
Desarrollo y evaluación de algoritmos para el mapeo de entornos y generación de trayectorias utilizando sistemas robóticos multi-agente

Fredy Josue Godoy Lucero



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



Desarrollo y evaluación de algoritmos para el mapeo de entornos y generación de trayectorias utilizando sistemas robóticos multi-agente

Trabajo de graduación presentado por Fredy Josue Godoy Lucero para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

(f) _____
Dr. Luis Alberto Rivera Estrada

Tribunal Examinador:

(f) _____
Dr. Luis Alberto Rivera Estrada

(f) _____
(f) _____

Fecha de aprobación:

Lista de figuras	viii
Resumen	ix
Abstract	x
1. Introducción	1
2. Antecedentes	3
2.1. Mapeo de entorno	3
2.2. Aplicaciones de <i>Ant Colony Optimization</i> (ACO)	5
2.3. <i>Particle Swarm Optimization</i> (PSO)	6
2.4. Implementación de algoritmos de inteligencia de enjambre en UVG	6
3. Justificación	11
4. Objetivos	13
5. Alcance	14
6. Marco teórico	15
6.1. <i>Particle Swarm Optimization</i> (PSO)	15
6.2. <i>Ant Colony System</i> (ACS) with <i>Travel Salesman Problem</i> (TSP)	17
6.3. Robots diferenciales	19
6.4. <i>Linear-quadratic regulator</i> (LQR)	20
6.5. Mapeo de entornos con <i>Multi-Stage Optimization</i> (MSO)	21
6.6. Filtro de Kalman extendido (EKF)	21
6.7. Topología de comunicación dinámica autoorganizada (DSCT)	22
6.8. Filtrado de información en múltiples capas (MLIF)	25
6.9. Light Detection and Ranging (LiDAR)	25
6.10. Vehículo diferencial <i>Pololu 3pi 32U4</i>	26
6.11. Entorno de simulación Webots	30

7. Construcción virtual de robot móvil con ruedas y tracción diferencial	31
7.1. Prototipo 1 de base para sensores de distancia	33
7.2. Prototipo 2 de base para sensores de distancia	34
7.3. Sensores de distancia	36
7.3.1. Sensor de distancia	36
7.3.2. Consideración para sensor de distancia tipo sonar	39
7.4. Otros sensores del vehículo diferencial	40
7.4.1. Sensores de posición de ruedas	40
8. Mundo simulado en Webots	42
9. Mapeo de entorno	45
9.1. Algoritmo de exploración - Prototipo 1	45
9.1.1. Posicionamiento de vehículo	47
9.1.2. Condiciones de giro	52
9.1.3. Resultados usando sensores de distancia tipo sonar	60
9.1.4. Resultados usando sensores de distancia tipo láser	71
9.2. Algoritmo de exploración - Prototipo 2	80
9.2.1. Condiciones de giro	82
10. Planificación de Trayectoria	84
10.1. Algoritmo para generar trayectoria	85
10.2. Resultados con mapa generado previamente	87
10.3. Resultados con mapa estimado del entorno	91
11. Sensores de distancia para aplicaciones en físico	93
11.1. Sensor VL53L0X	93
11.2. Sensor VL53L1X	94
12. Conclusiones	96
13. Recomendaciones	98
14. Bibliografía	99
15. Anexos	101
15.1. Resultados usando sensores de distancia tipo sonar	101
15.2. Resultados usando sensores de distancia tipo láser	109
16. Glosario	117

Lista de figuras

1.	Mapa generado a partir de mediciones del Robat [4].	4
2.	Concepto físico del Robat [4].	4
3.	Ruta generada a partir de las modificaciones sobre ACO [6].	5
4.	Grupo de robots ejecutando trayectoria de desplazamiento [8].	6
5.	Grupo de robots en formación triangular [8].	7
6.	Grupo de Robots E-Puck ejecutando rutina de movimiento [11].	8
7.	Rutina para validar en Webots [2].	9
8.	Mapa implementado en Webots [2].	9
9.	Comparación de mapa 1 generado con metodología síncrona (izquierda) y asíncrona (derecha) [12].	10
10.	Sistema de coordenada de referencia de un robot móvil con ruedas y tracción diferencial. XY muestra el marco de referencia mundial. (x_b, y_b) es la coordenada del centro de masa del robot, $\dot{\mathbf{x}}$ e $\dot{\mathbf{y}}$ son las velocidades de este centro a lo largo de los ejes X_r e Y_r , respectivamente [17].	20
11.	Diagrama de flujo del proceso que sigue EKF [19].	22
12.	Red de comunicación básica con un maestro [19].	24
13.	Proceso de DSCT en el método MSO [19].	24
14.	Proceso de MLIF en el método MSO [19].	25
15.	Funcionamiento LiDAR [20].	26
16.	Pololu 3pi+ 32U4 OLED Robot [1].	27
17.	Características del Pololu 3pi+ 32U4 OLED Robot [1].	29
18.	Dimensiones vista superior del Pololu 3pi+ 32U4 OLED Robot (mm [in]) [1].	29
19.	Mundo simulado en Webots [9].	30
20.	Representación 3D del bumper de Pololu 3pi+.	32
21.	Representación 3D del Pololu 3pi+.	32
22.	Representación 3D de Pololu 3pi+ en el entorno de Webots.	33
23.	Modelo 3D de base para sensores de distancia para Pololu 3pi+.	34
24.	Incorporación de base para sensores de distancia para Pololu 3pi+ en entorno de simulación Webots.	34
25.	Modelo 3D de base para sensores de distancia para Pololu 3pi+.	35

26.	Incorporación del prototipo 2 de la base para sensores de distancia para Pololu 3pi+ en entorno de simulación Webots.	35
27.	Distribución por nombre de sensores de distancia sobre base en modelo diferencial.	36
28.	Ejemplo de parámetros para un sensor de distancia [9].	37
29.	Respuesta del sensor vs. distancia del obstáculo [9].	37
30.	Distribución de rayos en sensores de distancia [9].	38
31.	Apertura de 0.05 m del sensor láser.	39
32.	Apertura de 0.01 m del sensor láser.	39
33.	Respuesta del sensor sonar con base al ángulo de apertura [9].	40
34.	Articulación con un solo eje de rotación [9].	41
35.	Mundo simulado en Webots.	43
36.	Mundo simulado en Webots.	43
37.	Secciones del mundo simulado en Webots.	44
38.	Diagrama de flujo general de algoritmo de exploración prototipo 1.	46
39.	Cambio de ángulo de la rueda respecto a un desplazamiento.	47
40.	Trayectoria de exploración en entorno de simulación Webots, 1 segmento de linea recta con 45° de rotación	48
41.	Trayectoria de exploración estimada, 1 segmento de linea recta con 45° de rotación, unidades en m.	49
42.	Trayectoria de exploración obtenida por GPS, 1 segmento de linea recta con 45° de rotación, unidades en m.	49
43.	Error acumulado en trayectoria de exploración, 1 segmento de linea recta con 45° de rotación.	51
44.	Diagrama de flujo para considerar un cambio de giro en algoritmo de exploración.	53
45.	Valor de sensor frontal mayor que lateral.	54
46.	Valor de sensor frontal menor que lateral.	55
47.	Valor de sensor trasero mayor a todos los demás sensores.	56
48.	Espacio del entorno de simulación en Webots para realizar trayectoria de exploración con cambio de orientación	57
49.	Trayectoria estimada de exploración con cambio de orientación del vehículo, unidades en mt	58
50.	Trayectoria de exploración con cambio de orientación del vehículo, obtenida con GPS, unidades en mt.	58
51.	Error acumulado en trayectoria de exploración con cambio de orientación . . .	59
52.	Posición inicial de vehículo en el entorno de simulación.	61
53.	Trayectoria de exploración usando sensores de distancia tipo sonar con simulación de 5 min.	62
54.	Error acumulado en trayectoria de exploración usando sensor de distancia tipo sonar en simulación de 5 min.	63
55.	Trayectoria de exploración usando sensores de distancia tipo sonar con simulación de 60 min.	64
56.	Error acumulado en trayectoria de exploración usando sensor de distancia tipo sonar en simulación de 60 min.	65
57.	Posición inicial del vehículo en la Sección 6 del entorno de simulación.	66

58.	Trayectoria de exploración segunda posición usando sensores de distancia tipo sonar con simulación de 5 min.	67
59.	Error acumulado en trayectoria de exploración iniciando en sección 5 del entorno simulado, usando sensor de distancia tipo sonar en simulación de 5 min.	68
60.	Colisión de vehículo con obstáculo.	69
61.	Trayectoria de exploración segunda posición usando sensores de distancia tipo sonar con simulación de 60 min.	70
62.	Error acumulado en trayectoria de exploración iniciando en sección 5 del entorno simulado, usando sensor de distancia tipo sonar en simulación de 60 min.	71
63.	Trayectoria de exploración usando sensores de distancia tipo láser con simulación de 5 min.	73
64.	Error acumulado en trayectoria de exploración usando sensor de distancia tipo sonar en simulación de 5 min.	74
65.	Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo láser con simulación de 60 min.	75
66.	Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo láser en simulación de 60 min.	76
67.	Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 5 min.	77
68.	Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 5 min.	78
69.	Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 60 min.	79
70.	Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 60 min.	80
71.	Diagrama de flujo general de algoritmo de exploración prototipo 2.	81
72.	Diagrama de flujo para considerar un cambio de giro en algoritmo de exploración prototipo 2.	83
73.	Espacio de trabajo con ruta valida representado en 2D.	86
74.	Espacio de trabajo con ruta valida representado en 3D.	87
75.	Espacio de trabajo 20×20 con 0 % de obstáculos.	88
76.	Espacio de trabajo 20×20 con 10 % de obstáculos.	89
77.	Espacio de trabajo 20×20 con 20 % de obstáculos, ruta valida próxima al objetivo.	90
78.	Espacio de trabajo 20×20 con 20 % de obstáculos.	90
79.	Espacio de trabajo estimado en exploración de 10 minutos, con objetivo de trayectoria posición inicial en sección 1 del mapa.	91
80.	Espacio de trabajo estimado en exploración de 10 minutos, con objetivo de trayectoria posición inicial en sección 6 del mapa.	92
81.	Modulo sensor de distancia VL53L0X [21].	94
82.	Modulo sensor de distancia VL53L1X [22].	95
83.	Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo sonar con simulación de 5 min.	101

84.	Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo sonar en simulación de 5 min.	102
85.	Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo sonar con simulación de 60 min.	103
86.	Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo sonar en simulación de 60 min.	104
87.	Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo sonar con simulación de 5 min.	105
88.	Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo sonar en simulación de 5 min.	106
89.	Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo sonar con simulación de 60 min.	107
90.	Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo sonar en simulación de 60 min.	108
91.	Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo láser con simulación de 5 min.	109
92.	Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo láser en simulación de 5 min.	110
93.	Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo láser con simulación de 60 min.	111
94.	Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo láser en simulación de 60 min.	112
95.	Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 5 min.	113
96.	Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 5 min.	114
97.	Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 60 min.	115
98.	Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 60 min.	116

Resumen

Este trabajo de investigación se enfocó en la creación y combinación de tres algoritmos principales relacionados con la navegación autónoma de vehículos de tracción diferencial. Los tres algoritmos principales de este estudio comprendieron la exploración de entornos, el mapeo en dos dimensiones de los entornos y la generación óptima de trayectorias. Estos algoritmos fueron específicamente diseñados y adaptados para su implementación en vehículos móviles de tracción diferencial.

La motivación principal detrás de esta investigación es la necesidad de comprender cómo un vehículo de tracción diferencial puede interactuar de manera efectiva en un entorno desconocido. Se asumió que el vehículo está equipado únicamente con sensores de distancia colocados estratégicamente en posiciones específicas, sensores para determinar la posición de las ruedas y una brújula. Además, se consideró un sistema de posicionamiento respecto a entorno de simulación para el vehículo, únicamente con el propósito de comparar y validar la posición estimada.

Para lograr que el vehículo interactuara con éxito en su entorno, se realizaron diferentes iteraciones de los tres algoritmos principales utilizando dos tipos diferentes de sensores de distancia en el vehículo. Los resultados de estas iteraciones permitieron determinar la opción óptima de sensor de distancia para aplicar en los algoritmos desarrollados.

La validación de los algoritmos desarrollados se llevó a cabo mediante simulaciones computarizadas, en un entorno que comprende entornos con diferentes características. Durante este proceso, se evaluó la precisión de los algoritmos, teniendo en cuenta factores como el tiempo de simulación y la posición inicial del vehículo. Estas pruebas proporcionaron una comprensión más profunda de cómo los algoritmos se desempeñan en condiciones diversas y cómo pueden optimizarse para aplicaciones prácticas en la navegación autónoma de vehículos de tracción diferencial.

Abstract

This research work focused on the creation and combination of three main algorithms related to the autonomous navigation of differential wheeled robot. The three main algorithms in this study comprised environment exploration, two-dimensional mapping of environments, and optimal trajectory generation. These algorithms were specifically designed and adapted for implementation in differential wheeled robot.

The primary motivation behind this research is the need to understand how a differential wheeled robot can interact effectively in an unknown environment. It was assumed that the robot is equipped only with distance sensors strategically placed in specific positions, sensors for determining the position of the wheels and a compass. In addition, a positioning system was considered with respect to the simulation environment for the wheeled robot, only for the purpose of comparing and validating the estimated position.

To get the robot to successfully interact with its environment, different iterations of the three main algorithms were performed using two different types of distance sensors on the robot. The results of these iterations made it possible to determine the optimal distance sensor option to apply in the developed algorithms.

The validation of the developed algorithms was carried out through computer simulations, in an environment that includes spaces with different characteristics. During this process, the accuracy of the algorithms was evaluated, taking into account factors such as simulation time and the initial position of the robot. These tests provided a deeper understanding of how the algorithms perform under diverse conditions and how they can be optimized for practical applications in autonomous navigation of differential wheeled robot.

CAPÍTULO 1

Introducción

En este trabajo se aborda el tema de la navegación autónoma de sistemas robóticos, específicamente de vehículos con tracción diferencial. El enfoque principal se centra en el desarrollo y evaluación de algoritmos que permitan la interacción efectiva de estos sistemas en entornos desconocidos. Esta necesidad se vuelve cada vez más relevante a medida que la robótica desempeña un papel crucial en una variedad de aplicaciones, desde la exploración de entornos peligrosos hasta la logística de almacenes.

La problemática que motiva esta investigación radica en los desafíos inherentes a la navegación autónoma. En entornos desconocidos, los robots deben tomar decisiones para evitar obstáculos, trazar trayectorias eficientes y lograr desplazarse de un punto a otro. Para abordar esta problemática, se plantea la necesidad de desarrollar algoritmos para la exploración y mapeo de entornos, así como la generación de trayectorias.

Uno de los aspectos clave de esta investigación es la elección y evaluación de sensores de distancia. La información precisa sobre el entorno es esencial para que los robots tomen decisiones. Por lo tanto, se investigó y evalúo la efectividad de dos diferentes sensores de distancia y se buscó su integración en robots de tracción diferencial.

En el desarrollo de esta investigación, se construyó un modelo virtual de un vehículo con tracción diferencial basado en el modelo Pololu 3pi+ [1]. Este modelo virtual es un agente con el que se llevaron a cabo pruebas y experimentos.

En la etapa de mapeo del entorno, se implementó un algoritmo de exploración que priorizó tres aspectos fundamentales: el preciso posicionamiento del vehículo, la gestión de las condiciones de giro y la detección de obstáculos. Para alcanzar estos objetivos, se realizaron pruebas tanto con sensores de distancia tipo sonar como sensores láser.

En el transcurso de esta fase, se realizaron pruebas que implicaron diversas configuraciones de sensores. Estas pruebas evaluaron la capacidad de los sensores para determinar con precisión la ubicación del vehículo con respecto al entorno. Además, se analizaron las condiciones de giro que permitirían al vehículo desplazarse de manera eficiente y segura en

el entorno de simulación.

Uno de los resultados críticos obtenidos en estas pruebas fue la identificación de la opción óptima de sensor de distancia. Esta elección se volvió de suma importancia, dado que el sensor seleccionado sería empleado en los demás algoritmos que se desarrollarían en el proyecto. Esta selección aseguró que los algoritmos posteriores operaran con la máxima precisión y eficacia en la percepción del entorno.

En el ámbito de la planificación de trayectorias, se desarrolló un algoritmo específico que permite la generación de trayectorias utilizando el mapa previamente generado del entorno. Asimismo, basándose en la trayectoria calculada, el vehículo tiene la capacidad de seguir dicha trayectoria de manera precisa.

Finalmente, se evaluaron opciones de sensores para futuras implementaciones de los algoritmos desarrollados en sistemas robóticos físicos. La elección adecuada de sensores es crucial para garantizar el éxito de las soluciones propuestas en aplicaciones del mundo real.

CAPÍTULO 2

Antecedentes

El mapeo de entorno es una tarea fundamental para que los sistemas de robots multi-agente puedan operar de manera eficiente y segura. El mapeo del entorno implica la creación de representaciones precisas y actualizadas del espacio en el que los robots se desplazan, lo cual es esencial para la planificación de rutas y el posicionamiento adecuado de los robots. El desarrollo de este trabajo implica la integración de técnicas de percepción del entorno, procesamiento de datos, algoritmos de mapeo, algoritmos para obtener rutas de desplazamiento y algoritmos de posicionamiento, esto para aplicarlo en sistemas de robots multi-agente [2].

2.1. Mapeo de entorno

En [3] se describe un algoritmo en tiempo real para el mapeo tridimensional en robots móviles que tiene aplicaciones en el ámbito de mapeo multi-robot. Se utilizan técnicas de estimación probabilística para fusionar información de sensores y construir un mapa del entorno en tiempo real. El algoritmo generado permite que los robots móviles se localicen y mapeen simultáneamente en entornos desconocidos y dinámicos, superando desafíos como la incertidumbre de la posición y la presencia de obstáculos en movimiento. El artículo presenta resultados experimentales que demuestran la eficacia y la precisión del algoritmo en diferentes escenarios, lo que lo convierte en una contribución importante en el campo de la robótica y el mapeo de entornos.

En [4] se desarrolla e implementa un robot totalmente autónomo denominado el Robat, con características similares a la ecolocalización de los murciélagos. Se presenta un sistema robótico móvil equipado con micrófonos direccionales y técnicas de procesamiento de señales para la detección y la localización de objetos en el entorno. El Robat también se caracteriza por su capacidad para generar y seguir rutas basadas en la detección de obstáculos y evitar colisiones. El Robat delinea los bordes de los objetos que va encontrando y los clasifica usando redes neuronales, con esto se genera un mapa más completo del entorno. El Robat a diferencia de otros sistemas que implementan sonar, este utiliza únicamente una salida y

dos entradas.

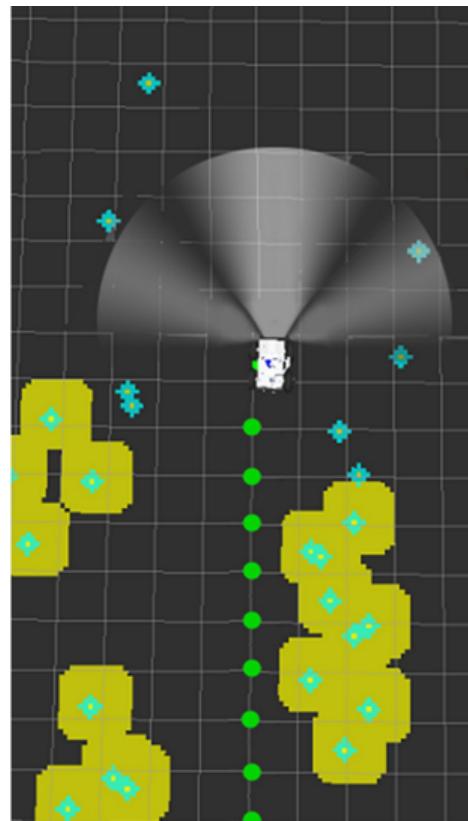


Figura 1: Mapa generado a partir de mediciones del Robat [4].

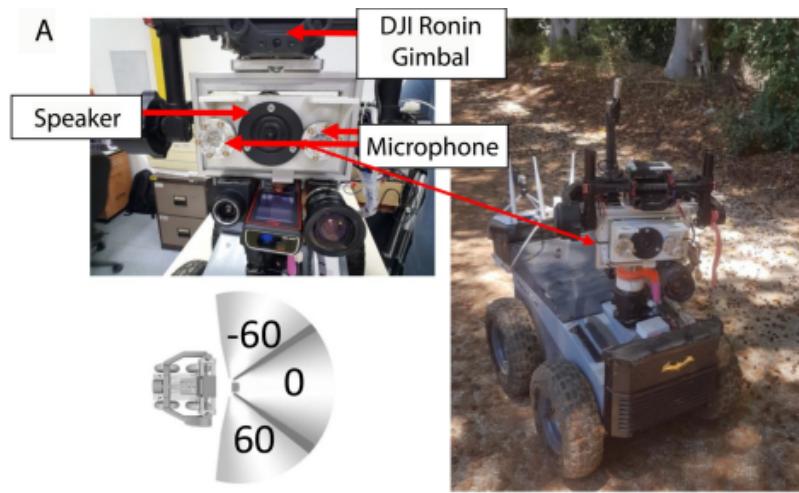


Figura 2: Concepto físico del Robat [4].

2.2. Aplicaciones de *Ant Colony Optimization* (ACO)

En [5] se presentan dos algoritmos basados en ACO, los cuales son aplicados para obtener la estimación de flujo de tráfico en la Ciudad de La Habana. Estos obtuvieron resultados satisfactorios siendo más efectivos que los algoritmos clásicos. El mapa en donde se aplicaron los algoritmos tiene 8000 bordes, estos bordes representan las secciones de carretera, y tiene 2000 nodos, que representan las intersecciones de las carreteras. Los algoritmos utilizados para este problema son el Ant Colony Algorithm for Multi-commodity Problems (ACAM) y el Ant Commodity Routing Algorithm (ACRA). Se determinó que con ACO se obtuvieron buenos resultados con simulaciones en poco tiempo, esto es bueno ya que se aplican a problemas de trayectorias de la vida real.

En [6] se centra en la evasión de obstáculos y en la planificación de trayectorias para robots móviles, con el fin de mejorar los algoritmos de ACO. Los autores identifican ciertas limitaciones en los algoritmos existentes, proponen un algoritmo de ACO. El algoritmo propuesto garantiza que los robots puedan encontrar una trayectoria satisfactoria en presencia de pasillos estrechos. Los resultados de simulación demuestran la efectividad del algoritmo propuesto.

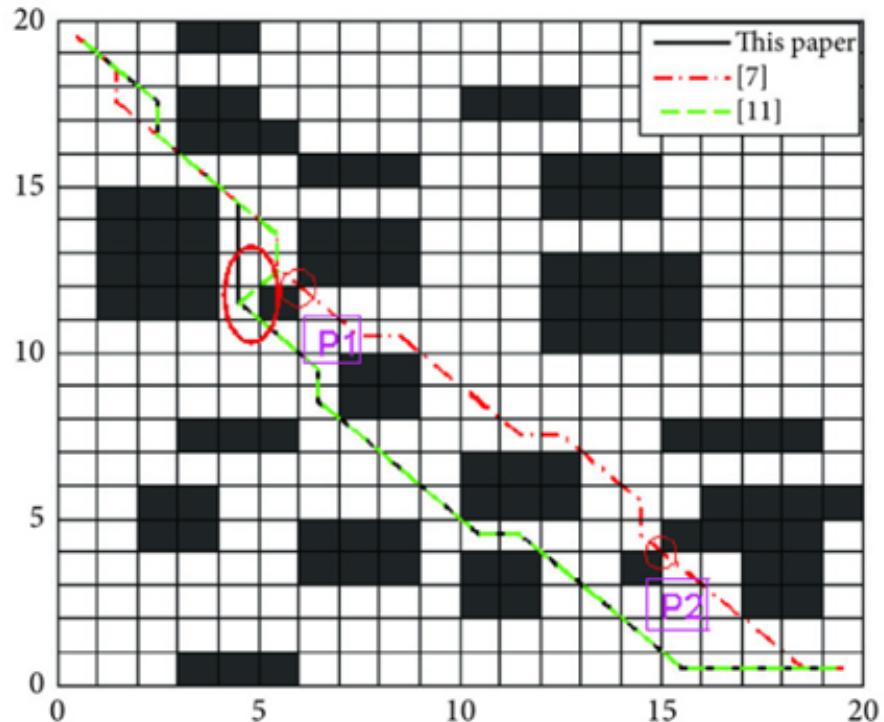


Figura 3: Ruta generada a partir de las modificaciones sobre ACO [6].

2.3. *Particle Swarm Optimization* (PSO)

En [7] se muestra la implementación de PSO, el cual es un método de inteligencia de enjambre. PSO utiliza la evaluación del entorno para compararlo e imitarlo. PSO se basa en el uso de un conjunto de partículas o agentes, que se comunican entre si, que corresponde a estados de un problema de optimización, donde cada partícula se moverá para la búsqueda de una posición óptima.

2.4. Implementación de algoritmos de inteligencia de enjambre en UVG

En el trabajo de investigación [8] se desarrolló e implementó un algoritmo de control para un sistema de robots multi-agente orientado a misiones de búsqueda, basado en los métodos teóricos de grafos y control moderno. En este trabajo se concluyó que la combinación del control de formación y el control contra colisiones en una sola función racional permite que los agentes alcancen formaciones con un mínimo error cuadrático medio al usar grafos totalmente rígidos. Sin embargo, también menciona que el utilizar grafos totalmente rígidos reduce el porcentaje de éxito del sistema en llegar a la meta establecida, ya que el exceso de restricciones dificulta el atravesar los obstáculos. Por lo que es mejor utilizar un grafo mínimamente rígido, para con el control de formación se tenga un mejor ajuste. La implementación se llevó a cabo a nivel de simulación en el entorno de Webots de Cyberbotics [9], en las cuales se obtuvo un éxito del 11% en el control de formación, sin embargo, el algoritmo completo tuvo un éxito del 80%, por lo que se puede decir que a nivel de simulación el conjunto de robots logró llegar a la meta pero no logró posicionarlos en la formación establecida. Para realizar las pruebas, se seleccionaron 2 grafos de 10 nodos cada uno. Los grafos fueron construidos por el método de inserción de Henneberg, esto con el fin de obtener grafos mínimamente rígidos.

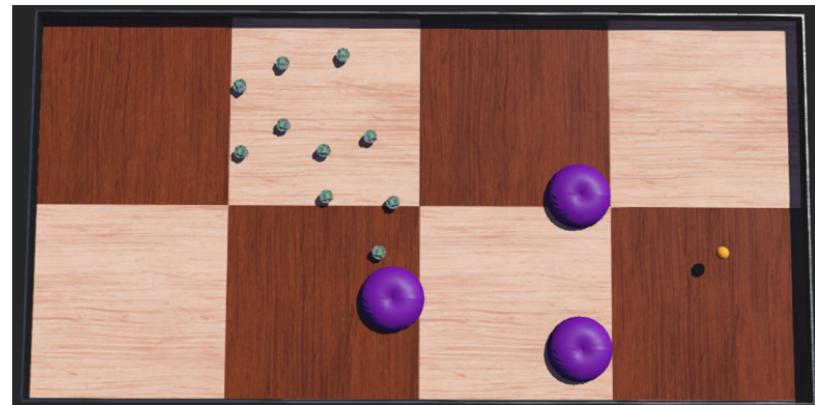


Figura 4: Grupo de robots ejecutando trayectoria de desplazamiento [8].

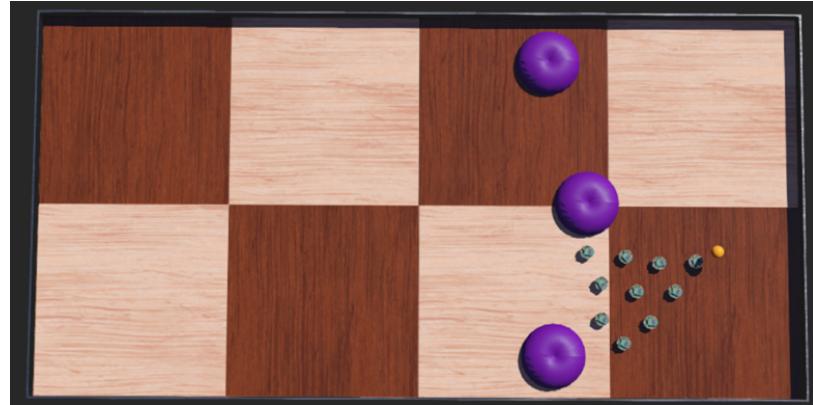


Figura 5: Grupo de robots en formación triangular [8].

En el trabajo de investigación [10] se desarrolló una herramienta para ajustar los parámetros del algoritmo PSO, la cual se denominó PSO Tuner, esto con el fin de mejorar el desempeño del algoritmo PSO. Este método consiste en una red neuronal recurrente que toma diferentes métricas propias de las partículas PSO y las convierte en una predicción de los hiper parámetros que debería emplear el algoritmo. En este trabajo se concluyó que el método de generación de trayectorias basado en programación dinámica posee una alta efectividad, sin embargo, cuenta con limitantes, las cuales son: baja reacción en situaciones con obstáculos dinámicos, y que su capacidad de escalabilidad es limitada.

En [11] a partir de la versión base del algoritmo de PSO se trabajaron modificaciones que permitieran ser implementada en robots diferenciales reales. Dado que el movimiento de las partículas es demasiado irregular causo problemas al enlazar directamente el movimiento de las partículas PSO con el movimiento de los robots. Por lo que, se propuso hacer que cada uno de los robots en lugar de seguir el movimiento exacto de una partícula del PSO, estos mejor tomarían la posición de la misma como una sugerencia de hacia donde realizar su desplazamiento. La implementación se llevó a cabo a nivel de simulación en el entorno de Webots, lograr plantear la forma adecuada de trasladar el algoritmo de PSO de un espacio de movimiento de partículas a un espacio tridimensional, por lo que en [11] se recomienda que en la implementación física se trabaje con una estructura cilíndrica para el cuerpo de los robots y que las medidas sean las mismas que las de E-Puck para que los controladores desarrollados en [11] sean aplicables directamente y no tener que hacer cambios.

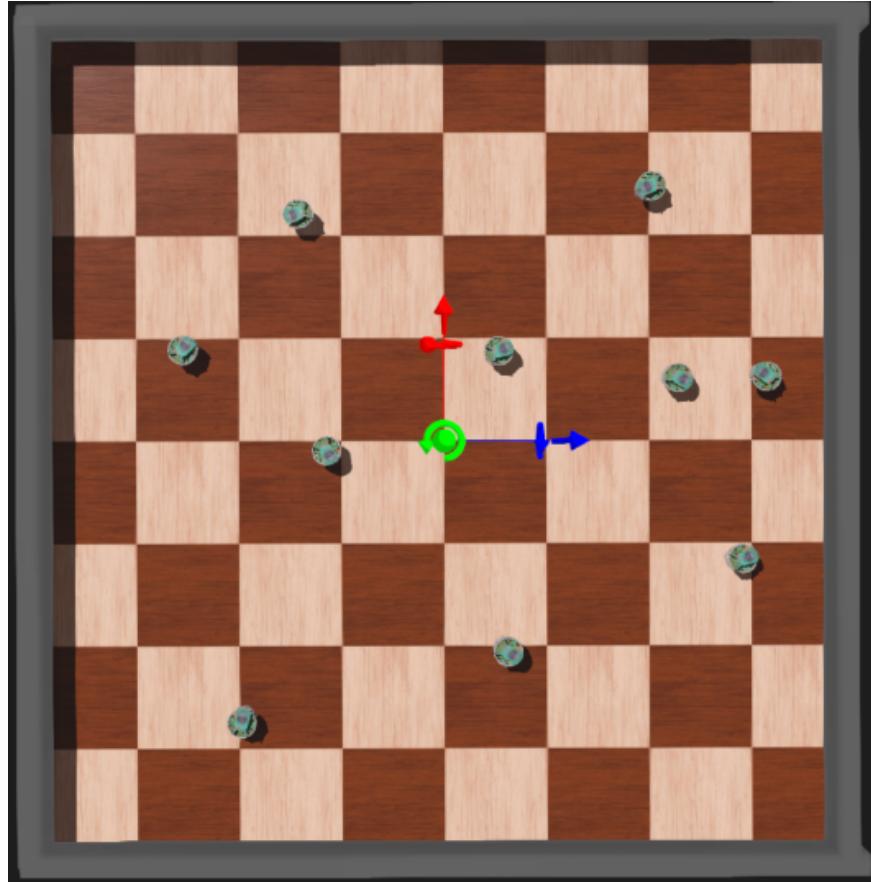


Figura 6: Grupo de Robots E-Puck ejecutando rutina de movimiento [11].

En [2] se modificó el algoritmo ACO para que se pueda resolver problemas de exploración de terrenos, planificación de terrenos, planificación de trayectorias y evasión de obstáculos. Para esta implementación se realizaron validaciones a nivel de simulación. Para los problemas de exploración de terrenos, planificación de trayectorias y evasión de obstáculos se propusieron dos algoritmos, el primero para la planificación de trayectorias y evasión de obstáculos, y el segundo para explorar terrenos desconocidos y realzar mapeos de estos. Los algoritmos se implementaron en Matlab y se tomó el algoritmo de ACO desarrollado en fases anteriores. El algoritmo de planificación de trayectorias se validó con la plataforma de Webots, implementando robots diferenciales utilizando controladores LQI, de pose y de pose de Lyapunov.

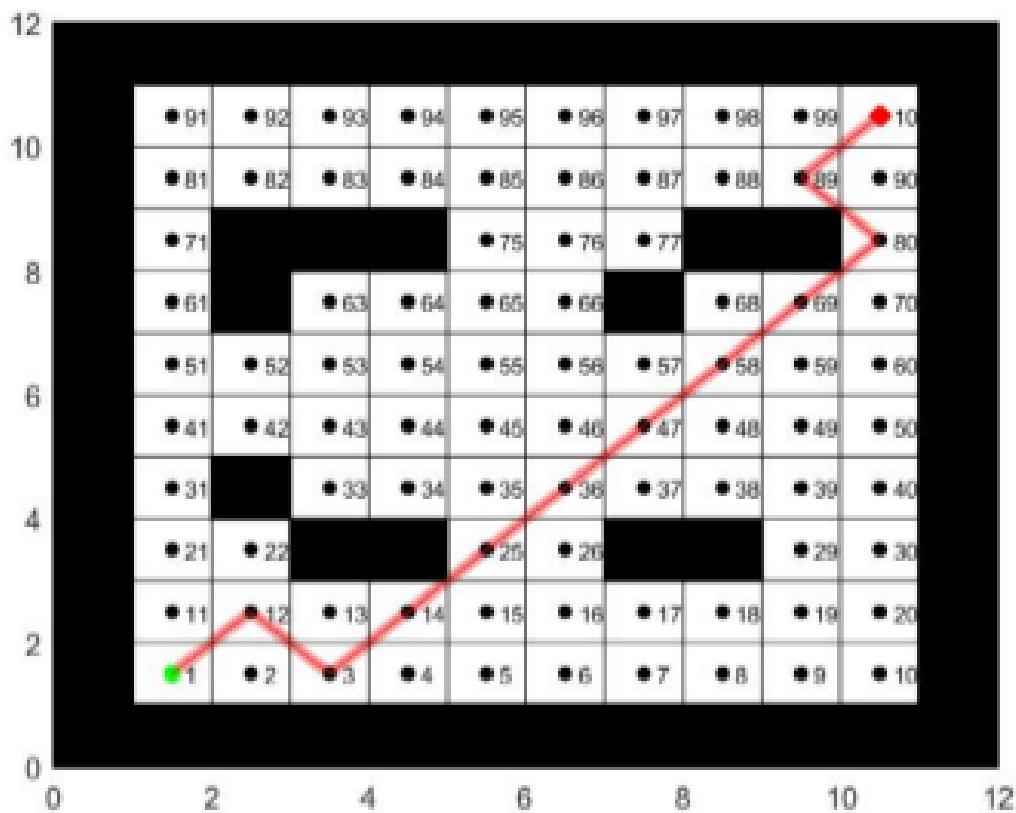


Figura 7: Rutina para validar en Webots [2].



Figura 8: Mapa implementado en Webots [2].

En [12] se abarcó tanto la simulación como la aplicación física de un método de SLAM utilizando ROS2. Se utilizó el paquete SLAM Toolbox desarrollado por Steve Macenski, que emplea algoritmos de Pose Graph SLAM para generar mapas 2D en tiempo real de los espacios recorridos por el Rover UVG. Estos mapas, conocidos como Cost Maps, asignan un valor a cada píxel de una imagen. Para la simulación se utilizó Gazebo y para la aplicación física se empleó un Lidar Hokuyo URG-04LX-UG01 para las lecturas de distancias, junto con los nodos de ROS2 desarrollados en otros módulos del proyecto Rover UVG para obtener la odometría, localización y orientación del robot.



Figura 9: Comparación de mapa 1 generado con metodología síncrona (izquierda) y asincrónica (derecha) [12].

CAPÍTULO 3

Justificación

En los trabajos en fases previas se han implementado los algoritmos de ACO y PSO para obtener las trayectorias de desplazamiento aplicados en sistemas de robots multi-agente. Para implementar y hacer uso de estos algoritmos es necesario tener información del entorno en el que los robots se van a movilizar. La idea es desarrollar un robot diferencial con la capacidad de obtener información del entorno para mapearlo. El mapeo de entornos es fundamental para que los algoritmos y robots puedan comprender y navegar de manera efectiva. Generar un mapa en 2D proporciona una base sólida para la toma de decisiones. Estas decisiones se ven reflejadas en la selección de la trayectoria generada para el desplazamiento del sistema de robots multi-agente, el cual es conformado por los robots diferenciales Pololu 3pi+ [1] .

El uso de algoritmos como ACO y PSO junto con el mapa generado a tiempo real permiten a los robots adaptarse a cambios en el entorno o a nuevas situaciones. Pueden analizar y actualizar continuamente el mapa a medida que exploran y recopilan nueva información, lo que les permite tomar decisiones con más informadas y ajustar las trayectorias de desplazamiento según sea necesario. La validación de los algoritmos a nivel de simulación utilizando herramientas como Webots y Matlab permite evaluar su rendimiento en escenarios controlados y reproducibles. Esto permite realizar ajustes y mejoras antes de llevar los algoritmos a pruebas físicas.

Los sistemas multi-agente que tienen la capacidad de crear mapas del entorno a medida que se desplazan y que planifican rutas eficientes para cubrir toda el área pueden explorar de manera autónoma entornos desconocidos, como áreas peligrosas o desastres naturales. Esto permite que los humanos no se expongan a situaciones de riesgo.

En situaciones de seguridad, como patrullaje y vigilancia, los sistemas de robots multi-agente pueden utilizar algoritmos de mapeo y planificación de trayectorias para patrullar de manera autónoma y eficiente. Pueden identificar áreas de interés, planificar rutas de patrullaje óptimas y realizar un monitoreo continuo del entorno.

En aplicaciones de logística y transporte, los sistemas de robots multi-agente pueden utilizar algoritmos de mapeo y planificación para optimizar la entrega de bienes en almacenes

o entornos industriales. Pueden crear mapas detallados de los espacios de trabajo, identificar las ubicaciones de los productos y planificar rutas eficientes para la recolección y entrega.

CAPÍTULO 4

Objetivos

Objetivo General

Desarrollar y evaluar algoritmos para el mapeo de entornos y generación de trayectorias utilizando sistemas robóticos multi-agente.

Objetivos Específicos

- Investigar y evaluar sensores de distancia, e integrarlos a robots diferenciales en simulaciones computarizadas.
- Desarrollar un algoritmo que permita mapear entornos en 2D utilizando los robots diferenciales simulados.
- Desarrollar un algoritmo que genere trayectorias utilizando el mapa generado del entorno.
- Validar el algoritmo de generación de trayectorias por medio de simulaciones computarizadas.
- Evaluar opciones de sensores para futuras implementaciones en físico de los algoritmos desarrollados.

CAPÍTULO 5

Alcance

El alcance de este trabajo abarca aspectos esenciales en el campo de la robótica y la navegación autónoma. Se llevó a cabo la implementación simulada de un algoritmo destinado a la gestión de trayectorias de exploración, lo que implicó la capacidad de guiar de manera autónoma un vehículo en la toma de decisiones con respecto a su movimiento en un entorno desconocido o poco explorado. Esta fase de desarrollo permitió la creación detallada de un mapa del entorno, un elemento fundamental para construir una representación precisa de la ubicación y posición relativa de objetos, obstáculos y elementos significativos dentro del área de trabajo.

El proceso de mapeo del entorno se estableció como una parte crucial, ya que proporcionó la base necesaria para que el vehículo autónomo tomara decisiones informadas. Esto incluyó la capacidad de evitar obstáculos, identificar rutas óptimas y alcanzar destinos específicos de manera eficiente. Es importante destacar que todo este proceso de mapeo se llevó a cabo en un entorno de simulación, específicamente en Webots. Esta elección proporcionó un nivel adicional de versatilidad al proyecto, permitiendo experimentar y probar algoritmos y controladores en un ambiente seguro y controlado.

Con base en las validaciones simuladas de los algoritmos desarrollados que permiten que un agente, en un entorno desconocido, pueda explorar, mapear y generar trayectorias en dicho entorno, crea la base para que en futuras fases del proyecto se implementen estos algoritmos de manera física. Esto podría lograrse mediante el uso de los robots Pololu 3pi+ o mediante el diseño y desarrollo de un vehículo explorador específico para tal propósito.

CAPÍTULO 6

Marco teórico

6.1. *Particle Swarm Optimization (PSO)*

En 1995, Kennedy y Eberhart propusieron el algoritmo PSO [13], el cual es metaheurístico, adecuado para optimizar funciones continuas no lineales. Los autores derivaron el algoritmo inspirado en el concepto de inteligencia de enjambre, que se observa con frecuencia en grupos de animales. El algoritmo PSO utiliza la idea de que las partículas en un enjambre pueden aprender de su propia experiencia y de las experiencias de las demás para encontrar soluciones óptimas en el espacio de búsqueda. Este enfoque ha demostrado ser efectivo en la optimización de funciones complejas y ha encontrado aplicaciones en diversas áreas, como la ingeniería, la economía y la inteligencia artificial.

El objetivo de un problema de optimización es determinar una variable representada por un vector $X = [x_1, x_2, x_3, \dots, x_n]$ que se minimiza o maximiza dependiendo de la formulación de optimización propuesta de la función $f(X)$. El vector de variables X se conoce como vector de posición. Este vector representa un modelo variable y es un vector de n dimensiones, donde n representa el número de variables que se pueden determinar en un problema. Por otro lado, la función $f(X)$ se llama función de aptitud u objetivo, que es una función que evalúa qué tan buena o mala es una posición X , por ejemplo, qué tan bueno piensa un ave que es un cierto punto de aterrizaje después de encontrarlo, y esta evaluación en este caso se realiza a través de varios criterios de supervivencia [13].

Considerando un enjambre con P partículas, se tiene un vector de posición y un vector de velocidad, respectivamente $X_i^t = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{in})^T$ y $V_i^t = (v_{i1}, v_{i2}, v_{i3}, \dots, v_{in})^T$ en la iteración t para cada partícula i que lo componen. Estos vectores se actualizan a través de la dimensión j de acuerdo con las siguientes ecuaciones:

$$V_{ij}^{t+1} = wV_{ij}^t + c_1 r_1^t (pbest_{ij} - X_{ij}^t) + c_2 r_2^t (gbest_j - X_{ij}^t) \quad (1)$$

$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1} \quad (2)$$

donde $i = 1, 2, \dots, P$ y $j = 1, 2, \dots, n$.

La Ecuación (1) indica que existen tres contribuciones diferentes al movimiento de una partícula en una iteración. La Ecuación (2) actualiza las posiciones de las partículas. El parámetro ω es la constante de peso de inercia y, para la versión clásica del PSO, es un valor constante positivo. Este parámetro es importante para equilibrar la búsqueda global, también conocida como exploración (cuando se establecen valores más altos), y la búsqueda local, conocida como explotación (cuando se establecen valores más bajos). En términos de este parámetro, se puede observar que es una de las principales diferencias entre la versión clásica del PSO y otras versiones derivadas de ella [13].

El primer término de la Ecuación de actualización de la velocidad es el producto entre el parámetro ω y la velocidad previa de la partícula, lo cual explica por qué denota el movimiento anterior de la partícula en el movimiento actual. Por ejemplo, si $\omega = 1$, el movimiento de la partícula está totalmente influenciado por su movimiento previo, por lo que la partícula puede seguir en la misma dirección. Por otro lado, si $0 \leq \omega < 1$, dicha influencia se reduce, lo que significa que la partícula se mueve hacia otras regiones en el dominio de búsqueda. Por lo tanto, a medida que se reduce el parámetro de peso de inercia, el enjambre puede explorar más áreas en el dominio de búsqueda, lo que aumenta las posibilidades de encontrar un óptimo global. Sin embargo, existe un costo al usar valores de ω más bajos, que es que las simulaciones requieren más tiempo [13].

El término de cognición individual, que es el segundo término de la Ecuación (1), se calcula mediante la diferencia entre la mejor posición propia de la partícula, por ejemplo, $pbest_{ij}$, y su posición actual X_{ij}^t . Se puede observar que la idea detrás de este término es que a medida que la partícula se aleja más de la posición $pbest_{ij}$, la diferencia ($pbest_{ij} - X_{ij}^t$) debe aumentar, por lo tanto, este término aumenta, atrayendo a la partícula hacia su mejor posición propia. El parámetro c_1 que aparece como un producto en este término es una constante positiva y es un parámetro de cognición individual, y pondera la importancia de las experiencias previas de la partícula. El otro parámetro que compone el producto del segundo término es r_1 , y este es un parámetro de valor aleatorio en el rango $[0, 1]$. Este parámetro aleatorio desempeña un papel importante, ya que evita convergencias prematuras, aumentando la probabilidad de encontrar los óptimos globales más probables [13].

En la Ecuación (1) el tercer término es el de aprendizaje social. Debido a este término, todas las partículas en el enjambre pueden compartir la información del mejor punto alcanzado sin importar qué partícula lo haya encontrado, por ejemplo, $gbest_j$. Su formato es similar al del segundo término, el de aprendizaje individual. Así, la diferencia ($gbest_j - X_{ij}^t$) actúa como una atracción para las partículas hacia el mejor punto encontrado hasta ese momento en la iteración t . De manera similar, c_2 es un parámetro de aprendizaje social y pondera la importancia del aprendizaje global del enjambre. Y r_2 desempeña exactamente el mismo papel que r_1 [13].

Por último, se muestran los pasos generales del algoritmo PSO y cómo se lleva a cabo la optimización en busca de mínimos, donde todos los vectores de posición son evaluados por la función $f(X)$ [13].

1. Inicialización

- a) Para cada partícula i en el enjambre con población P :
 - 1) Inicializa X_i con un valor aleatorio.
 - 2) Inicializa V_i con un valor aleatorio.
 - 3) Evalúa la aptitud de $f(X_i)$
 - 4) Inicializa $pbest_i$ con el valor de X_i
- b) Inicializa $gbest$ con el valor de X_i con la mejor aptitud.

2. Repetir hasta que se satisfaga el criterio de parada:

- a) Para cada partícula i :
 - 1) Actualizar V_i^t y X_i^t correspondiendo a las Ecuaciones (1) y (2)
 - 2) Evaluar la aptitud $f(X_i^t)$
 - 3) $pbest_i \leftarrow X_i^t$ if $f(pbest_i) < f(X_i^t)$
 - 4) $gbest \leftarrow X_i^t$ if $f(gbest) < f(X_i^t)$

6.2. *Ant Colony System (ACS) with Travel Salesman Problem (TSP)*

A principios de la década de 1990, se descubrió a través de experimentos que cuando las hormigas están buscando alimento, dejan una sustancia química, llamada feromona, en el camino por donde transitan. Las hormigas eligen el siguiente camino basándose en la concentración de feromonas: cuanto mayor sea la concentración, mayor será la probabilidad de que sea seleccionado. Al mismo tiempo, debido a la naturaleza volátil de las feromonas, la feromona en el camino menos favorable se va reduciendo cada vez más, lo que permite seleccionar el camino óptimo [14].

Ant Colony System (ACS) es una variante del algoritmo de optimización por colonia de hormigas (ACO) que fue propuesta por Marco Dorigo en 1997. ACS se utiliza para resolver problemas de optimización combinatoria y se basa en el comportamiento de las colonias de hormigas. El algoritmo se divide en dos fases: construcción de soluciones y actualización de feromonas. En la fase de construcción de soluciones, las hormigas construyen soluciones mediante la selección de una ciudad inicial y la selección de la siguiente ciudad a visitar en función de la cantidad de feromonas depositadas en las aristas que conectan las ciudades. ACS utiliza una regla de transición estocástica para la selección de la siguiente ciudad, que combina la información de feromonas con información heurística basada en la distancia entre las ciudades. En la fase de actualización de feromonas, se actualizan las cantidades de feromonas en función de la calidad de las soluciones construidas por las hormigas. ACS utiliza una estrategia de actualización de feromonas que deposita una cantidad adicional en las aristas que forman parte de la mejor solución encontrada hasta el momento [15]. ACS ha sido utilizado en una amplia variedad de aplicaciones, como la resolución del Problema del Agente Viajero, el diseño de rutas de recolección de basura, la optimización de la distribución de plantas industriales, entre otros. El algoritmo ha sido mejorado a lo largo del tiempo, con mejoras en la adaptación automática, la diversidad de grupos, la búsqueda local y la

combinación con otros algoritmos de optimización. En resumen, Ant Colony System (ACS) es una variante del algoritmo de optimización por colonia de hormigas (ACO) que se utiliza para resolver problemas de optimización combinatoria. El algoritmo se divide en dos fases: construcción de soluciones y actualización de feromonas. ACS ha sido utilizado en una amplia variedad de aplicaciones y ha sido mejorado a lo largo del tiempo [16].

La fórmula de selección para cada hormiga en el ACS desde la ciudad i hacia la ciudad j :

$$J = \begin{cases} \arg \max_s \left\{ \tau_{ij} [\eta_{ij}]^\beta \right\} & \\ & \end{cases} \quad (3)$$

$$\begin{cases} q \leq q_0 \\ q > q_0 \end{cases} \quad (4)$$

donde η_{ij} representa el inverso de la distancia entre la ciudad i y la ciudad j , τ_{ij} representa la concentración de feromonas entre la ciudad i y la ciudad j , q_0 es un parámetro con un valor constante. q es una variable aleatoria sujeta a una distribución uniforme entre 0 y 1. J es la siguiente ciudad a seleccionar. s es igual a la Ecuación (5). La Ecuación (3) muestra que las ciudades con alta concentración de feromonas o con distancias relativamente cercanas tienen más probabilidades de ser seleccionadas. Cuando $q \leq q_0$, se utiliza la Ecuación (3); de lo contrario, se utiliza la Ecuación(5) [14].

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & j \in N_i^k \\ 0, & j \notin N_i^k \end{cases} \quad (5)$$

donde α es el factor heurístico de información. β es el factor heurístico de expectativa; N_i^k es una colección de ciudades a las que las hormigas pueden llegar directamente y aún no han visitado. η_{ij} es una función heurística, su expresión se encuentra en la Ecuación (6) [14].

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (6)$$

Actualización de la feromona local: Cuando la hormiga lleva a cabo la construcción del camino, se mueve desde la ciudad actual i hacia la siguiente ciudad j y actualiza inmediatamente la feromona en el camino, lo cual puede expresarse mediante la Ecuación (7).

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\tau_0 \quad (7)$$

Donde ρ es el coeficiente de evaporación local de feromonas cuyo rango es $[0, 1]$; τ_0 es el valor inicial de feromonas [14].

Actualización de la feromona global: Después de que todas las hormigas hayan completado su recorrido, solo el camino global óptimo puede actualizar las feromonas, lo cual acelera la convergencia del algoritmo, y su expresión es la Ecuación (8).

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\Delta\tau_{ij}^{bs} \quad (8)$$

$$\Delta\tau_{ij}^{bs} = \frac{1}{C^{bs}} \quad (9)$$

Donde ξ es el coeficiente de evaporación global de feromonas, C^{bs} es la longitud del camino global óptimo. $\Delta\tau_{ij}^{bs}$ es la feromona añadida al camino global óptimo, su expresión se muestra en la Ecuación (9) [14].

6.3. Robots diferenciales

Los robots móviles con ruedas (WMRs, por sus siglas en inglés) se utilizan en diversas áreas como agricultura, logística, atención médica, cuidado en el hogar, exploración planetaria, transporte urbano, operaciones de vigilancia, inspección y mantenimiento, y seguridad y defensa. La precisión del movimiento es crucial para los WMRs; el robot debe ser calibrado y cualquier inexactitud posicional debe ser mejorada antes de su implementación en campo. Algunas de las técnicas que se han desarrollado para obtener precisión en la pose incluyen detección de errores de cámaras 3D, balizas activas, giroscopios, brújulas magnéticas y odometría. La odometría, que utiliza datos de sensores posicionales como encoders conectados a cada actuador del robot, mide los cambios en la posición del robot a lo largo del tiempo. El método de odometría se aplica para reducir la inexactitud posicional en diversos WMRs [17].

Las fuentes de inexactitud posicional sistemática y no sistemática afectan la precisión del movimiento del WMRs con mecanismo de tracción diferencial. Los elementos de control y mecánicos que causan inexactitud sistemática se deben a imperfecciones e irregularidades que pueden ocurrir en cualquier etapa de producción, desde el diseño original hasta la fabricación de componentes. La inexactitud no sistemática, como sugiere su nombre, es independiente de la estructura del robot. Este tipo de inexactitud es el resultado de irregularidades en la superficie y otros factores [17].

Se emplea un método de reducción de errores basado en la cinemática de los robots para reducir la inexactitud posicional. Luego se compara esta inexactitud con un conjunto de índices cuantificables. En esta técnica, utilizamos los índices lateral y longitudinal para medir la inexactitud posicional. Estos índices requieren modelado cinemático de un robot para especificar cómo el robot mismo afecta su localización [17].

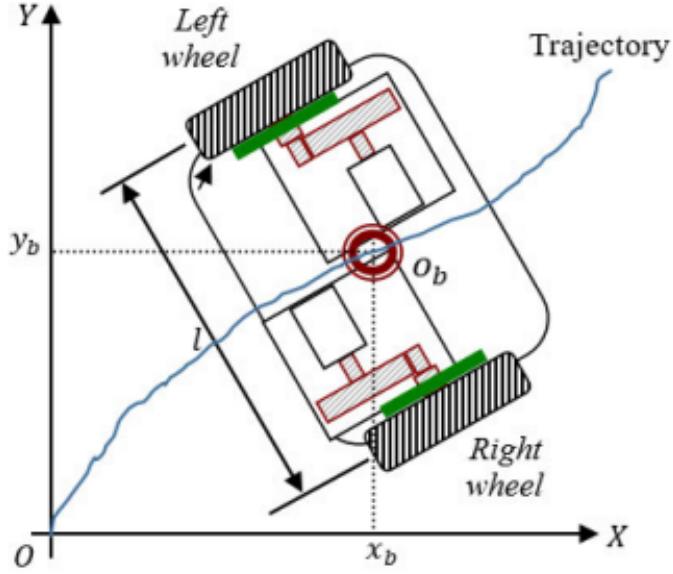


Figura 10: Sistema de coordenada de referencia de un robot móvil con ruedas y tracción diferencial. XY muestra el marco de referencia mundial. (x_b, y_b) es la coordenada del centro de masa del robot, \dot{x} e \dot{y} son las velocidades de este centro a lo largo de los ejes X_r e Y_r , respectivamente [17].

Comprender el movimiento de los WMR con tracción diferencial comienza con la descripción de la contribución de cada rueda al movimiento del robot. Así como cada rueda contribuye al movimiento, cada rueda también impone limitaciones. La Figura 10 muestra la estructura de un robot con ruedas comúnes que utiliza tracción diferencial. En este dibujo, las ruedas motrices se utilizan para proporcionar potencia a la plataforma. En la medida en que las dos ruedas motrices ruedan y no resbalan, tenemos [17].

$$\dot{x}_b \cos \theta + \dot{y}_b \sin \theta = \left(D_L \dot{\theta}_L + D_R \dot{\theta}_R \right) / 4 \quad (10)$$

$$2l\dot{\theta} = D_R \dot{\theta}_R - D_L \dot{\theta}_L \quad (11)$$

donde (x_b, y_b) es la coordenada del centro de masa del robot, \dot{x} e \dot{y} son las velocidades de este centro a lo largo de los ejes X_R e Y_R , respectivamente. θ representa la orientación del robot con respecto a la posición inicial en el punto de inicio del movimiento, y $\dot{\theta}$ es su derivada con respecto al tiempo. D_R y D_L son los diámetros nominales de las ruedas derecha e izquierda. Las velocidades angulares de las ruedas derecha e izquierda se denotan por $\dot{\theta}_R$ y $\dot{\theta}_L$. La distancia nominal entre las dos ruedas se representa por l [17].

6.4. Linear-quadratic regulator (LQR)

El Regulador Cuadrático Lineal (LQR) es un controlador óptimo que garantiza la estabilidad del sistema en un lazo cerrado mediante el uso de ganancias de retroalimentación.

Su objetivo es lograr la respuesta deseada utilizando la mínima cantidad de control. Sin embargo, debido a que el sistema a controlar es un Robot Diferencial y no es controlable, es necesario utilizar el difeomorfismo [18].

$$u^* = -k(x - x_{ss}) + u_{ss} \quad (12)$$

6.5. Mapeo de entornos con *Multi-Stage Optimization* (MSO)

En el método MSO el proceso de construcción de mapas se divide en tres etapas diferentes basadas en la relación entre el robot individual y los múltiples robots en el sistema multi-agente: el filtro de Kalman extendido (EKF) en la etapa de adquisición de mapas del robot individual, la Topología de comunicación dinámica autoorganizada (DSCT) en la etapa de transmisión de mapas multi-robot, y el filtrado de información en múltiples capas (MLIF) en la etapa de fusión de mapas multi-agente. En el sistema multi-agente, cada etapa coopera para completar la tarea de construir un mapa global en el entorno interior [19].

En la etapa de adquisición de mapas del robot individual, primero se optimiza el mapa global desde la perspectiva del robot individual. Según el proceso de obtención del mapa local, la precisión del mapa global se basa en el resultado de posicionamiento de los robots. Cuando los robots atraviesan un entorno desconocido, la incertidumbre sobre su posicionamiento aumenta y la construcción de un mapa global se vuelve ardua. Para mejorar la precisión del mapa local por parte del robot individual, el método MSO utiliza el EKF. El EKF integra la información de odometría de las ruedas y la información de odometría láser del robot para obtener resultados de posicionamiento más precisos, mejorando así la precisión de la construcción del mapa para el robot individual [19].

En segundo lugar, en la etapa de transmisión de mapas multi-agente, el método MSO utiliza el DSCT para agregar un mecanismo dinámico autoorganizado basado en una red de comunicación básica para lidiar con las fallas de los robots. Para permitir una comunicación robusta entre los robots, el método MSO predice los cambios dinámicos en el proceso de construcción de un mapa global y establece un maestro de respaldo para completar la transferencia del centro de la topología [19].

Finalmente, el método MSO utiliza el MLIF para optimizar el mapa global en la etapa de fusión de mapas multi-robot. El MLIF representa y optimiza diferentes informaciones erróneas a través de diferentes capas, filtra el mapa en orden y luego produce un mapa global con mayor precisión [19].

6.6. Filtro de Kalman extendido (EKF)

En este proceso, se utilizan dos métodos para estimar la posición relativa de un robot: la odometría de ruedas y la odometría láser. La odometría de ruedas se basa en contar el número de vueltas de las ruedas para determinar la posición relativa del robot. Por otro lado, la odometría láser procesa los datos escaneados por el láser, realiza coincidencias de

transformación de posición entre fotogramas consecutivos y obtiene la transformación de posición relativa mediante un método incremental [19].

La ventaja de la odometría de ruedas es que tiene una alta frecuencia de muestreo, lo que permite medir la posición relativa del robot en poco tiempo. Sin embargo, en la realidad, las ruedas siempre giran y resbalan, lo que resulta en deficiencias significativas en la precisión de posicionamiento. Afortunadamente, la odometría láser no se ve afectada por la condición de las ruedas. El EKF se utiliza para compensar el error de la situación, mejorando así la precisión de posicionamiento y, en última instancia, mejorando la precisión del mapa local [19].

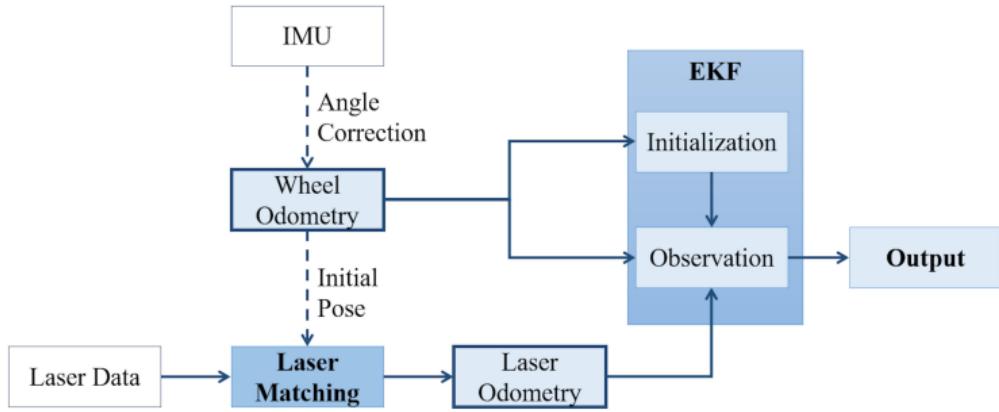


Figura 11: Diagrama de flujo del proceso que sigue EKF [19].

6.7. Topología de comunicación dinámica autoorganizada (DSCT)

Para que el sistema multi-agente pueda completar la construcción del mapa en un entorno real, la robustez de la comunicación es un punto clave al que se debe prestar atención. En primer lugar, se construye una red de comunicación básica y luego se diseña una red de comunicación dinámica que tiene una topología autoorganizada para hacer frente a la complejidad y variabilidad del entorno real [19].

La red de comunicación básica se refiere al mecanismo de colaboración de los robots bajo condiciones de comunicación estables. En este método, se construye un sistema centralizado basado en el esquema de maestro y esclavos. El robot maestro tiene ventajas de implementación simple y asegura la optimización de toda la tarea. Los robots de exploración (r_1, \dots, r_i) obtienen el mapa local en la primera etapa y luego lo transmiten al robot maestro (r_0). El robot maestro r_0 envía el mapa global a cada robot de exploración a tiempo para facilitar la planificación de su ruta. Este marco de colaboración garantiza la ejecución efectiva de la construcción del mapa global en el sistema multi-agente [19].

Sin embargo, la red de comunicación no siempre puede mantenerse estable en un entorno real debido a posibles cambios causados por la falta de energía de la batería del robot, fallas repentinas en la red, entre otras posibles causas. La metodología MSO adopta el DSCT para manejar estas incertidumbres. El proceso muestra el contenido específico del DSCT [19].

El DSCT mantiene una lista de robots en la red de comunicación en el robot maestro r_0 y registra el número de robot, la dirección IP y el número más pequeño de todos los robots de exploración durante el proceso de organización de la red. Los cambios en la red de comunicación multi-robot incluyen principalmente tres situaciones: (1) el robot de exploración ingresa a la red, (2) el robot de exploración sale de la red y (3) el robot maestro sale de la red [19].

Cuando un robot de exploración ingresa a la red, se crea una conexión entre el robot maestro y el nuevo robot de exploración en la red, siempre que el esquema anterior de maestro y esclavos se mantenga sin cambios. Por ejemplo, si el robot maestro r_0 recibe una solicitud de unirse a la red por parte de r_4 , agrega el número y la IP de r_4 a la lista de robots y el número más pequeño en la red no cambia. Los otros robots de exploración no se ven afectados y continúan completando sus tareas [19].

Cuando un robot de exploración sale de la red, el robot maestro actualiza la estructura de la red, elimina la conexión con los robots de exploración que han salido de la red e informa a los otros robots de exploración. Por ejemplo, si el robot maestro r_0 no recibe información de r_1 durante mucho tiempo, asume que r_1 tiene un mal funcionamiento, corta la conexión con él e informa a los otros robots de exploración. El robot maestro r_0 elimina la información de r_1 en la lista de robots y verifica si es necesario actualizar el número más pequeño en la red [19].

Cuando el robot maestro se retira de la red, se destruye el esquema anterior de maestro y esclavos, y los robots de exploración determinados por el número más pequeño deben reconstruir una nueva red de topología estrella. Por ejemplo, cuando todos los robots de exploración no reciben información del robot maestro r_0 durante mucho tiempo, se asume que r_0 ha perdido la conexión. r_2 , determinado por el número más pequeño, se convertirá automáticamente en el próximo robot maestro y asumirá la tarea de volver a organizar la red. El nuevo robot maestro heredará el mapa global fusionado por la red anterior y continuará completando la construcción del mapa y las tareas de exploración del entorno [19].

A través del DSCT en el método MSO, la colaboración de los robots no se ve afectada por la calidad de la comunicación del entorno real y se puede completar la tarea de construcción de un mapa global. Este método garantiza la eficacia de la construcción del mapa multi-agente [19].

Basic Communication Network

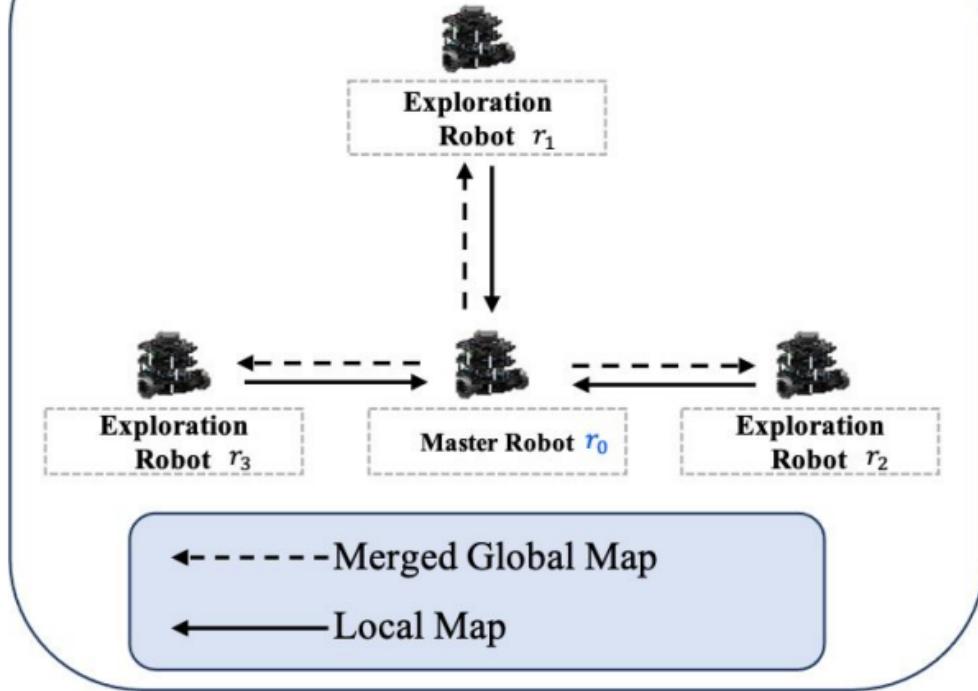


Figura 12: Red de comunicación básica con un maestro [19].

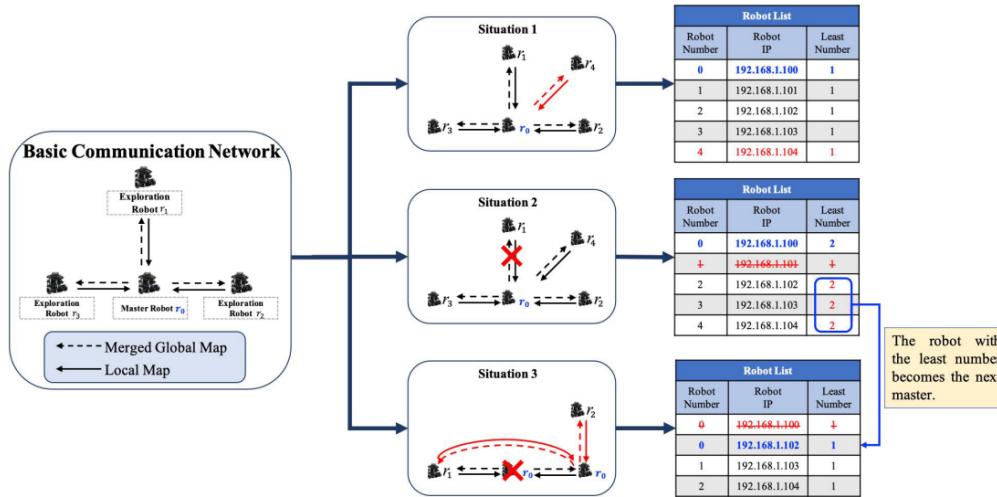


Figura 13: Proceso de DSCT en el método MSO [19].

6.8. Filtrado de información en múltiples capas (MLIF)

En la etapa de fusión de mapas de varios robots, el método propuesto utiliza el MLIF para mejorar la precisión del mapa global. El mapa global construido por el sistema multi-robot puede contener información incorrecta, como información de robots no conectados, información de obstáculos dinámicos e información de obstáculos estáticos no uniformes. Teniendo en cuenta las reglas generadas por la situación real de la exploración multi-agente en el entorno interior, filtramos el mapa global para eliminar los estados de cuadrícula incorrectos [19].

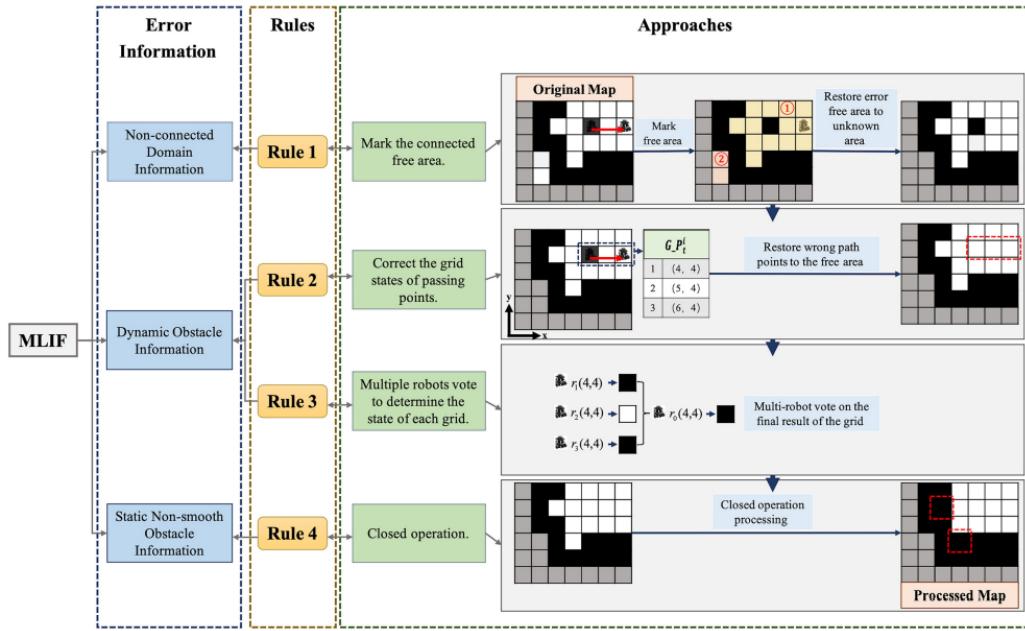


Figura 14: Proceso de MLIF en el método MSO [19].

6.9. Light Detection and Ranging (LiDAR)

LiDAR es una tecnología de teledetección que utiliza luz láser para medir distancias y crear modelos tridimensionales de objetos y entornos. LiDAR funciona emitiendo pulsos láser que rebotan en los objetos y regresan al sensor, que luego calcula la distancia entre el sensor y el objeto. Esta tecnología se utiliza comúnmente en campos como la arqueología, silvicultura, geología y planificación urbana, entre otros. LiDAR se puede utilizar para crear modelos digitales de elevación altamente precisos, que son útiles para cartografiar terrenos e identificar características como edificios, árboles y cuerpos de agua. En los últimos años, LiDAR se ha vuelto cada vez más popular en la industria automotriz para su uso en vehículos autónomos, ya que puede ayudar a los vehículos a detectar y evitar obstáculos.

El sensor RP LiDAR A1 es un escáner láser 2D de bajo costo desarrollado por SLAMTEC. Según los datos del fabricante, tiene un rango de operación de 12 metros y produce una nube de puntos que se puede utilizar para la localización de objetos y modelado del

entorno. Trabaja a una frecuencia de entre 2 Hz y 10 Hz, siendo 5.5 Hz la frecuencia óptima, generando en promedio 1450 puntos por cada rotación. Se recomienda su uso en interiores, donde ofrece mejores resultados, especialmente en condiciones de poca luz ambiental o artificial. Internamente, el LiDAR consta de un sensor infrarrojo que emite luz y otro que la recibe, lo que permite triangular y calcular el tiempo que tarda en regresar la luz después de rebotar en los objetos que lo rodean [20].

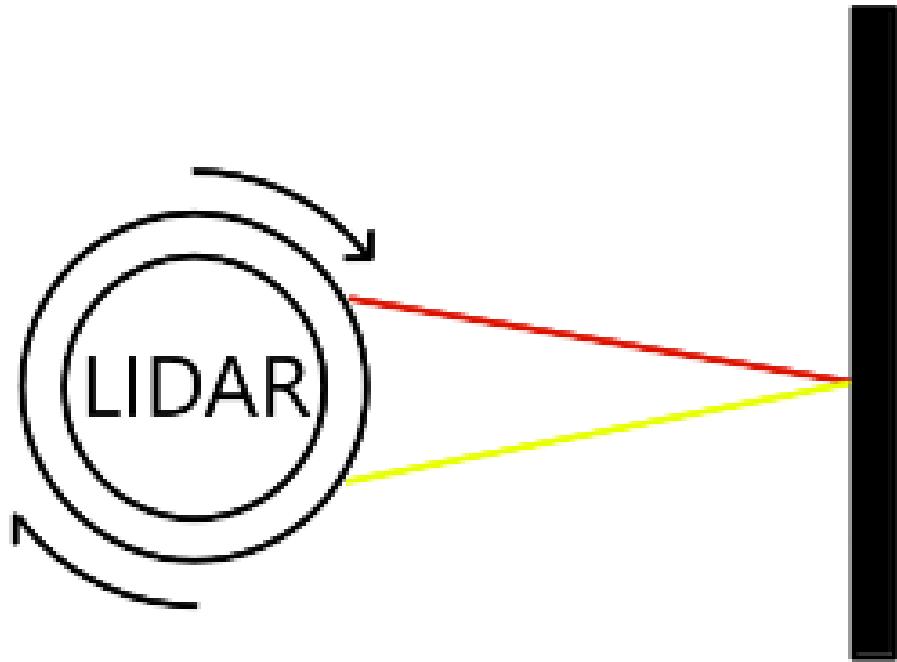


Figura 15: Funcionamiento LiDAR [20].

6.10. Vehículo diferencial *Pololu 3pi 32U4*

El vehículo diferencial Pololu 3pi+ 32U4 es un robot versátil, programable por el usuario, con un diámetro de 9.7 cm o 3.8 in. En su núcleo, se encuentra un microcontrolador ATmega32U4 de Microchip. Cuenta con una interfaz USB y un gestor de arranque compatible con Arduino. Este robot tiene la capacidad de incorporar periféricos adicionales. Las líneas de alimentación del robot, los controles del sistema de alimentación y las líneas de entradas y salidas del microcontrolador se pueden acceder a través de varios puertos de expansión de paso de 0.1 in [1].



Figura 16: Pololu 3pi+ 32U4 OLED Robot [1].

El Pololu 3pi+ 32U4 cuenta con dos controladores de motor, puente H, y una variedad de sensores integrados, que incluyen dos encoder para el control de motor, una unidad de medición inercial completa (acelerómetro de 3 ejes, giroscopio y magnetómetro), cinco sensores de reflexión hacia abajo para seguir líneas o detectar bordes, y sensores de golpes izquierdo y derecho en la cara frontal del robot. También incluye tres botones a bordo, un zumbador y LEDs indicadores que ofrecen una interfaz conveniente para la interacción del usuario [1].

En [1] se especifica que el vehículo diferencial Pololu 3pi+ 32U4 cuenta con los siguientes demostraciones de funciones, las cuales son accesibles desde el menú:

1. *Power* (Energía): Esta función muestra el voltaje de la batería en milivoltios, que debe estar por encima de 5000 para un juego de baterías completamente cargadas e indica si hay alimentación USB presente.
2. *LineSens* (Sensores de Línea): Esta función muestra las lecturas actuales de los cinco sensores de infrarrojos en una gráfica de barras. Barras más grandes significan una reflectancia más baja, por ejemplo, cuando el sensor está sobre algo oscuro. Colocar un objeto reflectante, como el dedo, debajo de uno de los sensores debería hacer que la lectura correspondiente disminuya visiblemente en la gráfica.
3. *BumpSens* (Sensores de Golpe): Esta función le permite probar los sensores de golpe a lo largo del lado frontal del robot. Mientras el sensor izquierdo esté presionado, la pantalla mostrará "L" y el LED amarillo estará encendido. Mientras el sensor derecho esté presionado, la pantalla mostrará "R" y el LED rojo estará encendido. El zumbador también sonará cada vez que se presione un sensor de golpe, con un tono diferente para el lado izquierda y derecha.

4. *Inertial* (Inercial): Esta función le permite probar los sensores iniciales del robot. A medida que haga rodar, incline o gire el robot en sus manos, la línea superior de la pantalla mostrará en qué eje (X, Y o Z) está rotando el robot, según lo medido por el giroscopio de tres ejes integrado. Cuando el robot esté inmóvil, solo debería mostrar “*Rot*”, sin ningún eje listado.
5. *Compass* (Brújula): Esta función utiliza el magnetómetro de tres ejes del robot para medir el campo magnético de la tierra y determinar en qué dirección de la brújula está direccionado. La demo se calibra constantemente, y para funcionar bien, el robot debe girarse inicialmente sobre varios ejes, la pantalla comenzará mostrando “¡Gírame!” para indicar esto.
6. *Motors* (Motores): Esta función le permite probar el giro de los motores.
7. *Encoders*: Esta demo presenta la visualización de la cuenta de los encoders de los motores izquierdo y derecho en la pantalla. Al girar manualmente las ruedas, se puede observar cómo las cuentas aumentan o disminuyen en función de la dirección de rotación.
8. *Spin* (Girar): Esta función hace que el Pololu 3pi+ 32U4 gire en su lugar en ambas direcciones.
9. *LEDs*: Esta función recorre la iluminación de los LEDs de usuario rojos, verdes y amarillos.
10. *OLED o LCD*: Esta función muestra varios caracteres de texto que puede mostrar en la pantalla OLED o LCD. Usando los botones se puede desplazar por las páginas de caracteres.
11. *Music*: Reproduce una adaptación de la Fuga en Re Menor de J. S. Bach para microcontrolador y zumbador, mientras desplaza un texto en pantalla. Esto demuestra la capacidad del Pololu 3pi+ 32U4 para reproducir música en segundo plano.

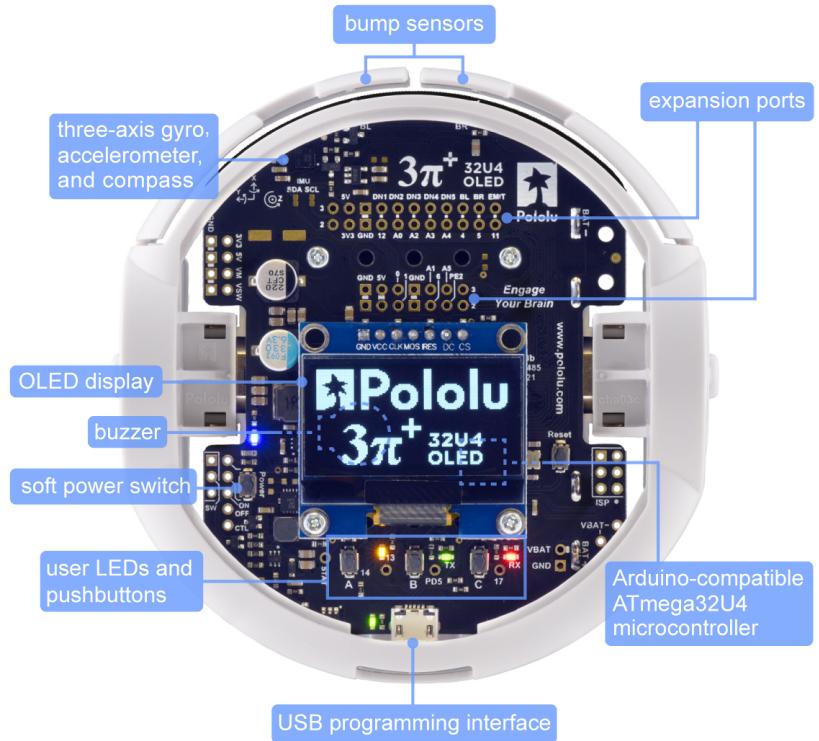


Figura 17: Características del Pololu 3pi+ 32U4 OLED Robot [1].

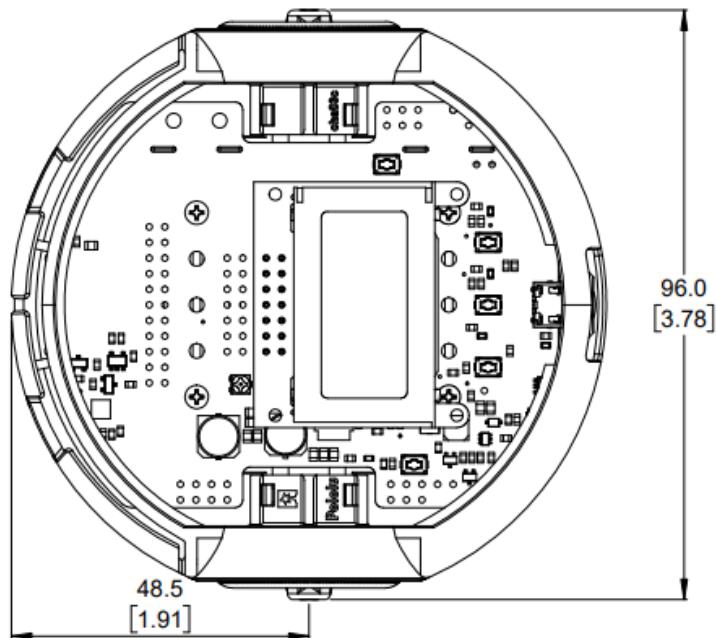


Figura 18: Dimensiones vista superior del Pololu 3pi+ 32U4 OLED Robot (mm [in]) [1].

6.11. Entorno de simulación Webots

Webots es una aplicación multiplataforma de código abierto, que proporciona un entorno de desarrollo completo para la simulación de robots, permitiendo un prototipado rápido en mundos virtuales 3D, como el que se puede observar en la Figura 19. Webots es ampliamente adoptado en la industria, la educación y la investigación. La simulación en Webots se basa en una interfaz gráfica moderna, un motor de física y un motor de renderizado [9].

Una característica destacada es la versatilidad de programación de robots en Webots, que admite lenguajes como C, C++, Python, Java, MATLAB y ROS, a través de una API intuitiva que abarca las necesidades fundamentales de la robótica. Además, Webots ofrece amplia documentación y una comunidad activa en Discord, lo que garantiza respuestas rápidas a las preguntas de los usuarios [9].

En [9] se especifican las siguientes propiedades que permiten la simulación de sistemas robóticos:

1. **Mundos:** Los mundos en Webots son entornos virtuales en 3D donde se pueden simular robots y otros objetos. Los mundos se pueden crear desde cero o se pueden importar desde otros programas de modelado 3D. Los mundos en Webots se pueden personalizar y ajustar para simular diferentes condiciones y situaciones.
2. **Nodos y Funciones API:** Los nodos en Webots son objetos que se utilizan para construir mundos virtuales. Los nodos se pueden agregar, eliminar y modificar para crear objetos complejos. Las funciones de API en Webots son funciones que se utilizan para programar robots y otros objetos en los mundos virtuales. Las funciones de API se pueden utilizar para controlar el movimiento, la posición y la orientación de los objetos en el mundo virtual.
3. **PROTO:** El mecanismo PROTO permite a los usuarios extender el conjunto de nodos agregando sus propios nodos. De esta manera, los usuarios pueden construir y reutilizar objetos complejos. Los nodos PROTO se definen en un archivo de texto y se pueden utilizar en cualquier mundo de Webots. Los nodos PROTO se pueden personalizar y ajustar para adaptarse a las necesidades específicas de cada usuario.



Figura 19: Mundo simulado en Webots [9].

CAPÍTULO 7

Construcción virtual de robot móvil con ruedas y tracción diferencial

Se ha creado una representación tridimensional del vehículo diferencial Pololu 3pi+ como parte fundamental de la plataforma utilizada para investigar y desarrollar algoritmos con el fin de implementarlos en el entorno de simulación Webots. El modelo 3D del Pololu 3pi+ se ha diseñado para reflejar las dimensiones, la forma y las características físicas del robot real. Esto incluye la disposición de las ruedas, la ubicación de los sensores y cualquier otro detalle relevante que sea esencial para una simulación precisa. Este modelo en 3D actúa como la base sobre la cual se desarrollan, prueban y perfeccionan los algoritmos de control y navegación.

La precisión en el modelado del bumper es esencial, ya que afecta directamente la capacidad del robot simulado para evitar obstáculos y mantener un comportamiento realista en el entorno de simulación. Con un modelo de bumper preciso, los algoritmos desarrollados se pueden probar, detectar fallas y mejorar.

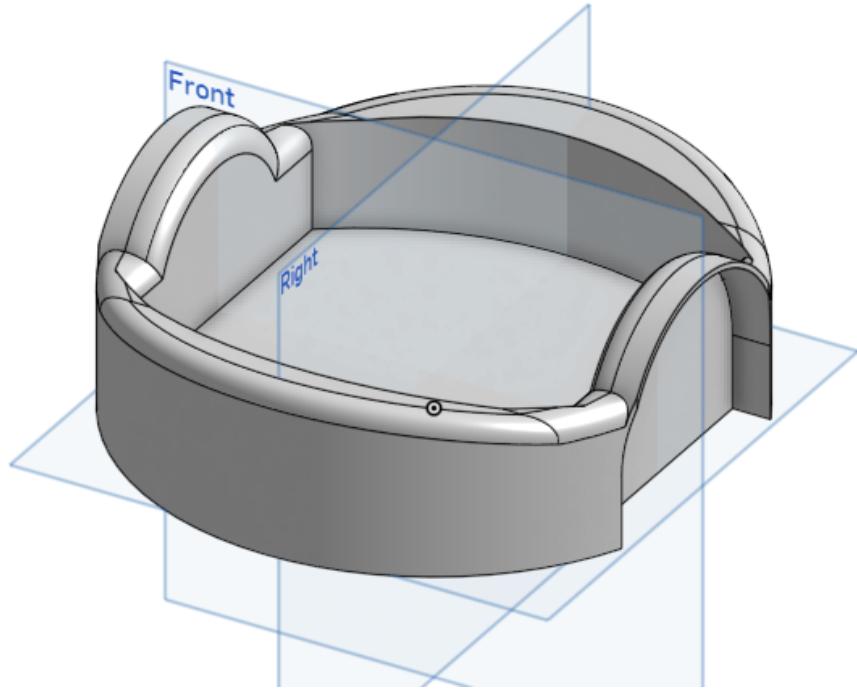


Figura 20: Representación 3D del bumper de Pololu 3pi+.

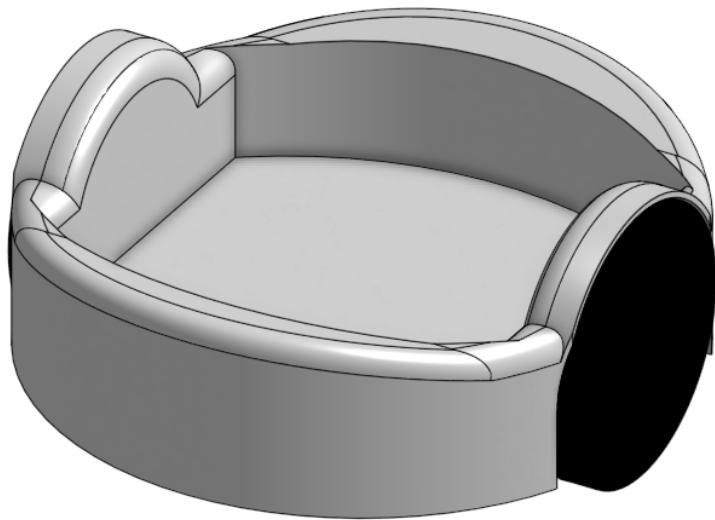


Figura 21: Representación 3D del Pololu 3pi+.

Para integrar el modelo 3D del vehículo Pololu 3pi+ en Webots es necesario utilizar el nodo denominado “ROBOT” en el espacio de trabajo. Este nodo tiene la función de incorporar el modelo previamente desarrollado del Pololu 3pi+ y permitir su uso dentro del entorno de simulación Webots.



Figura 22: Representación 3D de Pololu 3pi+ en el entorno de Webots.

7.1. Prototipo 1 de base para sensores de distancia

Adicionalmente al modelo 3D del vehículo, se ha creado un modelo 3D que representa la base en la que se colocarán los sensores de distancia. Esta base desempeña un papel fundamental al posicionar de manera precisa los sensores en el centro del modelo del vehículo, lo que garantiza una configuración realista de los sensores.

Este diseño de base para el prototipo 1 implica la incorporación de sensores que permanecen inmóviles con respecto al vehículo. Este conjunto consta de 6 sensores dispuestos en una configuración angular de 60 grados entre sí, lo que les permite abarcar un alcance completo de 360 grados. Para visualizar de manera óptima la distribución de estos sensores, en el modelo 3D se utiliza un hexágono como guía, con cada uno de sus lados coincidiendo con la ubicación de un sensor específico.

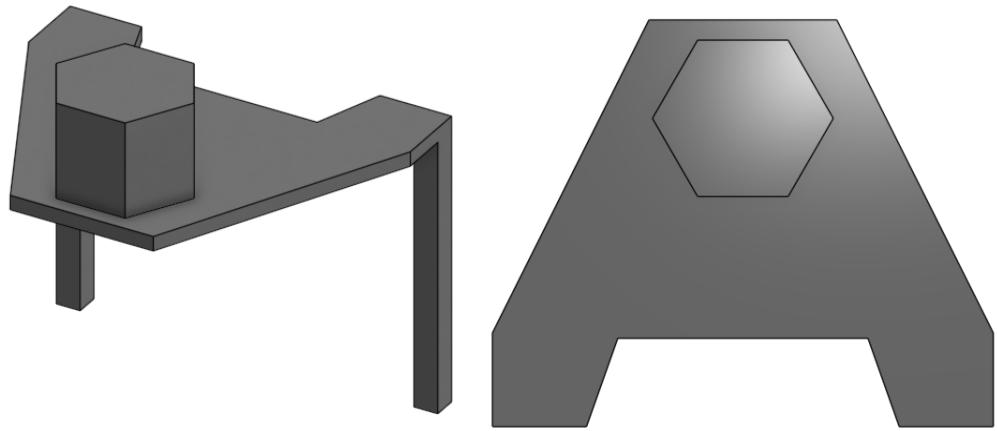


Figura 23: Modelo 3D de base para sensores de distancia para Pololu 3pi+.

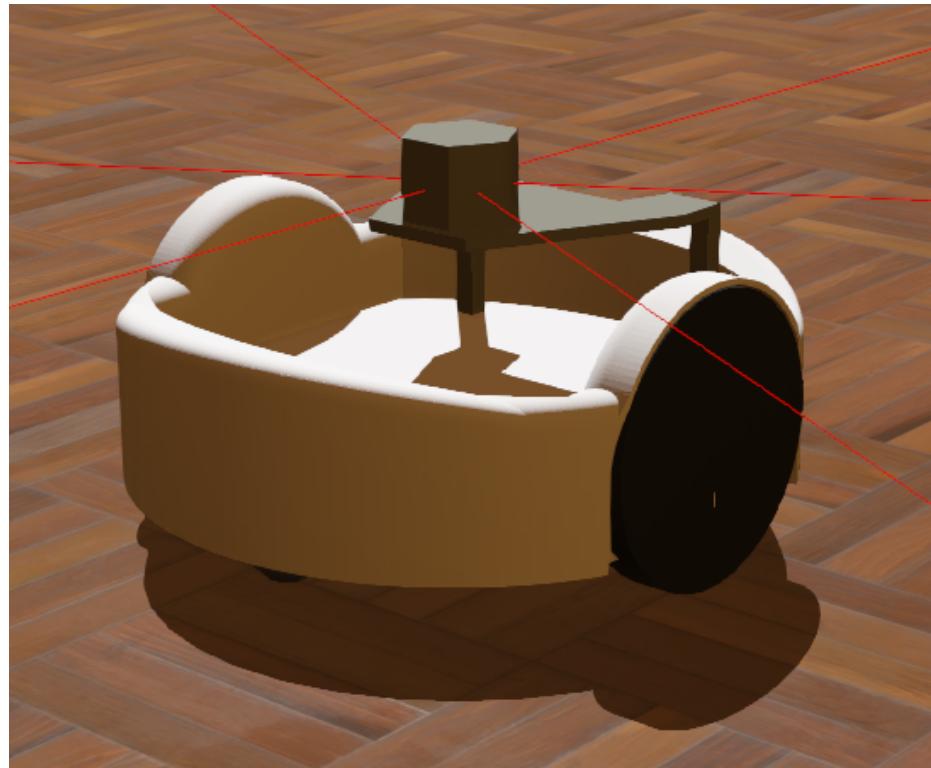


Figura 24: Incorporación de base para sensores de distancia para Pololu 3pi+ en entorno de simulación Webots.

7.2. Prototipo 2 de base para sensores de distancia

El diseño de base para el prototipo 2, al igual que el prototipo 1, implica la incorporación de sensores que permanecen inmóviles con respecto al vehículo.

Este conjunto está compuesto por 6 sensores dispuestos en una distribución angular

específica: 5 sensores frontales están separados por un ángulo de 45 grados entre sí, mientras que hay un único sensor colocado en la parte trasera del vehículo. Esta configuración se diseña con el propósito de tener una cobertura de detección más completa, abarcando mayormente el frente, y ambos lados del vehículo, pero de igual forma sin descuidar la parte trasera del vehículo. Esta distribución de sensores se puede observar en la Figura 26

La distribución angular de 45 grados entre los sensores frontales permite detectar obstáculos en el campo de visión frontal, lo que es crucial para evitar colisiones. El sensor trasero se encarga de supervisar la parte posterior del vehículo, lo que resulta fundamental al determinar giros de 180 grados. En conjunto, esta disposición angular asegura una mejor percepción del entorno y una capacidad de respuesta más completa para el sistema de detección del vehículo.

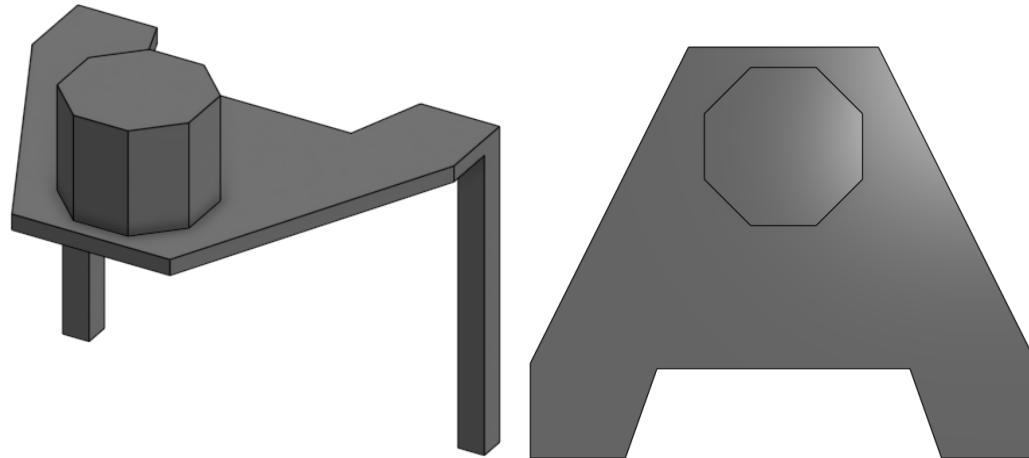


Figura 25: Modelo 3D de base para sensores de distancia para Pololu 3pi+.

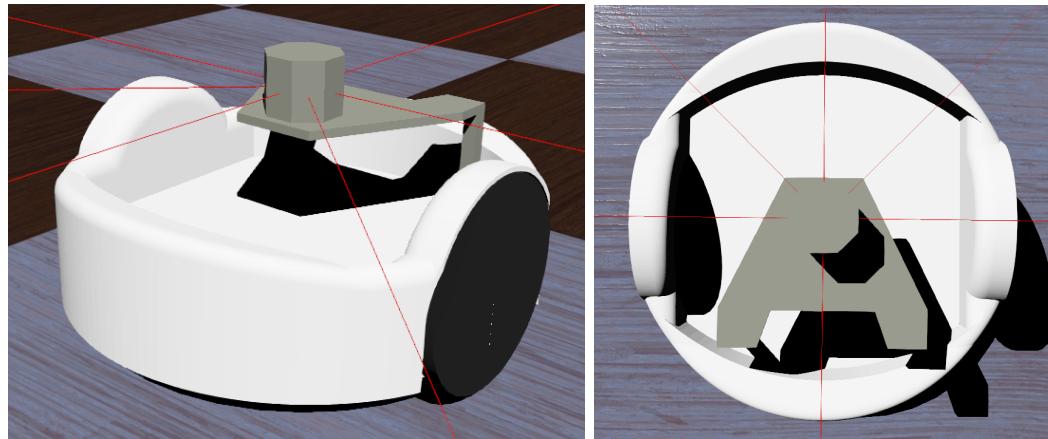


Figura 26: Incorporación del prototipo 2 de la base para sensores de distancia para Pololu 3pi+ en entorno de simulación Webots.

Para el desarrollo y validación de los algoritmos de exploración, mapeo y seguimiento de trayectoria se considera la siguiente identificación de cada uno de los sensores de distancia:

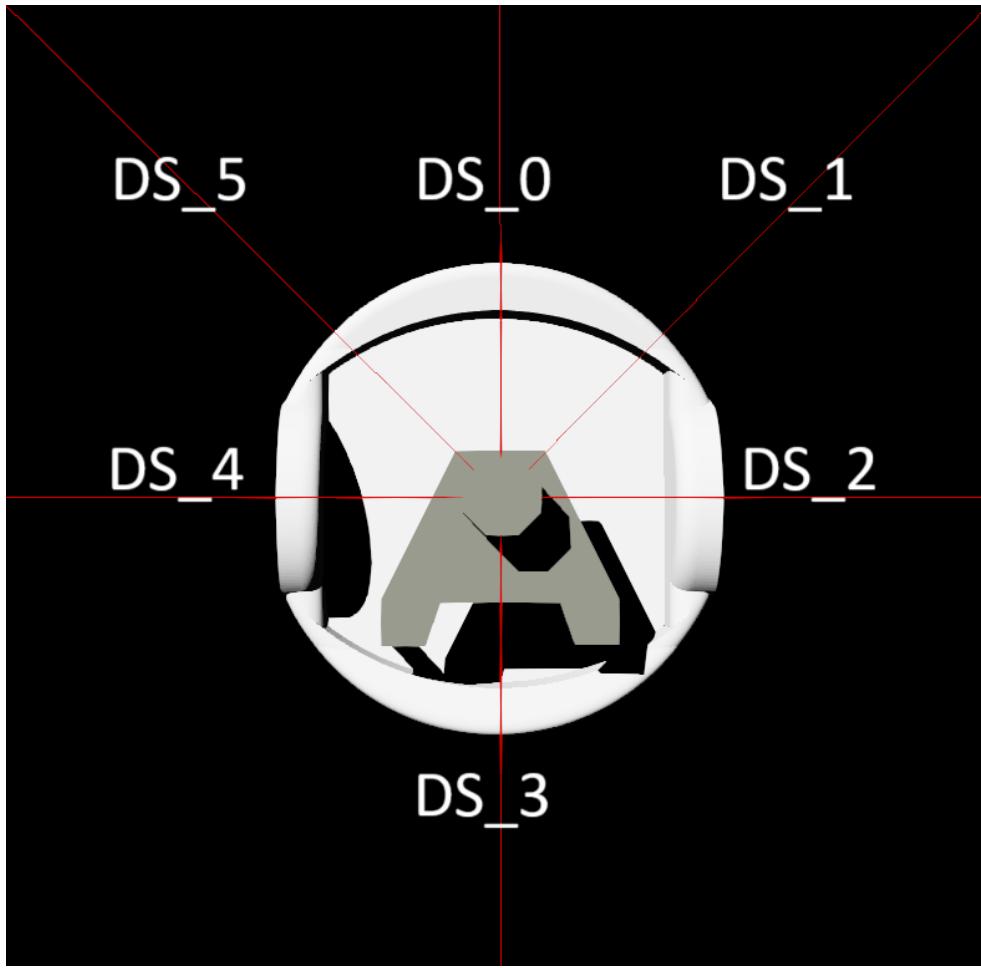


Figura 27: Distribución por nombre de sensores de distancia sobre base en modelo diferencial.

7.3. Sensores de distancia

Webots puede simular una amplia variedad de sensores robóticos estándar. Estos sensores pueden utilizarse para mejorar la percepción del entorno del robot simulado pero utilizando con condiciones reales del sensor.

7.3.1. Sensor de distancia

El sensor de distancia en Webots es un nodo que se puede utilizar para modelar un sensor genérico, un sensor infrarrojo, un sensor de sonar o un medidor de distancia láser. Esta simulación del dispositivo se lleva a cabo detectando las colisiones entre uno o varios rayos del sensor y los objetos en el entorno. Independientemente que sensor sea, el nodo de sensor de distancia tiene las siguientes configuraciones:

1. *LookupTable*: Esta tabla indica cómo el valor medido en la simulación debe mapearse

a los valores de respuesta devueltos por el sensor (la distancia devuelta por la función `wb_distance_sensor_get_value`). La primera columna de la tabla especifica las distancias de entrada, la segunda columna especifica los valores de respuesta deseados correspondientes, y la tercera columna indica la desviación estándar deseada del ruido. El ruido en el valor de retorno se calcula de acuerdo con una distribución de números aleatorios gaussianos, cuyo rango se calcula como un porcentaje del valor de respuesta. Es importante destacar que los valores de entrada de una tabla de búsqueda deben ser siempre positivos y estar ordenados de manera creciente.

Se presenta el siguiente ejemplo:

```
lookupTable [ 0      1000  0,
               0.1    1000  0.1,
               0.2    400   0.1,
               0.3    50    0.1,
               0.37   30    0 ]
```

Figura 28: Ejemplo de parámetros para un sensor de distancia [9].

La tabla anterior indica que, para una distancia de 0 metros, el sensor devolverá un valor de 1000 sin ruido, para una distancia de 0.1 metro, el sensor devolverá 1000 con un ruido de desviación estándar del 10%; para un valor de distancia de 0.2 metros, el sensor devolverá 400 con una desviación estándar del 10%, y así sucesivamente. Los valores de distancia que no estén especificados directamente en la tabla de búsqueda se interpolan linealmente. Esto se puede entender mejor en la siguiente figura.

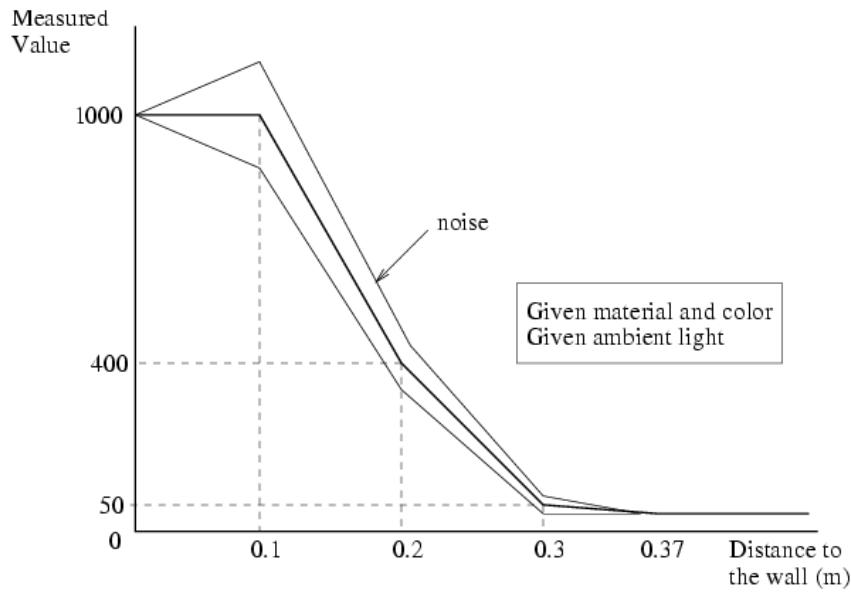


Figura 29: Respuesta del sensor vs. distancia del obstáculo [9].

2. Número de rayos: Es el número de rayos emitidos por el sensor. El número de rayos debe ser igual o mayor que 1 para los sensores “infrarrojos” y “sonar”. Para los sensores “láser”, el número de rayos debe ser exactamente 1.

Para un sensor, si el número de rayos es mayor que 1, el valor de medición del sensor se calcula a partir del promedio ponderado de las respuestas de los rayos individuales. Al utilizar múltiples rayos, se obtiene un modelo más preciso del sensor infrarrojo o sonar físico. Los rayos del sensor se distribuyen dentro de conos tridimensionales cuyos ángulos de apertura se pueden ajustar mediante el campo de apertura.

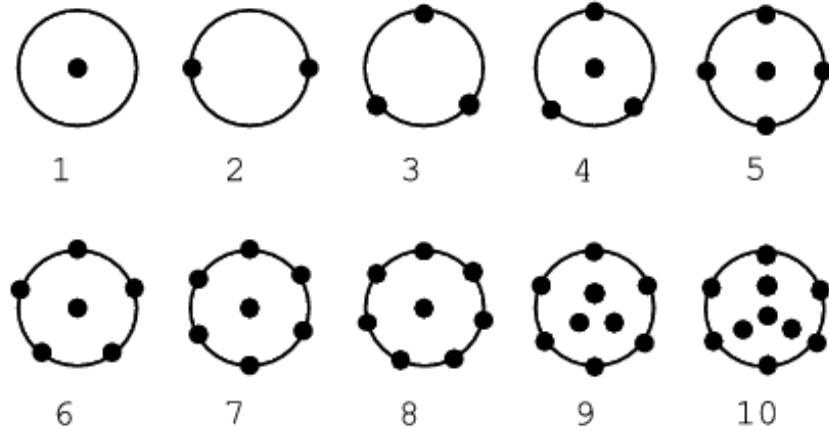


Figura 30: Distribución de rayos en sensores de distancia [9].

En la Figura 30 se puede ver las distribuciones de rayos de uno a diez. La distribución espacial de los rayos es lo más uniforme posible y presenta una simetría izquierda/derecha. No hay un límite superior en el número de rayos; sin embargo, el rendimiento de Webots disminuye a medida que aumenta el número de rayos.

3. Apertura: Este es el ángulo de apertura del sensor o radio del haz láser. Para los tipos de sensor infrarrojo y sonar, este campo controla el ángulo de apertura, en radianes, del cono de rayos cuando se utilizan múltiples rayos. Para el sensor láser, este campo especifica, en metros, el radio del punto rojo dibujado donde el haz láser impacta un obstáculo.

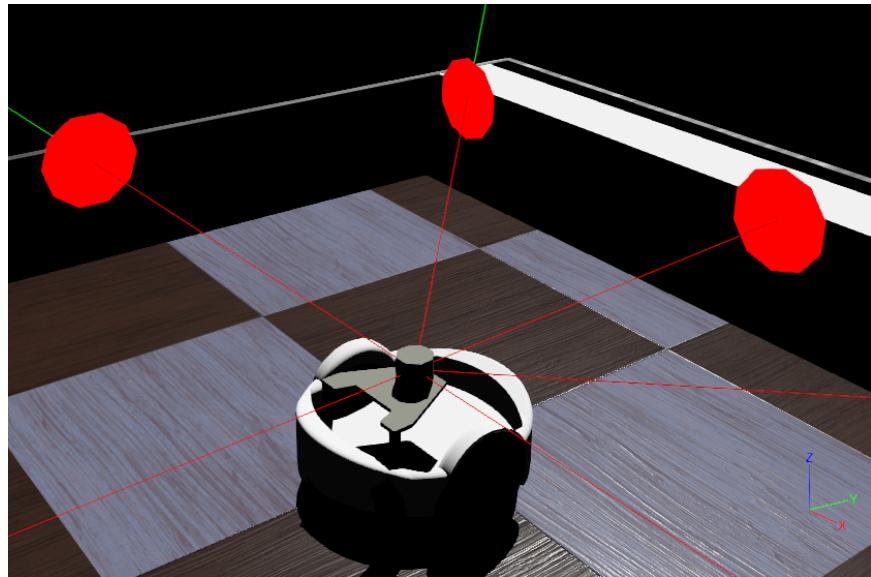


Figura 31: Apertura de 0.05 m del sensor láser.

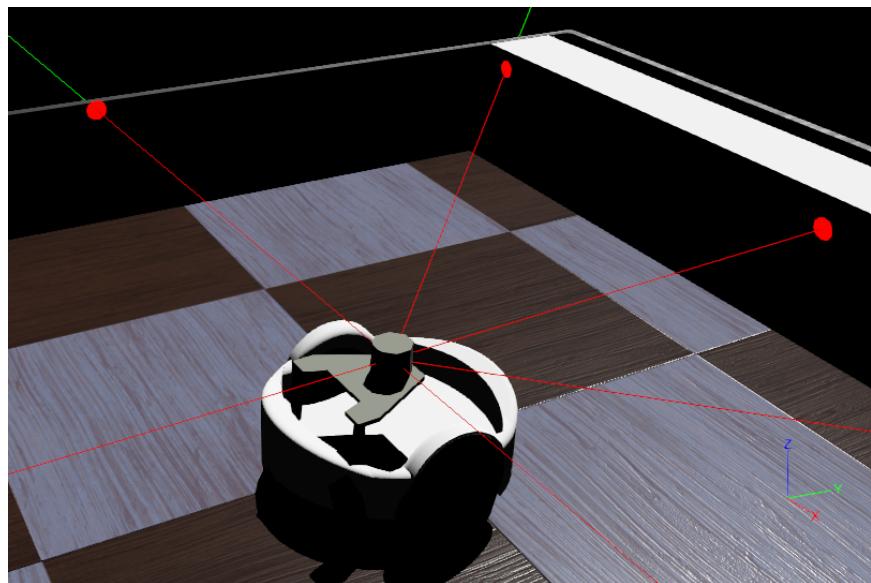


Figura 32: Apertura de 0.01 m del sensor láser.

7.3.2. Consideración para sensor de distancia tipo sonar

Para los sensores sonares el valor de retorno dependerá de los parámetros ingresados en la *lookupTable*, es decir, el valor correspondiente al alcance del sensor sonar, esto si el ángulo de incidencia es mayor de 22.5 grados ($\pi/8$ radianes). En otras palabras, los rayos del sonar que se encuentren fuera del cono de reflexión con una apertura de 45 grados nunca regresarán y, por lo tanto, se perderán en el cálculo de la distancia.

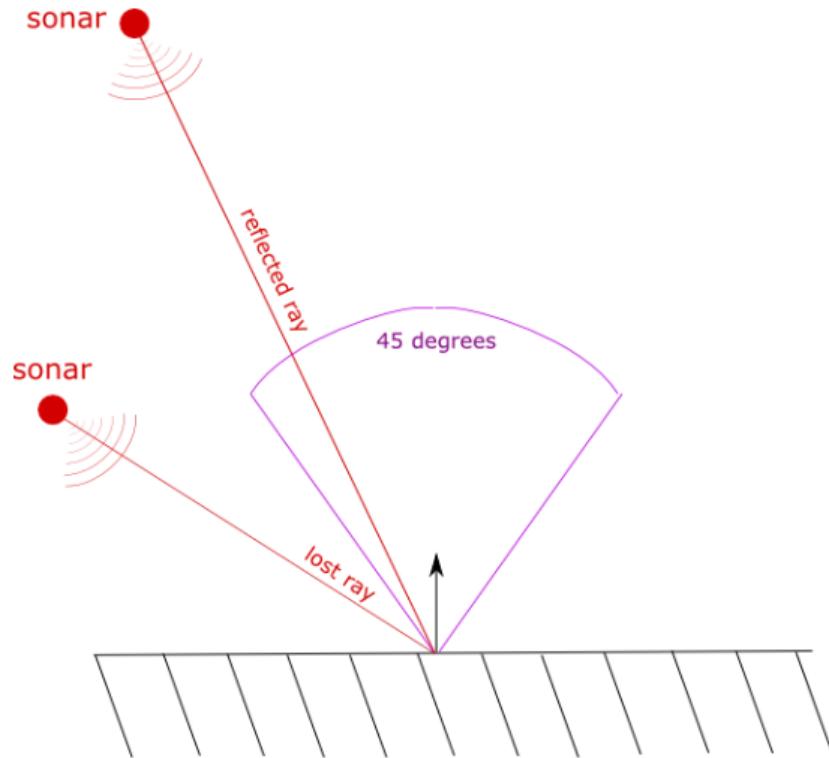


Figura 33: Respuesta del sensor sonar con base al ángulo de apertura [9].

7.4. Otros sensores del vehículo diferencial

El vehículo diferencial no solo está equipado con sensores de distancia, sino que también se han incorporado sensores adicionales que proporcionan información para el desarrollo y complemento en la toma de decisiones de los algoritmos.

7.4.1. Sensores de posición de ruedas

El sensor de posición es un nodo que puede utilizarse en una simulación mecánica para supervisar la posición de una articulación. En este trabajo se utiliza este sensor para supervisar la posición de las ruedas del vehículo diferencial. El sensor de posición se insertó en nodos de articulación de un solo eje de rotación, como se observa en la Figura 34. Dada la articulación, el sensor medirá la posición angular en radianes. Esto se hizo para cada una de las ruedas.

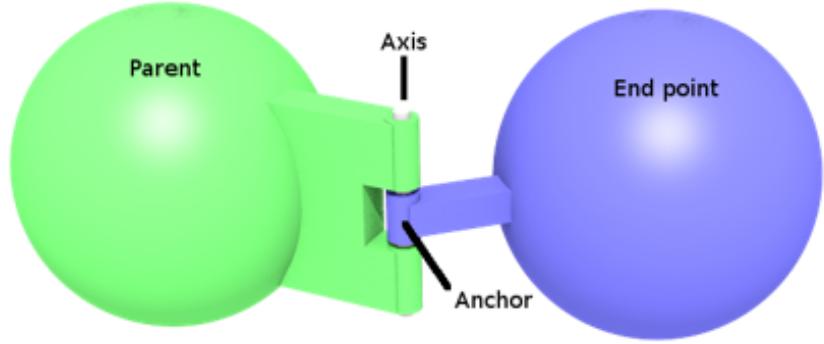


Figura 34: Articulación con un solo eje de rotación [9].

Brújula:

El vehículo diferencial incorpora una brújula digital. La brújula en Webots es un nodo que se puede utilizar para modelar una brújula digital de 1, 2 o 3 ejes. Este nodo devuelve un vector que indica la dirección norte especificada por el campo de coordenadas del mundo en Webots.

CAPÍTULO 8

Mundo simulado en Webots

El espacio de trabajo creado para realizar las pruebas de los algoritmos desempeña un papel crítico en el proceso de validación. El espacio de trabajo es un entorno controlado que garantiza resultados confiables y reproducibles. Validar los algoritmos en un espacio de trabajo adecuado permite evaluar el rendimiento de manera precisa y objetiva. Esto es esencial para confirmar que los algoritmos cumplen con los requisitos y expectativas previamente definidos, lo que a su vez aumenta la confianza en su capacidad para funcionar en situaciones del mundo real.

La representación de entornos diversos en el espacio de trabajo es esencial para abordar la variabilidad que los algoritmos pueden encontrar en situaciones reales. Los espacios deben capturar no solo situaciones típicas, sino también casos extremos y situaciones inesperadas que podrían poner a prueba la robustez y adaptabilidad de los algoritmos. La diversidad en el espacio de trabajo permite evaluar cómo los algoritmos responden a diferentes condiciones y desafíos, lo que resulta en una validación más completa y útil.

Para el desarrollo del entorno de pruebas también se consideró la facilidad de edición. Los espacios de trabajo no son estáticos; a menudo, es necesario ajustar las condiciones de prueba, modificar parámetros o agregar nuevos elementos de evaluación a medida que se desarrollan los algoritmos. Un espacio de trabajo que sea fácil de editar y adaptar permite realizar ajustes según sea necesario sin comprometer la integridad de las pruebas. Esto facilita la iteración y la mejora continua de los algoritmos a medida que se recopilan más datos y se obtienen nuevos conocimientos sobre su funcionamiento.

En las Figura 35 y Figura 36 se presenta el entorno destinado a llevar a cabo las pruebas, el cual se caracteriza por contar con dos tipos distintos de espacios: uno que comprende pasillos y otro que se distingue por su amplitud. La configuración de estas áreas se logró mediante la instalación de paredes con especificaciones precisas: un grosor de 0.08 metros y una altura de 0.1 metros. La longitud de estas paredes varía en función del segmento al que

están asignadas, lo que contribuye a la diversificación del entorno.

Adicionalmente, se incorporaron tres cajas como obstáculos en el escenario. Estas cajas tienen dimensiones uniformes de 0.3 metros de ancho, 0.3 metros de largo y 0.3 metros de alto, lo que agrega una complejidad adicional a las pruebas al generar obstáculos tridimensionales.

Para facilitar la orientación y referencia en el entorno, se ha introducido un marcador de color verde. Este marcador representa el punto de inicio de un vehículo, sirviendo como punto de partida para las actividades y pruebas que se llevarán a cabo en este espacio.

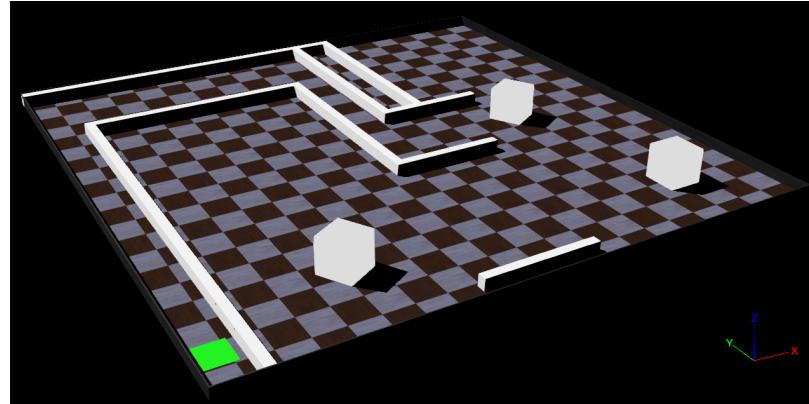


Figura 35: Mundo simulado en Webots.

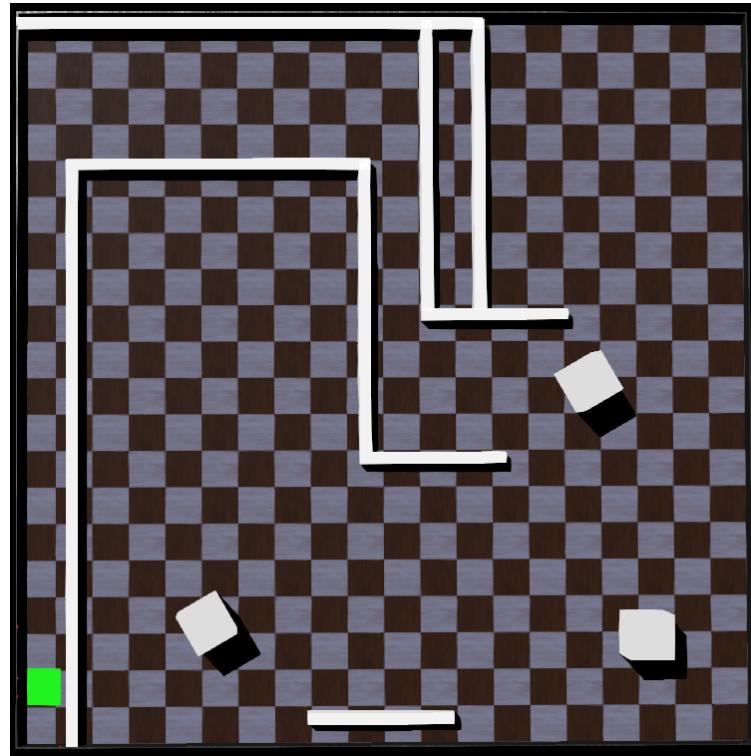


Figura 36: Mundo simulado en Webots.

Para llevar a cabo la validación de los algoritmos desarrollados, se empleó el mapa representado en la Figura 37, el cual se ha subdividido en un total de 8 secciones estratégicamente definidas. Esta división en secciones simplificó el proceso de detección y análisis de las áreas cubiertas durante las múltiples simulaciones que se ejecutaron. Subdividir el mapa en secciones específicas fue esencial para el proceso de validación, ya que permitió evaluar de manera precisa cómo se comportan los algoritmos cuando el vehículo se enfrenta a diferentes transiciones de secciones en el entorno.

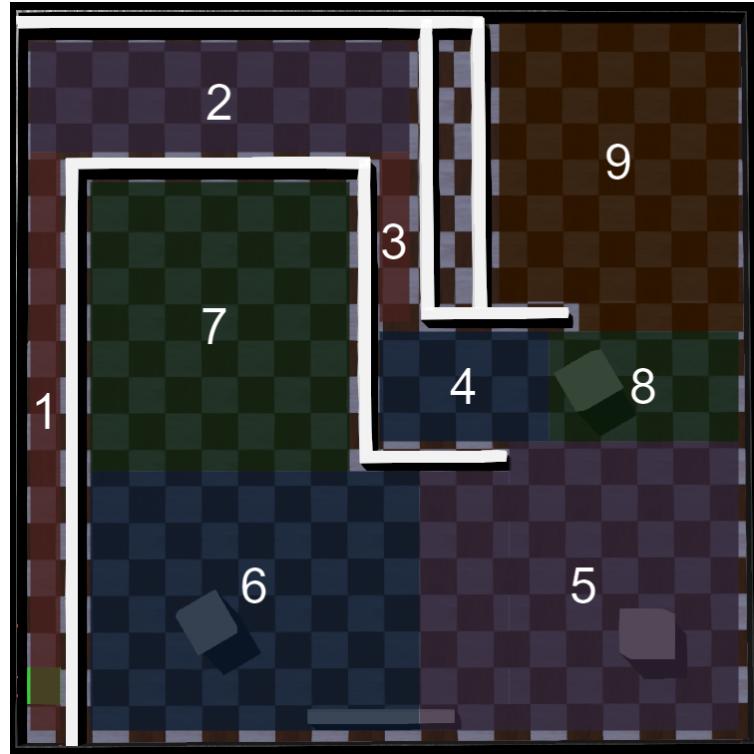


Figura 37: Secciones del mundo simulado en Webots.

En la Figura 37, se puede observar la distribución de estas secciones. Es importante destacar que las secciones 1 y 3 se consideraron como “pasillos”, dado que el espacio disponible solo permite el tránsito de un vehículo a la vez en estas áreas. Por otro lado, se consideraron las secciones restantes como “espacios abiertos”, ya que en estas áreas es posible que varios vehículos circulen simultáneamente sin restricciones.

CAPÍTULO 9

Mapeo de entorno

En este capítulo, se establecen los parámetros necesarios para realizar el mapeo 2D del entorno. Para abordar el problema del mapeo, se considera el modelo del robot diferencial mencionado previamente, junto con el desarrollo de un algoritmo que realice trayectorias de exploración, y detección de obstáculos. Es importante señalar que los algoritmos de exploración y mapeo del entorno pueden ser ejecutados por un solo agente robótico, por lo tanto, en este capítulo, las pruebas se enfocarán en validar el rendimiento utilizando un único agente robótico. En capítulos posteriores, se llevarán a cabo pruebas con sistemas robóticos multi-agente.

9.1. Algoritmo de exploración - Prototipo 1

En la Figura 38 se muestra el proceso de exploración en términos generales.

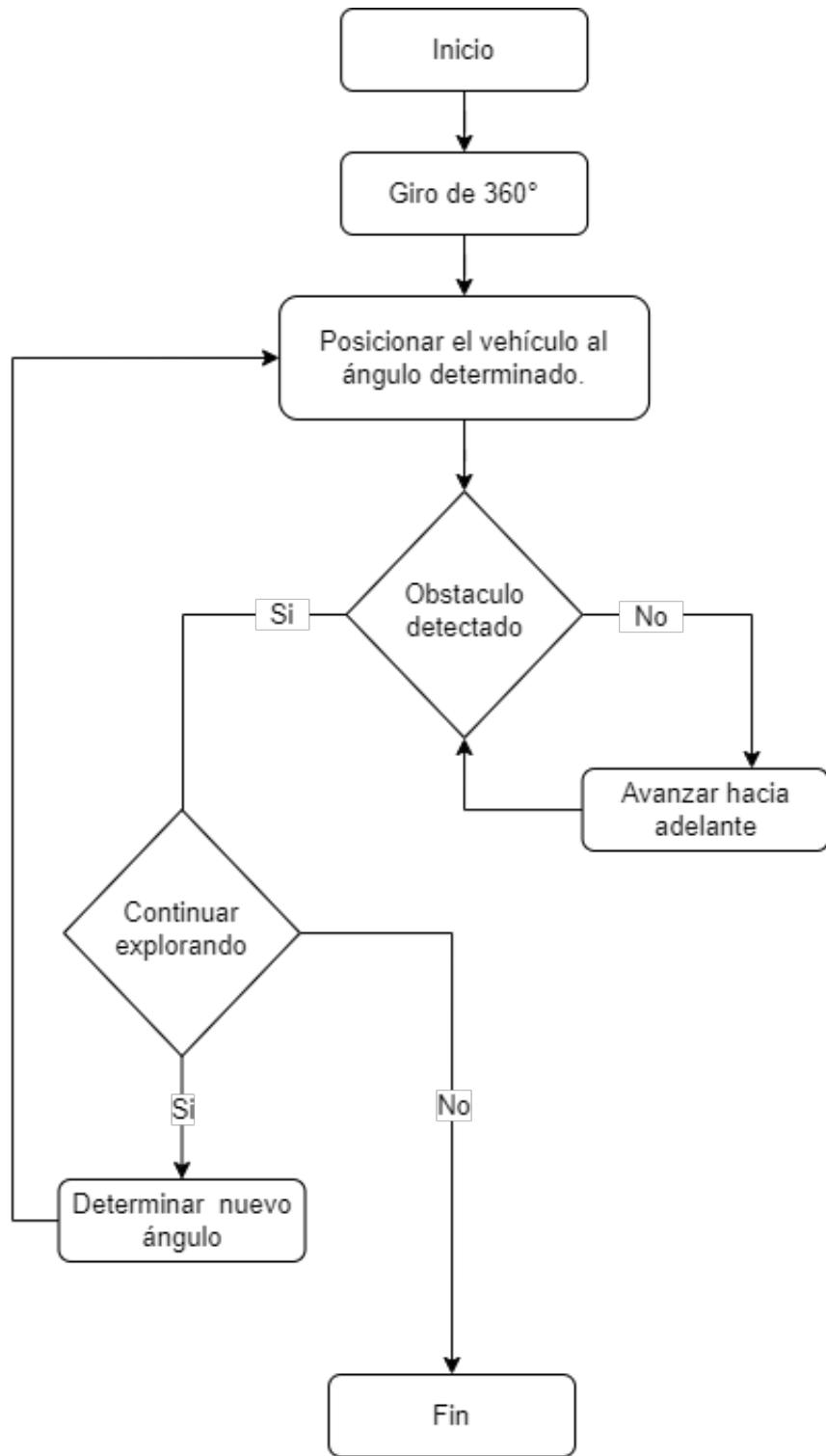


Figura 38: Diagrama de flujo general de algoritmo de exploración prototipo 1.

9.1.1. Posicionamiento de vehículo

Como se puede ver en la Figura 38, los desplazamientos del vehículo son únicamente en línea recta. Avanzar en línea recta permite determinar la distancia recorrida con una rueda y validar la distancia con la distancia obtenida con la otra rueda.

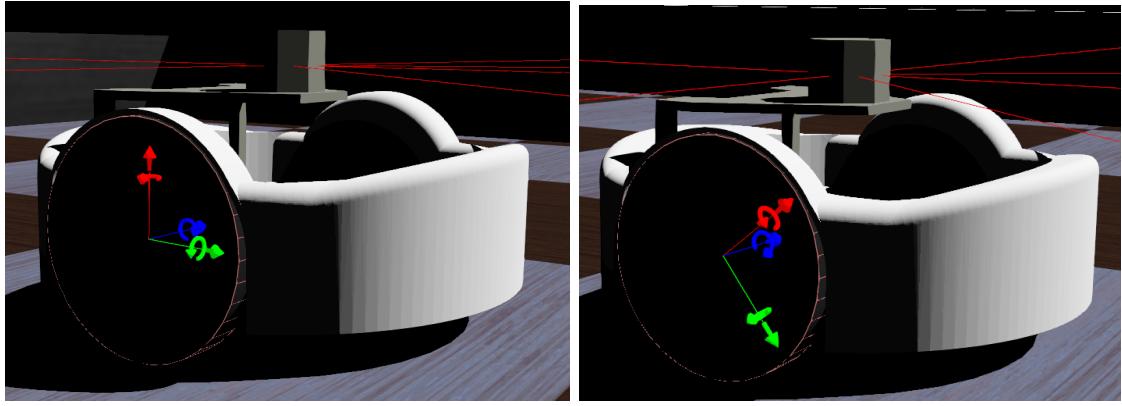


Figura 39: Cambio de ángulo de la rueda respecto a un desplazamiento.

En la Figura 39 se observa el cambio de ángulo en las ruedas cuando se hace un movimiento, en este algoritmo en línea recta. Este cambio de ángulo se utiliza la Ecuación (13) para determinar la distancia recorrida.

$$D = \Delta\Theta \cdot r \quad (13)$$

Donde:

- D : Es la distancia recorrida en m.
- $\Delta\Theta$: Es el cambio de ángulo de la rueda en radianes.
- r : Es el radio de la rueda en m.

En el diagrama de la Figura 38 se observa que, la distancia estimada es desde el punto en que se determina un nuevo ángulo para el vehículo hasta el siguiente punto en el cual se vuelve a determinar el ángulo del vehículo. Este proceso de estimación de distancia resulta en un componente de dos para conocer la posición. Para obtener el segundo componente, se utiliza el módulo de brújula, el cual brinda la orientación del vehículo diferencial respecto al mundo simulado, esto con la función `wb_compass_get_values`. Esta función devuelve un vector que indica la dirección del norte respecto al mundo simulado. Por lo que para obtener la rotación del vehículo se aplica la siguiente conversión:

$$\text{rot } z = \arctan\left(\frac{\text{wb_compass_get_values}[0]}{\text{wb_compass_get_values}[1]}\right) \quad (14)$$

Donde:

- $rot z$: Ángulo de rotación del vehículo sobre el eje z en radianes.
- $wb_compass_get_values[0]$: Representa la componente en la dirección x (horizontal) del vector magnético detectado por el sensor de brújula en la simulación.
- $wb_compass_get_values[0]$: representa la componente en la dirección y (vertical) del vector magnético detectado por el sensor de brújula en la simulación.

Utilizando la distancia recorrida y la orientación durante el trayecto, se calculan los desplazamientos en los ejes x e y . Usando trigonometría, se descompone la distancia recorrida total en las dos componentes.

$$\text{Componete } x = \cos(rotZ) \cdot D \quad (15)$$

$$\text{Componete } y = \sin(rotZ) \cdot D \quad (16)$$

En la Ecuación (15) y Ecuación (16) se muestra como se determina los desplazamientos en x e y .

En la Figura 40 se muestra un desplazamiento en linea recta estando el vehículo a 45° . Con este movimiento se estima el desplazamiento en x e y , en la Figura 41 se observa la representación del movimiento realizado.

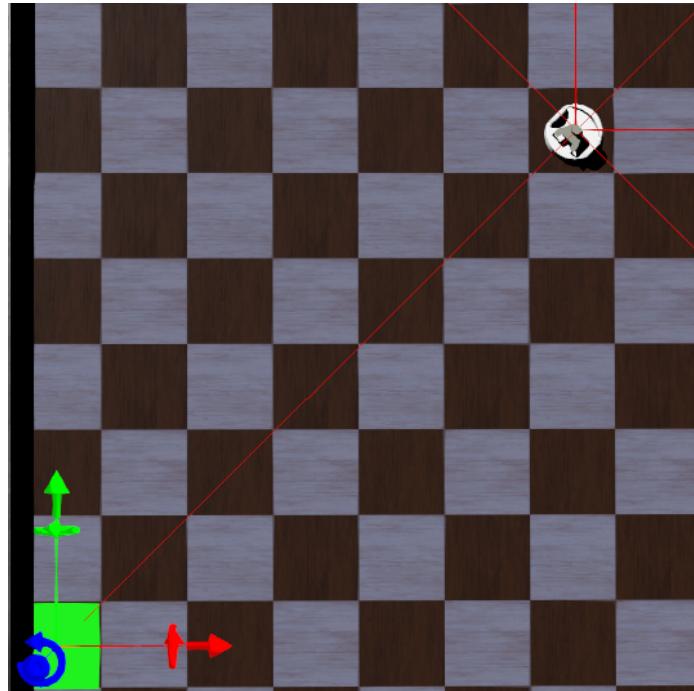


Figura 40: Trayectoria de exploración en entorno de simulación Webots, 1 segmento de linea recta con 45° de rotación

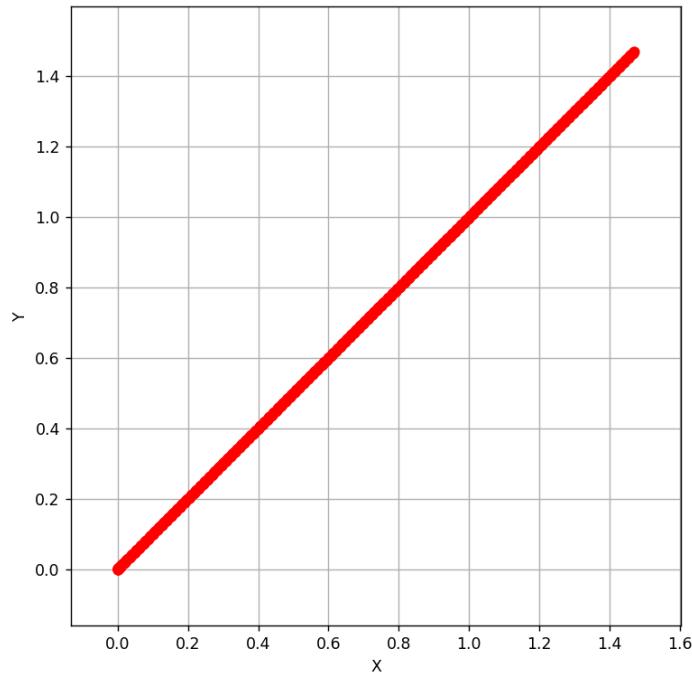


Figura 41: Trayectoria de exploración estimada, 1 segmento de linea recta con 45° de rotación, unidades en m.

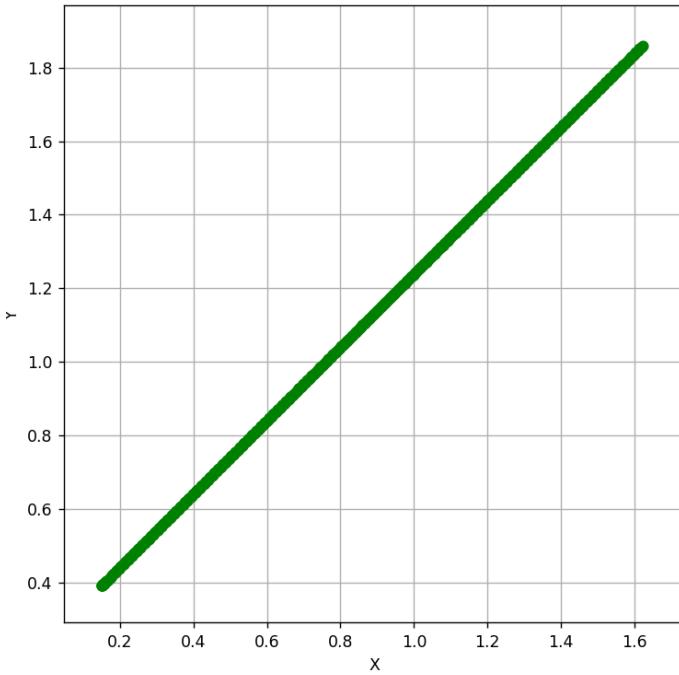


Figura 42: Trayectoria de exploración obtenida por GPS, 1 segmento de linea recta con 45° de rotación, unidades en m.

En la Figura 41, se inicia la estimación de la trayectoria desde las coordenadas $(0, 0)$. Sin embargo, en el entorno de simulación, el vehículo no se encuentra en estas coordenadas,

como se muestra en la Figura 40. Para facilitar la visualización precisa de la trayectoria, en la Figura 42 se presenta la misma ruta, pero con las coordenadas correspondientes al entorno de simulación.

Es importante señalar que los algoritmos se ejecutan con referencia a la trayectoria estimada, es decir, el punto de inicio de la trayectoria se establece en $(0, 0)$. Sin embargo, para validar los resultados, se utiliza el módulo GPS del vehículo para compararlos con la trayectoria estimada. Para llevar a cabo esta comparación, antes de iniciar el movimiento, se registra la posición del vehículo mediante el GPS. Las coordenadas obtenidas en ese punto inicial se consideran como desfases. A lo largo de la simulación, estos desfases se restan a las coordenadas del GPS, lo que permite obtener tanto un valor real como un valor estimado. De esta manera, es posible calcular el porcentaje de error en la estimación y validar la precisión del algoritmo utilizado.

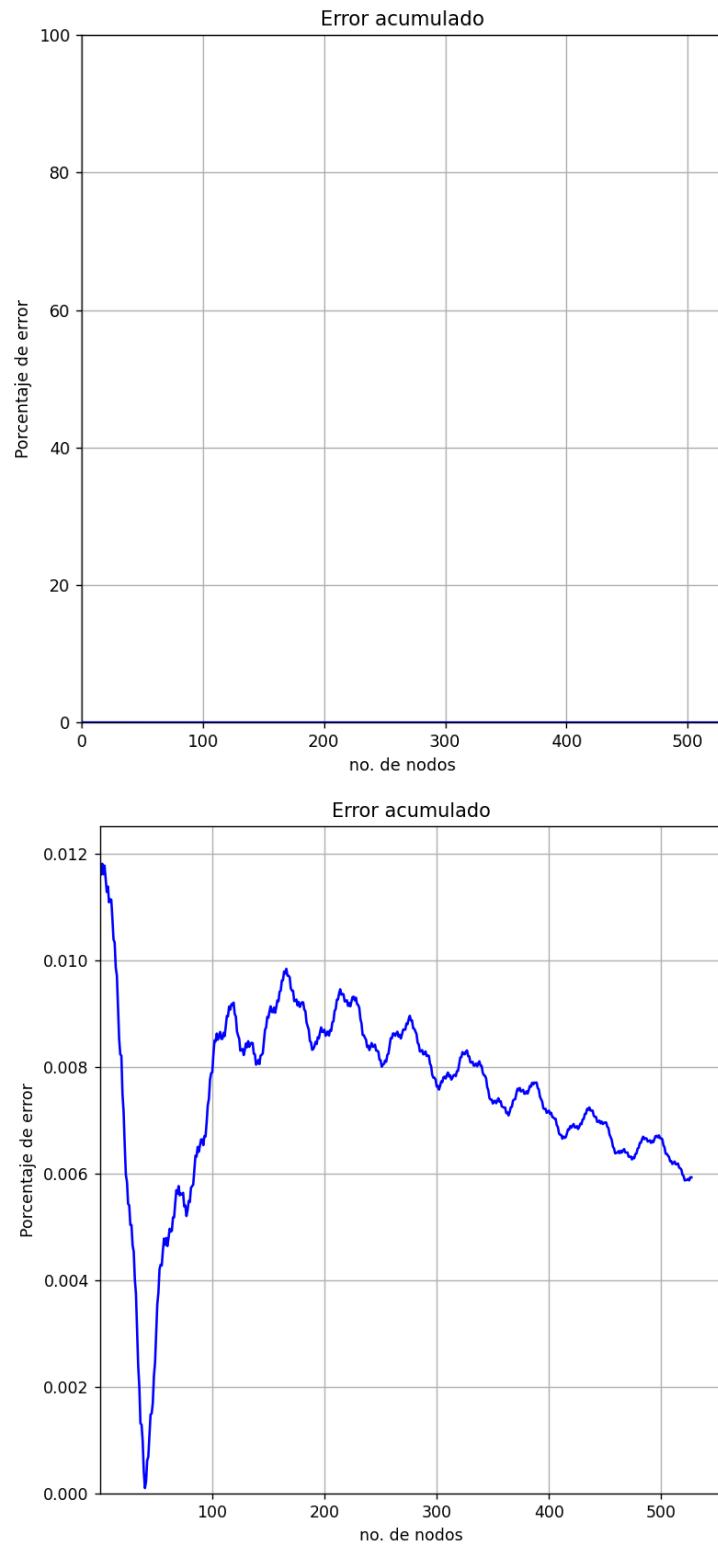


Figura 43: Error acumulado en trayectoria de exploración, 1 segmento de linea recta con 45° de rotación.

En la Figura 43 se muestra el error de la trayectoria estimada comparado con el valor del GPS. El eje x de la Figura 43 se denota como número de nodos. Los nodos son los puntos en los que se evalúa si existe un nuevo obstáculo. En la prueba realizada no se detecta obstáculos por lo que no hay un cambio de ángulo, las condiciones para cambio de ángulo se determinan posteriormente. El eje y de la Figura 43 es el error acumulado. Como se puede observar, para esta prueba sin cambio de dirección, el error no es mayor del 0.012 %.

9.1.2. Condiciones de giro

Cuando el vehículo se encuentra en movimiento, constantemente evalúa su entorno a través de la distribución de sensores de distancia, como se ilustra en la Figura 27. Cuando un sensor detecta un obstáculo, se compara el valor de los otros sensores, y el algoritmo toma una decisión dependiendo de los resultados de dicha comparación. De este modo, se determina cual es la orientación donde posiblemente sea mejor opción para transitar.

En la Figura 44 se observa el bucle en el que se encuentra el vehículo cuando esta realizando un desplazamiento. El vehículo avanza hacia adelante, y esta constantemente obteniendo los datos de los sensores ds_0 , ds_2 y ds_4 . También se contempla una distancia mínima para considerar cambiar la orientación.

El valor de la distancia mínima se compara con el valor del sensor ds_0 para determinar si se detiene el proceso de exploración. Este valor de distancia mínima puede adoptar uno de dos posibles valores, dependiendo de las mediciones de los sensores ds_2 y ds_4 . Si ambos sensores, ds_2 y ds_4 , registran distancias menores a 100 cm, el valor de distancia mínima es de 30 cm. Por otro lado, si ds_2 y ds_4 registran distancias iguales o mayores a 100 cm, el valor de distancia mínima es de 100 cm.

Esta comparación de las lecturas de los sensores ds_2 y ds_4 determina si el vehículo se encuentra en un entorno de exploración amplio y abierto, o si está operando en un espacio más restringido, como un pasillo estrecho. La elección de la distancia mínima se adapta en consecuencia, lo que permite una respuesta adecuada del vehículo ante las variaciones en el entorno, asegurando una exploración en función de las condiciones específicas en las que se encuentra.

El proceso de evaluación de los valores de los sensores consiste en comparar los sensores laterales, ds_2 y ds_4 , con el valor del sensor frontal, ds_0 . La interpretación de esta comparación implica identificar un espacio lateral que presente mejores oportunidades de exploración en lugar de continuar avanzando en línea recta hacia adelante. Se usan los sensores laterales, ds_2 y ds_4 , ya que, el cambio de giro puede ser de 90° o -90°, lo cual garantiza el libre transito del vehículo en dicho cambio de orientación.

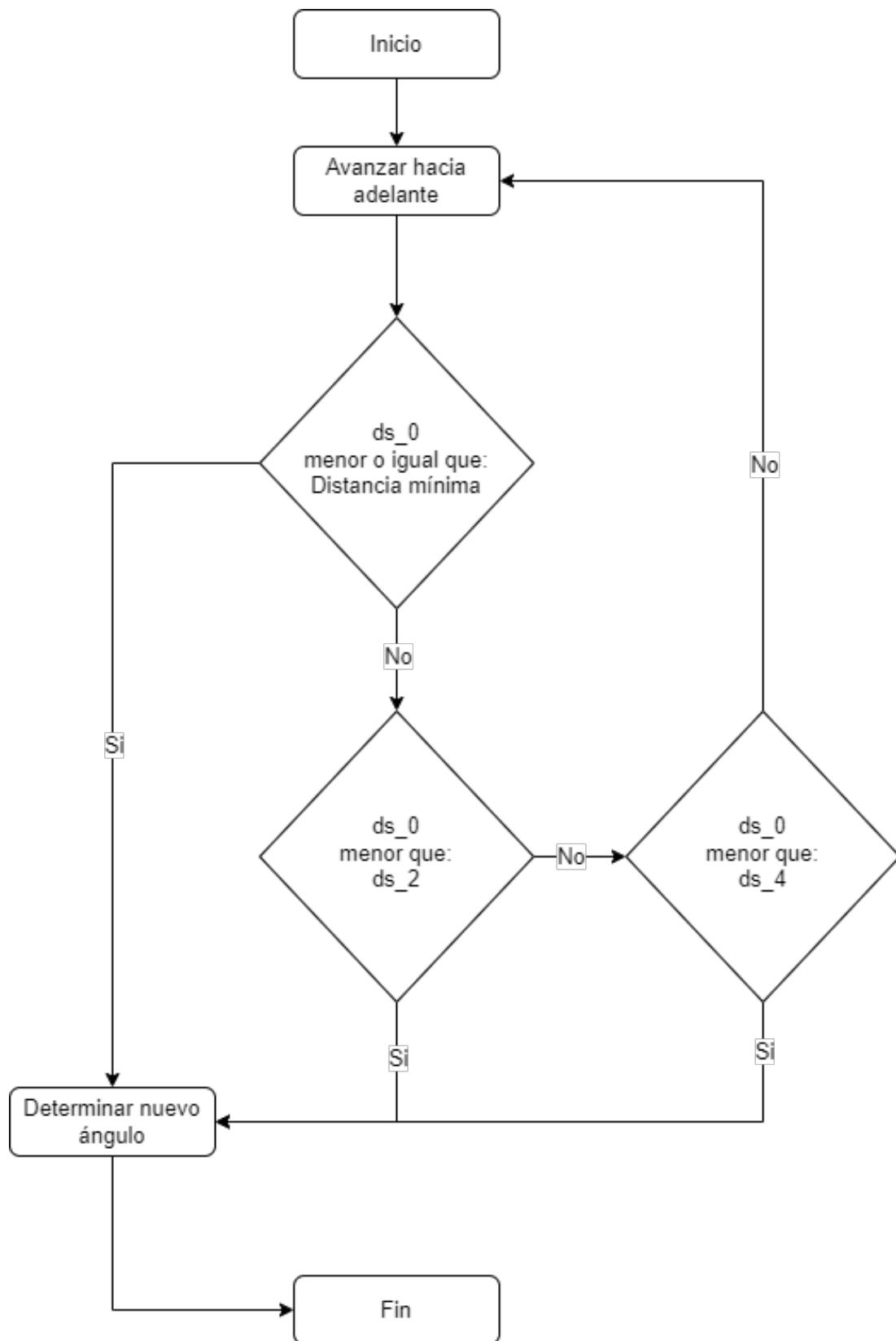


Figura 44: Diagrama de flujo para considerar un cambio de giro en algoritmo de exploración.

Para determinar cuál es el cambio de giro que se debe hacer, se consideran las siguientes condiciones comunes a las que se enfrenta el vehículo en el proceso de exploración.

En la Figura 45 se observa el caso cuando el valor del sensor frontal, ds_0 , es mayor que el valor de los sensores laterales, ds_2 y ds_4 . Como se mencionó previamente, este caso corresponde a continuar avanzando hacia adelante. En la Figura 46 se observa el caso cuando el valor del sensor frontal, ds_0 , es menor que el valor del sensor lateral, ds_2 . Siguiendo el diagrama de la Figura 44, este caso corresponde para considerar un cambio de giro. En la Figura 47 se observa el caso cuando el valor del sensor trasero, ds_3 , es mayor que el valor del resto de sensores. Usando el diagrama de la Figura 44, se considera cambiar de ángulo cuando el valor del sensor frontal, ds_0 , sea menor a la distancia mínima establecida.

Para determinar la dirección del giro necesario, se toman en cuenta los escenarios mencionados previamente y se analizan los valores de los otros sensores de distancia. Es importante notar que el vehículo siempre mantiene un ángulo de rotación constante. En consecuencia, las condiciones detectadas pueden resultar en un ajuste de giro de 45 grados, ya sea incrementando o disminuyendo el ángulo existente. El único caso en el que se hace un ajuste de 180° es cuando el valor del sensor de distancia trasero, ds_3 , es mayor al valor del resto de sensores de distancia.

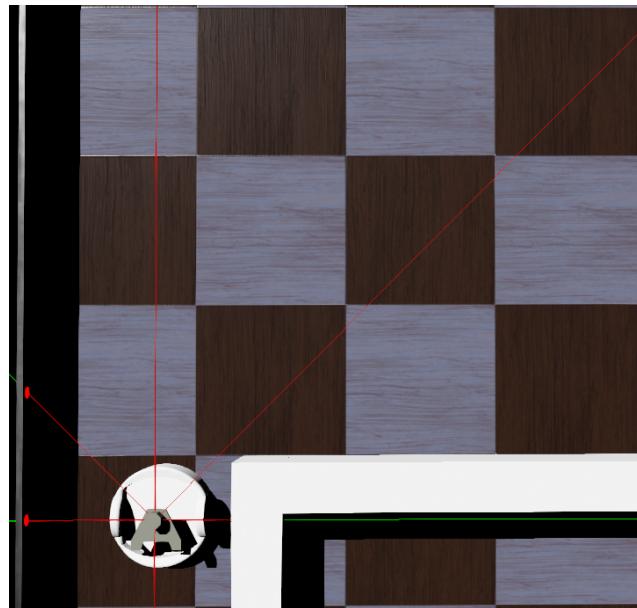


Figura 45: Valor de sensor frontal mayor que lateral.

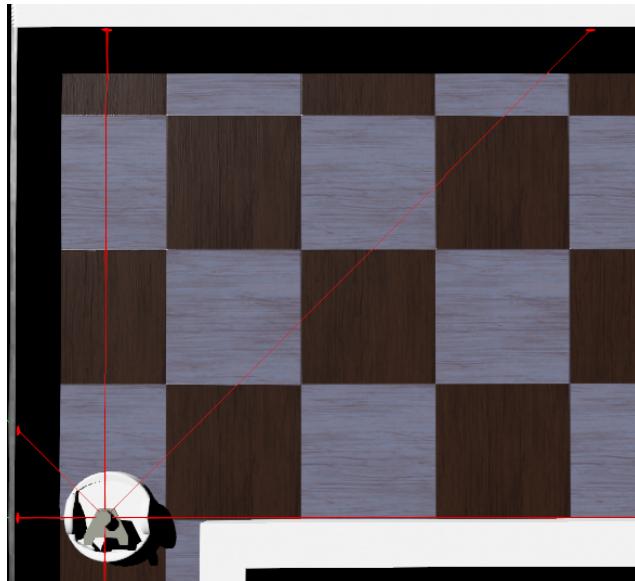


Figura 46: Valor de sensor frontal menor que lateral.

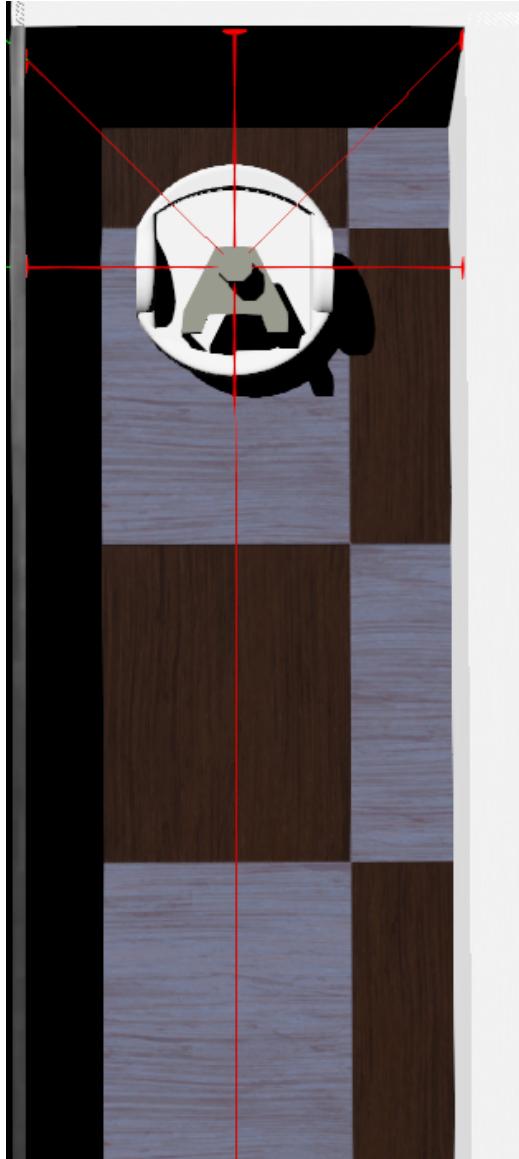


Figura 47: Valor de sensor trasero mayor a todos los demás sensores.

Para evaluar la capacidad de cambio de dirección y desplazamiento del vehículo, se utiliza el entorno de simulación representado en la Figura 48. Este entorno consta de un tramo inicial en línea recta, seguido por un espacio más amplio que facilita la realización de pruebas de cambio de orientación del vehículo.

La Figura 49 presenta la trayectoria estimada durante la exploración con cambios de orientación. De manera similar, en la Figura 50, se muestra la trayectoria registrada mediante el GPS del vehículo. Se observa que ambas trayectorias son similares, lo que se confirma mediante la Figura 51, donde el error acumulado se mantiene por debajo del 0.25 %.

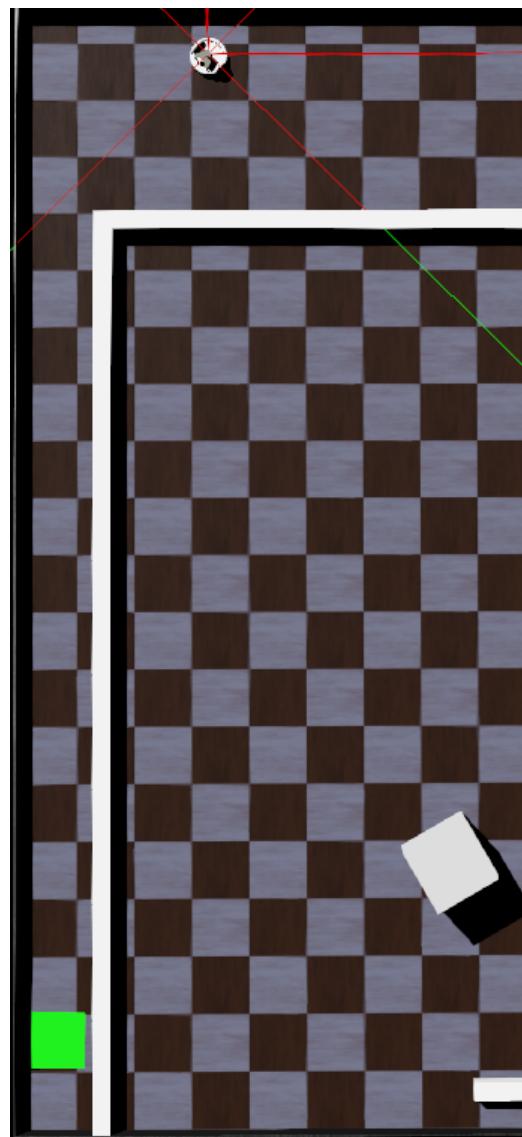


Figura 48: Espacio del entorno de simulación en Webots para realizar trayectoria de exploración con cambio de orientación

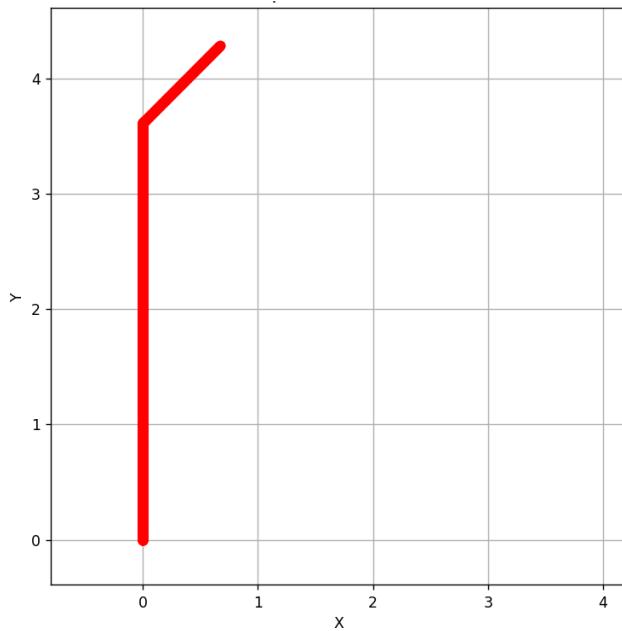


Figura 49: Trayectoria estimada de exploración con cambio de orientación del vehículo, unidades en mt

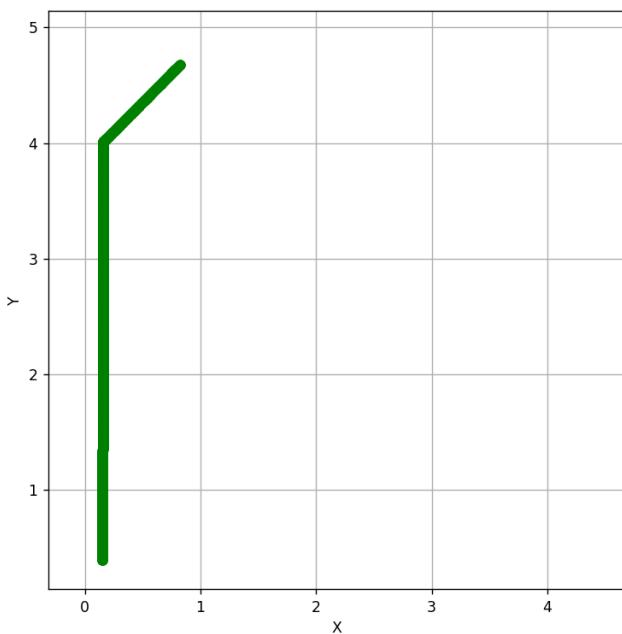


Figura 50: Trayectoria de exploración con cambio de orientación del vehículo, obtenida con GPS, unidades en mt.

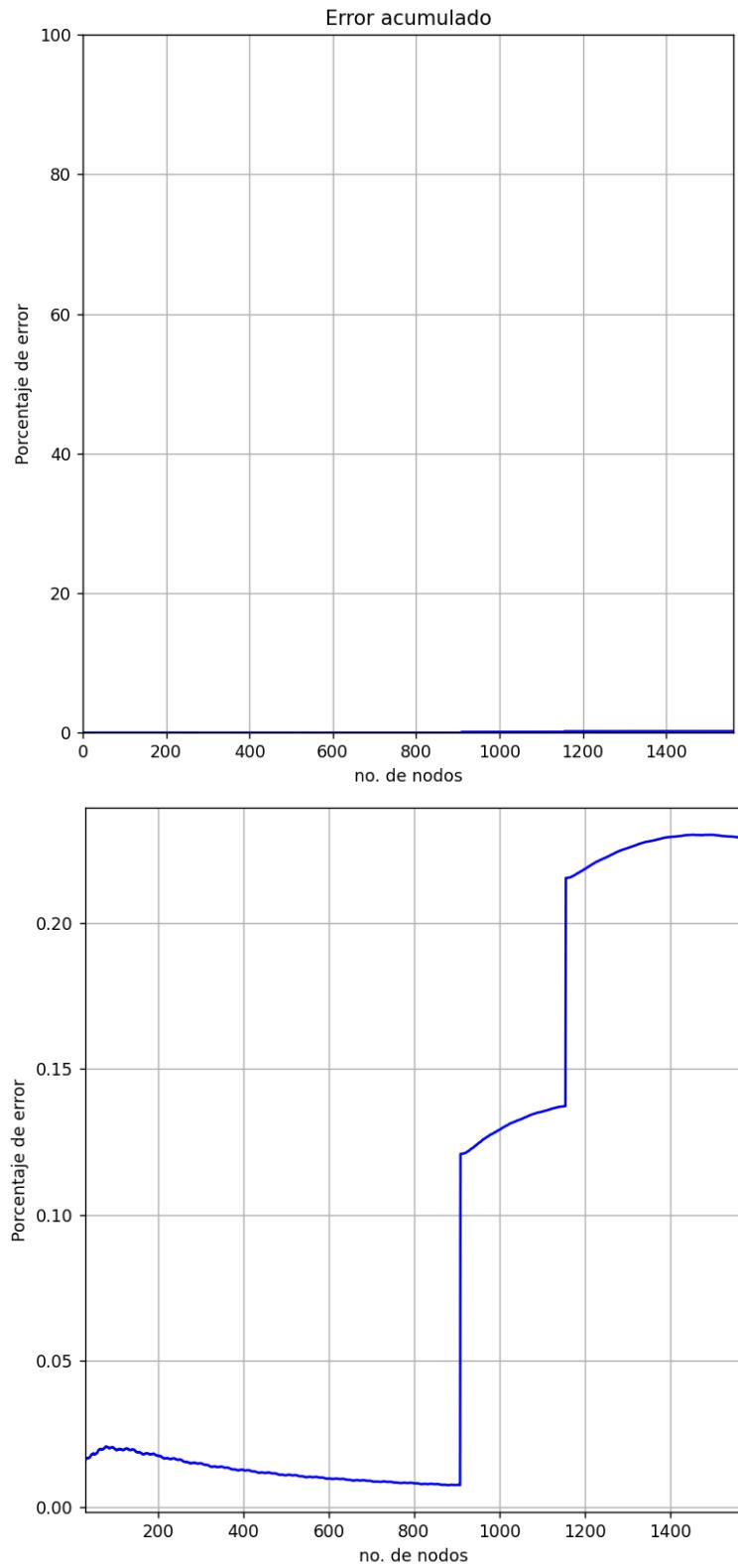


Figura 51: Error acumulado en trayectoria de exploración con cambio de orientación

9.1.3. Resultados usando sensores de distancia tipo sonar

Con base en el algoritmo de exploración, se emplearon sensores de distancia tipo sonar para realizar pruebas en el entorno simulado. Se consideró todo el espacio disponible, como se muestra en la Figura 36. Esto implica que el vehículo tenía la libertad de moverse en cualquier dirección. Los sensores de distancia tipo sonar estaban configurados para realizar mediciones en todas las direcciones posibles, como se muestra en la Figura 27.

Las pruebas se llevaron a cabo con el propósito de validar el comportamiento de exploración del vehículo en condiciones establecidas en el entorno de simulación. Se expuso al vehículo a dos ubicaciones iniciales diferentes, y además, se varió la duración de la simulación. Se realizaron pruebas específicas de exploración con tiempos de simulación de 5 minutos y 60 minutos. Esto con el fin de observar el comportamiento del vehículo en simulaciones de poca duración, 5 minutos, y simulaciones de larga duración, 60 minutos. De la misma manera, se realizaron múltiples pruebas con tiempos variados de simulación y así observar el comportamiento del vehículo, estos resultados se muestran en la sección de anexos. Este enfoque de prueba permitió evaluar la adaptabilidad y la eficacia del algoritmo de exploración en más de un escenario. Al variar las ubicaciones iniciales, se pudo observar si el vehículo era capaz de iniciar la exploración exitosamente desde puntos diferentes. Por otro lado, la variación en el tiempo de simulación permitió evaluar cómo el algoritmo se comportaba en términos de cobertura y eficiencia.

Uno de los aspectos más importantes que se destacó en estos resultados fue la limitación inherente de los sensores de distancia tipo sonar. Como se mencionó previamente, estos sensores presentan lecturas válidas cuando el ángulo de incidencia es menor al 22.5 grados, como se ilustra en la Figura 33. Esto significa que los sensores tienen una zona ciega significativa en su campo de visión. Esta limitación en el ángulo de incidencia llevó a resultados engañosos o incompletos en ciertos escenarios, dando como resultado colisiones del vehículo con el entorno. Esto puede tener importantes implicaciones en términos de seguridad y capacidad de navegación en entornos desconocidos.

Para la primera prueba de este algoritmo, como ya se mencionó, se emplearon sensores de distancia tipo sonar. La posición inicial de la prueba se encuentra claramente identificada en verde claro en la Figura 52. En la Figura 37 se puede observar que esta posición inicial se encuentra ubicada al final del pasillo correspondiente a la sección 1. Esta ubicación es de importancia, ya que planteó un desafío para el vehículo. El vehículo debe determinar la orientación correcta para avanzar hacia la única sección con la que tiene conexión, la cual es la sección 2.

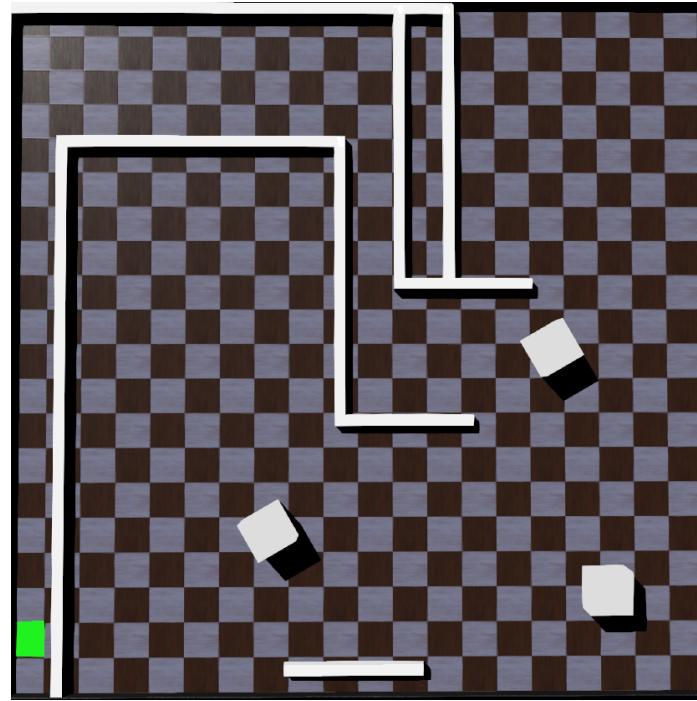
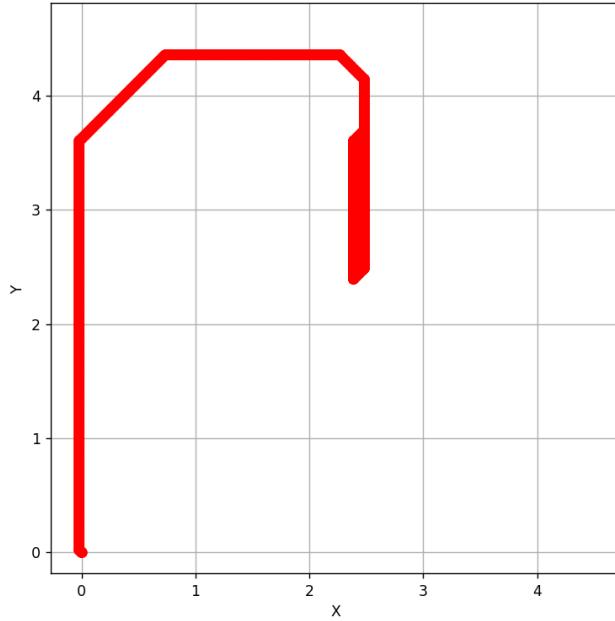


Figura 52: Posición inicial de vehículo en el entorno de simulación.

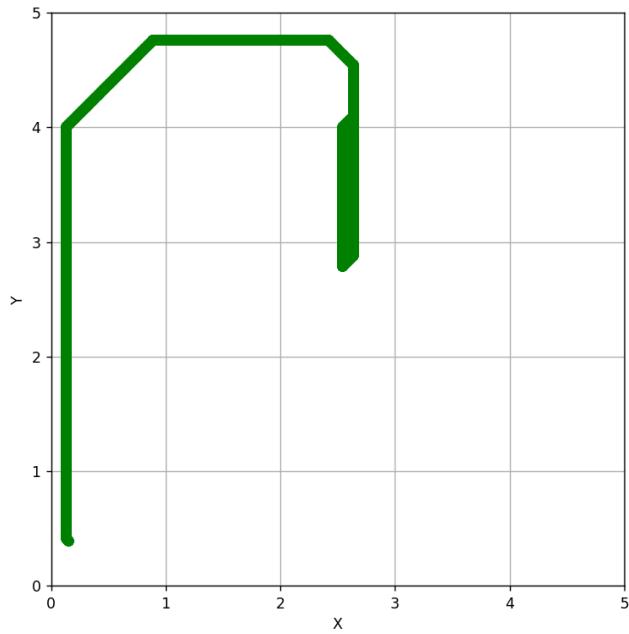
En la simulación de 5 minutos, se obtuvo tanto la trayectoria de exploración estimada como la trayectoria obtenida por GPS, estas se muestra en la Figura 53. En esta prueba, el vehículo exploró tres secciones diferentes.

En primer lugar, se encontraba la Sección 1, que representaba la ubicación inicial del vehículo y correspondía a un pasillo. Posteriormente, el vehículo se adentró en la Sección 2, un entorno más amplio y abierto. En esta sección no mostró trazos de exploración redundante, lo que perfiló al vehículo para explorar otra sección.

Finalmente, el vehículo exploró la Sección 3, otra área que también se asemejaba a un pasillo. Sin embargo, en lugar de avanzar hacia la siguiente sección después de llegar al final de esta, el vehículo optó por regresar a la Sección 2 del mapa en lugar de continuar explorando otras secciones.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 53: Trayectoria de exploración usando sensores de distancia tipo sonar con simulación de 5 min.

Como se puede observar en la Figura 54, en la exploración de 5 minutos de simulación que cubrió tres secciones diferentes del mapa, se registró un error máximo por debajo del 0.175 %. Mantener un porcentaje de error bajo en la estimación de la posición del vehículo proporciona una referencia confiable para llevar a cabo el mapeo del entorno.

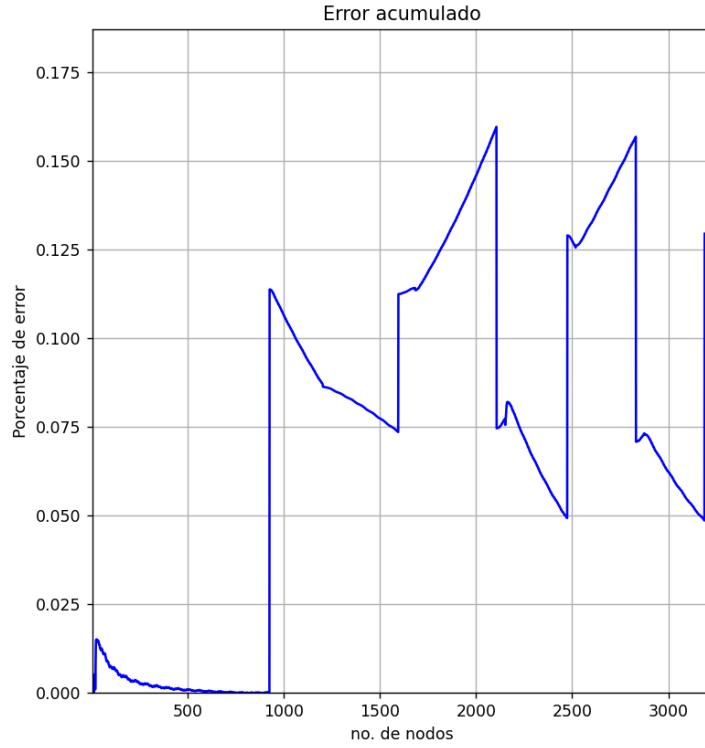
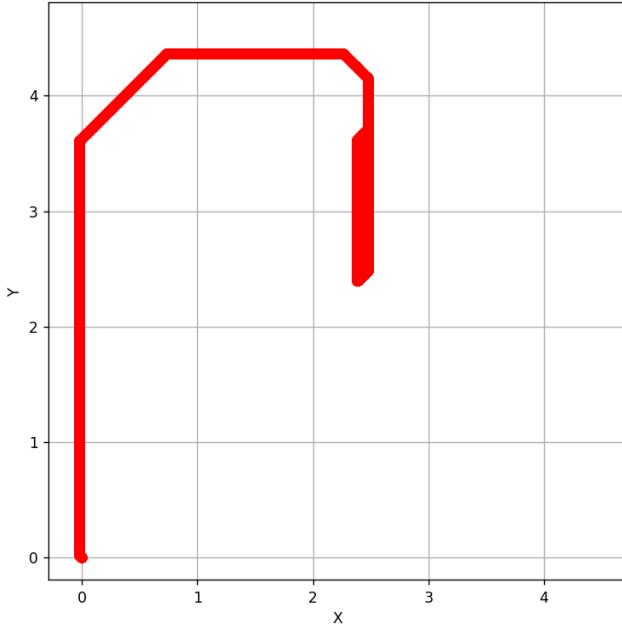


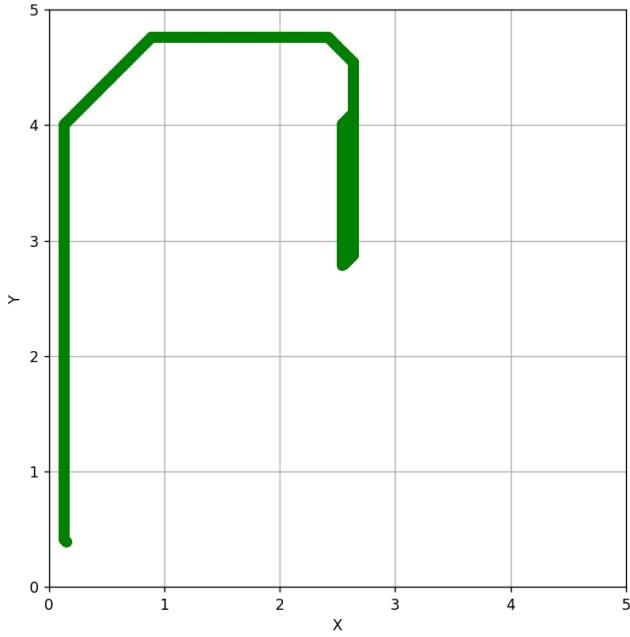
Figura 54: Error acumulado en trayectoria de exploración usando sensor de distancia tipo sonar en simulación de 5 min.

En la simulación que se extendió a lo largo de 60 minutos, se obtuvo tanto la trayectoria de exploración estimada como la trayectoria obtenida por GPS, estas se muestran en la Figura 55. En esta prueba el vehículo exploró las mismas tres secciones del entorno que previamente recorrió en la simulación de 5 minutos.

A pesar de extender el período de simulación a una duración de 60 minutos, el vehículo no logró expandir su exploración hacia áreas o secciones adicionales del entorno. Esto apunta hacia posibles limitaciones en el algoritmo de navegación utilizado o en las capacidades inherentes de los sensores de tipo sonar, lo que podría estar restringiendo la capacidad de exploración del vehículo autónomo.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 55: Trayectoria de exploración usando sensores de distancia tipo sonar con simulación de 60 min.

El error en la estimación de la posición del vehículo, a lo largo de la exploración de 60 minutos, se mantuvo por debajo del 0.25 %, lo cual aún se considera una referencia sólida y precisa para el mapeo del entorno. La forma de la gráfica que se presenta en la Figura 56 ilustra la acumulación de error causada por un bucle en la trayectoria de exploración. Desde los 5 minutos hasta el final de la simulación, la exploración se desarrolló exclusivamente en la sección 3. Es importante destacar que esta sección corresponde a un pasillo no muy largo,

y en este tramo, cada giro se efectuó poco tiempo después del anterior. Además, se observó que los giros realizados en este bucle se ejecutaron en el mismo sentido.

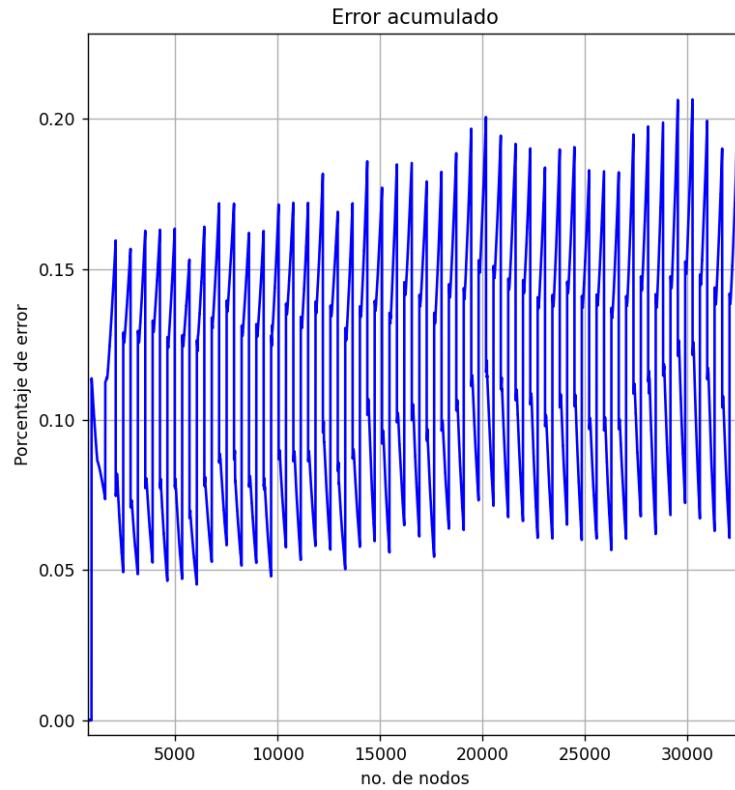


Figura 56: Error acumulado en trayectoria de exploración usando sensor de distancia tipo sonar en simulación de 60 min.

Para la segunda prueba del algoritmo de exploración usando sensores de distancia tipo sonar, la posición inicial se encuentra claramente identificada en verde claro, como se muestra en la Figura 57. En la Figura 37, se puede apreciar que esta posición inicial está ubicada en la sección 6 del entorno.

A diferencia de la ubicación inicial utilizada en la prueba anterior, esta sección representa un entorno más extenso y abierto. Además, se destaca por su acceso a dos secciones adicionales, la 5 y la 7, lo que proporciona una mayor variedad de opciones para llevar a cabo la exploración.

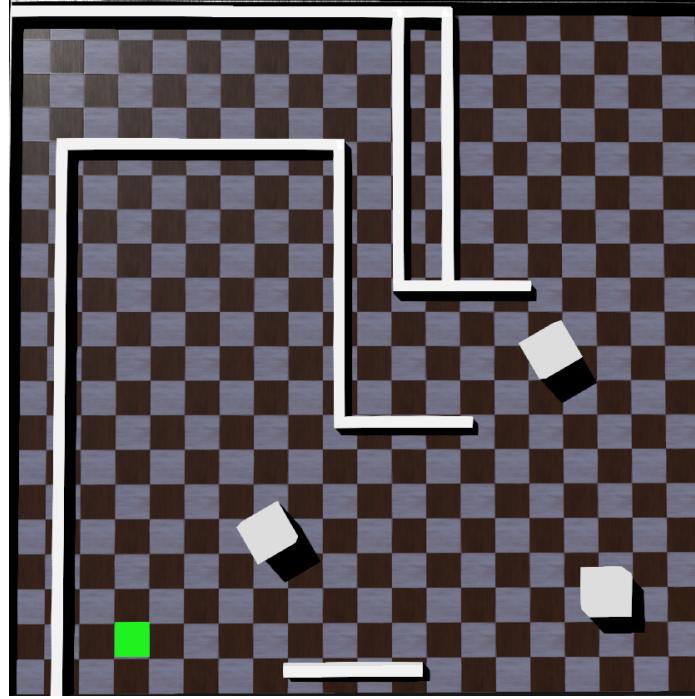
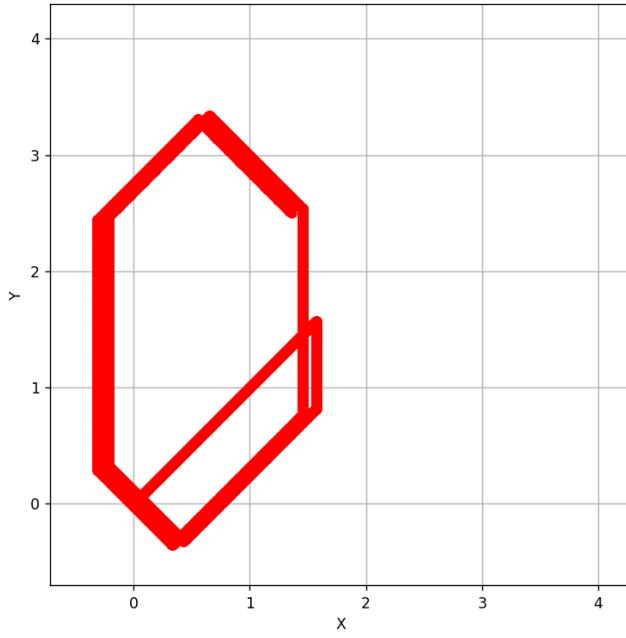


Figura 57: Posición inicial del vehículo en la Sección 6 del entorno de simulación.

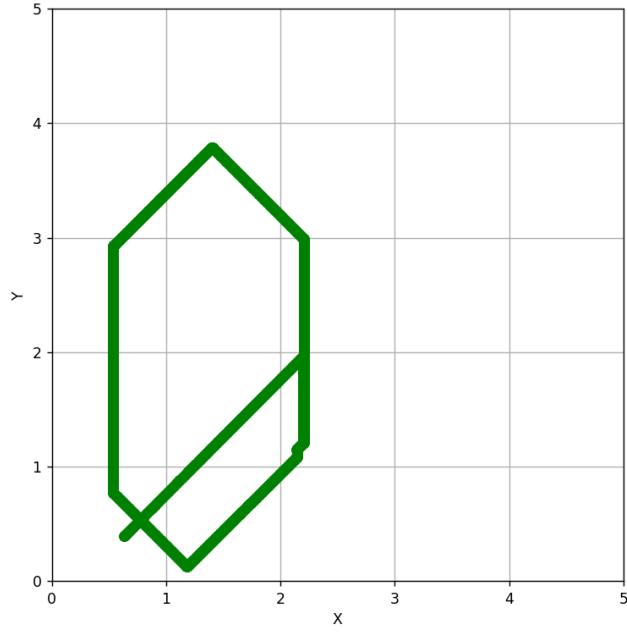
En la simulación de 5 minutos, que se inició desde la sección 6 del entorno, se obtuvo tanto la trayectoria de exploración estimada como la trayectoria obtenida por GPS, ambas representadas en la Figura 58. En esta prueba, el vehículo exploró exclusivamente una sección adicional, la sección 7.

El vehículo se adentró en la sección 7, la cual presenta un entorno igual de amplio y abierto que la sección 6. En esta sección, no se observaron trazos de exploración redundantes, lo que permitió al vehículo perfilarse para explorar nuevamente la sección 6. Una vez de regreso en la sección 6, al igual que durante la exploración de la sección 7, no se registró redundancia en el trazado de la trayectoria. Posteriormente, estando en esta sección, el vehículo volvió a explorar la sección 7, resultando en una exploración nula en las demás secciones.

La Figura 58 muestra claramente que la trayectoria estimada, Figura 58a, no coincide exactamente con la trayectoria obtenida mediante GPS, Figura 58b. A lo largo de esta simulación de 5 minutos, se exploraron las secciones 6 y 7 en tres ocasiones. Sin embargo, la trayectoria estimada presenta desfase en las diferentes ocasiones en las que transitó por el mismo punto, a diferencia de la trayectoria obtenida con GPS, que muestra que en cada una de las exploraciones realizadas en estas dos secciones, se recorrieron los mismos puntos de manera precisa.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 58: Trayectoria de exploración segunda posición usando sensores de distancia tipo sonar con simulación de 5 min.

Como se muestra en la Figura 59, el error acumulado llegó a ser mayor del 20 %, esto indica que la trayectoria estimada no es una referencia sólida para el mapeo del entorno. Este desfase previamente mencionado se origina por una colisión con la esquina de un obstáculo situado en la sección 6. Dicha colisión se aprecia claramente en la Figura 60.

Esta colisión de menor da lugar a un desplazamiento que no se toma en cuenta en la

estimación de la trayectoria. Como consecuencia, el algoritmo asume erróneamente que la colisión nunca ocurrió. Este problema se deriva del hecho de que el algoritmo carece de un mecanismo de freno de emergencia para detectar y evitar colisiones inminentes.

Esto implica que, en situaciones similares, el algoritmo no es capaz de reaccionar de manera adecuada frente a obstáculos imprevistos, lo que compromete su capacidad para proporcionar una estimación precisa de la trayectoria del vehículo y, por ende, su utilidad en el proceso de mapeo del entorno. La colisión y su impacto en la precisión de la trayectoria estimada son cuestiones que requieren una revisión y una posible mejora en el diseño y funcionamiento del algoritmo.

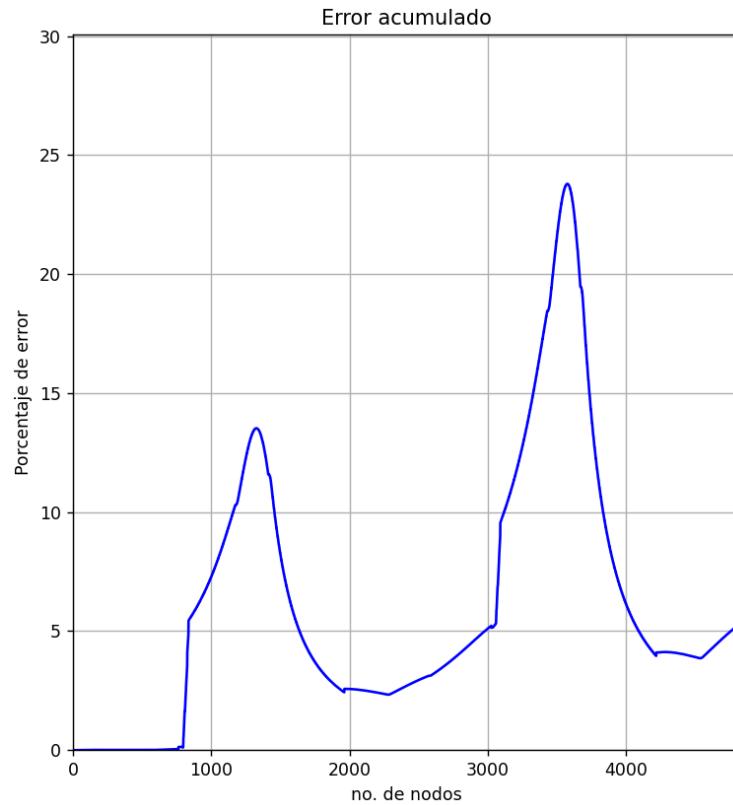


Figura 59: Error acumulado en trayectoria de exploración iniciando en sección 5 del entorno simulado, usando sensor de distancia tipo sonar en simulación de 5 min.

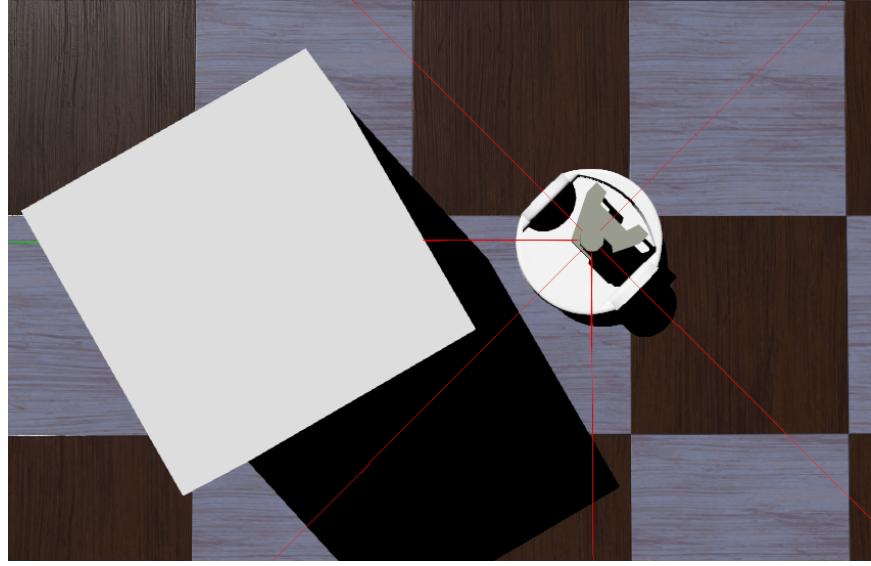


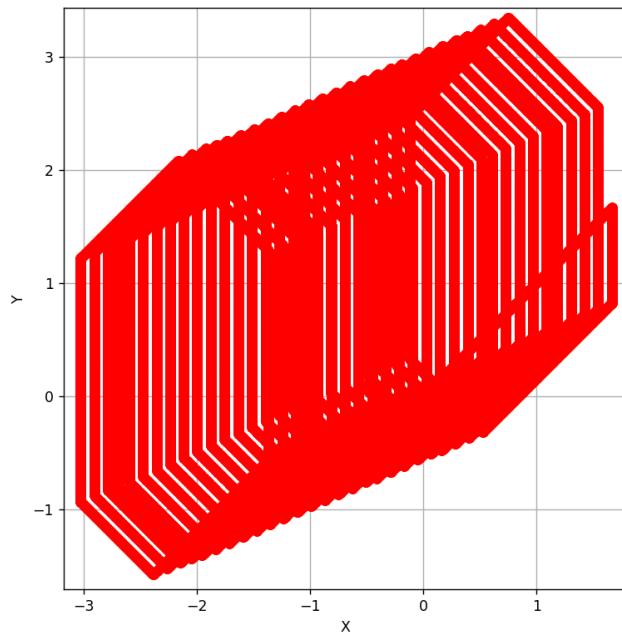
Figura 60: Colisión de vehículo con obstáculo.

En la simulación que se extendió a lo largo de 60 minutos, partiendo desde la sección 6 del entorno de simulación, se obtuvieron tanto la trayectoria de exploración estimada como la trayectoria obtenida por GPS, las cuales se presentan en detalle en la Figura 61. En esta prueba, se pueden identificar desfases significativos en la trayectoria estimada, lo que conlleva a una representación poco clara y precisa de la exploración del vehículo autónomo.

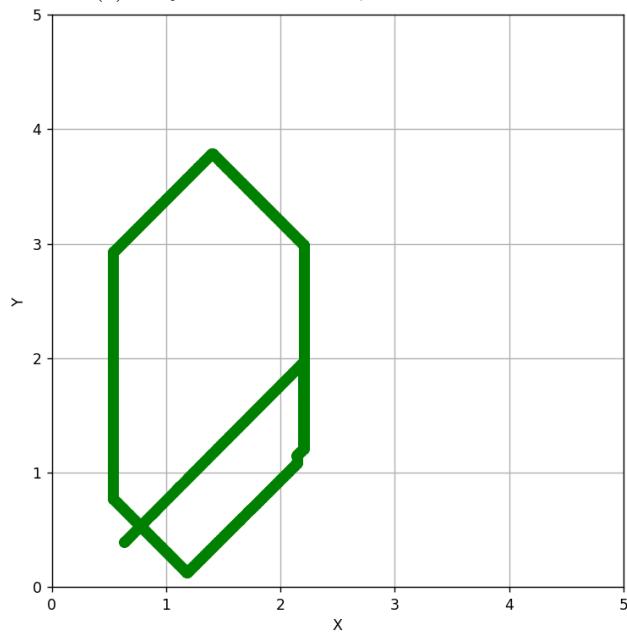
A pesar de la extensión del período de simulación a 60 minutos, el vehículo no logró explorar otras secciones del mapa, lo que indica una limitación en su capacidad de exploración. Sin embargo, lo más destacado en esta prueba es la propagación continua del desfase, ocasionado por las colisiones recurrentes con la esquina del obstáculo en la sección 6.

Este desfase tiene un impacto significativo en la precisión de la trayectoria estimada, lo que a su vez afecta la base de la calidad del mapeo del entorno. La colisión recurrente, que no es detectada ni corregida adecuadamente por el algoritmo debido a la ausencia de un mecanismo de freno de emergencia, contribuye en gran medida a la acumulación de error en la estimación de la trayectoria.

Como resultado, el error en la estimación de la trayectoria supera el 160 %, ver Figura 62, lo cual confirma de manera contundente que esta trayectoria estimada no es una referencia adecuada ni confiable para el mapeo del entorno. Esta limitación en la precisión y la capacidad del algoritmo para explorar usando sensores de distancia de tipo sonar tiene consecuencias directas en el mapeo del entorno.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 61: Trayectoria de exploración segunda posición usando sensores de distancia tipo sonar con simulación de 60 min.

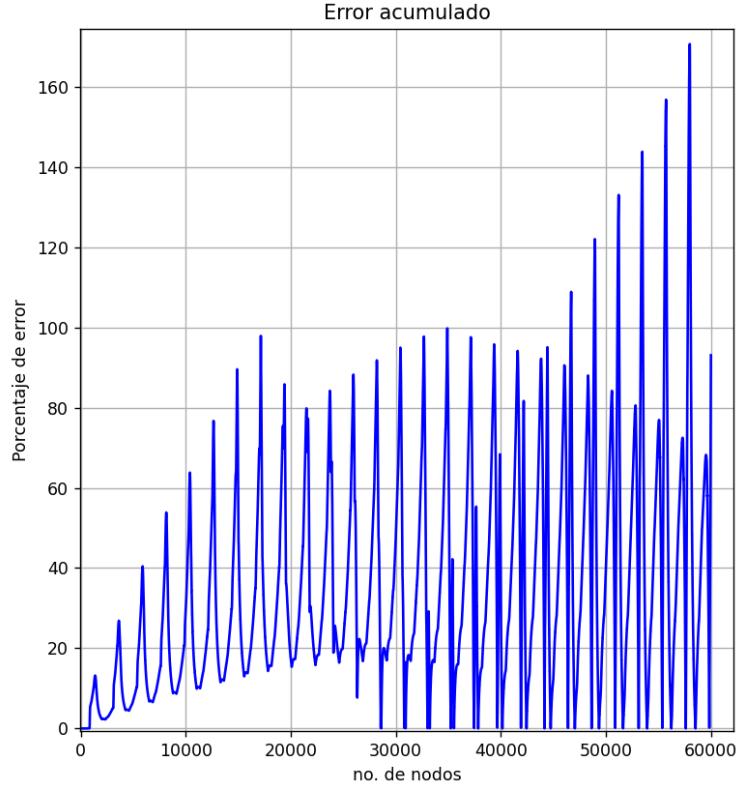


Figura 62: Error acumulado en trayectoria de exploración iniciando en sección 5 del entorno simulado, usando sensor de distancia tipo sonar en simulación de 60 min.

9.1.4. Resultados usando sensores de distancia tipo láser

Con base en el algoritmo de exploración, se emplearon sensores de distancia tipo sonar para realizar pruebas en el entorno simulado. Se consideró todo el espacio disponible, como se muestra en la Figura 36. Esto implica que el vehículo tenía la libertad de moverse en cualquier dirección. Los sensores de distancia tipo láser fueron configurados para realizar mediciones en las direcciones que se muestran en la Figura 27.

Las pruebas se llevaron a cabo con el propósito de validar el comportamiento de exploración del vehículo en condiciones establecidas en el entorno de simulación utilizando sensores de distancia tipo láser. Se expuso al vehículo a dos ubicaciones iniciales diferentes, y además, se varió la duración de la simulación. Se realizaron pruebas específicas de exploración con tiempos de simulación de 5 minutos y 60 minutos. Esto con el fin de observar el comportamiento del vehículo en simulaciones de poca duración, 5 minutos, y simulaciones de larga duración, 60 minutos. De la misma manera, se realizaron múltiples pruebas con tiempos variados de simulación y así observar el comportamiento del vehículo, estos resultados se muestran en la sección de anexos. Este enfoque de prueba permitió evaluar la adaptabilidad y la eficacia del algoritmo de exploración en más de un escenario. Al variar las ubicaciones iniciales, se pudo observar si el vehículo era capaz de iniciar la exploración exitosamente desde puntos diferentes. Por otro lado, la variación en el tiempo de simulación permitió evaluar cómo el algoritmo se comportaba en términos de cobertura y eficiencia.

Uno de los aspectos más importantes que resaltaron en estos resultados fue la versatilidad de los sensores de distancia tipo láser. A diferencia de los sensores sonar, los sensores láser no poseen una zona ciega significativa en su campo de visión. Esto facilitó la obtención de resultados más precisos, proporcionando valores de utilidad para el algoritmo de exploración. Esto se tradujo en una mayor cobertura del mapa, explorando secciones que no fueron alcanzadas utilizando sensores de distancia tipo sonar.

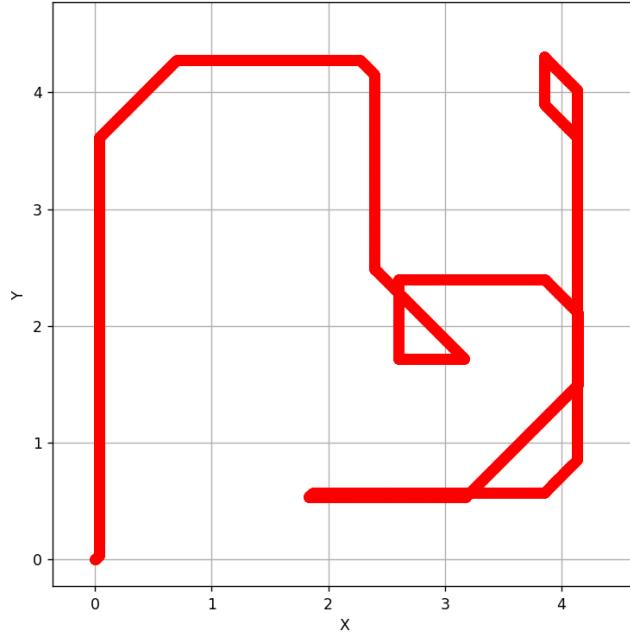
Para la primera prueba de este algoritmo, como ya se mencionó, se emplearon sensores de distancia tipo láser. La posición inicial de la prueba se encuentra claramente identificada en verde claro en la Figura 52.

En la simulación de 5 minutos, se obtuvieron tanto la trayectoria de exploración estimada como la trayectoria obtenida por GPS, las cuales se presentan en la Figura 63. Durante esta prueba, el vehículo exploró ocho de las nueve secciones disponibles.

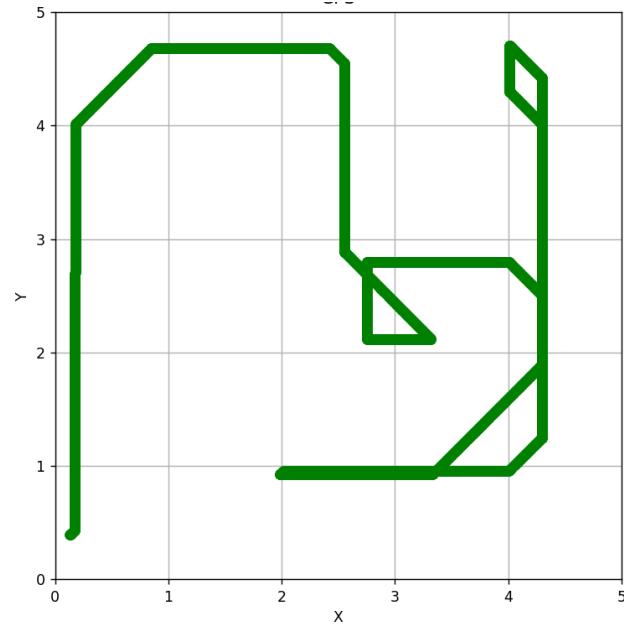
En primer lugar, se encontraba la Sección 1, que representaba la ubicación inicial del vehículo y correspondía a un pasillo. Posteriormente, el vehículo ingresó a la sección 2 del mapa, este es un entorno más amplio y abierto. En esta sección, no se detectaron trazos de exploración redundante, lo que condujo al vehículo a explorar la sección 3. En la sección 3 del mapa, el vehículo avanzó sin problemas hacia la sección 4, a diferencia de las pruebas con sensores de distancia de tipo sonar, en las cuales, en lugar de avanzar hacia la siguiente sección después de llegar al final de la misma, el vehículo optó por regresar a la sección 2 del mapa.

En la Figura 63, se puede observar que en el espacio correspondiente a la sección 4 del mapa, se produjo un giro que podría considerarse “redundante”. Después de este movimiento, el vehículo exploró la sección 8 del mapa, eludiendo un obstáculo presente en dicha sección. Esta sección está conectada con otras dos secciones, la sección 5 y la sección 9.

Posteriormente, el vehículo exploró la sección 5 del mapa, también eludiendo un obstáculo presente en esta área. La sección 5 es amplia y abierta, lo que permitía el tránsito en diversos puntos de la misma. La trayectoria de exploración en esta sección fue directa, sin presentar redundancias, lo que facilitó su cobertura sin problemas. Luego, el vehículo comenzó a explorar la sección 6, sin embargo, esta no fue completamente cubierta. Cuando el vehículo detectó el obstáculo presente en esta sección, se dirigió de regreso a la sección 5 del mapa. De igual manera, la exploración de esta sección del mapa se llevó a cabo sin dificultades. La sección 8 del mapa fue atravesada rápidamente. Esto condujo al vehículo a explorar la sección 9 del mapa. Como se puede apreciar en la Figura 63, en la sección 9 del mapa, el vehículo realizó un recorrido en forma de equilátero; sin embargo, al salir de esta sección, lo hizo por la misma trayectoria de entrada, sin variaciones en el recorrido de salida.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 63: Trayectoria de exploración usando sensores de distancia tipo láser con simulación de 5 min.

Al observar la Figura 63, se puede apreciar que la trayectoria estimada no muestra ningún desfase y presenta una notable semejanza con la trayectoria obtenida con GPS. Esta semejanza se respalda por el poco porcentaje de error que se puede constatar en la Figura 64, donde el error en la estimación de la trayectoria de exploración se registró por debajo del 1 %. Este nivel de precisión hace que la trayectoria estimada sea una referencia adecuada

para el proceso de mapeo del entorno. Es importante destacar que este nivel de precisión se logró al explorar 8 de las 9 secciones disponibles, lo que significa que usando el mismo algoritmo de exploración con sensores de distancia tipo láser se cubrió la mayoría del entorno sin comprometer la exactitud en la estimación de la posición del vehículo.

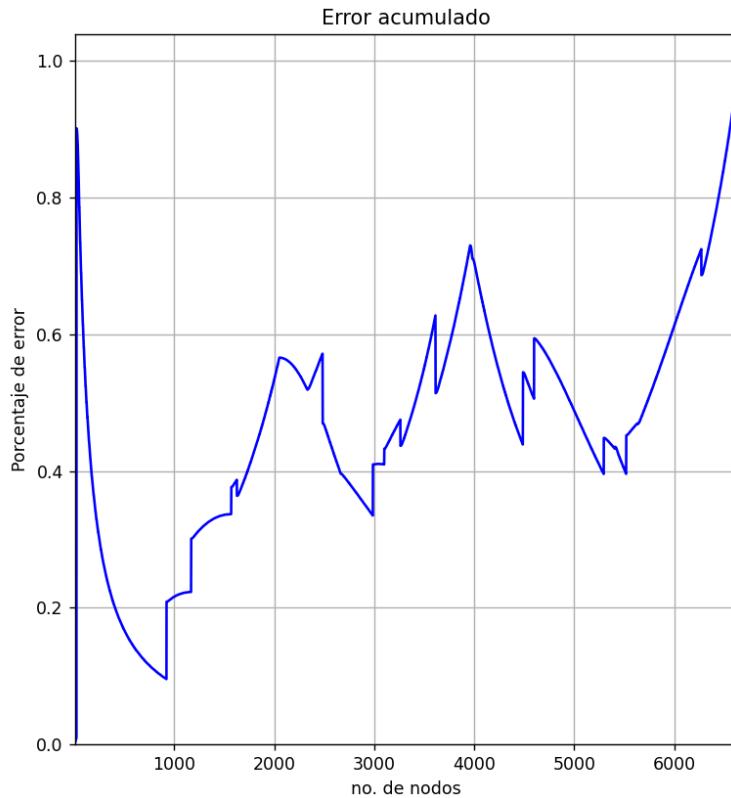
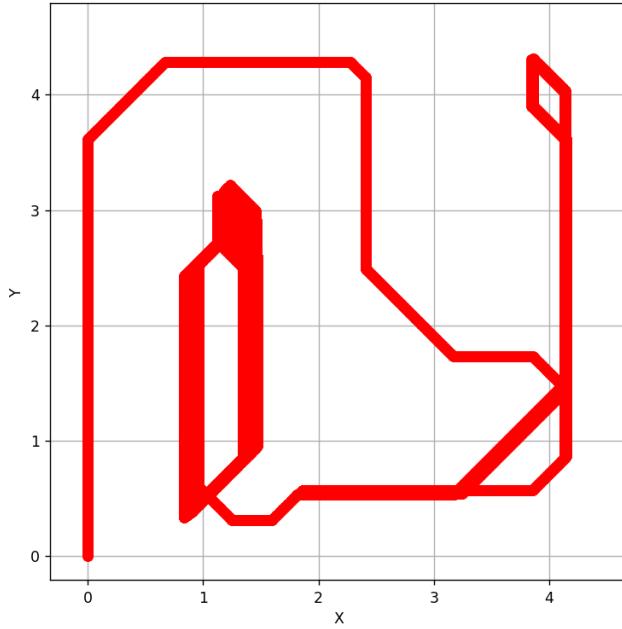


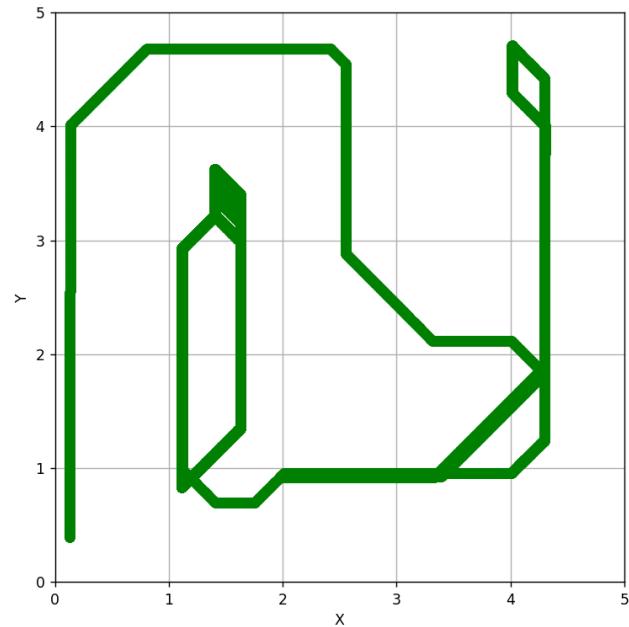
Figura 64: Error acumulado en trayectoria de exploración usando sensor de distancia tipo sonar en simulación de 5 min.

En la simulación que se extendió a lo largo de 60 minutos, se obtuvieron tanto la trayectoria de exploración estimada como la trayectoria obtenida por GPS, y ambas fueron representadas en la Figura 65. Durante esta prueba, el vehículo llevó a cabo la exploración de todas las secciones del mapa simulado.

Sin embargo, al observar la Figura 65a, se pudo notar que la trayectoria estimada de la exploración en las secciones 7 y 6 del mapa presentaba desfases. Durante los primeros 5 minutos de la prueba, el vehículo exploró exitosamente 8 de las 9 secciones, pero a partir de ese punto y hasta los 60 minutos totales, el vehículo se mantuvo explorando únicamente las secciones 6 y 7. A pesar de que la estimación de la trayectoria presentaba desfases, estos eran de menor magnitud en comparación con la estimación al usar sensores de distancia tipo sonar. En esta repetición de la trayectoria de exploración, no se registraron colisiones con el obstáculo de la sección 6 del mapa, sin embargo, sí se presentaron desfases en la trayectoria estimada.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 65: Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo láser con simulación de 60 min.

Basándose en la trayectoria de exploración realizada, se puede observar en la Figura 66 se obtiene un error de estimación en la trayectoria de exploración que es menor al 14 %. Sin embargo, es importante destacar que este error comienza a aumentar considerablemente a partir de los 10,000 nodos explorados. En la Figura 64, se presenta el error de estimación de la trayectoria de exploración en la simulación de 5 minutos. En esta exploración, se logró cubrir 8 de las 9 secciones del mapa, dejando sin explorar únicamente la sección 7. Con base

en los resultados anteriores, se puede deducir que el incremento en el error de estimación de la trayectoria en la simulación de 60 minutos coincide con la repetitiva exploración de las secciones 6 y 7 del mapa.

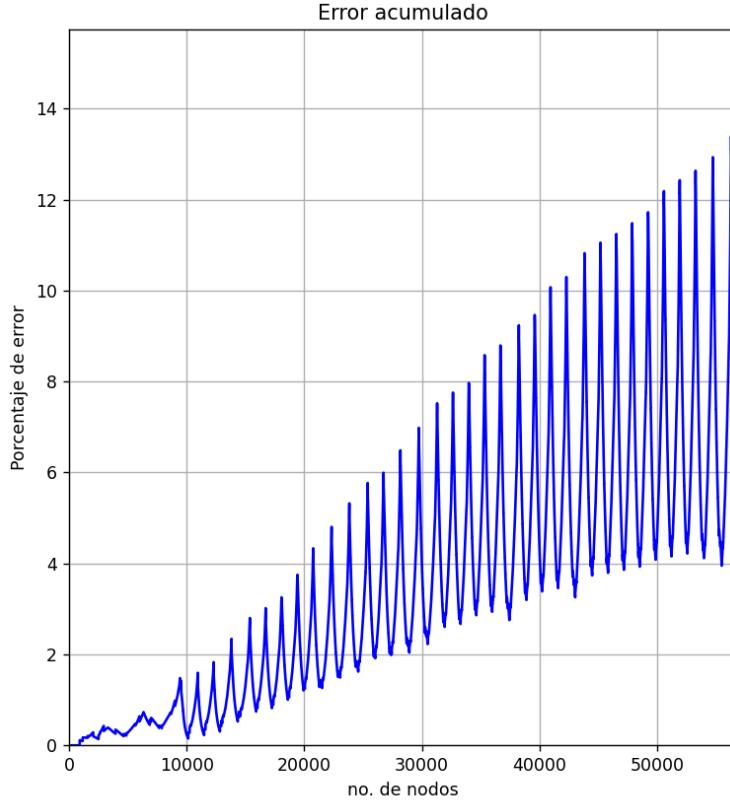


Figura 66: Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo láser en simulación de 60 min.

Para la segunda prueba del algoritmo de exploración usando sensores de distancia tipo láser, la posición inicial se encuentra claramente identificada en verde claro, como se muestra en la Figura 57. En la Figura 37, se puede apreciar que esta posición inicial está ubicada en la sección 6 del entorno.

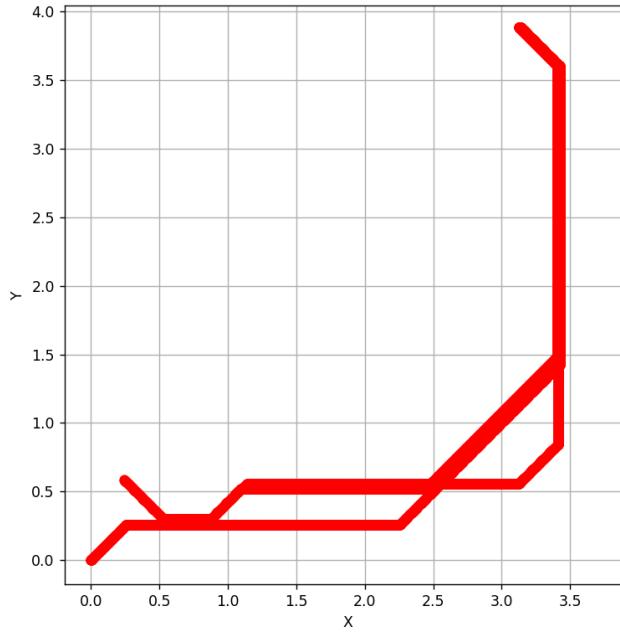
A diferencia de la ubicación inicial utilizada en la prueba anterior, esta sección del entorno representa un espacio más extenso y abierto. Además, se destaca por su acceso a dos secciones, la 5 y la 7, lo que proporciona una mayor variedad de opciones para llevar a cabo la exploración.

En la Figura 67, se presenta la trayectoria estimada de la exploración junto con la trayectoria obtenida a través del sistema de GPS. Durante esta prueba simulada de 5 minutos, se puede observar que la trayectoria de exploración abarcó las secciones 6, 5, 8 y 9 del entorno.

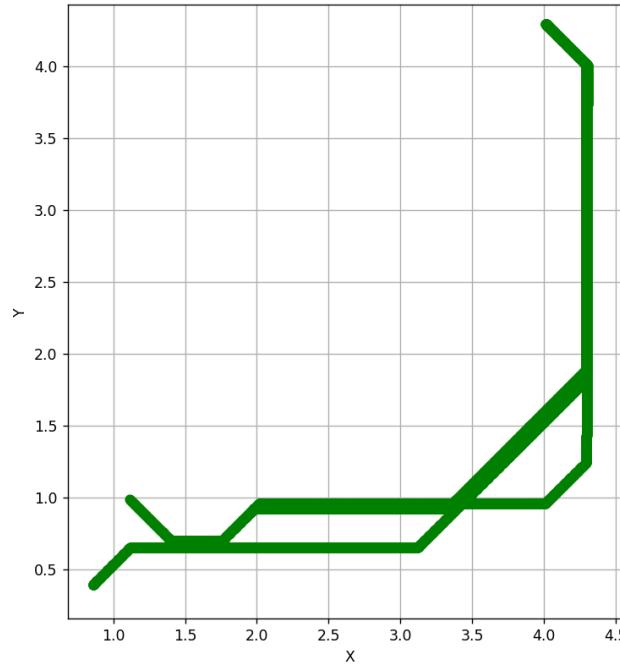
Inicialmente, la exploración se centró en la sección 6 del entorno, evitando el obstáculo que se encontraba en su camino. Después, el vehículo se desplazó hacia la sección 5, de igual forma, se logró evitar el obstáculo presente en esta área. Posteriormente, se produjo un desplazamiento rápido hacia la sección 8, sin encontrar obstáculos en su camino. Finalmente,

la exploración se dirigió a la sección 9 de manera fluida, sin ningún problema notable.

Después de completar esta secuencia de exploración, el vehículo regresó a la sección 6 del entorno, siguiendo el mismo orden en el que había llegado inicialmente.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 67: Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 5 min.

Como se puede observar en la Figura 68, el error en la estimación de la trayectoria de la

exploración se mantuvo por debajo del 1.4 %. Este nivel de precisión hace que la trayectoria estimada sea una referencia adecuada para el proceso de mapeo del entorno.

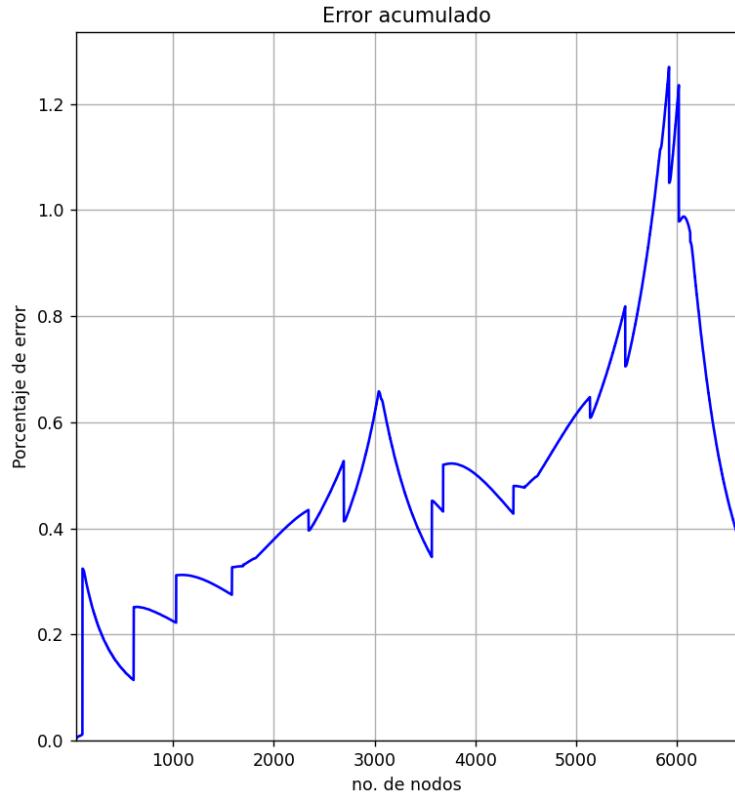
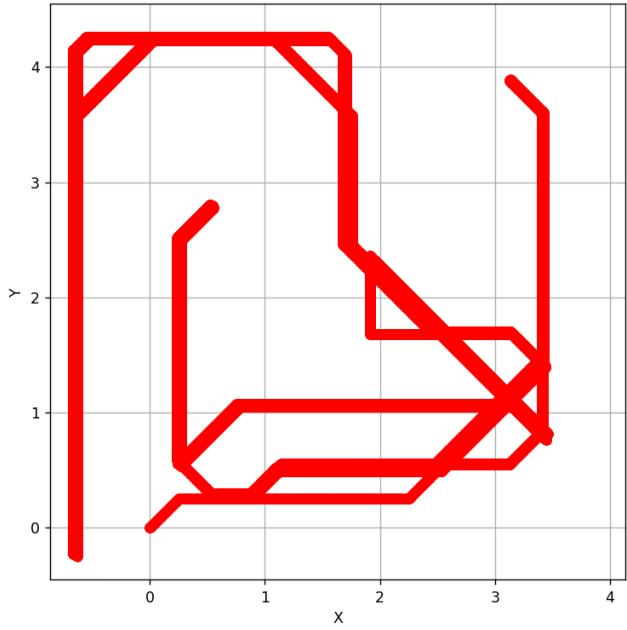


Figura 68: Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 5 min.

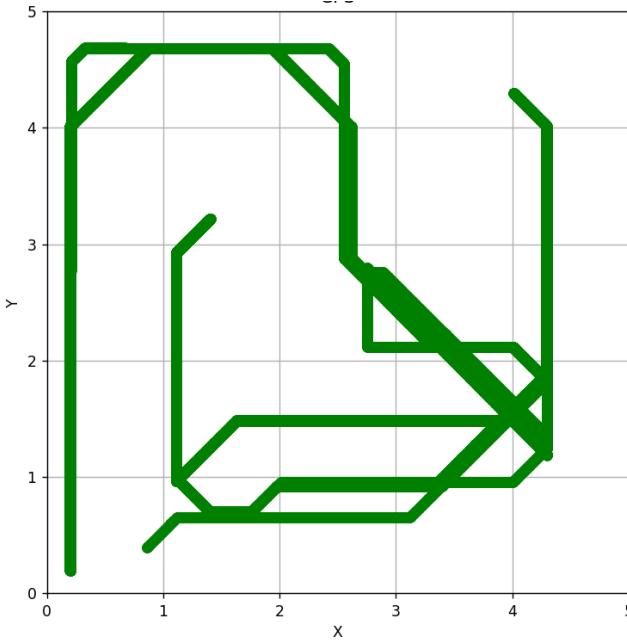
En la prueba del algoritmo de exploración, que se inició desde la sección 6 del mapa y tuvo una duración de 60 minutos, se obtuvo la trayectoria que se puede observar la Figura 69. Se muestra la trayectoria estimada como la obtenida con GPS.

Durante esta simulación, el vehículo logró explorar el entorno en su totalidad, abarcando las 9 secciones del mapa. La secuencia de la trayectoria seguida por el vehículo en la exploración comenzó con la exploración de la sección 6 del entorno. Luego se desplazó hacia la sección 5, evitando un obstáculo en su camino. Posteriormente, se movilizó hacia la sección 8, la cual fue explorada sin dificultades. Luego, avanzó hacia la sección 9, la cual también se exploró sin contratiempos, y el punto de entrada en esta sección coincidió con el punto de salida. Luego, regresó a la sección 8 y desde allí se dirigió a la sección 5. En esta sección, se desplazó cerca del borde con la sección anterior para conectar con la trayectoria hacia la sección 4. La sección 4, siendo un espacio pequeño, se exploró sin complicaciones y condujo a la exploración de la sección 3, la cual es un pasillo. Tras completar la exploración de la sección 3, el vehículo prosiguió con la exploración de la sección 2 y luego la sección 1. La sección 1 se exploró en su totalidad y luego se regresó a la sección 2. La trayectoria seguida en la sección 2 difirió de la realizada previamente antes de ingresar a la sección 1, en esta ocasión, la trayectoria se mantuvo más cerca de las paredes. Posteriormente, el vehículo transitó por las secciones 3, 4, 5 y 6 antes de alcanzar la sección 7. A partir de esta exploración, se repitió

el proceso de exploración cinco veces adicionales, manteniendo el mismo orden previamente descrito. Sin embargo, en la última secuencia de exploración, el vehículo quedó atrapado en un ciclo de repetición entre las secciones 4 y 5.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 69: Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 60 min.

Es importante resaltar que en esta prueba, la trayectoria estimada no muestra desfases notables en su mayor parte. Visualmente, la trayectoria estimada guarda una similitud con-

siderable con la trayectoria obtenida mediante el GPS. Sin embargo, en la ultima parte de la simulación, durante el desplazamiento y cambio de sección entre la 4 y la 5, se observa una leve desviación en la trayectoria. Al analizar el error en la estimación de la trayectoria en la Figura 70, se aprecian picos en la trayectoria, los cuales se sitúan por debajo del 7%. En la parte final del gráfico, a partir de los 50,000 nodos, se nota una tendencia al aumento del error. Esta sección del gráfico corresponde al ciclo repetitivo que ocurre en el cambio de sección entre la 4 y la 5.

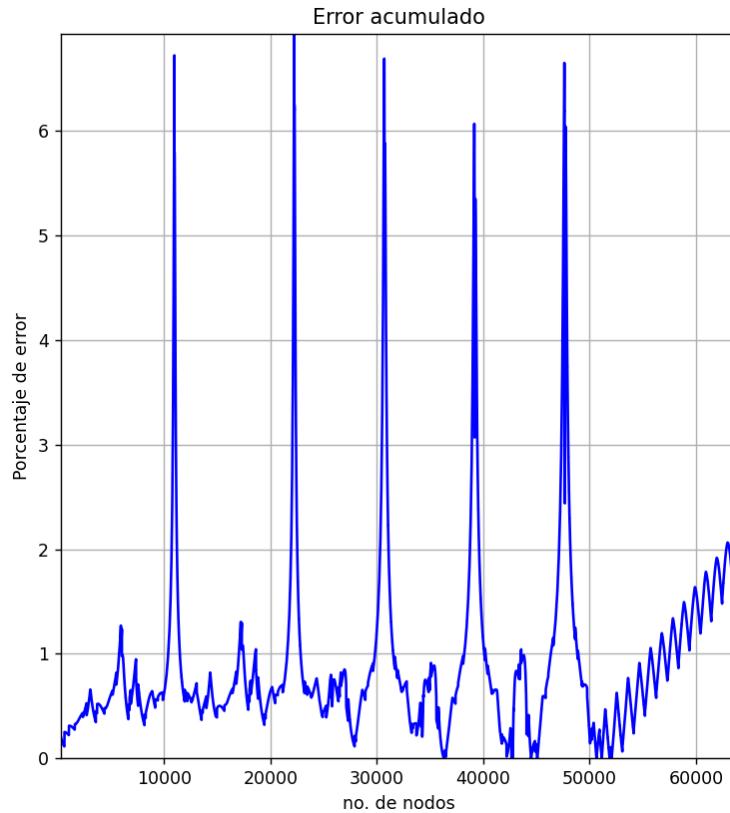


Figura 70: Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 60 min.

9.2. Algoritmo de exploración - Prototipo 2

Con base en los resultados obtenidos en las pruebas realizadas al algoritmo de exploración de entornos, se desarrolló un segundo prototipo. En esta segunda iteración del algoritmo de exploración, se destaca especialmente la mejora en la evasión de obstáculos y la mejora en la selección de ángulos de giro para la exploración de las diferentes secciones del mapa.

Se ha agregado una atención especial a la elección de los ángulos de giro del robot durante su exploración. El algoritmo ahora analiza de manera más detallada el entorno circundante y selecciona los ángulos de giro que maximizan la cobertura del área a explorar. Esto asegura que el vehículo pueda recopilar datos de manera más completa y eficiente, lo que es esencial en aplicaciones de exploración y mapeo.

Otra mejora importante es la incorporación de condiciones para la activación del frenado de emergencia. El algoritmo monitorea constantemente las condiciones del entorno y puede tomar decisiones de frenado de emergencia cuando se detectan situaciones que lo requieran.

Los cambios implementados han contribuido a la mejora de la eficiencia del algoritmo al reducir los tiempos de detención y orientación del vehículo, lo que se traduce en una exploración óptima para el mapeo de entornos. A pesar de estas modificaciones, el segundo prototipo mantiene el método del posicionamiento del vehículo usado en el primer prototipo de este algoritmo de exploración. En la Figura 71 se puede observar el proceso que sigue este segundo prototipo para la exploración de los entornos.

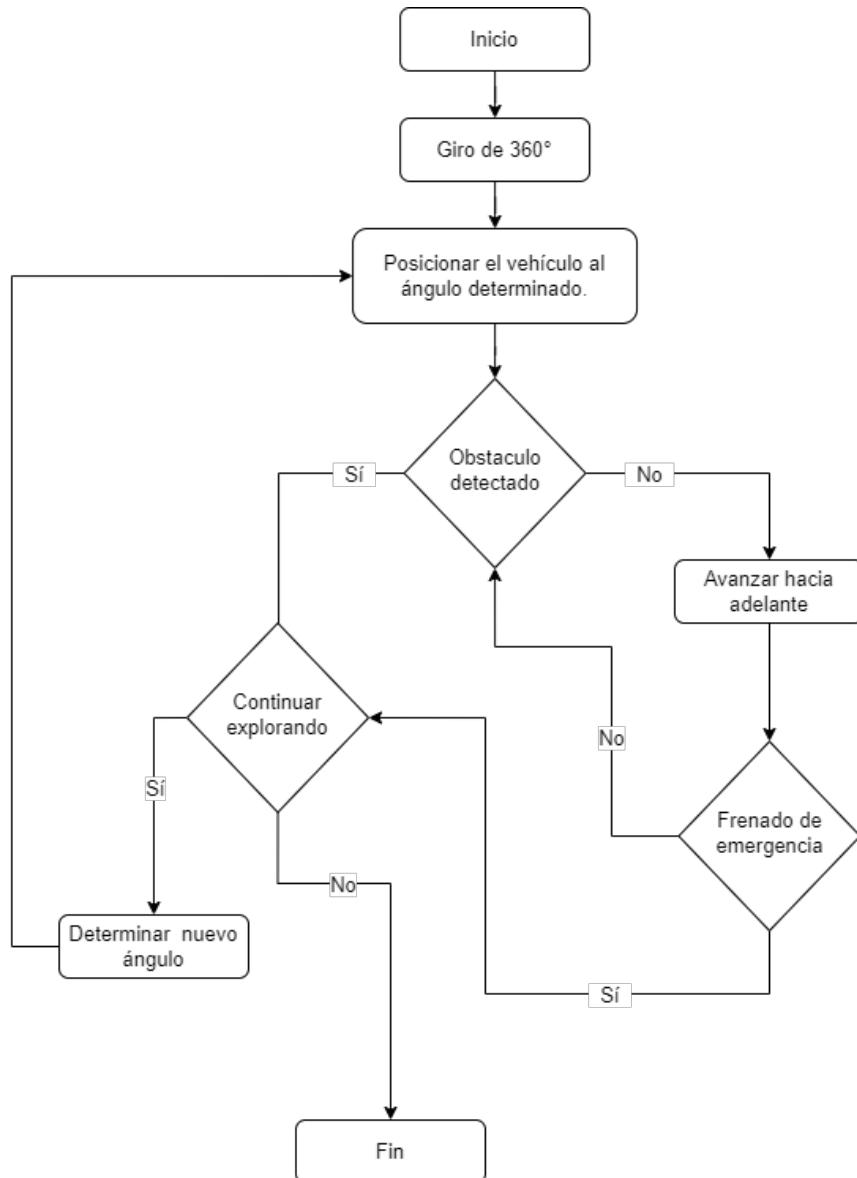


Figura 71: Diagrama de flujo general de algoritmo de exploración prototipo 2.

9.2.1. Condiciones de giro

Al igual que en el prototipo 1 de exploración, cuando el vehículo se encuentra en movimiento, constantemente evalúa su entorno a través de la distribución de sensores de distancia, como se ilustra en la Figura 27.

En la Figura 72 se observa el bucle en el que se encuentra el vehículo cuando esta realizando un desplazamiento. El vehículo avanza hacia adelante, y esta constantemente obteniendo los datos de los sensores ds_0, ds_1, ds_2, ds_4, ds_5. También se determina una distancia mínima para detener el movimiento de exploración y se contempla una distancia de frenado de emergencia.

El valor de la distancia mínima se compara con el valor del sensor ds_0 para determinar si se detiene el proceso de exploración. Este valor de distancia mínima puede ser uno de dos posibles valores, dependiendo de las mediciones de los sensores ds_2 y ds_4. Si ambos sensores, ds_2 y ds_4, registran distancias menores a 100 cm, el valor de distancia mínima es de 30 cm. Por otro lado, si ds_2 y ds_4 registran distancias iguales o mayores a 100 cm, el valor de distancia mínima es de 100 cm.

Esta comparación de las lecturas de los sensores ds_2 y ds_4 determina si el vehículo se encuentra en un entorno de exploración amplio y abierto, o si está operando en un espacio más restringido, como un pasillo estrecho. La elección de la distancia mínima se adapta con base en el entorno, lo que permite una respuesta adecuada del vehículo, asegurando una exploración en función de las condiciones específicas en las que se encuentra.

Esto da paso al proceso de evaluación de los valores de los sensores, el cual consiste en comparar los sensores laterales, ds_2 y ds_4, con el valor del sensor frontal, ds_0. La interpretación de esta comparación implica identificar un espacio lateral que presente mejores oportunidades de exploración en lugar de continuar avanzando en línea recta hacia adelante.

El valor de distancia de frenado de emergencia se compara con los datos de los sensores ds_0, ds_1 y ds_5 para evaluar la necesidad de aplicar un frenado de emergencia y calcular un nuevo ángulo. En las pruebas realizadas, el valor predefinido para la distancia de frenado de emergencia es de 10 cm, lo que representa medio centímetro más que el diámetro del vehículo. Esta comparación es de importancia, como se puede observar en la Figura 71, ya que la activación del frenado de emergencia puede llevar a la determinación de un nuevo ángulo de orientación y evitar colisiones con los obstáculos.

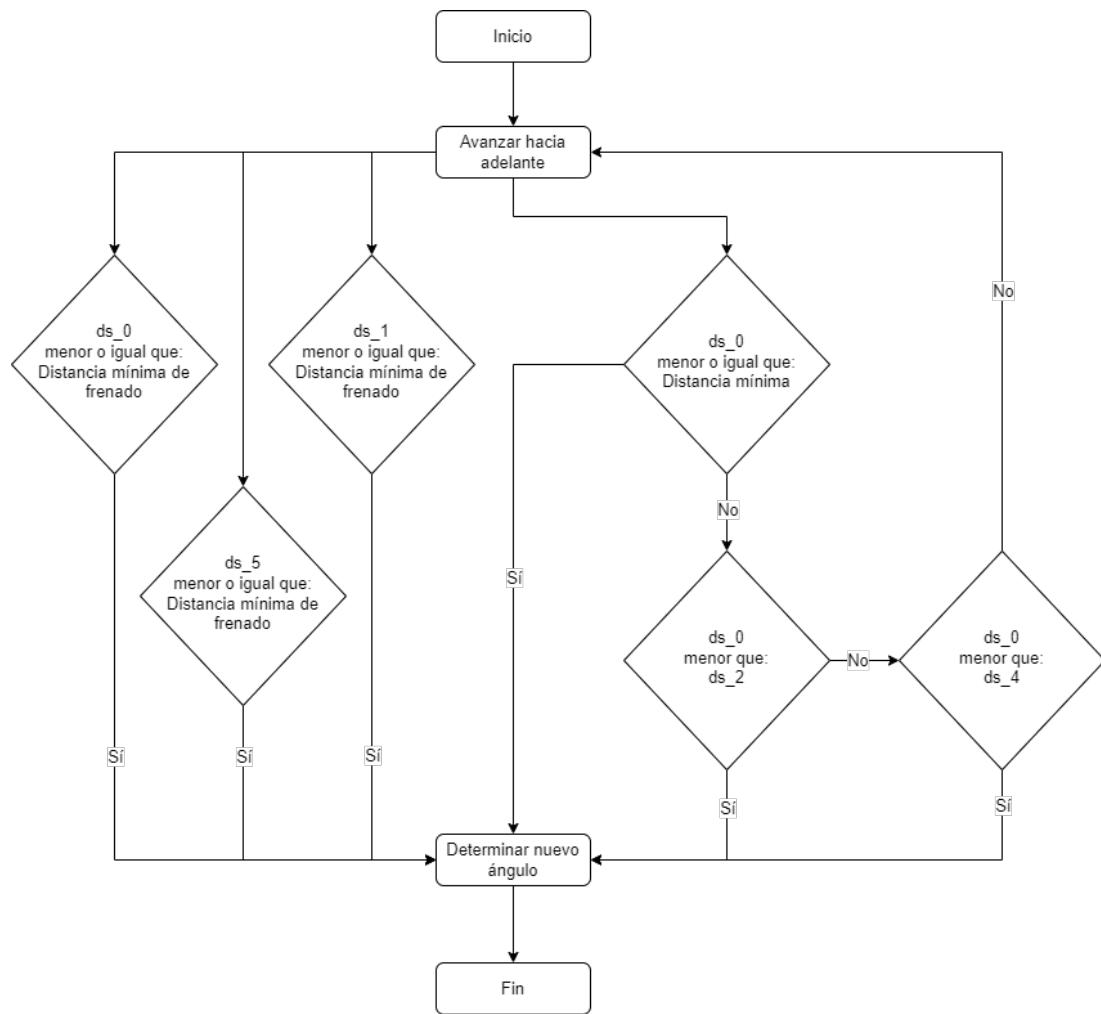


Figura 72: Diagrama de flujo para considerar un cambio de giro en algoritmo de exploración prototipo 2.

CAPÍTULO 10

Planificación de Trayectoria

En este capítulo, se establecen los parámetros que se emplean en la planificación de trayectorias. El algoritmo encargado de generar estas trayectorias se apoya en un espacio de trabajo como su principal punto de referencia. En este espacio de trabajo, es esencial representar y visualizar de manera precisa las características del entorno en el que se desea llevar a cabo la navegación.

Durante el desarrollo de este algoritmo se realizaron pruebas con dos tipos de mapa: un mapa preexistente que ha sido previamente generado aleatoriamente y el mapa estimado que se va construyendo a medida que el sistema explora su entorno.

En primer lugar, el mapa preexistente proporciona una base sólida para la planificación de trayectorias, ya que contiene información detallada sobre las características permanentes y totales de un entorno, como paredes y obstáculos fijos. Por otro lado, el mapa estimado se crea a medida que el sistema se mueve y explora su entorno en tiempo real. Este proceso implica recopilar datos a través de sensores y, en función de la información capturada, estimar la presencia de estos obstáculos en el entorno. La combinación de los resultados de estos dos mapas, el preexistente y el estimado, permitió el desarrollo y validación del algoritmo de planificación de trayectorias.

Sin importar la fuente del espacio de trabajo, este se considera como una matriz en la que la presencia de un obstáculo se representa con un 1 en la casilla, mientras que un cero denota un espacio libre en el que se puede transitar.

10.1. Algoritmo para generar trayectoria

El desarrollo del algoritmo encargado de realizar la planificación de trayectorias implicó el uso del algoritmo de Dijkstra para encontrar la ruta óptima entre un punto de inicio y un punto final en un mapa estimado o generado aleatoriamente. Al final de este algoritmo se incluye la representación gráfica del mapa y la ruta, lo que facilita la visualización del resultado obtenido.

El algoritmo comienza definiendo el tipo de mapa que se utilizará. Puede ser un mapa aleatorio, uno predefinido, o un mapa estimado en la exploración llevada a cabo en la simulación en Webots. Dependiendo de la elección, se genera o se carga el mapa correspondiente, estableciendo obstáculos y dimensiones.

Luego, se especifican las coordenadas del punto de inicio y el punto final. En el caso del mapa aleatorio, si el punto de inicio está ubicado en un obstáculo, se realizan ajustes automáticos para garantizar que el punto de inicio sea accesible. Estos ajustes automáticos consideran mover el punto inicial a la casilla superior derecha y validar que no sea una casilla ocupada por un obstáculo.

Se definen las posibles direcciones de movimiento desde cada posición en el mapa, tanto en las cuatro direcciones principales (arriba, abajo, izquierda y derecha) como en las diagonales. Esto permite que el algoritmo de planificación considere una variedad de rutas posibles.

Luego se identifica las posiciones vecinas válidas desde una ubicación dada, asegurando de que no se colisione con obstáculos.

Para calcular la distancia entre dos ubicaciones en el espacio de trabajo se utiliza la conocida fórmula euclíadiana, que se basa en el teorema de Pitágoras, para medir la distancia entre dos puntos en un plano bidimensional.

La fórmula euclíadiana se expresa como:

$$\text{distancia} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (17)$$

El bucle principal en este algoritmo desempeña un papel crucial, ya que hace uso del algoritmo de Dijkstra para encontrar la ruta más corta en un mapa 2D. Este algoritmo es ampliamente utilizado en la planificación de trayectorias y se basa en la búsqueda exhaustiva de un grafo ponderado y dirigido. En este contexto, el mapa se modela como un grafo, donde cada celda representa un nodo, y las conexiones entre celdas adyacentes reflejan los posibles movimientos de un nodo a otro.

La primera fase de la implementación implica la inicialización de dos matrices esenciales: “distancia” y “visitado”. La matriz “distancia” almacena las distancias más cortas desde el punto de inicio hasta cada nodo en el mapa. Inicialmente, todas las distancias se establecen en infinito, excepto la distancia al punto de inicio, que se fija en cero. La matriz “visitado” se utiliza para rastrear qué nodos se han visitado durante la exploración del grafo, con todos los nodos marcados como no visitados al principio.

El algoritmo Dijkstra se caracteriza por su uso de una cola de prioridad llamada “heap” para explorar eficazmente las posiciones disponibles y ajustar continuamente las distancias

para encontrar la ruta óptima. En cada iteración de este bucle principal, se extrae el elemento con la distancia mínima de la cola de prioridad, lo que garantiza que se explore primero la posición más prometedora en términos de distancia acumulada desde el punto de inicio.

En caso de que la posición actual sea igual al punto final, el algoritmo ha encontrado la ruta óptima y finaliza. La ruta óptima se encuentra en la ruta parcial almacenada en el elemento extraído de la cola de prioridad. Sin embargo, si la posición actual no corresponde al punto final, se exploran las posiciones vecinas válidas desde la posición actual. Para cada vecino, se calcula la distancia acumulada desde el punto de inicio hasta ese vecino a través de la posición actual. Si esta distancia acumulada es menor que la distancia previamente registrada para ese vecino en la matriz “distancia”, se actualiza la distancia y se agrega el vecino a la cola de prioridad con su nueva distancia acumulada y la ruta parcial actualizada.

Este proceso se repite hasta que se haya explorado todo el grafo o hasta que se haya encontrado la ruta óptima. Utilizar una cola de prioridad garantizó que el algoritmo explore primero las posiciones más prometedoras, teniendo un impacto directo en el tiempo en el que se llega a la solución. Si no se encuentra una ruta válida, el algoritmo lo señala.

Para visualizar los resultados, se muestra el mapa en dos dimensiones con obstáculos y la ruta óptima en color verde si se encuentra una ruta válida, o en rojo si no se encuentra una ruta. En otra subfigura se representa el mismo mapa pero en tres dimensiones, mostrando obstáculos como cajas negras. En la Figura 73, se presenta un espacio de trabajo en 2D en el que se ha representado una ruta válida. El espacio de trabajo se visualiza como un plano en el que se identifican las posiciones o celdas por las que puede moverse un agente o vehículo. La ruta válida se traza a lo largo de este plano, lo que permite una representación clara y simple de la trayectoria determinada. Por otro lado, en la Figura 74, se presenta el mismo espacio de trabajo y la ruta válida, pero se muestra en 3D.

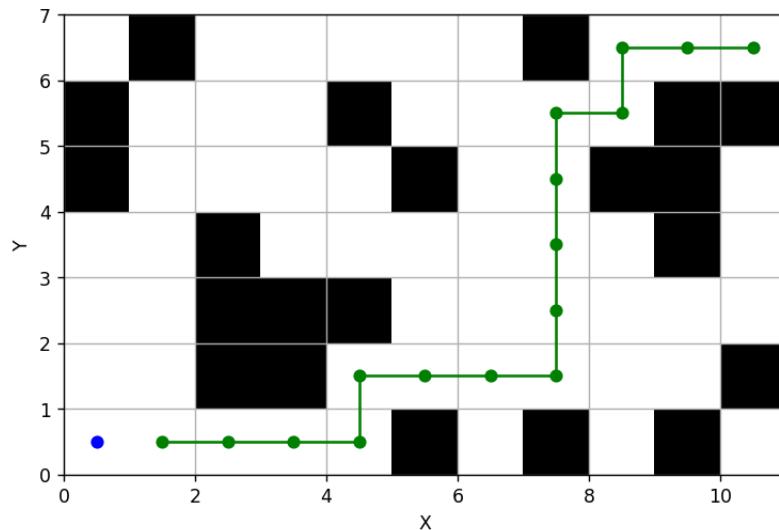


Figura 73: Espacio de trabajo con ruta valida representado en 2D.

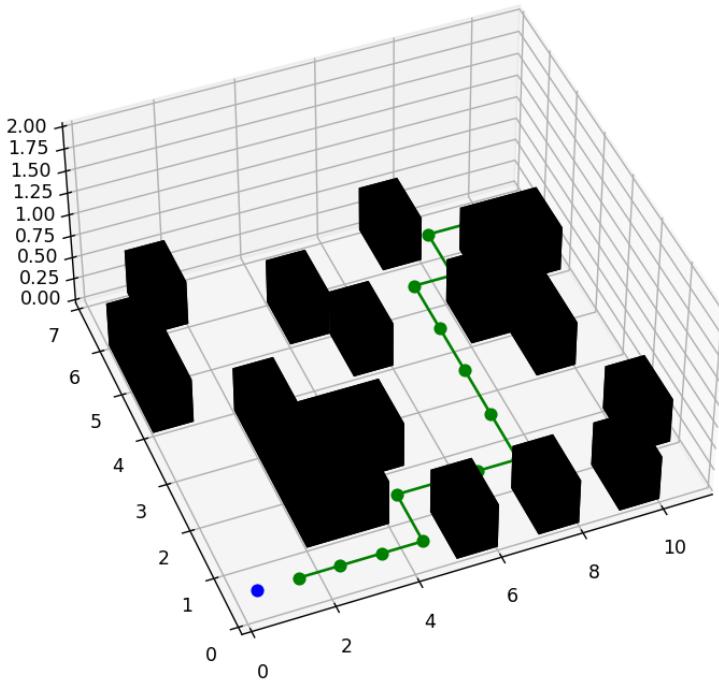


Figura 74: Espacio de trabajo con ruta valida representado en 3D.

10.2. Resultados con mapa generado previamente

En el proceso de evaluación del algoritmo de generación de trayectorias, se realizaron a cabo pruebas utilizando un mapa de dimensiones 20×20 casillas. Un aspecto crítico evaluado en estas pruebas es la influencia de la cantidad de obstáculos presentes en el mapa, y para ello se varía la densidad de obstáculos en el espacio de trabajo. La cantidad de obstáculos se mide en términos de porcentaje de cobertura en relación con el total de casillas disponibles en el mapa.

Específicamente, se realizaron pruebas con tres niveles de cobertura de obstáculos: 0 %, 10 % y 20 %. En el caso de 0 % de cobertura, no se generan obstáculos en el mapa, lo que representa un entorno completamente libre de obstáculos. En el caso del 10 % de cobertura, se introduce una cantidad limitada de obstáculos, lo que significa que el 10 % de las casillas del mapa se marcan como obstáculos, dejando el 90 % restante como espacio transitable. Finalmente, en el caso del 20 % de cobertura, se aumenta la densidad de obstáculos, marcando el 20 % de las casillas como obstáculos.

Estos niveles de cobertura de obstáculos permiten evaluar la capacidad del algoritmo para generar trayectorias válidas en diferentes condiciones. Al variar la densidad de obstáculos, se pueden identificar los posibles desafíos y limitaciones del algoritmo en la planificación de rutas en entornos con obstáculos. Las pruebas se centran en determinar si el algoritmo es capaz de encontrar rutas válidas en cada uno de estos escenarios, lo que es esencial para su aplicación en situaciones donde la presencia de obstáculos es variable y debe ser sorteado de manera efectiva para alcanzar una posición de manera segura y eficiente.

La Figura 75 muestra la trayectoria estimada en el espacio de trabajo previamente descrito, con densidad de obstáculos del 0 %.

En esta representación gráfica, se puede observar que en la esquina inferior izquierda del mapa es el punto de inicio, mientras que el punto alcanzado exitosamente se encuentra en la esquina superior derecha. Lo que destaca en esta observación es que el algoritmo ha logrado conectar estos dos puntos de manera exitosa y directa, sin ningún obstáculo en el camino.

Este resultado ilustra una situación óptima en la que el algoritmo de generación de trayectorias opera de manera efectiva en un entorno libre de obstáculos. La ruta generada es directa y eficiente, lo que demuestra la capacidad del algoritmo para encontrar soluciones sin problemas en condiciones ideales donde no hay obstrucciones que dificulten la navegación.

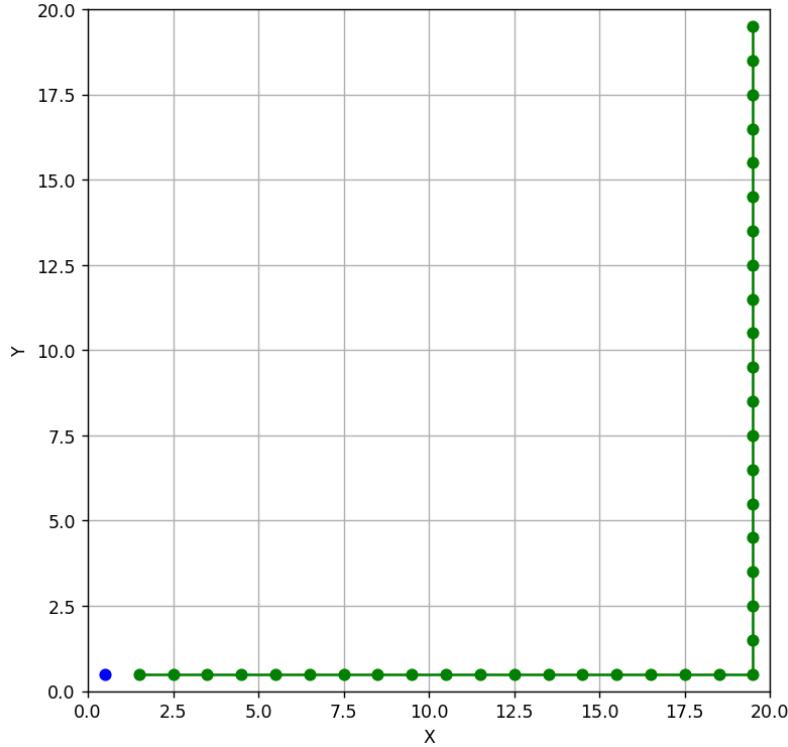


Figura 75: Espacio de trabajo 20×20 con 0 % de obstáculos.

La Figura 76 presenta una trayectoria estimada en el mismo espacio de trabajo de dimensiones 20×20 casillas, pero en este caso, se ha aumentado la complejidad del entorno al introducir obstáculos que representan el 10 % de la cobertura del mapa.

En esta representación gráfica, se mantiene la esquina inferior izquierda del mapa como el punto de inicio, y nuevamente se busca alcanzar con éxito la esquina superior derecha como destino. En este caso a diferencia del escenario sin obstáculos, se han añadido obstáculos al mapa.

El algoritmo de generación de trayectorias mostró su capacidad de sortear con éxito estos obstáculos en su camino hacia el punto de destino. Esto denota la capacidad del algoritmo para encontrar rutas válidas y evitar obstáculos en condiciones de obstáculos moderados.

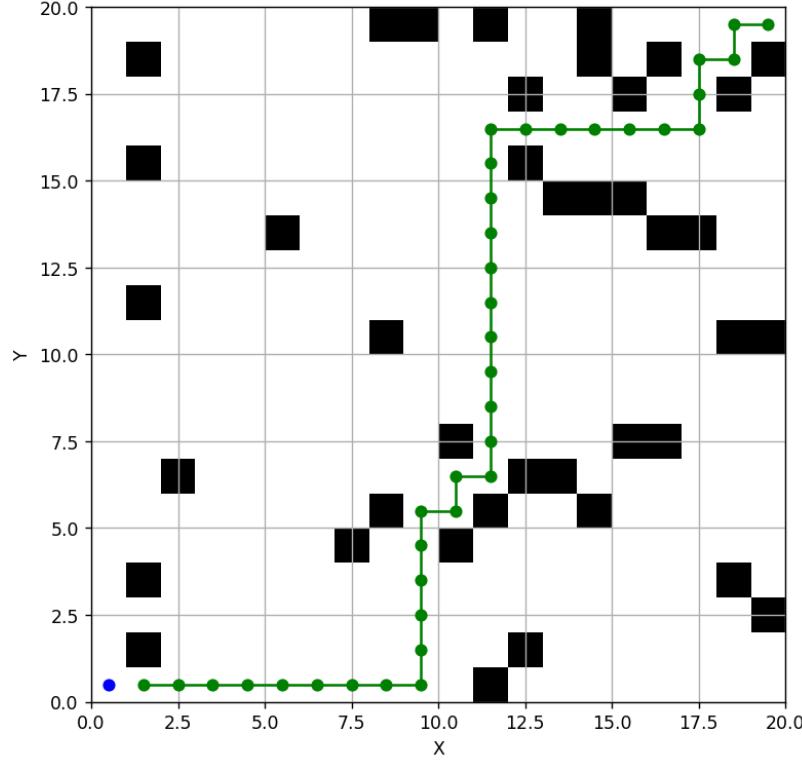


Figura 76: Espacio de trabajo 20×20 con 10 % de obstáculos.

Las Figuras 77 y 78 representan trayectorias estimadas en un espacio de trabajo con dimensiones de 20×20 casillas. En este escenario, se ha incrementado la complejidad del entorno al introducir obstáculos que ahora representan el 20 % de la cobertura total del mapa.

Al igual que en los escenarios anteriores, el punto de inicio se ubica en la esquina inferior izquierda del mapa, y el objetivo es llegar exitosamente a la esquina superior derecha. Sin embargo, en este caso, la cantidad de obstáculos se ha incrementado significativamente. En la Figura 77 la ruta hacia el punto de destino se encuentra bloqueada por obstáculos que rodean la casilla objetivo. Mientras que en la Figura 78 la casilla objetivo se encuentra libre en sus alrededores.

En la Figura 77, se destaca un aspecto importante: a pesar de que el algoritmo ha generado una trayectoria válida hasta el punto en que los obstáculos bloquean el camino, no ha logrado alcanzar la casilla objetivo. Esta observación muestra la capacidad del algoritmo para encontrar rutas válidas hasta donde las condiciones del entorno lo permiten. También resalta sus limitaciones cuando la densidad de obstáculos es tan alta en áreas específicas que obstruye por completo el camino hacia el destino.

En contraste, en el mapa mostrado en la Figura 78, que también tiene una densidad de obstáculos del 20 % pero una distribución diferente, se logró alcanzar exitosamente la casilla objetivo. Esto indica que la capacidad del algoritmo para encontrar una ruta válida depende en gran medida de la disposición y la distribución de los obstáculos en el espacio de trabajo.

En este caso, la distribución de obstáculos permitió que el algoritmo encontrara una ruta que evitara los obstáculos y alcanzara con éxito el destino, a pesar de la presencia de un alto porcentaje de obstáculos en el mapa.

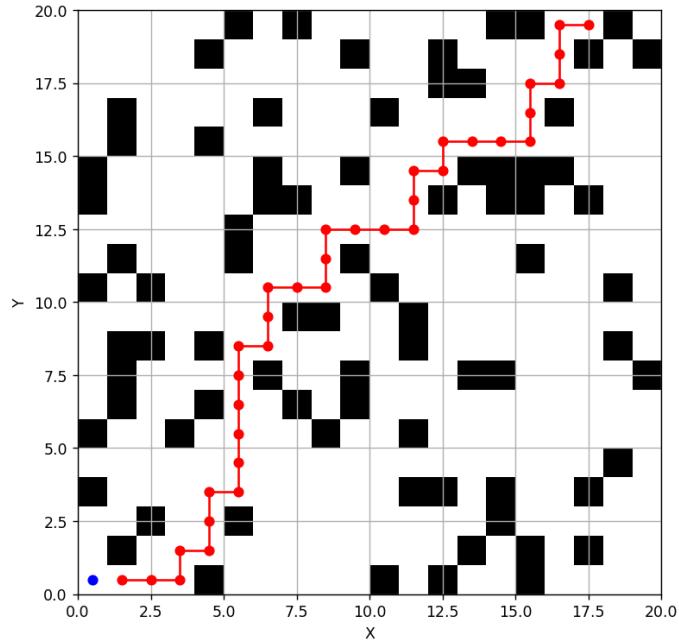


Figura 77: Espacio de trabajo 20×20 con 20 % de obstáculos, ruta valida próxima al objetivo.

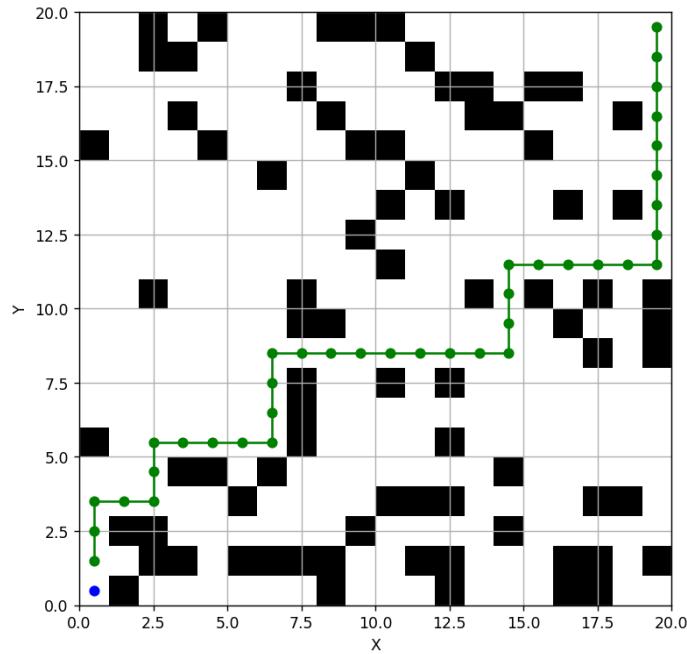


Figura 78: Espacio de trabajo 20×20 con 20 % de obstáculos.

10.3. Resultados con mapa estimado del entorno

Se llevaron a cabo pruebas utilizando el mapa estimado generado durante la fase de exploración. En estas pruebas, se estableció que el punto de partida de la trayectoria correspondía a la posición del vehículo en el momento en que se estimó la trayectoria, mientras que el punto objetivo de la trayectoria se fijó en la posición inicial del vehículo al inicio de la simulación.

La Figura 79 proporciona una representación visual del mapa estimado con el vehículo posicionado inicialmente en la sección 1 del mapa, como se puede apreciar en la Figura 52. La fase de exploración del mapa se prolongó durante un período de 10 minutos. Una vez transcurridos estos 10 minutos, se procedió a calcular la trayectoria requerida para que el vehículo retornara a su punto de partida en la simulación. Los resultados de esta evaluación mostraron su capacidad para determinar trayectorias en espacios de trabajo determinados, ya que la trayectoria fue estimada adecuadamente, permitiendo que el vehículo tenga una ruta que lo regrese a la posición desde la cual había comenzado la simulación.

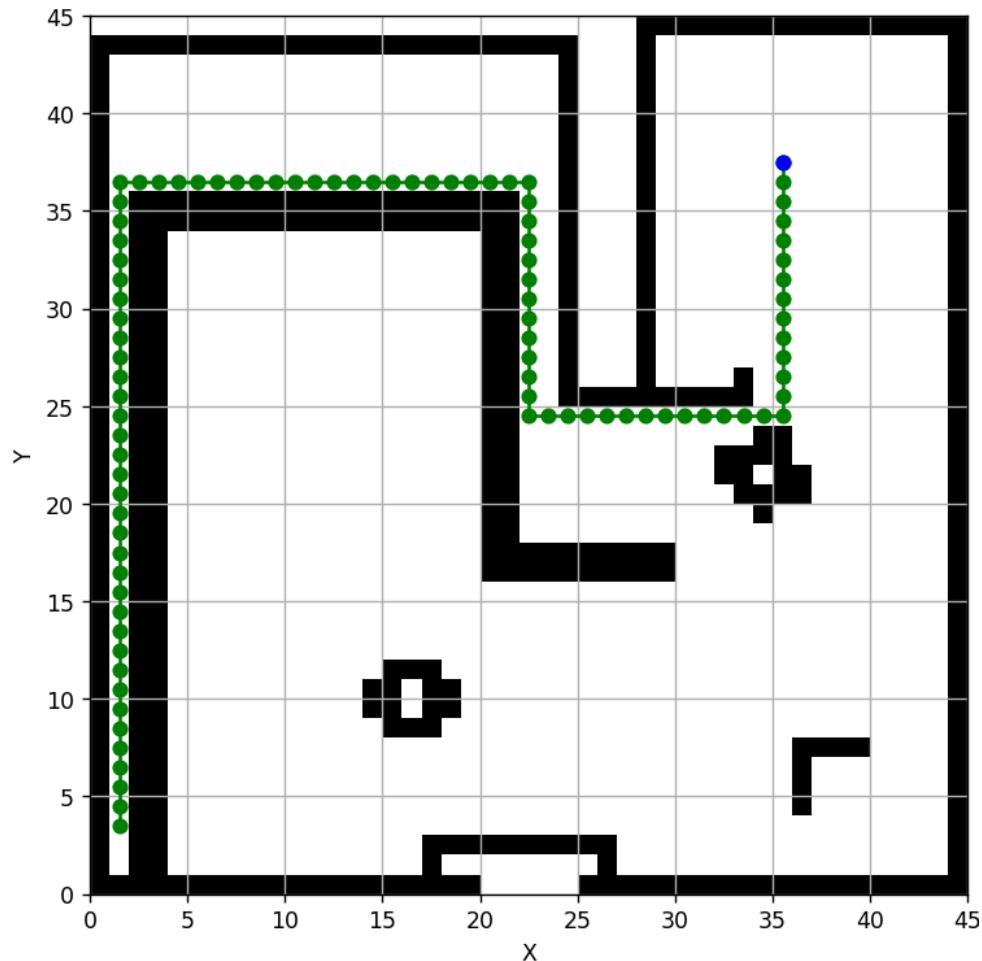


Figura 79: Espacio de trabajo estimado en exploración de 10 minutos, con objetivo de trayectoria posición inicial en sección 1 del mapa.

La Figura 80 proporciona una representación visual del mapa estimado con el vehículo posicionado inicialmente en la sección 6 del mapa, como se puede apreciar en la Figura 57. De igual forma, la fase de exploración del mapa se prolongó durante un período de 10 minutos. Una vez transcurridos estos 10 minutos, se procedió a calcular la trayectoria requerida para que el vehículo retornara a su punto de partida en la simulación. En los mapas estimados a partir de la exploración del vehículo se logró generar trayectorias validadas.

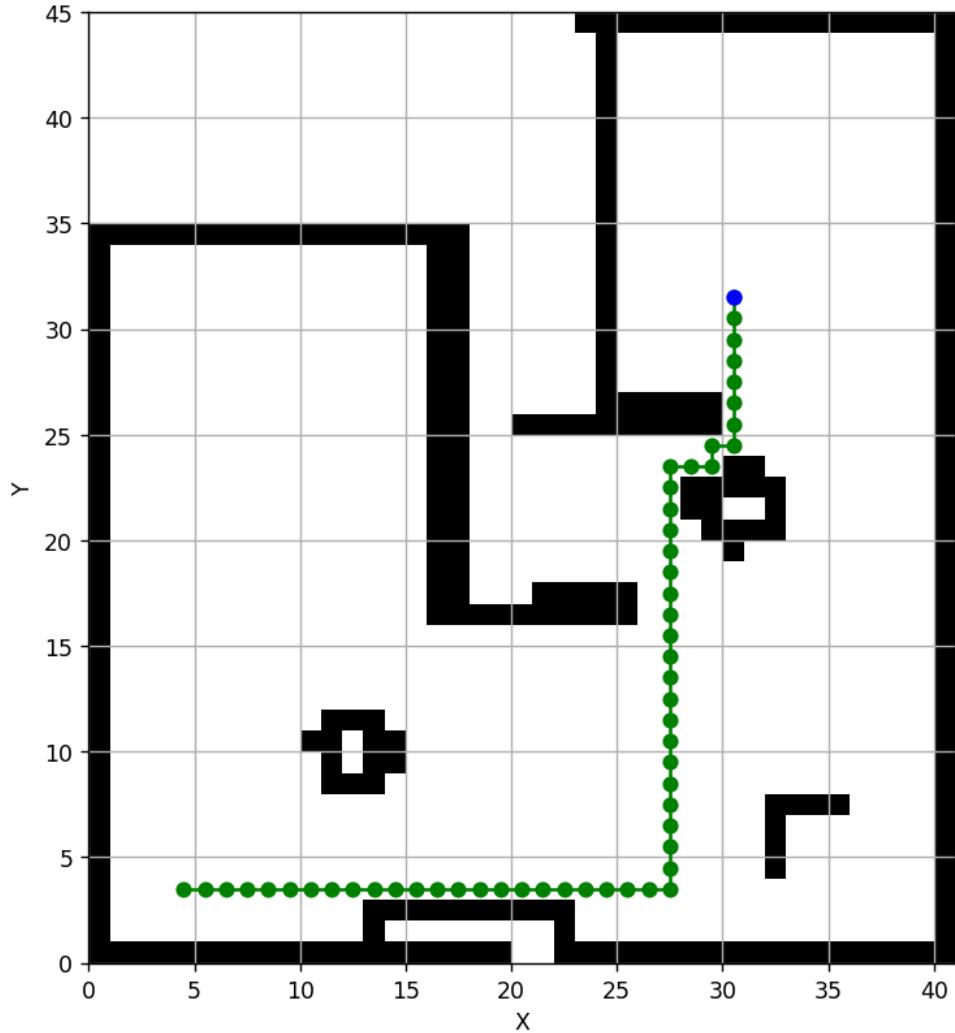


Figura 80: Espacio de trabajo estimado en exploración de 10 minutos, con objetivo de trayectoria posición inicial en sección 6 del mapa.

CAPÍTULO 11

Sensores de distancia para aplicaciones en físico

En este capítulo se presentan sensores de distancia comerciales que pueden emplearse para aplicar en físico los algoritmos desarrollados en este estudio.

En primer lugar, se realizó un análisis de las características y capacidades de diferentes sensores de distancia. Esto implica considerar aspectos como la precisión de las mediciones, el rango de detección y la velocidad de respuesta. Además, se exploran las ventajas y desventajas de cada sensor en términos de tamaño, consumo de energía e interconexión con otros componentes, como un microcontrolador. Este análisis permite identificar las opciones de sensores que mejor se ajustan a las características y restricciones de las aplicaciones previamente definidas.

11.1. Sensor VL53L0X

El sensor láser VL53L0X es un dispositivo miniatura completamente integrado que utiliza la tecnología de *Time-of-Flight* (ToF) para medir distancias absolutas de hasta 2 metros. Este sensor se ha convertido en una herramienta esencial en una amplia gama de aplicaciones, desde robots y sistemas de detección de movimiento hasta drones y aspiradoras robóticas, gracias a su capacidad para proporcionar mediciones de distancia altamente precisas [21].

Una característica destacada de este sensor es su emisor láser de 940 nm, que emite luz infrarroja segura para los ojos humanos. Esto cumple con las normas de seguridad láser y garantiza que el sensor pueda utilizarse sin riesgos para la salud. Además, el VL53L0X cuenta con una matriz SPAD de última generación, que mejora significativamente la sensibilidad y la precisión en la medición de distancias [21].

Otra ventaja del VL53L0X es su capacidad para medir distancias de manera independiente

diente de la reflectancia del objetivo. Esto significa que proporciona mediciones precisas sin verse afectado por el material o la superficie del objeto que se está midiendo. Además, el sensor incluye una sofisticada compensación de *crosstalk* óptico, que reduce las interferencias entre el láser y el receptor, lo que aumenta aún más su precisión. La integración del VL53L0X en proyectos electrónicos es sencilla gracias a su interfaz I^2C estándar, que facilita el control del dispositivo y la transferencia de datos [21].

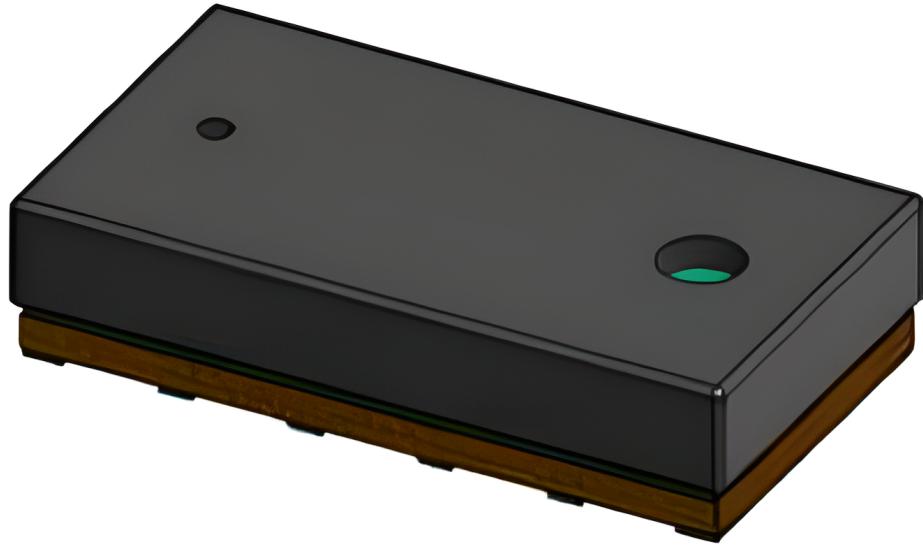


Figura 81: Modulo sensor de distancia VL53L0X [21].

11.2. Sensor VL53L1X

El sensor VL53L1X es un sensor láser basado en la tecnología (ToF), capaz de proporcionar mediciones precisas de distancia de hasta 4 metros y una frecuencia de rango rápida de hasta 50 Hz. Este sensor tiene un tamaño de módulo de $4.9 \times 2.5 \times 1.56$ mm. Este sensor permite medir la distancia de manera absoluta sin importar el color ni la reflectancia del objeto en cuestión [22].

El sensor tiene un campo de visión (FoV) típico de 27 grados. Además, cuenta con un tamaño de región de interés (ROI) programable en la matriz receptora, lo que permite reducir el FoV del sensor, y una posición de ROI programable en la matriz receptora, lo que brinda control de operación multizona desde el host. El sensor funciona con una sola fuente de alimentación de hasta 3.5 voltios y ofrece una interfaz I^2C , de hasta 400 kHz, con pines de apagado y de interrupción [22].



Figura 82: Modulo sensor de distancia VL53L1X [22].

CAPÍTULO 12

Conclusiones

- Los sensores de distancia tipo láser son mas precisos y eficaces en la detección de obstáculos comparado a los sensores de distancia tipo sonar. Esta diferencia notoria hace que los sensores láser sean la mejor opción para aplicaciones que requieren una detección de obstáculos en vehículos diferenciales y entornos similares.
- La disposición de los sensores de distancia de manera fija en el vehículo tiene implicaciones significativas para la percepción del entorno y la navegación autónoma. Esta configuración proporciona una base sólida para la detección constante de obstáculos y la toma de decisiones, lo que resulta en una mayor estabilidad y previsibilidad en la operación del vehículo. Sin embargo, es importante considerar cuidadosamente la ubicación y orientación de estos sensores para garantizar una cobertura óptima y minimizar puntos ciegos en el entorno de trabajo.
- Utilizar 6 sensores de distancia y una referencia de orientación, permite la movilidad y la exploración de un vehículo. Esta cantidad de sensores de distancia proporcionan una cobertura adecuada del entorno, lo que permite al vehículo detectar y evadir obstáculos de manera confiable durante su operación autónoma.
- La estrategia de limitar el desplazamiento del vehículo a únicamente en línea recta en el proceso de exploración y definir claramente los cambios de giro, se presenta como una metodología efectiva para obtener una estimación sólida de la posición del vehículo en entornos desconocidos
- La correcta y precisa estimación de la trayectoria de exploración es un elemento importante para lograr una precisa estimación del espacio de trabajo. La planificación de una trayectoria efectiva y estratégica permite al vehículo recorrer el entorno de manera sistemática y obtener datos confiables sobre el espacio circundante.
- Utilizar el algoritmo de Dijkstra en la generación de trayectorias es una opción efectiva. El desarrollo del algoritmo de generación de trayectorias basado en la búsqueda de caminos más cortos en grafos ponderados, ha demostrado ser confiable y eficiente para la planificación de rutas.

- Webots es una plataforma que proporciona un entorno virtual versátil y controlado que permite probar algoritmos, evaluar estrategias y validar soluciones antes de la implementación en el mundo real. El uso de simulaciones en Webots brinda ventajas, como la reducción de tiempos en pruebas. Además de la capacidad de realizar experimentos repetibles y variados. Sin embargo, es importante tener en cuenta que la precisión de los resultados en el mundo simulado depende en gran medida de la fidelidad de la simulación y la calidad de los modelos utilizados.

CAPÍTULO 13

Recomendaciones

- En aplicaciones físicas que demandan una detección de obstáculos precisa y fiable, se aconseja utilizar sensores de distancia tipo láser. Estos sensores destacan por su mayor precisión y resolución en contraste con otros tipos de sensores, lo que los convierte en la elección ideal para entornos donde la seguridad y la precisión son factores cruciales. Por lo tanto, resulta esencial mantener una línea de investigación centrada en robots móviles que implementen sensores láser, incluyendo sensores Lidar. Estos dispositivos contribuyen de manera significativa a la capacidad de los robots para navegar y operar de manera eficiente y segura en entornos físicos complejos.
- En aplicaciones físicas que requieren exploración precisa y eficiente, se aconseja el uso de vehículos de 4 ruedas. Estos vehículos ofrecen ventajas notables, ya que la redundancia de información proporcionada por múltiples ruedas mejora significativamente la precisión de la odometría y la capacidad de avanzar en línea recta de manera más confiable.

CAPÍTULO 14

Bibliografía

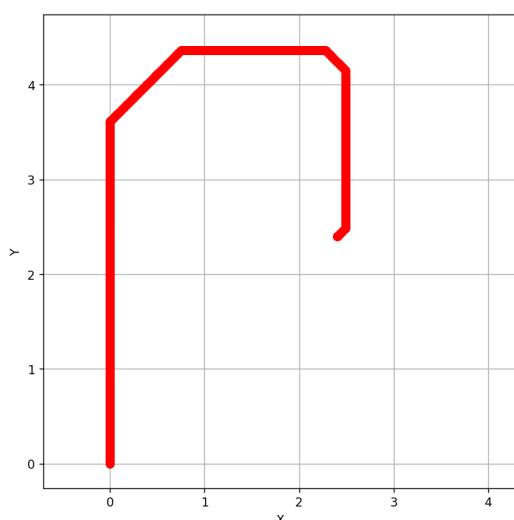
- [1] Pololu Robotics and Electronics, *Pololu 3pi+*, <https://www.pololu.com/product/3737>, 2023.
- [2] D. Baldizón, “Aplicaciones Prácticas para Algoritmos de Inteligencia y Robótica de Enjambre,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.
- [3] S. Thrun, W. Burgard y D. Fox, “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping,” en *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, 321-328 vol.1. DOI: 10.1109/ROBOT.2000.844077.
- [4] I. Eliakim, Z. Cohen, G. Kosa e Y. Yovel, “A fully autonomous terrestrial bat-like acoustic robot,” *PLOS Computational Biology*, vol. 14, n.º 9, págs. 1-13, sep. de 2018. DOI: 10.1371/journal.pcbi.1006406. dirección: <https://doi.org/10.1371/journal.pcbi.1006406>.
- [5] A. Rohler, J. Otero y S. Gonzales, “Traffic Flow Estimation Using Ant Colony Optimization Algorithms,” vol. 18, n.º 1, págs. 37-50, 2014.
- [6] H. Jun, Y. Fu y Z. Qun, “An Improved Ant Colony Algorithm of Robot Path Planning for Obstacle Avoidance,” *Journal of Robotics*, vol. 2019, n.º 1, pág. 8, 2019.
- [7] S. de los Cobos, M. Gutiérrez y E. García, “Colonia de abejas artificiales y optimización por enjambre de partículas para la estimación de parámetros de regresión no lineal,” *Revista de Matemática Teoría y Aplicaciones*, vol. 21, n.º 1, págs. 107-126, 2014.
- [8] A. M. Peña, “Algoritmo de sincronización y control de sistemas de robots multi-agente para misiones de búsqueda,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.
- [9] Cyberbotics Ltd., *Webots*, <https://cyberbotics.com>, 2022.
- [10] E. A. Santizo, “Aprendizaje Reforzado y Aprendizaje Profundo en Aplicaciones de Robótica de Enjambre,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

- [11] A. S. Aguilar, “Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO),” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.
- [12] K. Senn, “Implementación de un sistema de localización y mapeo simultáneo mediante un escáner Lidar Hokuyo como un nodo en ROS para el Rover UVG,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.
- [13] B. S. G. de Almeida y V. C. Leite, “Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems,” en *Swarm Intelligence*, J. D. Ser, E. Villar y E. Osaba, eds., Rijeka: IntechOpen, 2019, cap. 3. DOI: 10.5772/intechopen.89633. dirección: <https://doi.org/10.5772/intechopen.89633>.
- [14] D. Zhang, X. You, S. Liu y K. Yang, “Multi-Colony Ant Colony Optimization Based on Generalized Jaccard Similarity Recommendation Strategy,” *IEEE Access*, vol. 7, págs. 157303-157317, 2019. DOI: 10.1109/ACCESS.2019.2949860.
- [15] M. Dorigo y L. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, n.º 1, págs. 53-66, 1997. DOI: 10.1109/4235.585892.
- [16] C. Robles, “Optimización por colonia de hormigas: aplicaciones y tendencias,” *Journal of Engineering Education*, vol. 6, págs. 83-89, 2019.
- [17] Y. Maddahi y K. Zareinia, “Nonparametric Bootstrap Technique to Improve Positional Accuracy in Mobile Robots With Differential Drive Mechanism,” *IEEE Access*, vol. 8, págs. 158502-158511, 2020. DOI: 10.1109/ACCESS.2020.3020864.
- [18] G. Uribe y C. David, “Comparación de un Controlador Lqr Vs un Controlador Pid Implementados en un Helicóptero de dos Grados de Libertad Pivotado,” 2016.
- [19] H. Lu, S. Yang, M. Zhao y S. Cheng, “Multi-Robot Indoor Environment Map Building Based on Multi-Stage Optimization Method,” *Complex System Modeling and Simulation*, vol. 1, n.º 2, págs. 145-161, 2021. DOI: 10.23919/CSMS.2021.0011.
- [20] E. Cabrera, R. Moreno y M. Torres, “Integración de tecnología lidar en vehículo para escaneo con ROS,” *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 2022.
- [21] life.augmented, *Time-of-Flight ranging sensor*, <https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html>, 2022.
- [22] life.augmented, <https://www.st.com/resource/en/datasheet/vl53l1x.pdf>, <https://www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html>, 2022.
- [23] K. Ruohonen, *Graph Theory*. Flooved.com, 2008. dirección: <https://archive.org/details/flooved3467/page/n9/mode/2up>.

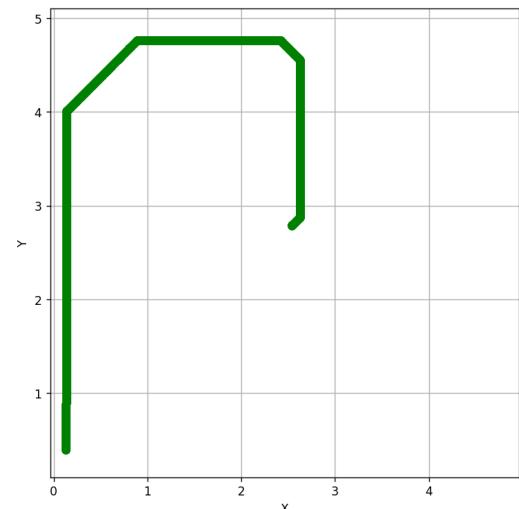
CAPÍTULO 15

Anexos

15.1. Resultados usando sensores de distancia tipo sonar



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 83: Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo sonar con simulación de 5 min.

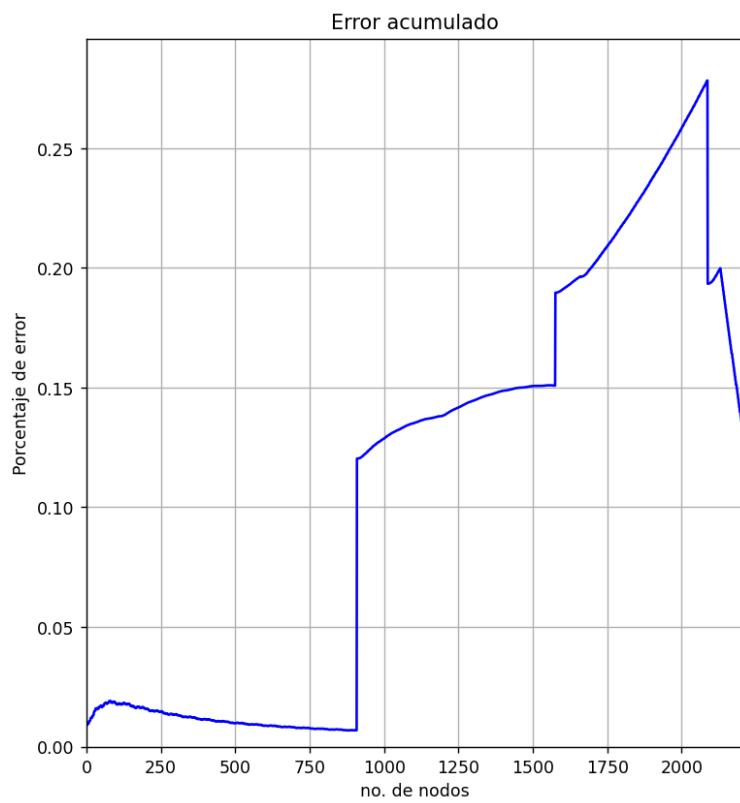
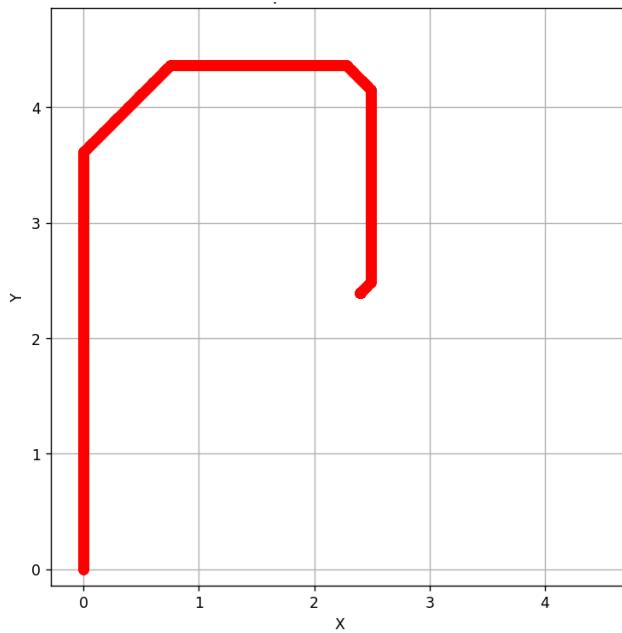
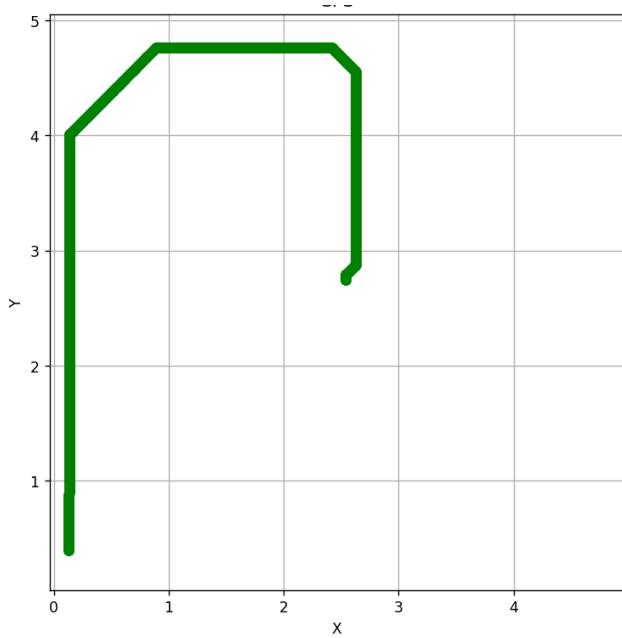


Figura 84: Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo sonar en simulación de 5 min.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 85: Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo sonar con simulación de 60 min.

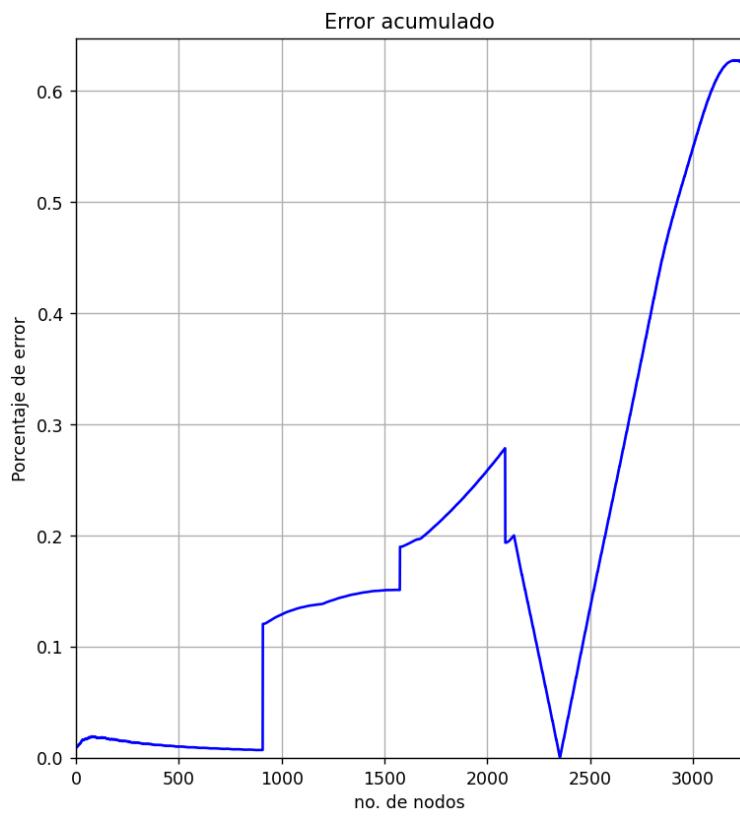
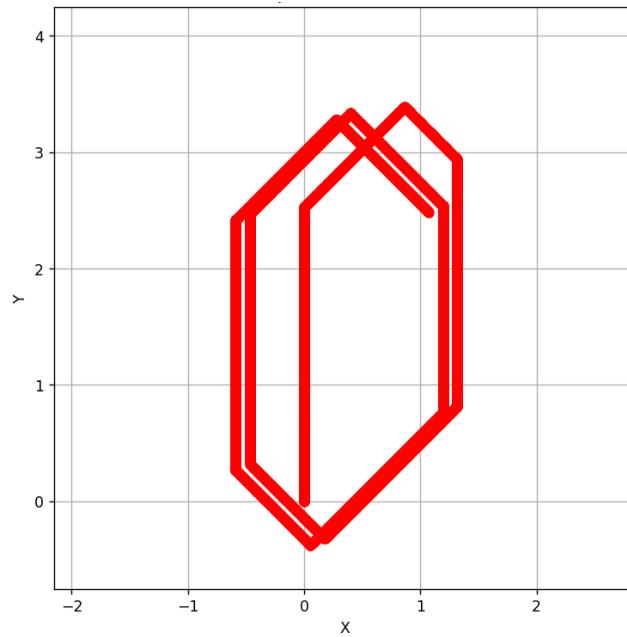
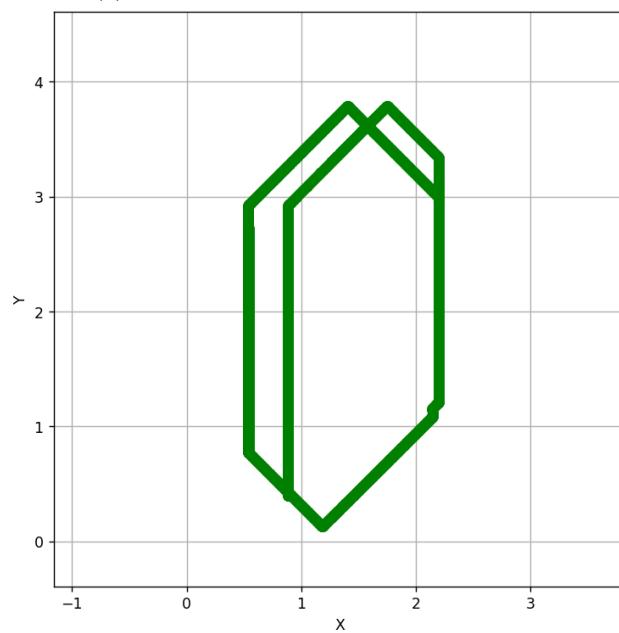


Figura 86: Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo sonar en simulación de 60 min.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 87: Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo sonar con simulación de 5 min.

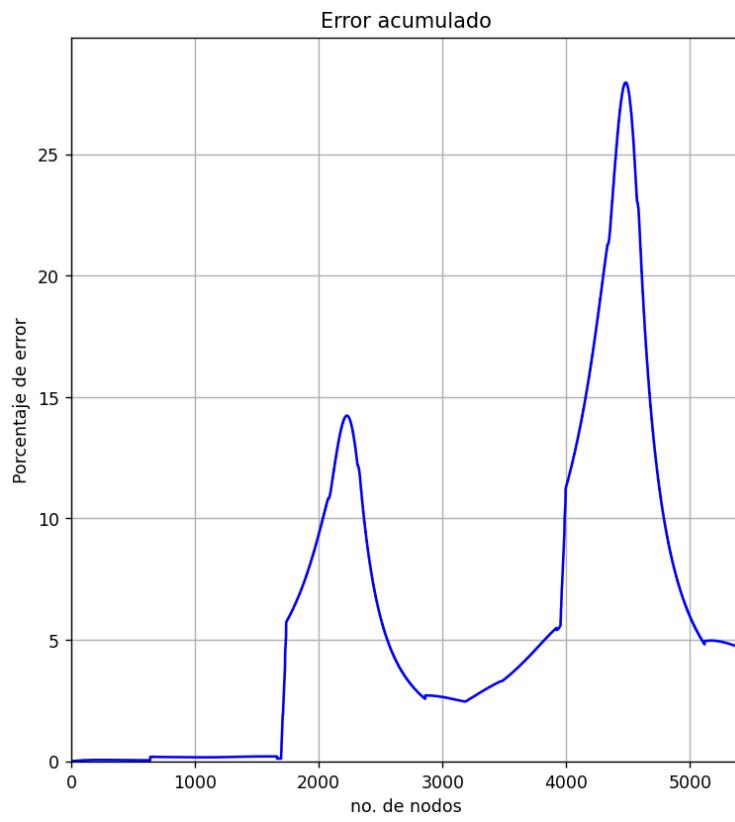
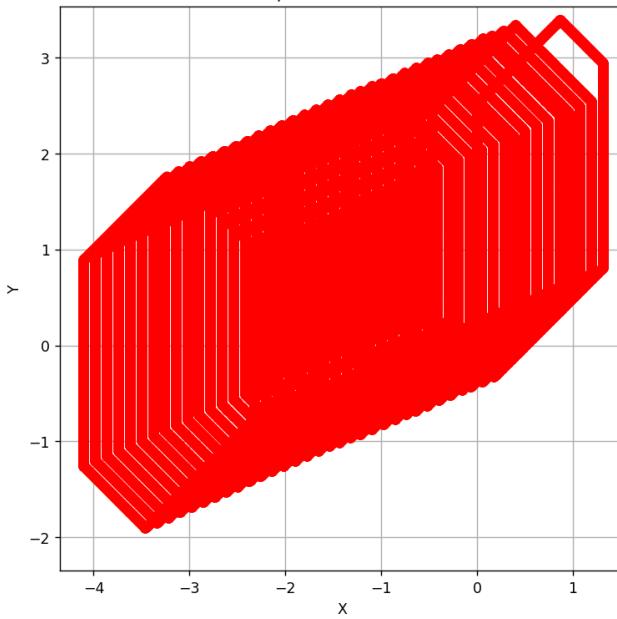
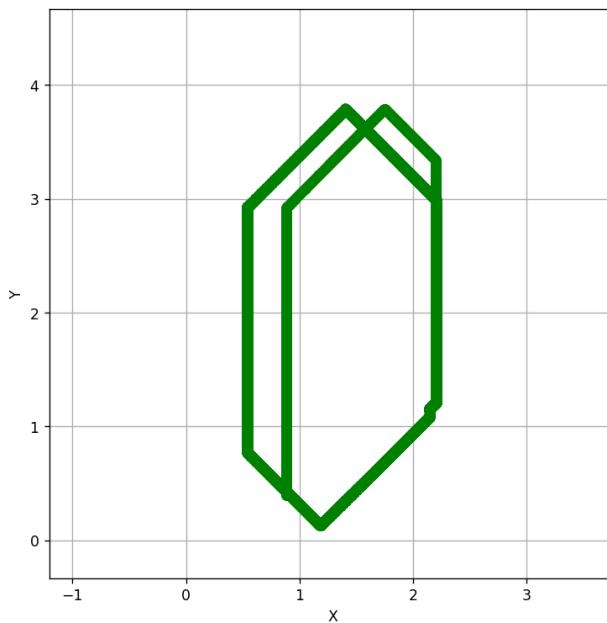


Figura 88: Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo sonar en simulación de 5 min.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 89: Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo sonar con simulación de 60 min.

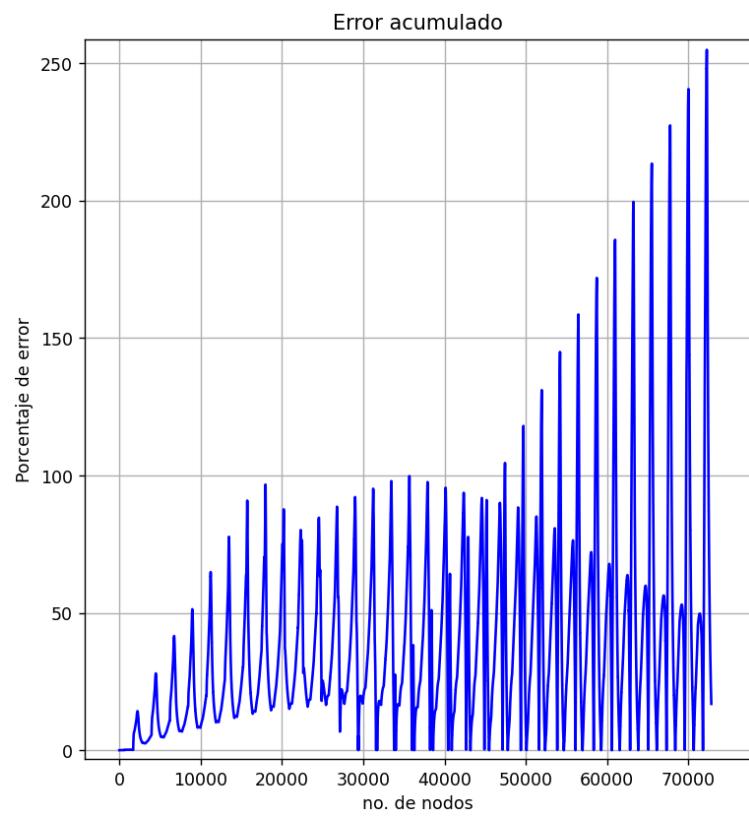
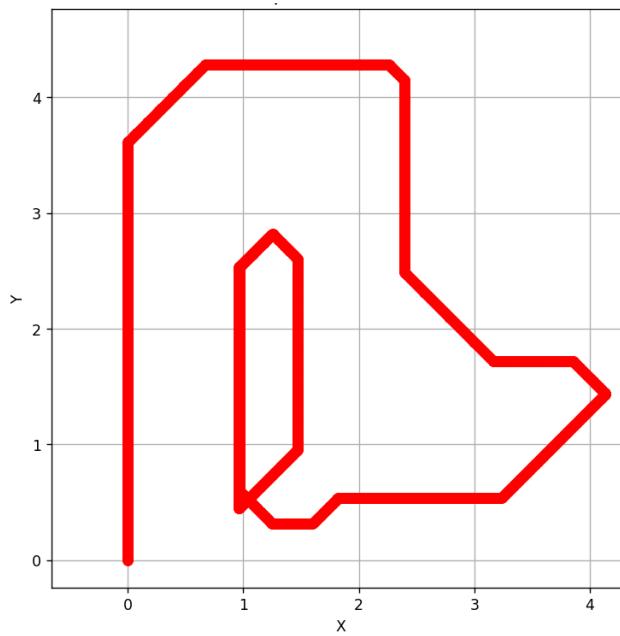
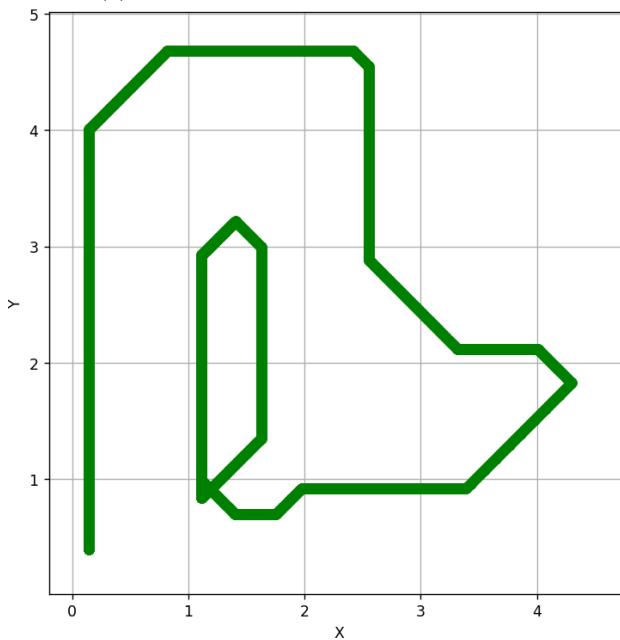


Figura 90: Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo sonar en simulación de 60 min.

15.2. Resultados usando sensores de distancia tipo láser



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 91: Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo láser con simulación de 5 min.

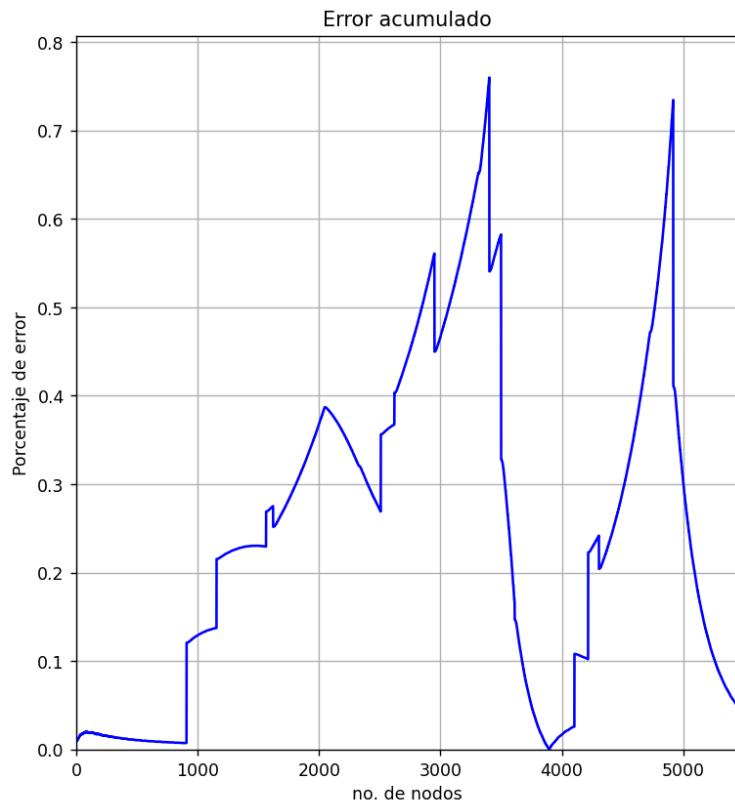
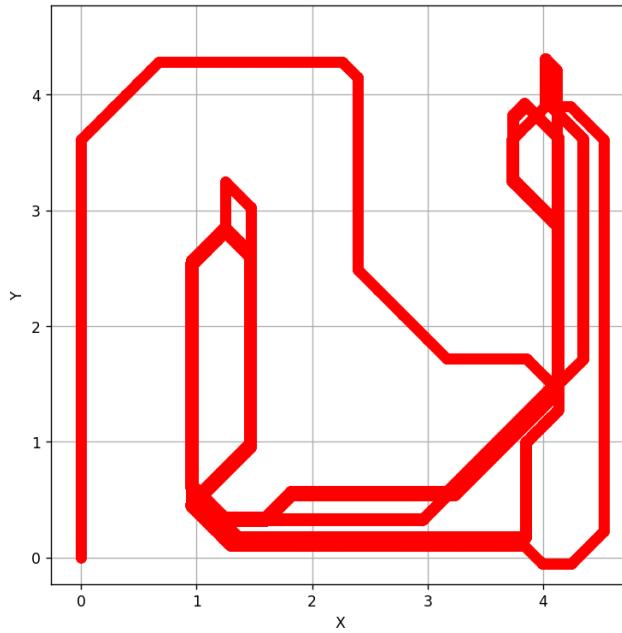
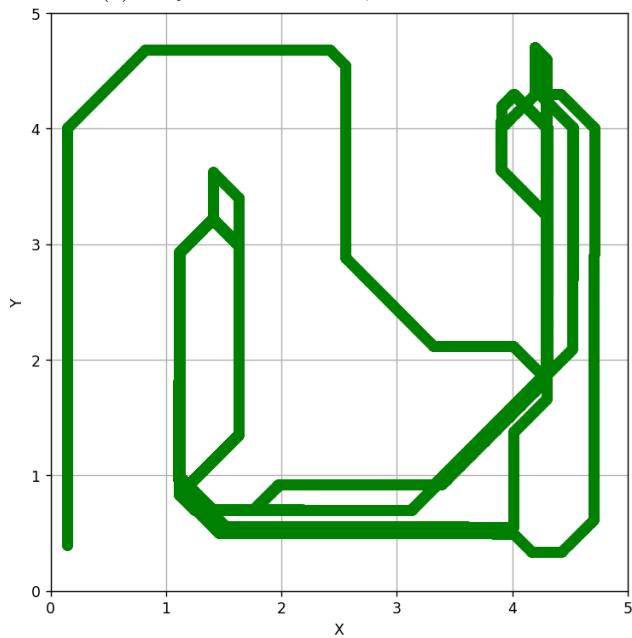


Figura 92: Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo láser en simulación de 5 min.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 93: Trayectoria de exploración iniciando desde sección 1 del mapa usando sensores de distancia tipo láser con simulación de 60 min.

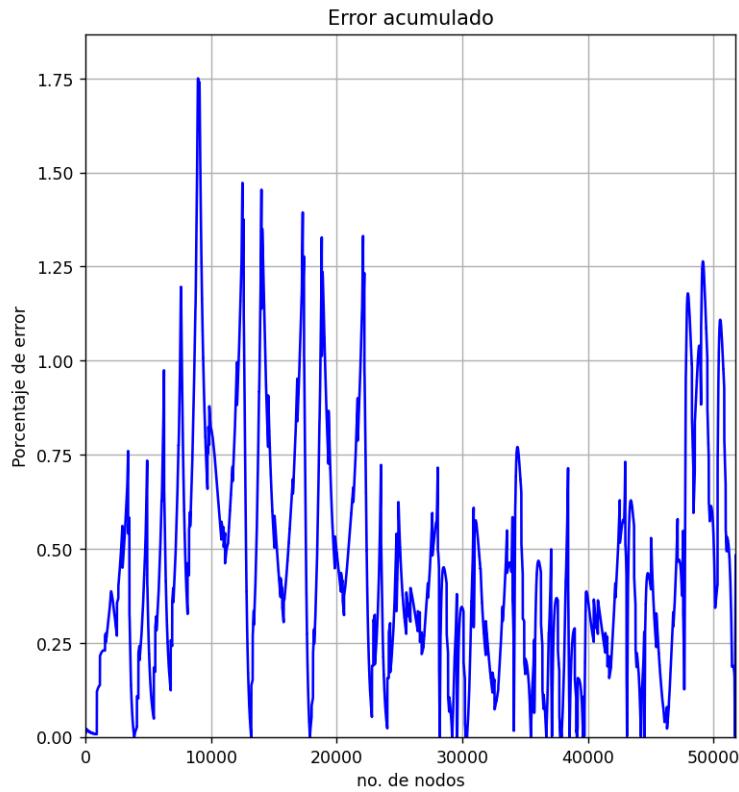
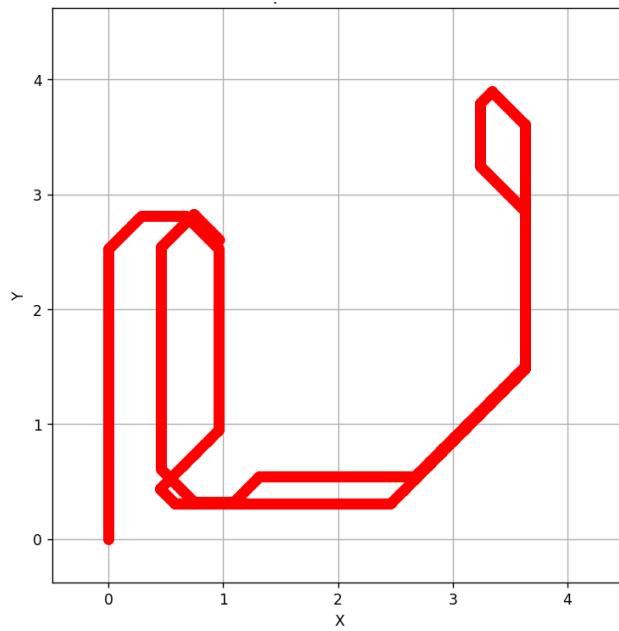
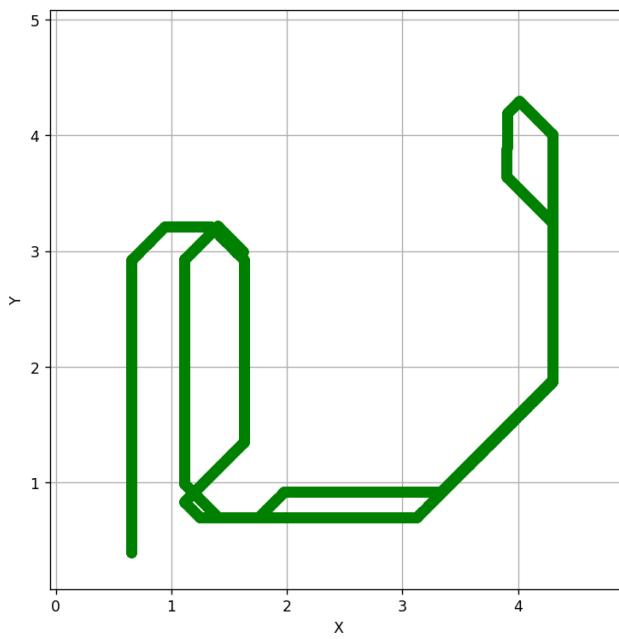


Figura 94: Error acumulado en trayectoria de exploración iniciando desde sección 1 usando sensor de distancia tipo láser en simulación de 60 min.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 95: Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 5 min.

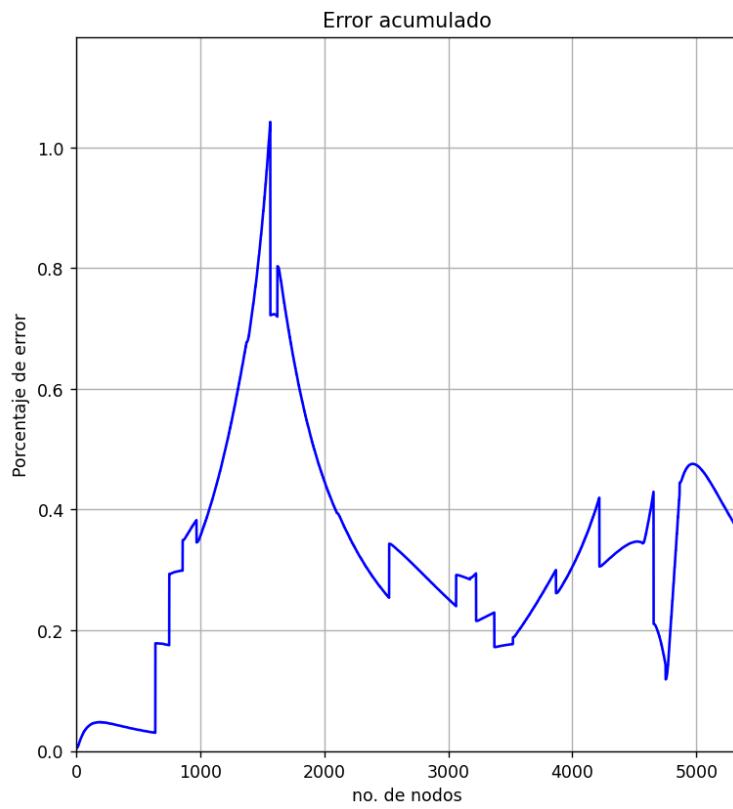
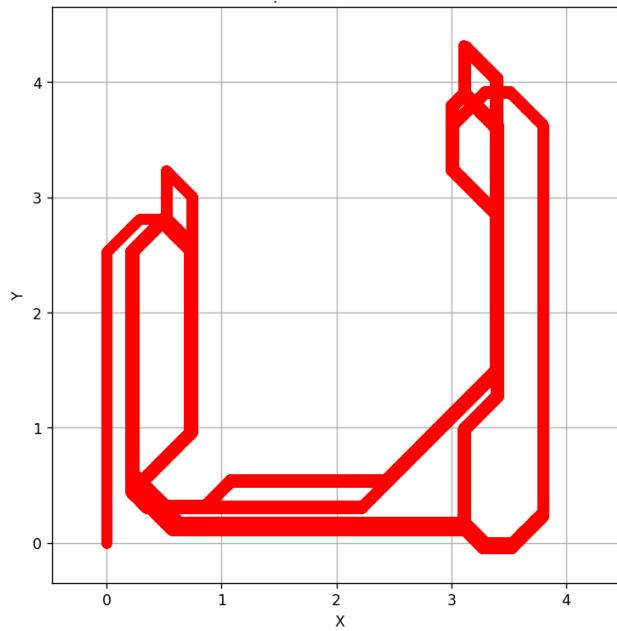
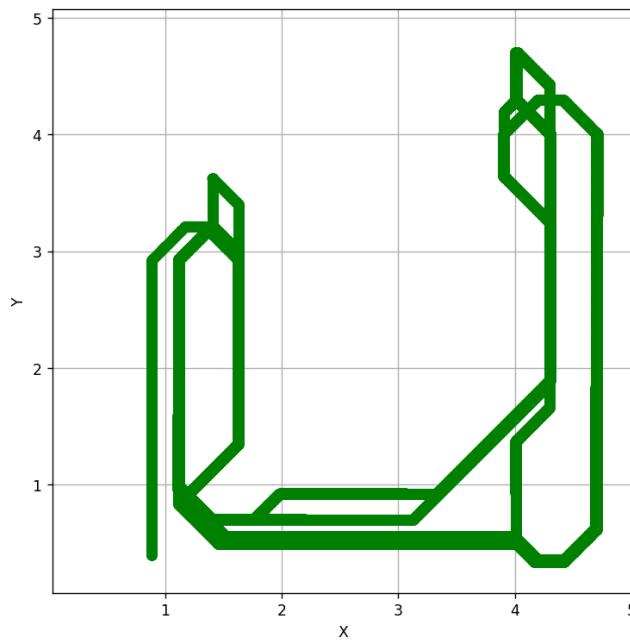


Figura 96: Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 5 min.



(a) Trayectoria estimada, unidades en m.



(b) Trayectoria obtenida con GPS, unidades en m.

Figura 97: Trayectoria de exploración iniciando desde sección 6 del mapa usando sensores de distancia tipo láser con simulación de 60 min.

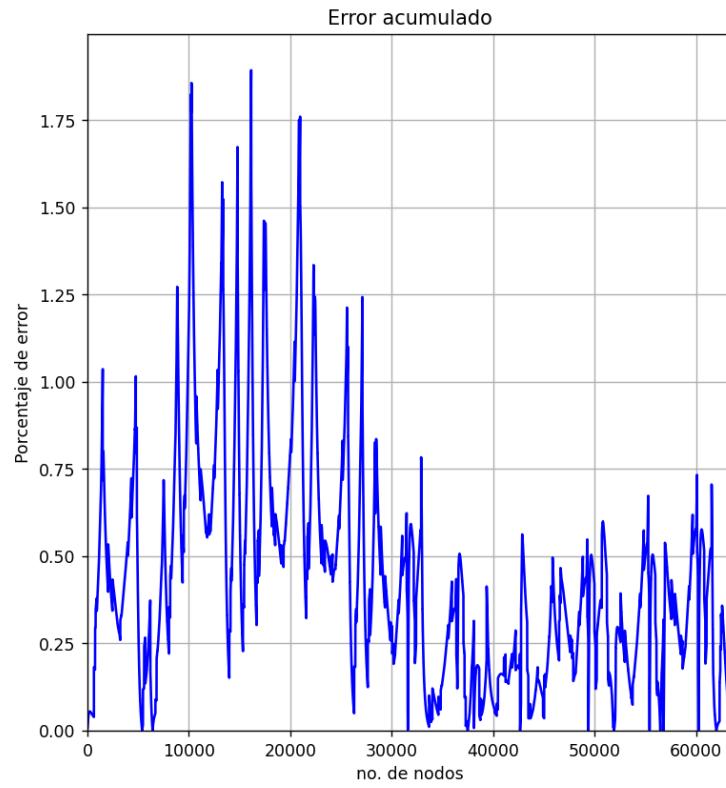


Figura 98: Error acumulado en trayectoria de exploración iniciando desde sección 6 usando sensor de distancia tipo láser en simulación de 60 min.

CAPÍTULO 16

Glosario

grafo Es un par de conjuntos (V, E), donde V es el conjunto de vértices y E es el conjunto de aristas, formado por pares de vértices. E es un multiconjunto, es decir, sus elementos pueden ocurrir más de una vez, de modo que cada elemento tiene una multiplicidad. A menudo, etiquetamos los vértices con letras (por ejemplo: a, b, c, \dots o v_1, v_2, \dots) o números ($1, 2, \dots$). [23]. 85