2022

INFO 802

Master Advanced Mechatronics

Luc Marechal

UNIVERSITÉ SAVOIE MONT BLANC

POLYTECH ANNECY-CHAMBÉRY

ROS

ROS Command Tools Tutorial 1

Objectives

At the end of this lecture, you are excepted to :

- Use ROS command line tools to get information on nodes, topics and message type
- Know what a ROS message is made up of.
- Find which library a ROS message comes from.
- Create a custom launch file.
- Achieve at least grade 80% of the Assignement

ROS Command Tools

Turtlesim

Turtle_teleop_key node

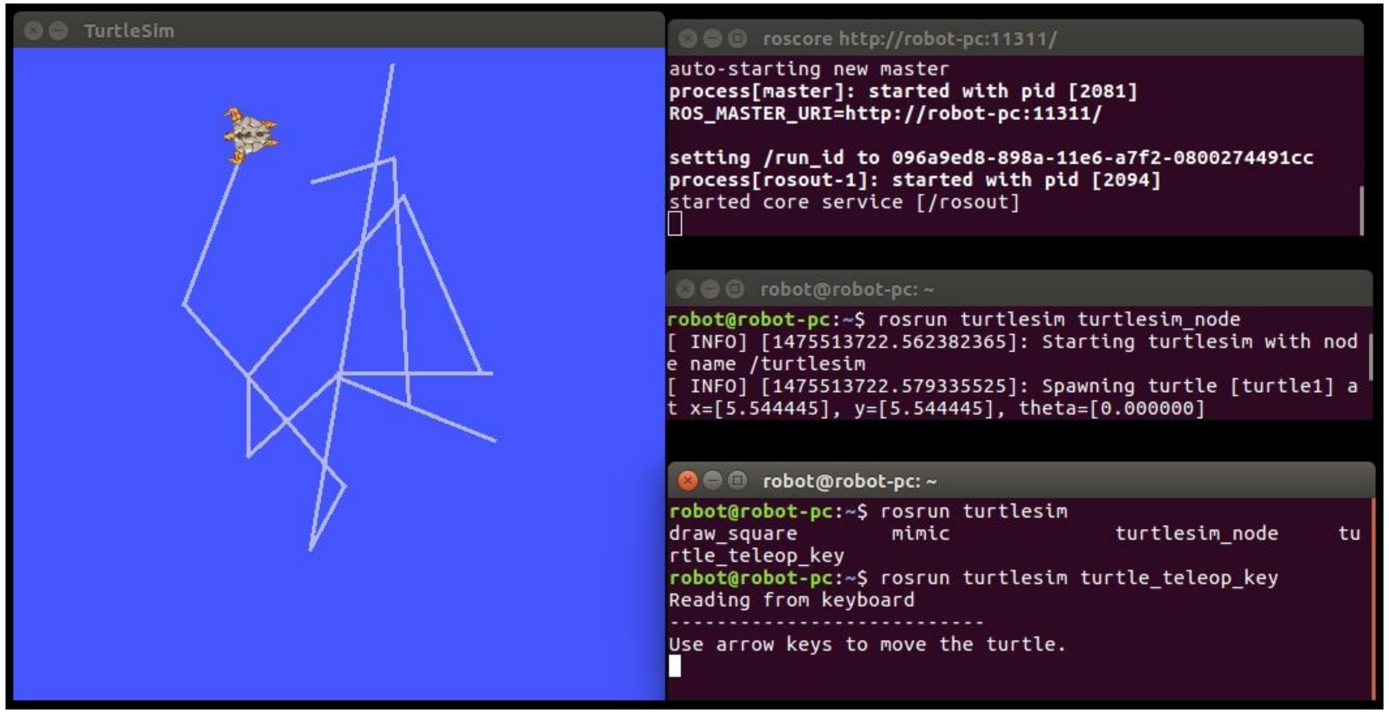
Test moving the turtle
(with the *turtle_teleop_key* node)

Recall: Open a terminal for each command

```
> roscore
```

```
> rosrun turtlesim turtlesim_node
```

```
> rosrun turtlesim turtle_teleop_key
```



```
auto-starting new master
process[master]: started with pid [2081]
ROS_MASTER_URI=http://robot-pc:11311/

setting /run_id to 096a9ed8-898a-11e6-a7f2-0800274491cc
process[rosout-1]: started with pid [2094]
started core service [/rosout]
```

```
robot@robot-pc:~$ rosrun turtlesim turtlesim_node
[ INFO] [1475513722.562382365]: Starting turtlesim with nod
e name /turtlesim
[ INFO] [1475513722.579335525]: Spawning turtle [turtle1] a
t x=[5.544445], y=[5.544445], theta=[0.000000]
```

```
robot@robot-pc:~$ rosrun turtlesim
draw_square        mimic              turtlesim_node     tu
rtle_teleop_key
robot@robot-pc:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

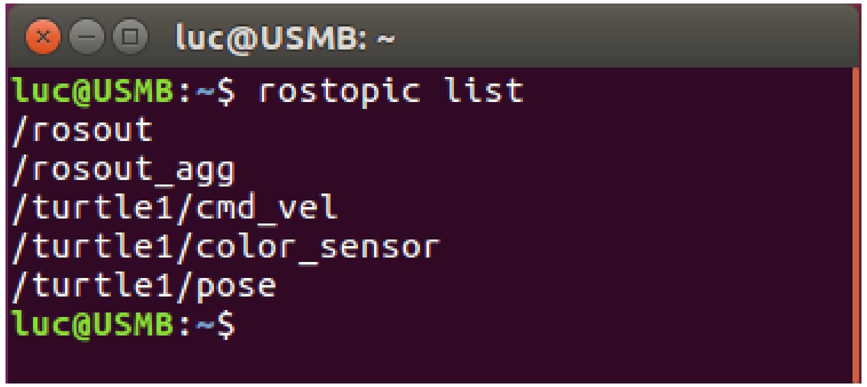
The terminal which *turtle_teleop_key* is running on MUST be selected.
Change the turtle's position by pressing arrow keys on the keyboard.

ROS Command Tools

topic

List all active topics on ROS:

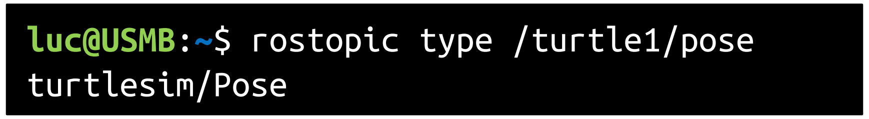
```
> rostopic list
```



```
luc@USMB:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
luc@USMB:~$
```

Display which message is used on a topic:

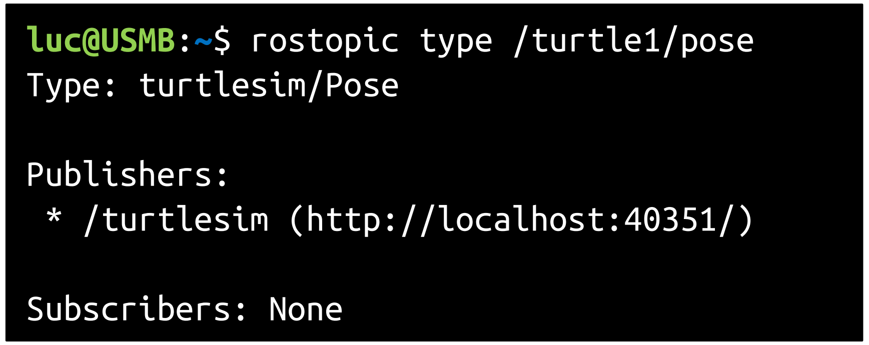
```
> rostopic type [topic_name]
```



```
luc@USMB:~$ rostopic type /turtle1/pose
turtlesim/Pose
```

Get more information on a topic:

```
> rostopic info [topic_name]
```



```
luc@USMB:~$ rostopic type /turtle1/pose
Type: turtlesim/Pose

Publishers:
 * /turtlesim (http://localhost:40351/)

Subscribers: None
```

ROS Command Tools

node

List all active node running on ROS:

```
> rosnode list
```

```
luc@USMB:~$ rosnode list
/rosout
/teleop_turtle
/turtlesim
```

Display information including publication/subscription:

```
> rosnode info [node_name]
```

```
luc@USMB:~$ rosnode info turtlesim
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Services:
 * /clear
 * /kill
 * /reset
 * /spawn
 * /turtle1/set_pen
 * /turtle1/teleport_absolute
```

ROS Command Tools

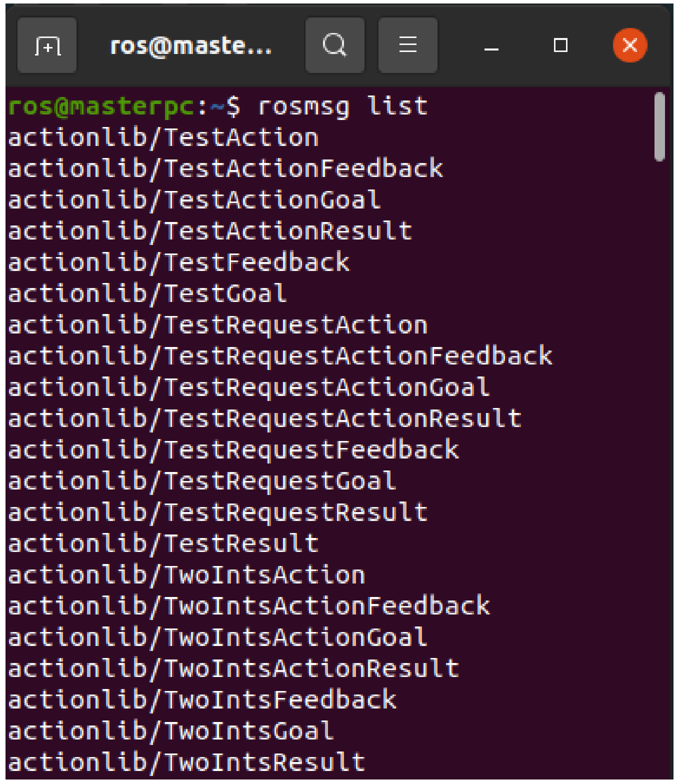
msg

Show all messages available in ROS:

```
> rosmsg list
```

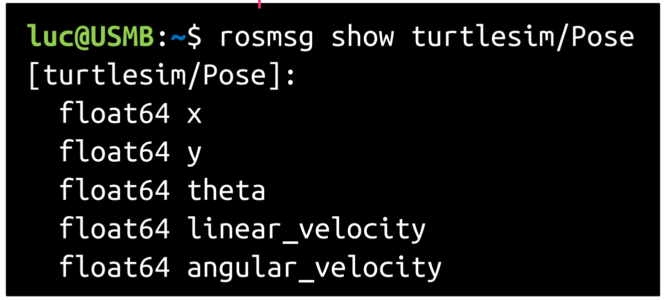
Show the content of a message type:

```
> rosmsg show [message_type]
```



```
ros@masterpc:~$ rosmsg list
actionlib/TestAction
actionlib/TestActionFeedback
actionlib/TestActionGoal
actionlib/TestActionResult
actionlib/TestFeedback
actionlib/TestGoal
actionlib/TestRequestAction
actionlib/TestRequestActionFeedback
actionlib/TestRequestActionGoal
actionlib/TestRequestActionResult
actionlib/TestRequestFeedback
actionlib/TestRequestGoal
actionlib/TestRequestResult
actionlib/TestResult
actionlib/TwoIntsAction
actionlib/TwoIntsActionFeedback
actionlib/TwoIntsActionGoal
actionlib/TwoIntsActionResult
actionlib/TwoIntsFeedback
actionlib/TwoIntsGoal
actionlib/TwoIntsResult
```

```
> rosmsg show turtlesim/Pose
```



```
luc@USMB:~$ rosmsg show turtlesim/Pose
[turtlesim/Pose]:
  float64 x
  float64 y
  float64 theta
  float64 linear_velocity
  float64 angular_velocity
```

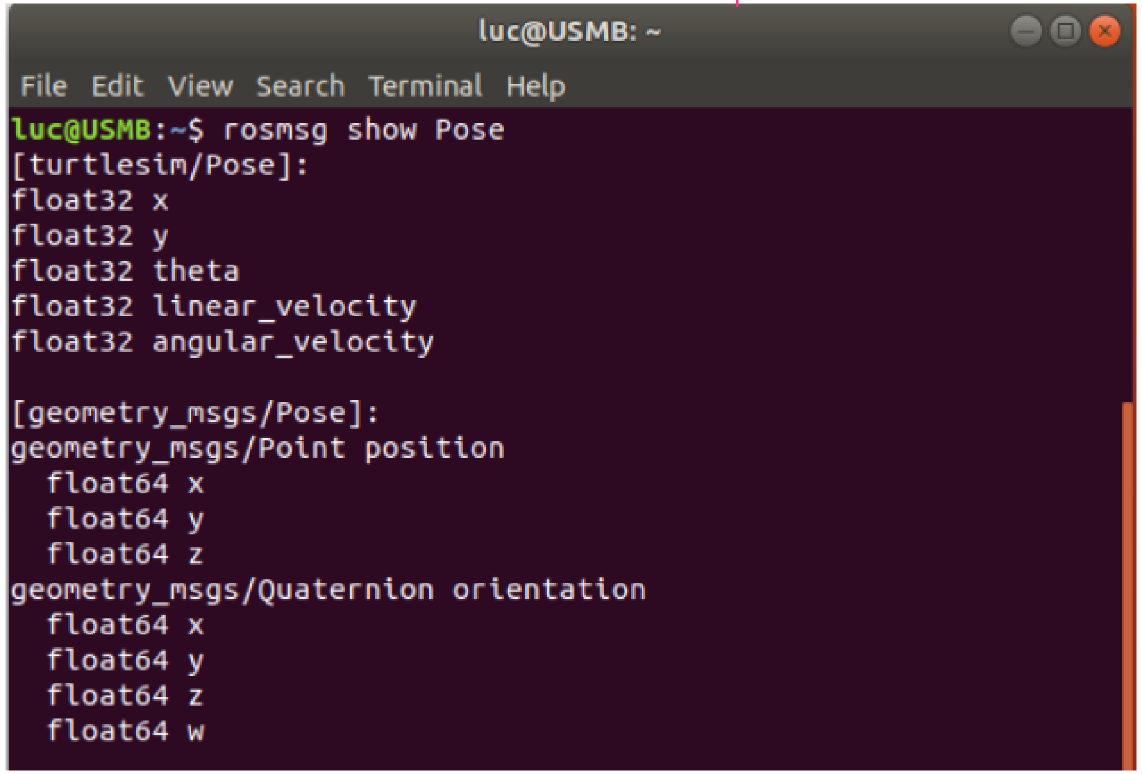
ROS Command Tools

msg

See message definition information:

```
> rosmsg show [message_type]
```

```
> rosmsg show Pose
```


luc@USMB: ~

```
luc@USMB:~$ rosmsg show Pose
[turtlesim/Pose]:
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity

[geometry_msgs/Pose]:
geometry_msgs/Point position
  float64 x
  float64 y
  float64 z
geometry_msgs/Quaternion orientation
  float64 x
  float64 y
  float64 z
  float64 w
```

The message of type *Pose* is defined in the package *turtlesim* but also in the package *geometry_msgs* but they are not the sames !

ROS Command Tools

System File

Get information on packages

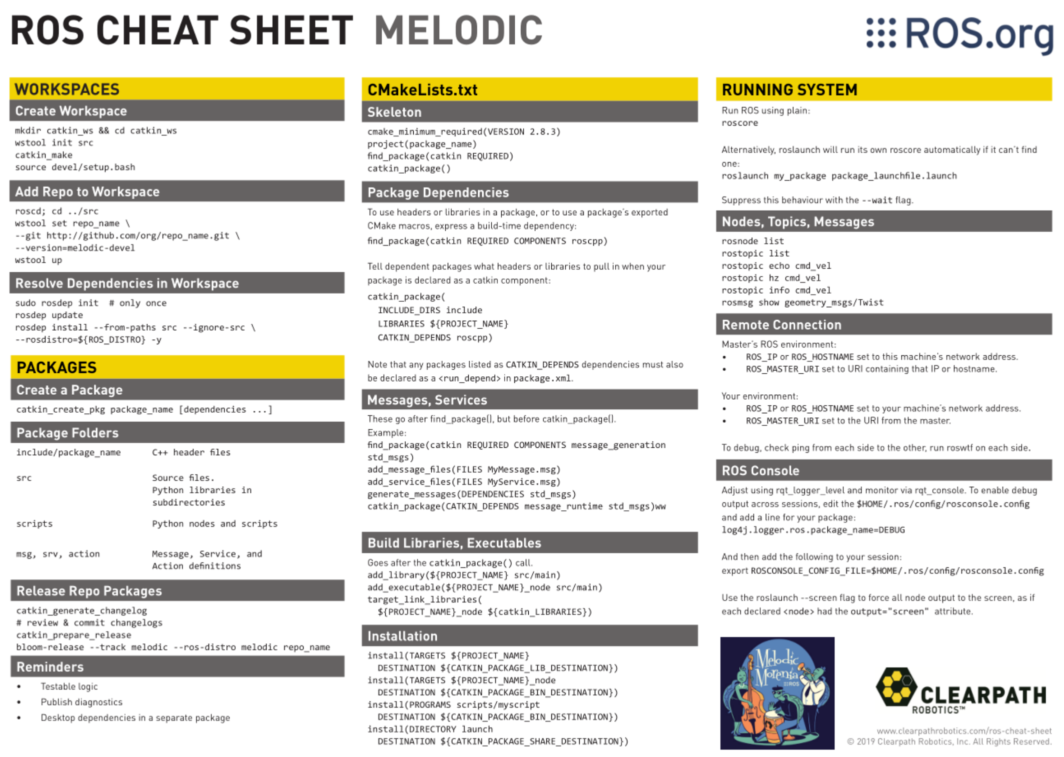
```
> rospack find [package_name]
```

Change directory (cd) directly to a package or a stack

```
> roscd [location_name[/subdir]]
```

ls directly in a package by name rather than by absolute path

```
> rosls [location_name[/subdir]]
```



ROS CHEAT SHEET MELODIC

ROS.org

WORKSPACES

Create Workspace

```
mkdir catkin_ws && cd catkin_ws
wstool init src
catkin_make
source devel/setup.bash
```

Add Repo to Workspace

```
roscd; cd ../src
wstool set repo_name \
--git http://github.com/org/repo_name.git \
--version=melodic-devel
wstool up
```

Resolve Dependencies in Workspace

```
sudo rosdep init  # only once
rosdep update
rosdep install --from-paths src --ignore-src \
--rosdistro=${ROS_DISTRO} -y
```

PACKAGES

Create a Package

```
catkin_create_pkg package_name [dependencies ...]
```

Package Folders

```
include/package_name      C++ header files

src                       Source files.
                          Python libraries in
                          subdirectories

scripts                   Python nodes and scripts

msg, srv, action          Message, Service, and
                          Action definitions
```

Release Repo Packages

```
catkin_generate_changelog
# review & commit changelogs
catkin_prepare_release
bloom-release --track melodic --ros-distro melodic repo_name
```

Reminders

- Testable logic
- Publish diagnostics
- Desktop dependencies in a separate package

CMakeLists.txt

Skeleton

```
cmake_minimum_required(VERSION 2.8.3)
project(package_name)
find_package(catkin REQUIRED)
catkin_package()
```

Package Dependencies

To use headers or libraries in a package, or to use a package's exported CMake macros, express a build-time dependency:

```
find_package(catkin REQUIRED COMPONENTS roscpp)
```

Tell dependent packages what headers or libraries to pull in when your package is declared as a catkin component:

```
catkin_package(
  INCLUDE_DIRS include
  LIBRARIES ${PROJECT_NAME}
  CATKIN_DEPENDS roscpp)
```

Note that any packages listed as CATKIN_DEPENDS dependencies must also be declared as a <run_depend> in package.xml.

Messages, Services

These go after find_package(), but before catkin_package().
Example:

```
find_package(catkin REQUIRED COMPONENTS message_generation
std_msgs)
add_message_files(FILES MyMessage.msg)
add_service_files(FILES MyService.msg)
generate_messages(DEPENDENCIES std_msgs)
catkin_package(CATKIN_DEPENDS message_runtime std_msgs)ww
```

Build Libraries, Executables

Goes after the catkin_package() call.

```
add_library(${PROJECT_NAME} src/main)
add_executable(${PROJECT_NAME}_node src/main)
target_link_libraries(
  ${PROJECT_NAME}_node ${catkin_LIBRARIES})
```

Installation

```
install(TARGETS ${PROJECT_NAME}
  DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION})
install(TARGETS ${PROJECT_NAME}_node
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
install(PROGRAMS scripts/myscript
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
install(DIRECTORY launch
  DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION})
```

RUNNING SYSTEM

Run ROS using plain:
```
roscore
```

Alternatively, roslaunch will run its own roscore automatically if it can't find one:
```
roslaunch my_package package_launchfile.launch
```

Suppress this behaviour with the --wait flag.

Nodes, Topics, Messages

```
rosnode list
rostopic list
rostopic echo cmd_vel
rostopic hz cmd_vel
rostopic info cmd_vel
rosmsg show geometry_msgs/Twist
```

Remote Connection

Master's ROS environment:
- ROS_IP or ROS_HOSTNAME set to this machine's network address.
- ROS_MASTER_URI set to URI containing that IP or hostname.

Your environment:
- ROS_IP or ROS_HOSTNAME set to your machine's network address.
- ROS_MASTER_URI set to the URI from the master.

To debug, check ping from each side to the other, run roswtf on each side.

ROS Console

Adjust using rqt_logger_level and monitor via rqt_console. To enable debug output across sessions, edit the $HOME/.ros/config/rosconsole.config and add a line for your package:
```
log4j.logger.ros.package_name=DEBUG
```

And then add the following to your session:
export ROSCONSOLE_CONFIG_FILE=$HOME/.ros/config/rosconsole.config

Use the roslaunch --screen flag to force all node output to the screen, as if each declared <node> had the output="screen" attribute.

CLEARPATH ROBOTICS™

www.clearpathrobotics.com/ros-cheat-sheet

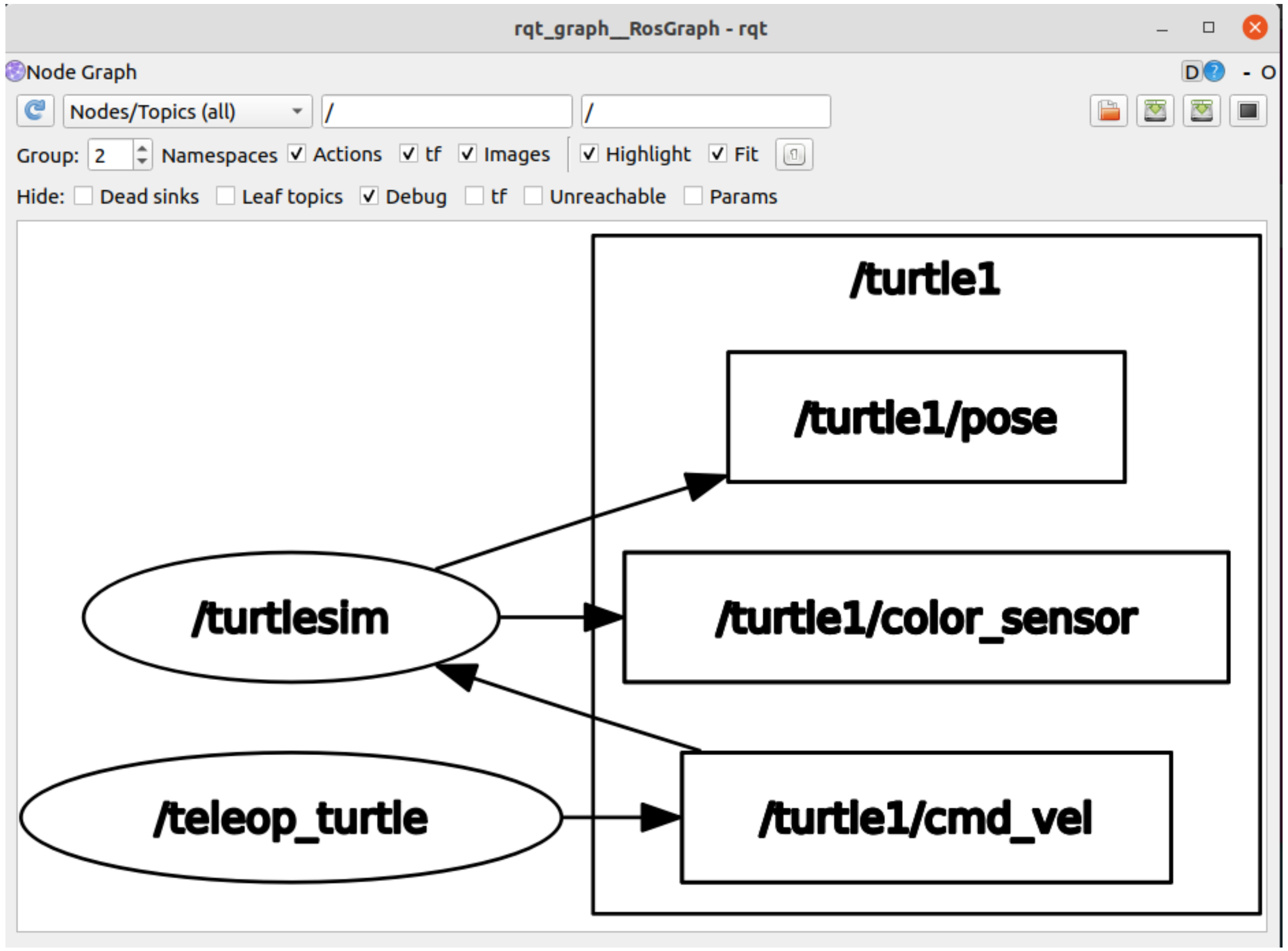
More info

http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem

ROS computation graph *rqt*

Visualize running topics and nodes

```
> rosrun rqt_graph rqt_graph
```


rqt_graph__RosGraph - rqt

Node Graph

Nodes/Topics (all) | / | /

Group: 2 Namespaces [✓] Actions [✓] tf [✓] Images [✓] Highlight [✓] Fit

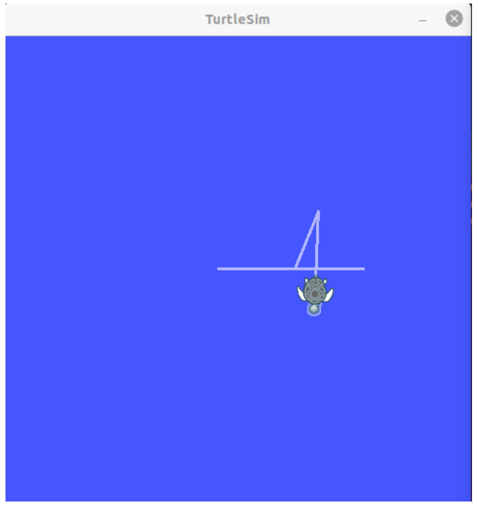
Hide: [ ] Dead sinks [ ] Leaf topics [✓] Debug [ ] tf [ ] Unreachable [ ] Params

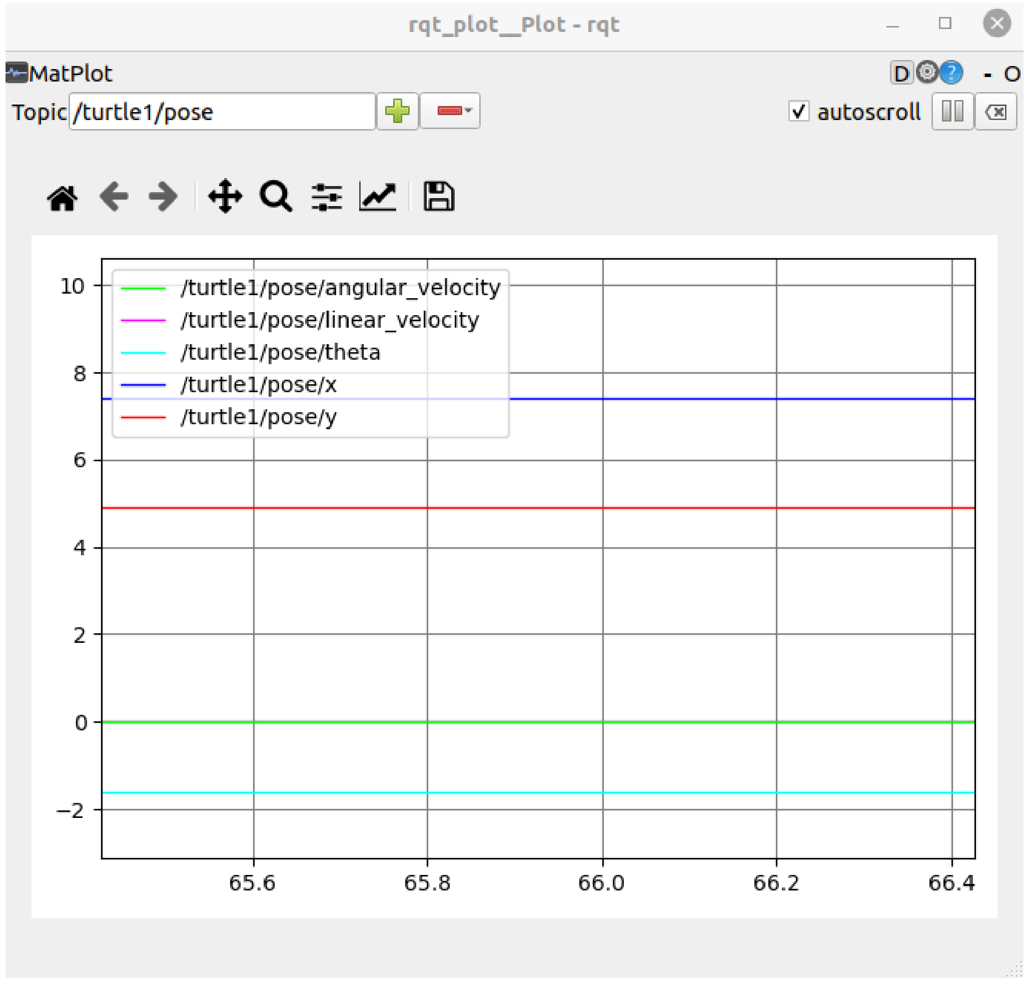
ROS computation graph *rqt*

Visualize running topics and nodes

```
> rosrun rqt_plot rqt_plot
```

It shows the values published on a topic





ROS computation graph *rqt*

- *rqt_graph* creates a dynamic graph of what's going on in the system
- *rqt_console* attaches to ROS's logging framework to display output from nodes. rqt_logger_level allows us to change the verbosity level (DEBUG, WARN, INFO, and ERROR) of nodes as they run.

- Prerequisit: Install rqt package

```
> sudo apt-get install ros-noetic-rqt ros-noetic-rqt-common-plugins
```

Launch rqt_console

```
> rosrun rqt_console rqt_console
```

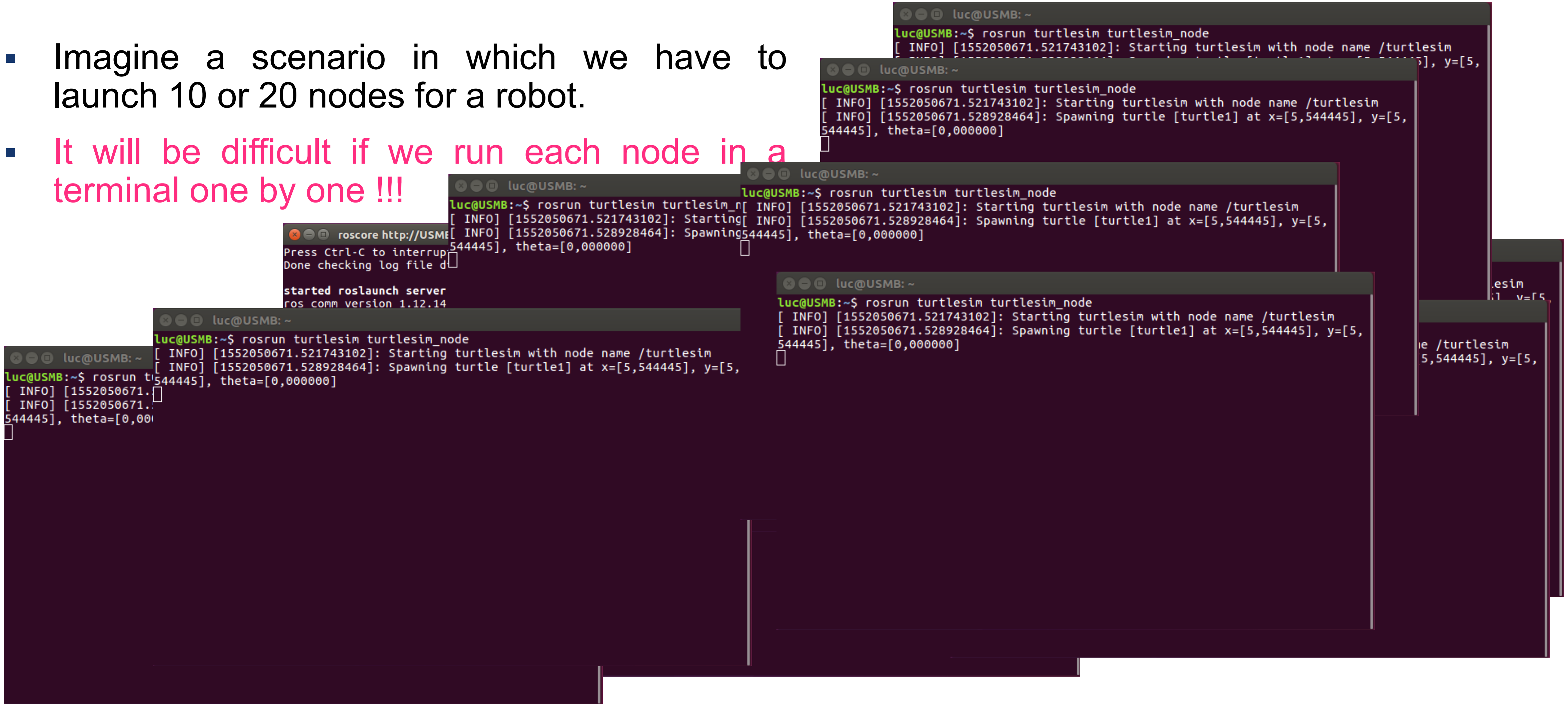
Launch rosrun rqt_logger_level rqt_logger_level (in an other terminal)

```
> rosrun rqt_logger_level rqt_logger_level
```

ROS Launch files

ROS Launch

- Imagine a scenario in which we have to launch 10 or 20 nodes for a robot.
- It will be difficult if we run each node in a terminal one by one !!!



```
luc@USMB: ~
luc@USMB:~$ rosrun turtlesim turtlesim_node
[ INFO] [1552050671.521743102]: Starting turtlesim with node name /turtlesim
```

```
luc@USMB: ~
luc@USMB:~$ rosrun turtlesim turtlesim_node
[ INFO] [1552050671.521743102]: Starting turtlesim with node name /turtlesim
[ INFO] [1552050671.528928464]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
luc@USMB: ~
luc@USMB:~$ rosrun turtlesim turtlesim_node
[ INFO] [1552050671.521743102]: Starting turtlesim with node name /turtlesim
[ INFO] [1552050671.528928464]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
luc@USMB: ~
luc@USMB:~$ rosrun turtlesim turtlesim_node
[ INFO] [1552050671.521743102]: Starting turtlesim with node name /turtlesim
[ INFO] [1552050671.528928464]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
luc@USMB: ~
luc@USMB:~$ rosrun turtlesim turtlesim_n
[ INFO] [1552050671.521743102]: Starting
[ INFO] [1552050671.528928464]: Spawning
544445], theta=[0,000000]
```

```
roscore http://USMB
Press Ctrl-C to interrup
Done checking log file d

started roslaunch server
ros_comm version 1.12.14
```

```
luc@USMB: ~
luc@USMB:~$ rosrun turtlesim turtlesim_node
[ INFO] [1552050671.521743102]: Starting turtlesim with node name /turtlesim
[ INFO] [1552050671.528928464]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
luc@USMB: ~
luc@USMB:~$ rosrun tu
[ INFO] [1552050671.
[ INFO] [1552050671.
544445], theta=[0,00
```

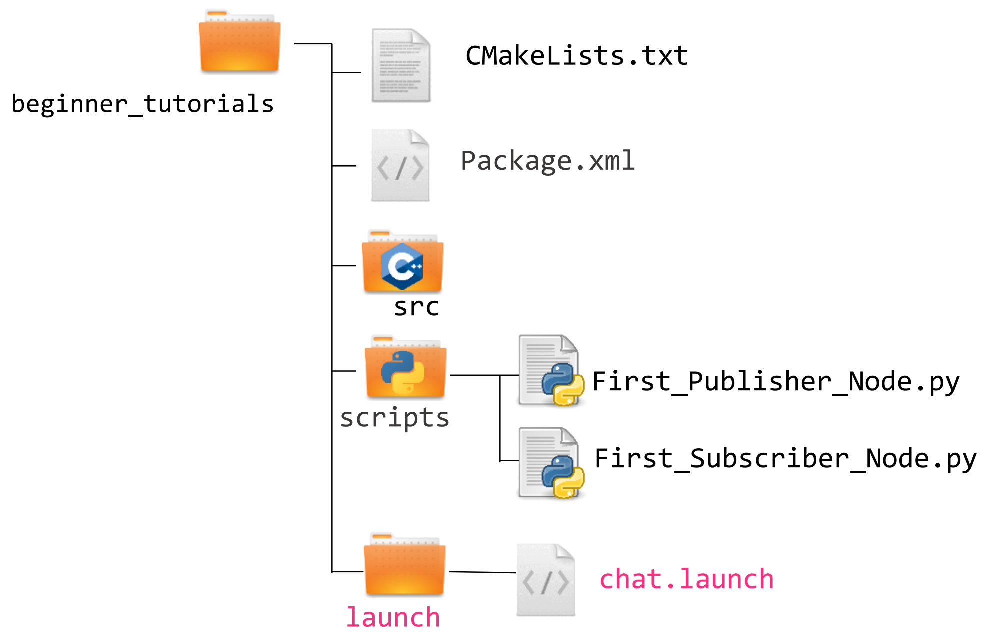
```
esim
], y=[5,
```

```
e /turtlesim
5,544445], y=[5,
```

ROS Launch

- *launch* is a tool for launching multiple nodes (as well as setting parameters)
- written in XML but file suffix: **.launch*
- the launch file needs to be located in a folder named “launch” inside de package folder

- If not yet running, launch automatically starts a roscore



Example

The file *chat.launch* is created in order to launch the node :

First_Publisher_Node.py and *First_Subscriber_Node.py*

More info
http://wiki.ros.org/roslaunch

ROS Launch

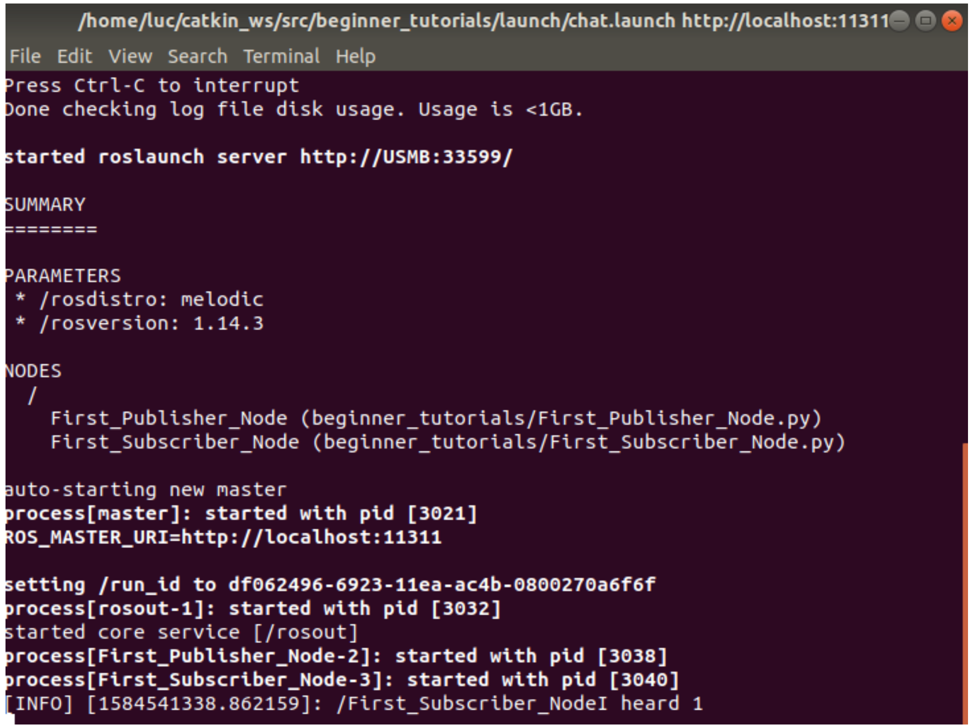
Start a launch file from a package with

```
> roslaunch [package_name] [file_name.launch]
```

Or browse to the folder and start a launch file with

```
> roslaunch [file_name.launch]
```

Example console output for:
> `roslaunch beginner_tutorials chat.launch`


/home/luc/catkin_ws/src/beginner_tutorials/launch/chat.launch http://localhost:11311
File Edit View Search Terminal Help

```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://USMB:33599/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.3

NODES
  /
    First_Publisher_Node (beginner_tutorials/First_Publisher_Node.py)
    First_Subscriber_Node (beginner_tutorials/First_Subscriber_Node.py)

auto-starting new master
process[master]: started with pid [3021]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to df062496-6923-11ea-ac4b-0800270a6f6f
process[rosout-1]: started with pid [3032]
started core service [/rosout]
process[First_Publisher_Node-2]: started with pid [3038]
process[First_Subscriber_Node-3]: started with pid [3040]
[INFO] [1584541338.862159]: /First_Subscriber_NodeI heard 1
```

More info
http://wiki.ros.org/roslaunch

ROS Launch

Other example

Turtle.launch

```
<launch>
  <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node"/>
  <node name="turtle_teleop_key" pkg=" turtlesim " type="turtle_teleop_key"/>
</launch>
```

- **launch**: root element of the Launch files. This is an XML document, and every XML document has one
- **node**: each <node> tag specifies a node to be launched
- **name**: name of the node (free to choose)
- **pkg**: package containing the node
- **type**: the executable name (if the executable is a python file, don't forget the .py extention)
- **output**: specifies where to output log messages (screen -> consol, log -> log file)

  output="screen" makes the ROS log messages appear on the launch terminal window

ROS Launch

Other example

`turtlesim_node` is NOT a python script

`turtlesim_target_node.py` IS a python script
The .py extention is needed

Turtle.launch

```
<launch>
  <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node"/>
  <node name="turtlesim_target_node" pkg="beginner_tutorials" type="turtlesim_target_node.py" output="screen"/>
</launch>
```

- **launch**: root element of the Launch files. This is an XML document, and every XML document has one
- **node**: each <node> tag specifies a node to be launched
- **name**: name of the node (free to choose)
- **pkg**: package containing the node
- **type**: the executable name (if the executable is a python file, don't forget the .py extention)
- **output**: specifies where to output log messages (screen -> consol, log -> log file)

  output="screen" makes the ROS log messages appear on the launch terminal window

ROS Launch

File Structure

if you don't put the option output="screen" you will not be able to see messages on the Terminal

chat.launch

```
<launch>
  <node name="First_Publisher_Node" pkg="beginner_tutorials" type="First_Publisher_Node.py"/>
  <node name="First_Subscriber_Node" pkg="beginner_tutorials" type="First_Subscriber_Node.py" output="screen"/>
</launch>
```

- **launch**: root element of the Launch files. This is an XML document, and every XML document has one
- **node**: each <node> tag specifies a node to be launched
- **name**: name of the node (free to choose)
- **pkg**: package containing the node
- **type**: the executable name (if the executable is a python file, don't forget the .py extention)
- **output**: specifies where to output log messages (screen -> consol, log -> log file)

  output="screen" makes the ROS log messages appear on the launch terminal window

ROS Launch

Arguments

- Create re-usable launch files with `<arg>` tag, which works like a parameter (default optional)

```
<arg name="arg_name" default="default_value"/>
```

- Use arguments in launch file with

```
$(arg arg_name)
```

- When launching, arguments can be set with

```
> roslaunch launch_file.launch arg_name:=value
```

range_world.launch (simplified)

```
<?xml version="1.0"?>
<launch>
  <arg name="use_sim_time" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>

  <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

  <include file="$(find gazebo_ros)
                              /launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/
                      test/test_worlds/$(arg world).world"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
  </include>
</launch>
```

ROS Launch

Including Other Launch Files

- Include other launch files with <include> tag to organize large projects

```
<include file="package_name"/>
```

- Find the system path to other packages with

```
$(find package_name)
```

- Pass arguments to the included file

```
<arg name="arg_name" value="value"/>
```

range_world.launch *(simplified)*

```
<?xml version="1.0"?>
<launch>
  <arg name="use_sim_time" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>

  <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

  <include file="$(find gazebo_ros)
                             /launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/
                        test/test_worlds/$(arg world).world"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
  </include>
</launch>
```

Assignement

- Due to Wednesday April 8, 2020. (6pm max. deadline)
- Answer to question 1 to 5 and leave your responses on Moodle

run the node *my_position.py* from the package *assignments_pkg*

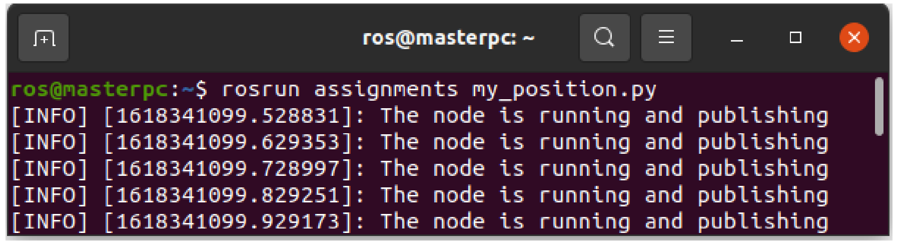
20% 1. Find which topic the *my_position.py* publishes to

20% 2. Find what message is used on this topic

40% 3. Edit the node *get_position.py* in the package *assignments* so it subscribes to the topic …

20% 4. … and prints in the Terminal only the y value of the coordinates.

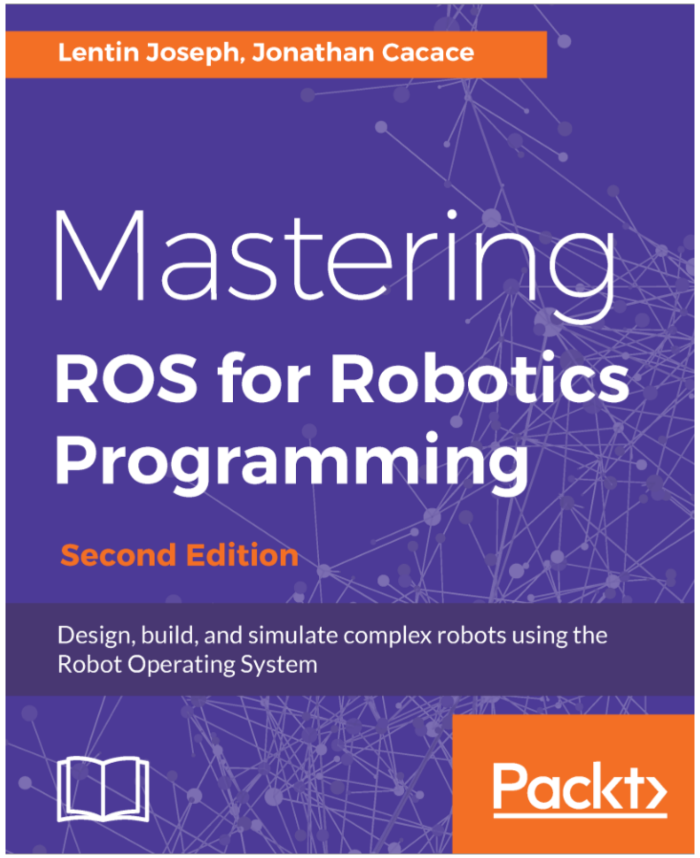
5. run and validate the node *get_position.py*


```
ros@masterpc:~$ rosrun assignments my_position.py
[INFO] [1618341099.528831]: The node is running and publishing
[INFO] [1618341099.629353]: The node is running and publishing
[INFO] [1618341099.728997]: The node is running and publishing
[INFO] [1618341099.829251]: The node is running and publishing
[INFO] [1618341099.929173]: The node is running and publishing
```

Further References

- ROS Wiki
  - http://wiki.ros.org/
- Installation
  - http://wiki.ros.org/ROS/Installation
- Tutorials
  - http://wiki.ros.org/ROS/Tutorials
- Available packages
  - http://www.ros.org/browse/
- ROS Cheat Sheet
  - https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/
  - https://kapeli.com/cheat_sheets/ROS.docset/
- ROS Best Practices
  - https://github.com/leggedrobotics/ros_best_practices/wiki
- ROS Package Template
  - https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template

Relevant books

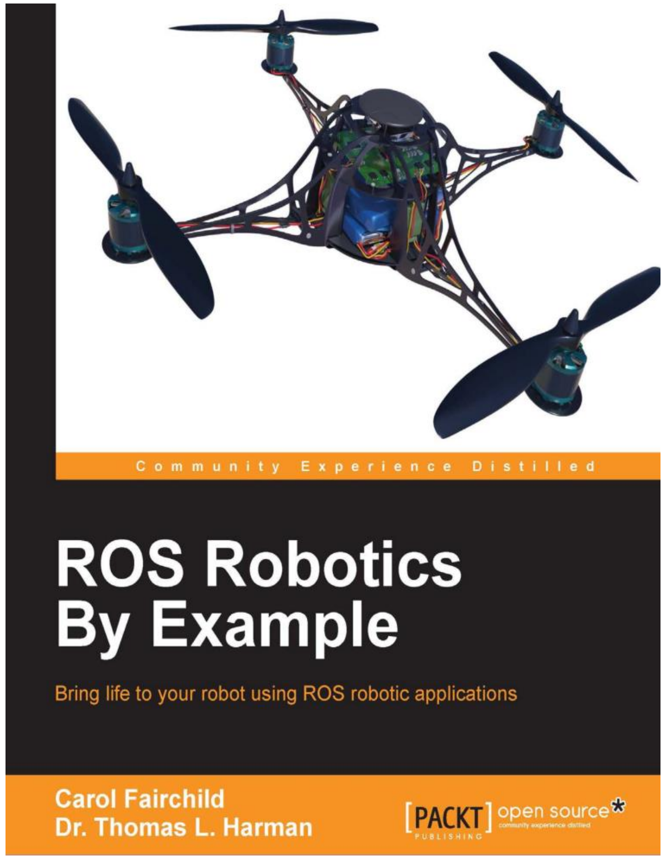

Lentin Joseph, Jonathan Cacace

Mastering ROS for Robotics Programming

Second Edition

Design, build, and simulate complex robots using the Robot Operating System

Packt>

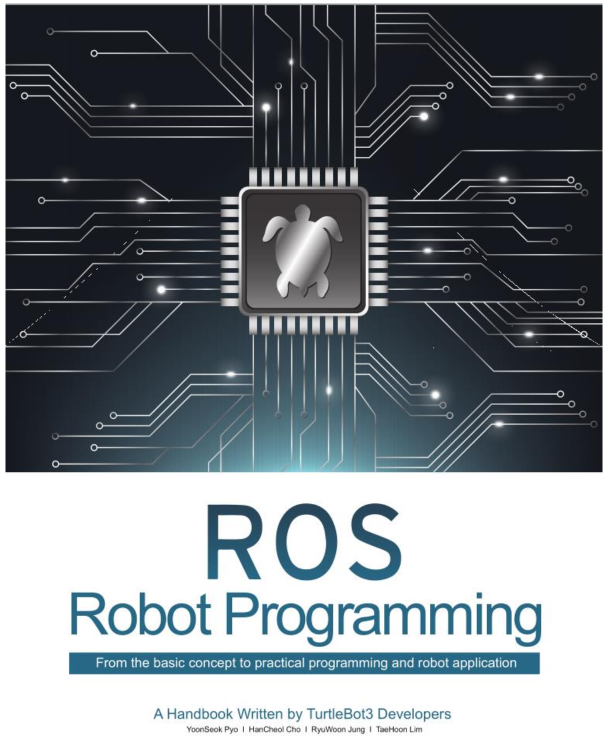

Community Experience Distilled

ROS Robotics By Example

Bring life to your robot using ROS robotic applications

Carol Fairchild
Dr. Thomas L. Harman

[PACKT] open source* PUBLISHING


ROS
Robot Programming

From the basic concept to practical programming and robot application

A Handbook Written by TurtleBot3 Developers
YoonSeok Pyo I HanCheol Cho I RyuWoon Jung I TaeHoon Lim

Contact Information

Université Savoie Mont Blanc

Polytech' Annecy Chambery
Chemin de Bellevue
74940 Annecy
France

https://www.polytech.univ-savoie.fr

Lecturer

Luc Marechal (luc.marechal@univ-smb.fr)
SYMME Lab (Systems and Materials for Mechatronics)