Robot Operating System - ROS

Daniela Arely Morales Hernández 142976
Ana Carolina Sandoval Mejía 152808
Stephanie Lizeth Malvaes Diaz 135515

I. Introducción

En esta práctica aprendimos cómo utilizar ROS (Robot Operating System) y también cómo conectarlo con Arduino. ROS provee librerías y herramientas para ayudar a los desarrolladores de Software a crear aplicaciones para robots. También repasamos la programación y compilación en C/C++.

II. Conceptos.

ROS

El Robot Operating System (ROS) es un marco flexible para escribir software robótico. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robótico complejo y robusto a través de una amplia variedad de plataformas robóticas. ROS provee abstracción de hardware, controladores de dispositivos, librerías, herramientas de visualización, comunicación por mensajes, administración de paquetes y más.

Turtlesim

Turtlesim es un simulador que se utiliza para aprender cómo utilizar ROS y sus paquetes. El simulador es una ventaja gráfica que nos muestra un robot en forma de tortuga, la cual recibe mensajes y se mueve de acuerdo a ellos.

III. Desarrollo

I. ROS

En esta parte se nos dio un programa con archivos iniciales y se nos pidió compilarlos y que se probara su funcionamiento. Este programa incluye dos archivos: un subscriptor (*ejemplo_sub*) y un publicador (*ejemplo_pub*). Nosotras implementamos la siguiente funcionalidad: el suscriptor le responde al publicador con un mensaje tipo String cuando un nuevo mensaje es recibido. El publicador lee e imprime a la terminal el mensaje que reciba.

Para esto corrimos los archivos que ya tenían implementadas estas funcionalidades, lo complicado fue saber cómo ejecutar estos archivos de ROS correctamente desde la terminal. A continuación se muestran los comandos utilizados en diferentes terminales para implementar las funcionalidades requeridas:

//
Terminal 1:
roscore //El comando ejecuta la plataforma de ROS.

Terminal 2:
cd repo // nos ubica en el repositorio en donde se encuentra el archivo
catkin_make
source devel/setup.bash
rosrun ejemplo ejemplo_sub

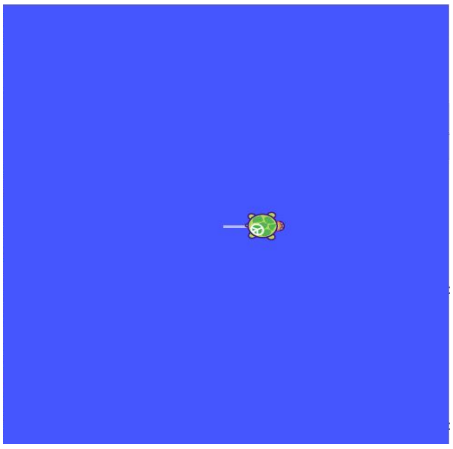
Terminal 3:
source devel/setup.bash
rosrun ejemplo ejemplo_pub
//

II. Turtlesim

En esta parte del laboratorio utilizamos el simulador Turtlesim de ROS en el cual se muestra una tortuga que se mueve con las teclas de flecha, los requerimientos de esta sección fueron implementar un programa que mueva la tortuga utilizando el teclado de la siguiente manera:

- 2: da vuelta en sentido de las manecillas del reloj
- 4: se mueve hacia atrás.
- 6: se mueve hacia adelante.
- 8: da vuelta en sentido contrario a las manecillas del reloj

Para realizar esto se debía crear un archivo en python que indicará las instrucciones anteriores a la tortuga



A continuación se muestra la ejecución del programa de Turtlesim,

III. Arduino-ROS

En esta sección se nos pidió implementar en Arduino un programa con las siguientes funcionalidades:

- Escuchar por nuevos mensajes
- Al recibir un mensaje, prender un LED por 3 segundos
- Al pasar los 3 segundos, enviar un mensaje de respuesta.

También se nos pidió ejecutar Turtlesim en una computadora y controlarla desde otra utilizando el programa ya implementado.

IV. Resultados

I. ROS

Nos tardamos en entender el funcionamiento de ROS ya que no habíamos trabajado anteriormente con él.

Correr las diferentes terminales al mismo tiempo es algo que nos costó bastante descubrir, así como entender los diferentes elementos de ROS, uno de ellos los tópicos a los que ambos elementos estaban suscritos.
La manera en la que se tienen que compilar los elementos de ROS no es muy claro en un inicio, por lo cual en un inicio no logramos comunicarnos elementos debido a que no se habían compilado los archivos

II. Turtlesim

Para esta parte se tuvo que desarrollar un pequeño programa en python que recibiera las instrucciones y publicará la dirección en la que se tenía que desplazar la tortuga. Tuvimos que crear un nuevo archivo y realizar los catkin_make correspondientes.

III. Arduino-ROS

En esta parte nosotros tuvimos que implementar nuestros suscriptores y publicadores que se encargaran de la comunicación de los elementos. No pudimos terminar esta parte debido que la comunicación entre los módulos XBee no se logró realizar debido a que no

aceptaba los comandos que indicaba el tutoria*l.*

V. Conclusiones

Arely: La práctica resultó muy extensa y complicada debido a que no estamos familiarizadas con ROS. Una vez que logramos entender la manera en que se mueve la tortuga fue más sencillo hacer los cambios correspondientes.

Ana Carolina: Esta práctica se me hizo complicada pues nunca había utilizado ROS previamente y personalmente creo que es un programa muy poco amigable para nuevos usuarios pues su ejecución se realiza desde la terminal pero me parece que tanto el proceso para utilizar ROS como los comandos necesarios no están muy explicados en el manual de ROS. Sin embargo, este programa es de gran utilidad para desarrollar software para robótica y además se puede conectar con Arduino para desarrollar aplicaciones muy interesantes.

Stephanie: Una parte complicada de esta práctica fue cambiar de computadoras, pues teníamos que verificar las carpetas constantemente, sin embargo, esto ayudó a tener una mayor comprensión sobre la función y utilidad de las carpetas y métodos utilizados.

VI. Referencias

http://wiki.ros.org/es

http://wiki.ros.org/turtlesim

https://es.wikipedia.org/wiki/Sistema_Operativo_Robótico