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

Proyecto Webots

Alex Alfaro
Manuel Aguilera
Eliseo Guarda
Pedro Nordenflycht
Vicente Rosales

Asignatura: Ingeniería de Software
Profesora: Sandra Cano
Ciudad: Valparaíso

Índice

1 Descripción del robot móvil y sus características

El robot desarrollado para este proyecto es un vehículo móvil autónomo de tipo terrestre, diseñado con una configuración de **cuatro ruedas motrices independientes**, controladas por motores de corriente continua (DC) conectados directamente a cada rueda. Esta arquitectura le proporciona un mayor maniobrabilidad y tracción, ideal para desplazarse en entornos de simulación con obstáculos o superficies variadas.

Características físicas y funcionales

Entre las características físicas y funcionales más importantes del robot tenemos:

- Tipo de locomoción:
    - Robot con tracción en las cuatro ruedas (**4WD**)
- Actuadores:
    - **4 motores rotacionales DC**, uno para cada rueda, que permiten un control un control fino de velocidad y dirección.
- Sensores incorporados
    - **LIDAR 360°**: Sensor de escaneo láser de alta resolución que permite la detección de obstáculos en un radio completo alrededor del robot. Ideal para mapeo y evasión de obstáculos en tiempo real.
    - **GPS**: Sensor de posicionamiento global que proporciona las coordenadas absolutas (X,Y, Z) dentro del entorno simulado, útil para localización y navegación global.
    - **Brújula (compass)**: Sensor de orientación que mide el ángulo respecto al norte magnético. Se utiliza para determinar la orientación (θ) del robot en el plano, y es clave para calcular trayectorias o giros.

El robot es capaz de:

- **Detectar obstáculos en tiempo real** mediante un sensor **LIDAR 360°**, permitiendo identificar objetos en su entorno inmediato.
- **Construir un mapa de ocupación** a partir de los datos entregados por el LIDAR, representando el entorno en una grilla de **200 x 200 celdas**, donde cada celda refleja la presencia o ausencia de obstáculos.
- **Localizar su posición y orientación** en el entorno simulado a través de sensores de **GPS** y **brújula**, información que se integra al proceso de mapeo

El robot no realiza ruteo ni planificación de trayectorias, sino que recorre el entorno mapeando progresivamente los obstáculos detectados.

2 Explicación del entorno simulado en Webots

Para la validación y prueba del comportamiento del robot móvil, se diseñó un entorno simulado personalizado de webots, el cual permite replicar escenarios controlados con condiciones ajustables para realizar experimentos de navegación autónoma, detección de obstáculos y planificación de rutas.

Características del entorno:

- **Dimensiones:**
  - El escenario cuenta con un tamaño de 2 metros por 2 metros, proporcionando un espacio adecuado para la navegación y prueba de algoritmos de evasión y planificación en un entorno reducido pero suficientemente complejo
- **Tipo de obstáculos:**
  - Se incorporaron **cajas y barriles** distribuidos en distintas posiciones dentro del entorno, simulando obstáculos de distintas formas y tamaños. esto permite evaluar la capacidad del robot para detectar objetos y ajustar su trayectoria en consecuencia
- **Objetivo de la navegación**:
  - El objetivo principal de la simulación es que el robot se desplace desde un **punto de partida** hacia una **meta predefinida**, evitando colisiones con los obstáculos dispuestos en el entorno.
    - **Punto de partida:** identificado de color verde
    - **Punto de llegada:** identificado de color amarillo
- **Entorno personalizado:**
  - El escenario fue construido manualmente usando el editor de entornos de Webots, lo que permitió definir de forma precisa la ubicación de cada obstáculo, el punto de inicio y la posición objetivo (como se ve en la Figura 1). Esto asegura la replicabilidad de las pruebas y el control de las variables de entorno para evaluar distintas configuraciones del algoritmo de navegación.

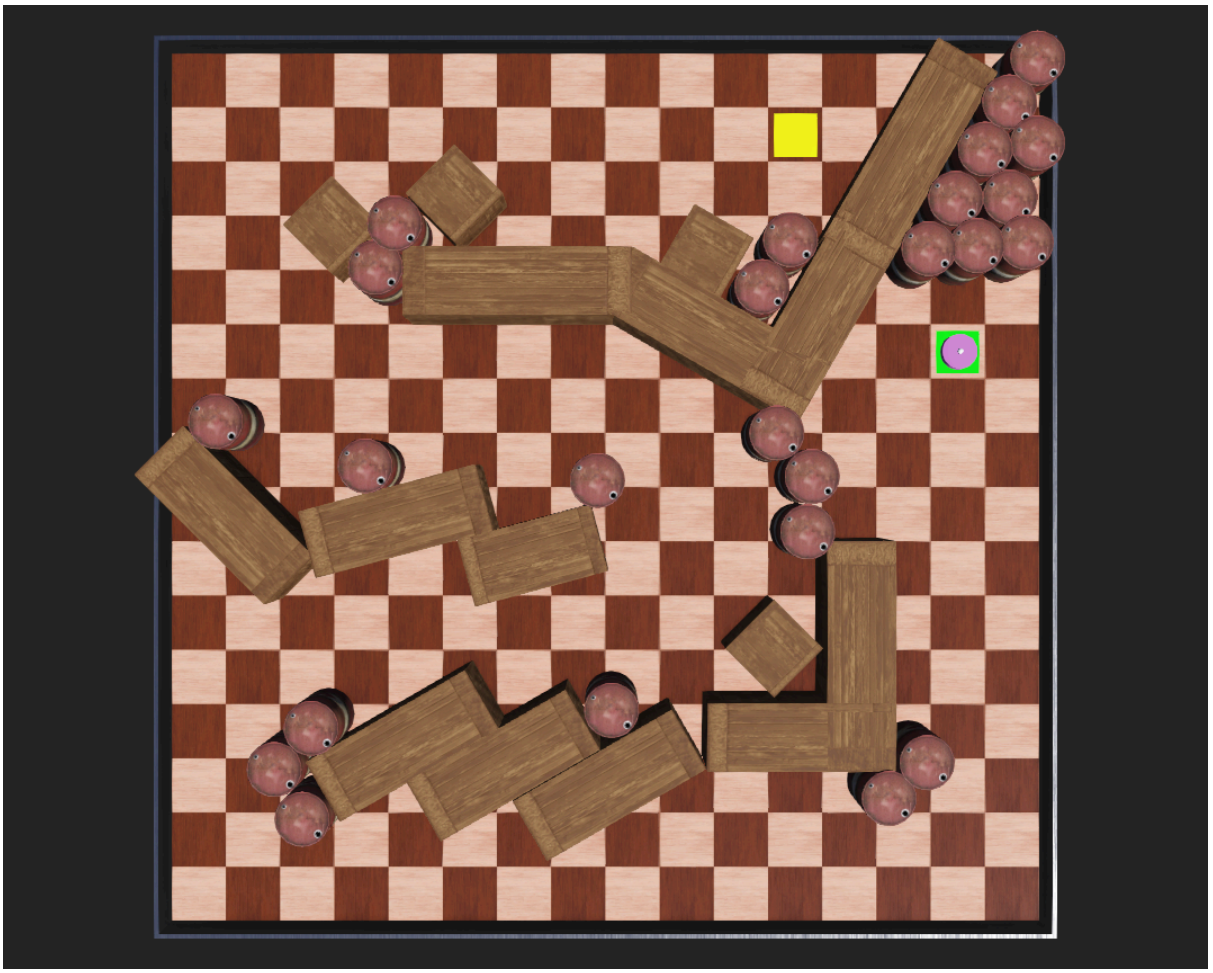


Figura 1: Entorno simulado en webots con inicio y meta.

Propósito de este entorno:
Este entorno compacto y controlado permite:

- Validar el correcto funcionamiento del sistema de percepción mediante LIDAR y sensores complementarios
- Evaluar la eficacia de los algoritmos de evasión de obstáculos y planificación de rutas.
- Medir métricas de desempeño como el tiempo de navegación, longitud del recorrido y porcentaje de mapa explorado.

3 Arquitectura del software: sensores, actuadores y módulos de control

El sistema de control del robot móvil se estructura en tres componentes funcionales principales: percepción, planificación y control, los cuales trabajan de manera coordinada para permitir la navegación autónoma dentro del entorno simulado.

Percepción
La capa de percepción es responsable de obtener información en tiempo real del entorno circundante y del estado del propio robot, a partir de los sensores disponibles:

- **Sensor LIDAR 360**:
    - Realiza un escaneo completo del entorno en cada ciclo de simulación
    - Detecta obstáculos en un rango comprendido entre **0,05 cm y 0,5 cm**
    - Los datos obtenidos se utilizan para construir dinamica
    - mente un **mapa de ocupación bidimensional de 200 x 200 celdas**, donde cada celda representa **0,1cm²**
    - Las celdas se marcan como ocupadas o libres según la detección de obstáculos, lo que permite planificar rutas seguras.
- **GPS**:
    - Proporciona la posición actual del robot en coordenadas absolutas dentro del entorno simulado
    - Es esencial para la localización global y definir la posición relativa respecto al objetivo
- **Brújula**:
    - Mide la **orientación angular del robot en radianes** respecto al norte magnético simulado.
    - Se utiliza para orientar correctamente al robot durante las fases de navegación y para alinear la dirección de movimiento hacia los puntos intermedios de la trayectoria planificada.

Planificación:
La capa de planificación se encarga de determinar la trayectoria óptima desde la posición actual hasta la meta, evitando las zonas marcadas como ocupadas en el mapa de ocupación.

- **Grilla de ocupación**:
    - Representa el entorno como una matriz de celdas cuadradas, cada una etiquetada como libre u ocupada en función de los datos entregados por el LIDAR.
- **Algoritmo de planificación Dijkstra**:
    - Se implementa el algoritmo Dijkstra, un método clásico de búsqueda de caminos mínimos en grafos ponderados sin heurística.
    - Permite encontrar el camino más corto desde el nodo inicial (posición actual) hasta el objetivo final, considerando las restricciones impuestas por los obstáculos detectados en el mapa.
    - Se ejecuta cada vez que se actualiza el mapa de ocupación, garantizando una planificación reactiva frente a los cambios dinámicos del entorno.

Control

La capa de control se encarga de traducir las decisiones de planificación en acciones físicas sobre los actuadores del robot

- **Controlador de motores**
    - Se utiliza un **control cinemático diferencial**, donde las velocidades de las ruedas izquierda y derecha se ajustan de manera independiente para definir la dirección y velocidad de avance.
    - El robot ajusta su movimiento en función de la orientación actual (obtenida desde la brújula) y el próximo punto de la trayectoria planificada.

Este modelo de arquitectura modular permite una integración eficiente de sensores y algoritmos, favoreciendo una navegación autónoma precisa y adaptable

4 Algoritmos a utilizar (evitación de obstáculos, mapeo, planificación de rutas)

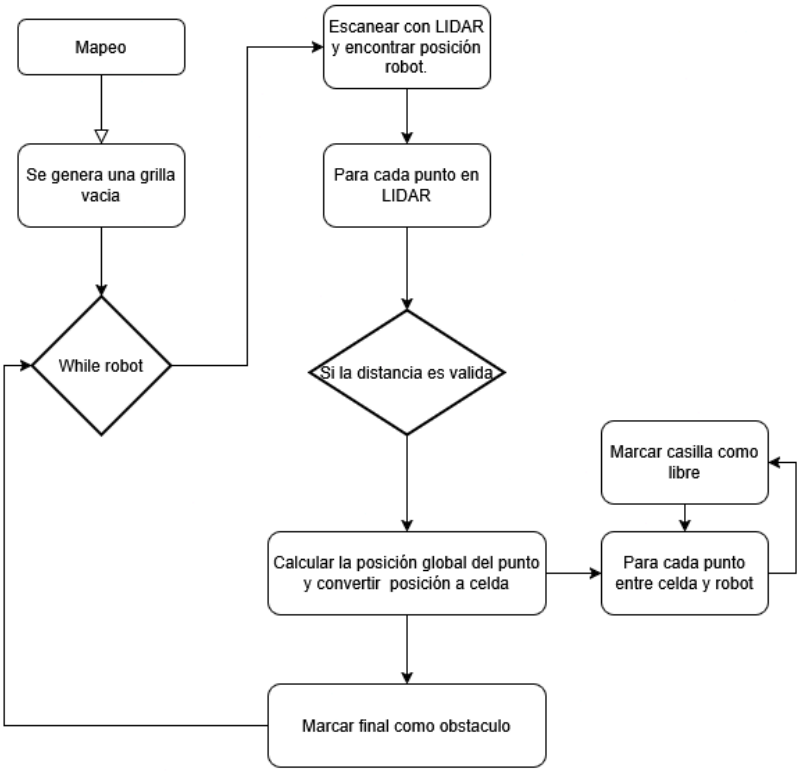
Para la navegación y exploración autónoma del entorno simulado, se implementó el siguiente enfoque algorítmico:

- **Algoritmo de planificación Dijkstra**:
    - Se empleó el algoritmo clásico de Dijkstra para determinar rutas óptimas sobre la grilla de ocupación generada a partir de los datos del sensor LIDAR. Este algoritmo permite calcular el camino más corto entre un nodo de inicio y una posición objetivo, evitando las celdas marcadas como ocupadas.
- **Actualización dinámica del mapa de ocupación**:
    - En cada ciclo de simulación, se realiza un escaneo completo con el sensor LIDAR, y se actualiza la grilla de ocupación (200x200 celdas), marcando obstáculos detectados en función de su posición relativa y distancia.
- **Control cinemático diferencial:**
    - El movimiento del robot se controla ajustando la velocidad de cada una de sus ruedas en función de su orientación y posición respecto a la siguiente dirección de avance

5 Diagramas de flujo y pseudocódigo de la solución

Pseudocódigo y diagrama de flujos para el mapeo:

```
Inicializar grid de ocupación (todo desconocido)

Mientras el robot esté activo:
    Obtener posición y orientación del robot (x, z, theta)
    Leer datos del LIDAR (distancias)

    Para cada rayo del LIDAR:
        Si la distancia es válida:
            Calcular la posición global del punto detectado
            Convertir esa posición a celda del grid

            Para cada celda entre el robot y el punto:
                Marcar como libre

            Marcar la celda final como obstáculo

    Mostrar o guardar el grid de ocupación
```



6 Resultados obtenidos (métricas de desempeño del robot)

Para evaluar el desempeño del sistema de exploración autónoma, se midió el **tiempo total necesario para que el robot explorará la totalidad del entorno simulado de 2m x 2m**, identificando todos los obstáculos presentes.

Métricas registradas:

- **Tiempo total de exploración**: 3 min 12 seg
- **Porcentaje de mapa explorado**: 74.94 %

La métrica principal es el tiempo de exploración total, permitiendo evaluar la eficiencia del mapeo autónomo bajo las condiciones del entorno personalizado.

7 Análisis de los algoritmos utilizados (precisión, eficiencia)

El sistema de exploración basado en el sensor LIDAR y el mapeo sobre una grilla de ocupación demostró ser eficaz para detectar obstáculos de distintas formas y tamaños (cajas y barriles) en un entorno cerrado de pequeñas dimensiones.

El uso de un algoritmo clásico como Dijkstra en entornos de alta resolución (200x200 celdas) resultó computacionalmente manejable, permitiendo planificaciones rápidas a nivel local.

Se observó una buena precisión en la detección y representación espacial de los obstáculos, logrando evitar colisiones directas durante la exploración. No obstante, la planificación reactiva puede presentar demoras en escenarios muy densos en obstáculos debido a la necesidad de constantes actualizaciones del mapa.

8 Reflexión sobre mejoras y optimización del sistema

A partir de la experiencia obtenida en este proyecto, se identifican varias oportunidades de mejora que permitirían incrementar la eficiencia y funcionalidad del sistema:

- **Implementar de forma completa y funcional la planificación de rutas** mediante el algoritmo de Dijkstra, permitiendo al robot desplazarse de forma deliberada hacia una meta definida, optimizando los tiempos de navegación.
- **Optimizar la resolución del mapa de ocupación** o aplicar técnicas de submuestreo adaptativo, para reducir la carga computacional en entornos de mayor tamaño o densidad de obstáculos
- **Incorporar estrategias de evasión reactiva** complementarias basadas en detección local (por ejemplo, detección de obstáculos frontales) para aumentar la robustez en entornos dinámicos.