PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Informe Proyecto
Robótica y Sistemas Autónomos

Integrantes:

- **Claudio Cabello V.**
- **David Martínez C.**
- **Eduardo Pérez M.**

Robótica y Sistemas Autónomos
Junio, 2025

Índice

1. Introducción

El avance de la robótica autónoma ha abierto múltiples posibilidades en áreas como la logística, exploración espacial, agricultura de precisión y asistencia en desastres. En este contexto, el desarrollo de robots móviles capaces de desplazarse de forma autónoma en entornos desconocidos representa un desafío técnico y formativo altamente relevante para los futuros ingenieros en informática y robótica.

Este proyecto tiene como propósito central diseñar e implementar un robot móvil autónomo en el entorno de simulación Webots, que sea capaz de percibir su entorno utilizando sensores, generar un mapa básico a partir de los datos percibidos, evitar obstáculos en tiempo real y planificar rutas para alcanzar objetivos específicos dentro de un entorno simulado. Para ello, se utilizan sensores como el LIDAR, sensores de distancia y GPS, junto con algoritmos clásicos como el de A* para planificación de rutas y mapeo basado en celdas ocupadas.

Más allá del desarrollo técnico, este proyecto busca fomentar un aprendizaje práctico y profundo en múltiples áreas clave de la robótica autónoma, entre ellas:

- **Percepción y sensado**: Comprender cómo los sensores interpretan el entorno y cómo sus datos deben ser procesados para construir una representación útil.
- **Mapeo y localización**: Generar modelos del entorno a partir de datos incompletos o ruidosos.
- **Planificación y toma de decisiones**: Aplicar algoritmos de búsqueda heurística (como A*) para encontrar caminos óptimos hacia un objetivo.
- **Control de movimiento**: Traducir una planificación abstracta en acciones concretas que permitan al robot desplazarse de forma fluida y eficiente.
- **Simulación e integración de sistemas**: Adquirir habilidades en entornos de simulación como Webots, fundamentales en la etapa de prototipado antes de implementaciones reales.

Este trabajo también fortalece competencias transversales como el trabajo en equipo, la gestión de versiones mediante repositorios Git y la documentación profesional de soluciones técnicas.

En resumen, este proyecto no solo representa un reto de implementación técnica, sino también una oportunidad de formación integral en los fundamentos de los sistemas autónomos modernos, sentando una base sólida para aplicaciones más complejas en el futuro profesional de cada integrante del equipo.

2. Objetivos

2.1. Objetivo General

Desarrollar un robot móvil autónomo en Webots capaz de navegar de forma segura y eficiente en un entorno simulado, utilizando técnicas de percepción, mapeo, planificación y control para evitar obstáculos y alcanzar metas definidas.

2.2. Objetivos Específicos

- **Implementar un sistema de percepción sensorial** utilizando LIDAR, sensores de distancia y GPS, que permita al robot interpretar su entorno para detectar obstáculos y ubicarse espacialmente.

- **Desarrollar un algoritmo de planificación de rutas** mediante el uso de A*, sobre un mapa básico generado en tiempo real, que permita al robot calcular trayectorias óptimas hacia un destino predeterminado.

- **Integrar un controlador autónomo de movimiento** capaz de seguir la ruta planificada y evitar colisiones, combinando información sensorial con decisiones en tiempo real.

3. Diseño del Proyecto

3.1. Descripción de Robot Móvil

El robot móvil simulado dispone de:

- **Motores diferenciales**: Cuatro ruedas con control independiente de velocidad para permitir movimiento hacia adelante y giros.

- **Sensores de distancia (ultrasonidos o infrarrojos)**: Dos sensores ubicados a izquierda y derecha para detección cercana de obstáculos.

- **Sensor LIDAR**: Permite obtener un mapa 360° del entorno a una resolución determinada, detectando obstáculos a distancias mayores.

- **GPS**: Para obtener la posición actual del robot en el entorno simulado, usando el plano X-Z.

El robot se mueve en un entorno simulado donde debe percibir obstáculos estáticos, construir un mapa simple y planificar rutas para desplazarse hacia un objetivo.

3.2. Explicación Webots

Webots es un simulador de robots que permite diseñar entornos 3D y programar el comportamiento de robots utilizando APIs en C, C++ o Python. En este proyecto, Webots se utilizó para:

- Simular el entorno con obstáculos y el robot móvil equipado con sensores reales.
- Proveer interfaces para acceder a datos de sensores (LIDAR, distancia, GPS) y controlar motores.
- Ejecutar el ciclo de simulación con un paso temporal definido (64 ms).
- Visualizar el comportamiento del robot y verificar su capacidad de navegación autónoma.

La integración con Webots facilita la prueba y validación del robot sin necesidad de hardware físico, permitiendo iterar en el diseño y los algoritmos antes de una posible implementación real.

3.3. Arquitectura de Software

El software del robot se estructura en módulos funcionales, cada uno con una responsabilidad específica dentro del ciclo de navegación autónoma:

Percepción:

- Procesa los datos crudos entregados por los sensores.
- Extrae información útil como obstáculos detectados y posición actual.
- Alimenta el sistema de mapeo con datos del entorno.

Mapeo y planificación:

- Construye un mapa básico mediante grilla de ocupación.
- Aplica el algoritmo A* para generar una ruta desde la posición actual al objetivo, evitando zonas bloqueadas.

Control de movimiento:

- Convierte la ruta planificada en comandos de velocidad diferencial.
- Ajusta la trayectoria en tiempo real frente a desviaciones o detección de obstáculos cercanos.

Motor de simulación (Webots):

- Coordina el ciclo de ejecución a intervalos regulares.
- Permite la interacción entre sensores, actuadores y lógica del robot.

3.4. Algoritmos a utilizar

➢ **Mapeo del entorno**: El robot crea un mapa simple en forma de cuadrícula, marcando qué lugares están libres y cuáles tienen obstáculos, usando la información del sensor LIDAR.

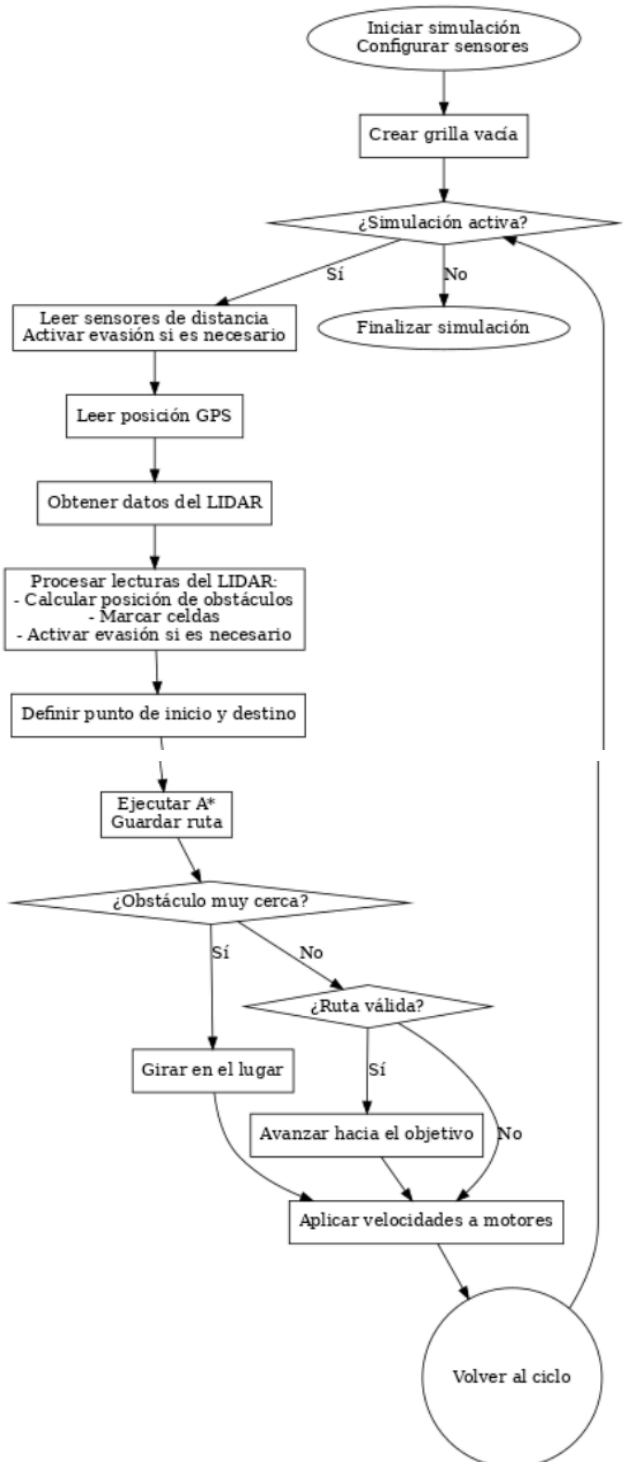
➢ **Planificación de rutas (A*)**: Se usa un algoritmo que busca el mejor camino para llegar al objetivo, evitando las zonas marcadas como ocupadas en el mapa.

➢ **Evitación de obstáculos cercanos**: Si el robot detecta un objeto muy cerca con sus sensores laterales, gira o se mueve para evitar chocar, incluso si no estaba en el mapa.

➢ **Control de movimiento**: El robot calcula hacia dónde debe girar y qué tan rápido avanzar para seguir el camino correcto hasta la meta.

3.5. Diagramas de flujo y pseudocódigo de la solución

Diagrama de flujo:



Pseudocódigo:

```
Iniciar simulación y configurar sensores, LIDAR, GPS y motores
Crear grilla vacía para el mapeo del entorno

Mientras la simulación esté activa:
    Leer sensores de distancia
    Activar evasión si hay obstáculo cercano

    Leer posición actual desde GPS
    Obtener datos del LIDAR

    Para cada lectura del LIDAR:
        Calcular posición del obstáculo
        Convertir a celda de grilla y marcar como ocupada si aplica
        Si está muy cerca, activar evasión

    Definir punto de inicio y destino en la grilla
    Ejecutar A* para planificar ruta
    Guardar camino en lista de puntos

    Si hay obstáculo muy cercano:
        Girar en el lugar
    Sino si hay ruta válida:
        Avanzar hacia el objetivo

    Aplicar velocidades a motores

Finalizar simulación
```

3.6. Resultados esperados

Se espera que el robot móvil sea capaz de cumplir correctamente su tarea principal: navegar de forma autónoma hacia un objetivo definido, evitando colisiones y ajustando su trayectoria en función de los obstáculos del entorno.
En particular, se espera que el sistema:

- Genere un mapa básico que represente correctamente las zonas ocupadas y libres.
- Calcule rutas eficientes y sin bloqueos hacia el destino.
- Reaccione adecuadamente ante obstáculos cercanos no previstos.
- Mantenga un movimiento fluido y estable en diferentes escenarios simulados.

Estos resultados confirman que los módulos de percepción, planificación y control están correctamente integrados y cumplen su función dentro del entorno simulado.

3.7. Análisis de los algoritmos propuestos

- **Mapeo por celdas ocupadas**: Es una técnica sencilla pero efectiva para representar entornos estáticos. Permite al robot construir una visión básica del espacio a partir del LIDAR, sin necesidad de algoritmos complejos como SLAM.

- **Planificación con A***: A* es un algoritmo clásico y confiable para encontrar caminos óptimos. Su uso es adecuado en entornos discretizados como una grilla, y garantiza que se encuentre un camino (si existe) de forma eficiente.

- **Evitación de obstáculos reactiva**: Esta estrategia permite responder rápidamente ante obstáculos muy cercanos o imprevistos, complementando el planificador global con maniobras simples y efectivas.

- **Control de movimiento proporcional**: Este tipo de control es fácil de implementar y suficiente para guiar al robot por rutas suaves. Si bien puede presentar desvíos menores, se espera que logre mantener el rumbo hacia los objetivos planificados.

4. Implementación

4.1. Código funcional del robot en Webots

El código funcional del robot en Webots implementa diversas funcionalidades que abarcan desde la inicialización de dispositivos hasta la navegación autónoma. A continuación, se detallan las principales funcionalidades integradas en el código:

1. Inicializar el robot y sus dispositivos:
    - Motores (ruedas)
    - Sensores de distancia (izquierdo y derecho)
    - LIDAR (escáner láser)
    - GPS
2. Construir un mapa del entorno usando el LIDAR:
    - Detecta obstáculos cercanos.
    - Marca en una grilla cuáles celdas están ocupadas.
3. Obtener la posición del robot usando el GPS.
4. Detectar obstáculos cercanos con sensores de distancia y LIDAR.
5. Planificar una ruta en una grilla desde el punto actual hasta un objetivo usando el algoritmo A*.
6. Decidir cómo moverse:
    - Si hay obstáculos cerca, gira para evitarlos.
    - Si hay una ruta disponible, avanza hacia el objetivo.
7. Controlar la velocidad de los motores para moverse o girar.
8. Se repite constantemente en el bucle principal mientras el robot está activo.

4.2. Configuración e integración de sensores

Los sensores se utilizaron para permitir que el robot percibiera su entorno y navegue de forma autónoma. Los sensores de distancia detectan obstáculos cercanos para evitar colisiones inmediatas, el LIDAR proporciona un escaneo detallado del entorno que permite construir un mapa y planificar rutas evitando obstáculos, y el GPS se encarga de ubicar al robot dentro del espacio simulado para guiar su desplazamiento hacia un objetivo.

A continuación se especifica configuración y uso de los sensores:

1. **Sensores de distancia (izquierdo y derecho)**: Se configuraron activándolos con un intervalo de tiempo fijo. Se utilizan para detectar obstáculos cercanos y activar maniobras de evasión básicas cuando el robot se aproxima demasiado a un objeto.

2. **Sensor LIDAR**: Se activó para obtener imágenes de rango y nube de puntos. Se utiliza para detectar obstáculos en un área más amplia y construir un mapa del entorno en forma de grilla (celdas ocupadas o libres), lo que permite la planificación de rutas.

3. **Sensor GPS**: Se configuró para enviar la posición del robot a intervalos regulares. Se usa para ubicar al robot dentro del entorno simulado y relacionar su posición con las celdas del mapa.

4.3. Algoritmos implementados para navegación, detección de obstáculos y mapeo

Los algoritmos de navegación, detección de obstáculos y mapeo permitieron que el robot se desplazará de forma autónoma y segura dentro de su entorno. El mapeo convierte los datos del LIDAR en una representación del espacio en forma de grilla, marcando las zonas ocupadas por obstáculos. Esta información es fundamental para el algoritmo de planificación de rutas (A*), que calcula el camino más eficiente hacia un objetivo evitando esas zonas.

Lo mencionado anteriormente se detalla a continuación:

1. Algoritmo de planificación de rutas A*: Para la navegación se utiliza el algoritmo A* para encontrar la ruta más corta desde la posición actual del robot hasta un punto objetivo dentro de una grilla. A* considera tanto el costo de movimiento como una heurística (distancia Manhattan). El resultado es una secuencia de puntos que el robot puede seguir para llegar a su destino evitando obstáculos.

2. Detección de obstáculos – Sensores de distancia y LIDAR: La detección de obstáculos se realiza combinando dos fuentes:

- Sensores de distancia: activan una respuesta inmediata (giro) si un objeto está demasiado cerca.
- LIDAR: escanea el entorno y detecta obstáculos en un rango amplio. Si el obstáculo está a menos de 0.3 metros, también se activa una maniobra evasiva. Esto permite al robot reaccionar ante obstáculos tanto cercanos como a distancia.

3. Mapeo del entorno: A partir de los datos del LIDAR, el robot convierte las distancias y ángulos en coordenadas cartesianas. Estas coordenadas se traducen en celdas dentro de una grilla 2D que representa el mapa del entorno. Las celdas donde hay obstáculos se marcan como ocupadas (valor 1), permitiendo al algoritmo A* planificar rutas solo a través de zonas libres.

4.4. Pruebas en escenarios simulados

Para verificar el correcto funcionamiento del sistema desarrollado, se realizaron pruebas en un entorno simulado utilizando Webots. El escenario de prueba consistía en un entorno estático con obstáculos distribuidos, en el cual el robot debía desplazarse desde su posición inicial hasta un objetivo, evitando colisiones y adaptándose en tiempo real a su entorno percibido.

Durante la simulación, se pudo observar lo siguiente:

- **Percepción correcta del entorno:** El robot escanea continuamente su entorno usando el sensor LIDAR, generando un mapa simple en forma de grilla donde se marcaron las celdas ocupadas por obstáculos.

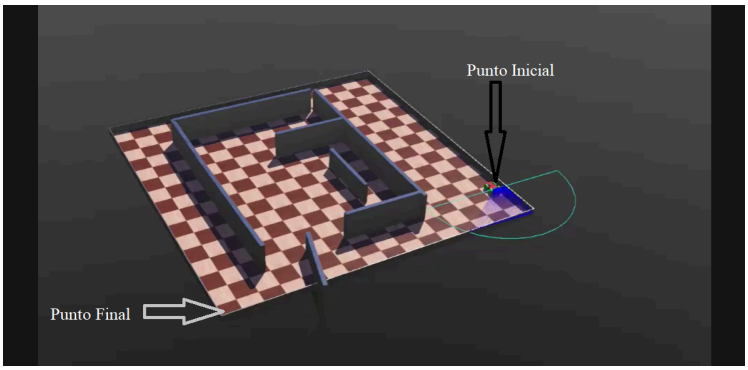
- **Evitación de obstáculos eficaz:** Gracias a los sensores de distancia y la lectura directa del LIDAR, el robot fue capaz de reaccionar ante obstáculos cercanos mediante maniobras de giro, evitando colisiones incluso cuando estos no estaban previamente mapeados.

- **Planificación de ruta funcional:** El algoritmo A* se ejecuta periódicamente para calcular una nueva ruta desde la posición actual del robot hasta el objetivo. El camino fue actualizado dinámicamente en base al entorno detectado.

- **Desplazamiento fluido:** El robot avanzó siguiendo la ruta planificada, ajustando su trayectoria cuando fue necesario. Al alcanzar el punto objetivo, el robot se detuvo automáticamente.

- **Prueba visual satisfactoria:** El video de la simulación (duración de 62 segundos) muestra una ejecución completa y estable del sistema, en la cual se validan todas las capacidades planificadas: percepción, mapeo, planificación y control autónomo.

Estas pruebas confirman que los distintos módulos del sistema están correctamente integrados y que el robot es capaz de navegar de forma autónoma y segura dentro del entorno simulado. Esto valida el enfoque híbrido utilizado, que combina planificación deliberativa (A*) con evasión reactiva ante obstáculos.



5. Resultados

5.1. Resultados obtenidos

Tiempo total de navegación: 10 segundos
Longitud del path (celdas): 24 celdas
Tiempo de planificación (A*): 250 milisegundos
Porcentaje del mapa explorado: 9,38% (24 de 256 en total).

5.2. Análisis de los algoritmos utilizados

A modo de análisis se presentan ventajas y desventajas de los algoritmos utilizados junto al tipo de algoritmo:

Técnica	Ventajas	Desventajas	Tipo
1. Algoritmo A*	Encuentra rutas óptimas si la heurística está bien diseñada. Eficiente en grillas. Permite planificar antes de moverse.	Puede ser lento en mapas grandes con muchos obstáculos. No responde bien a cambios dinámicos. Requiere mapa preciso.	Planificación de rutas
2. Grilla de ocupación	Implementación simple. Compatible con A*. Buen rendimiento en espacios pequeños.	Depende del tamaño de la celda. No almacena detalles. Alto consumo de memoria en entornos grandes.	Mapeo
3. Estrategia reactiva (sensores)	Respuesta rápida ante obstáculos. Fácil de implementar. Eficaz para evitar colisiones.	No tiene visión global. No distingue entre tipos de obstáculos.	Comportamiento reactivo

En el código combina un enfoque planificado refiriéndose algoritmo A* + mapeo con LIDAR con una estrategia reactiva . Esta combinación permite al robot navegar de forma inteligente, pero también adaptarse a situaciones imprevistas, logrando una navegación robusta y flexible en entornos parcialmente conocidos

5.3. Reflexión sobre mejoras y optimización del sistema

Durante el desarrollo del sistema de navegación autónoma, se logró implementar una solución funcional que combina planificación con A*, mapeo con LIDAR y estrategias reactivas para la evasión de obstáculos. No obstante, el proceso permitió identificar diversas oportunidades de mejora que podrían optimizar significativamente el rendimiento y la robustez del sistema.

Una de las principales áreas de mejora es la frecuencia y eficiencia del proceso de replanteamiento de ruta. Actualmente, cualquier cambio detectado en el entorno genera un nuevo cálculo de la trayectoria completa. Si bien esto garantiza que el robot siempre siga un camino actualizado, también puede generar sobrecarga computacional o movimientos innecesarios. Una optimización podría consistir en aplicar un replanteamiento parcial o condicional, solo cuando el nuevo obstáculo afecta directamente la ruta actual.

Otra mejora importante sería implementar un algoritmo de mapeo y SLAM, que permitiría al robot construir y actualizar dinámicamente un mapa del entorno mientras se localiza con mayor precisión.

En cuanto a la planificación, optimizar el uso del algoritmo A* mediante estructuras de datos más eficientes como colas de prioridad puede reducir el tiempo de búsqueda, especialmente en mapas más grandes o complejos. También se podría integrar planificación en tiempo real o replanificación continua para adaptarse a cambios repentinos en el entorno.

5.4. Lecciones aprendidas y posibles extensiones del proyecto

A lo largo del desarrollo del proyecto, se reforzaron conocimientos clave sobre sensores, planificación de rutas, mapeo y control de movimiento en entornos simulados. La implementación práctica permitió comprender la importancia de integrar distintos módulos de forma coherente y de manejar datos ruidosos en tiempo real.

También se evidenció el valor del trabajo iterativo, ya que fue necesario ajustar parámetros, corregir errores y validar el comportamiento del robot en múltiples escenarios.

Como extensiones futuras, se podría:

- Incorporar un sistema de mapeo probabilístico (como occupancy grids).
- Usar SLAM para mapeo y localización simultánea.
- Integrar visión por computadora o redes neuronales para detección de objetos.
- Implementar un control PID para mejorar la precisión del movimiento.

Estas mejoras permitirían escalar el sistema hacia entornos más complejos y dinámicos.

6. Conclusión

Este proyecto permitió diseñar y programar un robot móvil autónomo en un entorno simulado utilizando Webots. A lo largo del desarrollo, se integraron sensores, algoritmos de planificación y técnicas de control para que el robot fuera capaz de detectar obstáculos, construir un mapa básico y moverse de forma segura hacia un objetivo.

Gracias a la combinación del sensor LIDAR, el GPS y los sensores de distancia, el robot pudo percibir su entorno con suficiente precisión. El uso del algoritmo A* ayudó a calcular rutas eficientes dentro de una grilla del entorno, mientras que el comportamiento reactivo frente a obstáculos cercanos mejoró su capacidad de adaptación ante situaciones inesperadas.

En resumen, se logró construir un sistema que mezcla planificación y reacción, lo cual es clave en robótica autónoma. Además de aprender a trabajar con sensores y algoritmos, este proyecto también fortaleció habilidades prácticas como la programación estructurada, la simulación de robots y el trabajo colaborativo. Todo esto sienta una base sólida para futuros desafíos en el desarrollo de sistemas inteligentes y autónomos.

7. Referencias

- Cano, S. (2025). Proyecto Final – Robótica y Sistemas Autónomos. GitHub. https://github.com/sancano22/robotica-autonomos/tree/main/ProyectoFinal

- Pontificia Universidad Católica de Valparaíso. (2025). Curso Robótica y Sistemas Autónomos (ICI4150-1, Semestre 1). Escuela de Ingeniería Informática.

- Pérez Miranda, E., Cabello Vásquez, C., & Martínez Canto, D. (2025). Proyecto_Robotica. GitHub. https://github.com/EduardoPerezMir/Proyecto_Robotica

- Cyberbotics Ltd. (2024). Webots User Guide. https://cyberbotics.com/doc/guide/index

- Russell, S., & Norvig, P. (2021). Artificial intelligence: A modern approach (4th ed.). Pearson.

- Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. MIT Press.

- LaValle, S. M. (2006). Planning algorithms. Cambridge University Press. https://planning.cs.uiuc.edu/