PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO FACULTAD DE INGENIERÍA ESCUELA DE INGENIERÍA INFORMÁTICA

Proyecto Final: Robótica y Sistemas Autónomos

Cesar Anabalón, Ulysses Barreto, Felipe Bravo, Matias Guerrero, Fabiana Piña
Robótica y Sistemas Autónomos - Primer Semestre 2025
26 de junio del 2025

Índice

1. Diseño del Proyecto

1.1. Descripción del Robot Móvil y Características

El robot móvil diseñado en este proyecto corresponde a una plataforma con locomoción diferencial de cuatro ruedas, simulada completamente en el entorno Webots. Su arquitectura física está orientada a la exploración y navegación autónoma en entornos estructurados con obstáculos.

Las principales características del robot son las siguientes:

1. **Locomoción:** El robot posee cuatro ruedas motrices con tracción independiente. En donde, cada rueda está conectada a un motor eléctrico que puede ser controlado individualmente, permitiendo avanzar, retroceder, girar sobre su eje o ejecutar maniobras de evasión. El control de los motores se realiza mediante comandos de velocidad angular, utilizando la función wb_motor_set_velocity(). Los motores están configurados en modo de posición infinita, permitiendo un control continuo de la velocidad en lugar de un control por posición.

2. **Autonomía:** El robot opera de forma autónoma dentro del entorno simulado, sin intervención externa directa. Por lo que, toma decisiones en tiempo real basándose en la información obtenida desde sus sensores.

3. **Sensores y percepción:** El robot cuenta con sensores que le permiten percibir su entorno, detectar obstáculos, conocer su ubicación y construir un modelo interno del espacio, es decir el mapa.

4. **Capacidad de mapeo:** Utiliza la información del LIDAR y GPS para construir una grilla de ocupación, que en este caso es una matriz binaria, que representa el entorno, indicando qué celdas están ocupadas por obstáculos.

5. **Navegación autónoma:** Se implementa, el algoritmo A* para encontrar rutas óptimas desde su ubicación actual hacia un objetivo, evitando zonas marcadas como ocupadas en el mapa.

Por lo que, este diseño integra todos los elementos necesarios para abordar tareas de navegación, mapeo y evitación de obstáculos de manera eficiente y modular.

1.2. Explicación del Entrono Webots

El entorno simulado fue desarrollado en el simulador Webots, el dónde se utiliza, su editor de mundos (proyecto.wbt). Y este entorno emula un espacio cerrado con obstáculos dispersos, donde el robot debe navegar de forma autónoma.

Componentes del entorno:

1. **Escenario rectangular:** El área de simulación es un plano de dimensiones finitas que representa una sala o espacio cerrado. Los límites del entorno impiden que el robot se salga del área de trabajo.

2. **Obstáculos:** Se han ubicado bloques de tamaño variable en posiciones predefinidas para representar paredes o elementos que dificultan el paso. Estos objetos actúan como obstáculos estáticos dentro del mapa que el robot debe detectar y evitar.

3. **Robot móvil:** Se define como un nodo Robot en Webots. Contiene sensores (GPS, LIDAR, sensores de distancia) y actuadores (motores para las 4 ruedas). Se inicializa en una posición central dentro del entorno.

4. **Sensores virtuales:**
   - GPS: Proporciona la posición global (X, Z) del robot.
   - LIDAR: Devuelve distancias a obstáculos en un rango angular definido.
   - Sensores de distancia: Detectan objetos cercanos en los laterales del robot.

Por es esto, todo el entorno fue construido con el objetivo de permitir el desarrollo de tareas de localización, planificación de rutas, evasión de obstáculos y mapeo del espacio.

1.3. Arquitectura de Software

El sistema de control del robot fue programado en lenguaje C utilizando la API proporcionada por Webots. La arquitectura sigue un enfoque modular, dividiendo las responsabilidades en lectura de sensores, lógica de planificación y control de actuadores.

1.3.1. Sensores

1. **LIDAR:** Es el sensor principal para detección de obstáculos. El que genera un conjunto de distancias, que es el rango de imagen, en un campo de visión de aproximadamente de $160°$, pero este valor depende del modelo. Los datos del LIDAR se procesan para detectar obstáculos cercanos y para actualizar la grilla del mapa.

2. **GPS:** Proporciona la posición absoluta del robot en coordenadas X y Z. Esta información se utiliza para ubicar correctamente las lecturas del LIDAR dentro del sistema global de coordenadas. Lo que permite determinar la celda actual que ocupa el robot dentro del mapa.

3. **Sensores de distancia:** Son sensores infrarrojos montados a los lados del robot (izquierdo y derecho). Los cuales se utilizan para detectar obstáculos cercanos (menores a $0.95\,m$) que el LIDAR podría no captar en rangos estrechos.

1.3.2. Actuadores

Motores de las ruedas: Cada una de las cuatro ruedas cuenta con un motor individual (wheel1 a wheel4). Se configuran para funcionar en modo velocidad continua (posición = INFINITY). La velocidad de cada rueda se controla mediante valores entre 0 y $6.28\,rad/s$, la cual es la velocidad máxima. Permiten avanzar en línea recta, girar en el lugar (rotación diferencial) y ejecutar maniobras de retroceso ante obstáculos.

1.3.3. Módulos de Control

El programa principal se estructura en tres módulos funcionales:

1. **Movimiento reactivo:** Controla el avance y giro del robot. Si no hay obstáculos detectados, se avanza hacia el siguiente punto de la ruta. Si se detecta un obstáculo cercano, se ejecuta un giro evasivo temporal.

2. **Planificación de rutas (A*):** Se implementa el algoritmo A* sobre una grilla de 8x8. Calcula el camino más corto desde la celda actual del robot hasta una celda destino. Evita celdas marcadas como ocupadas por obstáculos.

3. **Mapeo:** A partir de los datos del LIDAR y del GPS, se calcula la posición global de los obstáculos detectados. Estos se proyectan sobre una grilla, marcando las celdas correspondientes como ocupadas (valor 1).

1.4. Algoritmo a Utilizar

Durante el desarrollo del proyecto se implementan tres tipos de algoritmos principales:

1. **Planificación A*:** Algoritmo heurístico de búsqueda en grafos que permite encontrar el camino más corto desde una celda inicial hasta una celda objetivo dentro de una matriz bidimensional. Por lo que se utiliza una heurística de distancia Manhattan ($|x1 - x2| + |y1 - y2|$) para estimar el costo restante. La matriz de ocupación es de 8×8, donde cada celda representa un área de $0.5\ m^2$.

2. **Evitación reactiva:** Si los sensores de distancia o el LIDAR detectan un obstáculo a menos de $0.3\ m$, se interrumpe el avance y el robot gira sobre su eje por unos pasos definidos. Este comportamiento no requiere planificación, es una respuesta rápida ante detección crítica.

3. **Mapeo local:** Los datos del LIDAR son transformados usando trigonometría y coordenadas del GPS para ubicar los obstáculos en el espacio global. Luego se convierten en índices de celda para actualizar la grilla de ocupación.

Por lo que, estos algoritmos trabajan de manera complementaria, permitiendo que el robot explore su entorno, evite colisiones y se desplace eficientemente hacia un objetivo.

1.5. Diagramas de Flujo y Pseudocodigo de la Solución

Algorithm 1 Algoritmo General del Robot Autónomo en Webots

```
Inicializar robot, motores, sensores de distancia, LIDAR y GPS
Crear grilla vacía
Definir punto de inicio y punto meta
while la simulación esté activa do
    Leer sensores de distancia
    if se detecta un obstáculo cercano then
        Activar flag_obstaculo_cercano
    end if
    Leer posición actual del robot (GPS)
    Usar LIDAR para actualizar obstáculos en la grilla
    if el LIDAR detecta obstáculo muy cerca then
        Activar flag_obstaculo_cercano
    end if
    Calcular ruta óptima con A* desde inicio hasta meta
    if flag_obstaculo_cercano = true then
        Girar en el lugar
    else if ruta válida existe then
        Avanzar hacia el siguiente punto del camino
    end if
    Enviar velocidades calculadas a los motores
end while
Finalizar ejecución del robot
```

Figura 1 Pseudocodigo del Robot Autónomo

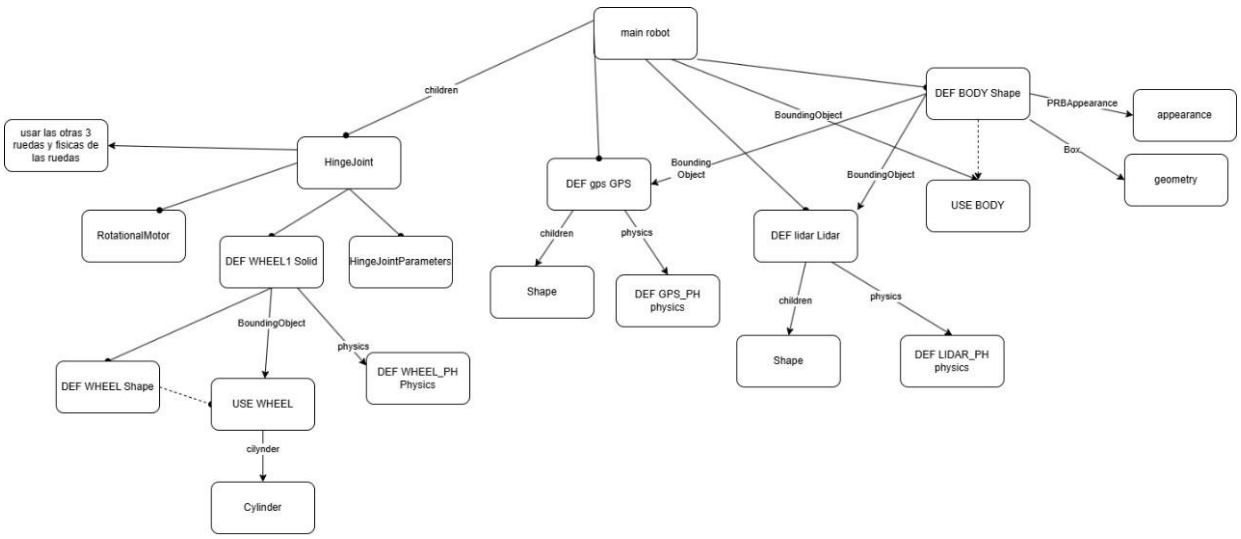

Figura 2 Diagrama del Robot

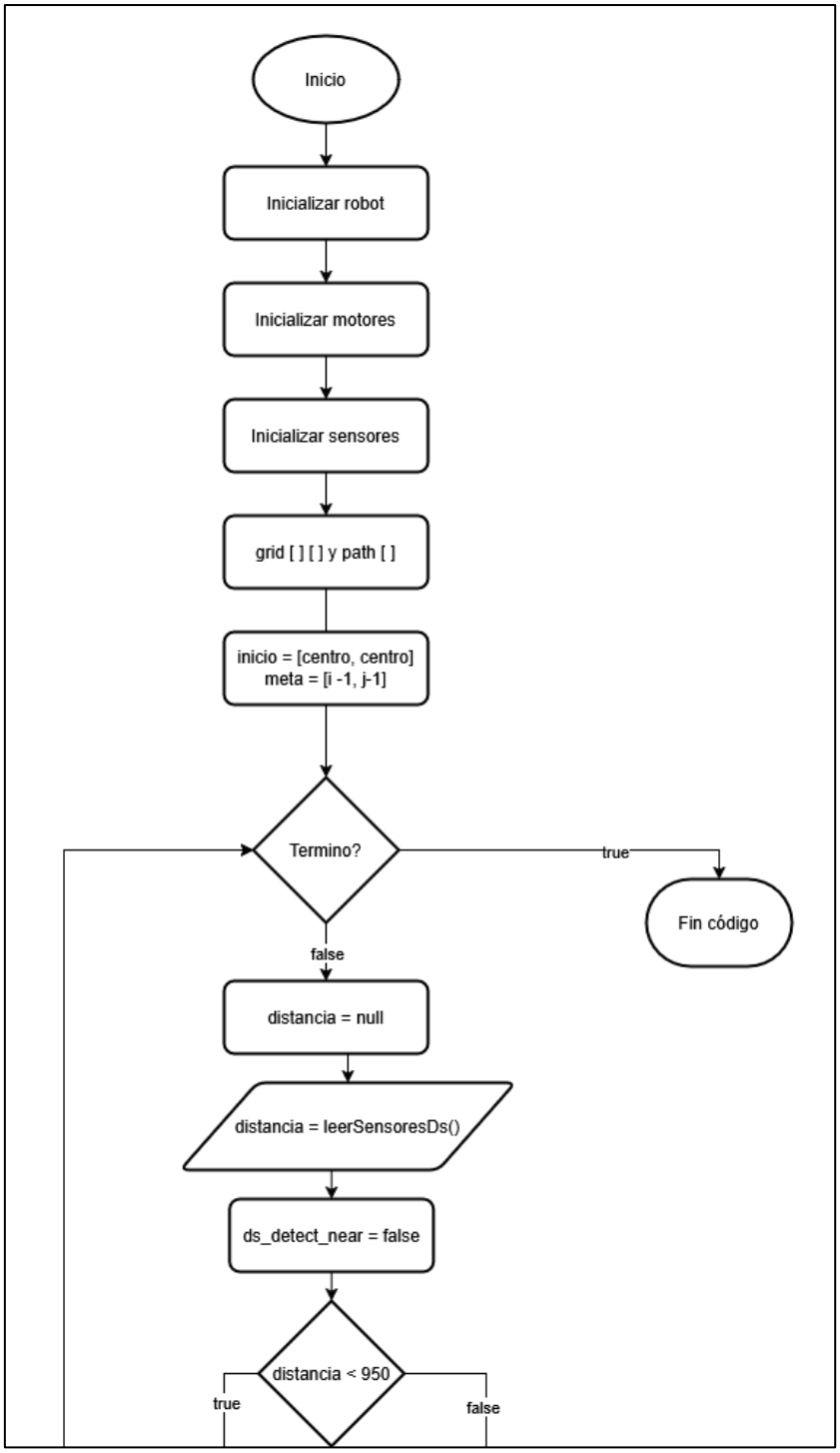

Figura 3 Diagrama de Acción del Robot Parte 1

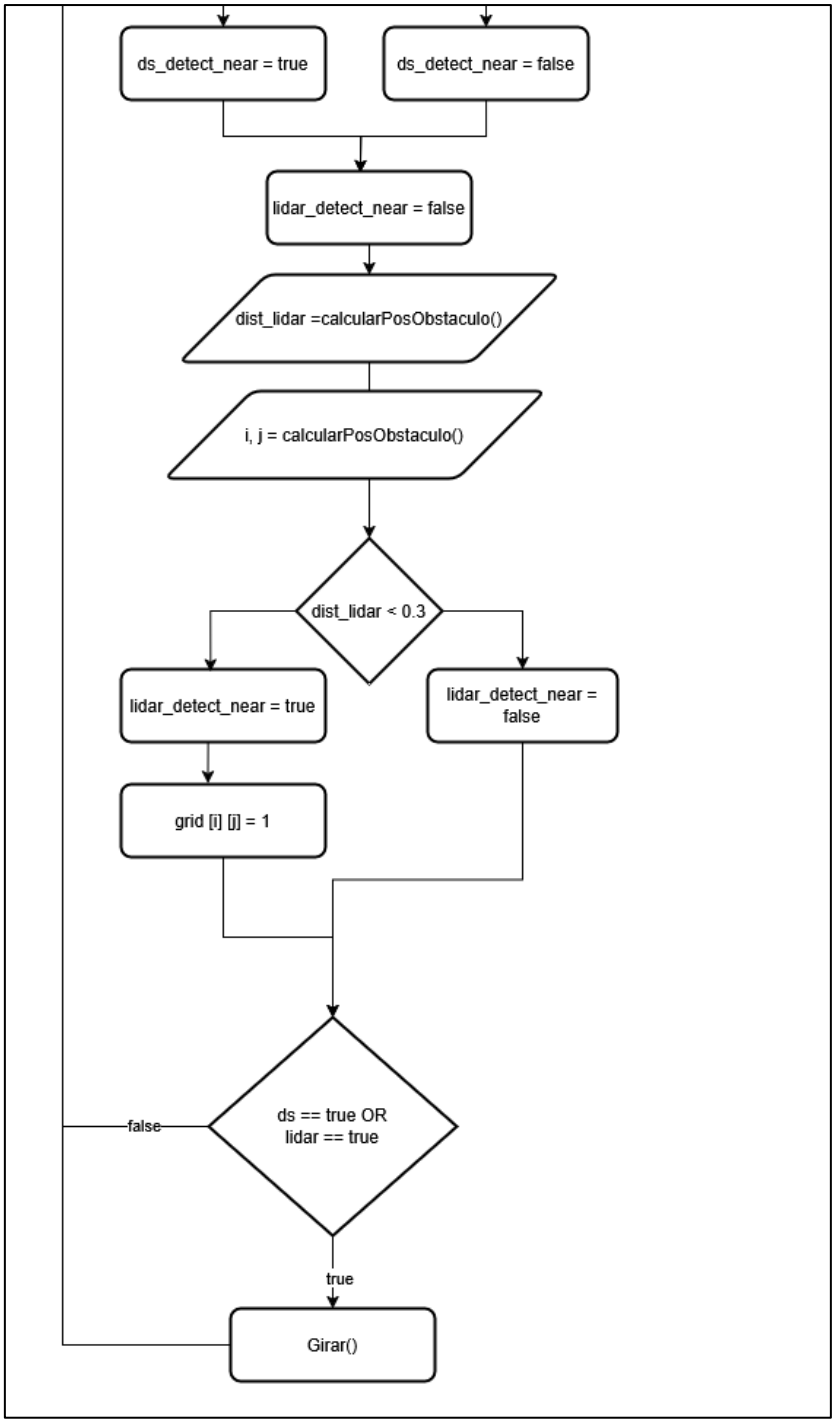

Figura 4 Diagrama de Acción del Robot Parte 2

2. Implementación del Robot en Webots

2.1. Código Funcional del Robot en Webots

El código principal que gobierna el comportamiento del robot está implementado en lenguaje C, dentro del archivo controller.c, ubicado en el directorio controllers del proyecto. Este archivo contiene toda la lógica necesaria para la operación autónoma del robot: desde la inicialización de los sensores y motores, hasta la ejecución de algoritmos de navegación, mapeo y evasión de obstáculos.

El controlador se estructura como un bucle principal que se ejecuta de forma cíclica cada 64 milisegundos (definido por TIME_STEP), lo que permite al robot responder en tiempo real a los estímulos del entorno simulado. Además, el código contiene funciones auxiliares para el cálculo del camino óptimo (planificación A*), procesamiento de datos del LIDAR y control de movimientos.

2.2. Configuración e Integración de Sensores

1. **LIDAR:** El sensor LIDAR fue activado utilizando wb_lidar_enable() y configurado para generar una nube de puntos mediante wb_lidar_enable_point_cloud(). Su campo de visión es de aproximadamente 2.78 radianes y su resolución puede alcanzar hasta 512 puntos por barrido. Esta información permite detectar obstáculos a diferentes distancias y ángulos, y es clave para el proceso de mapeo del entorno.

2. **GPS:** El sensor GPS entrega la posición absoluta del robot dentro del entorno simulado (coordenadas X y Z). Se activó con wb_gps_enable(), y sus lecturas permiten transformar los datos del LIDAR a coordenadas globales, facilitando su proyección sobre una grilla de ocupación.

3. **Sensores de distancia:** Se incorporaron dos sensores ultrasónicos (izquierdo y derecho) montados lateralmente. Fueron activados con wb_distance_sensor_enable() y su función principal es detectar obstáculos cercanos que el LIDAR pueda pasar por alto. Son esenciales para ejecutar maniobras de evasión rápida.

2.3. Algoritmos Implementados

2.3.1. Algoritmos Implementados para Navegación

El sistema de navegación del robot se basa en dos estrategias combinadas, la primera es el movimiento lineal por defecto, utilizado cuando no se detectan obstáculos inmediatos ni condiciones adversas. Y el segundo es la planificación mediante A*, en donde este algoritmo calcula la ruta óptima desde la posición actual del robot hasta un objetivo en la grilla (por ejemplo, una celda destino), evitando las zonas marcadas como ocupadas. El resultado es una secuencia de puntos que el robot sigue para alcanzar su meta.

2.3.2. Algoritmos Implementados para Detección de Obstáculos

El robot implementa una estrategia de evitación reactiva basada en la información de sus sensores, en donde, si alguno de los sensores de distancia detecta un obstáculo a menos de 950 unidades, o si el LIDAR detecta un punto a menos de 0.3 metros, se interpreta como una amenaza inminente. En respuesta, el robot interrumpe su avance y ejecuta una maniobra de giro sobre su eje, cambiando temporalmente su trayectoria hasta evitar la colisión.

Por lo que este comportamiento es rápido, autónomo y no depende de la planificación de ruta, permitiendo una reacción inmediata ante imprevistos.

2.3.3. Algoritmos Implementados para Mapeo

El proceso de mapeo se basa en la integración de datos del LIDAR y la posición GPS, en donde, cada punto de la nube del LIDAR es proyectado al espacio global mediante trigonometría, sumando la posición actual del robot. Por lo que, estas coordenadas globales se convierten en índices discretos sobre una grilla de tamaño 8×8, donde cada celda representa 0.5 m^2. Y las celdas correspondientes a puntos detectados se marcan con el valor 1, indicando la presencia de un obstáculo. Esta grilla de ocupación se actualiza dinámicamente en cada ciclo de simulación.

2.4. Pruebas de Escenarios Simulados

Se realizaron múltiples pruebas dentro de escenarios simulados en Webots, variando la distribución y cantidad de obstáculos en el entorno. Durante estas pruebas se observó lo siguiente:

- El robot fue capaz de generar un mapa coherente del entorno a partir de los datos del LIDAR y del GPS.
- En presencia de obstáculos cercanos, el robot reaccionó rápidamente gracias a los sensores de distancia y al LIDAR, evitando colisiones mediante giros.
- Cuando el camino directo al objetivo estaba bloqueado, el algoritmo A* fue capaz de encontrar rutas alternativas válidas.

Por lo que, estas pruebas confirmaron el correcto funcionamiento del sistema, validando la integración efectiva de sensores, actuadores y algoritmos de control. El robot mostró un comportamiento autónomo robusto y confiable dentro del entorno de simulación.

3. Resultados

3.1. Resultados Obtenidos

Como métricas de desempeño, se consideraron los siguientes parámetros:

- Tiempo total de navegación: Al robot le tomó 186 segundos completar el recorrido.
- Longitud del path (celdas): El robot recorrió un total de 48 celdas.
- Porcentaje del mapa explorado: Considerando la cantidad de celdas alcanzadas por el sensor del robot, entonces se exploró el 89,5 % del mapa.

3.2. Análisis de los Algoritmos Utilizados

El análisis se realizó respecto a tres aspectos importantes: precisión, eficiencia y robustez.

- **Precisión:** El algoritmo A* implementado permite calcular rutas óptimas en una grilla de $16x16$ celdas, desde una posición inicial hacia una meta fija. Se utiliza una heurística Manhattan, adecuada para entornos de grilla sin movimiento diagonal, lo que garantiza trayectorias eficientes y realistas para robots con restricciones direccionales. La precisión del mapa de obstáculos se mejora con datos LIDAR de alta resolución, que actualizan dinámicamente la grilla de navegación según la detección de obstáculos.

- **Eficiencia:** La ejecución del algoritmo A* está optimizada con listas abiertas y cerradas en arreglos estáticos, lo que permite una ejecución rápida dentro de los límites del simulador Webots. El uso de sensores (LIDAR, GPS, sensores de distancia) está habilitado a un intervalo de 64 ms (TIME_STEP), asegurando una buena frecuencia de actualización sin saturar el sistema.

- **Robustez:** El sistema combina múltiples sensores (LIDAR + sensores de distancia laterales + GPS), lo que permite una mejor detección de obstáculos, tanto estáticos como dinámicos. De esta forma, cuando se detecta un obstáculo cercano ($dist < 0.3$ por LIDAR o sensor < 950 en sensores laterales), se aplica una maniobra de evasión inmediata ($left\_speed = 1.0$; $right\_speed = -1.0$), lo que permite evitar colisiones sin depender exclusivamente de la planificación. El sistema, además, es capaz de reconstruir dinámicamente el mapa de obstáculos durante la ejecución, actualizando las celdas de la grilla en función de nuevas detecciones, aumentando la adaptabilidad a entornos cambiantes.

3.3. Reflexión sobre Mejoras y Optimización del Sistema

A pesar de que el sistema demostró un funcionamiento autónomo confiable en los escenarios simulados, es posible identificar diversas áreas donde se pueden aplicar mejoras significativas para optimizar su rendimiento, escalabilidad y capacidad de respuesta.

1. **Optimización de la planificación A*:** Actualmente, el algoritmo A* recalcula la ruta completa en cada ciclo de simulación, lo cual es innecesario si no hay cambios sustanciales en el entorno. Se recomienda implementar una lógica de replanificación condicional, activada sólo cuando se detectan nuevos obstáculos que invalidan el camino actual. Esto reduciría la carga computacional y mejoraría la eficiencia global del sistema.

2. **Mejora en la granularidad del mapeo:** La grilla utilizada puede ser insuficiente para entornos complejos. Aumentar la resolución de la grilla permitiría representar con mayor precisión obstáculos cercanos y zonas libres, mejorando la navegación y reduciendo errores en celdas mal clasificadas. Esto, no obstante, requeriría también optimizaciones adicionales para mantener el rendimiento.

3. **Incorporación de memoria histórica del entorno:** Actualmente, el mapa se actualiza en tiempo real, pero no se guarda una memoria persistente del entorno. Incorporar un sistema de memoria o un mapa global acumulativo permitiría al robot recordar ubicaciones previamente exploradas, evitando re-detecciones innecesarias y mejorando el comportamiento en misiones prolongadas o en exploración sistemática.

4. **Adaptación dinámica de velocidades:** El sistema actual mantiene una velocidad constante (por ejemplo, SPEED = 4.0) salvo en casos de evasión. Una mejora sería incorporar un controlador proporcional que adapte la velocidad del robot en función de la distancia a obstáculos, la curvatura del trayecto o la proximidad al objetivo, haciendo el movimiento más fluido y seguro.

5. **Expansión del modelo de evasión de obstáculos:** La evasión actual es reactiva y binaria (giro si detecta obstáculo). Este comportamiento puede ser mejorado mediante estrategias más avanzadas como campos potenciales, vectores de velocidad o el uso de curvas de Bézier para esquivar obstáculos sin detener completamente la navegación.

6. **Mejora del manejo de sensores redundantes:** El uso combinado del LIDAR y sensores de distancia ofrece redundancia, pero su integración es aún básica. Una fusión sensorial más sofisticada —por ejemplo, mediante filtros de Kalman o lógica

difusa— permitiría mejorar la confiabilidad en la detección y clasificación de obstáculos, especialmente en bordes o zonas parcialmente observadas.

7. **Modularización del código y reutilización:** El código principal implementa toda la lógica en un solo archivo. Modularizar las funciones de planificación, mapeo y evasión en archivos separados o estructuras de datos dedicadas mejoraría la legibilidad, mantenibilidad y permitiría la integración de nuevas funcionalidades de manera más ordenada.

8. **Pruebas más extensivas y métricas avanzadas:** El sistema fue evaluado en entornos simulados estáticos. Sería recomendable incorporar escenarios dinámicos, con obstáculos móviles, y métricas adicionales como el número de colisiones evitadas, tiempo promedio entre maniobras de evasión o consumo computacional por ciclo.

3.4. Lecciones Aprendidas y Posibles Extensiones del Proyecto

3.4.1. Importancia de la Modularidad en el Diseño de Software

Al inicio, la integración de todos los módulos (sensores, actuadores, planificación, mapeo, evasión) en un solo archivo controller.c demostró ser funcional, pero a medida que el proyecto crecía, la complejidad aumentaba. Aprendimos que una arquitectura de software más modular, con funciones y responsabilidades bien definidas, es crucial para la legibilidad, mantenibilidad y escalabilidad del código. Esto facilita la depuración y la introducción de nuevas características sin afectar al sistema completo.

- **Comprensión Profunda de los Sensores:** La precisión del mapeo y la efectividad de la evasión de obstáculos dependen directamente de la correcta configuración y el procesamiento de los datos de los sensores. Experimentar con el LIDAR para generar nubes de puntos y transformar coordenadas con el GPS nos enseñó la importancia de la calibración y la sincronización de los datos. Asimismo, la utilidad de los sensores de distancia como respaldo para detectar obstáculos cercanos que el LIDAR podría pasar por alto resaltó la importancia de la redundancia sensorial.

- **Desafíos y Oportunidades de los Algoritmos de Planificación:** El algoritmo A* demostró ser robusto para encontrar rutas óptimas en un entorno de grilla. Sin embargo, el costo computacional de recalcular la ruta completa en cada ciclo de simulación nos llevó a entender la necesidad de estrategias de planificación

condicional. Esta lección subraya el equilibrio entre la precisión del camino y la eficiencia del sistema en tiempo real.

- **Balance entre Comportamiento Reactivo y Deliberativo:** La combinación de la evasión reactiva (respuesta rápida a obstáculos inmediatos) y la planificación deliberativa (A* para la ruta a largo plazo) fue fundamental para el desempeño del robot. Aprendimos que, en entornos dinámicos, un sistema puramente deliberativo podría ser demasiado lento, mientras que uno puramente reactivo podría quedar atrapado en ciclos. La sinergia de ambos enfoques es clave para un comportamiento autónomo efectivo.

- **Validación a Través de la Simulación:** Webots resultó ser una herramienta invaluable para el diseño, implementación y prueba del robot. La capacidad de simular diferentes escenarios y visualizar el comportamiento del robot en un entorno controlado nos permitió identificar y corregir errores de manera eficiente antes de una posible implementación en hardware real. Esto aceleró el ciclo de desarrollo y permitió una iteración rápida.

3.4.2. Posibles Extensiones del Proyecto

Este proyecto abre múltiples líneas de desarrollo futuro que permitirían mejorar significativamente la autonomía, inteligencia y adaptabilidad del robot. Una primera extensión sería implementar mapeo 3D y técnicas de SLAM, reemplazando la grilla 2D por una representación tridimensional del entorno basada en la nube de puntos del LIDAR, lo que permitiría operar en espacios con obstáculos a diferentes alturas y prescindir del GPS para la localización. Asimismo, mejorar la navegación en entornos dinámicos incorporando predicción de movimiento mediante filtros de Kalman o redes neuronales permitiría evitar obstáculos móviles como personas u otros robots de forma anticipada. Otra línea de mejora relevante es el desarrollo de una interfaz de interacción humano-robot que facilite el control remoto, la supervisión en tiempo real y la toma de decisiones en casos de emergencia. También se propone optimizar la planificación de rutas considerando no solo la distancia, sino también el consumo energético y la robustez frente a errores sensoriales, incrementando así la autonomía y confiabilidad en misiones prolongadas. Finalmente, explorar técnicas de aprendizaje por refuerzo permitiría que el robot aprenda estrategias de navegación eficientes y adaptativas mediante la experiencia directa, mejorando su desempeño en entornos complejos sin necesidad de programación explícita.