

ECE 478-578

Intelligent Robotics I

PhD. Husnu Melih Erdogan – Electrical & Computer Engineering

herdogan@pdx.edu Teaching Assistant

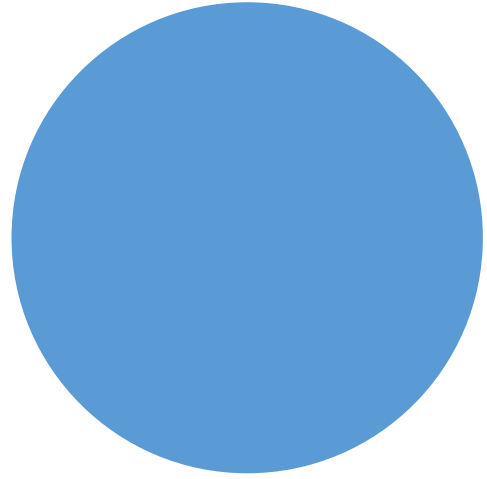


Introduction to ROS Part – 7



Course Structure

- **Part 1 - Overview**
 - What is ROS?
 - Introduction to ROS
 - ROS architecture, philosophy, history
 - How to install ROS?
 - Examples
 - Installation
 - ROS Master
 - ROS Nodes
 - ROS Topic
 - ROS Messages
 - Console Commands
 - ROS Packages
 - ROS Launch-files
 - Catkin Workspace and Build System
 - Turtlesim
- **Part 2 - Basics**
 - ROS File System
 - ROS Package
 - How to create a package?
 - How to build a package?
 - Creating a Publisher Node
 - Creating a Subscriber Node
 - Assignment 3
- **Part 3 - Debug**
 - ROS Launch File
 - How to use ROS .bagfiles?
 - ROS Parameters
 - ROS Namespace
- **Part 4 - Speech**
 - ROS Services
 - Speech Recognition
 - Speech Synthesis
 - Google Dialogflow
- **Part 5 - Speech**
 - Amazon Polly
 - ROS Actions
 - Assignment 4 (Optional)
- **Part 6 - Fuzzy**
 - 2D Multi-Robot Simulator
 - Fuzzy Logic
 - Assignment 5
- **Part 7 - Network**
 - ROS Messages
 - ROS Networking
 - ROS and RaspberryPi



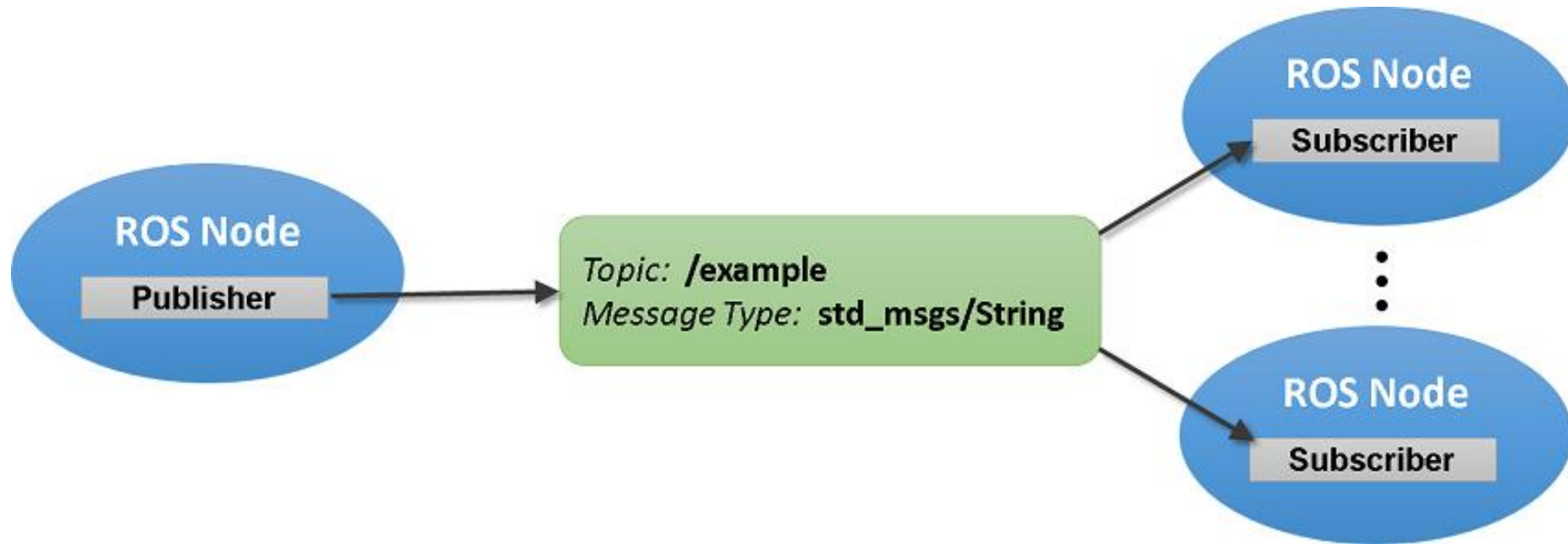
ROS Messages



ROS Messages

- ROS message describes the data values that ROS nodes publish.
- Messages help ROS tools to automatically generate code for the message type in several target languages.
- Message descriptions are in .msg files in the msg/ directory.
- Messages can be nested in each other.
- Each ROS distribution can have a different description for a message
 - You can go to <http://wiki.ros.org/msg> and check the message type
- Publish a message to a subscriber ROS node on command line.
 - `rostopic pub /mytopic std_msgs/String "data: 'Portland State University'"`

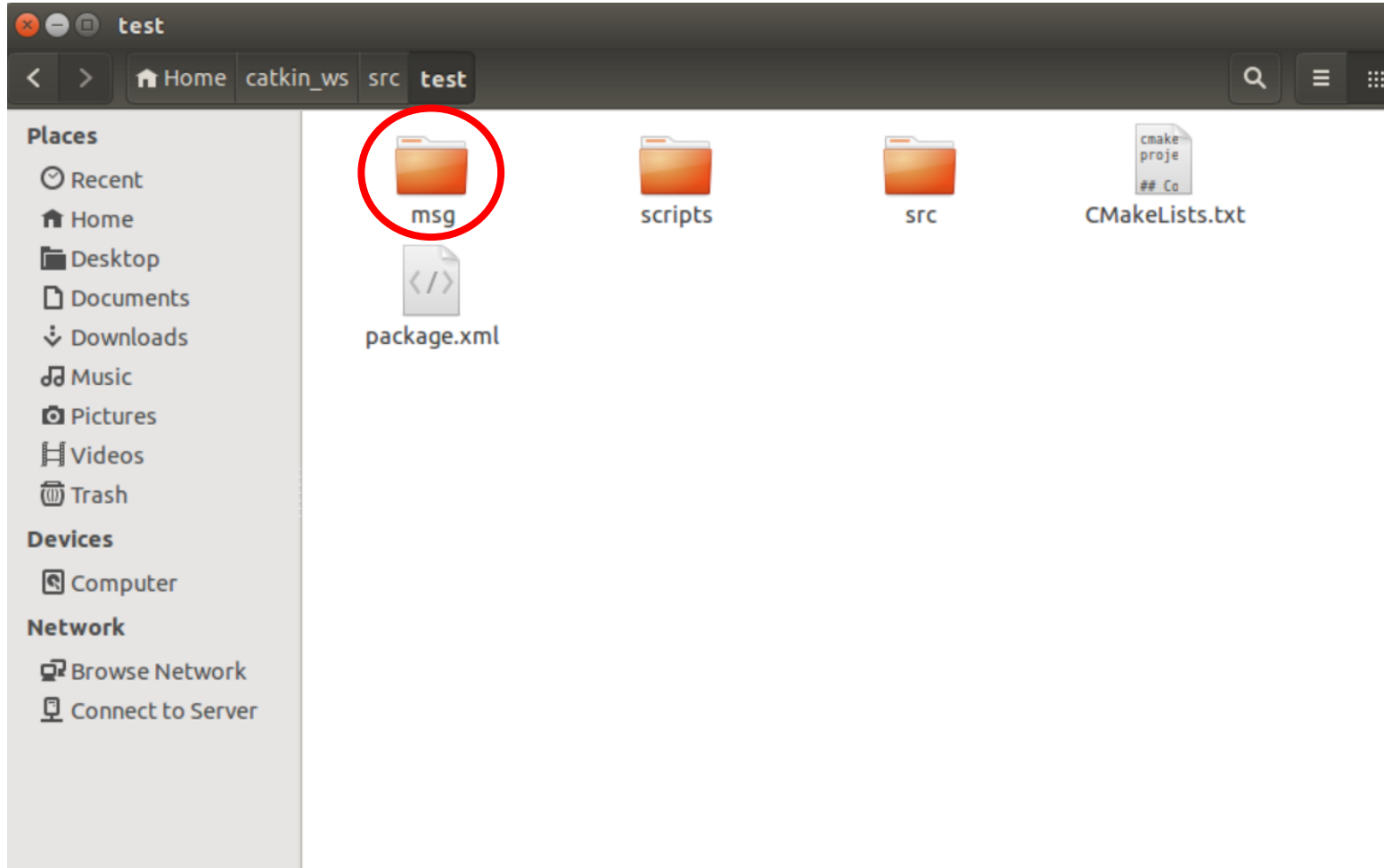
Summary of ROS Messages



[Image Source Link](#)

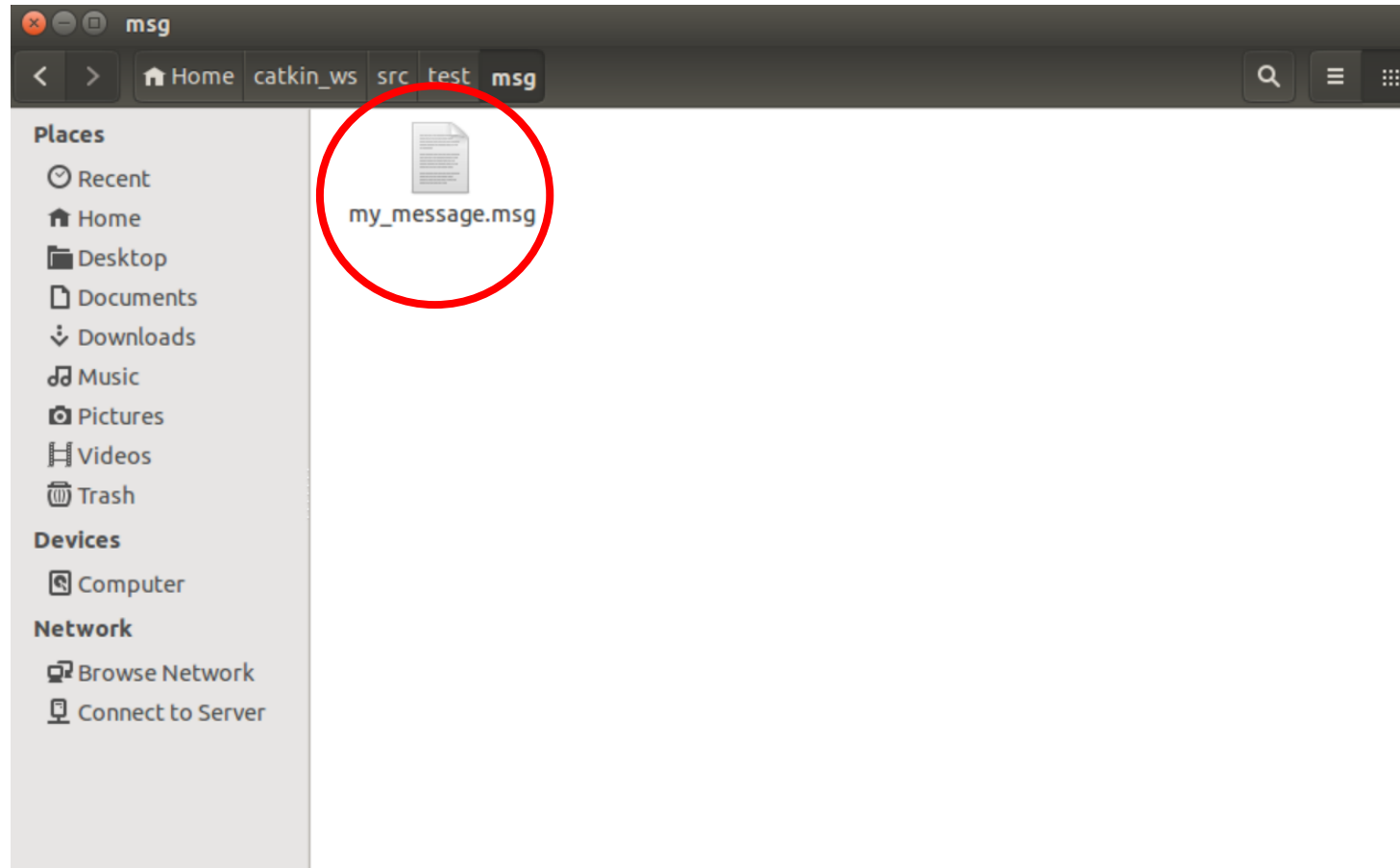
ROS Custom Messages

- Create a directory called msg



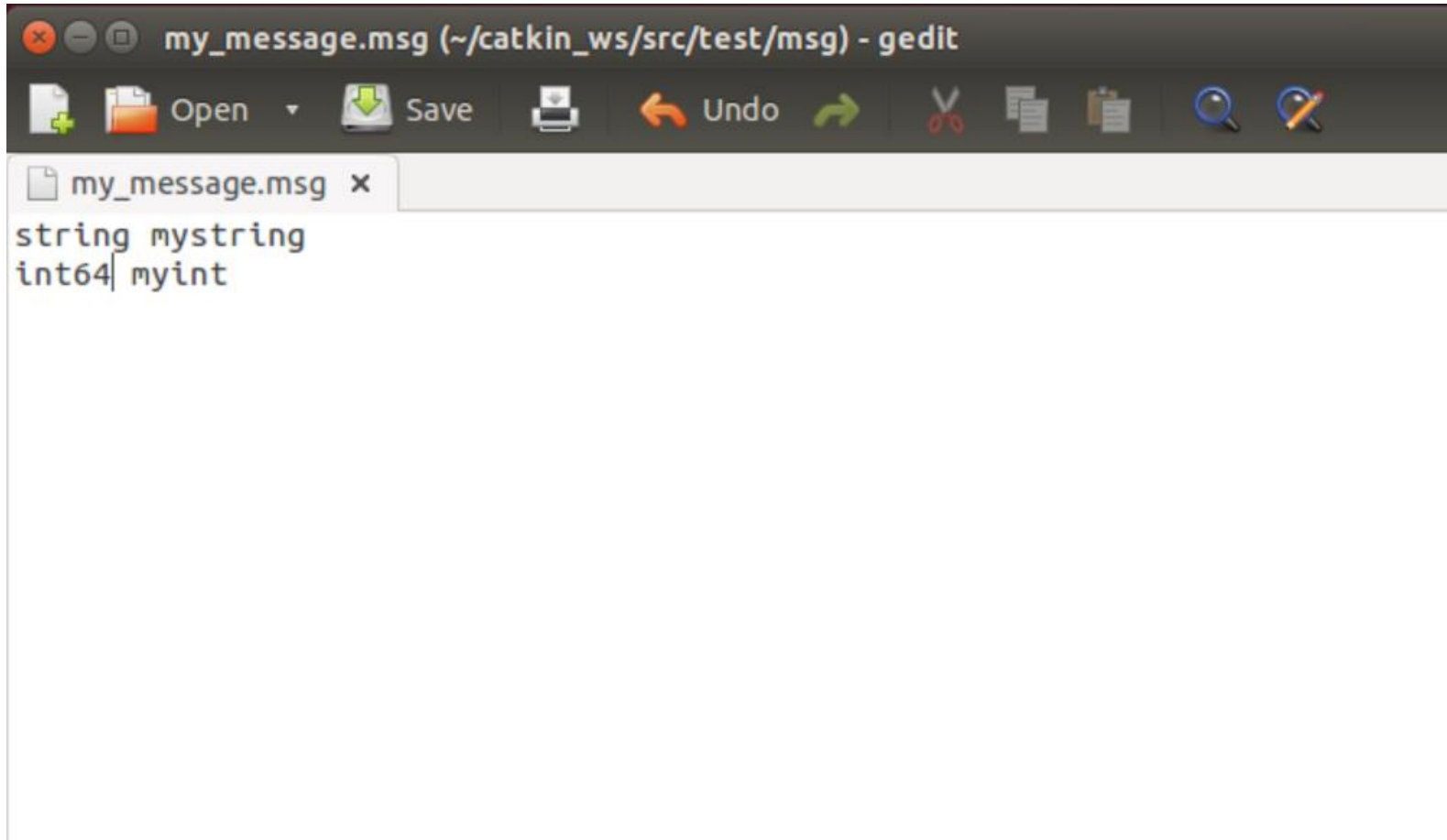
ROS Custom Messages

- Create .msg file for our custom message



ROS Custom Messages

- Define our custom message



The screenshot shows a gedit editor window titled "my_message.msg (~catkin_ws/src/test/msg) - gedit". The window has a menu bar with "Open", "Save", "Undo", and other standard editing functions. Below the menu bar, there is a tab labeled "my_message.msg x". The main text area contains the following code:

```
string mystring  
int64 myint
```

ROS Custom Messages

- Modify CMakeLists.txt in our package – add required package names

```
CMakeLists.txt x
cmake_minimum_required(VERSION 2.8.3)
project(test)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  message_generation
  std_msgs
)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)

## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user\_guide/setup\_dot\_py.html
# catkin_python_setup()
```

ROS Custom Messages

- Modify CMakeList.txt in our package – add our new message file and its dependencies

```
## Generate messages in the 'msg' folder
add_message_files(
  FILES
  my_message.msg
  #   Message2.msg
)
```

```
## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )
```

```
## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )
```

```
## Generate added messages and services with any dependencies listed here
generate_messages(
  DEPENDENCIES
  std_msgs  # Or other packages containing msgs
)
```

ROS Custom Messages

- Modify CMakeList.txt in our package – add catkin package dependency

```
#####  
## catkin specific configuration ##  
#####  
## The catkin_package macro generates cmake config files for your package  
## Declare things to be passed to dependent projects  
## INCLUDE_DIRS: uncomment this if your package contains header files  
## LIBRARIES: libraries you create in this project that dependent projects also need  
## CATKIN_DEPENDS: catkin_packages dependent projects also need  
## DEPENDS: system dependencies of this project that dependent projects also need  
catkin_package(  
#   INCLUDE_DIRS include  
#   LIBRARIES test  
    CATKIN_DEPENDS message_runtime  
#   DEPENDS system_lib  
)
```

ROS Custom Messages

- Modify package.xml in our package – add new dependencies

```
<buildtool_depend>catkin</buildtool_depend>
<build_depend>rospy</build_depend>
<build_export_depend>rospy</build_export_depend>
<exec_depend>rospy</exec_depend>
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>

<!-- The export tag contains other, unspecified, tags -->
<export>
  <!-- Other tools can request additional information be placed here -->

</export>
</package>
```

ROS Custom Messages

- Don't forget to source your setup.bash file

```
melih@melih-VirtualBox:~/catkin_ws$ source devel/setup.bash
```

- Let's check if our custom message is successfully created

```
melih@melih-VirtualBox:~/catkin_ws$ rosmmsg show test/my_message  
string mystring  
int64 myint
```

ROS Custom Messages

- Create a listener node in a different package that uses our custom message

```
listener.py x
#!/usr/bin/env python

import rospy
from std_msgs.msg import String
from test.msg import my_message

def callback(data):
    print data.mystring
    print data.myint

def listener():

    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber('chatter', my_message, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```

ROS Custom Messages

- Run roscore

```
roscore http://melih-VirtualBox:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://melih-VirtualBox:41922/
ros_comm version 1.11.21

SUMMARY
=====

PARAMETERS
* /roscdistro: indigo
* /rosversion: 1.11.21

NODES

auto-starting new master
process[master]: started with pid [3722]
ROS_MASTER_URI=http://melih-VirtualBox:11311/

setting /run_id to 4ff0d60e-f348-11e8-9143-080027535782
process[rosout-1]: started with pid [3735]
started core service [/rosout]
```

- Run our listener node

```
melih@melih-VirtualBox:~/catkin_ws$ rosrn my_first_rospackag listener.py
```

ROS Custom Messages

- List all the available topic names

```
melih@melih-VirtualBox: ~/catkin_ws
melih@melih-VirtualBox:~/catkin_ws$ rostopic list
/chatter
/rosout
/rosout_agg
```

- Get information about the chatter topic

```
melih@melih-VirtualBox: ~/catkin_ws
melih@melih-VirtualBox:~/catkin_ws$ rostopic info /chatter
Type: test/my_message

Publishers: None

Subscribers:
* /listener_3811_1543435279004 (http://melih-VirtualBox:45684/)
```

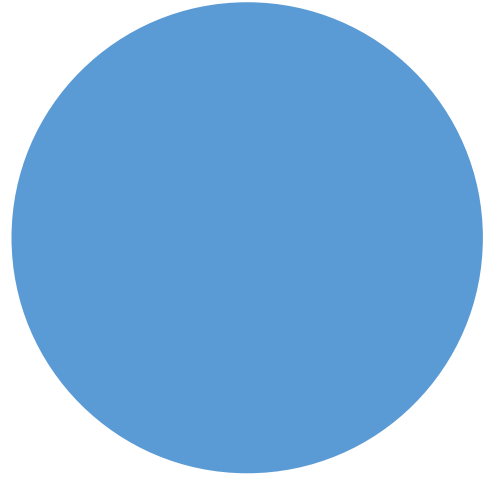
ROS Custom Messages

- Publish to the topic

```
melih@melih-VirtualBox: ~/catkin_ws
melih@melih-VirtualBox:~/catkin_ws$ rostopic pub -1 /chatter test/my_message melih 1903
publishing and latching message for 3.0 seconds
```

- Listener node will print that message

```
melih@melih-VirtualBox:~/catkin_ws$ rosrun my_first_rospackage listener.py
melih
1903
```



ROS Networking



ROS Networking

- kinetic-server



- roscore
- turtlesim

- raspberrypi: 192.168.0.100
- kinetic-server: 192.168.0.101



- raspbeerypi



- Rostopic pub -1

ROS Networking

- **On kinetic-server**
 - **sudo nano etc/hosts**
 - **Add these lines**
 - kinetic-server: 192.168.0.101
 - raspberrypi: 192.168.0.100
- **On raspberrypi**
 - **sudo nano etc/hosts**
 - **Add these lines**
 - raspberrypi: 192.168.0.100
 - kinetic-server: 192.168.0.101

ROS Networking

Add these two lines into your .bashrc file on kinetic-server

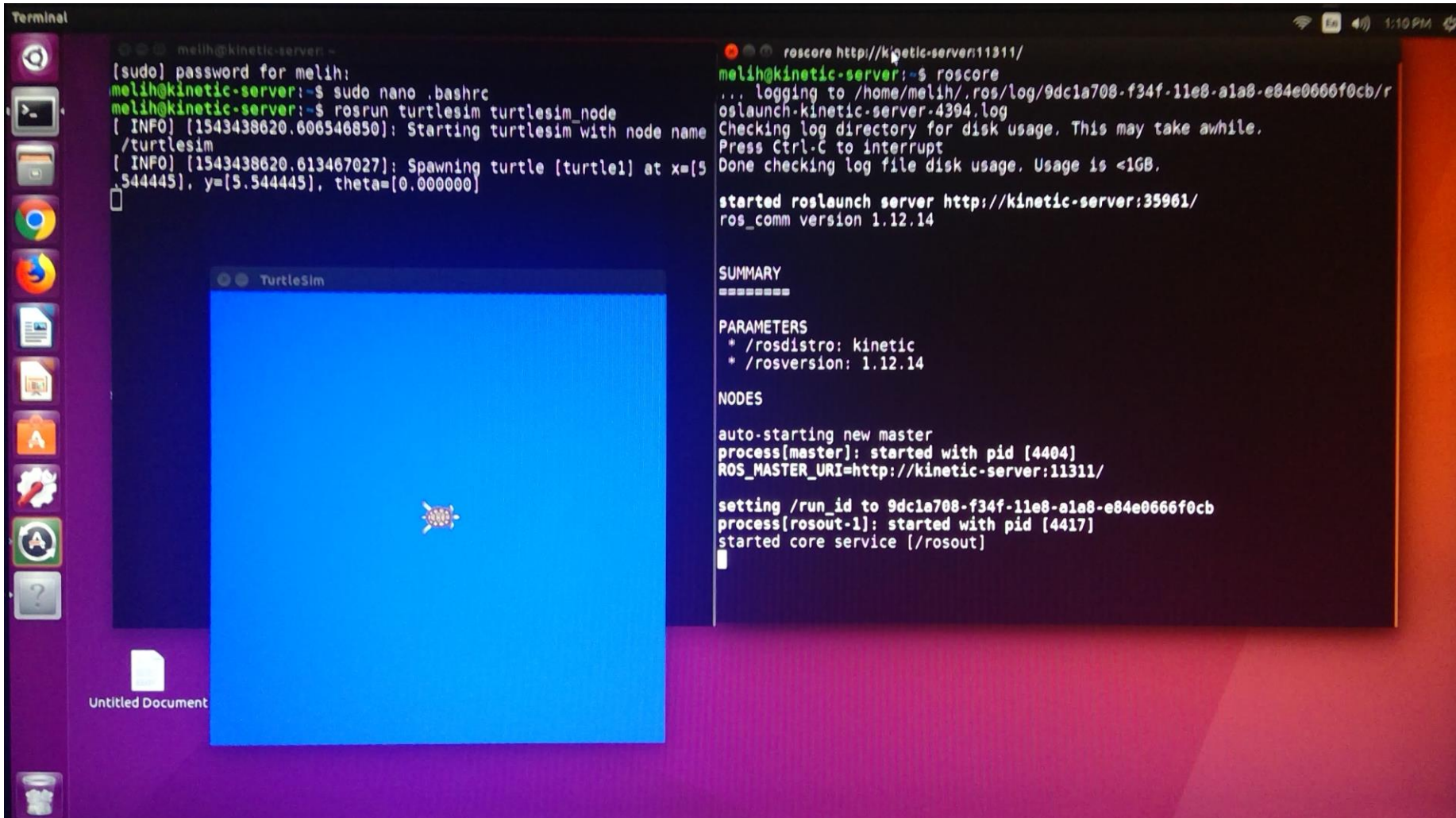
- export ROS_HOSTNAME=**kinetic-server**
- export ROS_MASTER_URI=http://kinetic-server:11311

Add these two lines in to the .bashrc file on raspberrypi

- export ROS_HOSTNAME=**raspberrypi**
- export ROS_MASTER_URI=http://kinetic-server:11311

ROS Networking

- On kinetic-server



The screenshot shows a Linux desktop environment with a terminal window and a TurtleSim window. The terminal window is split into two panes. The left pane shows the user 'melih' at the 'kinetic-server' machine, running 'sudo nano .bashrc' and then 'roslaunch turtlesim turtlesim_node'. The right pane shows the output of the 'roslaunch' command, including the ROS master URI 'http://kinetic-server:11311/' and the start of the 'turtlesim' node. The TurtleSim window is a blue square with a small turtle icon in the center. The desktop background is a purple and red gradient. The terminal window has a title bar that says 'Terminal'.

```
Terminal
melih@kinetic-server:~$ sudo nano .bashrc
melih@kinetic-server:~$ roslaunch turtlesim turtlesim_node
[ INFO] [1543438620.606546850]: Starting turtlesim with node name /turtlesim
[ INFO] [1543438620.613467027]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

roscore http://kinetic-server:11311/
melih@kinetic-server:~$ roscore
... logging to /home/melih/.ros/log/9dc1a708-f34f-11e8-a1a8-e84e0666f0cb/r
oslaunch-kinetic-server-4394.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://kinetic-server:35961/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

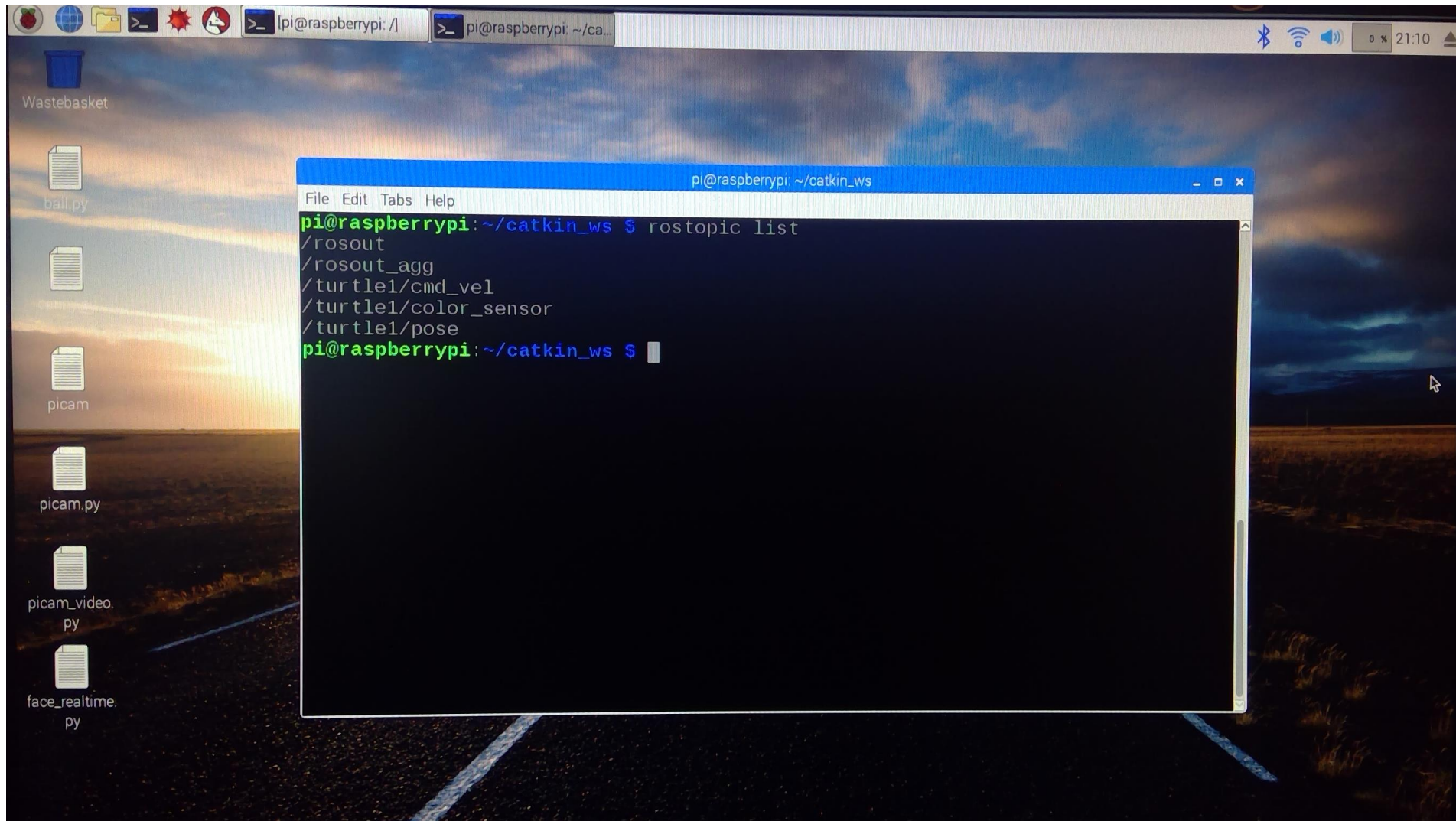
NODES

auto-starting new master
process[master]: started with pid [4404]
ROS_MASTER_URI=http://kinetic-server:11311/

setting /run_id to 9dc1a708-f34f-11e8-a1a8-e84e0666f0cb
process[rosout-1]: started with pid [4417]
started core service [/rosout]
```

ROS Networking

- On raspberrypi



The screenshot shows a Raspberry Pi desktop environment with a sunset background. On the left, there is a sidebar with icons for a Wastebasket and several Python files: ball.py, picam, picam.py, picam_video.py, and face_realtime.py. At the top, there is a taskbar with application icons and system status icons (Bluetooth, Wi-Fi, volume, and battery). A terminal window is open in the center, titled 'pi@raspberrypi: ~/catkin_ws'. The terminal shows the command 'rostopic list' being executed, which returns the following output:

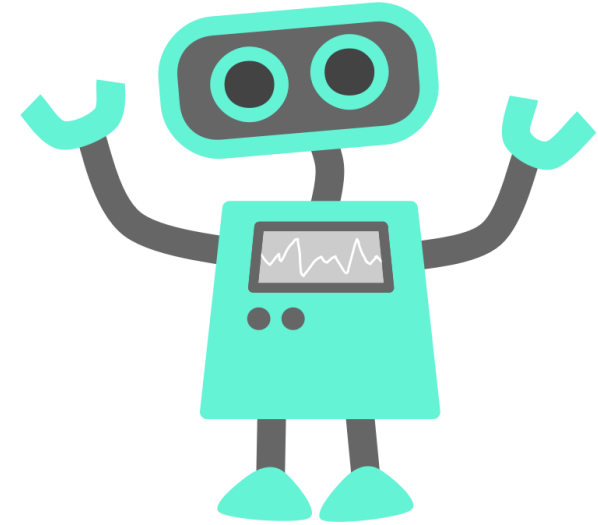
```
pi@raspberrypi:~/catkin_ws $ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
pi@raspberrypi:~/catkin_ws $
```

Intelligent Robotics 1

- **You learned about:**

- 3D Designing with Fusion 360
- Current 3D Printing Technologies and CURA
- Image Processing and OpenCV
- Python Programming
- Virtual Machines and Linux Ubuntu
- Robot Operating System (All the Basics!!!)
- Arduino and Raspberry Pi
- Sensors, Cameras, Actuators, Displays, Microphone, and Speakers
- Speech Recognition and Synthesis with Google Dialogflow and Amazon Polly
- Classical Logic and Fuzzy Logic

- **You built and programmed a real robot!**

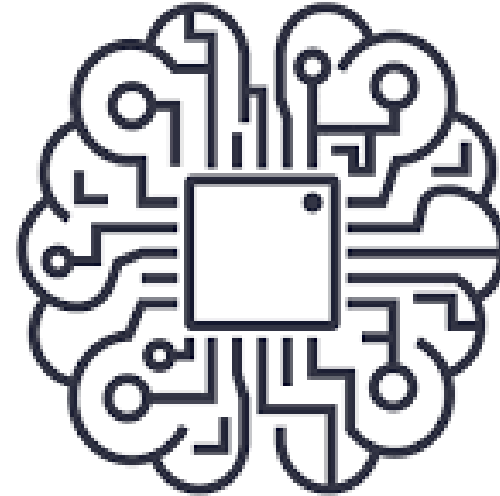




Thank you for
your hard work
and dedication!
Hope to see you all
in Intelligent Robotics 2!

What is in Intelligent Robotics 2?

- Introduction to Artificial Intelligence
 - Expert Systems
 - Prolog + Python
- Introduction to Machine Learning
 - Machine Learning
 - Classification Methods
 - Orange
 - TensorFlow
- Cloud Machine Learning Tools
 - Google Cloud
 - Vision API
 - Speech API
 - Natural Language API
- Robot Operating System (Kinetic)
 - Gazebo
 - ROS URDF
 - MoveIt
 - Indoor Mapping and Navigation with ROS
 - Image Processing with ROS.
 - Machine Learning with ROS
- Introduction to ROS 2



Interested in Becoming a Robotics Engineering

Electrical and Computer Engineering	Computer Science	Mechanical Engineering
Intelligent Robotics 1 ECE 478/578	Spoken Language Interfaces CS	Introduction to Robotics ME 410/510
Intelligent Robotics 1 ECE 479/579	Human Computer Interaction CS	Mechatronics ME 410
Intelligent Robotics 3 ECE 410/510	Object Oriented Programming CS	
Underwater Robotics ECE 510	Artificial Intelligence CS	
Mathematical Foundations of Machine Learning ECE	Machine Learning CS	
Embedded System Programming ECE	Advance Machine Learning CS	
Python Workshop ECE	Machine Learning Seminar CS	
	Computer Vision CS	
	Computational Photography CS	