Instituto Tecnológico de Estudios Superiores de Monterrey

TE3002B.502

Implementación de Robótica Inteligente

Challenge 2.

Gpo 502

Profesor:

Rigoberto Cerino Jiménez

Alumnos:

Daniela Berenice Hernández de Vicente	A01735346
Alejandro Armenta Arellano	A01734879
Dana Marian Rivera Oropeza	A00830027

Fecha: 25 de Abril del 2023

Resumen

En este reporte se podrá observar una descripción de lo que son los custom messages, parameters y launch file, cómo funcionan, todo esto con tal de comprender más allá de simplemente resolver el challenge semanal impuesto por Manchester Robotics.
Aunado a estas descripciones y definiciones, se puede encontrar la solución paso por paso al challenge de esta tercera semana de clases.

Objetivos

En este segundo challenge se espera que los estudiantes repasen los conceptos vistos durante las sesiones con Manchester Robotics..
De igual manera es importante destacar que esta actividad se divide en dos partes.
Por lo cual la primera parte consiste en lo siguiente:

- Usar un controlador PID para realizar el movimiento del robot
- La actividad debe ser implementada tanto en el simulador como en el robot físico.
- Cada estudiante debe crear sus propios archivos, ya que no ha sido provista una plantilla.
- El robot debe seguir una trayectoria cuadrada de 2 metros de longitud en cada lado.
- La posición inicial del robot debe ser [0,0,0] hasta la posición deseada.
- El estudiante debe definir la robustez del controlador y las estrategias implementadas para solución del challenge.
- El controlador debe estar tuneado apropiadamente y considerar la perturbación y el ruido que el sistema pueda causar.
- Se recomienda, pero no se requiere, para el estudiante para usar un archivo de configuración o un parámetro en el archivo de lanzamiento para establecer los objetivos de modo que se pueden cambiar fuera del código (no codificado).

Por otro lado la segunda parte consiste en lo siguiente:

- Usar un controlador PID para realizar el movimiento del robot.
- La actividad debe ser implementada tanto en el simulador como en el robot físico.
- Se debe crear un nodo "***path_generator***", que publique el actual y el próximo objetivo una vez que el robot complete el objetivo actual.
- El nodo mencionado anteriormente puede tomar otras decisiones como cambiar la ganancia del controlador y su metodología, sólo si se considera necesario por el estudiante y las circunstancias.
- El nodo debe dar a conocer al usuario si el punto deseable se puede alcanzar de acuerdo con el comportamiento dinámico del robot.
- El número de x deseadas debe ser especificado por el usuario, como mínimo deben ser 3 diferentes posiciones, sin incluir la posición inicial.
- La posición inicial del robot debe ser [0,0,0] hasta la posición deseada.
- El estudiante debe definir la robustez del controlador y las estrategias implementadas para solución del challenge.
- El controlador debe estar tuneado apropiadamente y considerar la perturbación y el ruido que el sistema pueda causar.

- Se recomienda, pero no se requiere, para el estudiante para usar un archivo de configuración o un parámetro en el archivo de lanzamiento para establecer los objetivos de modo que se pueden cambiar fuera del código (no codificado).
- El mensaje para el tópico "/***goals***" debe ser un mensaje personalizado basado en la posición de geometry_msgs.

Introducción

Semanalmente se presenta un reto por Manchester Robotics, los cuales serán resueltos con la ayuda de programas especializados como ROS y sus complementos como Gazebo, además de ciertos lenguajes de programación como Python o C++.

Aunado a esto cada semana es proporcionada una clase base con respecto a lo que se manejara durante el challenge semanal y ciertos conceptos fundamentales para la resolución del mismo, así como una sesión de preguntas y respuestas donde el objetivo es resolver dudas con respecto al mismo challenge semanal.

Como complemento se proporcionan algunas actividades las cuales de alguna manera llevan de la mano al alumno con el fin de lograr que este pueda desarrollar el challenge semanal con éxito.

En esta primera semana al ya contar con una noción fuerte de la arquitectura de trabajo de ROS y a la simulación de Gazebo, se nos introdujo ante los siguientes conocimientos:

- Control de lazo cerrado para un robot móvil:
    - Un robot móvil se presenta como un control de lazo cerrado, el cual puede realizar un cambio de posición de sus elementos, esto en función de la información captada de su entorno mediante sus sensores. [1]
    - Es importante reconocer que es casi imposible tunear el controlador para que este genere una trayectoria perfecta hacia la posición deseada en el robot.
    - Ante esta situación también es importante reconocer que al no ser exactamente perfecto, se busca generar una tasa de error muy baja. [2]
    - De igual manera es importante destacar que este es un sistema que utiliza fundamentalmente la retroalimentación para mantener una variable controlada, al contar con la medición de dicha variable, esta se compara con el valor de referencia dado anteriormente, y cualquier diferencia entre estos dos valores es utilizada para ajustar el controlador empleado.
- PID aplicado a un Robot Móvil diferencial:
    - Un controlador PID es el encargado de la modulación de la señal de control del sistema de acuerdo con el error entre el valor medido y el valor deseado. [3]
    - Este tipo de controlador se recomienda en los procedimientos industriales, aun cuando se presentan interferencias externas.
    - El Control Proporcional produce una salida del controlador basado en el error presentado por el sistema; en otros términos es un amplificador con ganancia ajustable, por lo que este reduce el tiempo de subida e incrementa el sobretiro para reducir el error a un estado estable. [3]
    - El Control Diferencial es capaz de predecir el error y toma medidas oportunas para poder corregirlo. Dicha acción reacciona a la velocidad de la entrada y

ajusta la señal de salida, por lo que va en función de la tasa de cambio del error y se corrige antes de que el error aumente. [3]
    - El Control Integral proporciona una salida del controlador proporcional al error acumulativo convertido en tiempo de respuesta. De igual manera se encarga de notificar a la salida que tan rápido hay que moverse cuando se llega a producir un error. [3]

- Cálculo del Error:
    - El error en un controlador es la medida de la diferencia entre el valor deseado y el valor real de la variable controlada en el sistema. [4]
    - Por lo que la función del error es utilizarlo como entrada para nuestro controlador, con el fin de ajustar la salida para que con esto se reduzca el error hasta que sea igual a cero. [4]
- Robustez de un controlador:
    - Esta es la capacidad de un controlador para mantener un buen rendimiento y función cuando se presentan ciertas alteraciones y/o incertidumbres ante el sistema controlado. [5]
    - De igual manera es capaz de manejar ciertas perturbaciones, no linealidades, variaciones, entre otros factores en las condiciones de operación, los cuales podrían afectar el comportamiento del sistema controlado y que en algunas ocasiones puedan pasar desapercibidos.

Solución del problema

Para la solución de ambas partes del reto se tienen dos nodos, ***path_generator*** y ***controller***.

Cabe mencionar que por temas de practicidad los códigos de cada uno de los nodos se encontrarán específicamente en la sección de anexos donde se incluirá una liga al github donde se encuentran cada uno de ellos.

Primera parte y Segunda parte:

Primero el nodo ***Path_Generator*** es el encargado de la lectura de los parametros, es decir es el encargado de la lectura de las posiciones en x y.
Posteriormente dichos valores se mandan hacia el nodo controller, donde son interpretados para obtener correctamente las posiciones.
Destacamos que la función del nodo controller es de lo más importante para la realización del challenge, ya que mediante el puzzlebot se están obteniendo las posiciones del encoder, por lo que conocemos cuánto avanza.

Ahora bien, mediante un controlador el cual determina la distancia que nosotros hayamos ingresado, como por ejemplo 1, mientras que la condición inicial es 0.1, ambas distancias se restan para generar el error, siendo esta por ejemplo de 0.9.
Dicho error se ingresa al controlador empleado, en este caso PI, este se encargará de ajustar la velocidad para que el error se pueda volver cero, siendo lo mismo lo que sucede con los ángulos, es decir que si el robot tiene que girar hacia un lado, se genera

otro error, por lo que cuando el error del ángulo sea cero, significa que el robot girara en línea recta, mientras que no sea así el robot seguirá moviéndose.

Nuestro robot se detendrá cuando llegue al punto indicado, para esperar un segundo y posteriormente proseguir con los demás puntos de su trayectoria.

Resultados

Es importante destacar que cada uno de los resultados obtenidos se grabaron y se encuentran en la sección de anexos dentro del link de github que es proporcionado en dicha sección.

Ahora bien, se utilizaron los siguientes valores para la realización de diferentes simulaciones.

- Resultado_Simulación_1 la trayectoria es de x = 2, y = 1.
- Resultado_Simulación_2 la trayectoria es de x = 1, y = 2.
- Resultado _Simulación_3 la trayectoria es de x : '1.0,1.0,1.0', y: '0.0,1.0,0.5'.

Con esto hemos podido observar que si al programa se le proporciona un punto exacto (siendo el caso de los Resultados 1 y 2), este puede llegar con un error muy pequeño, siendo este de 0.5, mientras que si en vez de proporcionarle un solo punto se le proporciona una trayectoria (siendo el caso del Resultado 3) el error se va acumulando, por lo que el punto exacto se puede ver distorsionado.

Conclusiones

Aún cuando en algunos momentos parecía inalcanzable, se consiguió realizar con éxito cada uno de los objetivos antes mencionados.

Se logra realizar de manera correcta el recorrido de las trayectorias solicitadas, más sin embargo al toparnos con el pequeño detalle de la acumulación del error, podemos concluir que hay dos posibles respuestas ante esto.

Primero que el controlador no esté optimizado, ya que al ser estos valores que se sacaron mediante prueba y error, pudiera ser que si están correctos aunque no al cien por ciento.

Por otro lado, también podría ser el hecho de que es difícil conseguir un controlador que te de los resultados cien por ciento exactos, por lo que se busca algo no muy lejano al resultado deseado.

Bibliografía

1. *Control y Robótica*. (s. f.). robotica1. Recuperado 25 de abril de 2023, de http://centros.edu.xunta.es/ieseduardopondal/tecnoweb/temas_informatica/robotica1.pdf
2. Manchester Robotics [MCR2]. (s. f.). Closed Loop Control_v2. *Github*. Recuperado 25 de abril de 2023, de https://github.com/ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation/blob/main/Week%203/Challenges/MCR2_Mini_Challenge2.pdf
3. Prado Martin, C. (2023, 1 marzo). *Controlador PID - Control Automático - Picuino*. Picuino. Recuperado 25 de abril de 2023, de https://www.picuino.com/es/control-pid.html
4. Perez, M. A., Perez Hidalgo, A., & Perez Berenguer, E. (2007). *INTRODUCCION A LOS SISTEMAS DE CONTROL Y MODELO MATEMÁTICO PARA SISTEMAS LINEALES INVARIANTES EN EL TIEMPO.* Universidad Nacional de San Juan. Recuperado 25 de abril de 2023, de http://dea.unsj.edu.ar/control1/apuntes/unidad1y2.pdf
5. Mariana. (2022). Control robusto. *Fisicotrónica*. http://fisicotronica.com/control-robusto/

Anexos

Link al Github:
https://github.com/Bere901/-Retos_Manchester_Robotics_IRI/tree/main/Week%202