

Instituto Tecnológico de Estudios Superiores de Monterrey



Implementación de robótica inteligente & Manchester Robotics

Manchester Robotics: Challenge 2

Profesores:

Rigoberto Cerino Jiménez

Dr. Mario Martinez

Integrantes

Daniel Castillo López A01737357

Emmanuel Lechuga Arreola A01736241

Paola Rojas Domínguez A01737136

10 de Abril de 2025

Índice

Índice.....	1
Resumen.....	2
Objetivos.....	3
Introducción.....	4
Mensajes Pose.....	4
Lazo abierto de control en un robot móvil.....	4
Cálculo de la distancia y ángulo recorrido por un robot móvil.....	5
Solución del problema.....	7
Nodos y funcionalidades.....	7
• Path_generator.py.....	7
Resultados.....	10
Conclusiones.....	11
Referencias.....	12

Resumen

Este proyecto desarrolló un sistema integral de control para el robot puzzlebot, implementando trayectorias tanto cuadradas como personalizadas mediante un enfoque modular en ROS 2. El sistema combina un generador de trayectorias interactivo con controladores robustos basados en máquinas de estados finitos (FSM), y para esto se necesitó un poco de investigación o revisar algunos conceptos como lo es la pose de un robot móvil, la el lazo abierto de control además del cálculo de la distancia y ángulo recorrido.

Los componentes clave incluyen:

- Un generador de trayectorias: que permite definir waypoints mediante coordenadas (x,y) y seleccionar entre control por velocidad o tiempo, con cálculos automáticos y límites de seguridad integrados.
- Un controlador especializado: desarrollado para trayectorias cuadradas de 2 m, con FSM de 8 estados y un ajuste automático de parámetros.
- Un controlador general.

De esta manera presentamos los resultados del mini reto de manera en la que se demuestra la capacidad del robot para completar trayectorias con precisión mientras mantenía los parámetros operativos seguros en caso de que los parámetros puestos por el usuario no cumplan con los rangos establecidos.

Objetivos

En esta sección se presentan el objetivo general y los objetivos particulares del reto, los cuales están enfocados en la implementación de un control de lazo abierto para la trayectoria de un Puzzlebot.

Objetivo general: Diseñar e implementar un sistema de control en ROS2 que permita conducir un robot móvil a través de dos trayectorias, utilizando un controlador de lazo abierto robusto según parámetros definidos por el usuario.

Objetivos específicos:

- Implementar un controlador de lazo abierto robusto, autoajutable y que se adapte a perturbaciones en el Puzzlebot
- Trazar con el robot una trayectoria cuadrada de 2 metros por lado
- Trazar una trayectoria con puntos de posición establecidos por el usuario
- Verificar la velocidad mínima posible (dead zone) y velocidad máxima permisible en el robot (saturación)
- Analizar si es posible alcanzar ciertas posiciones y ángulos.
- Calibrar de manera adecuada las velocidades lineales y angulares, o los tiempos, para alcanzar los puntos y giros deseados con mayor precisión.

Introducción

En robótica, la planificación de trayectorias es esencial para aplicaciones que van desde la logística hasta la búsqueda y exploración. Este mini reto incorpora un nodo en ROS configurable que genera trayectorias personalizadas basadas en algunos parámetros que el usuario deberá ingresar (puntos, velocidades y/o tiempos), garantizando la robustez en el modelo para soportar las incertidumbres que lleguen a presentarse en el entorno.

El desafío implica diseñar un sistema que interprete parámetros del usuario, como son los puntos, las velocidades o un tiempo y calcular valores de manera óptima. Además de incluir la robustez, que es la “capacidad de un sistema para manejar errores y entradas incorrectas” Para ello se implementa una máquina de estados finitos, donde se manejan los movimientos (Avance, giros y pausa) y un mensaje ROS integrado que integre pose, velocidades y tiempo. las pruebas en gazebo y el robot real demostrarán la adaptabilidad del sistema bajo perturbaciones, validando su uso en entornos dinámicos.

Mensajes Pose

En robótica móvil, el estado de un robot se refiere a la descripción de su posición, orientación, velocidad y aceleración respecto a un sistema de referencia. Los mensajes tipo Pose, ayudan a representar de forma matemática el estado de un robot, lo cual es útil al momento de planear trayectorias o la navegación del robot.

Lazo abierto de control en un robot móvil

El control de lazo abierto es un tipo de control donde la salida no se mide ni se retroalimenta para su corrección. En otras palabras, el sistema de control de lazo abierto es un sistema en el que la acción de control es independiente de la salida del sistema. (Manchester Robotics Ltd, s.f.-f)

En el contexto de la robótica móvil, un sistema de lazo abierto implica que el robot se desplaza y realiza acciones de acuerdo con una secuencia preprogramada de instrucciones, sin utilizar sensores para verificar si su movimiento se está ejecutando correctamente. Por lo tanto, aunque el control en lazo abierto puede ser adecuado para tareas simples y entornos controlados, su aplicación práctica en robótica suele estar limitada por la falta de precisión y adaptabilidad ante cambios del entorno o condiciones dinámicas.

Por ejemplo, en una tarea de navegación donde el robot debe seguir una trayectoria cuadrada, un sistema de lazo abierto puede controlar los motores para avanzar durante un tiempo determinado y luego girar un ángulo estimado basado también en tiempo. Sin embargo, si el robot se desplaza más lento, por ejemplo por no tener suficiente batería, podría no recorrer la distancia esperada, provocando una trayectoria incorrecta la cual no se puede corregir debido a que no existe retroalimentación en el sistema de control.

A pesar de sus limitaciones, el control en lazo abierto tiene como ventajas:

- Al no requerir sensores ni procesamiento adicional para interpretar datos, los sistemas mantienen un diseño más sencillo.
- En comparación con otros controladores, un control de lazo abierto presenta un menor costo, debido a que reduce el uso de hardware y software de monitoreo.
- Debido a que no es necesario realizar cálculos de retroalimentación ni análisis del entorno en tiempo real, este tipo de sistema representa una menor carga computacional.
- Es ideal para prototipos iniciales, pruebas y tareas repetitivas en condiciones conocidas

Sin embargo, los sistemas de control de lazo abierto presentan desventajas en entornos impredecibles:

- El sistema no puede corregir errores si la acción realizada no produce el resultado esperado
- Los cambios en el entorno en el cual opera el robot móvil, afectan el desempeño del sistema y no hay manera de compensarlos.
- Las pequeñas imprecisiones o errores en cada paso pueden acumularse, produciendo grandes desviaciones en trayectorias o tareas.
- El sistema no se ajusta automáticamente ante condiciones cambiantes, es decir, no es adaptable.

El término de robustez en un controlador de lazo abierto se refiere a un diseño de controlador que, a pesar de no utilizar retroalimentación, es capaz de mantener un desempeño aceptable aun en presencia de incertidumbres o perturbaciones limitadas en el sistema o en el entorno. La robustez se logra, por ejemplo, seleccionando tiempos, velocidades y trayectorias que toleran cierto margen de error sin comprometer el comportamiento general del sistema.

Cálculo de la distancia y ángulo recorrido por un robot móvil

En un sistema de control de lazo abierto el cálculo de la distancia y el ángulo recorrido se realiza sin retroalimentación, es decir, sin sensores que confirmen la posición real del robot. Este tipo de estimación se basa en modelos cinemáticos y tiempos de operación, utilizando directamente los comandos enviados al robot (velocidades y tiempos) para inferir la posición alcanzada.

Cálculo de distancia recorrida

Para un movimiento en línea recta, la distancia recorrida d se estima mediante la fórmula:

pongo fórmula después.

- v es la velocidad lineal (en metros por segundo, m/s),
- t es el tiempo durante el cual el robot se desplaza a esa velocidad (en segundos, s).

Cálculo del ángulo girado

Para los giros, se considera la velocidad angular w (en radianes por segundo) y el tiempo de rotación t

Solución del problema

Nodos y funcionalidades.

- ***Path_generator.py***
 - Genera trayectorias a partir de puntos (x,y) ingresados por el usuario, con opción de control por velocidad o tiempo.
- Implementación:
 - Solicita coordenadas (x,y) y el modo de control (v para velocidad, t para tiempo).
- Límites dinámicos:
 - Aplica clamp() para asegurar que las velocidades lineales (Linear_min/max) y angulares (Angular_min/max) estén dentro de rangos seguros.
- Mensajes Personalizado:
 - Publica en el tópico **pose** un mensaje TimedPose (definido en custom_interfaces/msg/) que incluye:
 - Pose (posición + orientación).
 - Velocidades lineales/angulares.
 - Tiempo estimado para movimientos.
- ***Controller.py***
 - Controla el robot para seguir waypoints usando una máquina de estados finitos (FSM).
- Implementación:
 - Estados:
 - **Rotate_to_goal**: Gira el robot hacia el waypoint.
 - **Forward**: Avanza en línea recta.
 - **IDLE**: Espera nuevos comandos.
 - Autoajuste:

- Calcula velocidades o tiempos según el mensaje recibido (TimedPose).
- Robustez:
 - Detiene el robot con `Twist ()` si no hay datos válidos.

Implementación de la trayectoria cuadrada.

El objetivo es mover el robot en un cuadrado de 2 metros de lado con giros de 90 grados, permitiendo al usuario controlar el movimiento mediante velocidades o tiempos específicos.

Características clave:

1. Máquina de estados finitos (FSM):
 - a. 8 estados (4 tramos rectos + 4 giros).
 - b. Transiciones basadas en tiempo (forward time para tramos rectos, rotate time para giros).
- ***Controller_square.py***
 - Mueve el robot en un cuadrado de 2m con lógica de autoajuste.
 - Solo acepta valores positivos.
 - Rechaza entradas no numéricas.
 - Implementación:
 - FSM de 8 Estados:
 - 4 tramos rectos + 4 giros de 90°.
 - Transiciones basadas en tiempo (Forward_time, rotate_time).
 - Modos de control:
 - Velocidad fija: Usuario define (**v**, **w**) velocidad lineal y angular.


```
forward_time = 2.0 / 0.1 = 20.0 s # Si linear_speed = 0.1 m/s

rotate_time = 1.57 / 0.5 ≈ 3.14 s # Si angular_speed = 0.5 rad/s
```
 - Tiempo fijo: Usuario define **t**, y el nodo calcula (**v**, **w**).


```
linear_speed = 2.0 / 4.0 = 0.5 m/s # Ajustado a LINEAR_MAX si excede
```

```
angular_speed = 1.57 rad / 1.5 s ≈ 1.05 rad/s # 90° en  
radianes
```

- Límites:

- Use la función **clamp ()** para ajustar velocidades a los rangos seguros del robot, con el fin de garantizar que las velocidades estén dentro de los rangos seguros del robot.

```
LINEAR_MIN = 0.0468 # m/s
```

```
LINEAR_MAX = 0.2557 # m/s
```

```
ANGULAR_MIN = 0.3874 # rad/s
```

```
ANGULAR_MAX = 2.6531 # rad/s
```

Link de video para visualización:

En el siguiente video se presentan como es que se ejecuta en terminal y la comunicación entre el robot y la computadora, además que el robot va ejecutando los pasos para hacer un cuadrado con su ruta.

<https://youtu.be/BkcUbPOqRTY>

Resultados

Se realizaron pruebas en simulación y en el robot real para validar el sistema de control. El robot completó con éxito la trayectoria cuadrada de 2 metros por lado, siguiendo una máquina de estados finitos con transiciones por tiempo. Se observaron trayectorias estables y sin desviaciones significativas, siempre que las velocidades se mantuvieran dentro de los rangos seguros.

En trayectorias personalizadas, el generador interpretó correctamente los puntos (x, y) ingresados, generando mensajes Timed Pose adecuados. El controlador FSM ejecuta los movimientos de forma ordenada (giros, avances y pausas), y el sistema fue capaz de rechazar entradas inválidas para evitar errores.

El video en youtube (<https://youtu.be/BkcUbPOqRTY>) muestra la ejecución correcta de la trayectoria cuadrada, confirmando la funcionalidad del sistema y su robustez en lazo abierto.

Conclusiones

En este proyecto pudimos implementar un sistema de control en lazo abierto para el Puzzlebot usando ROS 2. Lo interesante fue que, a pesar de no tener retroalimentación del entorno, logramos que el robot siguiera trayectorias cuadradas y personalizadas con buena precisión, siempre y cuando se respetaran los límites de velocidad.

Nos dimos cuenta de que con una buena calibración de tiempos y velocidades, el sistema es capaz de funcionar de forma estable y segura. Además, la máquina de estados ayudó mucho a organizar los movimientos del robot, haciendo más claro cuándo tenía que avanzar, girar o detenerse.

También vimos que, aunque el control en lazo abierto tiene sus limitaciones como que no puede corregirse si hay errores, con un diseño robusto y parámetros bien ajustados se puede obtener un comportamiento bastante aceptable.

En general, este reto nos permitió entender mejor cómo planear trayectorias, cómo manejar mensajes personalizados en ROS y cómo estructurar un sistema modular. Como siguiente paso, sería interesante integrar sensores para mejorar la precisión y hacer un sistema de control con retroalimentación.

Referencias

En este apartado se anexan los elementos consultados para el desarrollo del tema de investigación.

ManchesterRoboticsLtd. (s. f.-f).

TE3002B_Intelligent_Robotics_Implementation_2025/Week2/Presentations/PDF/MC

R2_Open_Loop_Control_V3_watermark.pdf at main ·

ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation_2025.

GitHub.

https://github.com/ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation_2025/blob/main/Week2/Presentations/PDF/MCR2_Open_Loop_Control_V3_watermark.pdf