Actividad R_2. Cálculo del error de un robot móvil diferencial

Oscar Ortiz Torres A01769292

Yonathan Romero Amador A01737244

Ana Itzel Hernández García A01737526

Implementación de robótica inteligente

Grupo 501

Tecnológico de Monterrey Campus Puebla

Martes 22 de abril de 2025

Resumen

En esta actividad se desarrollaron dos nodos en ROS2 para el robot móvil diferencial Puzzlebot. El primero estima su posición en un plano 2D mediante odometría diferencial a partir de los datos de los encoders de las ruedas. El segundo nodo calcula en tiempo real el error en distancia y orientación con respecto a una posición objetivo definida por el usuario. Para ello, se emplea el modelo cinemático del robot y se integran los datos de velocidad lineal. Los resultados mostraron un error acumulativo del 10% en la distancia recorrida y del 9.8% en la rotación, cifras razonables considerando que se trabajó sin sensores de corrección externos.

Objetivos

Objetivo general

Desarrollar e implementar nodos en ROS2 que permitan estimar la posición del Puzzlebot mediante odometría y calcular en tiempo real los errores de distancia y orientación con respecto a una posición objetivo.

Objetivos particulares

1. Implementar un nodo en ROS2 que utilice los datos de los encoders del Puzzlebot para estimar su posición mediante odometría.
2. Verificar el correcto funcionamiento del nodo de localización empleando el paquete teleop_twist_keyboard para el control manual del robot.
3. Desarrollar un nodo adicional que calcule en tiempo real los errores de distancia (*ed*) y orientación ($e\theta$) respecto a una posición objetivo introducida por el usuario.

Introducción

La odometría es el uso de datos obtenidos a través de sensores de movimiento para así poder calcular el cambio de la posición con respecto al tiempo, durante la navegación. Esto

siendo necesario para robots autónomos ya que permite realizar tareas sobre terrenos tras saber su propia localización.

Para trabajar en este reto se necesita saber la pose, esta es la posicion que algun cuerpo en un plano, en este reto es un espacio bidimensional, Se guarda las coordenadas cartesianas del robot y el rumbo del robot, esto es un término para la dirección hacia la que está orientada la parte frontal del robot.

El cálculo del error se obtiene al determinar el cambio de posición del robot, en este caso la odometría nos permite obtener los cambios reales del robot indicados por los sensores, esto se compara con el objetivo deseado, esta diferencia es conocida como el error

Solución del problema

El primer nodo desarrollado de la práctica consistió en estimar la posición del Puzzlebot en un plano 2D utilizando los datos de los encoders de sus motores. Se utilizó la técnica de odometría diferencial, la cual permite conocer el desplazamiento del robot a partir de la información de sus ruedas.

Diseño e implementación del nodo

- Suscripción a datos de sensores: se hacen suscripciones a los tópicos `VelocityEncL` y `VelocityEncR`, los cuales contienen los datos de las velocidades angulares de las ruedas izquierda y derecha, respectivamente.
- Cálculo de velocidades lineales: utilizando el radio de las ruedas, se calculan las velocidad tangenciales v_r y v_l, respectivas a cada rueda
- Estimación de la posición y orientación: de acuerdo con el modelo cinemático de un robot diferencial, se calcularon la velocidad lineal V, la velocidad angular Ω, y

se actualizó la posición estimada (X, Y) y la orientación θ del robot en cada iteración. Dicho modelo fue:

$$V = \frac{r}{2}(\omega_r + \omega_l),\ \Omega = \frac{r}{l}(\omega_r - \omega_l)$$

$$X_{k+1} = X_k + Vcos(\theta) \cdot dt,\ X_{k+1} = Y_k + Vsin(\theta) \cdot dt,\ \theta_{k+1} = \theta_k + \Omega \cdot dt$$

- Publicación del estado estimado: el nodo publica los datos de posición y orientación estimadas en el tópico odom utilizando el mensaje tipo `nav_msgs/Odometry`.

Para el desarrollo del segundo nodo encargado del cálculo del error en la distancia y en el ángulo, comenzamos con la declaración de los

Diseño e implementación del nodo

- Parámetros: se declaran los parámetros `distancia_obj`, la cual es la distancia deseada y `theta_obj` que es el ángulo deseado. Dichos parámetros pueden ser modificados por el usuario con `rqt_reconfigure` debido al `parameters_callback` implementado.
- Suscripción a datos de sensores: se reciben las velocidades angulares de las ruedas por medio de suscripciones a los tópicos `VelocityEncR` y `VelocityEncL`.
- Cálculo de la distancia estimada: se calcula sumando en cada instante de tiempo el valor absoluto de la velocidad lineal del robot por el tiempo transcurrido representando el avance total sin tomar en cuenta una dirección.

$$distancia_{robot} = \sum_{i=0}^{n} |V_i| \cdot dt$$

- Cálculo del error: el error de la distancia, almacenado en la variable `error_dist` por la diferencia de la variable `distancia_obj` y la distancia estimada. Mientras que con el ángulo del robot se almacena en la variable `error_theta` convirtiendo el ángulo `theta_obj` de grados a radianes para posteriormente ajustarlo para mantenerlo en el rango del círculo.
- Publicación del error: Ambos errores se publican como mensajes tipo `Float32` para que puedan ser visualizados desde `rqt_plot`.

Resultados

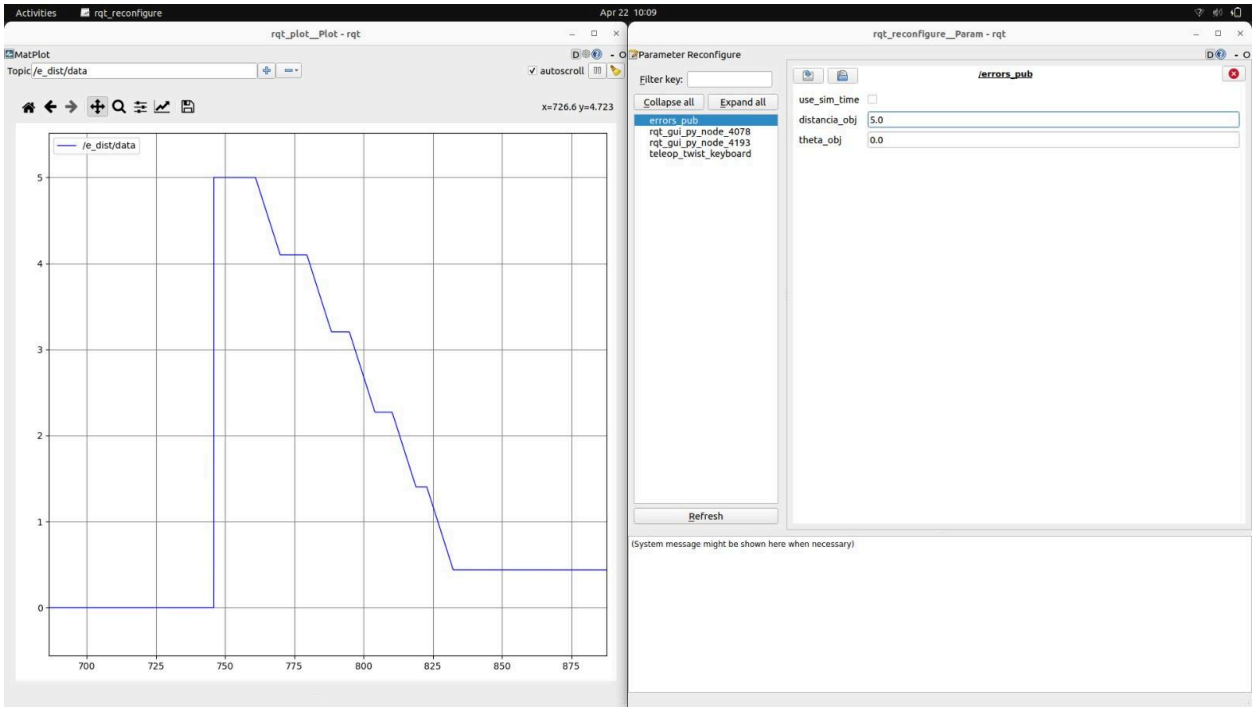

Figura 1 Distancia recorrida según los encoder.

Como podemos observar en la Figura 1, de acuerdo con los resultados obtenidos tras leer los sensores cuando el robot recorre 5 metros, deteniéndose cada metro para la evaluación del

error generado en ese fragmento de distancia. El comportamiento de la gráfica se debe a que se busca observar la acumulación del error en cada segmento.

Al analizar dichos resultados, tenemos que el error va aumentando en cada metro en el que el robot se detiene, obteniendo alrededor de 50 centímetros de error al recorrer los 5 metros, dando un porcentaje de error de 10%. Dicho comportamiento se puede observar en el apartado Anexos.

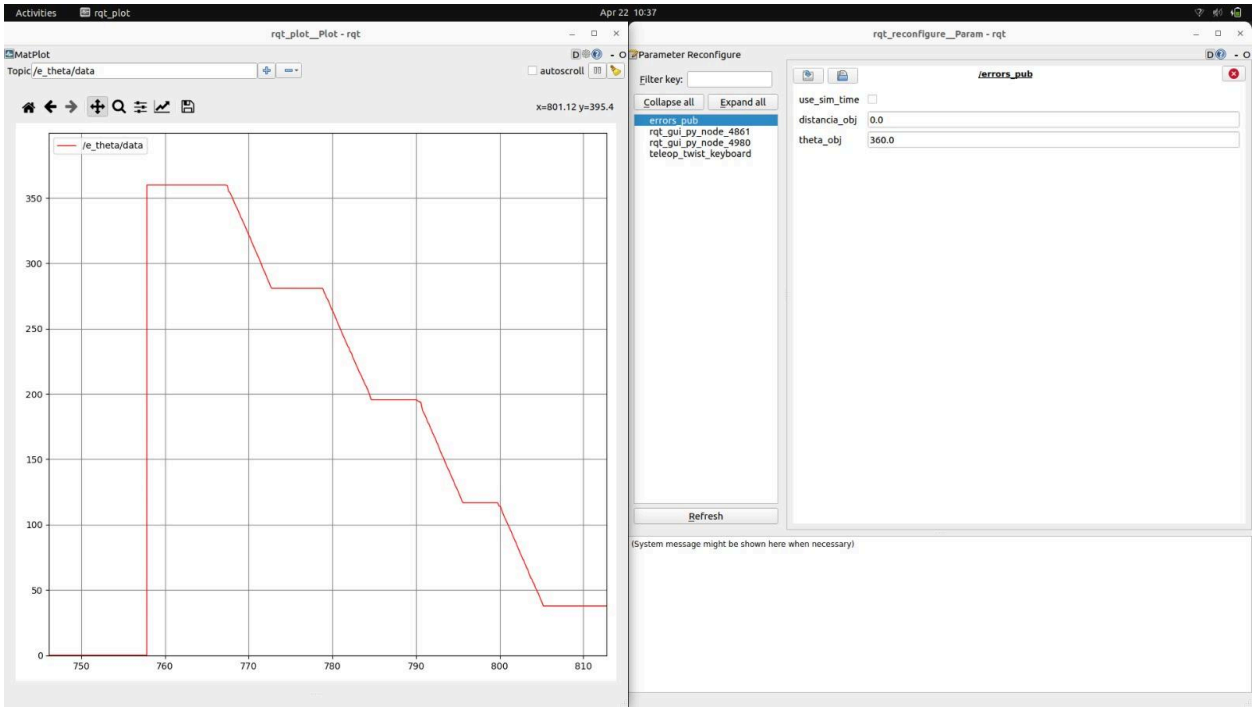

Figura 2 Rotación realizada según el encoder.

En la Figura 2, obtenemos los resultados de la lectura y comparación de la rotación del Puzzlebot, el cual tiene como objetivo girar 360 grados, deteniéndose cada 90 grados. Al igual que la prueba anterior se grafican la acumulación del error en cada ángulo, alcanzando un error total de 35° al completar una vuelta entera, dando un porcentaje de error del 9.8%. Igualmente se puede observar el comportamiento en el apartado de Anexos.

Conclusiones

Para concluir, se cumplieron los objetivos establecidos para esta actividad, implementar los nodos en ROS2 para la lectura de los encoders. Con esto estimando la posición del robot y calculando el error en el cambio de la posición. A través de la odometría diferencial, se logra la correcta estimación de la trayectoria y su orientación teniendo errores acumulativos del 10% en la distancia y del 9.8% en la rotación, valor aceptable en sistemas que no cuentan con sensores externos de corrección

Bibliografía

Odometría. (s. f.). Game Manual 0.

https://gm0.org/es/latest/docs/software/concepts/odometry.html

Anexos

Repositorio Github (reporte y códigos desarrollados):

https://github.com/oscarOT09/actividadR2_ROSario

Videos de funcionamiento del robot publicando el error calculado en tiempo real:

Comportamiento del error en la distancia: https://youtu.be/6pZyvsp4K2o

Comportamieinto del error en el angulo: https://youtu.be/8YeexC0GRxc