EE405A
Robotics Operating System (ROS) - 1

(TA) Hyunki Seong
School of Electrical Engineering
KAIST

March 12, 2021

hynkis@kaist.ac.kr

Experiment Objectives

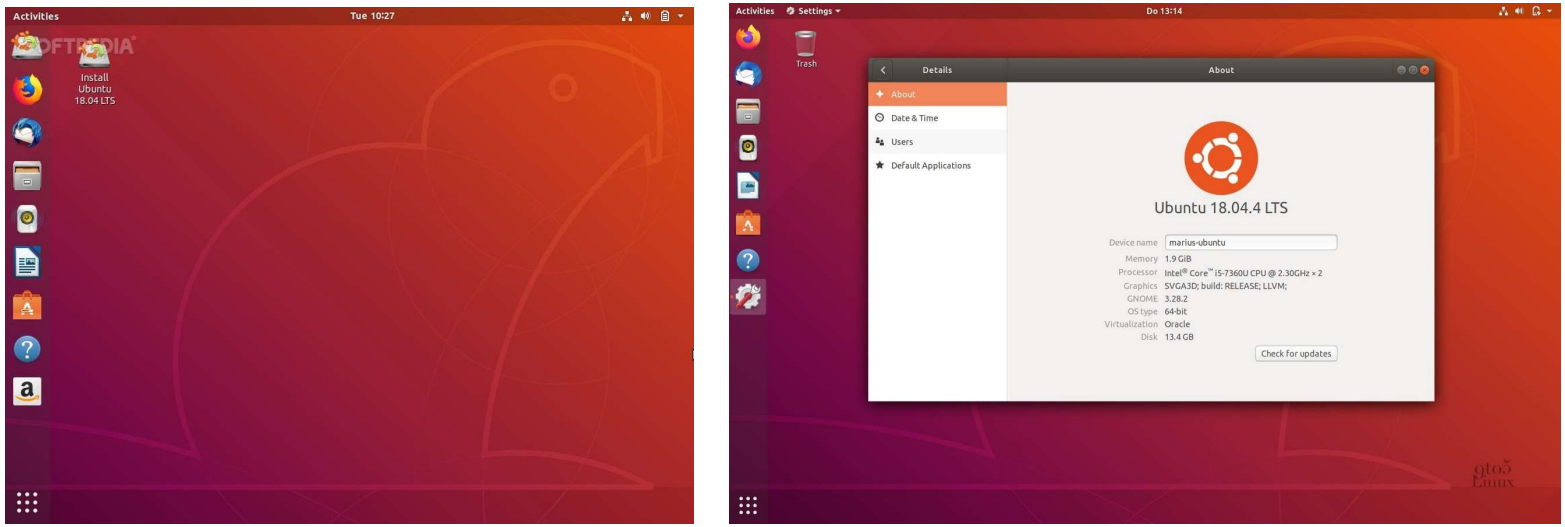
In this week, you will do the following:

- Get brief tips on installing Ubuntu(Linux-based OS).
- **Understand the Robotics Operating System (ROS).**
- Install & Setup ROS.
- Run ROS tutorial.

Ubuntu Installation

Ubuntu Installation

- Ubuntu 18.04
  - Ubuntu is a Linux kernel-based Operating System(OS).
  - It is the **most popular developer OS** for personal desktops and laptops.
  - Recently, **Ubuntu 18.04** release version is suitable for use.



Several figures from the Ubuntu 18.04 desktop

Ubuntu Installation

➢ Ubuntu 18.04 Installation

❑ There are two ways of Ubuntu Installation in your PC.

1. Install the Ubuntu alongside with Window **in Dual Boot**.
   - Link(Eng) : https://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/how-to-install-ubuntu-18-04-alongside-with-windows-10-or-8-in-dual-boot.html
   - Link(Kor) : https://coding-factory.tistory.com/494
2. Install the Ubuntu **in a Virtual Machine**.
   - Link(Eng) : https://codebots.com/library/techies/ubuntu-18-04-virtual-machine-setup

❑ **We strongly recommend you install Ubuntu in Dual Boot** (The virtual machine is too heavy and slow.)

Background

1. Robotics Operating System (ROS)
2. Node, Topic, Publish/Subscribe
3. Applications of ROS

Robotics Operating System(ROS)

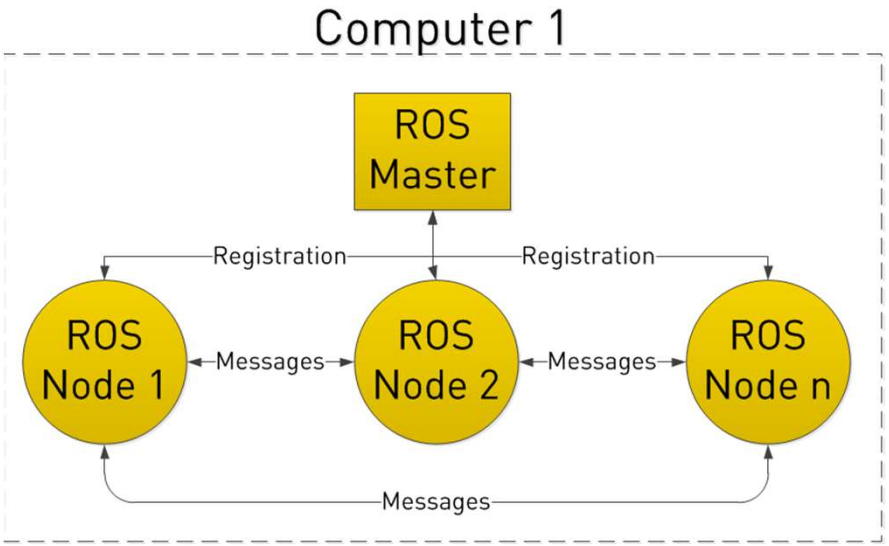
- What is ROS?
  - **The Robot Operating System(ROS)** is a flexible framework **for writing robot software**.
  - It is a collection of **tools, libraries, and conventions** that aim to **simplify the task of creating complex and robust robot behavior** across a wide variety of robotic platforms.
  - It has a large **ecosystem** to encourage **collaborative robotics software development.**

ROS distributions for Ubuntu 16.04(Kinetic), 18.04(Melodic), 20.04(Noetic)

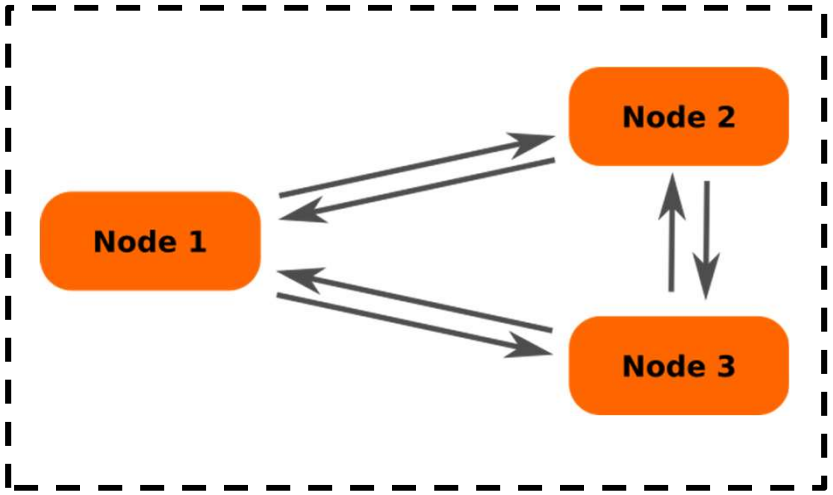
ROS for robotics software development

Robotics Operating System(ROS)

- What is ROS?

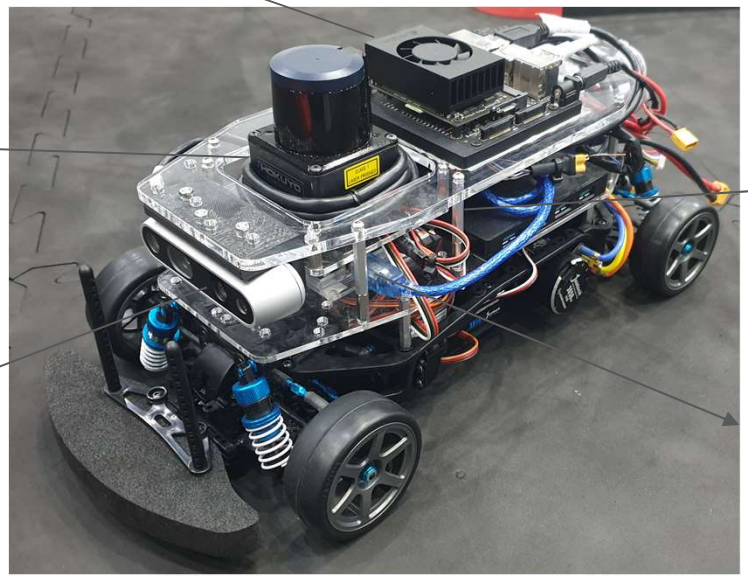


A robot system



Robotics Operating System(ROS)

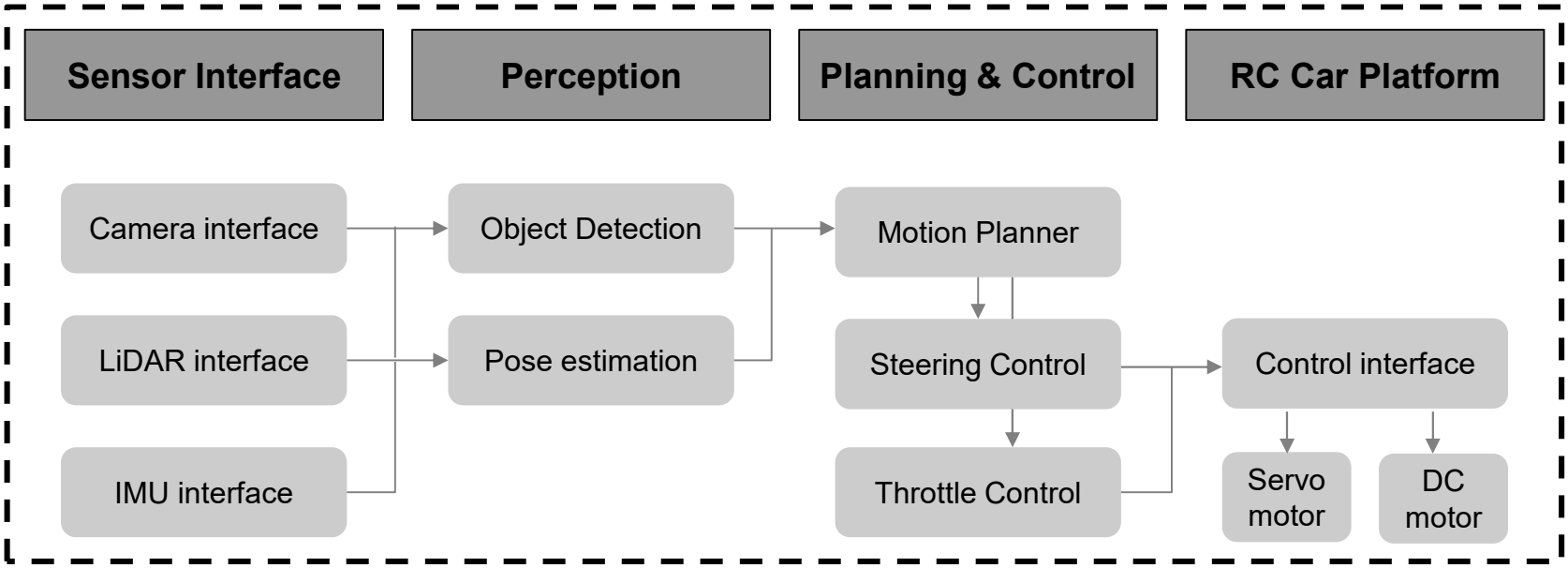
- What is ROS?

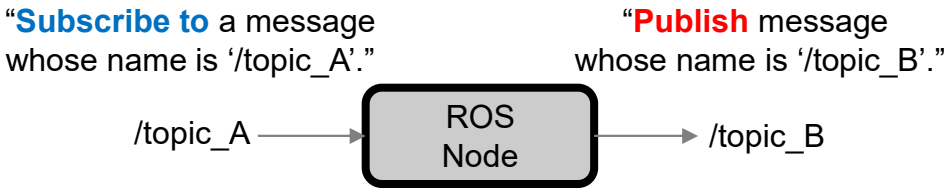


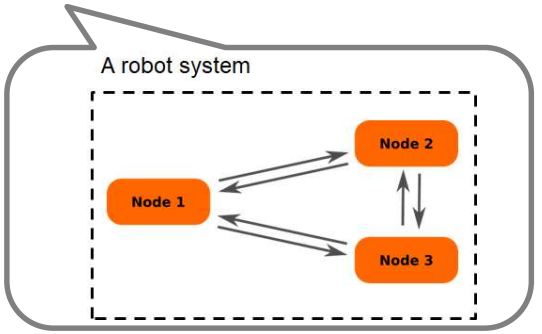
Robotics Operating System(ROS)

➢ What is ROS?

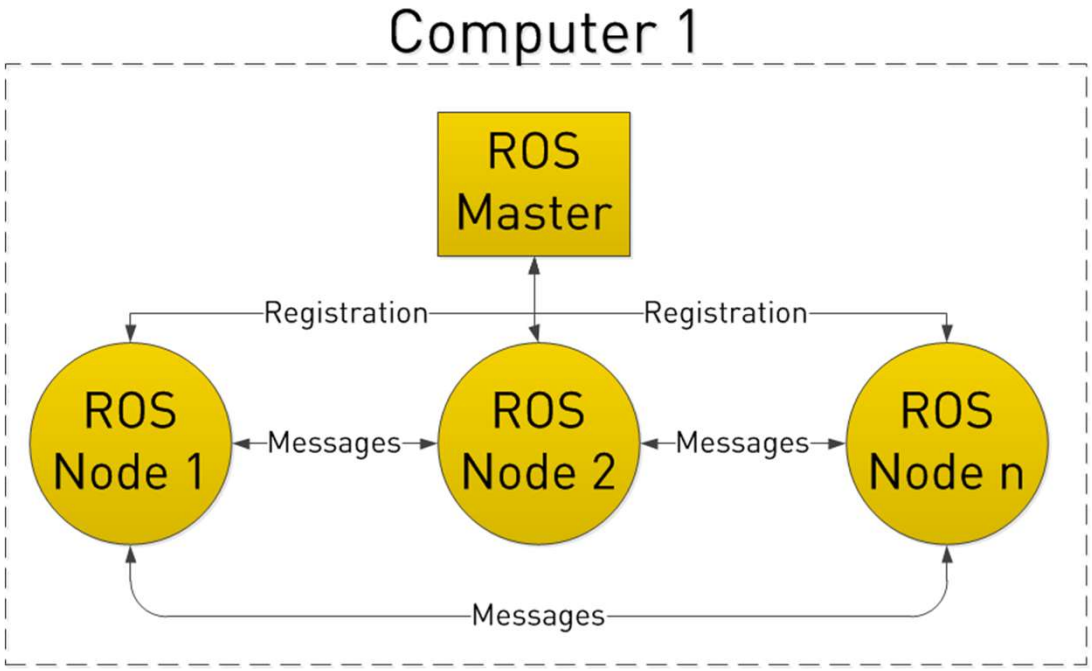
A vehicle system







Robotics Operating System(ROS)



- Important terminologies in ROS
  - **Node**
  - **Master**
  - **Message**
  - **Topic**
  - **Bag**

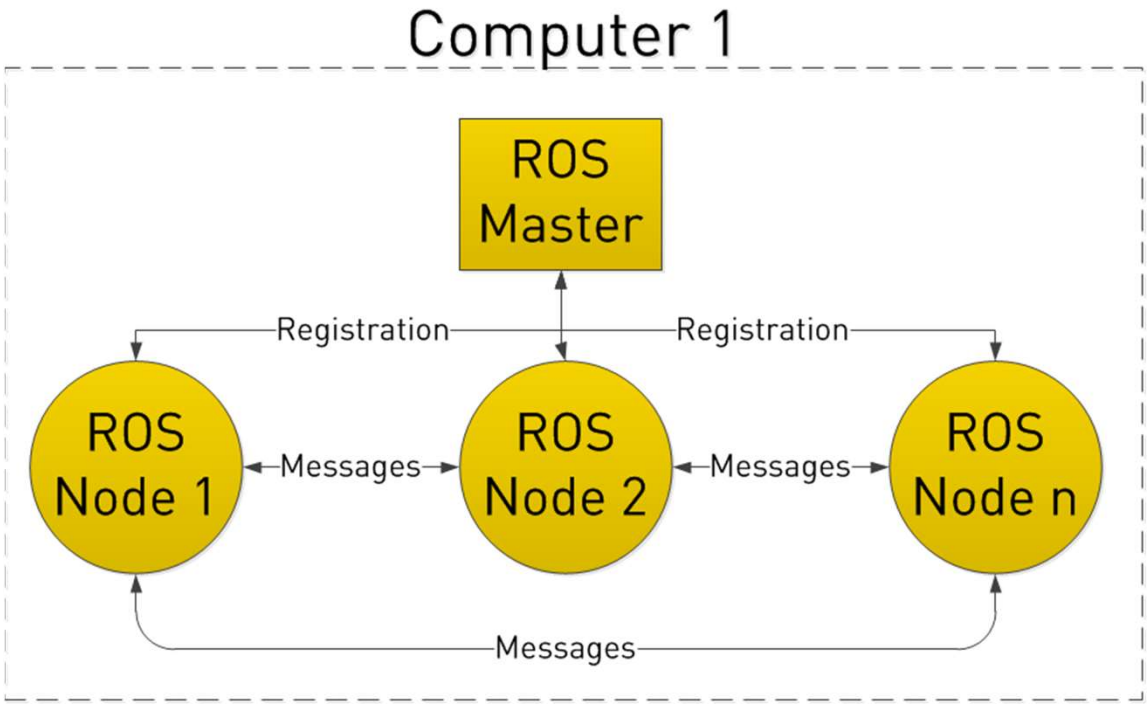
Reference : http://www.clearpathrobotics.com/assets/guides/kinetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

Robotics Operating System(ROS)

- Concepts
  - ROS is processed in the **peer-to-peer network** that are processing data together.
  - Important terminologies: **Node, Master, Message, Topic, Bag**, …
  - **Nodes :** Nodes are processes that perform computation. A robotic system will usually comprise many nodes; one node to control a laser range-finder, one node to control wheel motors, one node to perform localization, control, and so on.
  - **Master :** Master provides name registration of all nodes and lookup information, much like a DNS server.
  - **Message :** Nodes communicate with each other by **passing messages**. A message is simply a data structure, comprising typed fields.
  - **Topic :** The topic is a **name** that is used to identify the content of the message.
  - **Bags** : Bags are a format for **saving and playing back ROS message data**. Bags are important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

Reference : http://library.isr.ist.utl.pt/docs/roswiki/ROS(2f)Concepts.html

Robotics Operating System(ROS)



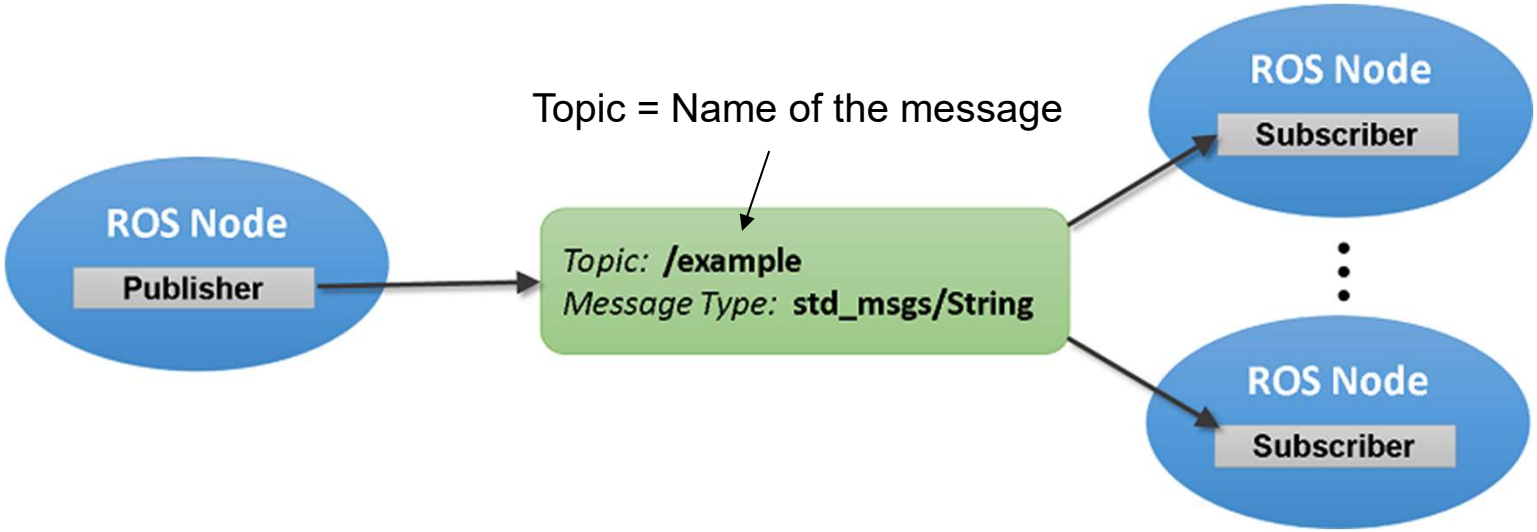
- **ROS Master** stores topics registration information for ROS nodes. Nodes communicate with the Master to report their registration information.
- **Nodes** connect to other nodes directly; the Master only provides lookup information, much like a DNS server.

※ **registration info** : node name, msg info to publish, msg info to subscribe, …

Reference : http://www.clearpathrobotics.com/assets/guides/kinetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

Robotics Operating System(ROS)

- **Message, Topic**
  - Messages are routed via a transport system with **publish** / **subscribe** semantics.
  - In message type, standard primitive types (Integer, Floating point, Boolean, etc.) are supported, as are arrays of primitives types.
  - Messages can be arbitrary nested structures and arrays (much like C structs).

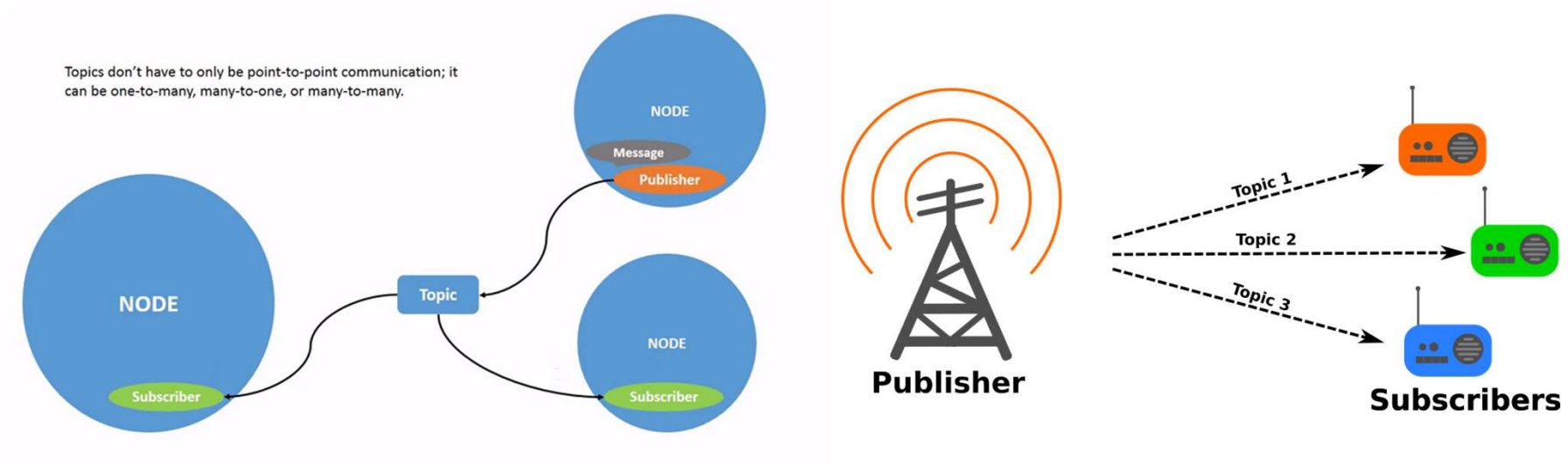


Reference : https://kr.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html

Robotics Operating System(ROS)

- **Publish / Subscribe**
  - A node **sends** out a message by **publishing(=broadcasting)** it to a given topic.
  - A node that is interested in a certain kind of data will **subscribe to(=receive)** the appropriate topic.
  - There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics.

    → It can be **One-to-One**, **One-to-Many**, or **Many-to-Many** communication!

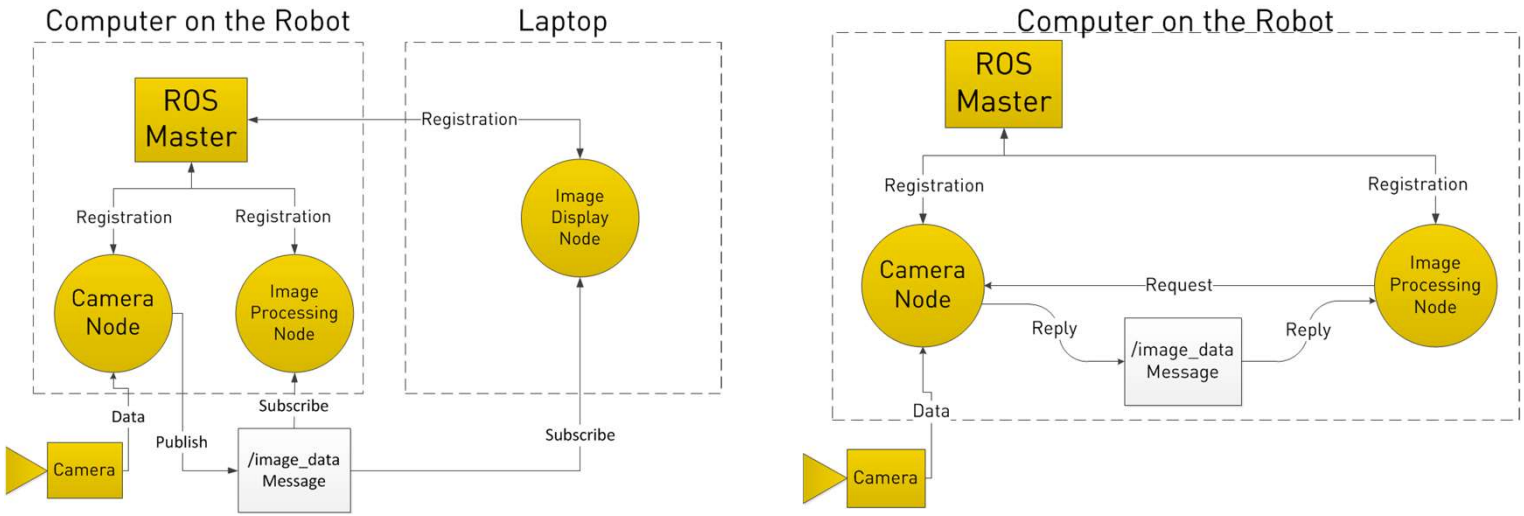


Reference : https://davesroboshack.com/the-robot-operating-system-ros/overview-of-ros/

Robotics Operating System(ROS)

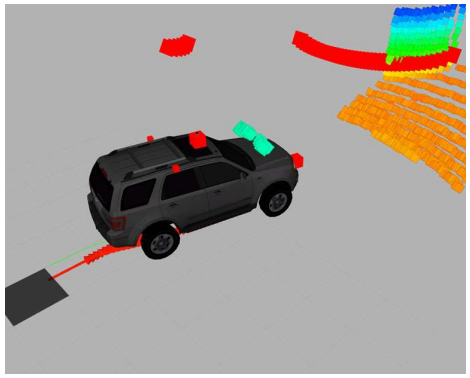
➢ Example: Image processing with Camera node

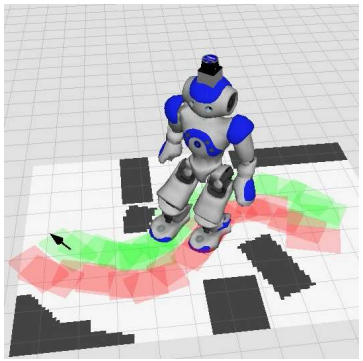
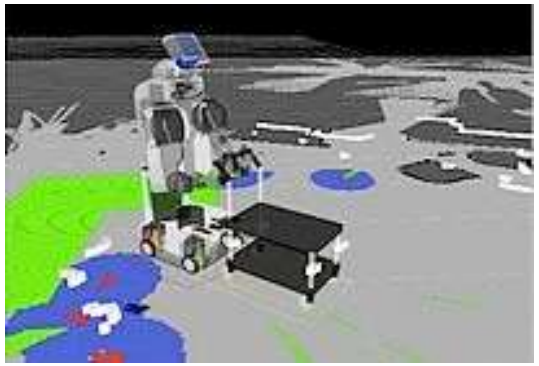
- ❑ There is a **Camera Node** to control a camera and receive image streaming data.
- ❑ There is an **Image Processing Node** to process images from the camera.
- ❑ Both nodes are registered to **ROS Master node** and data communication is established between them.
- ❑ Camera Node **publishes** **image data** by the **topic name** **'/image_data',** and the Image Processing Node **subscribes** **the topic message** whose **topic name** **is '/image_data'** to process the image.
- ❑ Another Node can be added to display the image data by subscribing the topic '/image_data' at the Image Display Node.

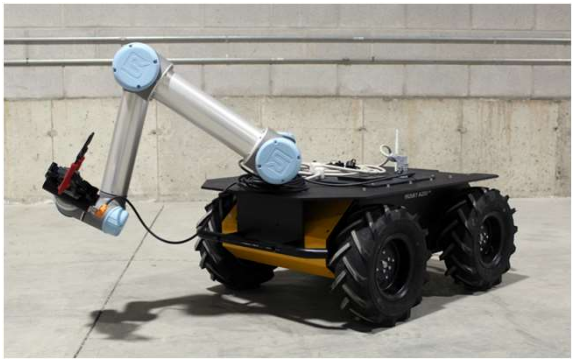


Reference : http://www.clearpathrobotics.com/assets/guides/kinetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

Applications of ROS






Applications of ROS

ROS Application for an Autonomous Delivery Robot [1]

ROS Application for a Drone Challenge [2]

[1] : https://www.youtube.com/watch?v=YmQ2W5y8cwU
[2] : https://www.youtube.com/watch?v=9UYjUIWnzwQ

Applications of ROS

Example of ROS Application for Urban Autonomous Driving [1]

[1] : https://www.youtube.com/watch?v=qfV3RK9moy4&t=6s

ROS Tutorial

1. Installation & Configuration of ROS
2. ROS Package Tutorial
3. ROS Topic
4. ROS Commands

Installation & Configuration of ROS

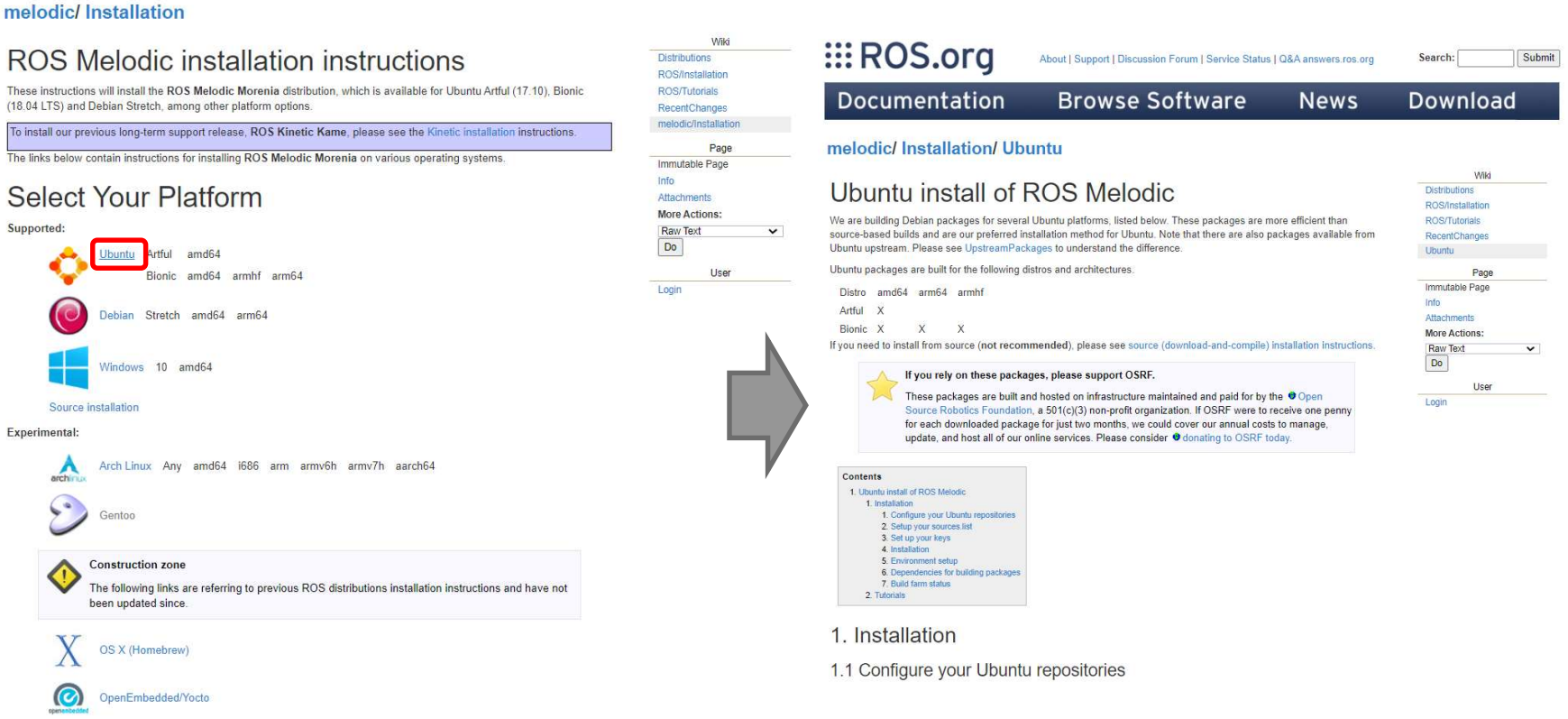
- Simply follow the homepage installation tutorial.
- Install homepage: http://wiki.ros.org/ROS/Installation.

ROS Kinetic (for Ubuntu 16.04)

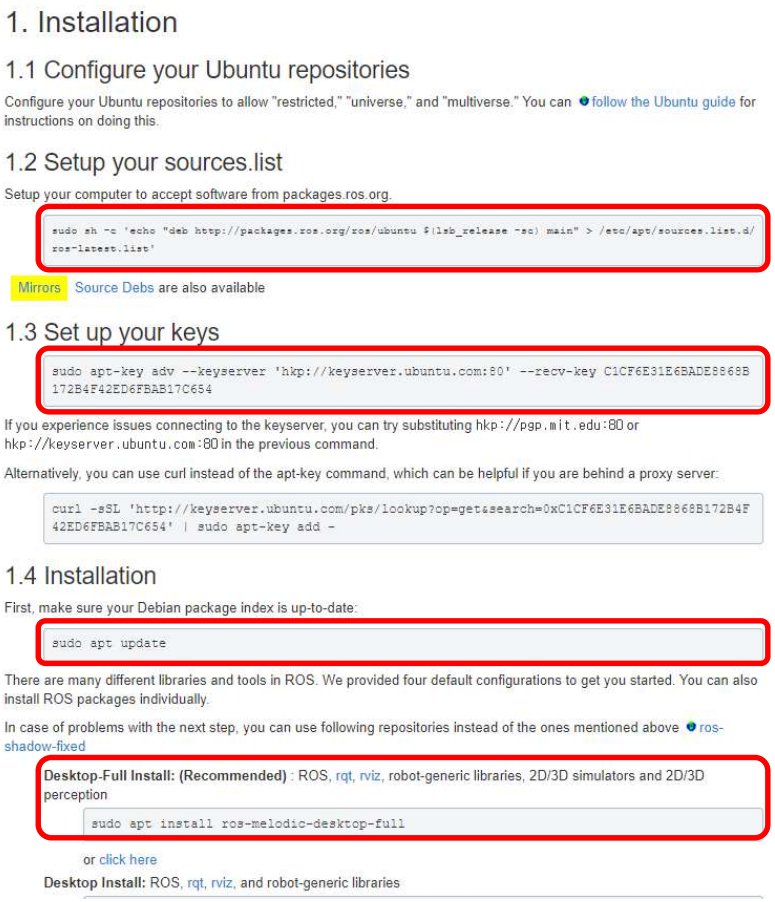
ROS Melodic (for Ubuntu 18.04)

Installation & Configuration of ROS

- Simply follow the homepage installation tutorial.
- Install homepage: http://wiki.ros.org/ROS/Installation.



Installation & Configuration of ROS



1. Installation

1.1 Configure your Ubuntu repositories

Configure your Ubuntu repositories to allow "restricted," "universe," and "multiverse." You can follow the Ubuntu guide for instructions on doing this.

1.2 Setup your sources.list

Setup your computer to accept software from packages.ros.org.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Mirrors Source Debs are also available

1.3 Set up your keys

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

If you experience issues connecting to the keyserver, you can try substituting hkp://pgp.mit.edu:80 or hkp://keyserver.ubuntu.com:80 in the previous command.

Alternatively, you can use curl instead of the apt-key command, which can be helpful if you are behind a proxy server:

```
curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654' | sudo apt-key add -
```

1.4 Installation

First, make sure your Debian package index is up-to-date:

```
sudo apt update
```

There are many different libraries and tools in ROS. We provided four default configurations to get you started. You can also install ROS packages individually.

In case of problems with the next step, you can use following repositories instead of the ones mentioned above ros-shadow-fixed

Desktop-Full Install: (Recommended) : ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators and 2D/3D perception

```
sudo apt install ros-melodic-desktop-full
```

or click here

Desktop Install: ROS, rqt, rviz, and robot-generic libraries

Reference :
http://wiki.ros.org/melodic/Installation/Ubuntu

- **Setup your sources.list**

  sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

- **Set up your keys**

  sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

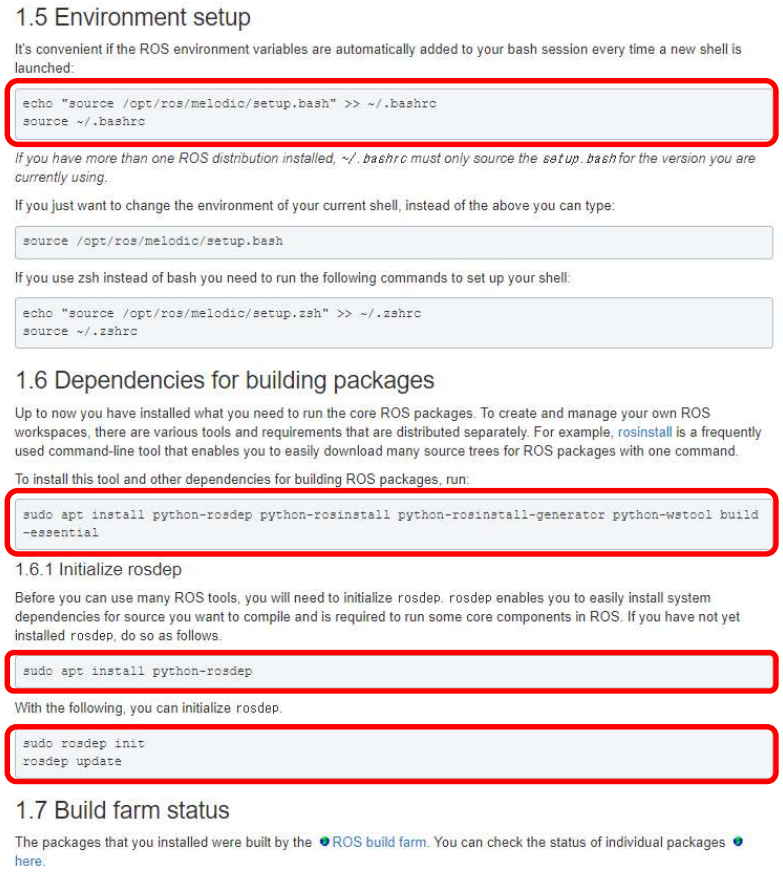
- **Update your Ubuntu package index**

  sudo apt update

- **Desktop-Full install**

  sudo apt install ros-melodic-desktop-full

Installation & Configuration of ROS



1.5 Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

If you have more than one ROS distribution installed, ~/.bashrc must only source the setup.bash for the version you are currently using.

If you just want to change the environment of your current shell, instead of the above you can type:

```
source /opt/ros/melodic/setup.bash
```

If you use zsh instead of bash you need to run the following commands to set up your shell:

```
echo "source /opt/ros/melodic/setup.zsh" >> ~/.zshrc
source ~/.zshrc
```

1.6 Dependencies for building packages

Up to now you have installed what you need to run the core ROS packages. To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, rosinstall is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command.

To install this tool and other dependencies for building ROS packages, run:

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build
-essential
```

1.6.1 Initialize rosdep

Before you can use many ROS tools, you will need to initialize `rosdep`. `rosdep` enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS. If you have not yet installed `rosdep`, do so as follows.

```
sudo apt install python-rosdep
```

With the following, you can initialize `rosdep`.

```
sudo rosdep init
rosdep update
```

1.7 Build farm status

The packages that you installed were built by the ROS build farm. You can check the status of individual packages here.

Reference :
http://wiki.ros.org/melodic/Installation/Ubuntu

- **Environment setup**

  echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
  source ~/.bashrc

  You can check the above added line as below

  gedit ~/.bashrc

  ※ '~/.bashrc' is same with '/home/user/.bashrc'

- **Install dependencies for building packages**

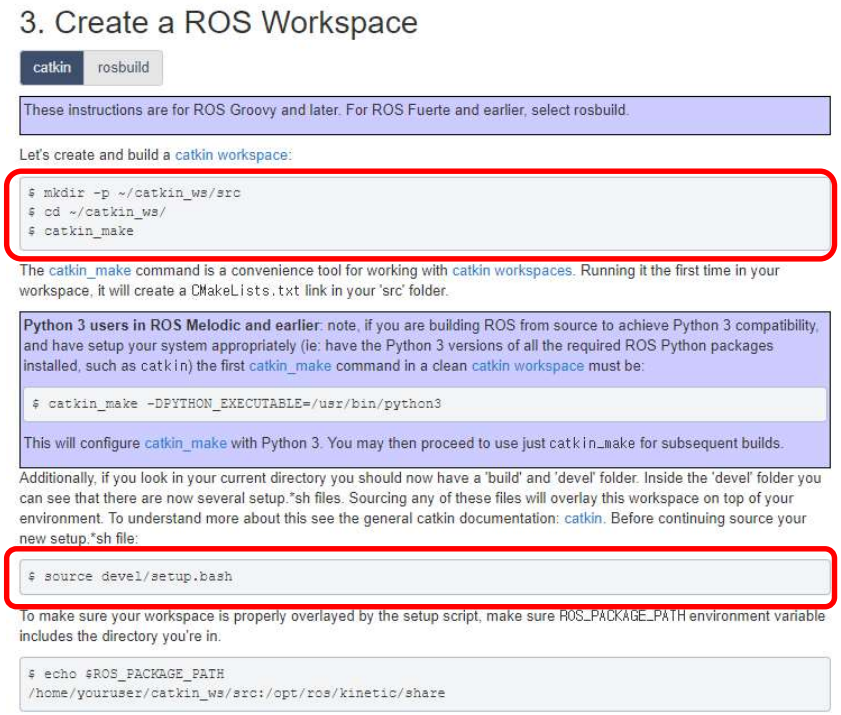
  sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential

- **Initialize rosdep**

  sudo apt install python-rosdep

  sudo rosdep init
  rosdep update

Installation & Configuration of ROS



3. Create a ROS Workspace

catkin | rosbuild

These instructions are for ROS Groovy and later. For ROS Fuerte and earlier, select rosbuild.

Let's create and build a catkin workspace:

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
```

The catkin_make command is a convenience tool for working with catkin workspaces. Running it the first time in your workspace, it will create a CMakeLists.txt link in your 'src' folder.

Python 3 users in ROS Melodic and earlier: note, if you are building ROS from source to achieve Python 3 compatibility, and have setup your system appropriately (ie: have the Python 3 versions of all the required ROS Python packages installed, such as catkin) the first catkin_make command in a clean catkin workspace must be:

```
$ catkin_make -DPYTHON_EXECUTABLE=/usr/bin/python3
```

This will configure catkin_make with Python 3. You may then proceed to use just catkin_make for subsequent builds.

Additionally, if you look in your current directory you should now have a 'build' and 'devel' folder. Inside the 'devel' folder you can see that there are now several setup.*sh files. Sourcing any of these files will overlay this workspace on top of your environment. To understand more about this see the general catkin documentation: catkin. Before continuing source your new setup.*sh file:

```
$ source devel/setup.bash
```

To make sure your workspace is properly overlayed by the setup script, make sure ROS_PACKAGE_PATH environment variable includes the directory you're in.

```
$ echo $ROS_PACKAGE_PATH
/home/youruser/catkin_ws/src:/opt/ros/kinetic/share
```

Reference :
http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

- **Create and build a catkin workspace (~/catkin_ws)**

cd ~ "go to home directory"
mkdir –p ~/catkin_ws/src "make a workspace (catkin_ws) and source (src) directory "

cd ~/catkin_ws "go back to workspace"
catkin_make "do catkin_make"

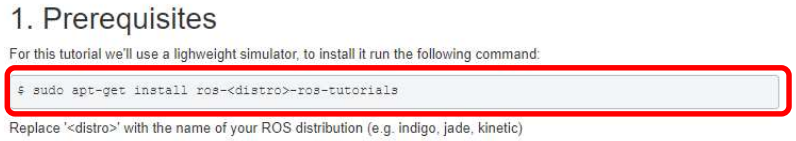
※ 'catkin_make' command should be executed in workspace (~/catkin_ws), not src directory.

- **Sourcing the files in devel after catkin_make process**

source devel/setup.bash

※ Usually, people add above line in '~/.bashrc' so that they don't need do 'sourec devel/setup.bash' every catkin_make process.
Open the bashrc file (gedit ~/.bashrc) and add the line (source devel/setup.bash) in it.

ROS Package Tutorial – TurtleSim


1. Prerequisites

For this tutorial we'll use a lighweight simulator, to install it run the following command:

```
$ sudo apt-get install ros-<distro>-ros-tutorials
```

Replace '<distro>' with the name of your ROS distribution (e.g. indigo, jade, kinetic)

Reference :
http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

- **Prerequisites**

  sudo apt-get install ros-melodic-ros-tutorials "for Ubuntu 18.04"

  or

  sudo apt-get install ros-kinetic-ros-tutorials "for Ubuntu 16.04"

- **Run roscore (in terminal 1)**

  roscore

- **Run tutorial ros package (in another terminal 2)**

  rosrun turtlesim turtlesim_node

  ※ rosrun [package_name] [node_name]

ROS Package Tutorial – TurtleSim

1. Setup

1.1 roscore

Let's start by making sure that we have roscore running, **in a new terminal**:

```
$ roscore
```

If you left roscore running from the last tutorial, you may get the error message:

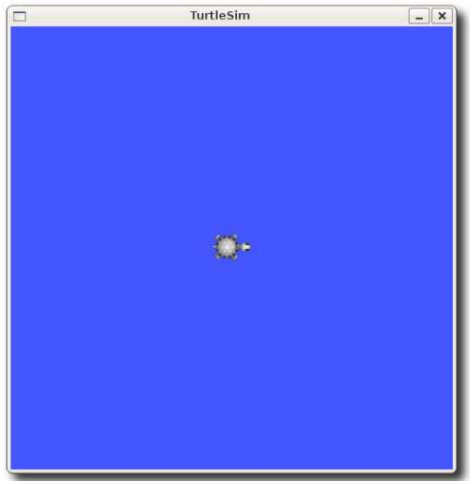
```
roscore cannot run as another roscore/master is already running.
Please kill other roscore/master processes before relaunching
```

This is fine. Only one roscore needs to be running.

1.2 turtlesim

For this tutorial we will also use turtlesim. Please run **in a new terminal**:

```
$ rosrun turtlesim turtlesim_node
```



Reference :
http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

➢ **Prerequisites**

sudo apt-get install ros-melodic-ros-tutorials "for Ubuntu 18.04"

or

sudo apt-get install ros-kinetic-ros-tutorials "for Ubuntu 16.04"

➢ **Run roscore (in terminal 1)**

roscore

➢ **Run tutorial ros package (in another terminal 2)**

rosrun turtlesim turtlesim_node

※ rosrun [package_name] [node_name]

ROS Package Tutorial – TurtleSim

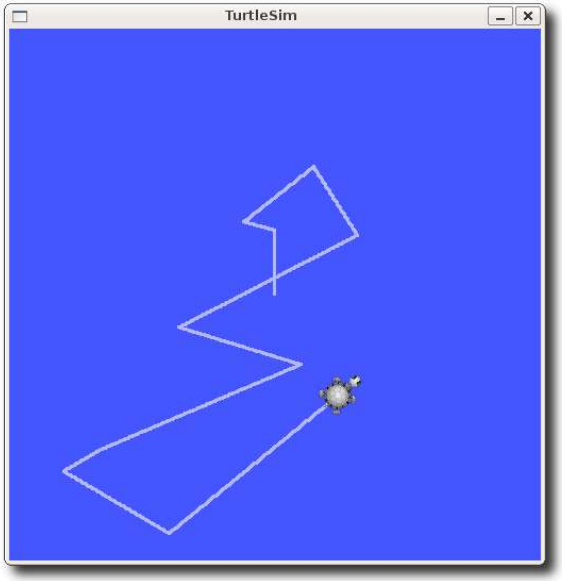
1.3 turtle keyboard teleoperation

We'll also need something to drive the turtle around with. Please run **in a new terminal**:

```
$ rosrun turtlesim turtle_teleop_key
```

```
[ INFO] 1254264546.878445000: Started node [/teleop_turtle], pid [5528], bound on [aqy], xm
lrpc port [43918], tcpros port [55936], logging to [~/ros/ros/log/teleop_turtle_5528.log],
using [real] time
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

Now you can use the arrow keys of the keyboard to drive the turtle around. If you can not drive the turtle **select the terminal window of the turtle_teleop_key** to make sure that the keys that you type are recorded.



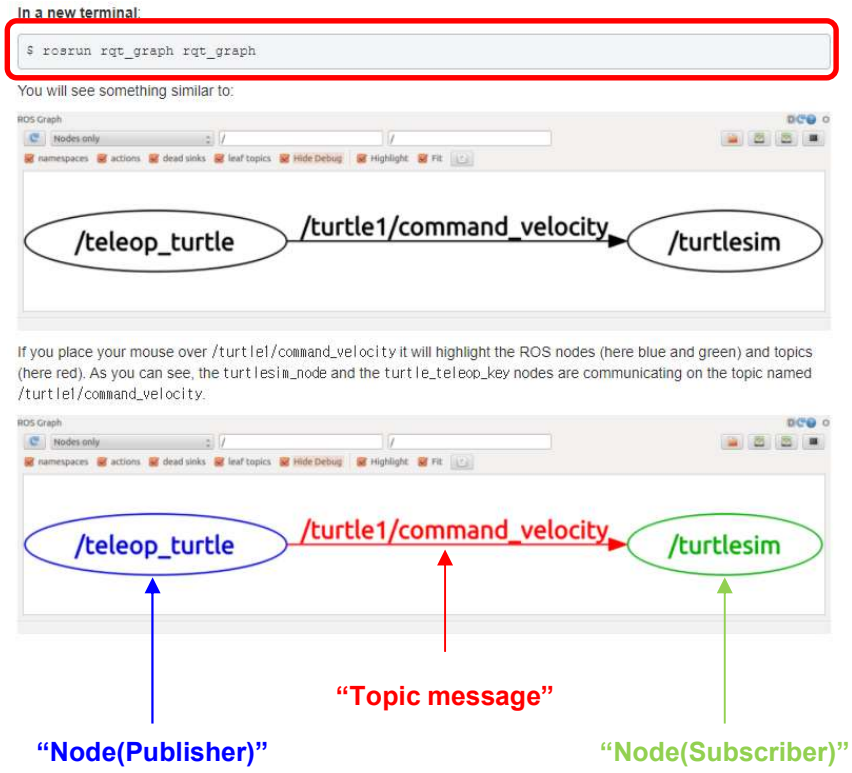
Reference :
http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

- **Run turtle keyboard package (in a new terminal 3)**

  rosrun turtlesim turtle_teleop_key

  You can use the arrow keys of keyboard to drive the turtle around!

ROS Topic



Reference :
http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

- **Run rqt_graph for checking the nodes and topics currently running (in a new terminal 4)**

  rosrun rqt_graph rqt_graph

  or simply

  rqt_graph

- **You can see the name of the publisher node (/teleop_turtle) and subscriber node (/turtlesim)**

- **The topic message name (/turtle1/command_velocity) is represented as an arrow from the publisher to subscriber**

ROS Topic

2.2 Introducing rostopic

The `rostopic` tool allows you to get information about ROS **topics**.

You can use the help option to get the available sub-commands for `rostopic`

```
$ rostopic -h
```

```
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
```

Or pressing tab key after `rostopic` prints the possible sub-commands:

```
$ rostopic
bw    echo  find  hz    info  list  pub   type
```

Let's use some of these topic sub-commands to examine turtlesim.

2.3 Using rostopic echo

`rostopic echo` shows the data published on a topic.

Usage:

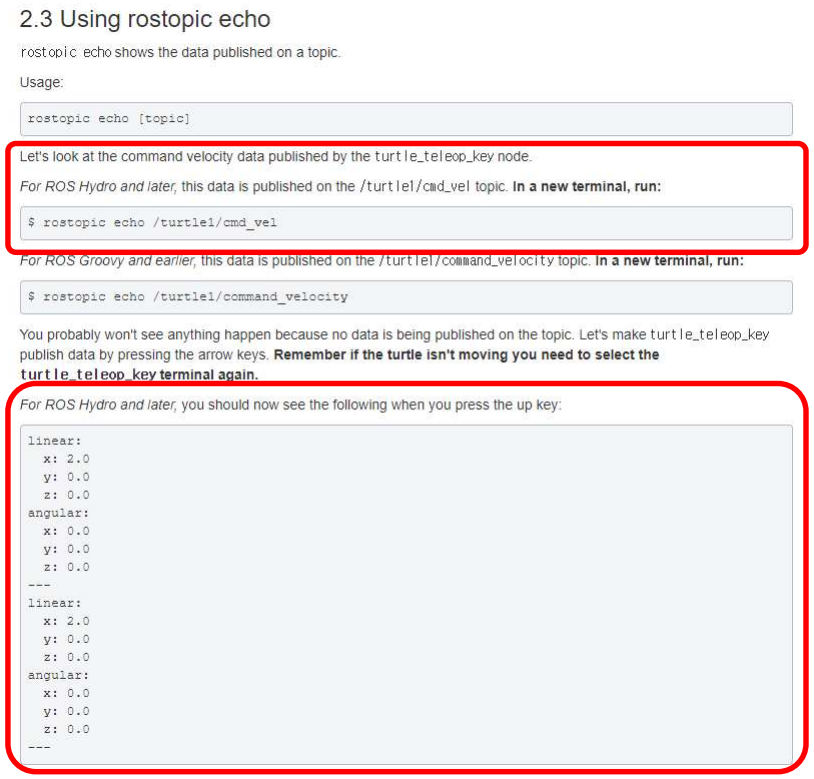
```
rostopic echo [topic]
```

Reference :
http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

- **"rostopic" allows you to get information about ROS topics**
  - rostopic list : print the list of active topics
  - rostopic echo : print messages of specific topic
  - rostopic pub : publish message data to topic
  - rostopic type : print message type of specific topic
  - rostopic hz : display publishing rate of topic

※ In order of frequent use

ROS Topic



2.3 Using rostopic echo

rostopic echo shows the data published on a topic.

Usage:

```
rostopic echo [topic]
```

Let's look at the command velocity data published by the turtle_teleop_key node.

For ROS Hydro and later, this data is published on the /turtle1/cmd_vel topic. **In a new terminal, run:**

```
$ rostopic echo /turtle1/cmd_vel
```

For ROS Groovy and earlier, this data is published on the /turtle1/command_velocity topic. **In a new terminal, run:**

```
$ rostopic echo /turtle1/command_velocity
```

You probably won't see anything happen because no data is being published on the topic. Let's make turtle_teleop_key publish data by pressing the arrow keys. **Remember if the turtle isn't moving you need to select the turtle_teleop_key terminal again.**

For ROS Hydro and later, you should now see the following when you press the up key:

```
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

Reference :
http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

- **Use "rostopic echo" for displaying the command velocity topic from the turtle keyboard package, turtle_teleop_key.**

  rostopic echo /turtle1/cmd_vel

  ※ rostopic echo {topic_name}

- **When you press the arrow keys, you will see the message published from the turtle_teleop_key package.**

  linear:
  x: 2.0
  y: 0.0
  z: 0.0
  angular:
  x: 0.0
  y: 0.0
  z: 0.0

ROS Commands

➢ **roscore :** run the ROS Master Node.

➢ **rosrun [package_name] [node_name] :** run a ROS package node.

- rosrun turtlesim turtle_sim_node
- rosrun turtlesim turtle_teleop_key

✓ **Do not need to memorize commands.**
✓ **Just use double 'Tap key' to display available commands or auto complete.**

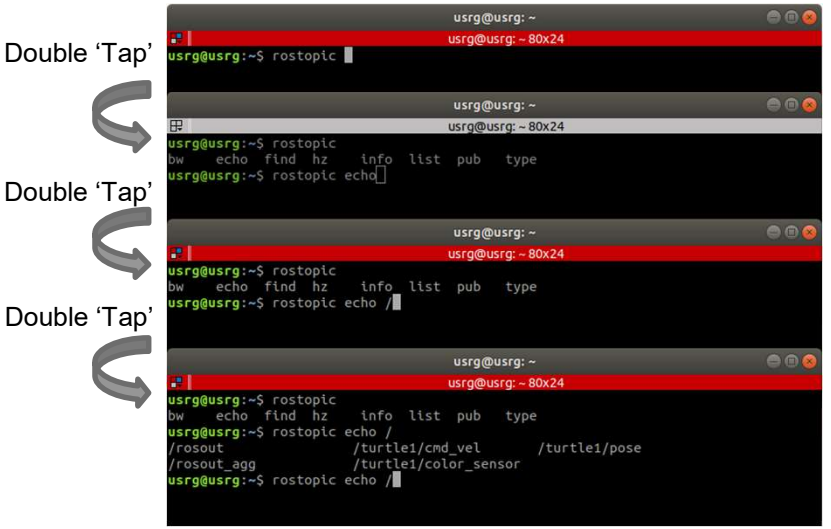
➢ **rostopic :** allows you to get information about ROS topics.

- **rostopic -h** : display the sub-commands for 'rostopic'.
  (bw, echo, find, hz, info, list, pub, type)
- **rostopic list** : print information about active topics.
- **rostopic echo [topic_name]** : print messages to screen.
- **rostopic pub [topic_name] [message_type] [data]**
  : publish data to topic.

Double 'Tap'

Double 'Tap'

Double 'Tap'

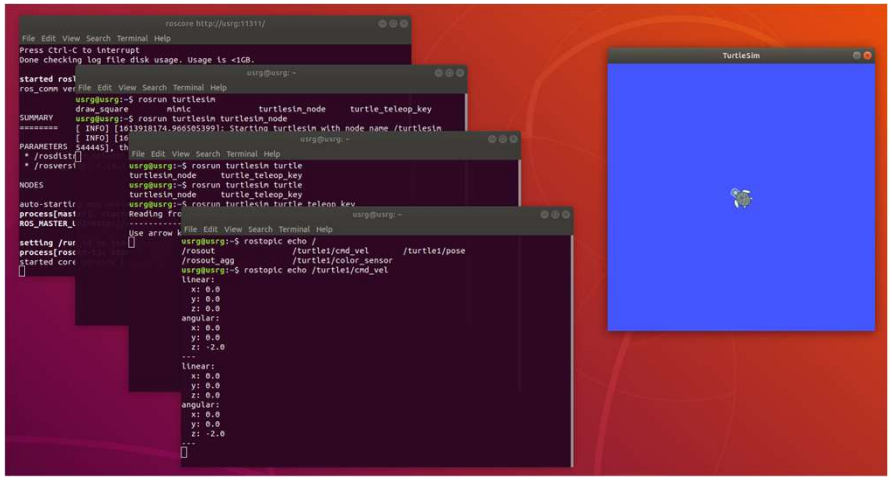


Tips – Terminator

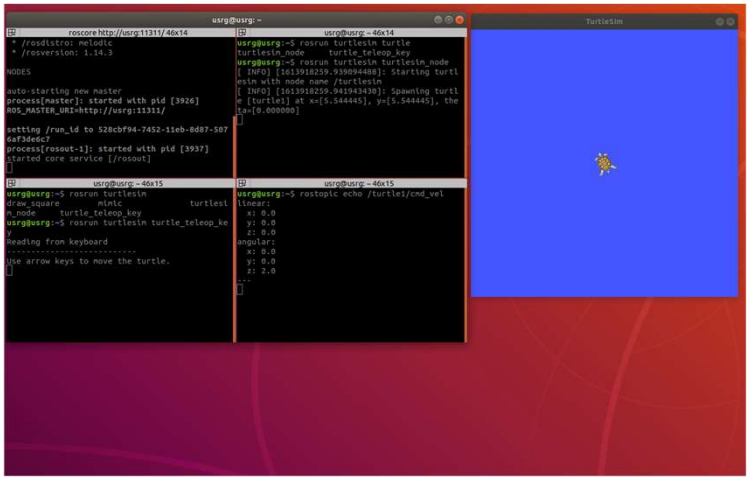
➢ **Recommend Terminal Program in Ubuntu: Terminator**

- In many case, programmer needs to **process multiple programs** in Ubuntu.
- Opening new terminal window whenever you need to run new one is inconvenient.
- Terminator has a function to **split a terminal window so that we can run multiple programs in one window**.
- **Installation**:

  sudo apt-get install terminator



Default Terminal Windows



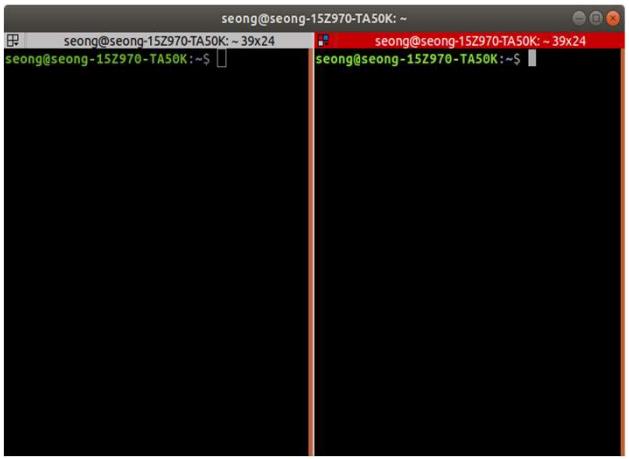
Terminator window

Tips – Terminator

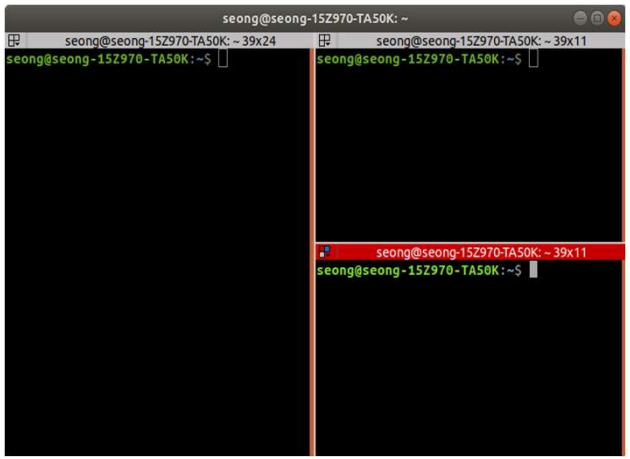
➢ **Recommend Terminal Program in Ubuntu: Terminator**

- **Ctrl + Alt + T :** open a terminal window (same with default terminal of Ubuntu).
- **Ctrl + Shift + E** : split terminal window vertically.
- **Ctrl + Shift + O** : split terminal window horizontally.
- **Ctrl + Shift + T** : add new terminal window tab.

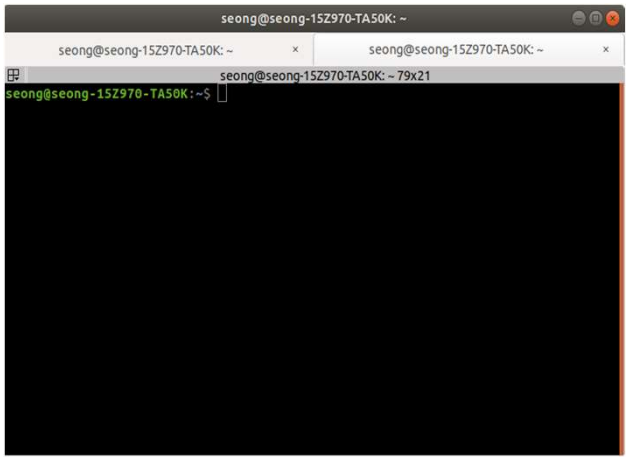
  → Convenient to run and control multiple ros packages in one window!



Ctrl + Shift + E



Ctrl + Shift + O



Ctrl + Shift + T

Experiment Summaries

- Get brief tips on installing Ubuntu(Linux-based OS).
  - Linux installation by Dual Boot is recommended.

- **Understand the Robotics Operating System (ROS).**
  - **Important terminologies : Publish / Subscribe / Topic**

- Install & Setup ROS.
  - Follow the homepage installation tutorial

- Run ROS tutorial.
  - Demonstrate the Turtlesim simulator to be familiar with the ROS system.

Experiment Objectives

Next week, you will do the following:

- Understand how to use the ROS Tools (rviz, rosbag)
- Learn ROS Programming
- Programming Assignment :
  - Programming ROS topic publisher & subscriber.
  - Use ‘roslaunch’ to run your publisher & subscriber node together.

Class Hours (TA Session)
: Online experiment class by TAs:
Wednesday(7~8:30pm)
※ Offline experiment will be determined soon.

Q & A

Email : hynkis@kaist.ac.kr