

# COMP20170

## Introduction to Robotics

### ::: ROS.org

**Assoc. Prof. Eleni Mangina**

Room B2.05

Teaching Assistant & Mentor:

**Evan O’Keeffe (PhD student – IRC Scholar)**

School of Computer Science

Ext. 2858

eleni.mangina@ucd.ie

# ROS.org

- ROS architecture & philosophy
- ROS master, nodes, and topics
- Console commands
- Catkin workspace and build system
- Launch-files
- Gazebo simulator

# ::: ROS.org

## What is ROS?

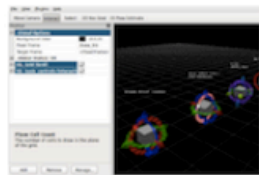
**ROS = Robot Operating System**



Plumbing

- Process management
- Inter-process communication
- Device drivers

+



Tools

- Simulation
- Visualization
- Graphical user interface
- Data logging

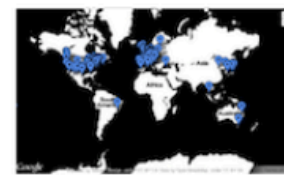
+



Capabilities

- Control
- Planning
- Perception
- Mapping
- Manipulation

+



Ecosystem

- Package organization
- Software distribution
- Documentation
- Tutorials

ros.org

## History of ROS

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory
- Since 2013 managed by OSRF
- Today used by many robots, universities and companies
- De facto standard for robot programming





## ROS Philosophy

- **Peer to peer**  
Individual programs communicate over defined API (ROS *messages*, *services*, etc.).
- **Distributed**  
Programs can be run on multiple computers and communicate over the network.
- **Multi-lingual**  
ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
- **Light-weight**  
Stand-alone libraries are wrapped around with a thin ROS layer.
- **Free and open-source**  
Most ROS software is open-source and free to use.



## ROS Workspace Environment

- Defines context for the current workspace
- Default workspace loaded with

```
source .bashrc
```

Overlay your *catkin workspace* with

```
> cd ~/catkin_ws  
> source devel/setup.bash
```

Check your workspace with

```
> echo $ROS_PACKAGE_PATH
```

This is  
already  
setup in the  
provided  
installation.

See setup with

```
> cat ~/.bashrc
```

### More info

<http://wiki.ros.org/indigo/Installation/Ubuntu>

<http://wiki.ros.org/catkin/workspaces>

```
lbros@lbros-VirtualBox:~/catkin_ws$ source devel/setup.bash  
lbros@lbros-VirtualBox:~/catkin_ws$ echo $ROS_PACKAGE_PATH  
/home/lbros/catkin_ws/src:/opt/ros/kinetic/share  
lbros@lbros-VirtualBox:~/catkin_ws$
```



## ROS Master

ROS Master

- Manages the communication between nodes
- Every node registers at startup with the master

Start a master with

```
> roscore
```

```
lbros@lbros-VirtualBox:~/catkin_ws$ roscore
... logging to /home/lbros/.ros/log/0e9e3440-f681-11e7-be49-08002776b3d2/roslaun
ch-lbros-VirtualBox-12722.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://lbros-VirtualBox:43513/
ros_comm version 1.12.12

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.12

NODES

auto-starting new master
process[master]: started with pid [12733]
ROS_MASTER_URI=http://lbros-VirtualBox:11311/

setting /run_id to 0e9e3440-f681-11e7-be49-08002776b3d2
process[rosout-1]: started with pid [12746]
started core service [/rosout]
```

**More info**

<http://wiki.ros.org/Master>



## ROS Nodes

- Single-purpose, executable program
- Individually compiled, executed, and managed
- Organized in *packages*

Run a node with

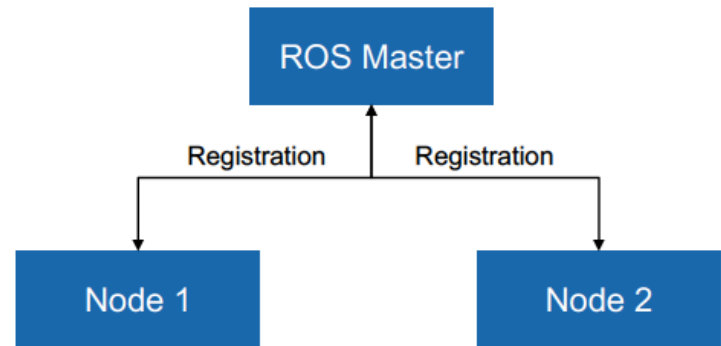
```
> rosrun package_name node_name
```

See active nodes with

```
> rosnodetool list
```

Retrieve information about a node with

```
> rosnodetool info node_name
```



**More info**

<http://wiki.ros.org/rosnode>



## ROS Topics

- Nodes communicate over *topics*
  - Nodes can *publish* or *subscribe* to a topic
  - Typically, 1 publisher and  $n$  subscribers
- Topic is a name for a stream of *messages*

List active topics with

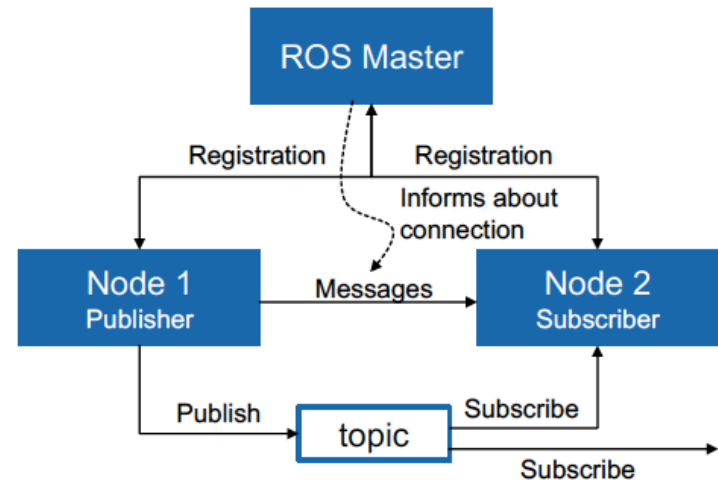
```
> rostopic list
```

Subscribe and print the contents of a topic with

```
> rostopic echo /topic
```

Show information about a topic with

```
> rostopic info /topic
```



**More info**

<http://wiki.ros.org/rostopic>

## ROS Messages

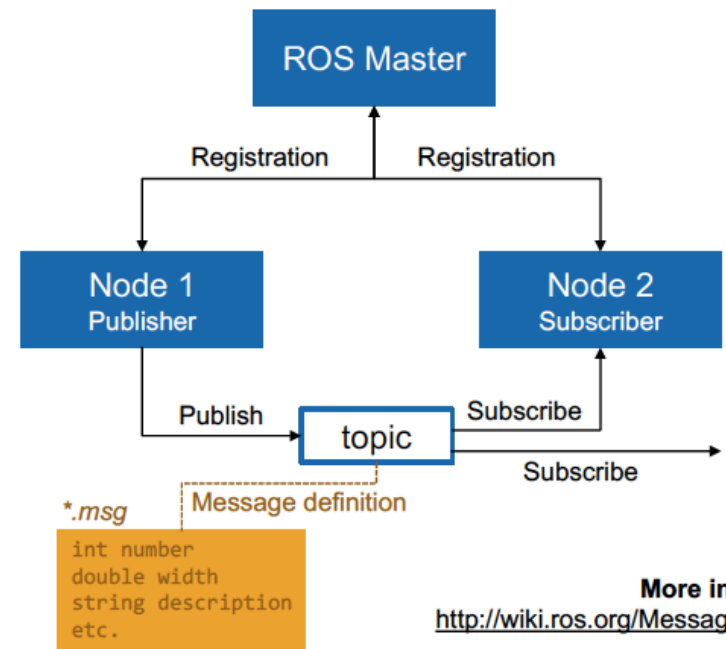
- Data structure defining the *type* of a topic
- Comprised of a nested structure of integers, floats, booleans, strings etc. and arrays of objects
- Defined in *\*.msg* files

See the type of a topic

```
> rostopic type /topic
```

Publish a message to a topic

```
> rostopic pub /topic type args
```



## ROS Messages

### Pose Stamped Example

*geometry\_msgs/Point.msg*

```
float64 x
float64 y
float64 z
```

*sensor\_msgs/Image.msg*

```
std_msgs/Header header
uint32 seq
time stamp
string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

*geometry\_msgs/PoseStamped.msg*

```
std_msgs/Header header
uint32 seq
time stamp
string frame_id
geometry_msgs/Pose pose
→ geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
    geometry_msgs/Quaternion
orientation
    float64 x
    float64 y
    float64 z
    float64 w
```



## Example

### Console Tab Nr. 1 – Starting a *roscore*

Start a roscore with

```
> roscore
```

A terminal window titled "roscore http://lbros-VirtualBox:11311/" with a menu bar (File, Edit, Tabs, Help). The terminal output shows the process starting, logging to a file, checking disk usage, and starting the roslaunch server. It then displays a summary of parameters and nodes.

```
setting /run_id to 85974078-f681-11e7-be49-08002776b3d2
process[rosout-1]: started with pid [12803]
started core service [/rosout]
^C[rosout-1] killing on exit
[master] killing on exit
shutting down processing monitor...
... shutting down processing monitor complete
done
lbros@lbros-VirtualBox:~/catkin_ws$ roscore
... logging to /home/lbros/.ros/log/f2a480ca-f692-11e7-be49-08002776b3d2/roslaunch-lbros-VirtualBox-13376.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://lbros-VirtualBox:38749/
ros_comm version 1.12.12

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.12

NODES

auto-starting new master
process[master]: started with pid [13387]
ROS_MASTER_URI=http://lbros-VirtualBox:11311/

setting /run_id to f2a480ca-f692-11e7-be49-08002776b3d2
process[rosout-1]: started with pid [13400]
started core service [/rosout]
```



## Example

### Console Tab Nr. 2 – Starting a *talker* node

Run a talker demo node with

```
> rosruncatkin_ws rosctutorials talker
```

A terminal window titled "lbros@lbros-VirtualBox: ~/catkin\_ws" with a menu bar (File, Edit, Tabs, Help) and two tabs ("roscore http..." and "lbros@lbros..."). The terminal displays a series of log messages from the "talker" node, each starting with "[ INFO]" followed by a timestamp and the message "hello world" and a sequence number. The sequence numbers range from 39 to 73. The terminal prompt at the bottom is "lbros@lbros-VirtualBox:~/catkin\_ws\$".

```
lbros@lbros-VirtualBox: ~/catkin_ws
File Edit Tabs Help
roscore http... x lbros@lbros... x
[ INFO] [1515649857.738393497]: hello world 39
[ INFO] [1515649857.838274996]: hello world 40
[ INFO] [1515649857.939045646]: hello world 41
[ INFO] [1515649858.038330999]: hello world 42
[ INFO] [1515649858.138507160]: hello world 43
[ INFO] [1515649858.239066178]: hello world 44
[ INFO] [1515649858.338445546]: hello world 45
[ INFO] [1515649858.438596424]: hello world 46
[ INFO] [1515649858.538680353]: hello world 47
[ INFO] [1515649858.641001702]: hello world 48
[ INFO] [1515649858.751274859]: hello world 49
[ INFO] [1515649858.838762107]: hello world 50
[ INFO] [1515649858.938506838]: hello world 51
[ INFO] [1515649859.039191743]: hello world 52
[ INFO] [1515649859.138581994]: hello world 53
[ INFO] [1515649859.238869128]: hello world 54
[ INFO] [1515649859.338677824]: hello world 55
[ INFO] [1515649859.438621819]: hello world 56
[ INFO] [1515649859.549089432]: hello world 57
[ INFO] [1515649859.648815116]: hello world 58
[ INFO] [1515649859.738269837]: hello world 59
[ INFO] [1515649859.841223203]: hello world 60
[ INFO] [1515649859.938687264]: hello world 61
[ INFO] [1515649860.038674566]: hello world 62
[ INFO] [1515649860.138613654]: hello world 63
[ INFO] [1515649860.238843187]: hello world 64
[ INFO] [1515649860.338535275]: hello world 65
[ INFO] [1515649860.438592784]: hello world 66
[ INFO] [1515649860.563749532]: hello world 67
[ INFO] [1515649860.639236155]: hello world 68
[ INFO] [1515649860.739459977]: hello world 69
[ INFO] [1515649860.838765688]: hello world 70
[ INFO] [1515649860.938947115]: hello world 71
[ INFO] [1515649861.038508801]: hello world 72
[ INFO] [1515649861.138786093]: hello world 73
lbros@lbros-VirtualBox:~/catkin_ws$
```



## Example

### Console Tab Nr. 3 – Analyze *talker* node

See the list of active nodes

```
> roscore list
```

Show information about the *talker* node

```
> roscore info /talker
```

See information about the *chatter* topic

```
> rostopic info /chatter
```

A screenshot of a terminal window titled "lbros@lbros-VirtualBox: ~/catkin\_ws". The terminal shows the execution of several ROS commands. First, "roscout http..." is run. Then, "roscout list" is executed, showing the output: "/rosout" and "/talker". Next, "roscout info talker" is run, displaying detailed information about the /talker node, including its publications, subscriptions, and services. Finally, "rostopic info chatter" is executed, showing information about the /chatter topic, including its type, publishers, and subscribers. The terminal output is as follows:

```
lbros@lbros-VirtualBox:~/catkin_ws$ roscout list
/rosout
/talker
lbros@lbros-VirtualBox:~/catkin_ws$ roscout info talker
-----
Node [/talker]
Publications:
* /chatter [std_msgs/String]
* /rosout [roscout_msgs/Log]

Subscriptions: None

Services:
* /talker/get_loggers
* /talker/set_logger_level

contacting node http://lbros-VirtualBox:43839/ ...
Pid: 13479
Connections:
* topic: /rosout
* to: /rosout
* direction: outbound
* transport: TCPROS

lbros@lbros-VirtualBox:~/catkin_ws$ rostopic info chatter
Type: std_msgs/String

Publishers:
* /talker (http://lbros-VirtualBox:43839/)

Subscribers: None

lbros@lbros-VirtualBox:~/catkin_ws$
```



## Example

### Console Tab Nr. 3 – Analyze *chatter* topic

Check the type of the *chatter* topic

```
> rostopic type /chatter
```

Show the message contents of the topic

```
> rostopic echo /chatter
```

Analyze the frequency

```
> rostopic hz /chatter
```

A terminal window titled "lbros@lbros-VirtualBox: ~/catkin\_ws" with standard window controls. It shows three tabs: "roscore http...", "lbros@lbros...", and "lbros@lbros...". The terminal output displays the results of three ROS commands:   
1. `rostopic type chatter` returns `std_msgs/String`.   
2. `rostopic echo chatter` shows a sequence of messages: `data: "hello world 3800"`, `data: "hello world 3801"`, `data: "hello world 3802"`, `data: "hello world 3803"`, and `data: "hello world 3804"`.   
3. `rostopic hz chatter` shows frequency statistics: `subscribed to [/chatter]`, `average rate: 10.000`, and then a series of lines for different window sizes (10, 20, 30, 40, 44) showing min, max, std dev, and average rate values.



## Example

### Console Tab Nr. 4 – Starting a *listener* node

Run a listener demo node with

```
> rosruncatkin_ws$ roscpp_tutorials listener
```

A terminal window titled "lbros@lbros-VirtualBox: ~/catkin\_ws" with a menu bar (File, Edit, Tabs, Help) and four tabs. The active tab shows the command "roscpp\_tutorials listener" being executed. The output consists of 20 lines of log messages, each starting with "[ INFO]" followed by a timestamp and the message "I heard: [hello world 6141]" through "6169".

```
lbros@lbros-VirtualBox: ~/catkin_ws$ roscpp_tutorials listener
[ INFO] [1515650629.883330468]: I heard: [hello world 6141]
[ INFO] [1515650629.985414755]: I heard: [hello world 6142]
[ INFO] [1515650630.082729639]: I heard: [hello world 6143]
[ INFO] [1515650630.183404147]: I heard: [hello world 6144]
[ INFO] [1515650630.285041209]: I heard: [hello world 6145]
[ INFO] [1515650630.386019910]: I heard: [hello world 6146]
[ INFO] [1515650630.482406502]: I heard: [hello world 6147]
[ INFO] [1515650630.585254983]: I heard: [hello world 6148]
[ INFO] [1515650630.685349428]: I heard: [hello world 6149]
[ INFO] [1515650630.786426201]: I heard: [hello world 6150]
[ INFO] [1515650630.883197809]: I heard: [hello world 6151]
[ INFO] [1515650630.985930668]: I heard: [hello world 6152]
[ INFO] [1515650631.085544590]: I heard: [hello world 6153]
[ INFO] [1515650631.186452282]: I heard: [hello world 6154]
[ INFO] [1515650631.285396694]: I heard: [hello world 6155]
[ INFO] [1515650631.385500894]: I heard: [hello world 6156]
[ INFO] [1515650631.485843060]: I heard: [hello world 6157]
[ INFO] [1515650631.582545588]: I heard: [hello world 6158]
[ INFO] [1515650631.682961369]: I heard: [hello world 6159]
[ INFO] [1515650631.782928551]: I heard: [hello world 6160]
[ INFO] [1515650631.882583373]: I heard: [hello world 6161]
[ INFO] [1515650631.990330651]: I heard: [hello world 6162]
[ INFO] [1515650632.082747375]: I heard: [hello world 6163]
[ INFO] [1515650632.182591196]: I heard: [hello world 6164]
[ INFO] [1515650632.282592494]: I heard: [hello world 6165]
[ INFO] [1515650632.383687290]: I heard: [hello world 6166]
[ INFO] [1515650632.483365123]: I heard: [hello world 6167]
[ INFO] [1515650632.582879182]: I heard: [hello world 6168]
[ INFO] [1515650632.682921573]: I heard: [hello world 6169]
```





## Example

### Console Tab Nr. 3 – Analyze

See the new *listener* node with

```
> rosnodet list
```

Show the connection of the nodes over the chatter topic with

```
> rostopic info /chatter
```

A terminal window titled "lbros@lbros-VirtualBox: ~/catkin\_ws" with a menu bar (File, Edit, Tabs, Help) and four tabs. The first tab is active and shows the output of the command `rostopic info chatter`. The output lists the topic name, type, publishers, and subscribers.

```
lbros@lbros-VirtualBox:~/catkin_ws$ rostopic info chatter
Type: std_msgs/String

Publishers:
 * /talker (http://lbros-VirtualBox:43839/)

Subscribers:
 * /listener (http://lbros-VirtualBox:35389/)

lbros@lbros-VirtualBox:~/catkin_ws$
```



## Example

### Console Tab Nr. 3 – Publish Message from Console

Close the *talker* node in console nr. 2 with Ctrl + C

Publish your own message with

```
[ INFO] [1515650917.681739261]: hello world 9019
[ INFO] [1515650917.783052753]: hello world 9020
^C[ INFO] [1515650917.882553801]: hello world 9021
lbros@lbros-VirtualBox:~/catkin_ws$ rostopic pub /chatter std_msgs/String "data:
COMP20170 Introduction to Robotics Module"
publishing and latching message. Press ctrl-C to terminate
```

Check the output of the *listener* in console nr. 4

```
[ INFO] [1515650917.582120782]: I heard: [hello world 9018]
[ INFO] [1515650917.682179493]: I heard: [hello world 9019]
[ INFO] [1515650917.783547353]: I heard: [hello world 9020]
[ INFO] [1515651125.734036397]: I heard: [COMP20170 Introduction to Robotics Mod
ule]
```

## catkin Build System

- *catkin* is the ROS build system to generate executables, libraries, and interfaces
- We suggest to use the *Catkin Command Line Tools*

→ Use **catkin build** instead of `catkin_make`

Navigate to your catkin workspace with

```
> cd ~/catkin_ws
```

Build a package with

```
> catkin build package_name
```

! Whenever you build a **new** package, update your environment

```
> source devel/setup.bash
```

The catkin  
command line  
tools are pre-  
installed in the  
provided  
installation.

**More info**

<http://wiki.ros.org/catkin/Tutorials>  
<https://catkin-tools.readthedocs.io/>

## catkin Build System

The catkin workspace contains the following spaces

Work here



src

The *source space* contains the source code. This is where you can clone, create, and edit source code for the packages you want to build.

Don't touch



build

The *build space* is where CMake is invoked to build the packages in the source space. Cache information and other intermediate files are kept here.

Don't touch



devel

The *development (devel) space* is where built targets are placed (prior to being installed).

If necessary, clean the entire build and devel space with

```
> catkin clean
```

**More info**

<http://wiki.ros.org/catkin/workspaces>



## catkin Build System

The catkin workspace setup can be checked with

```
> catkin config
```

For example, to set the *CMake build type* to Release (or Debug etc.), use

```
> catkin build --cmake-args  
    -DCMAKE_BUILD_TYPE=Release
```

### More info

[http://catkin-tools.readthedocs.io/en/latest/verbs/catkin\\_config.html](http://catkin-tools.readthedocs.io/en/latest/verbs/catkin_config.html)

[http://catkin-tools.readthedocs.io/en/latest/cheat\\_sheet.html](http://catkin-tools.readthedocs.io/en/latest/cheat_sheet.html)

```
student@ubuntu:~/catkin_ws$ catkin config
-----
Profile:                        default
Extending:                      [env] /opt/ros/indigo:/home/student/catkin_ws/devel
Workspace:                      /home/student/catkin_ws
-----
Source Space:                   [exists] /home/student/catkin_ws/src
Log Space:                      [exists] /home/student/catkin_ws/logs
Build Space:                    [exists] /home/student/catkin_ws/build
Devel Space:                    [exists] /home/student/catkin_ws/devel
Install Space:                  [unused] /home/student/catkin_ws/install
DESTDIR:                        [unused] None
-----
Devel Space Layout:             linked
Install Space Layout:          None
-----
Additional CMake Args:          -GEclipse CDT4 - Unix Makefiles -DCMAKE_CXX_COMP
ILER_ARG1=-std=c++11 -DCMAKE_BUILD_TYPE=Release
Additional Make Args:           None
Additional catkin Make Args:    None
Internal Make Job Server:      True
Cache Job Environments:        False
-----
Whitelisted Packages:          None
Blacklisted Packages:          None
-----
Workspace configuration appears valid.
```

Already  
setup in the  
provided  
installation.



## Example

Open a terminal and browse to your git folder

```
> cd ~/git
```

Clone the Git repository with

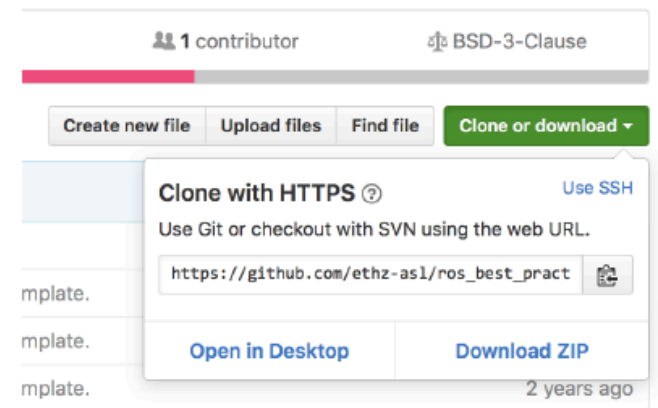
```
> git clone https://github.com/ethz-asl/ros_best_practices.git
```

Symlink the new package to your catkin workspace

```
> ln -s ~/git/ros_best_practices/ ~/catkin_ws/src/
```

*Note: You could also directly clone to your catkin workspace, but using a common git folder is convenient if you have multiple catkin workspaces.*

[https://github.com/ethz-asl/ros\\_best\\_practices](https://github.com/ethz-asl/ros_best_practices)





## Example

Go to your catkin workspace

```
> cd ~/catkin_ws
```

Build the package with

```
lbros@lbros-VirtualBox:~/catkin_ws$ catkin_make ros_package_template
```

Re-source your workspace setup

```
> source devel/setup.bash
```

Launch the node with

```
> roslaunch ros_package_template  
  ros_package_template.launch
```

```
/home/lbros/catkin_ws/src/git/ro...te.launch http://localhost:11311 - + x
File Edit Tabs Help
ch-lbros-VirtualBox-14134.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://lbros-VirtualBox:36471/

SUMMARY
=====
PARAMETERS
* /ros_package_template/subscriber_topic: /temperature
* /rostdistro: kinetic
* /rosversion: 1.12.12

NODES
/
  ros_package_template (ros_package_template/ros_package_template)

ROS_MASTER_URI=http://localhost:11311

process[ros_package_template-1]: started with pid [14151]
[ INFO] [1515653137.428379549]: Successfully launched node.
```



## ROS Launch

- *launch* is a tool for launching multiple nodes (as well as setting parameters)
- Are written in XML as \*.*launch* files
- If not yet running, launch automatically starts a roscore

Browse to the folder and start a launch file with

```
> roslaunch file_name.launch
```

Start a launch file from a package with

```
> roslaunch package_name file_name.launch
```

### More info

<http://wiki.ros.org/roslaunch>

Example console output for  
roslaunch roscpp\_tutorials talker\_listener.launch

A screenshot of a terminal window titled "lbros@lbros-VirtualBox: ~/catkin\_ws". The window shows the output of the command "roslaunch roscpp\_tutorials talker\_listener.launch". The output includes the ROS\_MASTER\_URI, the start of two processes (listener-1 and talker-2), and a series of "hello world" messages from the talker node, which are received by the listener node. The messages are timestamped and include the process IDs.

```
lbros@lbros-VirtualBox: ~/catkin_ws
File Edit Tabs Help
/home/lbros... x lbros@lbros... x
talker (roscpp_tutorials/talker)
ROS_MASTER_URI=http://localhost:11311
process[listener-1]: started with pid [14365]
process[talker-2]: started with pid [14366]
[ INFO] [1515653566.501365349]: hello world 0
[ INFO] [1515653566.609048861]: hello world 1
[ INFO] [1515653566.705561259]: hello world 2
[ INFO] [1515653566.737153923]: I heard: [COMP20170 Introduction to Robotics Mod
ule]
[ INFO] [1515653566.806794181]: hello world 3
[ INFO] [1515653566.808014464]: I heard: [hello world 3]
[ INFO] [1515653566.905684740]: hello world 4
[ INFO] [1515653566.906652486]: I heard: [hello world 4]
[ INFO] [1515653567.002569505]: hello world 5
[ INFO] [1515653567.003333932]: I heard: [hello world 5]
```





## ROS Launch File Structure

*talker\_listener.launch*

```
<launch>  
  <node name="listener" pkg="roscpp_tutorials" type="listener" output="screen"/>  
  <node name="talker" pkg="roscpp_tutorials" type="talker" output="screen"/>  
</launch>
```

! Notice the syntax difference  
for self-closing tags:  
▪ `<tag></tag>` and `<tag/>`

- **launch**: Root element of the launch file
- **node**: Each `<node>` tag specifies a node to be launched
- **name**: Name of the node (free to choose)
- **pkg**: Package containing the node
- **type**: Type of the node, there must be a corresponding executable with the same name
- **output**: Specifies where to output log messages (`screen`: console, `log`: log file)

### More info

<http://wiki.ros.org/roslaunch/XML>

<http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20tips%20for%20larger%20projects>



## 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>
```

**More info**

<http://wiki.ros.org/roslaunch/XML/arg>



## 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>
```

**More info**

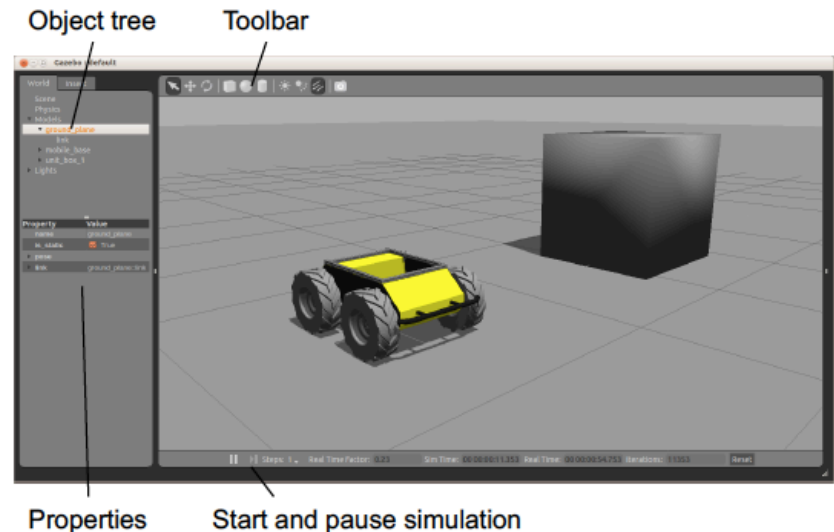
<http://wiki.ros.org/roslaunch/XML/include>

## Gazebo Simulator

- Simulate 3d rigid-body dynamics
- Simulate a variety of sensors including noise
- 3d visualization and user interaction
- Includes a database of many robots and environments (*Gazebo worlds*)
- Provides a ROS interface
- Extensible with plugins

Run Gazebo with

```
> rosrun gazebo_ros gazebo
```



**More info**

<http://gazebosim.org/>  
<http://gazebosim.org/tutorials>



## 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://github.com/ros/cheatsheet/releases/download/0.0.1/ROSCheatsheet\\_catkin.pdf](https://github.com/ros/cheatsheet/releases/download/0.0.1/ROSCheatsheet_catkin.pdf)
- **ROS Best Practices**
  - [https://github.com/ethz-asl/ros\\_best\\_practices/wiki](https://github.com/ethz-asl/ros_best_practices/wiki)
- **ROS Package Template**
  - [https://github.com/ethz-asl/ros\\_best\\_practices/tree/master/ros\\_package\\_template](https://github.com/ethz-asl/ros_best_practices/tree/master/ros_package_template)