

Ast x Oru Robotics Summer School

Andreas Persson, David Caceres Dominguez, Pedro Zuidberg Dos Martires

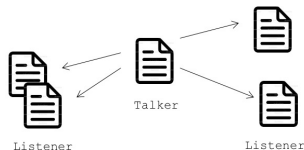
Örebro University

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)

```
1  #!/usr/bin/env python
2  import rospy # Import the ROS Python library
3  from std_msgs.msg import String # Import String message type from standard messages
4
5  # Main function
6  if __name__ == '__main__':
7      try:
8
9          # Init the connection with the ROS system
10         rospy.init_node('talker', anonymous=True)
11
12         # Create a publisher that will publish messages on topic named 'chatter'
13         pub = rospy.Publisher('chatter', String, queue_size=10)
14
15         # Start the ROS main loop, running with a frequency of 10Hz
16         rate = rospy.Rate(10)
17         while not rospy.is_shutdown():
18
19             # Create and publish a String message
20             str = "hello world %s" % rospy.get_time()
21             rospy.loginfo(str)
22             pub.publish(str)
23
24             # Call sleep to maintain the desired rate
25             rate.sleep()
26
27 except rospy.ROSInterruptException:
28     pass
```

Recap from Yesterday

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

```
1  #!/usr/bin/env python
2  import rospy # Import the ROS Python library
3  from std_msgs.msg import String # Import String message type from standard messages
4
5  # A callback function that is called every time there is new a message
6  # available on the topic of interest, i.e., the 'chatter' topic in this case.
7  def callback(msg):
8      rospy.loginfo(rospy.get_caller_id() + "I heard %s", msg.data)
9
10 # Main function
11 if __name__ == '__main__':
12
13     # Init the connection with the ROS system
14     rospy.init_node('listener', anonymous=True)
15
16     # Initialize a subscriber that will receive and handle message
17     # though a given callback function
18     rospy.Subscriber("chatter", String, callback)
19
20     # Keeps the node spinning until the node is stopped
21     rospy.spin()
```

Recap from Yesterday

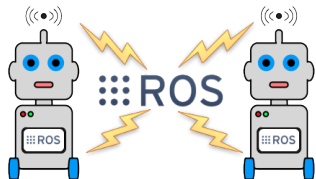
- ROS callback functions
- Used for receiving data (e.g. movement commands)
- Only "local" data
- Store received data outside the callback function:
 - Global variables
 - Class member variables

```
1  #!/usr/bin/env python
2  import rospy # Import the ROS Python library
3  from std_msgs.msg import String # Import String message type from standard messages
4
5  # A callback function that is called every time there is new a message
6  # available on the topic of interest, i.e., the 'chatter' topic in this case.
7  def callback(msg):
8      rospy.loginfo(rospy.get_caller_id() + "I heard %s", msg.data)
9
10 # Main function
11 if __name__ == '__main__':
12
13     # Init the connection with the ROS system
14     rospy.init_node('listener', anonymous=True)
15
16     # Initialize a subscriber that will receive and handle message
17     # though a given callback function
18     rospy.Subscriber("chatter", String, callback)
19
20     # Keeps the node spinning until the node is stopped
21     rospy.spin()
```

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://rp12:11311`
- One roscore for all devices



BrickPi3 Robots

BrickPi3:

How it Works

