Sensores y Actuadores

Clase 9

Sensores

Crear paquete “sensactu”

```
alex@alex-VirtualBox: ~/catkin_ws/src
alex@alex-VirtualBox:~$ cd catkin_ws/src/
alex@alex-VirtualBox:~/catkin_ws/src$ catkin_create_pkg sensactu rospy
Created file sensactu/CMakeLists.txt
Created file sensactu/package.xml
Created folder sensactu/src
Successfully created files in /home/alex/catkin_ws/src/sensactu. Please adjust t
he values in package.xml.
alex@alex-VirtualBox:~/catkin_ws/src$
```

```
alex@alex-VirtualBox: ~/catkin_ws
alex@alex-VirtualBox:~/catkin_ws$ catkin_make
Base path: /home/alex/catkin_ws
Source space: /home/alex/catkin_ws/src
Build space: /home/alex/catkin_ws/build
Devel space: /home/alex/catkin_ws/devel
Install space: /home/alex/catkin_ws/install

####
#### Running command: "cmake /home/alex/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/hom
e/alex/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/alex/catkin_ws/install -G Un
ix Makefiles" in "/home/alex/catkin_ws/build"
####
```

Compilación del paquete

```
alex@alex-VirtualBox: ~/catkin_ws
-- ==   - new_package
-- ==   - sensactu
-- ------------------------------------------------------------
-- +++ processing catkin package: 'new_package'
-- ==> add_subdirectory(new_package)
-- Using these message generators: gencpp;genlisp;genpy
-- new_package: 0 messages, 1 services
-- +++ processing catkin package: 'sensactu'
-- ==> add_subdirectory(sensactu)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/alex/catkin_ws/build
####
#### Running command: "make -j1 -l1" in "/home/alex/catkin_ws/build"
####
[  0%] Built target _new_package_generate_messages_check_deps_WordCount
[  0%] Built target std_msgs_generate_messages_lisp
[ 25%] Built target new_package_generate_messages_lisp
[ 25%] Built target std_msgs_generate_messages_py
[ 75%] Built target new_package_generate_messages_py
[ 75%] Built target std_msgs_generate_messages_cpp
[100%] Built target new_package_generate_messages_cpp
[100%] Built target new_package_generate_messages
alex@alex-VirtualBox:~/catkin_ws$
```

Habilitar la librería FakeSensor

- ➔ Descargamos el archivo “fake_sensor.py”.
- ➔ Se copia el archivo en la dirección indicada en la figura.
- ➔ Se da los permisos necesarios para trabajar con el archivo.
- ➔ Luego se importará algunos sensores o actuadores.

Obtener las medidas del sensor

```
#!/usr/bin/env python
from math import pi
from fake_sensor import FakeSensor
import rospy
import tf
from geometry_msgs.msg import Quaternion

def make_quaternion(angle):
    q = tf.transformations.quaternion_from_euler(0, 0, angle)
    return Quaternion(*q)

if __name__ == '__main__':
    sensor = FakeSensor()
    rospy.init_node('fake_sensor')
    pub = rospy.Publisher('angle', Quaternion, queue_size=10)
    rate = rospy.Rate(10.0)
    while not rospy.is_shutdown():
        angle = sensor.sensor.value() * 2 * pi / 100.0
        q = make_quaternion(angle)
        pub.publish(q)
        rate.sleep()
```

Se importa la librería FakeSensor

Tipo de dato para ubicación de un móvil

Cambio de tipo de dato, de ángulos a Quaternion

Variable tipo FakeSensor

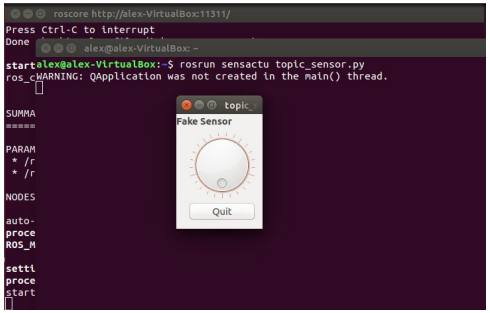
Obtener las medidas del sensor

```
#!/usr/bin/env python
from math import pi
from fake_sensor import FakeSensor
import rospy
import tf
from geometry_msgs.msg import Quaternion

def make_quaternion(angle):
    q = tf.transformations.quaternion_from_euler(0, 0, angle)
    return Quaternion(*q)

if __name__ == '__main__':
    sensor = FakeSensor()
    rospy.init_node('fake_sensor')
    pub = rospy.Publisher('angle', Quaternion, queue_size=10)
    rate = rospy.Rate(10.0)
    while not rospy.is_shutdown():
        angle = sensor.sensor.value() * 2 * pi / 100.0
        q = make_quaternion(angle)
        pub.publish(q)
        rate.sleep()
```

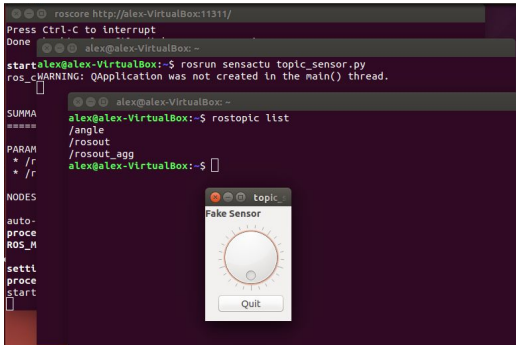
Se muestre recibe el valor del sensor cada 10 hz

Se hace la transformación del ángulo

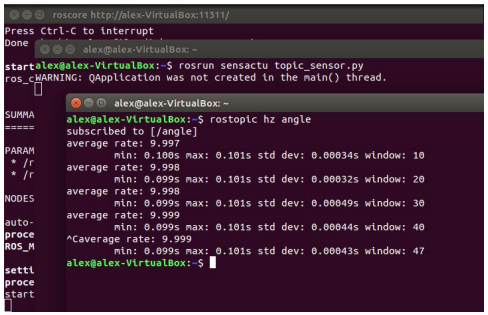
Obtener las medidas del sensor



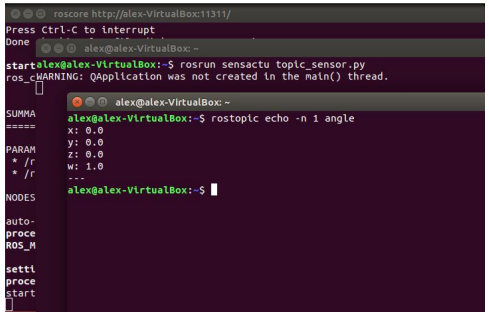
Lista de los tópicos



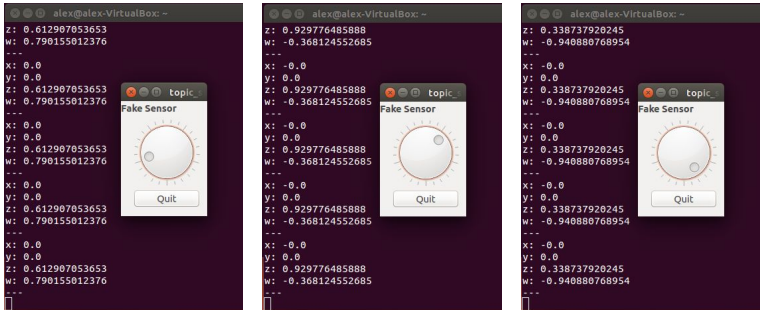
Transmisión de data



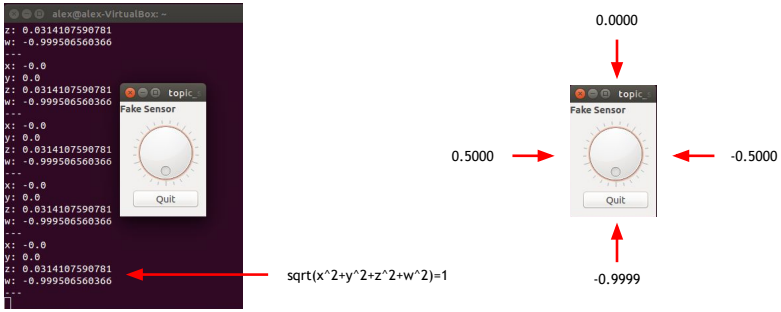
Mostrar la data del sensor



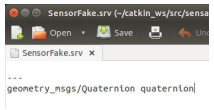
Mostrar la data del sensor



Mostrar la data del sensor

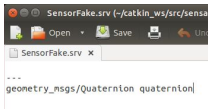


Servicio para un sensor



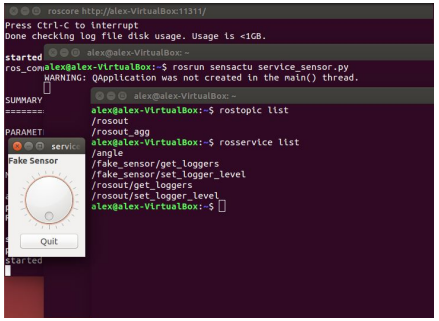

```python
#!/usr/bin/env python
from math import pi
from fake_sensor import FakeSensor

import rospy
import tf
from geometry_msgs.msg import Quaternion
from sensactu.srv import SensorFake, SensorFakeResponse

def make_quaternion(angle):
    q = tf.transformations.quaternion_from_euler(0, 0, angle)
    return Quaternion(*q)
```

Servicio para un sensor

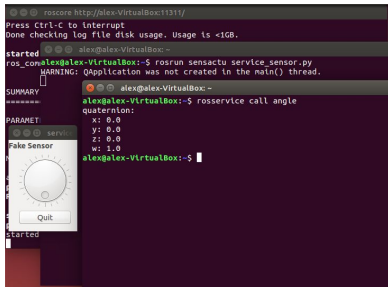


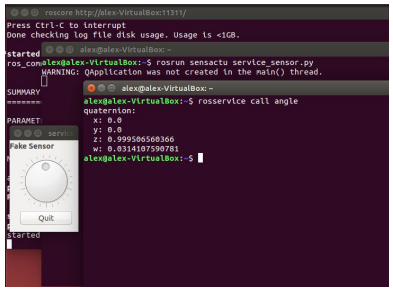

```python
def callback(request):
    global sensor
    angle = sensor.sensor.value * 2 * pi / 100.0
    q = make_quaternion(angle)

    return SensorFakeResponse(q)

if __name__ == '__main__':
    global sensor
    sensor = FakeSensor()

    rospy.init_node('fake_sensor')
    service = rospy.Service('angle', SensorFake, callback)

    rospy.spin()
```

Servicio y tópicos del sensor

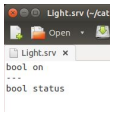


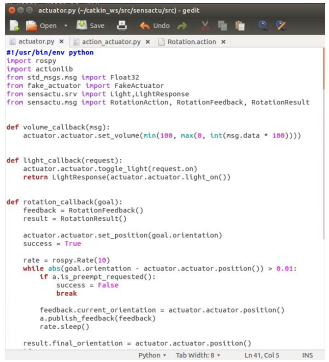
Servicio y tópicos del sensor



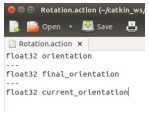


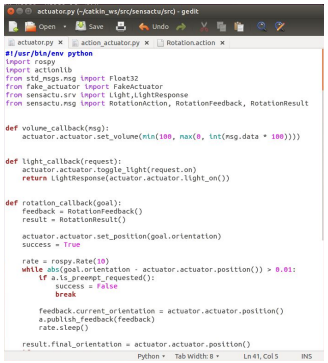
Actuadores



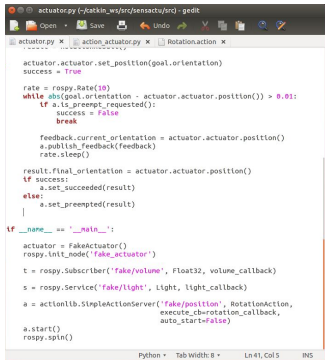


Actuadores





Actuadores



Implementar configuraciones

package.xml (~/catkin_ws/src/sensactu) - gedit

```xml
<?xml version="1.0"?>
<package>
  <name>sensactu</name>
  <version>0.0.1</version>
  <description>Virtual Sensor/Actuators package</description>
  <maintainer email="lopez.alexander@pucp.pe">Alexander Lopez</maintainer>
  <license>BSD</license>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>message_generation</build_depend>
  <build_depend>actionlib_msgs</build_depend>
  <run_depend>actionlib_msgs</run_depend>
  <run_depend>message_runtime</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>rospy</run_depend>

  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>
</package>
```

Implementar configuraciones

CMakeLists.txt (~/catkin_ws/src/sensactu) - gedit

```cmake
cmake_minimum_required(VERSION 2.8.3)
project(sensactu)

## Add support for C++11, supported in ROS Kinetic and newer
# add_definitions(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  rospy
  std_msgs
  message_generation
  geometry_msgs
  actionlib_msgs
)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
# catkin_python_setup()
```

Implementar configuraciones

CMakeLists.txt (~/catkin_ws/src/sensactu) - gedit

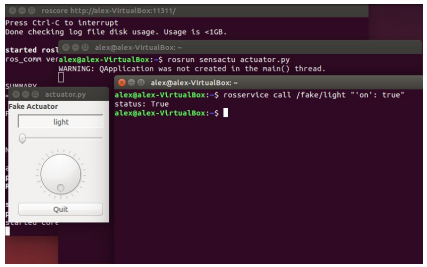
```cmake
# )

## Generate services in the 'srv' folder
add_service_files(
  FILES
  SensorFake.srv
  Light.srv
#   Service1.srv
#   Service2.srv
)

## Generate actions in the 'action' folder
add_action_files(
  DIRECTORY action
  FILES Rotation.action
#   Action1.action
#   Action2.action
)

## Generate added messages and services with any dependencies listed here
generate_messages(
  DEPENDENCIES
  std_msgs
  actionlib_msgs
  geometry_msgs
)
```

Implementar configuraciones

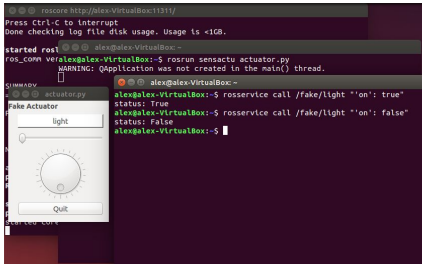
CMakeLists.txt (~/catkin_ws/src/sensactu) - gedit

```cmake
## CATKIN_DEPENDS: catkin_packages dependent projects also need
## DEPENDS: system dependencies of this project that dependent projects
also need
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES sensactu
   CATKIN_DEPENDS message_runtime rospy std_msgs actionlib_msgs
#  DEPENDS system_lib
)

###########
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
# include_directories(include)
include_directories(
  ${catkin_INCLUDE_DIRS}
)

## Declare a C++ library
# add_library(${PROJECT_NAME}
#   src/${PROJECT_NAME}/sensactu.cpp
# )
```

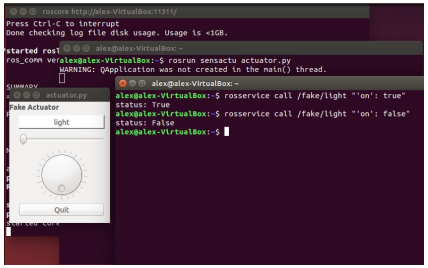
Publicación



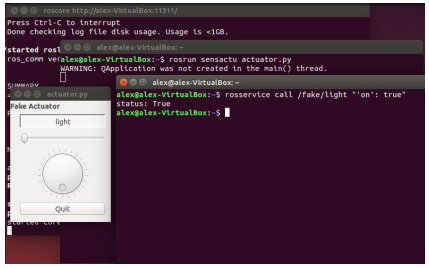
Publicación



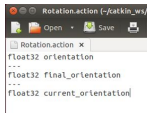
Servicio

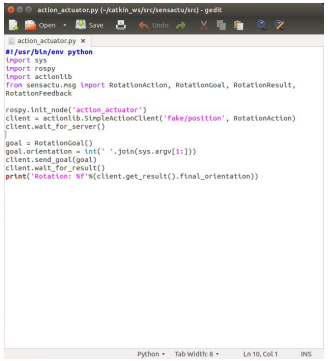


Servicio

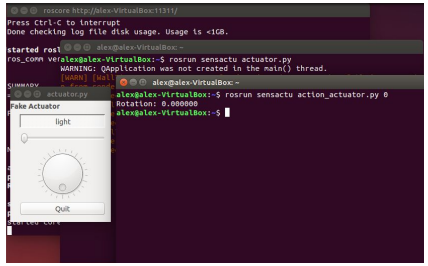


Implementación de una Acción

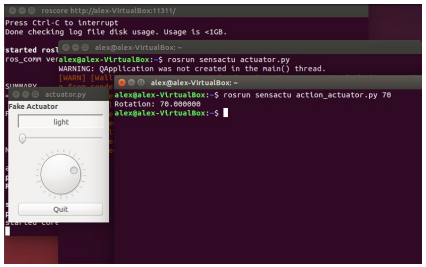




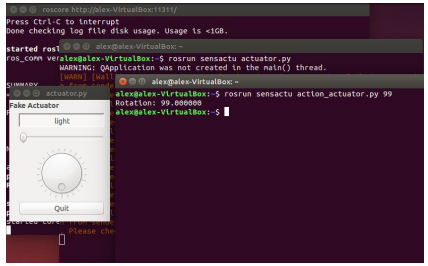
Acción



Acción



Acción



¡Gracias!

¡La única pregunta tonta es la que no se hace!