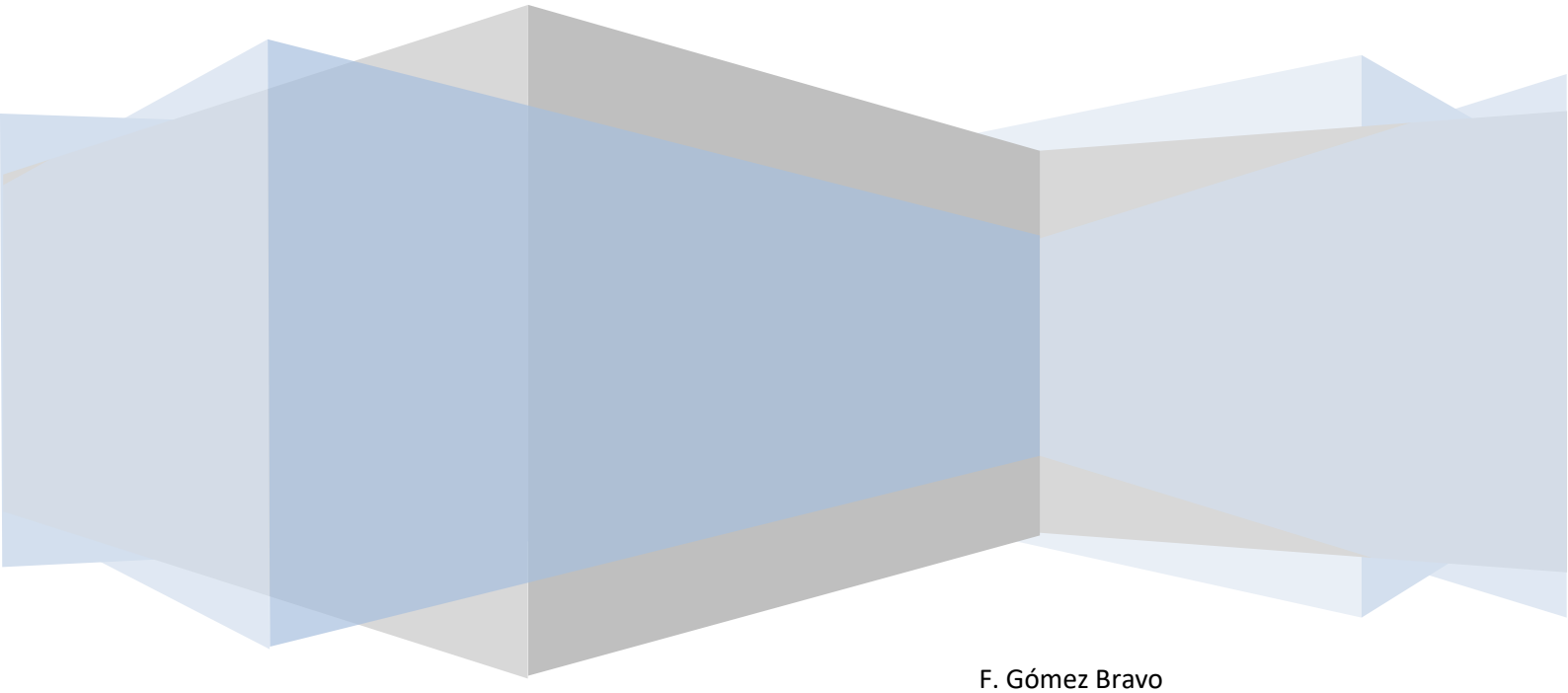
ROBÓTICA
CUARTO CURSO DEL GRADO EN
INGENIERÍA INFORMÁTICA

Universidad de Huelva

Práctica 6

ROS: GAZEBO Y TURTLEBOT



F. Gómez Bravo
R. López de Ahumada
José Manuel Lozano

En esta práctica se introduce el uso del paquete de simulación del robot Turtlebot dentro del entorno de Gazebo

1. Introducción

Gazebo (http://wiki.ros.org/gazebo_ros_pkgs) es un programa de simulación 3D, soportado por ROS y mantenido por Open Robotics (https://www.openrobotics.org/). Gazebo es una plataforma de software de código abierto que puede simular sistemas complejos, dentro de los cuales, para nosotros los más relevantes son robots. Gazebo Integró el motor de física ODE, la representación OpenGL y el código de soporte para la simulación de sensores y el control de actuadores. Permite simular varios robots interactuando entre sí, incluido su entorno formado por diversos tipos de objetos. Desde el punto de vista de su integración en ROS, Gazebo permite simular la estructura de nodos y topics utilizados por los controladores de los robots. De esta forma es posible programar controladores o software para interaccionar con el robot real testándolo primero en el robot simulado en Gazebo. El desarrollo de paquetes de simulación de robots en Gazebo excede este curso. Esta práctica está dedicada a introducir el uso de Gazebo con un modelo del robot con el que trabajaremos en adelante: Turtlebot. El paquete que incluye los nodos necesarios para simular el robot (turtlebot_gazebo) está instalado en la máquina virtual con la que estamos trabajando.

2. Lanzamiento de la simulación de Tutlebot en Gazebo

Para ejecutar el conjunto de nodos que implementan la simulación es necesario lanzar el fichero 'turtlebot_world.launch' de la siguiente forma:

```
turtlebot@rosindigo:~$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

si todo funciona bien aparecerá una ventana como la que se muestra en la figura 1

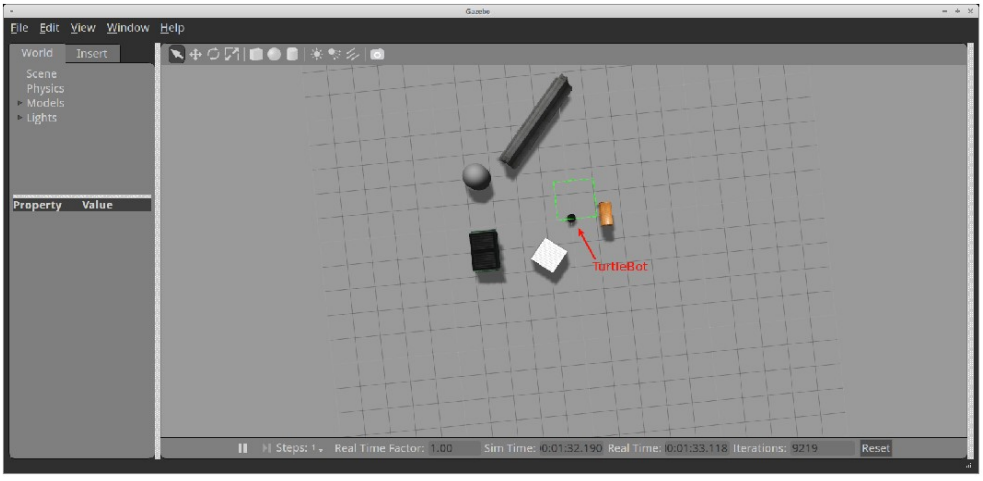


Figura. - 1

Cuando lanzamos un paquete de simulación en Gazebo, se lanza al mismo tiempo un 'roscore' de forma que no será necesario ejecutar este comando para que estén activas todas las funcionalidades de ROS.

El escenario que aparece en la figura 1 es el escenario que se representa por defecto. Actuando sobre las distintas posibilidades que ofrece el interfaz de esa ventana pueden añadirse objetos y modificar su configuración espacial, posición y ángulos de Euler.

También es posible cargar otros escenarios que vienen por defecto en el paquete, o descargarse de la página oficial de Gazebo. Para ello basta con configurar convenientemente la variable de entorno 'TURTLEBOT_GAZEBO_WORLD_FILE ' de la siguiente forma:

```
turtlebot@rosindigo:~$ export
TURTLEBOT_GAZEBO_WORLD_FILE=/opt/ros/indigo/share/turtlebot_gazebo/worlds/corridor.
world
```

O bien para cargar un mundo vacío sin obstáculos:

```
turtlebot@rosindigo:~$ export
TURTLEBOT_GAZEBO_WORLD_FILE=/opt/ros/indigo/share/turtlebot_gazebo/worlds/empty.w
orld
```

Si la carga de los elementos no es correcta, es necesario modificar el fichero '*setup.sh*' dentro del directorio /usr/share/gazebo:

```
turtlebot@rosindigo:~$ cd /usr/share/gazebo

turtlebot@rosindigo:~$ gedit setup.sh
```

Modificaremos la línea:

```
turtlebot@rosindigo:~$ export
GAZEBO_MODEL_DATABASE_URI=http://models.gazebosim.org/
```

3. Operando con el robot simulado en Gazebo

Para interaccionar pro primera vez con el robot utilizaremos el nodo para teleoperación con el teclado:

```
turtlebot@rosindigo:~$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

Si todo funciona correctamente, aparecerá en la ventana del terminal lo mostrado en la figura 2:

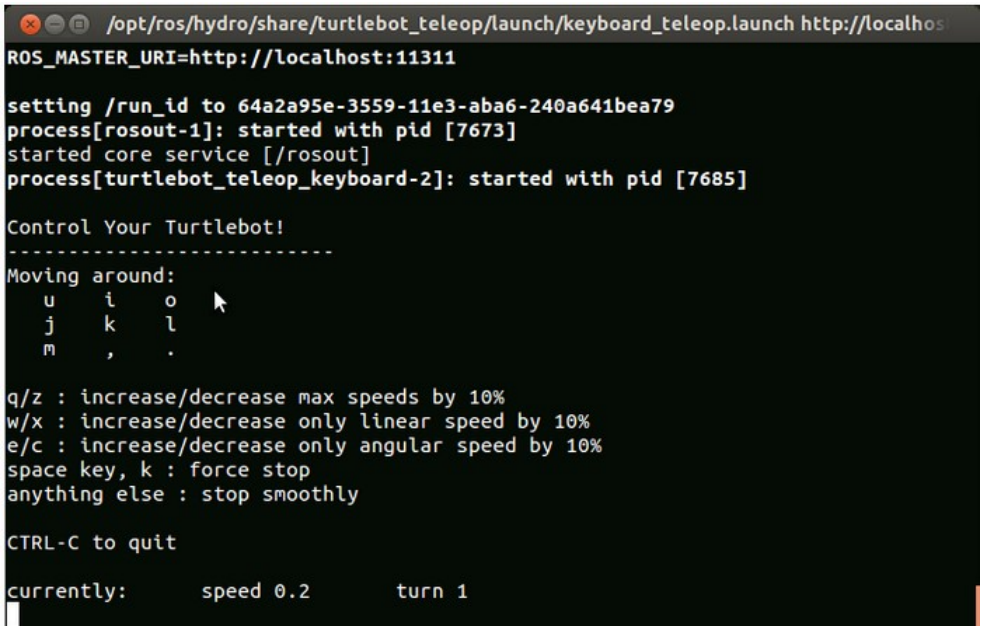


Figura.-2

Pulsando en el teclado la tecla ‘i’ el robot marcha en línea recta hacia adelante, y pulsando la letra k marcha en línea recta hacia detrás. Pulsando las letras ‘u’ ó ‘o’ el robot se mueve hacia delante a la izquierda o a la derecha respectivamente. Lo mismo, pero en movimiento hacia atrás ocurre si se pulsan las letras ‘j’ o ‘l’.

El topic para controlaremos la velocidad de movimiento del robot es: ‘/mobile_base/commands/velocity’

El cual contiene un mensaje tipo: ‘geometry_msgs/Twist’.

Podemos hacer que el robot trace circunferencias publicando en ese topic con la utilidad ‘rostopic’:

```
turtlebot@rosindigo:~$ rostopic pub -r 10 /mobile_base/commands/velocity
geometry_msgs/Twist '[0.15,0.0,0.0]' '[0.0,0.0,0.3]'
```

3.1. Utilización del robot con Rviz

Para visualizar el movimiento del robot en Rviz, una vez hayamos arrancdo el apquete de simulación de turtlebot en Gazebo, abriremos la aplicación Rviz

```
turtlebot@rosindigo:~$ rosrun rviz rviz
```

A continuación, escogeremos /odom como sistema de referencia global. Después añadiremos un objeto TF, y una vez cargado seleccionaremos que solo se muestre el sistema de referencia asociado a la base.

Posteriormente, para representar la trayectoria que recorre el robot ejecutaremos el nodo add_path_vf del paquete control_turtlebot

```
turtlebot@rosindigo:~$ rosrun control_ add_path_vf .py
```

Una vez hecho esto, si movemos el robot , podremos visualizar en Rviz el movimiento del sistema de referencia principal de éste y la trayectoria que recorre.

3.2. Utilización del Joystic para mover el robot

Para conocer la información asignada a cada uno de los ejes del joystick es necesario ver si éste ha sido detectado por el sistema. Podemos hacerlo ejecutando:

```
turtlebot@rosindigo:~$ ls  /dev/input/
```

Normalmente aparece como js0.

Una vez se sabe que el joystick es detectado por el sistema es posible visualizar la información asociada a los ejes y botones de este ejecutando:

```
turtlebot@rosindigo:~$ sudo jstest /dev/input/js0
```

Si comprobamos que con el dispositivo que aparece como js0 no responde al funcionamiento del joystick, buscaremos el puerto jsX que se corresponda con el joystick.

Si finalmente el joystick está configurado en el puerto jsX tendremos que configurar este puerto como dirección por defecto para ROS usando el siguiente comando:

```
rosparam set joy_node/dev "/dev/input/jsX"
```

```
turtlebot@rosindigo:~$ rosparam set joy_node/dev "/dev/input/jsX"
```

Una vez verificada la configuración del Joystick es posible usar este para mover el robot mediante el siguiente procedimiento.

En primer lugar, hay que ejecutar el nodo

```
turtlebot@rosindigo:~$ rosrun joy joy_node
```

Este nodo escribe en el topic '/joy' los distintos valores leídos de los ejes y botones del joystick usando el mensaje tipo sensor_msg/joy.

Posteriormente se debe ejecutar el nodo 'control_joy' que se encuentra dentro del paquete 'control_turtlebot'

```
turtlebot@rosindigo:~$ rosrun control_turtlebot control_joy.py
```

Observe el código que hay dentro del fichero que implementa ese nodo. Compruebe que el nodo se subscribe al topic '/joy' para leer los valores de referencia de las velocidades lineal y angular y publica en '/mobile_base/commands/velocity', para hacer que el robot se mueva de acuerdo a esos valores.

4. Tarea

Para finalizar se le proponen las siguientes tareas al alumnado para que prueben los conceptos explicados en la guía.

1) Usando como guía el nodo programado en 'turtlebot control_joy.py' programe un nodo que arranque el movimiento del robot realizando un arco de circunferencia cuando se pulse uno de los botones del joystick y se detenga cuando se pulse otro de ellos. Seleccione los botones de acuerdo con el joystick que esté utilizando.

2) Implemente un controlador, similar al realizado en las clase de teoría para que el robot siga un camino predefinido.