Práctica 5: Laboratorio de Principios de Mecatrónica

Rebeca Baños García, Leonardo García Osuna

Resumen—En esta práctica se repasó el lenguaje de C++ y su compilación, además se aprendió a utilizar ROS desde la terminal e integrar ROS con Arduino. También desde la terminal se aprendió a utilizar Turtlesim como simulador.

1. INTRODUCCIÓN

Esta práctica fue importante para nosotros ya que ROS nos ayuda a poder entrar a la programación de robótica más a fondo, es un herramienta excepcional que permite la comunicación, mediante publicaciones y suscripciones a *tópicos*, entre procesos ocurriendo en paralelo. ROS ayudó también a reforzar nuestro manejo de la terminal para desde ahí editar el código necesario. La integración de ROS y Arduino pone las bases para poder controlar desde ROS robots físicos. Por último fue importante saber establecer la conexión de ROS para múltiples computadoras.

2. MARCO TEÓRICO

Describimos brevemente los componentes principales para la realización de la práctica.

ROS (Robot Operating System) es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo –en realidad no es un sistema operativo, sino *middleware*–. ROS provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los *nodos*, que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros. La librería está orientada para un sistema UNIX (Ubuntu (Linux)) aunque también se está adaptando a otros sistemas operativos como Fedora, Mac OS X, Arch, Gentoo, OpenSUSE, Slackware, Debian o Microsoft Windows, considerados como 'experimentales'. ROS tiene dos partes básicas: la parte del sistema operativo, ros, como se ha descrito anteriormente y ros-pkg, una suite de paquetes aportados por la contribución de usuarios (organizados en conjuntos llamados pilas o *stacks*) que implementan funcionalidad tal como localización y mapeo simultáneo (SLAM), planificación, percepción, simulación, etc. ROS es software libre bajo términos de licencia BSD. Esta licencia permite libertad para uso comercial e investigador. Las contribuciones de los paquetes en ros-pkg están bajo una gran variedad de licencias diferentes.

Para poder ejecutar los nodos –procesos, programas– de ROS correctamente y habilitar la comunicación entre éstos mediante las publicaciones y suscripciones a tópicos como es debido, es necesario tener una instancia de *roscore* ejecutándose –en la computadora principal, en caso de usar varias, como en el caso de la última parte de la práctica. tener un roscore corriendo.

- **Turtlesim:** Turtlesim es un simulador que nos permite de forma simple aprender los conceptos básicos de ROS, es un paquete que forma parte de la instalación base de ROS. El simulador es una ventana gráfica que muestra un robot en forma de tortuga, los robots utilizados en Turtlesim se llaman turtlebots. Esta tortuga recibe mensajes del tipo "Twist" que es el tipo de mensajes estándar que se utiliza para controlar un robot.
- **rostopic:** Provee herramientas para la Terminal para tener información o hacer publicaciones a tópicos a manera de un CLI para los robots. Herramienta importante para depurar problemas cuando hay muchas partes interconectadas y es necesario verificar funcionalidad nodo por nodo.
- **rosserial:** Es una librería para el IDE de Arduino que provee el protocolo de comunicación de ROS a través del puerto UART del Arduino. Se debe importar al *sketch* para poder acceder al roscore.

3. DESARROLLO

Primero para la parte de ROS instalamos la librería como indicaba en la práctica y corrimos el ejemplo correspondiente en conjunto con el roscore. Para la parte de cambiar el mensaje mostrado, tuvimos que abrir los códigos del ejemplo, es decir el ejemplo subscriptor y el publicador y editamos la parte que arrojaba el mensaje deseado cambiando también el tipo de mensaje a String.

Para la parte de Turtlesim, verificamos que ROS tuviera instaladas las librerías necesarias en la computadora y poder correr el simulador debidamente, al correrlo apareció la ventana donde mostraba nuestra tortuga. Finalmente abrimos la terminal donde marcaban las coordenadas de la tortuga para moverse y cambiamos varias veces las coordenadas x y z, ya que x hacia que avanzara la tortuga y z hacía que girara.

Para la parte de Arduino, bajamos la libreria necesaria llamada rosserial para que ROS se pudiera comunicar con el Arduino. En el programa de Arduino también tuvimos que importar la librería necesaria para que se pudiera tener la conexión correcta con ROS. Posteriormente, en arduino escribimos el código necesario para poder encender el LED correspondiente. En esta parte tuvimos que escribir código tanto para la parte del suscriptor y del publicador de ROS, se requirió crear variables globales para el nodo y el publicador. Para la parte del suscriptor utilizamos las funciones necesarias para que el resultado fuera el esperado. Finalmente compilamos y mandamos el programa escrito en arduino a nuestra tarjeta y en la terminal escogimos el puerto donde ésta estaba conectada para después ejecutar roscore y el resto de los nodos.

Finalmente, para comunicar las dos computadoras para oder ejecutar Turtlesim en una mientras la controlamos de otra, primero nos aseguramos que el IP de cada máquina estuviera bien escrito en ambas, posteriormente corrimos los comandos necesarios para que se identificaran ambas compuadoras entre la maestra y la que correría a la tortuga. Finalmente ya que estaban comunicadas corrimos Turtlesim para visualizar los resultados.

4. RESULTADOS

La primera parte de la práctica fue sencilla ya que teniamos familiaridad con la terminal. Al correr el ejemplo pudimos ver como era necesario tener roscore ejecutándose para poder ver el resultado al esribir un número entero. Para modificar el mensaje que resultaba en el publicaro, fue algo difícil para nosotros ya que al editar el código dejaba de funcionar el programa. Nuestros errores fueron la mala importación de la libreria de Strings que la necesitabamos y el no cambiar por completo todas las variables int a String para que se desplegara el mensaje deseado. Al corregir estos errores, nuyestro programa compilo bien, pero el mensaje resultante lo arrojaba después de varios segundos; para corregir esto, cambiamos la manera en la que se ciclaba el programa y así logramos que el mensaje se desplegara cada que le insertaramos un número. Para la parte de Turtlesim fue relativamente fácil, la única complicación que tuvimos fue al inicio de importar la libreria con el tópico que contenía el simulador, pero una vez que la encontramos fue fáci encontrar las coordenadas y cambiarlas para que la tortuga se moviera en la dirección que le indicamos. En la conexión entre ROS y Arduino, tuvimos mucho cuidado ésta vez de importar las librerias necesarias desde un principio tanto en ROS como en Arduino. Finalmente para escribir el código necesario nos basams en un ejemplo ya incluído en Arduino, por lo que nos fue más fácil entender cómo se programaba el publicador y el subscriptor en Arduino. Para encender el LED, tuvimos etsar seguros primero de compilar bien el código escrito en Arduino así como mandarlo a la tarjeta y ya hecho esto ejecutar ROS junto con roscore para poder ver los resultados. Logramos los resultados esperados desde el inicio, al mandar un mensaje tipo char el LED indicado de la tarjeta se encendía por tres segundos, para finalmente enviar un mensaje diciendo qeu se había recibido un mensaje.

Para la comunicación entre dos computadoras tuvimos ciertos conflictos ya que al principio no se lograban comunicar entre ellas; esto era porque abriamos la terminal varias veces y no nos dimos cuenta que al hacer esto el IP de cada computadora se reseteaba y no guardaba cual era la maestra y cual la que ejecutaría la simulación. Al darnos cueta del problema, solo abrimos la terminal una vez para que no se reseteara el IP y se lograron comunicar entre ellas. Terminado esto simplemente corrimos Turtlesim como anteriormente lo hicimos en una sola computadora moviento los ejes x para avanzar y el z para girar.

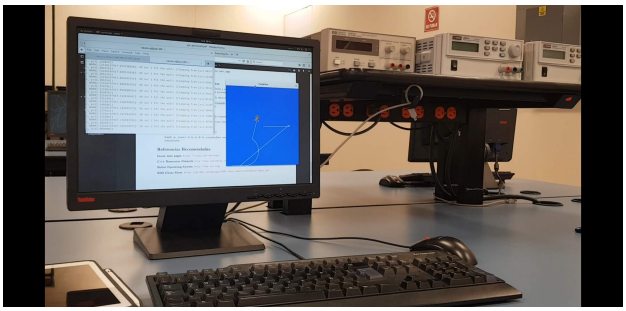

Figura 1. Turtlesim

5. CONCLUSIONES

- **Rebeca:** En esta práctica pude tener más contacto con la terminal y ver que muchas de las cosas que hacemos se logran hacer directamente desde la terminal. También fue interesante utilizar un simulador como Turtlesim para darme una idea de como funcionan los robots y bajo que indicaciones lo hacen. El tener un acercamiento a ROS me hizo saber que todavía hay cosas que me faltan por aprender, pero es bueno conocer más lenguajes y las funciones que éstos nos brindan para diferentes aplicaciones.
- **Leonardo:** Existen muchas soluciones alternas a ROS para resolver las necesidades de comunicación entre sistemas, pero ROS, si bien es un poco árido al principio, sin duda se ha vuelto muy popular dado que resuelve con versatilidad y portabilidad las necesidades de comunicación entre sistemas y subsistemas, requerido para el funcionamiento de uno o varios robots. Personalmente, tenía experiencia mínima modificando nodos existentes en lenguaje python para vehículos aéreos no tripulados, pero construir un workspace, crear un paquete desde cero y ejecutarlo fueron actividades nuevas y sin duda útiles.

6. ROL O PAPEL

- **Rebeca:** Consultaba ligas, leía comandos para la terminal, manejo de la terminal.
- **Leonardo:** Manejo de la terminal y de arduino. Investigar elementos faltantes para que el funcionamiento fuera correcto.

7. FUENTES CONSULTADAS

- **ROS:** http://wiki.ros.org/es https://es.wikipedia.org/wiki/Sistema_Operativo_Rob%C3%B3tico

- **ROS Tutorials:** https://wiki.ros.org/ROS/Tutorials/
- **rostopic Wiki:** https://wiki.ros.org/rostopic
- **Turtlesim:** https://openwebinars.net/blog/turtlesim-simulador-2d-para-robots-diferenciales-en-ros/
- **ROS Troubleshooting** https://wiki.ros.org/ROS/Troubleshooting
- **ROS Network Setup** https://wiki.ros.org/ROS/NetworkSetup
- **ROS Network Setup** https://wiki.ros.org/rosserial_arduino/Tutorials