UNIVERSIDAD TECNOLÓGICA INDOAMÉRICA

FACULTAD DE INGENIERIA, INDUSTRIA Y PRODUCCIÓN

CARRERA DE INGENIERIA EN CIENCIAS DE LA COMPUTACIÓN

TEMA:

DESARROLLO DE UNA APLICACIÓN BASADA EN ROS2 PARA EL CONTROL DE ROBOTS CON MOTORES INTELIGENTES DE LA FAINPRO

Trabajo de Integración Curricular previo a la obtención del título de Ingeniero en Ciencias de la Computación.

AUTOR

Jonathan Stalyn Toapanta Changoluisa

TUTOR

Ing. José Luis Varela Aldás PhD.

AMBATO – ECUADOR

FEBRERO, 2024

AUTORIZACIÓN POR PARTE DEL AUTOR PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL, Y PUBLICACIÓN ELECTRÓNICA DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Yo, Jonathan Stalyn Toapanta Changoluisa, declaro ser autor del Trabajo de Integración Curricular con el nombre "Desarrollo de una aplicación basada en ROS2 para el control de robots con motores inteligentes de la FAINPRO", como requisito para optar al grado de Ingeniero en Ciencias de la Computación y autorizo al Sistema de Bibliotecas de la Universidad Tecnológica Indoamérica, para que con fines netamente académicos divulgue esta obra a través del Repositorio Digital Institucional (RDI-UTI).

Los usuarios del RDI-UTI podrán consultar el contenido de este trabajo en las redes de información del país y del exterior, con las cuales la Universidad tenga convenios. La Universidad Tecnológica Indoamérica no se hace responsable por el plagio o copia del contenido parcial o total de este trabajo.

Del mismo modo, acepto que los Derechos de Autor, Morales y Patrimoniales, sobre esta obra, serán compartidos entre mi persona y la Universidad Tecnológica Indoamérica, y que no tramitaré la publicación de esta obra en ningún otro medio, sin autorización expresa de la misma. En caso de que exista el potencial de generación de beneficios económicos o patentes, producto de este trabajo, acepto que se deberán firmar convenios específicos adicionales, donde se acuerden los términos de adjudicación de dichos beneficios.

Para constancia de esta autorización, en la ciudad de Ambato, a los 12 días del mes de marzo de 2024, firmo conforme:

Autor: Jonathan Stalyn Toapanta Changoluisa
Firma: .....
Número de Cédula: 050476214-7
Dirección: Tungurahua, Píllaro, San Andrés, Chaupiloma.
Correo Electrónico: toapantasnick@gmail.com
Teléfono: 0999117522 - 0984359351

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Integración Curricular “Desarrollo de una aplicación basada en ROS2 para el control de robots con motores inteligentes de la FAINPRO” presentado por Jonathan Stalyn Toapanta Changoluisa, para optar por el Título Ingeniero en Ciencias de la Computación,

CERTIFICO

Que dicho Trabajo de Integración Curricular ha sido revisado en todas sus partes y considero que reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte los Lectores que se designe.

Ambato, 12 de marzo del 2024

Firmado electrónicamente por:
JOSE LUIS VARELA ALDAS

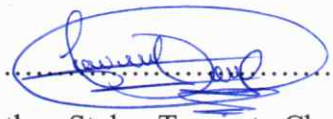
……

Ing. José Luis Varela Aldás PhD.

DECLARACIÓN DE AUTENTICIDAD

Quien suscribe, declaro que los contenidos y los resultados obtenidos en el presente Trabajo de Integración Curricular, como requerimiento previo para la obtención del Título de Ingeniero en Ciencias de la Computación, son absolutamente originales, auténticos y personales y de exclusiva responsabilidad legal y académica del autor

Ambato, 12 de marzo 2024



Jonathan Stalyn Toapanta Changoluisa
050476214-7

APROBACIÓN DE LECTORES

El Trabajo de Integración Curricular ha sido revisado, aprobado y autorizada su impresión y empastado, sobre el Tema: DESARROLLO DE UNA APLICACIÓN BASADA EN ROS2 PARA EL CONTROL DE ROBOTS CON MOTORES INTELIGENTES DE LA FAINPRO, previo a la obtención del Título de Ingeniero en Ciencias de la Computación, reúne los requisitos de fondo y forma para que el estudiante pueda presentarse a la sustentación del Trabajo de Integración Curricular.

Ambato,12 de marzo de 2024

……..

Ing. Patricio Gustavo Lara Álvarez

LECTOR

…….

Mgtr. Wilson Patricio Peñaherrera Acurio

LECTOR

DEDICATORIA

Dedico este trabajo a mi madre y a mi hermano, Blanca Changoluisa y Bryan Toapanta, cuyo amor incondicional y apoyo constante han sido mi mayor fuente de inspiración. Agradezco sus sacrificios y su fe en mis capacidades, que me han impulsado a alcanzar esta meta académica. Al Ing. José Varela, por su paciencia, aliento y comprensión durante las largas noches de estudio. A mis amigos y seres queridos, cuya motivación y positividad han sido un faro de luz en este camino. A todos aquellos que, de una u otra manera, han dejado una marca imborrable en mi viaje académico, gracias por ser parte integral de este logro.

AGRADECIMIENTO

Extendiendo mi profundo agradecimiento a mi familia, en especial a mi madre quiero reconocer el invaluable respaldo emocional y logístico que me brindaron a lo largo de este arduo proceso. Su apoyo incondicional, paciencia y comprensión fueron pilares fundamentales que me permitieron concentrarme en mi investigación. Agradezco especialmente por estar a mi lado durante las largas jornadas de trabajo, por compartir celebraciones de logros y por ser un constante recordatorio de que cada paso dado en este proyecto también era un triunfo compartido. Su amor y aliento han sido mi fuente de inspiración, y estoy profundamente agradecido por tenerlos como mi red de apoyo más cercana. También quiero expresar mi agradecimiento a mis amigos por su constante aliento y comprensión. Este trabajo no habría sido posible sin su apoyo inquebrantable. Agradezco sinceramente al Ing. José Varela Aldas por su orientación experta y apoyo constante a lo largo de este proyecto de tesis. Sus valiosos comentarios y dirección han sido fundamentales para el desarrollo de esta investigación, quien, con su vasta experiencia en el campo de la ingeniería, proporcionó una perspectiva única y valiosa que enriqueció significativamente mi investigación. Su compromiso con la excelencia académica y su disposición para compartir su conocimiento fueron cruciales en la resolución de desafíos técnicos y en la toma de decisiones fundamentadas. Además, agradezco la paciencia y la orientación que brindó en cada etapa del proceso, lo que me permitió abordar de manera efectiva los aspectos más complejos de este proyecto. La influencia positiva del Ing. Varela Aldas va más allá de la esfera académica y ha dejado una huella duradera en mi desarrollo profesional.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

UNIVERSIDAD TECNOLOGICA INDOAMERICA

FACULTAD DE INGENIERIA, INDUSTRIA Y PRODUCCIÓN

TEMA:

"Desarrollo de una aplicación basada en ROS2 para el control de robots con motores inteligentes de la FAINPRO"

AUTOR

Jonathan Stalyn Toapanta Changoluisa

Resumen

Este proyecto consiste en la implementación de ROS2 como middleware para el control remoto de robots emerge como una solución innovadora y eficaz. Así mismo se completa la creación de una aplicación basada en ROS 2 destinada al manejo de robots equipados con motores inteligentes de la FAINPRO. A través de una investigación documental exhaustiva y la aplicación de la metodología SCRUM que consta de 5 fases. Fase 1: Planificación del Sprint: Colaboración, Objetivos, Tareas, Duración, Priorización. Fase 2: Desarrollo del Sprint: Implementación, Adaptación, Comunicación, Transparencia, Entrega. Fase 3: Revisión del Sprint: Demostración, Retroalimentación, Mejora, Resultados, Aprendizaje. Fase 4: Retrospectiva del Sprint: Evaluación, Reflexión, Mejoras, Colaboración, Innovación. Y por último la fase 5: Planificación de la Siguiente Iteración: Ajustes, Priorización, Backlog, Planificación, Objetivos, se ha logrado diseñar y codificar una aplicación que no solo cumple con los objetivos establecidos, sino que también sienta las bases para futuros avances en la automatización y optimización de procesos en la FAINPRO. Los materiales utilizados para este proyecto son: motores inteligentes Dynamixel, un robot manipulador, tarjetas ESP32, Open CM y su expansión OpenCM485 EXP, la entrevista se aplica al técnico responsable del robot para la recolección de datos para los requerimientos de la aplicación. Los resultados obtenidos dan una aceptación del 98% dentro de la utilidad percibida y con el mismo porcentaje de aceptación en la factibilidad de uso percibida. La aplicación práctica de esta solución no solo optimiza los procesos actuales en la FAINPRO, sino que también abre la puerta a nuevas posibilidades y mejoras continuas en la integración de tecnologías de punta en entornos robóticos.

DESCRIPTORES: Manipulador móvil, ROS2, Micro-ROS, Actuadores inteligentes.

ABSTRACT

UNIVERSIDAD TECNOLÓGICA INDOAMÉRICA

Faculty of Engineering, Industry and Production Computing

AUTHOR: TOAPANTA CHANGOLUISA JONATHAN STALYN

TUTOR: PHD. VARELA ALDAS JOSE LUIS

ABSTRACT

DEVELOPMENT OF AN APPLICATION BASED ON ROS2 FOR THE CONTROL OF ROBOTS WITH FAINPRO INTELLIGENT MOTORS.

This project involves the implementation of ROS2 as middleware for remote robot control, emerging as an innovative and effective solution. It also entails the creation of an application based on ROS2 for managing robots equipped with intelligent motors from FAINPRO. Through thorough documentary research and the application of the SCRUM methodology consisting of 5 phases: Phase 1 - Sprint Planning: Collaboration, Objectives, Tasks, Duration, Prioritization. Phase 2 - Sprint Development: Implementation, Adaptation, Communication, Transparency, Delivery. Phase 3 – Sprint Review: Demonstration, Feedback, Improvement, Results, Learning. Phase 4 – Sprint Retrospective: Evaluation, Reflection, Improvements, Collaboration, Innovation. Finally, Phase 5 - Planning the Next Iteration: Adjustments, Prioritization, Backlog, Planning, Objectives. The result is the design and coding of an application that not only meets established objectives but also lays the groundwork for future advancements in automation and process optimization at FAINPRO. Project materials include Dynamixel smart motors, a robotic manipulator, ESP32 cards, Open CM, and its expansion OpenCM485 EXP. The interview with the robot's responsible technician collected data for application requirements. The results show a 98% acceptance rate in perceived utility and feasibility of use. The practical application of this solution not only optimizes current processes at FAINPRO but also opens the door to new possibilities and continuous improvements in the integration of cutting-edge technologies in robotic environments.

DESCRIPTORS: Intelligent actuators, mobile manipulators, Micro-ROS, ROS2.

CAPÍTULO I

1. INTRODUCCIÓN

1.1. CONTEXTUALIZACIÓN

1.1.1. Macro

La presente investigación se enmarca en el entorno dinámico de la robótica moderna, donde la adopción de tecnologías avanzadas es esencial para potenciar la funcionalidad y la eficiencia de los sistemas robóticos. En este contexto, el Sistema Operativo de Robots (ROS), y su versión más reciente, ROS2, emergen como herramientas fundamentales que han transformado la forma en que los robots son diseñados, desarrollados y operados.

A nivel mundial existen propuestas que implementan ROS2 para controlar diferentes robots. En la Universidad de ciencia y tecnología de Norway desarrollaron un sistema novedoso para controlar un robot móvil KMR iiwa utilizando ROS2. El KMR iiwa es un robot móvil con un manipulador montado en el base desarrollado por el fabricante de robots KUKA. El sistema desarrollado se integra con el sistema operativo Sunrise.OS del robot móvil y expone las interfaces de sensores y control a través de sockets UDP y TCP [1]. Por otro lado, tenemos el siguiente trabajo que fue realizado en la Universidad de Glasgow de Reino Unido, en este artículo, se aborda la importancia del software de simulación en la investigación en robótica, destacando su utilidad para representar virtualmente el mundo real.

La investigación se centra en evaluar paquetes de simulación compatibles con ROS versión 2, específicamente en el ámbito de la manipulación de brazos robóticos, una aplicación clave en la industria. La metodología incluye la comparación de estos paquetes en parámetros, tareas y escenarios similares, evaluando su desempeño en operaciones a largo plazo, éxito en la realización de tareas, repetibilidad y uso de recursos. Aunque no se identifica un software de simulación superior en general, se destaca que Ignition y Webots exhiben mayor estabilidad, mientras que PyBullet y Coppeliasim consumen menos recursos en comparación con otros competidores [2].

1.1.2. Meso

A nivel ecuador existen propuestas que implementan ROS2, El siguiente artículo aborda la cinemática de un robot paralelo 3UPS/RPU y sugiere un enfoque de control

centrado en rastrear la trayectoria de un prototipo virtual y real de este robot destinado a la rehabilitación de rodilla y tobillo. Se propone la aplicación de un control proporcional-derivativo (PD) con compensación de gravedad. La implementación de la posición de los actuadores del robot, así como de los sistemas de comunicación y control, se realiza en ROS2 (Sistema Operativo Robótico). Al emplear una trayectoria predefinida, los resultados destacan la robustez del controlador para mantener la estabilidad de la plataforma y lograr que los actuadores sigan la referencia deseada [3].

Por otra parte, tenemos el siguiente trabajo, donde se abordó esta problemática mediante el desarrollo de un Framework MultiRobot de código abierto, basado en ROS2, diseñado para facilitar la comunicación y coordinación eficiente entre sistemas mecatrónicos y sensores en entornos colaborativos cerrados. El enfoque incluyó la creación de un diseño de software y control, respaldado por simulaciones utilizando la herramienta Gazebo. Como resultado, se logró una arquitectura centralizada que incorpora un módulo de navegación autónoma para la planificación y seguimiento de rutas de los robots, un módulo de visión por computadora para la localización y gestión de incertidumbres, así como un módulo controlador de tareas para la asignación de misiones relacionadas con la movilización de objetos [4].

1.1.3. Micro

A nivel de la universidad Tecnológica Indoamérica existe la siguiente propuesta donde se lleva a cabo la construcción del robot omnidireccional móvil. Este trabajo de titulación expone la implementación de la industria 4.0 en los procesos educativos de la carrera de ingeniería industrial en la Universidad Tecnológica Indoamérica, mediante la aplicación de la metodología STEAM. La meta principal consiste en desarrollar y construir un robot educativo con funcionalidades de la industria 4.0, destinado a prácticas de laboratorio. El robot se compone de un manipulador móvil fabricado con diversos materiales, donde el brazo manipulador incorpora componentes impresos en 3D y la plataforma móvil se construye con acero ASTMA36. Se lleva a cabo un análisis mecánico detallado de los componentes físicos, incluyendo cálculos de velocidades y fuerzas máximas.

La parte electrónica se basa en un controlador Open CM9.04-C y actuadores inteligentes Dynamixel. Asimismo, se integra una cámara de video y sensores para medir variables físicas del entorno. El sistema se complementa con una tarjeta secundaria (ESP32) que se conecta de forma inalámbrica a internet, facilitando la transmisión de datos a la nube. Una aplicación móvil permite el control remoto del robot y la

visualización de datos e imágenes del entorno. La evaluación del desempeño del robot se lleva a cabo mediante pruebas, y se elaboran guías de prácticas para su implementación en los laboratorios de la FITIC [5].

Esta tesis aborda el desarrollo de un robot educativo que se fundamenta en un manipulador móvil con el propósito de enseñar competencias STEAM mediante una aplicación móvil. La meta es crear herramientas que mejoren los procesos de enseñanza-aprendizaje en el aula, incentivando a los estudiantes a explorar las nuevas tecnologías. El robot educativo se construye utilizando diversos materiales, entre ellos piezas metálicas e impresiones 3D con PLA. El circuito electrónico comprende dos tarjetas: un controlador para el sistema local y una tarjeta de comunicación WiFi. El rasgo distintivo del prototipo radica en la implementación de dispositivos inteligentes, posibilitando la recepción y envío de datos a través de una red local. Además, el robot se conecta de manera inalámbrica a la red global mediante tecnología System on Chip, almacenando datos en la nube para gestionar el sistema a través de Firebase.

La aplicación móvil proporciona control remoto del sistema y visualización de datos e imágenes desde la ubicación del robot. Finalmente, se lleva a cabo una evaluación de la aceptación y usabilidad de la propuesta, obteniendo resultados satisfactorios [6].

1.2. EL PROBLEMA

En la sociedad contemporánea, la robótica ha evolucionado de ser una disciplina especializada a convertirse en una necesidad imperante. La creciente relevancia de la robótica en diversos sectores de la vida cotidiana y la industria ha llevado a un cambio significativo en la perspectiva educativa. En respuesta a esta demanda, muchas universidades están optando por ofrecer carreras relacionadas con la robótica. Este enfoque se alinea con la necesidad de preparar a la próxima generación de profesionales con habilidades especializadas que aborden los desafíos y las oportunidades emergentes en este campo tecnológico.

La decisión de incorporar carreras en robótica en las Universidades se basa en la comprensión de que la robótica no solo es una disciplina técnica, sino que también se ha convertido en un componente esencial para la innovación y el progreso en diversas industrias. La formación en robótica no solo impulsa el conocimiento técnico, sino que

también fomenta habilidades cruciales como la resolución de problemas, la creatividad y la colaboración interdisciplinaria.

Al brindar carreras en robótica, las universidades no solo responden a la demanda del mercado laboral, sino que también contribuyen al desarrollo de una fuerza laboral altamente capacitada y adaptativa. Además, esta iniciativa abre oportunidades para que los estudiantes exploren y participen activamente en la vanguardia de la tecnología, desempeñando un papel fundamental en la conformación del futuro tecnológico de la sociedad.

Debido a que hay algoritmos deficientes de control para los robots en la manipulación se encuentra con el desgaste de los componentes puesto que está repetidamente la conexión y retiro de la alimentación a los componentes y corren el riesgo de quemar los elementos, también la latencia en la comunicación entre el robot y la estación de control es uno de los factores que dan problemas ya que hay veces la perdida de información durante la transmisión de información entonces se busca mejorar la comunicación entre componentes para un mejor rendimiento, otra de las causas de la problemática es la utilización de herramientas convencionales para la operación de los robots esto es debido a que se sigue acostumbrado a las tecnologías de uso más común dentro de la robótica por ello se requiere renovar para así no causar los daños en los componentes electrónicos ya que los costos no son bajos y para correr el riesgo de perder uno de los componentes por un mal uso o por manipulación continua, se toma la decisión de mejorar la comunicación de los elementos, para una mejor transmisión de información y con un mejor algoritmo mejorado de control.

Considerando las dificultades identificadas y los potenciales riesgos que afectan a cada componente durante las operaciones de robots en el laboratorio de robótica de la Facultad de Ingeniería, Industria y Producción de la Universidad Tecnológica Indoamérica, surge el siguiente problema: Ineficiencia en la operación de robots equipados con motores inteligentes en la FAINPRO.

En la figura1 se presenta el árbol del problema donde se da a notar varias causas con sus respectivos efectos; una de las causas que existen algoritmos deficientes de control el cual crea el mayor desgaste de los componentes, otra de las causas que presenta es la latencia en la comunicación entre el robot y la estación de control lo que provoca la perdida de datos durante la transmisión de información, como ultima causa, se especula

la utilización de herramientas convencionales para operación de robots el cual podría ser responsable del daño de los componentes electrónicos.

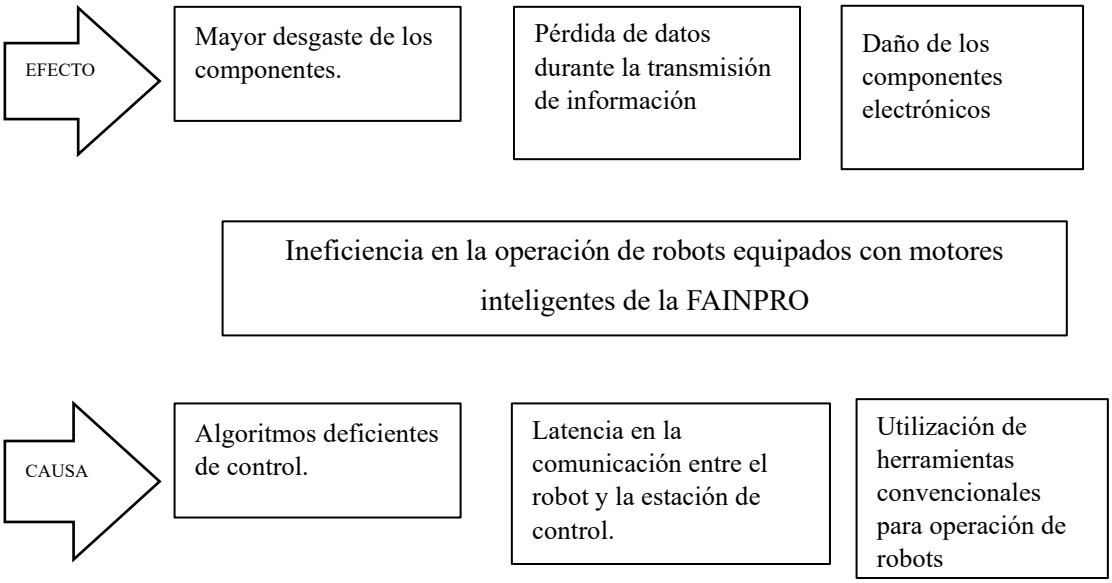


Fig. 1: Árbol del problema.

1.3. PROGNOSIS

El problema identificado radica en la ineficacia operativa de los robots, y se origina en tres causas fundamentales. En primer lugar, la presencia de algoritmos deficientes afecta negativamente la ejecución eficiente de las tareas programadas. La latencia en la comunicación entre el robot y la estación de control es otra causa identificada, lo que compromete la sincronización y respuesta en tiempo real, contribuyendo así a la ineficiencia global. Finalmente, la utilización de herramientas convencionales para operaciones de robots se presenta como una barrera adicional.

Estas causas tienen consecuencias directas, como el mayor desgaste de los componentes, la pérdida de datos durante la transmisión de información y el daño a los componentes electrónicos. Estos efectos adversos impactan significativamente en la vida útil y el rendimiento general de los robots, lo que subraya la urgencia de encontrar soluciones efectivas.

La no implementación de la aplicación basada en ROS2 para el control de robots con motores inteligentes en la FAINPRO podría acarrear consecuencias significativas en

términos de eficiencia operativa, seguridad y rendimiento general de los robots. En primer lugar, la persistencia de algoritmos deficientes y la latencia en la comunicación entre el robot y la estación de control continuarían siendo obstáculos fundamentales. Esto resultaría en operaciones robotizadas ineficaces, afectando la precisión, velocidad y capacidad de respuesta en tiempo real.

La utilización de herramientas convencionales para operaciones de robots también persistiría, limitando la versatilidad y la capacidad de adaptación de los robots a tareas más avanzadas y complejas. Los efectos de estas deficiencias se traducirían directamente en un mayor desgaste de los componentes, pérdida recurrente de datos durante la transmisión de información y daño continuo a los componentes electrónicos. A largo plazo, esto podría resultar en una disminución significativa de la vida útil de los robots y, en última instancia, en costos operativos más altos debido a reparaciones y reemplazos frecuentes.

Además, la falta de mejora en la eficiencia operativa podría afectar la competitividad de la FAINPRO en el ámbito de la investigación y la aplicación de tecnologías robóticas avanzadas. La institución podría quedarse rezagada en términos de innovación y eficiencia, lo que podría afectar su capacidad para liderar en proyectos tecnológicos y participar en colaboraciones de investigación avanzada.

1.4. ANTECEDENTES DE LA EMPRESA

Actualmente, dispone de un laboratorio equipado con motores inteligentes que, lamentablemente, no están siendo utilizados debido a la ausencia de un software adecuado para operar eficientemente sus componentes. La falta de esta herramienta ha limitado la capacidad de aprovechar plenamente las funcionalidades avanzadas de los motores, frustrando su potencial aprovechamiento. En consecuencia, se busca activamente una solución que permita la integración de un software especializado, con el objetivo de desbloquear el rendimiento óptimo de estos motores y maximizar los beneficios que pueden aportar a nuestras operaciones. Paralelamente se ha trabajado en ROS en la mayoría de los proyectos, pero migrar a ROS2 puede ser beneficioso para aquellos que buscan una mayor flexibilidad, escalabilidad, soporte para sistemas embebidos y características avanzadas de comunicación y seguridad. Sin embargo, la elección entre ROS y ROS2 dependerá de las necesidades específicas del proyecto y de la compatibilidad con las bibliotecas y herramientas existentes.

INDOAMÉRICA es una Institución de educación superior con más de tres décadas al servicio de la ciencia y la formación de profesionales de excelencia. Nuestros orígenes se remontan a los años ochenta, cuando fue constituido el Centro de Estudios de Computación “Servisistemas Informáticos Indoamérica”. Su nombre, INDOAMÉRICA, expresa un profundo sentido de pertenencia a la cultura latinoamericana, lo cual constituye la base fundamental de nuestra identidad.

A inicios de los noventa, la institución adquiere el estatus de “Instituto Técnico Superior Indoamérica”, y en 1992 se eleva a Instituto Tecnológico Superior. Posteriormente, para responder a las demandas del desarrollo social y científico, se plantea como objetivo transformarla en una universidad. En 1998, luego del cumplimiento de las exigencias académicas vigentes, se constituye la Universidad Tecnológica INDOAMÉRICA con domicilio en la ciudad de Ambato, mediante ley N° 112, publicada en el Registro Oficial N° 373. Debido a su calidad académica y reconocimiento social, en el año 2004, se crea legalmente la extensión de la Universidad en la ciudad de Quito. En los años inmediatos, su oferta académica se consolida en ambas ciudades, abarcando las tres modalidades de estudio: presencial, semipresencial y a distancia.

Al finalizar la primera década del presente milenio, la Universidad define objetivos mucho más ambiciosos, entre ellos, el fortalecimiento de la investigación científica y la vinculación con la sociedad; esto se desarrolla en un contexto de cambios en el Sistema de Educación Superior Ecuatoriano.

La restructuración del sistema conllevó a la evaluación de todas las universidades e institutos del país. INDOAMÉRICA, debido en gran parte a su producción científica, enfrentó este nuevo desafío de manera exitosa, en 2012, su sede en Quito fue la única aprobada en la evaluación realizada por el CEAACES a las extensiones universitarias del país. Tan solo un año después, la Universidad, a nivel institucional, obtiene la ACREDITACIÓN, destacándose como la mejor universidad del centro del país.

Más allá de los logros alcanzados, a partir de 2014, INDOAMÉRICA continúa con su empeño de ser una universidad de excelencia. La universidad ha consolidado algunos de sus centros de investigación logrando una producción científica de impacto internacional. Además, ha diversificado permanentemente su oferta académica para garantizar un servicio educativo pertinente. Actualmente, INDOAMÉRICA tiene como

visión convertirse en un referente universitario latinoamericano. En este sentido, la institución trabaja incansablemente para alcanzar mayores niveles de calidad académica, con esta finalidad se están desarrollando varios proyectos; entre ellos, el fortalecimiento del área académica que integrará efectivamente los procesos de investigación, vinculación y docencia, y la ampliación de la infraestructura en Ambato y Quito.

Somos la mejor universidad del centro del país y nos estamos consolidando como la mejor del norte de Quito.

MISION

Generar vínculos con la sociedad que fomente la ética, el emprendimiento, la competencia social y ser ambientalmente responsable, mediante el aseguramiento de la calidad en la ciencia, innovación, tecnología, cultura y desarrollo sostenible.

VISION

Ser una Universidad comprometida con la sociedad y un referente a nivel internacional.

1.5. JUSTIFICACIÓN

La realización de la tesis enfocada en el "Desarrollo de una aplicación basada en ROS2 para el control de robots con motores inteligentes de la FAINPRO" se presenta como una iniciativa estratégica que busca abordar y superar desafíos clave en el ámbito de la robótica. La necesidad de optimizar la eficiencia operativa en el área de robótica de la FAINPRO se convierte en un impulsor fundamental para la propuesta de esta aplicación innovadora.

La aplicación no solo promete mejorar la eficiencia operativa, sino que también ofrece una adaptabilidad crucial a cambios en los requisitos y en el entorno operacional. La flexibilidad inherente a la tecnología ROS2 (Robot Operating System) permite una personalización efectiva, asegurando que la aplicación pueda evolucionar de acuerdo con las necesidades específicas de la FAINPRO. El veneficio directo que aporta este trabajo es la unificación de los elementos para poder trabajar con más precisión y en un solo puesto de trabajo. El veneficio indirecto que nos brinda es el control de cada uno de los elementos un monitoreo de los datos que se envían y los que se reciben.

Entre los beneficios destacados se encuentra el aumento de la calidad en el trabajo realizado por los robots con motores inteligentes. La aplicación se presenta como una herramienta que no solo agiliza las operaciones cotidianas, sino que también permite una comunicación más directa, reduciendo significativamente la pérdida de tiempo en la transmisión de datos. Al aprovechar las funciones avanzadas de ROS2, se establece una base sólida para la mejora general de la eficiencia y la productividad en el entorno robótico.

Además de ampliar el conocimiento en el campo de la robótica, el desarrollo de esta aplicación tiene el potencial de generar contribuciones significativas. Desde una perspectiva práctica, la tesis proporcionará una plataforma aplicable y escalable para la implementación de sistemas robóticos avanzados en la FAINPRO. La mejora de la eficiencia operativa, la optimización del rendimiento y la adaptabilidad mejorada a entornos cambiantes son contribuciones específicas que se anticipan como resultados tangibles de este proyecto de investigación.

En resumen, la tesis sobre el desarrollo de una aplicación basada en ROS2 no solo responde a las necesidades inmediatas de la FAINPRO en términos de eficiencia robótica, sino que también representa una inversión estratégica en la capacidad de adaptación y progreso tecnológico para el futuro de la robótica en la institución.

1.6. OBJETIVOS

General

Desarrollar una aplicación basada en ROS2 para el control de robots con motores inteligentes en la FAINPRO

Específicos

- Diseñar la estructura de software basada en ROS2 para la interfaz de robots con motores inteligentes.
- Codificar la aplicación para el control de robots con motores inteligentes utilizando el middleware ROS2 para el control remoto.
- Evaluar la funcionalidad de la aplicación mediante diversas pruebas con el fin de garantizar su rendimiento.

CAPÍTULO II

2. FUNDAMENTACIÓN TEORICA

2.1. ANTECEDENTES INVESTIGATIVOS

En el siguiente trabajo de tesis del autor *Mate de Rose* con el tema *"LiDAR-based Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2",* fue llevado acabado en la fundación LINKS de Turín, el problema planteado en este trabajo es la navegación autónoma de vehículos, donde propone y se implementa un método para el manejo de obstáculos dinámicos. La propuesta presentada tiene como meta establecer los fundamentos de una estructura altamente modular en el ámbito de la navegación dentro del marco de ROS2, lo que facilitaría su posterior ajuste para adaptarse a situaciones particulares desarrollados en el proyecto LINKS. Por último, se han realizado pruebas en ambientes virtuales mediante el uso del simulador Webots y Rviz, identificando áreas de mejora y aspectos cruciales para futuros desarrollos en el proyecto LINKS.

Las evaluaciones mediante simulaciones llevadas a cabo en un entorno virtual en constante cambio han evidenciado mejoras significativas en cuanto a la reducción de colisiones y la optimización de los tiempos de desplazamiento, gracias a la incorporación del DOL en combinación con el Controlador DWB que está a disposición [7].

En el trabajo de grado que tiene por tema: *"Developing a platform for the KUKA iDo social robot"* con el autor *"Geert Mol"* en la empresa KUKA, se realiza un estudio en el rediseño de hardware para el control de robots en donde se implementa un controlador PID, una red de control del robot, un controlador de radio y una computadora central. En donde se utilizó el software de control ROS2 para el robot, el cual tiene la comunicación con el controlador del motor mediante enlaces seriales. Los resultados indicaron una ligera disminución en el tiempo requerido para la configuración, lo que sugiere la posibilidad de una menor pendiente en la curva de aprendizaje para las investigaciones posteriores que involucren la plataforma KUKA [8].

En el siguiente artículo que lleva de nombre *"Implementation and Performance Study of the Micro-ROS/ROS2 Framework to Algorithm Design for Attitude Determination and Control System"* con los autores *"Alfredo Mamani-Saico & Pablo*

Raúl Yanyachi" fue realizada en la universidad nacional de San Agustín de Arequipa y en el Instituto de Investigaciones Astronómicas y Aeroespaciales Pedro Paulet. En donde el estudio aprovecha las capacidades de ROS2 y Micro-ROS para desarrollar una plataforma experimental destinada a algoritmos de control de actitud. Esta plataforma incluye un entorno informático dedicado al procesamiento de datos (CEDDP) y un módulo para el sistema de control y determinación de actitudes (ADCS). La investigación se centra en medir el tiempo de respuesta de la unidad CEDDP en escenarios de uso de red, tanto exclusiva como compartida. Además, se cuantifica la pérdida de mensajes en tres segmentos durante un ciclo completo de tareas de control, bajo varias configuraciones de Calidad de Servicio (QoS). Además, se analiza la periodicidad de los mensajes en cuatro puntos clave dentro de las entidades ROS2 que participan en el sistema.

Los resultados de la experimentación revelan una plataforma experimental sólida construida sobre los marcos ROS2 y Micro-ROS. Se concluye que emplear una política de QoS de mejor esfuerzo, junto con ajustes menores a los perfiles de QoS, emerge como un enfoque óptimo para diseñar algoritmos de control de actitud [9].

En el siguiente trabajo de grado que tiene por título *"control de un robot cartesiano usando el sistema operativo robótico ros2"* con el autor *"Vázquez Ramos, Álvaro"* fue ejecutada en la universidad de Valladolid, desarrolla el software de control, se crearon programas para operar un robot cartesiano de manera manual o automática utilizando el controlador industrial AMC4030. Para la operación manual, se diseñó una interfaz gráfica en lenguaje C++/CLI en el entorno .NET, permitiendo que un operador introduzca la posición deseada del robot. En el modo automático, se empleó el entorno ROS2 para la comunicación, facilitando la reutilización del código en diversas aplicaciones. En cuanto a la implementación de la visión artificial, se presentó un ejemplo de sistema que determina la posición del robot cartesiano mediante cámaras de profundidad.

Los resultados obtenidos incluyen la consecución de programas capaces de controlar el robot cartesiano e integrarlos en el sistema de ROS2, cumpliendo los objetivos del proyecto. Se demostró la compatibilidad de ROS, permitiendo la fácil portabilidad del código a otros sistemas y su aplicación en diferentes máquinas. Además, se destacó la versatilidad del sistema robótico para su utilización en diversas aplicaciones, como sistemas de visión artificial. Se identificaron posibles mejoras y proyectos futuros, como la optimización del software y el desarrollo de un sistema para determinar la posición del robot, con potencial aplicación en la poda de viñedos. En resumen, la tesis se centró en el

desarrollo de programas de control, la implementación de una interfaz gráfica y la integración con ROS2, logrando resultados satisfactorios y abriendo perspectivas para proyectos venideros [10].

El siguiente trabajo de Máster con el título *"MERLIN2: Sistema cognitivo para ROS 2"* con el autor *"Miguel Ángel González Santamarta"*. La tesis fue llevada acabó en la universidad de León donde aborda el diseño y desarrollo de la arquitectura cognitiva híbrida denominada MERLIN2, diseñada para modelar el comportamiento de robots en la ejecución de tareas específicas. La implementación de esta arquitectura se llevó a cabo en Python3, haciendo uso de dos librerías especializadas para gestionar el conocimiento del robot y desarrollar comportamientos basados en máquinas de estados. Se eligió ROS 2 como el marco principal para el desarrollo de MERLIN2, aprovechando sus mejoras en los sistemas de comunicación y otras funcionalidades en comparación con ROS.

La comunicación entre los nodos de la arquitectura MERLIN2 se realizó mediante los sistemas de ROS2 basados en Data Distribution Service (DDS). Además, se crearon paquetes de ROS2 para implementar los diversos componentes de la arquitectura, incluyendo el sistema de gestión del conocimiento basado en PDDL y la librería de Python3 para desarrollar comportamientos basados en máquinas de estados. Para evaluar el rendimiento de MERLIN2, se llevaron a cabo pruebas en un entorno simulado con Gazebo y se compararon los resultados con los obtenidos mediante la arquitectura MERLIN y ROSPlan. Los resultados revelaron que los tiempos y distancias recorridas por el robot utilizando MERLIN2 eran comparables o superiores a los obtenidos con las otras arquitecturas evaluadas [11].

2.2. FUNDAMENTACIÓN TEORICA

En la Figura 2, se aprecia la distribución de la fundamentación teórica, que abarca diversos aspectos relacionados con el robot. Esta sección engloba conceptos generales como robótica, robótica móvil y actuadores inteligentes, proporcionando una visión integral de los fundamentos que respaldan el desarrollo y comprensión del robot en cuestión.

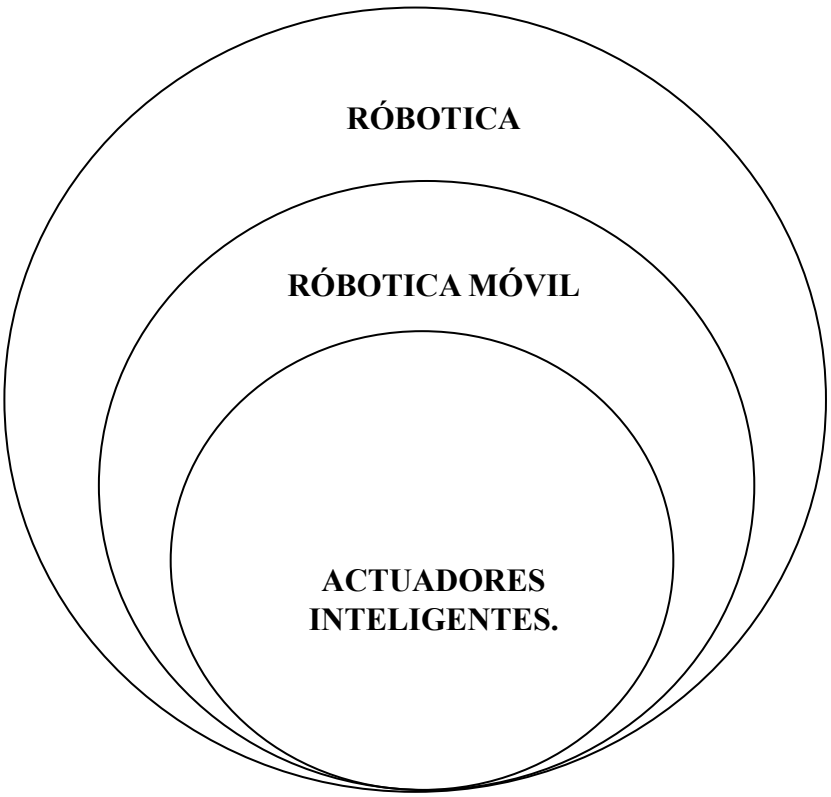


Fig. 2: Agrupación teórica sobre el robot.

2.2.1. La robótica

La revolución tecnológica ha generado nuevas perspectivas en el ámbito educativo, destacándose la integración de la robótica como uno de los sistemas más vanguardistas en entornos educativos. El propósito fundamental de la presente investigación consistió en examinar la evolución del concepto de "robótica" en el ámbito educativo, utilizando como punto de referencia la literatura recopilada en la Web of Science (WoS). La metodología empleada se basó en la bibliometría, que fue aplicada para analizar el desarrollo estructural y dinámico de dicho concepto. El rastreo de estudios sobre robótica en educación en la plataforma WoS se inició en el año 1975, y su progresión ha sido variable, alcanzando su punto máximo de producción en 2019. Aunque se focalizó en la recopilación de estudios vinculados a áreas de conocimiento educativo, también se observó la presencia de otras disciplinas como ingeniería e informática.

Se evidenció que los artículos de actas son los tipos de manuscritos más utilizados para presentar resultados científicos en esta área. Estados Unidos sobresale como el país con el mayor nivel de producción en este campo de estudio. Los resultados subrayan el

potencial de este tipo de investigaciones en el ámbito científico y resaltan la relevancia de la tecnología robótica en la formación de futuros cirujanos y en los logros que estos obtienen en su propio proceso de aprendizaje [12].

2.2.2. Evolución de la robótica

La robótica móvil ha experimentado un progreso constante en el último siglo, desde autómatas bioinspirados hasta vehículos autónomos, impactando campos como la exploración espacial y aplicaciones industriales. Este avance se debe a mejoras tecnológicas en procesadores, sensores como cámaras y eficiencia de baterías. Este análisis revisa la evolución histórica de la robótica móvil, desde sus inicios hasta aplicaciones recientes en industria y salud. Los primeros intentos de autómatas autónomos datan de épocas tempranas, mostrando el potencial tecnológico. A lo largo del tiempo, mejoras en hardware y software han optimizado la movilidad, procesamiento y percepción de estos robots. La eficiencia de las baterías ha sido crucial, permitiendo mayor autonomía.

La robótica móvil ha dejado una marca emblemática en la exploración espacial, desplegando robots para investigar en entornos hostiles. Este enfoque se ha aplicado en industrias, automatizando procesos y reduciendo riesgos laborales. En salud, robots móviles han mejorado la precisión en cirugías y atención a pacientes. En conclusión, la robótica móvil ha evolucionado de curiosidad técnica a herramienta indispensable en industria, exploración y salud. Su presencia creciente en diversas aplicaciones refleja su versatilidad y promete un papel crucial en el futuro [13].

2.2.3. Tipos de robots

Existen diversos tipos de robots, cada uno diseñado para cumplir funciones específicas en diferentes entornos y aplicaciones. Aquí hay una clasificación general basada en sus características y funciones:

- Robots Industriales: Utilizados en entornos de fabricación para realizar tareas como ensamblaje, soldadura, pintura, manejo de materiales, etc.
- Robots Móviles: Diseñados para moverse en el entorno y realizar tareas específicas. Pueden incluir robots terrestres, aéreos o acuáticos.

- Robots Médicos: Utilizados en aplicaciones médicas para cirugía asistida, telecirugía, entrega de medicamentos, entre otros.
- Robots de Servicio: Destinados a realizar tareas de servicio en entornos no industriales, como la limpieza, entrega de alimentos, asistencia en el hogar, etc.
- Robots de Exploración: Diseñados para explorar entornos peligrosos o inaccesibles para los humanos, como el espacio, el fondo del océano o áreas contaminadas.
- Robots Educativos: Utilizados con fines educativos para enseñar programación, electrónica y conceptos de robótica a estudiantes.
- Robots de Entretenimiento: Diseñados para proporcionar entretenimiento y diversión, como robots juguetes, mascotas robóticas y robots artísticos.
- Robots Humanoides: Diseñados con apariencia y movimientos similares a los humanos. Pueden ser utilizados en investigación, asistencia en el hogar o entretenimiento.
- Robots Colaborativos (Cobots): Diseñados para trabajar de forma segura y colaborativa con humanos en entornos compartidos.
- Robots Agrícolas: Utilizados en la agricultura para tareas como siembra, cosecha, riego y monitoreo de cultivos.
- Robots de Rescate: Diseñados para operaciones de rescate en situaciones peligrosas, como desastres naturales o estructuras colapsadas.
- Robots Militares: Utilizados en aplicaciones militares para tareas como vigilancia, desactivación de bombas, y exploración en terrenos hostiles [14].

2.2.4. Control de robots

Implica el desarrollo de algoritmos y estrategias que permitan al robot realizar tareas específicas de manera eficiente y segura. Esto puede incluir la implementación de sistemas de retroalimentación (feedback) para ajustar continuamente el comportamiento del robot en función de la información que recibe de sus sensores. La robótica abarca diversas disciplinas, como la mecánica, la electrónica, la informática y la inteligencia artificial, todas las cuales contribuyen al diseño y desarrollo de sistemas de control robótico.

2.2.5. Robótica móvil

Es un sistema basado en tecnología robótica diseñado para transportar diversos materiales o cargas de manera completamente autónoma. Su aplicación abarca los procesos logísticos, ya sea en entornos como almacenes automatizados o en integración fluida en las líneas de producción de fábricas. La eficacia y la capacidad de adaptación a las necesidades del cliente han facilitado la expansión de los robots móviles más allá del ámbito industrial, integrándose de manera natural en el sector de servicios, siendo visibles en lugares como hospitales, oficinas y restaurantes. La robótica móvil se distingue por su autonomía y versatilidad, además de su carácter colaborativo, garantizando total seguridad para los trabajadores y el entorno laboral mediante la incorporación de sensores que abarcan 360 grados, permitiéndoles reconocer objetos o personas y detenerse antes de cualquier posible colisión [15].

2.2.6. Actuadores inteligentes

Se define como un dispositivo integrado que incorpora controles, capacidades de comunicación, control de posición y retroalimentación, así como detección de fallos. Esta configuración proporciona una instalación sencilla, facilita su uso y programación, y permite un monitoreo eficiente, eliminando la necesidad de contar con controles externos. Con componentes de hardware y tecnología más avanzados en comparación con un actuador convencional, el actuador inteligente exhibe una robustez que lo capacita para manejar aplicaciones de alto ciclo de trabajo. Además, su capacidad de ofrecer un control plug-and-play lo convierte en una solución ideal para aplicaciones industriales de manejo de materiales, ofreciendo una integración eficaz y una operatividad mejorada en entornos industriales dinámicos [16].

En la Figura 3, se destaca la estructuración teórica relacionada con el software desarrollado, abordando conceptos fundamentales como la programación de robots, middleware, ROS (Robot Operating System) y ROS2. Estas teorías, esenciales en su naturaleza, desempeñan un papel crucial al proporcionar la base teórica necesaria para alimentar y orientar de manera integral el desarrollo de este proyecto.

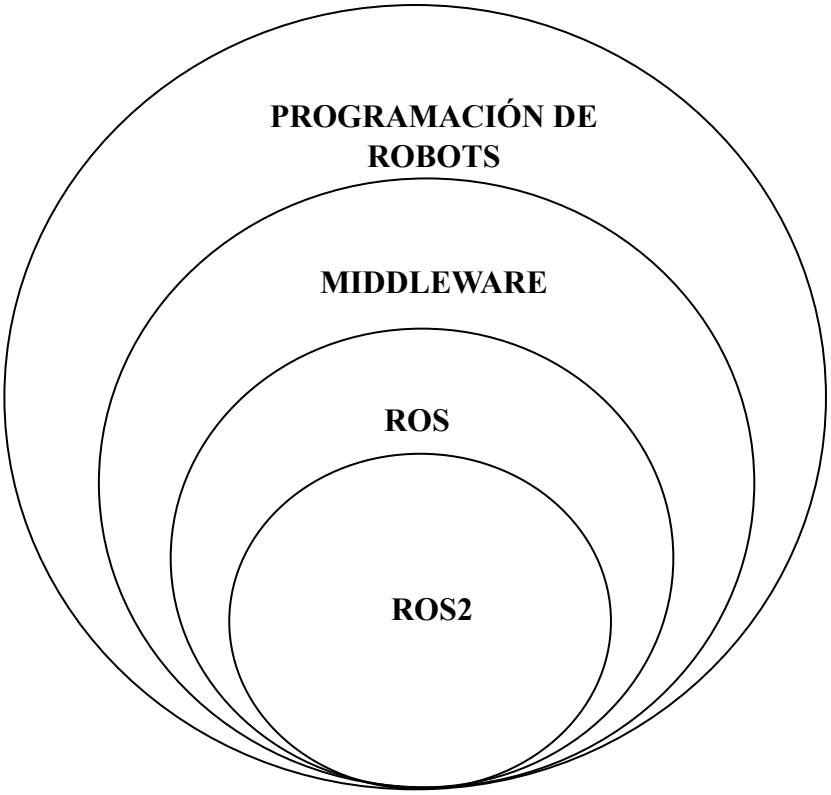


Fig. 3: Agrupación teórica sobre software.

2.3. Programación de robots

La programación robótica constituye el procedimiento integral que engloba el diseño, desarrollo e implementación de algoritmos y comandos destinados a supervisar el comportamiento y las funcionalidades de un robot. Este enfoque no solo se presenta como una herramienta esencial para cultivar habilidades técnicas y conceptuales, sino que también se erige como un recurso pedagógico valioso aplicable a estudiantes de diversas edades. Diversos lenguajes de programación, plataformas y herramientas se encuentran disponibles para abordar la programación robótica. Ejemplos destacados de estas plataformas y lenguajes utilizados en la enseñanza de la programación robótica incluyen LEGO Mindstorms, Arduino, Raspberry Pi, Python, Scratch y Blockly. A continuación, se enlistan los lenguajes de programación comúnmente empleados en robots destinados a la educación:

- Scratch
- Python
- Blockly
- JavaScript

- C++
- Java
- Swift
- Arduino (lenguaje de programación basado en C/C++)
- Mindstorms (basado en LabVIEW)
- Ruby

En el campo de la robótica educativa, se pueden identificar diversos enfoques y clasificaciones en términos de programación, entre los cuales se encuentran:

2.3.1. Programación Visual: Este método se basa en bloques de programación e interfaces de arrastrar y soltar. Estos lenguajes permiten a los estudiantes crear programas sin la necesidad de escribir código directamente. Ejemplos notables de programación gráfica son Scratch, Blockly y Mindstorms.

2.3.2. Programación Basada en Texto: Los alumnos redactan código utilizando lenguajes de programación textuales como Python, JavaScript, C++ y Java. Estos lenguajes requieren un nivel más avanzado de sofisticación, así como conocimientos en la sintaxis y estructuras de programación.

2.3.3. Programación Basada en Eventos: En este enfoque, los programas se construyen alrededor de eventos y acciones que responden a ellos. Los estudiantes definen cómo los robots deben comportarse frente a diversas situaciones o estímulos sensoriales. Ejemplos de lenguajes de programación orientados a eventos son Scratch y Mindstorms.

2.3.4. Programación Orientada a Objetos: Este enfoque se basa en el principio de objetos que representan entidades y propiedades del mundo real. Los alumnos pueden programar robots utilizando técnicas como encapsulamiento, herencia y polimorfismo en lenguajes como Python, Java, C++ y Ruby.

2.3.5. Programación de Bajo Nivel: Implica la interacción directa con el hardware del robot, como motores y sensores. Se utilizan lenguajes de programación como C, C++ o ensamblador.

2.3.6. Programación de Alto Nivel: A diferencia de la programación de bajo nivel, se centra en la lógica y el comportamiento global del robot. Utiliza lenguajes de programación de alto nivel como Python, JavaScript o Ruby [17].

2.3.7. Middleware ROS2

Este texto examina los obstáculos enfrentados durante el desarrollo de Apex.OS, una variante automotriz del sistema operativo robótico (ROS)2. Se abordan los desafíos mediante la exploración de las prácticas óptimas empleadas en la comunicación de datos y la ejecución de software en sistemas basados en OSEK. El enfoque se centra en comprender estos aspectos para mejorar la adaptabilidad del sistema. Además, se detallan las extensiones implementadas en ROS2, Apex.OS y Apex.Middleware para abordar las restricciones en tiempo real específicas de los sistemas automotrices. Este análisis y las modificaciones introducidas buscan optimizar la funcionalidad y la eficiencia de Apex.OS, ofreciendo soluciones a los desafíos particulares que surgen en el desarrollo de sistemas operativos para aplicaciones automotrices [18].

2.3.8. Concepto de ROS

ROS representa un sistema operativo de código abierto diseñado para robots, proporcionando servicios típicos de un sistema operativo convencional. Estos servicios incluyen la abstracción del hardware, el control de dispositivos de bajo nivel, la implementación de funciones comunes, la transferencia de mensajes entre procesos y la gestión de paquetes. Además, ofrece herramientas y bibliotecas que facilitan la obtención, construcción, escritura y ejecución de código a través de varias computadoras. En comparación con otras estructuras para robots como Player, YARP, Orocos, CARMEN, Orca, MOOS y Microsoft Robotics Studio, ROS comparte similitudes en su enfoque.

El gráfico de procesos en ROS constituye una red de procesos peer-to-peer, potencialmente distribuidos en diferentes máquinas, que se integran fácilmente a través de la infraestructura de comunicación de ROS. La comunicación en ROS se realiza mediante diversos estilos, tales como comunicaciones síncronas del tipo RPC a través de servicios, transmisiones asincrónicas de información en tópicos, y almacenamiento de datos en un servidor de parámetros (Parameter Server). Aunque ROS no está específicamente diseñado como un marco para despliegue en tiempo real, es posible integrar código en tiempo real. Por ejemplo, el robot PR2 de The Willow Garage utiliza un sistema denominado pr2_etherCAT para transportar mensajes ROS a través de procesos en tiempo real, demostrando la flexibilidad de integración de ROS. Además, ROS se integra de manera fluida con el conjunto de herramientas Orocos Real-time [19],[20].

2.3.9. Concepto de ROS2

Un enfoque conciso para la programación de robots con ROS2 proporciona a los lectores el conocimiento básico y las herramientas necesarias para programar y dar vida a los robots. Este recurso proporcionará a los lectores las habilidades necesarias para trabajar en proyectos utilizando ROS2, una versión actualizada de ROS. No es imprescindible tener experiencia con ROS2 ya que el texto explica los conceptos, herramientas y técnicas desde cero. Algunas características relevantes son: Utilizar dos lenguajes de programación soportados oficialmente por ROS2, principalmente C y Python. Explora ROS2 desde tres perspectivas diferentes pero complementarias: comunidad, gráficos por computadora y espacio de trabajo. Incluye boots simulados completos, estrategias de desarrollo y prueba, árboles de comportamiento y descripción, así como configuración y uso de Nav2. Proporciona un repositorio de GitHub que contiene código habilitado para lectura [21].

En la tabla 1 se puede observar cuales son las diferencias entre ROS y ROS2 el cual también ayuda a poder migrar a un mejor sistema de control para los robots con actuadores inteligentes.

Tabla I
Diferencias entre ROS Y ROS2.

Diferencias	ROS	ROS 2
Arquitectura y desempeño	Originalmente diseñado para robots grandes y complejos, utiliza una arquitectura de comunicación basada en grafo y está optimizado para sistemas de un solo procesador.	Diseñado para abordar limitaciones de ROS, ofrece una arquitectura más flexible con soporte para sistemas distribuidos, comunicación en tiempo real y mayor escalabilidad.
Soporte para sistemas embebidos	No está optimizado para sistemas embebidos con recursos limitados.	Ha mejorado el soporte para sistemas embebidos y de tiempo real, siendo más adecuado para aplicaciones en robots más pequeños y dispositivos con restricciones de recursos.
Comunicación		

	Usa el middleware de comunicación ROS 1 (basado en XML-RPC) que puede tener limitaciones en términos de eficiencia y flexibilidad.	Utiliza DDS (Data Distribution Service) como middleware predeterminado, ofreciendo una comunicación más robusta y eficiente, con soporte para calidad de servicio y seguridad.
Seguridad	Originalmente no fue diseñado con un enfoque fuerte en la seguridad.	Incluye mejoras significativas en seguridad, con la capacidad de cifrar la comunicación y gestionar el acceso a los nodos, lo que es crucial para aplicaciones en entornos críticos.
Desarrollo activo	Aunque sigue siendo ampliamente utilizado, el desarrollo activo se ha trasladado en gran medida a ROS 2.	Es la versión más reciente y recibe actualizaciones y mejoras regulares. La comunidad y el soporte están más orientados hacia ROS 2 para el desarrollo futuro.

CAPÍTULO III

3. METODOLOGIA DE LA INVESTIGACIÓN

3.1. MODALIDAD DE INVESTIGACIÓN

3.1.1. Investigación documental:

Esta investigación desempeña un papel crucial en el marco teórico, siendo el punto focal para la obtención de información relevante en relación con el tema de nuestra tesis. En esta fase, se exploran trabajos previos relacionados con nuestra área de estudio, además de profundizar en la comprensión de conceptos clave. Cada concepto es meticulosamente respaldado con citas bibliográficas, estableciendo una conexión directa con las fuentes originales. Este enfoque garantiza la validez y la autoridad de la información recopilada, brindando una sólida base teórica para nuestro trabajo. La inclusión de citas bibliográficas no solo respalda la información presentada, sino que también proporciona a nuestros lectores la oportunidad de explorar las fuentes originales para una comprensión más profunda. En conjunto, esta investigación en el marco teórico enriquece la calidad y la robustez de nuestro trabajo, asegurando que esté adaptado en una base conceptual sólida y respaldado por la erudición existente.

3.1.2. Investigación aplicada:

La investigación se centra en el progreso de las nuevas tecnologías en el ámbito de la robótica, abordando los avances que ha experimentado desde sus inicios hasta la actualidad. Dicha evolución se ha visto influenciada por la aparición de nuevas tecnologías y enfoques de trabajo. El objetivo principal de este estudio implica la aplicación de principios científicos y tecnológicos con el propósito de desarrollar una solución práctica. Esta solución no solo busca beneficiar a la universidad de manera específica, sino que también aspira a contribuir potencialmente al avance de la comunidad académica en general.

3.2. TÉCNICAS E INSTRUMENTOS

3.2.1. Observación:

Esta investigación se llevó a cabo en el laboratorio de robótica, donde se evidenció que el control de robots constituye la conexión convencional, abarcando la comunicación

entre robots y la interacción de movimientos. Este proceso, sin embargo, presenta riesgos inherentes, como el potencial riesgo de daño a elementos cruciales del robot, incluyendo tarjetas utilizadas, actuadores y sensores. Este análisis destaca la necesidad de abordar los desafíos relacionados con la seguridad y eficiencia en el control robótico, señalando áreas críticas que requieren atención para garantizar un rendimiento óptimo y prevenir posibles daños en el equipamiento.

3.2.2. Documentación:

La elaboración del documento integral de desarrollo constituye una fase crucial y meticulosa del proceso, destinada a registrar de manera detallada cada etapa, desde la identificación inicial de desafíos hasta la implementación de soluciones específicas. Este documento, elaborado con minuciosidad, no se limita únicamente al entorno temático y la problemática identificada, sino que también se enfoca en el logro integral de cada objetivo establecido.

En su esencia, el documento ofrece una visión completa y estructurada del proyecto, abordando no solo los hitos alcanzados, sino también los obstáculos superados. Cada desafío específico encontrado durante el desarrollo se destaca con claridad, proporcionando una comprensión profunda de las dificultades enfrentadas. Más crucial aún, se presenta una exhaustiva exposición de las soluciones estratégicas implementadas para abordar eficazmente cada problema identificado.

Este documento va más allá de ser simplemente un registro histórico del progreso; se erige como un recurso esencial para comprender el contexto en el que se llevó a cabo el proyecto. Proporciona una base sólida para evaluar el rendimiento del equipo en relación con los objetivos establecidos. Además, se convierte en una herramienta invaluable para extraer lecciones valiosas del proceso de investigación y desarrollo, contribuyendo así al crecimiento continuo y la mejora de futuros proyectos.

La documentación exhaustiva no solo sirve como un testimonio detallado de la evolución del proyecto, sino que también se posiciona como un componente fundamental para el aprendizaje organizacional. Facilita la transmisión de conocimientos y experiencias acumuladas, permitiendo a los equipos futuros comprender los desafíos previamente abordados y beneficiarse de las soluciones eficaces implementadas en este proceso de desarrollo meticuloso.

3.2.3. Entrevista

La entrevista fue realizada al Ingeniero Christian Junta, Técnico de Investigación de la Universidad Tecnológica Indoamérica, quien fue aportador de la información para poder llevar a cabo la realización de la aplicación para el control del robot. Esta entrevista se realizó con el fin de recopilar información importante en donde se pueda tomar en cuenta ciertos aspectos para el desarrollo del proyecto de a aplicación basada en ros2 para el control del robot. La entrevista se realizó únicamente al Ingeniero Christian ya que él es el creador del robot que es utilizado en esta tesis, por tal motivo únicamente se realizó al creador del robot quien tiene el conocimiento del funcionamiento completo y de la estructura del robot manipulador móvil. A continuación, se presenta las preguntas y respuestas realizadas en la entrevista.

¿Podría describir cómo se manejan actualmente las operaciones y tareas llevadas a cabo por el robot manipulador en su entorno?

"Se trata de un sistema fundamentado en motores inteligentes que operan mediante una comunicación serial utilizando una ESP32. Esta plataforma posibilita el envío de datos hacia la nube, donde se almacenan en una base de datos no estructurada. En esta base, se registran las actividades del robot de manera continua. Además, se implementa una comunicación a través de iOS que permite el control remoto del robot mediante un teléfono móvil con conexión a internet".

¿Considera usted factible el desarrollo de una aplicación basada en ROS2 para el control del robot manipulador con motores inteligentes de la FAINPRO? En caso afirmativo, ¿cuáles son los principales beneficios que anticipa?

"La realización de la aplicación es totalmente factible, especialmente al optar por ROS2, que representa una estandarización en el ámbito de la robótica. Con ROS2, podemos esperar una comunicación más rápida y eficiente, lo que optimiza el uso de los actuadores inteligentes. Además, al emplear un solo código para la conexión, simplificamos el proceso y aprovechamos al máximo el potencial de los componentes".

¿Actualmente se utiliza ROS2 como plataforma principal para el control y operación del robot manipulador en su institución o entorno? En caso contrario, ¿cuál es la plataforma preferida y por qué?

“No, en este caso aún no se emplea esta nueva tecnología. Actualmente, se utilizan las librerías de Arduino, con Arduino actuando como controlador a través de la tarjeta ESP32. Si bien los tiempos de respuesta son buenos, no alcanzan el nivel de eficiencia que ofrece ROS2”.

Desde su perspectiva, ¿cree que es necesario implementar un sistema centralizado basado en ROS2 para el control integral del robot manipulador, considerando las capacidades de los motores inteligentes de la FAINPRO? ¿Cuáles podrían ser los beneficios clave de esta centralización en términos de eficiencia y coordinación?

“El tiempo de respuesta y el tiempo de procesamiento han mejorado significativamente. Anteriormente, experimentábamos demoras de entre 3 a 4 segundos o incluso más, a pesar de la velocidad disponible. Siempre había retrasos al esperar que la información se cargara en la nube. Sin embargo, al migrar a ROS2 y centralizar nuestras operaciones, la comunicación ahora es más rápida y estable”.

¿Cómo se maneja la seguridad en el sistema del robot manipulador en su entorno actual y cómo cree que la implementación de una aplicación basada en ROS2 podría contribuir a mejorar estos aspectos?

“En términos de seguridad, aunque el robot cuenta con un sistema robusto, enfrenta problemas como la pérdida de datos, retrasos en el tiempo de respuesta y una comunicación poco estable. Por lo tanto, considero que mejorar estos aspectos clave sería beneficioso, especialmente mediante la implementación de un código más apropiado y limpio”.

¿Considera que una aplicación basada en ROS2 podría ofrecer mejoras significativas en la gestión?

“Sí, precisamente debido a su estandarización y centralización específicamente para el campo de la robótica, el uso de esta forma de comunicación permitirá que cualquier persona con conocimientos sobre ROS2 pueda manejar el entorno de manera apropiada”.

¿Cuáles son las principales capacidades y características de los motores inteligentes de la FAINPRO que considera más relevantes para la implementación de una aplicación de control para el robot manipulador omnidireccional?

“Los motores Dynamixel ofrecen una amplia gama de información gracias a su naturaleza inteligente plug and play. Estos datos incluyen el número de revoluciones, la velocidad

actual, la temperatura de funcionamiento, y disponen de un sistema de seguridad automático que desconecta los motores si se detecta algún comportamiento anormal o fuera de lo esperado, con el fin de proteger su integridad estructural. Además, cuentan con una memoria integrada que permite almacenar sus posiciones, lo que garantiza que, al reiniciar el robot, pueda retomar su posición inicial para llevar a cabo sus funciones. Otro aspecto destacado es la presencia de un procesador, que constituye una de sus características principales".

Conclusión de la entrevista:

En resumen, el sistema actual del robot manipulador de la FAINPRO se basa en motores inteligentes con comunicación serial a través de una ESP32, enviando datos a una base de datos no estructurada en la nube. Aunque el control se realiza mediante librerías de Arduino, se considera factible y beneficioso migrar a una aplicación basada en ROS2. Los beneficios anticipados incluyen una mejor respuesta en tiempo de comunicación, estandarización para la robótica y mayor eficiencia en la gestión centralizada. Aunque aún no se utiliza ROS2, se espera que mejore la seguridad, el tiempo de respuesta y la estabilidad de la comunicación. Las capacidades clave de los motores Dynamixel incluyen información detallada, sistema de auto seguridad y memoria para almacenar posiciones, destacando su relevancia para una aplicación de control omnidireccional. La implementación de ROS2 se percibe como una opción que mejoraría significativamente la eficiencia y coordinación en el manejo del robot manipulador.

CAPÍTULO IV

4. PROPUESTA Y RESULTADOS ESPERADOS

4.1. ESTUDIO DE FACTIBILIDAD

La factibilidad para llevar a cabo esta tesis fue rigurosamente evaluada y aprobada por el decano de la facultad de Ingeniería, Industria y Producción, asegurando así la viabilidad del proyecto. El enfoque principal de este estudio de factibilidad reside en la evaluación detallada de la posibilidad de desarrollar una aplicación basada en ROS2 para el control de robots que cuentan con motores inteligentes provenientes de la FAINPRO. La aplicación tiene como objetivo principal potenciar la eficiencia y versatilidad en el control de estos robots al aprovechar las capacidades avanzadas de los motores inteligentes incorporados. Se busca no solo implementar una solución tecnológica innovadora, sino también contribuir al avance y la aplicación práctica de tecnologías robóticas de desarrollo. El respaldo institucional y la aprobación de la factibilidad sientan las bases para llevar a cabo un proyecto que no solo aborda los desafíos tecnológicos actuales, sino que también promueve el progreso en la intersección de la robótica y la inteligencia artificial.

4.1.1. Factibilidad Operativa

La factibilidad operativa en este caso implica asegurarse de que la organización cuente con el personal capacitado, procesos operativos bien integrados, tecnología compatible y una sabiduría organizativa dispuesta a adoptar la aplicación basada en ROS2 para el control de los robots. Para ello, se deben tener en cuenta los siguientes aspectos.

Permisos: Para llevar a cabo el proyecto de desarrollo de la aplicación basada en ROS2 para el control de robots con motores inteligentes de la FAINPRO, se han gestionado los permisos necesarios, incluyendo la obtención de la aprobación del decano y del laboratorista responsable del laboratorio de prácticas. Estas autorizaciones son fundamentales para garantizar la conformidad con las políticas internas de la institución y asegurar la legalidad y ética en la ejecución del proyecto. Se ha llevado a cabo un proceso formal de solicitud y aprobación, obteniendo el respaldo necesario de las autoridades pertinentes para proceder con el desarrollo de la aplicación en el entorno del laboratorio.

Conocimientos programación: La ejecución exitosa del proyecto requiere un sólido dominio de la programación, centrándose especialmente en el lenguaje C y en el entorno ROS2. El equipo de desarrollo debe demostrar habilidades avanzadas en la programación en C, un lenguaje crucial para la implementación eficiente de algoritmos y el control de hardware. Además, se espera que los miembros del equipo posean un profundo conocimiento de ROS2, un marco de desarrollo robótico que facilitará la comunicación y la coordinación entre los diversos componentes de la aplicación y los robots. La familiaridad con las prácticas de programación en ROS2 permitirá una integración fluida y una gestión eficaz de los recursos robóticos.

Conocimientos electrónica: Dado que la aplicación implica el control de robots con motores inteligentes, es esencial contar con conocimientos profundos en electrónica. El equipo de desarrollo debe tener experiencia en el diseño y la implementación de circuitos electrónicos, así como en la integración de componentes electrónicos con sistemas robóticos. Esta competencia garantizará un control preciso y seguro de los motores y la electrónica asociada en los robots.

Conocimientos en comunicación: El proyecto requiere conocimientos en comunicación serial, Wi-Fi y Bluetooth. Se requiere un sólido entendimiento de la comunicación serial, fundamental para la interacción eficiente entre la aplicación y los componentes electrónicos, como los motores inteligentes de los robots. El equipo debe demostrar experiencia en la configuración y gestión de comunicaciones serie, asegurando una transmisión de datos fiable y en tiempo real. Además, se busca un conocimiento profundo de las redes Wi-Fi para garantizar una conectividad estable y eficiente, permitiendo la comunicación fluida entre la aplicación y los robots en entornos más amplios. Asimismo, se valora la experiencia en tecnología Bluetooth, la cual puede ser esencial para la comunicación cercana y la interacción con dispositivos periféricos.

4.1.2. Factibilidad Técnica

En la tabla 2 y 3 se puede observar la descripción de cada uno de los softwares y hardware que se utilizaran en el desarrollo de este proyecto. La Universidad Indoamérica proporciona un entorno completo para el desarrollo del proyecto, equipado con motores, tarjetas, un robot y una computadora. Además, ofrece licencias de software, como Visual Studio Code, asegurando compatibilidad, integración y posibles actualizaciones a largo plazo. Esta infraestructura robusta permite un desarrollo eficiente y pruebas realistas. La

presencia de tecnología avanzada, junto con el respaldo de licencias de software clave, no solo mejora la eficacia del proyecto, sino que también facilita la capacitación interna y garantiza la alineación con los estándares de la industria. La universidad demuestra un compromiso con la sostenibilidad al proporcionar bases para actualizaciones continuas, asegurando que el proyecto esté preparado para enfrentar los desafíos tecnológicos futuros.

Tabla II

Descripción de softwares.

SOFTWARE	Descripción
Licencias	Se deberá adquirir licencias de software necesario para el desarrollo y ejecución del proyecto.
Ubuntu 22.04	Ubuntu 22.04: Sistema operativo Linux de código abierto. Interfaz amigable, rendimiento mejorado, actualizaciones de software, seguridad fortalecida y soporte a largo plazo.
Visual Studio Code	Editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Altamente personalizable y compatible con varios lenguajes de programación.
Arduino IDE	"Arduino IDE: Plataforma de desarrollo intuitiva para programar proyectos electrónicos con facilidad y versatilidad creativa."

Tabla III

Descripción de hardware.

HARDWARE	Descripción
Motores	Se requerirá la adquisición de motores específicos para el desarrollo del proyecto.
Tarjetas	Serán necesarias tarjetas de control o procesamiento para la integración con los motores.
Robot	Se planifica utilizar un robot con capacidades específicas para el propósito del proyecto.
Computadora	Se utilizará una computadora con especificaciones técnicas adecuadas para el desarrollo.

4.1.3. Factibilidad Económica

La factibilidad económica de este proyecto se ve respaldada de manera significativa por la provisión integral de recursos por parte de la Universidad. La adquisición de motores Dynamixel, un robot, tarjetas OpenCM y ESP32, junto con otros elementos esenciales, representa un ahorro sustancial para el presupuesto del proyecto. La universidad asume la responsabilidad de suministrar estos materiales clave, eliminando así la necesidad de gastos adicionales en este sentido.

Esta situación beneficia la viabilidad económica del proyecto al reducir significativamente los costos de adquisición de hardware y tecnología. La disponibilidad de estos recursos también agiliza el proceso de implementación, ya que el equipo puede centrarse directamente en el desarrollo y la programación sin preocuparse por la obtención de componentes costosos.

La ausencia de gastos adicionales no solo simplifica la planificación financiera, sino que también contribuye a una mayor eficiencia en la ejecución del proyecto. Al minimizar los costos asociados con la adquisición de equipos y materiales, se optimizan los recursos financieros disponibles para otros aspectos críticos del proyecto, como capacitación, documentación y posibles mejoras futuras.

En la Tabla 4, se presenta de manera detallada la adquisición de cada elemento necesario para la construcción del robot manipulador móvil, así como los costos asociados a cada uno de estos componentes por parte de la universidad.

Tabla IV

Detalle de costos del robot.

Elementos	Costo
Motores inteligentes Dynamixel XC430	479,60
Infraestructura del robot manipulador móvil	417,47
Tarjeta ESP32	18,00

Tarjeta Open-CM	91,00
Extensión Open-CM 485 EXP	64,00
TOTAL $	**1070,07**

4.2. METODOLOGÍA

La elección de la metodología Scrum para el desarrollo de la aplicación basada en ROS2 para controlar robots con motores inteligentes de la FAINPRO se fundamenta en su flexibilidad y adaptabilidad a cambios tecnológicos emergentes, su enfoque interactivo e incremental que permite entregas regulares de funcionalidades tangibles fomenta la colaboración constante y la comunicación entre los miembros del equipo, y facilita la mejora continua mediante la reflexión y ajuste tras cada sprint. La capacidad de Scrum para proporcionar entregas rápidas de valor y su enfoque en satisfacer las necesidades del cliente resultan especialmente relevantes en proyectos tecnológicos complejos como el desarrollo de aplicaciones robóticas.

En este proyecto se utilizará la metodología SCRUM que cuenta con los siguientes pasos:

4.2.1. Inicio

La primera fase se encarga de estudiar y analizar el proyecto identificando las necesidades básicas del sprint.

4.2.2. Planificación y Estimación

El equipo Scrum planifica cómo llevará a cabo el trabajo, dividiéndolo en tareas más pequeñas si es necesario. La segunda fase de Scrum incluye normalmente los siguientes pasos:

- Crear, estimar y comprometer historias de usuario.
- Identificar y estimar tareas.
- Crear el sprint backlog o iteración de tareas.

4.2.3. Implementación

Implementación del diagrama de funcionamiento de la aplicación con ROS2.
Se realizan reuniones diarias cortas llamadas Daily Scrum para revisar el progreso y ajustar la planificación según sea necesario.

El objetivo es tener una versión potencialmente entregable del producto al final del sprint.

4.2.4. Revisión y Retrospectiva

Después de haber completado la maquetación e implementación, es necesario llevar a cabo una revisión del proceso, la cual implica realizar una autocrítica o evaluación interna del equipo en relación con su propio desempeño. Algunos de los pasos cruciales a realizar en esta etapa incluyen:

- Demostrar y validar el sprint.
- Retrospectiva del sprint.

4.2.5. Entrega del producto

Hacemos referencia a la conclusión del proyecto y la entrega del producto, donde se espera que completes dos tareas específicas:

- Enviar entregables.
- Enviar retrospectiva del proyecto.

4.2.6. Modelo de aceptación tecnológica (TAM)

El Modelo de Aceptación de Tecnología (TAM) se erige como una estructura conceptual diseñada para explorar y analizar los múltiples factores que inciden y, en última instancia, definen el comportamiento de las personas respecto a la aceptación de nuevas tecnologías. En esencia, el TAM propone que la percepción individual de la utilidad y la facilidad de uso desempeñan roles fundamentales al influir en la intención y la aplicación real de una tecnología de la información. Este marco teórico, propuesto por David en 1989, se ha convertido en un pilar en la investigación de la adopción tecnológica al proporcionar una estructura sistemática para entender cómo las actitudes y las percepciones moldean la interacción humana con la innovación.

Para profundizar en la comprensión del TAM, es esencial destacar que la percepción de utilidad se refiere a la creencia individual en que la tecnología en cuestión mejorará su desempeño o facilitará la realización de tareas específicas. En paralelo, la percepción de facilidad de uso aborda la evaluación subjetiva de la simplicidad y accesibilidad de la tecnología. Estas dos percepciones, según el TAM, se combinan de manera determinante para influir en la intención de adoptar la tecnología y, en última instancia, en su uso

práctico. Así, el modelo TAM no solo proporciona una estructura teórica robusta sino también un marco práctico para investigadores y profesionales que buscan comprender y mejorar la aceptación de tecnologías de la información [22].

En la Tabla 5 se presentan las preguntas que se emplearon para evaluar la aceptación de la nueva tecnología, específicamente en relación con el robot manipulador. Estas preguntas han sido seleccionadas siguiendo los principios delineados en el Modelo de Aceptación Tecnológica (TAM). La tabla distingue claramente entre dos categorías fundamentales: la utilidad percibida, compuesta por 7 preguntas que exploran la percepción de beneficios y mejoras en el desempeño, y la factibilidad de uso percibida, que incluye 7 preguntas centradas en la evaluación subjetiva de la facilidad y accesibilidad en la utilización de la tecnología. Este enfoque estructurado proporciona una base metodológica sólida para examinar de manera integral la aceptación de la nueva tecnología, permitiendo una evaluación detallada de las perspectivas tanto de utilidad como de factibilidad de uso.

Tabla V

Cuestionario para aceptación de la tecnología.

	Utilidad Percibida
UP 1	1 ¿En qué medida crees que esta aplicación mejora la eficiencia en el control de los robots en comparación con métodos de control tradicionales?
UP 2	2 ¿Qué tan útil encuentras la información proporcionada por la aplicación para monitorear y ajustar el rendimiento de los motores inteligentes durante las operaciones de los robots?
UP 3	3 ¿En qué medida crees que esta aplicación se adapta a las necesidades específicas de control de los robots con motores inteligentes de FAINPRO en comparación con otras soluciones de control disponibles en el mercado?
UP 4	4 ¿Qué tan probable sería que recomendaras esta aplicación a otros usuarios que trabajan con robots equipados con motores inteligentes de FAINPRO?
UP 5	5 ¿Cómo calificarías tu satisfacción general con la experiencia de usar esta aplicación para el control de robots con motores inteligentes de FAINPRO?

UP 6	6 ¿Qué tan valiosa encuentras la capacidad de esta aplicación para realizar análisis en tiempo real de los datos recopilados durante las operaciones de los robots?
UP 7	7 ¿En qué medida crees que esta aplicación mejora la precisión y fiabilidad del control de los robots en comparación con otros sistemas de control disponibles en el mercado?
	Factibilidad de Uso Percibida
FP 1	1 ¿Qué tan fácil de usar encuentras la interfaz de la aplicación para controlar los robots con motores inteligentes de FAINPRO en comparación con otras herramientas de control disponibles?
FP 2	2 ¿Qué tan claro te resulta el proceso de configuración y ajuste de los parámetros de los motores inteligentes utilizando esta aplicación en comparación con otros sistemas de control?
FP 3	3 ¿Qué tan confiable percibes la comunicación entre la aplicación y los motores inteligentes durante las operaciones de los robots?
FP 4	4 ¿En qué medida te sientes cómodo/a utilizando esta aplicación para el control de los robots con motores inteligentes en diferentes escenarios y entornos de trabajo?
FP 5	5 ¿Qué nivel de confianza tienes en la seguridad de la comunicación entre la aplicación y los motores inteligentes en términos de protección contra posibles vulnerabilidades?
FP 6	6 ¿Qué tan satisfactoria es la respuesta de la aplicación a tus comandos de control durante la operación de los robots?
FP 7	7 ¿Cuán fácil te resulta aprender a utilizar las funciones avanzadas de esta aplicación para el control de los robots con motores inteligentes?

4.3. APLICACIÓN DE LA METODOLOGÍA DE DESARROLLO

4.3.1. Fase 1 Inicio:

Basándonos en la entrevista previamente llevada a cabo, se identifica como una necesidad prioritaria la creación de un nuevo software controlador con el objetivo de mejorar de manera efectiva la operatividad del robot, facilitando la interacción entre sus

componentes. Además, en la recopilación de datos, se considera la creación de una aplicación que optimice el control del robot, eliminando la necesidad constante de utilizar diversas plataformas para activar los actuadores. La problemática principal que impulsa esta fase del proyecto es la ineficiencia operativa observada en robots equipados con motores inteligentes de la FAINPRO por falta de nuevas tecnologías para el uso de los mismos componentes.

4.3.2. Fase 2 Planificación y Estimación:

4.3.2.1. Identificación de Requisitos Clave:

En los encuentros iniciales, se trabajó en la identificación y definición de los requisitos clave del proyecto. Estos requisitos incluyeron, entre otros aspectos, la implementación de la comunicación con ROS2, el diseño del sistema de control para motores inteligentes. Tomando en cuenta la problemática encontrada en el funcionamiento y control del robot manipulador.

4.3.2.2. Creación de sprints:

Sprint 1: Diseño y Planificación Inicial

- Planificación del Sprint: Este esprint tiene como objetivo principal terminar el software de control y el diseño de la aplicación basada en ROS2 para sus pruebas respectivas.
- Investigación de Requisitos: Los requisitos específicos dentro de la aplicación son: conexión estable entre el robot y la aplican, un diseño fácil de usar y entender, los botones principales para cada acción que se realice al robot como son: (conectar, adelante, atrás, izquierda, derecha y salir).
- Diseño de la Arquitectura del Software: Se ha creado una estructura inicial para la aplicación basada en ROS2, considerando la interacción con los motores inteligentes. En la Figura 4 se muestra el diseño de la interfaz de usuario para el robot, la cual cuenta con todas las funciones básicas y es muy intuitiva de usar. Esta estructura inicial ha sido diseñada para proporcionar una experiencia de usuario fluida y eficiente. La interfaz de usuario ha sido desarrollada de manera que sea accesible para usuarios de todos los niveles de experiencia, facilitando así la operación del robot en diversos entornos.

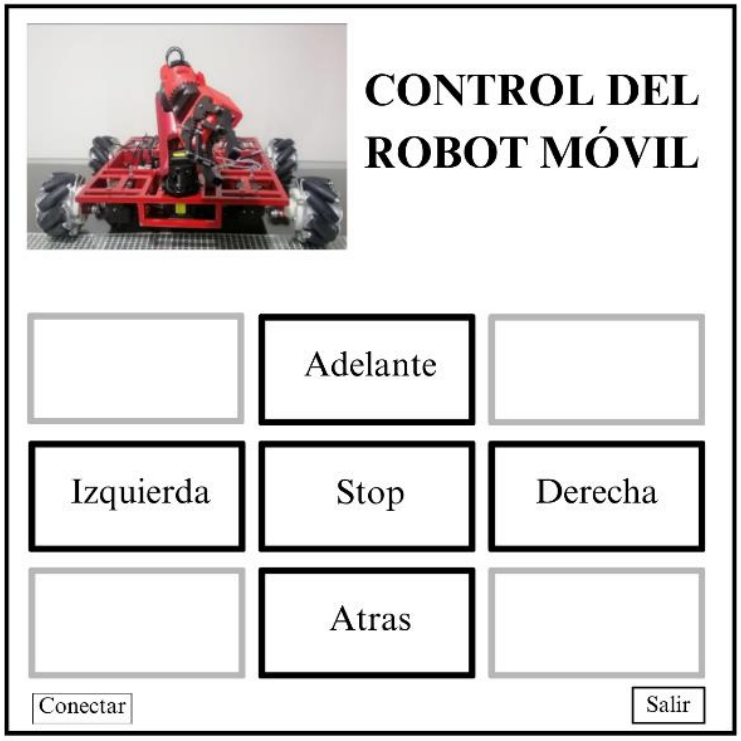


Fig. 4: Diseño de la interfaz del controlador.

4.3.2.3. Identificación de Componentes Necesarios:

Los componentes entre hardware y software identificados son los motores inteligentes Dynamixel, las tarjetas ESP32 y Open-CM con su respectiva expansión para la comunicación entre las tarjetas y los actuadores, que ayudaran a ser una memoria principal quien ayuda en la activación de los motores. Además, se crea el cronograma de trabajo tanto para el autor del proyecto como para el tutor de la tesis:

Semana 1-2: Codificación en ROS2 (C) y Desarrollo de Interfaz en Python

Inicio de la codificación de la aplicación móvil en ROS2 utilizando el lenguaje C.
Desarrollo de la interfaz de control del robot en Python.
Revisiones continuas de la documentación.

Semana 3-5: Avances y Revisión Continua
Implementación de funcionalidades específicas en la aplicación móvil.
Revisión continua de la documentación y ajustes.
Sesiones regulares de revisión con el Ing. José Varela.

Semana 6-7: Control y Optimización

Enfoque en la implementación del control remoto.
Optimización de la aplicación para mejorar rendimiento y eficiencia.
Entrevista con el Ing. Christian para obtener aportes técnicos.

Semana 8-9: Desarrollo de la Interfaz de Control
Refinamiento de la interfaz de control del robot.
Pruebas de usabilidad y ajustes.
Sesiones de revisión adicionales con el Ing. José Varela.

Semana 10-11: Revisiones Finales y Preparación para Presentación
Revisión final de la documentación y la aplicación.
Incorporación de aportes del Ing. Christian y ajustes finales.
Preparación para la presentación final.

Semana 12: Presentación Final y Entrega
Presentación de la aplicación y la tesis ante el Ing. José Varela.
Recolección de feedback y retroalimentación.
Finalización de la entrega de la aplicación y la documentación.

En la Figura 5 se evidencia el registro detallado de asistencia con fechas y horas, documentando cada avance tanto en el desarrollo del producto como en la elaboración del documento. Esta documentación se erige como una parte esencial para llevar a cabo el proceso de desarrollo de manera efectiva y asegurar una conclusión exitosa, cumpliendo así con cada uno de los objetivos establecidos en este trabajo.

Esta planificación detallada garantiza un enfoque estructurado y colaborativo para la entrega final, asegurando que tanto la aplicación como la tesis cumplan con los estándares establecidos y reflejen un alto nivel de calidad en el desarrollo del proyecto

Seguimiento a tutorías

Proyecto de Grado: DESARROLLO DE UNA APLICACIÓN BASADA EN ROS2 PARA EL CONTROL DE ROBOTS CON MOTORES INTELIGENTES DE LA FAINPRO

Fecha	Hora	Tutor	Lugar	Asistió	% Avance	Sugerencias	Documento	
2023-11-17	10:41 a.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	35%	CONEXCIÓN A MICROROS CON ESP32		CONFIRMADA
2023-11-22	01:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	40%	CODIFICACIÓN LED ROS2		CONFIRMADA
2023-11-29	01:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	45%	FUNDAMENTACIÓN TEORICA		CONFIRMADA
2023-12-06	01:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	50%	REVISIÓN DE CODIGO LIBRERIA TWIST		CONFIRMADA
2023-12-20	01:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	50%	PROGRAMACIÓN DEL LA COMUNICACIÓN CON ROS2		CONFIRMADA
2024-01-10	01:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	60%	DESARROLLO DE LA INTERFAZ DE USUARIO.		CONFIRMADA
2024-01-17	01:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN	SI	70%	REDACCIÓN DE LA FACTIBILIDAD		CONFIRMADA
2024-02-01	03:30 p.m.	VARELA ALDAS JOSE LUIS	DIRECCIÓN DE INVESTIGACIÓN					

Fig. 5: Evidencia de asistencia tutoría de tesis.

4.3.2.4.Roles y Responsabilidades:

Autor del Proyecto (Desarrollador y Codificador):

Como autor del proyecto y principal desarrollador, tengo la responsabilidad de llevar a cabo la codificación y elaboración de la aplicación móvil. Mis funciones abarcan desde el diseño estructural del software basado en ROS2 hasta la implementación de las funcionalidades específicas requeridas para el control de robots con motores inteligentes. Además, soy responsable de documentar de manera detallada cada fase del desarrollo, incluyendo desafíos enfrentados y soluciones implementadas.

Responsabilidades Principales:

- Codificación y desarrollo de la aplicación móvil.

- Diseño de la estructura de software basada en ROS2.

- Implementación de funcionalidades específicas para el control de robots.

- Documentación exhaustiva del proceso de desarrollo.

- Colaboración con el Ing. José Varela para la revisión y mejora continua del producto.

Tutor de Tesis y Revisor

(Ing. José Varela):

El Ing. José Varela cumple un papel crucial como tutor de tesis y revisor en la fase de entrega. Su experiencia y conocimientos se enfocan en garantizar que el proyecto cumpla con los estándares académicos y profesionales requeridos. Además de su rol

académico, el Ing. José Varela se involucra activamente en la revisión del documento y proporciona aportes significativos para mejorar tanto la calidad del informe como la funcionalidad de la aplicación.

Sprint 2: Desarrollo del Controlador

Proceso de Desarrollo:

Para la creación del software de control, se inició instalando la extensión de Ubuntu 22.04 en el entorno de Windows. Posteriormente, se procedió a la instalación del software ROS2, marcando el inicio efectivo del desarrollo del controlador. Se llevaron a cabo pruebas iniciales utilizando los ejemplos por defecto proporcionados por ROS2, y luego se avanzó en la implementación creando un controlador inicial. Este primer controlador tenía como objetivo encender el LED interno de la ESP32, estableciendo así una base funcional para el desarrollo del controlador del robot.

Para iniciar la creación del controlador del robot, se tomó como punto de partida el ejercicio del LED, el cual operaba como un suscriptor (subscriber). A partir de este punto, se procedió a instalar las bibliotecas necesarias tanto para ROS2 como para la ESP32. Estas bibliotecas fueron integradas directamente en el código desarrollado en lenguaje C, estableciendo así las bases fundamentales para la elaboración del controlador del robot.

Con las bibliotecas esenciales debidamente instaladas, se procedió a configurar la creación del ejecutor y la lectura de datos en el controlador del robot. La utilización de la librería 'Twist' facilitó la lectura de datos decimales, permitiendo así la construcción de la trama en el carácter denominado 'control'. En este proceso, se inicializaron las variables clave, a saber: U1, U2 y W, las cuales desempeñarían un papel esencial en el control del robot y en la lectura de la entrada de datos.

Una vez completadas las etapas anteriores, se procedió a implementar los controles correspondientes, centrándose en los signos de los valores ingresados. Para llevar a cabo este control, se emplearon estructuras condicionales “if”, asignando la letra 'a' en caso de valores positivos y 'b' para valores negativos. Además, se implementó un control adicional para los datos, donde se evaluaron los ceros mediante otra estructura “if”. En esta evaluación, si los valores se encontraban en el rango de 0 a 9, se incrementaban dos ceros; para valores de 10 a 99, se incrementaba un cero; y si el valor

era igual a 100 o superior, se mantenían los tres dígitos originales. Este proceso garantizó la adecuada formación de la trama, permitiendo así la activación del robot.

Con la configuración previa completada de manera exitosa, se llevó a cabo la transmisión de datos mediante el puerto serial hacia la placa OpenCM. Una vez concluido este proceso y contando con el código integral, se procedió a cargar el software en la tarjeta ESP32. Esta tarjeta juega un papel central al contener el código del controlador, y está conectada a la tarjeta OpenCM, la cual a su vez actúa como la interfaz principal para los motores inteligentes del robot. En este contexto, la tarjeta OpenCM se posiciona estratégicamente como el punto de conexión para la ejecución de los motores, permitiendo así una integración cohesiva entre la interfaz de control y la funcionalidad motora del robot.

Una vez completada la configuración del controlador, se emprendió la creación de la interfaz de control para el robot. Esta interfaz fue desarrollada en Python mediante Visual Studio Code, considerando la necesidad de utilizar bibliotecas de ROS2 para establecer la conexión con la tarjeta ESP32 y facilitar el envío de datos. La interfaz del control se compone de siete botones fundamentales, a saber: 'Adelante', 'Atrás', 'Izquierda', 'Derecha', 'Stop', 'Conectar' y 'Salir'. Cada uno de estos botones cumple una función específica en la gestión y dirección del robot, proporcionando así una interfaz intuitiva y funcional para el usuario

Desarrollo del Controlador: En la figura 6 se puede observar la codificación realizada en donde se utilizó la librería Kivy, que es una biblioteca de Python para crear interfaces de usuario multiplataforma, interactivas y creativas.

Implementación de la Interfaz: En la figura 7, como se puede mostrar esta ya la interfaz implementada codificada en el lenguaje Python ya que es compatible con ROS2.

Integración Inicial: Realizar pruebas de integración básicas para garantizar la comunicación efectiva entre la aplicación y los componentes de hardware.

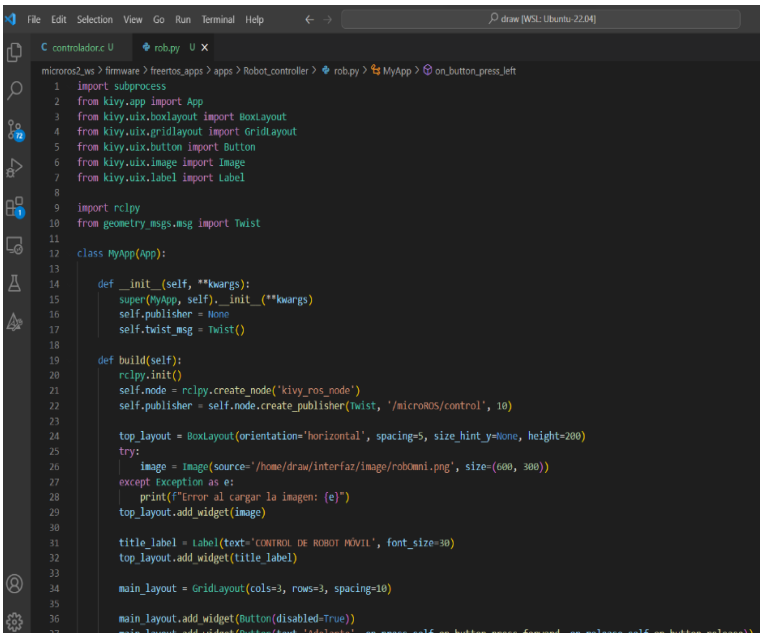


Fig. 6: Código de la interfaz de usuario.

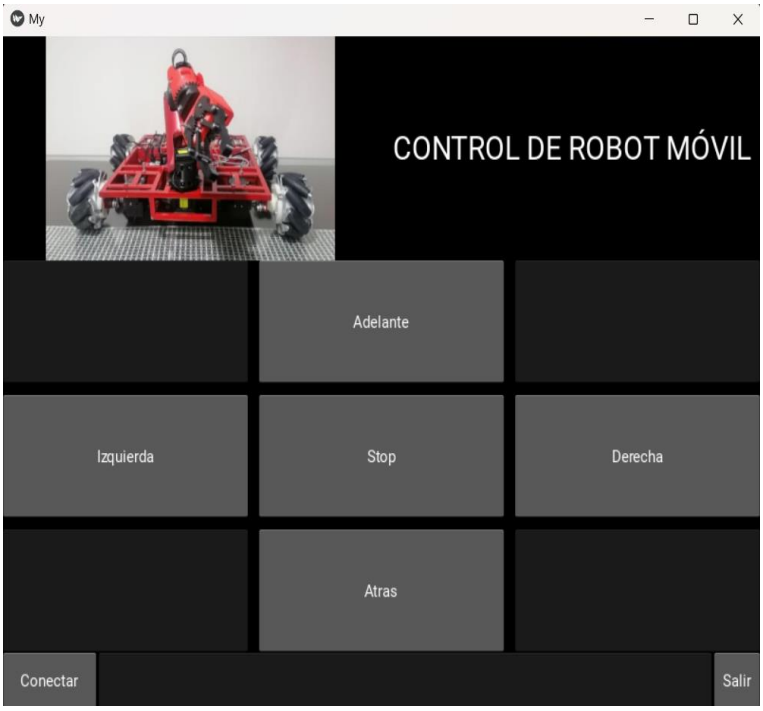


Fig. 7: Implementación de la interfaz.

Sprint 3: Pruebas y Evaluación de Funcionalidad

Pruebas Unitarias y de Integración: En la figura 8 se puede observar las pruebas realizadas del software controlador creada en el middleware ROS2.

Evaluación del Rendimiento: En la figura 9 se puede observar las pruebas realizadas con el controlador desarrollada, en donde se muestra en la consola la actividad presionada y enviada al robot.

Identificación y Resolución de Problemas: Durante el desarrollo del software, nos enfrentamos a diversos desafíos. Inicialmente, al enviar datos, nos encontramos con la presencia de caracteres desconocidos que no formaban parte de la trama de activación de los motores. Además, identificamos un fallo en la interfaz, relacionado con el botón de conexión, el cual no respondía ni se ejecutaba correctamente. Para abordar estos problemas, implementamos alternativas de conexión que permitieron solventar los errores detectados. Asimismo, para asegurar el envío de datos sin contratiempos, optamos por cambiar la señal de transmisión.

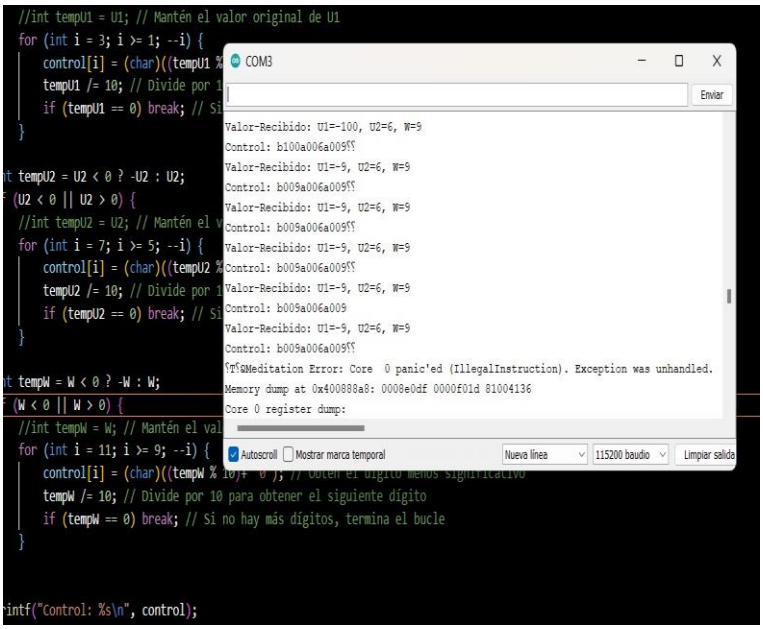


Fig. 8: Pruebas del funcionamiento del software creado.

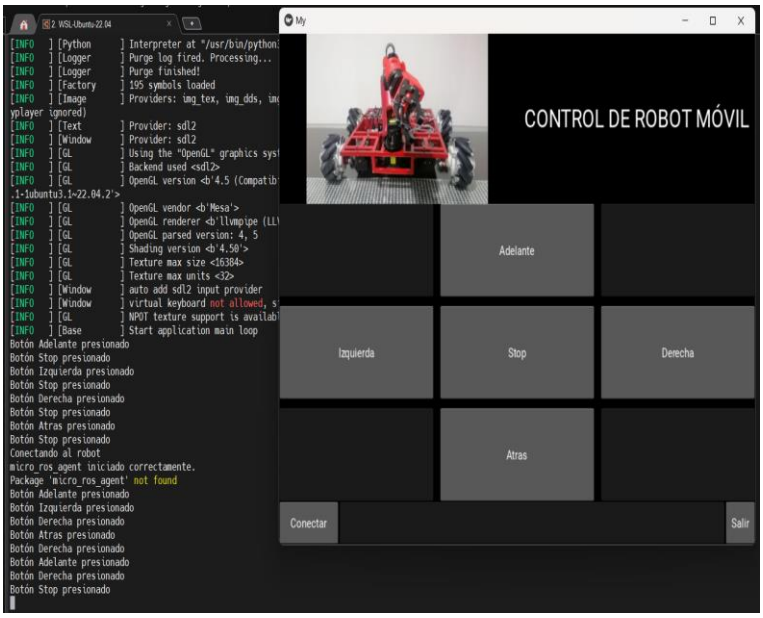


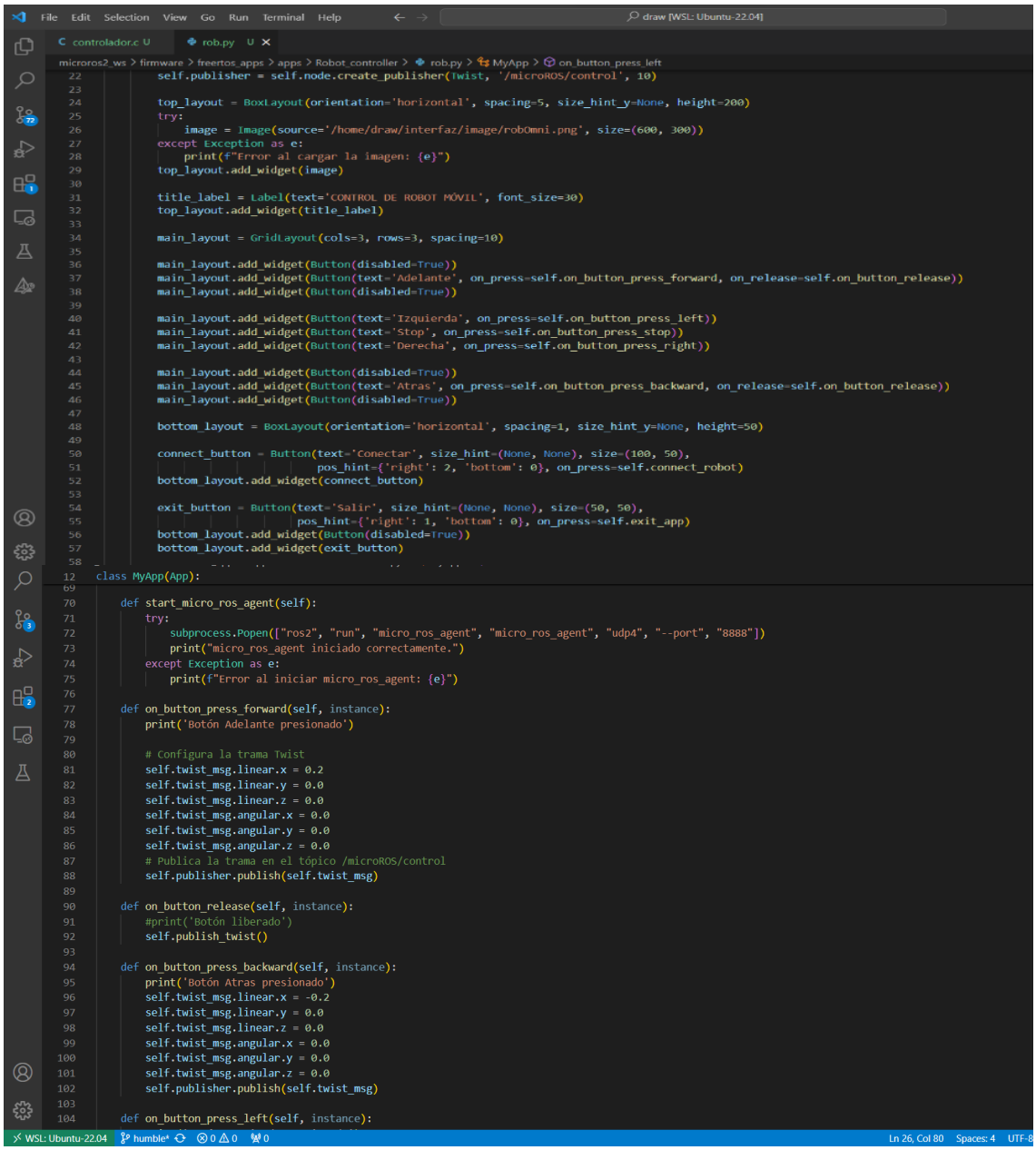
Fig. 9: Pruebas con la aplicación.

- Sprint 4: Optimización y Documentación Final

Optimización del Código: Refinar y optimizar el código de la aplicación para mejorar el rendimiento y la eficiencia quitando ciertas líneas de código que no tienen ninguna función en la ejecución y agrupando de mejor manera para recortar el código y sea mejor presentado.

4.3.3. Fase 3 Implementación:

Se centra en la ejecución de los Sprints planificados, donde se lleva a cabo el desarrollo de la aplicación basada en ROS2. Juntamente con el software creado para el control del robot. En la figura 10 se puede observar la implementación de todos los elementos y las acciones que realizara cada uno de los botones para la activación del robot.



```
        self.publisher = self.node.create_publisher(Twist, '/microROS/control', 10)

        top_layout = BoxLayout(orientation='horizontal', spacing=5, size_hint_y=None, height=200)
        try:
            image = Image(source='/home/draw/interfaz/image/robOmni.png', size=(600, 300))
        except Exception as e:
            print(f"Error al cargar la imagen: {e}")
        top_layout.add_widget(image)

        title_label = Label(text='CONTROL DE ROBOT MÓVIL', font_size=30)
        top_layout.add_widget(title_label)

        main_layout = GridLayout(cols=3, rows=3, spacing=10)

        main_layout.add_widget(Button(disabled=True))
        main_layout.add_widget(Button(text='Adelante', on_press=self.on_button_press_forward, on_release=self.on_button_release))
        main_layout.add_widget(Button(disabled=True))

        main_layout.add_widget(Button(text='Izquierda', on_press=self.on_button_press_left))
        main_layout.add_widget(Button(text='Stop', on_press=self.on_button_press_stop))
        main_layout.add_widget(Button(text='Derecha', on_press=self.on_button_press_right))

        main_layout.add_widget(Button(disabled=True))
        main_layout.add_widget(Button(text='Atras', on_press=self.on_button_press_backward, on_release=self.on_button_release))
        main_layout.add_widget(Button(disabled=True))

        bottom_layout = BoxLayout(orientation='horizontal', spacing=1, size_hint_y=None, height=50)

        connect_button = Button(text='Conectar', size_hint=(None, None), size=(100, 50),
                                pos_hint={'right': 2, 'bottom': 0}, on_press=self.connect_robot)
        bottom_layout.add_widget(connect_button)

        exit_button = Button(text='Salir', size_hint=(None, None), size=(50, 50),
                             pos_hint={'right': 1, 'bottom': 0}, on_press=self.exit_app)
        bottom_layout.add_widget(Button(disabled=True))
        bottom_layout.add_widget(exit_button)

class MyApp(App):

    def start_micro_ros_agent(self):
        try:
            subprocess.Popen(["ros2", "run", "micro_ros_agent", "micro_ros_agent", "udp4", "--port", "8888"])
            print("micro_ros_agent iniciado correctamente.")
        except Exception as e:
            print(f"Error al iniciar micro_ros_agent: {e}")

    def on_button_press_forward(self, instance):
        print('Botón Adelante presionado')

        # Configura la trama Twist
        self.twist_msg.linear.x = 0.2
        self.twist_msg.linear.y = 0.0
        self.twist_msg.linear.z = 0.0
        self.twist_msg.angular.x = 0.0
        self.twist_msg.angular.y = 0.0
        self.twist_msg.angular.z = 0.0
        # Publica la trama en el tópico /microROS/control
        self.publisher.publish(self.twist_msg)

    def on_button_release(self, instance):
        #print('Botón liberado')
        self.publish_twist()

    def on_button_press_backward(self, instance):
        print('Botón Atras presionado')
        self.twist_msg.linear.x = -0.2
        self.twist_msg.linear.y = 0.0
        self.twist_msg.linear.z = 0.0
        self.twist_msg.angular.x = 0.0
        self.twist_msg.angular.y = 0.0
        self.twist_msg.angular.z = 0.0
        self.publisher.publish(self.twist_msg)

    def on_button_press_left(self, instance):
```

Fig. 10: Código de los elementos y acción de cada uno.

4.3.4. Fase 4 Revisión y Retrospectiva:

4.3.4.1. Revisión: Durante la fase de revisión del Sprint, se llevó a cabo una reunión de evaluación con el tutor de la tesis para presentar y analizar el incremento del producto desarrollado. En este encuentro, se destacaron los avances significativos relacionados con la aplicación basada en ROS2 para el control de robots con motores inteligentes, diseñada para el Avance de la Investigación.

Se realizó una demostración integral del controlador implementado con ROS2. El enfoque principal fue resaltar las siguientes características clave del producto:

Comunicación mediante Twist: Se presentó el sistema de comunicación basado en el intercambio de mensajes Twist. Este método permitió la transmisión eficiente de datos entre la aplicación y el robot, facilitando el control preciso de los motores.

Integración con Motores Inteligentes: Se detalló la integración exitosa de los motores inteligentes con la aplicación ROS2. Estos motores están conectados a una unidad Open CM para garantizar un funcionamiento cohesivo y eficiente.

Funcionalidades Principales: Se destacaron las funcionalidades principales de la aplicación “salir/cerrar”, el control de movimientos “adelante”, “atrás”, “izquierda”, “derecha” y “stop/parar”, la capacidad de respuesta a comandos y la coordinación fluida con los motores para lograr un comportamiento controlado del robot.

4.3.4.2. Retrospectiva del Sprint: La fase de retrospectiva del Sprint constituyó un espacio fundamental para reflexionar sobre el desempeño del equipo de desarrollo durante la implementación de la aplicación basada en ROS2 para el control de robots con motores inteligentes, en el marco de la tesis vinculada a la Fundación para el Avance de la Investigación en Proyectos (FAINPRO).

4.3.4.3. Revisión de Procesos: Durante la revisión de procesos, se llevó a cabo un análisis exhaustivo para identificar elementos que contribuyeron al éxito del Sprint y aquellos que presentaron desafíos. Entre los aspectos destacados se encuentran

4.3.4.4. Logros y Aspectos Positivos: el logro alcanzado fue el desarrollo de la aplicación para controlar al robot de igual manera el software controlador que es cargada en la tarjeta esp32 para poder tener el contacto con la aplicación y el robot. Los aspectos positivos de este proyecto son, la comunicación eficiente sin demora hacia el robot no existe perdida de información al momento de ser ejecutada.

4.3.4.5. Desafíos y Problemas: se discutió sobre los problemas encontrados en el desarrollo y en la comunicación de la interfaz con el software controlador del robot, y el desafío que se tenía en el proyecto para un desarrollo totalmente funcional y que sea implementada correctamente.

4.3.5. Fase 5 Entrega del producto

La entrega del producto se realiza en el laboratorio de la Universidad Tecnológica Indoamérica al laboratorista quien está a cargo con todo su funcionamiento estable y seguro. También la parte técnica del software y hardware es entregada al ingeniero José Varela quien es docente de robótica y también tutor de este proyecto realizado. Tomando en cuenta todos los aspectos a continuación:

- Finalización de la codificación e implementación de todas las funcionalidades de la aplicación móvil.
- Completar la documentación técnica y académica asociada a la tesis.
- Realizar revisiones y mejoras basadas en los aportes del Ing. José Varela.
- Asegurar la alineación de la aplicación y la tesis con los estándares académicos y profesionales.
- Realizar una presentación de la aplicación y la tesis para validación y retroalimentación.

4.4. RESULTADOS

En la Tabla 6, se presenta la ficha de requerimientos de funcionalidad de la aplicación, detallando los intentos realizados y los logros obtenidos para cada requisito. Este registro proporciona una visión detallada de la implementación y el éxito alcanzado en cada aspecto funcional del sistema.

Tabla VI

Ficha de Requerimiento funcional.

ID	Requerimientos	Intentos	Logrados	Cumplimiento
RF 1	Conectar	40	38	Hecho
RF 2	Velocidad lineal	40	36	Hecho
RF 3	Velocidad angular	40	36	Hecho
RF 4	Stop	40	40	Hecho
RF 5	Salir	40	40	Hecho

4.4.1. Encuestas mediante un formulario de Google

De acuerdo con el cuestionario realizado en la tabla 4 basado en el modelo TAM (Modelo de Aceptación Tecnológica), se realizó la encuesta al ingeniero David laboratorista encargado de la Facultad de Ingeniería, Industria y Producción. La muestra final está representada por una persona lo cual representa una validación de eficiencia sobre el proyecto realizado.

En la Tabla 7, se detallan los niveles de calificación asociados al cuestionario de evaluación de aceptación utilizado en el contexto de este proyecto. El cuestionario, compuesto por 7 niveles, abarca un rango que va desde "1" para indicar una posición totalmente en desacuerdo hasta "7" que representa una posición totalmente de acuerdo. Esta escala de calificación estratificada permite capturar de manera matizada las respuestas de los participantes, ofreciendo una amplia gama de opciones para expresar su grado de acuerdo o desacuerdo con las afirmaciones propuestas. Esta estructuración precisa proporciona un marco claro para interpretar las respuestas recopiladas y facilita la obtención de datos cuantitativos significativos sobre la percepción y aceptación del proyecto por parte del participante.

Tabla VII

Niveles de calificación para evaluar la aceptación de tecnología.

1	2	3	4	5	6	7
Totalmente en Desacuerdo	Muy en Desacuerdo	En Desacuerdo	Indiferente	De Acuerdo	Muy De Acuerdo	Totalmente de Acuerdo

Conforme se aprecia en la Figura 11, los resultados de la encuesta revelan que el encuestado expresó una aceptación significativa, otorgando una calificación mayoritaria de 7, en relación con la utilidad del innovador software de control destinado a robots con motores inteligentes. Esta alta puntuación indica que el encuestado percibe que el software satisface eficazmente las necesidades y expectativas, situándose dentro del rango de utilidad requerido. Esta valoración positiva subraya la eficacia del nuevo software y destaca la alineación exitosa de sus características con las expectativas del usuario. La información proporcionada por la Figura 8 sugiere una sólida aceptación de la utilidad del software, respaldando así la efectividad de su diseño y funcionalidad para

satisfacer las demandas específicas en el ámbito de robots con motores inteligentes. Estos resultados constituyen una base valiosa para la validación y mejora continua de la tecnología propuesta.

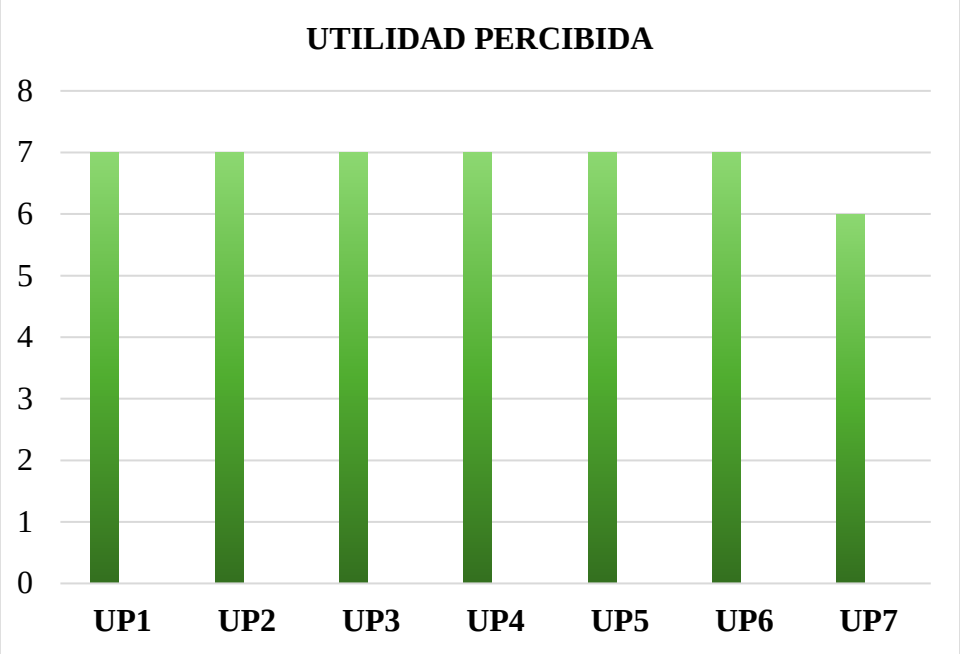


Fig. 11: Utilidad percibida

En la Figura 12, se evidencia que el encuestado ha asignado una calificación de 7 a la mayoría de las preguntas, lo cual constituye un indicativo positivo en relación con la aceptación de la factibilidad de uso del software desarrollado en conjunto con la aplicación de control para el robot manipulador. Estas calificaciones consistentemente altas refuerzan la percepción del encuestado sobre la viabilidad y facilidad de utilizar la herramienta propuesta. Esta evaluación positiva sugiere que el software y su interfaz de control han logrado satisfacer las expectativas del usuario en términos de accesibilidad y eficacia. La Figura 9, al resaltar estas respuestas favorables, proporciona valiosa retroalimentación que respalda la eficacia práctica y la aceptación del software y la aplicación de control en el entorno específico del robot manipulador. Estos hallazgos respaldan la validez de la solución tecnológica propuesta y ofrecen perspectivas clave para posibles refinamientos y mejoras futuras.

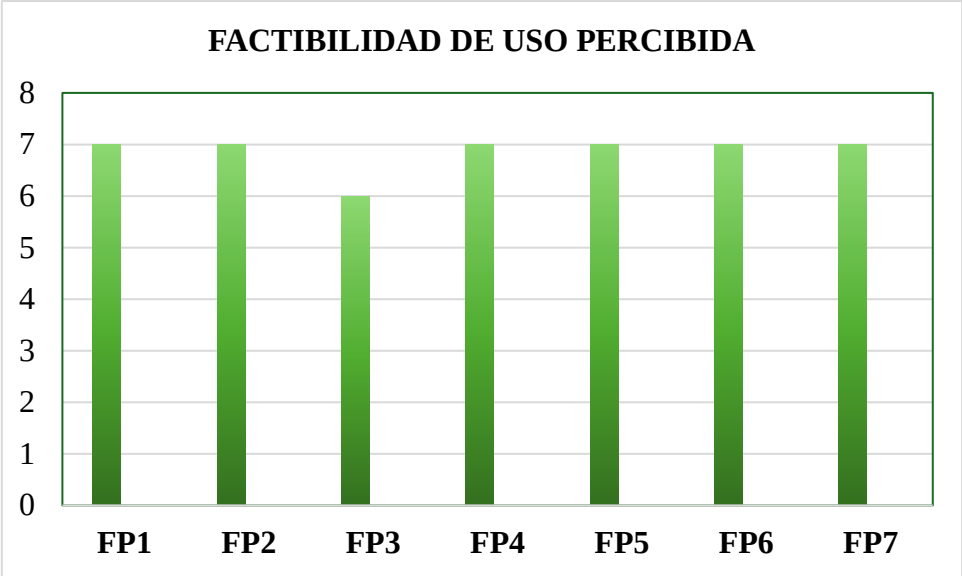


Fig. 12: Factibilidad de uso percibida.

En la Tabla 8, se presenta el promedio de la encuesta aplicada al laboratorista. De acuerdo con este promedio, se calcula un índice de aceptación del 98,0% en la utilidad percibida. Por otra parte, tenemos a la factibilidad de uso percibida con un porcentaje del 98,0% de aceptación. Este resultado altamente positivo indica que el rendimiento del software de control cumple con todas las metas establecidas en los objetivos y es fácil de usar y de entender.

Tabla VIII

Aceptación de la factibilidad de uso y utilidad percibida.

Utilidad Percibida		Factibilidad de Uso Percibida	
UP1	**7**	**FP1**	**7**
UP2	**7**	**FP2**	**7**
UP3	**7**	**FP3**	**6**
UP4	**7**	**FP4**	**7**
UP5	**7**	**FP5**	**7**
UP6	**7**	**FP6**	**7**
UP7	**6**	**FP7**	**7**
Promedio	**6,9**	**Promedio**	**6,9**
Aceptación en %	**98,0**	**Aceptación en %**	**98,0**

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

Se desarrollo un software y una aplicación de control para controlar robots con mototres inteligentes basados en ROS2.

- Se logró diseñar una estructura de software sólida y eficiente basada en ROS2 para la interfaz de robots con motores inteligentes. La arquitectura proporciona una base estable que facilita la comunicación y el control remoto, asegurando una integración efectiva de los motores inteligentes. El software controlador fue desarrollada en el lenguaje de programacion C utilizando la herramienta de Visual Studio Code. Así mismo se desarrollo de la interfaz (aplicación) fue desarrollada en el lenguaje de programación python. Estos dos elementos son esenciales para el funcionamiento y fue desarrollado dentro del subsistema para linux con el sistyema operativo Ubuntu 22.04 LTS.

- La aplicación se desarrolló utilizando la librería Kivy, que posibilita la creación de interfaces GUI multiplataforma, siendo compatible con Windows, macOS, Linux, Android e iOS. Para el software, se emplearon diversas librerías, como la destinada al manejo de errores en ROS 2, facilitando la gestión y reporte eficiente de errores. Para esto se realizó la programación de clientes ROS2 en entornos con restricciones de recursos, siendo ideal para sistemas embebidos. También se utilizó una biblioteca estándar en sistemas Unix para manipulación de descriptores de archivo, y otra para definir la estructura del mensaje "Twist" en ROS2. Además, se emplearon librerías estándar en C para operaciones de entrada/salida y manipulación de cadenas de caracteres, junto con FreeRTOS para sistemas operativos en tiempo real, posibilitando la ejecución determinista de tareas concurrentes en microcontroladores embebidos.

- El robot fue evaluado mediante las pruebas de funcionamiento, utilizando los elementos del test de aceptación como referencia. Esta herramienta fue

fundamental para llevar a cabo una evaluación efectiva y contribuir al proceso de aceptación.

- En la prueba de aceptación utilizando el método TAM, se alcanzó un promedio de 6.9/7, equivalente al 98.0% en términos porcentuales. Este resultado refleja un elevado nivel de aceptación, indicando plena funcionalidad del software y la aplicación para el control del robot manipulador móvil, Este rendimiento sobresaliente respalda la eficacia y confiabilidad del software, validando su capacidad para controlar de manera óptima el robot manipulador móvil. La obtención de un alto porcentaje de aceptación subraya su capacidad para cumplir con los requerimientos y expectativas, consolidando su posición como una solución funcional y robusta.

5.2. RECOMENDACIONES

Se sugiere la integración del brazo robótico tanto en la aplicación como en el software de control, con el objetivo de optimizar la funcionalidad del robot. Esta incorporación contribuirá significativamente a mejorar la integralidad del funcionamiento, permitiendo una operación más completa y eficiente del sistema robótico.

Se recomienda adoptar un enfoque diferente en el código para lograr una mayor intuición y una calidad superior, garantizando así un funcionamiento óptimo. Esta alternativa busca mejorar tanto el rendimiento como la calidad, elevando el estándar general de la aplicación para lograr un desempeño excepcional y una experiencia de usuario mejorada.

Se sugiere perfeccionar el proceso de prueba de aceptación de la aplicación con el fin de lograr una demostración de resultados más efectiva. Esta mejora contribuirá a una evaluación más precisa y detallada, garantizando una presentación más robusta y convincente de los logros obtenidos.

Se sugiere emplear el sistema operativo en una maquina física con el fin de garantizar un óptimo rendimiento tanto en el desarrollo de funciones futuras como en la ejecución de la interfaz con el robot manipulador, en el cual se ha creado la aplicación y codificado el controlador.

Se recomienda llevar a cabo múltiples pruebas del robot utilizando la aplicación desarrollada. Es crucial revisar cuidadosamente cada movimiento antes de su ejecución, así como verificar los tiempos de respuesta entre los componentes para garantizar su funcionalidad de manera constante.

BIBLIOGRAFIA

[1] C. Heggem, N. M. Wahl, and L. Tingelstad, “Configuration and Control of KMR iiwa Mobile Robots using ROS2,” in *2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, IEEE, jun. 2020, pp. 1–6. doi: 10.1109/SIMS49386.2020.9121554.

[2] F. P. Audonnet, A. Hamilton, and G. Aragon-Camarasa, “A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2,” in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, IEEE, nov. 2022, pp. 755–762. doi: 10.23919/ICCAS55662.2022.10003832.

[3] L. Arcos, K. Vicente, P. Cruz, J. Abad, and I. Zambrano, “A ROS2 based Trajectory Tracking Controller of a 3UPS-1RPU Parallel Robot for Knee Rehabilitation,” in *2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM)*, IEEE, oct. 2022, pp. 1–6. doi: 10.1109/ETCM56276.2022.9935755.

[4] Yumbla Arevalo, Francisco Xavier, irector Piguave Toala, Anthonny Ernesto, Ronquillo Manosalvas, and Diego Sebastian, “Diseño e implementación de un sistema Multi-Robot de código abierto para ambientes colaborativos en ROS2,” ESPOL, Guayaquil, Ecuador, 2023. Accessed: Jan. 18, 2024. [Online]. Available: http://www.dspace.espol.edu.ec/handle/123456789/58439

[5] J. L. Varela Aldas and F. V. Junta Andagana, “Aplicación de la industria 4.0 en los procesos de enseñanza en la carrera de Ingeniería Industrial de la Universidad Tecnológica Indoamérica usando metodología STEAM.,” tesis de grado, Universidad Tecnológica Indoamérica, Ambato, 2022. Accessed: Jan. 24, 2024. [Online]. Available: https://repositorio.uti.edu.ec//handle/123456789/3885

[6] Junta Andagana Christian Joseph and M. Ing. José Luis Varela Aldas, “implementación de un robot educativo para el desarrollo de competencias steam en estudiantes de educación básica superior a través de una aplicación móvil,” Tesis de Grado, universidad tecnológica Indoamérica, Ambato, 2022.

[7] Matteo De Rose, “LiDAR-based Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2,” Master Thesis, POLITECNICO DI TORINO, 2021. Accessed: oct. 25, 2023. [Online]. Available: https://webthesis.biblio.polito.it/21253/1/tesi.pdf

[8] Geert Mol, “Developing a platform for the KUKA iDo social robot,” trabajo de fin de grado, University of Twente, 2023. Accessed: oct. 25, 2023. [Online]. Available: http://essay.utwente.nl/97166/1/Mol_BA_EEMCS.pdf

[9] A. Mamani-Saico and P. R. Yanyachi, “Implementation and Performance Study of the Micro-ROS/ROS2 Framework to Algorithm Design for Attitude Determination and Control System,” *IEEE Access*, vol. 11, pp. 128451–128460, 2023, doi: 10.1109/ACCESS.2023.3330441.

[10] Á. Vázquez Ramos, “control de un robot cartesiano usando el sistema operativo robótico ROS2,” Grado en Ingeniería en Electrónica Industrial y Automática, universidad de valladolid escuela de ingenierias industriales, Valladolid, 2020.

[11] M. Á. González Santamarta, “MERLIN2: Sistema cognitivo para ROS2,” Máster Universitario en Ingeniería Informática, Universidad de León, España, 2022.

[12] J. López-Belmonte, A. Segura-Robles, A.-J. Moreno-Guerrero, and M.-E. Parra-González, “Robotics in Education: A Scientific Mapping of the Literature in Web of Science,” *Electronics (Basel)*, vol. 10, no. 3, p. 291, Jan. 2021, doi: 10.3390/electronics10030291.

[13] J. G. Guarnizo Marin, D. Bautista Díaz, and J. S. Sierra Torres, *Una revisión sobre la evolución de la robótica móvil*. Universidad Santo Tomás, 2021. doi: 10.15332/dt.inv.2021.02848.

[14] Robótica, “Clasificación de los robots según su función,” esneca businessschool. Accessed: Jan. 18, 2024. [Online]. Available: https://www.esneca.com/blog/clasificacion-de-los-robots-segun-su-funcion/

[15] Revista de robots, “Robótica móvil. Qué es la robótica móvil y para qué sirve,” *mayo 24, 2023*, May 24, 2023.

[16] Intelligent motion, “Actuadores inteligentes.”

[17] Educación Robótica, “Educación Robótica,” *14/04/2023*, Feb. 18, 2023.

[18] M. Pohnl, A. Tamisier, and T. Blass, “A Middleware Journey from Microcontrollers to Microprocessors,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, Mar. 2022, pp. 282–286. doi: 10.23919/DATE54114.2022.9774609.

[19] Riva Juan Eduardo, “ROS introducción,” *2021-08-18,* Aug. 18, 2021.

[20] J. Varela-Aldás and G. Palacios-Navarro, “A ROS-Based Open Tool for Controlling an Educational Mobile Robot.,” pp. 23–39, 2024.

[21] Francisco Martín Rico, *A Concise Introduction to Robot Programming with* ROS2. 2023.

[22] Correa Ramírez, PERÉZ Jorge Alfaro, and Alegre Pablo Durand, “Acceptance and use of websites of government transparency: An empirical study in Chile,” 15/10/2015, Chile, oct. 15, 2015.

ANEXOS

ANEXO 1: Guía de control

Guía de control e instalación de los elementos del robot manipulador móvil.

El Manual del Control del Robot Manipulador Móvil ofrece una guía exhaustiva destinada a la eficiente operación y control de un robot manipulador móvil. Este documento ha sido elaborado para usuarios de diversos niveles, desde principiantes hasta usuarios avanzados, y cubre de manera detallada aspectos clave relacionados con la interfaz de control, la configuración y la ejecución de acciones específicas

Para conectar la aplicación al robot primero se tendrá que abrir la terminal del sistema operativo Ubuntu, para ello se tendrá que ejecutar los siguientes comandos: *source install/local_setup.sh* que sirve para cargar el script que establece las variables necesarias para los ejecutables.

El siguiente comando: *ros2 run micro_ros_setup configure_firmware.sh pin_pong -t udp -i 172.18.34.130 -p 8888*, sirve para configurar el firmware Micro-ROS para comunicación UDP en dirección específica.

Para personalizar las opciones de configuración del firmware Micro-ROS se ejecuta el siguiente comando: *ros2 run micro_ros_setup build_firmware.sh menuconfig*. Después de realizar la ejecución del comando anterior procedemos a compila el firmware de Micro-ROS con el siguiente comando: *ros2 run micro_ros_setup build_firmware.sh*. Sí nos da un error al final puede volver a ejecutar el comando de la configuración del firmware y continuar con el comando para compilar el firmware.

Como siguiente paso debemos cagar el firmware de Micro-ROS en el hardware correspondiente, en este caso a la tarjeta ESP32 para ejecutar el comando a continuación se debe verificar la conexión de la tarjeta hacia la computadora y ejecutamos el siguiente comando: *ros2 run micro_ros_setup flash_firmware.sh*.

Una vez realizado todos los pasos anteriores con efectividad si puede ejecutar el controlador desde un software que ayuda a ver las gráficas de un sistema operativo WSL en este caso se utilizó el software MobaXterm en donde con el comando *pyhon3 nombre_proyecto.py* se ejecutara la

aplicación, una vez abierto la interfaz se puede ejecutar con el botón conectar para la conexión de la app con el robot y así manipular desde la aplicación sin ningún problema.

En la figura 13, se presenta la interfaz gráfica de la aplicación, donde se destaca el nombre de la aplicación y se incluye una representación visual del robot manipulador móvil. Esta interfaz está diseñada para el control del robot y consta de siete botones en total. Cinco de estos botones están destinados a dirigir los movimientos del robot, como "adelante", "atrás", "izquierda", "stop" y "derecha". Además, se han incorporado dos botones adicionales: uno para cerrar la aplicación y otro para establecer la conexión. Estos elementos proporcionan una interfaz intuitiva y funcional, permitiendo a los usuarios controlar el robot de manera eficiente y gestionar la aplicación de manera fácil.

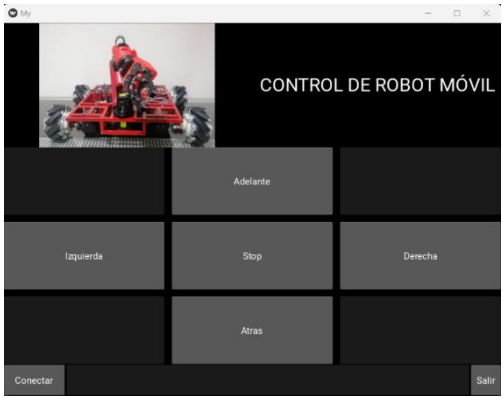


Fig. 13: Interfaz de control del robot manipulador.

En la figura 14, se destaca el botón "Conectar", el cual facilita la conexión con la ESP32 para el envío de datos a la tarjeta OpenCM tener en cuenta que solo se puede ejecutar mientras la terminal de Ubuntu este abierta. En caso de que no se envíen datos, simplemente presionar nuevamente el botón "Conectar" realizará un reinicio de la conexión, permitiendo así una reconexión efectiva. Este proceso asegura la optimización del uso de la interfaz, proporcionando una solución sencilla y eficiente para mantener la comunicación entre la aplicación y el robot manipulador móvil.

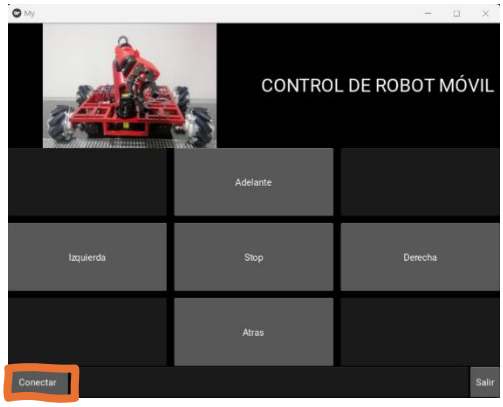


Fig. 14: Función del botón conectar.

En la figura 15, se destaca el botón superior que corresponde a la función "adelante". La funcionalidad de este botón, codificada en Python, consiste en enviar un valor positivo desde la tarjeta ESP32 a la tarjeta OpenCM. Este valor activará los motores y desencadenará el movimiento hacia adelante del robot. La implementación de esta acción mediante

el código Python garantiza una transmisión eficaz de comandos a la tarjeta OpenCM, permitiendo el control preciso y activación de los motores para el desplazamiento hacia adelante del robot manipulador móvil.

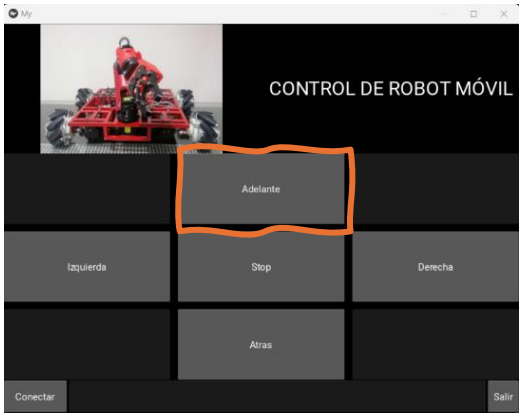


Fig. 15: Función del botón adelante.

En la figura 16, se observa el botón inferior que corresponde a la función "atrás". Para proporcionar una funcionalidad similar al botón "adelante", se debe implementar una función en el código Python que maneje esta acción específica. El cual activará a los motores y empezará a moverse el robot manipulador omnidireccional.

Fig. 16: Función del botón atrás.

En la figura 17 se observa el botón lateral izquierdo que corresponde a la función mover a la izquierda tiene una funcionalidad similar al botón los botones anteriores cambiando valores en la codificación que se ha realizado en Python, y así activando a los motores al enviar el dato y poniendo en marcha al robot.

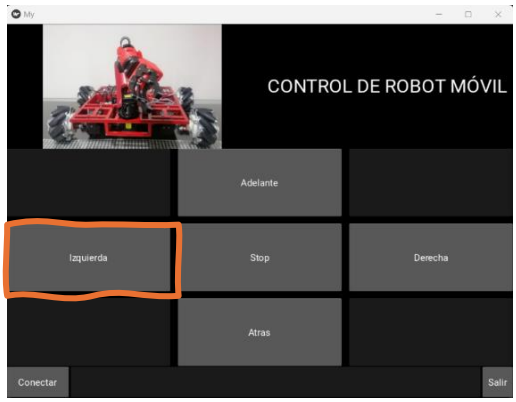


Fig. 17: Función del botón izquierda.

En la figura 18 se observa el botón lateral derecho correspondiente a la actividad mover hacia la derecha. Esta función es similar a todos los botones vistos anteriormente con la diferencia de

cambio de valores. Estos cambios se deben realizar en el código realizada en Python y así enviar el dato y activar los motores y ver la acción del robot.

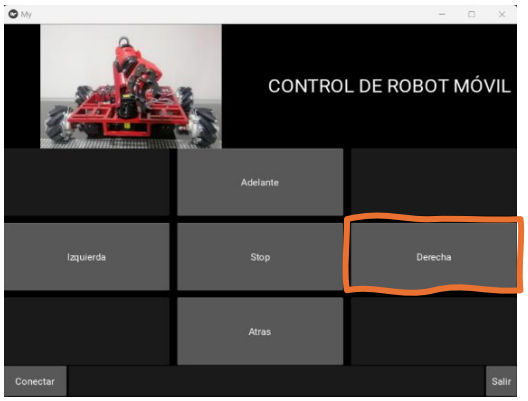


Fig. 18: Función del botón derecha.

En la figura 19 podemos observar el botón intermedio stop con la función detener al robot, esta función desactiva todos los valores enviados por los botones anteriores y dejar al robot parado sin ninguna acción.

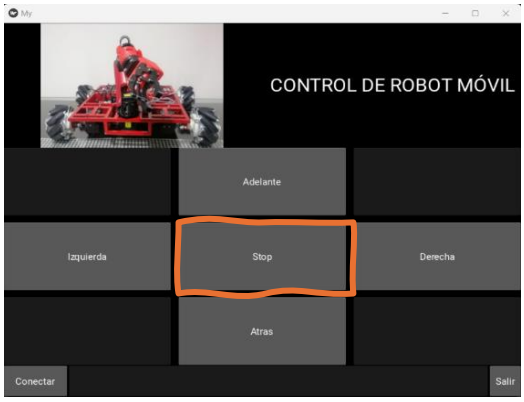


Fig. 19: Función del botón stop/parar.

En la figura 20 se observa el botón inferior que corresponde a la función "salir", esta función hace que se desconecte la aplicación del robot y se cierra automáticamente la interfaz y así quedaría el robot sin conexión hasta volver a conectar con la interfaz.

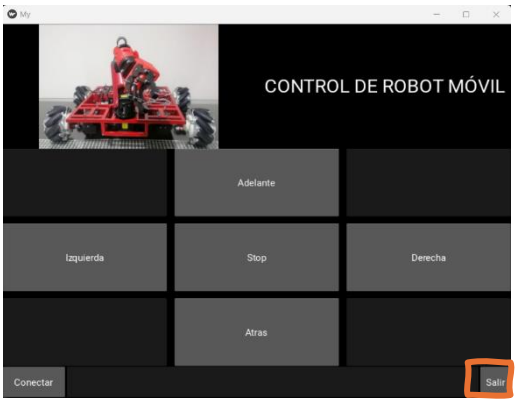


Fig. 20: Función del botón salir.

ANEXO 2: Instalación de Ubuntu 22.04

WSL (Windows Subsystem for Linux).

En la figura 21 se puede observar el instalador de Ubuntu 22.04 LTS que se encuentra dentro de la tienda de Microsoft Store.

Fig. 21: Sistema Operativo Ubuntu 22.04 LTS (WLS).

En la figura 22 se puede observar Ubuntu corriendo ya configurado con el usuario y contraseña listo para ser utilizado.

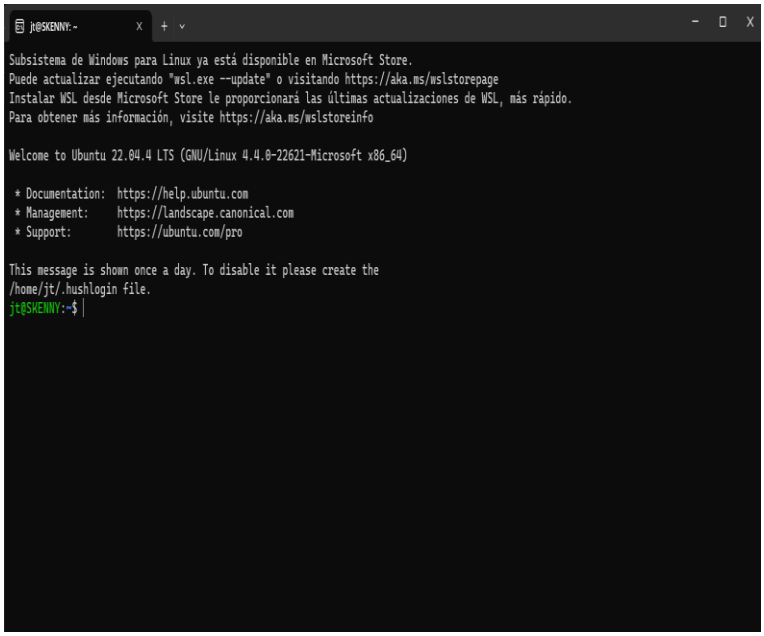


Fig. 22: Ubuntu 22.04 Instalado y Configurado con el respectivo usuario.

ANEXO 3: Instalación de ROS2

Humble Hawksbill.

A continuación, se puede observar estos comandos están relacionados con la configuración de la configuración regional y el conjunto de caracteres en un sistema basado en Linux, específicamente para asegurarse de que el sistema esté configurado para utilizar UTF-8, que es un estándar de codificación de caracteres que admite una amplia gama de caracteres y es compatible con muchos idiomas y sistemas de escritura.

locale

sudo apt update && apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale

Después de eso, ejecutamos los siguientes comandos que se puede observar, el primer comando se refiere a la instalación de un conjunto de utilidades comunes para gestionar repositorios de software en sistemas Ubuntu y basados en Debian. El segundo comando muestra información sobre las políticas de los paquetes disponibles en los repositorios configurados en el sistema, filtrando la salida para mostrar únicamente información relacionada con el repositorio "universe", que contiene paquetes mantenidos por la comunidad en sistemas Ubuntu.

sudo apt install software-propieties-common
sudo add-apt-repostery universe.

A continuación, se observa el comando sudo apt update && sudo apt install curl -y`: Actualiza la lista de paquetes disponibles y luego instala la herramienta "curl" de manera automática, sin solicitar confirmación. El segundo comando, sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key  o/usr/share/keyrings/ros-archive-keyring.gpg`: Descarga el archivo de clave de autenticación para el repositorio ROS y lo guarda en el directorio `/usr/share/keyrings/`, permitiendo la verificación de la autenticidad de los paquetes ROS durante la instalación.

```
sudo apt update && sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
```

Ahora, se observa el comando, “sudo apt update”: Actualiza la lista de paquetes disponibles en los repositorios configurados en el sistema. Este comando permite al sistema operativo conocer las versiones más recientes de los paquetes disponibles para su instalación. En el segundo comando, “sudo apt upgrade”: Descarga e instala las últimas versiones de los paquetes instalados en el sistema, asegurando que todas las aplicaciones y componentes estén actualizados a las versiones más recientes disponibles en los repositorios. Esto ayuda a garantizar la seguridad, estabilidad y rendimiento del sistema al mantenerlo actualizado con las últimas correcciones de errores, mejoras y parches de seguridad.

```
sudo update
sudo upgrade
```

Por último, se observa el comando "sudo apt install ros-humble-desktop" se utiliza para instalar el conjunto de paquetes ROS (Robot Operating System) conocido como "ros-humble-desktop". Este paquete proporciona una instalación básica del entorno de desarrollo de ROS en el escritorio, incluyendo herramientas y bibliotecas esenciales para comenzar a trabajar con ROS en un entorno de desarrollo. Estos paquetes suelen incluir herramientas de línea de comandos, bibliotecas para la comunicación entre nodos ROS, utilidades de simulación y visualización, entre otros.

```
sudo apt install ros-humble-descktop
```

ANEXO 4: Instalación de microROS2

MicroROS2 para esp32

En la figura 28 se observa el inicio de la instalación de Micro-ROS y creación de una carpeta en donde se almacenará cada uno de los ejemplos por defecto y los que se vayan creando.

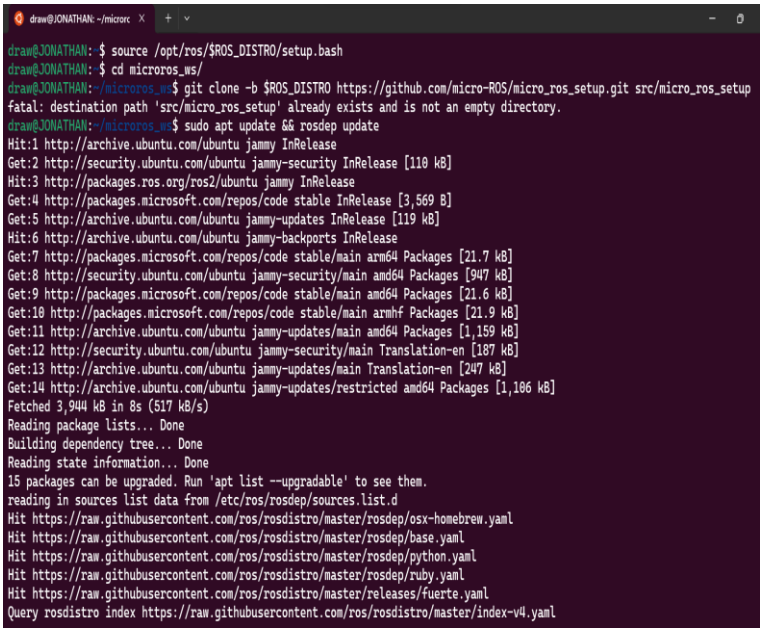


Fig. 23: Fuente de la instalación de ROS2.

En la figura 29 se puede observar los comandos para actualizar dependencias utilizando rosdep implica gestionar las bibliotecas y paquetes necesarios para un proyecto en ROS (Robot Operating System). La instalación del pip se refiere a la instalación del gestor de paquetes de Python, permitiendo la adquisición sencilla de paquetes y bibliotecas adicionales. Por último, crear herramientas micro-ROS implica la generación y obtención de herramientas específicas de micro-ROS, un marco diseñado para sistemas embebidos, facilitando así su implementación en entornos con recursos limitados.

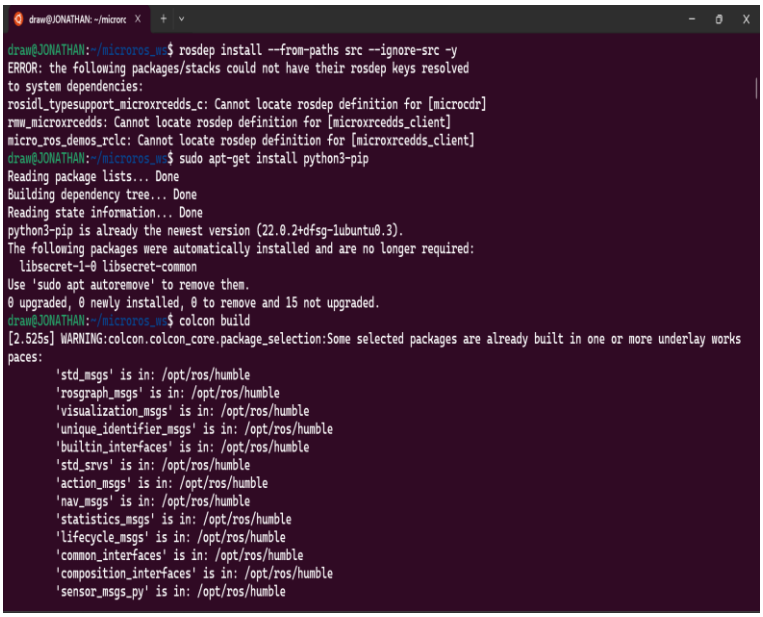


Fig. 24: Actualizar dependencias

Actualizar dependencias usando rosdep, Instalación del pip y Cree herramientas micro-ROS y consígalas.

En la figura 30 se puede observar el primer comando, es utilizado para establecer un entorno de desarrollo específico para la creación de firmware en el contexto de Micro-ROS en sistemas embebidos. El segundo comando se utiliza para construir el firmware necesario para habilitar la funcionalidad de Micro-ROS en sistemas embebidos específicos. La ejecución de este comando generalmente sigue la configuración previa del entorno de desarrollo utilizando otros scripts proporcionados por el paquete micro_ros_setup.

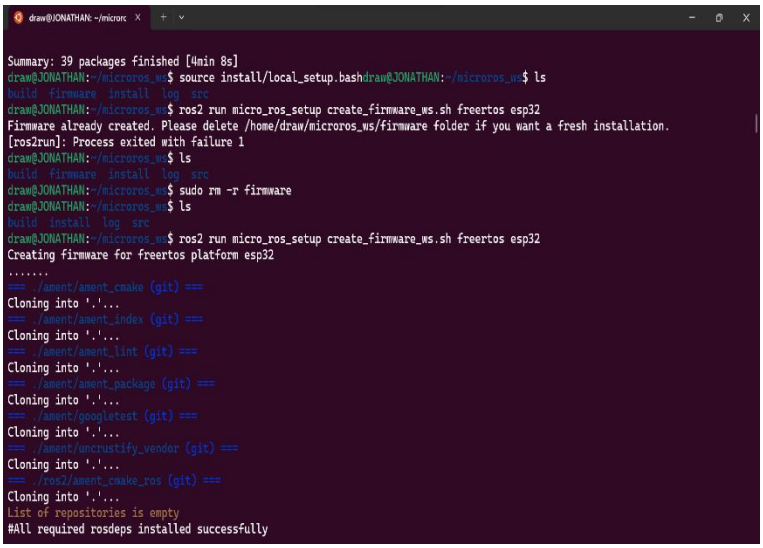


Fig. 25: Comandos para crear el firmware y crear el setup de micro-ROS.

En la figura 31 se puede observar el comando que ejecuta y crea el ejercicio ping_pong ejemplo interno que viene por default dentro de micro-ROS para poder saber si está funcionando o no la instalación realizada.

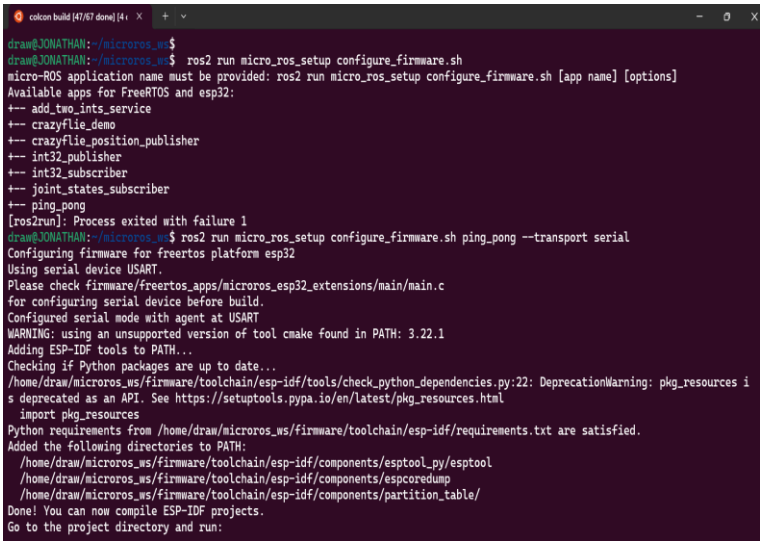


Fig. 26: Comando de ejecución de Micro-ROS.

Comando para ejecutar el ejercicio de ejemplo que viene dentro de Micro-ROS

En la figura 32 se puede observar la verificación de la carpeta donde será almacenado cada proyecto creado estará directamente dentro de la carpeta apps de Micro-ROS, así será más seguro poder ejecutar con más facilidad y sin problemas a la hora de buscar la ubicación de los proyectos.

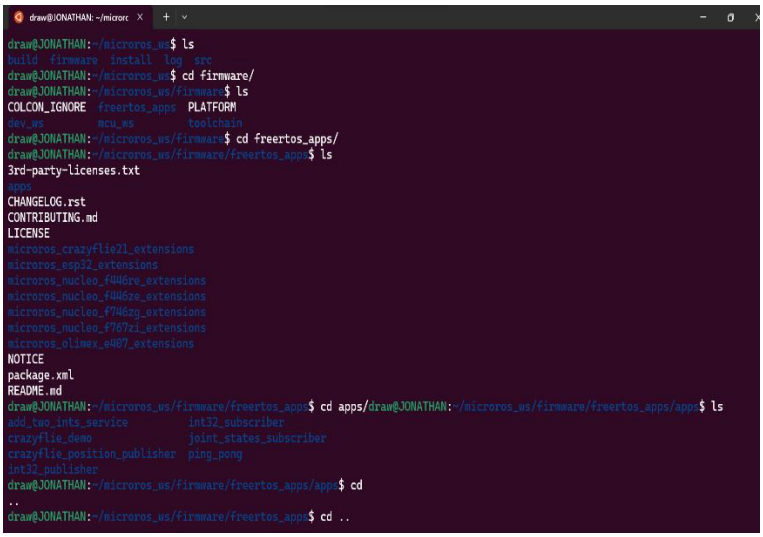


Fig. 27: Verificación de la carpeta.

Verificación de la carpeta donde se guardarán todos los proyectos que se vayan creando.

En la Figura 33, se visualiza el comando para cargar el software desarrollado en la ESP32. Este paso sigue a la creación de todos los paquetes esenciales para garantizar una ejecución exitosa. Este proceso es crucial para desplegar eficientemente el software en el dispositivo.

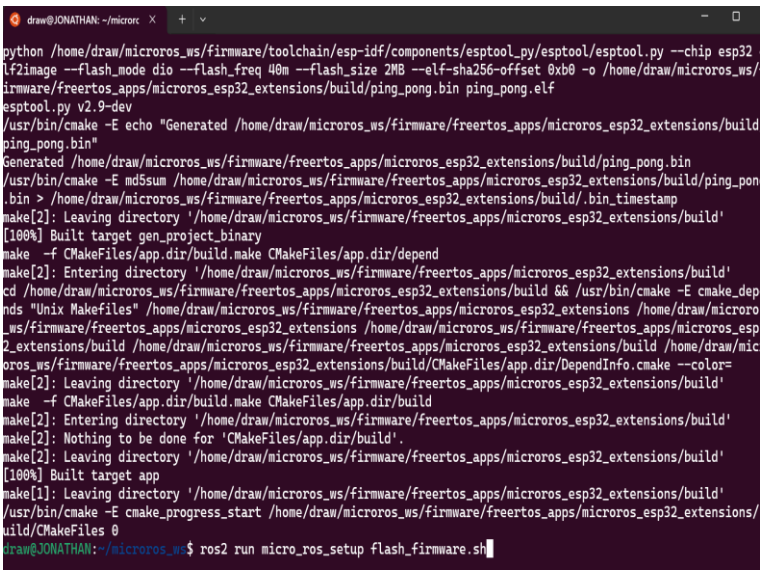


Fig. 28: Comando para cargar el software creado a la tarjeta ESP32.

ANEXO 5: Código del controlador

Código del controlador en ROS2.

```
#include <rcl/error_handling.h>

#include <rclc/rclc.h>

#include <rclc/executor.h>

#include <unistd.h>

#include <geometry_msgs/msg/twist.h>

#include <stdio.h>

#include <string.h>

#include <freertos/FreeRTOS.h>

#include <freertos/task.h>

#define RCCHECK(fn) { rcl_ret_t temp_rc = fn; if((temp_rc != RCL_RET_OK)){printf("Failed status on line %d: %d. Aborting.\n",__LINE__,(int)temp_rc);vTaskDelete(NULL);}}

//#define RCSOFTCHECK(fn) { rcl_ret_t temp_rc = fn; if((temp_rc != RCL_RET_OK)){printf("Failed status on line %d: %d. Continuing.\n",__LINE__,(int)temp_rc);}}

rcl_subscription_t subscriber2;

geometry_msgs__msg__Twist control_msg;

void control_callback(const void *msgin2)

{

   const geometry_msgs__msg__Twist *received_msg = (const geometry_msgs__msg__Twist *)msgin2;

   float U = received_msg->linear.x;

   float u2 = received_msg->linear.y;

   float w = received_msg->angular.z;

   int U1 = (int)(U*100);

   int U2 = (int)(u2*100);

   int W = (int)(w*100);

   //signo U1

   char signoU1; if (U1 < 0) { signoU1 = 'b'; } else {signoU1 = 'a'; }

   //signo U2

   char signoU2; if (U2 < 0) { signoU2 = 'b'; } else { signoU2 = 'a'; }

   //signo W
```

```
    char signoW; if (W < 0) { signoW = 'b'; } else { signoW = 'a'; }
    char control[38] = "x000x000x000a000a000a000a000a000a000a1";
    control[24] = signoU1; control[28] = signoU2; control[32] = signoW;
    int tempU1 = U1 < 0 ? -U1 : U1;
    if (U1 < 0 || U1 > 0) {
       for (int i = 27; i >= 25; --i) {
          control[i] = (char)((tempU1 % 10)+ '0'); // Obtén el dígito menos significativo
          tempU1 /= 10; // Divide por 10 para obtener el siguiente dígito
          if (tempU1 == 0) break; // Si no hay más dígitos, termina el bucle
       } }
    int tempU2 = U2 < 0 ? -U2 : U2;
    if (U2 < 0 || U2 > 0) {
       for (int i = 31; i >= 29; --i) {
          control[i] = (char)((tempU2 % 10)+ '0');
          tempU2 /= 10;
          if (tempU2 == 0) break;
       } }
    int tempW = W < 0 ? -W : W;
    if (W < 0 || W > 0) {
       for (int i = 35; i >= 33; --i) {
          control[i] = (char)((tempW % 10)+ '0');
          tempW /= 10;
          if (tempW == 0) break;
       }  }
     printf("%s\n",control);
}
void appMain(void *arg)
{
    rcl_allocator_t allocator = rcl_get_default_allocator();
    rclc_support_t support2;
    // create init_options
    RCCHECK(rclc_support_init(&support2, 0, NULL, &allocator));
```

```
   // create node
   rcl_node_t node2;
   RCCHECK(rclc_node_init_default(&node2, "twist_subscriber_rclc", "",
&support2));
   // create subscriber
   RCCHECK(rclc_subscription_init_default(
      &subscriber2,
      &node2,
      ROSIDL_GET_MSG_TYPE_SUPPORT(geometry_msgs, msg, Twist),
      "/microROS/control"));
   // create executor
   rclc_executor_t executor2;
   RCCHECK(rclc_executor_init(&executor2, &support2.context, 1, &allocator));
   RCCHECK(rclc_executor_add_subscription(&executor2, &subscriber2,
&control_msg, &control_callback, ON_NEW_DATA));
   while (1)
   {
       rclc_executor_spin_some(&executor2, RCL_MS_TO_NS(10));
      //usleep(100000);
   }
   // free resources
   RCCHECK(rcl_subscription_fini(&subscriber2, &node2));
   RCCHECK(rcl_node_fini(&node2));
   RCCHECK(rclc_executor_fini(&executor2)); // Liberar recursos del executor
   vTaskDelete(NULL); }
```

ANEXO 6: Código app Python

Código de la creación de la app en Python y configuración de la aplicación en Python y ROS2.

```
import subprocess
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button
from kivy.uix.image import Image
from kivy.uix.label import Label
import rclpy
from geometry_msgs.msg import Twist
class MyApp(App):
    def __init__(self, **kwargs):
        super(MyApp, self).__init__(**kwargs)
        self.publisher = None
        self.twist_msg = Twist()
    def build(self):
        rclpy.init()
        self.node = rclpy.create_node('kivy_ros_node')
        self.publisher = self.node.create_publisher(Twist, '/microROS/control', 10)
        top_layout = BoxLayout(orientation='horizontal', spacing=5, size_hint_y=None, height=200)
        try:
            image = Image(source='/home/draw/interfaz/image/Omni.png', size=(600, 300))
        except Exception as e:
            print(f"Error al cargar la imagen: {e}")
        top_layout.add_widget(image)
        title_label = Label(text='CONTROL DE ROBOT MÓVIL', font_size=30)
        top_layout.add_widget(title_label)
        main_layout = GridLayout(cols=3, rows=3, spacing=10)
        main_layout.add_widget(Button(disabled=True))
```

```
        main_layout.add_widget(Button(text='Adelante',
on_press=self.on_button_press_forward, on_release=self.on_button_release))
        main_layout.add_widget(Button(disabled=True))
        main_layout.add_widget(Button(text='Izquierda',
on_press=self.on_button_press_left))
        main_layout.add_widget(Button(text='Stop', on_press=self.on_button_press_stop))
        main_layout.add_widget(Button(text='Derecha',
on_press=self.on_button_press_right))
        main_layout.add_widget(Button(disabled=True))
        main_layout.add_widget(Button(text='Atras',
on_press=self.on_button_press_backward, on_release=self.on_button_release))
        main_layout.add_widget(Button(disabled=True))
        bottom_layout = BoxLayout(orientation='horizontal', spacing=1,
size_hint_y=None, height=50)
        connect_button = Button(text='Conectar', size_hint=(None, None), size=(100, 50),
                                pos_hint={'right': 2, 'bottom': 0}, on_press=self.connect_robot)
        bottom_layout.add_widget(connect_button)
        exit_button = Button(text='Salir', size_hint=(None, None), size=(50, 50),
                             pos_hint={'right': 1, 'bottom': 0}, on_press=self.exit_app)
        bottom_layout.add_widget(Button(disabled=True))
        bottom_layout.add_widget(exit_button)
        main_container = BoxLayout(orientation='vertical')
        main_container.add_widget(top_layout)
        main_container.add_widget(main_layout)
        main_container.add_widget(bottom_layout)
        return main_container
    def connect_robot(self, instance):
        print('Conectando al robot')
        self.start_micro_ros_agent()
    def start_micro_ros_agent(self): try:
            subprocess.Popen(["ros2", "run", "micro_ros_agent", "micro_ros_agent",
"udp4", "--port", "8888"])
            print("micro_ros_agent iniciado correctamente.")
        except Exception as e:
```

```
        print(f"Error al iniciar micro_ros_agent: {e}")
def on_button_press_forward(self, instance):
    print('Botón Adelante presionado')
    # Configura la trama Twist
    self.twist_msg.linear.x = 0.2
    self.twist_msg.linear.y = 0.0
    self.twist_msg.linear.z = 0.0
    self.twist_msg.angular.x = 0.0
    self.twist_msg.angular.y = 0.0
    self.twist_msg.angular.z = 0.0
    # Publica la trama en el tópico /microROS/control
    self.publisher.publish(self.twist_msg)
def on_button_release(self, instance):
    #print('Botón liberado')
    self.publish_twist()
def on_button_press_backward(self, instance):
    print('Botón Atras presionado')
    self.twist_msg.linear.x = -0.2
    self.twist_msg.linear.y = 0.0
    self.twist_msg.linear.z = 0.0
    self.twist_msg.angular.x = 0.0
    self.twist_msg.angular.y = 0.0
    self.twist_msg.angular.z = 0.0
    self.publisher.publish(self.twist_msg)
def on_button_press_left(self, instance):
    print('Botón Izquierda presionado')
    self.twist_msg.linear.x = 0.0
    self.twist_msg.linear.y = 0.2
    self.twist_msg.linear.z = 0.0
    self.twist_msg.angular.x = 0.0
    self.twist_msg.angular.y = 0.0
    self.twist_msg.angular.z = 0.0
```

```
        self.publisher.publish(self.twist_msg)
    def on_button_press_right(self, instance):
        print('Botón Derecha presionado')
        self.twist_msg.linear.x = 0.0
        self.twist_msg.linear.y = -0.2
        self.twist_msg.linear.z = 0.0
        self.twist_msg.angular.x = 0.0
        self.twist_msg.angular.y = 0.0
        self.twist_msg.angular.z = 0.0
        self.publisher.publish(self.twist_msg)
    def on_button_press_stop(self, instance):
        print('Botón Stop presionado')
        self.twist_msg.linear.x = 0.0
        self.twist_msg.linear.y = 0.0
        self.twist_msg.linear.z = 0.0
        self.twist_msg.angular.x = 0.0
        self.twist_msg.angular.y = 0.0
        self.twist_msg.angular.z = 0.0
        self.publisher.publish(self.twist_msg)
    def exit_app(self, instance):
        print('Saliendo de la aplicación')
        self.node.destroy_node()
        rclpy.shutdown()
        App.get_running_app().stop()
    def publish_twist(self):
        self.publisher.publish(self.twist_msg)
if __name__ == '__main__':
    MyApp().run()
```

ANEXO 7: Pruebas realizadas

Pruebas realizadas con el software controlador y la interfaz de control conectadas al robot manipulador.

En la Figura 34, la visualización detallada del código en la figura proporciona una comprensión profunda de la lógica de programación implementada basada en ROS2 y Micro-ROS, lo que facilita la identificación y corrección de posibles errores. La conexión directa entre la tarjeta ESP32 y el robot establece un entorno de prueba eficaz, permitiendo ajustes y optimizaciones en tiempo real. Este enfoque integral garantiza una evaluación exhaustiva y precisa del rendimiento del robot en escenarios variables, contribuyendo a un desarrollo más sólido y una implementación más eficiente de sus funciones.

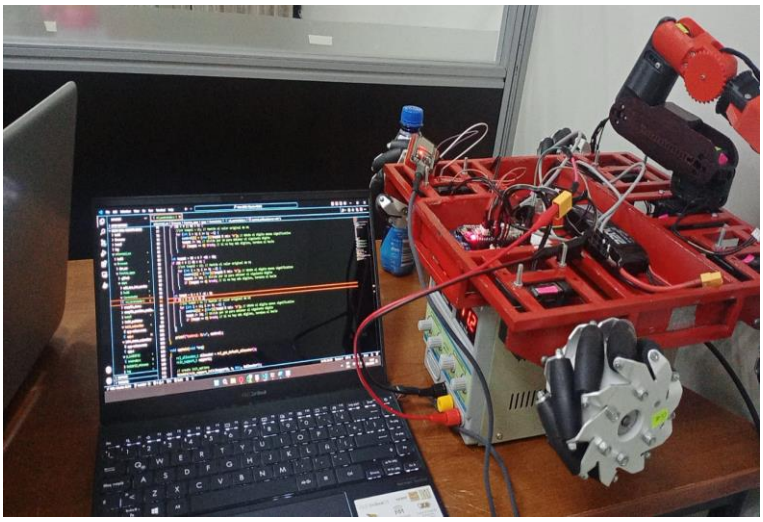


Fig. 29: Ejecución del software controlador hacia el robot.

En la Figura 35, la visualización detallada del código en la figura proporciona una comprensión profunda de la lógica de programación implementada basada en ROS2 y Micro-ROS, lo que facilita la identificación y corrección de posibles errores. La conexión directa entre la tarjeta ESP32 y el robot establece un entorno de prueba eficaz, permitiendo ajustes y optimizaciones en tiempo real. Este enfoque integral garantiza una evaluación exhaustiva y precisa del rendimiento del robot en escenarios variables, contribuyendo a un desarrollo más sólido y una implementación más eficiente de sus funciones.

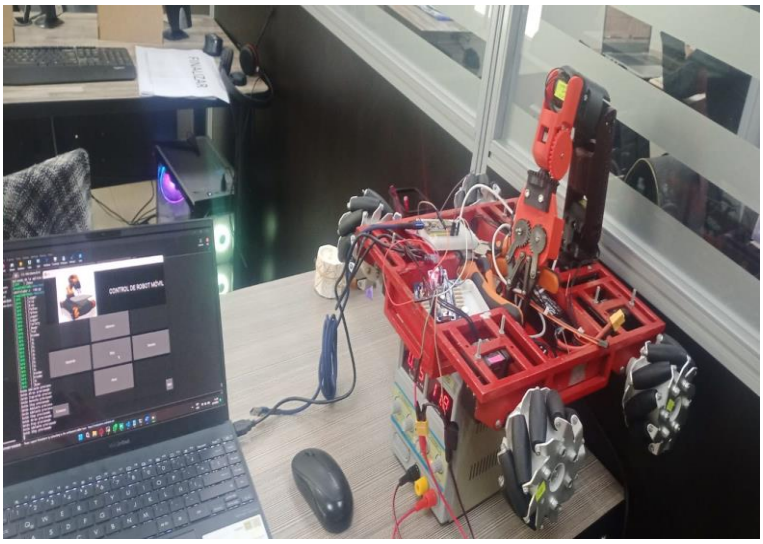


Fig. 30: Ejecución de la aplicación e interacción con el robot.

En la Figura 36, se aprecia la interacción fluida entre la aplicación desarrollada en Python y el robot. La interfaz proporciona botones que facilitan el envío de diferentes velocidades a las tarjetas, permitiendo así activar individualmente cada uno de los actuadores. Este sistema intuitivo y eficiente brinda un control preciso sobre el comportamiento del robot durante las pruebas. Además, la interfaz gráfica de la aplicación garantiza una experiencia de usuario mejorada, agilizando el proceso de ajuste y optimización de las velocidades de los actuadores para un rendimiento óptimo del robot.

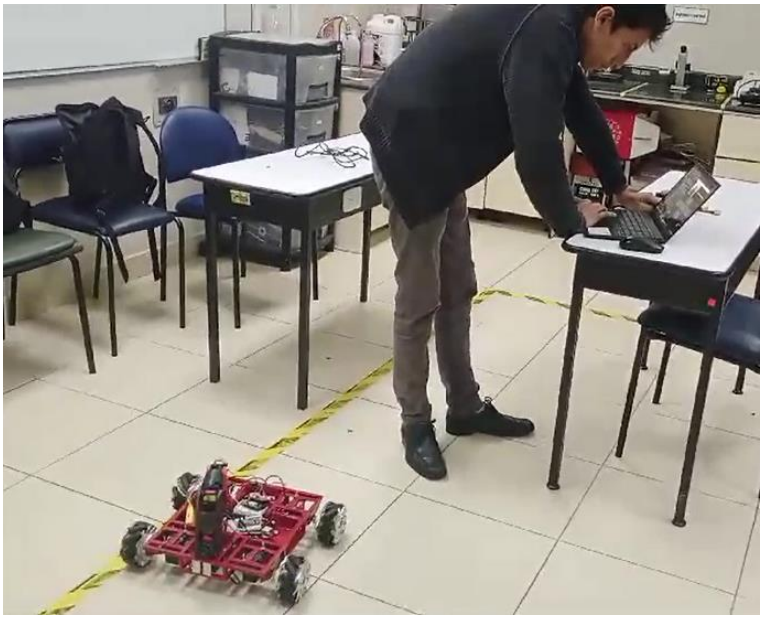


Fig. 31: Pruebas con el robot en el laboratorio.

ANEXO 8: Evidencia

Evidencia de la aplicación de las preguntas de aceptación.

La Figura 37 exhibe el formulario utilizado durante la aplicación al laboratorista a cargo del laboratorio de robótica en la Facultad de Ingeniería, Industria y Producción de la Universidad Tecnológica Indoamérica. Este instrumento es crucial para recopilar datos significativos sobre la aceptación de la tecnología en el contexto específico de este proyecto. La participación del laboratorista, como figura clave en el entorno de investigación y desarrollo, proporciona una perspectiva valiosa para evaluar cómo la tecnología propuesta se integra y satisface las necesidades específicas del laboratorio. La retroalimentación recabada a través de este formulario contribuirá significativamente a la comprensión de la viabilidad y utilidad de la tecnología implementada en el ámbito académico, estableciendo así una base sólida para posibles mejoras y desarrollos futuros.

Link del cuestionario: https://forms.gle/oHNACpKNpxFK8C1UA

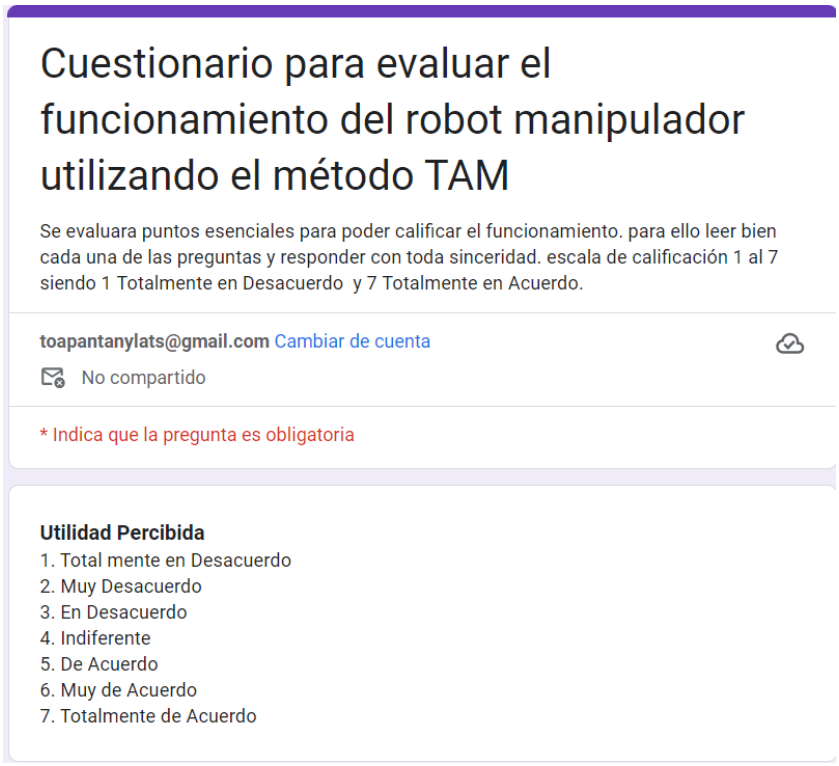


Fig. 32: Formulario de Google de la Aceptación de Tecnología TAM