Tecnológico de Monterrey

Campus Puebla

Materia

Fundamentación de Robótica TE3001B

Tema

Challenge II

Integrantes

José Jezarel Sánchez Mijares A01735226

Frida Lizett Zavala Pérez A01275226

Antonio Silva Martínez A01173663

Fecha

Abril 25 2023

Contenidos

Resumen

El documento de investigación propone desarrollar las habilidades de los estudiantes en el manejo de ROS (Robot Operating System) y Gazebo, a través de una actividad que involucra la interacción con un robot llamado Puzzlebot. El objetivo de la actividad es aumentar el nivel de interacción y la implementación de funciones disponibles en ROS, así como diseñar un controlador P, PI o PID para minimizar el error de posición del robot.

Objetivos

Los principales objetivos de este reporte son los siguientes:

1. Comprender los requisitos técnicos necesarios para desarrollar el reto planteado, como la creación de una carpeta en GitHub, el uso de códigos para la creación de nodos, archivos launch y config, entre otros aspectos.
2. Justificar la importancia y el valor del desarrollo del reto en términos de la aplicación práctica de la tecnología de robótica en el mundo real.
3. Describir el proceso de desarrollo del reto, incluyendo la identificación y resolución de problemas, las decisiones de diseño tomadas y los métodos utilizados para evaluar el rendimiento del robot.
4. Presentar las evidencias del funcionamiento del robot, incluyendo dos videos que muestren su desempeño en el simulador Gazebo y en el robot físico.
5. Proporcionar información sobre los equipos que participaron en la realización del reto, incluyendo los nombres de los integrantes de cada equipo.
6. Compartir el link del repositorio en forma individual y en la plataforma Canvas para su posterior evaluación.
7. Evaluar el éxito del proyecto en función de los objetivos planteados, identificando fortalezas y áreas de mejora.

Introducción

En la robótica móvil, uno de los principales desafíos es el control preciso del movimiento del robot en un entorno cambiante. El control en lazo cerrado es una técnica que se utiliza para mantener el movimiento del robot dentro de los parámetros deseados y corregir los errores de posición en tiempo real. En este sentido, el controlador proporcional integral derivativo (PID) es uno de los métodos más comúnmente utilizados en la implementación del control en lazo cerrado para robots móviles diferenciales.

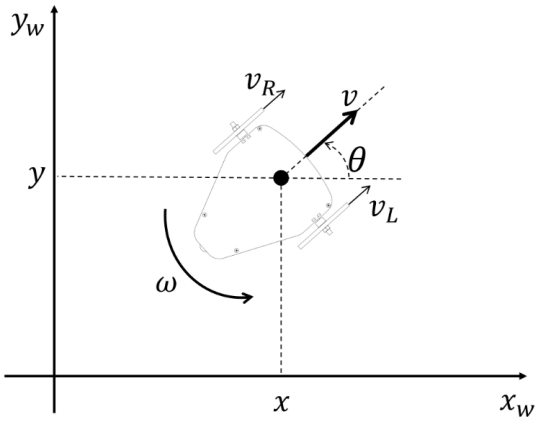
El PID se basa en el cálculo del error entre la posición deseada del robot y su posición real, y ajusta los parámetros del controlador para minimizar este error. Sin embargo, existen varios temas derivados al cálculo del error y la robustez de un controlador que deben ser considerados en la implementación del PID para garantizar un control preciso y confiable del robot.

En este reporte de investigación, se abordará el tema del control en lazo cerrado para un robot móvil, enfocándose en la aplicación del PID para un robot móvil diferencial. Se discutirán los temas relacionados con el cálculo del error y la robustez de un controlador, así como las soluciones propuestas para abordar estos desafíos. El objetivo principal es proporcionar una visión general de los conceptos clave y los métodos utilizados en el control en lazo cerrado para robots móviles, y explorar los avances más recientes en el desarrollo de controladores robustos para mejorar el rendimiento de los robots móviles en diferentes entornos.

Solución del problema

Posición y parámetros físicos

En la primera parte de este reto, lo que realizamos fue obtener las medidas físicas de nuestro robot con el objetivo de poder calcular la posición (x,y,z(ang)) en la que se encuentra nuestro robot, de esas medidas podemos obtener el siguiente modelo



Del modelo se obtienen las siguientes ecuaciones

$$v_{Robot} = \frac{v_R + v_L}{2} = r\frac{\omega_R + \omega_L}{2}$$

$$\omega_{Robot} = \frac{v_R - v_L}{l} = r\frac{\omega_R - \omega_L}{l}$$

La primera nos permite calcular la velocidad lineal del robot, como queremos pasarla a lineal se usa el radio más la suma de las velocidades angulares de cada rueda, para la velocidad angular es el mismo caso, solo que se usa la distancia entre ambas llantas para dividir la resta de las velocidades. A partir de este razonamiento obtenemos las siguientes ecuaciones

$$x_{k+1} = x_k + r\frac{\omega_R + \omega_L}{2}\, dt \cos\theta$$

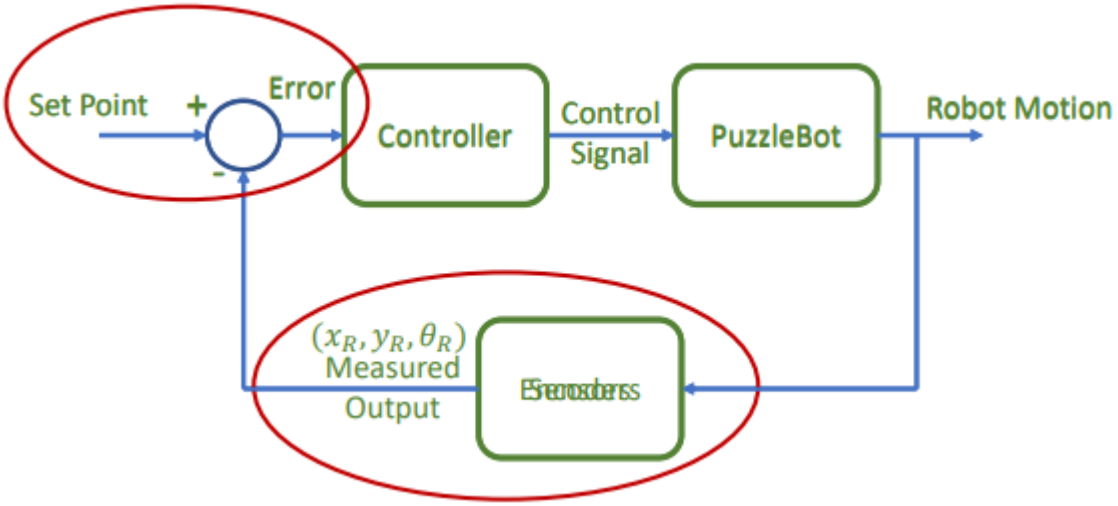
$$y_{k+1} = y_k + r\frac{\omega_R + \omega_L}{2} dt\ \sin\theta$$

$$\theta_{k+1} = \theta_k + r\frac{\omega_R - \omega_L}{l} dt$$

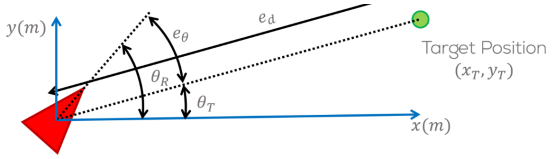
Aquí ya calculamos la posición de x,y,z(ang), en el caso de x,y nos apoyamos de identidades trigonométricas, por lo que colocamos el seno y cos del angulo respectivamente, se pone la posición “anterior”para que en teoría pueda ir actualizando las posición de manera constante

Sistema de control

Para que nuestro puzzlebot sea funcional necesitamos incorporar un sistema de control que nos permita corregir el error de manera automática. Empezamos describiendo nuestro sistema con el siguiente esquema:



Para comenzar a calibrar el controlador tenemos que calcular el error que tiene con respecto a la respuesta esperada, en este caso la diferencia que tiene entre el ángulo y la distancia a la que se encuentra para ello usamos el siguiente esquema



Como se puede apreciar existe una diferencia tanto en la distancia deseada y la distancia en la que se encuentra el objeto. Como tal podemos definir dos errores, el error angular y el error en la distancia que podemos obtener a través de distancias trigonométricas siendo la distancia la hipotenusa del triángulo formado por los dos puntos y el error angular estaría definido por el arcotangente del punto esperado y la resta del punto en el que se encuentra posicionado nuestro objeto con esto procedimos a realizar un análisis en lazo abierto para poder visualizar el comportamiento de la planta. Una vez contando con más parámetros iniciamos la realización de nuestro controlador.

Se utilizaron dos controladores diferentes, uno para el error angular y otro para la distancia, en la distancia un controlador P fue suficiente para reducir nuestro error y que la planta pudiera ser capaz de llegar al punto deseado, primero probamos con un punto, para seguir con dos y tres, y aunque el margen de error iba subiendo con el paso de las iteraciones era bastante eficiente, sin embargo en el caso del error angular no pudimos usar el mismo controlador ya que manejaban diferentes errores cada uno, al principio probamos con solo un p, aunque no reducía el error como lo esperábamos, en consecuencia agregamos la ki, lo que ayudó a acercarnos, poco a poco de manera experimental fuimos calibrando el controlador, uno de los principales errores que aparecieron fue que nuestro sistema se volvió inestable, aunque lo solucionamos moviendo la ki y modificando el límite permitido de margen de error que habíamos establecido.
Pudimos calcular todo esto gracias a varias pruebas hechas en GAZEBO, así como en físico ya que al final tuvimos que modificar un poco los valores del controlador en cada circunstancia

Programación

La siguiente parte corresponde a la incorporación de más puntos a alcanzar para el puzzlebot por lo cual se utilizaron archivos de parámetros que contenían una cadena de floats donde se encontraban los puntos, una vez que el error tanto en el ángulo como en la distancia fuera 0, lo que quiere decir que estábamos en el punto establecido, el robot comienza con el otro punto, así repitiendo este proceso, una mejora que es notable comparado con la entrega pasada ya que se usaron muchos if-else para que siguiera la ruta esperada, con esta mejora reducimos significativamente las líneas de código, además para mejorar nuestro entorno de ROS realizamos también archivos de mensajes donde se enviaron los resultados de las mediciones del error, la posición del robot y la velocidad a la que se encontraba. Esto dio una mayor versatilidad a nuestro código y además de un mayor control en el entorno de trabajo.

Resultados

Los resultados obtenidos fueron el desplazamiento de dos trayectorias con cuatro puntos cada una, siendo que la primera emula la figura de un cuadrado mientras que la segunda emula un zig zag, A continuación se puede visualizar el desplazamiento de las trayectorias.

Cuadrado: https://youtu.be/2A7cIpV-CA8

Zig zag: https://youtu.be/WDWz-631YHU

Conclusiones

Para la resolución de este reto se utilizaron dos controladores, un controlador PI para los ángulos y un controlador P para las distancias, esto se realizó de esta forma debido a que al utilizar dos medidas (Distancia y Ángulos), los errores que se generaban al trabajarlos juntos las necesidades de uno, no complementaban las necesidades del otro haciendo que la medición y la calibración no fueran las adecuadas, mientras que si separabamos ambos puntos podríamos calibrar de una mejor manera el desplazamiento de ambos, por otra parte pudimos apreciar que para poder mejorar la precisión de nuestro controlador es necesario el tomar en cuenta más entradas de sensores como lo son la cámara debido a que así podemos realizar de mejor manera un análisis del entorno del robot. Tenemos varías áreas de mejora en las que nos enfocaremos en las siguientes entregas, como la mejora de nuestro controlador PID .

Referencias

ManchesterRoboticsLtd. (2023). GitHub - ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation. Retrieved 26 April

2023, from
https://github.com/ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation