Práctica No. 5 Robot Operating System - ROS

Departamento Académico de Sistemas Digitales
Instituto Tecnológico Autónomo de México
Primavera 2019

JEAN PAUL VIRUEÑA ITURRIAGA 155265
FABIÁN ORDUÑA FERREIRA 159001

Abstract - En esta práctica se aprendió a usar ROS y sus componentes básicos al igual que un simulador para implementar programas que comuniquen nodos dentro de ROS y realicen funciones como intercambiar mensajes entre nodos, mover un robot dentro de una simulación, intercambiar mensajes desde una tarjeta Arduino e intercambiar mensajes desde distintas computadoras con el fin de familiarizarse con ROS y aprender a usar sus componentes y funcionalidades.

I. INTRODUCCIÓN

En la actualidad, la tecnología es un recurso que tenemos para optimizar nuestras actividades diarias. Los robots son un gran ejemplo de esto ya que se han utilizado desde el siglo pasado para realizar tareas de forma automatizada que el ser humano puede realizar. Se pueden encontrar en todos lados ya que se pueden encontrar en entornos de manufactura, de ensamblaje, embalaje, transporte, todo tipo de exploración, cirugías, armamento, producción industrial [10] por decir algunos ejemplos. Con el paso del tiempo, la robótica ha mejorado considerablemente haciendo que las funciones de los robots sean más rápidas, más precisas y más eficientes.

Para crear el diseño de un robot, existen sistemas operativos diseñados para que un robot genere sus funcionalidades mediante librerías. Un ejemplo de estos sistemas operativos es Robot Operating System (ROS). Por tal motivo, los objetivos de esta práctica son implementar 4 programas que realicen funciones utilizando *roscore* y otros componentes de ROS como *Node Handlers* y nodos de tipo *subscriber* y *publisher* modificando sus tópicos para tener un primer acercamiento a dicho sistema:

- El primero busca que cuando un publicador le mande un mensaje a un subscriptor, este último le conteste con un mensaje de recibido para el publicador pueda imprimir en la termina el mensaje de recibido.
- El segundo consisten en hacer que en Turtlesim una tortuga pueda moverse hacia arriba, hacia abajo, a la derecha y a izquierda con cuatro teclas del teclado.
- En el cuarto se quiere utilizar ROS desde Arduino con el fin de que la tarjeta Arduino pueda escuchar los mensajes que ROS le envió y prensa un led para avisar que recibió el mensaje y responder el mensaje.
- El cuarto se diseñó para poder hacer que se controle a una tortuga en Turtlesim que se encuentre corriendo en una computadora desde otra computadora

Este reporte está organizado en distintas secciones, comenzando con un marco teórico, en el que se presenta la información relevante y de importancia para una mejor comprensión de lo que aquí se aborda; seguido del desarrollo, donde se presenta la manera en que se trabajó para la obtención de los resultados; la sección de resultados, en la que se presenta un análisis de lo obtenido y la sección de conclusiones.

II. MARCO TEÓRICO

Robot Operating System (ROS) es un framework que provee librerías, herramientas y convenciones para desarrollar software para robots. ROS contiene funciones para crear robustos y complejos comportamientos para plataformas robóticas [6 y 8]. Dentro de ROS está *roscore*, una colección de nodos y programas que contiene prerrequisitos de un sistema basado en ROS. *Roscore* es ejecutado desde la terminal de una computadora que puedan correr los nodos de ROS dentro de la computadora [3].

Un nodo en ROS es una término utilizado para los ejecutables que están conectados a una red, también llamada grafo, de ROS para comunicar con otros nodos usando tópicos, usualmente un sistema de control de los robots está compuesto de diferentes tópicos [4]. Los tópicos (o *topics*) son *buses*,

sistemas digitales que transportan información entre dos componentes de un sistema computacional, con un nombre para intercambiar mensajes de manera unidireccional. Para que dos o más nodos se puedan comunicar entre ellos, es necesario que ambos anuncien su tópico y su tipo de datos (todos los mensajes de un tópicos tienen que tener el mismo tipo de dato) [7].

En esta práctica se usaron dos tipos de nodo: *publishers y subscribers*. Los nodos *publisher* o publicadores se encargan de enviar mensajes en broadcast de un tipo de dato a un tópico en particular [1]. Los nodos *subscriber* o suscriptores se encargan de esperar mensajes de cierto tipo de dato a un tópico [5]. Para poder realizar la comunicaciones entre nodos es necesario representarlos en ROS, para eso se utiliza la clase NodeHandle [4 y 5].

Es muy frecuente querer simular robots dentro de una computadora mediante una interfaz gráfica. Es por eso que existen simuladores en ROS como Turtlesim. Turtlesim es un simulador que ya viene incluido en ROS hecho para enseñar las funciones básicas de ROS y sus paquetes. La simulación consiste en una ventana gráfica que muestra un robot con forma de tortuga que se puede mover a lo largo de la ventana usando comandos de ROS o el teclado [2 y 9].

III. DESARROLLO

Para cumplir con los objetivos de esta práctica se realizaron cuatro ejercicios para familiarizarnos e implementar programas en el framework de desarrollo ROS (Robot Operating System). Consistió en implementar el código en C++ para ejecutarlo desde terminal, emplear una herramienta llamada TurtleSim para ampliar los conocimientos del lenguaje, conectar ROS con arduino y probar conexión entre computadoras usando los programas previamente mencionados. Los ejercicios se describen a continuación:

Para el primer ejercicio se tomaron los archivos proporcionados en el repositorio https://github.com/garygra/PM_pract_5, de allí empleamos dos archivos de forma principal: ejemplo_pub.cpp y el ejemplo_sub.cpp. En estos, se encontraba código que permitía establecer comunicación unilateral entre dos agentes empleando el framework ROS. Así que primero analizamos la lógica que se empleaba para poder establecer la comunicación unilateral y posteriormente modificamos el código para que pudiéramos establecer una comunicación bilateral, misma que respondería al primer emisor al recibir un mensaje.

Para el segundo ejercicio fue necesario emplear la herramienta TurtleSim, misma que ya se encontraba en la computadora del laboratorio. Primero ejecutamos la herramienta desde línea de comando para visualizar el despliegue inicial. Después, recopilamos información de cómo implementar ciertos movimientos para poder llevar a cabo esta tarea y creamos bloque de ros para que dada una entrada numérica, en este caso con los números dos, cuatro seis y ocho, sea capaz de desplazarse en cierta dirección.

Para el tercer ejercicio, se empleó el framework ROS junto con un arduino. Primero, usamos la librería roserial ya que es necesaria para poder establecer comunicación con el arduino. Implementamos un método, tomando como base los empleados del repositorio previamente mencionado, que pudiera establecer comunicación para el arduino. Una vez que esto se realizó, gracias a los conocimientos que adquirimos en previas prácticas, escribimos un método para que pudiera prender un led cada vez que se recibiera un mensaje.

Para el cuarto y último ejercicio, se necesitaban emplear ciertos métodos de los ejercicios descritos previamente para poder realizar una conexión entre computadoras de tal forma que se pudiera mover la tortuga implementada en el segundo ejercicio.

IV. RESULTADOS

En esta práctica, en lo que concierne al primer ejercicio fuimos capaces de aprender y comprender de forma básica la manera en que el framework ROS funciona; por un lado, al analizar y modificar el repositorio que se nos proporcionó para poder pasar de una conexión unilateral a una bilateral, y por otra con ayuda de internet para aclarar las dudas que nos surgieron. Además de eso, aprendimos qué comandos se necesitaban para

poder correr los programas desarrollados con el apoyo del profesor y leyendo documentación.

En cuanto al segundo ejercicio, tuvimos ciertas dificultades ya que al principio no pudimos ejecutar bien turtlesim porque había un archivo extra que lo impedía. Sin embargo, fuimos capaces de generar el código para trabajar con turtlesim gracias a las recomendaciones que miembros de otros equipos nos proporcionaron.

Para el tercer ejercicio, pusimos en práctica los conocimientos que teníamos previamente de arduino en cuanto asignar puertos, prender y apagar leds, a la estructura general. Pero, al tratar de implementar por primera vez ROS con arduino para establecer comunicación, tuvimos un poco de dificultades por desconocer la forma en que se debía implementar. A pesar de ello, logramos implementar comunicación y prender un led cuando esto ocurría.

En lo que concierne al último ejercicio, en el que buscamos realizar una conexión entre computadoras de tal forma que se pudiera mover la tortuga construida en el ejercicio dos, no pudimos concretarlo de forma exitosa. Consideramos que esto ocurrió debido a la forma que tratamos de asignar la conexión entre las computadoras.

V. CONCLUSIONES

Jean Paul Virueña Iturriaga
Fue muy interesante este primer acercamiento a ROS y conocer cómo funciona. La curva de aprendizaje me pareció muy tediosa ya que no fue tan intuitivo entender los distintos componentes del código. También descubrí una herramienta que tiene mucho potencial y me alegraría seguir trabajado con ella en el futuro.

Fabián Orduña Ferreira
Gracias a esta práctica aprendí a trabajar de forma básica con el framework ROS para desarrollar ciertas tareas. En un principio, me pareció un poco complicada la estructura que sigue ROS ya que de primera intención no es muy intuitivo y puede llegar a ser un poco confuso. Asimismo, aprendí la forma en que se puede conectar arduino con ROS y conocí la forma en que se puede emplear turtlesim, con el que considero se pueden poner en práctica proyectos muy interesantes, sobre todo si se combinan como lo que se buscó de la comunicación entre computadoras.

VI. ROLES

Para el desarrollo de las actividades, de las que mostramos previamente los resultados, ambos miembros del equipo colaboramos de forma conjunta. En lo que a este documento concierne, Paul se enfocó en mayor proporción al abstract, a la introducción y al marco teórico, mientras que Fabián se enfocó más al desarrollo y a los resultados.

VII. FUENTES DE CONSULTA

[1] Clearpath Robotics. "ROS 101: Creating a Publisher Node." Clearpath Robotics. October 12, 2017. Accessed April 25, 2019. https://www.clearpathrobotics.com/blog/2014/09/ros-101-creating-node/.

[2] Packtpub. "Turtlesim, the First ROS Robot Simulation." Packt Subscription. Accessed April 25, 2019. https://subscription.packtpub.com/book/hardware_and_creative/9781782175193/1/ch01lvl1sec14/turtlesim-the-first-ros-robot-simulation.

[3] ROS.org. "Rocore." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/roscore.

[4] ROS.org. "Nodes." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/Nodes.

[5] ROS.org. "Writing a Simple Publisher and Subscriber (C)." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(c).

[6] ROS.org. "Topics." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/Topics.

[7] ROS.org. "ROS." Ros.org. Accessed April 25, 2019. https://wiki.ros.org/.

[8] ROS.org. "About ROS." Ros.org. Accessed April 25, 2019. https://www.ros.org/about-ros/.

[9] ROS.org. "Turtlesim." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/turtlesim.

[10] Wikipedia. "Robot." Wikipedia. April 23, 2019. Accessed April 25, 2019. https://en.wikipedia.org/wiki/Robot.