

Lab 2

Adding Suction Feedback
(Updated Fall 2019)

Goal: Implement Suction Feedback

- We want to subscribe to the information about Suction Feedback
 - This means we want to know about a digital or analog input
- There is a one other subscriber in our code, so let's use that as a guide to help us
- Important commands
 - `rostopic list`
 - `rostopic info <topic_name>`
 - `rostopic echo <topic_name>`
 - `rosmmsg list`
 - `rosmmsg info <message_name>`

How does a Subscriber work?

```
241
242     # Initialize subscriber to ur3/position and callback fuction
243     # each time data is published
244     sub_position = rospy.Subscriber('ur3/position', position, position_callback)
245
```

- We can see that the function takes three arguments:
 - A topic name
 - A data type
 - A callback function
- The callback function is called each time the topic is published
- We assign the function return to “sub_position,” but we don’t use this variable elsewhere in the code
- <http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers>
- <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>

The callback function

```
87 """
88 Whenever ur3/position publishes info, this callback function is called.
89 """
90 def position_callback(msg):
91     global thetas
92     global current_position
93     global current_position_set
94
95     thetas[0] = msg.position[0]
96     thetas[1] = msg.position[1]
97     thetas[2] = msg.position[2]
98     thetas[3] = msg.position[3]
99     thetas[4] = msg.position[4]
100     thetas[5] = msg.position[5]
101
102     current_position[0] = thetas[0]
103     current_position[1] = thetas[1]
104     current_position[2] = thetas[2]
105     current_position[3] = thetas[3]
106     current_position[4] = thetas[4]
107     current_position[5] = thetas[5]
108
109     current_position_set = True
110
111
```

- This function takes `msg` as the argument.
- It then passes the value of this data into global variables (`thetas []`)
- The data is stored in a data structure passed in as `msg`

The callback function (continued)

- We are using the callback function to bring message data into our program.
- The additional code in this callback function (e.g. `current_position`) is part of the way the our code is implemented and not a required part of how a callback function works.
- You can put additional code you might need within the callback function


How can we find this data?

- If we didn't have this data how could we find it?
- Let's explore using the commands from before
 - `rostopic list`
 - `rostopic info <topic_name>`
 - `rostopic echo <topic_name>`
 - `rosmmsg list`
 - `rosmmsg info <message_name>`
 - **Note:** `show` and `info` work the same
- This process is done while running ROS in one terminal and entering commands in a second
 - Run `roslaunch ur3_driver ur3_driver.launch` in the first terminal
 - Be sure to source `devel/setup.bash` in the second terminal
 - Run `roslaunch lab2pkg_py lab2_exec.py` at least once or you might not see some data.

rostopic list

- This gives a list of all the topics
- We can see /ur3/position

```
ur3@ur3-8: ~/feedback_test
ur3@ur3-8:~/feedback_test$ source devel/setup.bash
ur3@ur3-8:~/feedback_test$ rostopic list
/rosout
/rosout_agg
/ur3/command
/ur3/gripper_input
/ur3/position
ur3@ur3-8:~/feedback_test$
```



```
rostopic info /ur3/position
```

- This gives more information about a specific topic
- Note that we can see the data type used for the callback function
- This is also the name of the message

```
ur3@ur3-8: ~/feedback_test
ur3@ur3-8:~/feedback_test$ source devel/setup.bash
ur3@ur3-8:~/feedback_test$ rostopic list
/rosout
/rosout_agg
/ur3/command
/ur3/gripper_input
/ur3/position
ur3@ur3-8:~/feedback_test$ rostopic info /ur3/position
Type: ur3_driver/position
Publishers:
 * /ur3_driver_1 (http://ur3-8:39225/)
Subscribers: None

ur3@ur3-8:~/feedback_test$
```


rosmmsg list

- This gives a list of all the messages
- We can see the `ur3_driver/position` message here
- If you don't see this, you probably have not run the lab 2 code.

```
ur3@ur3-8: ~/feedback_test
turtle_actionlib/ShapeActionResult
turtle_actionlib/ShapeFeedback
turtle_actionlib/ShapeGoal
turtle_actionlib/ShapeResult
turtle_actionlib/Velocity
turtlesim/Color
turtlesim/Pose
ur3_driver/command
ur3_driver/gripper_input
ur3_driver/position ←
ur_msgs/Analog
ur_msgs/Digital
ur_msgs/IOStates
ur_msgs/MasterboardDataMsg
ur_msgs/RobotStateRTMsg
ur_msgs/ToolDataMsg
uuid_msgs/UniqueID
variant_msgs/Test
variant_msgs/Variant
variant_msgs/VariantHeader
variant_msgs/VariantType
visualization_msgs/ImageMarker
visualization_msgs/InteractiveMarker
visualization_msgs/InteractiveMarkerControl
```

rosmmsg info ur3_driver/position

- This shows us all the members of the message data structure
- We can see 2 members:
 - Float64[] position
 - Note: [] indicates that this is an array
 - bool isReady
 - Not used in Python

```
ur3@ur3-8: ~/feedback_test
ur_msgs/IOStates
ur_msgs/MasterboardDataMsg
ur_msgs/RobotStateRTMsg
ur_msgs/ToolDataMsg
uuid_msgs/UniqueID
variant_msgs/Test
variant_msgs/Variant
variant_msgs/VariantHeader
variant_msgs/VariantType
visualization_msgs/ImageMarker
visualization_msgs/InteractiveMarker
visualization_msgs/InteractiveMarkerControl
visualization_msgs/InteractiveMarkerFeedback
visualization_msgs/InteractiveMarkerInit
visualization_msgs/InteractiveMarkerPose
visualization_msgs/InteractiveMarkerUpdate
visualization_msgs/Marker
visualization_msgs/MarkerArray
visualization_msgs/MenuEntry
ur3@ur3-8:~/feedback_test$ rosmmsg info ur3_driver/position
float64[] position
bool isReady
ur3@ur3-8:~/feedback_test$
```

rostopic echo /ur3/position

- echo, allows us to see the values of the topic
- Note that we can see the current values of `position` (there are six values in the array) and `isReady`
- echo continues to output until stopped

```
ur3@ur3-8: ~/feedback_test
80591021937, -1.5610006491290491, 2.7926506996154785]
isReady: True
---
position: [3.3602089881896973, -0.8035529295550745, 1.2048301696777344, -2.01137
3821889059, -1.5610244909869593, 2.7926506996154785]
isReady: True
---
position: [3.3602328300476074, -0.8035767714129847, 1.2048540115356445, -2.01136
1900960104, -1.5610244909869593, 2.7926745414733887]
isReady: True
---
position: [3.3602449893951416, -0.8035529295550745, 1.2048301696777344, -2.01136
1900960104, -1.5610125700580042, 2.7926387786865234]
isReady: True
---
position: [3.360197067260742, -0.8035410086261194, 1.204805850982666, -2.0113499
800311487, -1.5610244909869593, 2.7926626205444336]
isReady: True
---
^Cposition: [3.3602569103240967, -0.8035171667682093, 1.20484209060666895, -2.011
385742818014, -1.5610244909869593, 2.7926266193389893]
isReady: True
---
ur3@ur3-8:~/feedback_test$ rostopic echo /ur3/position
```

```
rostopic echo
/ur3/position/position[0]
```

```
isReady: True
---
ur3@ur3-8:~/feedback_test$ rostopic echo /ur3/position/position[0] -n 1
3.36023283005
---
ur3@ur3-8:~/feedback_test$ █
```

- We can look at elements within the data structure as well
 - Here we are only looking at the value of Theta1 (`position[0]`) in the position array
 - We could just as easily look at Theta5 or `isReady`
- `rostopic echo /ur3/position/position[0] -n 1`
 - This allows us to echo only one instance of data instead of streaming it

Putting it all together

```
241
242     # Initialize subscriber to ur3/position and callback fuction
243     # each time data is published
244     sub_position = rospy.Subscriber('ur3/position', position, position_callback)
245
```

- We know we want the values of `position`
- By searching the topics, we found the values in the topic `/ur3/position` and the message data type `position`
- We can now create our subscriber function
- We assign it to a convenient variable (`sub_position`)
- We select an appropriate callback function name (`position_callback`)

The callback function

```
89 """
90 def position_callback(msg):
91     global thetas
92     global current_position
93     global current_position_set
94
95     thetas[0] = msg.position[0]
96     thetas[1] = msg.position[1]
97     thetas[2] = msg.position[2]
98     thetas[3] = msg.position[3]
99     thetas[4] = msg.position[4]
100     thetas[5] = msg.position[5]
101
102
```

- We learned the data type and pass it in (`msg`)
- We create global variables to receive the information update (`thetas[0], ..., thetas[5]`)
- We extract the needed data from the data structure with:
 - `thetas[0] = msg.position[0]`
 - `thetas[1] = msg.position[1]`
 - And so on...

Questions to answer for suction feedback

- What is the topic?
- What is the data type?
- What is the name of the variable?
- Where is the data we want in the data structure?
 - Note: There are two solutions to this question: An analog and a digital one.

Applying to suction feedback

- Create a subscriber function call
- Create a callback function
- Implement the feedback into your code
- Remember: Suction feedback will not be updated immediately upon turning on the gripper