INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

"Proyecto 1: Generación de trayectorias en ROS."

Reporte que presenta:
Juan Alberto Muro López (000137592)

Materia:
Robótica: SDI-11911

Profesor:
Dr. Marco Morales Aguirre

México DF. 1 de septiembre de 2016

Índice

Índice de figuras

Índice de cuadros

1. Objetivo General

- Aprender los fundamentos de ROS y aplicarlos para construir nodos con capacidad de publicar y suscribirse a tópicos.

2. Introducción

Este trabajo aborda el problema de generar trayectorias para un robot. Es una primera aproximación al uso de ROS para comunicarse con un robot autómata. Se requirió de un manejo básico de Linux y de la herramienta Turtlesim para el manejo de nodos.

3. Marco Teórico

3.1. ¿Qué es ROS?

ROS (Sistema Operativo Robótico) es un framework para el desarrollo de software de robots que provee la funcionalidad de un sistema operativo en un clúster heterogéneo. ROS se desarrolló originalmente en 2007 bajo el nombre de switchyard por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte al proyecto del Robot con Inteligencia Artificial de Stanford (STAIR. Desde 2008, el desarrollo continua primordialmente en Willow Garage, un instituto de investigación robótico con más de veinte instituciones colaborando en un modelo de desarrollo federado. Esta unificación de lenguaje de programación fue muy útil ya que ayuda a que diferentes personas puedan tabajar en el mismo proyecto y facilitar el diseño y construcción ya que normalmente es un equipo de varios especialistas los que están a cargo de un sólo robot.

3.2. Conceptos de ROS aplicados en la práctica

Los conceptos utilizados fueron turtlesim, roscpp, rospack, workspace y node. El primero es un simulador que nos ayuda a visualizar y a practicar los conceptos básicos de ROS antes de aplicarlos a un robot, lo cual podría llegar a ser peligroso si no se tiene suficiente experiencia. Tutlesim funciona a base de nodos, que son ejecutables que sirven para comunicarse entre sí y enviar o recibir información relevante al funcionamiento del robot. Como la programación es a base de Linux, utiliza paquetes dentro de un workspace para poder ejecutar los programas. El lenguaje de programación para los ejecutables es C++ por lo que se utiliza roscpp para crear un ambiente

de trabajo en donde los programas se crean con C++ para después ejecutarse desde la terminal.

3.3. Trayectorias de robots

Una de las finalidades más básicas en un robot es el tener la capacidad de moverse a algún punto específico con una pose definida. Normalmente esto depende del diseño y de la cinemática del robot y de su entorno, pero al ser esta práctica únicamente elaborada sobre un simulador, sólo es importante considerar el espacio donde debía moverse, el cual es una cuadrícula de 11 x 11. A este tipo de movimientos de un robot se les conoce como trayectorias, las cuales tienen un punto de inicio y un punto destino. Normalmente hay varias maneras o rutas para hacer este recorrido y la decisión de cual tomar se hace con base en distintos criterios como velocidad, distancia, tiempo, estabilidad, etc. Existen varios métodos para realizar esta trayectoria, pero el método utilizado en este proyecto fue el de un controlador proporcional el cual nos da trayectorias suaves y bien definidas siempre y cuando el tiempo de ejecución no sea muy corto, de lo contrario se necesitan controladores más sofisticados como un controlador PID para poder contrarrestar el sobretiro que se genera cuando se escogen tiempos de ejecución muy pequeños en relación a la distancia recorrida.

4. Experimentos

Los experimentos realizados fueron divididos en varias etapas:

- Primero se instaló una máquina virtual con todos los paquetes necesarios para poder ejecutar Turtlesim de manera adecuada.
- Después se realizaron los tutoriales del libro para aprender a utilizar los comandos básicos de ROS.
- Más adelante se realizaron los tutoriales encontrados en la Wiki de Turtlesim, los cuales utilizan un controlador proporcional para que la tortuga complete una trayectoria.
- Por último se le hicieron los cambios necesarios a este último programa para que se pudiera modificar el tiempo de ejecución, presentándose así, nuevos problemas que antes no estaban presentes. Esto se debió a que al utilizar un controlador proporcional, no es tan fácil controlar la salida del proceso si los valores son muy pequeños, por lo que se debe utilizar un controlador más sofisticado.

5. Conclusiones

Se logró aprender los fundamentos de ROS y se aplicaron para poder construir nodos capaces de publicar y de suscribirse a tópicos. Además se aprendió un poco más acerca de los fundamentos de Ubuntu y se repasó el material visto en clase acerca de poses y trayectorias. El proyecto resultó ser más complicado de lo que parecía pues los conocimientos acerca de la programación en Ubuntu eran muy básicos, además al intentar utilizar un método de control de sistemas en lugar de una solución lineal, aparecieron problemas que son más difíciles de solucionar y para los cuales se requiere un conocimiento más avanzado acerca del lenguaje de programación utilizado.

6. Bibliografía

Referencias

[1] Jason M. O'Kane. A Gentle Introduction to ROS. `https://cse.sc.edu/~jokane/agitr/`, 28 08 2016

[2] Colaboradores de Wikipedia. Robot Operating System. Wikipedia. The free encyclopedia. `https://en.wikipedia.org/wiki/Robot_Operating_System`, 31 08 2016