https://wiki.bu.edu/roboclass/index.php?title=ROS_Quickref_for_new_nodes

4 min read

ROS Quickref for new nodes

The following is a quick summary of how to write a new node, depending on whether you have ok or not.

1. `#!/usr/bin/env python`, `import rospy`
2. import any message type you need to publish or subscribe
3. import any additional module (e.g., NumPy, OpenCV)
4. If your node has subscribers: write your callback function
5. Initialize your node
6. Create your subscriber and/or publisher objects
7. If you have a publisher, create a Rate object that you will use to pace your publishing. Optional: create a message object that you will reuse every time you need to publish
8. Main loop:
    1. If you don't have publishers: `rospy.spin()`
    2. If you have publishers:
        1. `while not rospy.is_shutdown():`
        2. Prepare message (set all attributes that are needed)
        3. Call `publish` method of the publisher with the prepared message
        4. Optional: perform any other operation that needs to happen
        5. rate.sleep()

Following the recipe above, there are mainly four types of nodes that you will encounter in this class (and in general), with the following structures.

Publisher-only[edit]

This node publishes messages at a constant rate. In the main function:

- Node initialization
- Publisher object with pub=rospy.Publisher()
- Create a rate=rospy.Rate() object
- While loop with pub.publish() command and rate.sleep()

An example of this is provided in talker.py.

Subscriber-only[edit]

This node only receives messages, without publishing anything.

Before the main function:

- Define a callback (with global variables or a class if it needs to remember data between calls)

In the main function:

- Node initialization
- Subscriber instance with rospy.Subscriber()
- Enter "waiting loop" with rospy.spin()

An example of this is provided in listener.py

Publish-upon-callback[edit]

Every time a message is received, this node modifies it and publishes it on another topic.

Before the main function:

- Define a callback, using a publisher pub as global variable

In the main function:

- Make publisher pub a global variable
- Node initialization
- Create global publisher, pub=rospy.Publisher()
- Subscriber instance with rospy.Subscriber()
- Enter "waiting loop" with rospy.spin()

An example of this is provided in repeater.py

Subscriber with regular publishing[edit]

This node "accumulates" the effects of multiple received messages, storing the result in a global variable. At a constant rate, it then publishes the global results on another topic.

Before the main function:

- Define a callback, using a global variable to keep track of the state ("accumulated" messages) across calls

In the main function:

- Make state variable a global variable
- Node initialization
- Create publisher, pub=rospy.Publisher() (note, the publisher does not need to be global)
- Subscriber instance with rospy.Subscriber()
- Create a rate=rospy.Rate() object
- While loop with pub.publish() command and rate.sleep()

An example of this is given as a question in Homework 1.