ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Desarrollo de un sistema para el manejo remoto de un robot de desinfección.

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Ciencias de la Computación

Presentado por:

Alan Sergio Coello Kondratova

GUAYAQUIL - ECUADOR

Año: 2022

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; **Alan Sergio Coello Kondratova** doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Alan Sergio Coello
Kondratova

EVALUADORES

Firmado electrónicamente por:
ERICK VICENTE LAVID CEDENO

Erick Vicente Lavid Cedeño

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
DENNYS FABIAN PAILLACHO CHILUIZA

Dennys Fabián Paillacho Chiluiza

PROFESOR TUTOR

RESUMEN

El CIDIS tiene en su posesión un robot móvil, el cual puede ser utilizado para desinfectar superficies cercanas gracias a sus lámparas de luz UV, sin embargo, no es posible controlarlo de manera remota sin tener conocimiento en varios conceptos avanzados como ROS y la línea de comandos de Linux. Por lo tanto, el objetivo de este proyecto es desarrollar una interfaz gráfica que permita al usuario navegar el robot, así como también controlar la activación y desactivación de las lámparas de desinfección.

Se hizo uso de HTML, CSS y JavaScript junto al framework Vue para desarrollar la interfaz gráfica y de ROS, ROS Web Tools, Roslibjs y otras herramientas relacionadas a ROS para realizar la conexión entre la simulación del robot y la interfaz gráfica de usuario. Además, se utilizó los programas Rviz y Gazebo para montar el entorno de simulación.

Se desarrolló una interfaz gráfica de usuario que permite navegar la simulación del robot, visualizar la imagen captada por la cámara frontal del robot en tiempo real, tener retroalimentación de los valores enviados al robot en tiempo real, tener retroalimentación de la activación del modo de desinfección y controlar el proceso de desinfección.

Se concluyó que la utilización de un joystick para navegar el robot es un modo preciso de navegación además de concluir que el uso de la interfaz no se limita al robot de desinfección, sino que a muchos robots que también hacen uso de ROS.

Palabras clave: Interfaz ROS, Robot de desinfección, GUI, Aplicación Web.

ABSTRACT

CIDIS has in its possession a mobile robot, which can be used to disinfect nearby surfaces thanks to its UV light lamps, however, it is not possible to control it remotely without having knowledge of various advanced concepts such as ROS and the Linux command line. Therefore, the objective of this project is to develop a graphical interface that allows the user to navigate the robot, as well as control the activation and deactivation of the disinfection lamps.

HTML, CSS, and JavaScript were used together with the Vue framework to develop the graphical interface and ROS, ROS Web Tools, Roslibjs and other ROS-related tools to make the connection between the robot simulation and the graphical user interface. In addition, the Rviz and Gazebo programs were used to set up the simulation environment.

A graphical user interface was developed that allows navigating the robot simulation, visualizing the image captured by the robot's front camera in real time, having feedback on the values sent to the robot in real time, having feedback on the activation of the disinfection mode and control the disinfection process.

It was concluded that the use of a joystick to navigate the robot is an accurate mode of navigation, in addition to concluding that the use of the interface is not limited to the disinfection robot, but to many robots that also make use of ROS.

Keywords: *ROS Interface, Disinfection Robot, GUI, Web Application.*

ÍNDICE GENERAL

ABREVIATURAS

CIDIS Centro de Investigación, Desarrollo e Innovación de Sistemas Computacionales

ESPOL Escuela Superior Politécnica del Litoral

ROS Robot Operating System

HTML The HyperText Markup Language

CSS Cascading Style Sheets

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

En el entorno existen varios virus y bacterias que influyen negativamente en el ser humano, infectando el cuerpo, multiplicándose y causando enfermedades. Después de dejar el cuerpo de una persona infectada estos pueden alojarse en superficies de diferentes materiales (metales, madera, concreto, etc.) durante varias horas durante las cuales pueden infectar seres vivos. Los virus son los causantes de aproximadamente 60% de las infecciones humanas alrededor del mundo y su transmisión puede ser por distintos medios, entre los cuales se encuentra el contacto con superficies infectadas [1].

Una de las formas de ayudar a evitar el contagio de enfermedades es desinfectar correctamente espacios comunes y públicos para disminuir el riesgo de contagio, especialmente en la actualidad en la cual los empleos y clases presenciales están tomándose en todo el mundo y el uso de mascarillas ya no es obligatorio en todos los lugares.

Desinfectar una superficie se puede realizar con líquidos desinfectantes como ácido hipocloroso, amonio cuaternario, peróxido de hidrógeno, etc. Pero estos no son eficientes cuando se quiere desinfectar superficies de gran extensión como aulas, oficinas y salas de espera, además de dejar restos peligrosos en lugares como salas de quirófano y salones de clase.

La desinfección por luz ultravioleta o UV ha demostrado ser una alternativa eficaz para eliminar distintos virus y bacterias de superficies y objetos, además de no dejar rastro de químicos o componentes que puedan hacer daño a la salud [2]. La radiación UV solo presenta efectos negativos en la salud si se tiene exposición directa a ésta, pero de otras formas es inofensiva y no supone daños en materiales (metales, plásticos, minerales de construcción, pintura, madera) con los cuales pueda tener contacto en el momento de la desinfección.

1.1 Descripción del problema

Actualmente, existe en posesión del CIDIS un prototipo de un robot móvil de desinfección manejado por comandos en ROS, pero debido a que el producto se desea en un futuro comercializar de manera que cualquier persona pueda utilizarlo sin una capacitación previa, se requiere una manera simplificada de que los futuros usuarios puedan manejar todas las funciones del robot.

La forma actual de control del robot no permite que este se aleje mucho del usuario, lo cual puede suponer problemas de salud debido a la exposición a la luz ultravioleta. Los sectores que pueden ser desinfectados por el robot pueden localizarse en lugares de acceso limitado para el usuario, como tuberías y alcantarillados, por lo que también es necesario que se pueda manejar a distancia.

El manejo por cables no es recomendable debido a que el prototipo puede desplazarse por el suelo y los cables pueden interferir con su desplazamiento o engancharse en objetos por el camino, obstaculizando el movimiento del robot. El robot tiene múltiples ruedas motrices y directrices que permiten el desplazamiento hasta la superficie objetivo para desinfección.

Aparte del desplazamiento, se desea controlar de manera remota la desinfección, esto es, encender y apagar la lámpara ultravioleta presente en el prototipo a voluntad para desinfectar la superficie hasta la cual se manejó el robot.

1.2 Justificación del problema

Actualmente se tiene un prototipo de robot desinfectante, pero en un futuro se desea comercializar el robot, para lo cual es necesario que pueda utilizarlo cualquier persona como, por ejemplo, los miembros del personal de limpieza de hospitales y colegios.

La radiación UV emitida por la lámpara de desinfección puede resultar dañina para la piel del usuario en caso de exposición extendida, por lo que es necesario que el

método de manejo del robot sea remoto, para así mantener al usuario a una distancia segura al momento de encender esta lámpara para iniciar la desinfección.

Además, el CIDIS tiene una línea de investigación y desarrollo de robots con distintas finalidades como transporte logístico y agentes comerciales inteligentes (robots para ayudar a los clientes de centros comerciales) por lo cual un método para controlar el robot de desinfección de manera remota podría ser utilizado como fundamento para desarrollar métodos similares para manejar el resto de los robots. Los robots de transporte logístico y agentes comerciales comparten muchas de las características del robot de desinfección por lo que una adaptación del método de manejo del prototipo podría ser implementado para cada uno.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar una aplicación web para el control remoto de un robot móvil de desinfección.

1.3.2 Objetivos Específicos

1. Elaborar de una interfaz gráfica de usuario que permita manejar el movimiento y la función de desinfección del robot de manera remota.
2. Conectar la interfaz gráfica de usuario al robot móvil de desinfección.
3. Realizar una prueba de concepto del uso de la interfaz gráfica de usuario en el control del robot.

1.4 Marco teórico

1.4.1 Simulación

Ubuntu

Ubuntu es una distribución de Linux compuesta por software libre y de código abierto basada en la distribución Debian [3]. La versión más reciente y utilizada como sistema operativo para la instalación de ROS, Rosbridge_server y Python 3 es Ubuntu 18.04.

Python 3

Python es un lenguaje de programación de alto nivel, interpretado, de uso general y dinámicamente tipado. Soporta distintos paradigmas de programación como estructurado, orientado a objetos, y programación funcional [4].

SimpleHttpServer

Es una herramienta que provee Python 3 para crear un servidor simple local, con esta se puede alojar aplicaciones y páginas web localmente [5].

ROS

ROS o Robot Operating System es un middleware utilizado en sistemas robóticos complejos. Es un conjunto de frameworks para desarrollo de software para robots y brinda servicios de abstracción del hardware [6].

Vue

Vue es un framework para construir interfaces de usuario, utilizado comúnmente para crear aplicaciones de una sola página [7]. Vue permite el fácil manejo de variables y funciones tratadas como métodos.

Características de ROS

- **Peer to Peer**

Ros tiene una arquitectura no centralizada en la cual distintos procesos en la misma red tienen una comunicación utilizando topología punto a punto. ROS nos permite de esta manera evitar el uso de un servidor central para la comunicación entre procesos.

- **Multilenguaje**

ROS tiene soporte para múltiples lenguajes de programación como C++, LISP, Python, etc.

1.4.2 Herramientas gráficas de ROS

- **RViz**

  RViz es una interfaz gráfica que permite visualizar información mediante la utilización de plugins.

- **Rqt**

  Rqt es un framework basado en Qt para desarrollo de interfaces gráficas de usuario para ROS consistente en 3 paquetes:

  -rqt

  -rqt_common_plugins: Suite de herramientas backend de ROS.

  -rqt_robot_plugins: Herramientas para interactuar con robots durante su tiempo de ejecución.

- **Gazebo**

  Gazebo es un simulador 3D con el cual es posible crear escenarios tridimensionales con obstáculos, objetos, robots, etc. En el computador [8]. Los simuladores son esenciales para probar el desempeño de un robot en distintos escenarios simulados y así determinar las capacidades y limitaciones de este.

1.4.3 Mapeo

Es la acción de dibujar la silueta del espacio circundante al robot. El mapa es una generalización del ambiente en el que se encuentra el robot, en este se indican puntos de referencia para la localización de este y localización de obstáculos [9]. Para generarse, el robot utiliza sensores como Lidar y Radar.

1.4.4 Localización

Es la calculación del posicionamiento del robot respecto al espacio que lo rodea, utilizando sensores de proximidad [9]. Los algoritmos de localización se sirven de sensores y cámaras de profundidad y los datos permiten crear caminos para el desplazamiento basado en la posición en el mapa.

1.4.5 Interfaz gráfica de usuario para operación de robots móviles

Las interfaces gráficas de usuario pueden ser desplegadas en servidores web en computadoras. Para la comunicación con el robot se puede utilizar cuadros de entrada de texto en los cuales se introduce el IP y el puerto del robot para luego realizar la conexión.

En proyectos anteriores, se han utilizado métodos de conexión similares con éxito, como en el caso de José Manuel Rapado [10] que realizó una interfaz gráfica de usuario que también permite controlar un robot de desinfección.

Compañías como Turtlebot también han desarrollado software para el manejo remoto de robots, específicamente los robots que cuenten con sus kits para construcción de robots remotos [11]. El Turtlebot4, versión más reciente de su kit, utiliza ROS para navegación tanto remota como autónoma gracias a su funcionalidad de trazado de mapas.

1.4.6 Paquetes de ROS para conectividad e interoperabilidad de sistemas robóticos

- **Rosbridge_server**

  Rosbridge_server es parte del paquete de rosbridge_suite y proporciona una capa de transporte WebSocket [12]. Un WebSocket es una capa de comunicación en 2 direcciones de baja latencia entre clientes (como navegadores web) y servidores. Al proporcionar una conexión WebSocket, el servidor rosbridge permite que las páginas web se comuniquen con ROS utilizando el protocolo rosbridge.

- **Robot Web Tools**

  Este proyecto busca aumentar la interoperabilidad de los sistemas robóticos y las interfaces gráficas de usuario mediante bibliotecas de visualización web, uso de robótica en la nube. [13]
  Entre las herramientas de Robot Web Tools encontramos:
  -Foxglove Studio: Herramienta de visualización y debugging para datos.
  -ROS Control Center: Centro de control para robots ROS.
  -ROSweb: Sistema de supervisión web para ROS.
  -ROSboard: Herramienta para convertir un robot en un servidor web.

1.4.7 Comunicación Interfaz-Robot

- **TCP**

  Protocolo de Control de Transmisión, es un protocolo de red que permite a 2 anfitriones o hosts conectarse e intercambiar flujos de datos [14]. Garantiza que los datos y paquetes se entreguen en el orden en el que fueron enviados.

- **WebSocket**

  Es un protocolo de red basado en TCP que determina la metodología de intercambio de datos entre redes y que proporciona canales de comunicación full-duplex en una sola conexión TCP [15]. Es una conexión persistente entre un cliente y un servidor, que se utiliza como protocolo principal en la conexión mediante Rosbridge.

- **API REST**

  Api Rest es una Interfaz de programación de aplicaciones (API) que cumple principios de diseño de la arquitectura REST [16], los cuales son:

- Interfaz uniforme.
- Desacoplamiento entre el servidor y el cliente.
- Sin estado.

- Capacidad de almacenamiento en Caché.
- Arquitectura en capas.

  APIs REST pueden ser utilizadas cuando un robot ROS se utiliza como servidor web y sus acciones pueden ser tomadas como servicios.

CAPÍTULO 2

2. METODOLOGÍA

2.1. Análisis

Para controlar el robot de manera remota es necesario desarrollar una interfaz gráfica la cual permita llevar a cabo las funcionalidades del robot como el movimiento y accionamiento de la desinfección. Existen varias opciones para desarrollar la interfaz gráfica como la creación de una aplicación móvil nativa o una aplicación de escritorio, sin embargo la opción que se ha escogido con anterioridad en interfaces gráficas como la de Rapado [8] es una aplicación web debido a su versatilidad y portabilidad, ya que una página web responsive puede ser ejecutada tanto en dispositivos móviles como computadoras de escritorio sin necesidad de desarrollar una aplicación nativa para cada sistema operativo.

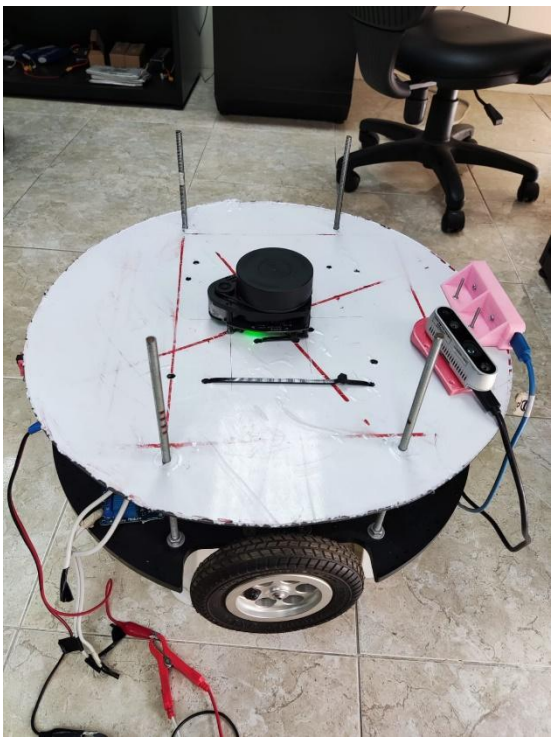

Figura 2.1 Robot sin módulo de desinfección [Autoría propia]

Varias herramientas se pueden utilizar para desarrollar interfaces gráficas web que conecten con ROS. Una de ellas es Roslibjs, la principal librería de JavaScript para interactuar con ROS desde un navegador que permite crear funciones para enviar instrucciones a distintos módulos del robot. Otra

herramienta necesaria es Rosbridge_suite, la cual nos permite una capa de abstracción, tratando a todo ROS como backend y, de esta manera, se puede abstraer las capacidades individuales de las partes del robot y manejar partes del robot por medio de mensajes.

Se definió que se desarrollaría una interfaz gráfica de usuario web la cual permite al usuario navegar el robot mediante botones que al ser presionados ejecutan comandos de navegación de ROS. La aplicación web contará con diseño responsive para que se pueda adaptar pantallas con distintos tamaños y de esta manera pueda ser utilizada en dispositivos como laptops y tabletas por igual.

El movimiento del robot funciona enviando un mensaje al topic cmd_vel. Estos mensajes modifican valores de aceleración en 2 vectores, uno lineal y otro angular, que al girar las ruedas direccionales y acelerando las ruedas motrices permiten la navegación del robot.

- Valores lineales: son utilizados para medir la aceleración lineal hacia adelante o hacia atrás.
- Valores angulares: sirven en conjunto con los lineales cuando se desea que el robot realice movimientos rotatorios o que gire hacia la izquierda o derecha.

Tabla 2.1 Vectores para desplazamiento en ROS [Autoría propia]

Vector de velocidad Lineal	Vector de velocidad angular
Vector3 linear	Vector3 angular
float64 x	float64 x
float64 y	float64 y
float64 z	float64 z

Valores positivos indican un movimiento hacia adelante, valores negativos indican movimiento hacia atrás y la magnitud de los valores determina la magnitud de la aceleración que tendrán las ruedas del robot.

2.2. Herramientas de software:

A continuación, se describirán las herramientas de software que serán utilizadas para diseñar la interfaz web, enlazar la aplicación al robot y crear simulaciones y cómo se realizarán estos procesos.

- **ROS Melodic:** Es la distribución más actual de ROS, un middleware que servirá para traducir los comandos en acciones del robot como el movimiento de las ruedas motrices, ruedas directrices, el accionamiento de la lámpara UV, etc.
- **Rosbridge:** Rosbridge nos permite realizar la conexión entre la aplicación web y el robot, además de traducir acciones como hacer clic en botones en pantalla a comandos en ROS.
- **Gazebo:** Es un simulador tridimensional para robótica que integra el motor de física Open Dynamics Engine, renderizado mediante OpenGL y que tiene soporte para simulación de sensores y control de actuadores.
- **Gazebo_ros_pkgs**: Es un conjunto de paquetes de ROS que provee de interfaces para realizar simulaciones del robot en Gazebo.
- **Robot Web Tools:** Las herramientas de Robot Web Tools, específicamente roslibjs, nos permiten interactuar con ROS desde el navegador y así controlar las acciones del robot desde la aplicación web.
- **Cartographer:** Es un sistema que provee de localización y mapeado en tiempo real en 2D Y 3D, esencial para el mapeo del área donde se desplazará el robot.

2.3. Requerimientos funcionales

A continuación, se realizará un análisis de los requisitos funcionales y no funcionales de la interfaz gráfica de usuario, además de presentar un

diagrama de casos de uso donde se especifican las acciones que el usuario podrá realizar dentro de la aplicación.

Requisitos funcionales:

1. La aplicación debe permitir el desplazamiento del robot hacia adelante.
2. La aplicación debe permitir el desplazamiento del robot hacia atrás.
3. La aplicación debe permitir la rotación de las ruedas direccionales del robot hacia ambas direcciones, izquierda y derecha.
4. La aplicación debe permitir encender la lámpara UV.
5. La aplicación debe permitir apagar la lámpara UV.

2.4. Prerrequisitos de funcionamiento

Para el correcto funcionamiento y pruebas del robot se deben cumplir los siguientes requisitos:

- Los espacios en los que debe navegar el robot deben ser cerrados.
- La batería del robot debe tener carga suficiente.
- El robot debe tener espacio suficiente para desplazarse (mayor a 2 metros cuadrados).
- Se debe tener acceso a internet.

2.5. Plan de implementación

Para implementar una interfaz gráfica, se definió un plan consistente en 4 fases: Recrear el entorno ROS con el robot, implementar la Interfaz Gráfica de Usuario, conectar la GUI con ROS y ejecución de pruebas experimentales.

2.5.1. Fase 1: Recreación del entorno ROS con el robot

Para crear una interfaz gráfica de usuario que permita controlar al robot primero es necesario tener una comprensión acerca de la forma en la que se controla el robot. Los requisitos actuales para conectarse con el robot y enviarle comandos son: una laptop con sistema operativo Ubuntu y un

router con conexión a internet de alta velocidad. La laptop necesita tener instalado todo el paquete de ROS, Rosbridge_suite y Husarnet.

Para realizar la conexión con el robot se debe crear una red que puede ser proveniente de un router o de un celular en modo punto de acceso móvil, pero es recomendado utilizar un router que tenga una frecuencia de 5Ghz.

De esta manera, el usuario puede conectarse a la microcomputadora Jetson Nano que se encuentra dentro del robot, la cual es la responsable de computar los comandos que se envíen y de convertirlos en acciones de los distintos componentes del robot.

Para visualizar la configuración y el estado de varios topics y mensajes en el robot se utiliza la herramienta Rviz, que sirve para visualizar todos los componentes del robot y permite realizar pruebas y detectar si existe comportamiento extraño en alguno de los parámetros de los nodos del robot.

Al ejecutar Rviz, se puede visualizar un mapa dibujado por los sensores localizados en el robot, específicamente el lidar y las cámaras de profundidad. Gracias a la visualización en Rviz también se puede determinar la posición del robot respecto al mapa trazado y si los sensores funcionan y dibujan correctamente el posicionamiento de las ruedas, la lámpara y otros componentes del robot.

2.5.2. Fase 2: Implementar la Interfaz Gráfica de Usuario

Para desarrollar la aplicación web primero se debe crear un servidor http basado en Python 3, para esto se configura un entorno virtual. Se debe crear una pantalla que dé al usuario la opción de conectarse con el robot, para que el usuario pueda elegir cuándo iniciar la conexión y para elegir a

que robot se va a conectar, en caso de que haya varios robots conectados a la misma red.

Dentro de una carpeta se debe crear un archivo HTML, el cual servirá para conectar nuestra aplicación web al robot y en este archivo incluiremos todas las funciones para publicar topics, crear nodos y enviar mensajes al robot. Los nodos en ROS son procesos que realizan alguna tarea y trabajan independientemente de otros, pero se comunican con otros nodos del sistema. Los topics en ROS proveen de información a otros nodos y los mensajes son valores de datos que los nodos publican.

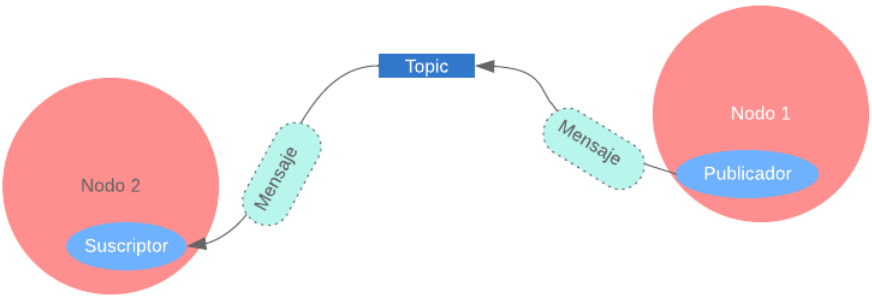

Figura 2.2 Publicación de mensajes en ROS [Autoría propia]

Para desarrollar la interfaz, se hará uso del framework Vue para desarrollo más eficiente con JavaScript y de Bootstrap para importar estilos a nuestra interfaz. La interfaz presentará al usuario varios botones, cada uno de estos botones publicará un mensaje en un topic para cumplir una función. Esta solución se asemejará a un 'D-pad' que permitirá, mediante el accionamiento de los botones de movimiento hacia adelante, atrás, y en ambas direcciones, publicar mensajes que serán atendidos por los nodos de ruedas presentes en el robot.

Se creará un archivo JavaScript en el cual se instancian las funciones que servirán para enlazar el accionamiento de botones de la interfaz gráfica con acciones del robot.

Para controlar los movimientos del robot, se creará una página en HTML en la cual se mostrará al usuario un grupo de botones con una distribución similar a la de un control remoto de un dron con 4 botones para controlar el desplazamiento del robot en distintas direcciones, un botón para detener su movimiento, un botón para encender la lámpara de desinfección y un botón para apagarla.

2.5.3. Fase 3: Conectar la Interfaz Gráfica de Usuario con ROS

La conexión entre la interfaz gráfica y el entorno ROS del robot consiste en traducir las acciones del usuario en funciones que realizará las partes del robot. Para realizar esto, se utilizará JavaScript y la librería Roslibjs para representar distintas funciones del robot como funciones de JavaScript.
El accionamiento de los botones de la interfaz gráfica será enlazado con eventos, estos eventos enviarán instrucciones al robot para que desempeñe las acciones requeridas por el usuario.

Las instrucciones que se van a enlazar son:

- Adelante: publica el mensaje para cambiar la aceleración en el eje x a +1.
- Atrás: publica el mensaje para cambiar la aceleración del eje x a -1.
- Detenerse: publica el mensaje para cambiar la aceleración del eje x a 0.
- Giro: publica el mensaje para cambiar la aceleración linear del eje x a +0.5 y aceleración angular del eje z a +0.3 o -0.3 dependiendo de si va a girar a la izquierda o a la derecha.
- Iniciar desinfección: publica el mensaje para encender la lámpara de luz ultravioleta.
- Acabar desinfección: publica el mensaje para apagar la lámpara de luz ultravioleta.

El componente que permitirá la conexión entre ROS y la GUI es el paquete rosbridge_suite. Este paquete contiene rosbridge, paquetes de frontEnd para rosbridge como un paquete de WebSockets y paquetes de ayuda.

También se utilizará Catkin, un paquete incluido en ROS con la función de crear un espacio de trabajo.

2.5.4. Fase 4: Pruebas funcionales y simulación

Finalmente, una vez implementada la interfaz gráfica de usuario, se deben realizar pruebas funcionales con el simulador y pruebas funcionales con el robot. En caso de haber algún inconveniente con el robot, el simulador es una representación realista de toda la funcionalidad que tiene el robot, por lo que el funcionamiento correcto de la interfaz con el simulador es equivalente a un funcionamiento correcto con el robot.

El programa utilizado para simular el robot se llama Gazebo, este nos otorga un entorno completo con todas las funciones que presenta el robot, una representación gráfica del mismo completa con texturas y un mapa en el cual el robot va a desplazarse.

El entorno 3d que se utilizará en Gazebo para pruebas es el encontrado en la Figura 1.

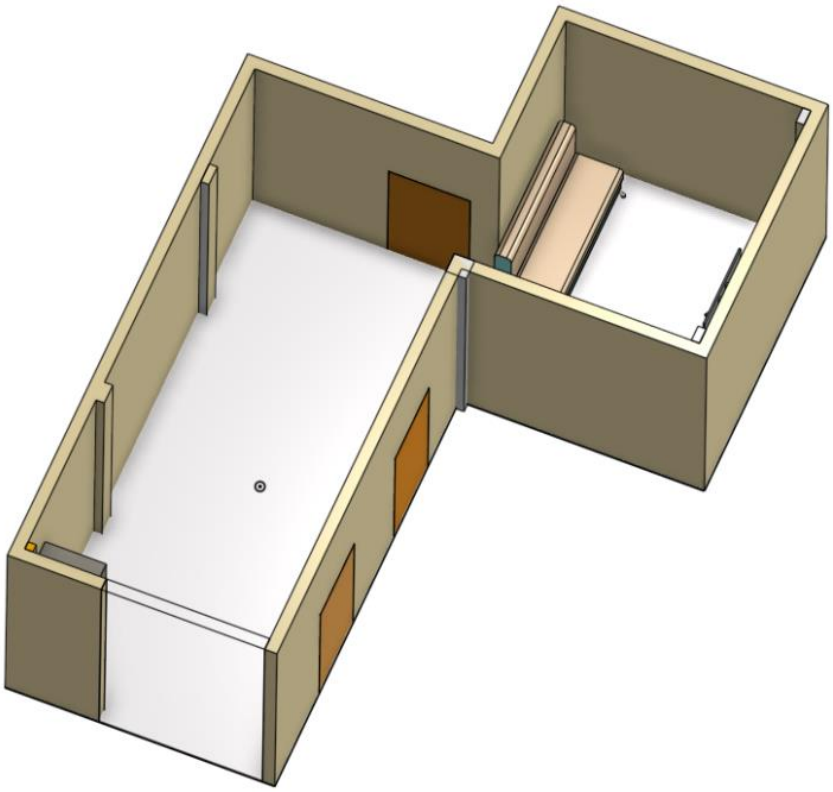

Figura 2.3 Entorno para la simulación. [Autoría propia]

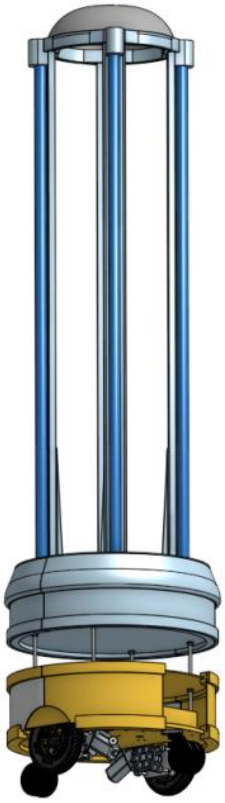

Figura 2.4 Diseño del robot. [Autoría propia]

El diseño del robot utilizado para ingresar en la simulación es el encontrado en la figura 4. El entorno y el diseño fueron provistos por el tutor.

Con Gazebo, crearemos una simulación en la cual el modelo mostrado en la figura 4 se insertará en el entorno de la figura 3, y tendrá las capacidades de moverse y de accionar la lámpara de desinfección, permitiendo así probar la aplicación web.

Se utilizará Cartographer para realizar un mapeo en simulación, utilizando como entorno el indicado anteriormente. En la prueba de simulación se utilizará la aplicación web desarrollada para probar todas las funciones del robot. Después de realizar la simulación en el robot, se procederá a realizar una prueba con el robot físico.

2.5.5. Limitaciones de la solución

La solución elegida, una aplicación web, presenta múltiples ventajas sobre otras opciones como aplicaciones móviles y aplicaciones de escritorio que fueron mencionadas anteriormente, sin embargo, también presenta sus propias limitaciones, tales como:

- Latencia:

Al conectarse la aplicación al robot por medio de una red, el tiempo que demoran los paquetes en ser enviados desde el dispositivo máster al robot y viceversa siempre existirá, por lo que no se podrá enviar comandos al robot en tiempo real. Siempre habrá un retraso entre la publicación de mensajes y la recepción de dichos mensajes por el robot.

- Tiempo de instalación:

Debido a la gran cantidad de plugins y librerías utilizadas, el tiempo de instalación del software puede ser de varios minutos, dependiendo de

factores como la velocidad de internet y de procesamiento del sistema donde sea instalado.

2.6. Cronograma de actividades

Tabla 2.2 Cronograma de actividades. [Autoría propia]

Actividades	Fecha de Inicio	Fecha de finalización
Fase 1: recreación del ambiente en ROS.	01/07/2022	03/07/2022
Fase 1: Pruebas del ambiente y robot.	04/07/2022	16/07/2022
Fase 2: Creación del servidor http.	17/07/2022	20/07/2022
Fase 2: Desarrollo de la interfaz gráfica de usuario	21/07/2022	28/07/2022
Fase 3: Creación de las funciones de movimiento	29/07/2022	06/08/2022
Fase 3: Conexión de GUI con ROS.	07/08/2022	12/08/2022
Fase 4: Pruebas experimentales con simulación	13/08/2022	17/08/2022
Fase 4: Pruebas experimentales con robot físico	18/08/2022	20/08/2022
Entrega Final	21/08/2022	24/08/2022

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

Se implementó un entorno web y una interfaz gráfica de usuario para permitir el control remoto del movimiento y activación de un robot de desinfección utilizando conocimientos sobre desarrollo de aplicaciones web, desarrollo de entornos web en Python y herramientas de ROS. El desarrollo de la aplicación web fue realizado en 4 fases detalladas anteriormente en el capítulo 2. Debido a problemas técnicos en el funcionamiento del robot que se presentaron antes del desarrollo de la aplicación, se acordó con el cliente que la aplicación sea dirigida al manejo de la simulación del robot, como se muestra en el apéndice A.

3.1. Resultados de la fase 1: Recreación del entorno ROS con el robot

Para recrear el entorno ROS primero se realizó la instalación de ROS y de varios paquetes necesarios para el correcto funcionamiento de la simulación, los cuales fueron extraídos del repositorio. Este repositorio fue clonado dentro del directorio home del computador.

El entorno de simulación fue ejecutado utilizando Gazebo y Rviz. Con la ayuda de Rviz se puede realizar el proceso de mapeo, para así generar un mapa del entorno y facilitar la navegación autónoma entre otras funciones del robot.

Para ejecutar el mapeo en simulación se siguió una serie de pasos de configuración:

- **Mapeo en simulación**

Antes de generar el mapa se compilo los archivos del simulador utilizando catkin, y para generar el mapa se hace uso de Cartographer, mediante el comando:

roslaunch arlobot_uv_launcher nav_demo.launch.

Una vez ejecutado el comando, para manejar el robot desde la terminal utilizando botones del teclado se hizo uso del aplicativo turtlebot3_teleop de ROS. Este es uno de los 2 métodos para poder navegar el robot o la simulación.

El otro método utilizado para navegar el robot en simulación fue mediante el uso del topic echo, el cual en conjunto con la funcionalidad de 2d nav goal de Rviz permite manejar el robot a sitios específicos que son señalados dentro de Rviz.
Una vez recreado y probado el entorno ROS, se comenzó el desarrollo de la interfaz gráfica de usuario.

3.2. Resultados de la fase 2: Implementación de la Interfaz Gráfica de Usuario.

Para implementar la GUI, primero se implementó un servidor http haciendo uso de la herramienta de Python SimpleHttpServer. Este servidor cumple el propósito de alojar la aplicación web localmente para que de esta manera se pueda acceder a la misma utilizando un navegador y conectándose al local host. El puerto local utilizado para alojar la aplicación es el 7000. El servidor se ejecutó en un directorio nuevo creado dentro del directorio del repositorio clonado anteriormente.

Se creó dentro del directorio nuevo un archivo HTML que servirá para mostrar los módulos de la aplicación. Los módulos que se desarrollaron dentro de la aplicación son:

1. Módulo de conexión.
2. Módulo de joystick.
3. Módulo de valores de joystick.
4. Módulo de activación de desinfección.
5. Módulo de cámara.

Figura 3.1 Aplicación en pantalla horizontal [Autoría propia]

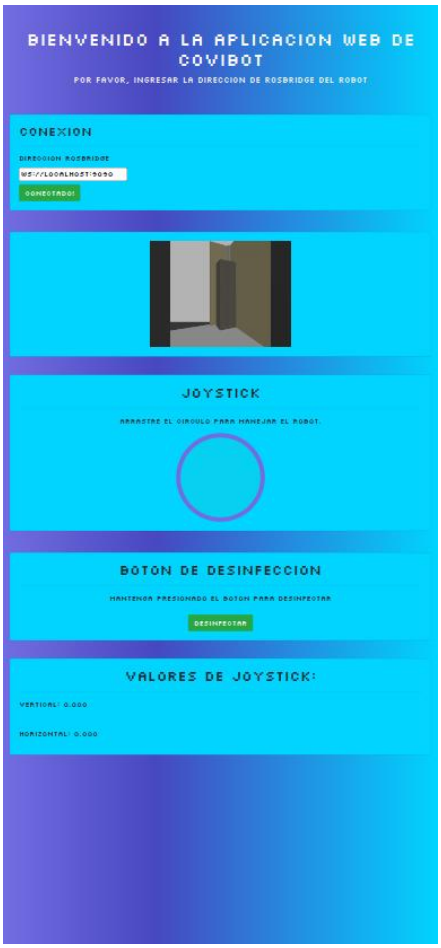

Figura 3.2 Aplicación en pantalla vertical [Autoría propia]

Para determinar el comportamiento de los módulos de la aplicación, se creó un archivo JavaScript y se hizo uso del framework Vue para construir la

interfaz web. Para darle diseño a la interfaz, se hizo uso de la librería Bootstrap y el lenguaje Css.

3.2.1. Módulo de conexión

El módulo de conexión consiste en un contenedor con 2 elementos: un input y un botón.

El input es donde el usuario ingresará la dirección de WebSocket del robot al que desee conectarse, y será por defecto la dirección local en la que fue montada la simulación anteriormente. El botón, al ser presionado, indicará si fue exitosa la conexión con la dirección indicada y de ser así iniciará la conexión entre Rosbridge y la interfaz gráfica.

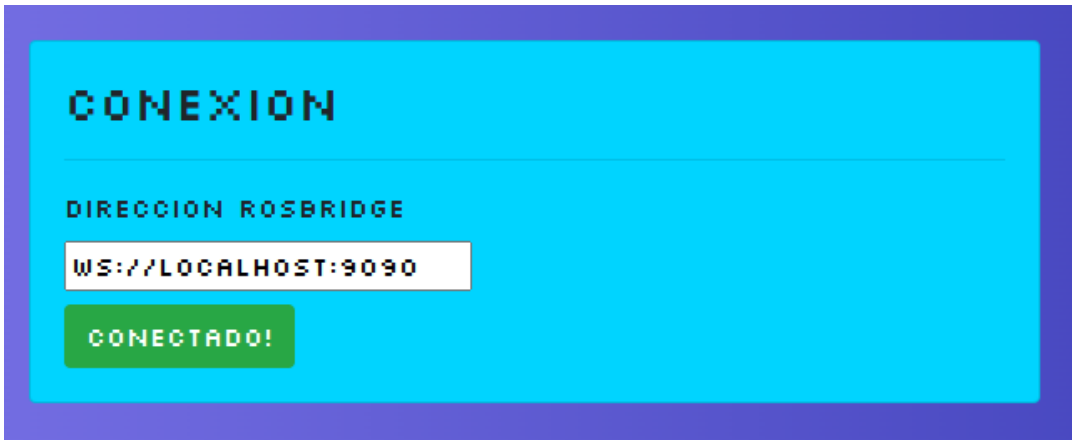

Figura 3.3 Módulo de conexión [Autoría propia]

3.2.2. Módulo de joystick

El módulo de joystick sirve para dar control del movimiento del robot simulado al usuario. En este módulo se encuentra un círculo, cuando el usuario mantiene su cursor sobre este, mantiene clic y arrastra el nuevo círculo interior en distintas direcciones la simulación del robot se moverá en la dirección indicada. Este módulo funciona enviando a los topics de las ruedas del robot valores variables de dirección y aceleración. De esta manera, al arrastrar el joystick el robot se podrá mover con velocidad constante en la misma dirección determinada.

El joystick se maneja con 2 ejes: El eje X se extiende horizontalmente por el círculo del joystick y el eje Y se extiende verticalmente, formando un

plano. Cuando el usuario arrastra el joystick el robot rotará acorde a la dirección en la cual se arrastra.

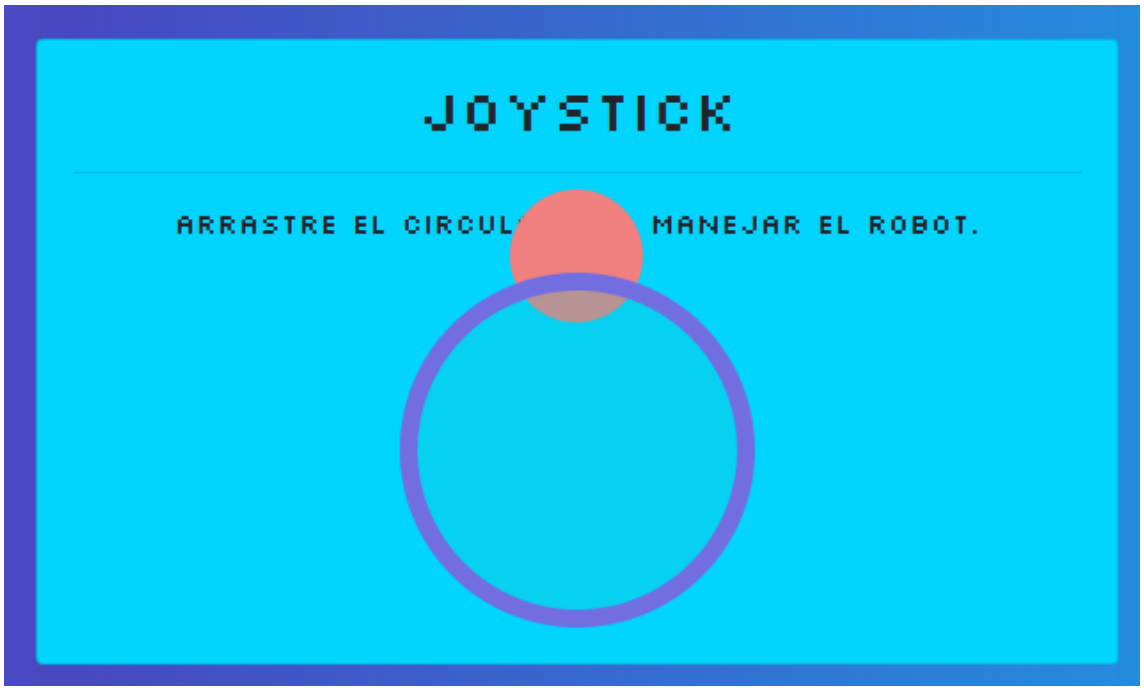

Figura 3.4 Módulo de joystick [Autoría propia]

3.2.3. Módulo de valores de joystick

En este módulo se muestran los valores de ejes vertical y horizontal que se están enviando al robot actualmente. El propósito de este módulo es dar retroalimentación al usuario de los valores que están siendo enviados y que así, en caso de haber un malfuncionamiento del robot, se pueda captar inmediatamente y tomar los pasos debidos para evitar accidentes.

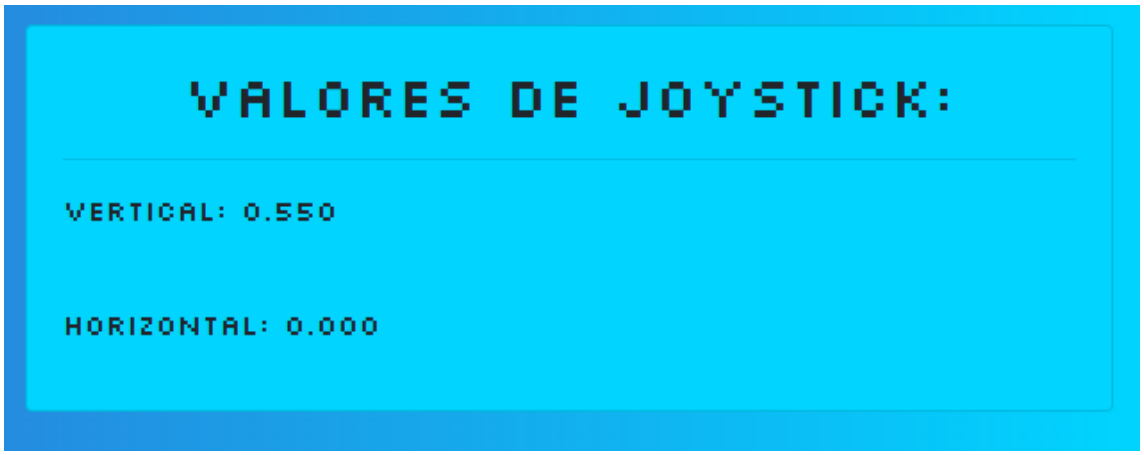

Figura 3.5 Módulo de valores de joystick [Autoría propia]

3.2.4. Módulo de desinfección

El módulo de desinfección contiene un botón para iniciar la desinfección al encender las lámparas de luz ultravioleta encontradas en la parte superior del robot. La desinfección se simboliza con el cambio en la coloración de la cámara mientras el botón se mantenga presionado.

Figura 3.6 Módulo de desinfección [Autoría propia]

3.2.5. Módulo de cámara

El módulo de cámara consiste en una vista que permite al usuario ver lo que se encuentra directamente en frente del robot. De este modo, el usuario puede navegar el robot sin necesidad de observar la simulación y también puede observar la superficie a la cual se le hará la desinfección.

Cuando el botón de desinfección este activado, los colores de la cámara cambiaran a negativo, para demostrar que se está realizando la desinfección.

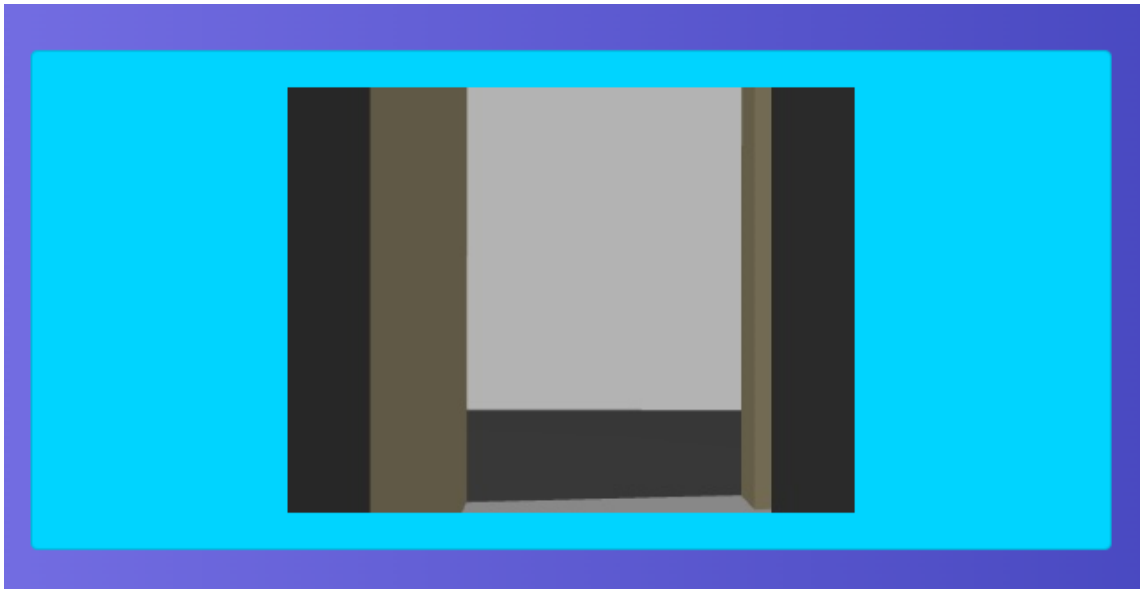

Figura 3.7 Módulo de cámara [Autoría propia]

Después de la implementación de la GUI, se procedió a realizar la siguiente fase, en la cual se conectó la GUI con ROS dándole funcionalidad a todos los módulos mediante la utilización del framework Vue, JavaScript y el paquete Roslibjs.

3.3. Resultados de la fase 3: Conexión de la Interfaz Gráfica de Usuario con ROS.

Se creó varios métodos en el archivo JavaScript para enviar mensajes a los topics en ROS. A continuación, se describirá los métodos asignados para cada uno de los módulos de la aplicación.

3.3.1. Métodos: Módulo de conexión

Al ingresar la dirección de Rosbridge en el input y presionar el botón de conectar, se inicia la conexión utilizando, creando un nuevo objeto Ros y se establece su atributo url como la dirección de Rosbridge ingresada.
En caso de que la conexión sea exitosa se mostrará conectado y se procederá a activar los demás módulos.

3.3.2. Métodos: Módulo de joystick

El módulo de joystick utiliza 2 valores, vertical y horizontal, que cambiarán según la posición del círculo interior relativa al centro del círculo exterior y estos valores se utilizan para modificar el mensaje enviado al topic /cmd_vel, que se encarga de la rotación, aceleración angular y linear de las ruedas del robot.

Mover el círculo interior hacia la derecha o izquierda del centro del círculo exterior hace que el robot gire hacia la derecha o izquierda respectivamente y mover el círculo interior hacia arriba o hacia abajo hace que el robot se mueva hacia adelante y hacia atrás respectivamente.

3.3.3. Métodos: Módulo de valores de joystick

En este módulo se muestra en valores del -1 al 1 tanto el valor de la aceleración vertical que corresponde al movimiento hacia adelante en caso de ser positivo o negativo en caso de ser hacia atrás y la aceleración horizontal que crea una rotación hacia la izquierda en caso de ser negativa o hacia la derecha en caso de ser positiva.

3.3.4. Métodos: Módulo de Desinfección

El módulo de desinfección consta de un botón que va a servir para cambiar la imagen mostrada en la cámara, para representar que la desinfección está en progreso. La desinfección se mantendrá en curso mientras el botón sea presionado.

3.3.5. Métodos: Módulo de Cámara

Para mostrar la retroalimentación de la cámara del robot se utilizó uno de los paquetes de ros, web_video_server, que provee de una transmisión de video de un topic de ROS que transporta video, en este caso la cámara que se encuentra transmitiendo la visión delantera del robot. Está transmisión de video se encuentra alojada en el puerto local 8080 y se accede a ella creando una función que rellenará un canvas del paquete Mjpegcanvas e

insertando este canvas en el módulo. El módulo de cámara solo está disponible cuando se establece una conexión exitosa con el websocket y con el servidor web de video.

3.4. Resultados de la fase 4: Pruebas funcionales y simulación

Luego de montar la simulación y la aplicación web se procedió a realizar pruebas funcionales de los distintos módulos de la aplicación. Las pruebas fueron exitosas, esto es, todos los módulos funcionan de manera correcta y la aplicación se adapta a distintos tamaños de pantalla sin perder funcionalidad.

Los resultados de las pruebas de los distintos módulos se presentan a continuación:

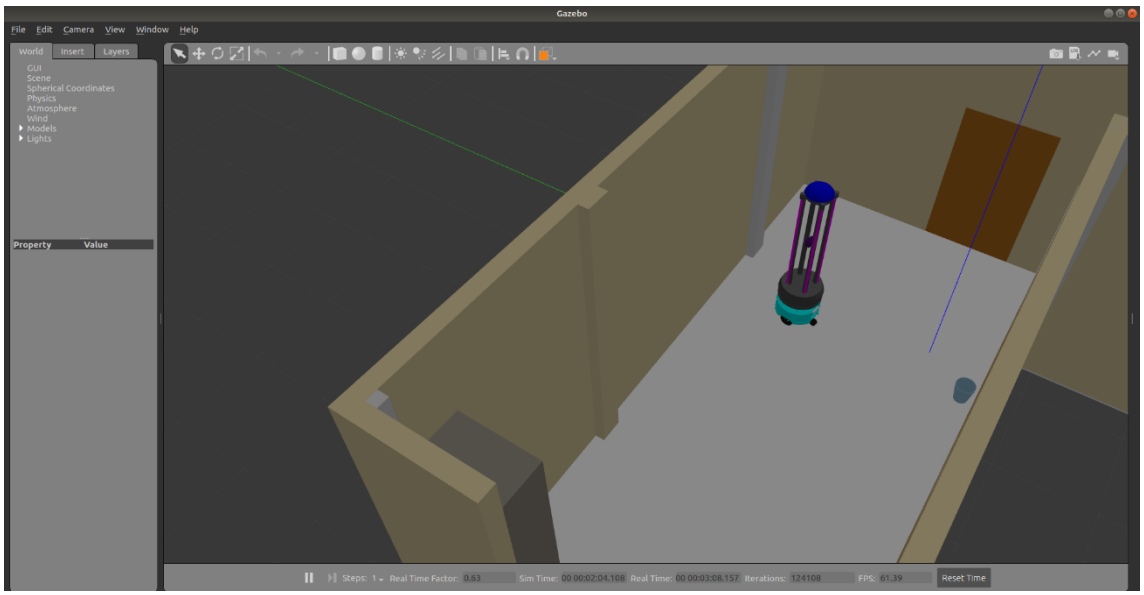

Figura 3.8 Robot desplazado en Gazebo [Autoría propia]

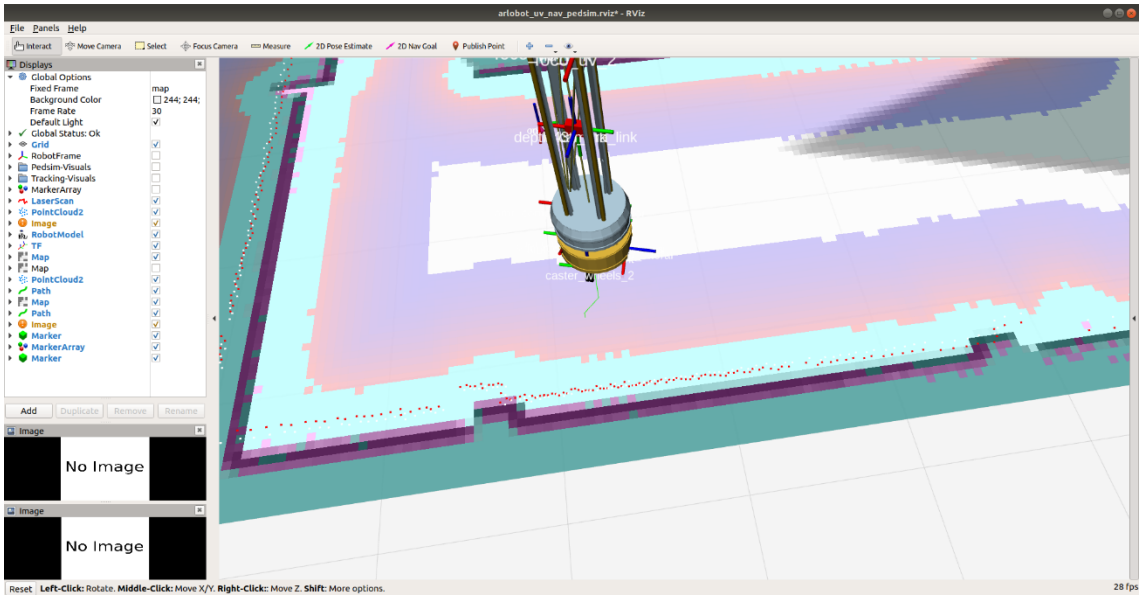

Figura 3.9 Robot desplazado en Rviz [Autoría propia]

- El módulo de conexión representa correctamente la conexión con ROS.
- El módulo de cámara es capaz de transmitir la imagen de la cámara simulada constantemente.
- El movimiento del robot no presenta latencia considerable cuando se mueve el joystick.
- Los valores presentados por el módulo de valores de joystick son correctos.
- El módulo de desinfección cambia correctamente el color de la cámara mientras se mantiene presionado, representando correctamente la desinfección.

De esta manera, se cumplen todos los requerimientos antes del desarrollo del proyecto y se procede a realizar el análisis de costos de este.

3.5. Análisis de costos

El desarrollo de este proyecto no tuvo costo alguno ya que fue realizado en su totalidad por un estudiante de la carrera de Ingeniería en Computación. En caso de que el proyecto fuese realizado por un ingeniero en

computación, se puede calcular el costo del proyecto en base a las semanas de trabajo necesarias para el desarrollo de la aplicación.

A continuación, se detalla las semanas utilizadas para cada fase de desarrollo del proyecto:

Tabla 3.1 Semanas por fase de desarrollo [Autoría propia]

Fase de desarrollo	Semanas de desarrollo
1. Recreación del entorno ROS con el robot	2
2. Implementación de la Interfaz Gráfica de Usuario.	3
3. Conexión de la Interfaz Gráfica de Usuario con ROS.	3
4. Pruebas funcionales y simulación	2

Esto nos da un total de 10 semanas. Tomando como referencia que el sueldo mensual promedio de un ingeniero en computación es de USD 1,500, se calcula que la ganancia semanal de un ingeniero sería de USD 375, por lo que el costo aproximado del proyecto sería de USD 3,750.

3.6. Cierre

Una vez implementada y probada la interfaz, se presentó al cliente el entregable. Este entregable es la integración de todos los cambios realizados incluyendo la interfaz y todos los archivos necesarios para el desarrollo de esta en el repositorio principal. Esto se realizó mediante un commit en el repositorio en Gitlab.

Además, se añadió un tutorial para instalar todas las dependencias y pasos para montar el entorno web, el entorno ROS e ingresar a la interfaz web. El acta de cierre firmada por el cliente se encuentra en la sección de anexos.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se concluyó a partir de múltiples pruebas que implementar un Joystick brinda un control más preciso de los movimientos y el frenado del robot que crear 4 botones para moverlo hacia adelante, atrás, izquierda y derecha, por lo que se decidió utilizar el joystick como método de navegación.

- Se concluyó que la implementación del módulo de cámara es acertada ya que permite que el usuario pueda navegar el robot sin observar la simulación, y provee de suficiente información para no estrellar ni atascar el robot.

- Se concluyó que la implementación del módulo de valores de joystick es acertada debido a que en caso de algún malfuncionamiento de la interfaz se puede observar esta para revisar si se está enviando algún valor erróneo.

- Se concluyó que la implementación del módulo de conexión es útil no solo para utilizar la GUI con el robot de desinfección, sino también para poder navegar robots similares basados en ROS.

4.2. Recomendaciones

- Se recomienda, en caso de implementarse nuevos módulos en la aplicación, la creación de nuevas páginas para alojar estos y así mantener la claridad visual.

- Se recomienda que, al implementar la aplicación para el manejo del robot real, se cree un módulo de desinfección que active las lámparas de desinfección ya que la función de desinfección en la simulación es distinta a la del robot real.

- Al desempeñar la acción de desinfección en el robot real, se recomienda mantener una distancia prudente con el robot y utilizar equipo protector porque los rayos de luz ultravioleta pueden resultar dañinos para la salud.

- En caso de utilizarse la aplicación para navegar el robot a través de la web, se recomienda tener internet de alta velocidad debido a que esto ayuda a disminuir la latencia entre la interacción con la interfaz y las acciones realizadas por el robot.

- Se recomienda tener conocimientos básicos de física si se desea implementar módulos que interactúen con las ruedas debido a que se hace uso de conceptos como la aceleración angular y la aceleración lineal.

BIBLIOGRAFÍA

[1] P. Vasickova, I. Pavlik, M. Verani and A. Carducci, "Issues Concerning Survival of Viruses on Surfaces," *Food Environ Virol,* vol. 2, no. 1, 4 Feb 2010.

[2] F. Chiappa, B. Frascella, G. P. Vigezzi, M. Moro, L. Diamanti, L. Gentile, P. Lago, N. Clementi, C. Signorelli, N. Mancini and A. Odone, "The efficacy of ultraviolet light-emitting technology against coronaviruses: a systematic review," *Journal of Hospital Infection,* vol. 114, 2021.

[3] "Ubuntu," [Online]. Available: https://ubuntu.com/community/debian. [Accessed 24 Jun 2022].

[4] "Python.org," [Online]. Available: https://www.python.org/doc/essays/blurb/.

[5] Python, "Python documentation," [Online]. Available: https://docs.python.org/2/library/simplehttpserver.html.

[6] Global, Bosch, "Bosch Global," 23 Jun 2022. [Online]. Available: https://www.bosch.com/stories/bringing-robotics-middleware-onto-tiny-microcontrollers/.

[7] Vue, "Vue.js," 2022. [Online]. Available: https://es.vuejs.org/v2/guide/.

[8] V. Mazzari, "Generation Robots - Blog," 06 Jul 2021. [Online]. Available: https://www.generationrobots.com/blog/en/robotic-simulation-scenarios-with-gazebo-and-ros.

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

[10] J. M. Rapado, "Diseño e implementación de una interfaz gráfica de usuario para mapeado de entornos y navegación en ROS," p. 7, 2016.

[11] "Clearpath Robotics," 21 Jun 2022. [Online]. Available: https://clearpathrobotics.com/turtlebot-4/.

[12] "Ros.org," [Online]. Available: http://wiki.ros.org/rosbridge_server.

[13] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills and S. Chernova, "Robot Web Tools: Efficient messaging for cloud robotics," *RSJ International Conference on Intelligent Robots and Systems (IROS),* 2015.

[14] B. Lutkevich, "SearchNetworking," 04 Oct 2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/TCP.

[15] K. Sookocheff, "Kevin Sookocheff," Abr 04 2019. [Online]. Available: https://sookocheff.com/post/networking/how-do-websockets-work/.

[16] "Code Institute Global," 13 Ene 2022. [Online]. Available: https://codeinstitute.net/global/blog/what-is-a-rest-api/.

APÉNDICES

APÉNDICE A

Acta de reunión

Temas tratados/acuerdos		
1. Aceptación de cambio de enfoque del proyecto. La interfaz desarrollada solamente será destinada a la navegación del robot en simulación.		
Firma		
Cargo	**Nombre**	**Firma**
Tutor/Cliente	Dennys Paillacho	 Firmado electrónicamente por: DENNYS FABIAN PAILLACHO CHILUIZA

APÉNDICE B

Acta de reunión

Temas tratados/acuerdos		
1. Aceptación de entregables del proyecto		
2. Presentación de producto final		
Firma		
Cargo	**Nombre**	**Firma**
Tutor/Cliente	Dennys Paillacho	 Firmado electrónicamente por: DENNYS FABIAN PAILLACHO CHILUIZA