

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CARRERA DE INGENIERÍA MECATRÓNICA




**OPTIMIZACIÓN DE TRAYECTORIA DE UN ROBOT
MÓVIL OMNIDIRECCIONAL SEMIAUTÓNOMO
PARA LA DESINFECCIÓN DE AMBIENTES CON EL
USO DE RADIACIÓN UV TIPO C**

TESIS

Para optar el título profesional de Ingeniera Mecatrónica

AUTORA:

Maria del Martin Vera Tudela Bustamante 

ASESORA:

Ruth Vanessa Canahuire Cabello 

Lima - Perú

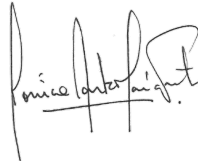
2023

DECLARACIÓN JURADA

Yo, Mónica Cecilia Santa María Fuster identificada con DNI No 18226712 en mi condición de autoridad responsable de validar la autenticidad de los trabajos de investigación y tesis de la UNIVERSIDAD DE INGENIERIA Y TECNOLOGIA, DECLARO BAJO JURAMENTO:

Que la tesis denominada “OPTIMIZACIÓN DE TRAYECTORIA DE UN ROBOT MÓVIL OMNIDIRECCIONAL SEMIAUTÓNOMO PARA LA DESINFECCIÓN DE AMBIENTES CON EL USO DE RADIACIÓN UV TIPO C” ha sido elaborada por la señorita Maria del Martin Vera Tudela Bustamante, con la asesoría de Ruth Vanessa Canahuire Cabello, identificada con DNI 42141373, y que se presenta para obtener el título profesional de Ingeniero mecatrónico, ha sido sometida a los mecanismos de control y sanciones anti plagio previstos en la normativa interna de la universidad, encontrándose un porcentaje de similitud de 1%.

En fe de lo cual firmo la presente.



Dra. Mónica Santa María Fuster
Directora de Investigación

En Barranco, el 16 de enero de 2024

Dedicatoria:

A Ernesto y Maria, mis padres.

Agradecimientos:

A Dios, por iluminar mi camino.

A mi asesora, Ruth Canahuire, por guiarme en todo momento.

A mis profesores, por brindarme un mundo de conocimientos.

A mis padres y amigos, por su constante soporte y motivación.

Índice general

	Pág.
RESUMEN	1
ABSTRACT	2
CAPÍTULO 1 INTRODUCCIÓN	3
1.1 Problemática	3
1.2 Objetivos	4
1.2.1 Objetivo general	4
1.2.2 Objetivos específicos	4
1.3 Justificación	5
1.4 Alcance y Limitaciones	5
1.5 Organización del Trabajo	6
CAPÍTULO 2 REVISIÓN CRÍTICA DE LITERATURA	7
2.1 Estado del Arte	7
2.2 Antecedentes	8
CAPÍTULO 3 MARCO TEÓRICO	11
3.1 Desinfección con radiación UV-C	11
3.1.1 Detección de presencia de Sars-CoV-2 en superficies	12
3.2 Robots móviles con ruedas	13
3.2.1 Robot no holonómico	14
3.2.2 Robot holonómico	14

3.3	Modelo cinemático de robots móviles holonómicos	15
3.3.1	Cinemática directa	18
3.3.2	Cinemática inversa	18
3.4	Control de velocidad del robot móvil	19
3.4.1	Método de control LQG	21
3.5	Localización del robot móvil	25
3.6	Planificación de Movimiento	27
3.7	Criterio de optimización	28
3.7.1	Programación lineal	30
CAPÍTULO 4 METODOLOGÍA		35
4.1	Diseño mecánico del robot desinfectante	35
4.2	Selección de waypoints	37
4.3	Localización del robot móvil	39
4.4	Generación de trayectoria	40
4.5	Cinemática y Control LQG	43
4.6	Análisis de dosis UV-C por colorimetría	44
CAPÍTULO 5 RESULTADOS		45
5.1	Resultados simulados	45
5.1.1	Selección de waypoints	45
5.1.2	Cinemática del robot móvil omnidireccional	48
5.1.3	Control de velocidad LQG	49
5.1.4	Simulación de navegación semiautónoma	56
5.2	Resultados experimentales	60
5.2.1	Control de velocidad LQG	60
5.2.2	Navegación semiautónoma	61
5.2.3	Análisis colorimétrico	65
CONCLUSIONES		67

ANEXOS 75

Índice de tablas

3.1	Valores RGB de cada dosis del dosímetro de UVC STERILIZER X™. . .	13
3.2	Solución con el método Dijkstra al grafo presentado en la Fig. 3.9 (a). . .	28
3.3	Tabla símplex	31
5.1	Dimensiones del mapa real de trabajo.	48
5.2	Parámetros del robot.	48
5.3	Parámetros del motor DC	49
5.4	Resultados del controlador LQG.	60
5.5	Resultados del análisis colorimétrico	66

Índice de figuras

1.1	Cuadro de programación de actividades.	6
3.1	Representación gráfica de un robot móvil diferencial [32].	14
3.2	Ruedas de robots móviles omnidireccionales [33]. (a) Rueda <i>swedish</i> . (b) Rueda mecanum.	15
3.3	Posición del robot móvil en el plano [34].	16
3.4	Ángulo de inclinación del <i>roller</i> en una rueda omnidireccional, γ	17
3.5	Rueda omnidireccional y sus parámetros [34].	17
3.6	Diagrama mecánico-eléctrico del motor DC.	20
3.7	Diagrama de bloques del control óptimo LQG con efecto integral.	24
3.8	Algoritmo AMCL basada en puntos de referencia [43].	26
3.9	Algoritmo Dijkstra. (a) Ejemplo de grafo. (b) Ejemplo de grafo resuelto.	28
3.10	Trayectorias del robot simuladas de un robot mediante DWA [46]	29
4.1	Diagrama de bloques de la metodología empleada en este trabajo.	36
4.2	Diseño mecánico del robot desinfectante. (a) En Autodesk Inventor. (b) Implementado.	37
4.3	Esquema del control de trayectoria LQG.	44
5.1	Waypoints de un laberinto de 5 metros por 5 metros.	46
5.2	Dimensiones del mapa real de trabajo. Ver Tabla 5.1	47
5.3	Waypoints en el mapa real de trabajo.	47
5.4	Parámetros de restricción y distribución geométrica del robot móvil omnidireccional.	49

5.5	Velocidad de las ruedas del robot aplicando el controlador LQG.	52
5.6	Tiempo de establecimiento y error en estado estable del controlador con respuesta al escalón.	53
5.7	Mapa de polos y ceros del controlador LQG diseñado.	53
5.8	Velocidad del robot aplicando el controlador LQG y cinemática directa. .	54
5.9	Esfuerzo del control LQG en cada rueda del robot.	55
5.10	Simulación de la navegación semiautónoma (a) El robot parte del primer waypoint hasta el segundo waypoint. (b) El robot se encuentra en el se- gundo waypoint. (c) El robot se encuentra en el tercer waypoint. (d) El robot se encuentra en el cuarto waypoint. (e) El robot se encuentra en el quinto waypoint. (f) El robot llega al último waypoint.	57
5.11	Velocidades del robot móvil. (a) Control de velocidad en el eje x . (b) Con- trol de velocidad lineal en el eje y . (c) Control de velocidad angular θ . . .	62
5.12	Trayectoria deseada VS. trayectoria obtenida sin considerar el peso del robot móvil.	63
5.13	Trayectoria deseada VS. trayectoria obtenida considerando el peso del ro- bot móvil.	63
5.14	Diagrama de caja del error de posición (a) en x . (b) en y	64
5.15	Ubicación de dosímetros en el ambiente de trabajo	65

RESUMEN

La desinfección de ambientes, bajo el contexto del COVID-19, resulta importante para contener la propagación del virus Sars-CoV-2. Diversas investigaciones han sido desarrolladas con la finalidad de asegurar la desinfección de superficies mediante insumos químicos o máquinas de desinfección que salvaguarden la salud de las personas. En ese sentido, la optimización de trayectoria es una metodología que tiene como objetivo principal generar una trayectoria definida por una serie de *waypoints* que asegure la evasión de obstáculos y garantice la ruta óptima según diversos objetivos, razón por la cual resulta atractiva su aplicación en la desinfección de ambientes debido a las necesidades de su comportamiento.

Este trabajo comprende la optimización de trayectoria de desinfección de un robot móvil omnidireccional semiautónomo para maximizar la dosis UV-C suministrada a las superficies para asegurar la inactivación del virus Sars-CoV-2 en un 99.9 %. Para la navegación del robot móvil, se pre-carga un mapa donde el robot pueda navegar, su localización se obtiene por odometría, la generación de trayectoria se obtiene con el algoritmo de Dijkstra y, para lograr inactivar el virus en las superficies, se halló los *waypoints* mediante programación lineal utilizando el modelo de irradiación. La validación de desinfección se realizó mediante el análisis off-line de la dosis suministrada por las lámparas UV-C, ubicadas en los *waypoints*, a las superficies con marcadores colorimétricos sensibles a los rayos UV-C situados en el sótano 1 de la Universidad de Ingeniería y Tecnología (UTEC).

Palabras clave:

UV-C; Robótica exploratoria; COVID-19; Optimización de trayectoria; Programación lineal

ABSTRACT

**OPTIMIZATION OF THE PATH OF A
SEMI-AUTONOMOUS OMNIDIRECTIONAL
MOBILE ROBOT FOR THE DISINFECTION OF
INDOOR ENVIRONMENTS WITH UV-C RADIATION**

Disinfection of environments, in the context of COVID-19, is important to contain the spread of the Sars-CoV-2 virus. Several researches have been developed in order to ensure the disinfection of surfaces by means of chemical inputs or disinfection machines that safeguard people's health. In this sense, trajectory optimization is a methodology whose main objective is to generate a trajectory defined by a series of waypoints that ensures the avoidance of obstacles and guarantees the optimal route according to different objectives, which is why its application in the disinfection of environments is attractive due to the needs of its behavior.

This work comprises the optimization of the disinfection trajectory of a semi-autonomous omnidirectional mobile robot to maximize the UV-C dose delivered to the surfaces to ensure 99.9 % inactivation of the Sars-CoV-2 virus. For the navigation of the mobile robot, a map is pre-loaded where the robot can navigate, its location is obtained by odometry, trajectory generation is obtained with Dijkstra's algorithm and, to achieve inactivation of the virus on the surfaces, waypoints were found by linear programming using the irradiation model. Disinfection validation was performed by off-line analysis of the dose delivered by UV-C lamps, located at the waypoints, to surfaces with UV-C sensitive colorimetric markers located in basement 1 of the University of Engineering and Technology (UTEC).

Keywords:

UV-C; Exploratory robot; COVID-19; Trajectory optimization, Linear programming

Capítulo 1

INTRODUCCIÓN

En este primer capítulo se presenta la motivación de la presente tesis, la cual consiste en la optimización de trayectoria de desinfección de un robot móvil para la desinfección de ambientes interiores con radiación ultravioleta tipo C (UV-C). Asimismo, se presentarán trabajos recientes de la robótica móvil aplicada para la inactivación del virus Sars-CoV-2 con UV-C Posteriormente, se plantearán los objetivos que tendrá este trabajo. Finalmente, se establecerán los alcances y las limitaciones que se tendrán en cuenta.

1.1 Problemática

La propagación del virus Sars-CoV-2 ha llevado a la Organización Mundial de la Salud (OMS) a declarar la pandemia por el COVID-19 en el 2020. Dicha pandemia ha tenido un impacto significativo en los centros de trabajo y de salud a nivel mundial, obligándolos a encontrar técnicas efectivas para la desinfección de áreas de alto y mediano tránsito. Por tal motivo, la OMS, en conjunto con la Organización Internacional del Trabajo (OIT), recomienda a todos los centros de trabajo desarrollar y emplear programas que garanticen la salud de los trabajadores y prevengan la infección por Sars-CoV-2, pues ellos se encuentran más expuestos a este nuevo flagelo [1].

Por otro lado, se ha evidenciado que las condiciones climáticas influyen en la propagación del COVID-19. Las altas temperaturas, la alta humedad específica, la alta radiación ultravioleta y los fuertes vientos pueden inactivar el virus Sars-CoV-2. Además, según estudios, existen 3 herramientas fundamentales que se deben tomar en cuenta para

frenar la propagación de la enfermedad: vacunas, cuarentena focalizada y correcta desinfección de ambientes [2].

En los últimos años, se ha utilizado la radiación UV-C como alternativa de desinfección de ambientes, cuyo rango de longitud de onda está comprendido entre 200 a 280 nm, para lograr un efecto germicida [3]. Esta tecnología ha cobrado gran notoriedad en el sector salud, pues la utilización de sustancias químicas germicidas para la desinfección de superficies no solo inactiva los microorganismos, sino también causa afecciones en la salud de las personas debido a que la sustancia permanece en el ambiente dejando a las personas expuestas causando inmunodepresión [4].

1.2 Objetivos

1.2.1 Objetivo general

El objetivo general de esta tesis es optimizar la trayectoria de desinfección de un robot móvil omnidireccional usando programación lineal considerando la geometría del ambiente para la dosificación de UV-C con el fin de desinfectar las superficies.

1.2.2 Objetivos específicos

1. Seleccionar la instrumentación electrónica incluyendo el sistema de desinfección por UV-C y diseñar el sistema mecánico del robot móvil en Autodesk Inventor.
2. Modelar y simular la cinemática, el control de trayectoria del robot móvil y la optimización de los puntos de desinfección en Matlab.
3. Generar la trayectoria de desinfección optimizada del robot móvil usando programación lineal en función del área geométrica de la región a explorar en ROS.

4. Implementar el mecanismo para la desinfección y el control de trayectoria optimizada del robot.
5. Analizar la desinfección de las superficies a través de colorimetría.

1.3 Justificación

Este trabajo es importante porque utiliza la tecnología UV-C como medio que permitirá controlar y reducir la propagación del virus Sars-CoV-2 mediante la desinfección óptima de superficies contaminadas en menos tiempo [5] en comparación con la esterización por calor que puede causar daños materiales o el uso de desinfectantes químicos que pueden sufrir escasez en el mercado [6] y causar inmunodepresión en las personas [4], por lo cual su aplicación resulta idónea en zonas de alta afluencia de personas. Por otro lado, salvaguarda la integridad y salud de las personas al no tener contacto directo con la radiación UV-C, la cual puede causar cáncer de piel [4]. Además, permite el desarrollo de sistemas de desinfección con tecnología propia.

1.4 Alcance y Limitaciones

Este trabajo se centra en la optimización de la trayectoria, más no en la optimización de la instrumentación electrónica o mecánica del mismo. El sistema robótico móvil omnidireccional se limita al desarrollo de un prototipo funcional que demuestre su funcionalidad para desinfectar las superficies con UV-C de manera semiautónoma. De igual manera, el sistema se desplaza sobre una superficie plana y se considera una superficie lisa. El sistema robótico fue probado en el sótano de la Universidad de Ingeniería y Tecnología (UTEC) debido a que el entorno de trabajo debe ser estático y las superficies deben tener una misma altura. Finalmente, la investigación culmina con la validación de desinfección mediante el análisis colorimétrico de dosímetros que indican la dosis suministrada

a las superficies por la radiación UV-C. El objetivo es inactivar el virus Sars-CoV-2 en un 99.9 %. Para dicha inactivación se utilizaron 3 lámparas UV-C.

1.5 Organización del Trabajo

Para el desarrollo de la presente tesis se organizaron las actividades a realizar, tal como se puede observar en la Fig. 1.1.

Nombre de tarea	Duración	Comienzo	Fin
Cronograma	102 días	9/6/2021	12/17/2021
Inicio	0 días	9/6/2021	9/6/2021
Diseño del robot móvil	19 días	9/6/2021	9/25/21
Selección de la instrumentación	7 días	9/6/2021	9/13/2021
Diseño del robot móvil completo	5 días	9/13/2021	9/18/21
Adquisición de materiales	7 días	9/18/21	9/25/21
Simulación de movimiento del robot	62 días	9/25/21	11/26/21
URDF del robot móvil	12 días	9/25/21	10/7/21
Algoritmo de selección de waypoints	14 días	10/7/21	10/21/21
Cinemática de robot móvil omnidireccional	6 días	10/21/21	10/27/21
Control de trayectoria LQG	9 días	10/27/21	11/5/21
Generación de trayectoria en Gazebo	21 días	11/5/21	11/26/21
Implementación del sistema	21 días	11/26/21	12/17/21
Implementación del control en Atmega 328P (Arduino UNO)	9 días	11/26/21	12/5/21
Implementación de la generación de trayectoria en Raspberry Pi3	12 días	12/5/21	12/17/21
Fin	0 días	12/17/21	12/17/21

FIGURA 1.1: Cuadro de programación de actividades.

Capítulo 2

REVISIÓN CRÍTICA DE LITERATURA

2.1 Estado del Arte

En el año 2020, la compañía UVD Robots®, especializada en el desarrollo de nuevas tecnologías robóticas en conjunto con herramientas germicidas UV-C para la erradicación de patógenos dañinos, lanzó la tercera generación de robots de desinfección UV-C. Este robot móvil diferencial se mueve de forma autónoma en un área previamente definida después de que el usuario haya ingresado los parámetros de tiempo de exposición y distancia de superficies, además cuenta con 8 lámparas UV-C. Para permitir el movimiento autónomo, el robot debe estar preprogramado con un mapa detallado de la posición de los objetos en el área a tratar con radiación UV-C al igual que los puntos donde el robot parará por 3 minutos durante el ciclo de desinfección. Una vez que se configuran todos los parámetros, y la ubicación de los objetos, estos deben permanecer exactamente en el mismo lugar para permitir un funcionamiento autónomo [7].

En [8], los autores presentaron un robot móvil diferencial autónomo equipado con 8 lámparas UV-C alrededor de una columna central y 2 lámparas UV-C ubicadas en la parte superior. El funcionamiento del robot es teleoperado y se da a través de una aplicación que puede ser vista desde un teléfono o una tablet mediante comunicación Wi-Fi [8]. Sin embargo, existe una falta de garantías sobre la desinfección de todas las superficies [9]. Eso se debe a que ninguno de los robots presentados anteriormente cuenta con pruebas que validen su poder de desinfección sobre la totalidad de las superficies del área donde el robot se encuentre [10]. En [11], analizaron la desinfección de superficies con un robot UV-C. Para corroborar la desinfección utilizaron puntos colorimétricos, los cuales obtienen la dosis acumulada suministrada, y el 33 % de estos no llegó a ser desinfectado.

En el año 2023, un estudio realizado por investigadores de la Universidad de Pisa evaluó la efectividad de desinfección en la reducción de la contaminación microbiana ambiental cuando se aplica como complemento del protocolo operativo estándar de desinfección en zonas críticas con alto riesgo de transmisión de infecciones sanitarias del robot de desinfección *Hyper Light UV-C* desarrollado por la corporación empresarial asiática Mediland. El funcionamiento de dicho robot consiste en el posicionamiento manual por un operador, quien puede determinar el tiempo de desinfección por cada punto de parada [12]. Según Mediland, al tener una tecnología patentada de reflector protector, el tiempo necesario de desinfección por cada punto de parada se encuentra en un intervalo de 5 a 15 minutos [13]. El protocolo operativo estándar de desinfección que se realizaba en el hospital donde se ejecutó dicho estudio consistía en la aplicación de un detergente a base de cloro, Antisapril Detergente 10 %, Angelini, seguido de un desinfectante a base de cloro en las superficies de los muebles y los aparatos electromédicos, posterior al protocolo operativo estándar, el robot UV-C fue empleado en ciclos de 5 minutos. El estudio demostró que de la muestra de 160 puntos de desinfección, solo con el protocolo operativo estándar el 64.3 % dieron positivo, mientras que solo el 17.5 % dieron positivo después de su exposición a la radiación UV-C. En ese sentido, la adición de la desinfección con el robot UV-C al procedimiento estándar de desinfección tuvo resultados eficaces en la reducción de las fallas de higiene [12].

2.2 Antecedentes

Los patógenos, incluidos los virus, pueden propagarse y transmitirse por vías ambientales, como el aire y las superficies inertes, o indirectamente al tocar una superficie contaminada [14]. La Organización Mundial de la Salud (OMS) insta a todas las personas a prevenir la propagación de la infección mediante el lavado regular de las manos y el uso de máscaras faciales o cualquier otra barrera física contra la transmisión [15].

Asimismo, se recomienda evitar o mantener distancia en zonas concurridas, como colegios, universidades, restaurantes, teatros y cines [16]. Sin embargo, la eficacia de estas acciones preventivas es limitada, sobre todo en ambientes interiores donde el aire circulante biocontaminado o las superficies que se tocan con frecuencia pueden mediar en la transmisión.

La tecnología de sanitización con luz ultravioleta UV-C ha tomado notoriedad debido a que se descubrió que las superficies de las habitaciones se limpiaban de manera inapropiada mediante métodos manuales tradicionales de desinfección. A pesar de la realización de capacitaciones para lograr la mejora de la técnica de limpieza para la correcta desinfección de superficies, los efectos fueron limitados [17]. A longitudes de onda particulares, como 254 nm, la luz UV-C es capaz de destruir los enlaces moleculares y romper el ADN o ARN, lo cual provoca la muerte de una variedad de microorganismos presentes en el ambiente [18]. Para efectos de la inactivación del virus SARS-CoV-2, en [19] se determinó que la dosis de UV-C requerida para la inactivación del 99.9% de un microorganismo, corresponde a 3.7 mJ/cm^2 y con una dosis de 16.9 mJ/cm^2 para lograr una completa inactivación del virus.

En 1940, el ingeniero sanitario Wells probó la funcionalidad de la luz UV-C para desinfectar matando microorganismos presentes en el aire mediante lámparas UV-C colocadas en la parte superior de las paredes de un salón de clases para combatir la propagación de sarampión [20]. En [21], se comparó la eficacia de un robot móvil para desinfectar el *Staphylococcus aureus* resistente a la meticilina (SARM) con UV-C que opera en posición estacionaria o móvil. Este estudio evidenció que el dispositivo que opera en una posición estacionaria fue significativamente menos efectivo que el dispositivo móvil debido a que la desinfección no llegaba a distancias mayores. En consecuencia, se evidenció que el robot UV-C autónomo es eficaz para reducir la contaminación en los sitios de la habitación.

Un estudio realizado por la Universidad de Bihac demuestra y describe cómo cada vez más robots móviles con ruedas de servicio desinfectante están contribuyendo a una desinfección muy rápida, sencilla y eficaz en las instituciones médicas, pues reduce el costo de limpieza y desinfección, y el riesgo de infecciones asociadas a la atención médica [22]. La generación de movimiento para un robot móvil necesita de un control de trayectoria adecuado. En [23], se desarrolló y construyó un robot móvil direccional en el cual se implementó un controlador lineal cuadrático (LQR, por sus siglas en inglés) para controlar la posición de 2 motores DC. El resultado obtenido reflejó que el control LQR implementado en el robot móvil diferencial contaba con un error nominal de 0.144 m.

En [24], se desarrolló y construyó un robot móvil omnidireccional en el cual también se implementó un control LQR para controlar, en este caso, la posición de 4 motores DC. El resultado obtenido reflejó que el control LQR implementado en el robot móvil omnidireccional contaba con un error nominal de 0.03 m. Comparando los resultados de ambos estudios, el robot móvil omnidireccional presenta un error menor que el del robot móvil diferencial. Además, en el estudio del robot móvil omnidireccional se implementó un control Proporcional – Integral (PI) para controlar la posición de los 4 motores DC. El resultado obtenido reflejó que el control PI implementado en el robot contaba con un error nominal de 0.11 m. Comparando el control PI con el control LQR implementado en el robot móvil omnidireccional, el control LQR tiene una mejor performance.

Capítulo 3

MARCO TEÓRICO

En este capítulo se presentarán las técnicas para validar la inactivación del virus Sars-CoV-2 y se mostrará el modelo cinemático del robot móvil. Luego, se presenta el modelo matemático de la planta y se muestra la representación en espacio de estados. Seguidamente, se expone el algoritmo de control LQG utilizado para el control de trayectoria del robot móvil y algoritmo utilizado para la generación de trayectoria. Finalmente, se introducirá el algoritmo de optimización utilizado para obtener la mejor trayectoria de desinfección posible.

3.1 Desinfección con radiación UV-C

Dentro del espectro electromagnético, que muestra las ondas electromagnéticas clasificadas por longitud de onda, frecuencia y temperatura, la luz ultravioleta se encuentra entre los rayos X y la luz visible y está entre un rango de longitud de onda de 100 nm a 400 nm [25]. La luz ultravioleta es utilizada como un agente físico desinfectante, el cual inactiva patógenos. Con inactivación, se hace referencia a la incapacidad que se genera en los microorganismos de reproducirse y, en consecuencia, causar enfermedades. La diferencia entre radiación UV e irradiación UV consiste en que este último término hace referencia al proceso de desinfección en el que se utiliza como agente desinfectante la radiación ultravioleta [26].

Dentro de la radiación electromagnética ultravioleta, existe una segunda clasificación por longitud de onda: onda larga (UV-A), onda media (UV-B) y onda corta (UV-C). La radiación UV de onda corta (UV-C) es la que contiene el rango de longitud de onda

considerada germicida, entre 220 nm y 320 nm. Siendo más específicos, para aplicaciones de inactivación de patógenos, longitudes de onda UV entre 254 nm y 265 nm son más efectivas [25].

La eficacia de la desinfección con radiación ultravioleta depende de la dosis suministrada. Esto quiere decir que la intensidad de radiación ultravioleta multiplicada por el tiempo en el que los microorganismos están expuestos determina el resultado de la desinfección. La ecuación de la dosis está dada de la siguiente manera [26]:

$$D = I_{media}t \quad (3.1)$$

donde D es la dosis de radiación ultravioleta suministrada en mJ/cm^2 , I_{media} es la intensidad de radiación ultravioleta media en mW/cm^2 y t es el tiempo de exposición en segundos.

3.1.1 Detección de presencia de Sars-CoV-2 en superficies

Una vez realizada la desinfección, es fundamental detectar la presencia de Sars-CoV-2 a través del monitoreo del aire y las superficies para garantizar la desinfección y, de esta manera, evitar la propagación de la enfermedad [27]. Existen 2 formas de detectar su presencia: directa e indirectamente. El método más común para detectar de forma directa la presencia del virus Sars-CoV-2 es mediante el muestreo de superficies para luego trasladar dicha muestra a un laboratorio donde se extrae y amplifica el ARN viral mediante el método de reacción en cadena de la polimerasa (RPC, por sus siglas en inglés) [28]. El método indirecto es detectando la dosis de UV-C recibida por la superficie mediante dosímetros electrónicos o dosímetros cuyo indicador es por colorimetría, donde el color cambia dependiendo a la dosis detectada. En este caso, la dosis que se debe detectar para obtener un 99.9 % de inactivación del virus Sars-CoV-2 es de $3.7 mJ/cm^2$ [19].

Color	RGB			Dosis (mJ/cm ²)
	Valor de R	Valor de G	Valor de B	
	255	180	225	25
	255	117	208	50
	255	82	180	75
	254	60	156	100
	243	15	141	200
	236	3	135	300

TABLA 3.1: Valores RGB de cada dosis del dosímetro de UVC STERILIZER X™.

En diversas investigaciones sobre desinfección de superficies, contaminadas con el virus Sars-CoV-2, con radiación UV-C han validado dicha desinfección con dosímetros colorimétricos [11][12]. En el mercado, existen distintas marcas de dosímetros colorimétricos como por ejemplo CLEANLIFE®, Chemdye®, UVC dosimeter™ y UVC STERILIZER X™. Este último tiene la distribución de dosis por color mostrada en la Tab. 3.1.

3.2 Robots móviles con ruedas

Los robots móviles con ruedas representan la mayoría de los robots móviles usados. Normalmente están conformados por un sistema de ruedas que provee el movimiento con respecto al suelo y un cuerpo rígido (base). Otros cuerpos rígidos, también equipados con ruedas, pueden estar conectados a la base mediante uniones de revolución [29]. Los robots móviles con ruedas consumen menos energía y se mueven más rápido que otros mecanismos de locomoción. Desde el punto de vista de control, menos esfuerzo de control es requerido, debido a sus mecanismos simples y los reducidos problemas de estabilidad. Aunque es difícil superar las condiciones accidentadas del terreno o las condiciones desiguales del suelo, los robots móviles con ruedas son adecuados para una gran clase de entornos [30].



FIGURA 3.1: Representación gráfica de un robot móvil diferencial [32].

3.2.1 Robot no holonómico

Los robots no holonómicos son robots que están sujetos a una restricción de velocidad y, en consecuencia, su trayectoria no depende únicamente de su posición y orientación. Esto genera que el movimiento del robot sea limitado pues no se puede mover en cada dirección del espacio. Sin embargo, el robot puede alcanzar cualquier configuración en un plano libre de obstáculos como un automóvil [31]. Una de las más conocidas y simples configuraciones de un robot no holonómico es el robot móvil diferencial. Este tiene un modelo matemático simple y costo computacional bajo lo cual permite una fácil implementación para tareas específicas [31]. Un ejemplo de esto se puede observar en la Fig. 3.1.

3.2.2 Robot holonómico

Los robots holonómicos, también conocidos como robots móviles omnidireccionales, son robots que, a diferencia de los robots no holonómicos, no tienen restricciones

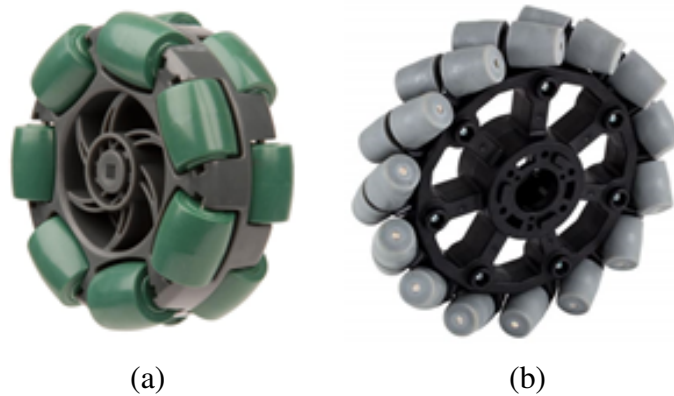


FIGURA 3.2: Ruedas de robots móviles omnidireccionales [33]. (a) Rueda *swedish*. (b) Rueda mecanum.

en la posición ni en la orientación con respecto a la velocidad. Esto genera que el movimiento del robot sea libre pues se puede mover en cualquier dirección del espacio sin haber rotado previamente [31].

Los robots móviles omnidireccionales suelen utilizar ruedas *swedish* o ruedas mecanum, estas se pueden observar en la Fig. 3.2. Las ruedas *swedish* poseen *rollers* en su circunferencia interior, los cuales giran libremente sobre los ejes en el plano de la rueda y tangencial a la circunferencia exterior de la rueda. Las ruedas mecanum son similares a las ruedas omnidireccionales, la diferencia radica en que los ejes de giro de los *rollers* no se encuentran en el plano de la rueda [31].

3.3 Modelo cinemático de robots móviles holonómicos

La posición del robot móvil en el plano (2D) es descrita en la Fig. 3.3, donde un sistema de referencia de base inercial arbitrario $\{b\}$ se fija en el plano de movimiento, mientras que un sistema de referencia $\{m\}$ está conectado al robot móvil.

En robótica móvil 2D la posición y la orientación se suelen representar en un solo vector respecto al sistema de referencia inercial $\{b\}$ [34]:

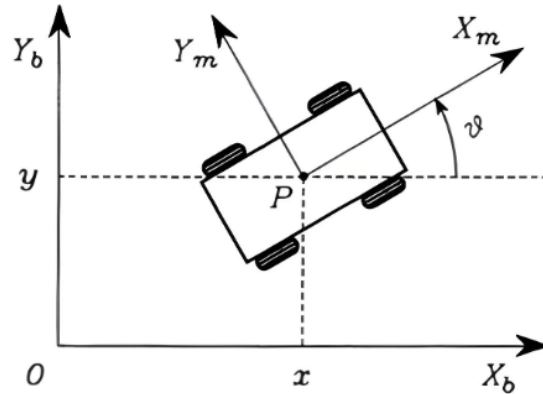


FIGURA 3.3: Posición del robot móvil en el plano [34].

$$\xi^b = \begin{bmatrix} x \\ y \\ \vartheta \end{bmatrix} \quad (3.2)$$

donde la posición es descrita por las coordenadas x e y , y ϑ representa la orientación del robot respecto al sistema de referencia inercial con origen en O . De tal forma, derivando el vector mostrado en la ecuación (3.2), se obtiene la velocidad del robot con respecto al sistema de referencia inercial $\{b\}$:

$$\dot{\xi}^b = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} \quad (3.3)$$

Antes de construir el modelo cinemático que represente al robot móvil, es necesario delimitar la restricción que las ruedas le proveen al movimiento del robot. En el caso de los robots holonómicos, las ruedas son omnidireccionales, las cuales poseen pequeñas ruedas, denominadas *rollers*, no actuadas e inclinadas en cierto ángulo γ [34]. Esto se puede notar en la Fig. 3.4.

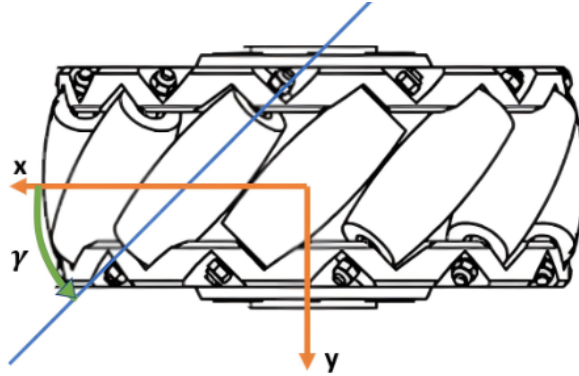


FIGURA 3.4: Ángulo de inclinación del *roller* en una rueda omnidireccional, γ .

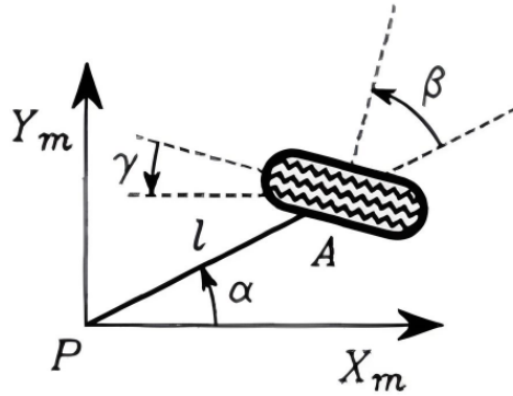


FIGURA 3.5: Rueda omnidireccional y sus parámetros [34].

Tomando en cuenta los parámetros que se pueden observar en la Fig. 3.5, la ecuación de restricción de rodadura de la rueda omnidireccional toma la siguiente forma [35]:

$$\begin{bmatrix} \sin(\alpha + \beta + \gamma) & -\cos(\alpha + \beta + \gamma) & -l\cos(\beta + \gamma) \end{bmatrix} \dot{\xi}^m = r\dot{\varphi}\cos(\gamma) \quad (3.4)$$

donde α es el ángulo formado entre el centro de giro y el centro de la rueda, β es el ángulo de rotación de la rueda, γ es el ángulo entre los *rollers* de la rueda y el eje de la rueda, l es la distancia entre el centro de giro y el centro de la rueda, r es el radio de la rueda, $\dot{\varphi}$ representa la velocidad angular de las ruedas.

El modelo cinemático se obtiene a partir de las restricciones que impone cada rueda. Las restricciones se pueden agrupar de forma matricial de la siguiente manera [35]:

$$A\dot{\xi}^m = B\dot{\varphi} \quad (3.5)$$

donde A está comprendida de filas que contienen las restricciones que las ruedas generan en el movimiento del robot, B contiene los radios de las ruedas y $\dot{\xi}^m$ contiene la velocidad del robot con respecto a su sistema de referencia. Luego de esto se puede aplicar cinemática directa o cinemática inversa.

3.3.1 Cinemática directa

La cinemática directa determina la velocidad del robot en función de la velocidad angular de las ruedas, resultando en la siguiente ecuación [35]:

$$\dot{\xi}^m = A^\# B\dot{\varphi} \quad (3.6)$$

donde $A^\#$ es la pseudo-inversa de Moore-Penrose que sirve para despejar la velocidad del robot con respecto a su sistema de referencia ($\dot{\xi}^m$) y se define de la siguiente manera:

$$A^\# = (A^T A)^{-1} A^T \quad (3.7)$$

3.3.2 Cinemática inversa

La cinemática inversa funciona de manera opuesta a la cinemática directa debido a que determina las velocidades angulares de las ruedas en función de la velocidad del robot, resultando en la siguiente ecuación [35]:

$$\dot{\varphi} = B^\# A\dot{\xi}^m \quad (3.8)$$

donde $B^\#$ es la pseudo-inversa de Moore-Penrose que sirve para despejar la velocidad angular de las ruedas ($\dot{\varphi}$) y se define de la siguiente manera:

$$B^\# = (B^T B)^{-1} B^T \quad (3.9)$$

3.4 Control de velocidad del robot móvil

El control de un robot móvil es una tarea importante en la robótica móvil. Lo que se busca es seguir una trayectoria descrita por la posición o velocidad en base a modelos matemáticos en función del tiempo. En el caso de los robots holonómicos, la velocidad es integrable [30]. Para implementar un controlador en un robot móvil es necesario primero obtener el modelo cinemático del robot. Este debe relacionar la velocidad lineal y angular del robot con la velocidad de las ruedas con respecto al sistema de referencia inercial [29].

La planta a controlar es la velocidad de cada motor DC, los cuales generan el movimiento de las ruedas del robot móvil omnidireccional. Para ello, el diagrama del motor DC [36] se muestra en la Fig. 3.6. El sistema puede modelarse mediante las siguientes ecuaciones diferenciales [37]:

$$J\ddot{\theta} + b\dot{\theta} = KI \quad (3.10)$$

$$L\frac{dI}{dt} + RI = V - K\dot{\theta} \quad (3.11)$$

donde θ es la posición del eje, I es la corriente de la armadura, V es la tensión de la fuente (es decir, la entrada del sistema), J es el momento de inercia del motor, b es el coeficiente de rozamiento viscoso, K es la constante contraelectromotriz, R es la resistencia eléctrica y L es la inductancia eléctrica.

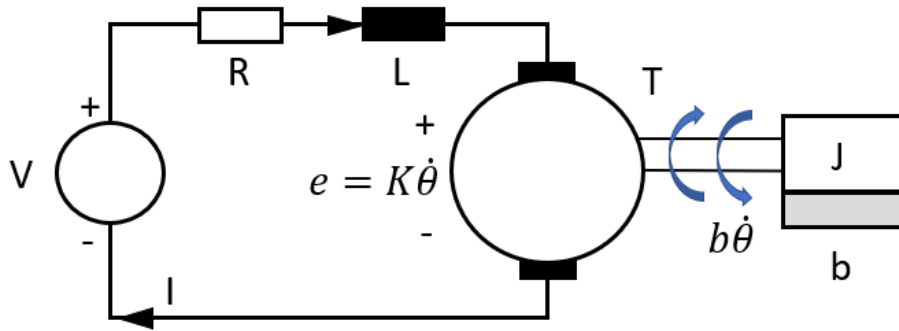


FIGURA 3.6: Diagrama mecánico-eléctrico del motor DC.

Escribiendo las ecuaciones (3.10) y (3.11) en la forma de espacio de estados ($\dot{x} = Ax + Bu$), se obtiene lo siguiente:

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} -\frac{b}{J} & -\frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_x + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}}_B \underbrace{V}_u \quad (3.12)$$

donde x_1 es $\dot{\theta}$, x_2 es I , x representa el vector de espacio de estados y u es el vector que representa el esfuerzo de control.

En ese sentido, la salida del sistema en espacio de estados ($y = Cx$) queda de la siguiente manera:

$$y = x_1 = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_C \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_x \quad (3.13)$$

3.4.1 Método de control LQG

El controlador LQG, *Linear Quadratic Gaussian* por sus siglas en inglés, se considera un método de control robusto porque para su diseño se consideran los ruidos de medición y las perturbaciones [38]. Dicho controlador se deriva del controlador LQR, *Linear Quadratic Regulator* por sus siglas en inglés, el cual considera que todas las variables de estado se encuentran disponibles para la retroalimentación. Sin embargo, cuando algunas o todas las variables de estado no se encuentran disponibles o no se pueden medir, resulta necesario utilizar un observador o estimador. En ese sentido, el controlador LQG resulta una combinación de un controlador LQR y un estimador de Kalman; es decir, un estimador lineal cuadrático.

Para asegurar que el sistema pueda ser controlado, debe ser completamente controlable. Un sistema es completamente controlable si existe una señal de control $u(t)$ sin restricciones que pueda transferir un estado inicial $x(t_0)$ a algún otro estado $x(t)$ en un intervalo de tiempo finito $t_0 \leq t \leq T$. Para comprobar la controlabilidad de sistemas continuos que tienen la representación de estados mostrada en la ecuación (3.12), el rango de la matriz de controlabilidad debe ser igual a la cantidad de estados del sistema (n) [39]:

$$\text{rank} \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} = n \quad (3.14)$$

Para estimar las variables de estado del sistema que no se pueden medir, el sistema debe ser completamente observable. Un sistema es completamente observable si y solo si existe un tiempo finito T de modo tal que el estado inicial $x(0)$ puede ser determinado a partir de la observación histórica $y(t)$ dado el control $u(t)$ donde $0 \leq t \leq T$. Para comprobar la observabilidad de sistemas continuos que tienen la representación de estados mostradas en las ecuaciones (3.12) y (3.13), el rango de la matriz de observabilidad debe

ser igual a la cantidad de estados del sistema (n) [39]:

$$\text{rank} \begin{bmatrix} C^* & A^*C^* & \dots & (A^*)^{n-1}C^* \end{bmatrix} = n \quad (3.15)$$

donde la conjugada compleja de una matriz se establece con el símbolo (*). Si las matrices no son complejas, entonces se reemplaza la operación de conjugada compleja (*) con la operación transpuesta (T).

El método de control LQR es una técnica eficiente y comúnmente utilizada para diseñar controladores para sistemas complejos. Dado el sistema representado en la ecuación (3.12), el control LQR busca encontrar una matriz de ganancia (K) para los estados de modo que la ley de control (3.17) minimice la siguiente función de costo [40]:

$$J = \int_0^{\infty} ((x_{ref}(t) - x(t))^T Q (x_{ref}(t) - x(t)) + u(t)^T R u(t)) dt \quad (3.16)$$

donde Q es una matriz definida o semidefinida positiva y R es una matriz definida positiva. Los valores de la matriz Q y R se escogen de acuerdo con los requerimientos del diseño.

$$u(t) = -Kx(t) \quad (3.17)$$

Reemplazando la ecuación (3.17) en la ecuación (3.12), se obtiene lo siguiente:

$$\dot{x} = (A - BK)x \quad (3.18)$$

Para asegurar el funcionamiento de la ley de control presentada en la ecuación (3.17), la matriz $A - BK$ debe ser estable; es decir, las raíces de la ecuación característica debe tener parte real negativa.

Derivando J respecto a $u(t)$, la ley de control queda de la siguiente manera:

$$u(t) = -R^{-1}B^T Px(t) \quad (3.19)$$

donde P una matriz simétrica, definida semipositiva, mayor a 0, y es la solución a la ecuación de Ricatti [40]:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \quad (3.20)$$

El filtro de Kalman es un algoritmo que proporciona estimaciones de algunas variables desconocidas dadas las medidas observadas a lo largo del tiempo. Primero se puede encontrar una señal óptima de estimación de estado $x(t)$, que minimiza la covarianza $E[(x - \hat{x})(x - \hat{x})^T]$, y luego utilizar la señal estimada $\hat{x}(t)$ para reemplazar las variables de estado reales de tal manera que el problema original se puede reducir a un problema de control óptimo LQ ordinario [41]. En ese sentido, el modelo dinámico del estimador de estados es el siguiente:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \quad (3.21)$$

donde L es la matriz de ganancia de retroalimentación del observador.

Como se mencionó a principios de esta sección, el controlador LQG es una integración del controlador LQR y el filtro de Kalman. De manera tal que su diseño consta de 2 pasos principales: determinar la ganancia K de la retroalimentación de los estados mediante la solución de la ecuación de Ricatti (LQR) y determinar la ganancia del observador L . Por otro lado, si la planta no tiene integradores, la salida del sistema debe ser integrada. En ese sentido, se tendrá la matriz de ganancia del integrador.

Tomando en cuenta lo presentado anteriormente, el diagrama de bloques del control óptimo LQG con efecto integral se muestra en la Fig. 3.7. De manera tal que cerrando

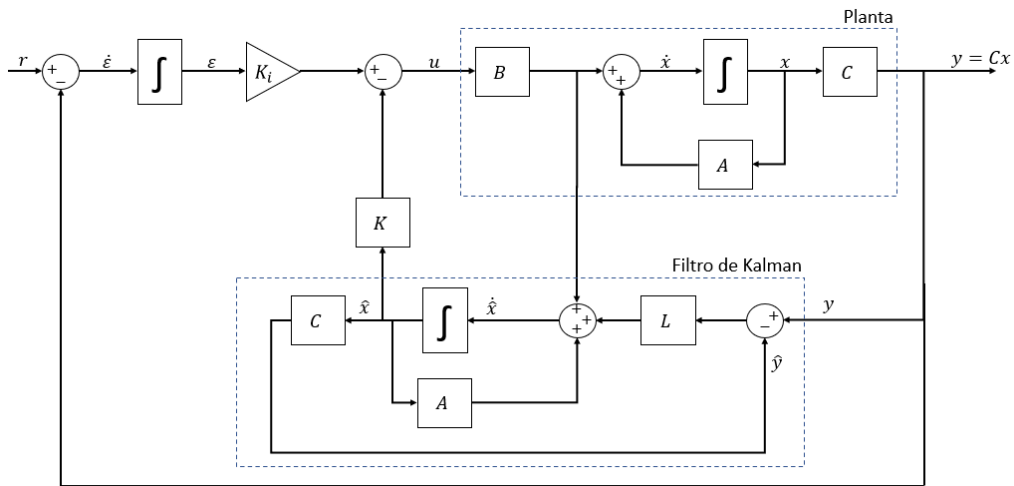


FIGURA 3.7: Diagrama de bloques del control óptimo LQG con efecto integral.

el lazo en las ecuaciones (3.12) y (3.13), se obtiene lo siguiente:

$$\dot{\epsilon} = r - Cx \quad (3.22)$$

$$\dot{x} = Ax + B(K_i\epsilon - K\hat{x}) \quad (3.23)$$

$$\dot{\hat{x}} = A\hat{x} + B(K_i\epsilon - K\hat{x}) + L(Cx - C\hat{x}) \quad (3.24)$$

Escribiendo las ecuaciones en la forma de espacio de estados, se obtiene lo siguiente:

$$\begin{bmatrix} \dot{x} \\ \dot{\epsilon} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{Bmatrix} x \\ \epsilon \end{Bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r \quad (3.25)$$

$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{Bmatrix} x \\ \epsilon \end{Bmatrix} \quad (3.26)$$

donde la ley de control puede escribirse de la siguiente manera:

$$u = -K\hat{x} + K_i\epsilon = - \begin{bmatrix} K & K_i \end{bmatrix} \begin{Bmatrix} \hat{x} \\ \epsilon \end{Bmatrix} \quad (3.27)$$

3.5 Localización del robot móvil

La localización del robot se puede realizar utilizando sensores propioceptivos como la unidad de medición inercial (IMU, por sus siglas en inglés) o la odometría mediante los encoders de los motores que accionan las ruedas. La odometría consiste en la estimación de la posición del robot mediante el movimiento de sus ruedas. Si el entorno es conocido desde un principio de la navegación, entonces se puede obtener la localización del robot con un sensor exteroceptivo como por ejemplo un sensor LiDAR e integrarlo con el cálculo odométrico para darle mayor precisión a la localización del robot móvil.

Existen dos tipos de localización: local y global. La localización local requiere que la ubicación inicial del robot sea aproximadamente conocida y normalmente no pueden recuperarse si pierden la pista de la posición del robot, mientras que la localización global puede localizar un robot sin conocimiento previo de su posición; es decir, pueden tratar la reubicación del robot llevándolo a un lugar desconocido. Las técnicas de localización global son más potentes que las locales y pueden hacer frente a situaciones en las que es probable que el robot experimente graves errores de posicionamiento [42].

El algoritmo de localización adaptativa de Monte Carlo (AMCL) es un algoritmo probabilístico que es aplicable tanto a problemas de localización local como global. Utiliza un filtro de partículas para estimar la posición y orientación (pose) 2D de un robot a partir de los datos de los sensores. El algoritmo funciona representando la pose del robot como una distribución de partículas, donde cada partícula representa una posible pose del robot frente a un mapa conocido. El número de partículas M es un parámetro que permite

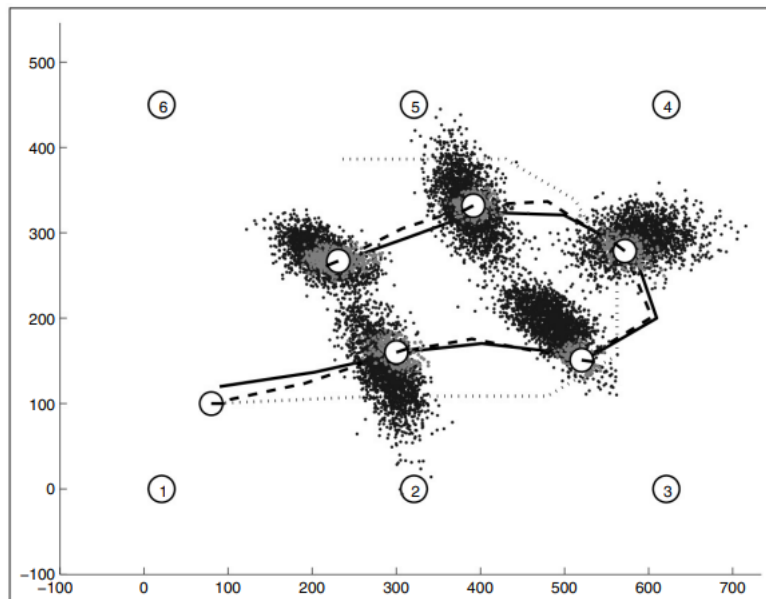


FIGURA 3.8: Algoritmo AMCL basada en puntos de referencia [43].

al usuario sopesar la precisión del cálculo y los recursos informáticos necesarios para ejecutar el algoritmo AMCL. Aumentar el número total de partículas incrementa la precisión de la aproximación [43].

La Fig. 3.8 muestra una secuencia de conjuntos de muestras generadas por el algoritmo AMCL. La línea continua representa la trayectoria real del robot, la línea de puntos representa la trayectoria basada en la información de control, y la línea discontinua representa la trayectoria media estimada por el algoritmo MCL. En cada pose, el robot toma medidas de los sensores y las compara con un mapa del entorno para determinar la probabilidad de que cada partícula sea la verdadera pose del robot. Las partículas con bajas probabilidades se descartan, mientras que las partículas con altas probabilidades se vuelven a muestrear para generar un nuevo conjunto de partículas para la siguiente pose [43].

3.6 Planificación de Movimiento

Una vez se obtiene el mapa del entorno de trabajo, se busca planificar y generar la trayectoria del robot móvil para que este llegue a un punto específico evitando los obstáculos en el camino. La generación de trayectoria se puede dividir en 2 categorías: generación de trayectoria global y generación de trayectoria local. En la generación de trayectoria global el robot móvil conoce su entorno en su totalidad antes de comenzar. A diferencia de la generación de trayectoria global, en la generación de trayectoria local el robot móvil desconoce casi todo su entorno antes de comenzar [44].

El algoritmo de Dijkstra es un algoritmo de planificación global de trayectorias que resuelve el problema del camino más corto en un grafo, el cual es un conjunto de vértices unidos por aristas. El algoritmo mantiene un conjunto de vértices donde dos de ellos se denominan A y B. El vértice inicial A está vacío al ser el punto de partida y B es el punto a donde se desea llegar. Cada arista que une 2 vértices mantiene un peso de manera que cuando se obtiene el camino más corto desde el vértice inicial, al resto de vértices que componen el grafo, el algoritmo se detiene. Este algoritmo funciona correctamente cuando el entorno de trabajo del robot móvil es estático [44].

En la Fig. 3.9 (a) se muestra un ejemplo de grafo cuyo objetivo es encontrar el camino más corto del punto A al punto B. La Tabla 3.2, encuentra la solución mediante el método de Dijkstra, donde el Paso 1 parte del origen A y, al ser el punto inicial, no tiene una arista por lo cual el Paso 1 en el vértice A es $(0, A)$. Lo mismo se replica en los siguientes pasos por cada vértice, las celdas sombreadas de azul implican el camino más corto del paso anterior para calcular sobre ello el peso acumulado de las aristas y el vértice de dónde proviene. Finalmente, en el Paso 4, se selecciona el camino más corto para llegar al objetivo (B) el cual es $(10, C)$ lo que significa que el peso acumulado de las aristas es 10 y proviene del vértice C. En la Fig. 3.2 (b) se muestra el resultado del camino más corto (aristas azules) obtenido por dicho método.

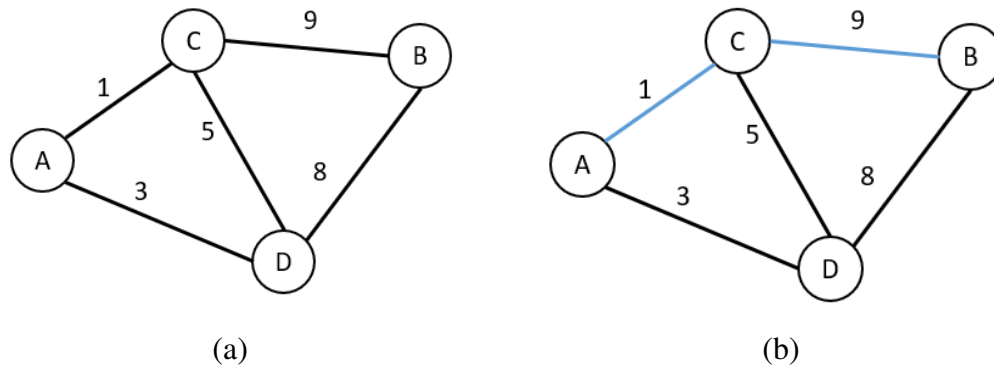


FIGURA 3.9: Algoritmo Dijkstra. (a) Ejemplo de grafo. (b) Ejemplo de grafo resuelto.

Vértice	Paso 1	Paso 2	Paso 3	Paso 4
A	(0, A)	*	*	*
B	—	(10, C)	(14, D)	(10, C)
C	(1, A)	(1, A)	*	*
D	(3, A)	(6, C)	(6, C)	*

TABLA 3.2: Solución con el método Dijkstra al grafo presentado en la Fig. 3.9 (a).

El algoritmo DWA, *dynamic window approach* por sus siglas en inglés, es un algoritmo de evitación de obstáculos en tiempo real altamente eficaz que transforma el problema de planificación de la trayectoria en un problema de optimización restringida del espacio de velocidad y controla el movimiento del robot mediante la salida de la velocidad óptima en tiempo real. Sin embargo, el DWA se enfrenta a problemas como los óptimos locales y una baja tasa de éxito en la evitación de obstáculos dinámicos [45]. La idea clave del DWA es muestrear una serie de trayectorias previstas en función de la posición actual (x_t, y_t) , la dirección θ_t y la velocidad v_r del robot, como se muestra en la Fig. 3.10 [46].

3.7 Criterio de optimización

Existen muchos factores que pueden ser considerados como criterios de optimización para el planeamiento del camino del robot móvil. En el presente trabajo, lo que

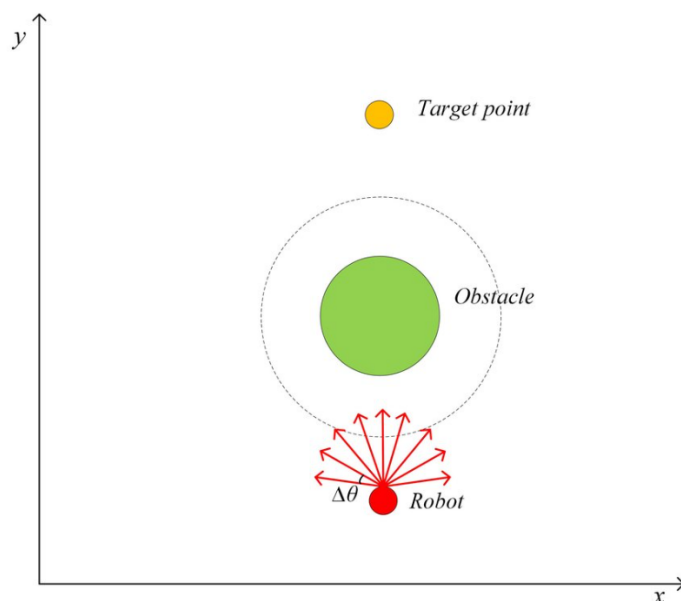


FIGURA 3.10: Trayectorias del robot simuladas de un robot mediante DWA [46]

se busca encontrar los puntos óptimos de parada para que el robot móvil desinfecte todas las superficies posibles del ambiente. La selección del algoritmo de optimización depende del tipo de función a minimizar como también del costo computacional que conlleva el procesamiento de búsqueda los puntos óptimos.

El método de algoritmos genéticos es uno de los métodos más populares y usados en los últimos años. Sin embargo, su costo computacional es elevado por lo cual no es recomendable optimizar de manera *online* sino de manera *offline* [47]. A diferencia de este método, la programación cuadrática secuencial (SQP, por sus siglas en inglés) sí es recomendable para trabajar de manera *online* debido a su bajo costo computacional. SQP es un método iterativo que resuelve una secuencia de programas cuadráticos que se acercan a la solución del problema de sistemas no lineales mediante la linealización de las restricciones y el uso de una aproximación cuadrática a la función objetivo [48]. Sin embargo, cuando se trata de sistemas que son lineales cuya función de costo y restricciones son lineales, es recomendable utilizar el método de programación lineal, pues esta clase de problemas de optimización presentan una sola solución global.

3.7.1 Programación lineal

La optimización lineal es una técnica de optimización para un sistema en el que la función objetivo a minimizar es lineal en las incógnitas y las restricciones consisten en igualdades y desigualdades lineales. La función objetivo define la cantidad a optimizar y el objetivo de dicha técnica es encontrar los valores de las variables que maximicen o minimicen la función objetivo. La forma exacta de estas restricciones puede variar de un problema a otro, pero como se muestra a continuación, cualquier programa lineal puede transformarse en la siguiente forma estándar [49]:

$$\begin{array}{ll}
 \text{minimizar} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
 \text{sujeto a} & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
 & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
 & \vdots \qquad \qquad \qquad \vdots \\
 & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \\
 \text{y} & x_1 \geq 0, x_2 \geq 0, \cdots, x_n \geq 0,
 \end{array}$$

donde b_i, c_i y a_{ij} son constantes reales fijas, y x_i son números reales a ser determinados, también llamados variables básicas.

En notación vectorial más compacta, este problema estándar se convierte en:

$$\begin{array}{ll}
 \text{minimizar} & \mathbf{c}^T \mathbf{x} \\
 \text{sujeto a} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ y } \mathbf{x} \geq \mathbf{0}.
 \end{array}$$

donde \mathbf{x} es un vector columna de dimensión n , \mathbf{c}^T es un vector fila de dimensión n , \mathbf{A} es una matriz $m \times n$, y \mathbf{b} es un vector columna de dimensión m . La desigualdad vectorial $\mathbf{x} \geq \mathbf{0}$ significa que cada componente de \mathbf{x} es no negativo.

Los problemas de programación lineal se pueden resolver utilizando técnicas como el método de punto interior o el método simplex. Estos algoritmos permiten encontrar la solución óptima del problema de forma eficiente. El solver *linprog* de Optimization Toolbox™ de Matlab implementa estos algoritmos para resolver el problema de programación lineal.

El método simplex consiste en pasar de una solución factible básica (es decir, un punto extremo) del conjunto de restricciones de un problema en forma estándar a otra, de manera que disminuya continuamente el valor de la función objetivo hasta alcanzar un mínimo. El método simplex se desarrolla a partir de un examen minucioso del sistema de ecuaciones lineales que define las restricciones y las soluciones básicas factibles del sistema [49].

El método simplex se considera desde un enfoque teórico matricial, que se centra en todas las variables juntas. Este punto de vista más sofisticado conduce a una representación notacional compacta, a una mayor comprensión del proceso simplex y a métodos alternativos de aplicación. Ante la desigualdad $\mathbf{Ax} \leq \mathbf{b}$, el método crea una variable de holgura (\mathbf{S}) para generar una igualdad en las restricción. Una vez las inecuaciones se igualaron al coeficiente de la restricción y se convirtieron en ecuaciones ($\mathbf{Ax} + \mathbf{S} = \mathbf{b}$), se procede a crear la tabla simplex, la cual se puede observar en la Tab. 3.3, que se completa con las constantes que acompañan a las variables.

Variabes básicas	f	x_1	x_2	\dots	x_n	S_1	S_2	\dots	S_m	Coef. restricción
S_1	0	a_{11}	a_{12}	\dots	a_{1n}	1	0	\dots	0	b_1
S_2	0	a_{21}	a_{22}	\dots	a_{2n}	0	1	\dots	0	b_2
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots	\ddots	\vdots	\dots
S_m	0	a_{m1}	a_{m2}	\dots	a_{mn}	0	0	\dots	1	b_m
f	1	$-c_1$	$-c_2$	\dots	$-c_n$	0	0	0	0	0

TABLA 3.3: Tabla simplex

Luego de haber creado la tabla simplex, se debe encontrar la columna pivote, la cual es la columna de la tabla simplex con el valor más negativo en la fila de la función

f . Seguidamente, se debe encontrar la fila pivote, la cual es la fila de la tabla *símplex* que tenga el menor valor luego de dividir el coeficiente de restricción entre la variable de la columna pivote. Una vez encontrada la columna y la fila pivote, se encuentra el elemento pivote, el cual es la intersección de dicha columna y fila.

De esta manera, la fila pivote se convierte en la fila saliente en la siguiente iteración de la tabla *símplex* y la fila entrante, la que reemplaza a la fila saliente, contiene los mismos valores que la fila saliente pero divididos entre el elemento pivote. En el caso de las demás filas, se reemplazan por los mismos valores de la fila de la iteración anterior menos el coeficiente pivote de la fila por la fila entrante. Las iteraciones de la tabla *símplex* se realizan hasta encontrar la solución óptima, lo cual se refleja cuando no hay valores negativos en la función f . El resultado de dichas iteraciones son los coeficientes que se obtienen en las variables básicas.

Aunque en la práctica el método *símplex* funciona bien, visitando sólo una pequeña fracción del número total de vértices, no se disponía de una teoría definitiva del rendimiento del método *simplex*. Sin embargo, se demostró mediante ejemplos que para ciertos programas lineales el método *simplex* examinará todos los vértices. Estos ejemplos demostraron que, en el peor de los casos, el método *simplex* requiere un número de pasos que es exponencial en el tamaño del problema. A la vista de este resultado, muchos investigadores creyeron que podría idearse un buen algoritmo, distinto del método *simplex*, cuyo número de pasos fuera polinómico en lugar de exponencial en el tamaño del programa [49].

En ese sentido, el método de punto interior resulta especialmente útil para programas lineales de gran escala que tienen estructura o pueden definirse utilizando matrices dispersas. En primer lugar, el algoritmo intenta simplificar el problema eliminando redundancias y simplificando las restricciones. En segundo lugar, genera un punto inicial lo cual es especialmente importante para resolver eficazmente los algoritmos de punto interior.

Finalmente, se realizan las iteraciones predictor-corrector de manera que el método de punto interior intenta encontrar un punto en el que se cumplan las condiciones del teorema Karush-Kuhn-Tucker (KKT). Para describir estas ecuaciones para el problema de programación lineal, considere la forma estándar del problema de programación lineal después del preprocesamiento [50]:

$$\begin{aligned} & \min f(x) \\ & \text{sujeto a } \begin{cases} \bar{A}x = \bar{b} \\ x + t = u \\ x, t \geq 0 \end{cases} \end{aligned} \quad (3.28)$$

donde \bar{A} es la matriz lineal ampliada que incluye tanto las desigualdades lineales como las igualdades lineales, \bar{b} es el vector de igualdad lineal correspondiente, u es la restricción superior de x y t es el vector de holguras que convierten los límites superiores en igualdades.

El Lagrangiano del sistema presentado en la ecuación (3.28) involucra 3 vectores: y , contiene los multiplicadores de Lagrange asociados a las igualdades lineales, v , multiplicadores de Lagrange asociados al límite inferior y w , multiplicadores de Lagrange asociados al límite superior. En ese sentido, las condiciones KKT para el sistema son[50]:

$$\begin{aligned} f - \bar{A}^T y - v + w &= 0 \\ \bar{A}x &= \bar{b} \\ x + t &= u \\ v_i x_i &= 0 \\ w_i t_i &= 0 \\ (x, v, w, t) &\geq 0 \end{aligned}$$

El algoritmo predice primero un paso a partir de la fórmula Newton-Raphson y, a continuación, calcula un paso corrector. Tras calcular el paso Newton corregido, el algoritmo realiza más cálculos para obtener tanto un paso actual más largo, como para prepararse mejor para los pasos posteriores. Estos múltiples cálculos de corrección pueden mejorar tanto el rendimiento como la robustez. El algoritmo predictor-corrector itera hasta que alcanza un punto que es factible (satisface las restricciones dentro de las tolerancias) y en el que los tamaños relativos de los pasos son pequeños.

Capítulo 4

METODOLOGÍA

En esta sección se muestra la metodología utilizada para realizar la presente tesis empezando a explicar la manera en que se encuentran los puntos óptimos de desinfección en un mapa construido previamente mediante programación lineal. Seguidamente, se presenta la localización del robot y la planificación de movimiento que realizó el robot para lograr llegar a su objetivo final en base al modelo de irradiación. Luego, el análisis de la cinemática directa e inversa para determinar la movilidad del robot. Finalmente, se explica el método de colorimetría para validar la desinfección de las superficies en un 99.9 %.

En la Fig. 4.1, se muestra la integración de las definiciones descritas en el Capítulo 3 para el desarrollo del presente trabajo. Se utilizaron dos paquetes para la localización y la generación de trayectoria en ROS, *Robot Operating System* por sus siglas en inglés, el cual es un conjunto de bibliotecas y herramientas de software que se utilizan para la creación de aplicaciones robóticas.

4.1 Diseño mecánico del robot desinfectante

El diseño mecánico del robot, en el que se implementará el presente trabajo de tesis, consta de 2 partes. La primera es la estructura del carrito y la segunda es la estructura que sostiene las lámparas UV-C. La integración de ambos diseños en Autodesk Inventor, se pueden observar en la Fig. 4.2(a). En la Fig. 4.2(b) se puede observar la implementación del diseño mecánico del robot desinfectante. El robot utiliza 4 ruedas mecanum y 3 lámparas UV-C.

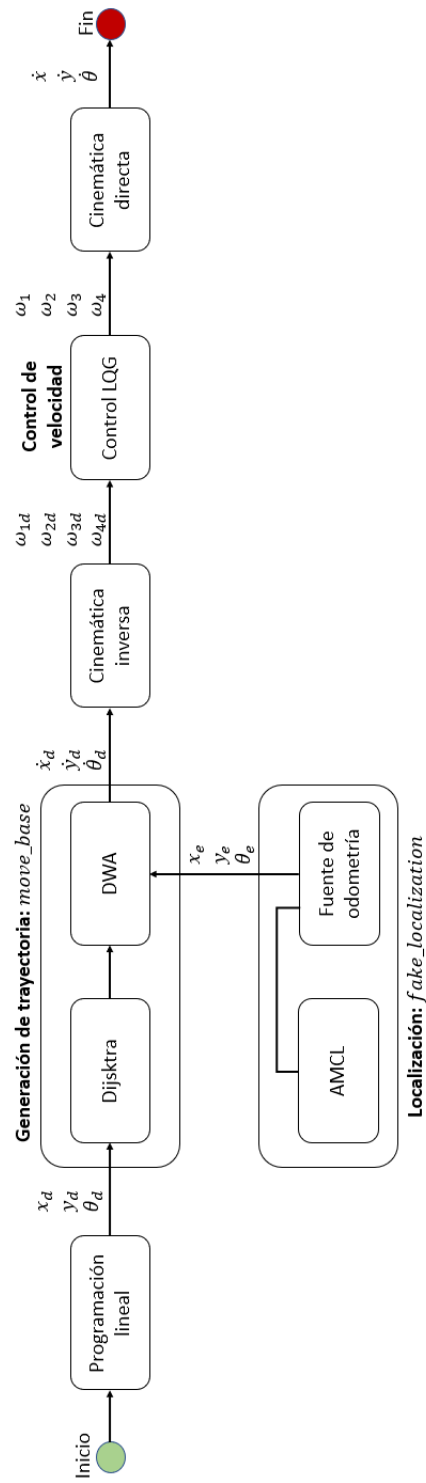


FIGURA 4.1: Diagrama de bloques de la metodología empleada en este trabajo.

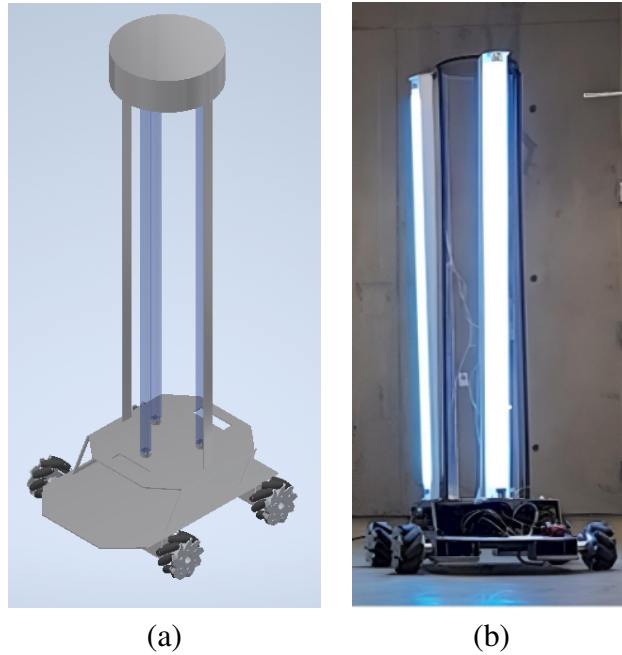


FIGURA 4.2: Diseño mecánico del robot desinfectante. (a) En Autodesk Inventor. (b) Implementado.

4.2 Selección de waypoints

Con el mapa del entorno de trabajo construido, se escogen los puntos específicos dentro del ambiente en los que el robot debe parar para desinfectar la sala. Dichos puntos de parada se denominan *waypoints*. Para dicha selección existen 3 pasos principales: discretización del mapa, cantidad de irradiación recibida por la superficie y el establecimiento de la función objetivo y sus restricciones. Además, es necesario que el mapa ingresado a dicho proceso sea en blanco y negro, esto con el objetivo de facilitar la discretización y diferenciar los obstáculos de los puntos libres.

Para establecer cuánta irradiancia le suministra un punto libre en el mapa, o también posible *waypoint*, a un punto perteneciente a un obstáculo, se toman las siguientes consideraciones:

$$Ir_u(v) = \begin{cases} \frac{P}{2\pi \times L \times r} & \text{si } u \text{ ve a } v \\ 0 & \text{caso contrario} \end{cases}$$

donde u es un posible *waypoint*, v es un punto parte de un obstáculo, $Ir_u(v)$ es la irradiancia recibida por cada posible u hacia cada v , P es la potencia de la lámpara, L es la longitud de la lámpara y r es la distancia que existe entre u y v .

Seguidamente, se establece el problema de optimización:

$$\begin{array}{ll} \text{minimizar} & t_{u_1} + t_{u_2} + \dots + t_{u_n} \\ \text{sujeto a} & Ir_{u_1v_1}t_{u_1} + Ir_{u_2v_1}t_{u_2} + \dots + Ir_{u_nv_1}t_{u_n} \geq 3.7 \\ & Ir_{u_1v_2}t_{u_1} + Ir_{u_2v_2}t_{u_2} + \dots + Ir_{u_nv_2}t_{u_n} \geq 3.7 \\ & \vdots \hspace{10em} \vdots \\ & Ir_{u_1v_m}t_{u_1} + Ir_{u_2v_m}t_{u_2} + \dots + Ir_{u_nv_m}t_{u_n} \geq 3.7 \\ \text{y} & t_{u_1} \geq 0, t_{u_2} \geq 0, \dots, t_{u_n} \geq 0, \end{array}$$

Dichas ecuaciones del problema de optimización se pueden sintetizar en:

$$f = \sum_u t_u \tag{4.1}$$

donde

$$t_u \geq 0 \tag{4.2}$$

$$\sum_u t_u Ir(v) \geq 3.7 \tag{4.3}$$

Con las ecuaciones descritas previamente, se hallan los puntos óptimos para garantizar la desinfección en un 99.9 % de las superficies.

4.3 Localización del robot móvil

Para que el robot móvil pueda ubicarse mientras navega su entorno, es necesario que conozca su pose. Por tal motivo, se utiliza la odometría de las ruedas mecanum del robot las cuales son accionadas por motores DC. La manera de calcular la odometría será mediante los sensores propieceptivos con los que cuentan los motores DC: encoders incrementales.

El paquete *fake_localization* de ROS que se utiliza para obtener la localización del robot proporciona un único nodo, el cual sustituye a un sistema de localización, proporcionando un subconjunto de la API ROS utilizada por AMCL. Para contextualizar, en ROS, los nodos son ejecutables escritos en diversos lenguajes como C++, Python, Java, entre otros. El principal mecanismo utilizado por los nodos ROS para comunicarse es el envío y recepción de mensajes. Los mensajes se organizan en categorías específicas denominadas tópicos. Los nodos se configuran mediante parámetros y pueden publicar mensajes sobre un tópico concreto o suscribirse a un tópico para recibir información.

En ese sentido, *fake_localization* convierte los datos de odometría en datos de pose, nube de partículas y transformación de la forma publicada por AMCL. En ROS, la configuración del nodo consta en los siguientes parámetros:

- **odom_frame_id**: Nombre del marco odométrico del robot.
- **delta_x**: Desplazamiento x entre el origen del marco de coordenadas del simulador y el marco de coordenadas del mapa publicado por *fake_localization*.
- **delta_y**: Desplazamiento y entre el origen del marco de coordenadas del simulador y el marco de coordenadas del mapa publicado por *fake_localization*.
- **delta_yaw**: Rotación en z entre el origen del marco de coordenadas del simulador y el marco de coordenadas del mapa publicado por *fake_localization*.

- **global_frame_id**: Marco de referencia global.
- **base_frame_id**: Marco de referencia con respecto a la base.

4.4 Generación de trayectoria

Para el planeamiento de trayectoria en ROS se utiliza el paquete de navegación *move_base*, el cual permite utilizar el algoritmo de Dijkstra y DWA para generar tanto una trayectoria global como también una local. Para enviar los *waypoints* obtenidos mediante la optimización lineal, se crea un nodo que manda mensajes al paquete *move_base*, el cual le permite establecer la pose como también el tiempo de parada para la desinfección.

El paquete *move_base* proporciona un nodo que mueve el robot desde su posición actual a una posición objetivo. Cuando se realiza la planificación de la ruta, la meta de navegación 2D, posición (x,y,z) y orientación (x,y,z,w) se publican en el tópico *move_base/goal*. El nodo *move_base* enlaza un planificador global y uno local para llevar a cabo su tarea de navegación. Algunos de los parámetros para la configuración del nodo son los siguientes:

- **base_global_planner**: Se encarga de calcular la trayectoria segura para llegar a la posición objetivo. El paquete ofrece 3 tipos de planeadores globales (NavfnPlanner, CarrotPlanner y GlobalPlanner).
- **base_local_planner**: Proporciona órdenes de velocidad al robot y hace que siga la trayectoria local. El paquete ofrece 4 tipos de planeadores locales (*dwa_local_planner*, *base_local_planner*, *eband_local_planner*, *teb_local_planner*)

En el caso del planeador global *GlobalPlanner*, los parámetros que deben ser configurados en ROS para su funcionamiento son los siguientes:

- **allow_unknown**: Especifica si se permite o no al planificador crear planes que atravesen un espacio desconocido.
- **visualize_potential**: Especifica si se visualiza o no el área potencial calculada mediante un PointCloud2.
- **use_dijkstra**: Si es verdadero, utiliza el algoritmo de Dijkstra. Si no, A*.
- **use_quadratic**: Si es verdadero, utiliza la aproximación cuadrática del potencial. En caso contrario, utiliza un cálculo más sencillo.
- **use_grid_path**: Si es verdadero, crea una ruta que sigue los límites de la malla. En caso contrario, utiliza un método de descenso por gradiente.

En el caso del planeador local *dwa_local_planner*, los parámetros que deben ser configurados en ROS para su funcionamiento son los siguientes:

- **max_vel_trans**: Valor absoluto de la velocidad máxima de traslación del robot en *m/s*.
- **min_vel_trans**: Valor absoluto de la velocidad mínima de traslación del robot en *m/s*.
- **max_vel_x**: La velocidad x máxima del robot en *m/s*.
- **min_vel_x**: La velocidad x mínima del robot en *m/s*.
- **max_vel_y**: La velocidad y máxima del robot en *m/s*.
- **min_vel_y**: La velocidad y mínima del robot en *m/s*.
- **max_vel_theta**: La velocidad rotacional máxima del robot en *rad/s*.
- **min_vel_theta**: La velocidad rotacional mínima del robot en *rad/s*.

- **acc_lim_x**: El límite de aceleración x del robot en m/s^2 .
- **acc_lim_y**: El límite de aceleración y del robot en m/s^2 .
- **acc_lim_theta**: El límite de aceleración rotacional del robot en rad/s^2 .
- **xy_goal_tolerance**: La tolerancia en metros para el controlador en la distancia x e y al alcanzar un objetivo.
- **yaw_goal_tolerance**: La tolerancia en radianes para el controlador en rotación cuando alcanza su objetivo.
- **path_distance_bias**: La ponderación de qué tanto el controlador debe permanecer cerca de la ruta que se le dio.
- **goal_distance_bias**: La ponderación de qué tanto debe intentar el controlador alcanzar su objetivo local.

El nodo *move_base* también mantiene dos mapas de costes, uno para el planificador global y otro para un planificador local que se utilizan para realizar tareas de navegación. Los parámetros para el mapa de costes de los planificadores en ROS son los siguientes:

- **global_frame**: El marco global en el que operará el mapa de costes.
- **robot_base_frame**: El nombre de la estructura para el eslabón base del robot.
- **update_frequency**: La frecuencia en Hz para que se actualice el mapa.
- **publish_frequency**: La frecuencia en Hz para que se publique la información de visualización del mapa.
- **transform_tolerance**: Especifica el retardo en los datos de transformación que es tolerable en segundos. Este parámetro sirve de salvaguarda para evitar la pérdida de un enlace en el árbol de datos de transformación.

- **inflation_layer**: Añade nuevos valores alrededor de los obstáculos letales (es decir, infla los obstáculos) para que el mapa de costes represente el espacio de configuración del robot.
- **static_map**: Define si el mapa es estático o no.
- **map_type**: Define que es un mapa de costes.
- **rolling_window**: Si se utiliza o no una versión de ventana móvil del mapa de costes. Si el mapa es estático, este valor debe ser falso.
- **width**: La anchura del mapa en metros.
- **height**: La altura del mapa en metros.
- **resolution**: La resolución del mapa en metros/celda.

4.5 Cinemática y Control LQG

Utilizando el planeador local DWA, el cual forma parte del paquete *move_base* de ROS, se obtiene la velocidad deseada en x , y y $theta$ del robot para llegar a los *waypoints*. A dichas velocidades se le aplica la cinemática inversa para obtener las velocidades angulares deseadas (ω_{1d} , ω_{2d} , ω_{3d} y ω_{4d}) para cada rueda del robot a utilizar, que en este caso son 4. En ese sentido, las variables de estado de cada controlador LQG son la velocidad angular de la rueda del robot y la corriente de la armadura del motor DC. Una vez obtenidas las velocidades angulares deseadas, ingresan como referencia al control de velocidad LQG con lo cual se asegura que las velocidades deseadas se repliquen en las velocidades reales. Con la cinemática directa e inversa se establece el control de trayectoria del robot móvil con el controlador LQG tal como se puede observar en la Fig. 4.3.

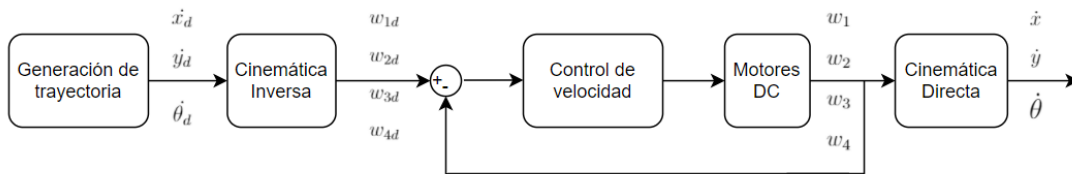


FIGURA 4.3: Esquema del control de trayectoria LQG.

4.6 Análisis de dosis UV-C por colorimetría

Para validar la maximización de dosis UV-C suministrada a las superficies, se colocarán marcadores colorimétricos sensibles a los rayos UV-C de la marca Intellego en diferentes ubicaciones y orientaciones en el lugar donde el robot móvil navegará. Los datos de dosis acumulada en los marcadores colorimétricos serán calculados adquiriendo los colores de los marcadores y extrayendo los valores de la matriz G del modelo de color RGB, tal como se realizó en [11]. Según la tabla de referencia Tab. 3.1, se usará la matriz G para calcular el valor de dosis en cada marcador basado en el ajuste de mínimos cuadrados tomando como referencia los colores disponibles y la dosis que representan. En ese sentido, la ecuación logarítmica resultante para obtener el valor de la dosis acumulada es la siguiente:

$$D = -68.32 \ln(G) + 378.52 \quad (4.4)$$

donde D es la dosis acumulada expresada en mJ/cm^2 y G son los valores de color de la matriz RGB que oscilan entre 0 y 255.

Capítulo 5

RESULTADOS

En este capítulo se presentarán los resultados obtenidos en el presente trabajo. En primer lugar, se mostrarán los resultados de optimización de puntos de desinfección. Luego, se describirá al robot y sus parámetros para obtener el modelo cinemático. Seguidamente, se mostrarán los parámetros establecidos del controlador LQG para asegurar el desempeño de la planeación de trayectoria y los resultados del comportamiento del robot. Posteriormente, se presentará las simulaciones obtenidas en ROS. Finalmente, se valida la desinfección mediante el análisis colorimétricos de los dosímetros. La prueba experimental de la trayectoria se implementó en el sótano uno de la Universidad de Ingeniería y Tecnología (UTEC).

5.1 Resultados simulados

5.1.1 Selección de waypoints

Para establecer la generación de trayectoria del robot móvil, es necesario que el robot conozca los puntos de desinfección que formarán parte de su trayectoria. Por tal motivo, al tratarse de un problema con función y restricciones lineales, la obtención de dichos puntos de desinfección se tomó como un problema de optimización lineal.

Para efectos de simulación, se utilizó el software Matlab, en el cual un mapa a blanco y negro es ingresado al algoritmo, dicho algoritmo se encarga de discretizar el mapa lo cual ayuda a obtener los posibles waypoints; es decir, los puntos donde el robot puede parar, como también los obstáculos. Con dicha información, el algoritmo determina

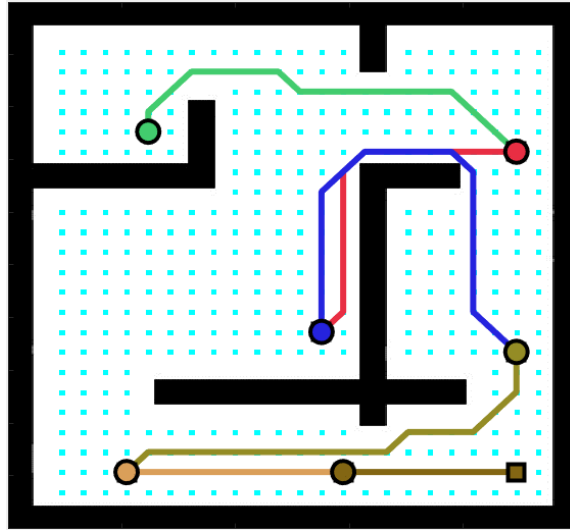


FIGURA 5.1: Waypoints de un laberinto de 5 metros por 5 metros.

cuanta dosis le suministra cada posible waypoint a cada punto que representa un obstáculo para luego utilizar la función objetivo y las restricciones, definidas en las ecuaciones (4.1), (4.2) y (4.3), dadas por la dosis que se desea suministrar. Finalmente, con la optimización lineal, se obtienen los puntos óptimos de desinfección como también el tiempo de desinfección necesario para suministrar una dosis de 3.7 mJ/cm^2 .

En la Fig. 5.1 se puede observar el primer mapa con el que se trabajó a modo de experimento, un laberinto de 5 metros por 5 metros. En dicho laberinto se encontraron 7 puntos de desinfección con un tiempo estimado de desinfección de aproximadamente 48 minutos. En la Fig 5.2 se puede observar el mapa de trabajo real cuyas medidas principales se pueden observar en la Tabla 5.1. En dicho mapa se encontraron 6 puntos de desinfección, los cuales se pueden observar en la Fig. 5.3, con un tiempo de desinfección estimado de 68 minutos.

En términos de rendimiento del código de optimización, utilizando como mapa de trabajo el sótano 1 de la UTEC, el tiempo que tarda el programa en ejecutarse para encontrar la solución al problema de optimización es 0.79 segundos y el tiempo total

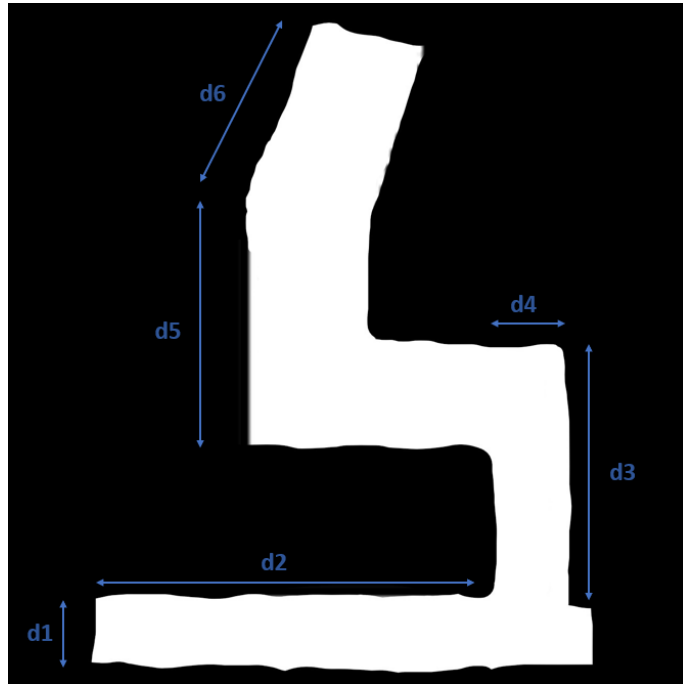


FIGURA 5.2: Dimensiones del mapa real de trabajo. Ver Tabla 5.1

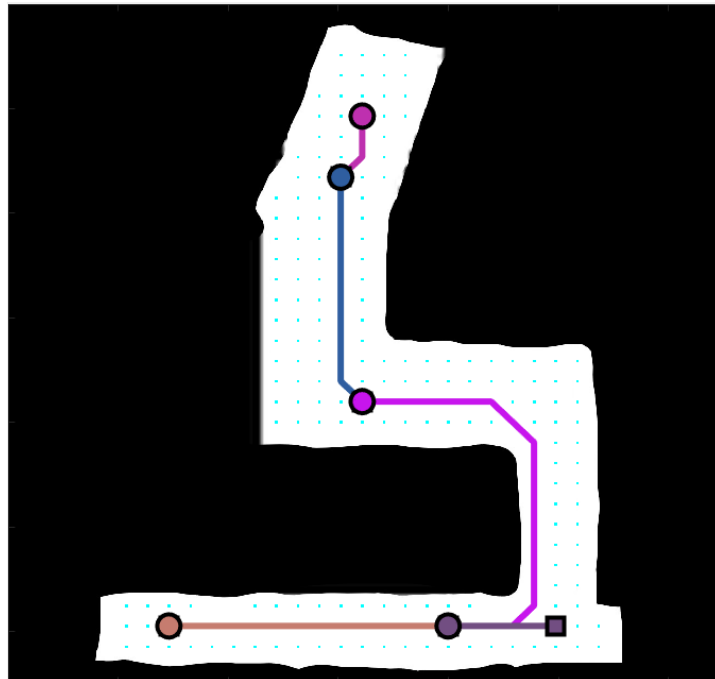


FIGURA 5.3: Waypoints en el mapa real de trabajo.

Medida	Distancia (m)
$d1$	1.8
$d2$	8.15
$d3$	4.9
$d4$	1.75
$d5$	4.6
$d6$	4.4

TABLA 5.1: Dimensiones del mapa real de trabajo.

que tarda en discretizar el mapa, identificar puntos libres y obstáculos, y dar solución al problema de optimización es 7.99 segundos. A nivel del comando *linprog*, el valor de la función objetivo en la solución es 0, lo que significa que la función objetivo se ha optimizado hasta su valor óptimo, en este caso el mínimo, y no se puede mejorar más.

5.1.2 Cinemática del robot móvil omnidireccional

El prototipo de robot construido presenta una configuración omnidireccional debido a que esta configuración le permite una mayor movilidad gracias a sus ruedas mecanum. Para obtener las ecuaciones matemáticas del modelo cinemático del robot móvil omnidireccional, se necesita conocer las dimensiones de este. Las medidas requeridas son el radio de cada rueda mecanum, el cual mide $0.5m$ en las 4 ruedas, y los datos que se pueden observar en la Fig. 5.4 y la Tabla 5.2, donde $d_x = 0.135m$ y $d_y = 0.2075m$. Con dichos valores se determinan la matriz A y la matriz B para reemplazarlas en la ecuación descrita en la ecuación (3.6) con el fin de hallar la cinemática directa. Asimismo, empleando la ecuación descrita en la ecuación (3.8), se halla la cinemática inversa.

Rueda	α_i	β_i	γ_i	l_x	l_y
$i = 1$	$\frac{19}{60}\pi$	$\frac{11}{60}\pi$	$\frac{\pi}{4}$	d_x	d_y
$i = 2$	$\frac{109}{60}\pi$	$-\frac{11}{60}\pi$	$-\frac{\pi}{4}$	d_x	$-d_y$
$i = 3$	$\frac{49}{60}\pi$	$-\frac{11}{60}\pi$	$-\frac{\pi}{4}$	$-d_x$	d_y
$i = 4$	$\frac{79}{60}\pi$	$\frac{11}{60}\pi$	$\frac{\pi}{4}$	$-d_x$	$-d_y$

TABLA 5.2: Parámetros del robot.

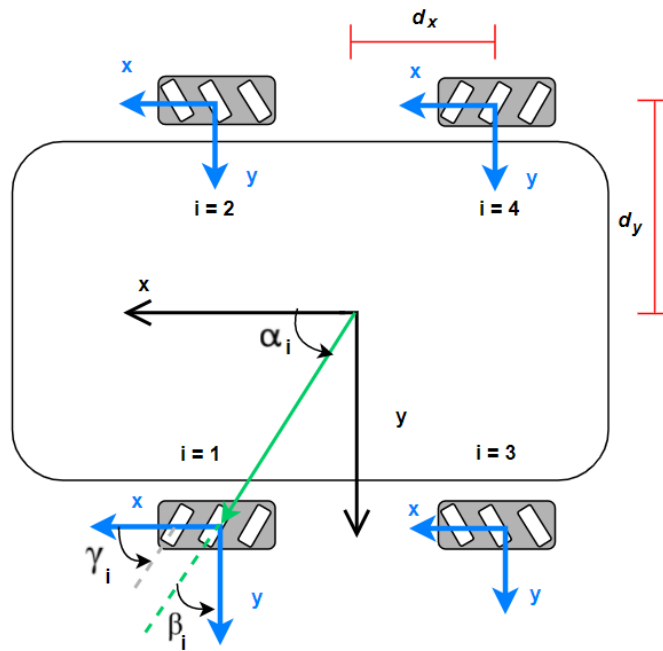


FIGURA 5.4: Parámetros de restricción y distribución geométrica del robot móvil omnidireccional.

5.1.3 Control de velocidad LQG

Con la cinemática directa e inversa se establece el control de trayectoria del robot móvil con el controlador LQG tal como se puede observar en la Fig. 4.3. Dicho controlador será implementado en un Atmega328P, mediante la placa Arduino UNO, por cada motor y la ganancia K necesaria para la ley de control será hallada previamente en MATLAB mediante la función integrada lqr y el estimador lqe .

b	0.000002 N.m.s
J	$5.7 \times 10^{-7} \text{ kg.m}^2$
K	0.0134
R	1.9Ω
L	$65 \times 10^{-6} \text{ H}$

TABLA 5.3: Parámetros del motor DC

La planta del controlador es el motor DC, el cual acciona cada rueda omnidireccional del robot. En ese sentido, evocando la ecuación (3.12), los parámetros de los motores DC a utilizar son los especificados en la Tabla 5.3. Para poder utilizar el controlador LQG, la matriz presentada en la ecuación (3.18) debe ser una matriz estable, por tal motivo se comprobó que sus raíces tengan parte real negativa. Asimismo, se obtuvo los valores propios de la matriz A , $[-170; -29064]$, lo que comprobó que la planta no tiene integradores por lo cual se aplica el efecto integral al controlador.

Para poder utilizar el controlador LQG, es necesario también determinar que el sistema sea completamente controlable y observable, según lo especificado en las ecuaciones (3.14) y (3.15) respectivamente. En ese sentido, se encontró que el rango de la matriz de controlabilidad y observabilidad fue 2, lo que garantiza que el sistema es completamente controlable y observable.

Las matrices Q y R sintonizadas para asegurar los requerimientos del diseño, evocando la función de costo descrita en la ecuación (3.16), son las siguientes:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 500 \end{bmatrix}$$

$$R = [1]$$

La ley de control se define con la ganancia del controlador LQR considerando el efecto integral, tal cual se define en la ecuación (3.27). En ese sentido, la ganancia se define de la siguiente manera:

$$K = [0.9889 \quad 0.8626] \quad (5.1)$$

$$K_i = [22.3607] \quad (5.2)$$

Para determinar la matriz de ganancia de retroalimentación del observador, L , se utiliza la función *lqe* de Matlab, donde se requiere establecer las matrices de covarianza del ruido del proceso, del sensor y del ruido Gaussiano (Q_e , R_e y G), las cuales se definen por prueba y error:

$$Q_e = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$R_e = [1]$$

$$G = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

En ese sentido, la matriz de ganancia de retroalimentación del observador resultante es la siguiente:

$$L = \begin{bmatrix} 4.8799 \times 10^{-5} \\ -2.054 \times 10^{-7} \end{bmatrix} \quad (5.3)$$

Simulando el comportamiento del controlador LQG para verificar su funcionamiento y el correcto establecimiento de sus parámetros, se establece, para efectos de prueba, que la velocidad deseada en x es 2 m/s , en y es 2 m/s y en θ es 1 rad/s , el resultado del controlador aplicado a cada rueda del robot, después de aplicar la cinemática inversa, se puede observar en la Fig. 5.5. Asimismo, el tiempo de establecimiento, error en estado estable y el porcentaje de sobreimpulso del controlador, como respuesta al escalón, se muestran en la Fig. 5.6, siendo el tiempo de establecimiento 0.175 segundos, el error en estado estable 0 y el porcentaje de sobreimpulso 0% . En la Fig. 5.7 se observa que el sistema es estable a lazo cerrado al ser su raíz mayor negativa (-22.4).

En la Fig. 5.8 se puede observar el resultado de la velocidad del robot después de aplicar la cinemática directa. En la Fig. 5.9 se puede observar el esfuerzo de control cuando se especifica que la velocidad notando que está dentro del límite de voltaje del motor DC que es 12V .

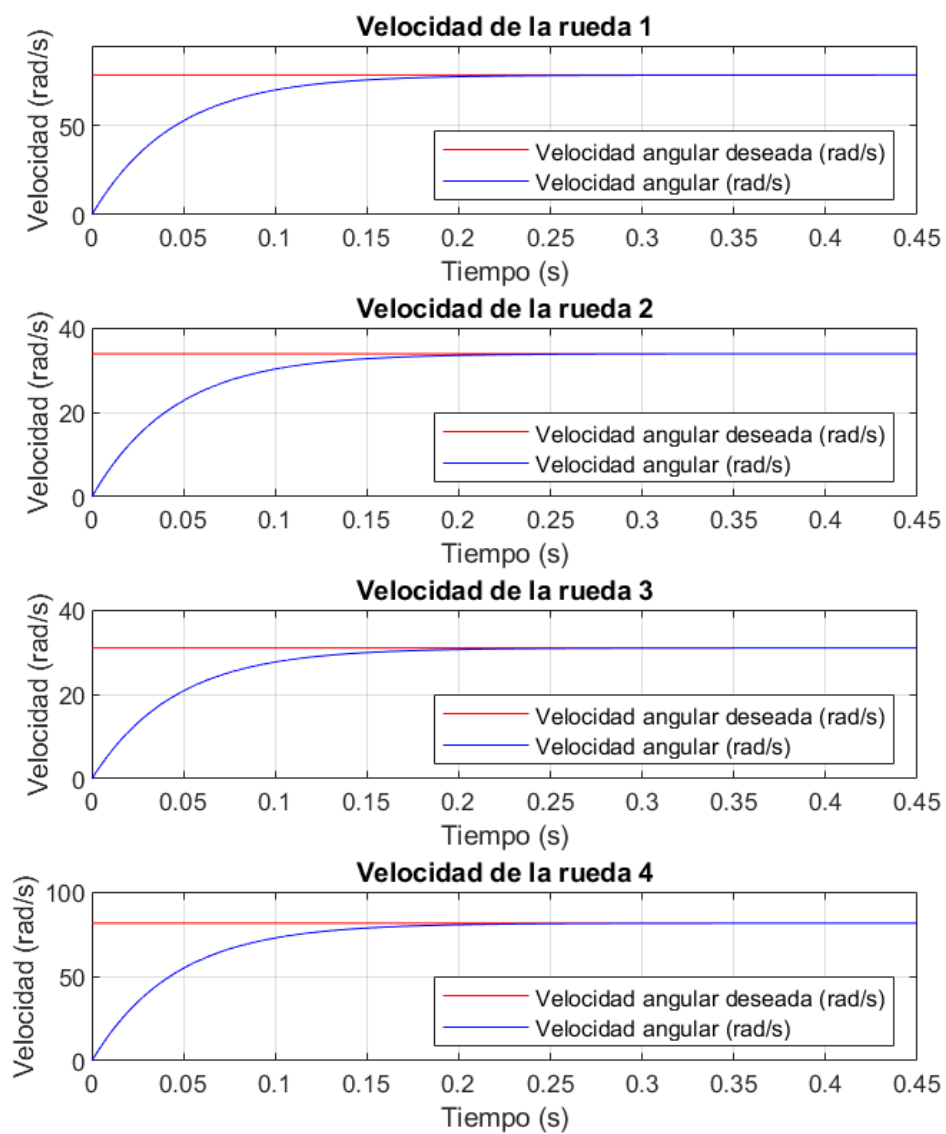


FIGURA 5.5: Velocidad de las ruedas del robot aplicando el controlador LQG.

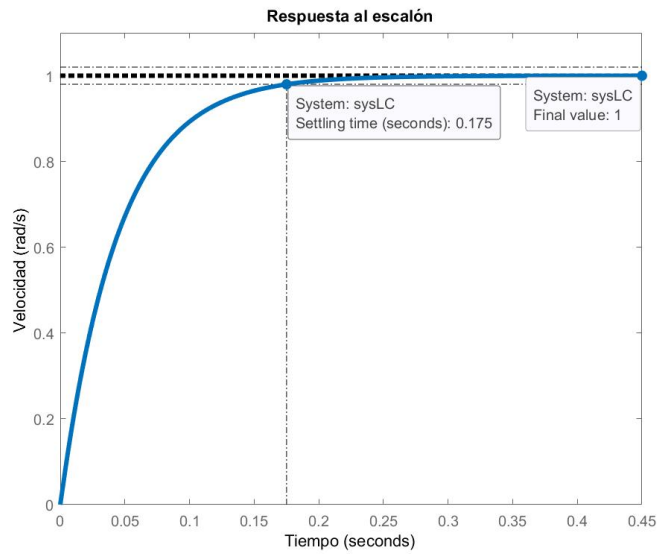


FIGURA 5.6: Tiempo de establecimiento y error en estado estable del controlador con respuesta al escalón.

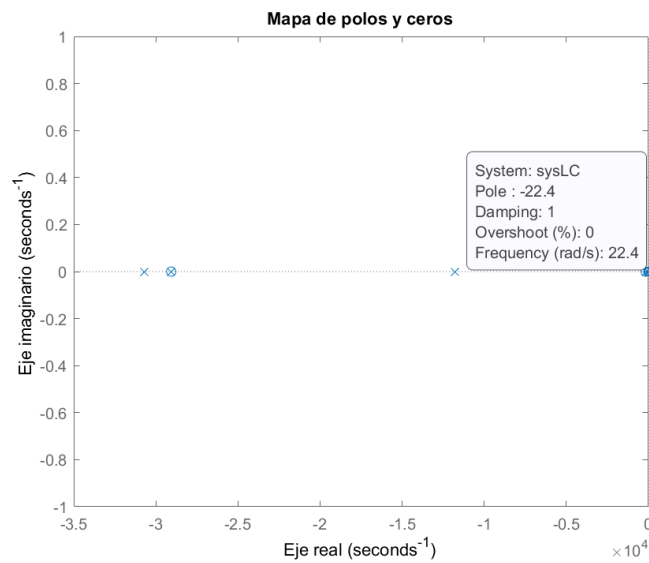


FIGURA 5.7: Mapa de polos y ceros del controlador LQG diseñado.

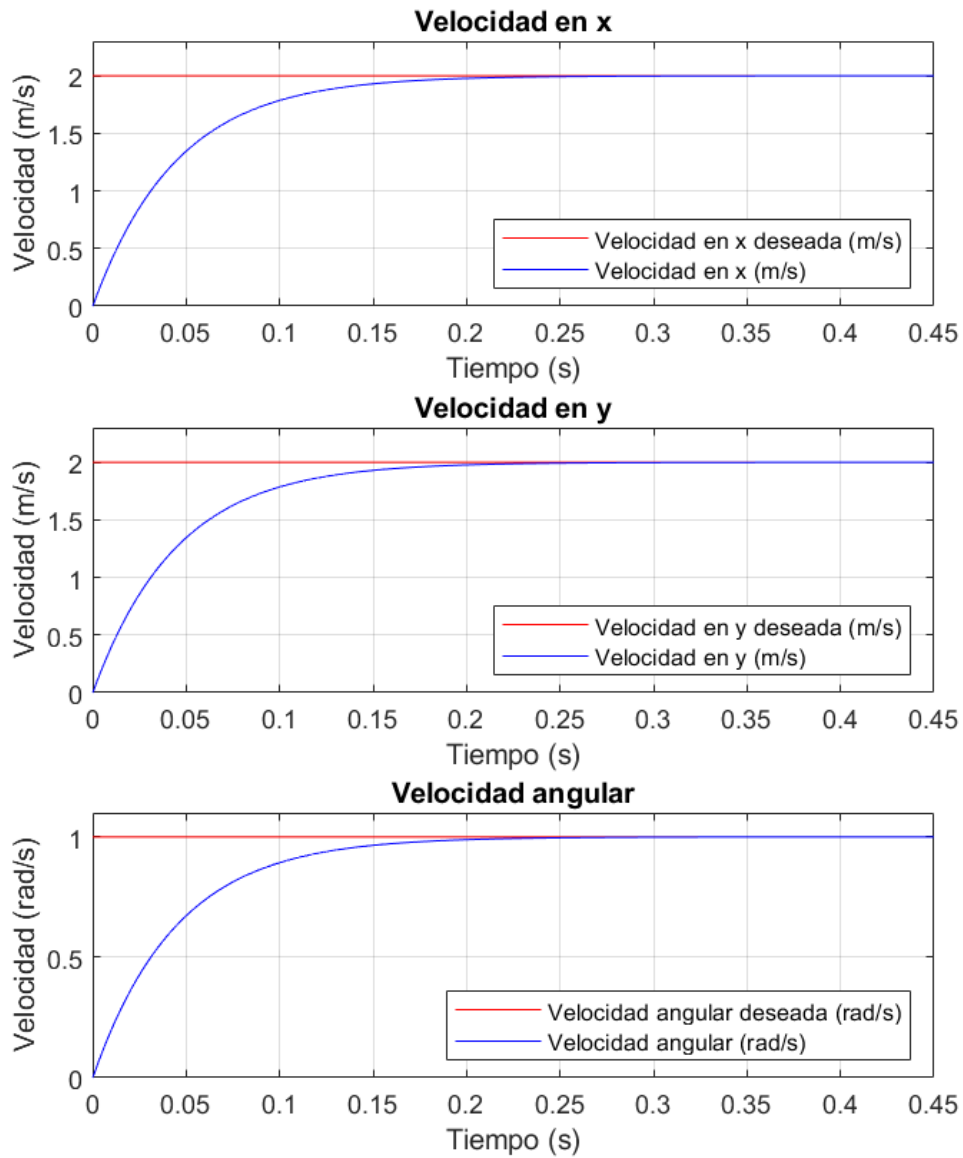


FIGURA 5.8: Velocidad del robot aplicando el controlador LQG y cinemática directa.

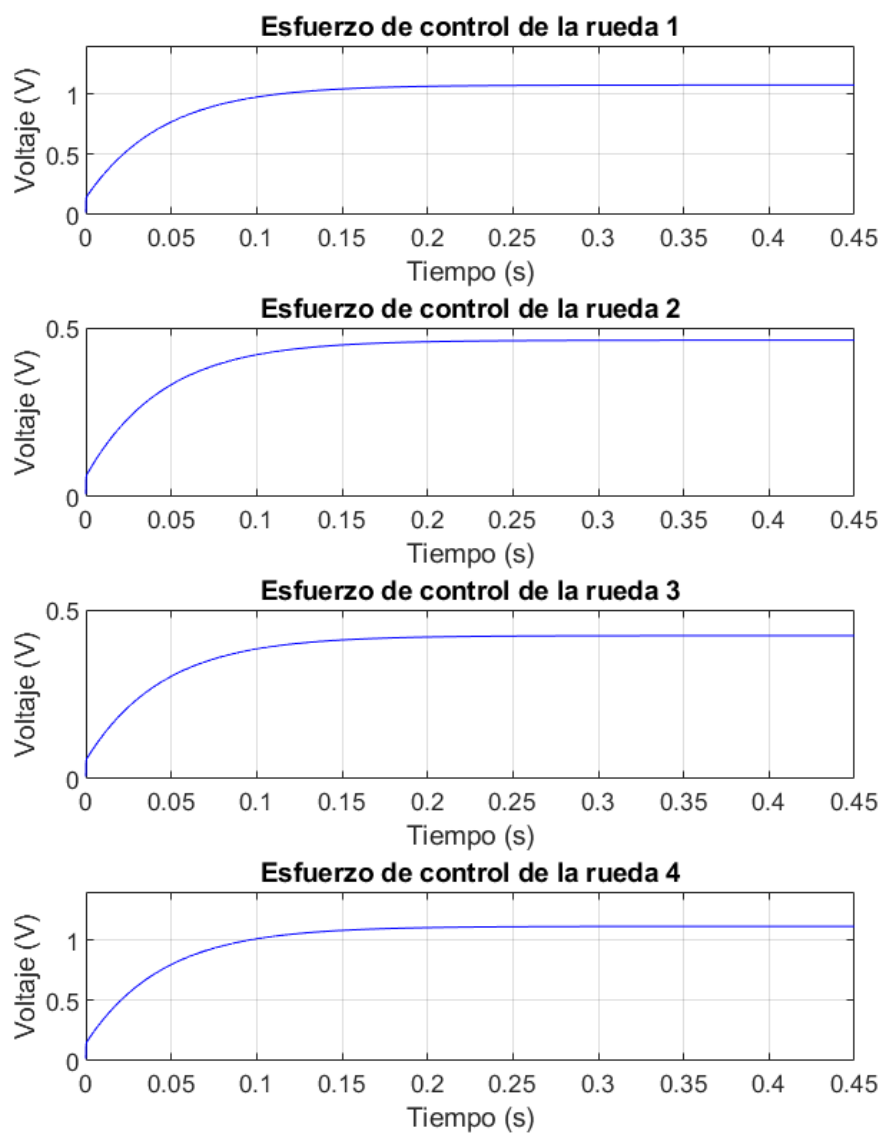


FIGURA 5.9: Esfuerzo del control LQG en cada rueda del robot.

5.1.4 Simulación de navegación semiautónoma

Antes de implementar el algoritmo de navegación se simuló el sistema mediante ROS. Cabe resaltar, que ROS permite una simulación dinámica gracias a Gazebo, el cual considera el robot con pesos e inercias como también establece un controlador para el correcto funcionamiento del sistema robótico. En la Fig. 5.10 se muestra el proceso del robot móvil para llegar al punto deseado.

Para comenzar con la navegación es necesario ubicar al robot dentro del mapa, tanto en posición como orientación. Una vez realizado lo planteado, el algoritmo establece una trayectoria entre los puntos de desinfección, donde estos son los puntos objetivos. Cabe resaltar que una vez un punto de desinfección es alcanzado, este se convierte en el punto inicial y el siguiente punto de desinfección se convierte en el punto objetivo. Para evitar que el robot móvil genere su trayectoria muy cerca de los obstáculos, lo cual le podría tomar menos tiempo en llegar a su punto objetivo, puede generar colisiones. Por tal motivo, se establece un grosor de los obstáculos presentes en el mapa lo cual no impide su paso cercano a los obstáculos pero sí lo evita. Tal como se puede observar en la Fig. 5.10 el algoritmo es capaz de generar trayectorias curvas y diagonales, lo cual también es capaz de realizar debido a la configuración omnidireccional del robot móvil.

La presente simulación permite la sintonización de los parámetros adecuados para la navegación como la velocidad máxima y mínima con la que debe navegar el robot y, de esta manera, lograr una correcta navegación sin colisiones. En ese sentido, evocando los parámetros explicados en la Sección 4.3, los parámetros en *ROS* establecidos en el paquete *fake_localization* son los siguientes:

- **odom_frame_id:** /odom
- **delta_x:** 0.0
- **delta_y:** 0.0

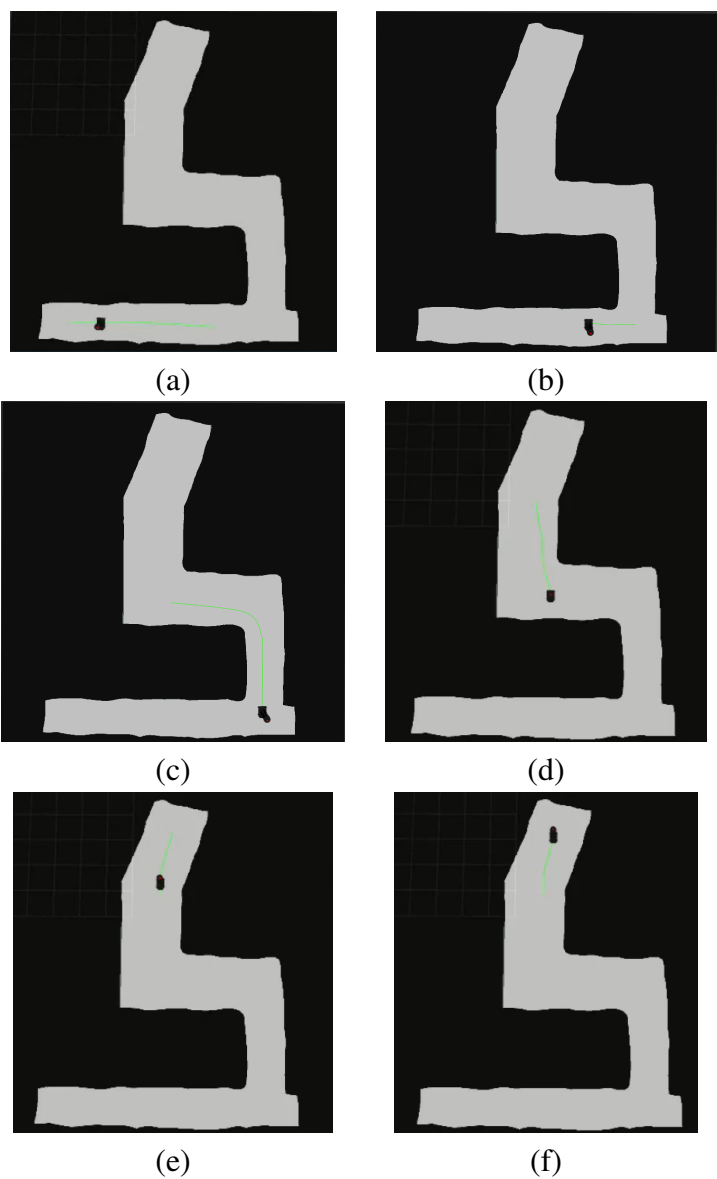


FIGURA 5.10: Simulación de la navegación semiautónoma (a) El robot parte del primer waypoint hasta el segundo waypoint. (b) El robot se encuentra en el segundo waypoint. (c) El robot se encuentra en el tercer waypoint. (d) El robot se encuentra en el cuarto waypoint. (e) El robot se encuentra en el quinto waypoint. (f) El robot llega al último waypoint.

- **delta_yaw:** 0.0
- **global_frame_id:** /map
- **base_frame_id:** omni_base_footprint

Para lograr que el robot tenga una correcta trayectoria fue necesario establecer parámetros en el paquete *move_base* dentro del entorno de *ROS* que van desde las velocidades máximas y mínimas que el robot debe presentar durante su navegación como también el grosor de obstáculos mencionado anteriormente, el cual permite evitar la colisión con dichos obstáculos. Dichos parámetros se definieron en la Sección 4.4. Los parámetros establecidos para el planeador global *GlobalPlanner* son los siguientes:

- **allow_unknown:** true
- **visualize_potential:** true
- **use_dijkstra:** true
- **use_quadratic:** true
- **use_grid_path:** false

Los parámetros establecidos para el planeador local *dwa_local_planner* son los siguientes:

- **max_vel_trans:** 0.15 *m/s*.
- **min_vel_trans:** -0.15 *m/s*.
- **max_vel_x:** 0.15 *m/s*.
- **min_vel_x:** -0.15 *m/s*.

- **max_vel_y:** -0.15 m/s .
- **min_vel_y:** -0.15 m/s .
- **max_vel_theta:** 0 rad/s .
- **min_vel_theta:** 0 rad/s .
- **acc_lim_x:** 2.5 m/s^2 .
- **acc_lim_y:** 2.5 m/s^2 .
- **acc_lim_theta:** 0 rad/s^2 .
- **xy_goal_tolerance:** 0.05
- **yaw_goal_tolerance:** 0.5
- **path_distance_bias:** 32
- **goal_distance_bias:** 20

Los parámetros establecidos para el mapa de costes de los planificadores antes mencionados son los siguientes:

- **global_frame:** map
- **robot_base_frame:** omni_base_footprint
- **update_frequency:** 5.0 Hz
- **publish_frequency:** 0.0 Hz
- **transform_tolerance:** 1.9 s
- **inflation_layer:** 0.35

- **static_map:** true
- **map_type:** costmap
- **rolling_window:** false
- **width:** 10 *m*
- **height:** 10 *m*
- **resolution:** 0.02 *m/celda*

5.2 Resultados experimentales

5.2.1 Control de velocidad LQG

Si bien se desea llegar a una posición específica dentro del mapa, el planeamiento de trayectoria genera velocidades traslacionales y rotacionales para lograr llegar a dicha posición. Por tal motivo, se implementó un controlador LQG para cada motor DC que acciona cada rueda utilizando las mismas ganancias presentadas en las ecuaciones (5.1), (5.2) y (5.3), cuyos resultados se pueden observar en la Tabla 5.4, donde T_s representa el tiempo de establecimiento, e_{ss} representa el error en estado estable y $\%OS$ el porcentaje de sobreimpulso.

$T_s(s)$	e_{ss}	$OS(\%)$
0.525	$\pm 0.04\%$	0

TABLA 5.4: Resultados del controlador LQG.

Debido a que se controlan las ruedas del robot móvil, se requiere de la cinemática inversa presentada en la Sección 3.3 para obtener las velocidades del robot móvil. El análisis de la cinemática es realizado en el microcomputador Raspberry Pi 3B+, además, este se encarga de comandar las velocidades del robot móvil. Para efectos de prueba,

se comandaron velocidades por cada eje; es decir, x , y y θ , durante aproximadamente 9 segundos.

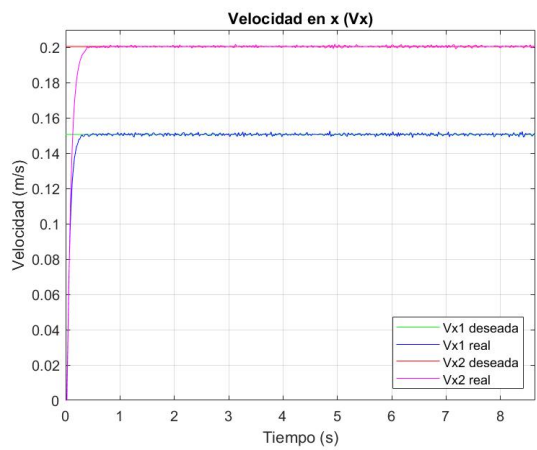
En la Fig. 5.11 se pueden observar los resultados de dichas pruebas. Para la velocidad en x y en y se realizaron 2 pruebas partiendo de una velocidad inicial de 0 m/s y para la velocidad en θ se realizó una prueba partiendo de una velocidad inicial de 4.7 rad/s.

5.2.2 Navegación semiautónoma

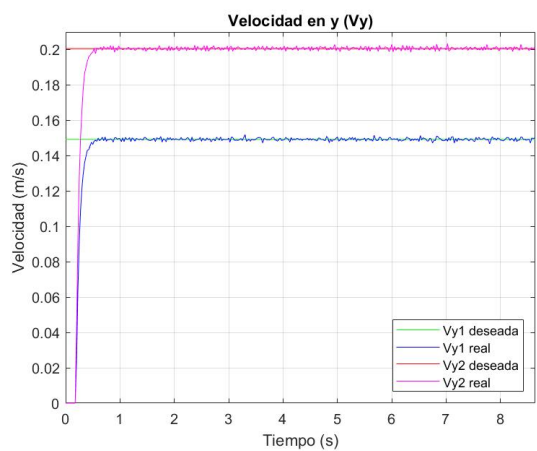
Para el planeamiento de trayectoria, se trabajó con el entorno de trabajo real del robot. El mapa fue diseñado con píxeles en blanco y negro para facilitar el proceso de optimización. La placa Arduino Uno será la encargada de leer la información de los pulsos que genera el encoder y esta transmite la información al microcomputador Raspberry Pi3, el cual hará los cálculos necesarios para publicar la data de la odometría en ROS. Asimismo, el algoritmo *move_base* y *fake_localization* es implementado en el microcomputador Raspberry Pi3.

Para comprobar el funcionamiento de la generación de trayectoria en conjunto con el control cinemático, se dispuso a probar el algoritmo de optimización y navegación sin considerar el peso; es decir, se colocó el robot móvil sobre una superficie de tal manera que las ruedas puedan moverse libremente. Tal como se puede observar en la Fig. 5.12, la trayectoria real sigue fielmente a la trayectoria deseada con 0 % de error en x e y .

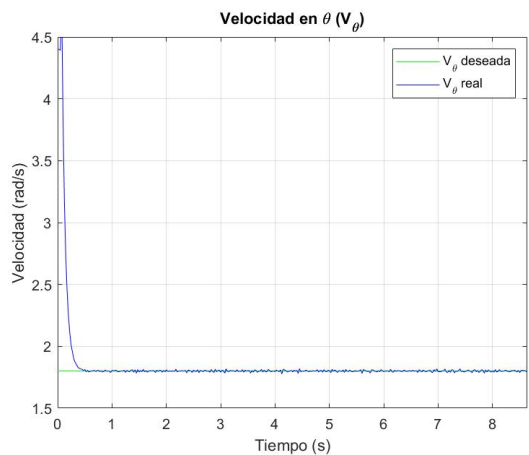
A continuación se realizó la prueba de la generación de trayectoria considerando el peso del robot; es decir, se colocó sobre el suelo. En la Fig. 5.13 se puede observar la trayectoria deseada versus la trayectoria obtenida. En la Fig. 5.14 (a), se observa que el error mínimo obtenido en x es 0 %, el error máximo es 104.9 %, siendo este un valor atípico, y el error medio es 14.9 %. En la Fig. 5.14 (b), se observa que el error mínimo obtenido en y es 0 %, el error máximo es 5.8 % y el error medio es 2.3 %.



(a)



(b)



(c)

FIGURA 5.11: Velocidades del robot móvil. (a) Control de velocidad en el eje x . (b) Control de velocidad lineal en el eje y . (c) Control de velocidad angular θ .

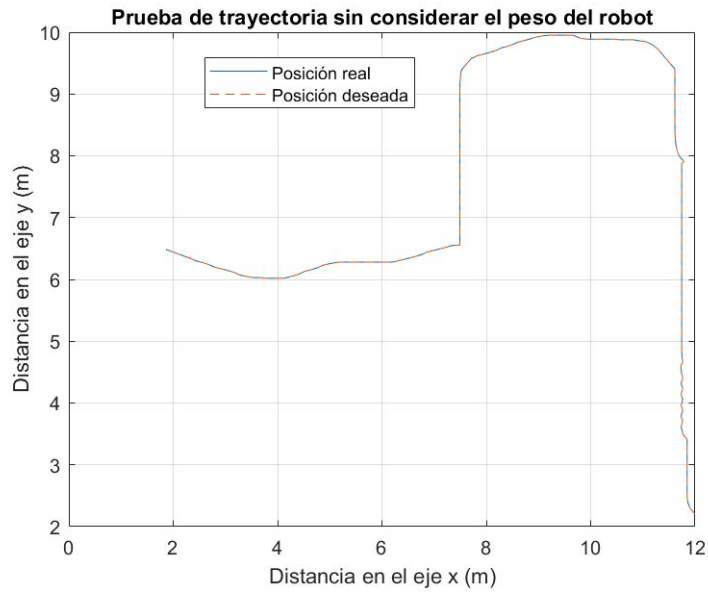


FIGURA 5.12: Trayectoria deseada VS. trayectoria obtenida sin considerar el peso del robot móvil.

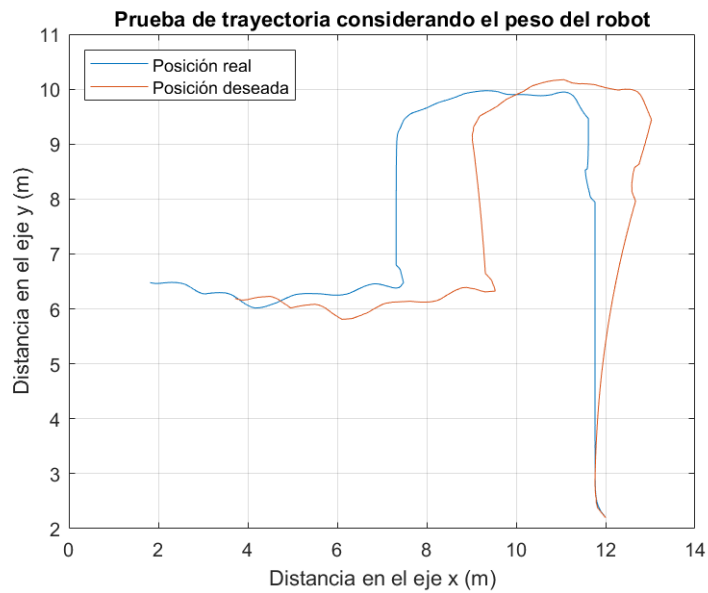
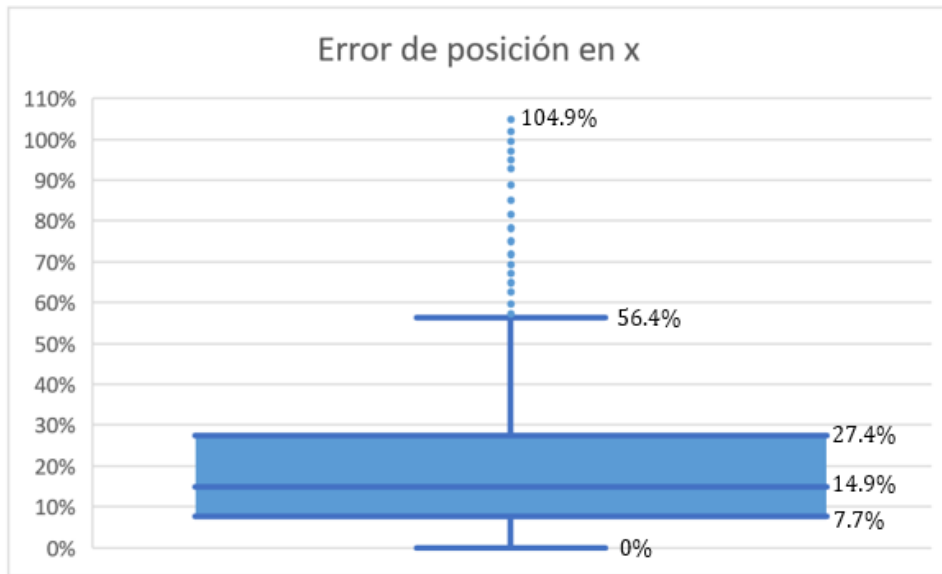
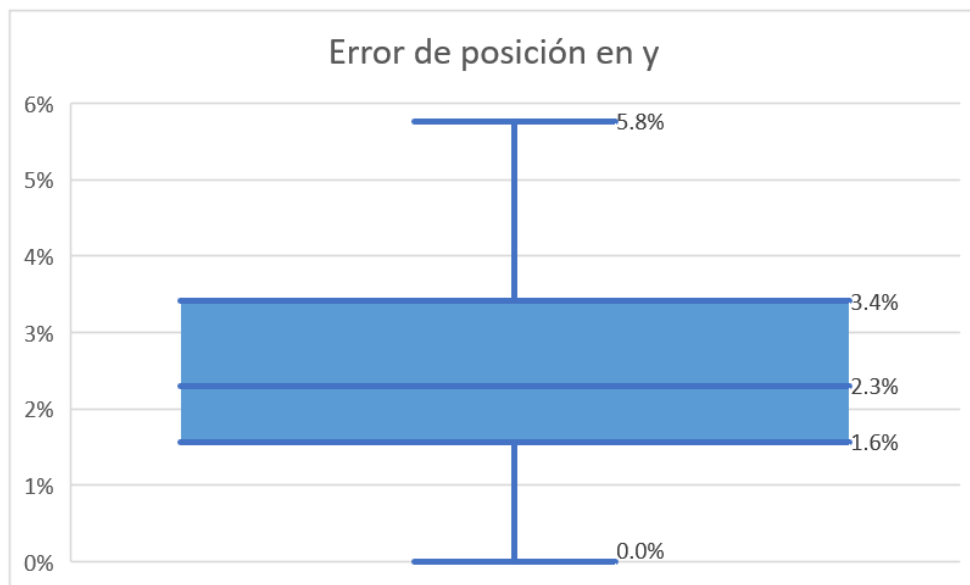


FIGURA 5.13: Trayectoria deseada VS. trayectoria obtenida considerando el peso del robot móvil.



(a)



(b)

FIGURA 5.14: Diagrama de caja del error de posición (a) en x. (b) en y.

Es claro que el resultado se ve afectado debido a que la única data recibida para estimar la posición del robot es odométrica, como también el peso, la inercia y la fricción afectan el comportamiento del robot móvil. Esto sucede debido a que se utiliza un control cinemático, el cual no considera los puntos mencionados anteriormente.

5.2.3 Análisis colorimétrico

Para corroborar que la optimización lineal para la obtención de los waypoints es correcta, se utiliza un análisis de dosis suministrada a las superficies mediante colorimetría. Se colocaron 27 puntos dentro del ambiente de trabajo, cuya distribución se puede observar en la Fig. 5.15 y los resultados obtenidos se pueden observar en la Tabla 5.5, aplicado la ecuación (4.4).

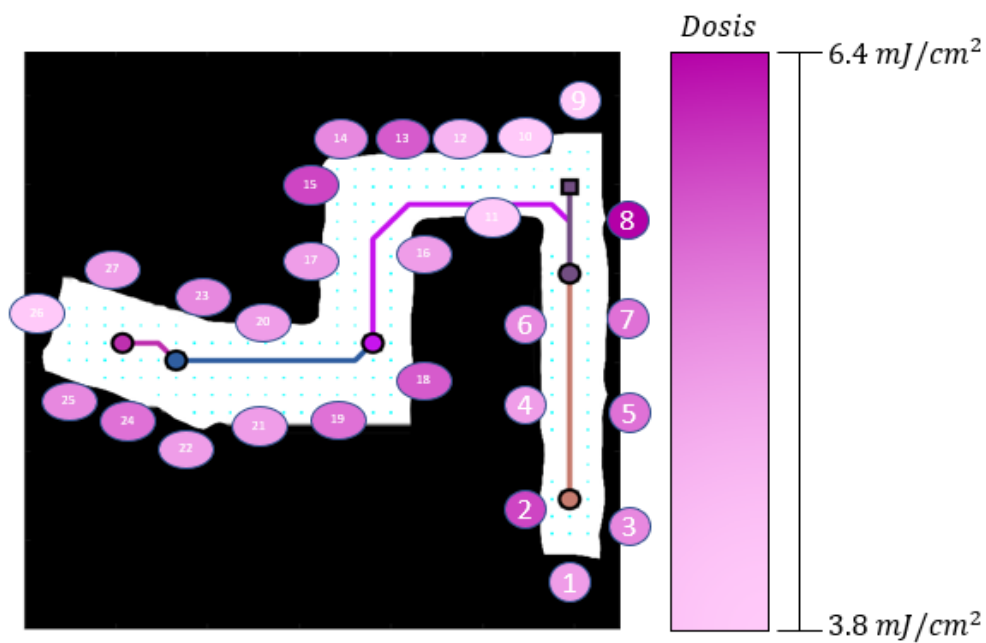


FIGURA 5.15: Ubicación de dosímetros en el ambiente de trabajo

	Valor matriz G	Dosis recibida
Punto 1	239	4.37 mJ/cm ²
Punto 2	235	5.52 mJ/cm ²
Punto 3	238	4.65 mJ/cm ²
Punto 4	239	4.37 mJ/cm ²
Punto 5	237	4.94 mJ/cm ²
Punto 6	238	4.65 mJ/cm ²
Punto 7	237	4.94 mJ/cm ²
Punto 8	232	6.40 mJ/cm ²
Punto 9	241	3.80 mJ/cm ²
Punto 10	241	3.80 mJ/cm ²
Punto 11	241	3.80 mJ/cm ²
Punto 12	240	4.08 mJ/cm ²
Punto 13	236	5.23 mJ/cm ²
Punto 14	239	4.37 mJ/cm ²
Punto 15	235	5.52 mJ/cm ²
Punto 16	239	4.37 mJ/cm ²
Punto 17	239	4.37 mJ/cm ²
Punto 18	236	5.23 mJ/cm ²
Punto 19	237	4.94 mJ/cm ²
Punto 20	240	4.08 mJ/cm ²
Punto 21	239	4.37 mJ/cm ²
Punto 22	239	4.37 mJ/cm ²
Punto 23	238	4.65 mJ/cm ²
Punto 24	237	4.94 mJ/cm ²
Punto 25	238	4.65 mJ/cm ²
Punto 26	241	3.80 mJ/cm ²
Punto 27	239	4.37 mJ/cm ²

TABLA 5.5: Resultados del análisis colorimétrico

Con la data expresada en la Tabla 5.5, se evidencia que el algoritmo de optimización lineal para la desinfección de superficies para la inactivación del virus Sars-CoV-2 en un 99.9 % se logra de manera satisfactoria, pues todos los puntos cuentan con una dosis mayor a 3.7 mJ/cm².

CONCLUSIONES

En la presente tesis, se propuso un algoritmo de optimización de puntos de desinfección, el cual fue implementado en un robot móvil de configuración omnidireccional, mostrando la trayectoria de desinfección que el robot realiza. El algoritmo de optimización para la obtención de los waypoints, utilizando como mapa el sótano de la UTEC, obtuvo 6 waypoints para garantizar la desinfección de todas las superficies con un tiempo de 68 minutos aproximadamente.

Asimismo, se logró implementar experimentalmente la metodología de control LQG con la finalidad de controlar las velocidades generadas por la generación de trayectoria para lograr llegar al waypoint deseado. Dicho controlador en simulación logró un tiempo de establecimiento de 0.175 segundos, un porcentaje nulo de sobreimpulso y un error en estado estable de 0. En la implementación experimental se obtuvo un tiempo en estado estable de 0.525 segundos, un porcentaje nulo de sobreimpulso y un error en estado estable de 0.04 %.

El Arduino UNO, donde se implementó el bloque de control comandado por las velocidades que le envía el Raspberry Pi3, tiene recursos limitados en términos de memoria RAM, capacidad de procesamiento y almacenamiento. Si el Arduino UNO está sobrecargado o si está tratando de manejar una cantidad excesiva de datos, podría experimentar retrasos que podrían generar pequeños errores. En ese sentido, como trabajo futuro se recomienda tomar en cuenta los errores del hardware utilizado sobre lo ya mostrado en resultados.

Utilizando el algoritmo de Dijkstra y DWA se logró planificar una trayectoria que ayuda al robot llegar a una posición deseada con error nulo de porcentaje (0 %) en x e y sin considerar el peso del robot. Sin embargo, considerando el peso del robot se obtuvo un error medio de 14.9 % en x y 2.3 % en y . Por tal motivo, se ubicó al robot en los waypoints obtenidos en el algoritmo de optimización.

Para usar el paquete de navegación *move_base* hay que tener en cuenta que requiere de sensores exteroceptivos. En el presente trabajo, se colocó un Lidar dentro del diseño en *xacro* mas no uno en físico, lo cual no causó ningún problema de funcionamiento. Sin embargo, se evidenció que el proceso de localización del robot móvil mediante odometría únicamente no es lo ideal y el resultado mejoraría con el uso de un sensor exteroceptivo como por ejemplo un LiDAR.

Por otro lado, en el presente trabajo se utilizó un controlador cinemático; sin embargo, el resultado presentó errores significativamente grandes debido a que el controlador no considera masa, inercias, fricción y, además, localizar el robot solo con data odométrica presenta serias limitaciones.

Asimismo, para la sincronización de las transformadas y posterior navegación es necesario contar con un *clock* que se puede obtener con conexión a internet o de manera física. En este caso, se requirió de internet para lograr la implementación de la trayectoria del robot móvil.

En términos de desempeño del código para la obtención de waypoints, el tiempo que tarda el programa en ejecutarse para encontrar la solución al problema de optimización es 0.79 segundos y el tiempo total que tarda en discretizar el mapa, identificar puntos libres y obstáculos, y dar solución al problema de optimización es 7.99 segundos.

Comparando lo desarrollado en la presente tesis con sus pares comerciales e investigaciones presentadas en la Sección 2.1, todos los trabajos definen los puntos, sea de manera manual, teleoperada o semi-autónoma, y el tiempo de desinfección mediante el

criterio del operador, además de no validar la desinfección. En el caso del robot *Hyper Light UV-C*, si bien el posicionamiento y el tiempo de desinfección es definido por el operador, la recomendación es de 5 a 15 minutos por punto de desinfección debido a que el enfoque es su tecnología patentada de reflector protector; sin embargo, se evidenció que su desinfección se alcanzó en el 82.5 % de sus puntos de muestreo a diferencia del presente trabajo. Es importante recordar que los datos obtenidos en esta tesis son para un espacio diferente al utilizado por las soluciones comerciales.

Finalmente, el objetivo general de la tesis, dosificar de UV-C las superficies para su desinfección, fue alcanzado satisfactoriamente dado a que los 27 dosímetros colorimétricos, colocados en el sótano de la UTEC, obtuvieron en su totalidad una dosis mayor a 3.7 mJ/cm^2 , lo cual asegura una inactivación del virus Sars-CoV-2 en un 99.9 % en todos sus puntos.

REFERENCIAS BIBLIOGRÁFICAS

- [1] World Health Organization, “Preventing and mitigating covid-19 at work: policy brief,” no. 19, Mayo 2021.
- [2] E. Asher et al., “Optimal covid-19 infection spread under low temperature, dry air, and low uv radiation,” *New Journal of Physics*, vol. 23, Marzo 2021.
- [3] N. Briones et al., “Luz ultravioleta para desinfección en áreas de salud, frente alcovid-19. revisión de literatura.” *OACTIVA*, vol. 5, no. 3, pp. 111–118, Septiembre 2020.
- [4] G. Byrns, “Alternative room disinfection modalities – pros and cons,” *Springer*, Julio 2014.
- [5] M. Guettari et al., “Uvc disinfection robot,” *Environmental Science and Pollution Research*, 2020.
- [6] M. Raeiszadeh y B. Ade, “A critical review on ultraviolet disinfection systems against covid-19 outbreak: Applicability, validation, and safety considerations,” *ACS Photonics*, 2020.
- [7] A. Füzsl et al., “The use of a uv-c disinfection robot in the routine cleaning process: a field study in an academic hospital,” *Antimicrobial Resistance and Infection Control*, vol. 10, 2021.
- [8] G. Moez et al., “Uvc disinfection robot,” *Environmental Science and Pollution Research*, vol. 98, pp. 434–441, Agosto 2021.

- [9] M. Lindblad et al., “Ultraviolet-c decontamination of a hospital room: Amount of uv light needed,” *Burns : journal of the International Society for Burn Injuries*, 2020.
- [10] J. Conroy et al., “Robot development and path planning for indoor ultraviolet light disinfection,” 04 2021.
- [11] L. Tiseni et al., “Uv-c mobile robots with optimized path planning: Algorithm design and on-field measurements to improve surface disinfection against sars-cov-2,” *IEEE Robotics Automation Magazine*, vol. 28, no. 1, pp. 59–70, 2021.
- [12] B. Casini et al., “Implementation of an environmental cleaning protocol in hospital critical areas using a uv-c disinfection robot,” *Environmental Research and Public Health*, 2023.
- [13] Mediland Enterprise Corporation, “Hyper light disinfection robot.”
- [14] L. Tan et al., “Air and surface contamination by sars-cov-2 virus in a tertiary hospital in wuhan, china,” *International Journal of Infectious Diseases*, 07 2020.
- [15] World Health Organization, “Brote de enfermedad por coronavirus (covid-19): orientaciones para el público,” Mayo 2022.
- [16] Pavan B. Mandavkar, “Coronavirus: Basic information and precautionary measures,” *SSRN*, 04 2020.
- [17] P. Carling et al., “Improving environmental hygiene in 27 intensive care units to decrease multidrug-resistant bacterial transmission,” *Crit Care Med*, vol. 38, no. 4, pp. 1054–1059, Abril 2010.
- [18] T. Cutler y J. Zimmerman, “Ultraviolet irradiation and the mechanisms underlying its inactivation of infectious agents,” *Animal Health Research Reviews*, vol. 12, no. 1, pp. 15–23, Junio 2011.

- [19] A. Bianco et al., “Uv-c irradiation is highly effective in inactivating and inhibiting sars-cov-2 replication,” *Scientific Reports*, Marzo 2021.
- [20] W. Wells, “Air disinfection in day schools,” *Am J Public Health Nations Health*, Diciembre 1943.
- [21] J. Cadnum et al., “Next-generation uv: Evaluation of a robotic ultraviolet-c room disinfection device,” *Open Forum Infectious Diseases*, Octubre 2017.
- [22] A. Begić, *Application of Service Robots for Disinfection in Medical Institutions*, 2018, pp. 1056–1065.
- [23] J. Cornejo et al., “Trajectory tracking control of a differential wheeled mobile robot: a polar coordinates control and lqr comparison,” in *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, 2018, pp. 1–4.
- [24] S. Morales et al., “Lqr trajectory tracking control of an omnidirectional wheeled mobile robot,” in *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, 2018, pp. 1–5.
- [25] The American Water Works Association (AWWA) y The American Society of Civil Engineers (ASCE), *Water Treatment Plant Design, Fifth Edition*. McGraw-Hill, 2012.
- [26] T. Asano et al., “Disinfection processes for water reuse applications,” in *Water Reuse: Issues, Technologies, and Applications*, 2007.
- [27] V. Corman, “Aerosol and surface distribution of severe acute respiratory syndrome coronavirus 2 in hospital wards, wuhan, china, 2020,” *Emerging infectious diseases*, Enero 2020.
- [28] C. Pasquarella, “Detection of sars-cov-2 on hospital surfaces,” *Acta Biomed.*, Julio 2020.

- [29] B. Siciliano et al., "Robotics: Modeling, planning, and control," *Springer*, Diciembre 2009.
- [30] B. Siciliano et al., "Springer handbook of robotics," *Springer*, 2008.
- [31] K. Lynch et al., "Modern robotics: Mechanics, planning, and control," 2017.
- [32] Robotnik, "Robot móvil summit-xl." [Online]. Available: robotnik.eu/es/productos/robots-moviles/summit-xl-es/
- [33] V. Robotics, "Ruedas omnidireccionales." [Online]. Available: www.vexrobotics.com.mx
- [34] C. Canudas et al., "Theory of robot control," *Springer*, 1996.
- [35] R. Siegwart et al., "Introduction to autonomous mobile robots," *MIT Press*, 2004.
- [36] I. C. et al., "Modelo matemático del motor de corriente directa," 02 2017.
- [37] M. Fadali y A. Visiol, "Digital control engineering : analysis and design," *Elsevier*, 2013.
- [38] K. Ogata, "Designing linear control systems with matlab," 1993.
- [39] N. Nise, "Control systems engineering," *Wiley*, 2014.
- [40] K. Ogata, "Modern control engineering," *Pearson*, 2010.
- [41] D. Xue et al., "Linear feedback control: Analysis and design with matlab," *SIAM*, 2007.
- [42] S. Se et al., "Local and global localization for mobile robots using visual landmarks," *IEEE*, 2001.
- [43] S. Thrun et al., "Probabilistic robotics," 2005.

- [44] H. Zhang et al., "Path planning for the mobile robot: A review," *Symmetry*, Octubre 2018.
- [45] L. Chang et al., "Multi-robot formation control in unknown environment based on improved dwa," *Control Decis.*, 2021.
- [46] J. Sun et al., "Smart obstacle avoidance using a danger index for a dynamic environment," *MDPI*, 2019.
- [47] F. Gross, "Frontiers in antennas: Next generation design and engineering," *McGraw-Hill*, 2011.
- [48] D. Green y M. Southard, "Perry's chemical engineers' handbook," *McGraw-Hill*, 2019.
- [49] D. Luenberger y Y. Ye, "Linear and nonlinear programming," *Springer*, 2008.
- [50] T. M. Inc., "Linear programming algorithms," Natick, Massachusetts, United States, 2023. [Online]. Available: <https://la.mathworks.com/help/optim/ug/linear-programming-algorithms.html#brnpenw>

ANEXOS

Los archivos que fueron usados para el diseño del controlador LQG y la obtención de los waypoints con programación lineal mediante el software Matlab y los archivos que fueron implementados en el Arduino UNO y la Raspberry Pi mediante el lenguaje de programación Python se encuentran en el siguiente repositorio de GitHub: [Repositorio](#). Este repositorio contiene 6 archivos los cuales se describen a continuación:

- **Control_Cinematica.m**: Este archivo de Matlab contiene el diseño del controlador LQG con efecto integral mediante el procedimiento descrito anteriormente. Se comprueba la observabilidad y controlabilidad del sistema, y la parte final del código grafica los resultados de velocidad de las ruedas del robot, la velocidad del robot y la señal de control. Asimismo, se grafica el mapa de polos para evaluar la estabilidad del sistema a lazo cerrado.
- **test5.m**: Este archivo de Matlab contiene el diseño del algoritmo de optimización mediante programación lineal. Al inicio del código se discretiza el mapa y se calcula la irradiación que cada posible waypoint le da a cada obstáculo. Finalmente, se grafican los waypoints en el mapa ingresado.
- **CreateGridGraph.m**: Este archivo de Matlab contiene la función que permite discretizar el mapa de entrada (.png, .jpg, .bmp) en **test5.m** en un gráfico de cuadrícula (celdas de cuadrícula) en forma de matriz dispersa.

- **Checkobs.m:** Este archivo de Matlab contiene la función que permite identificar si un punto es libre (color blanco) o es un obstáculo (color negro) en el mapa ingresado en **test5.m**.
- **Pwaypoint.m:** Este archivo de Matlab contiene la función que permite identificar las coordenadas de los posibles waypoints del mapa ingresado en **test5.m** y la distancia que existe en sus 8 celdas vecinas.
- **PPPwaypoint.m:** Este archivo de Matlab contiene la función que permite identificar las coordenadas de los obstáculos del mapa ingresado en **test5.m** y la distancia que existe en sus 8 celdas vecinas.
- **ShowPath.m:** Este archivo de Matlab contiene la función que grafica los waypoints en el map ingresado en **test5.m**.
- **rueda1.ino:** Este archivo de Arduino inicia con la definición de pines de entrada y salida de la placa Arduino UNO a utilizar para implementar el controlador LQG en el motor de la rueda 1 del robot móvil omnidireccional.
- **rueda2.ino:** Este archivo de Arduino inicia con la definición de pines de entrada y salida de la placa Arduino UNO a utilizar para implementar el controlador LQG en el motor de la rueda 2 del robot móvil omnidireccional.
- **rueda3.ino:** Este archivo de Arduino inicia con la definición de pines de entrada y salida de la placa Arduino UNO a utilizar para implementar el controlador LQG en el motor de la rueda 3 del robot móvil omnidireccional.
- **rueda4.ino:** Este archivo de Arduino inicia con la definición de pines de entrada y salida de la placa Arduino UNO a utilizar para implementar el controlador LQG en el motor de la rueda 4 del robot móvil omnidireccional.
- **/src/omni_description:** Esta carpeta ubicada en el RaspberryPi 3 contiene el modelo del robot, con el fin de hacer las simulaciones en Gazebo.

- **/src/my_robot/maps:** Esta carpeta ubicada en el RaspberryPi 3 contiene el mapa del entorno a trabajar.
- **/src/my_robot/params:** Esta carpeta ubicada en el RaspberryPi 3 contiene los parámetros del planeador global y local, como también los parámetros de sus mapas de costos.
- **/move_base.launch:** Este archivo XML ubicado en el RaspberryPi 3 contiene la inicialización del paquete *move_base* junto a los parámetros necesarios para su ejecución.
- **my_robot_configuration.launch:** Este archivo XML ubicado en el RaspberryPi 3 empieza con la inicialización del mapa y el modelo del robot. En la parte final, se inicializa el paquete *fake_localization* para obtener la posición mediante la odometría del robot.
- **waypoint_nav.py:** Este archivo Python ubicado en el RaspberryPi 3 contiene la serie de waypoints previamente calculados, como también el tiempo de parada en cada uno de los waypoints.