Ast x Oru Robotics Summer School

Andreas Persson, David Caceres Dominguez, Pedro Zuidberg Dos Martires

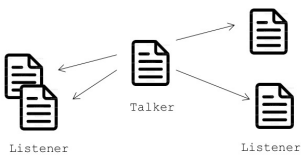
Örebro University

github.com/pedrozudo/astxoru-roboticssummerschool

Recap from Yesterday

ROS nodes are program that communicate with each other.

- Message passing (through "topics")
- Publisher (talker)
- Subscriber (listener)
- Uses ROS message files



Recap from Yesterday

- ROS Rate loop
- Fixed frequency
- Based on Timers
- Used for processing and sending data (e.g. sensor readings)

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

# Main function
if __name__ == '__main__':
    try:

        # Init the connection with the ROS system
        rospy.init_node('talker', anonymous=True)

        # Create a publisher that will publish messages on topic named 'chatter'
        pub = rospy.Publisher('chatter', String, queue_size=10)

        # Start the ROS main loop, running with a frequency of 10Hz
        rate = rospy.Rate(10)
        while not rospy.is_shutdown():

            # Create and publish a String message
            str = "hello world %s" % rospy.get_time()
            rospy.loginfo(str)
            pub.publish(str)

            # Call sleep to maintain the desired rate
            rate.sleep()

    except rospy.ROSInterruptException:
        pass
```

Recap from Yesterday

- ROS spin loop
- No timers
- Only used to trigger subscriber callbacks

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

# A callback function that is called every time there is new a message
# available on the topic of interest, i.e., the 'chatter' topic in this case.
def callback(msg):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", msg.data)

# Main function
if __name__ == '__main__':

    # Init the connection with the ROS system
    rospy.init_node('listener', anonymous=True)

    # Initialize a subscriber that will receive and handle message
    # though a given callback function
    rospy.Subscriber("chatter", String, callback)

    # Keeps the node spinning until the node is stopped
    rospy.spin()
```

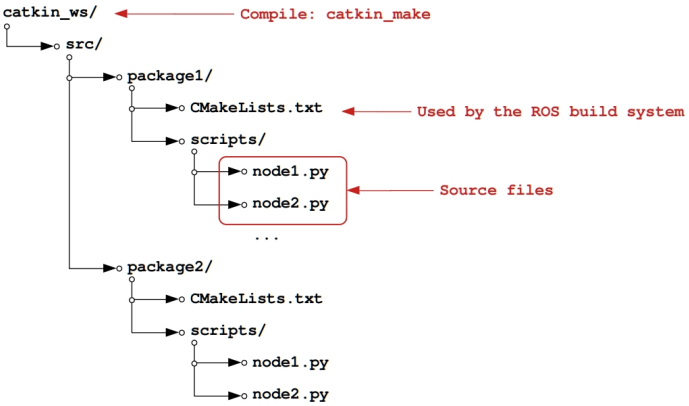
Recap from Yesterday

- ROS callback functions
- Used for receiving data (e.g. movement commands)
- Only "local" data
- Store received data outside the callback function:
  - Global variables
  - Class member variables

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

# A callback function that is called every time there is new a message
# available on the topic of interest, i.e., the 'chatter' topic in this case.
def callback(msg):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", msg.data)

# Main function
if __name__ == '__main__':

    # Init the connection with the ROS system
    rospy.init_node('listener', anonymous=True)

    # Initialize a subscriber that will receive and handle message
    # though a given callback function
    rospy.Subscriber("chatter", String, callback)

    # Keeps the node spinning until the node is stopped
    rospy.spin()
```

Recap from Yesterday

The structure of a ROS.



ROS Network

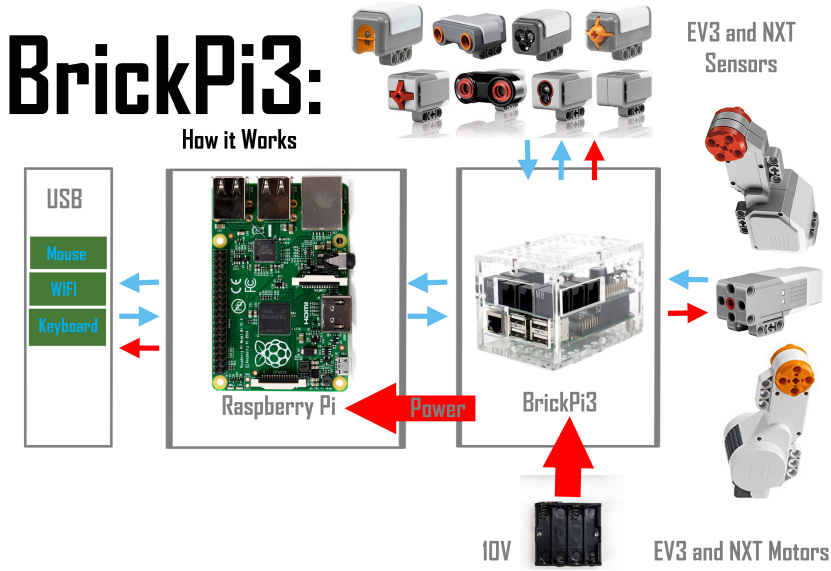
ROS Network

Configuring a ROS network across multiple devices is, in fact, very easy.

- *ROS Master* environment variable
- Changed by `export` command:

  `export ROS_MASTER_URI=http://192.168.1.112:11311`
- One `roscore` for all devices

BrickPi3 Robots


BrickPi3:
How it Works
EV3 and NXT
Sensors
USB
Mouse
WIFI
Keyboard
Raspberry Pi
Power
BrickPi3
10V
EV3 and NXT Motors