

Laboratorio 2 - Robótica de Desarrollo, Intro a ROS



1. Requisitos:

- Ubuntu versión 20.xx preferible 20.04 LTS (Sirve en máquina virtual aunque se recomienda una instalación nativa).
- Tutorial para instalar ROS Noetic: Disponible en el repo del laboratorio.
<https://github.com/PedroFCardenas/IntroROS>
- MATLAB 2015b o superior instalado en el equipo.
- Robotics toolbox de matworks (Disponible desde la versión 2015 en adelante).
- Tutoriales de ROS (Son excelente fuente de ayuda).

2. Resultados de aprendizaje:

- Conocer y explicar los conceptos básicos de ROS (Rootic Operative System).
- Usar los comandos fundamentales de ROS.
- Conectar nodos de ROS con Matlab.
- Realizar conexion inicial de motores Dynamixel con ROS.

3. Ejercicio en el laboratorio:

a) Familiarizarse con los comandos de mayor uso para la consola de Linux (veáse <http://www.informit.com/blogs/blog.aspx?uk=The-10-Most-Important-Linux-Commands>).

b) Conexión de ROS con Matlab:

Procedimiento:

- Con Linux operando lanzar 2 terminales. En la primera terminal escribir el comando *roscore* para iniciar el nodo maestro.
- En la segunda terminal escribir *roslaunch turtlesim turtlesim_node*.
- Lanzar una instancia de Matlab para Linux (es imperativo que tenga el *toolbox* de robótica de Mathworks).
- Crear un script con el siguiente código:

```
%%
rosinit; %Conexión con nodo maestro
%%
velPub = rospublisher('/turtle1/cmd_vel','geometry_msgs/Twist'); %Creación publicador
velMsg = rosmesssage(velPub); %Creación de mensaje
%%
velMsg.Linear.X = 1; %Valor del mensaje
send(velPub,velMsg); %Envio
pause(1)
```

- Ejecutar las tres secciones del script y observar los resultados con la pose de la tortuga.

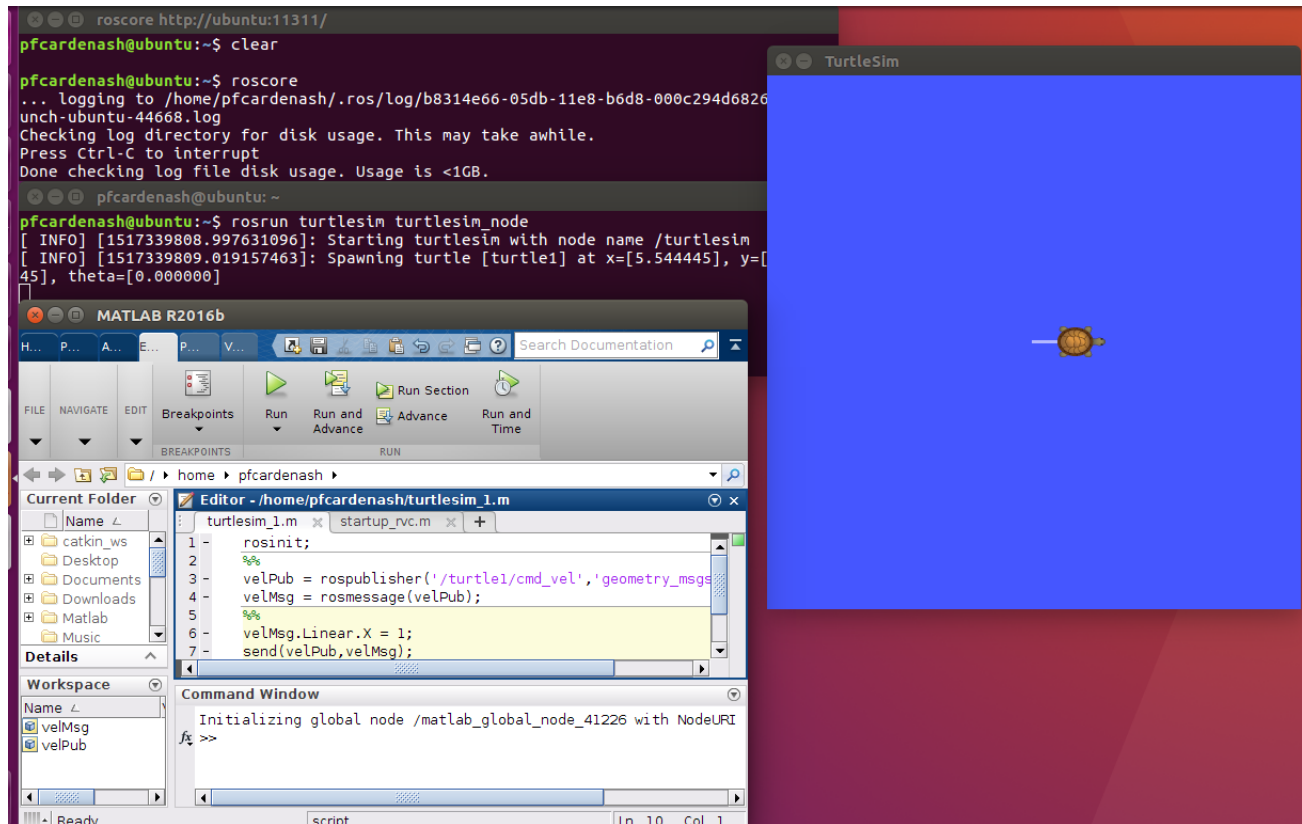
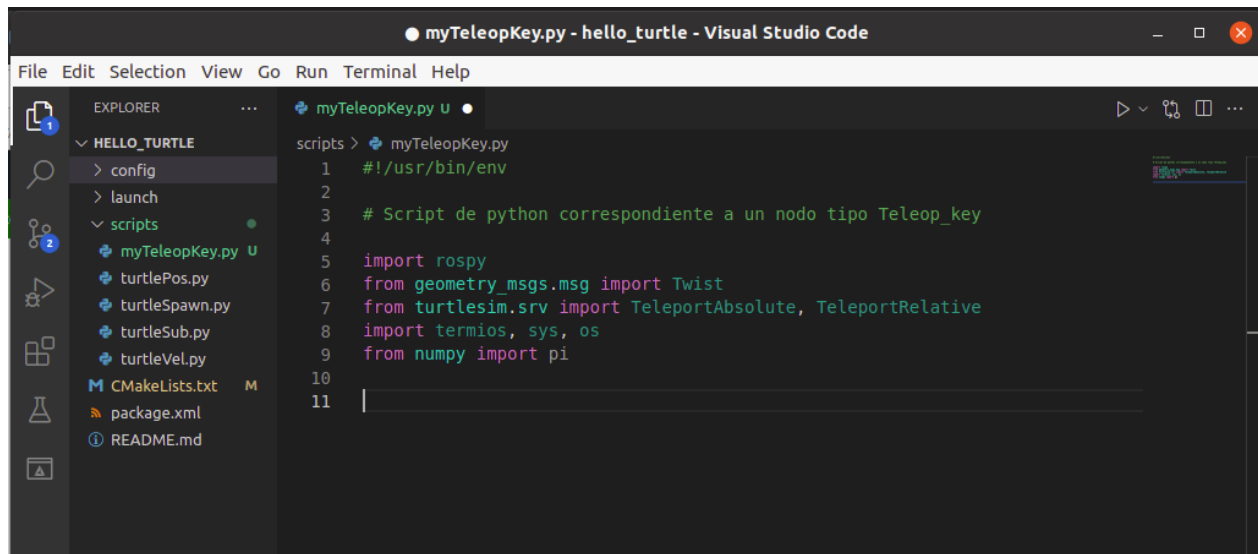


Figura 1: Conexión con Matlab.

- Crear un script en Matlab que permita suscribirse al tópico de pose de la simulación de *turtle1*.
Tip: Usar la instrucción *rossubscriber* con los argumentos ('TOPICNAME', 'MESSAGETYPE'), luego utilizar la opción *latteest message* para captura el último mensaje obtenido.
- Crear un script en Matlab que permita enviar todos los valores asociados a la pose de *turtle1*.
Tip: El tópico *pose* únicamente sirve para suscribirse, consultar los servicios de *turtlesim* para modificar la *pose* de la tortuga.
- Consultar de qué manera se finaliza el nodo maestro en Matlab.

c) Utilizando Python: **Procedimiento:**

- En el paquete *hello_turtle* de ROS [3], en la carpeta de *scripts*, crear un script de Python, de nombre *myTeleopKey.py*



```

1  #!/usr/bin/env
2
3  # Script de python correspondiente a un nodo tipo Teleop_key
4
5  import rospy
6  from geometry_msgs.msg import Twist
7  from turtlesim.srv import TeleportAbsolute, TeleportRelative
8  import termsios, sys, os
9  from numpy import pi
10
11

```

Figura 2: Script de Python

- Escribir un código que permita operar una tortuga del paquete *turtlesim* con el teclado, que cumpla con las siguientes especificaciones:
 - Se debe mover hacia adelante y hacia atrás con las teclas W y S
 - Debe girar en sentido horario y antihorario con las teclas D y A.
 - Debe retornar a su posición y orientación centrales con la tecla R
 - Debe dar un giro de 180° con la tecla ESPACIO

Tip 1: Es necesario lograr que se detecten las teclas presionadas. Hay una forma común de hacerlo con la librería *keyboard* de Python, pero da algunos problemas en Linux. Una alternativa es mediante el código disponible en este enlace: <http://python4fun.blogspot.com/2008/06/get-key-press-in-python.html>.

Tip 2: Los movimientos con las teclas A, S, D y W se pueden conseguir mediante el tópico *turtle1/cmd_vel*, mientras que los movimientos con las teclas R y ESPACIO se pueden conseguir mediante los servicios *turtle1/teleport_absolute* y *turtle1/teleport_relative*. Tener presente que para usar el tópico y los servicios hay que importar a Python los tipos de mensajes y servicios (los comandos *rostopic type [NOMBRE_DEL_TOPIC]* y *rosservice type [NOMBRE_DEL_SERVICE]* pueden ser útiles).

- Incluir el script que se acaba de crear en el apartado de *catkin_install_python* del archivo *CMakeLists.txt*, siguiendo la misma estructura de los otros scripts ya incluidos.
- Lanzar una terminal, dirigirse al directorio del workspace de *catkin* y escribir el comando *catkin_make* para hacer build en el paquete modificado.
- Con Linux operando lanzar 3 terminales. En la primera terminal escribir el comando *roscore* para iniciar el nodo maestro.
- En la segunda terminal escribir *roslaunch turtlesim turtlesim_node*.
- En la tercera terminal dirigirse al directorio que contiene el workspace de *catkin* y escribir *source devel/setup.bash*. Acto seguido escribir *roslaunch hello_turtle myTeleopKey.py*. En este punto, la terminal ya debería estar esperando el ingreso de teclas.

- Observar el movimiento de la tortuga con las teclas A, S, W y D, así como los cambios en la posición instantáneos con las teclas R y ESPACIO.

Observaciones:

- a) Se debe crear un repositorio en GitHub en donde se explique la metodología, los resultados, los análisis y las conclusiones.
- b) **Forma de trabajo:** Grupos de laboratorio.
- c) Los puntos que requieran implementación de funciones deberán tener comentarios de cómo se utilizan, y adjuntar archivos **.m** y **.py** dentro del repositorio.

Referencias

- [1] ROS Tutorials. Disponibles en la wiki de ROS: <http://wiki.ros.org/ROS/Tutorials>
- [2] Robotics - UNAL - LAB2 [Repositorio en GitHub]. Disponible en: https://github.com/fegonzalez7/rob_unal_clase2
- [3] Hello Turtle. [Repositorio en GitHub]. Disponible en: https://github.com/felipe17/hello_turtle