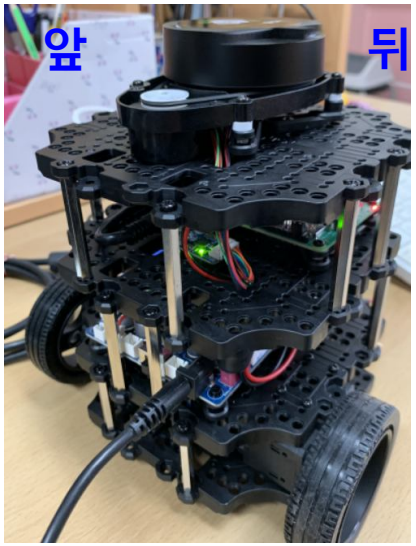
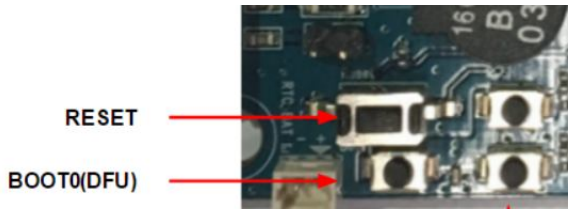


만일 실험을 마치려면, terminal은 CTL+C로 중단 후에 종료

ROS Programming

(file #3 / ?) ver1.0 Noetic

Yongseok Chi

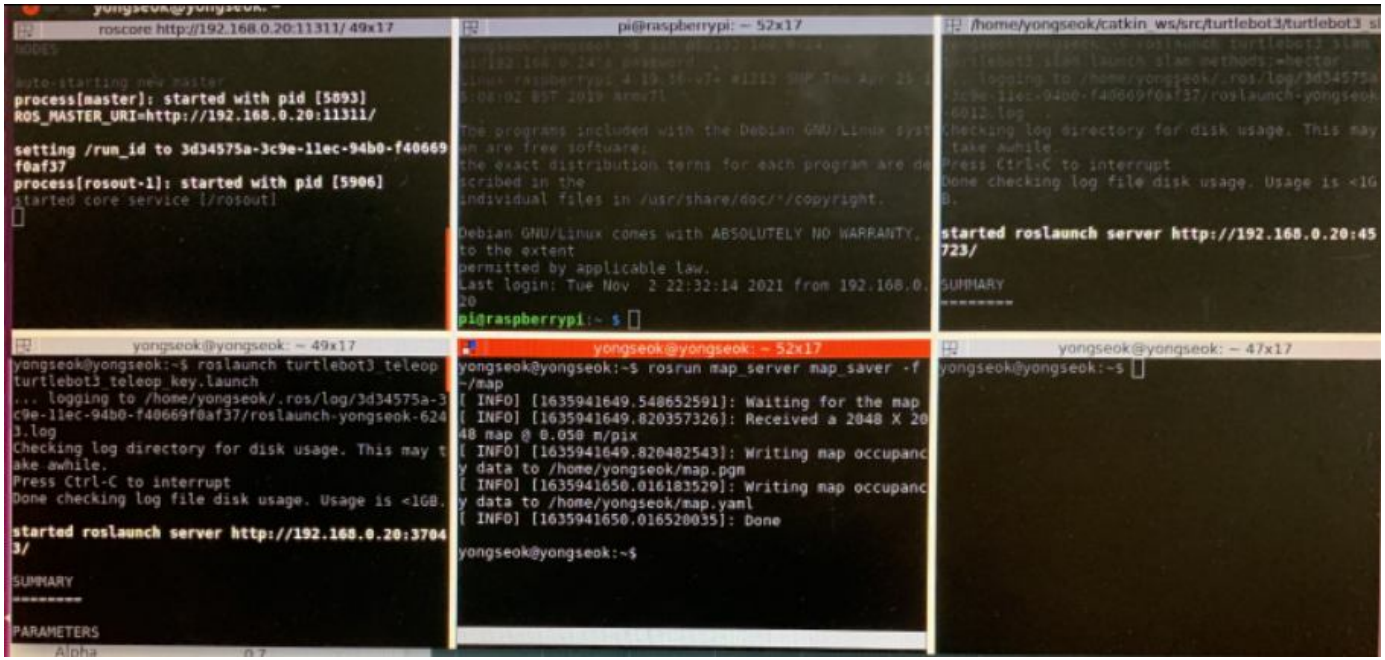


만일 회전 안하면, openCR의 RESET누르고, Ctl+c, 새 터미널
roslaunch turtlebot3_bringup turtlebot3_robot.launch

TIP

새 터미널 열기 Ctrl + Alt + T

* 수평 분할 : Ctrl + Shift + O
* 수직 분할 : : Ctrl + Shift + E
* 다음 창 활성화 : Ctrl + Tab or Ctrl + Shift + N
* 이전 창 활성화 : Ctrl + Shift + Tab or Ctrl + Shift + P
* 현재 활성화 된 창 닫기 : Ctrl + Shift + W
* 터미네이터 실행 : Ctrl + Alt + T
* 터미네이터 종료 : Ctrl + Shift + Q

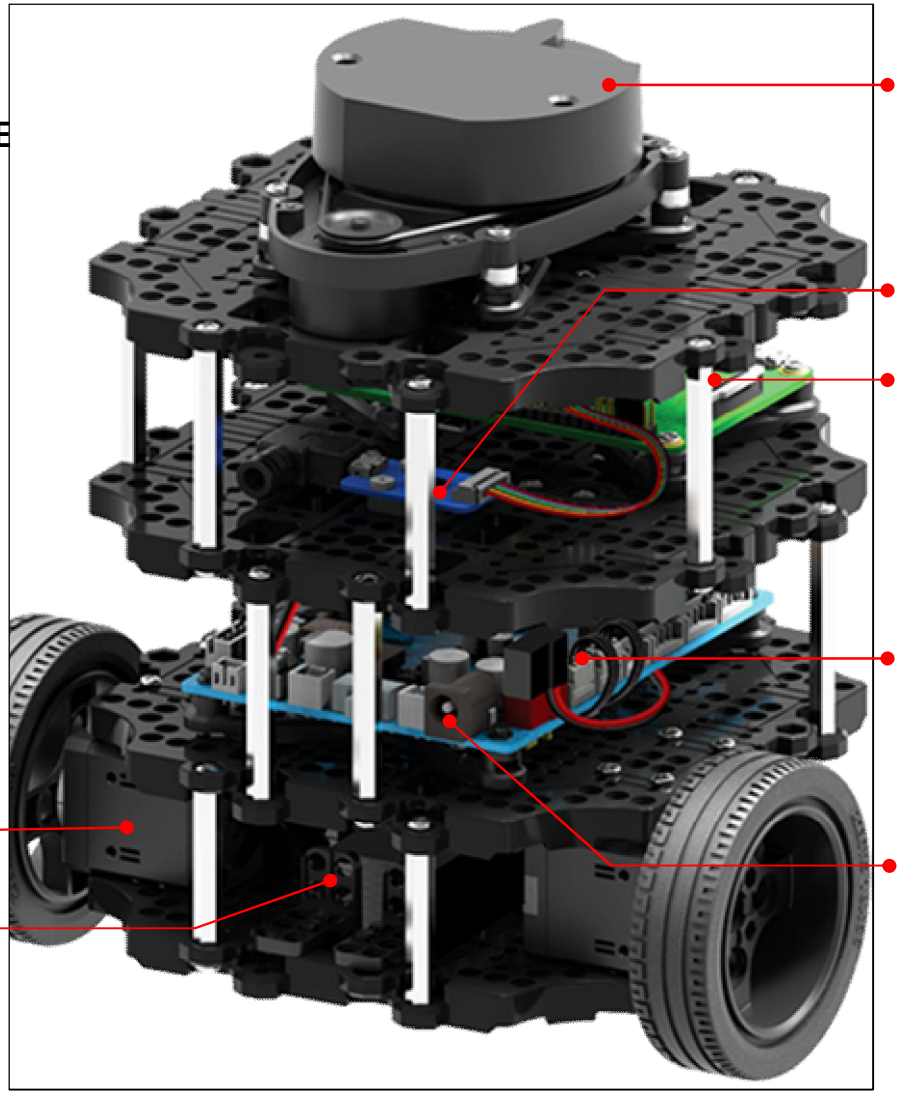


Turtlebot3 burger : Standard Specification

1. Specification

(1) OPEN SOURCE SOFTWARE

(2) OPEN SOURCE HARDWARE



360° LiDAR
: HLS-LFCD LDS(Laser Distance Sensor)
: Detection distance 120mm~3500mm
: Angular Resolution 1 degree

: USB2LDS (115200 baudrate)

SBC(Single Board Computer)
: Raspberry PI 3 B+ / PI 4
- Broadcom BCM2837B0 / 2711
- Cortex-A53 64bits Quad-core
- 1.4GHz frequency

OpenCR(Control module for ROS)
: STM32F746
- ARM Cortex-M7 32bits RISC core
- 216MHz frequency

Input 12V 5A

DYNAMIXEL
: XL430-W350-T
: Stall Torque
- 4.1 [N.m] (at 12.0 [V], 2.3 [A])
: No Load Speed
- 46 [rev/min] (at 12.0 [V])

Li-Po battery 11.1V 1800mAh

Reference books

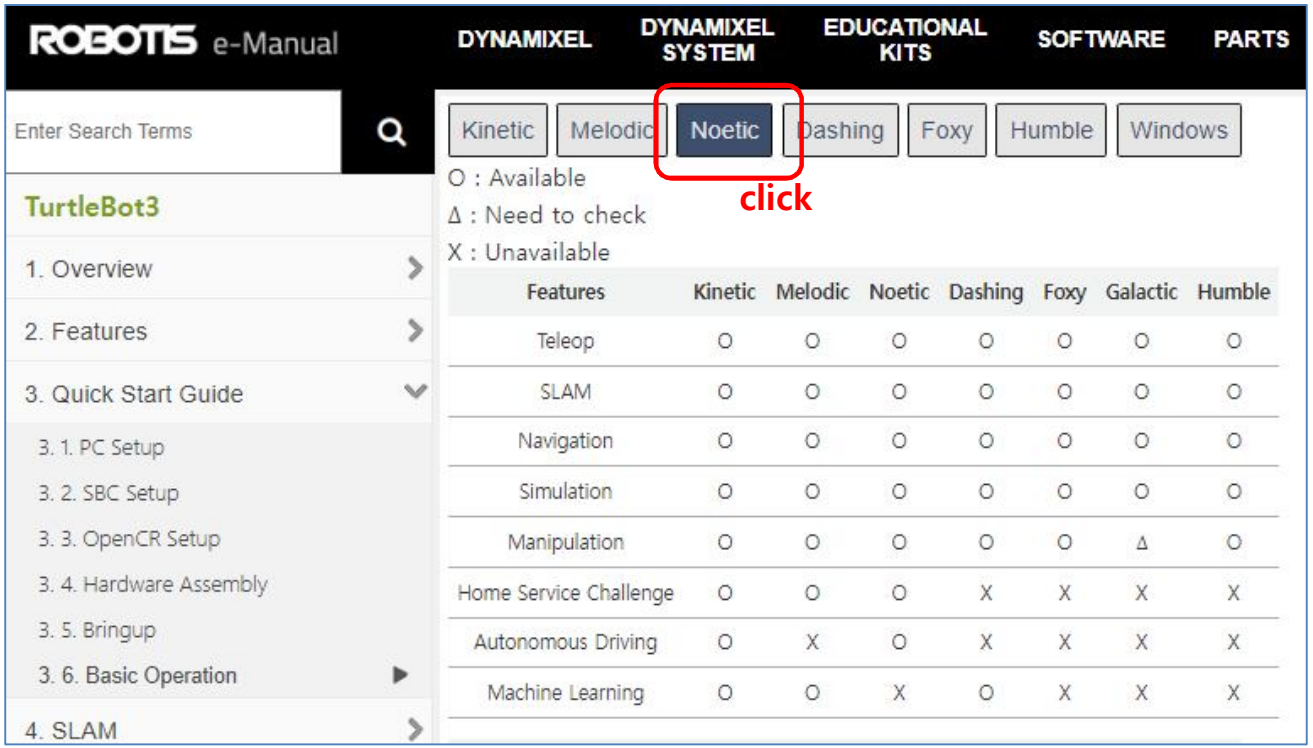
2. Reference books

(1) 주교재

: ROS 로봇 프로그래밍 : Ruby paper 3-1 ROS 설치

: Robotis e-manual 3장

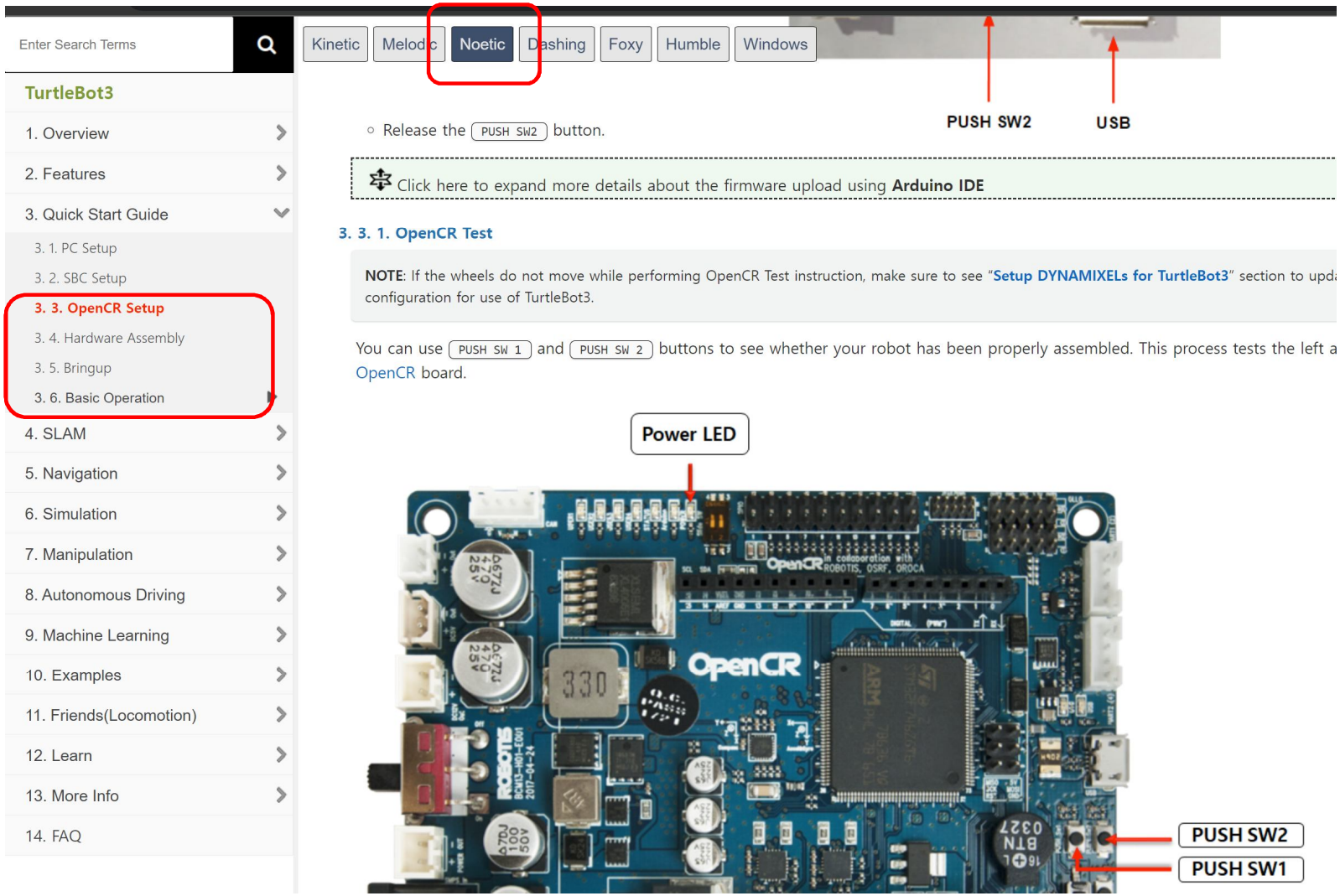
https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/



Features	Kinetic	Melodic	Noetic	Dashing	Foxy	Galactic	Humble
Teleop	O	O	O	O	O	O	O
SLAM	O	O	O	O	O	O	O
Navigation	O	O	O	O	O	O	O
Simulation	O	O	O	O	O	O	O
Manipulation	O	O	O	O	O	Δ	O
Home Service Challenge	O	O	O	X	X	X	X
Autonomous Driving	O	X	O	X	X	X	X
Machine Learning	O	O	X	O	X	X	X

(2) 실험 재료 : Turtlebot3 burger

Reference books



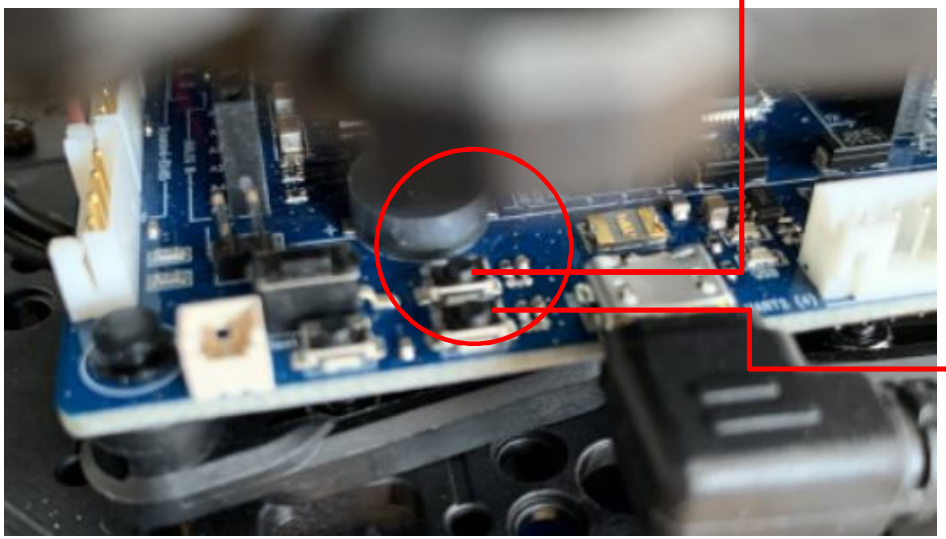
4. ROS 개발환경 구축

OpenCR Test 테스트

: TurtleBot3 burger 조립 후 DYNAMIXEL's configuration 실험하기

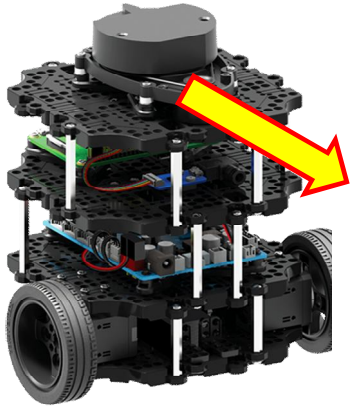
책상 위가 아닌 바닥에서 실험하기

➔ Turtlebot3의 전원케이블 주의



PUSH SW1

몇 초 동안 누르기

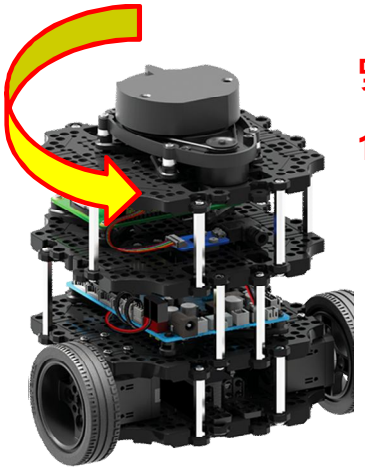


몇 초 동안

수십 30cm 전진

PUSH SW2

몇 초 동안 누르기



몇 초 동안

180도 회전

4. ROS 개발환경 구축

Bringup(3.5.1)

: https://emanual.robotis.com/docs/en/platform/turtlebot3/bringup/#bringup

(3) Bringup TurtleBot3

[Remote PC 에서 SBC를 원격 제어]

새 터미널(CTRL + ALT + T) 열고

$ roscore

새 터미널 (CTRL + ALT + T) 열고

$ ssh ubuntu@192.168.0.21 (for Raspberry)

password 는 turtlebot (암호 입력 안보임, enter)

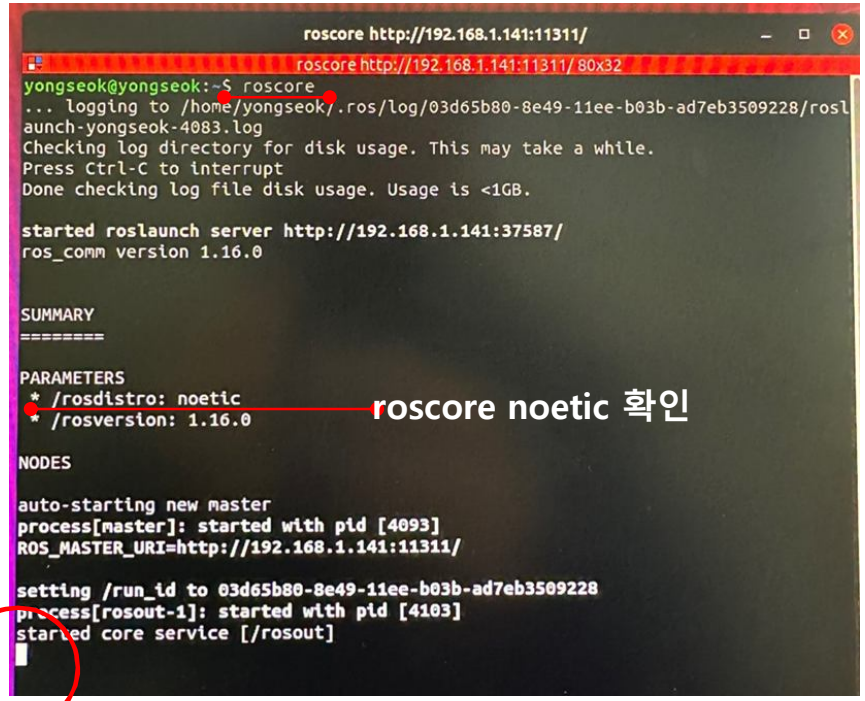
[SBC (TurtleBot burger) 제어]

$ export TURTLEBOT3_MODEL=burger

$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

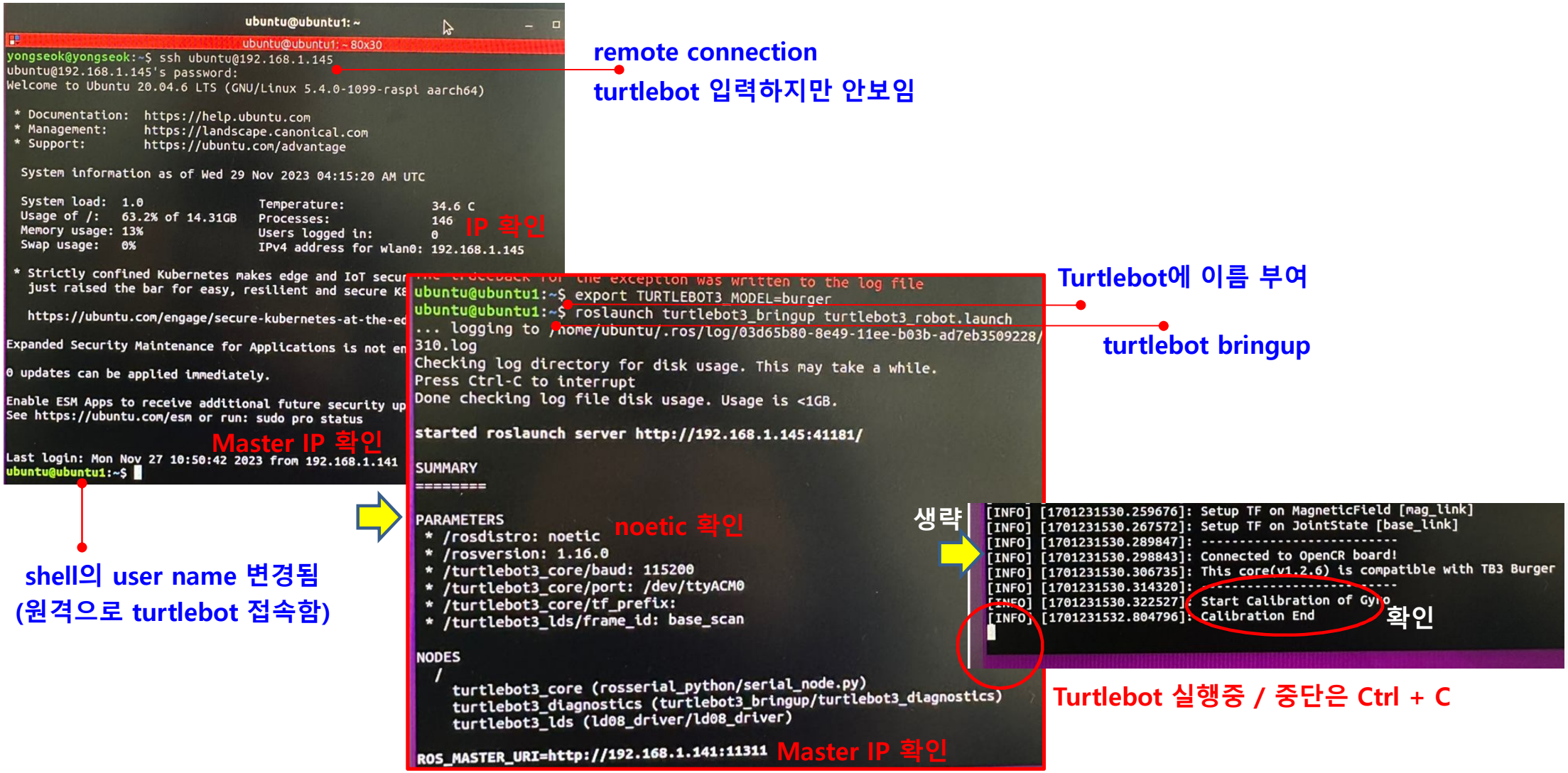
: TurtleBot3 applications 시작하기 위한 기본 package 불러오기

ROS master 실행



ROS master 실행중으로 중단은 Ctrl + C

Turtlebot3 bringup 실험 화면



: https://emanual.robotis.com/docs/en/platform/turtlebot3/basic_operation/#basic-operation [3.6.1.1 teleop_key]

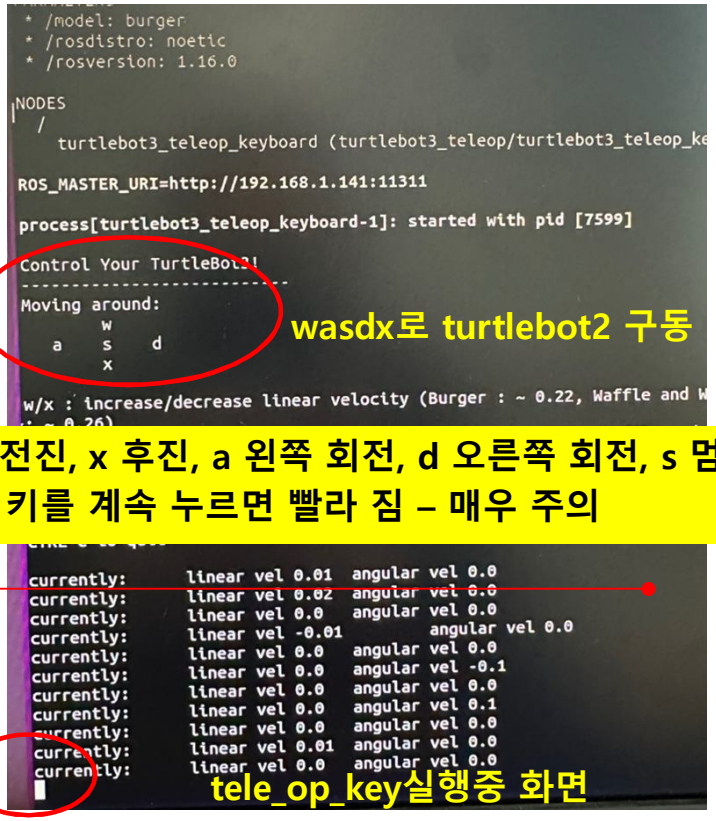
[Basic operation]

[Remote PC 에서] 새 터미널 (CTRL + ALT + T) 열고

$ export TURTLEBOT3_MODEL=burger

➔ 실제 bashrc 에 있기때문에 실행 안해도 됨

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

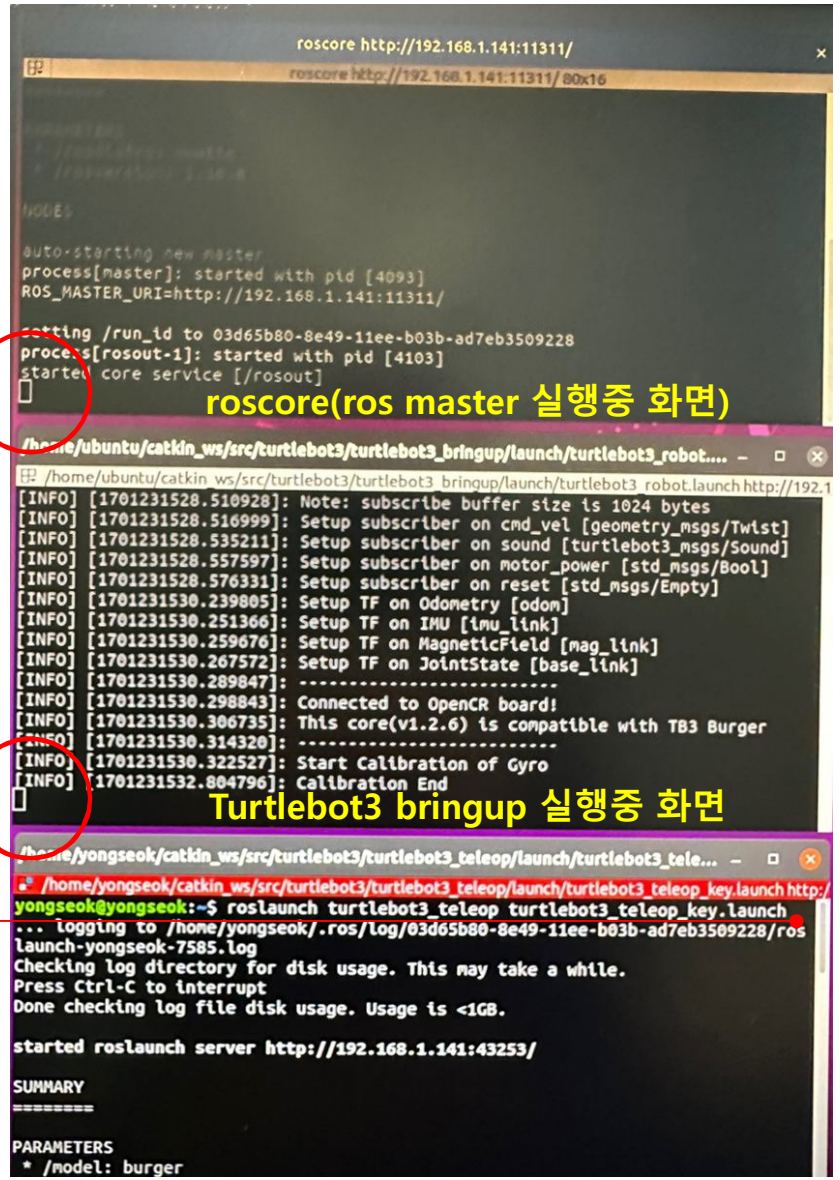


wasdx로 turtlebot2 구동

w 전진, x 후진, a 왼쪽 회전, d 오른쪽 회전, s 멈출
위 키를 계속 누르면 빨라 짐 – 매우 주의

선속도
각속도

tele_op_key실행중 화면



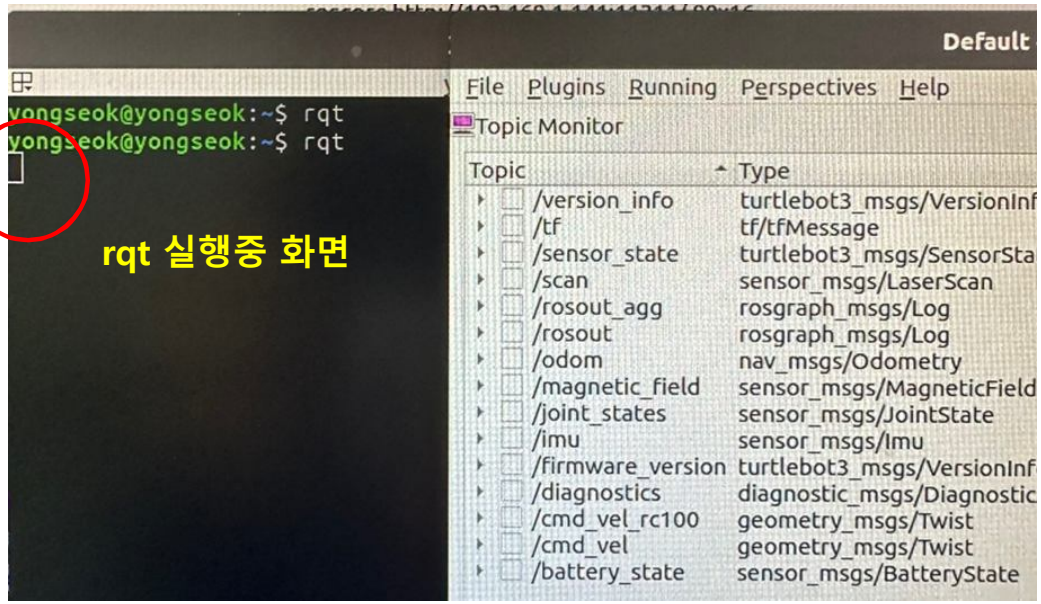
Remote pc에서
Teleop_key 실행화면

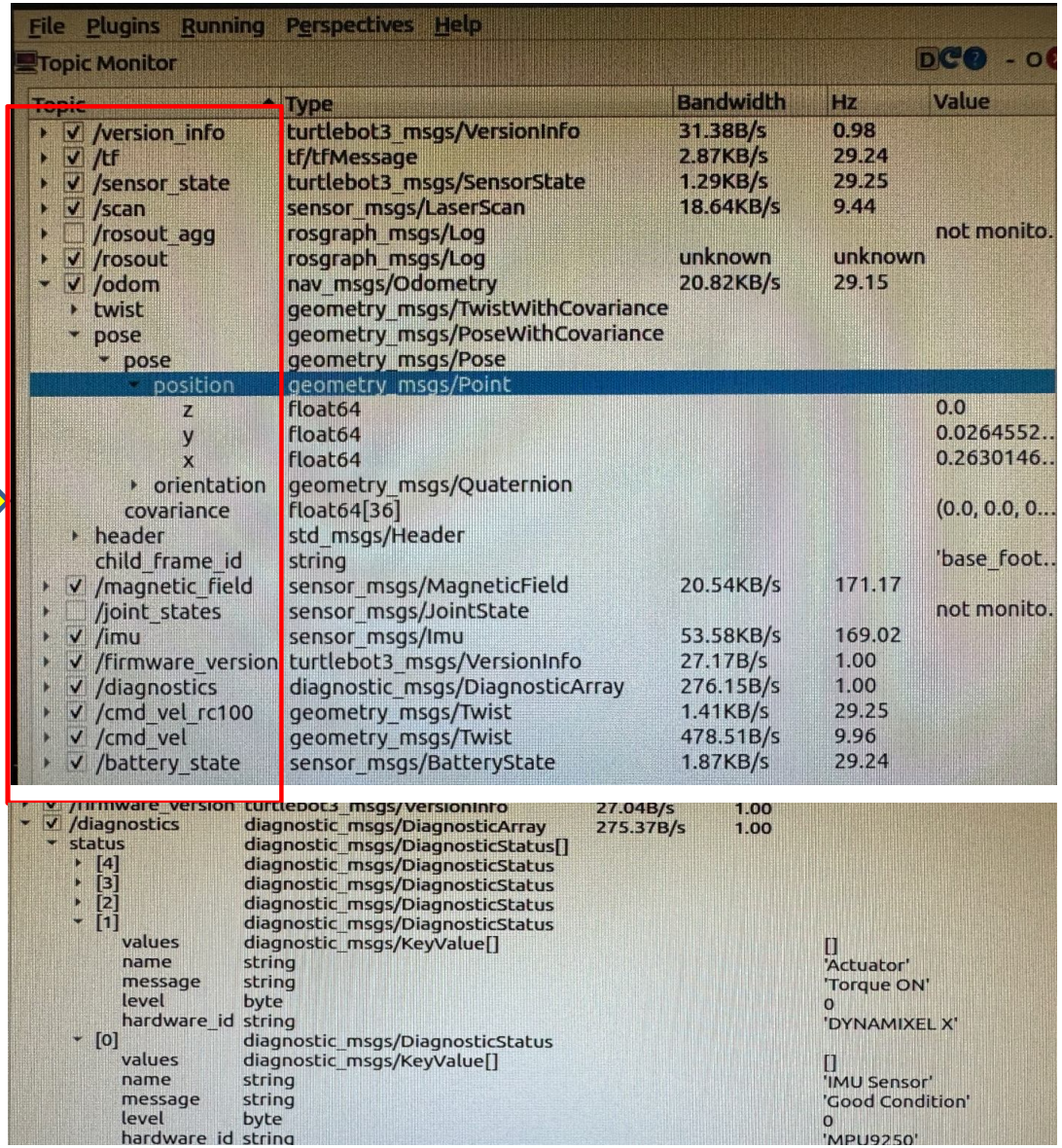
: https://emanual.robotis.com/docs/en/platform/turtlebot3/basic_operation/#basic-operation [3.6.2 Topic monitor]

[rqt]

[Remote PC 에서] 새 터미널 (CTRL + ALT + T) 열고

$ rqt





topic monitor window is not displayed,
select the **plugin** -> **Topics** -> **Topic Monitor**

Click the checkbox next to each topic to monitor the topic

Remote PC

원격제어 방법2

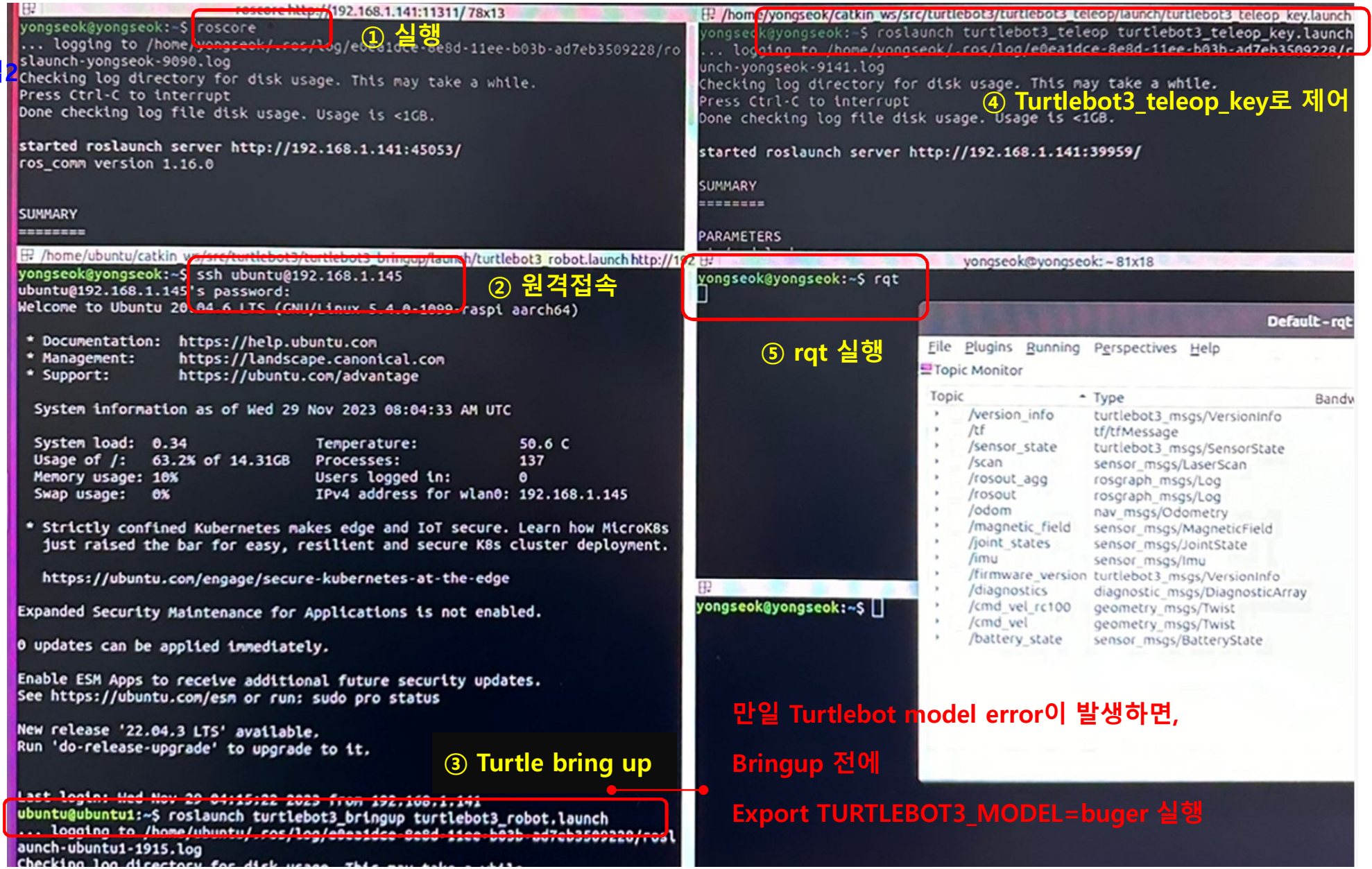
만일

마치려면,

terminal은

CTL+C로

중단후 종료



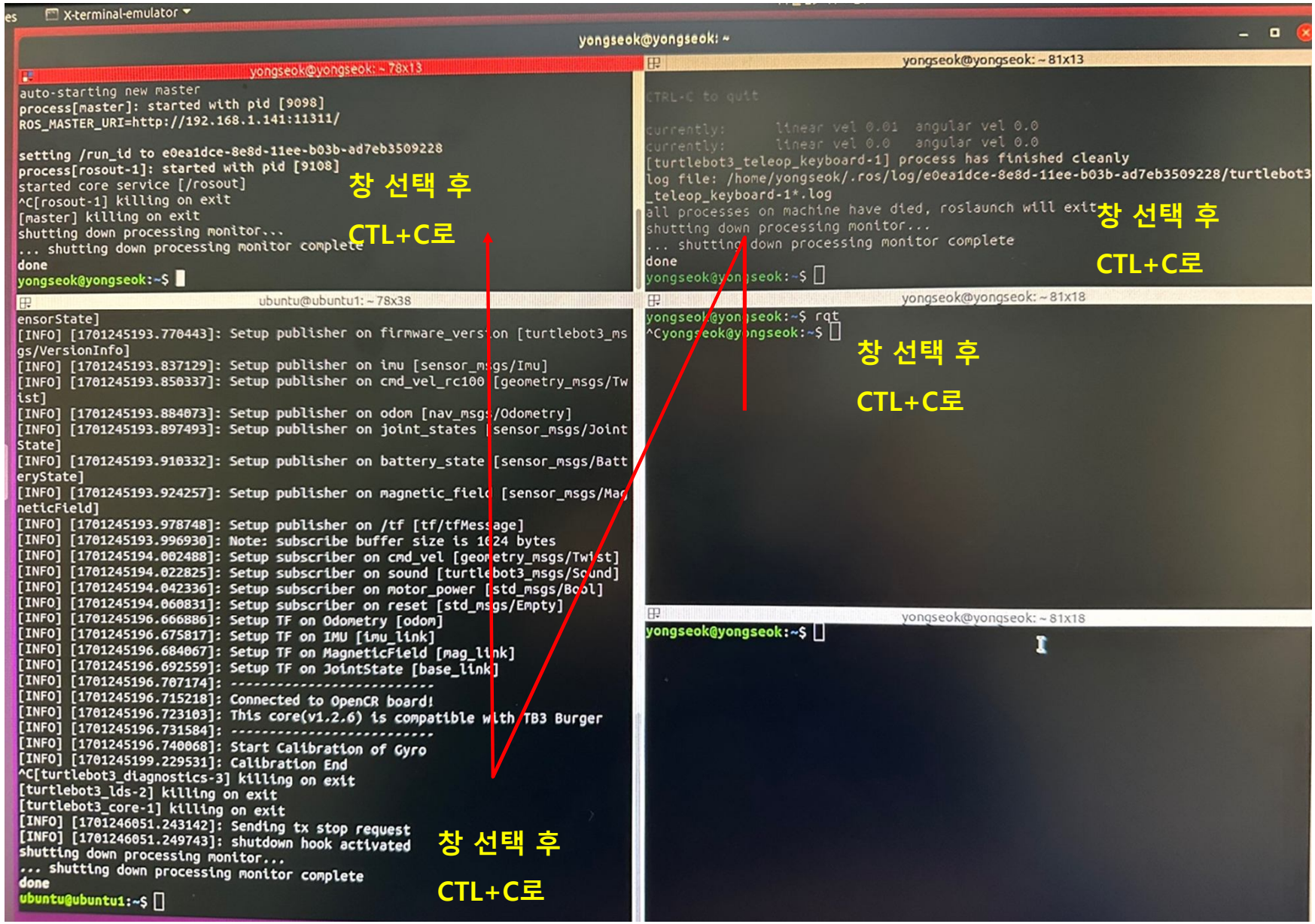
만일

마치려면,

terminal은

CTL+C로

중단후 종료



[SBC (TurtleBot burger) 환경변수에 이름 반영]

: export TURTLEBOT3_MODEL=burger 반영하기

$ nano ~/.bashrc

```
ubuntu@ubuntu1:~$ nano ~/.bashrc
```

```
alias cm='cd ~/catkin_ws && catkin_make'
alias eb='nano ~/.bashrc'
alias nb='nano ~/.bashrc'
alias sb='source ~/.bashrc'

# Replace {IP_ADDRESS_OF_REMOTE_PC} with the IP address of
# Both Remote PC and Raspberry Pi should be connected in t
export ROS_MASTER_URI=http://192.168.1.141:11311
# Replace {IP_ADDRESS_OF_RASPBERRY_PI} with the IP address
export ROS_HOSTNAME=192.168.1.145

export LDS_MODEL=LDS-02
export TURTLEBOT3_MODEL=burger

^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text
^X Exit       ^R Read File   ^\ Replace    ^U Paste Text
```

화살표 키로 맨 밑으로 내려서 아래 작성

export TURTLEBOT3_MODEL=burger

저장 Ctrl +S / 나가기 Ctrl + X

환경변수 반영하기

$ source ~/.bashrc

```
ubuntu@ubuntu1:~$ nano ~/.bashrc
ubuntu@ubuntu1:~$ source ~/.bashrc
ubuntu@ubuntu1:~$
```

Bringup 방법

: Remote PC 원격제어 방법2 (단, Raspberry의 IP 정확해야 함)

Remote PC 새 터미널 열고

$ roscore

만일

마치려면, terminal은 CTL+C로 중단후 종료

Bringup

새 터미널 열고

$ ssh ubuntu@192.168.0.21 (IP is your Raspberry)

password 는 turtlebot (암호 입력 안보임, enter) 연결

~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

$ sudo shutdown now

만일

마치려면, terminal은 CTL+C로 중단후 종료

새 터미널에서 방향키 실행하기

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

만일

마치려면, terminal은 CTL+C로 중단후 종료

새 터미널에서 방향키 실행하기

$ rqt

만일

마치려면, terminal은 CTL+C로 중단후 종료

4. ROS 개발환경 구축

4-2-9. rqt

: GUI 개발에 사용되는 Qt framework 기반의 ROS software framework

: rqt graph 처럼 node간의 관계를 확인할 때 사용

: 3개의 meta package 로 구성하여 복잡한 GUI를 쉽게...

- rqt ➔ rqt 구동 core module
- rqt_common_plugins ➔ Turtlebot 작동 중이거나 정지 중에 사용 가능한 도구 모음
- rqt_robot_plugins ➔ Turtlebot 작동하고 있을 때 로봇과 상호작용하기 위한 도구 모음

[Remote PC 에서 : 강의자료 4-2-8 실행 중에]

: https://emanual.robotis.com/docs/en/platform/turtlebot3/basic_operation/#basic-operation

새 터미널 열기

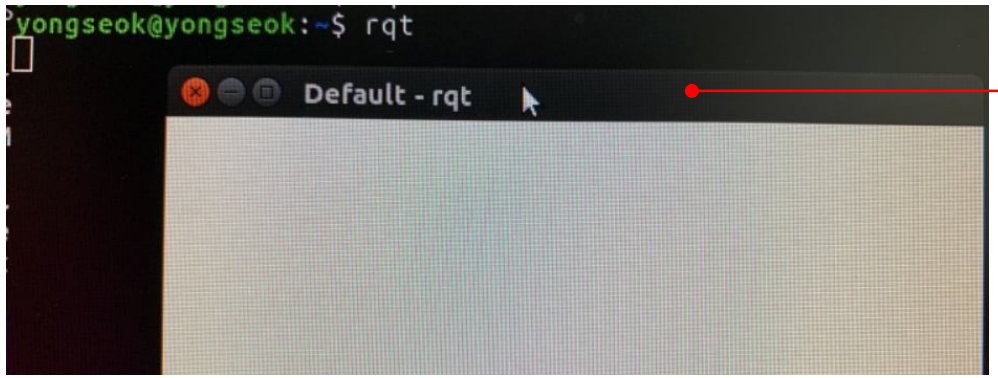
$ roscore

Turtlebot bringup이후에

새 터미널 열기

$ rqt

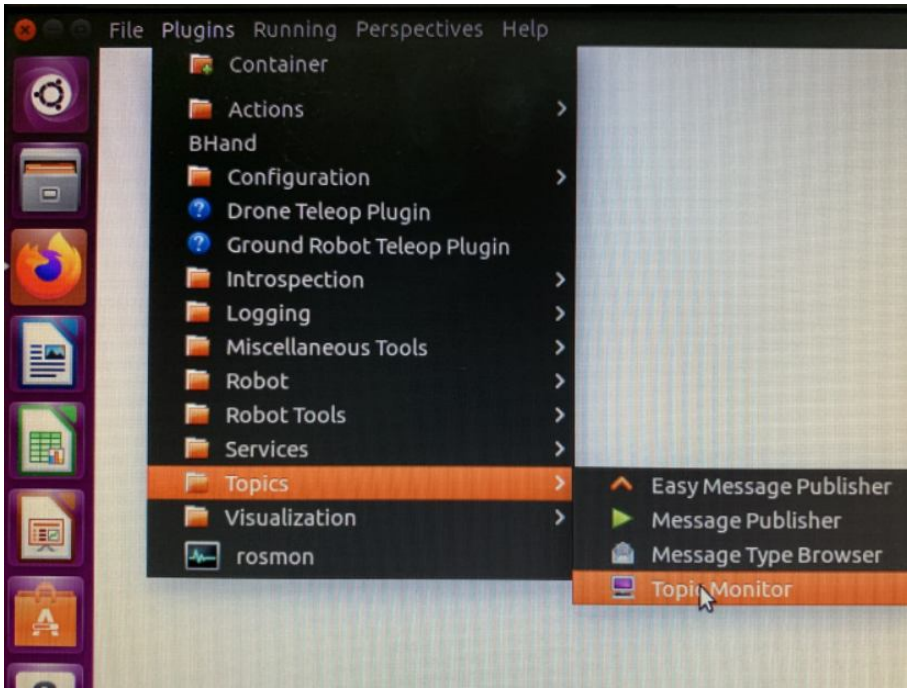
실행하면 옆의 rqt 기본 화면이 열림



여기를 double click 하면

화면이 크게 열림

4. ROS 개발환경 구축

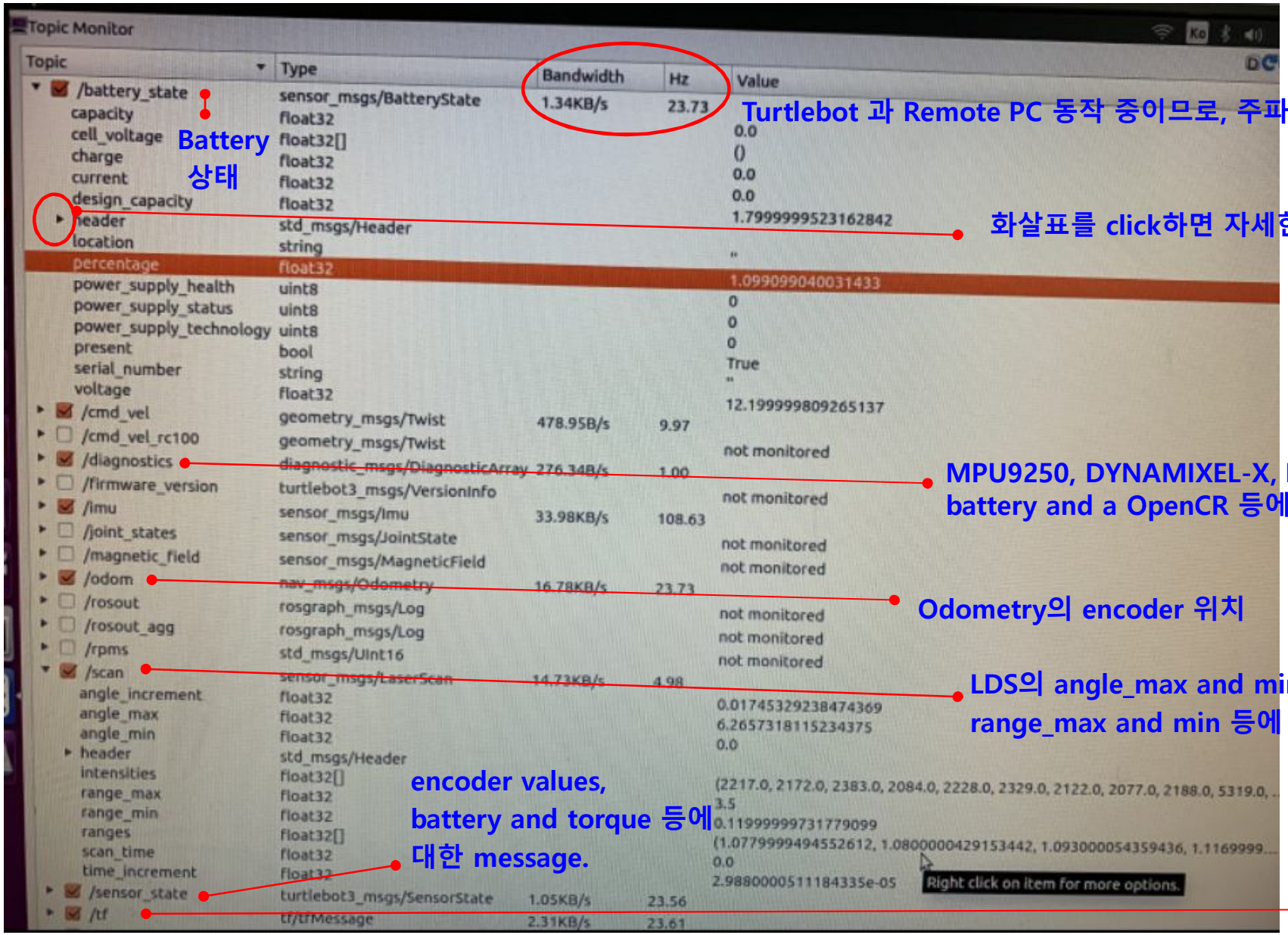


plugins 선택 ➔ Topics 선택 ➔ Topic Monitor 선택

Checkbox를 선택하기

: battery, cmd_vel, diagnostic, odom, scan, sensor_state, tf

: Checkbox를 선택해야 모니터링이 가능함



Battery 상태

Turtlebot 과 Remote PC 동작 중이므로, 주파수 실시간 변화

화살표를 click하면 자세한 내용 확인

MPU9250, DYNAMIXEL-X, HLS-LFCD-LDS, battery and a OpenCR 등에 대한 자가진단

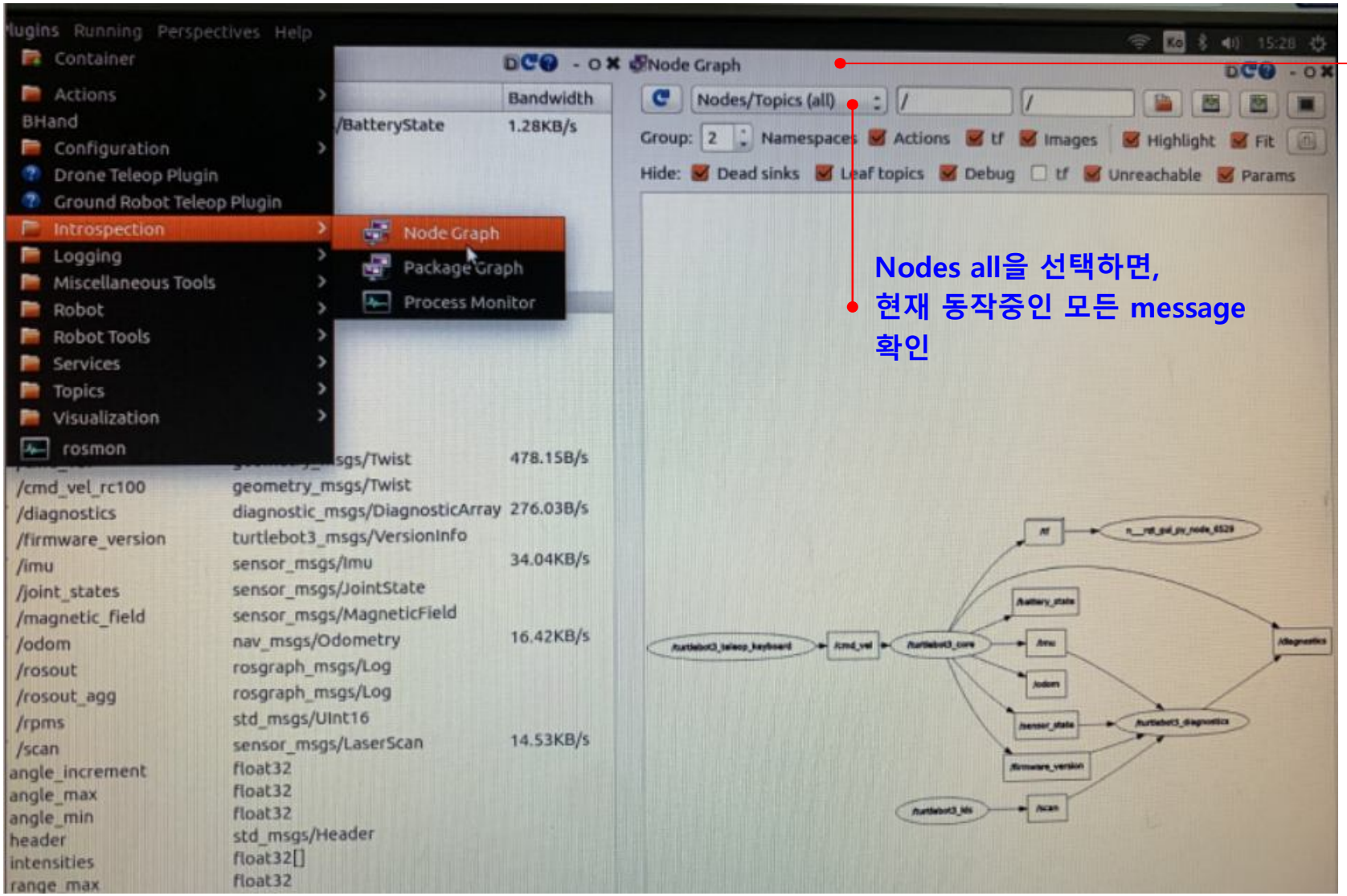
Odometry의 encoder 위치

LDS의 angle_max and min, range_max and min 등에 대한 message

encoder values, battery and torque 등에 대한 message.

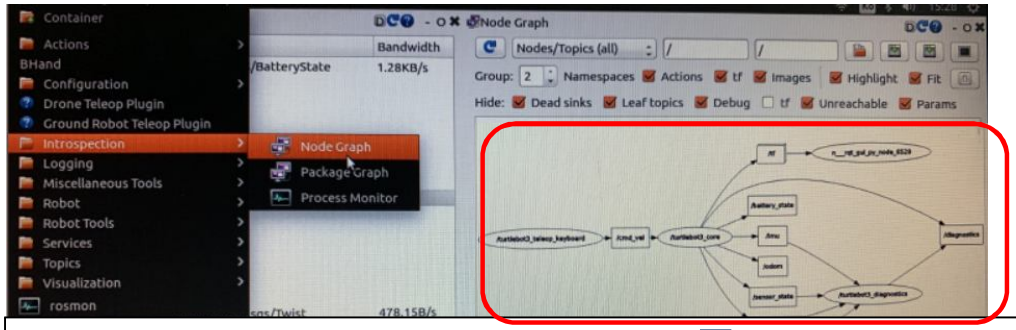
tf(transformation) 로봇의 각 부분에 연결된 Joint 확인

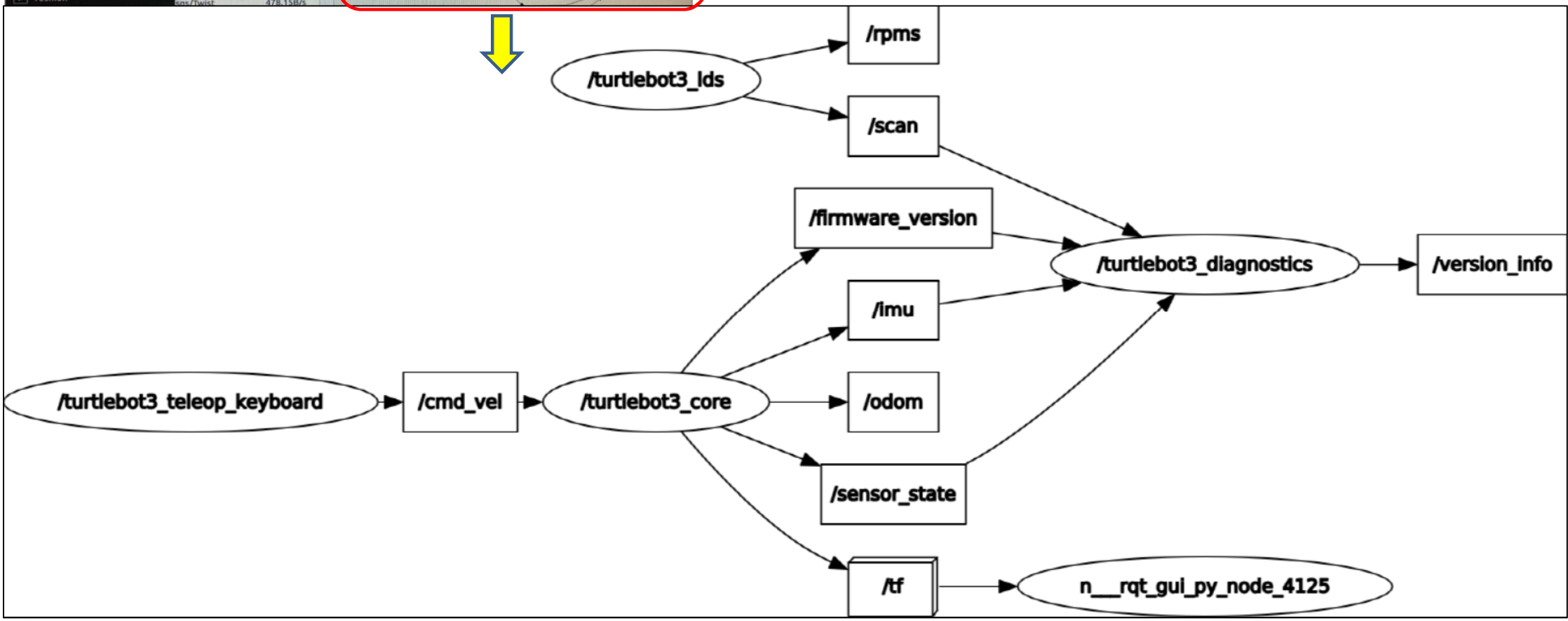
plugins 선택 ➔ Introspection 선택 ➔ Node Graph 선택 (node와 message 관계를 도식화 하여 나타내는 rqt graph 같음)



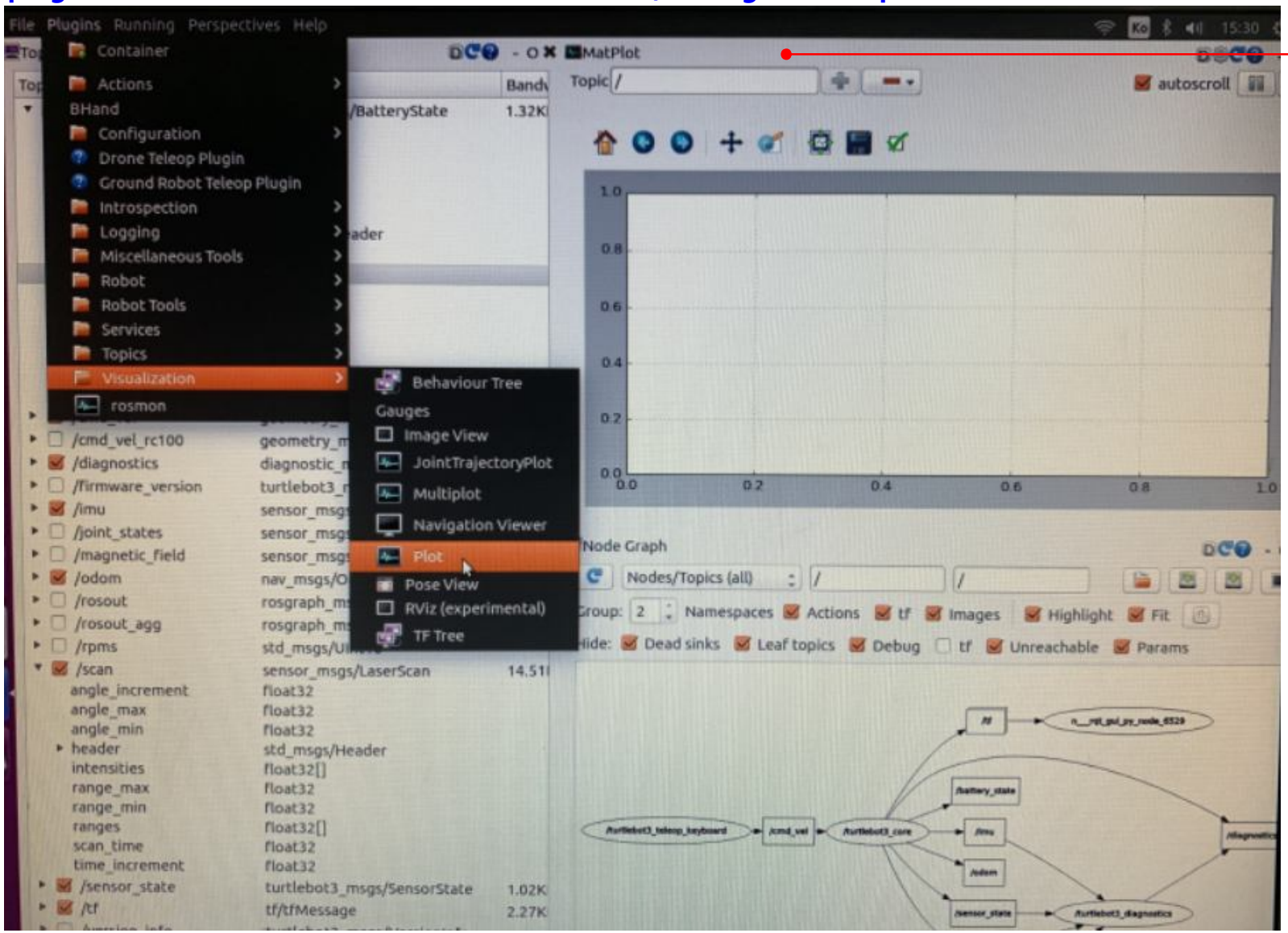
새로 생성된 창을 끌어다가 원하는 위치에 배치하기

Nodes all을 선택하면, 현재 동작중인 모든 message 확인


Container
Actions
BHand
Configuration
Drone Teleop Plugin
Ground Robot Teleop Plugin
Introspection
Logging
Miscellaneous Tools
Robot
Robot Tools
Services
Topics
Visualization
rosmon
Node Graph
Package Graph
Process Monitor
Bandwidth
/BatteryState
1.28KB/s
Node Graph
Nodes/Topics (all)
Group:
Namespaces
Actions
tf
Images
Highlight
Fit
Hide:
Dead sinks
Leaf topics
Debug
Unreachable
Params


/rpms
/turtlebot3_lds
/scan
/firmware_version
/turtlebot3_diagnostics
/version_info
/imu
/turtlebot3_teleop_keyboard
/cmd_vel
/turtlebot3_core
/odom
/sensor_state
/tf
n__rqt_gui_py_node_4125

plugins 선택 ➔ Visualization 선택 ➔ Plot 선택 (message 또는 topic data를 그래프로 나타내는 rqt)

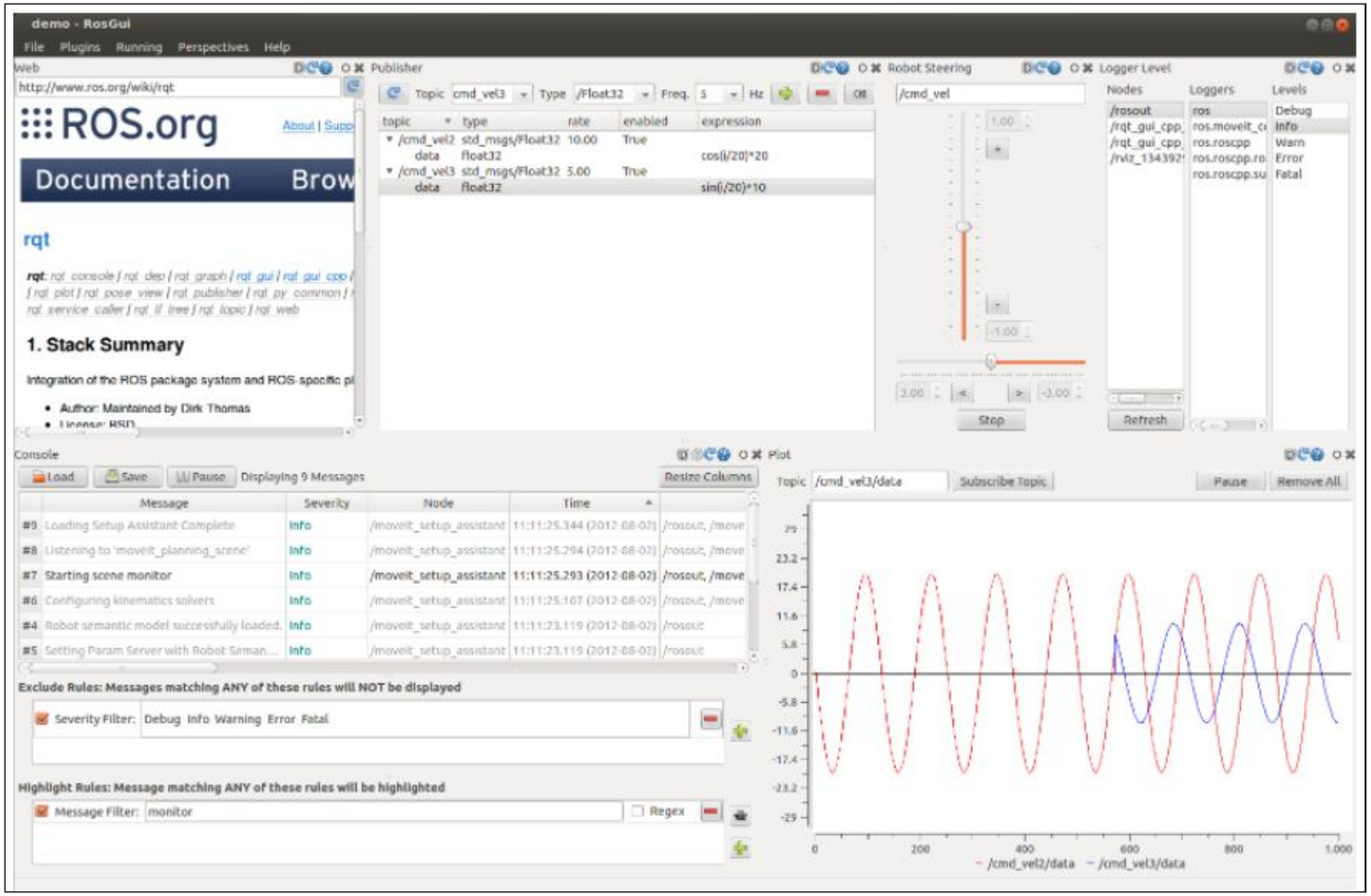


새로 생성된 창을
끌어다가 원하는 위치에
배치하기

4. ROS 개발환경 구축

rqt manual : http://wiki.ros.org/rqt

: kinetic version의 rqt 사용법에 대해 학습하기



4-3. 파일 시스템

4-3-1. 파일 구성

: ROS 소프트웨어 구성을 위한 기본 단위 – package, ROS 응용프로그램은 package 단위로 개발됨

: package는 ROS 최소 단위 실행 프로세서인 node를 하나 이상 포함하거나 다른 node 를 실행하기 위한 설정 파일을 포함

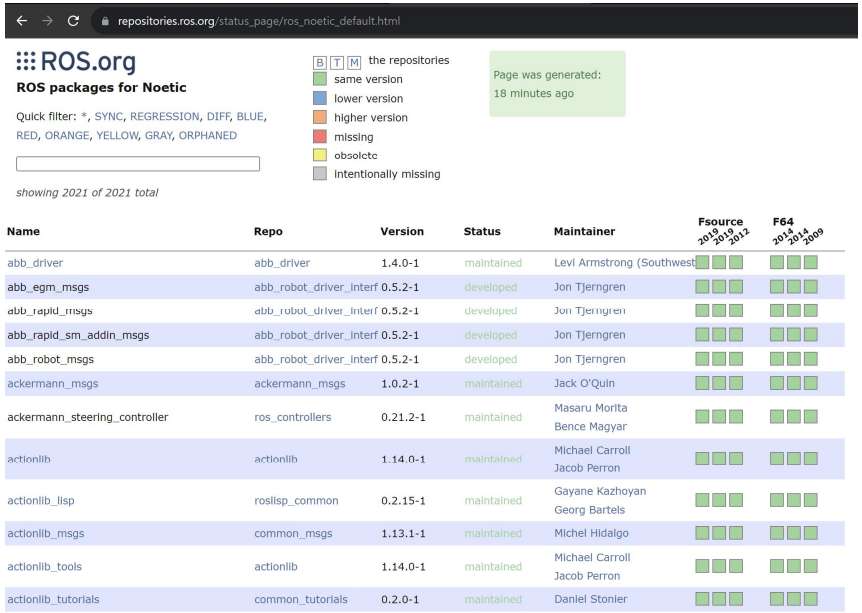
: ROS noetic package http://repositories.ros.org/status_page/ros_noetic_default.html

: Metapackage - 공통된 목적을 지닌 패키지들의 집합(ROS의 file system concepts)

: 각 package는 package.xml 파일을 포함하고 있으며, 이는 패키지의 정보를 보유

- package.xml 은 package 의 이름, 저작자, 라이선스, 의존성 패키지 등을 정의
- ROS 빌드 시스템 catkin은 기본적으로 CMake를 이용하고,

★ package 폴더에 CMakeLists.txt 파일에 빌드 환경을 기술



4. ROS 개발환경 구축

[파일 구성]

: ROS 파일 시스템은 설치 폴더와 사용자 작업 폴더로 구분

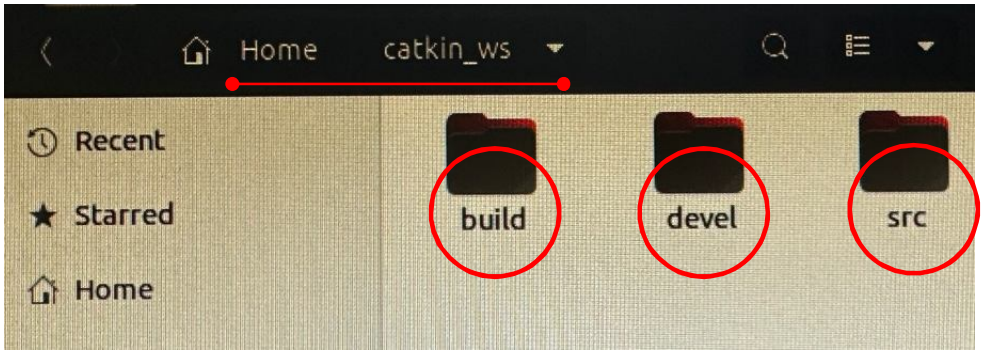
- ROS 설치 폴더는 ROS를 설치하면 /opt 폴더에 ros 폴더 생성

➔ ROS 설치 폴더 경로 /opt/ros/noetic

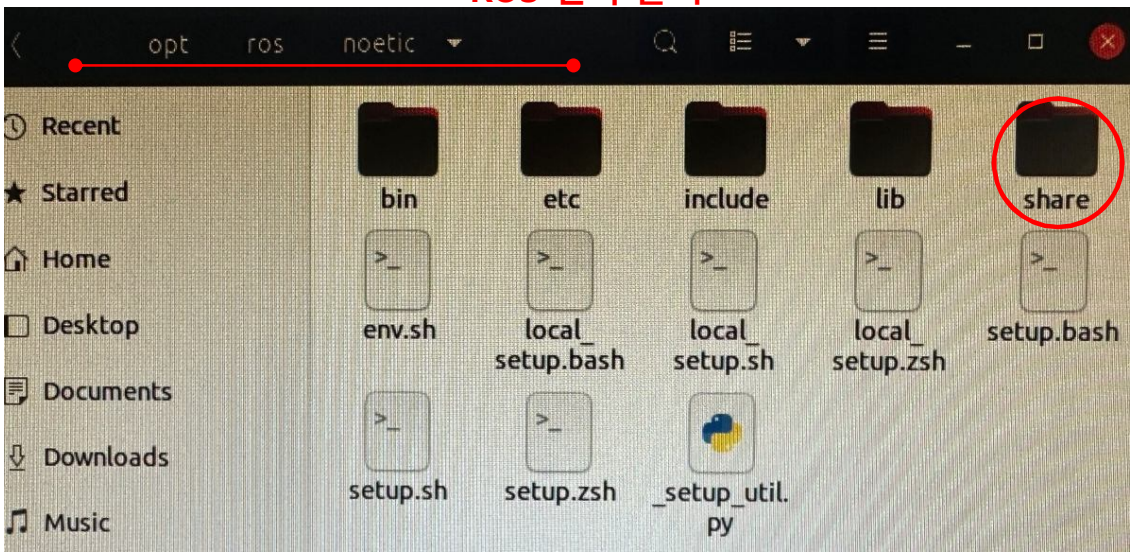
➔ ros 폴더에 roscore를 포함한 핵심 유틸리티와 rqt, Rviz, 로봇 관련 라이브러리, 시뮬레이션, 내비게이션 등 설치

- 사용자 작업 폴더 는 사용자 폴더인 ~/catkin_ws 사용

사용자 작업 폴더

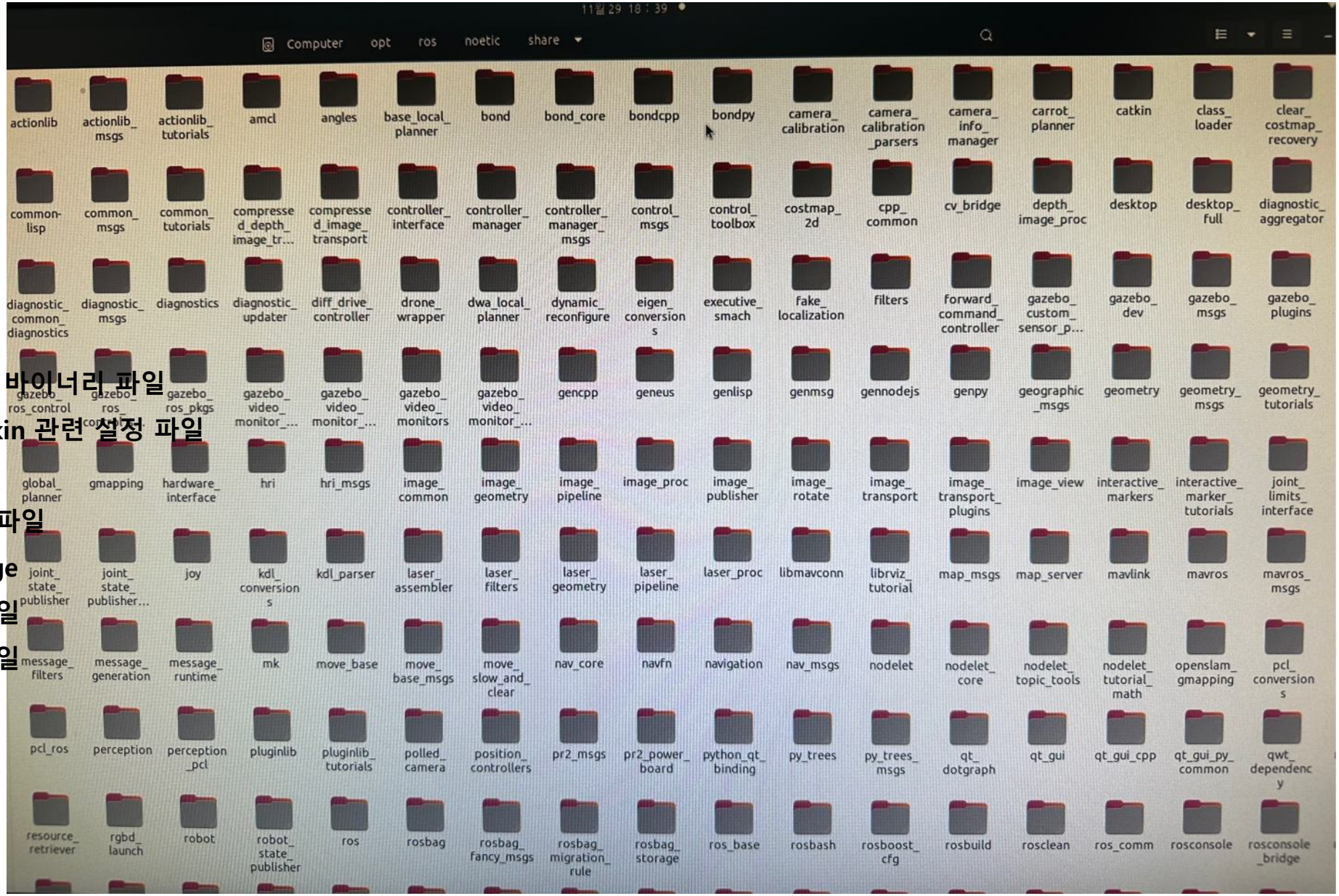


ROS 설치 폴더



ROS 설치 폴더 경로

/opt/ros/noetic/share



/bin 실행 가능한 바이너리 파일

/etc ROS 및 catkin 관련 설정 파일

/include 헤더 파일

/lib 라이브러리 파일

/share ROS package

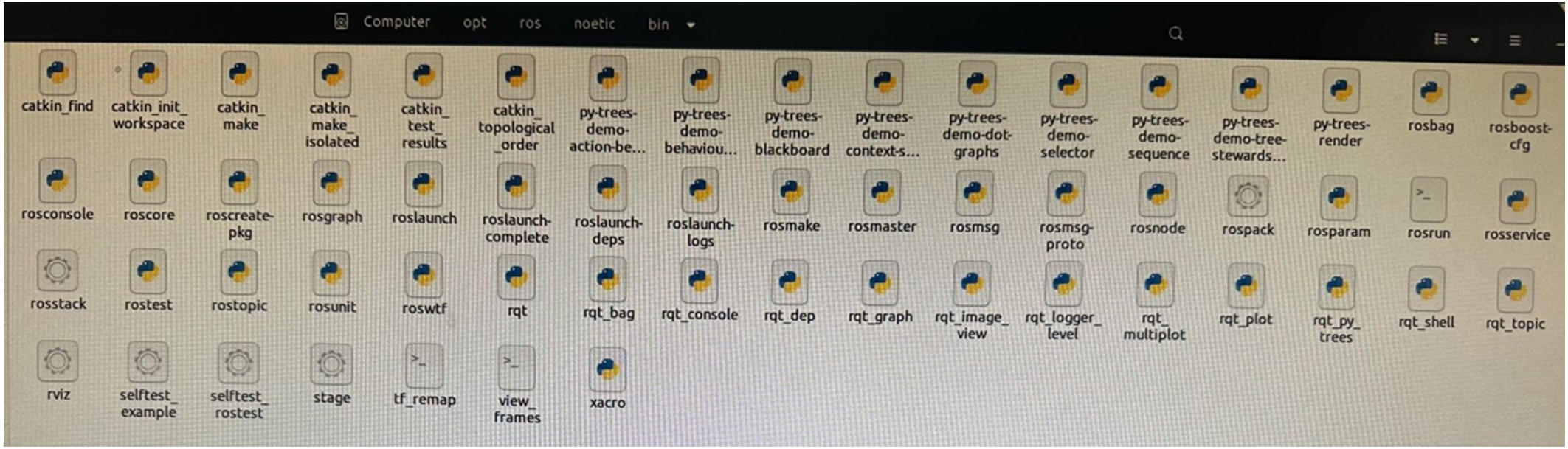
env.* 환경설정 파일

setup.* 환경설정 파일

4. ROS 개발환경 구축

ROS 설치 폴더 경로

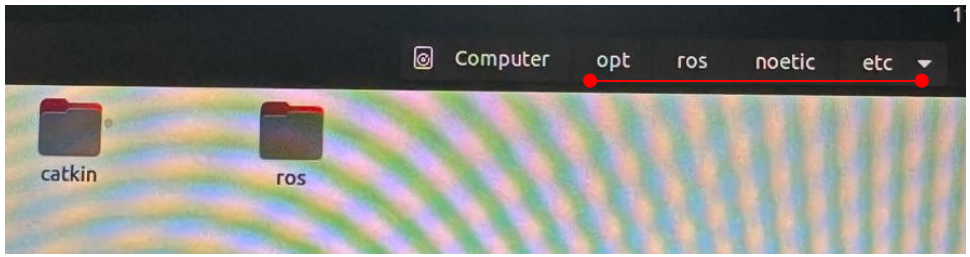
/opt/ros/noetic/bin

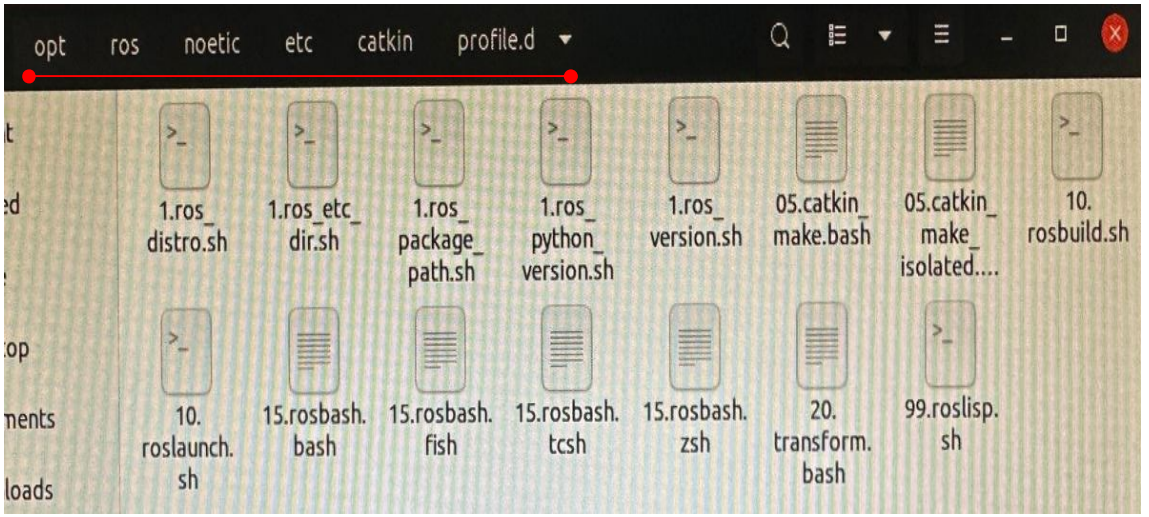


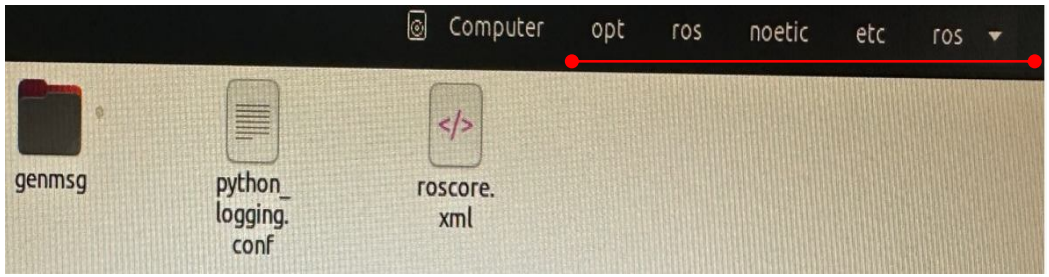
4. ROS 개발환경 구축

4-3-2. 설치 폴더

: ROS 설치 폴더 경로 /opt/ros/noetic/etc







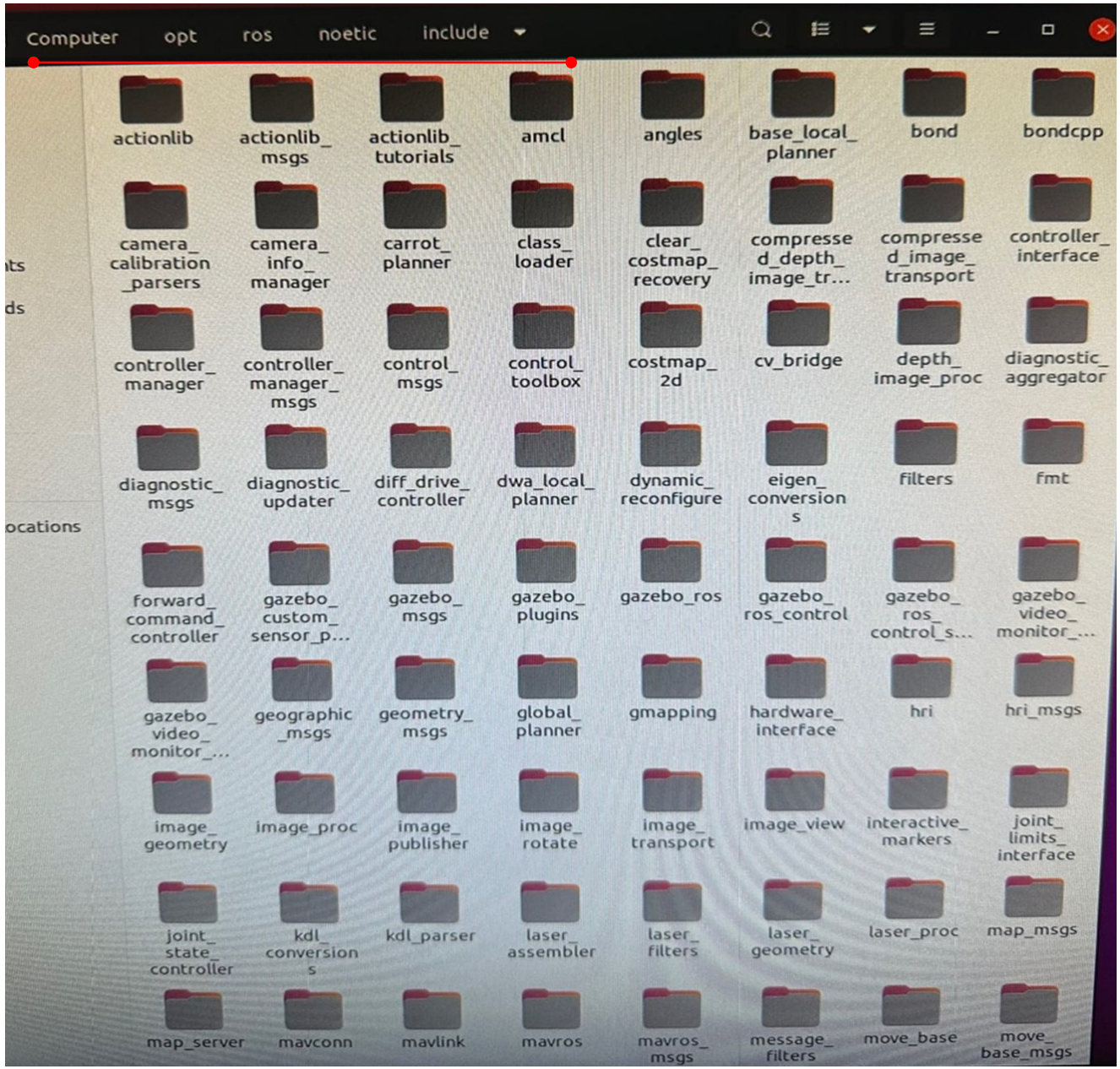
/bin	실행 가능한 바이너리 파일
/etc	ROS 및 catkin 관련 설정 파일
/include	헤더 파일
/lib	라이브러리 파일
/share	ROS package
env.*	환경설정 파일
setup.*	환경설정 파일

4. ROS 개발환경 구축

4-3-2. 설치 폴더

: ROS 설치 폴더 경로 /opt/ros/noetic/include

/bin	실행 가능한 바이너리 파일
/etc	ROS 및 catkin 관련 설정 파일
/include	헤더 파일
/lib	라이브러리 파일
/share	ROS package
env.*	환경설정 파일
setup.*	환경설정 파일

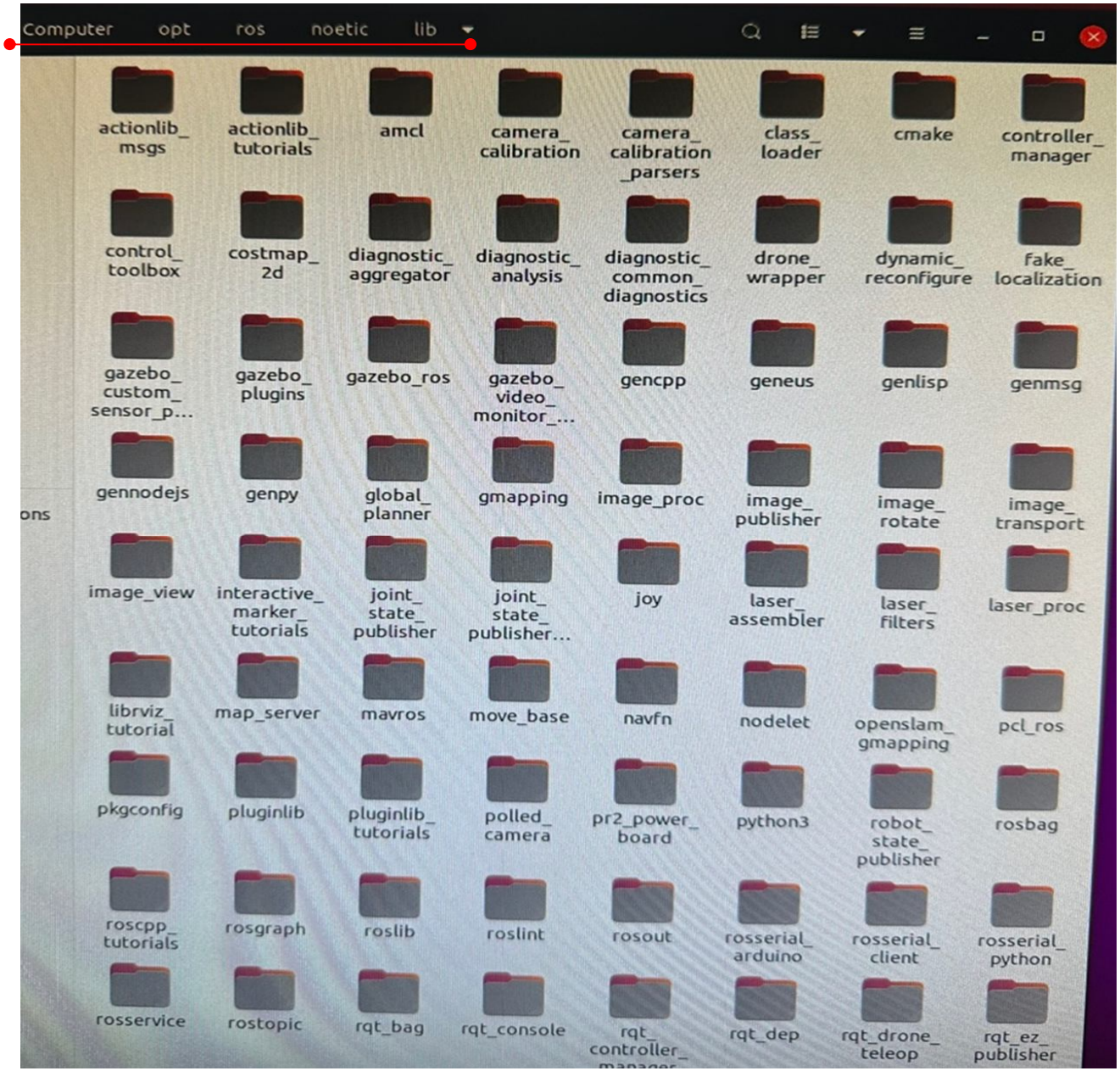


4. ROS 개발환경 구축

4-3-2. 설치 폴더

: ROS 설치 폴더 경로 /opt/ros/noetic/lib

/bin	실행 가능한 바이너리 파일
/etc	ROS 및 catkin 관련 설정 파일
/include	헤더 파일
/lib	라이브러리 파일
/share	ROS package
env.*	환경설정 파일
setup.*	환경설정 파일

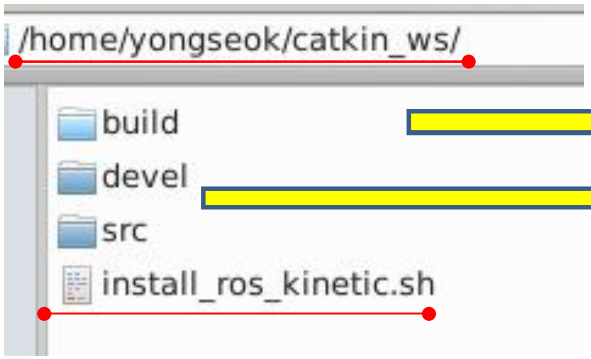


4. ROS 개발환경 구축

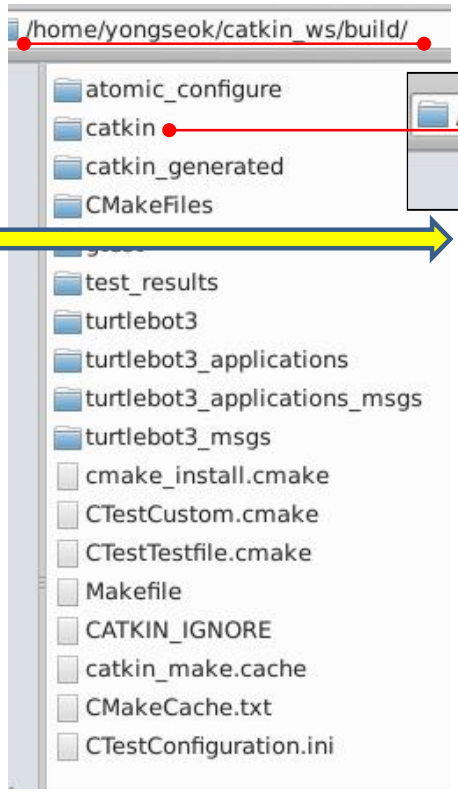
4-3-3. 작업 폴더

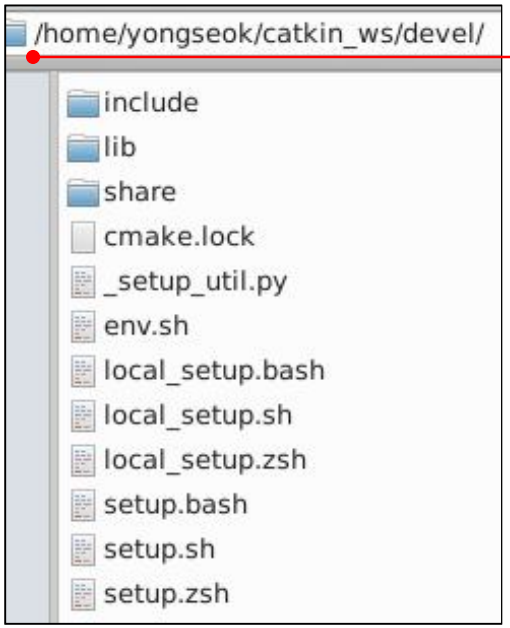
: 작업 폴더 경로 /home/yongseok/catkin_ws/

: build와 devel 폴더는 catkin_make 후에 생성



/build





/devel

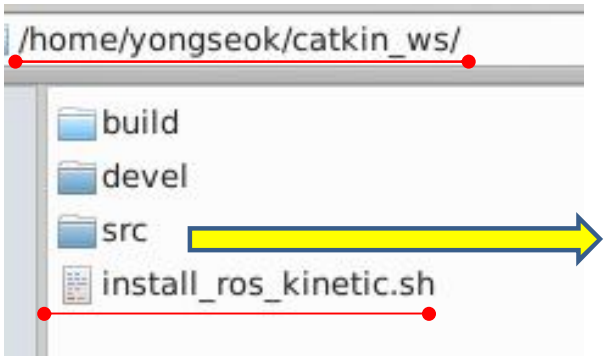
/build	빌드 관련 파일
/devel	msg, srv 헤더 파일, 사용자 package library, 실행 파일
/src	사용자 package

4. ROS 개발환경 구축

4-3-3. 작업 폴더

: 작업 폴더 경로 /home/yongseok/catkin_ws/

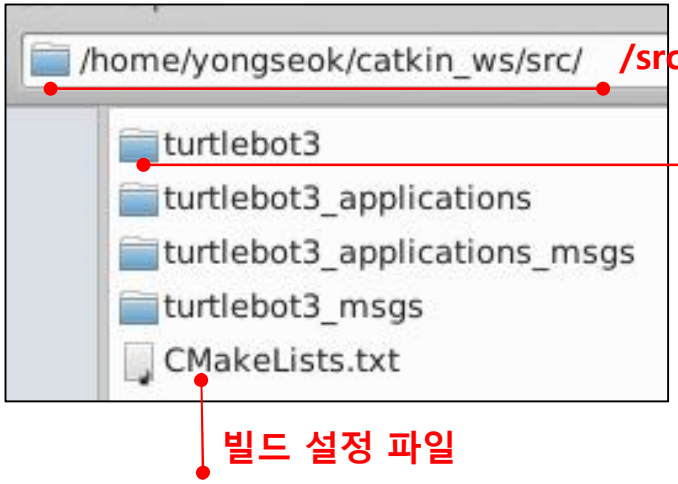
: build와 devel 폴더는 catkin_make 후에 생성

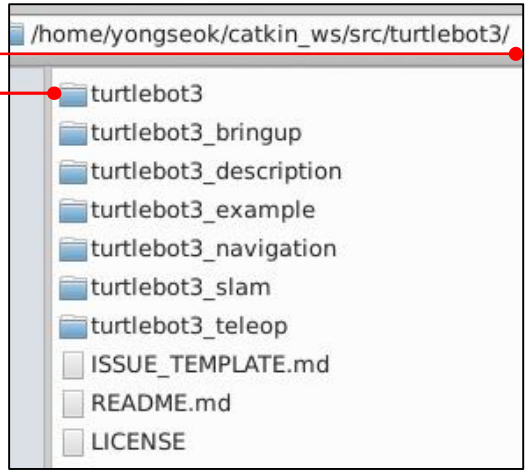


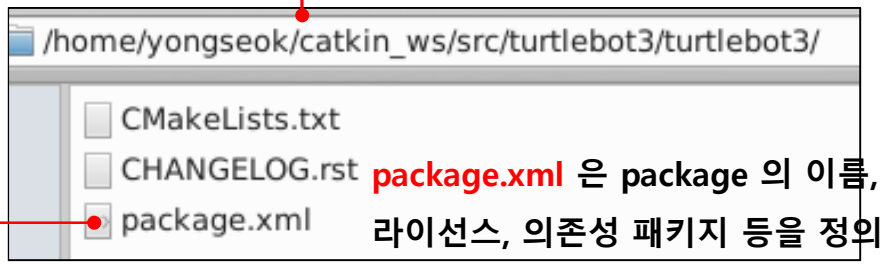
/build 빌드 관련 파일

/devel msg, srv 헤더 파일,
사용자 package library, 실행 파일

/src 사용자 package







package 설정 파일

package.xml 은 package 의 이름, 저작자, 라이선스, 의존성 패키지 등을 정의

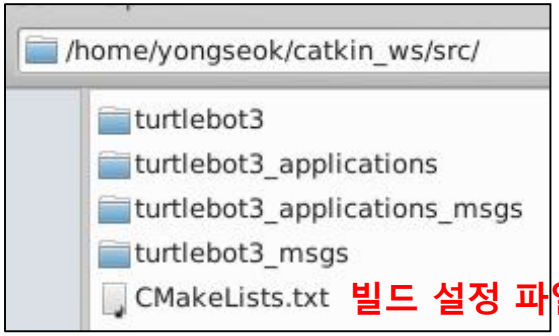
실습 주의 or SKIP

4-3-4. Build system

: ROS 빌드 시스템은 CMake(Cross Platform Make) 사용

: ROS 빌드 환경은 package 폴더의 CMakeLists.txt 파일에 정의

: CMake를 ROS에 맞도록 수정하여 특화된 catkin 빌드 시스템 제공(catkin_make)



(1) package 생성 실습

: package 생성 명령어 catkin_create_pkg

- catkin_create_pkg 는

사용자 package 작성할 때 catkin 빌드 시스템에 필요한 CMakeLists.txt와 package.xml 포함한 package 폴더 생성

: 실습(새 터미널 열기)

① 작업 폴더 이동 $ cd ~/catkin_ws/src

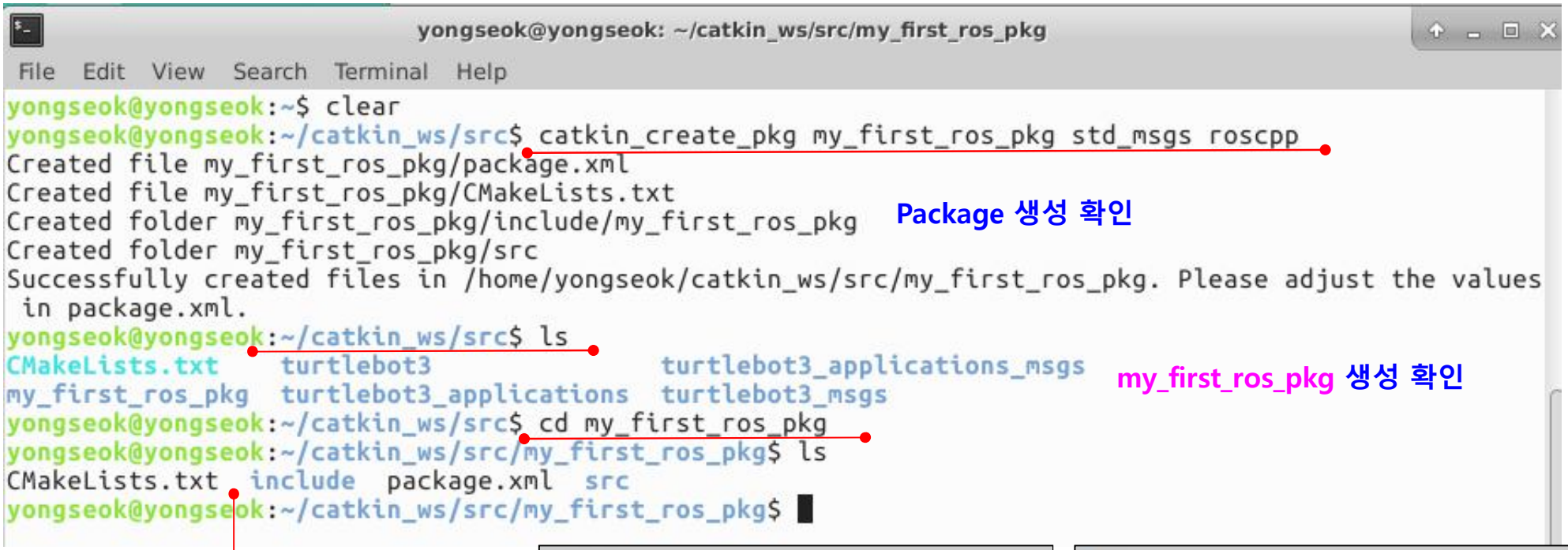
② 생성할 패키지 이름은 my_first_ros_pkg $ catkin_create_pkg my_first_ros_pkg std_msgs roscpp

package 이름

ROS의 표준 메시지 package

C/C++를 위한 Client library

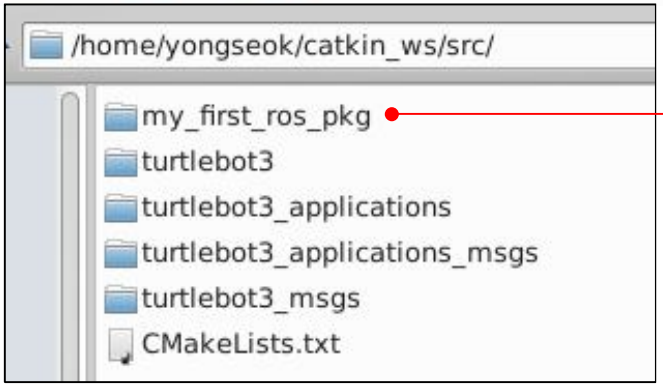
실습 주의 or SKIP

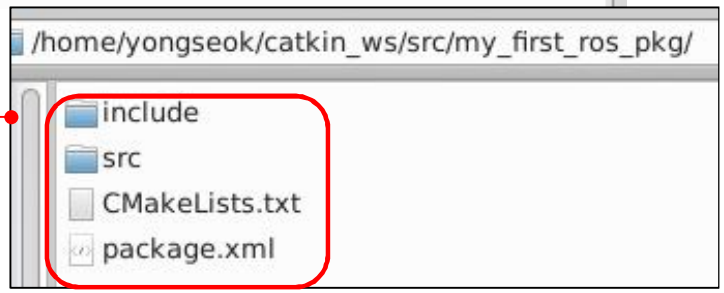


Package 생성 확인

my_first_ros_pkg 생성 확인

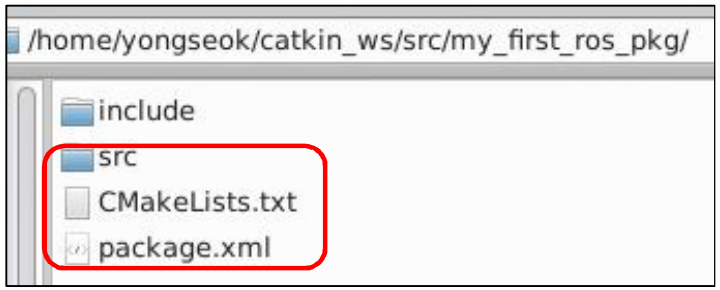
CMakeLists.txt와 package.xml 생성





Directory 위치 주의

실습 주의 or SKIP



③ 터미널에서 gedit를 사용하여 package.xml 열기

$ gedit package.xml

(디렉토리 위치 확인 필요)

```
yongseok@yongseok:~/catkin_ws/src$ cd my_first_ros_pkg
yongseok@yongseok:~/catkin_ws/src/my_first_ros_pkg$ ls
CMakeLists.txt  include  package.xml  src
yongseok@yongseok:~/catkin_ws/src/my_first_ros_pkg$ gedit package.xml
```

package
수정전.xml

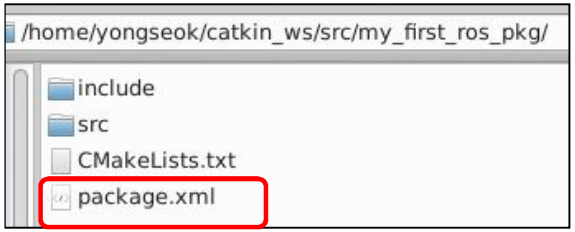
Open + package.xml ~/catkin_ws/src/my_first_ros_pkg Save

File Edit View Search Tools Documents Help

```
<?xml version="1.0"?>
<package format="2">
  <name>my_first_ros_pkg</name>
  <version>0.0.0</version>
  <description>The my_first_ros_pkg package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="yongseok@todo.todo">yongseok</maintainer>


  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>


  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/my_first_ros_pkg</url> -->


  ional, multiple are allowed, one per tag -->
     to be maintainers, but could be -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->


  <!-- The *depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -
  <!--   <depend>roscpp</depend> -->
```

④ package.xml 이해

실습 주의 or SKIP



: .xml 확장자는 XML(Extensible Markup Language) 파일

- 사용자 정의 태그를 사용하여 문서의 구조 및 기타 기능을 설명하는 일반 텍스트 파일
- XML은 W3C(World Wide Web Consortium)에서 만든 markup language(꺾쇠 < 과 > 로 표현)로 사람과 기계 모두가 읽을 수 있는 문서를 인코딩하기 위한 구문을 정의

: package.xml http://docs.ros.org/en/jade/api/catkin/html/howto/format2/catkin_library_dependencies.html

Open + package.xml ~/catkin_ws/src/my_first_ros_pkg Save

File Edit View Search Tools Documents Help

```
<?xml version="1.0"?>
<package format="2">
  <name>my_first_ros_pkg</name>
  <version>0.0.0</version>
  <description>The my_first_ros_pkg package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="yongseok@todo.todo">yongseok</maintainer>


  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>
```

<?xml>: 문서 문법을 정의하는 문구로 아래의 내용은 xml 버전 1.0을 따름

<package>: 이 구문부터 맨 끝의 </package>까지가 ROS 패키지 설정 부분

<name>: 패키지의 이름. 패키지를 생성할 때 입력한 패키지 이름이 사용

<version>: 패키지의 버전. 자유롭게 지정

<description>: 패키지에 대한 설명

<maintainer>: 패키지 관리자의 이름과 이메일 주소

<license>: 라이선스 기재 BSD, MIT, Apache, GPLv3, LGPLv3 등

실습 주의 or SKIP

```
<!-- Url tags are optional, but multiple are allowed, one per tag -->
<!-- Optional attribute type can be: website, bugtracker, or repository -->
<!-- Example: -->
<!-- <url type="website">http://wiki.ros.org/my_first_ros_pkg</url> -->

<!-- Author tags are optional, multiple are allowed, one per tag -->
<!-- Authors do not have to be maintainers, but could be -->
<!-- Example: -->
<!-- <author email="jane.doe@example.com">Jane Doe</author> -->

<!-- The *depend tags are used to specify dependencies -->
<!-- Dependencies can be catkin packages or system dependencies -->
<!-- Examples: -->
<!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
<!--   <depend>roscpp</depend> -->
<!--   Note that this is equivalent to the following: -->
<!--   <build_depend>roscpp</build_depend> -->
<!--   <exec_depend>roscpp</exec_depend> -->
<!-- Use build_depend for packages you need at compile time: -->
<!--   <build_depend>message_generation</build_depend> -->
<!-- Use build_export_depend for packages you need in order to build against this package: -->
<!--   <build_export_depend>message_generation</build_export_depend> -->
<!-- Use buildtool_depend for build tool packages: -->
<!--   <buildtool_depend>catkin</buildtool_depend> -->
<!-- Use exec_depend for packages you need at runtime: -->
<!--   <exec_depend>message_runtime</exec_depend> -->
<!-- Use test_depend for packages you need only for testing: -->
<!--   <test_depend>gtest</test_depend> -->
<!-- Use doc_depend for packages you need only for building documentation: -->
<!--   <doc_depend>doxygen</doc_depend> -->
<buildtool_depend>catkin</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>std_msgs</build_depend>
<build_export_depend>roscpp</build_export_depend>
<build_export_depend>std_msgs</build_export_depend>
<exec_depend>roscpp</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

<url>: 패키지를 설명하는 웹 페이지, 버그 관리, 저장소 등 주소 기재

<author>: 패키지 개발에 참여한 개발자 이름과 이메일 주소

<buildtool_depend>: 빌드 시스템 의존성 기술. Catkin 빌드 시스템이므로 catkin 입력

<build_depend>: 패키지를 빌드할 때 의존하는 패키지 이름

<build_export_depend > : roscpp, std_msgs를 내보낼 때(export)
header file을 포함하여 build_export_depend에 추가해야 함

<exec_depend> : package를 running중에 의존성이 필요하다면 추가

실습 주의 or SKIP

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

  </export>
</package>
```

<export>: ROS에서 명시하지 않은 태그명을 사용할 때 사용

<package>: 이 구문부터 맨 끝의 </package>까지가 ROS 패키지 설정 부분

실습 주의 or SKIP

⑤ package.xml 수정하기

```
<?xml version="1.0"?>
<package>
  <name>my_first_ros_pkg</name>
  <version>0.0.1</version>
  <description>The my_first_ros_pkg package</description>
  <author email="ys.chi@dongseo.ac.kr">yongseok chi</author>
  <maintainer email="ys.chi@dongseo.ac.kr">yongseok chi</maintainer>
  <license>Apache License 2.0</license>

  <url type="bugtracker">https://github.com/ROBOTIS-GIT/ros_tutorials/issues</url>
  <url type="repository">https://https://github.com/ROBOTIS-GIT/ros_tutorials</url>
  <url type="website">http://www.robotis.com</url>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>roscpp</build_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>roscpp</run_depend>
  <export></export>
</package>
```

실습 주의 or SKIP

(2) CMakeLists.txt (빌드 설정파일) 실습

① CMakeLists.txt (빌드 설정파일)이해

: ROS 빌드 시스템은 CMake(Cross Platform Make) 사용

: ROS 빌드 환경은 package 폴더의 CMakeLists.txt 파일에 정의

: 실행 파일 생성, 의존성 패키지 우선 빌드, 링크 생성 등을 설정

② 터미널에서 gedit를 사용하여 CMakeLists.txt 열기

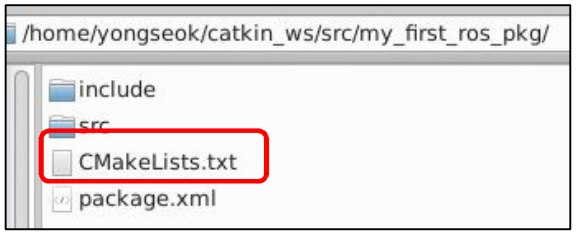
$ cd ~/catkin_ws/src/my_first_pkg (디렉토리 위치 확인)

$ ls (디렉토리 내용 확인)

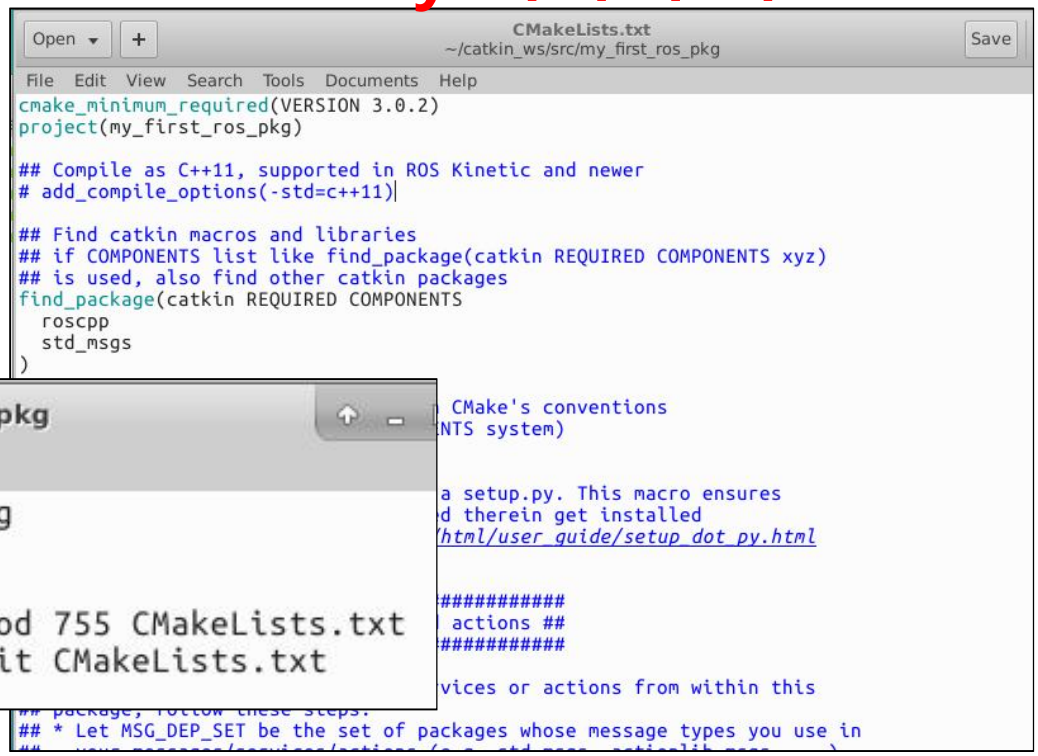
$ chmod 755 CMakeLists.txt (권한 변경)

$ gedit CMakeLists.txt (gedit로 파일 열기)

CMakeLists수정 전.txt



Directory 위치 주의



실습 주의 or SKIP

CMakeLists.txt 이해 https://wiki.ros.org/catkin/CMakeLists.txt

CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.0.2)
2  project(my_first_ros_pkg)
3
4  ## Compile as C++11, supported in ROS Kinetic and newer
5  # add_compile_options(-std=c++11)
6
7  ## Find catkin macros and libraries
8  ## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
9  ## is used, also find other catkin packages
10 find_package(catkin REQUIRED COMPONENTS
11   roscpp
12   std_msgs
13 )
14
15 ## System dependencies are found with CMake's conventions
16 # find_package(Boost REQUIRED COMPONENTS system)
17
18
19 ## Uncomment this if the package has a setup.py. This macro ensures
20 ## modules and global scripts declared therein get installed
21 ## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
22 # catkin_python_setup()
```

운영체제에 설치된 cmake의 최소 요구 버전. 현재 3.0.2

project : package 이름. package.xml에서 입력한 package 이름
만약 package.xml의 <name> 태그에 기재한 package 이름과 다르면 빌드할 때 에러 발생

find_package : catkin 빌드를 할 때 요구되는 구성 요소 package
의존성 package로 roscpp 와 std_msgs가 추가됨
여기에 입력된 package가 없다면 catkin 빌드 에러 발생

ROS 이외 package 사용할 때 사용되는 방법
예, Boost를 사용할 때 system package가 설치되어야 함
(Boost는 C++ 프로그래밍 언어를 위한 선형대수, 의사 난수 발생, 멀티스레딩, 영상 처리, 정규 표현식 같은 작업들과 구조들을 지원하는 라이브러리)

catkin_python_setup : 파이썬, 즉 rospy를 사용할 때 설정하는 옵션
파이썬 설치 프로세스인 setup.py를 부르는 역할

실습 주의 or SKIP

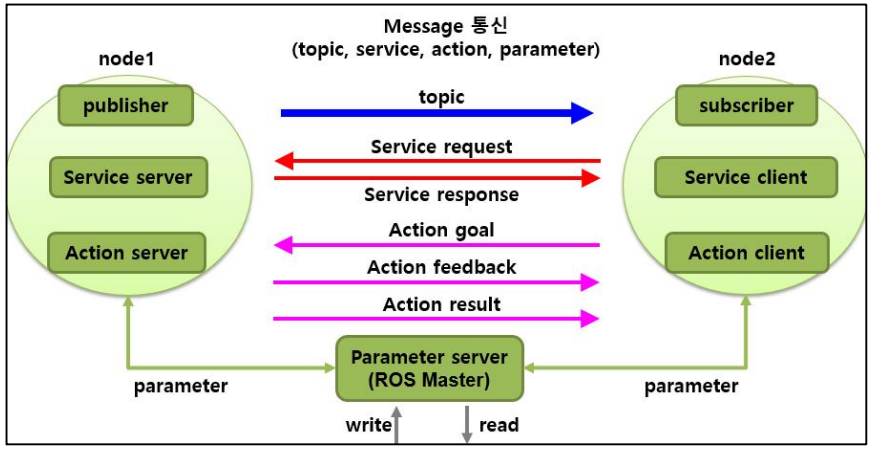
```
46    # * add every package in MSG_DEP_SET to generate_messages(DEPENDENCIES
47
48    ## Generate messages in the 'msg' folder
49    # add_message_files(
50    #   FILES
51    #   Message1.msg
52    #   Message2.msg
53    # )
54
55    ## Generate services in the 'srv' folder
56    # add_service_files(
57    #   FILES
58    #   Service1.srv
59    #   Service2.srv
60    # )
61
62    ## Generate actions in the 'action' folder
63    # add_action_files(
64    #   FILES
65    #   Action1.action
66    #   Action2.action
67    # )
68
69    ## Generate added messages and services with any dependencies listed
70    # generate_messages(
71    #   DEPENDENCIES
72    #   std_msgs
73    # )
74
75    ################################################
76    ## Declare ROS dynamic reconfigure parameters ##
77    ################################################
78
79    ## To declare and build dynamic reconfigure parameters within this
```

add_message_files : message 파일 추가 옵션
FILES를 사용하면 현재 package 폴더의 msg 폴더 안에
.msg 파일 참조하여 헤더 파일(*.h)을 자동 생성
여기에서는 Message1.msg와 Message2.msg 파일 사용 옵션

add_service_files : service 파일 추가 옵션
FILES를 사용하면 package 폴더의 srv 폴더 안의 .srv 파일 참조
여기에서는 Service1.srv와 Service2.srv의 서비스 파일 사용 옵션

add_action_files : action 파일 추가 옵션
FILES를 사용하면 package 폴더의 action 폴더 안의 .action 파일 참조
여기에서는 Action1.action와 Action2.action 의 파일 사용 옵션

generate_messages : 의존하는 메시지를 설정 옵션
DEPENDENCIES 옵션에 의해
std_msgs 메시지 패키지 사용 설정



실습 주의 or SKIP

```
88  # Generate dynamic reconfigure parameters in the 'cfg' folder
89  # generate_dynamic_reconfigure_options(
90  #   cfg/DynReconf1.cfg
91  #   cfg/DynReconf2.cfg
92  # )
93
94  ###################################
95  ## catkin specific configuration ##
96  ###################################
97  ## The catkin_package macro generates cmake config files for your package
98  ## Declare things to be passed to dependent projects
99  ## INCLUDE_DIRS: uncomment this if your package contains header files
100 ## LIBRARIES: libraries you create in this project that dependent projects also need
101 ## CATKIN_DEPENDS: catkin_packages dependent projects also need
102 ## DEPENDS: system dependencies of this project that dependent projects also need
103 catkin_package(
104 #  INCLUDE_DIRS include
105 #  LIBRARIES my_first_ros_pkg
106 #  CATKIN_DEPENDS roscpp std_msgs
107 #  DEPENDS system_lib
108 )
109
110 ###########
111 ## Build ##
112 ###########
113
114 ## Specify additional locations of header files
115 ## Your package locations should be listed before other locations
116 include_directories(
117 # include
118   ${catkin_INCLUDE_DIRS}
119 )
120
```

generate_dynamic_reconfigure_options :
dynamic_reconfigure 사용할 때 참조하는 설정 파일 불러오는 설정
dynamic Reconfigure : node들에서 제공하는 설정 값 변경을 위한 GUI 설정 변경

Catkin build option :
INCLUDE_DIRS 뒤에 설정한 패키지 내부 폴더인 include의 헤더 파일 사용 설정
LIBRARIES 뒤에 설정한 패키지의 라이브러리를 사용 설정
CATKIN_DEPENDS 뒤에 roscpp나 std_msgs 등 의존하는 패키지 지정
DEPENDS는 시스템 의존 패키지를 정의하는 설정

include_directories : include 폴더 지정 옵션
${catkin_INCLUDE_DIRS}는 각 패키지 안의 include 폴더를 의미하고
이 안의 헤더 파일을 이용하겠다는 설정

실습 주의 or SKIP

add_library : 빌드 후 생성할 라이브러리를 선언
my_first_ros_pkg 패키지의 src 폴더 my_first_ros_pkg.cpp 파일 참조하여
my_first_ros_pkg 라이브러리를 생성하는 명령

```
## Declare a C++ library
# add_library(${PROJECT_NAME}
#   src/${PROJECT_NAME}/my_first_ros_pkg.cpp
# )

## Add cmake target dependencies of the library
## as an example, code may need to be generated before libraries
## either from message generation or dynamic reconfigure
# add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/my_first_ros_pkg_node.cpp)

## Rename C++ executable without prefix
## The above recommended prefix causes long target names, the following renames the
## target back to the shorter version for ease of user use
## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg someones_pkg_node"
# set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX "")

## Add cmake target dependencies of the executable
## same as for the library above
# add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )
```

add_dependencies
: 라이브러리 및 실행 파일을 빌드 전에 생성해야 할 의존성 메시지 및
dynamic reconfigure가 있다면 우선적으로 수행하라는 설정
my_first_ros_pkg 라이브러리에 의존성 메시지, dynamic reconfigure 생성

add_executable : 빌드 후 생성할 실행 파일에 대한 옵션
src/my_first_ros_pkg_node.cpp 파일 참조하여
my_first_ros_pkg_node 실행 파일 생성

target_link_libraries :
지정 실행 파일을 생성하기에 앞서 링크해야 하는 라이브러리와
실행 파일을 링크시키는 옵션

실습 주의 or SKIP

③ CMakeLists.txt 수정하기

```
cmake_minimum_required(VERSION 3.0.2)
project(my_first_ros_pkg)
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
catkin_package(CATKIN_DEPENDS roscpp std_msgs)
include_directories(${catkin_INCLUDE_DIRS})
add_executable(hello_world_node src/hello_world_node.cpp)
target_link_libraries(hello_world_node ${catkin_LIBRARIES})
```

④ hello_world_node.cpp 소스 코드 작성하기

CMakeLists.txt 에서 add_executable(hello_world_node src/hello_world_node.cpp) 설정하여,

아래 경로에 hello_world_node.cpp 생성하고 작성하기

```
$ cd ~/catkin_ws/src/my_first_ros_pkg/src/
$ gedit hello_world_node.cpp
```

⑤ hello_world_node.cpp 수정하기

실습 주의 or SKIP

```
#include <ros/ros.h>
#include <std_msgs/String.h>
#include <sstream>
int main(int argc, char **argv)
{
  ros::init(argc, argv, "hello_world_node");
  ros::NodeHandle nh;
  ros::Publisher chatter_pub = nh.advertise<std_msgs::String>("say_hello_world", 1000);
  ros::Rate loop_rate(10);
  int count = 0;

  while (ros::ok())
  {
     std_msgs::String msg;
     std::stringstream ss;
     ss << "hello world!" << count;
     msg.data = ss.str();
     ROS_INFO("%s", msg.data.c_str());
     chatter_pub.publish(msg);
     ros::spinOnce();
     loop_rate.sleep();
     ++count;
  }
return 0;
}
```

실습 주의 or SKIP

Directory 위치 주의

(4) package 빌드 및 node 실행

① ROS 패키지의 프로파일을 갱신 및 빌드

$ rospack profile

$ cd ~/catkin_ws && catkin_make

② node 실행

: 에러 없이 빌드 완료되면

~/catkin_ws/devel/lib/my_first_ros_pkg에

hello_world_node 파일 생성됨

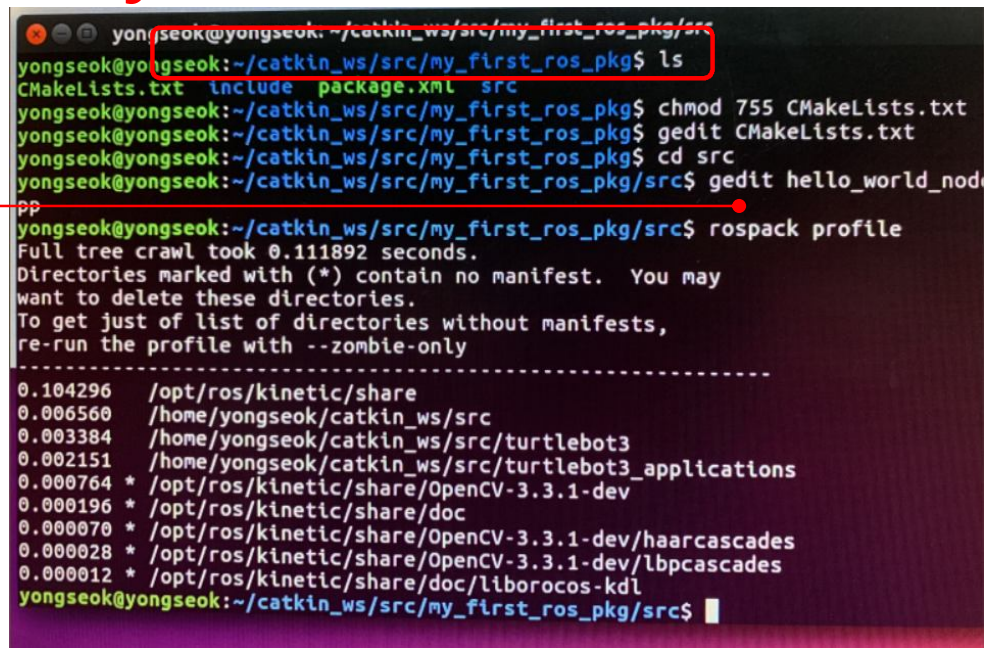
: 새 터미널에서

$ roscore

: 새 터미널에서 my_first_ros_pkg 패키지의 hello_world_node 노드 실행

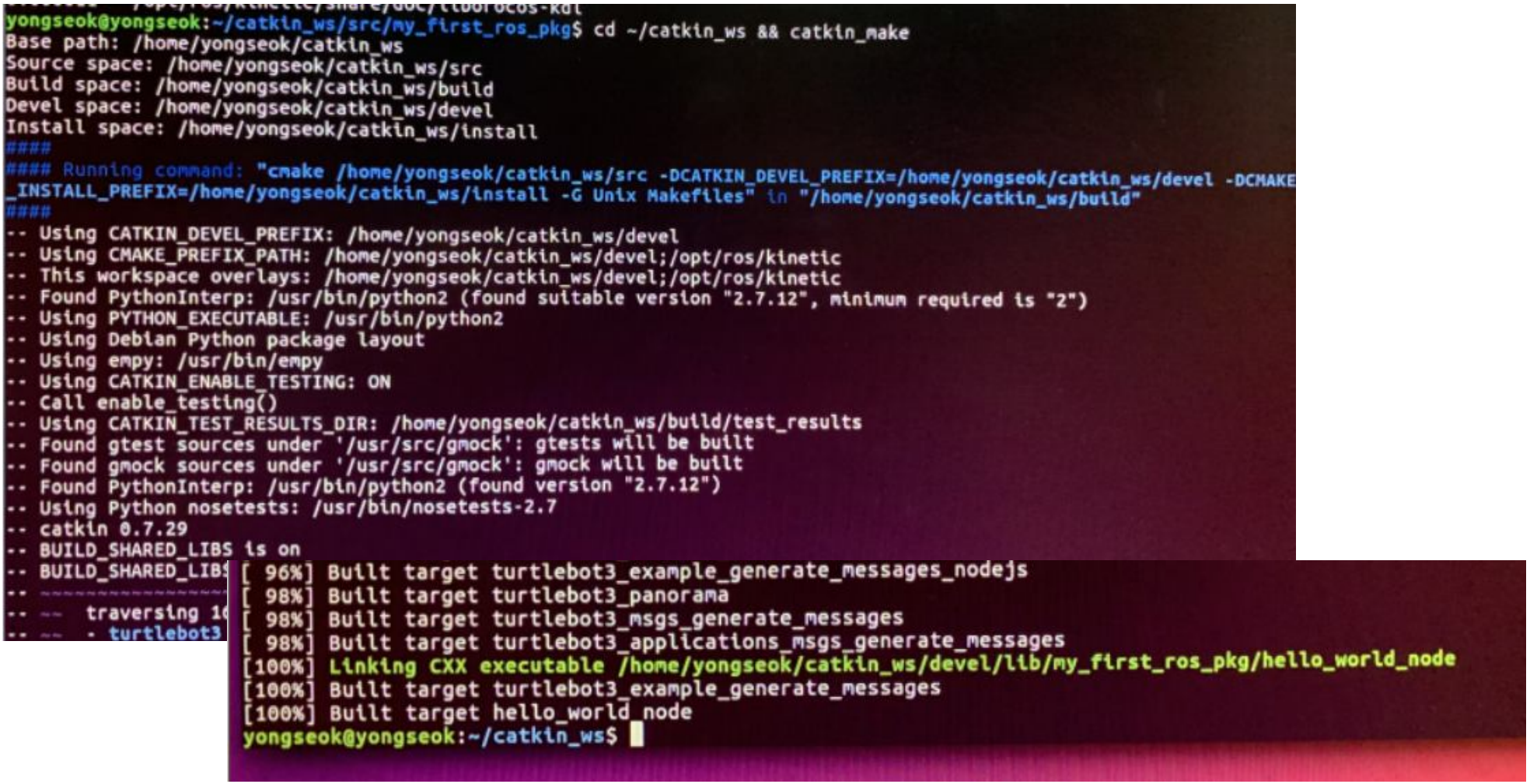
$ rosrun my_first_ros_pkg hello_world_node

: 중단할때는 ctl + c



실습 주의 or SKIP

$ cd ~/catkin_ws && catkin_make



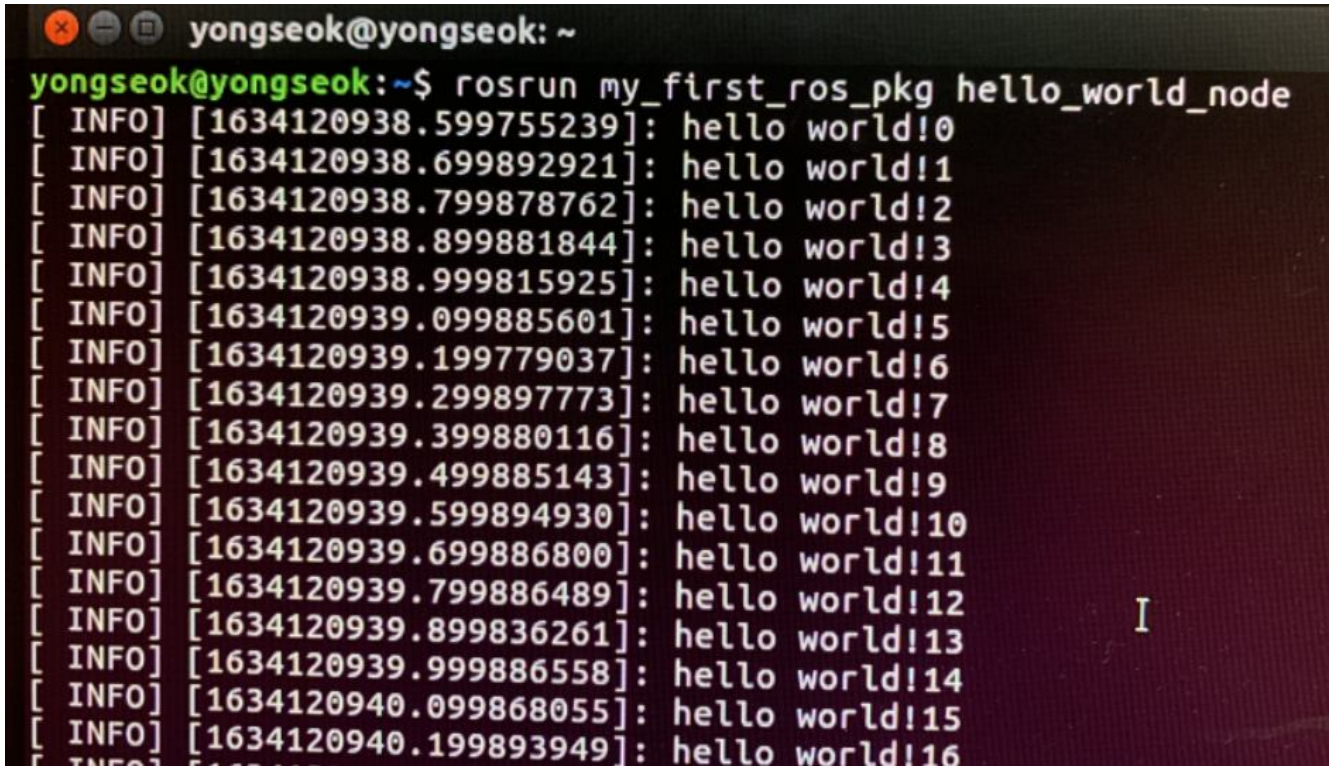
실습 주의 or SKIP

: 새 터미널에서

$ roscore

: 새 터미널에서 my_first_ros_pkg 패키지의 hello_world_node 노드 실행

$ rosrun my_first_ros_pkg hello_world_node

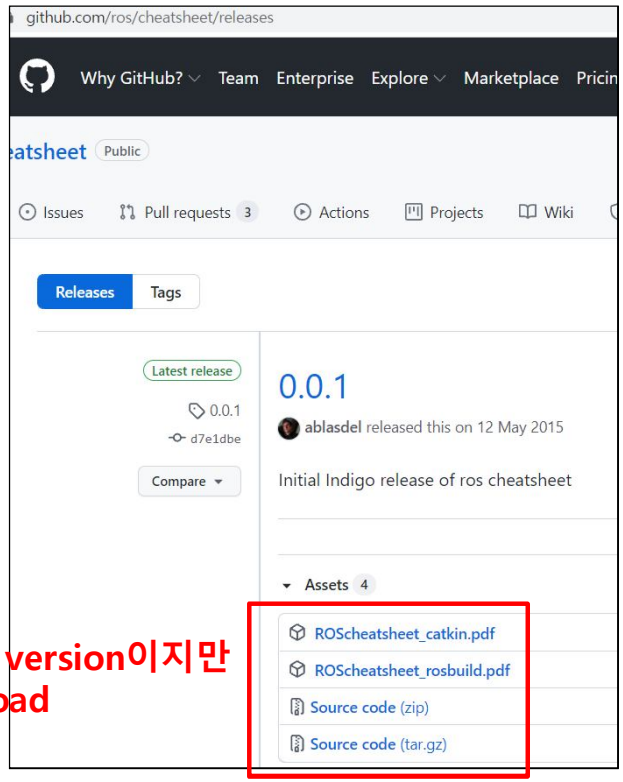


5. ROS 명령어

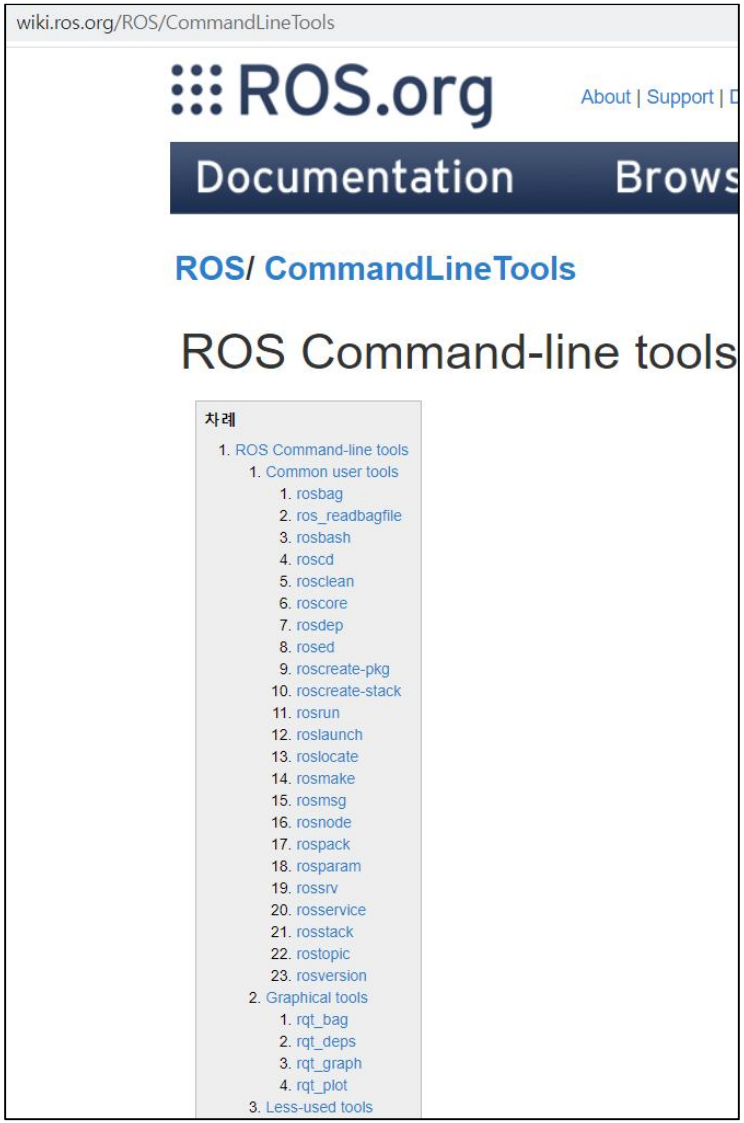
5-1. ros 명령어

: https://wiki.ros.org/ROS/CommandLineTools

: https://github.com/ros/cheatsheet/releases



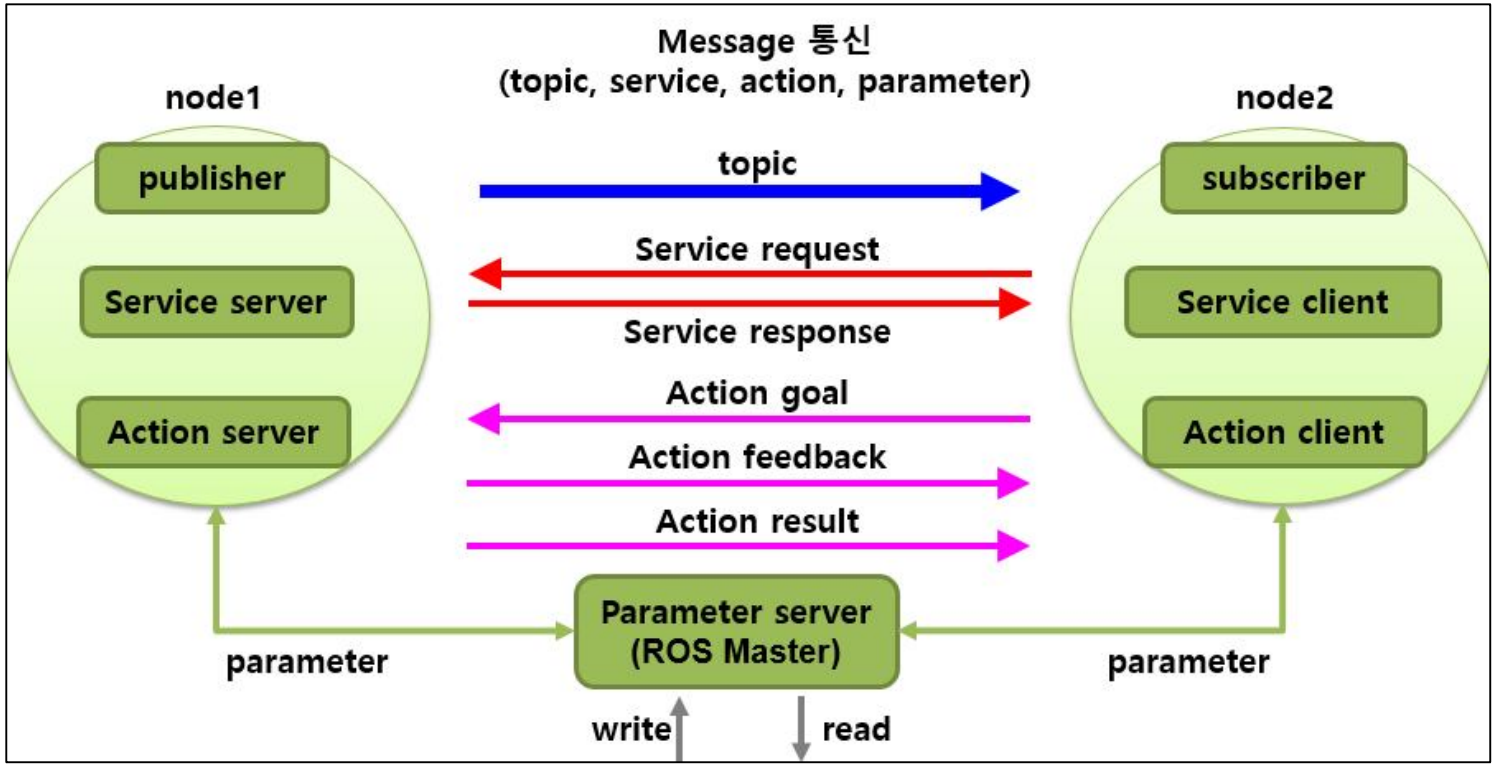
Indigo version이지만 download



5. ROS 명령어

5-1. ros 명령어

: node 와 node 간의 통신



: message는 node 간 topic을 통해 전송되는 data

- topic은 전송선로, message는 선로 안에서 전달되는 data

5. ROS 명령어

5-1. ros 명령어

: ROS는 shell 환경에서 명령어를 입력하여 파일 시스템, 소스 코드 편집, 빌드, 디버깅, 패키지 관리 등 처리

: ROS shell 명령어

명령어	중요도	명령어 풀이	세부 설명
roscd	★★★	ros+cd(changes directory)	지정한 ROS 패키지의 디렉터리로 이동
rosls	★☆☆	ros+ls(lists files)	ROS 패키지의 파일 목록 확인
rosed	★☆☆	ros+ed(editor)	ROS 패키지의 파일 편집
roscp	★☆☆	ros+cp(copies files)	ROS 패키지의 파일 복사
rospd	☆☆☆	ros+pushd	ROS 디렉터리 인덱스에 디렉터리 추가
rosd	☆☆☆	ros+directory	ROS 디렉터리 인덱스 확인

: ROS 실행 명령어

명령어	중요도	명령어 풀이	세부 설명
roscore	★★★	ros+core	master(ROS 네임 서비스) + rosout(로그 기록) + parameter server(파라미터 관리)
rosrun	★★★	ros+run	노드 실행
roslaunch	★★★	ros+launch	노드를 여러 개 실행 및 실행 옵션 설정
rosclean	★★☆	ros+clean	ROS 로그 파일을 검사하거나 삭제

5. ROS 명령어

: ROS 정보 명령어

명령어	중요도	명령어 풀이	세부 설명
rostopic	★★★	ros+topic	ROS 토픽 정보 확인
rosservice	★★★	ros+service	ROS 서비스 정보 확인
rosnode	★★★	ros+node	ROS 노드 정보 확인
rosparam	★★★	ros+param(parameter)	ROS 파라미터 정보 확인, 수정
rosbag	★★★	ros+bag	ROS 메시지 기록, 재생
rosmsg	★★☆	ros+msg	ROS 메시지 정보 확인
rossrv	★★☆	ros+srv	ROS 서비스 정보 확인
rosversion	★☆☆	ros+version	ROS 패키지 및 배포 릴리즈 버전 정보 확인
roswtf	☆☆☆	ros+wtf	ROS 시스템 검사

: ROS catkin 명령어

명령어	중요도	세부 설명
catkin_create_pkg	★★★	캐킨 빌드 시스템으로 패키지 자동 생성
catkin_make	★★★	캐킨 빌드 시스템에 기반을 둔 빌드
catkin_eclipse	★★☆	캐킨 빌드 시스템으로 생성한 패키지를 이클립스에서 사용할 수 있게 변경
catkin_prepare_release	★★☆	릴리즈할 때 사용되는 로그 정리 및 버전 태깅
catkin_generate_changelog	★★☆	릴리즈할 때 CHANGELOG.rst 파일 생성 또는 업데이트
catkin_init_workspace	★★☆	캐킨 빌드 시스템의 작업 폴더 초기화
catkin_find	★☆☆	캐킨 검색

5. ROS 명령어 -- 직접 명령어를 typing하여 실행하기

: ROS package 명령어

명령어	중요도	명령어 풀이	세부 설명
rospack	★★★	ros+pack(age)	ROS 패키지와 관련된 정보 보기
rosinstall	★★☆	ros+install	ROS 추가 패키지 설치
rosdep	★★☆	ros+dep(endencies)	해당 패키지의 의존성 파일 설치
roslocate	☆☆☆	ros+locate	ROS 패키지 정보 관련 명령어
roscreate-pkg	☆☆☆	ros+create-pkg	ROS 패키지 자동 생성(구 rosbuild 시스템에서 사용)
rosmake	☆☆☆	ros+make	ROS 패키지를 빌드(구 rosbuild 시스템에서 사용)

Linux shell
: 명령어와 프로그램 실행을 위한 인터페이스
: kernel과 사용자간의 bridge
: 검정 터미널을 의미
: bash shell 은 한 종류

5-1-1. ros shell 명령어

: ros shell 명령어는 rosbash 로 명칭

- Linux 에서 사용하는 bash shell 명령어를 ROS 개발환경에서 사용할 수 있음
- ros 접두사에 cd, pd, d, ls, ed, cp, run 등 접미사 붙여 사용

: roscd package_이름

- ROS 디렉터리 이동

roscd turtlesim

```
yongseok@yongseok:~$ roscd turtlesim
yongseok@yongseok:/opt/ros/kinetic/share/turtlesim$ ls
cmake  images  msg  package.xml  srv
yongseok@yongseok:/opt/ros/kinetic/share/turtlesim$ roscd my_first_ros_pkg
yongseok@yongseok:~/catkin_ws/src/my_first_ros_pkg$ ls
CMakeLists.txt  include  package.xml  src
yongseok@yongseok:~/catkin_ws/src/my_first_ros_pkg$
```

5. ROS 명령어

: rosls package_이름 

- ROS 파일 목록

rosls turtlesim

```
yongseok@yongseok:~$ rosls turtlesim
cmake  images  msg  package.xml  srv
yongseok@yongseok:~$
```

5-1-2. ros 실행 명령어

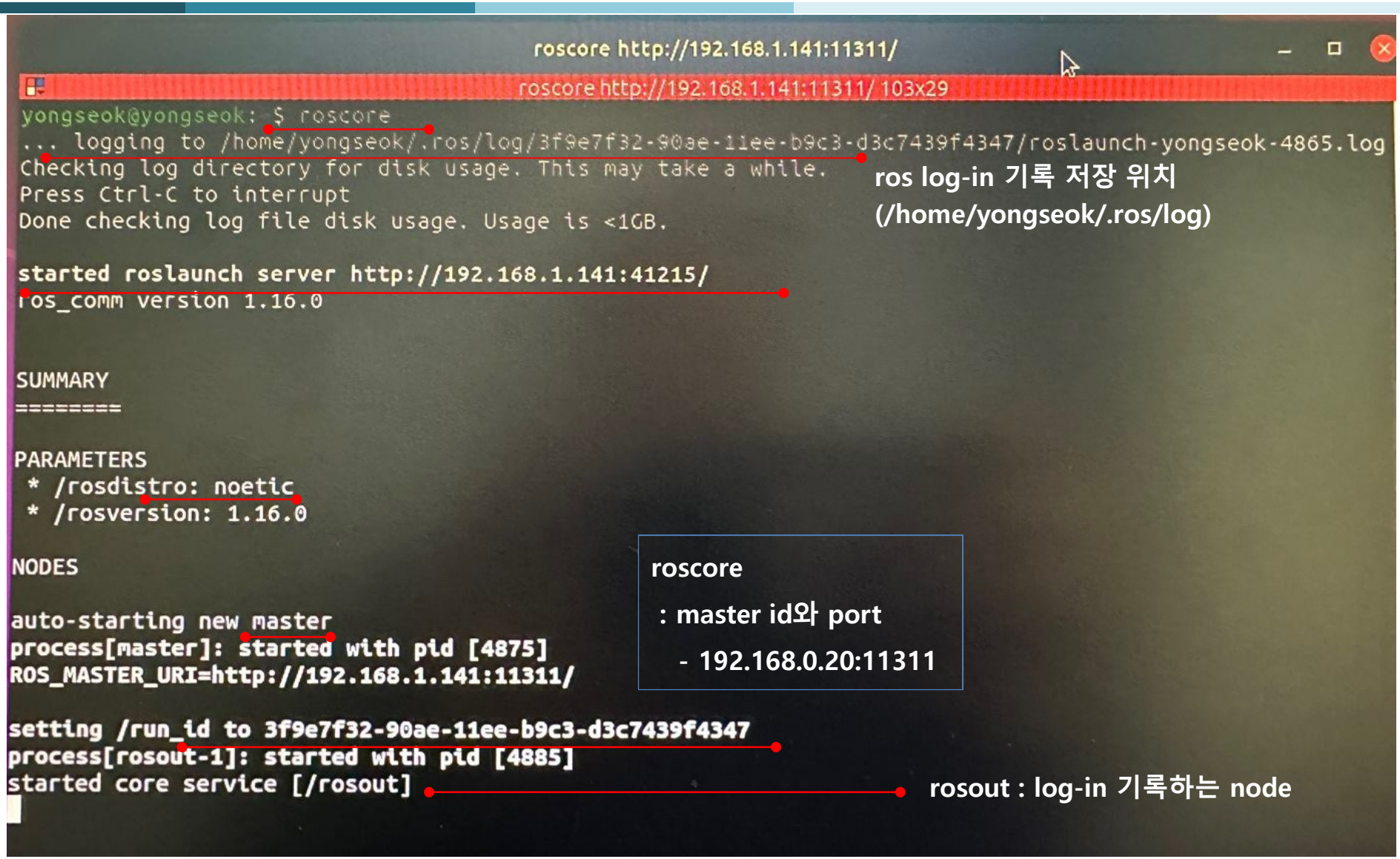
: roscore, rosrun, roslaunch, rosclean

: roscore

- ros 사용 위해 제일 먼저 구동되는 필수 요소
- roscore 실행하면, node 사이 message 통신에서 연결 정보를 관리하는 ros master 구동 (설정해 둔 ROS_MASTER_URI를 사용)
- ROS_MASTER_URI 환경설정 ~/.bashrc에서 설정 (bashrc – 사용자가 새로운 shell을 열 때마다 실행되는 shell script)
- ros master는 XMLRPC(Extensible Markup Language Remote Procedure Call) 서버로 구동하게 됨
- ros master는 node 사이의 접속을 위하여 node들의 이름, topic, service 이름, message 형태, URI 주소와 포트를 등록 받고 요청이 있을 때 정보를 다른 node에 알려주는 역할
- rosout 은 ros log-in 기록인 DEBUG, INFO, WARN, ERROR, FATAL 등 ROS 표준 출력 로그 기록을 담당
- ros master는 parameter 를 관리하는 parameter server를 실행

5. ROS 명령어

roscore



5. ROS 명령어

: rosrun package_name node_name

- rosrun은 지정한 package에서 하나의 node를 실행하는 명령어

rosrun turtlesim turtlesim_node

```
yongseok@yongseok:~$ rosrun turtlesim turtlesim_node
QXcbConnection: Failed to initialize XRandr
Qt: XKEYBOARD extension not present on the X server.
[ INFO] [1634457183.268399184]: Starting turtlesim with node name /turtlesim
[ INFO] [1634457183.281373253]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445
], theta=[0.000000]
```

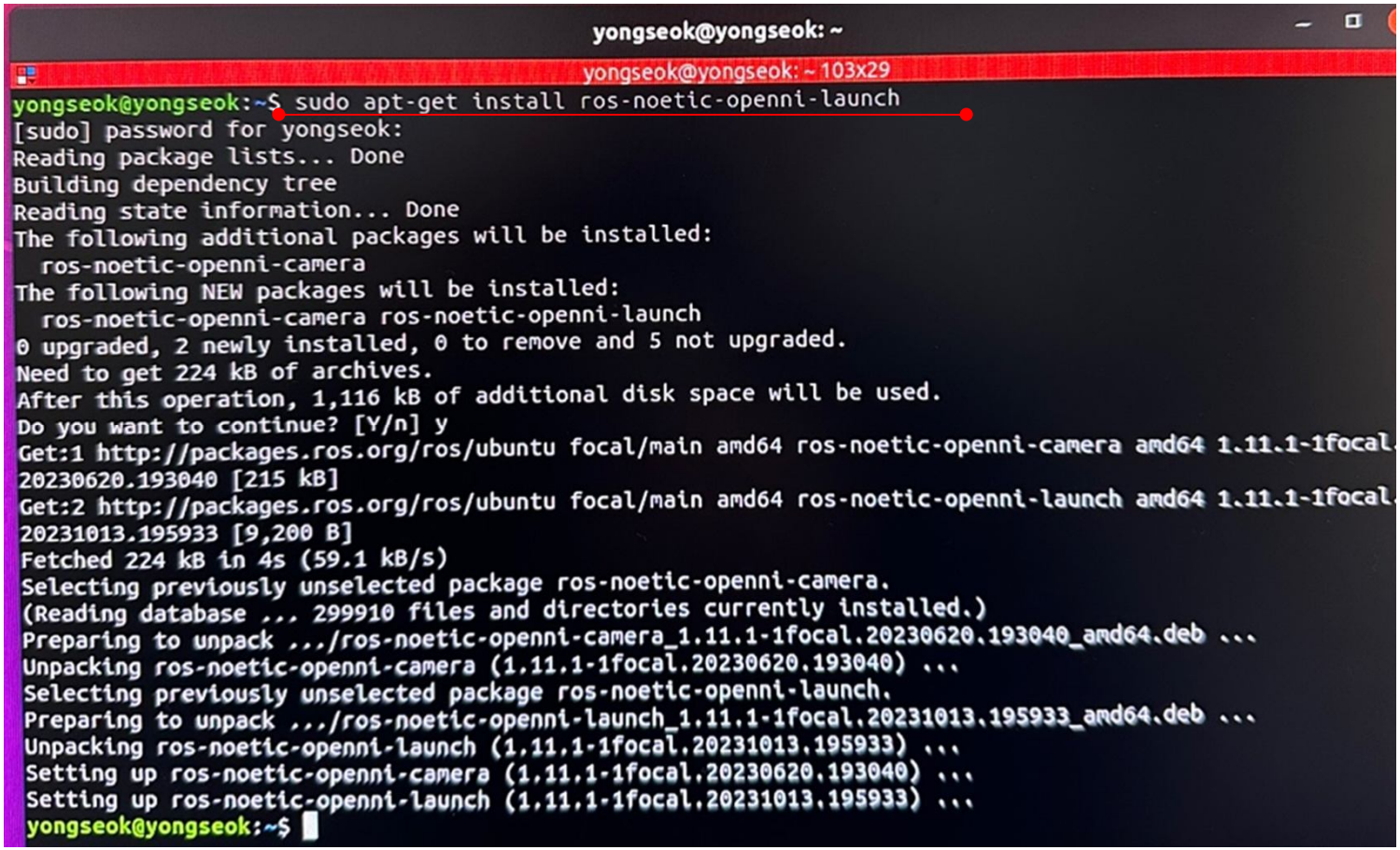
turtlesim
: turtlesim_node의 실제 node 이름

5. ROS 명령어

: roslaunch package_name node_name

- 지정한 패키지에서 하나 이상의 node를 실행하거나 실행 옵션을 설정하는 명령어
- 실습을 위해 새터미널에서 sudo apt-get install ros-noetic-openni-launch

openni : 인간과 장치간의 Natural Interaction 지원하는
오픈소스 API 및 gesture 인식을 지원하는 middleware
https://structure.io/openni
https://github.com/OpenNI



5. ROS 명령어

roslaunch openni_launch openni.launch

```
yongseok@yongseok:~$ roslaunch openni_launch openni.launch
... logging to /home/yongseok/.ros/log/15026ba2-2f1d-11ec-8ca2-f40669f0af37/roslaunch
-yongseok-5403.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://192.168.0.20:45009/

SUMMARY
========

PARAMETERS
 * /camera/camera_nodelet_manager/num_worker_threads: 4
 * /camera/depth_rectify_depth/interpolation: 0
 * /camera/depth_registered_rectify_depth/interpolation: 0
 * /camera/disparity_depth/max_range: 4.0
 * /camera/disparity_depth/min_range: 0.5
 * /camera/disparity_registered_hw/max_range: 4.0
 * /camera/disparity_registered_hw/min_range: 0.5
 * /camera/disparity_registered_sw/max_range: 4.0
 * /camera/disparity_registered_sw/min_range: 0.5
 * /camera/driver/depth_camera_info_url:
 * /camera/driver/depth_frame_id: camera_depth_opti...
 * /camera/driver/depth_registration: False
 * /camera/driver/device_id: #1
 * /camera/driver/rgb_camera_info_url:
 * /camera/driver/rgb_frame_id: camera_rgb_optica...
 * /rosdistro: kinetic
 * /rosversion: 1.12.17

NODES
  /camera/
    camera_nodelet_manager (nodelet/nodelet)
    depth_metric (nodelet/nodelet)
```

20개 이상의 node와
10여개의 parameter sever 실행

```
^C[camera_base_link3-24] killing on exit
[camera_base_link2-23] killing on exit
[camera_base_link1-22] killing on exit
[camera_base_link-21] killing on exit
[camera/disparity_registered_hw-20] killing on exit
[camera/disparity_registered_sw-19] killing on exit
[camera/depth_registered_metric-17] killing on exit
[camera/depth_registered_hw_metric_rect-16] killing on exit
[camera/disparity_depth-18] killing on exit
[camera/points_xyzrgb_hw_registered-15] killing on exit
[camera/depth_registered_rectify_depth-14] killing on exit
[camera/depth_registered_sw_metric_rect-13] killing on exit
[camera/points_xyzrgb_sw_registered-12] killing on exit
[camera/register_depth_rgb-11] killing on exit
[camera/depth_points-10] killing on exit
[camera/depth_metric-9] killing on exit
[camera/depth_metric_rect-8] killing on exit
[camera/depth_rectify_depth-7] killing on exit
[camera/ir_rectify_ir-6] killing on exit
[camera/rgb_rectify_color-5] killing on exit
[camera/rgb_rectify_mono-4] killing on exit
[camera/rgb_debayer-3] killing on exit
[camera/driver-2] killing on exit
[camera/camera_nodelet_manager-1] killing on exit
[camera/depth_metric_rect-8] escalating to SIGTERM
[camera/depth_rectify_depth-7] escalating to SIGTERM
[camera/rgb_rectify_mono-4] escalating to SIGTERM
[camera/ir_rectify_ir-6] escalating to SIGTERM
[camera/camera_nodelet_manager-1] escalating to SIGTERM
shutting down processing monitor...
... shutting down processing monitor complete
done
yongseok@yongseok:~$
```

CTL + C : 중단하기

5. ROS 명령어

: rosclean option

- ROS 로그 검사 및 삭제
- roscore 구동과 함께 모든 node에 대한 기록은 로그 파일에 기록되고 축적됨

rosclean check

```
yongseok@yongseok:~$ rosclean check
7.4M ROS node logs
yongseok@yongseok:~$
```

ROS node 로그 총 사용량 7.4MB
Roscore 실행 시 1GB 초과시 warning 발생

rosclean purge

```
yongseok@yongseok:~$ rosclean purge
Purging ROS node logs.
PLEASE BE CAREFUL TO VERIFY THE COMMAND BELOW!
Okay to perform:

rm -rf /home/yongseok/.ros/log
(y/n)?
y
```

ROS 로그 저장소
(/home/yongseok/.ros/log)의 로그를 모두 삭제

5. ROS 명령어

5-1-3. ros 정보 명령어

: ROS에서 제공하는 turtlesim 이용하여 node, topic, service 실습을 위해 아래 사전 단계 준비

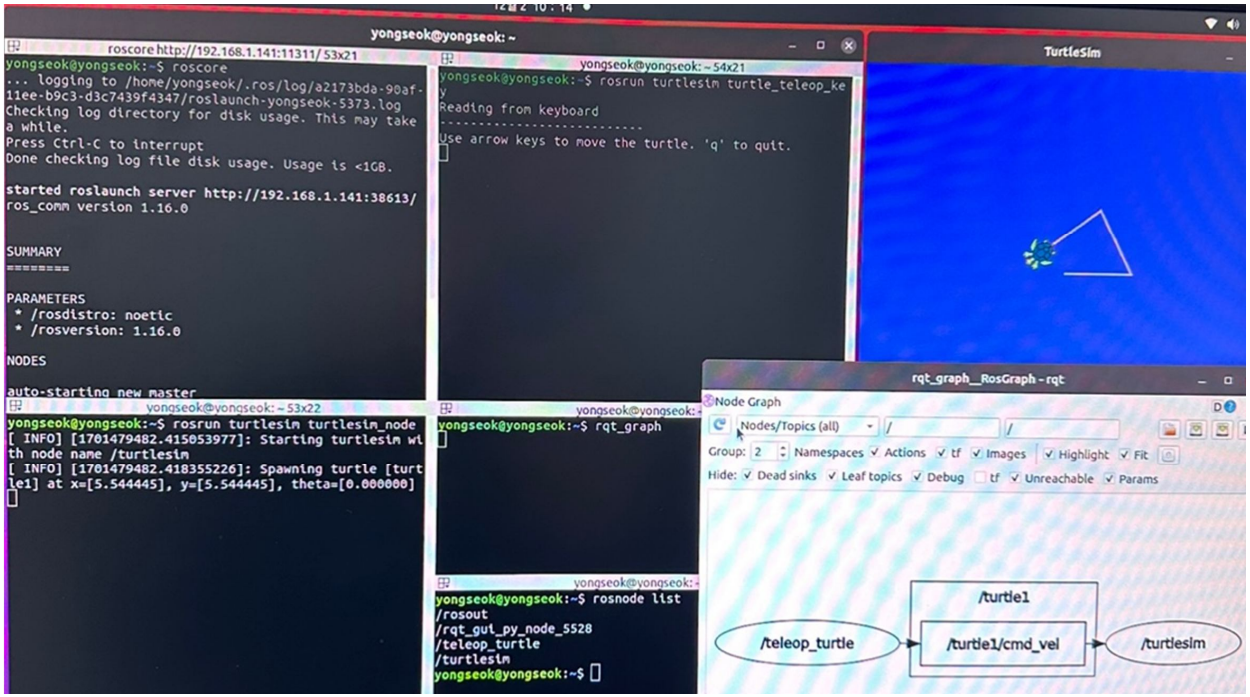
① 기존의 roscore 중지(CTL + C) 및 모든 터미널 닫기

② 새 터미널 열고 roscore

③ 새 터미널 열고 rosrun turtlesim turtlesim_node

④ 새 터미널 열고 rosrun turtlesim turtle_teleop_key

⑤ 새 터미널에서 rqt_graph



새 터미널에서 ★

: rosnode list 실행중인 node 목록 확인

```
yongseok@yongseok:~$ rosnode list
/rosout
/teleop_turtle
/turtlesim
yongseok@yongseok:~$
```

위에서 실행한 node list를 보여줌

로그 기록을 위한 rosout node

위에서 turtlesim_node 이지만,
실제 source code는 turtlesim라고 표시 설정

위에서 turtle_teleop_key 이지만,
source code에서는
ros::init(argc, argv, "teleop_turtle"); 로 설정됨

5. ROS 명령어

: rosnode ping /node 이름

- 현재 사용중인 노드와 Remote PC와의 연결 테스트
- 연결되어 있다면 해당 node로부터 XMLRPC 응답을 받게 됨

rosnode ping /turtlesim

rosnode ping /teleop_turtle

rosnode ping /rosout

XMLRPC 응답

XMLRPC(Extensible Markup Language Remote Procedure Call)
: http 를 트랜스포트로 사용해서 전달되는 XML을 사용하는 원격 procedure 호출 방법

turtlesim node와의 연결 상태 확인

roscore 실행 마다 PORT 달라짐

```
yongseok@yongseok:~$ rosnode ping /turtlesim
rosnode: node is [/turtlesim]
pinging /turtlesim with a timeout of 3.0s
xmlrpc reply from http://192.168.0.20:33007/     time=0.570059ms
xmlrpc reply from http://192.168.0.20:33007/     time=1.139879ms
xmlrpc reply from http://192.168.0.20:33007/     time=1.197100ms
xmlrpc reply from http://192.168.0.20:33007/     time=1.163960ms
xmlrpc reply from http://192.168.0.20:33007/     time=1.148939ms
^Cping average: 1.043987ms
yongseok@yongseok:~$ rosnode ping /teleop_turtle
rosnode: node is [/teleop_turtle]
pinging /teleop_turtle with a timeout of 3.0s
xmlrpc reply from http://192.168.0.20:41599/     time=0.602961ms
xmlrpc reply from http://192.168.0.20:41599/     time=1.138926ms
xmlrpc reply from http://192.168.0.20:41599/     time=1.124144ms
xmlrpc reply from http://192.168.0.20:41599/     time=1.132011ms
^Cping average: 0.999510ms
yongseok@yongseok:~$ rosnode ping /rosout
rosnode: node is [/rosout]
pinging /rosout with a timeout of 3.0s
xmlrpc reply from http://192.168.0.20:36367/     time=0.571966ms
xmlrpc reply from http://192.168.0.20:36367/     time=1.133204ms
xmlrpc reply from http://192.168.0.20:36367/     time=1.144886ms
xmlrpc reply from http://192.168.0.20:36367/     time=1.194000ms
^Cping average: 1.011014ms
yongseok@yongseok:~$
```

turtlesim id와 port
- 192.168.0.20:33007

중단 하려면 CTL + C

teleop_turtle와의 연결 상태 확인

teleop_turtle id와 port
- 192.168.0.20:41599

중단 하려면 CTL + C

rosout와의 연결 상태 확인

rosout id와 port
- 192.168.0.20:36367

중단 하려면 CTL + C

5. ROS 명령어

roscore

master
http://192.168.0.20:11311
- node정보 관리

① subscriber node 정보 outbound
- node : turtlesim
- url : http://192.168.0.20:33007
- 필요 정보 : turtle1/cmd_vel
geometry_msgs/Twist

② Publisher node 정보 outbound
- node : teleop_turtle
- url : http://192.168.0.20:41599
- 정보 : turtle1/cmd_vel
geometry_msgs/Twist

③ Publisher node 정보 outbound
- node : teleop_turtle
- url : http://192.168.0.20:41599
- 정보 : turtle1/cmd_vel
geometry_msgs/Twist

Publisher

teleop_turtle

http://192.168.0.20:41599
- 정보 전송

④ message 전송
- node : teleop_turtle
- topic : turtle1/cmd_vel

: message는 node 간 topic을 통해 전송되는 data
- topic은 전송선로, message는 선로 안에서 전달되는 data

Subscriber

turtlesim

http://192.168.0.20:33007
- 정보 수신

: rosnode info /node_이름

- 지정된 노드 정보 확인
- 기본적으로 Publications, Subscriptions, Services 등을 확인
- node 실행 URI와 topic 입출력에 대한 정보 확인

rosnode info /turtlesim

```
yongseok@yongseok:~$ rosnode info /turtlesim
--------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Services:
 * /clear
 * /kill
 * /reset
 * /spawn
 * /turtle1/set_pen
 * /turtle1/teleport_absolute
 * /turtle1/teleport_relative
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level

contacting node http://192.168.0.20:33007/ ...
Pid: 7615
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /turtle1/cmd_vel
    * to: /teleop_turtle (http://192.168.0.20:41599/)
    * direction: inbound
    * transport: TCPROS
```

node 이름

turtlesim node에서 rosout node에 Login 정보를 발행

turtlesim node에서 구독하는(필요로 하는) 구독내용(topic)

service 내용

turtlesim id와 port

rosout 으로 log-in 정보 송출(outbound)

teleop_turtle 로부터 turtle1/cmd_vel 인입(inbound)

: message는 node 간 topic을 통해 전송되는 data
- topic은 전송선로,
  message는 선로 안에서 전달되는 data

➔ topic turtle1/cmd_vel 에 전달되는 message는 geomety_msgs/Twist 임.

➔ rostopic type /turtle1/cmd_vel (message 확인)

➔ rosmsg show geomety_msgs/Twist (message 내용 확인)

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

: message는 node 간 topic을 통해 전송되는 data

- topic은 전송선로, message는 선로 안에서 전달되는 data

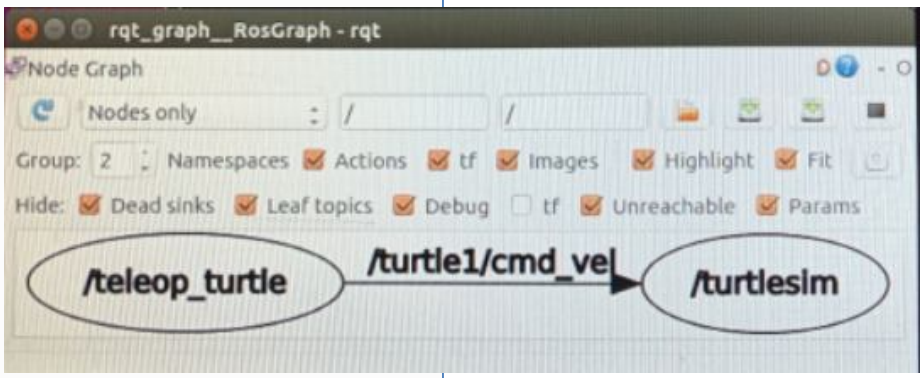
➔ topic turtle1/cmd_vel 에 전달되는 message는 geomety_msgs/Twist 임.

새 터미널 열고

➔ rostopic type /turtle1/cmd_vel (message type 확인)

➔ rosmsg show geometry_msgs/Twist (message 내용 확인)

```
yongseok@yongseok:~$ rostopic type /turtle1/cmd_vel
geometry_msgs/Twist
yongseok@yongseok:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z

yongseok@yongseok:~$
```



teleop_turtle node는 turtlesim node에게 turtle1/cmd_vel topic을 outbound 하고

turtle1/cmd_vel topic의 message type은 geomety_msgs/Twist 이며

geomety_msgs/Twist 내용은 linear, angular(선속도 m/s, 각속도 rad/s) 이다

rosnode info /teleop_turtle

```
yongseok@yongseok:~$ rosnode info /teleop_turtle
--------------------------------------------------
--
Node [/teleop_turtle]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Subscriptions: None

Services:
 * /teleop_turtle/get_loggers
 * /teleop_turtle/set_logger_level


contacting node http://192.168.0.20:41599/ ...
Pid: 7871
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /turtle1/cmd_vel
    * to: /turtlesim
    * direction: outbound
    * transport: TCPROS
```

node 이름

teleop_turtle node가 발행하는
: rosout topic 과 message type
: turtle1/cmd_vel topic과 message type

subscription 없음

teleop_turtle id와 port

rosout 으로 정보 송출(outbound)

turtlesim 으로
turtle1/cmd_vel 송출(outbound)

rosnode info /rosout

```
yongseok@yongseok:~$ rosnode info /rosout
--------------------------------------------------------
--
Node [/rosout]
Publications:
 * /rosout_agg [rosgraph_msgs/Log]

Subscriptions:
 * /rosout [rosgraph_msgs/Log]

Services:
 * /rosout/get_loggers
 * /rosout/set_logger_level


contacting node http://192.168.0.20:36367/ ...
Pid: 7386
Connections:
 * topic: /rosout
    * to: /turtlesim (http://192.168.0.20:33007/)
    * direction: inbound
    * transport: TCPROS
 * topic: /rosout
    * to: /teleop_turtle (http://192.168.0.20:41599/)
    * direction: inbound
    * transport: TCPROS
```

node 이름

rosout node가 발행하는
topic 과 message type

subscription

teleop_turtle id와 port

Teleop_turtle 으로부터 정보
인입(inbound)

turtlesim 으로부터 정보 인입(inbound)

: rosnode machine remote_PC_ID

- 해당 PC에서 실행되고 있는 노드 목록

rosnode machine 192.168.0.20

```
yongseok@yongseok:~$ rosnode machine 192.168.0.20
/rosout
/teleop_turtle
/turtlesim
yongseok@yongseok:~$
```

: rosnode kill /node_이름

- 실행 중인 노드를 종료하는 명령어 = node가 실행중인 터미널 창에서 [Ctrl+c]

rosnode kill /teleop_turtle

```
yongseok@yongseok:~$ rosnode kill /teleop_turtle
killing /teleop_turtle
killed
yongseok@yongseok:~$
```

: rosnode cleanup

- node가 비정상 종료되었을 때, 이 명령어로 연결 정보가 끊어진 node를 목록에서 삭제

rosnode cleanup

```
yongseok@yongseok:~$ rosnode cleanup
yongseok@yongseok:~$
```

: ROS에서 제공하는 turtlesim 이용하여 node, topic, service 실습을 위해 아래 사전 단계 준비

① 기존의 roscore 중지(CTL + C) 및 모든 터미널 닫기 ② 새 터미널 열고 roscore

③ 새 터미널 열고 rosrun turtlesim turtlesim_node ④ 새 터미널 열고 rosrun turtlesim turtle_teleop_key

새 터미널에서

: rostopic -h - 두 node 사이의 통신에 사용되는 topic 나타냄 (-h는 option)

실행해 보기

```
yongseok@yongseok:~$ rostopic -h
rostopic is a command-line tool for printing information about ROS Topics.

Commands:
        rostopic bw     display bandwidth used by topic
        rostopic delay  display delay of topic from timestamp in header
        rostopic echo   print messages to screen
        rostopic find   find topics by type
        rostopic hz     display publishing rate of topic
        rostopic info   print information about active topic
        rostopic list   list active topics
        rostopic pub    publish data to topic
        rostopic type   print topic or field type

Type rostopic <command> -h for more detailed usage, e.g. 'rostopic echo -h'
```

: rostopic list - 활성화된 topic 목록 표시

```
yongseok@yongseok:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
yongseok@yongseok:~$
```

: rostopic list –v - publish topic, subscribe topic 구분

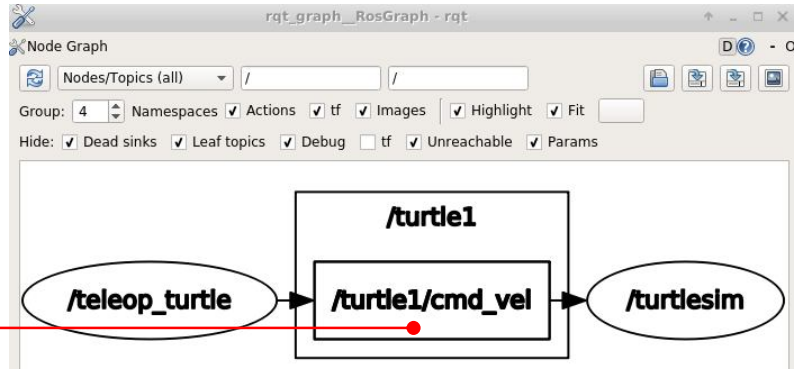
```
yongseok@yongseok:~$ rostopic list -v

Published topics:
 * /turtle1/color_sensor [turtlesim/Color] 1 publisher
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 publisher
 * /rosout [rosgraph_msgs/Log] 2 publishers
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher
 * /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber
 * /rosout [rosgraph_msgs/Log] 1 subscriber
```

새 터미널에서

: rostopic echo /topic_이름 - 지정한 토픽의 메시지 내용 실시간 표시(현재 발행되는 data 확인)

- 먼저 새터미널에서 rqt_graph 실행하여 topic 확인하기

```
yongseok@yongseok:~$ rqt_graph
QXcbConnection: Failed to initialize XRandr
Qt: XKEYBOARD extension not present on the X server.
```



topic : /turtle1/cmd_vel 확인

새터미널에서 rostopic echo /turtle1/cmd_vel 실행하기

```
yongseok@yongseok:~$ rostopic echo /turtle1/cmd_vel
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

turtle1/cmd_vel topic
message type은 geomety_msgs/Twist 이며
geomety_msgs/Twist 내용은 linear, angular

Linear, angular 값이 초기에는 변화가 없으며, 0.0 값을 나타냄

rostopic echo /turtle1/cmd_vel 실행한 터미널 값이 변함

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

Linear, angular 값이 변함

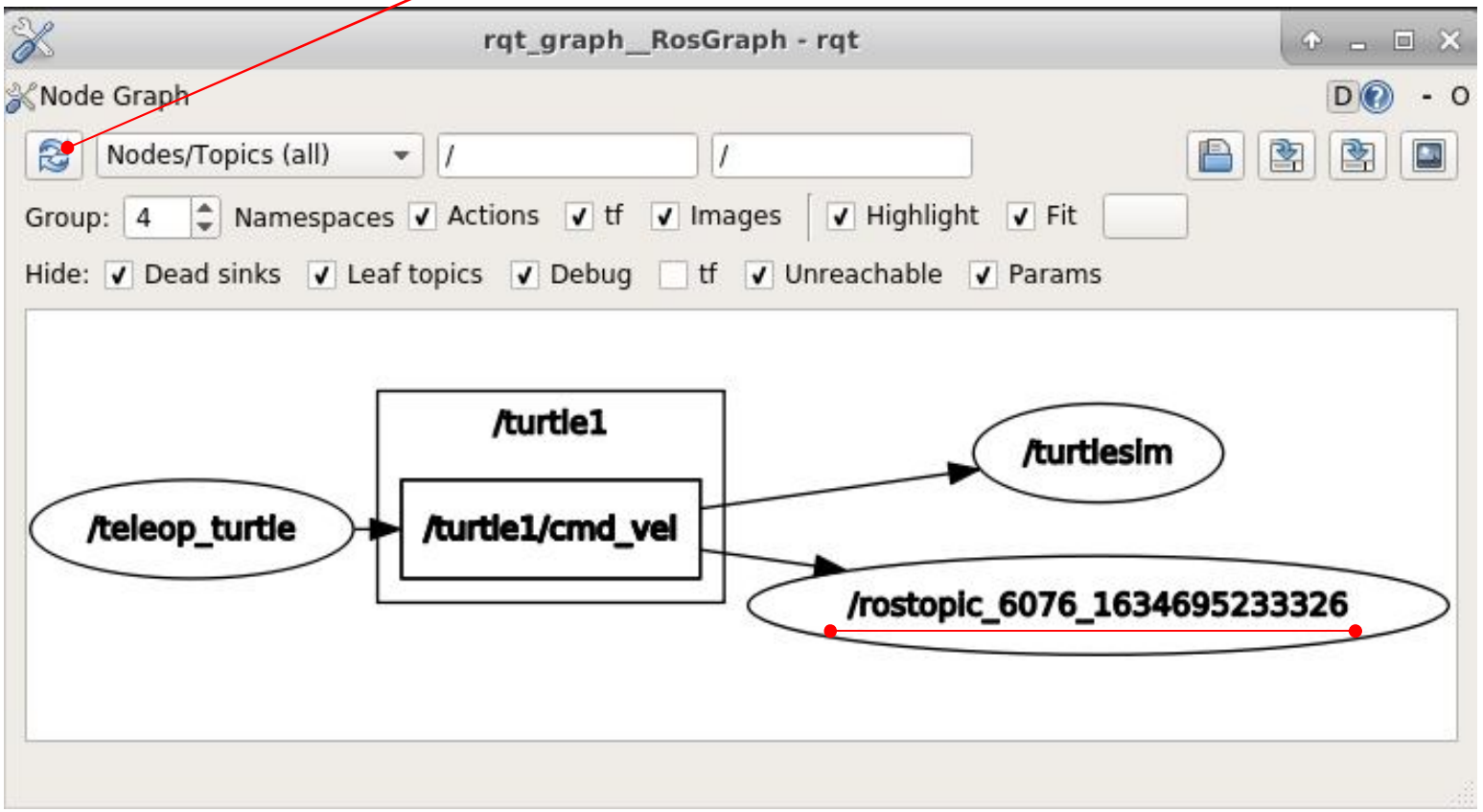
x, y : 0.02m/s 속도,
z : 2rad/s 속도로 회전

rosrun turtlesim turtle_teleop_key를 실행한 터미널을 클릭 후

화살표(up, down, right, left)를 움직이기

```
yongseok@yongseok: ~
File Edit View Search Terminal Help
yongseok@yongseok:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

rqt_graph 실행화면에서 refresh 누르기



turtle1/cmd_vel 라는 topic을 받는 node에 새로운 rostopic 이 추가 됨을 확인

➔ Topic은 1:1 형식이 아닌 1:N 등의 형식을 갖음

5. ROS 명령어

새 터미널에서 ☆

: rostopic pub /topic_name message_type parameter

- 지정한 topic_이름으로 message를 published

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

/turtle1/cmd_vel
: topic name

geometry_msgs/Twist
: published 되는 message type

/turtle1/cmd_vel 의 message가 ['linear', 'angular'] 형태
➔ '[x, y, z]' '[x, y, z'] 형태로 입력
➔ '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
: x축 좌표로 초당 2.0m의 속도 이동,
z축 중심으로 초당 1.8rad 회전하는 값

-1 : 메시지를 한 번만 publish
: 실제로 한 번 실행되기는 하지만 3초간 실행

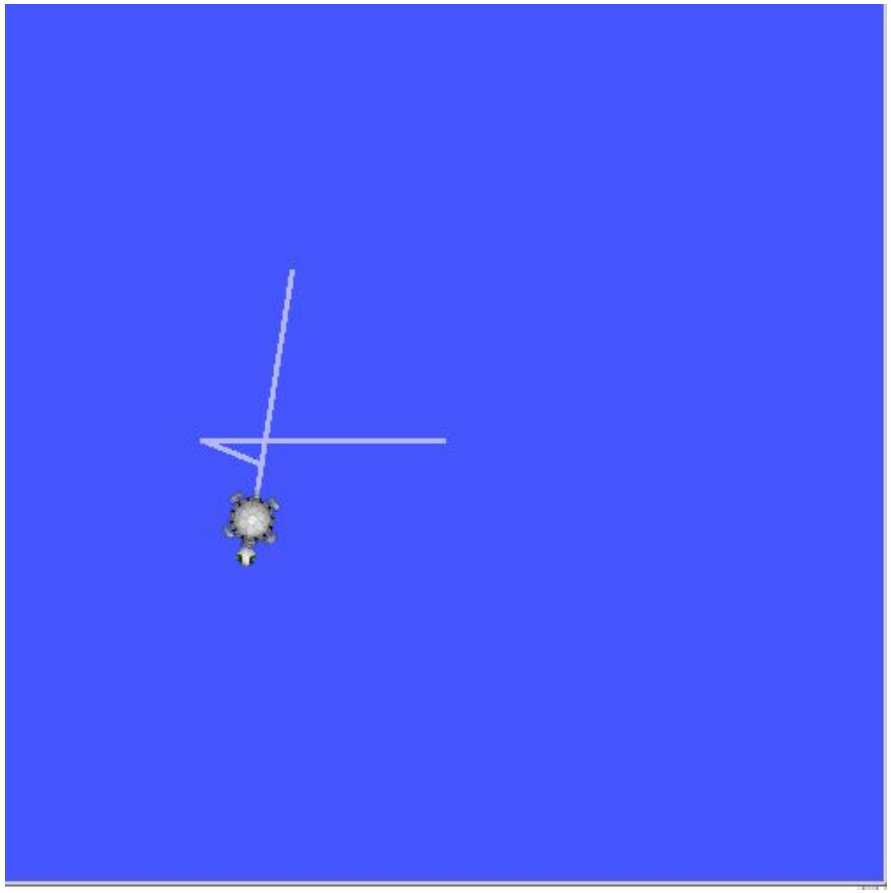
rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

띄어 쓰기

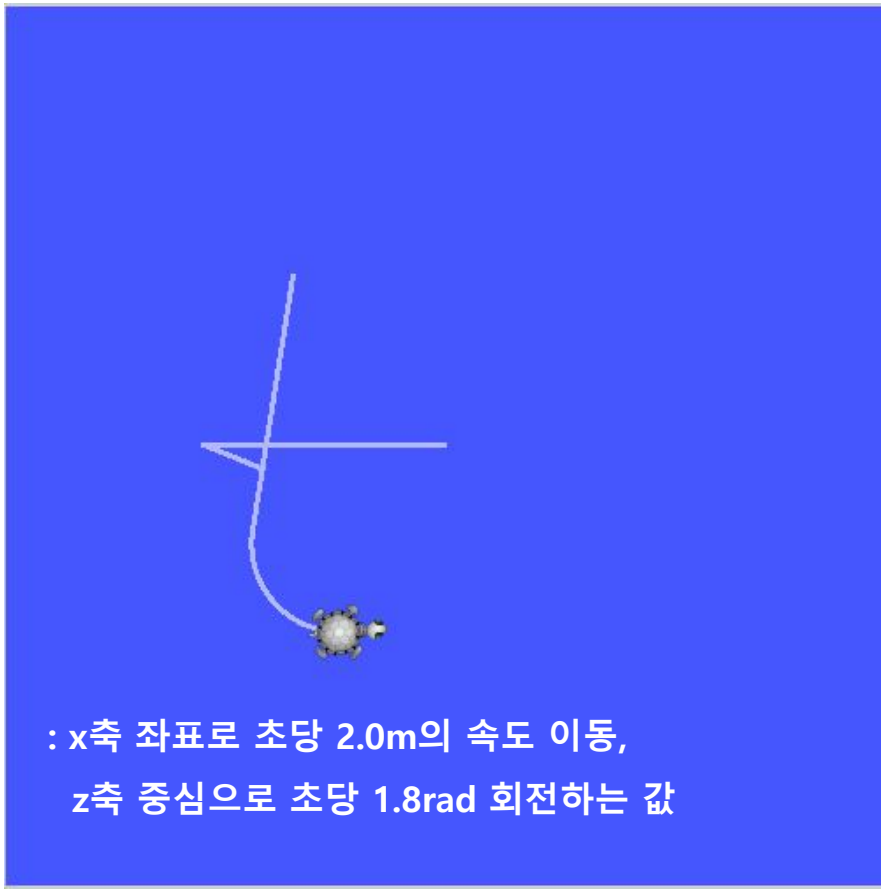
5. ROS 명령어

```
yongseok@yongseok:~$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2
.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
publishing and latching message for 3.0 seconds
yongseok@yongseok:~$
```

AS IS



TO BE



: x축 좌표로 초당 2.0m의 속도 이동,
z축 중심으로 초당 1.8rad 회전하는 값

5. ROS 명령어

Ref. https://www.ros.org/reps/rep-0103.html

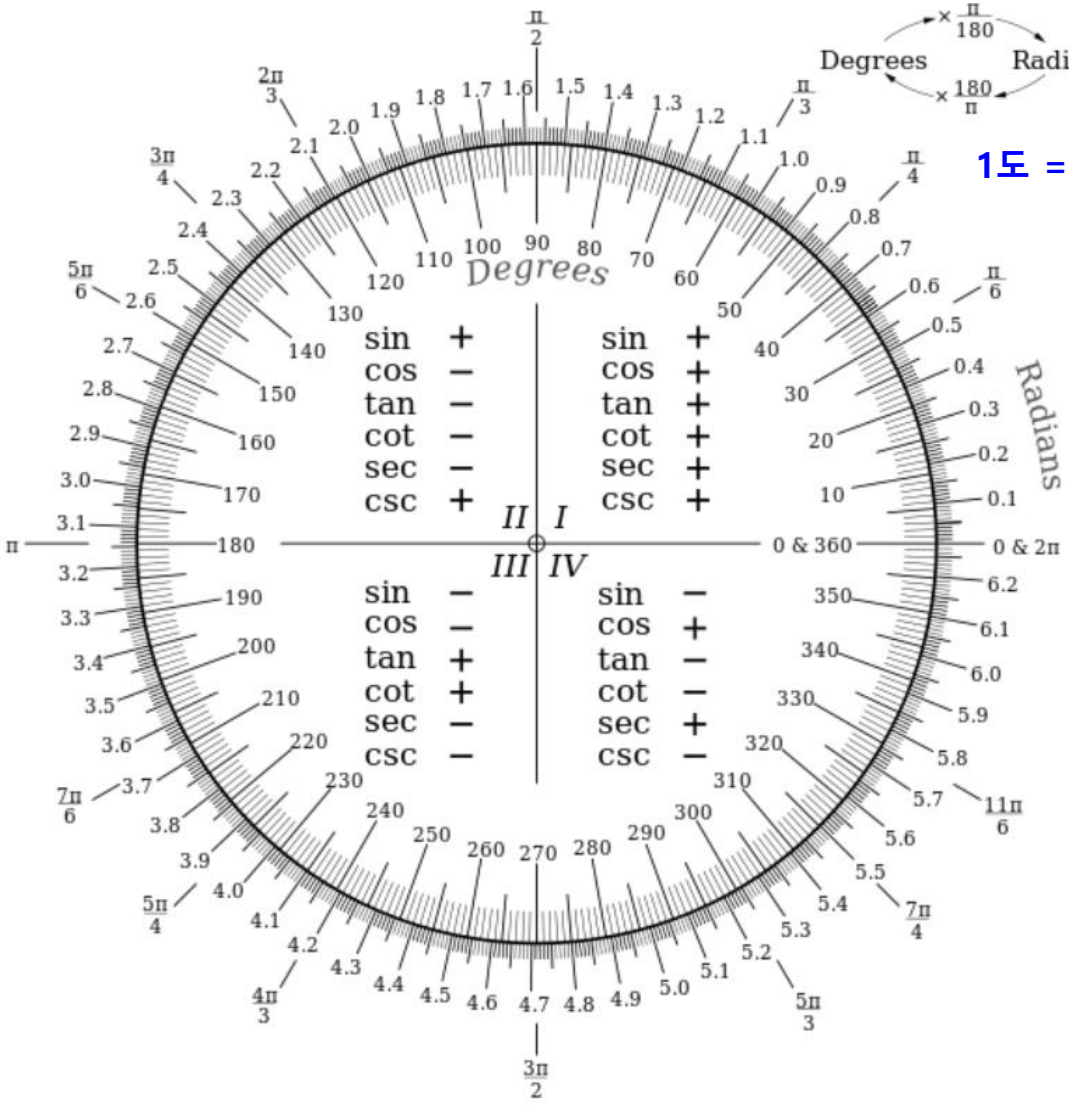
※ ros 표준 단위

: 표준 단위 SI 권장. REP-01031 명시함

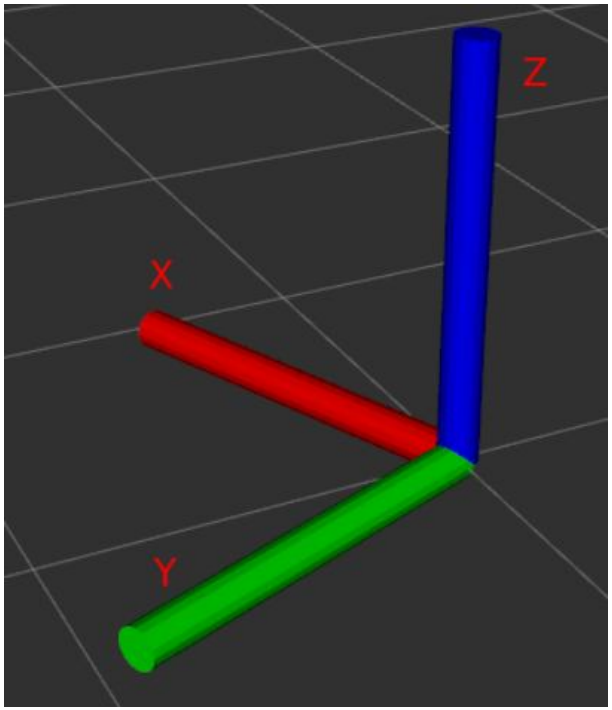
- https://www.ros.org/reps/rep-0103.html

Quantity	Unit
length	meter
mass	kilogram
time	second
current	ampere

Quantity	Unit
angle	radian
frequency	hertz
force	newton
power	watt
voltage	volt
temperature	celsius
magnetism	tesla



5. ROS 명령어



※ ros 표준 단위

: 회전축 x, y, z

x forward (정면 : x +방향)

y left (왼쪽 : y +방향)

z up (위쪽: z +방향)

: turtlebot burger 회전 방향

- 오른손의 법칙(오른손 감는 방향이 + 방향)

: ROS에서 제공하는 turtlesim 이용하여 service 실습을 위해 아래 사전 단계 준비

① 기존의 roscore 중지(CTL + C) 및 모든 터미널 닫기 ② 새 터미널 열고 roscore

③ 새 터미널 열고 rosrun turtlesim turtlesim_node ④ 새 터미널 열고 rosrun turtlesim turtle_teleop_key

새 터미널에서

: rosservice -h - service option 나타냄 (-h는 option)

```
yongseok@yongseok:~$ rosservice -h
Commands:
        rosservice args print service arguments
        rosservice call call the service with the provided args
        rosservice find find services by service type
        rosservice info print information about service
        rosservice list list active services
        rosservice type print service type
        rosservice uri  print service ROSRPC uri

Type rosservice <command> -h for more detailed usage, e.g. 'rosserv
```

: rosservice list

- 활성화된 서비스에 대한 정보를 출력

```
yongseok@yongseok:~$ rosservice list
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/rostopic_6076_1634695233326/get_loggers
/rostopic_6076_1634695233326/set_logger_level
/rqt_gui_py_node_6021/get_loggers
/rqt_gui_py_node_6021/set_logger_level
/spawn
/teleop_turtle/get_loggers
/teleop_turtle/set_logger_level
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/get_loggers
/turtlesim/set_logger_level
yongseok@yongseok:~$
```

5. ROS 명령어

파라미터(r, g, b, width, off) off는 0(false)이므로 선이 보이도록 함

: rosservice info /서비스_이름 : 지정한 서비스의 정보 표시

rosservice info /turtle1/set_pen

: rosservice type 서비스_이름 : 지정한 서비스 타입 출력

rosservice type turtle1/set_pen

: rosservice uri /서비스_이름 : ROSRPC uri 서비스 출력

rosservice uri /turtle1/set_pen

: rosservice args /서비스_이름 : 서비스 parameter 출력

rosservice args /turtle1/set_pen

```
yongseok@yongseok:~$ rosservice info /turtle1/set_pen
Node: /turtlesim
URI: rosrpc://192.168.0.20:47009
Type: turtlesim/SetPen
Args: r g b width off
yongseok@yongseok:~$ rosservice type turtle1/set_pen
turtlesim/SetPen
yongseok@yongseok:~$ rosservice uri /turtle1/set_pen
rosrpc://192.168.0.20:47009
yongseok@yongseok:~$ rosservice args /turtle1/set_pen
r g b width off
yongseok@yongseok:~$
```

r, g, b, width, off 파라미터

/turtle1/set_pen 서비스의 ROSRPC URI

5. ROS 명령어

: rosservice call /서비스_이름 parameter : 입력된 parameter로 서비스 요청

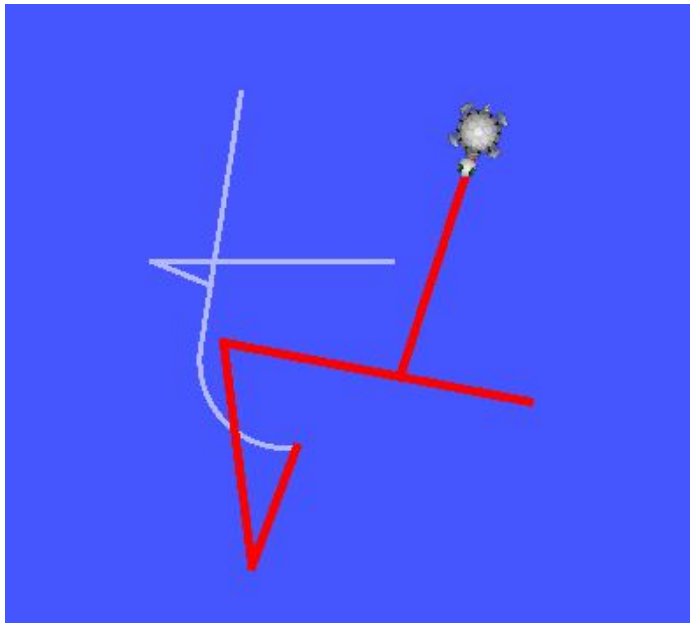
rosservice call /turtle1/set_pen 255 0 0 5 0

- /turtle1/set_pen 서비스를 요청하는 명령어
- 사용된 '255 0 0 5 0'은 /turtle1/set_pen 서비스에 사용되는 파라미터(r, g, b, width, off) 수치
- 빨간색 r은 최대치인 255, g, b는 모두 0이므로 펜의 색은 빨간색
- width는 5의 굵기로 설정, off는 0(false)이므로 선이 보이도록 함

```
yongseok@yongseok:~$ rosservice call /turtle1/set_pen 255 0 0 5 0

yongseok@yongseok:~$
```

rosrun turtlesim turtle_teleop_key를 실행한 터미널을 클릭 후
화살표(up, down, right, left)를 움직이기

```
yongseok@yongseok: ~
File  Edit  View  Search  Terminal  Help
yongseok@yongseok:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```



5. ROS 명령어

① 기존의 roscore 중지(CTL + C) 및 모든 터미널 닫기 ② 새 터미널 열고 roscore

③ 새 터미널 열고 rosrun turtlesim turtlesim_node ④ 새 터미널 열고 rosrun turtlesim turtle_teleop_key

새 터미널에서

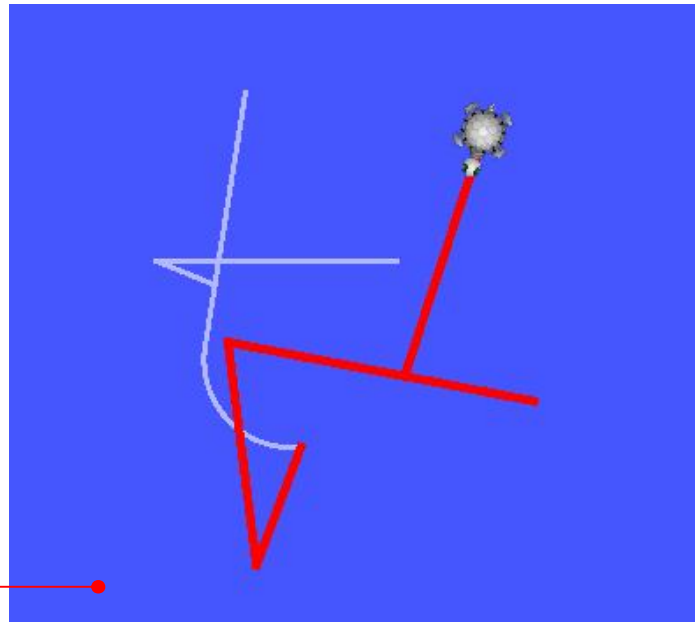
: rosparam list - 같은 네트워크에서 사용 중인 파라미터 목록 표시

```
yongseok@yongseok:~$ rosparam list
/background_b
/background_g
/background_r
/rosdistro
/roslaunch/uris/host_192_168_0_20__36559
/rosversion
/run_id
yongseok@yongseok:~$
```

: rosparam get /파라미터_이름 - 파라미터 값 불러오기

rosparam get /background_b

```
yongseok@yongseok:~$ rosparam get /background_b
255
```



5. ROS 명령어

: rosparam get / - 모든 파라미터의 값을 확인

```
yongseok@yongseok:~$ rosparam get /
background_b: 255
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_192_168_0_20__36559: 'http://192.168.0.20:36559/'}
rosversion: '1.12.17

  '
run_id: b5b10df2-3142-11ec-8dd3-f40669f0af37

yongseok@yongseok:~$
```

5. ROS 명령어

추후 카메라 세팅 값 저장 방법

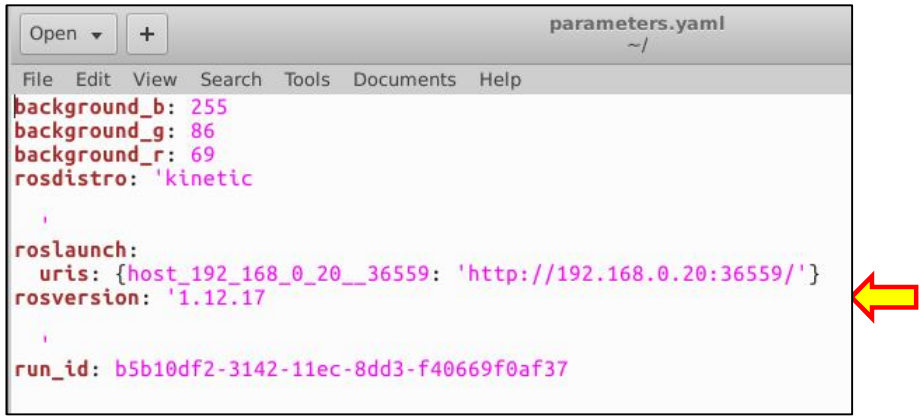
: rosparam dump ~파일_이름 - 파라미터를 지정한 파일에 저장 ~/ 사용자 home 폴더

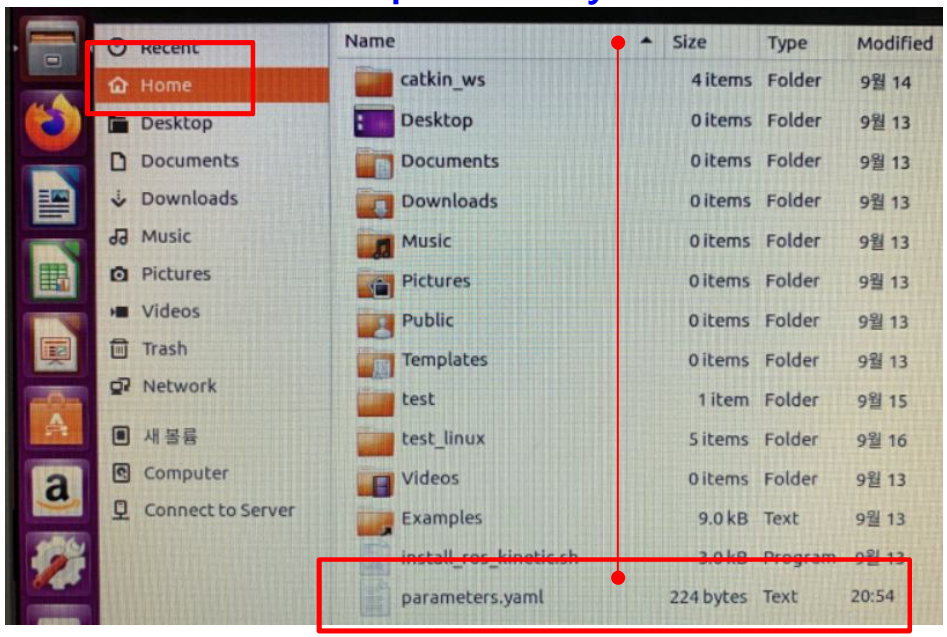
rosparam dump ~/parameters.yaml - 현재 파라미터 값을 parameters.yaml 파일에 저장하는 명령어

```
yongseok@yongseok:~$ rosparam get /
background_b: 255
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_192_168_0_20__36559: 'http://192.168.0.20:36559/'}
rosversion: '1.12.17

  '
run_id: b5b10df2-3142-11ec-8dd3-f40669f0af37

yongseok@yongseok:~$ rosparam dump ~/parameters.yaml
yongseok@yongseok:~$
```

현재 모든 parameters를 parameters.yaml 파일에 저장

사용자 home 폴더의 parameters.yaml을 열어서 확인





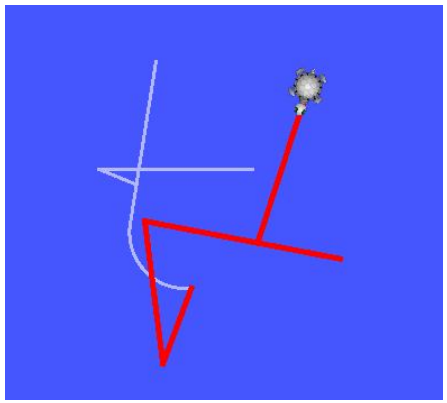
5. ROS 명령어

: rosparam set 파라미터_이름 - 파라미터 값 설정

- turtlesim 노드의 배경색 관련 파라미터 background_b = 0으로 설정

rosparam set background_b 0

```
yongseok@yongseok:~$ rosparam get /
background_b: 255
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_192_168_0_20__36559: 'http://192.168.0.20:36559/'}
rosversion: '1.12.17

  '
run_id: b5b10df2-3142-11ec-8dd3-f40669f0af37

yongseok@yongseok:~$ rosparam set background_b 0
yongseok@yongseok:~$
```



실행했지만,

turtlesim 바탕은 아직 변화지 않음

rosservice call clear

```
yongseok@yongseok:~$ rosparam set background_b 0
yongseok@yongseok:~$ rosservice call clear

yongseok@yongseok:~$ rosparam get /
background_b: 0
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_192_168_0_20__36559: 'http://192.168.0.20:36559/'}
rosversion: '1.12.17

  '
run_id: b5b10df2-3142-11ec-8dd3-f40669f0af37
```

turtlesim 화면 갱신

B = 0

rosservice call clear 명령어로 turtlesim 화면 갱신해야 변함

그러나 parameters.yaml 파일에 반영되지 않음

5. ROS 명령어

: rosparam load ~/파일_이름 - 파라미터를 불러온다

- rosparam dump와 반대로 parameters.yaml 파일 불러, 현재의 파라미터 값으로 사용

rosparam load ~/parameters.yaml

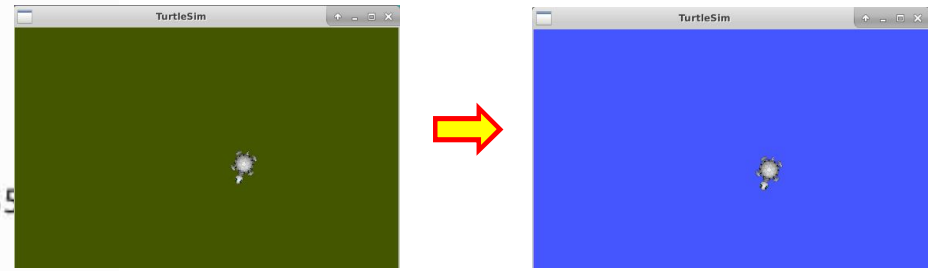
rosservice call clear

```
yongseok@yongseok:~$ rosparam load ~/parameters.yaml
yongseok@yongseok:~$ rosservice call clear

yongseok@yongseok:~$ rosparam get /
background_b: 255
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_192_168_0_20__36559: 'http://192.168.0.20:3655
rosversion: '1.12.17

  '
run_id: b5b10df2-3142-11ec-8dd3-f40669f0af37
```

parameters.yaml 파일 불러와서 사용하기

turtlesim 화면 갱신

현재의 파라미터 값으로 반영됨



그외

: rosparam delete /background_b - 지정된 파라미터 삭제 명령어

5. ROS 명령어

① 기존의 roscore 중지(CTL + C) 및 모든 터미널 닫기

② 새 터미널 열고 roscore

③ 새 터미널 열고 rosrun turtlesim turtlesim_node

④ 새 터미널 열고 rosrun turtlesim turtle_teleop_key

새 터미널에서

: rosmsg list - 모든 메시지 목록 표시(ROS에 설치된 패키지의 모든 메시지를 목록으로 표시)

```
yongseok@yongseok:~$ rosmsg list
actionlib/TestAction
actionlib/TestActionFeedback
actionlib/TestActionGoal
actionlib/TestActionResult
actionlib/TestFeedback
actionlib/TestGoal
actionlib/TestRequestAction
actionlib/TestRequestActionFeedback
actionlib/TestRequestActionGoal
actionlib/TestRequestActionResult
actionlib/TestRequestFeedback
actionlib/TestRequestGoal
actionlib/TestRequestResult
actionlib/TestResult
actionlib/TwoIntsAction
actionlib/TwoIntsActionFeedback
actionlib/TwoIntsActionGoal
actionlib/TwoIntsActionResult
actionlib/TwoIntsFeedback
actionlib/TwoIntsGoal
actionlib/TwoIntsResult
```

내용 생략

```
turtlebot3_example/Turtlebot3Feedback
turtlebot3_example/Turtlebot3Goal
turtlebot3_example/Turtlebot3Result
turtlebot3_msgs/SensorState
turtlebot3_msgs/Sound
turtlebot3_msgs/VersionInfo
turtlesim/Color
turtlesim/Pose
uuid_msgs/UniqueID
variant_msgs/Test
variant_msgs/Variant
variant_msgs/VariantHeader
variant_msgs/VariantType
visualization_msgs/ImageMarker
visualization_msgs/InteractiveMarker
visualization_msgs/InteractiveMarkerControl
visualization_msgs/InteractiveMarkerFeedback
visualization_msgs/InteractiveMarkerInit
visualization_msgs/InteractiveMarkerPose
visualization_msgs/InteractiveMarkerUpdate
visualization_msgs/Marker
visualization_msgs/MarkerArray
visualization_msgs/MenuEntry
yongseok@yongseok:~$
```

5. ROS 명령어

: rosmsg show message_이름 - 지정한 메시지 정보 표시

- rosmsg list 에서 표시된 msgs중에서 turtlesim/Pose 메시지 정보를 출력

```
turtlebot3_msgs/Sound
turtlebot3_msgs/VersionInfo
turtlesim/Color
turtlesim/Pose
uuid_msgs/UniqueID
variant_msgs/Test
```

rosmsg show turtlesim/Pose

```
yongseok@yongseok:~$ rosmsg show turtlesim/Pose
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity

yongseok@yongseok:~$
```

Float32 부동소수점형 변수로 x, y, theta, linear_velocity, angular_velocity 정보가 포함된 message

rosmsg package turtlesim

```
yongseok@yongseok:~$ rosmsg package turtlesim
turtlesim/Color
turtlesim/Pose
```

- turtlesim 패키지에서 사용되는 메시지 목록

rosmsg packages - 메시지를 사용하는 모든 패키지 목록

```
yongseok@yongseok:~$ rosmsg packages
actionlib
actionlib_msgs
actionlib_tutorials
ar_track_alvar_msgs
base_local_planner
bhand_controller
bond
control_msgs
controller_manager_msgs
costmap_2d
diagnostic_msgs
dynamic_reconfigure
```

내용 생략

```
tf
tf2_msgs
theora_image_transport
trajectory_msgs
turtle_actionlib
turtlebot3_applications_msgs
turtlebot3_example
turtlebot3_msgs
turtlesim
uuid_msgs
variant_msgs
visualization_msgs
yongseok@yongseok:~$
```

5. ROS 명령어

5-2. ros catkin 명령어

: ROS catkin 명령어는 catkin 빌드 시스템을 사용하여 package를 빌드할 때 사용 

명령어	중요도	세부 설명
catkin_create_pkg	★★★	캐킨 빌드 시스템으로 패키지 자동 생성
catkin_make	★★★	캐킨 빌드 시스템에 기반을 둔 빌드
catkin_eclipse	★★☆	캐킨 빌드 시스템으로 생성한 패키지를 이클립스에서 사용할 수 있게 변경
catkin_prepare_release	★★☆	릴리즈할 때 사용되는 로그 정리 및 버전 태깅
catkin_generate_changelog	★★☆	릴리즈할 때 CHANGELOG.rst 파일 생성 또는 업데이트
catkin_init_workspace	★★☆	캐킨 빌드 시스템의 작업 폴더 초기화
catkin_find	★☆☆	캐킨 검색

강의자료 3~4장

강의자료 3장

catkin_eclipse는 통합개발환경(IDE) Eclipse를 사용하여 package환경 구축

5. ROS 명령어

: package 생성 명령어 catkin_create_pkg

- catkin_create_pkg 는

  사용자 package 작성할 때 catkin 빌드 시스템에 필요한 CMakeLists.txt와 package.xml 포함한 package 폴더 생성

$ catkin_create_pkg my_first_ros_pkg std_msgs roscpp

package 이름

ROS 표준 메시지 package

C/C++를 위한 Client library

의존성 package

: catkin 빌드 시스템에 기반을 둔 명령어 catkin_make

- catkin_make 는 사용자가 만든 package 또는 다운로드한 package를 빌드하는 명령어
- ~/catkin_ws/src 폴더에 있는 모든 package를 빌드
- 그러나 build 위치는 ~/catkin_ws

  $ cd ~/catkin_ws && catkin_make
- 모든 package가 아닌 일부 package 빌드 : --pkg 패키지_이름 지정

  $ catkin_make --pkg user_ros_tutorials

5. ROS 명령어

: catkin 검색, 작업 공간 표시 catkin_find

- 프로젝트 작업 폴더를 표시하는 명령어

catkin_find

```
yongseok@yongseok:~$ catkin_find
/home/yongseok/catkin_ws/devel/include
/home/yongseok/catkin_ws/devel/lib
/home/yongseok/catkin_ws/devel/share
/opt/ros/kinetic/bin
/opt/ros/kinetic/etc
/opt/ros/kinetic/include
/opt/ros/kinetic/lib
/opt/ros/kinetic/share
yongseok@yongseok:~$
```

프로젝트 작업 폴더를 표시

: catkin 검색, 작업 공간 표시

- catkin_find 패키지_이름

catkin_find turtlesim

```
yongseok@yongseok:~$ catkin_find turtlesim
/opt/ros/kinetic/include/turtlesim
/opt/ros/kinetic/lib/turtlesim
/opt/ros/kinetic/share/turtlesim
yongseok@yongseok:~$
```

turtlesim 작업 폴더를 표시

감사합니다