Ast x Oru Robotics Summer School

Andreas Persson, David Caceres Dominguez, Pedro Zuidberg Dos Martires

Örebro University

github.com/pedrozudo/astxoru-roboticssummerschool

Recap from Last Week

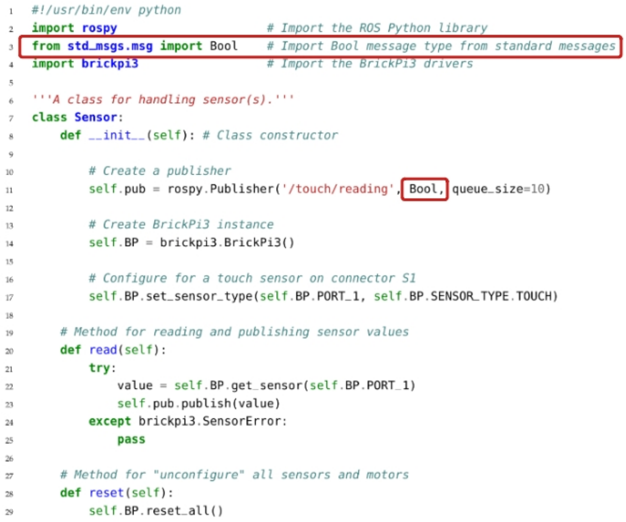
Recap from Last Week

- `Sensor` class for...
- Reading BrickBi3 sensor
- Publishing ROS message

```python
#!/usr/bin/env python
import rospy                         # Import the ROS Python library
from std_msgs.msg import Bool        # Import Bool message type from standard messages
import brickpi3                      # Import the BrickPi3 drivers

'''A class for handling sensor(s).'''
class Sensor:
    def __init__(self): # Class constructor

        # Create a publisher
        self.pub = rospy.Publisher('/touch/reading', Bool, queue_size=10)

        # Create BrickPi3 instance
        self.BP = brickpi3.BrickPi3()

        # Configure for a touch sensor on connector S1
        self.BP.set_sensor_type(self.BP.PORT_1, self.BP.SENSOR_TYPE.TOUCH)

    # Method for reading and publishing sensor values
    def read(self):
        try:
            value = self.BP.get_sensor(self.BP.PORT_1)
            self.pub.publish(value)
        except brickpi3.SensorError:
            pass

    # Method for "unconfigure" all sensors and motors
    def reset(self):
        self.BP.reset_all()
```

Recap from Last Week

- Sensor reading published as...
- `Bool` message type
- From standard messages (`std_msgs`)

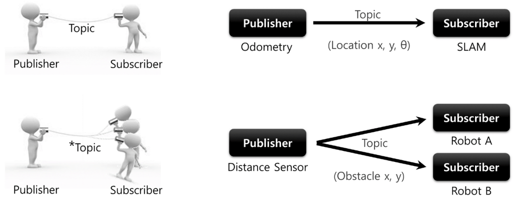


```python
#!/usr/bin/env python
import rospy                        # Import the ROS Python library
from std_msgs.msg import Bool       # Import Bool message type from standard messages
import brickpi3                     # Import the BrickPi3 drivers

'''A class for handling sensor(s).'''
class Sensor:
    def __init__(self): # Class constructor

        # Create a publisher
        self.pub = rospy.Publisher('/touch/reading', Bool, queue_size=10)

        # Create BrickPi3 instance
        self.BP = brickpi3.BrickPi3()

        # Configure for a touch sensor on connector S1
        self.BP.set_sensor_type(self.BP.PORT_1, self.BP.SENSOR_TYPE.TOUCH)

    # Method for reading and publishing sensor values
    def read(self):
        try:
            value = self.BP.get_sensor(self.BP.PORT_1)
            self.pub.publish(value)
        except brickpi3.SensorError:
            pass

    # Method for "unconfigure" all sensors and motors
    def reset(self):
        self.BP.reset_all()
```

Recap from Last Week

In fact, there are many different ROS message types...

- `std_msgs` standard messages
- `sensor_msgs` sensor messages
- `geometry_msgs` geometric primitives
- `nav_msgs` navigation messages
- ...

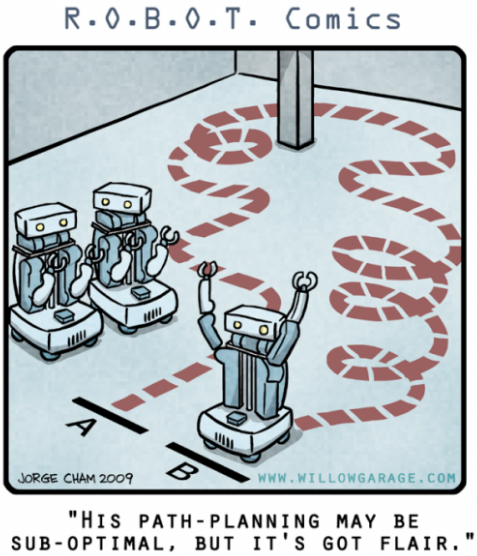


*Topic not only allows 1:1 Publisher and Subscriber communication, but also supports 1:N, N:1 and N:N depending on the purpose.

Navigation

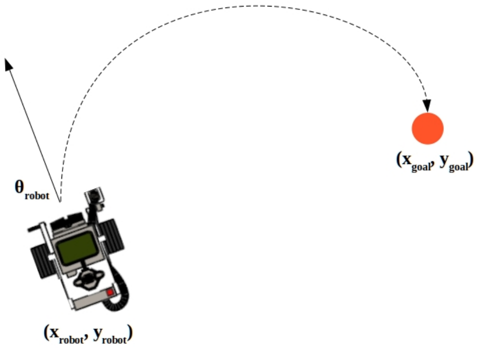
Navigation

- *GoTo Behaviour*
- Make the robot go from point **A** to point **B**



Navigation

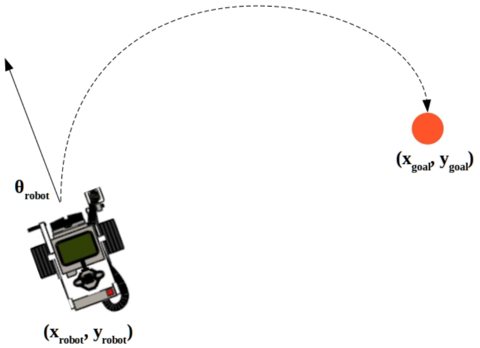
- *Whats needed?*
- Position and orientation of the **robot**
- Position of the **goal** (position)
- Robot wheel configuration and dimensions



- Also, see Luis slides:
  `github.com/pedrozudo/astxoru-roboticssummerschool/lectures_luis/slides.pdf`

Navigation

- *Today (before lunch):*
  1. Assume that the **robot start** at position and orientation `(0.0, 0.0, 0.0)`, and;
  2. Write the *GoTo behaviour* that makes the robot go the a given arbitrary **goal position**



To be continue (after lunch)...