Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Diseño e implementación de un software de gestión y planificación de rutas de un robot de servicio

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero/a en Ciencias de la Computación

Presentado por:

Gabriela Regina Ramos Baque

Joby Jorge Farra Solis

Guayaquil - Ecuador

Año: 2024

Dedicatoria

El presente proyecto lo dedico al Ingeniero Carlos Carvajal por haber sido un gran amigo y mentor durante mi carrera universitaria, también a mi compañero de tesis Joby Farra quien ha sido mi apoyo tanto en lo personal, como universitario y profesional.

Finalmente, a mis padres, amigos y familiares por creer en mí y estar a mi lado siendo mi aliento, permitiéndome llegar a donde estoy.

Gabriela Regina Ramos Baque

Dedicatoria

El presente proyecto lo dedico a mi mamá, abuelo y hermana que no dejaron que me rinda, tanto en el inicio como en el final de mi carrera, también a mi compañera de tesis Gabriela Ramos quien ha sido mi apoyo incondicional.

Finalmente, a mi papá quien siempre busco desafíos nuevos para mí, haciéndome autodidacta e independiente, y su apoyo en momentos cruciales de mi carrera.

Joby Jorge Farra Solis

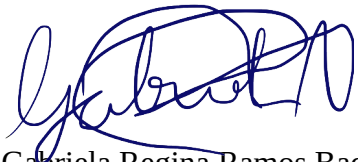
Declaración Expresa

Nosotros, Gabriela Ramos y Joby Farra, acordamos y reconocemos que: La titularidad de los derechos patrimoniales de autor del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores. La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique a los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 16 de enero del 2024.

Joby Jorge Farra Solis



Gabriela Regina Ramos Baque

Evaluadores

Firmado electrónicamente por:
LUIS EDUARDO
MENDOZA MORALES

Luis Eduardo Mendoza Morales, Ph.D.

Profesor de Materia

Firmado electrónicamente por:
DENNYS FABIAN
PAILLACHO CHILUIZA

Dennys Paillacho Chiluiza, Ph.D.

Tutor de proyecto

Resumen

Actualmente, el CIDIS está llevando a cabo un proyecto enfocado en el desarrollo de un robot mesero autónomo, inicialmente concebido para abordar la necesidad de desinfectar espacios interiores durante la pandemia, con capacidades de desplazamiento autónomo y reconocimiento de entorno, teniendo como principal limitación la carencia de una interfaz de usuario eficiente. Por lo tanto, el objetivo del presente proyecto es el desarrollo de dicha interfaz, la cual permita al usuario navegar y controlar el robot mesero según sus necesidades. Para la implementación y desarrollo de la interfaz se hizo uso del marco de desarrollo React Native, con lenguaje JavaScript, y con la técnica responsive para la adaptabilidad de dispositivos móviles Android, funcionando esencialmente como un joystick virtual. En cuanto a la integración, se utilizó la librería Roslibjs que permite la conexión con el robot mediante websockets, garantizando comunicación eficiente y en tiempo real, posibilitando el control del robot. Finalmente, se concluyó que la interfaz permite la navegación y gestión del robot mesero, además de que, al cumplir su propósito principal, se consolida como una herramienta indispensable que mejora la experiencia del usuario y maximiza la utilidad del robot en diversos escenarios.

Palabras claves: ROS, Roslibjs, Websockets, JavaScript, React Native.

Abstract

Currently, CIDIS is carrying out a project focused on the development of an autonomous waiter robot, initially conceived to address the need of disinfecting indoor spaces during the pandemic, with autonomous movement and environment recognition capabilities, having as main limitation the lack of an efficient user interface. Therefore, the objective of the present project is the development of such an interface, which allows the user to navigate and control the waiter robot according to his needs. For the implementation and development of the interface, the React Native development framework was used, with JavaScript language, and with the responsive technique for the adaptability of Android mobile devices, working essentially as a virtual joystick. In terms of integration, the Roslibjs library was used, which allows the connection with the robot through websockets, ensuring efficient and real-time communication, enabling the control of the robot. Finally, it was concluded that the interface allows the navigation and management of the waiter robot, in addition to fulfilling its main purpose, it is consolidated as an indispensable tool that improves the user experience and maximizes the usefulness of the robot in various scenarios.

Keywords: *ROS, Roslibjs, Websockets, JavaScript, React Native.*

Índice general

Abreviaturas

CIDIS	Centro de Investigación Desarrollo e Innovación de Sistemas Computacionales
ROS	Robot Operating System
LIDAR	Light Detection and Ranging
JSON	JavaScript Object Notation
iOS	iPhone Operating System
API	Application Programming Interface

Índice de figuras

Índice de tablas

CAPÍTULO 1

1. Introducción

En la actualidad, la tecnología forma parte de nuestra vida cotidiana por lo cual se ha adaptado y evolucionado para satisfacer las necesidades de los seres humanos en diversos contextos y circunstancias. Uno de los campos que más ha evolucionado es la tecnología robótica, la cual ha encontrado aplicaciones en todo tipo de industrias y escenarios, llegando a nuevos niveles en desarrollo de hardware y software.

La transición tecnológica ha abierto las puertas a innovaciones asombrosas, ejemplificado por el prototipo de robot desarrollado por el Centro de Investigación Desarrollo e Innovación de Sistemas Computacionales (CIDIS). Originalmente concebido como una respuesta a la pandemia del COVID-19 a finales del año 2019, este robot fue inicialmente utilizado para la desinfección de espacios interiores, una necesidad emergente en aquel momento, equipado con sensores y capacitado para navegar y reconocer objetos y personas en su entorno con precisión.

Con el paso del tiempo y el cambio en las circunstancias, la visión para este robot se ha modificado, siendo su nuevo objetivo el servir como un robot mesero, ampliando así su ámbito de aplicación y servicio. A nivel industrial, se tiene como objetivo el agilizar actividades y optimizar el tiempo de respuesta.

1.1 Descripción del Problema

En la actualidad, el CIDIS ha desarrollado un prototipo de robot mesero diseñado para la entrega de productos en espacios interiores. Este robot está diseñado con sensores avanzados que le permite la capacidad de navegar y reconocer tanto objetos como personas que se encuentran en su entorno. Aunque el robot ha sido construido y ajustado físicamente, enfrenta una limitación fundamental: la falta de una interfaz de usuario eficiente y amigable para su gestión y control.

Dicha carencia impacta de manera negativa en su operatividad y su habilidad para adaptarse de forma óptima a las variaciones en el entorno en el que opera.

1.2 Justificación del Problema

La eficiencia y efectividad de cualquier sistema robótico no solo dependen de su diseño físico y capacidades técnicas, sino también de cómo los humanos interactúan con él. En el contexto del CIDIS y su prototipo de robot mesero, la ausencia de una interfaz de usuario intuitiva y eficiente podría generar múltiples inconvenientes, como tiempos de respuesta lentos, errores en la entrega de productos o incluso posibles accidentes si el robot no responde adecuadamente a las instrucciones dadas por los usuarios. Además, una interfaz no amigable podría generar resistencia por parte del personal o clientes al utilizar el robot, lo que reduciría su aceptación y, por lo tanto, su utilidad.

Una interfaz bien diseñada permitirá que el robot mesero pueda ser empleado de manera más extensa, permitiendo a los usuarios gestionar y controlar sus acciones con confianza y precisión. Esto no solo mejoraría la experiencia del usuario al interactuar con el robot, sino que también maximizaría la inversión realizada en su desarrollo, garantizando que el robot pueda operar a su máxima capacidad y adaptarse rápidamente a cualquier cambio o situación inesperada en el entorno.

Por lo tanto, es importante abordar esta carencia para asegurar que el robot mesero pueda ser integrado de manera efectiva en espacios interiores, cumpliendo su propósito de entregar productos de manera eficiente y segura, y garantizando una experiencia positiva para todos los usuarios involucrados.

1.3 Objetivos

1.3.1 Objetivo General

Automatizar, parametrizar e integrar el robot mesero desarrollado por el CIDIS mediante el diseño e implementación de una interfaz de usuario amigable para personal no técnico, con el objetivo de facilitar su gestión y control en entornos interiores.

1.3.2 Objetivos Específicos

- Diseñar e implementar una interfaz de usuario que permita navegar y visualizar en tiempo real los movimientos del robot.
- Integrar la interfaz de usuario desarrollada al sistema existente del robot.
- Realizar pruebas para asegurar la correcta funcionabilidad y respuesta en tiempo real de la interfaz al interactuar con el robot.

1.4 Marco Teórico

1.4.1 Conceptos básicos

Sistema Operativo Ubuntu. Ubuntu, un sistema operativo de código abierto basado en Debian, se selecciona como plataforma para este proyecto debido a su enfoque en la accesibilidad y la amigabilidad hacia usuarios no familiarizados con Linux. La versión 20.04.6 de Ubuntu se elige específicamente para la instalación de ROS, asegurando estabilidad y compatibilidad con lo desarrollado por el CIDIS.

Sistema Operativo para Robots (ROS). También conocido como Sistema operativo para robots (en inglés, Robot Operating System) emerge como un conjunto avanzado de librerías y herramientas diseñadas para facilitar el desarrollo de software para robots [6]. ROS no es un framework en tiempo real, aunque actualmente existen herramientas que permite integrar con

código que permita obtener la información en tiempo real [14]. Para este proyecto ROS Noetic se adopta por su capacidad de abstracción del hardware y su continua evolución mediante actualizaciones y correcciones de errores.

Desplazamiento autónomo en Robótica. La capacidad de desplazamiento autónomo en robótica representa un avance significativo. Este concepto se refiere a la habilidad intrínseca de un robot para navegar y moverse de manera autónoma en su entorno, sin intervención humana directa [3]. Este enfoque implica la integración de diversos algoritmos y sistemas, respaldados por sensores, que permiten al robot tomar decisiones informadas y ejecutar acciones de manera independiente. [2]

Mapeo Robótico. En la esencia del proyecto, el mapeo robótico se presenta como una herramienta estratégica. Este proceso implica la creación de representaciones visuales y modelos del entorno circundante del robot, facilitando la toma de decisiones autónomas. A través de algoritmos y sensores, el mapeo robótico asigna puntos de referencia críticos para que el robot comprenda y adapte su posición en el espacio [9]. La creación del entorno está relacionada con la localización determinando grandes cantidades de información donde se indica los obstáculos [5] y el tamaño del entorno.

Herramientas de Comunicación en ROS. Para la comunicación del robot con la interfaz gráfica para el control y manejo de este debe existir un intermediario que permite el envío de información entre ambos, para esto se considera Rosbridge.

Rosbridge Suite. Rosbridge Suite se destaca como un conjunto integral de módulos ROS esenciales para la interacción con aplicaciones externas. Facilita la comunicación bidireccional mediante la transferencia de información en formato JSON, permitiendo una conexión efectiva entre ROS y plataformas web.

1.4.2 Herramientas gráficas de ROS

En el ámbito del desarrollo de software del robot, las herramientas gráficas son esenciales al ofrecer interfaces visuales intuitivas que facilitan la interacción con el robot. En este sentido, las siguientes herramientas cuenta con los medios efectivos para observar y poner a prueba el robot, evaluando el desempeño de sus sensores y algoritmos en entornos virtuales.

Rqt. Rqt se posiciona como un framework basado en Qt, esencial para el desarrollo de interfaces gráficas de usuario integradas con ROS [4]. Proporciona flexibilidad y eficiencia en la creación de interfaces intuitivas.

RViz. RViz, una herramienta de visualización 3D, se convierte en un recurso esencial para la observación y comprensión de la percepción del robot en su entorno tridimensional, contribuyendo a la toma de decisiones autónomas.

Gazebo. Gazebo, un simulador 3D, se presenta como un entorno virtual completo para el diseño, prototipado y prueba de robots. Con sus herramientas y bibliotecas de desarrollo en la nube, Gazebo facilita una evaluación exhaustiva del comportamiento del robot en diversos escenarios simulados.

CAPÍTULO 2

2.Metodología

2.1 Análisis General

En el marco del proyecto actual, se incorpora un robot avanzado, proporcionado por el CIDIS, que se fundamenta en la plataforma Arlo y se complementa con un sensor LIDAR. Esta combinación específica confiere al robot una notable capacidad de movilidad, gracias a su diseño de dos ruedas bidireccionales y motores operando de forma autónoma, vital para la exploración en espacios interiores. El sensor LIDAR juega un papel crucial, permitiendo la detección eficaz de obstáculos y la realización de un mapeo exhaustivo del entorno, lo que habilita una navegación autónoma y refinada del robot.

Con el objetivo de manejar de manera efectiva y maximizar la funcionalidad, se ha previsto el desarrollo de una interfaz gráfica asegurando una interacción fluida y directa con el sistema ROS, facilitando la ejecución de comandos y el seguimiento del estado del robot en tiempo real. La creación de esta interfaz gráfica es un elemento fundamental del proyecto, ya que no solo enriquece la interacción usuario-robot, sino que también proporciona los medios para controlar con precisión las operaciones del robot, ampliando su versatilidad y aplicabilidad en distintos ámbitos prácticos.

2.2 Análisis de Requerimientos

El desarrollo de una interfaz gráfica robusta y eficiente para el control del robot móvil conlleva una comprensión detallada de los requerimientos necesarios. Estos requerimientos abarcan tanto las capacidades funcionales como las consideraciones estéticas y de usabilidad, asegurando que la interfaz final sea tanto operativa como accesible para los usuarios, independientemente de su experiencia técnica.

2.2.1 Requerimientos No Funcionales

La interfaz debe ser diseñada con un enfoque intuitivo y de fácil uso. Esto implica:

- **Usabilidad:** La interfaz deberá ser comprensible y manejable para usuarios de todos los niveles técnicos, permitiendo una experiencia de usuario fluida y sin frustraciones.
- **Estética:** Un diseño atractivo y coherente contribuye a una mejor experiencia del usuario, incentivando su uso continuado y facilitando la interacción.
- **Accesibilidad:** Consideración de normas de accesibilidad para asegurar que la interfaz sea utilizable por una amplia gama de usuarios, incluyendo aquellos con discapacidades.

2.2.2 Requerimientos Funcionales

La interfaz debe contar con funcionalidades claves para el control efectivo del robot:

- **Conexión por IP:** Capacidad para conectar y desconectar con el robot mediante una dirección IP específica, facilitando un método de comunicación directo y fiable.
- **Control de Desplazamiento:** Implementación de controles intuitivos para el movimiento del robot, incluyendo la navegación manual y la opción de mapeo para que el robot se dirija automáticamente a un punto designado.
- **Transmisión de Video:** Integración de una función de transmisión de video en tiempo real para visualizar el entorno desde la perspectiva del robot, mejorando la interacción del usuario con el dispositivo y su entorno.

Estos requerimientos funcionales son fundamentales para el diseño y desarrollo de una interfaz que no solo permita a los usuarios interactuar de manera efectiva con el robot, sino que también enriquezca su experiencia al ofrecer una visión detallada del entorno operativo del robot.

El enfoque metodológico adoptado garantiza una implementación estructurada y eficaz, desde el análisis inicial hasta las pruebas finales, asegurando que el robot cumpla con los estándares requeridos para su operación autónoma y controlada.

2.3 Historias de Usuario

Conforme a los requisitos identificados, se han desarrolló historias de usuario que se centran en las funcionalidades clave que desempeñará el robot dentro del contexto de servicio y entrega de pedidos (véase Tabla 1). Estas historias de usuario están diseñadas para alinear las expectativas del comportamiento del robot con las necesidades operativas del interior.

Tabla 1

Historias de Usuario

Enunciado de la Historia			Criterios de Aceptación				
Rol	Característica / Funcionabilidad	Razón / Resultado	Número (#) de Escenario	Criterio de Aceptación (Título)	Contexto	Evento	Resultado
Como dueño de un restaurante	Necesito agilizar las entregas dentro de mi local.	Con la finalidad de dar servicio de calidad y eficiencia en el menor tiempo posible.	1	Entrega de pedidos sin larga espera.	Restaurante tiene alta demanda de pedidos.	Cuando es hora de almuerzo o fin de semana.	Robot mesero toma pedido y el encargado lo recepta para su preparación.
			2	Entrega de pedidos sin larga espera.	Restaurante no cuenta con suficiente personal para la entrega de pedidos	Existen cierta cantidad de meseros, pero no lo suficiente para llevar muchos pedidos.	Robot mesero toma pedido y el encargado lo recepta para su preparación.
			3	Entrega de pedidos sin larga espera.	Personal con permiso médico.	Al existir menos cantidad de personal no se puede cubrir la demanda.	Robot mesero toma pedido y el encargado lo recepta para su preparación.
Como encargado de la entrega de pedidos	Necesito visualizar en pantalla los puntos de mis mesas.	Con la finalidad de facilitar entregas, sin necesidad de llevar el control.	1	Entregas autónomas.	Agilizarlas entregas.	Seleccionar un punto especifico de entrega	Robot mesero se dirige al punto señalado y entrega la orden.

Nota: Historias de usuarios establecidas por el cliente, donde cuenta con el rol, características, razón y número de escenario

La Tabla 1 presenta un conjunto de historias de usuario organizadas en varias columnas que delinean las metas, características, escenarios y criterios de aceptación para el despliegue del robot en un entorno de restaurante. Cada historia describe un requisito funcional del robot, como la entrega de pedidos y la navegación entre mesas, junto con el contexto de aplicación, los criterios específicos que determinarán el éxito de la tarea, y los resultados esperados. Estos resultados incluyen respuestas rápidas a los pedidos, la capacidad de manejar picos de demanda y la entrega precisa en la ubicación designada.

2.4 Prototipo

En base a los requerimientos que se desean cubrir con la aplicación, se realizó un prototipo para poder visualizar las interfaces y funcionales que se incluirá dentro de la misma.

Para ello se cuenta con 5 vistas, de las cuales se tiene la página de inicio, como en la figura 1, que es la primera que visualizará el usuario al iniciar la aplicación móvil, y mostrarán las opciones de conexión, control, mapeo y configuración, las cuales podemos ver en las figuras 2,3,4 y 6 respectivamente.

Figura 1

Pantalla de Inicio

Figura 2

Pantalla de Conexión Pendiente

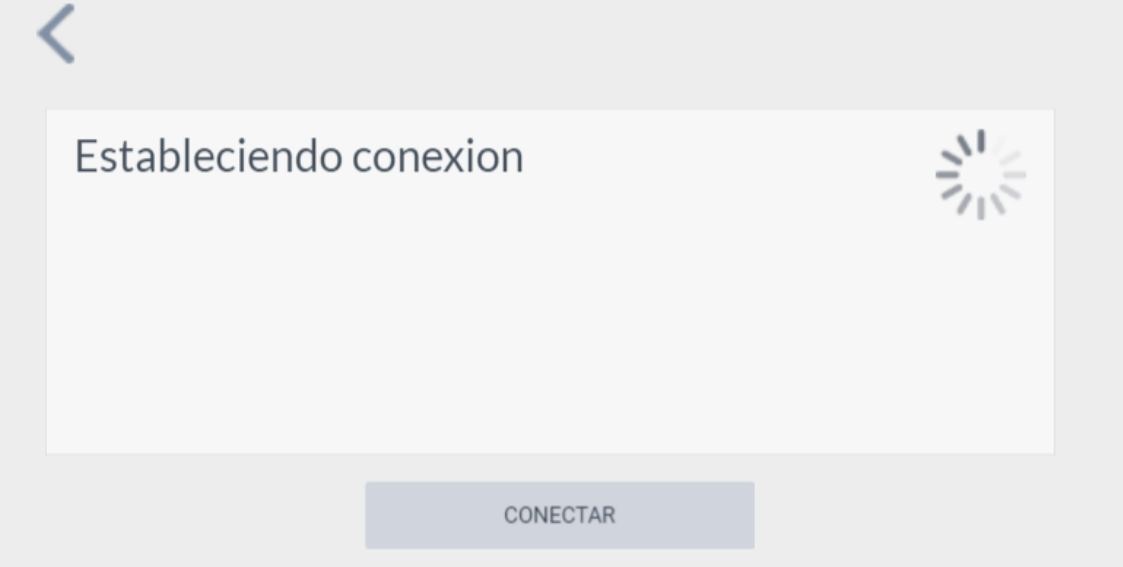


Figura 3

Pantalla de Control

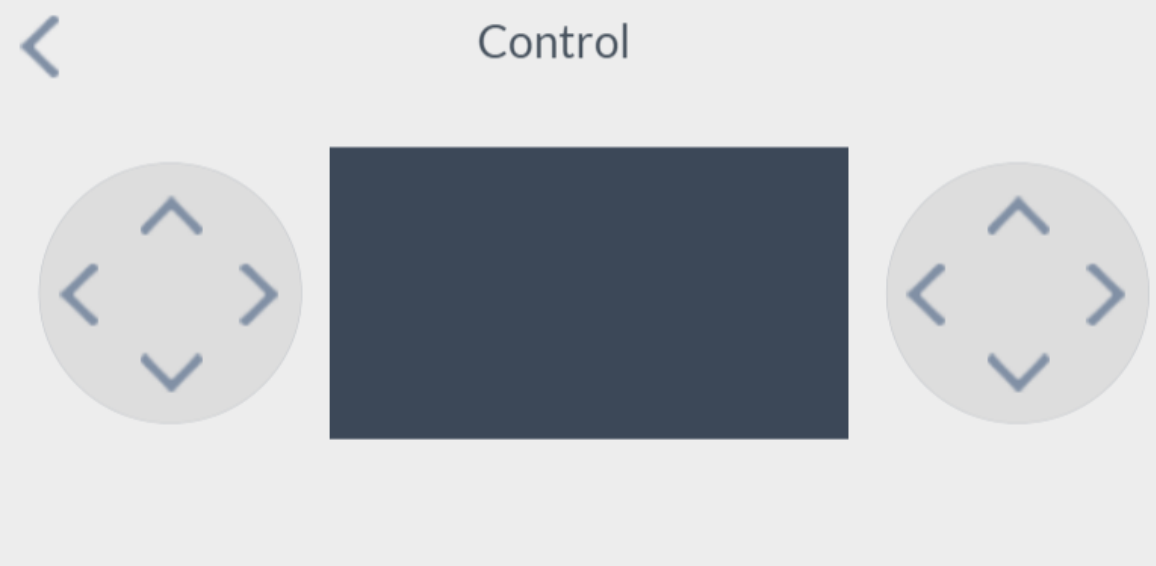


Figura 4

Pantalla de Mapeo

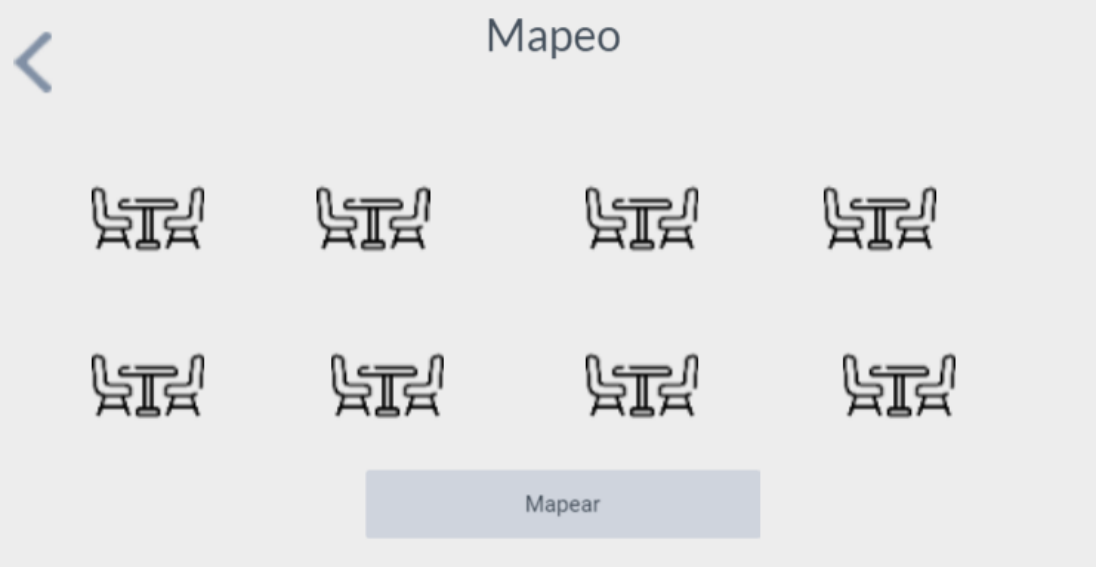


Figura 5

Pantalla de Configuración

2.5 Diseño de la solución

La solución propuesta busca satisfacer los requerimientos esenciales del proyecto mediante la implementación de funcionalidades clave para el control manual y autónomo del robot, así como para el escaneo del ambiente interior.

La arquitectura de la solución se centra en aspectos técnicos fundamentales, incluyendo la selección de una distribución adecuada de ROS, la compatibilidad entre diferentes versiones de paquetes, la configuración óptima de los nodos para asegurar una comunicación fluida entre ROS y la aplicación, y la identificación de las dependencias críticas para el funcionamiento del sistema.

2.5.1 Vista de Desarrollo

El diagrama de componentes, ilustrado en la Figura 6, presenta la arquitectura del sistema diseñado para el control y monitoreo de un robot. La aplicación móvil, desarrollada utilizando React Native, sirve como la interfaz de usuario y se comunica con el sistema operativo de robótica (ROS) a través de Roslibjs utilizando Websockets. Esta comunicación es bidireccional, permitiendo tanto el envío de comandos y la publicación de tópicos como la recepción de datos en tiempo real desde el robot.

En la figura, se muestra el componente de la Plataforma Arlo, el cual ejecuta scripts que traducen las instrucciones enviadas desde la aplicación al robot. Este componente gestiona diversos nodos de ROS que procesan y transmiten datos entre la aplicación y el robot.

Además, se incluye el componente Sensor LIDAR en el sistema, que opera utilizando paquetes y scripts específicos de ROS para facilitar el escaneo ambiental a través de la aplicación y permitir la navegación autónoma del robot utilizando mapas previamente escaneados.

El componente de Websockets actúa como un canal de comunicación crucial para el intercambio de mensajes entre la aplicación móvil y ROS, asegurando que las instrucciones y la telemetría fluyan de manera eficiente y en tiempo real.

2.6 Plan de Implementación

El desarrollo de la solución propuesta se organizó en cinco fases críticas: Desarrollo de la Interfaz Gráfica, Implementación de la Interfaz, Levantamiento del ROS Bridge, Integración del Sistema ROS con la Interfaz Gráfica, y Pruebas en Entorno Virtual. Esta estructura se diseñó para abordar tanto la funcionalidad como la usabilidad del sistema.

1. Desarrollo de la Interfaz Gráfica: Se seleccionará React Native como la tecnología subyacente debido a sus ventajas en la creación de aplicaciones tanto para iOS como para

Android. Esta elección se fundamenta en la reutilización de componentes y la disponibilidad de bibliotecas que agilizan el desarrollo y las pruebas de la interfaz.

2. Implementación de la Interfaz: La implementación se centrará en asegurar una experiencia de usuario intuitiva y funcional, aprovechando las capacidades de React Native para crear interfaces responsivas y adaptativas.

3. Levantamiento del ROS Bridge: Esta fase implica establecer un puente de comunicación entre ROS y la aplicación mediante el uso de ROS Bridge, facilitando el intercambio de mensajes y comandos.

4. Conexión entre ROS y la Aplicación: Se utilizará la biblioteca Roslibjs para la integración con el sistema ROS. Dado su diseño en JavaScript y la facilidad de conexión a través de web sockets, Roslibjs se presenta como una solución eficiente para la comunicación entre la interfaz gráfica y ROS.

5. Pruebas en Entorno Virtual: La fase final comprende pruebas exhaustivas en simuladores y entornos virtuales para validar la funcionalidad y el rendimiento del sistema antes de su implementación real.

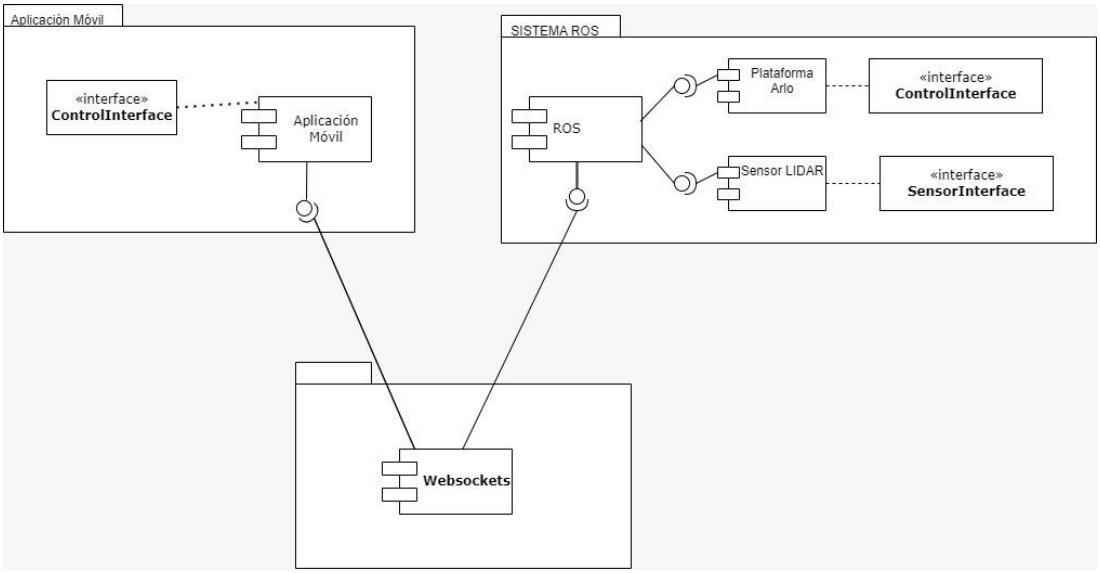
Tabla 2

Plan de desarrollo

TAREA			FECHA DE INICIO	FECHA DE FIN	DURACIÓN (DÍAS)	% DE TAREA COMPLETADO
ID	**TÍTULO**	**DUEÑO**				
1	**Definición, planificación y estimación**					**100,00%**
1,1	Introducción al proyecto	Joby Farra - Gabriela Ramos	25/9/23	27/9/23	3	**100%**
1.2	Levantamiento de requerimientos	Joby Farra - Gabriela Ramos	5/10/23	9/10/23	3	**100%**
1.3	Revisión de lineamientos	Joby Farra - Gabriela Ramos	10/10/23	12/10/23	3	**100%**
1.4	Introducción a ROS	Joby Farra - Gabriela Ramos	12/10/23	18/10/23	5	**100%**
1,5	Documento V1 (Marco Teórico - Problemática - Objetivos)	Gabriela Ramos	13/10/23	19/10/23	5	**100%**
2	**Diseño de solución**					**100%**
2,1	Diseño de estructura base (Escenario de interior - Restaurante)	Joby Farra	13/10/23	16/10/23	2	**100%**
2,2	Dieño de estructura base (Escenario de interior - Oficina)	Joby Farra	15/10/23	17/10/23	2	**100%**
2,3	Diseño de interfaz gráfica (Prototipo)	Gabriela Ramos	19/10/23	21/10/23	2	**100%**
3	**Desarrollo**					**100%**
3,1	Interfaz gráfica versión 1, requerimientos bases	Gabriela Ramos	22/10/23	24/10/23	2	**100%**
3.2	Conexion Robot - Interfaz	Joby Farra	26/10/23	30/10/23	3	**100%**
3.3	Pruebas - conexion	Joby Farra - Gabriela Ramos	31/10/23	1/11/23	2	**100%**
3.4	Movimiento - Joystick	Joby Farra	6/11/23	8/11/23	3	**100%**
3.5	Pruebas - Movimiento	Joby Farra - Gabriela Ramos	9/11/23	11/11/23	2	**100%**
3.6	Interfaz gráfica versión 2, vision restaurante	Gabriela Ramos	13/11/23	15/11/23	3	**100%**
3.7	Mapeo Ambiente	Joby Farra - Gabriela Ramos	15/11/23	16/11/23	2	**100%**
3.8	Movimiento - Autonomo	Joby Farra	17/11/23	18/11/23	1	**100%**
3.9	Pruebas - Autonomo	Joby Farra - Gabriela Ramos	18/11/23	21/11/23	2	**100%**
4	**Revisión y retroalimentación**					**100%**
4,1	Pruebas con el cliente	joby Farra - Gabriela Ramos	24/11/23	27/11/23	2	**100%**
4,2	Modificaciones	joby Farra - Gabriela Ramos	7/12/23	8/12/23	2	**100%**
4,3	Pruebas con el cliente final	joby Farra - Gabriela Ramos	17/12/23	19/12/23	2	**100%**

Figura 6

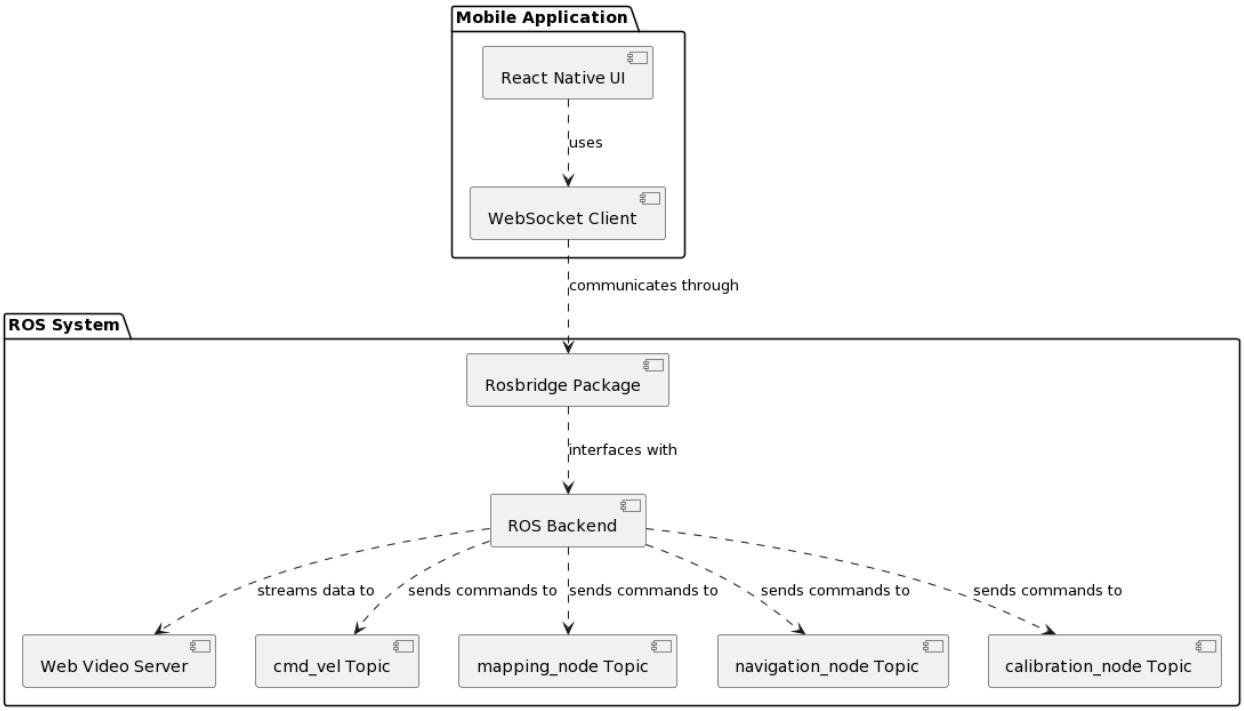
Diagrama de Componentes



En la Figura 7, se ilustra la conexión entre la interfaz y el sistema ROS, destacando cómo los mensajes se envían desde la aplicación hasta interactuar con el robot. La conexión mencionada es a través de websockets, donde se puede evidenciar que para ciertos paquetes son de recepción de datos mientras otros presentan la transferencia bidireccional. Tanto los nodos de movimiento, escaneo, y navegación se le envía instrucciones al robot para que luego se interpreten en sus respectivos paquetes y se observe que el robot realice alguna actividad.

Figura 7

Diagrama de comunicación



CAPÍTULO 3

3.Resultados y Análisis

3.1 Fase 1: Desarrollo de la interfaz gráfica

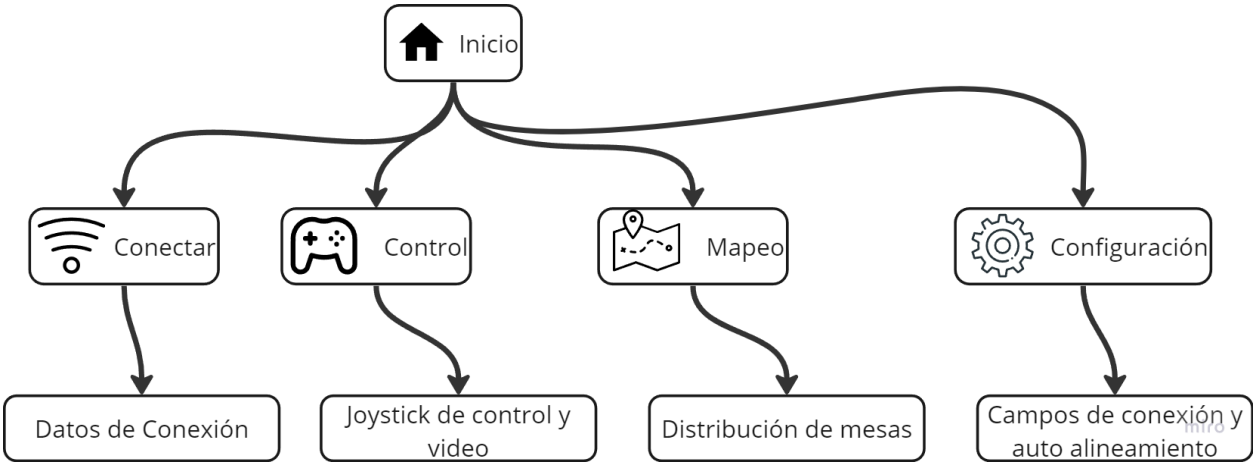
Para construir una interfaz de usuario eficiente que permita el manejo efectivo del robot, se desarrolló un flujo de actividades detallado, ilustrado en la Figura 8. Bajo la estructura facilitada por React Native, hemos desarrollado cuidadosamente la interfaz gráfica, con un enfoque específico en dispositivos móviles que operan con el sistema operativo Android.

Además, durante el proceso de diseño, se implementó una estrategia responsive para adaptarse con destreza a la diversidad de tamaños de pantalla presentes en la amplia gama de dispositivos móviles. Este enfoque proactivo asegura que la interfaz gráfica no solo sea adaptable, sino también visualmente coherente, proporcionando una experiencia de usuario consistente y optimizada en cualquier dispositivo utilizado.

La interfaz diseñada consta de cinco pantallas principales, cada una con un propósito específico para garantizar una experiencia de usuario fluida y funcional.

Figura 8

Diagrama de flujo de aplicación



3.2 Fase 2: Implementación de la Interfaz grafica

Es importante mencionar que, en comparación con el prototipo inicial, se realizaron modificaciones significativas en la interfaz lo cual tenemos:

Pantalla de Inicio, mostrada en la Figura 9, actúa como el núcleo central ya que con este se logrará navegar a las demás pantallas de la aplicación. La interfaz está diseñada para ser intuitiva y de fácil manejo, incorporando iconos representativos como botones de acceso directo que facilitan al usuario la identificación rápida de cada funcionalidad clave, permitiendo así una navegación rápida y eficiente dentro de la aplicación.

Figura 9

Pantalla de Inicio

Pantalla "Conectar" está diseñada específicamente para gestionar la conexión entre la aplicación y el sistema operativo de robótica ROS del robot. Esta pantalla informa al usuario sobre el estado actual de la conexión, indicando si la aplicación ya está sincronizada con el robot o si se necesita establecer una nueva conexión, detalles que se ilustran en las Figuras 10 y 11. En caso de que la aplicación no esté conectada, se desplegará una ventana emergente, como se muestra en la Figura 12, solicitando al usuario que introduzca la información necesaria para establecer la conexión, incluyendo la dirección IP y el puerto específico del robot.

Figura 10

Nueva vista de Conexión

Figura 11

Conexión exitosa

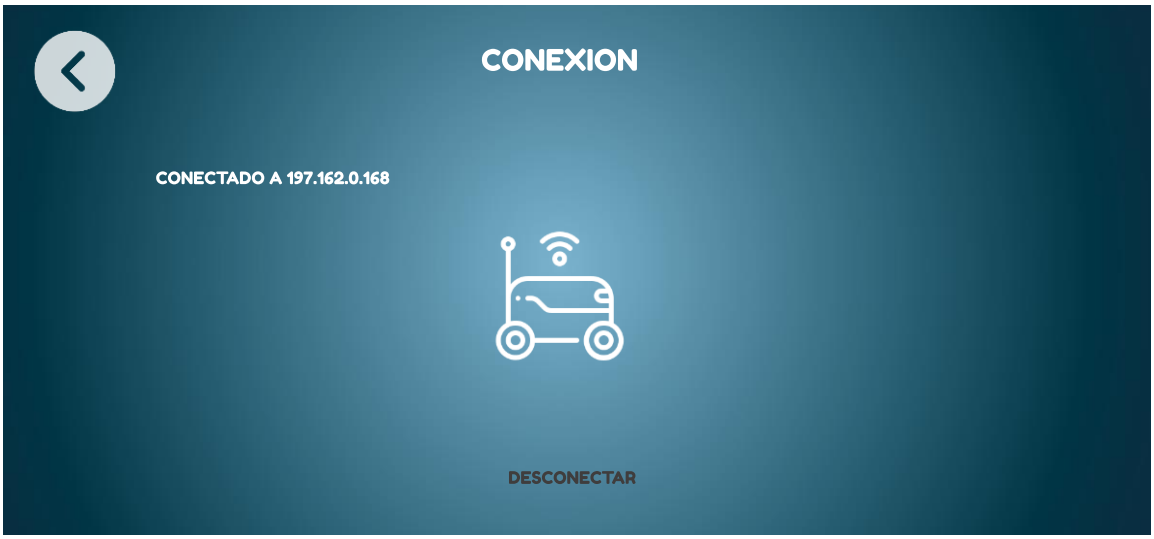
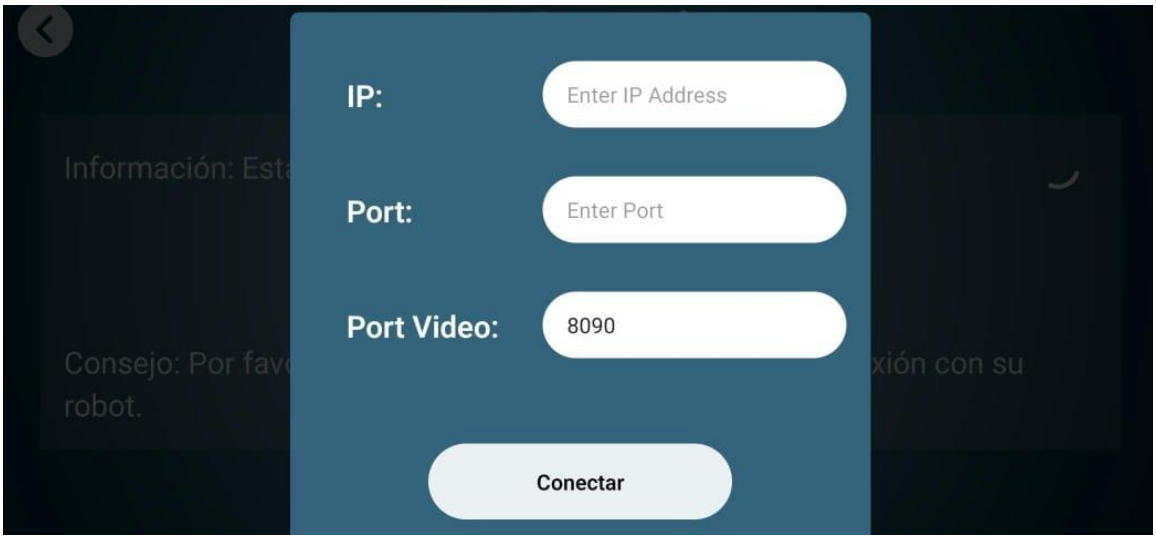


Figura 12

Formulario de Conexión

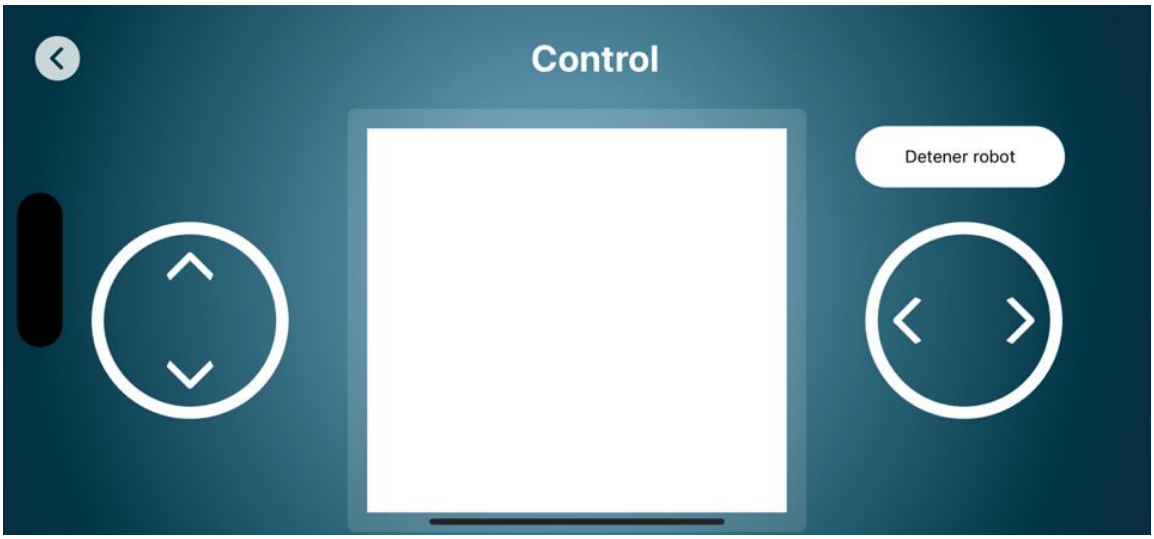


Pantalla “Control”, representada en la Figura 13, es la encargada de facilitar el movimiento a través de un joystick virtual. En esta se realizaron ajustes fundamentales para adecuarse a la naturaleza del robot mesero, el cual utiliza exclusivamente dos ruedas. Esta particularidad implica un movimiento bidireccional en lugar del movimiento completo, originalmente propuesto.

Además, incluye una transmisión en vivo de vídeo, capturado por la cámara del robot, proporcionando una vista en tiempo real del entorno en el que se encuentra dicho robot. Dichos ajustes se implementaron con el objetivo de permitir una gestión más precisa, segura y específica del desplazamiento del robot, asegurando un rendimiento óptimo y alineándose de manera más exacta con las capacidades reales del dispositivo.

Figura 13

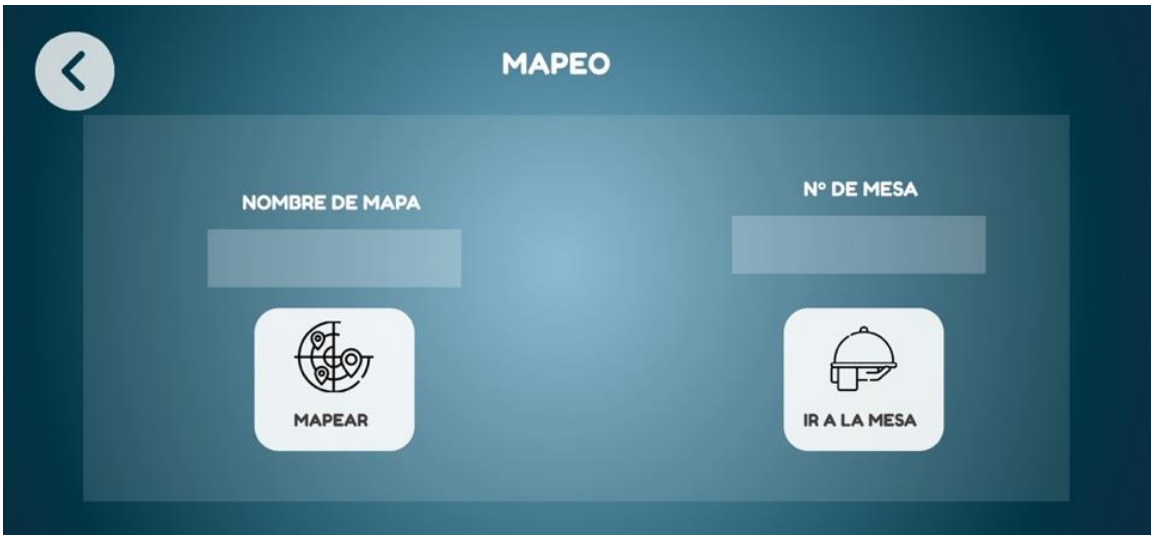
Nueva vista de Control



La pantalla de Mapeo, ilustrada en la Figura 14, permite al usuario señalar un punto específico al robot, indicándole la dirección a seguir hacia dicho punto. Sin embargo, durante el proceso de implementación, se observó que la presentación de la distribución de las mesas genera un consumo considerable de recursos. Ante esta consideración, se han realizado ajustes en esta pantalla, buscando un equilibrio óptimo entre la representación visual de la disposición de las mesas y el rendimiento general del sistema. Este cambio estratégico se orienta hacia una gestión más eficiente de los recursos disponibles, asegurando una experiencia de usuario fluida y operaciones del sistema más ágiles. En esta, se debe enviar el topic correspondiente para que se realice el mapeo y una vez logrado, ingresar en el campo, la mesa (punto) a donde se deba dirigir.

Figura 14

Nueva vista de Mapeo



Por último, la pantalla Configuración, como se ve en la Figura 15, contiene opciones para ajustar y personalizar la configuración del robot. Esto incluye la posibilidad de cambiar puertos para distintos servicios y un área dedicada al auto alineamiento. En esta pantalla, los usuarios pueden introducir el identificador de un código QR específico que el robot necesita escanear, lo que es crucial para tareas de auto alineamiento y orientación espacial.

Figura 15

Nueva vista de Configuraciones

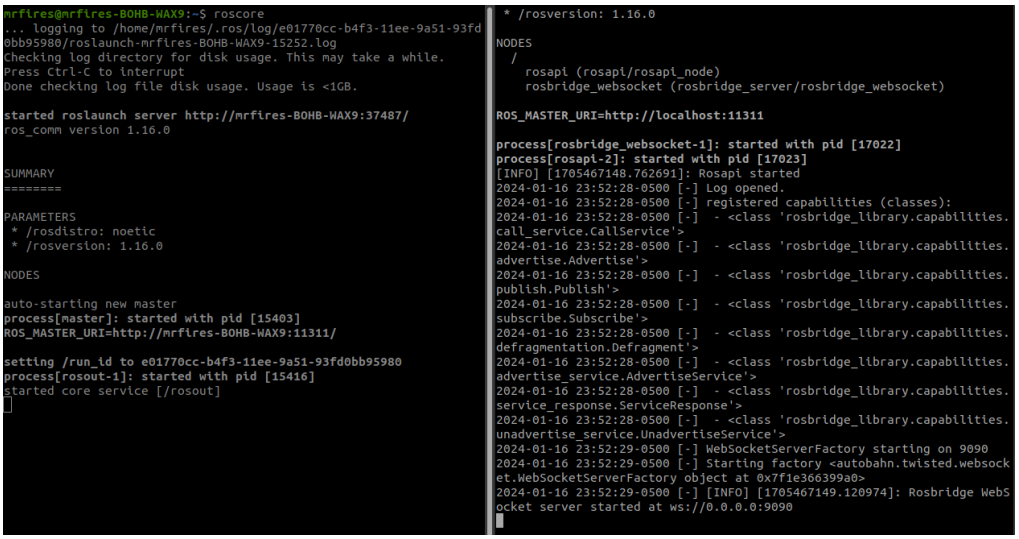
3.3 Fase 3: Levamiento ROS Bridge

Un componente clave para la integración exitosa de nuestra aplicación con el robot fue la activación del servicio rosbridge_server, utilizando el puerto 9090 para facilitar la comunicación entre el sistema operativo de robótica (ROS) y la interfaz gráfica de usuario. Este paso era esencial para establecer un canal de comunicación confiable y eficiente, permitiendo el intercambio de comandos y datos en tiempo real entre la aplicación y el robot.

La confirmación de una conexión exitosa, como se muestra en la Figura 16, validó la efectividad de nuestra configuración y el uso de rosbridge_server como un enlace crítico para la operación del sistema. Este logro subraya la importancia de una arquitectura de comunicaciones robusta, asegurando la interactividad y el control preciso del robot a través de la interfaz gráfica, fundamentales para la funcionalidad y autonomía del proyecto.

Figura 16

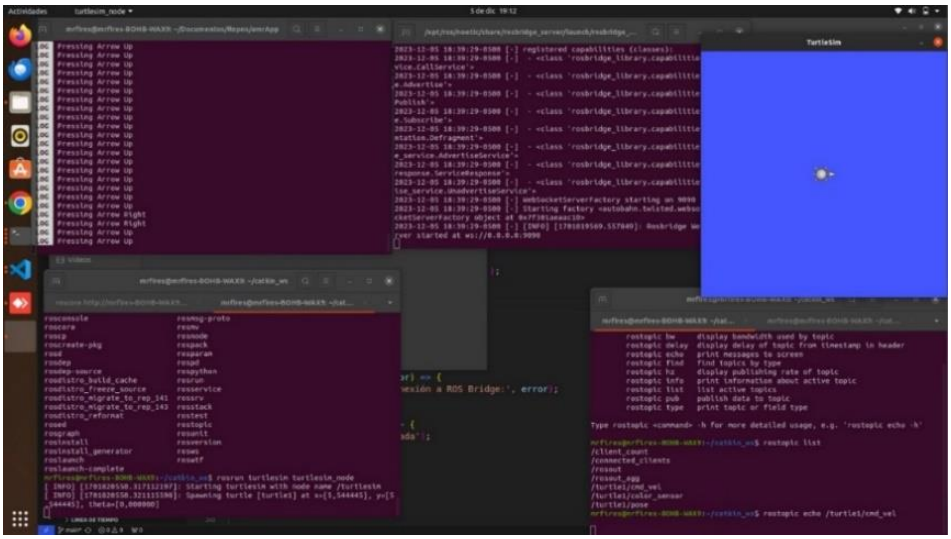
Levantamiento de RosCore y Rosbridge



Durante esta fase de desarrollo, fue crucial implementar diversos servicios, incluyendo el web_video_server y el turtlesim, entre otros. Para optimizar la gestión de estos servicios y aumentar la resiliencia del sistema, se adoptó la estrategia de ejecutar cada servicio en una terminal independiente. Esta aproximación permitió una rápida recuperación ante posibles fallos, asegurando que cualquier servicio afectado pudiera ser reiniciado de inmediato sin interrumpir el funcionamiento global del sistema.

Figura 17

Levantamiento de los nodos para pruebas en entorno virtual



3.4 Fase 4: Implementación de la conexión entre ROS y la Aplicación

Para facilitar una comunicación efectiva entre la aplicación y el robot, hemos implementado el paquete rosbridge_suite, una herramienta crucial dentro de ROS que proporciona una API versátil. Este paquete incluye soporte para diversas interfaces y un sistema de websockets, lo que lo hace indispensable para nuestra configuración. La principal ventaja de rosbridge_suite es su capacidad para establecer una conexión directa mediante una dirección IP y un puerto específico, lo que permite una interacción fluida y confiable con los diferentes nodos que operan dentro del ecosistema ROS.

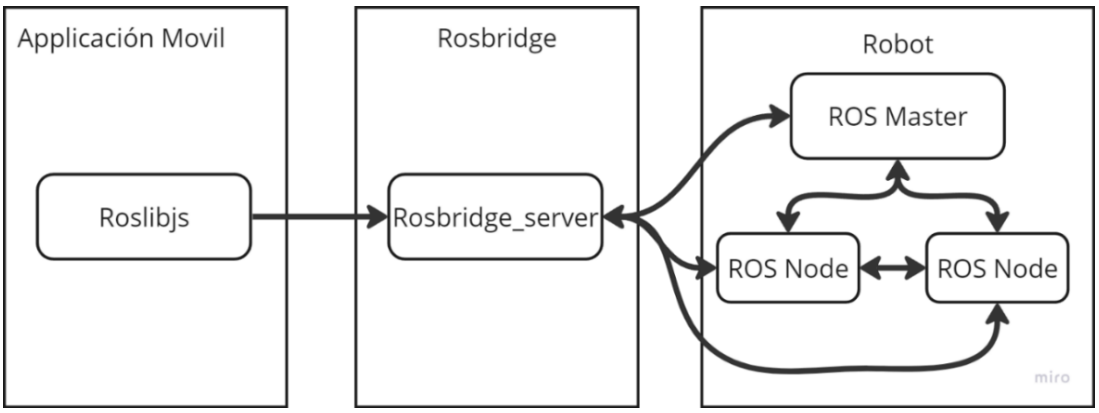
Del lado de la aplicación, hemos seleccionado Roslibjs, una biblioteca de JavaScript específicamente diseñada para trabajar con ROS a través de websockets desempeñando la comunicación bidireccional.

Para ilustrar detalladamente el proceso de cómo se establece esta conexión y la forma en que los datos fluyen entre la aplicación y ROS, recomendamos revisar la Figura 7. Esta

ilustración proporciona una visión clara de nuestra infraestructura de comunicación, mostrando el esquema detallado de conexión y el flujo de información, lo que facilita la comprensión de la integración efectiva de estas tecnologías en nuestro proyecto.

Figura 18

Conexión entre App y ROS con Rosbridge



3.4.1 Generación de movimientos del robot a través de la interfaz

Tras establecer con éxito la conexión mediante el paquete rosbridge_suite, la siguiente etapa implica el control efectivo del movimiento del robot utilizando el nodo "/cmd_vel". Este nodo es crucial para enviar instrucciones de movimiento al robot, utilizando el tipo de mensaje geometry_msgs/Twist. Dentro de la aplicación, se configura una instancia de ROS topic específicamente para este nodo, lo que nos permite publicar mensajes dirigidos al control del movimiento. Las instrucciones de movimiento se codifican de la siguiente manera, asegurando que el robot interprete y ejecute las acciones deseadas correctamente:

- Adelante: Publicamos un mensaje con un valor positivo en el eje x (e.g., 0.2, 0.0, 0.0), impulsando al robot hacia adelante.
- Atrás: Un valor negativo en el eje x resulta en el movimiento hacia atrás del robot.

- Detener: Publicando un mensaje con todos los valores en cero, el robot se detiene completamente.
- Giro a la Izquierda: Un valor positivo en el eje z hace que el robot gire hacia la izquierda.
- Giro a la Derecha: De manera similar, un valor negativo en el eje z orienta al robot hacia la derecha.

La implementación de estos comandos permite una manipulación detallada y precisa del robot, facilitando el control sobre su dirección y velocidad de manera intuitiva. Esta capacidad de control es fundamental para la navegación autónoma del robot, permitiéndole responder a las instrucciones de movimiento de forma eficaz y realizar tareas con mayor precisión.

3.4.2 Integración de transmisión de video desde las cámaras del robot

Para habilitar la transmisión en tiempo real de la cámara integrada en el robot, se establece una conexión con el nodo /d435/color/image_raw. Este nodo es crucial ya que actúa como el canal a través del cual las imágenes capturadas por la cámara del robot se hacen accesibles para su visualización. Para facilitar la distribución y acceso a estos datos visuales, se utiliza el paquete web_video_server, el cual sirve las imágenes capturadas por el nodo a través de un puerto específico, haciendo posible su visualización en diferentes dispositivos.

En el lado del cliente, específicamente dentro de la interfaz de la aplicación, se emplea un componente webview para mostrar la transmisión de video en tiempo real. Este enfoque permite integrar de manera eficiente la visualización de video dentro de la aplicación, ofreciendo a los usuarios la capacidad de ver directamente lo que el robot 've' en su entorno, sin demoras significativas.

3.4.3 Implementación del escaneo utilizando el sensor LIDAR

En esta fase del proyecto, el objetivo es activar el escaneo del entorno utilizando el sensor LIDAR, que es fundamental para generar un mapa detallado del área circundante. Este mapa incluirá elementos críticos como paredes y objetos, los cuales son esenciales para la navegación eficiente del robot. Para manejar este proceso, se implementa el nodo mapping_node, diseñado específicamente para gestionar las operaciones de escaneo.

El funcionamiento de mapping_node se basa en la recepción de mensajes con una estructura específica: estado, nombre_mapa. Aquí, 'estado' indica si el escaneo debe comenzar o terminar, con '0' asignado para el inicio del escaneo y '1' para su finalización. Es crucial, además, incluir en el mensaje el 'nombre_mapa', que se utilizará para guardar el mapa generado en la memoria del robot para su referencia y uso futuro. Esta estructura de mensaje permite un control preciso sobre el proceso de escaneo, asegurando que los mapas se creen y almacenen de manera efectiva según las necesidades del proyecto.

3.4.4 Generación de rutas autónomas desde la interfaz

En la generación de rutas autónomas, se utilizará un nodo llamado “navigation_node” en el cual se debe publicar un mensaje de forma “nombre_mapa, id_mesa”, al momento de iniciar la navegación del robot, este recibirá el mensaje anteriormente descrito y busca en el mapa pre guardado, además del identificador de la mesa que contiene las coordenadas donde el robot generará una ruta, con la opción de detectar objetos y esquivarlos.

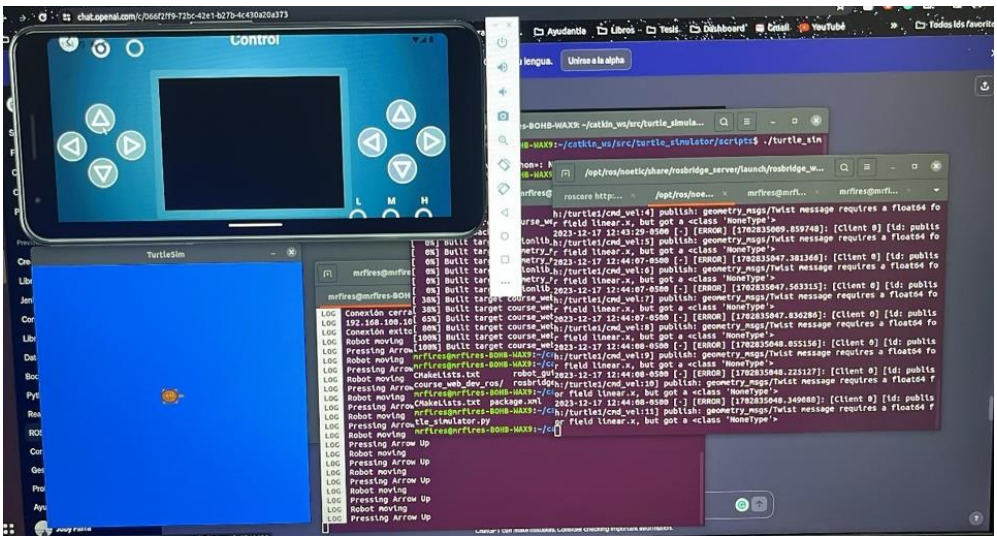
3.5 Fase 5: Pruebas en entorno virtual

Se llevaron a cabo múltiples pruebas exhaustivas en un entorno virtual con el propósito de evaluar minuciosamente el desempeño de la aplicación. En este contexto, se verificó con éxito la generación de movimiento en el nodo designado, tal como se detalla en la Figura 19. Estas

pruebas se centraron en asegurar que la aplicación respondiera de manera precisa y coordinada a las instrucciones de control, especialmente aquellas relacionadas con el movimiento del robot mesero.

Figura 19

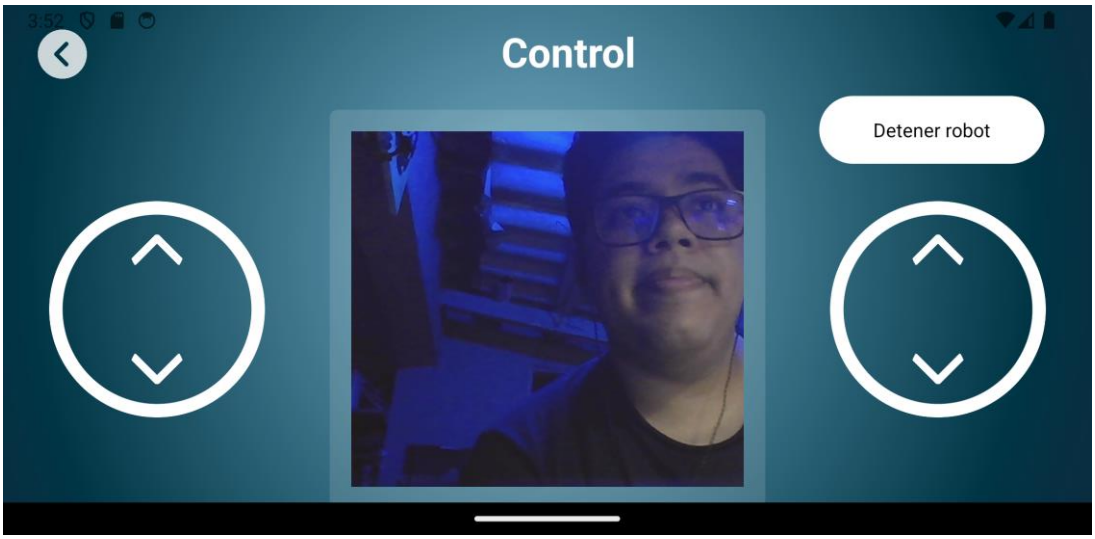
Prueba de conexión en entorno simulado



Adicionalmente, se sometió a prueba la funcionalidad de transmisión de video en la pantalla de control, conforme se muestra en la Figura 20. La verificación exitosa de la transmisión de vídeo respalda la aplicación para integrar información visual de manera eficiente y confirma que la interfaz de control proporciona al usuario una representación precisa y actualizada de la perspectiva del robot, contribuyendo a una interacción más efectiva y segura.

Figura 20

Prueba de transmisión de video



Estas pruebas en el entorno virtual no solo validaron la implementación técnica de la aplicación, sino que también sentaron las bases para un despliegue exitoso en entornos del mundo real, al proporcionar insights valiosos sobre el rendimiento integral de la aplicación y su capacidad para cumplir con los requisitos operativos previstos.

3.6 Limitaciones de la solución

La solución planteada, aplicación móvil, presenta grandes ventajas como son la portabilidad y la adaptabilidad en los diferentes dispositivos, pero al brindar ventajas también debe tener sus puntos débiles, tales como:

- **Licencias de distribución:** Para generar la distribución de la aplicación en tiendas oficiales, se necesita un proceso largo de trámites y pruebas para que la aplicación cumpla lo solicitado por estos distribuidores.

- **Cambios o Actualización:** Si se requiere que realizar cambios sobre un topic o la implementación de un nuevo apartado, esto requiere que se genere una nueva versión de la aplicación por lo que esta corrección llevaría tiempo hasta que pase las pruebas de calidad.
- **Latencia:** Debido a que la aplicación y el robot deben estar en una misma red, esto lleva a que, si existen más dispositivos en la red, pueden generar tráfico en la red, por lo que existirá retraso en los envíos de comandos o en la transmisión de video.
- **Almacenamiento:** La aplicación solicita al dispositivo en que se va a instalar tenga el almacenamiento suficiente por lo generaría un conflicto para usuarios que tienen problemas de almacenamiento. Lo que generaría que el uso de aplicación tenga un dispositivo dedicado.

Cabe destacar que el éxito de este proyecto y su cambio a la propuesta inicial radica, en gran medida, en la implementación exitosa de una interfaz amigable que se adapta dinámicamente para optimizar la eficiencia, versatilidad y los recursos del sistema en su conjunto para la operatividad de diversos escenarios.

CAPÍTULO 4

4. Conclusiones y Recomendaciones

4.1 Conclusiones

- En el presente proyecto se ha logrado automatizar y facilitar el control del robot mesero mediante una interfaz de usuario diseñada para personal no técnico. Esta interfaz intuitiva permite una gestión eficiente en entornos interiores, marcando un avance en la tecnología robótica aplicada a la hospitalidad. Este éxito abre caminos para futuras mejoras y adaptaciones del robot, promoviendo una mayor integración de soluciones robóticas en el servicio. Se destaca el potencial de la robótica para transformar servicios, demostrando el impacto positivo de la tecnología en la interacción humana y operativa.
- La efectiva conexión entre la aplicación y el robot ha facilitado una comunicación fluida y eficiente a través de la interfaz, mientras que la implementación de un sistema de control manual ha permitido al robot emular con precisión el movimiento característico de un tanque. Esta funcionalidad brinda a los usuarios un control intuitivo y preciso sobre el dispositivo.
- La creación de una interfaz de usuario diseñada específicamente para aquellos sin conocimientos técnicos especializados representa un avance significativo. La integración eficiente de esta interfaz con el robot mesero desarrollado por el CIDIS marca un progreso destacado en la gestión y control del robot en entornos interiores. Estos logros se traducen en una experiencia de usuario mejorada y en una optimización notable de las operaciones del sistema automatizado.
- La interfaz amigable no solo mejora la accesibilidad para usuarios no técnicos, sino que también contribuye a la eficiencia general del sistema. En conjunto, estas mejoras subrayan

la capacidad del sistema para adaptarse a diversas situaciones operativas, proporcionando un rendimiento óptimo y una experiencia de usuario intuitiva.

Para concluir, la combinación de una interfaz eficiente, un sistema de control preciso y una comunicación efectiva establece las bases para un sistema automatizado robusto y versátil, consolidando así el éxito y la utilidad de este proyecto.

4.2 Recomendaciones

Para mejorar el rendimiento del sistema y optimizar la eficiencia operativa, se proponen las siguientes recomendaciones estratégicas:

- Para garantizar una comunicación más estable y minimizar la latencia en la transmisión de órdenes, es aconsejable que el dispositivo ejecutando la aplicación esté conectado a la misma red local que el robot o los robots. Esta medida es crucial para mantener una sincronización fluida entre las instrucciones emitidas y las acciones del robot.
- Es esencial asegurar que la estación base del robot se mantenga fija en una ubicación invariable. Esta estabilidad es clave no sólo para la precisión en el reconocimiento de las mesas y otros puntos de referencia dentro del entorno operativo, sino también para el correcto autoajuste y alineación del robot, lo cual es indispensable para su funcionamiento adecuado y autonomía.

Referencias

[1] C. Crick, G. Jay, S. Osentoski, B. Pitzer y O. C. Jenkins, "Rosbridge: Ros for non-ros users," en Robotics Research: The 15th International Symposium ISRR, Springer International Publishing, 2017, pp. 493-504.

[2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, "Autonomous Mobile Robots," 2nd ed. The MIT Press, 2011.

[3] V. Narváez et al., "Diseño e Implementación de un Sistema de Localización y Mapeo Simultáneos (SLAM) para la Plataforma Robótica ROBOTINO®," Rev. Politécnica, vol. 33, núm. 1, febrero 2014.

[4] ROS Wiki, "rqt," ROS Wiki. Available at: http://wiki.ros.org/rqt

[5] Cabrera, A. and Delgado, G. (2014) 'https://dspace.uazuay.edu.ec/bitstream/datos/3610/1/10292.pdf'. Cuenca: Universidad del Azuay.

[6] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open-source software, vol. 3, no. 3.2, May 2009, p. 5.

[7] Ochoa Zambrano, J.S. (2014) *Diseño e implementación de un asistente móvil con desplazamiento Autónomo Basado en dispositivos Android*, *Repositorio Institucional de la Universidad Politécnica Salesiana: Página de inicio*. Available at: https://dspace.ups.edu.ec/handle/123456789/7225.

[8] Coello Kondratova, A.S. (2022) *Desarrollo de una aplicación para el manejo remoto de un robot de desinfección.* thesis. ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL.

[9] J. M. Rapado García, "Diseño e implementación de una interfaz gráfica de usuario para mapeado de entornos y navegación en ROS," Ph.D. dissertation, Universitat Politècnica de València, 2016.

[10] Joseph, L. (2018) *Robot Operating System for Absolute Beginners: Robotics Programming*, *SpringerLink*. Aluva, Kerala: Apress. Available at: https://link.springer.com/book/10.1007/978-1-4842-7750-8 (Accessed: 2024).

[11] C. Stachniss, "Robotic Mapping and Exploration," Springer Tracts in Advanced Robotics, 2009

[12] *Wiki ros.org*. Available at: https://wiki.ros.org/rosbridge.

[13] *Wiki ros.org*. Available at: https://wiki.ros.org/roslibjs.

[14] *Wiki Ros* (2018) *ros.org*. Available at: https://wiki.ros.org/ROS/Introduction.

Apéndice A

MANUAL DE INSTALACIÓN BACKEND

1. Requerimientos

Se requiere un computador con Ubuntu 20.04 Focal, el cual pueden encontrar en la página oficial de Ubuntu. Y para ROS Noetic, se necesita seguir los pasos en la documentación oficial de ROS.

2. Instalación paquetes ROS

Para instalar los paquetes adicionales se necesita ejecutar el siguiente comando:

- sudo apt-get install ros-noetic-turtlesim ros-noetic-web-video-server ros-noetic-rosbridge-suite ros-noetic-image-transport ros-noetic-cv-bridge.

3. Crear un catkin Workspace

Se necesita crear un workspace para poder incluir scripts de paquetes personalizados. Para lo cual necesita ejecutar los siguientes comandos

a. mkdir -p ~/catkin_ws/src
b. cd ~/catkin_ws/src
c. catkin_init_workspace
d. git clone https://github.com/jjfarra/test_robot
e. cd ~/catkin_ws
f. catkin_make

g. chmod +x ~/catkin_ws/streamcam/src/camera_publisher.py

h. chmod +x ~/catkin_ws/src/robot_test/src/turtlesim_listener.py

i. source devel/setup.bash

j. En una nueva terminal ejecutar: roscore

k. En una nueva terminal ejecutar: rosrun streamcam camera_publisher.py

l. En una nueva terminal ejecutar: rosrun robot_test turtlesim_listener.py

m. En una nueva terminal ejecutar: roslaunch rosbridge_server rosbridge_websocket.launch

n. En una nueva terminal ejecutar: rosrun turtlesim turtlesim_node

o. En una nueva terminal ejecutar: rosrun web_video_server web_video_server _port:=8090

p. En una nueva terminal ejecutar: roslaunch usb_cam usb_cam-test.launch

Para conocer la ip del equipo con ros, se puede ejecutar el comando: ip a

Apéndice B

MANUAL DE INSTALACIÓN FRONTEND

En esta sección se encuentra lo necesario para el alzamiento del proyecto en el ambiente de desarrollo a nivel de frontend, es decir, la interfaz gráfica.

Requerimientos:

- Node js (preferencia versión 18.17.1)
- Simulador. Puede ser Android Studio o Expo para sistema operativo OS.

Pasos:

1. git clone *https://github.com/gabrielaraamoss/amrApp.git*
2. Ubicarse en la ruta del proyecto
3. Eliminar la carpeta *node_modules* en caso de existir
4. En el terminal escribir *npm install*
5. Una vez completado el proceso, en el terminal mismo escribir *npm start*
6. Según el simulador, seleccionar el de preferencia.

Nota: Los cambios que se realicen en el código fuente, se verá reflejado al instante sin necesidad de refrescar o alzar el proyecto por cada cambio.