Introdução ao ROS

Andrey Masiero

5 de fevereiro de 2016

Agenda

1. Turtle
2. Dual Shock 3 + ROS
3. Exercício
4. Arquivo launch
   - Criando um launch
   - Executando um launch

Olha a tortuguita

Instalando o pacote `turtlesim`. Abra um terminal e digite:

```
$ sudo apt-get install ros-indigo-turtlesim
```

Olha a tortuguita

No terminal execute o roscore:

```
$ roscore
```

Em um segundo terminal execute:

```
$ rosrun turtlesim turtlesim_node
```

Agora, abra o terceiro terminal e execute:

```
$ rosrun turtlesim turtle_teleop_key
```

Ferramentas de Auxilio

`rostopic list`: lista todos os tópicos ativos. Exemplo de saída:

```
$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

Ferramentas de Auxilio

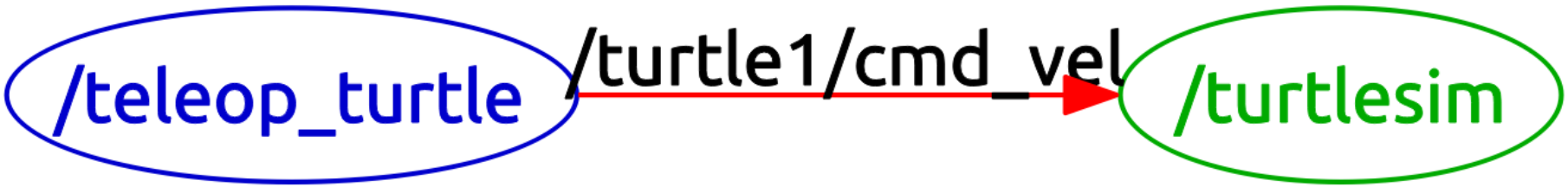
`rostopic echo`: verifica o que é publicado em um tópico. Exemplo de saída:

```
$ rostopic echo /turtle1/cmd_vel
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

Ferramentas de Auxilio

`rqt_graph`: monta o grafo de ligação entre os nós. Exemplo de saída:

```
$ rosrun rqt_graph rqt_graph
```



Dual Shock 3 + ROS

Primeiro é necessário instalar o driver do controle. Abra um terminal e digite:

```
$ sudo apt-get install xboxdrv
```

Na sequência instale o pacote para joystick do ROS.

```
$ sudo apt-get install ros-indigo-joy
```

Dual Shock 3 + ROS

Vamos criar um pacote. Na pasta `src` do seu workspace execute:

```
$ catkin_create_pkg ros_aula_02 rospy
   geometry_msgs sensor_msgs
```

Agora vamos criar um arquivo para receber as informações do controle. Execute os passos:

```
$ cd ~/catkin_ws/src/ros_aula_02
$ mkdir scripts
$ cd scripts
$ gedit ds3_controller.py
```

Dual Shock 3 + ROS

```
#!/usr/bin/env python
import rospy
from sensor_msgs.msg import Joy

def controller(joyMsg):
    if 1 in joyMsg.buttons[0:17]:
        select, l3, r3, start, up, right, down, left, l2, r2, l1, r1, triangle,
             circle, x, square, ps_button = [
            button == 1 for button in joyMsg.buttons[0:17]
        ]
        if up:
            print "Pressionado cima"
        elif down:
            print "Pressionado baixo"
        elif right:
            print "Pressionado direita"
        elif left:
            print "Pressionado esquerda"
        else:
            print "Utilize apenas os comandos de direcao digitais"

def receiver():
    rospy.init_node('ds3_controller', anonymous=True)
    rospy.Subscriber('/joy', Joy, controller)
    rospy.spin()

if __name__ == '__main__':
    receiver()
```

Dual Shock 3 + ROS

É necessário dar permissão de execução ao arquivo `.py`.
Para isso, execute:

```
$ chmod +x ds3_controller.py
```

Hora de testar!

Execute:

```
$ roscore
```

Agora execute o nó que irá publicar as informações do controle:

```
$ rosrun joy joy_node
```

E por fim, o código criado:

```
$ rosrun ros_aula_02 ds3_controller.py
```

Exercício

Controlar a tortuguita através do direcional digital presente no controle Dual Shock 3.

Criando um launch

launch é um arquivo no formato `xml` que contém uma sequência de nós que devem ser executados para realizar uma determinada tarefa. Dessa forma, é possível utilizar apenas um terminal para chamar todos os programas da tarefa.
Os seguintes passos são necessários para criar um launch:

```
$ cd ~/catkin_ws/src/ros_aula_02
$ mkdir launch
$ cd launch
$ gedit turtle_ds3_teleop.launch
```

Criando um launch

```
<?xml version="1.0"?>
<launch>
    <node name="turtle" pkg="turtlesim" type="turtlesim_node" output="screen" />
    <node name="joy" pkg="joy" type="joy_node" output="screen" />
    <node name="ds3_controller" pkg="ros_aula_02" type="ds3_controller.py" output="
        screen" />
</launch>
```

Executando um launch

Para executar um launch basta colocar o seguinte comando no terminal:

```
$ roslaunch ros_aula_02 turtle_ds3_teleop.
    launch
```

Obs: Veja, não é necessário utilizar o comando `roscore` antes de executar o launch.

Referências

Todas as informações e exemplos retirados do site oficial do ROS.
`http://wiki.ros.org/ROS/Tutorials`