ECE 478-578 Intelligent Robotics I

PhD. Husnu Melih Erdogan – Electrical & Computer Engineering

herdogan@pdx.edu Teaching Assistant



Introduction to ROS Part - 1





ROS (Robot Operating system)



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
 - Assignment 3 Part1

- Part 2 Details
 - ROS File System
 - ROS Package
 - How to create a package?
 - How to build a package?
 - Creating a Publisher Node
 - Creating a Subscriber Node
 - Creating a Launch File
 - How to use ROS .bagfiles?
 - Assignment 3- Part2

• Part 3 - Details

٠

- ROS Parameters
- **ROS Messages**
- ROS Services
- ROS Actions
- Assignment 4

- Part 4 Project
 - Rviz
 - Mapping
 - Localization
 - 2D Multi-Robot Simulator
 - Assignment 5



Books



Morgan Quigley, Brian Gerkey & William D. Smart



Mastering ROS for Robotics Programming

Design, build, and simulate complex robots using Robot Operating System and master its out-of-the-box functionalities

Lentin Joseph

PACKT open source®





Learning Robotics Using Python

Design, simulate, program, and prototype an interactive autonomous mobile robot from scratch with the help of Python, ROS, and Open-CV!

tin Joseph Copyrighted Material [PACKT] open source*



Courses

http://wiki.ros.org/Courses

- 3. University (Undergraduate & Graduate) Courses
- International ROS Summer School (University of Applied Sciences in Aachen, Germany)
- 240AR060: Introduction to ROS (IRI-UPC) (Barcelona) (● IRI ● UPC)
- Programming for Robotics Introduction to ROS (ETH Zurich) (Video lectures)
- Gaitech Education Portal, China
- Introduction to Robotics (CS460) (Prince Sultan University, Saudi Arabia)
- Introduction to Robotics (Bar-Ilan University, Israel)
- Advanced Robotics (CU Boulder)
- Advanced Robotics Systems (KU Leuven)
- Autonomous robotics (DEI Università di Padova)
- Autonomous Vehicles Freshman Research Initiative (University of Texas at Austin)
- CSCE 574: Robotics (University of South Carolina)
- CSE553: Mobile Robotics (Washington University in St Louis)
- CS1480: Building Intelligent Robots (Brown University)
- CS225B: Robot Programming Laboratory (Stanford University)
- CS324: Perception for Robotics (Stanford University, 2011)
- CS324: Perception for Manipulation (Stanford University, 2010)
- MEAM620: Robotics (University of Pennsylvania)
- CS445/660: Social Robotics Seminar (University at Albany)
- GEI740 "Programmation de robots mobiles" Behavior-based mobile navigation with Gazebo (Université de Sherbrooke)
- RTM-ROS Robotics in Japanese (Tokyo University)
- Robot Learning (Cornell University)
- Robot Programming (Sapienza University of Rome)
- Intelligent Robotics (University of Birmingham)
- RobotChallenge (Leibniz Universität Hannover)
- CSIS401: Introduction to Robotics (Siena College, Loudonville NY)
- ME 495: Embedded Systems in Robotics (Northwestern University, Evanston IL)
- Intelligent Robotics (Kyushu University, Japan)
- Programming for the Robot Operating System (University of Zagreb, Croatia)



What is ROS?

- ROS stands for Robot Operating System.
- However, it is not a real operating system such a Linux, Windows, OSX
- It is framework.
- It is more like a middleware that sits between your operating system and your program in C++, Python, Java, and MATLAB etc.



Why ROS is useful?

- A distributed, modular design
- A very large community
- Permissive Licensing
 - "lets people do anything they want with your code as long as they provide attribution back to you and don't hold you liable."
- Collaborative Environment
- It is free.
- Supports different programming languages
 - Matlap, C++, Python, Lisp
- An important technical skill to have in your resume.

Rights in Copyright NonProtective Proprietary







Why ROS is good for us?

- We use ROS in the Intelligent Robotics Lab because:
 - Some of our robots already supports ROS
 - ROS is a distributed system
 - It allows us do fast prototyping
 - There are already so many ready to use packages
 - Robot Theater project
 - It will also help you find Robotics Engineering jobs



Robotics Engineering Jobs

in 🛛	२ robot operating system	Vinited States	Search	6 2 6 6	45 🚯		Reactivate Premium			
Jobs 🔻	Date Posted LinkedIr	Features Company	Experience Leve	I ▼ All filters						
Sort by: Re	Sort by: Relevance 🔻 🔲 Split View 👻									
Robot 202 resul	operating system in United States	A Job alert Off	D	obotics Software F	ngineer		<i>с</i>)			
Q	Robotics Software Engineer Quest Groups LLC Hayward, California SQL, Python, GPU programming, JavaScript. Exp Operating System (ROS) and Gazebo. MS or Ph.I 6 days ago - T Easy Apply	erience with Robot D in Computer Science,	Robotics Software Engineer Formation Quest Groups LLC · Hayward, California Posted 6 days ago · 1,390 views Save							
fellow ROBOTS	Robotics Software Engineer Fellow Robots Burlingame, California		1	See how you compare to Reactivate Premium	407 applicant	S.				
	Fellow Robots is expanding its team and is looki Engineers that have extensive experience working New · in Easy Apply	ng for Robotics Software g on auton	Job Company • 407 applicants • 11-50 employees		Connections You have 0 connections at this					
+HINK	Robotics Engineer THINK Surgical San Francisco Bay Area		Mid-Senior level	Staffing and Recruiting	Add	>				
	At least three years of experience in software de operating system environment in C/C++. Experie 3 days ago - in Easy Apply	velopment in a real-time ence transl	Job description Required Skills:							
Motek Teoreelaats	Senior SLAM and Robotics Engineer MoTek Technologies Redwood City, California		 MS or Ph.D in Computer Science, Robotics, or a related field 4+ years of robotic software development 							
	Experience with the Robot Operating System (ROS). MS or PhD in Computer Science, Robotics, or a related field or BS with 5+ years Be an early applicant 3 weeks ago • In Easy Apply		 experience Excellent program in C++ Demonstrated ability 	Jen Burns Partner at Palo Alto, C	n Burns 2nd Irtner at Quest Groups LLC Io Alto, California					
⋎ fetch	Robotics Engineer Fetch Robotics San Jose, California Experience with the Robot Operating System (Rd Science, Robotics, or a related field or BS with 2- 4 weeks ago - T Easy Apply	DS). MS in Computer - years applica	(including extende pose estimation of Experience with Rc (ROS) and Gazebo Experience control Obstacle detection range sensors, etc	Send InMail Seniority Level Mid-Senior level Industry						



History of ROS

- It is developed in 2007 at the Stanford Artificial Intelligence Lab.
- It was managed by Willow Garage.
- Open Source Robotics Foundations has been managing it since 2013.





Current Distributions



Install

Get ROS Indigo Igloo on Ubuntu Linux





Get ROS Kinetic Kame on Ubuntu Linux

(Recommended for Latest LTS)





Get ROS Lunar Loggerhead on Ubuntu Linux

(Recommended for Latest)



ROS Melodic Morenia Released May, 2018 Latest LTS, supported until May, 2023





ROS Industrial

ROS-Industrial is an open-source project that extends the advanced capabilities of ROS software to manufacturing.



http://rosindustrial.org/



Some of Our ROS Powered Robots







ROS Philosophy

- Peer to Peer
 - Individual programs communicate over defined API (ROS messages, services)
- Distributed
 - Programs can be run on multiple computers and communicate over the network.

Supports Multiple Languages

- ROS modules can be written in any language for which a client library exist (C++, Python, MATLAB, Java, etc.)
- Light-weight
 - Stand-alone libraries are wrapped around with in a thin ROS layer.
 - It doesn't mean ROS doesn't use your CPU or don't need a powerful PC.
 - Some packages could use all almost your system resources.

• Free and Open-Source

- It is totally free and supported by a considerably large community.
- It is backed by many universities and industrial companies.
- It becomes a standard in the robotics field.



ROS Core Components

• There are three core components:

Communications Infrastructure:

- Message Passing
- Remote Procedure Calls
- Recording and Playback of Messages
- Distributed Parameter System

Tools:

- Command-Line Tools
- Rviz
- Rqt

Robot-Specific Features:

- Standard Message Definitions
- Robot Geometry Library
- Robot Description Language
- Localization
- Mapping
- Navigation



Communications Infrastructure

- At the lowest level, ROS offers a message passing interface that provides inter-process communication and is commonly referred to as a middleware.
- The ROS middleware provides these facilities:
 - publish/subscribe anonymous message passing
 - recording and playback of messages
 - request/response remote procedure calls
 - distributed parameter system



Communications Infrastructure Message Passing

- A communication system is often one of the first needs to arise when implementing a new robot application.
- ROS's built-in and well-tested messaging system saves you time by managing the details of communication between <u>distributed nodes</u> <u>via the anonymous publish/subscribe mechanism</u>.
- Another benefit of using a message passing system is that it forces you to implement clear interfaces between the nodes in your system, thereby improving encapsulation and promoting code reuse.
- The structure of these message interfaces is defined in the message IDL (Interface Description Language).



Communications Infrastructure Remote Procedure Calls

- The asynchronous nature of publish/subscribe messaging works for many communication needs in robotics, but sometimes you want synchronous request/response interactions between processes.
- The ROS middleware provides this capability using <u>services</u>. Like topics, the data being sent between processes in a service call are defined with the same simple message IDL (Interface Description Language).



Communications Infrastructure: Recording and Playback of Messages

- Because the publish/subscribe system is anonymous and asynchronous, the data can be easily captured and replayed without any changes to code.
- Say you have Task A that reads data from a sensor, and you are developing Task B that processes the data produced by Task A. ROS makes it easy to capture the data published by Task A to a file, and then republish that data from the file at a later time.
- This is a powerful design pattern that can significantly reduce your development effort and promote flexibility and modularity in your system.
- We can run the robot itself only a few times, recording the topics we care about, and then replay the messages on those topics many times, experimenting with the software that processes those data.



Communications Infrastructure Distributed Parameter System

- The ROS middleware also provides a way for tasks to share configuration information through a global key-value store.
- This system allows you to easily modify your task settings, and even allows tasks to change the configuration of other tasks.



Tools

- One of the strongest features of ROS is the powerful development toolset.
- These tools support introspecting, debugging, plotting, and visualizing the state of the system being developed.
- The underlying publish/subscribe mechanism allows you to spontaneously do self-analysis of the data flowing through the system, making it easy to comprehend and debug issues as they occur.
- The ROS tools take advantage of this introspection capability through an extensive collection of graphical and command line utilities that simplify development and debugging.



Tools - Command-Line Tools

- ROS can be used 100% without a GUI.
- All core functionality and introspection tools are accessible via one of more than 45 ROS command line tools.
- There are commands for launching groups of nodes;
 - introspecting topics,
 - services,
- and actions;
 - recording and playing back data;
 - and a host of other situations.



Tools - rviz

- Perhaps the most well-known tool in ROS, rviz provides general purpose, three-dimensional visualization of many sensor data types and any URDFdescribed robot.
- rviz can visualize many of the common message types provided in ROS, such as
 - laser scans,
 - three-dimensional point clouds,
 - and camera images.
- Visualizing all of your data in the same application not only looks impressive, but also allows you to quickly see what your robot sees, and identify problems such as sensor misalignments or robot model inaccuracies.









rviz





rviz





Tools - rqt

- ROS provides rqt, a Qt-based framework for developing graphical interfaces for your robot.
- You can create custom interfaces by composing and configuring the extensive library of built-in rqt plugins into tabbed, split-screen, and other layouts.
- You can also introduce new interface components by writing your own rqt plugins .



Tools - rqt

demo - RosGui	•										● 🛛 😣
Web		Publisher					Robot Steering		🗶 Logger Level		
http://www.ros.org/wiki/rqt	C	C Topic cmd vel3	Type //Float32	Treg. 5 The	Iz 🕒		/cmd vel		Nodes	Loggers	Levels
III ROS.org	About Supp	topic v type v /cmd_vel2 std_msg data float32	rate en s/Float32 10.00 Tru	abled expressio	n 20			1.00 +	/rosout /rqt_gui_cpp /rqt_gui_cpp /rviz_1343921	ros ros.moveit_co ros.roscpp	Debug Info Warn Error
Documentation	Brow	▼ /cmd_vel3 std_msg data float32	s/Float32 5.00 Tru	sin(i/20)*	10	-			, <u>.</u>	ros.roscpp.su	Fatal
rqt	U										
rqt: rat console rat dep rat graph rat gui rat plot rat pose view rat publisher rat py rat service caller rat tf tree rat topic rat w	<u>rat gui cpp</u> / common r /eb							-			
1. Stack Summary											
Integration of the ROS package system and ROS-specific pl							3.00 🗘 <	> -3.00 🗘			
Author: Maintained by Dirk Thomas License: BSD								Stop	Refresh	(())	
Console					🕷 Plot						
🕒 Load 🛛 🖉 Save 🔲 Pause Displayi	ng 9 Messages			Resize Columns	Торіс	/cmd_vel3/da	ta Sut	oscribe Topic		Pause	Remove All
Message	Severity	Node	Time	•							
#9 Loading Setup Assistant Complete	Info	/moveit_setup_assistant	11:11:25.344 (2012-08-	-02) /rosout, /move	- 29 -						
#8 Listening to 'moveit_planning_scene'	Info	/moveit_setup_assistant	11:11:25.294 (2012-08-	-02) /rosout, /move	- 23.2 -						
#7 Starting scene monitor	Info	/moveit_setup_assistant	11:11:25.293 (2012-08-	-02) /rosout, /move	-	Λ	Δ		Δ		Λ
#6 Configuring kinematics solvers	Info	/moveit_setup_assistant	11:11:25.107 (2012-08-	-02) /rosout, /move	17.4 -		$- \wedge$	$\Lambda = \Lambda$		\cap	$\Lambda = I \Lambda = 1$
#4 Robot semantic model successfully loaded.	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-	-02) /rosout	11.6 -		$ \rangle$	$1 \wedge 1 \wedge$	$ \wedge $		$ \wedge $
#5 Setting Param Server with Robot Seman	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-	-02) /rosout	5.8 -		$ \rangle$	$(\land) \land)$	$-\Lambda \Lambda \Lambda$	(/)	$\Lambda / \Lambda ' \perp$
Exclude Rules: Messages matching ANY of the	ese rules will N	OT be displayed			-5.8 -			[- I \ V I	$\langle V \rangle \rangle$	$V \mid $
Severity Filter: Debug Info Warning Err	or Fatal				-116-	$ \rangle \rangle \rangle$			$\left(\right) $	MI	$\mathcal{A} \mid \mathcal{A} \mid$
					-	$\langle \langle \rangle \rangle$	()		(1 + 1)		
								V			
Highlight Rules: Message matching ANY of these rules will be highlighted -23.2 -											
Message Filter: monitor				Regex 💻 🚘	-29 -						
				+		0	200	400 - /cmd_vel2/data	600 – /cmd_vel3/data	800	1.000



Tools – rqt_graph

- The rqt_graph plugin provides introspection and visualization of a live ROS system
- It shows nodes and the connections between them
- It allows you to easily debug and understand your running system and how it is structured.



Tools – rqt_graph





Tools – rqt_plot

- The rqt_plot plugin help you monitor encoders, voltages, or anything that can be represented as a number that varies over time.
- The rqt_plot plugin allows you to choose the plotting backend (e.g., matplotlib, Qwt, pyqtgraph) that best fits your needs.



Tools – rqt_plot





Tools – rqt_topic

- The rqt_plot plugin help you monitor encoders, voltages, or anything that can be represented as a number that varies over time.
- The rqt_plot plugin allows you to choose the plotting backend (e.g., matplotlib, Qwt, pyqtgraph) that best fits your needs.



Tools – rqt_topic

😣 🔿 🗊 rqt_topicTopicPlugin - rqt										
Q Topic Monitor D @ − O										
То	pic 🔻	Туре	Bandwidth	Hz	Value					
	 ✓ /imu/data ✓ /imu/mag ✓ /imu/rpy ▶ header ▼ vector x y z /imu/temperature /rosout 	sensor_msgs/Imu geometry_msgs/Vector3Stamped geometry_msgs/Vector3Stamped std_msgs/Header geometry_msgs/Vector3 float64 float64 float64 std_msgs/Float32 rosgraph_msgs/Log	6.24KB/s 935.23B/s 934.96B/s	19.46 19.46 19.46	-0.30104295895060923 -0.08551920999488644 -2.7362312256211423 not monitored not monitored					
	/rosout_agg	rosgraph_msgs/Log			not monitored					



Tools – rqt_publisher

- The rqt_plot plugin help you monitor encoders, voltages, or anything that can be represented as a number that varies over time.
- The rqt_plot plugin allows you to choose the plotting backend (e.g., matplotlib, Qwt, pyqtgraph) that best fits your needs.


Tools – rqt_publisher

ublisher ublisher Copic Topic Type Type Type Type Tate Type Type Type	File Plugins Running Pers	spectives Help
Topic /move_group/result Type moveit_msgs/MoveGroupActionResult Freq. 1 Hz Image: Content of the second	ublisher	D@@ 0
topic v type rate expression ▼ /move_group/cancel actionlib_msgs/GoalID 1.00 stamp time genpy.Time[0] id string " ▼ /rqt_marble sensor_msgs/NavSatFix 1.00 ▶ header std_msgs/Header 1.00 ▶ status sensor_msgs/NavSatStatus 0.0 latitude float64 0.0 longitude float64 0.0	C Topic /move_group/resu	Jlt 🔻 Type moveit_msgs/MoveGroupActionResult 💌 Freq. 1 💌 Hz 🖶 💻 🗷
✓ /move_group/cancel actionlib_msgs/GoalID 1.00 stamp time genpy.Time[0] id string " ✓ /rqt_marble sensor_msgs/NavSatFix 1.00 ▶ header std_msgs/Header	topic 🔻	type rate expression
attitude rtoat64 0.0 position_covariance float64[9] [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] position_covariance_type uint8 0 ✓ /place/feedback moveit_msgs/PlaceActionFeedback 1.00 ▶ header std_msgs/Header - ▶ status actionlib_msgs/GoalStatus - ▶ feedback moveit_msgs/PlaceFeedback 1.00 ▼ /move_group/result moveit_msgs/NoveGroupActionResult 1.00 ▶ header std_msgs/Header - ▶ faeder std_msgs/Header - ▶ faeder std_msgs/GoalStatus - ▶ faeder std_msgs/MoveGroupActionResult 1.00 ▶ header std_msgs/Header - ▶ status actionlib_msgs/GoalStatus - ▶ tatus actionlib_msgs/GoalStatus - ▶ result moveit_msgs/MoveGroupResult -	 /move_group/cancel stamp id /rqt_marble header status latitude longitude altitude position_covariance_type /place/feedback header status feedback move_group/result header status result 	actionlib_msgs/GoalID 1.00 time genpy.Time[0] string " sensor_msgs/NavSatFix 1.00 std_msgs/Header 0.0 sensor_msgs/NavSatStatus 0.0 float64 0.0 float64[9] [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,



Robot-Specific Features

- In addition to the core middleware components, ROS provides common robot-specific libraries and tools that will get your robot up and running quickly. Here are just a few of the robot-specific capabilities that ROS provides:
 - Standard Message Definitions for Robots
 - Robot Geometry Library
 - Robot Description Language
 - Pre-emptiable Remote Procedure Calls
 - Diagnostics
 - Pose Estimation
 - Localization
 - Mapping
 - Navigation



Robot-Specific Features – Pose Estimation





Robot-Specific Features - Mapping



How to Install ROS?

- Install ROS Kinetic on PC Ubuntu Recommended
- Install ROS Kinetic on Virtual Machine
- Install ROS Kinetic on Raspberry Pi 3 Stretch Project2

ROS Workspace Environment

- Standard name for it catkin_ws
- However, you can use whatever name you want.
- You can create it in any user location

- Before you use a work space first you need to source your setup.*sh.
- The setup.bash files setup your environment variables and paths.

ROS Workspace Environment

- setup.bash Environment Setup File
- The binary catkin package includes a set of environment setup files that are used to extend your shell environment, so that your terminal can find and use any resources that have been installed to that location.
- the setup file included in the root of the distribution install directory (usually /opt/ros/<ros_distribution_name>)
- For example with ROS Kinetic on Ubuntu the files would be located as such:

```
/

opt/

ros/

kinetic/

setup.bash -- Environment setup file for Bash shell

setup.sh -- Environment setup file for Bourne shell

setup.zsh -- Environment setup file for zshell

...
```


ROS Workspace Environment

- If you want to change the environment of your current shell
 source /opt/ros/kinetic/setup.bash
- If you want to overlay a workspace on top of your environment.
 source devel/setup.bash
- If you want to source your terminal automatically.
 - echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
 - source ~/.bashrc

What is catkin?

A catkin workspace is a folder where you modify, build, and install catkin packages. (wiki.ros.org)

Catkin packages can be built as a standalone project, in the same way that normal cmake projects can be built, but <u>catkin also provides the</u> <u>concept of workspaces, where you can build multiple, interdependent</u> <u>packages together all at once.</u>

Create a Catkin Workspace

- mkdir -p ~/catkin_ws/src

(create a new directory called src in another directory called catkin ws)

- cd ~/catkin_ws/src
- (go into the src directory)
- catkin_init_workspace

(initialize the workspace)

- cd ~/catkin_ws/

(go into your brand new workspace)

- catkin_make

(build your code in a catkin workspace)

source devel/setup.bash

(Before continuing make sure you source your new setup.*sh file)

- echo \$ROS_PACKAGE_PATH

Test your path: /home/youruser/catkin_ws/src:/opt/ros/kinetic/share 🥰

- You can also use **catkin build** command to build your code in your package.
- It is as same as the **catkin_make** command.

- <u>Do not use catkin_make and catkin build at the same time</u>
- <u>Do not mix them</u>

Catkin_ws						
く 〉 ûHome cat	kin_ws				۹ :	
 Recent Home Desktop Documents Downloads Music 	build	devel	src			
 ▶ Pictures ▶ Videos ₩ Trash ₽ Network 						
Computer Connect to Server						

Catkin Workspace – src folder

- The source space is the folder is where catkin will be expected to look for packages when building.
- Each catkin project desired to be compiled from source should be checked out into subdirectories inside this directory.
- Packages are found recursively so they do not have to be direct subfolders.

Catkin Workspace – dev folder

- The development space is where catkin generates the binaries and runtime libraries which are executable before installation.
- After the build step, inside this folder is expected everything needed to run nodes in packages which have been built.
- The development space can not be a folder which contains ROS packages in subfolders.
 - Ex: It can not equal to the workspace root as this would make the source space a subfolder which would lead to packages being found multiple times.

Catkin Workspace – build folder

• The build space is the folder in which cmake is invoked and generates artifacts such as the CMakeCache

Catkin Workspace – Overlying Workspaces

- In catkin style workspaces, overlaying of one workspace on top of another workspace is supported. E
- Each overlay can be built on top of another overlay, and any packages in a higher overlay will mask out packages in lower overlays.
- When overlaying any package all packages which depend on that package must be overlayed if they are intended to be used.
- This must be checked by the user creating the overlays. If a core package is overlayed and changes how it works, any package which depends on it and relies on the old behavior (whether runtime, or link time) will crash.

Catkin Workspace – Overlying Workspaces

- mkdir ws1/src
- mkdir ws2/src
- mkdir ws3/src
- ws1 as an overlay of /opt/ros/kinetic/setup.bash
- ws2 as an overlay of ws1
- ws3 as an overlay of ws2
- <u>http://wiki.ros.org/catkin/Tutorials/workspace_overlaying</u>

Catkin Workspace – Overlying Workspaces

- cd ~/catkin_ws1
- catkin_make
- source /opt/ros/kinetic/setup.bash
- cd ~/catkin_ws2
- catkin_make
- source ~/catkin_ws1/devel/setup.bash
- cd ~/catkin_ws3
- catkin_make
- source ~/catkin_ws2/devel/setup.bash

ROS Master

- In charge of managing the communication between nodes
- Every node registers with the master when you run them
- It tracks publishers and subscribers to topics as well as services.
- The role of the Master is to enable individual ROS nodes to locate one another.
- Once these nodes have located each other they communicate with each other peer-to-peer.
- The Master also provides the parameter server.

ROS Master

- In order to start a master type:
 - roscore

melih@melih-ros1:~/catkin_ws\$ roscore
... logging to /home/melih/.ros/log/1cfb8bd2-e0e1-11e8-bc2c-08002780f4a0/roslaun
ch-melih-ros1-2429.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.</pre>

started roslaunch server http://melih-ros1:36959/
ros_comm version 1.12.14

SUMMARY

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES

auto-starting new master process[master]: started with pid [2440] ROS_MASTER_URI=http://melih-ros1:11311/

setting /run_id to 1cfb8bd2-e0e1-11e8-bc2c-08002780f4a0
process[rosout-1]: started with pid [2453]
started core service [/rosout]

Ros Nodes

- Single-purpose, executable program written in Python, C++, Matlab etc
- Individually complied, executed and managed.
- You can run, stop, and kill nodes whenever you want.
- They run in parallel.
- In order to run a node type:
 - rosrun package_name node_name
- See active nodes running on ROS
 - rosnode list
- In order to get more information about nodes type:
 - rosnode info node_name

ROS Master and Nodes

ROS Master Example

ROS Master Example

ROS Topics

- ROS Nodes communicate over topics.
- Each node can **publish** or **subscribe** topics
- When one of the nodes register with the master, node also tells master that I am the publisher of these topics. Therefore, other nodes can know who publishes that topic and subtribe that topic.
- Topics is used to transfer data from publisher node to subscriber node
- List all active topics :
 - rostopic list
- Subscribe and print the contents of a topic
 - rostopic echo /topic-name
- Show information about the topic
 - rostopic info /topic-name
- Show all the other possible rostopic commands
 - rostopic -h

ROS Master and Nodes

tate

Other ROS Topic Commands

- **rostopic bw** display bandwidth used by topic
- rostopic echo print messages to screen
- **rostopic hz** display publishing rate of topic
- **rostopic list** print information about active topics
- **rostopic pub** publish data to topic
- rostopic type print topic type

Using rqt-graph

- rqt_graph tool is used to create a dynamic graph of what's going on in the system that includes all nodes and topics
- To run rqt-graph
 - rosrun rqt_graph rqt_graph

Using rqt-graph

 Tqt_graph_RosGraph - rqt Node Graph Nodes/Topics (all) Oroup namespaces Group actions Hide dead sinks Hide leaf topics Hide Debug Highligi 	D@ - (
master	
mobile_base_nodelet_manager	robot_state_publisher
/mobile_base_nodelet_manager/bond/mobile_base_nodelet_manager	<pre>/robot_state_publisher</pre>
mobile_base mobile_base/commands/velocity mobile_base/sensors/core mobile_base (mobile_base) (mobile_base (mobile_base)	diagnostic_aggregator diagnostic_aggregator
capability_server	
/capability_server/bonds(capability_server	
app_manager turtlebot (/app_manager +turtlebot/status	interactions /interactions

ROS Messages

- ROS message describes the data values that ROS nodes publish.
- Messages help ROS tolls 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 <u>http://wiki.ros.org/msg</u> and check the message type
- Publish a message to a subscriber ROS node on command line.
 - rostopic pub /mytopic std msg/String "data: 'Portland State University'"

Summary of ROS Messages

Image Source Link

Example ROS Message Definition:

geometry_msgs/Point Message

File: geometry_msgs/Point.msg

Raw Message Definition

This contains the position of a point in free space
float64 x
float64 y
float64 z

Compact Message Definition

float64 x float64 y float64 z

autogenerated on Tue, 17 Jan 2017 20:57:22

Message Types

You must not use the names of built-in types or header when constructing your own message types.

Built-in types:

Primitive Type	Serialization	C++	Python
bool (1)	unsigned 8-bit int	uint8_t(2)	bool
int8	signed 8-bit int	int8_t	int
uint8	unsigned 8-bit int	uint8_t	int(3)
int16	signed 16-bit int	int16_t	int
uint16	unsigned 16-bit int	uint16_t	int
int32	signed 32-bit int	int32_t	int
uint32	unsigned 32-bit int	uint32_t	int
int64	signed 64-bit int	int64_t	long
uint64	unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ascii string (4)	std::string	str
time	secs/nsecs unsigned 32-bit ints	• ros::Time	rospy.Time
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

ROS Messages - Remapping

Image Source Link

ROS Messages - Remapping

Node 1 : rosrun tutorials listener.py /topic:=/topic1
Node 2 : rosrun tutorials talker.py /topic:=/topic1

ROS Package

- Software in ROS is organized in packages.
- A package can contain your ROS nodes, any libraries, a dataset, configuration files, message types, scripts, service types, some third-party software.
- When you want to create a project in ROS, first you need to create a package
- You can also download and install many packages written by other ROS users around the world
- You can submit the ROS package you write to the ROS package repository
- Every ROS package is for certain ROS distributions
- Packages have dependencies
- <u>http://www.ros.org/browse/list.php?package_type=package&distro=indigo</u>



ROS Package – Package.xml

Toggle line numbers

- 1 <?xml version="1.0"?>
- 2 <package>
- 3 <name>beginner tutorials</name>
- 4 <version>0.1.0</version>
- 5 <description>The beginner_tutorials package</description>
- 7 <maintainer email="you@yourdomain.tld">Your Name</maintainer>
- 8 <license>BSD</license>
- 9 <url type="website">http://wiki.ros.org/beginner tutorials</url>
- 10 <author email="you@yourdomain.tld">Jane Doe</author>
- 11

6

12 <buildtool_depend>catkin</buildtool_depend>

- 13
- 14 <build depend>roscpp</build depend>
- 15 <build depend>rospy</build depend>
- 16 <build_depend>std_msgs</build_depend>
- 17
- 18 <run_depend>roscpp</run_depend>
- 19 <run_depend>rospy</run_depend>
- 20 <run_depend>std_msgs</run_depend>
- 21
- 22 </package>



ROS Launch

- Sometimes you have so many nodes with many different arguments and parameters to run in a project.
- Roslaunch is a tool for easily launching multiple ROS nodes as well as setting parameters in the parameter server.
- The basic idea is to list, in a specific XML format, a group of nodes that should be started at the same time.



ROS Launch – Simple Example

<launch>

<node name="telephone1" pkg="ros_lecture1" type="telephone.py" output="screen"/>
</launch>



Example Launch File

```
<launch>
1
     <node
2
       pkg="turtlesim"
3
       type="turtlesim_node"
4
       name="turtlesim"
5
       respawn="true"
6
     />
7
     <node
8
       pkg="turtlesim"
9
       type="turtle_teleop_key"
10
       name="teleop_key"
11
       required="true"
12
       launch-prefix="xterm -e"
13
     />
14
     <node
15
       pkg="agitr"
16
       type="subpose"
17
       name="pose_subscriber"
18
       output="screen"
19
     />
20
   </launch>
21
```

Listing 6.1: A launch file called example.launch that starts three nodes at once.



Some useful ROS command-line tools

- <u>rospack</u>: find and retrieve information about packages
- <u>catkin create pkg</u>: create a new package
- <u>catkin make</u>: build a workspace of packages
- <u>rosdep</u>: install system dependencies of a package
- <u>rqt</u>: In <u>rqt</u> there is a plugin called "Introspection/Package Graph", which visualizes package dependencies as a graph



Turtlesim Example



http://wiki.ros.org/turtlesim

Roscore

- Nodes
 - rosrun turtlesim turtlesim_node
 - rosrun turtlesim_turtle_teleop_key

- Topic

rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'



geometry_msgs/Twist

- Twist message should have:
 - a linear component for the x, y, and z axes velocities,
 - an angular component for the for the x, y, and z axes angular rate

geometry_msgs/Twist Message

File: geometry_msgs/Twist.msg

Raw Message Definition

This expresses velocity in free space broken into its linear and angular parts. Vector3 linear Vector3 angular

Compact Message Definition

geometry_msgs/Vector3 linear geometry_msgs/Vector3 angular



Ubuntu Terminal – Open New Tab

e melih@melih-ros1:~ elih@melih-ros1:~\$	
	S 🖨 🗊 Preferences
	General Shortcuts Profiles Encodings Show menubar by default in new terminals Enable mnemonics (such as Alt+F to open the File menu) Enable the menu accelerator key (F10 by default) Enable the menu accelerator key (F10 by default)
	Open new terminals in: Tab
	Help



Resources

- <u>http://www.iris.ethz.ch/the-institute/robotics-systems-lab.html</u>
- http://www.ros.org/

