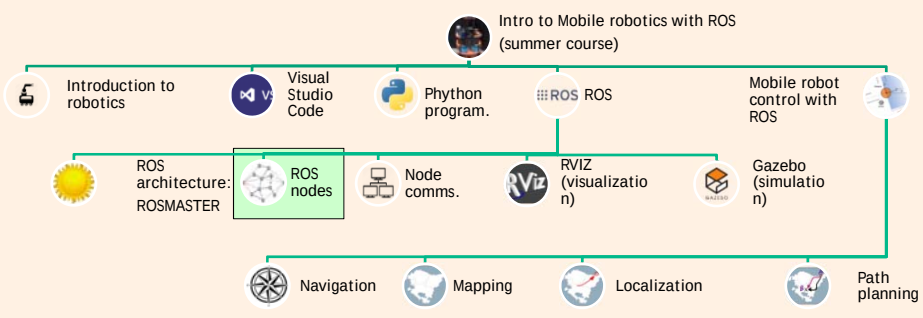
ROS packages

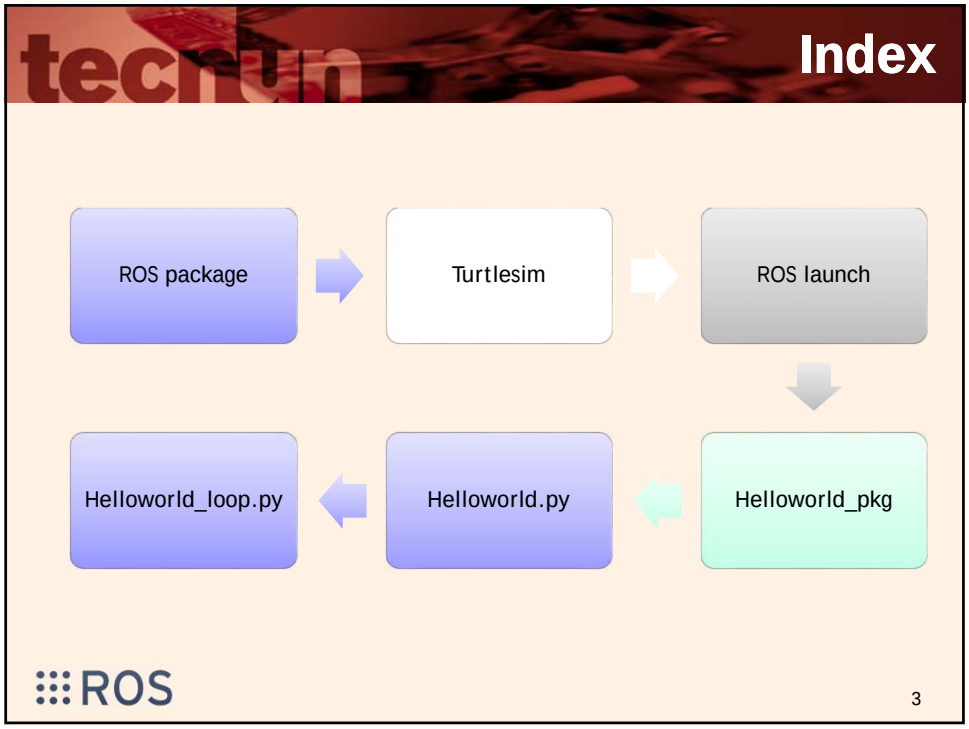
Dr. Emilio Sánchez Tapia

tecrun

Mobile robots with ROS



ROS


Index
ROS package
Turtlesim
ROS launch
Helloworld_pkg
Helloworld.py
Helloworld_loop.py
ROS
3


ROS package
ROS

ROS package

- ROS uses **packages** to organize its programs
- **All the files that a specific ROS program contains**; all its cpp files, python files, configuration files, compilation files, launch files, and parameters files.
- With the following structure:
  - **launch** folder: Contains launch files
  - **src** folder: Source files (cpp, python)
  - **CMakeLists.txt**: List of cmake rules for compilation
  - **package.xml**: Package information and dependencies

ROS

5

Turtesim

ROS

ROS package:turtlesim example

- Terminal 1

```
$ roscore
```

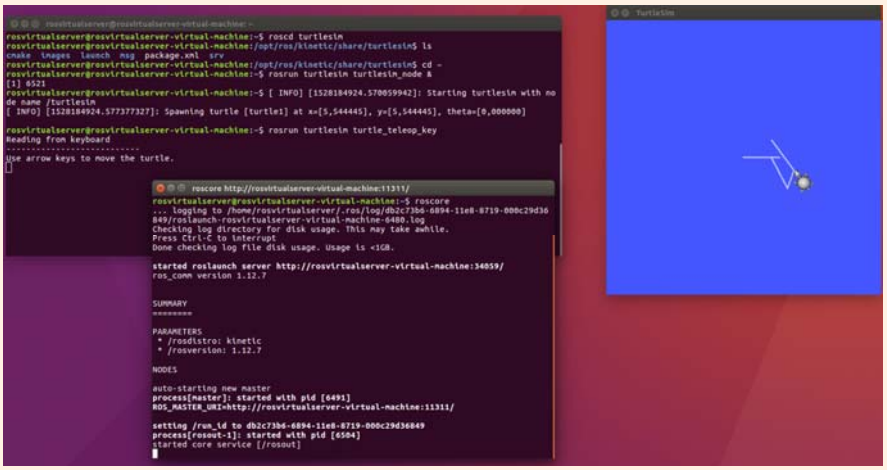
- Terminal 2

```
$ roscd turtlesim
$ ls
$ cd ~
$ rosrun turtlesim turtlesim_node &
$ rosrun turtlesim turtle_teleop_key
```

ROS

7

ROS package:turtesim example



ROS

8

ROS launch

ROS

tecrun

ROSlaunch

- It is a tool for launching multiple nodes (as well as setting parameters)
- Base on *.launch files, written in XML
- If roscore not yet running, launch automatically starts it
- Usage: Browse to the folder and start a launch file with

```
roslaunch package_name file_name.launch
```

ROS

10

ROSlaunch: turtlesim example

- turtlesim.launch

```
<launch>
  <!-- the turtle -->
  <node pkg="turtlesim" type="turtlesim_node"
name="turtesim_node"  output="screen">
  </node>

  <!-- turtle controls -->
  <node pkg="turtlesim" type="turtle_teleop_key"
name="turtesim_teleop_key"  output="screen">
  </node>

</launch>
```

- Usage: Browse to the folder and start a launch file with

```
roslaunch turtlesim turtlesim.launch
```

ROS

11

ROSlaunch: xml structure

- `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)

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

ROS

12

ROSlaunch: arguments

- Launch files with <arg> tag→ which works like a parameter (default optional)

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

- Use the arguments in the launch file as:

```
$(arg arg_name)
```

- When launching, from the terminal:

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

- Include other launch files:

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

- Find the system path to other packages

```
$(find package_name)
```

- Pass arguments to the included file

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

ROS

13

ROSlaunch: example

```
<?xml version="1.0"?>
<launch>
  <arg name="sim_time_on" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>
  <group if="$(arg sim_time_on)">
    <param name="/sim_time_on" value="true" />
  </group>
  <include file="$(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

14

ROS package: creating

- To create a new package in the current directory:

```
> catkin_create_pkg [package_name] [depend1] [depend2] [depend3]
> roscreate-pkg [package_name] [depend1] [depend2] [depend3]
```

- The preferred one catkin method

```
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src$ catkin_create
_pkg  ros_tutorial1 std_msgs rospy
Created file ros_tutorial1/CMakeLists.txt
Created file ros_tutorial1/package.xml
Created folder ros_tutorial1/src
Successfully created files in /home/rosvirtualserver/catkin_ws/src/ros_tutorial1
. Please adjust the values in package.xml.
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src$
```

ROS

15

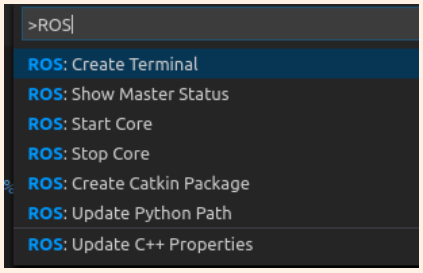
Helloworld_pkg

ROS

ROS package: creating with vscode

- We can also create catkin packages from visual code studio thanks to the ROS plugin.
- Other ROS commands available in Visual Studio Code:

`<crtl+p>→type ROS →`



ROS

17

ROS package: structure

- The typical dependencies:
  - `std_msgs` → standard structures to be used for topics
  - `rospy` → if we want to program it in Python
  - `roscpp` → if we want to program it in C++
- Two files are created

  > `CMakeLists.txt` → the input to the CMake build system for building software packages.

  > `package.xml` →meta information about the package.

ROS

18

ROS package: createing helloworld_pkg

```
~$ cd ~/catkin_ws/src
~/catkin_ws/src$ catkin_create_pkg helloworld_pkg

Created file helloworld_pkg/CMakeLists.txt
Created file helloworld_pkg/package.xml
Successfully created files in
/home/rosvirtualserver/catkin_ws/src/helloworld_pkg. Please
adjust the values in package.xml.

~/catkin_ws/src$ ls |grep hello
helloworld_pkg

~/catkin_ws/src$ cd helloworld_pkg/
~/catkin_ws/src/helloworld_pkg$ ls
CMakeLists.txt  package.xml
```

ROS 19

ROS package: package.xml

```
<?xml version="1.0"?>
<package format="2">
<name>hellowolrd</name>
<version>0.0.0</version>
<description>This is my first ROS
package.</description>
<maintainer email="noname@noplace.com">my
name</maintainer>
<license>BSD</license>
<url type="website">http://www.tecnun.es</url>
<author email=" noname@noplace.com"> my name
</author>
<buildtool_depend>catkin</buildtool_depend>
<depend>roscpp</depend>
<depend>sensor_msgs</depend>
</package>
```

ROS 20

ROS package: package.xml

- For C++

```
cmake_minimum_required(VERSION 2.8.3)
project(helloworld_pkg)
## Use C++11
add_definitions(--std=c++11)
## Find catkin macros and libraries
find_package(catkin REQUIRED
COMPONENTS
roscpp
std_msgss
)
```

- For Python

```
cmake_minimum_required(VERSION 2.8.3)
project(helloworld_pkg)

## Find catkin macros and libraries
find_package(catkin REQUIRED
COMPONENTS
rospy
std_msgss
)
```

ROS 21

ROS package: package.xml→cpp

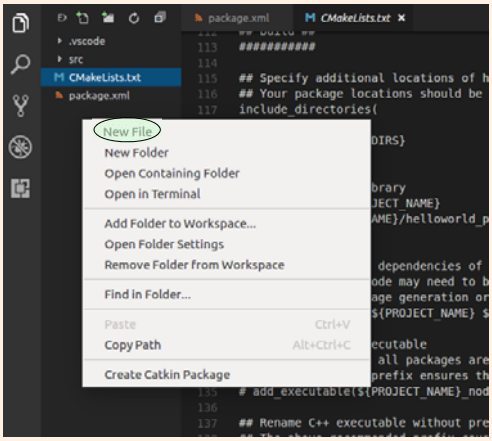
```
catkin_package(
INCLUDE_DIRS include
# LIBRARIES
CATKIN_DEPENDS roscpp std_msgs
# DEPENDS
)
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(${PROJECT_NAME} src/${PROJECT_NAME}_node.cpp
src/helloworld_pkg_node.cpp)

target_link_libraries(${PROJECT_NAME} ${catkin_LIBRARIES})
```

ROS 22

ROS package: helloworld.py file

- Create helloworld.py in src directory
- Add the following lines

```
#!/usr/bin/env python
import rospy
rospy.init_node('hello')
print "hello world. This
is my first ROS node"
```



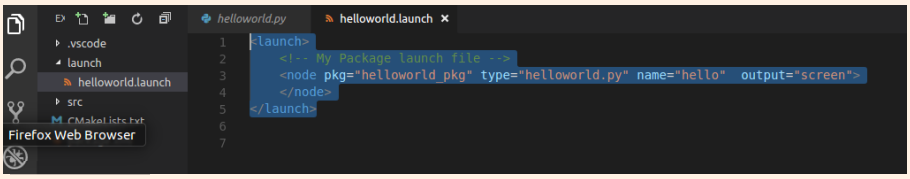
ROS

24

ROS package: helloworld.launch file

- Create helloworld.launch in launch directory
- Add the following lines

```
<launch>
    <!-- My Package launch file -->
    <node pkg="helloworld_pkg" type="helloworld.py"
name="hello"  output="screen">
    </node>
</launch>
```



ROS

25

ROS package: launch helloworld.py

```
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$ roslaunch
 helloworld_pkg helloworld.launch
... logging to /home/rosvirtualserver/.ros/log/04f3eea2-68ba-11e8-8719-000c29d36849/roslaunch-r
osvirtualserver-virtual-machine-28481.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://rosvirtualserver-virtual-machine:33307/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES
  /
    hello (helloworld_pkg/helloworld.py)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
ERROR: cannot launch node of type [helloworld_pkg/helloworld.py]: can't locate node [helloworld
.py] in package [helloworld_pkg]
No processes to monitor
shutting down processing monitor...
... shutting down processing monitor complete
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$ ls -la
total 12
drwxrwxr-x 2 rosvirtualserver rosvirtualserver 4096 jun  5 14:18 .
drwxrwxr-x 5 rosvirtualserver rosvirtualserver 4096 jun  5 14:28 ..
-rw-rw-r-- 1 rosvirtualserver rosvirtualserver  343 jun  5 14:18 helloworld.py
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$ chmod +x
helloworld.py
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$ ls -la
total 12
drwxrwxr-x 2 rosvirtualserver rosvirtualserver 4096 jun  5 14:18 .
drwxrwxr-x 5 rosvirtualserver rosvirtualserver 4096 jun  5 14:28 ..
-rwxrwxr-x 1 rosvirtualserver rosvirtualserver  343 jun  5 14:18 helloworld.py
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$
```

ROS

ROS package: launch helloworld.py

```
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$ roslaunch
 helloworld_pkg helloworld.launch
... logging to /home/rosvirtualserver/.ros/log/04f3eea2-68ba-11e8-8719-000c29d36849/roslaunch-r
osvirtualserver-virtual-machine-30477.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://rosvirtualserver-virtual-machine:45761/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES
  /
    hello (helloworld_pkg/helloworld.py)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[hello-1]: started with pid [30498]
hello world. This is my first ROS node
[hello-1] process has finished cleanly
log file: /home/rosvirtualserver/.ros/log/04f3eea2-68ba-11e8-8719-000c29d36849/hello-1*.log
all processes on machine have died, roslaunch will exit
shutting down processing monitor...
... shutting down processing monitor complete
done
rosvirtualserver@rosvirtualserver-virtual-machine:~/catkin_ws/src/helloworld_pkg/src$
```

ROS

27

Hellowold_loop.py

ROS

tecrun

ROS package: helloworld_loop

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

import rospy # Import the rospy, the Python library for ROS.
rospy.init_node('hello_loop')

print "hello world. This is my first ROS node" #a message sent to the
consolels

rate = rospy.Rate(0.2)                # We create a Rate object of 0.2Hz
while not rospy.is_shutdown():      # Endless loop until Ctrl + C
   print "hello loop"
   rate.sleep()                     # We sleep the needed time to
maintain the Rate fixed above
```

ROS

29

ROS package: exercise helloworld_loop

- Create helloworld_loop.py node
- Create helloworld_loop.launch
- Execute helloworld_loop node
  - roslaunch
  - rosrun
- Test the node is on
  - rosnode list
  - rosnode info :

ROS

30

ROS package: build helloworld_pkg

- When the package is finished
  - > cd ~/catkin_ws
  - > catkin_make
- This instruction → compiles the src directory and creates the build directory
- If only python programming → required

ROS

31