# ROS BASICS

{SUBSCRIBER, LAUNCH FILE }

## RECAP

- ROS AND ROS BUILD SYSTEM
- CATKIN WORKSPACE
- NODES
- PACKAGES
- TOPICS
- MESSAGES
- PUBLISHER
- ROSPY

# ROS Client Libraries

| Client Library | Language | Comments |
|---|---|---|
| roscpp | C++ | Most widely used, high performance |
| rospy | Python | Good for rapid-prototyping and non-critical-path code |
| roslisp | LISP | Used for planning libraries |
| rosjava | Java | Android support |
| roslua | Lua | Light-weight scripting |
| roscs | Mono/.Net | Any Mono/.Net language |
| roseus | EusLisp | |
| PhaROS | Pharo Smalltalk | |
| rosR | R | Statistical programming |

Experimental

## SUBSCRIBER

## SUBSCRIBER

- A subscriber is a node that **reads** information from a topic

- Topics implement a **publish/subscribe** communications mechanism

## WRITING A SUBSCRIBER - GENERAL GUIDELINES

- Defining the environment
- Importing libraries
- Defining the callback function
- Creating a subscriber instance
  - Topic declaration
  - Callback function call
- Initialize node
- Define Main function

## SUBSCRIBER - HELLO WORLD

```
#!/usr/bin/env python

# import libraries
import rospy
import time
from std_msgs.msg import String

# callback function
def callback(msg):
        print msg.data

# initialize node
rospy.init_node('listener',anonymous=True)
# initialize the subscriber
Sub = rospy.Subscriber('chatter',String,callback)

# keep the node running
rospy.spin()
```

# SUBSCRIBER - CODE EXPLAINED

```
# ! /usr/bin/env python
```

➔ The first line makes sure your script is executed as a Python script

```
import rospy
from std_msgs.msg import String
```

➔ You need to import rospy if you are writing a ROS Node

➔ The std_msgs.msg import is to use the std_msgs/String message type for publishing

```
def callback(msg):
        print msg.data
```

➔ The callback functions handles the message data that gets subscribed

```
rospy.init_node('listener', anonymous=True)
```

➔ It tells rospy the name of your node -- until rospy has this information, it cannot start communicating with the ROS Master

➔ anonymous = True ensures that your node has a unique name by adding random numbers to the end of NAME

```
sub = rospy.Subscriber('chatter', String, callback)
```

➔ It declares that your node is subscribing to the chatter topic

➔ Whenever new message is received, callback is invoked with the message as the first argument

```
rospy.spin()
```

➔ It simply keeps your node from exiting until the node has been shutdown

# LAUNCH FILES

- ROS uses launch files in order to execute multiple programs

- A simple launch file is an xml file with .xml as file extension and a structure

```
<launch>
    <node name=' node name ' pkg=' package name' type=' python script name' output = ‘type of output’ />
</launch>
```

- All launch files are contained within a **<launch>**tag. Inside that tag, you can see a **<node>** tag, where we specify the following parameters:

    - **node name** = Name of the ROS node that will launch our Python file
    - **pkg** = Name of the package that contains the code of the ROS program to execute
    - **type** = Name of the program file that we want to execute
    - **output** = Through which channel you will print the output of the Python file

- **Syntax** for launching a launch file

```
roslaunch <package name> <launch file name>
```

## LAUNCH FILES FOR BASIC PUBLISHER AND SUBSCRIBER

➔ Publisher launch file

```
<launch>
    <node name='talker' pkg='assignment1' type='sample_publisher.py' />
</launch>
```

➔ Subscriber launch file

```
<launch>
  <node name='listener' pkg='assignment1' type='sample_subscriber.py' output='screen' />
</launch>
```

➔ Combined launch file

```
<launch>
  <node name='talker' pkg='assignment1' type='sample_publisher.py' />
  <node name='listener' pkg='assignment1' type='sample_subscriber.py' output='screen' />
</launch>
```

## TURTLESIM SAMPLE SUBSCRIBER

```
#!/usr/bin/env python

import rospy
from turtlesim.msg import Pose
import time

def callback(msg):

    print ('x location: ' + str(msg.x))
    print ('y location: ' + str(msg.y))


rospy.init_node('location_turtle', anonymous=True)
sub=rospy.Subscriber('/turtle1/pose', Pose, callback)

rospy.spin()
```

# Thank You