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>
  • rosmsg list
  • rosmsg info <message_name>
How does a Subscriber work?

- 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/ROS/Tutorials/WritingPublisherSubscriber%28python%29](http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29)

```python
# Initialize subscriber to ur3/position and callback function
# each time data is published
sub_position = rospy.Subscriber('ur3/position', position, position_callback)
```
The callback function

• 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>
  • rosmsg list
  • rosmsg 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 rosrun 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
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
rosmsg 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.
rosmsg 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
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
rostopic echo /ur3/position/position[0]

• 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

- 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

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