

Guía para simulación de Cuadricóptero utilizando ROS y Gazebo

Ejemplo 1: Simulación en mundo simple

1. Abrir terminal y entrar al espacio de trabajo

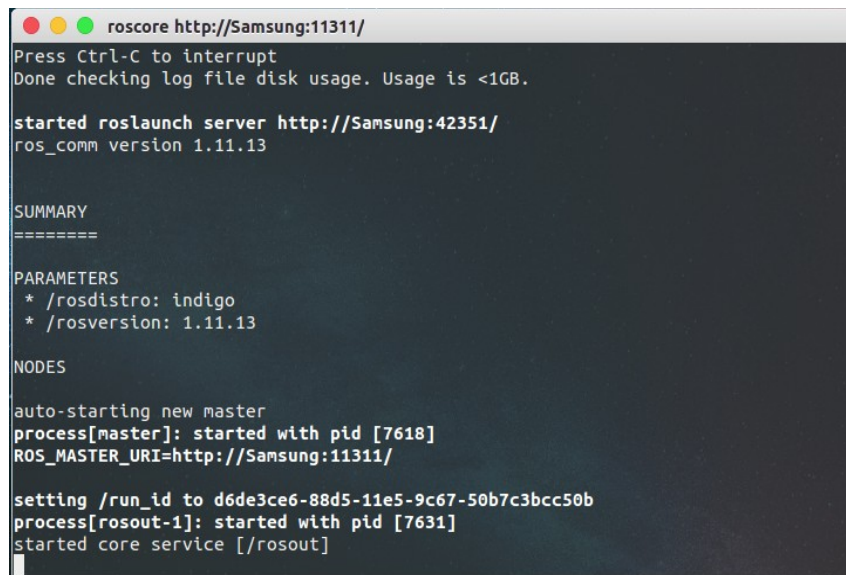
```
$ cd ~/catkin_ws/
```

2. Configurar el espacio de trabajo

```
$ source devel/setup.bash
```

3. Correr el sistema operativo ROS

```
$ roscore
```



```
roscore http://Samsung:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://Samsung:42351/
ros_comm version 1.11.13

SUMMARY
=====

PARAMETERS
* /roscdistro: indigo
* /rosversion: 1.11.13

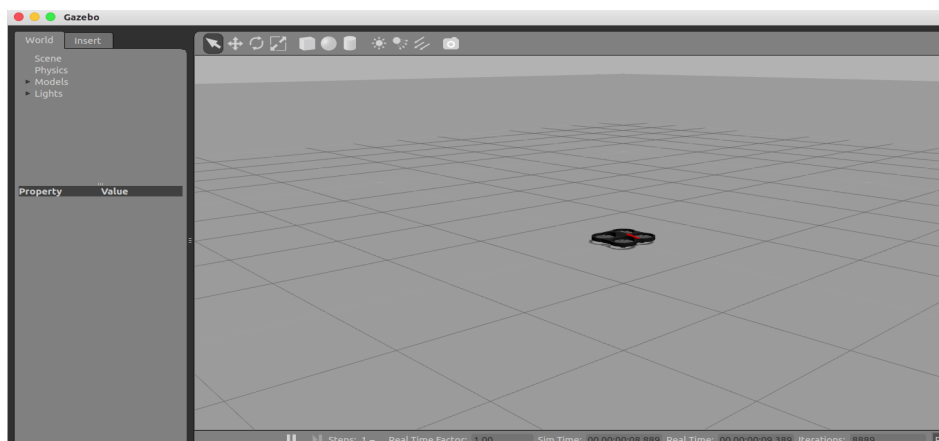
NODES

auto-starting new master
process[master]: started with pid [7618]
ROS_MASTER_URI=http://Samsung:11311/

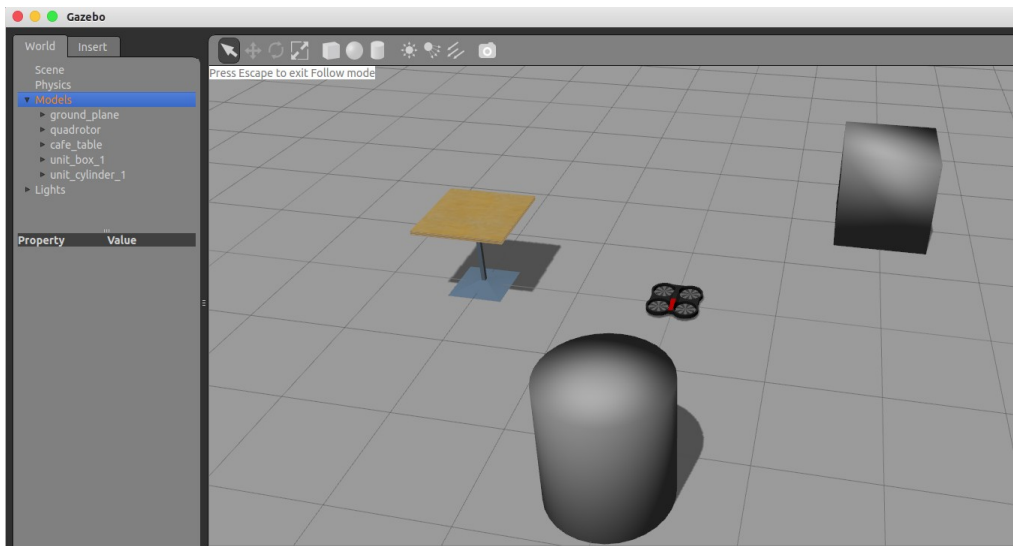
setting /run_id to d6de3ce6-88d5-11e5-9c67-50b7c3bcc50b
process[rosout-1]: started with pid [7631]
started core service [/rosout]
```

4. Abrir un nuevo terminal en el espacio de trabajo, configurar el espacio de trabajo y correr el mundo simple

```
$ roslaunch hector_quadrotor_gazebo quadrotor_empty_world.launch
```



5. Agrega objetos para que sean obstáculos en la simulación



6. En otro terminal en el espacio de trabajo, configurar el espacio de trabajo y correr el controlador de xbox

\$ roslaunch hector_quadrotor_teleop xbox_controller.launch

```
liseth@Samsung: ~/catkin_ws
nch-Samsung-12151.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://Samsung:55917/

SUMMARY
=====
PARAMETERS
* /joy/dev: /dev/input/js0
* /quadrotor_teleop/x_axis: 5
* /quadrotor_teleop/y_axis: 4
* /quadrotor_teleop/yaw_axis: 1
* /quadrotor_teleop/z_axis: 2
* /roscpp: indigo
* /rosversion: 1.11.13
ROSMaster URL: http://Samsung:11311/

NODES
  /
    joy (joy/joy_node)
    quadrotor_teleop (hector_quadrotor_teleop/quadrotor_teleop)
```

7. Conduce el cuadricóptero desde el controlador

Para mover en el eje x: 5

Para mover en el eje y: 4

Para rotar: 1

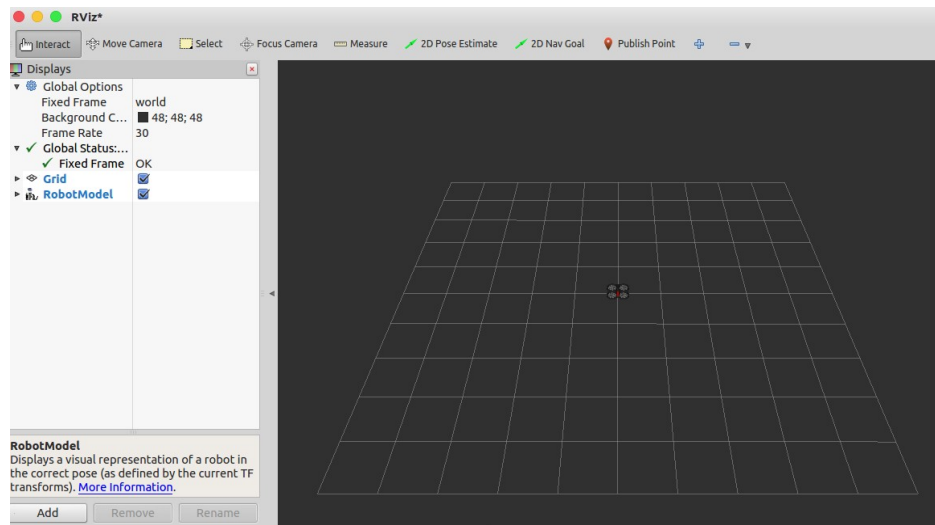
Para mover en el eje z: 2

8. Para ver el detalle del modelo, abrir otro terminal

\$ cd ~/catkin_ws/

\$ source devel/setup.bash

\$rviz rviz



Ejemplo 2: Simulación en un mundo complejo

1. Abrir terminal y entrar al espacio de trabajo

```
$ cd ~/catkin_ws/
```

2. Configurar el espacio de trabajo

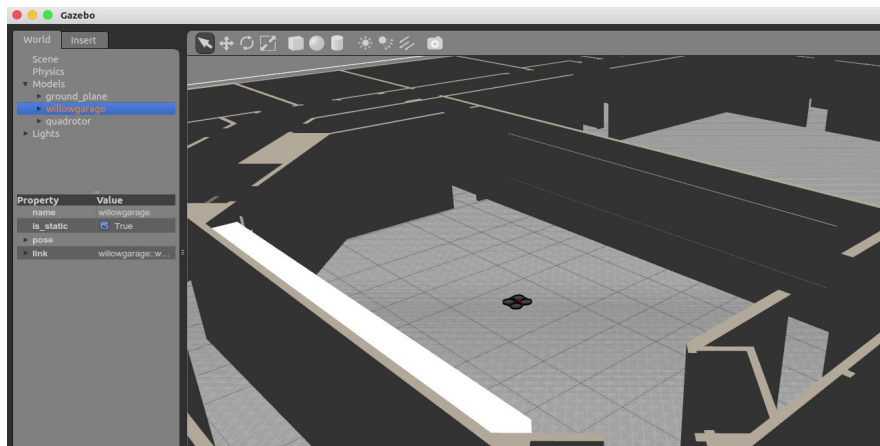
```
$ source devel/setup.bash
```

3. Correr el sistema operativo ROS

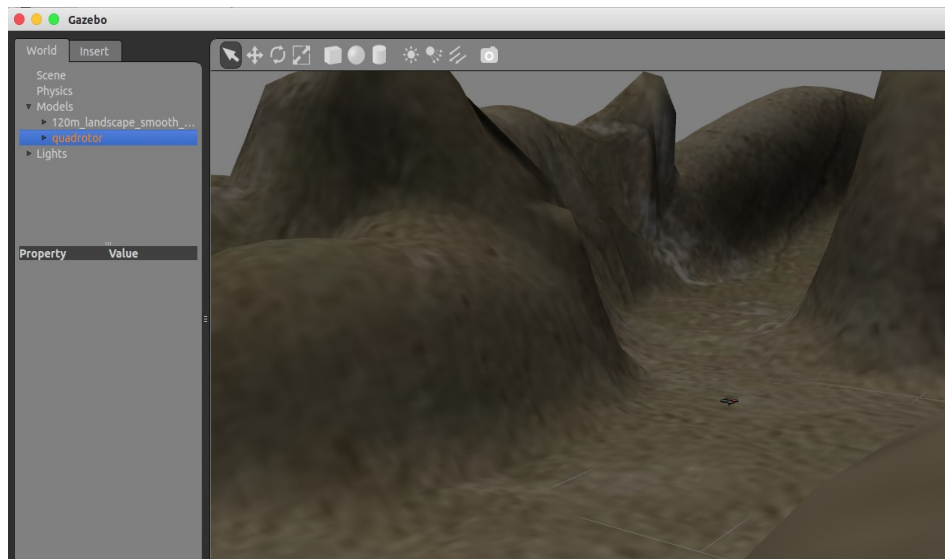
```
$ roscore
```

4. Abrir un nuevo terminal en el espacio de trabajo, configurar el espacio de trabajo y correr el mundo willow garage o el mundo rocoso

```
$ roslaunch hector_quadrotor_demo indoor_slam_gazebo.launch
```



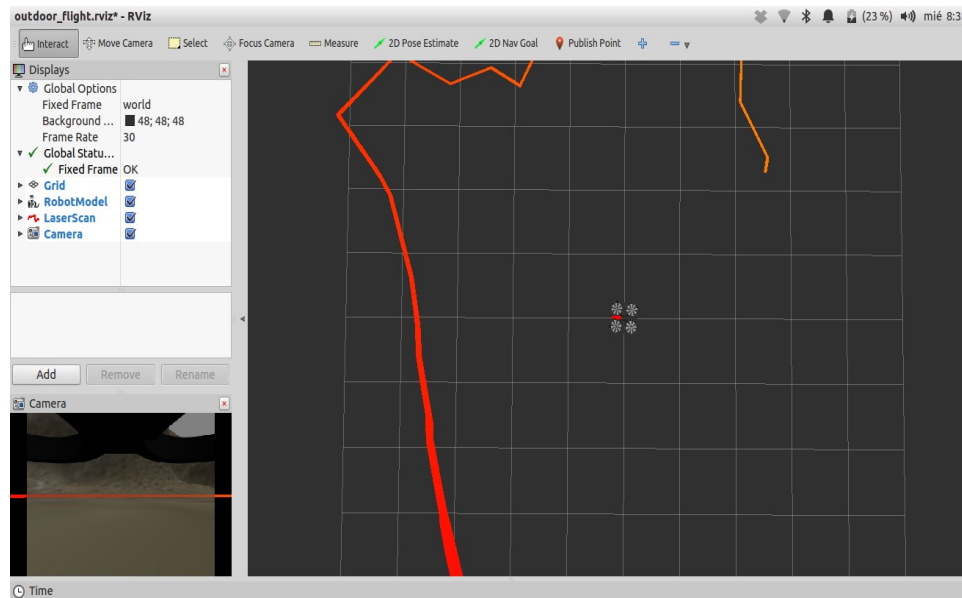
```
$ roslaunch hector_quadrotor_demo outdoor_flight_gazebo.launch
```



5. En otro terminal en el espacio de trabajo, configurar el espacio de trabajo y correr el controlador de xbox

```
$ roslaunch hector_quadrotor_teleop xbox_controller.launch
```

6. Pilotea el cuadricóptero y observa el comportamiento de la cámara en el entorno **Rviz**



Ejemplo 3: Simulación de ruta de control de vuelo

1. Abrir terminal y entrar al espacio de trabajo

```
$ cd ~/catkin_ws/
```

2. Configurar el espacio de trabajo

```
$ source devel/setup.bash
```

3. Correr el sistema operativo ROS

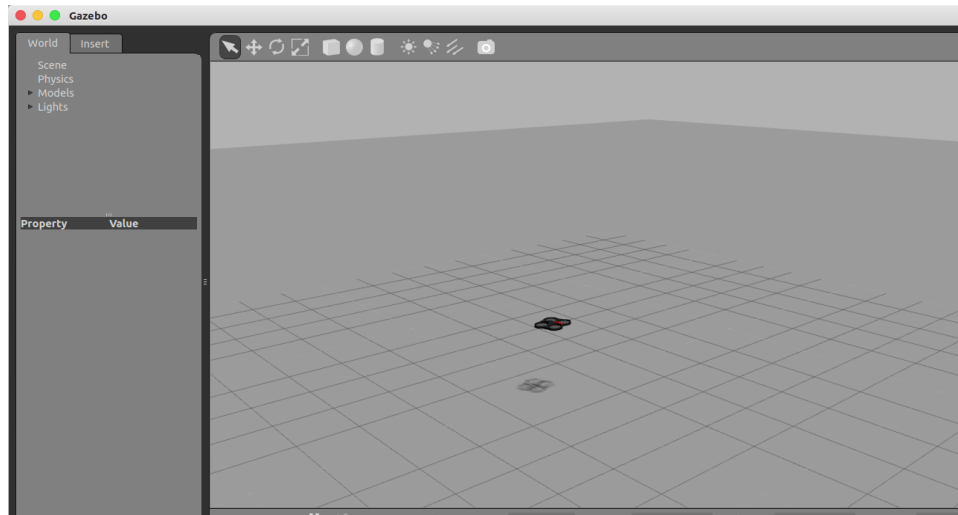
```
$ roscore
```

4. Abrir un nuevo terminal en el espacio de trabajo, configurar el espacio de trabajo y correr el mundo simple

```
$ roslaunch hector_quadrotor_gazebo quadrotor_empty_world.launch
```

5. Correr el nodo

```
$ rosrun quadcopter node
```



6. Para el detalle del modelo, abrir otro terminal

```
$ cd ~/catkin_ws/
```

```
$ source devel/setup.bash
```

```
$ rviz rviz
```