Dinámica de Robots.

Practica 2.

Ejercicio con Turtle.

Ingeniería en Mecatrónica.

UPZMG.

Chagoya de la Cruz Levi Hazael.

Viorato Arámbula Alexis Israel.

Vazquez Flavio Antonio.

Hernández Morales Fco Javier.

Gómez Carrillo Christian Salvador.

Introducción.

¿Qué es ROS?

Robot Operating System (ROS) es un middleware robótico, es decir, una colección de frameworks para el desarrollo de software de robots. ROS se desarrolló originariamente 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 (STAIR2). Desde 2008, el desarrollo continuó principalmente en Willow Garage, un instituto de investigación robótico con más de veinte instituciones colaborando en un modelo de desarrollo federado.

A pesar de no ser un sistema operativo, ROS provee los servicios estándar de uno de estos tales como la abstracción del hardware, el control de dispositivos de bajo nivel, la implementación de funcionalidad de uso común, el paso de mensajes entre procesos y el 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 a día de hoy como 'experimentales'.

ROS tiene dos partes básicas: la parte del sistema operativo, ros, y ros-pkg. Esta última consiste en una suite de paquetes aportados por la contribución de usuarios (organizados en conjuntos llamados pilas o en inglés stacks) que implementan las funcionalidades tales como localización y mapeo simultáneo, planificación, percepción, simulación, etc.

Aunque está en proceso de desarrollo de aplicaciones, las áreas que ya incluye ROS son:

- Un nodo principal de coordinación.
- Publicación o subscripción de flujos de datos: imágenes, estéreo, láser, control, actuador, contacto, etc.
- Multiplexación de la información.
- Creación y destrucción de nodos.
- Los nodos están perfectamente distribuidos, permitiendo procesamiento distribuido en múltiples núcleos, multiprocesamiento, GPUs y clústeres.
- Login.
- Parámetros de servidor.
- Testeo de sistemas.

En las futuras versiones se espera que las siguientes áreas vayan apareciendo entre las aplicaciones de los procesos de ROS:

- Percepción
- Identificación de Objetos
- Segmentación y reconocimiento
- Reconocimiento facial
- Reconocimiento de gestos
- Seguimiento de objetos
- Egomoción.
- Comprensión de movimiento
- Estructura de movimientos (SFM)
- Visión estéreo: percepción de profundidad mediante el uso de dos cámaras
- Movimientos
- Robots móviles
- Control
- Planificación
- Agarre de objetos

Objetivo.

Es realizar el ejercicio con turtlesim haciendo que la tortuga se mueva a base de coordenadas, se puede hacer a base de tiempo o que no se pare.

Procedimiento.

En las siguientes imágenes tenemos los pasos y los comandos que se necesitaron para iniciar con la práctica.

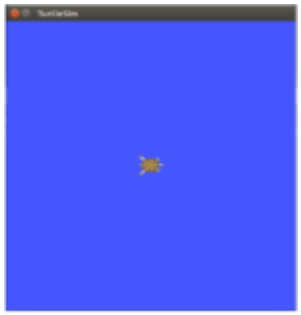
1.- Para comenzar abrimos la terminal e inicializamos ROS con la siguiente linea:

```
roscore
```

2.- Luego para abrir el simulador de turtlesim ejecutamos el siguiente comando en una ventana nueva:

```
rosrun turtlesim turtlesim_node
```

Y se abrirá una ventana como la siguiente.



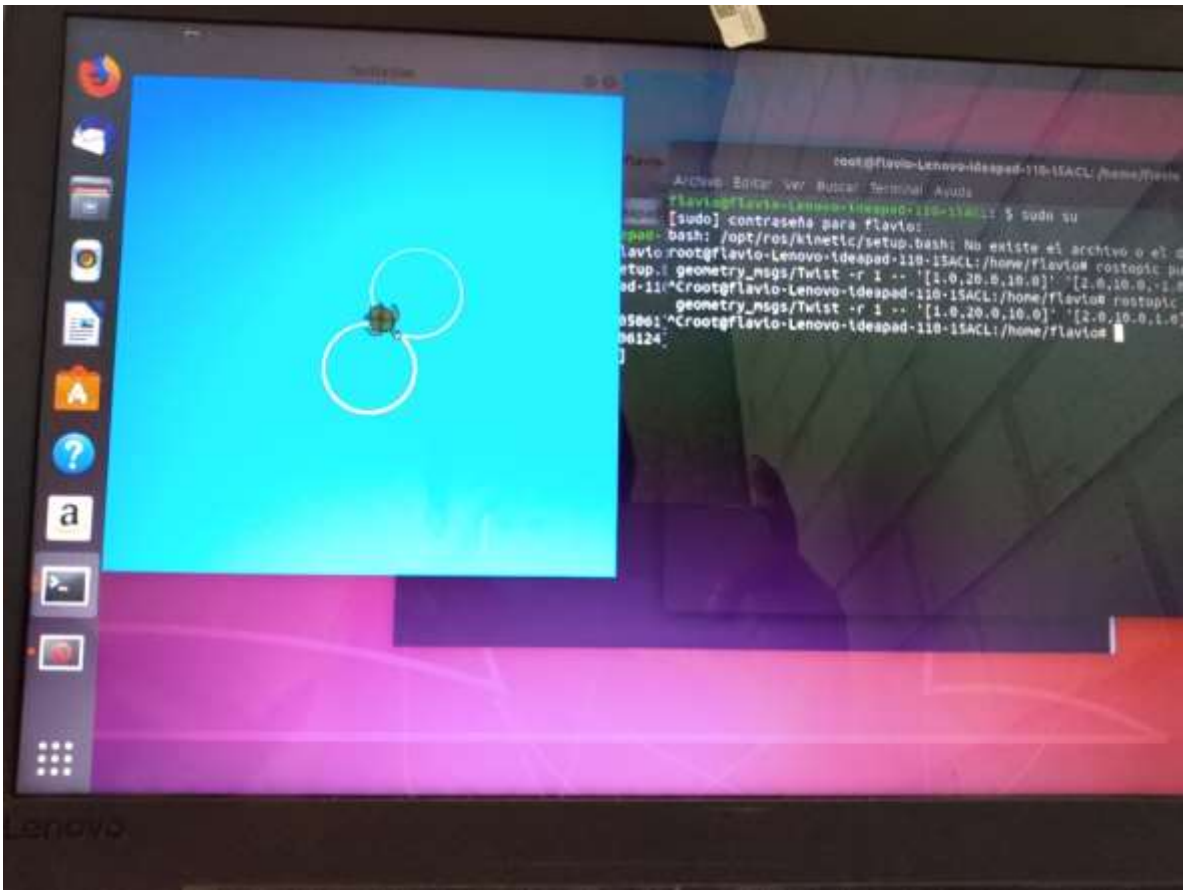
3.- A través del tópico **cmd_vel** pasamos el siguiente comando que enviará un mensaje de tipo Twist al turtlebot, de esta forma el turtlebot va a avanzar con un pequeño giro hacia la izquierda:

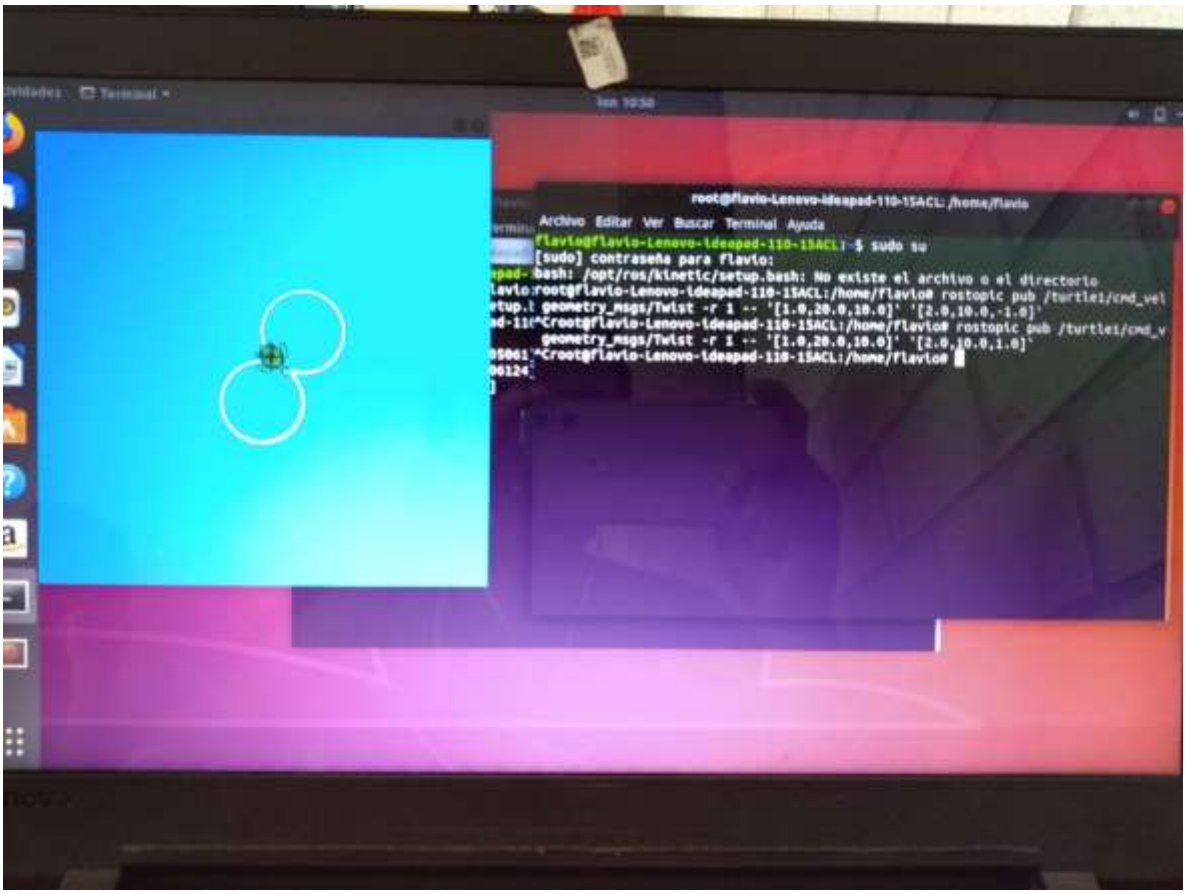
```
rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0,0.0,0.0]'
'[0.0,0.0,0.3]'
```

4.- Para darle un movimiento continuo al turtlebot podemos ejecutar:

```
rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0,0.0,0.0]'
'[0.0,0.0,1.0]'
```

Evidencias.


[sudo] contraseña para flavio:
bash: /opt/ros/kinetic/setup.bash: No existe el archivo o el
root@flavio-Lenovo-ideapad-110-15ACL:/home/flavio# rostopic pu
geometry_msgs/Twist -r 1 -- '[1.0,20.0,10.0]' '[2.0,10.0,1.0
^Croot@flavio-Lenovo-ideapad-110-15ACL:/home/flavio# rostopic
geometry_msgs/Twist -r 1 -- '[1.0,20.0,10.0]' '[2.0,10.0,1.0
^Croot@flavio-Lenovo-ideapad-110-15ACL:/home/flavio#



Conclusion:

En esta practica fue una gran base para entender las coordenadas que debe de tener un robot desde el terminal del pc ya que tuvimos que mover una tortuga mediante coordenadas lo cual no se complicó mucho ya que había una investigación de por medio una ves realizada la practica se comprendió los movimientos básicos con coordenadas