



Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences



ROS Nodes, Topics, and Messages

Foundation Course

August 29, 2019

Hassan Umari

1. Recap

2. ROS nodes in Python

- 2.1 A simple ROS node in Python
- 2.2 Writing a publisher node in Python
- 2.3 How to use ROS messages
- 2.4 Writing a subscriber node in Python

3. ROS messages



Recap

Summary of yesterday's session

- ROS is a collection of libraries and tools that helps you when you develop software for robots.
- ROS provides several ways to transfer data between nodes:
 1. ROS topics and messages (**publish/subscribe**).
 2. ROS services (**request/reply**).
 3. ROS actions (**request/reply**).
 4. Parameter server.

Recap

Summary of yesterday's session

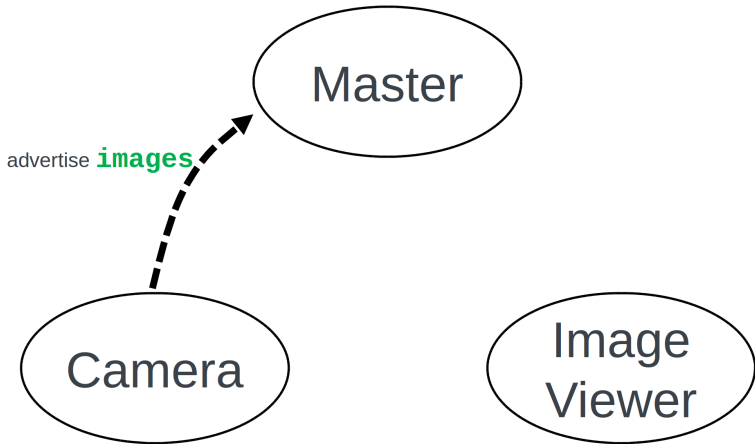
- We will focus today on ROS topics and messages..

```
graph TD; Master([Master]); Camera([Camera]); ImageViewer([Image Viewer]);
```

Master

Camera

Image
Viewer







1. Recap

2. ROS nodes in Python

- 2.1 A simple ROS node in Python
- 2.2 Writing a publisher node in Python
- 2.3 How to use ROS messages
- 2.4 Writing a subscriber node in Python

3. ROS messages



1. Recap

2. ROS nodes in Python

2.1 A simple ROS node in Python

2.2 Writing a publisher node in Python

2.3 How to use ROS messages

2.4 Writing a subscriber node in Python

3. ROS messages

A simple ROS node

../scripts/00_simple_node.py

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

import rospy
from time import sleep

rospy.init_node("print_text")

while True:
    print "Hello world!"
    sleep(1)
```

A simple ROS node

../scripts/01_simple_node.py

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

```
import rospy
```

```
rospy.init_node("print_text")  
rate = rospy.Rate(1)
```

```
while not rospy.is_shutdown():  
    print "Hello world!"  
    rate.sleep()
```

Writing a publisher node in Python

../scripts/02_simple_publisher.py

```
rospy.init_node('node name')
```

- nodes name must be unique. If you want to make sure the name of the node is unique:

```
rospy.init_node('node name', anonymous= True)
```

- node name will look like this:
/print_text_19637_1567065017476

Three ways to run a node

ROS Nodes

There are 3 ways to run a node:

1. Like you normally do (not recommended). Example (in case of python node):

```
python <file name>
```

2. using rosrun command:

```
rosrun <package name> <node name>
```

3. Using launch files. (we'll see it later)

Let's create a package first!

ROS Nodes

- ROS commands find your files (python scripts, cpp files, launch files, message definitions) if they are located in a package inside the workspace.
- Normally, a package looks like this:



include



msg



scripts



src



srv



CMakeLists.txt



package.xml

Let's create a package first!

ROS Nodes

- go to the README and do the steps for **creating a package**.

ROS commands

ROS Nodes

- Navigate to a ROS package directly:

```
roscd <package name>
```

- run a node without navigating to it's directory:

```
roslaunch <package name> <executable>
```

Let's create a package first!

ROS Nodes

- go to the README and do the steps for **running a node**.

More ROS commands

ROS Nodes, Topics, and Messages

- List all the running nodes::

```
rostopic list
```

- Get more info. about a certain node:

```
rostopic info <node name>
```

1. Recap

2. ROS nodes in Python

2.1 A simple ROS node in Python

2.2 Writing a publisher node in Python

2.3 How to use ROS messages

2.4 Writing a subscriber node in Python

3. ROS messages



Writing a publisher node in Python

ROS Nodes, Topics, and Messages

- Let's extend our previous node and make it publish a String ROS message.

Writing a publisher node in Python

../scripts/02_simple_publisher.py

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

import rospy

from std_msgs.msg import String

rospy.init_node('talker')

pub = rospy.Publisher('myFirstTopic', String, queue_size=10)

rate = rospy.Rate(1)

my_message = String()
my_message.data = "Hello there! How are you?"

while not rospy.is_shutdown():
    pub.publish(my_message)
    rate.sleep()
```

Writing a publisher node in Python

../scripts/02_simple_publisher.py

```
rospy.Publisher(name, data_class, queue_size)
```

- `name`: Name of the topic to publish on.
- `data_class`: The type of message. It is a ROS message class.
- `queue_size`: The size of the outgoing message queue.

Writing a publisher node in Python

../scripts/02_simple_publisher.py

```
rospy.Publisher(  
    name,  
    data_class,  
    subscriber_listener=None,  
    tcp_nodelay=False,  
    latch=False,  
    headers=None,  
    queue_size=None  
)
```


Writing a publisher node in Python

Things to note..

- ROS messages are implemented as classes.
- To publish a message you also need to define a **Publisher** class.
- Most of ROS concepts and functionalities are implemented as classes. This is why understanding OOP helps you understand ROS better.

Writing a publisher node in Python

ROS Nodes, Topics, and Messages

- Go to the README file and do the instructions of section:
some of ROS commands.

More ROS commands

ROS Nodes, Topics, and Messages

- Get the current list of topics:

```
rostopic list
```

- Print published messages:

```
rostopic echo <topic name>
```

More ROS commands

ROS Nodes, Topics, and Messages

- Publish a message from terminal:

```
rostopic pub <topic name> <msg type> <msg>
```

- Get message type of a topic:

```
rostopic type <topic name>
```

1. Recap

2. ROS nodes in Python

2.1 A simple ROS node in Python

2.2 Writing a publisher node in Python

2.3 How to use ROS messages

2.4 Writing a subscriber node in Python

3. ROS messages



How to use ROS messages

ROS Nodes, Topics, and Messages

- ROS messages are just classes with attributes you can fill.
- ROS messages are defined in a separate files and have to be placed in a package. (will be covered today).
- The following command can be used to see the class attributes, or the description, of a ROS message:

```
rosmmsg show <package/msg>
```

How to use ROS messages

Example

```
rosmmsg show std_msgs/String
```

The output is:

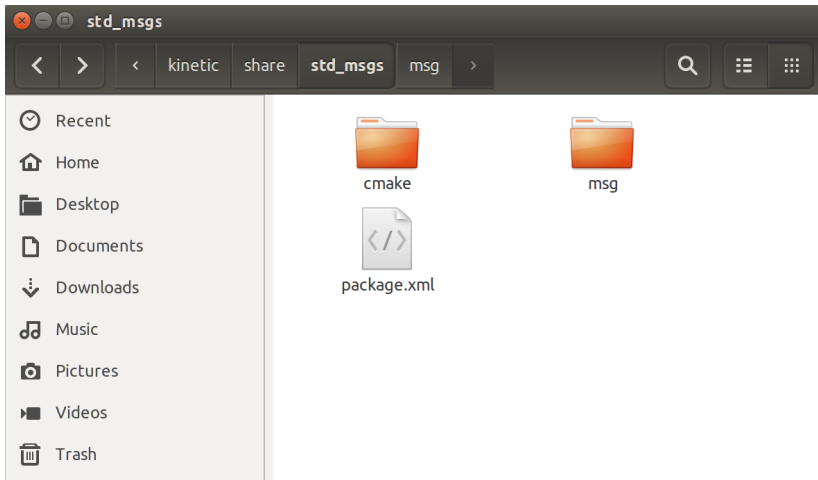
```
string data
```

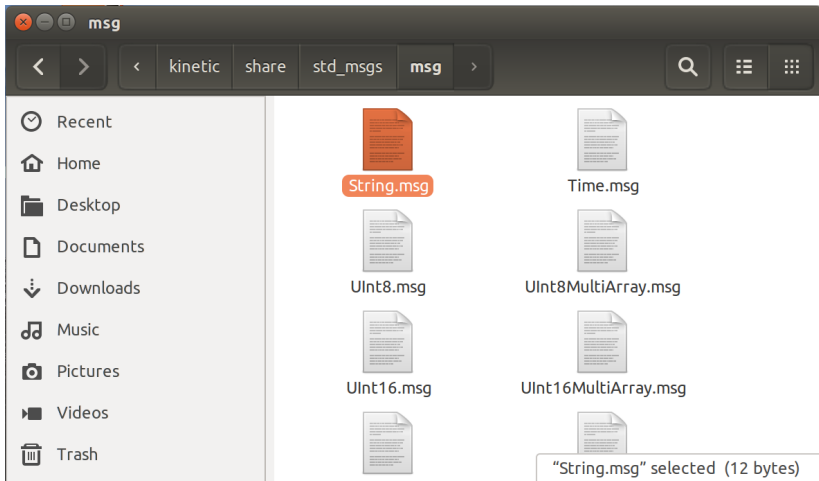
- It means ROS **String** message is a class with an attribute named **data** of type string (Python string).

How to use ROS messages

Importing a ROS message

- `String` message is located in the `std_msgs`.





How to use ROS messages

Importing a ROS message

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

Exercise 1

1. Recap

2. ROS nodes in Python

2.1 A simple ROS node in Python

2.2 Writing a publisher node in Python

2.3 How to use ROS messages

2.4 Writing a subscriber node in Python

3. ROS messages



Writing a subscriber node in Python

ROS Nodes, Topics, and Messages

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

Writing a publisher node in Python

../scripts/02_simple_publisher.py



1. Recap

2. ROS nodes in Python

- 2.1 A simple ROS node in Python
- 2.2 Writing a publisher node in Python
- 2.3 How to use ROS messages
- 2.4 Writing a subscriber node in Python

3. ROS messages



rospy reference

- rospy full documentation (all the classes, all the functions ..etc):

`http://docs.ros.org/kinetic/api/rospy/html/`