Laboratorio de Principios de Mecatrónica

30 de abril de 2021

Práctica #7

Robot Operating System

Grupo:

L002

Estudiante:

- Oceguera Urquiza Juan Manuel
- Jáuregui Tapia Jesús Enrique

Profesor:

Benito Granados–Rojas

Índice

1. Introducción

El sistema de operación robótica (ROS por sus siglas en inglés) es un marco de trabajo para escribir software a robots. Es una colección de herramientas, librerías y convenciones que buscan facilitar la tarea de crear comportamientos robóticos complejos a través de distintas plataformas. Debido a su capacidad de conectarse a distintas fuentes permite el desarrollo colaborativo de software robótico. [1]

En el ambiente de robótica, un agente usualmente se refiere a un código independiente que corre en función de las indicaciones de un usuario de red. Como su nombre lo indica, un agente móvil se distingue por su capacidad motriz. [2]

Turtlesim es una herramienta de ROS desarrollada por Josh Faust y Dirk Thomas que provee un entorno especializado para la simulación de agentes móviles. En este caso, los agentes móviles se representan por tortugas que se desplazan en un escenario cuadrado. Para desplazar dichos agentes, se envía a la tortuga la velocidad en línea recta y angular. Es común utilizar la plataforma para entender el funcionamiento de ROS y sus distintas librerías (y - posteriormente - aplicar el conocimiento en un entorno físico). [3]

Se pretende utilizar el entorno de Turtlesim para simular, comunicarse, y controlar un agente móvil de tipo diferencial a través de ROS.

2. Experimentos y Simulaciones

2.1. Configuración de paquetes

Se realizó la configuración del paquete con el que se trabajó en el resto de la práctica. Para ello, se creó un paquete de ROS a través de la consola en la carpeta de "catkin_WS". Después, se creó un directorio de scripts dentro del paquete, en donde desarrollamos los códigos explicados a continuación. Insertamos estos códigos en dicho directorio, y cambiamos los permisos de acceso para poder ejecutarlos desde una terminal.

2.2. Trayectoria rectangular en turtlesim

Inicialmente, para representar la trayectoria rectangular, se ubica una tortuga en la posición (1,1). A partir de ese punto, se orienta hacia la posición (10,1) con la función orientate, se da una ligera espera y se le indica que se aproxime a dicha posición con la función go-to-goal. Una vez llegado a este punto, se da otra espera y se orienta al siguiente extremo del cuadrado, esta vez en la posición (10,10). Este procedimiento de llegar a un punto, esperar, orientar al siguiente y avanzar, se repite con los 4 extremos en las posiciones (10,1), (10,10), (1,10) y (1,1), respectivamente.

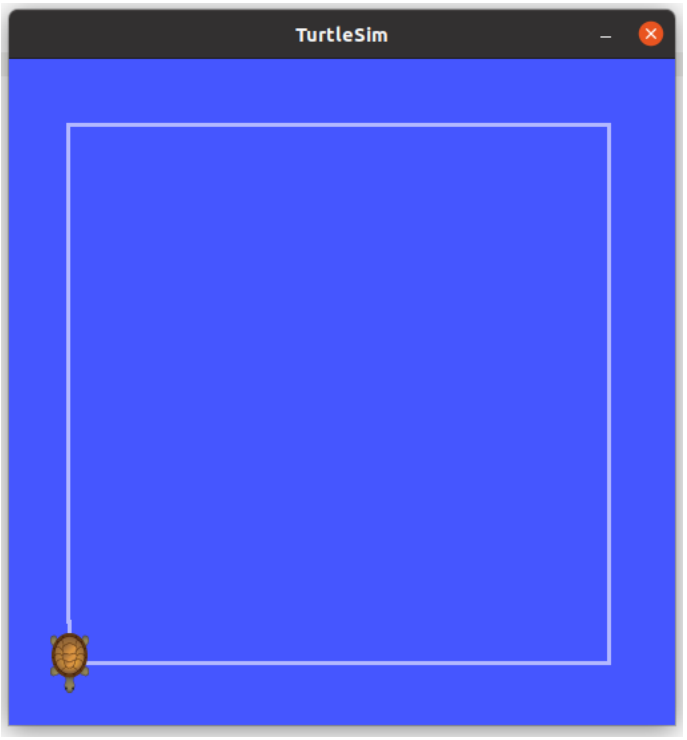

Figura 1: Trayectoria rectangular.

2.3. Trayectoria circular en turtlesim

Para la trayectoria circular se sigue una lógica similar en cuanto a orientarse, esperar, avanzar, esperar y volverse a orientar, pero esta vez se repite este ciclo con ayuda de un ciclo for. Inicialmente, se posiciona la tortuga en la posición (10,5.5), pero para la trayectoria de un círculo ideal se requeriría una cantidad infinita de puntos. Por ello, se realiza un ciclo de n iteraciones en donde n es el número de lados de un polígono que se utiliza para aproximar el círculo. En este caso, la aproximación fue con 77 iteraciones; es decir, se crea un polígono de 77 lados (pero visualmente es una buena aproximación a círculo). Para definir las direcciones de orientación y avance, se crea un arreglo con los n ángulos correspondientes del polígono bajo la regla:

$$\theta(i) = 2\pi\frac{i}{n} \tag{1}$$

Una vez se tienen estos ángulos, se procede al ciclo de orientarse y avanzar para cada uno de los n puntos considerando el centro del polígono en (5.5, 5.5), y definiendo las posiciones X y Y en función de cosenos y senos, respectivamente (y un radio de 4.5).

$$Coordenada_x = 4.5cos(\theta) \tag{2}$$

$$Coordenada_y = 4.5sin(\theta) \tag{3}$$

Por último, simplemente se ajustan los puntos con un desplazamiento vertical y horizontal para mantener nuestro centro en (5.5, 5.5) sumando 5.5 a cada coordenada.

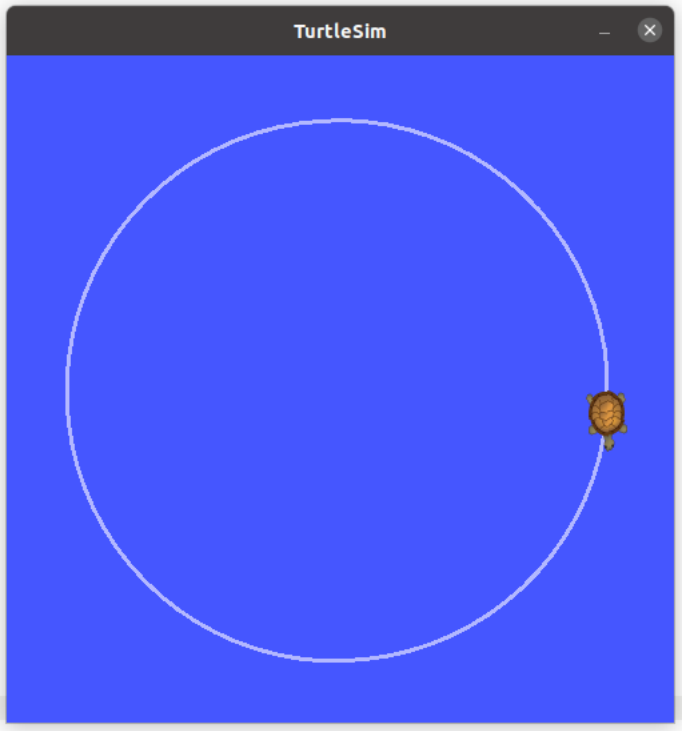

Figura 2: Trayectoria circular.

3. Conclusiones

ROS es un marco de trabajo sumamente útil para la simulación de software de robótica. Concretamente, el entorno especializado de Turtlesim nos permite trabajar con agentes móviles de una manera cercana a la realidad.

En este caso, se observó que la tortuga cuenta con dos movimientos principales: rotación alrededor del eje Z, y movimiento alrededor de X (en este caso X se refiere al frente de la tortuga). Esto levanta la necesidad de trabajar con referencia a un set de coordenadas que describa al entorno. Esto se logra a través de la publicación de la posición y, más tarde, la suscripción a dichas publicaciones.

Este sistema publicación/suscripción es útil para la simulación de varios agentes móviles en un mismo entorno de Turtlesim, pues es posible para las tortugas conocer las posiciones del resto de agentes y tomar decisiones en consecuencia.

Por último, vale la pena mencionar que el código para moverse de un punto a otro utilizado funciona a través de un controlador de velocidad con base en la distancia euclidiana entre el punto actual y el deseado multiplicada por una constante. Aún queda por optimizar dicha constante para el ejercicio del círculo debido a que toma demasiado tiempo en completar la figura.

4. Enlaces externos

`https://github.com/JuanOceguera/PrincipiosDeMecatronica`

`https://github.com/Jesus669/PrincipiosMecatronica`

Referencias

[1] Robot Operating System, About ROS, ROS, 2021. [En línea]. Disponible en: https://www.ros.org/about-ros/. [Visitado el: 23- Abr -2021].

[2] University of Pennsylvania, Mobile Agent Computing, A White Paper, Computer and Information Science U-Penn, 1997. [En línea]. Disponible en: https://www.cis.upenn.edu/ bcpierce/courses/629/papers/Concordia-WhitePaper.html. [Visitado el: 23- Abr -2021].

[3] J. Faust and D. Thomas, Turtlesim, ROS Wiki, 2020. [En línea]. Disponible en: http://wiki.ros.org/turtlesim. [Visitado el: 23- Abr -2021].