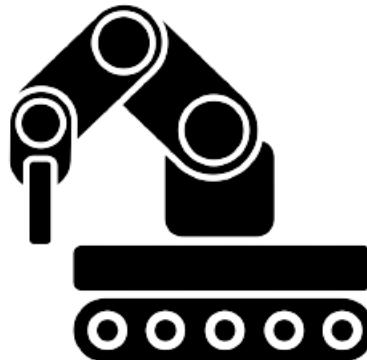
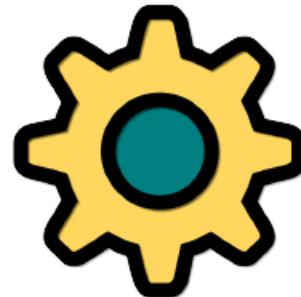




ROS2 커리큘럼

Basic tutorial

ROS



RLmodel
ybbaek



Contents

1. ROS 개요
2. ROS1 & ROS2 차이
3. ROS2
4. Turtlesim
5. dummy robot
6. Gazebo
7. Python code
8. Appendix
9. Linux 명령어



ROS개요

로봇개발의 문제점

기존 로봇 개발 방식과 한계

- 하드웨어 설계, 제어부터 제어기, 비전, 네비게이션 등 모든것을 개발해야 함.
- API마다의 interface가 다르고, 적용하는데 학습이 필요함.
- 하드웨어에 의존적인 소프트웨어적 성격때문에 로봇이 달라지면 소프트웨어 또한 수정이 필요함.
- 소프트웨어를 작성하는데 하드웨어에 대한 지식이 필요함.

ROS개요

ROS란

ROS는 로봇용 공개소스 메타 운영체제

ROS는 일반 운영체제에서 제공하는 하드웨어 추상화, 저수준 기기 제어, 빈번히 사용되는 기능들이 구현되어 있으며, 프로세스간 메시지 전달, 패키지 관리 기능 등을 제공.

또한, ROS는 여러 컴퓨터 시스템 작동하는 코드를 얻어오고, 빌드하고, 작성하고, 실행하기 위한 도구 및 라이브러리를 제공.





ROS개요

ROS의 목적

로보틱스 소프트웨어 개발을 전 세계 레벨에서 공동작업이 가능하도록 하는 환경을 구축하는 것
(로봇 소프트웨어 플랫폼, 미들웨어, 프레임 워크 보다는 연구, 개발에서의 코드 재사용을
극대화가 초점이다.)

ROS의 기능

분산프로세스 : 최소단위의 실행가능한 프로세스(Node, 노드) 형태로 프로그램하며, 각 프로세스는 독립적으로 실행되면서 유기적으로 데이터를 주고 받을 수 있다.

패키지 단위 관리 : 같은 작업을 수행하는 여러개의 프로세스를 패키지 단위로 관리 가능하여 개발 또는 사용이 편리하고 배포, 공유, 수정에 용이하다.

공개 저장소 : 개발자는 Github 등에 프로젝트를 공개하고 라이센스를 밝히게 되어 있다.

API 형태 : ROS는 API를 불러와 자신이 사용하던 코드를 쉽게 적용 가능하다

프로그래밍 지원 : ROS는 다양한 언어를 지원. (Python, C++, Lisp, Java, C#, Lua, Ruby 등)



ROS개요

ROS의 목적

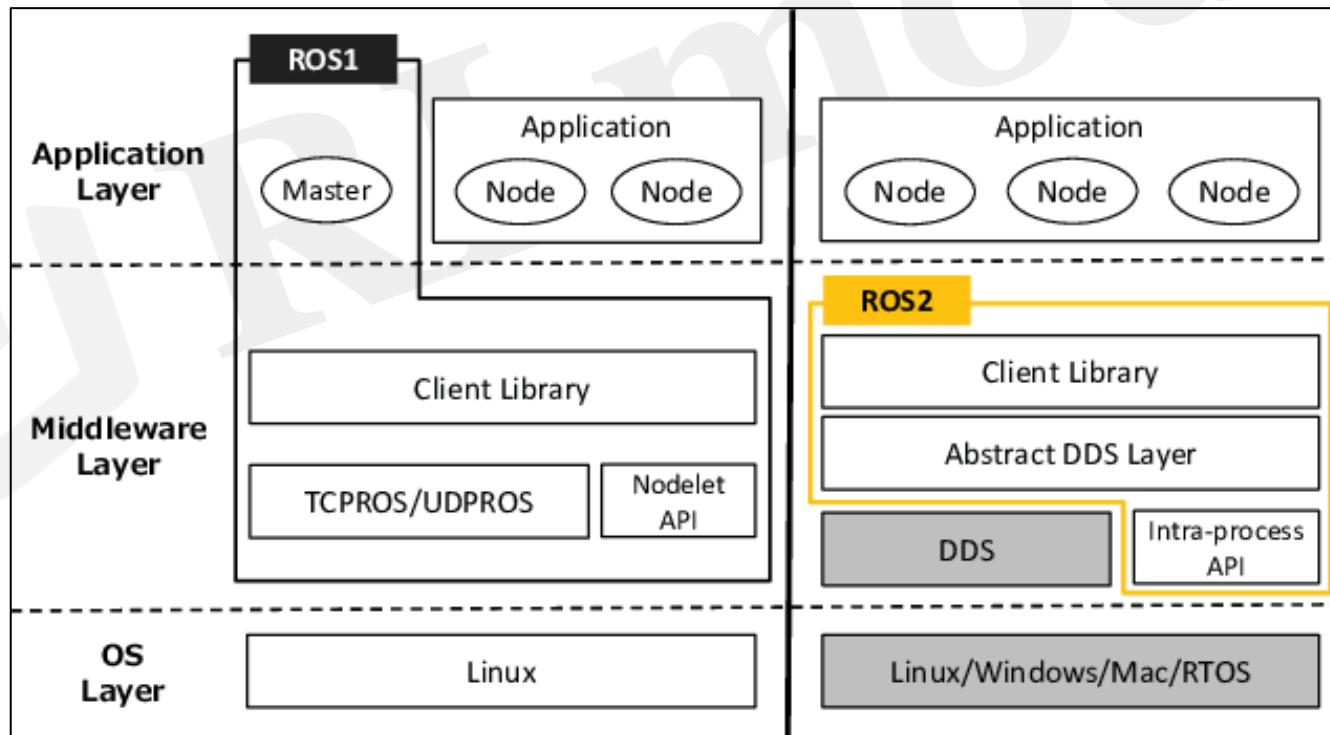
로봇 소프트웨어 플랫폼의 장점

- 높은 프로그램 재사용성
- 통신 기반 프로그램
- 개발도구 지원
- 활성화된 커뮤니티
- 생태계 조성
- 기존의 PC와 Phone의 역사를 Robot에 반영시키기 위한 생태계 형성

ROS1, ROS2 차이

ROS2 특징

- Platforms: [Linux, Windows](#)
- 통신방식: [DDS](#)
- 언어: [Python 3.x](#)
- Build system: [ament](#)

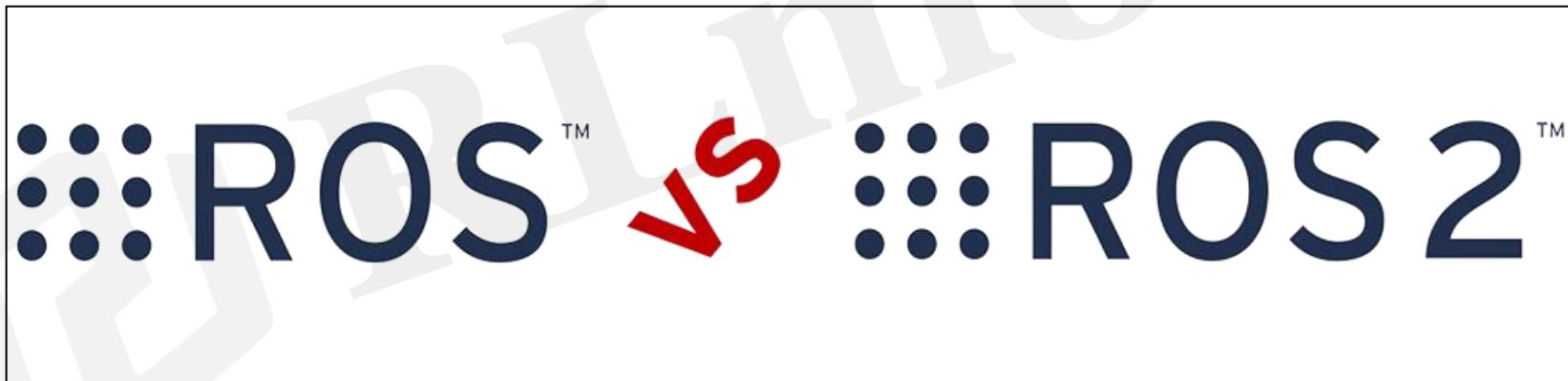




ROS₁, ROS₂ 차이

ROS₂ 특징

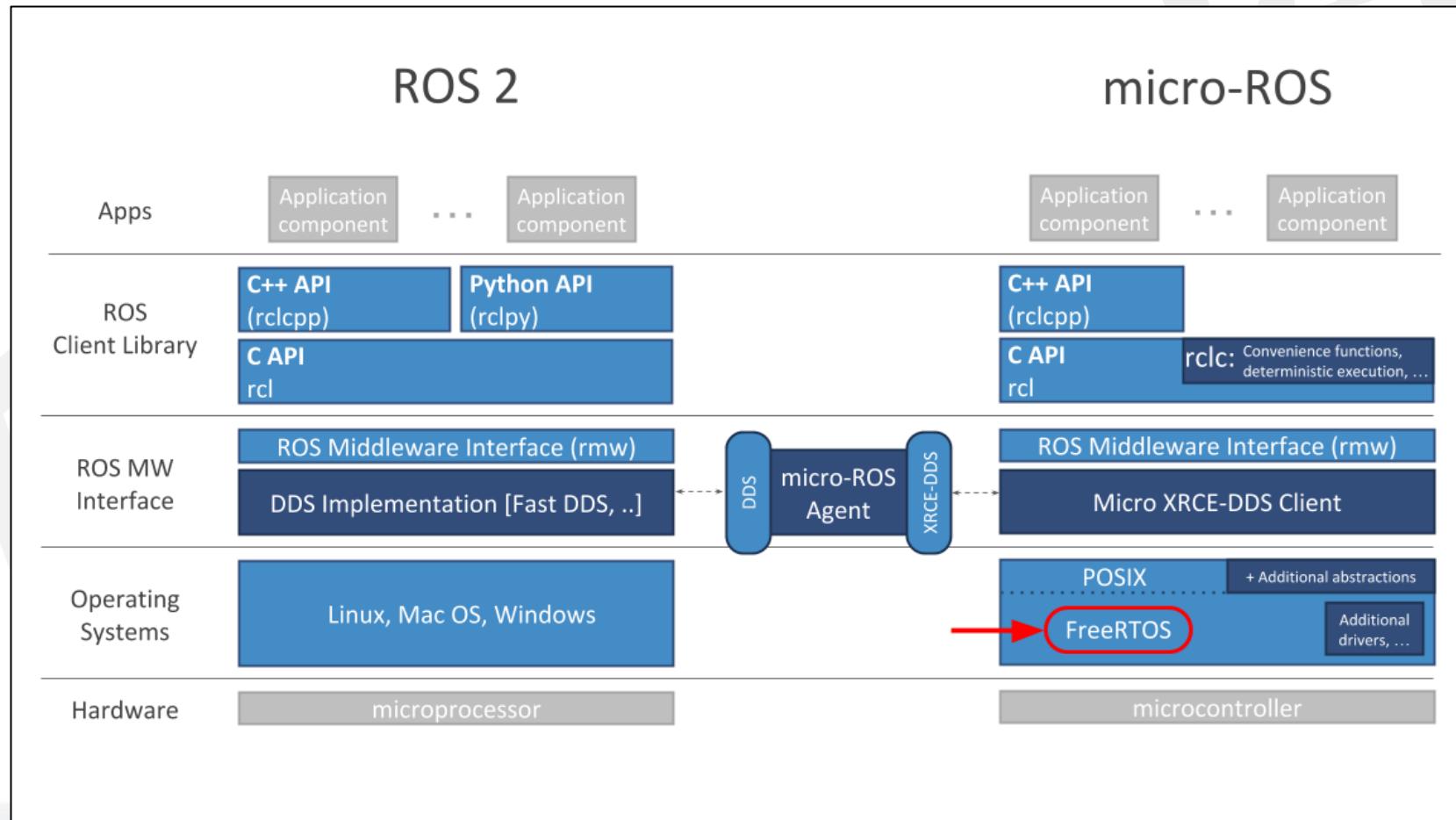
- 복수의 로봇
- 임베디드 시스템에서의 ROS 사용
- 실시간 제어
- 멀티 플랫폼



Micro ROS

정의/특징

- Micro-ROS는 리소스가 제한된 시스템 및 micro-controller를 위해 특별히 설계된 가벼운 실시간 프레임워크.





ROS2 구성

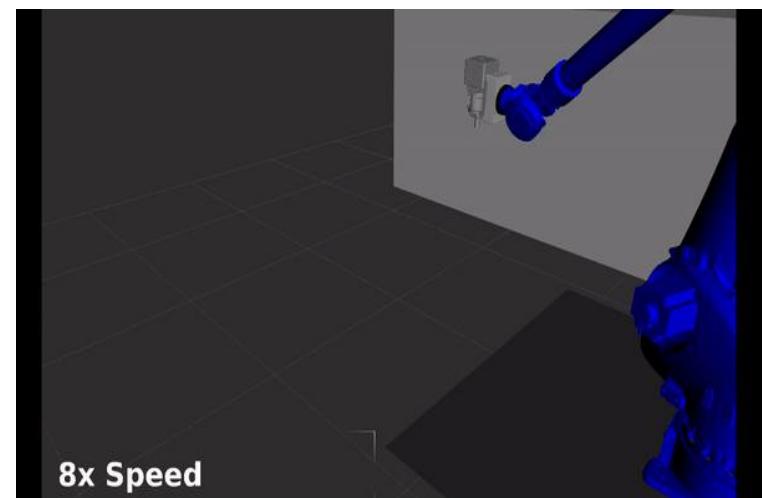
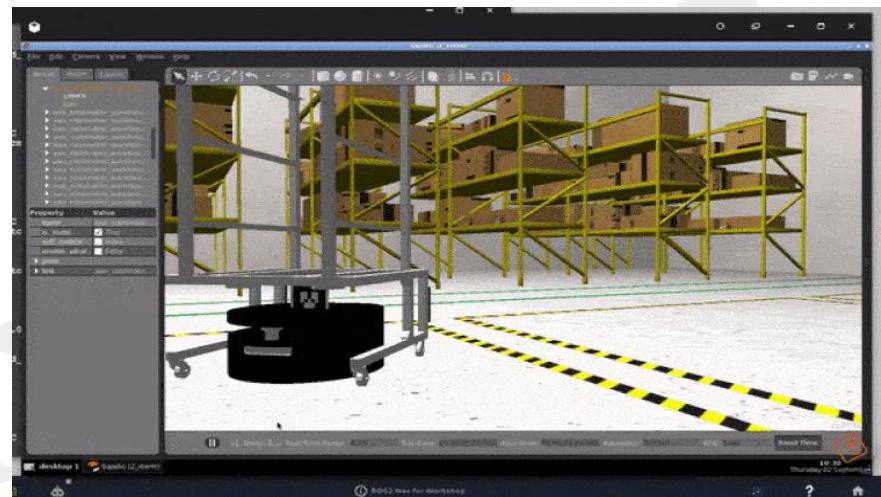
항목





ROS2 구성

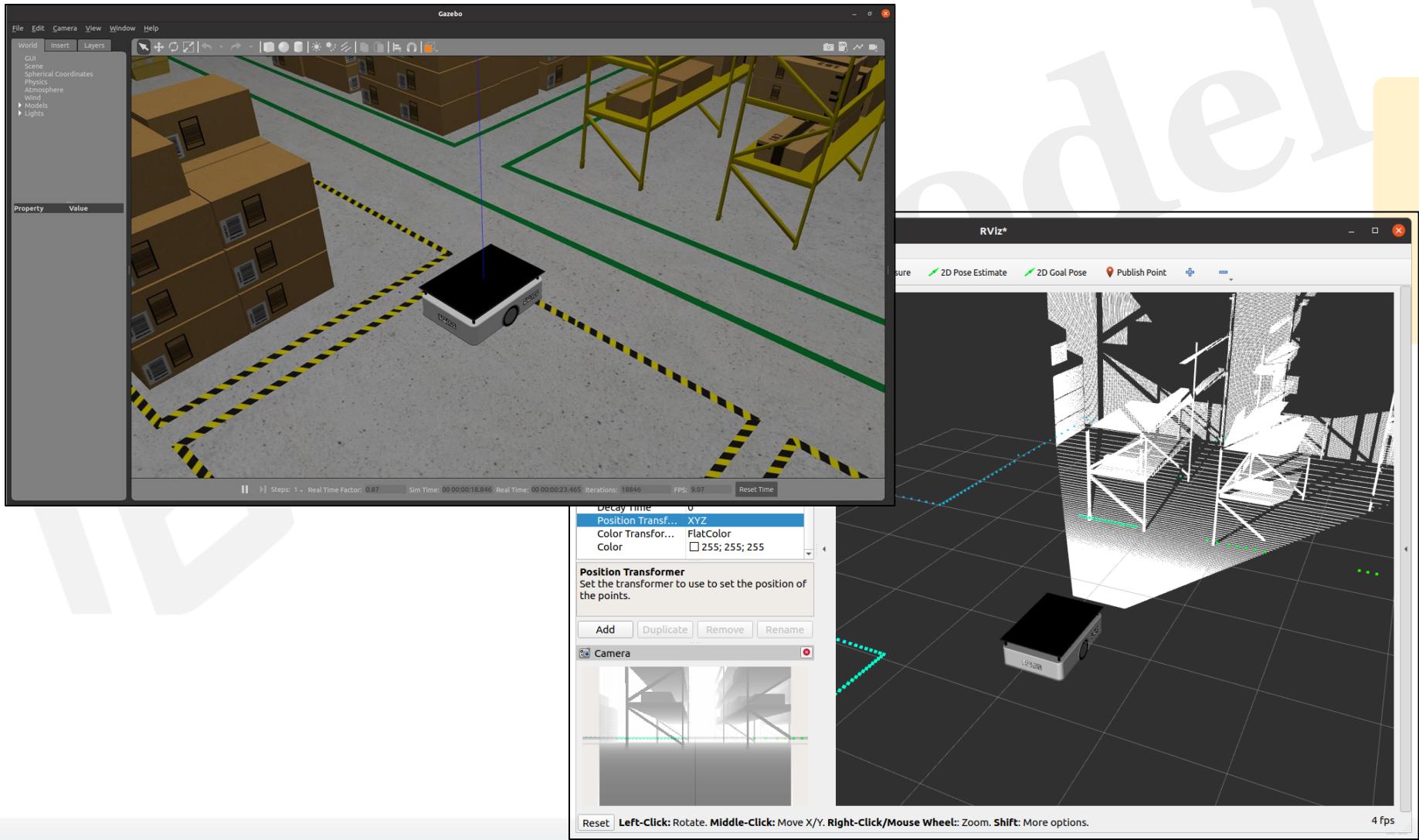
3D simulation with ROS2





ROS2 구성

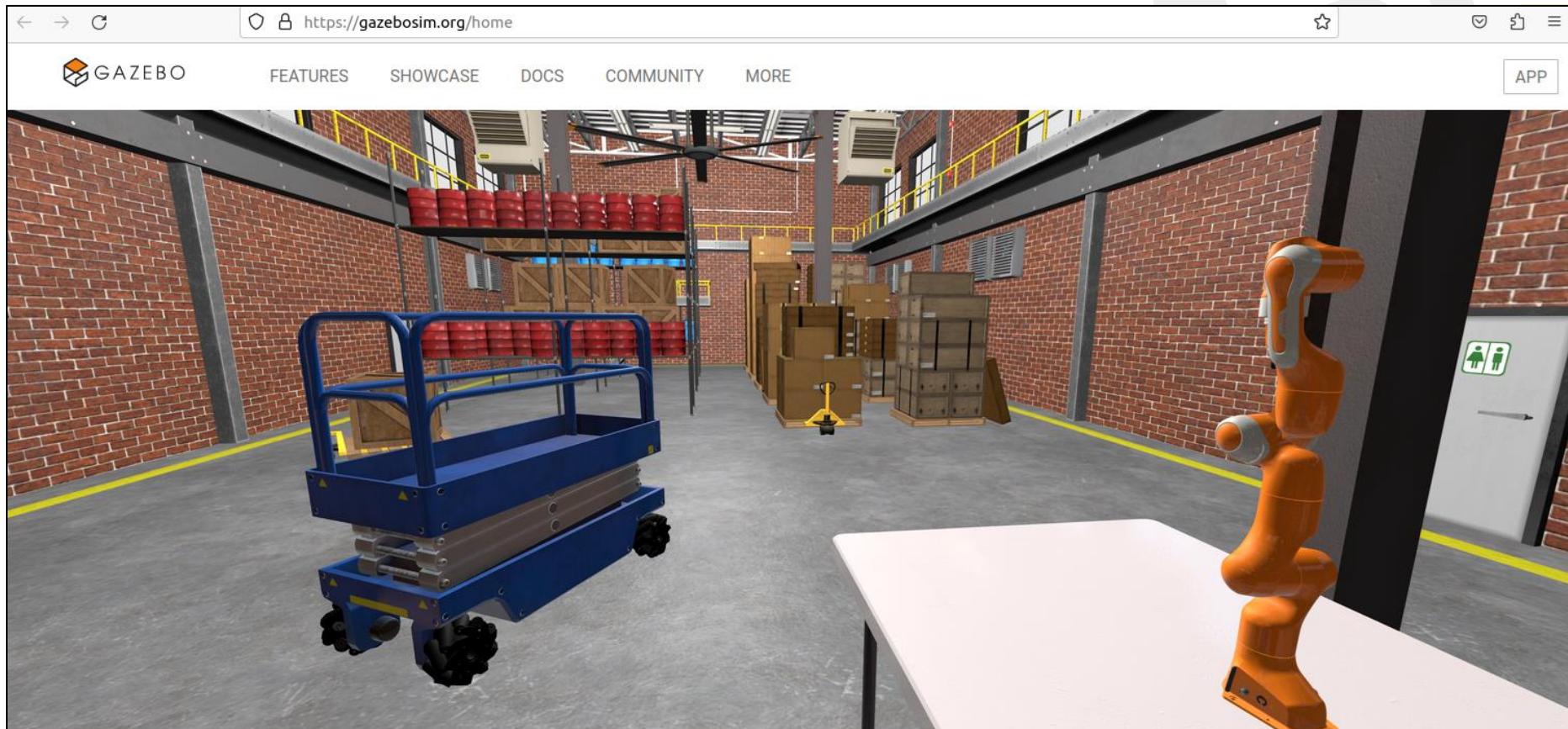
3D simulation with ROS2



ROS2 구성

Gazebo examples

<https://gazebosim.org/home>

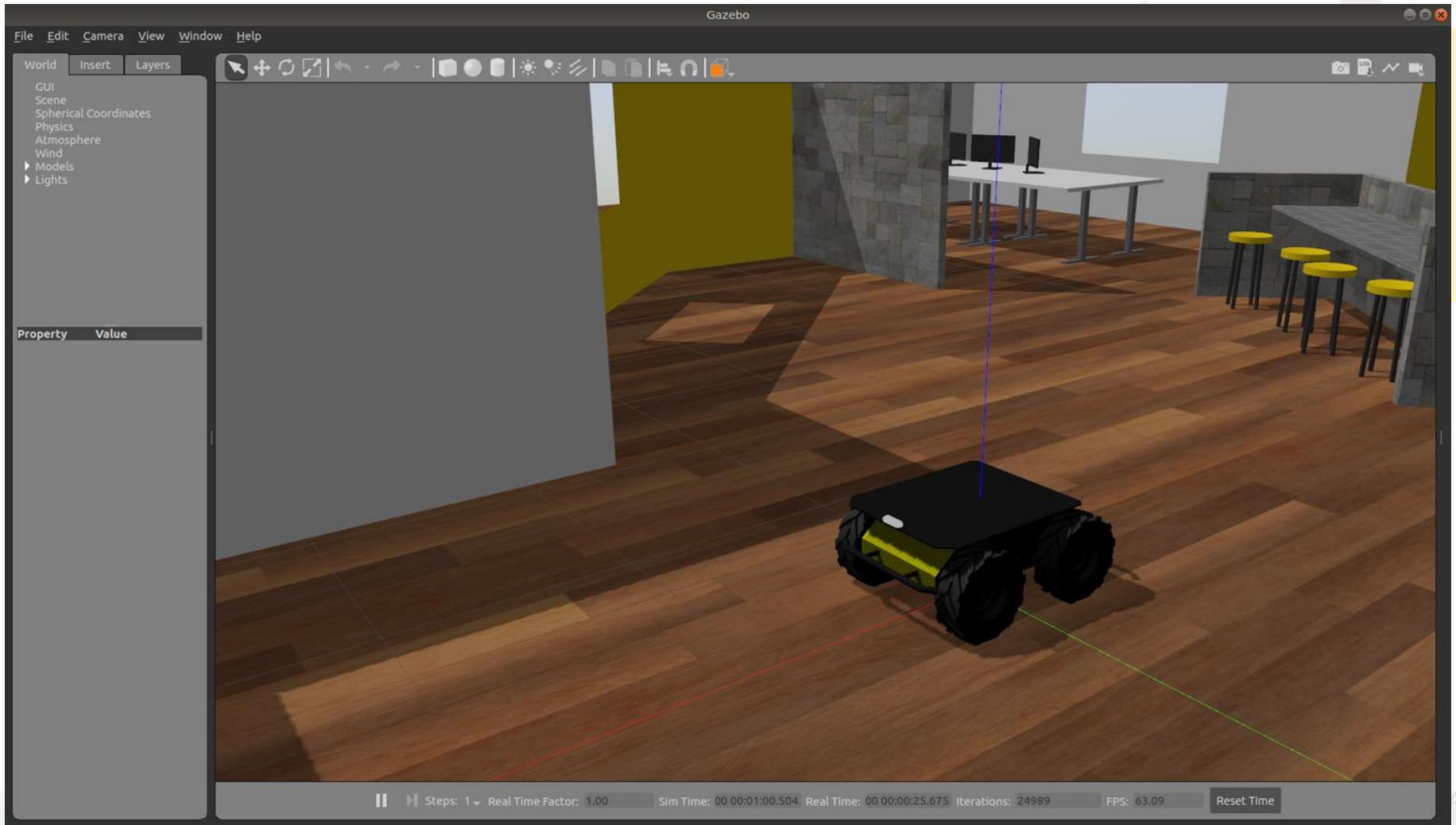




ROS2 구성

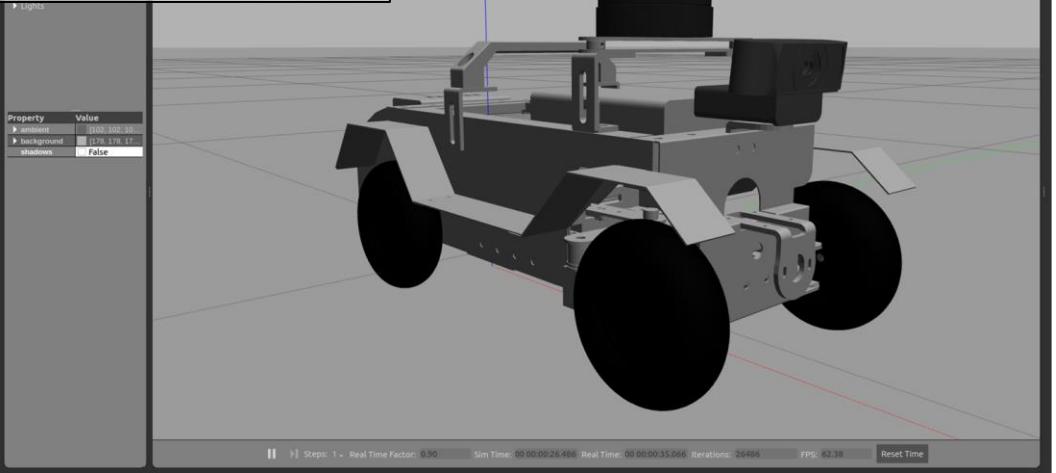
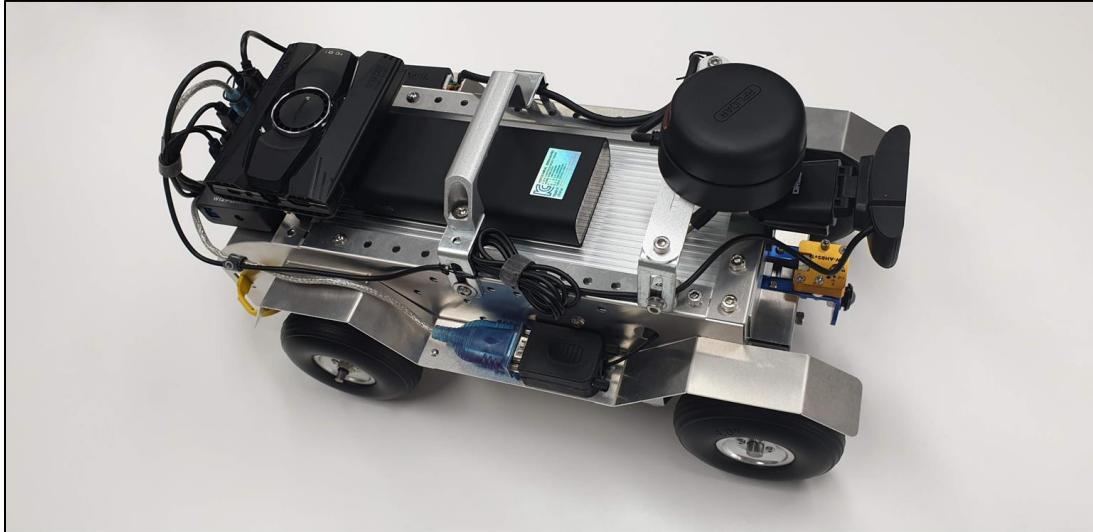
Clearpath Husky robot

https://www.clearpathrobotics.com/assets/guides/kinetic/husky/additional_sim_worlds.html



ROS2 구성

3D simulation with ROS2



ROS2 doc

ROS2 documentation : foxy

link: <https://docs.ros.org/en/foxy/index.html>

The screenshot shows a Firefox browser window displaying the ROS 2 Documentation for the 'Foxy' version. The page has a dark blue header with the title 'ROS 2 Documentation: Foxy' and a cartoon fox logo. A sidebar on the left lists navigation links: Installation, Distributions, Tutorials, How-to Guides, Concepts, Contact, The ROS 2 Project, Related Projects, Glossary, and Citations. The main content area starts with a note about the EOL status of the version. It then features a large section titled 'ROS 2 Documentation' with a brief introduction to what ROS is. Below this is another section about the history and goals of ROS 2. Further down, there's information on how to cite the documentation and a 'Getting started' section with a bulleted list of links to various guides.

You're reading the documentation for a version of ROS 2 that has reached its EOL (end-of-life), and is no longer officially supported. If you want up-to-date information, please have a look at [Iron](#).

ROS 2 Documentation

The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. From drivers and state-of-the-art algorithms to powerful developer tools, ROS has the open source tools you need for your next robotics project.

Since ROS was started in 2007, a lot has changed in the robotics and ROS community. The goal of the ROS 2 project is to adapt to these changes, leveraging what is great about ROS 1 and improving what isn't.

This site contains the documentation for ROS 2. If you are looking for ROS 1 documentation, check out the [ROS wiki](#).

If you use ROS 2 in your work, please see [Citations](#) to cite ROS 2.

Getting started

- [Installation](#)
 - Instructions to set up ROS 2 for the first time
- [Tutorials](#)
 - The best place to start for new users!
 - Hands-on sample projects that help you build a progression of necessary skills
- [How-to Guides](#)
 - Quick answers to your "How do I...?" questions without working through the [Tutorials](#)
- [Concepts](#)
 - High-level explanations of core ROS 2 concepts covered in the [Tutorials](#)
- [Contact](#)



ROS version

Generation

Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Noetic Ninjemy (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015
ROS Groovy Galapagos	December 31, 2012			July, 2014
ROS Fuerte Turtle	April 23, 2012			--
ROS Electric Emys	August 30, 2011			--
ROS Diamondback	March 2, 2011			--
ROS C Turtle	August 2, 2010			--
ROS Box Turtle	March 2, 2010			--

ROS



ROS2



20.04
ubuntu

18.04
ubuntu



ROS version

<https://www.ros.org/>

The screenshot shows the official ROS website at https://www.ros.org/. The page has a dark blue header with the ROS logo on the left. To the right of the logo are four navigation links: WHY ROS?, GETTING STARTED, COMMUNITY, and ECOSYSTEM. Below the header, a large white title reads "ROS - Robot Operating System". Underneath the title is a descriptive paragraph about ROS. To the right of the text is a large, stylized green illustration of a frog sitting on a green base, with a small blue bird perched on its head.

https://www.ros.org/

ROS

WHY ROS?

GETTING STARTED

COMMUNITY

ECOSYSTEM

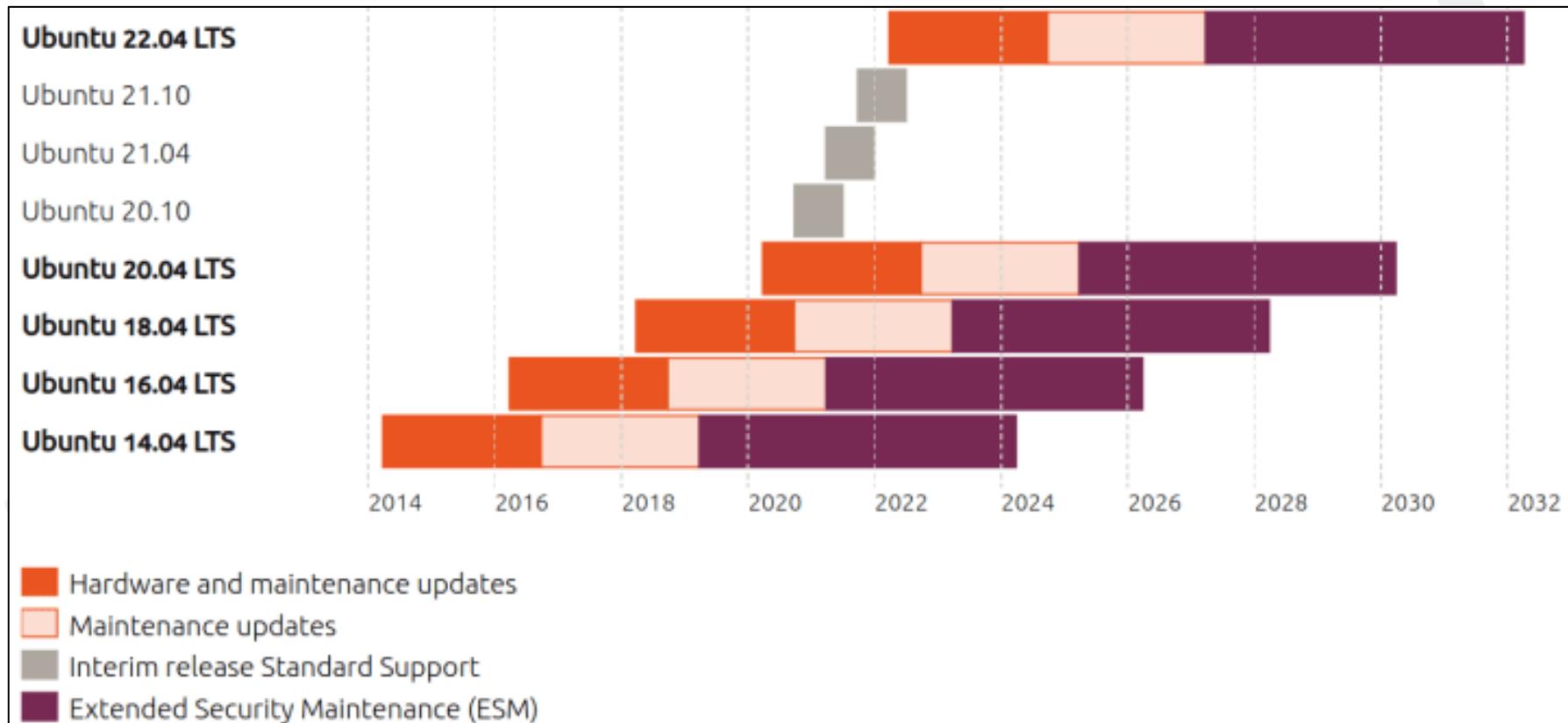
ROS - Robot Operating System

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.



ROS version

Ubuntu version



Build system

설명

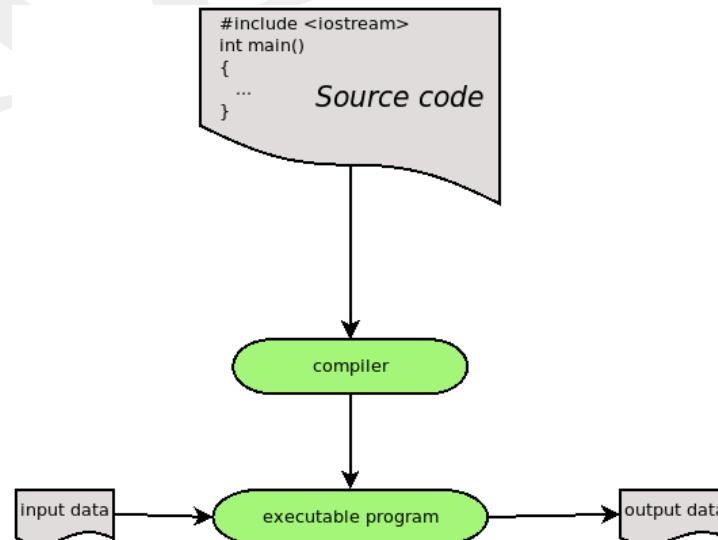
소스코드를 프로그램으로 바꾸어 주는 행위가 바로 빌드를 하는 것이고 이렇게 할 수 있도록 도와주는 도구들을 빌드 시스템이라고 말합

Compile 소스코드 전체를 기계어로 번역

Interpreted 소스코드를 한 줄씩 번역하면서 실행

C++ > compile type (전처리 > 컴파일 > 링크)

Python > interpreter type





ROS2 build system

ament

ament는 로봇 소프트웨어 개발을 위한 빌드 시스템, ROS 2에서 사용.

이전 ROS1에서는 Catkin이 주로 사용, ROS 2에서 ament가 새로운 빌드 시스템으로 채택

패키지 기반: ament는 개별 패키지를 중심으로 빌드를 수행합니다. 각 패키지는 독립적으로 빌드되며, 종속성 관리 및 빌드 옵션 설정이 용이합니다. 이를 통해 모듈화된 소프트웨어 시스템을 구성할 수 있습니다.

CMake 기반: ament는 CMake 빌드 시스템을 기반으로 하여 CMakeLists.txt 파일을 사용하여 패키지를 구성합니다. CMake는 강력하고 유연한 빌드 시스템으로 알려져 있으며, 다양한 플랫폼에서 이식성을 제공합니다.

Cross-컴파일 지원: ament는 크로스-컴파일 환경에서도 작동하도록 설계되었습니다. 이는 로봇 시스템과 같이 타겟 플랫폼과 개발 환경이 분리된 경우에 유용합니다.

메타빌드: ament는 패키지 간 종속성을 정의하고 관리하기 위한 메타빌드 시스템을 제공합니다. 이를 통해 패키지 간 종속성을 간편하게 설정하고, 패키지의 빌드 및 설치 순서를 관리할 수 있습니다.

빌드 도구 호환성: ament는 다양한 빌드 도구와 호환됩니다. 예를 들어, CMake로 작성된 패키지를 빌드할 수 있으며, Python 패키지와 같은 다른 유형의 패키지도 지원합니다.



ROS2 build 툴

colcon

Colcon Build는 로봇 소프트웨어 개발에 사용되는 툴 체인의 일부로, ROS(로봇 운영체제) 프레임워크에서 사용되는 빌드 시스템입니다. Colcon은 "COnsolidate Language-specific build CONfiguration"의 약자

다양한 언어 지원: Colcon Build는 ROS 패키지를 구성하는 다양한 언어(C++, Python 등)의 빌드를 지원합니다. 따라서 ROS 프로젝트에서 여러 언어로 작성된 소스 코드를 효율적으로 관리할 수 있습니다.

병렬 빌드: Colcon Build는 병렬 빌드를 지원하여 여러 패키지를 동시에 빌드할 수 있습니다. 이를 통해 빌드 시간을 단축하고 개발자들의 생산성을 향상시킬 수 있습니다.

빌드 종속성 관리: Colcon Build는 패키지 간의 종속성을 자동으로 관리합니다. 패키지 간의 의존성을 정확히 파악하고 필요한 종속성 패키지를 자동으로 설치하므로, 빌드 과정에서 발생할 수 있는 문제를 최소화할 수 있습니다.

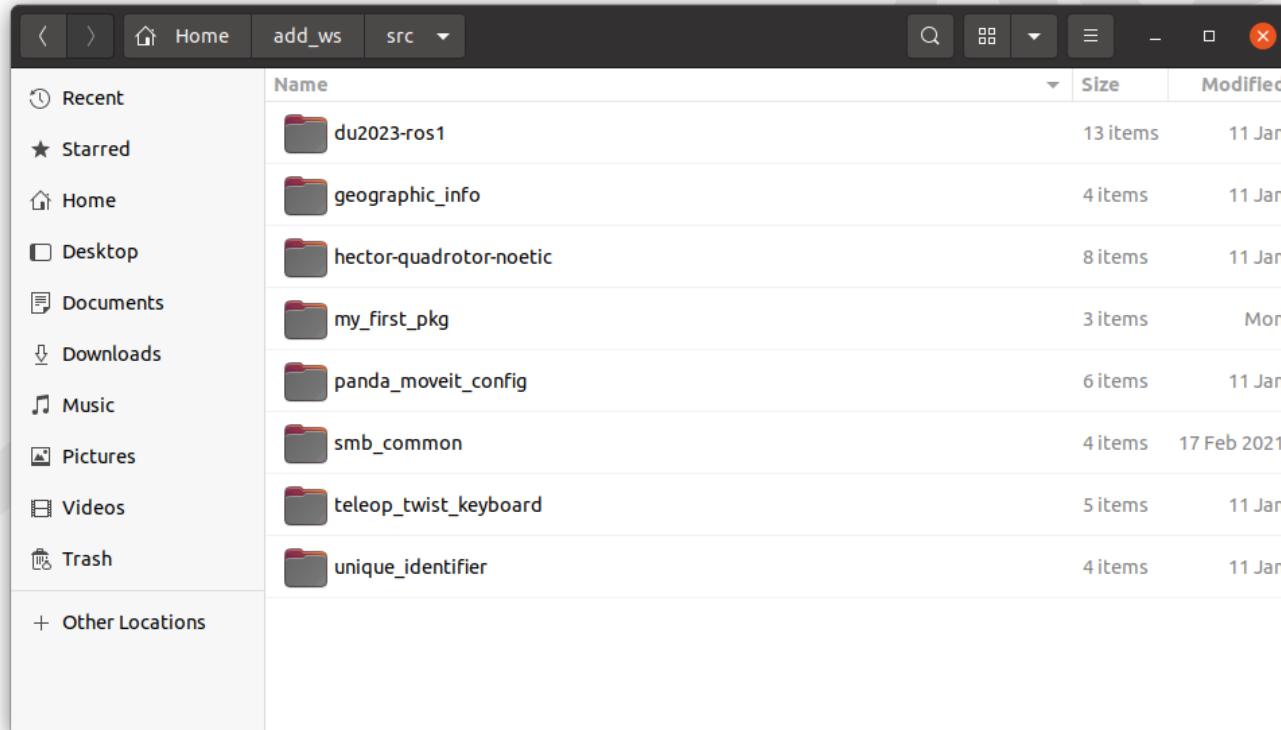
커스터마이징 가능: Colcon Build는 다양한 빌드 옵션을 제공하여 사용자가 빌드 프로세스를 커스터마이징할 수 있습니다. 빌드 타겟, 빌드 옵션, 환경 변수 설정 등을 조정하여 개발자들의 요구에 맞게 빌드를 구성할 수 있습니다.

Package

Package 정의

패키지(package)는 하나 이상의 노드(node)가 기능적 단위로 묶인 ROS 코드의 컨테이너입니다.

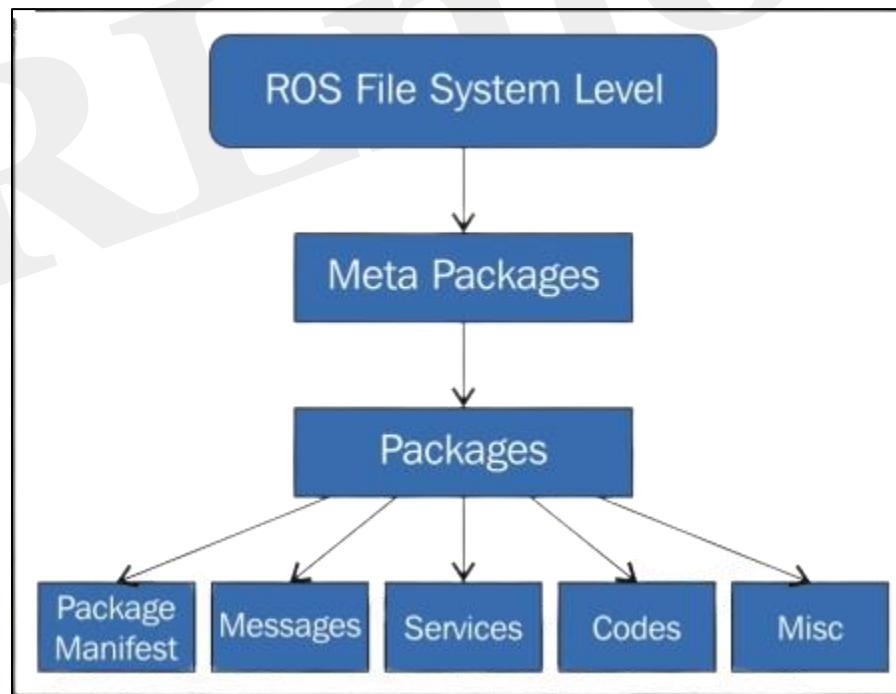
코드를 설치하거나 공유하기 위해서는 패키지를 구성해야합니다. ROS2는 ament을 빌드 시스템으로 사용하고 colcon을 빌드 툴로 사용합니다



ROS2 구성

패키지

- ROS2 코드의 컨테이너
- 최소 패키지 구조
 - package.xml : 패키지에 대한 메타 정보가 포함된 파일
 - CMakeLists.txt : 패키지 내에서 코드를 빌드하는 방법을 설명하는 파일
- 보통 저장공간은 ~/dev_ws/src/패키지 로 구성





ROS2 구성

package 확인 (내장)

cmd: \$ ros2 pkg executables

```
byb76@rlhp: ~/add_ros2_ws
byb76@rlhp: ~/add_ros2_ws 65x18
tf2_ros static_transform_publisher
tf2_ros tf2_echo
tf2_ros tf2_monitor
tf2_tools view_frames.py
topic_monitor data_publisher
topic_monitor topic_monitor
tracetools status
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
velodyne_driver vdump
velodyne_driver velodyne_driver_node
velodyne_laserscan velodyne_laserscan_node
velodyne_pointcloud velodyne_convert_node
velodyne_pointcloud velodyne_transform_node
xacro xacro
(ROS 2 foxy) byb76@rlhp:~/add_ros2_ws$ ros2 pkg executables
```

ROS2 패키지

package 항목

확인명령어: \$ ros2 pkg list

```
Tabed@abed-Lenovo-Z50-70: ~
File Edit View Search Terminal Help
std_msgs
std_srvs
stereo_msgs
teleop_twist_joy
teleop_twist_keyboard
tf2
tf2_eigen
tf2_geometry_msgs
tf2_kdl
tf2_msgs
tf2_py
tf2_ros
tf2_sensor_msgs
tlsf
tlsf_cpp
topic_monitor
tracetools
trajectory_msgs
uncrustify_vendor
unique_identifier_msgs
urdf
visualization_msgs
yaml_cpp_vendor
abed@abed-Lenovo-Z50-70:~$
```



ROS2 패키지

package 항목

도움말 명령어: \$ ros2 pkg --help

```
abed@abed-Lenovo-Z50-70: ~
File Edit View Search Terminal Help
tracetools
trajectory_msgs
uncrustify_vendor
unique_identifier_msgs
urdf
visualization_msgs
yaml_cpp_vendor
abed@abed-Lenovo-Z50-70:~$ ros2 pkg --help
usage: ros2 pkg [-h] Call `ros2 pkg <command> -h` for more detailed usage. ...

Various package related sub-commands

optional arguments:
  -h, --help            show this help message and exit

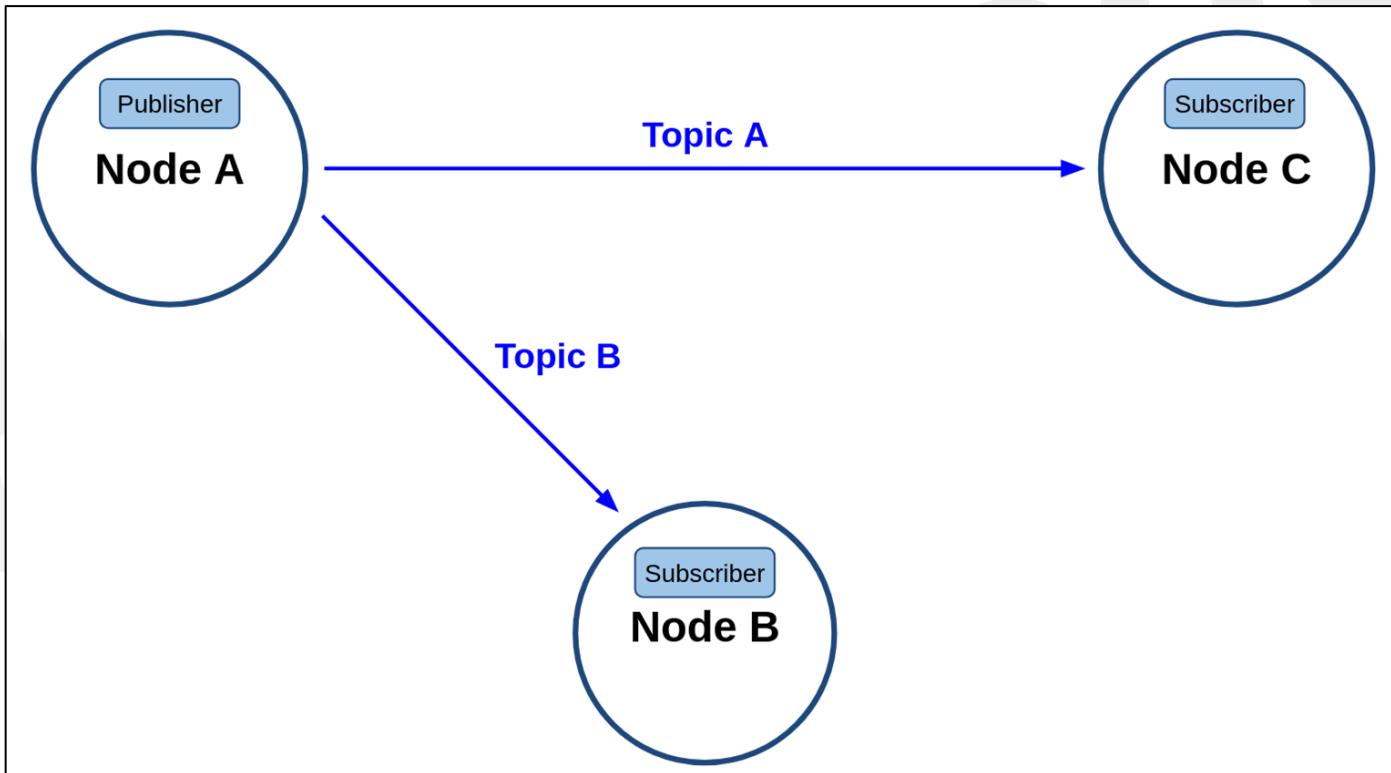
Commands:
  create      Create a new ROS2 package
  executables Output a list of package specific executables
  list        Output a list of available packages
  prefix      Output the prefix path of a package
  xml         Output the XML of the package manifest or a specific tag

  Call `ros2 pkg <command> -h` for more detailed usage.
abed@abed-Lenovo-Z50-70:~$
```

Node

node (노드)란

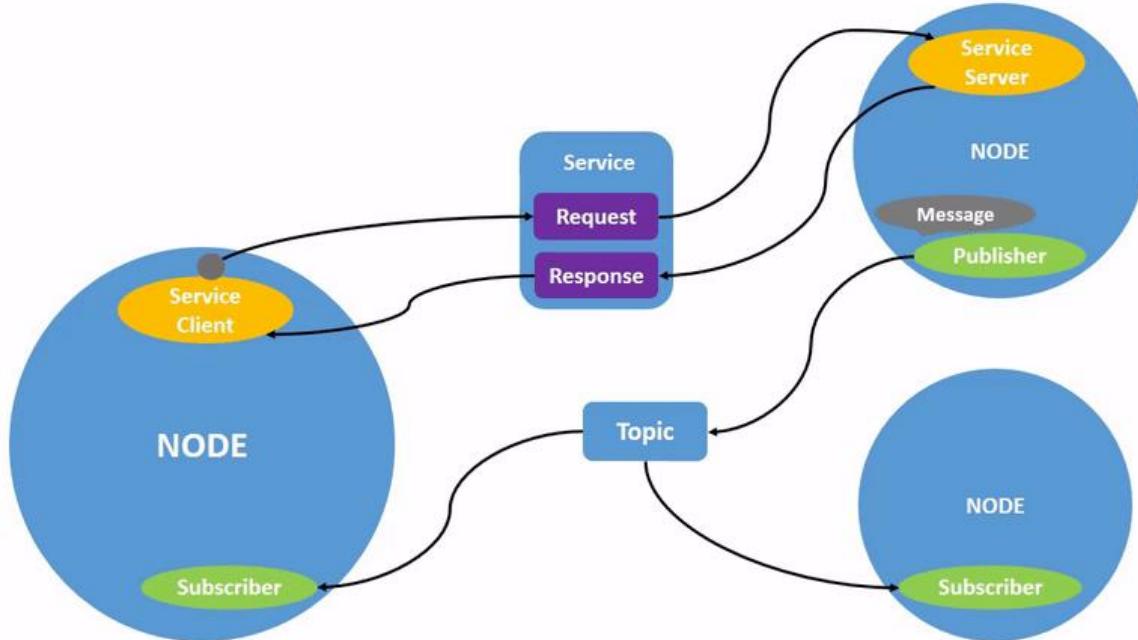
- Node는 필요한 연산을 실행하는 최소 프로세스이다.
- ROS는 노드 단위로 프로그램을 작성하며 노드사이에 메시지를 주고 받으면서 통신합니다



Node

node 특징

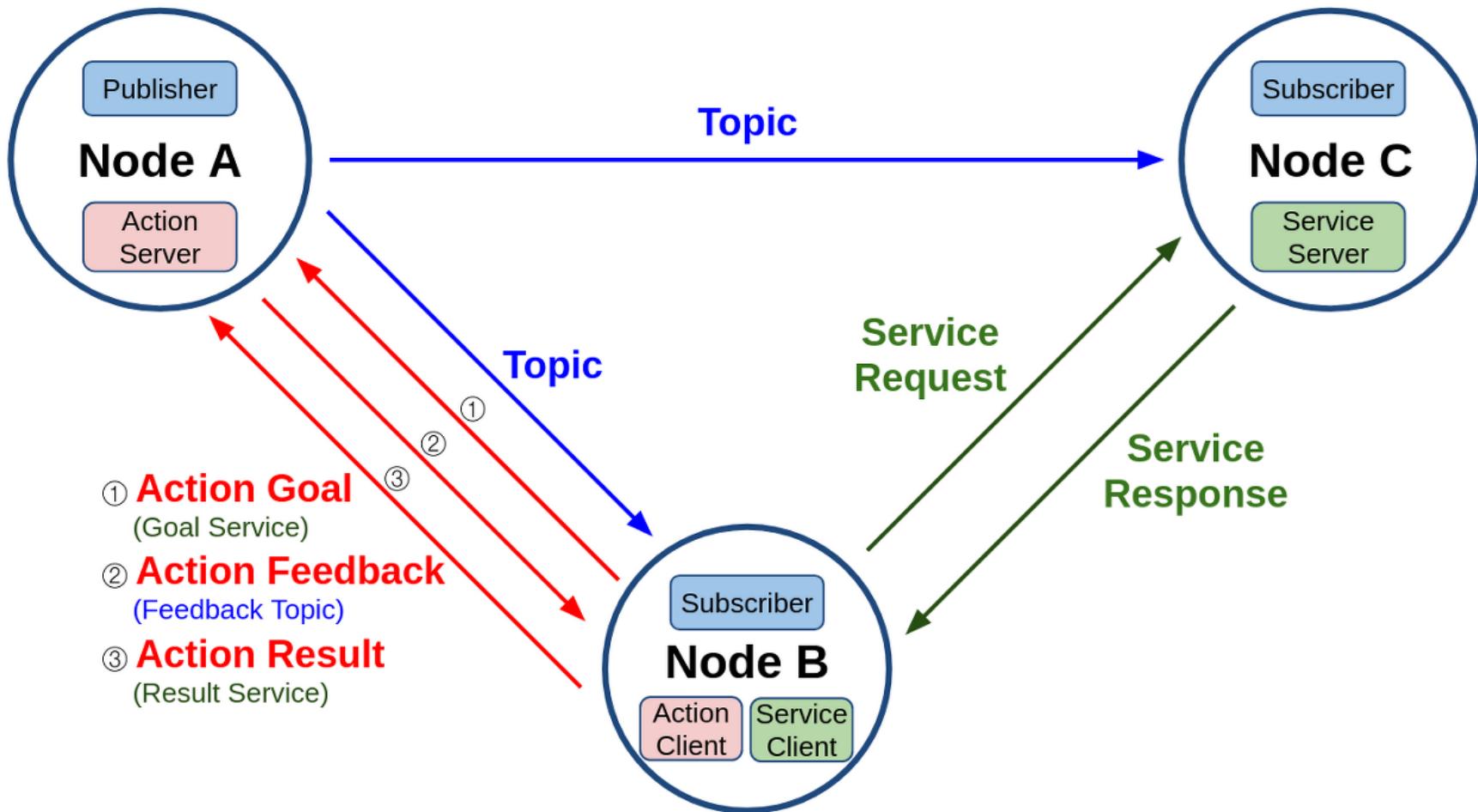
- 로봇 제어 시스템은 일반적으로 많은 Node로 구성. (예) Node1은 레이저 거리 측정기를 제어하고, Node2는 로봇의 휠 모터를 제어하고, 또 다른 하나의 Node3은 로컬라이제이션을 수행
- Node들을 묶어서 Package를 만들 수 있다.



Node

node 구성

다양한 Node 들의 구성으로 Package(패키지)를 만들 수 있다.





ROS2 interface

interface

ROS 노드 간에 데이터를 주고받기 위해서 토픽, 서비스, 액션을 사용하고, 이때 사용되는 데이터 형태를 ROS 2 인터페이스라고 한다.

이들은 각각 msg, srv, action interface를 사용하며, 정수, 부동 소수점, boolean과 같은 단순 자료형과, 데이터 구조와 메시지들이 나열된 배열같은 구조로도 사용이 가능.

ex) cmd_vel 토픽 > geometry_msgs/msg/Twist

```
ybbaek@ubuntu:~$ ros2 node info /turtlesim  
/turtlesim  
Subscribers:  
/parameter_events: rcl_interfaces/msg/ParameterEvent  
/turtle1/cmd_vel: geometry_msgs/msg/Twist  
...
```



ROS2 interface

interface

ex) **geometry_msgs** 정의

http://wiki.ros.org/geometry_msgs

[About](#) | [Support](#) | [Discussion Forum](#) | [Index](#) | [Service Status](#) | [Q&A answers.ros.org](#)

[Documentation](#) [Browse Software](#) [News](#) [Download](#)

geometry_msgs

[melodic](#) [noetic](#) Show EOL distros:

See [geometry_msgs](#) on [index.ros.org](#) for more info including anything ROS 2 related.

[Documentation Status](#)

[common_msgs](#): [actionlib_msgs](#) | [diagnostic_msgs](#) | [geometry_msgs](#) | [nav_msgs](#) | [sensor_msgs](#) | [shape_msgs](#) | [stereo_msgs](#) | [trajectory_msgs](#) | [visualization_msgs](#)

Package Summary

✓ Released ✓ Continuous Integration: 3 / 3 ✓ Documented

geometry_msgs provides messages for common geometric primitives such as points, vectors, and poses. These primitives are designed to provide a common data type and facilitate interoperability throughout the system.

- Maintainer status: maintained
- Maintainer: Michel Hidalgo <michel AT ekumenlabs DOT com>
- Author: Tully Foote <tfoote AT osrfoundation DOT org>
- License: BSD
- Source: git https://github.com/ros/common_msgs.git (branch: noetic-devel)

Package Links

[Code API](#)
[Msg API](#)
[Tutorials](#)
[Troubleshooting](#)
[FAQ](#)
[Changelog](#)
[Change List](#)
[Reviews](#)

Dependencies (4)
[Used by \(274\)](#)
[Jenkins jobs \(10\)](#)

ROS 2 Documentation

The ROS Wiki is for ROS 1. Are you using ROS 2 ([Foxy](#), [Galactic](#), [Humble](#), or [Rolling](#))?
Check out the [ROS 2 Project Documentation](#)

Package specific documentation can be found on [index.ros.org](#)

위키

[Distributions](#)
[ROS/Installation](#)
[ROS/Tutorials](#)
[RecentChanges](#)
[geometry_msgs](#)

문서

[못 고치는 문서](#)
[정보](#)
[첨부](#)

다른 작업:

[원문 보기](#) ▾

사용자

[로그인](#)



ROS2 interface

interface

ex) Twist Message 정의

http://docs.ros.org/en/melodic/api/geometry_msgs/html/msg/Twist.html

geometry_msgs/Twist Message

File: `geometry_msgs/Twist.msg`

Raw Message Definition

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

Compact Message Definition

```
geometry_msgs/Vector3 linear  
geometry_msgs/Vector3 angular
```

autogenerated on Mon, 28 Feb 2022 22:12:13



ROS2 interface

interface

ex) Twist Message 정의

geometry_msgs/msg/Vector3

ros2 interface show geometry_msgs/msg/Vector3

```
rlmodel@rlmodel:~/ros2_ws$ ros2 interface show geometry_msgs/msg/Vector3
# This represents a vector in free space.

# This is semantically different than a point.
# A vector is always anchored at the origin.
# When a transform is applied to a vector, only the rotational component is applied.

float64 x
float64 y
float64 z
rlmodel@rlmodel:~/ros2_ws$
```



ROS2 parameter

parameter 정의

ROS 2의 파라미터는 각 노드가 가진 Parameter Server를 통하여 외부의 Parameter Client와 통신하여 파라미터를 변경하는 것.

ROS 2의 Service와 비슷하지만, 파라미터는 서비스 통신 방법을 사용하여 노드 내부, 외부에서 노드 내 매개변수를 쉽게 지정, 변경, 가져와서 사용할 수 있게 하는 점에서 사용 목적이 다름.

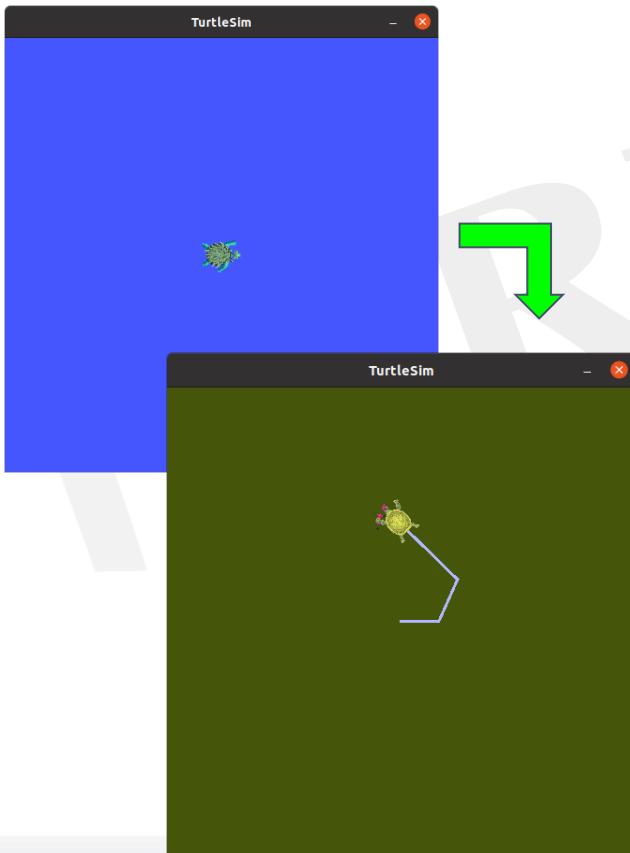
>확인 명령어 \$ ros2 param list

ROS2 구성

parameter

설명: 노드 내에서 사용되는 변수이고, 이 값을 노드 내 또는 ros2 param 명령어를 통해 확인하거나 수정

명령어: ros2 param (set, get, list, describe...)



```
rlmodel@rlmodel:~/ros2_ws$ ros2 param get /turtlesim background_b
Integer value is: 0
rlmodel@rlmodel:~/ros2_ws$ ros2 param set /turtlesim background_b 11
Set parameter successful
rlmodel@rlmodel:~/ros2_ws$ ros2 param get /turtlesim background_b
Integer value is: 11
rlmodel@rlmodel:~/ros2_ws$
```



ROS2 구성

node 확인

node 관련 명령어:

\$ros2 node list

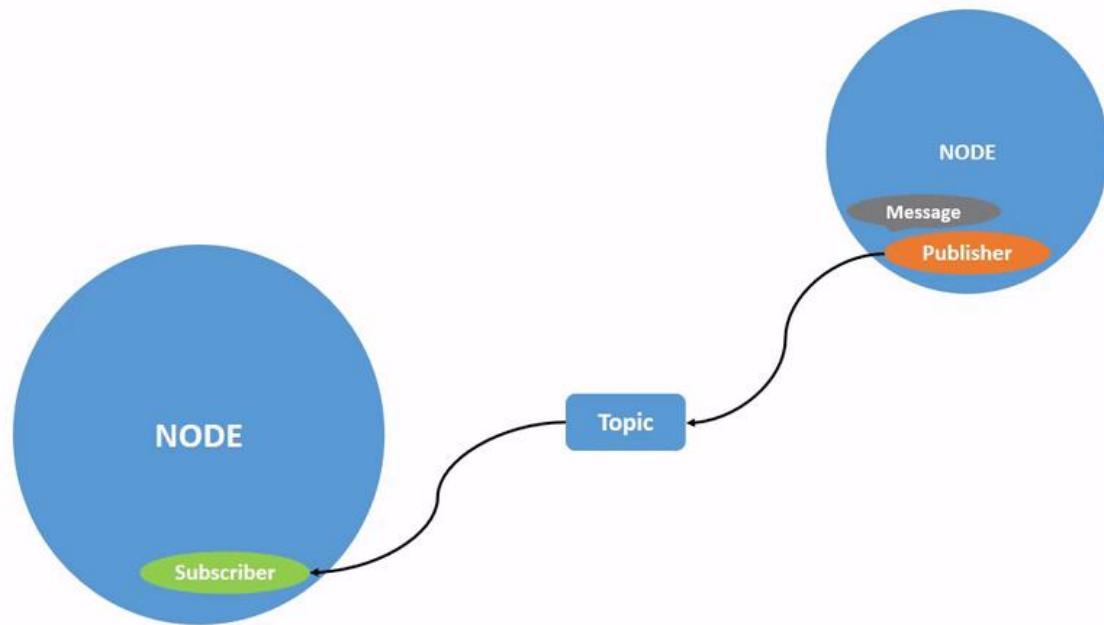
\$ros2 node info /xxx

```
byb76@ybbaek-dell:~/ros2_foxy$ ros2 node info /turtlesim --
/turtlesim
Subscribers:
/parameter_events: rcl_interfaces/msg/ParameterEvent
/turtle1/cmd_vel: geometry_msgs/msg/Twist
Publishers:
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/turtle1/color_sensor: turtlesim/msg/Color
/turtle1/pose: turtlesim/msg/Pose
Service Servers:
/clear: std_srvs/srv/Empty
/kill: turtlesim/srv/Kill
/reset: std_srvs/srv/Empty
/spawn: turtlesim/srv/Spawn
/turtle1/set_pen: turtlesim/srv/SetPen
/turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
/turtle1/teleport_relative: turtlesim/srv/TeleportRelative
/turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
/turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
/turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
/turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
/turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
/turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
Action Clients:
byb76@ybbaek-dell:~/ros2_foxy$ |
```

ROS2 구성

topic 정의

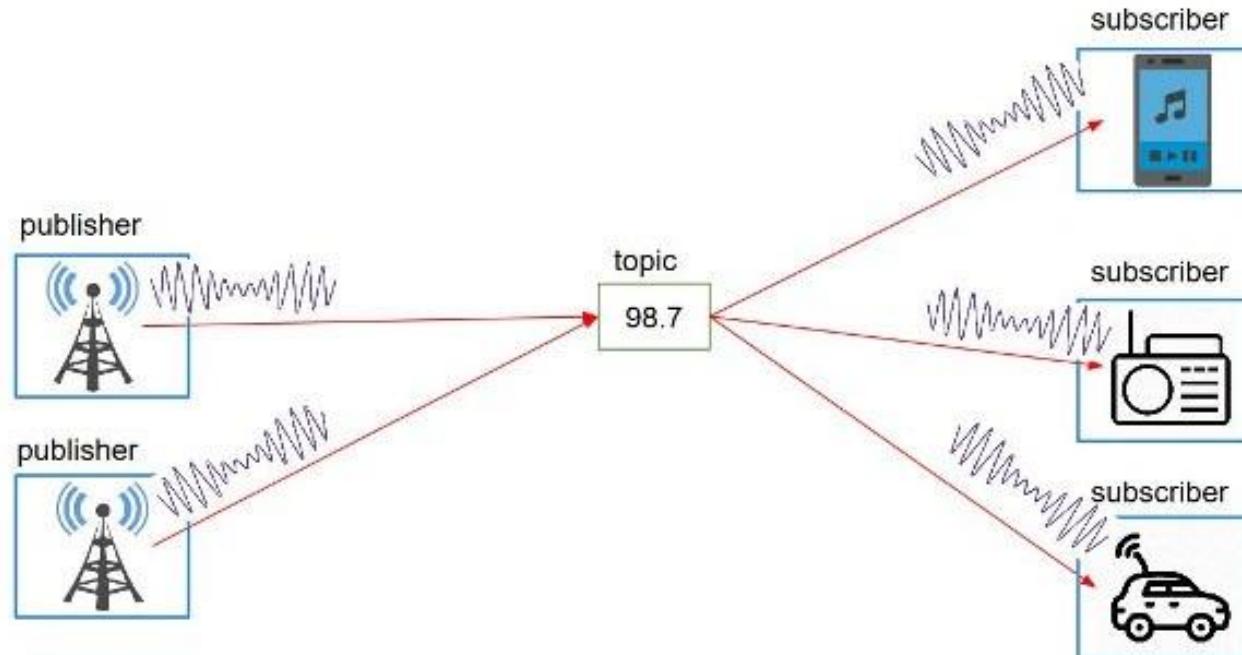
>Topic이란 노드들 간에 통신을 할 수 있는 채널이다. 두 프로그램 간에 어떤 통신이 발생하고, 어떤 메시지를 주고 받게 되는데 이 경로를 토픽이라고 한다. - 메세지는 Topic이 데이터나 정보를 주고 받을 때 사용하는 포맷임



ROS2 구성

topic 특징

- Topic 을 주는 쪽을 송신자 (Publisher), 받는 쪽을 수신자, 구독자 (Subscriber)라고 한다.
- 1:1, 1:N, N:N 등, 다양한 네트워크를 구성할 수 있다.
- 비동기식 단방향 메시 송수신 방식





ROS2 구성

topic 관련 명령어:

자주사용: list, info, echo

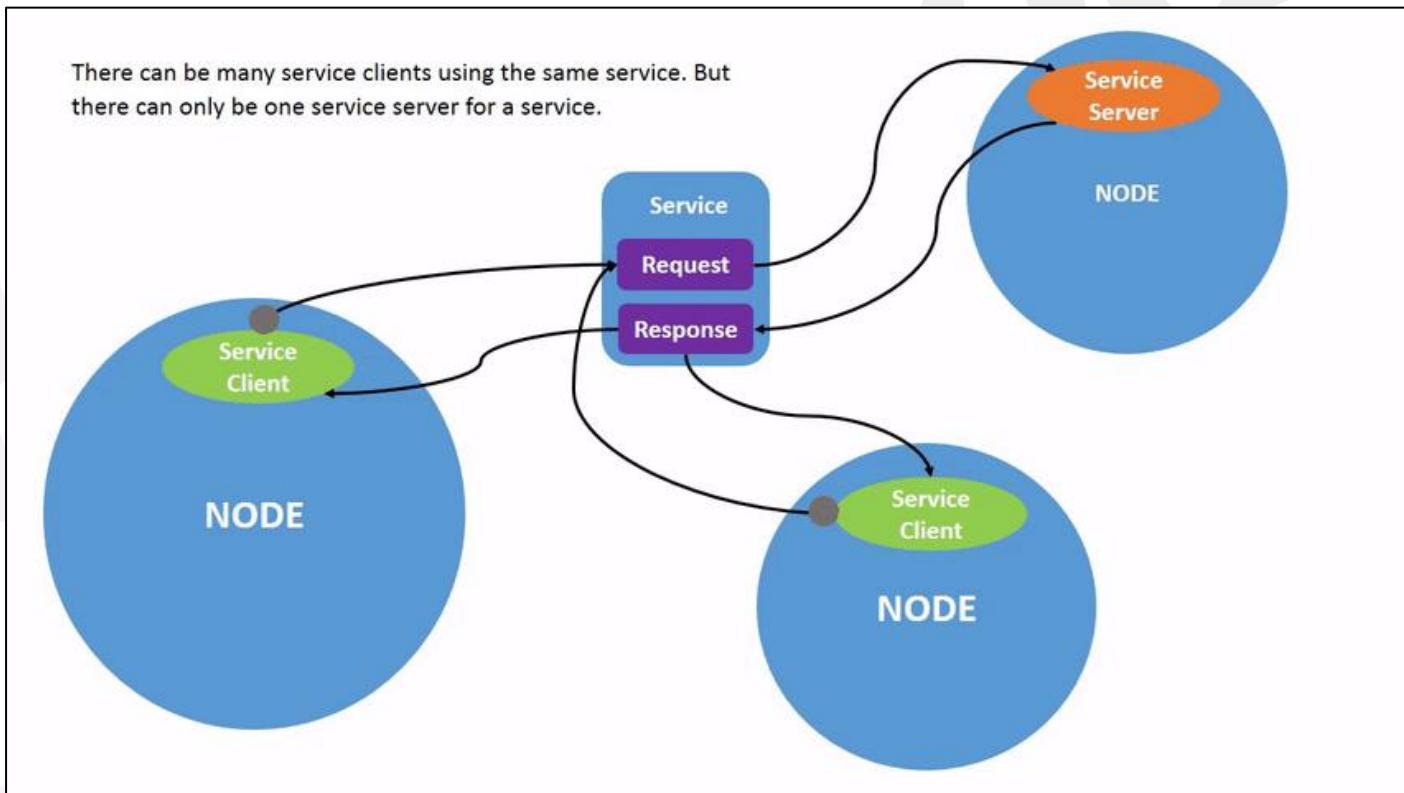
```
byb76@rlhp: ~/add_ws
byb76@rlhp: ~/add_ws 59x10
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 topic
--include-hidden-topics
bw
delay
echo
find
hz
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 topic
```

ROS2 구성

service 란

정의: 클라이언트(client)의 요청(request)과 서버(server)의 응답(response)으로 이루어진 통신방식

Service의 중요한 특징 한 가지 추가하자면, 하나의 Service Server에 여러 Client가 request 할 수 있지만, Server는 동시에 여러 request를 처리하지 못합니다.





ROS2 구성

service 설명

Service 통신 시 srv라는 데이터 형식이 사용됩니다. 특정 srv가 어떻게 구성되어 있는지 알고 싶을 때는 ros2 interface show를 사용합니다.

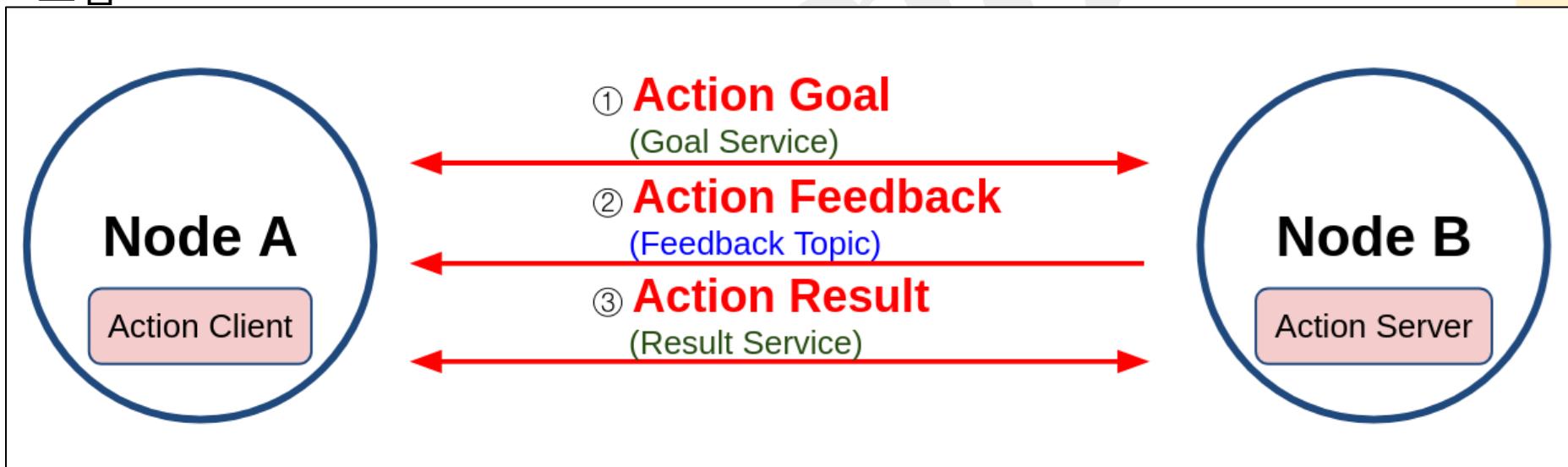
```
$ ros2 interface show gazebo_msgs/srv/SpawnEntity
string name # Name of the entity to be spawned (optional).
string xml # Entity XML description as a string, either URDF or SDF.
string robot_namespace # Spawn robot and all ROS interfaces under this namespace
geometry_msgs/Pose initial_pose # Initial entity pose.
string reference_frame # initial_pose is defined relative to the frame of this entity.
# If left empty or "world" or "map", then gazebo world frame is
# used.
# If non-existent entity is specified, an error is returned
# and the entity is not spawned.
-----
bool success # Return true if spawned successfully.
string status_message # Comments if available.
```

ROS2 구성

action 이란

설명: 액션(action)은 토픽(topic)과 서비스(service)가 혼합된 형태의 통신방식

액션 클라이언트(client)가 서비스를 통해 목표(goal)를 설정하면 액션 서버(server)는 결과(result)에 도달할때까지 토픽을 통해 피드백(feedback)을 보냄



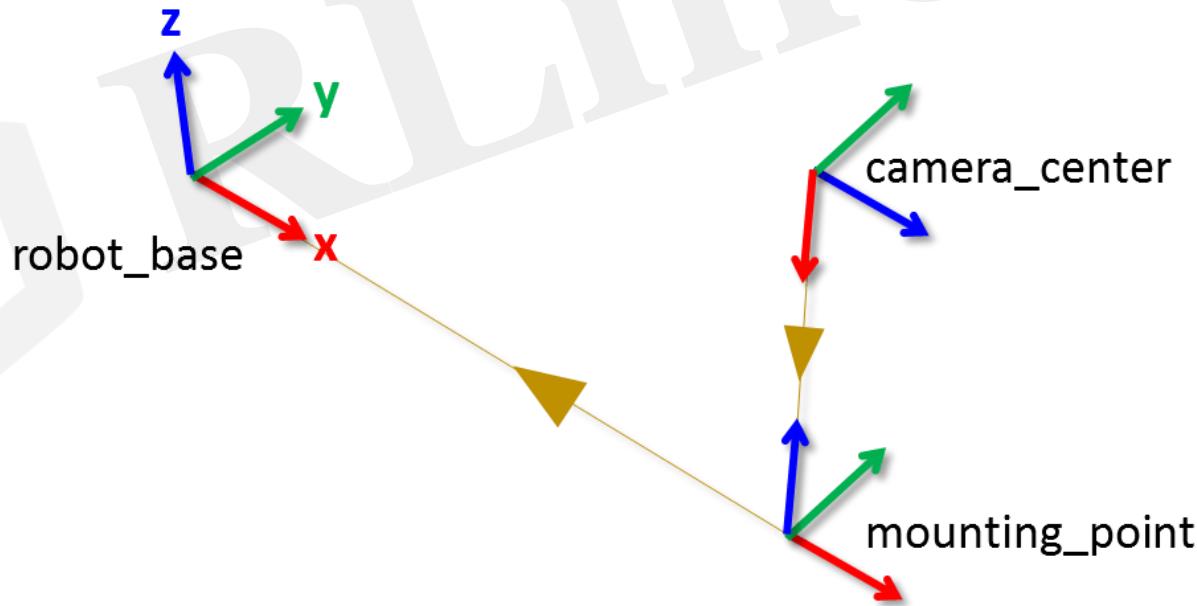
Transform

tf2 란

설명: ros에서 tf란 프레임을 추적할 수 있게 해주는 ros의 라이브러리

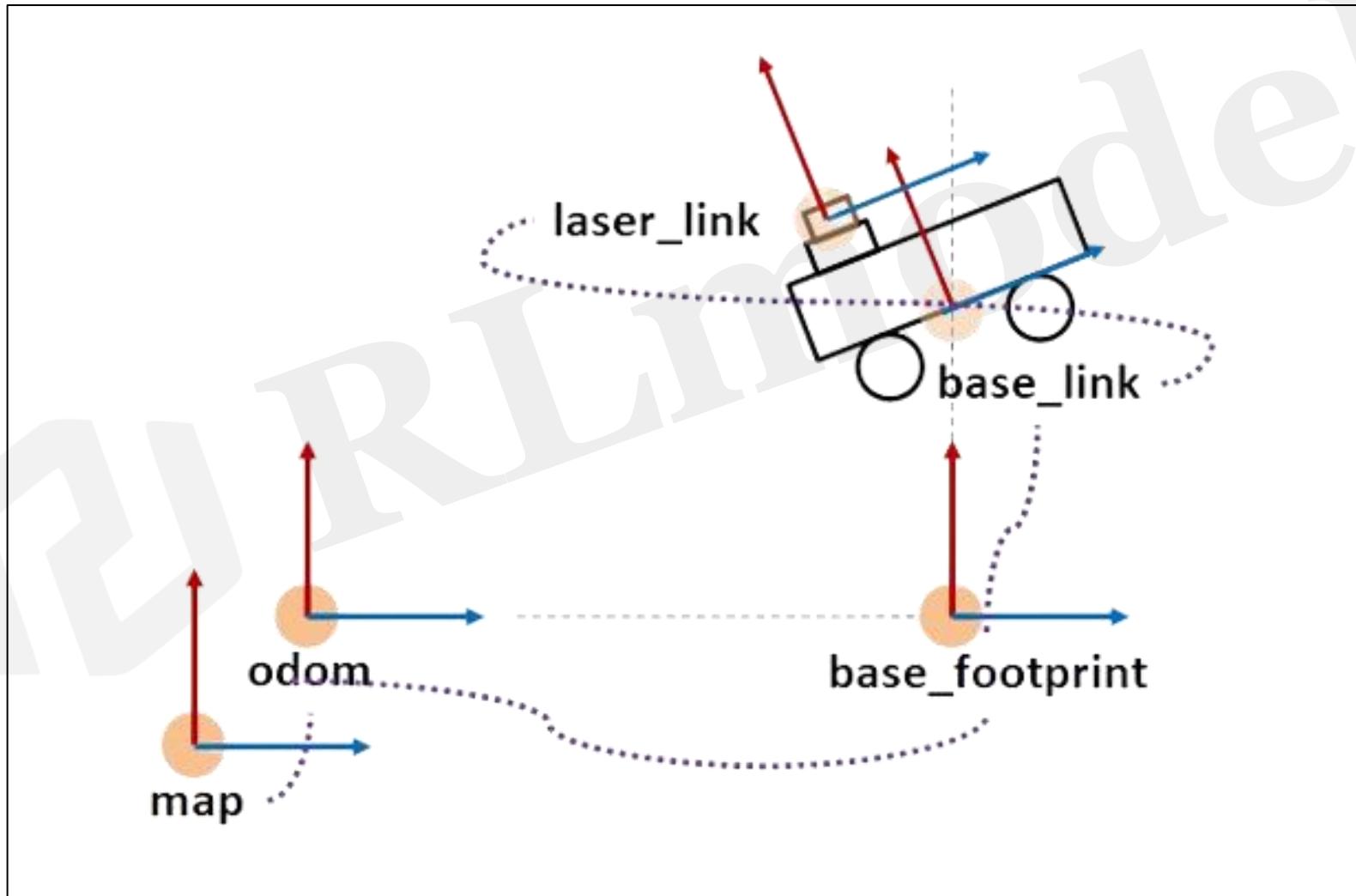
카메라, 로봇, 센서 등의 모듈마다 프레임이 존재하고 각 프레임간의 관계를 알아낼 수 있는 라이브러리

사용/설정: urdf , 혹은 broadcaster



Transform

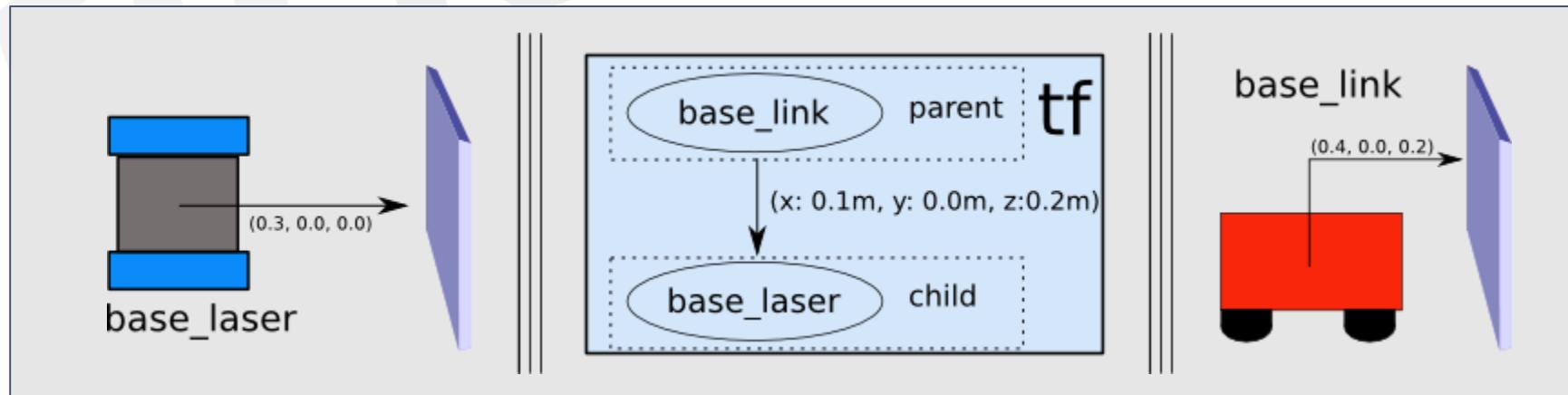
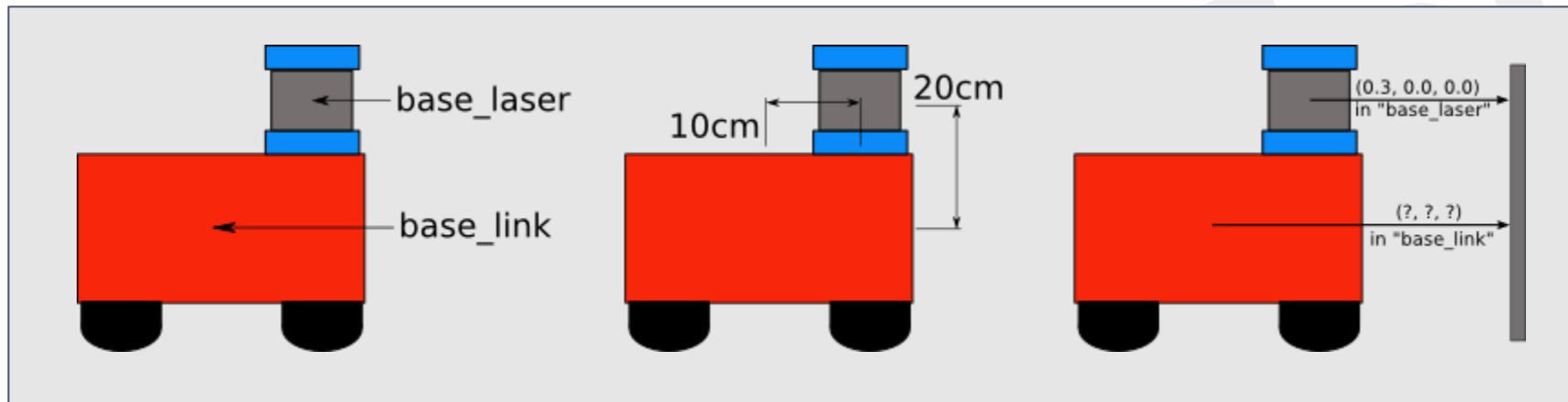
robot tf 구성 예시



Transform

tf2 란

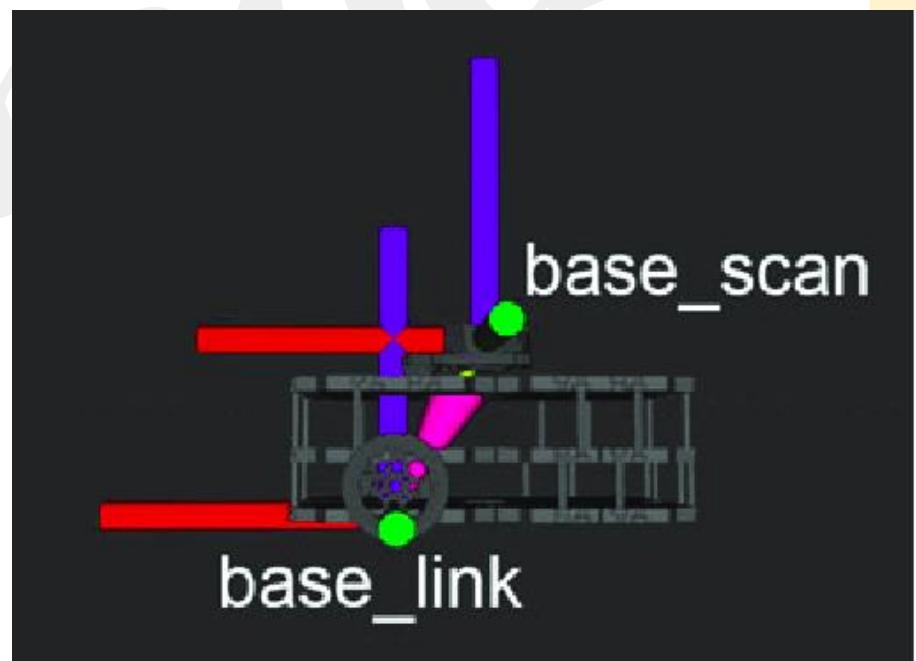
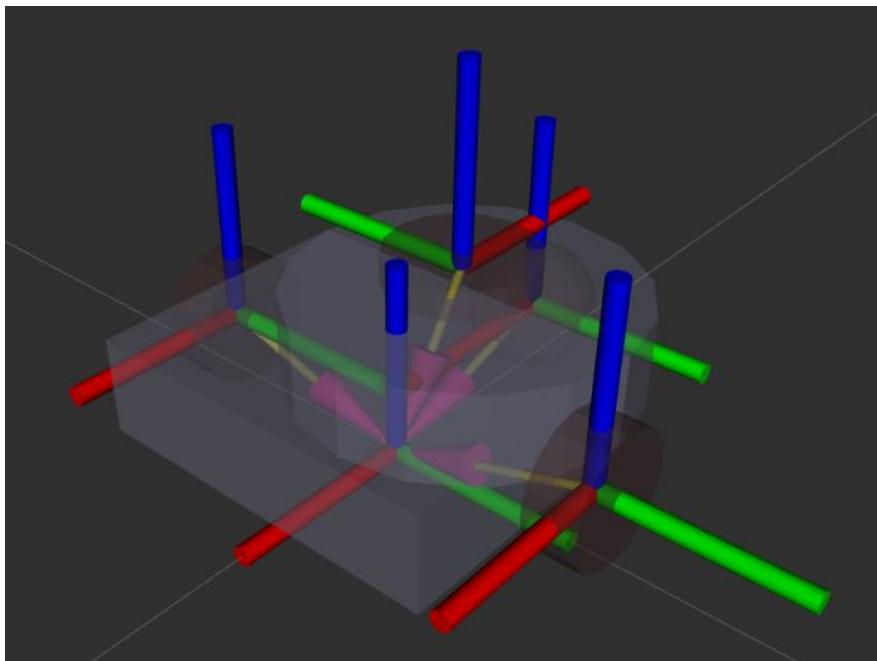
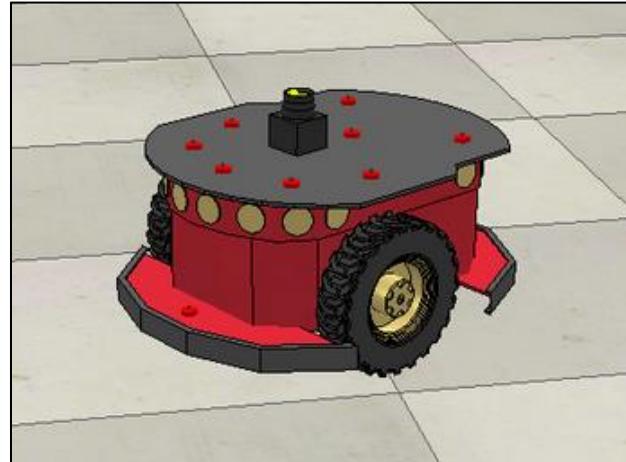
예시: rviz2



Transform

tf2 시작화

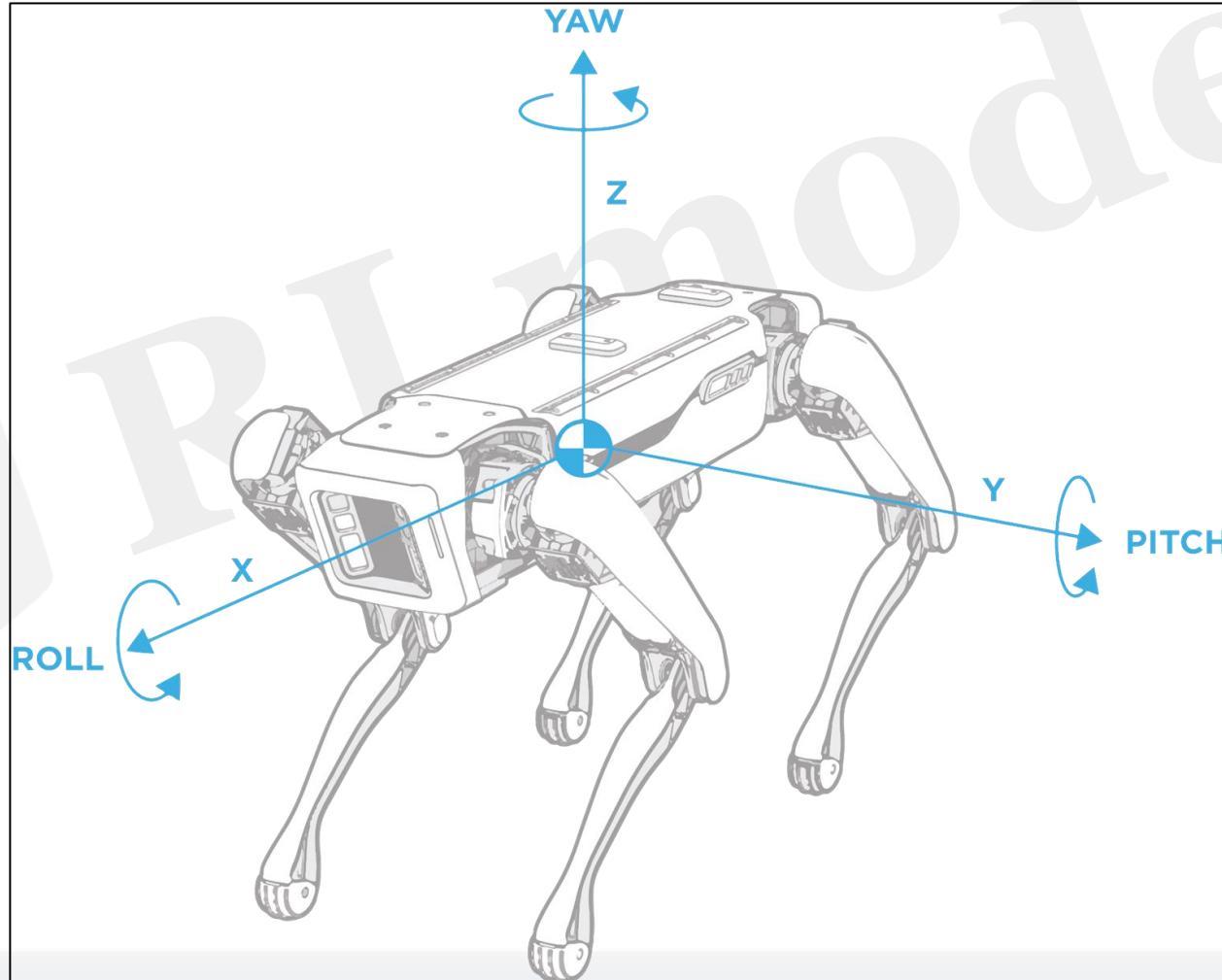
예시: rviz2



Transform

tf2 시작화

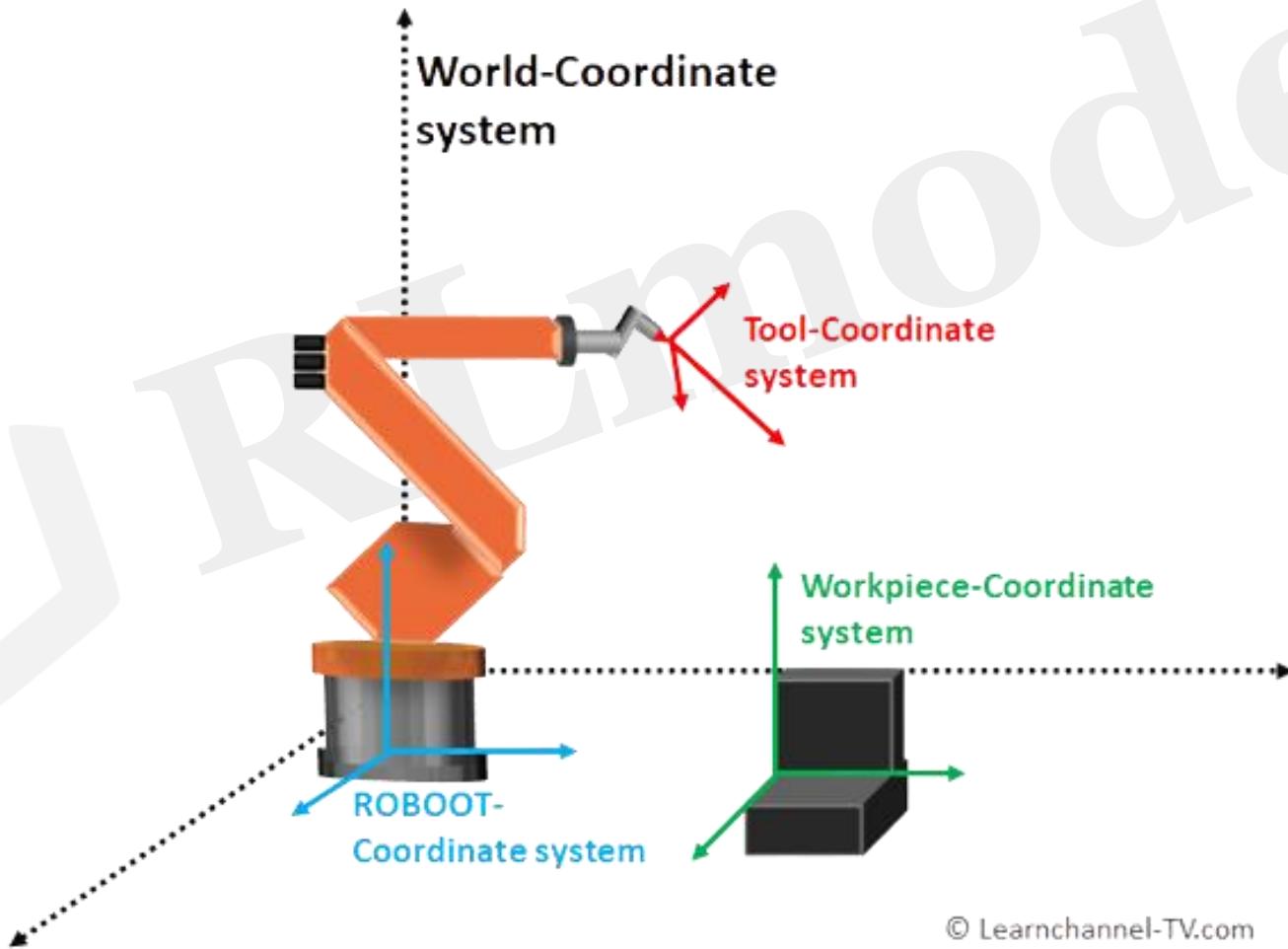
예시: robot dog



Transform

manipulator tf

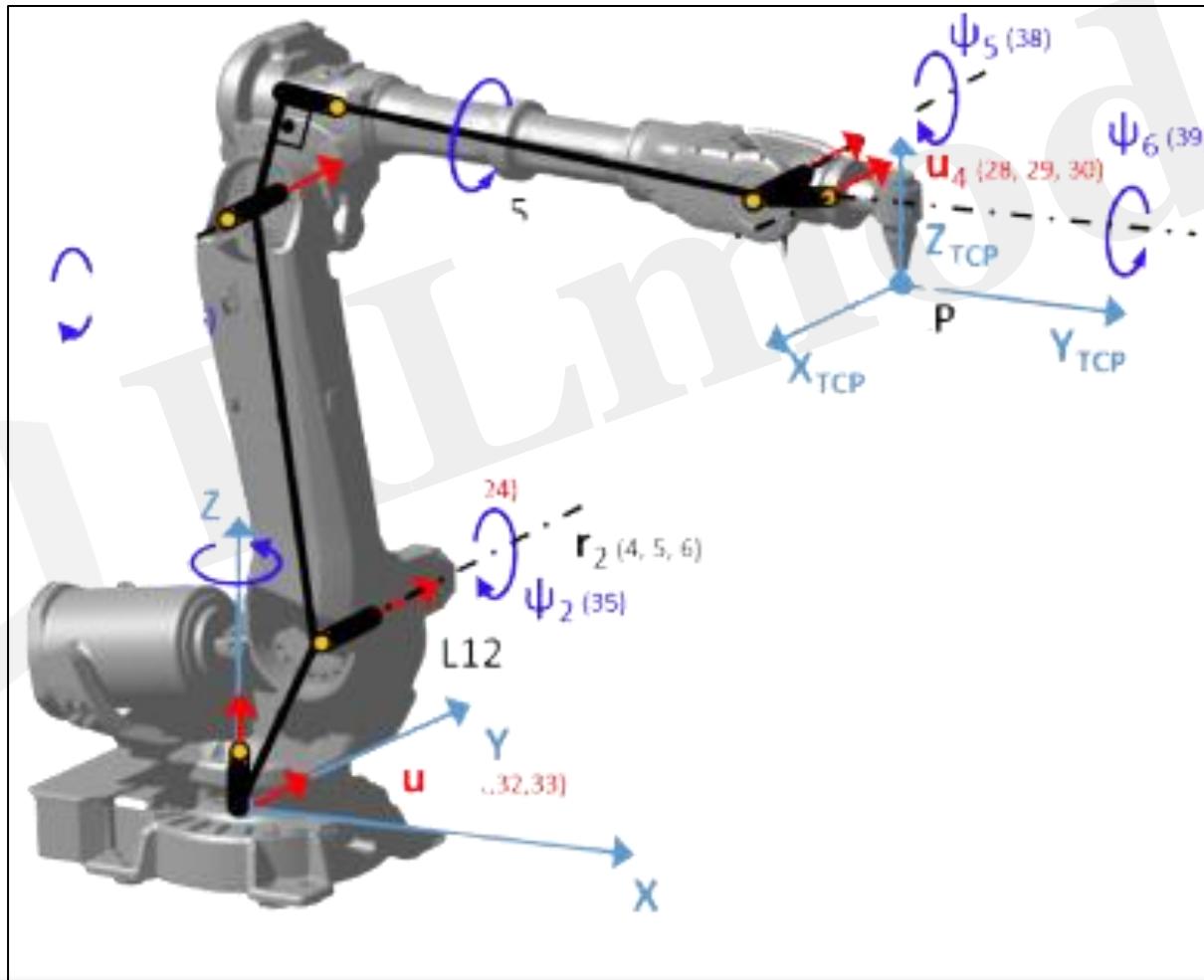
예시): robot arm DOF



Transform

manipulator tf

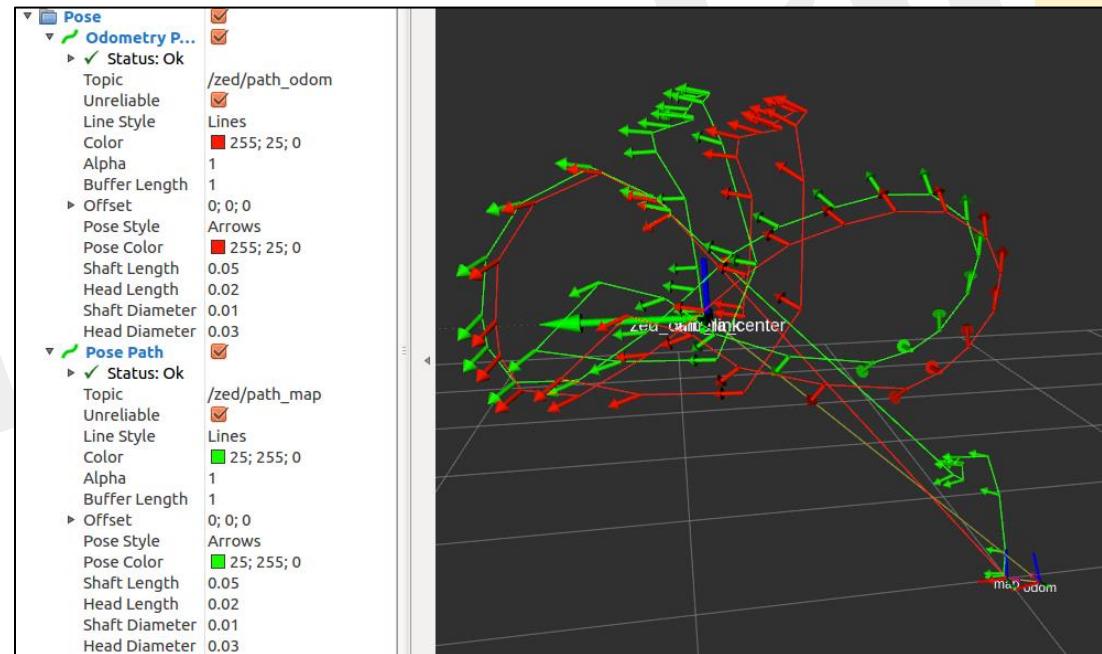
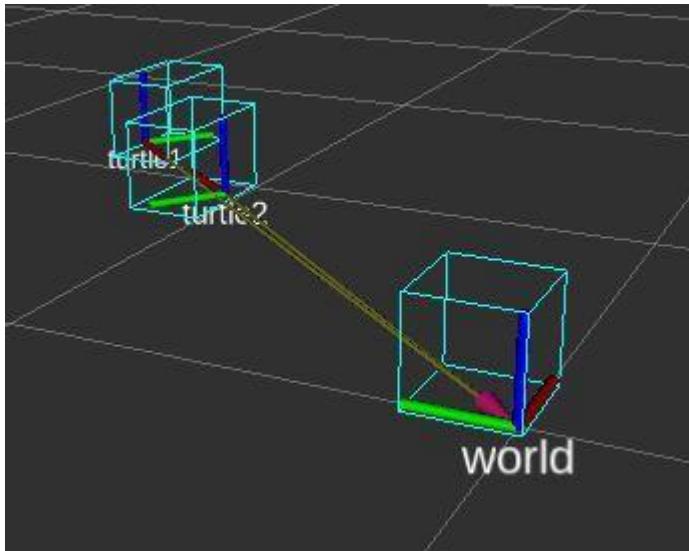
예시: 6 DOF(Degree Of Freedom) robot arm



Transform

tf2 표시

> rviz 상 시각화 상태

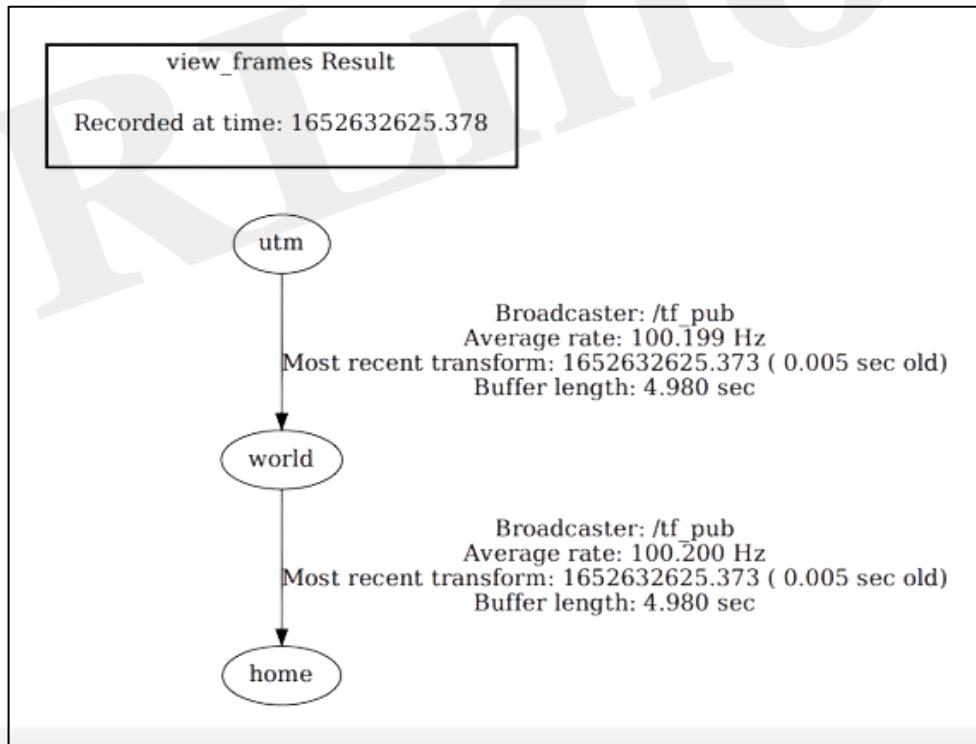


Transform

frame 이란

> frame은 ROS 상에 존재하는 수많은 좌표계를 의미하며, TF는 각 프레임 사이의 관계를 정의하는 것

- 모든 ROS의 프레임은 유클리드 좌표계로 구성
- 각각의 프레임은 서로 독립된 상태
- frame_id 를 통하여 각각의 좌표계를 서로 구분.



Transform

정리

map: world fixed frame(세계 고정 좌표 프레임)이다. Z축은 위를 향한다.

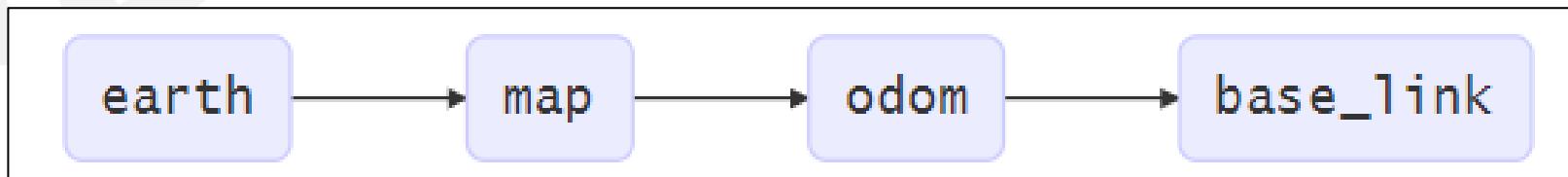
odom: 로봇의 Odometry(오도메트리) 정보를 기준으로 정의된 좌표계

base_footprint: base_footprint는 바닥에 대한 로봇 위치

base_link: base_link는 모바일 로봇 베이스에 부착.

gaze: head 위치 및 방향을 정의한 좌표 프레임이다. 카메라같은 센서와 관련해 보는 방향을 정의한다.

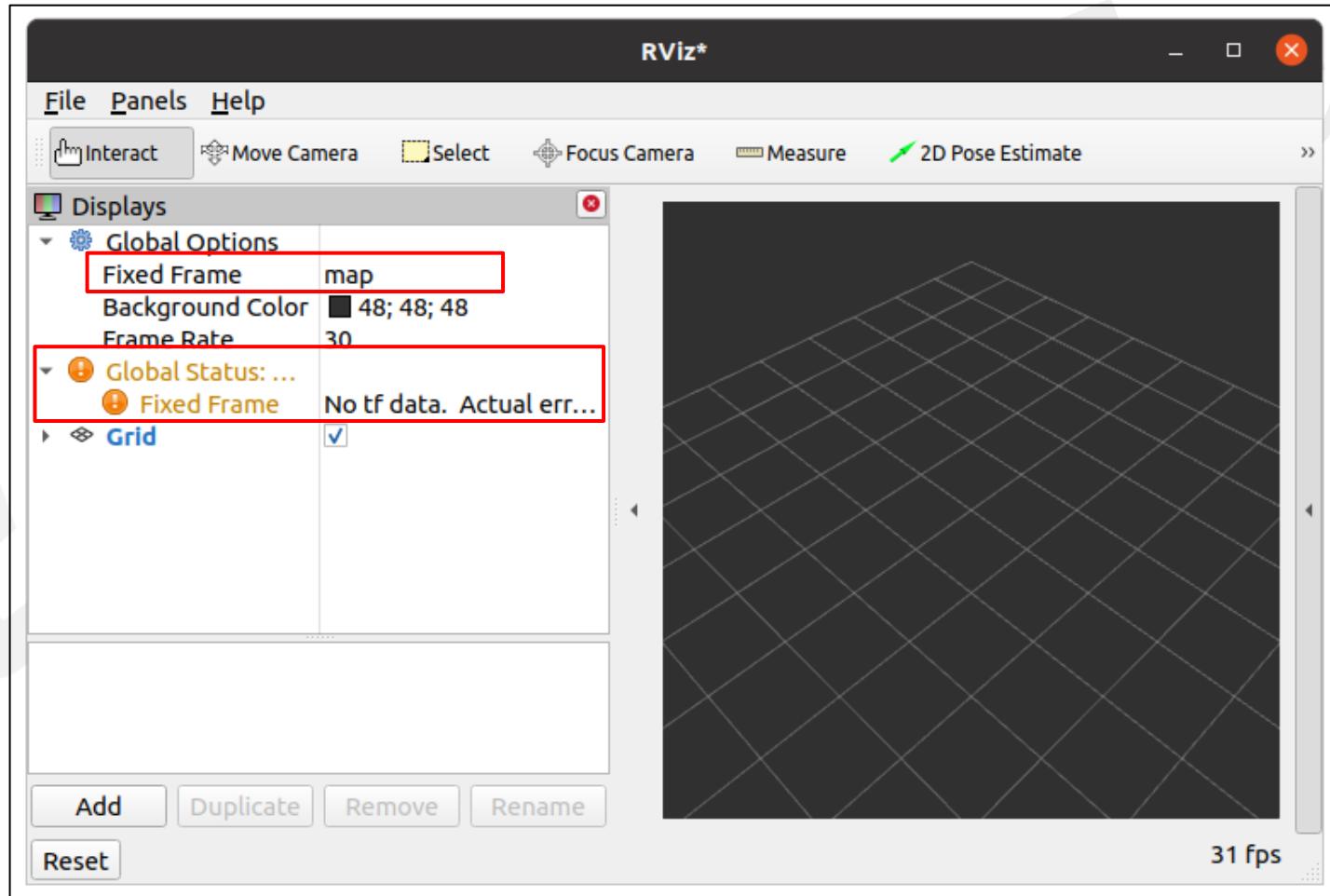
earth: ECEF의 원점이다. 이 프레임은 서로 다른 맵 프레임에서 여러 로봇의 상호 작용을 허용하도록 설계되었다.



Transform

Rviz

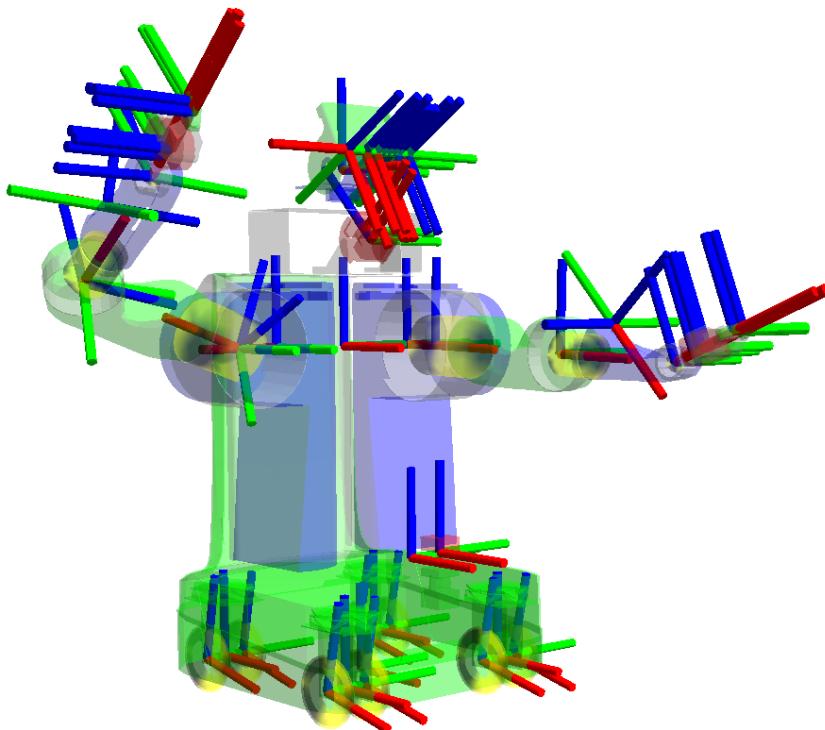
Fixed Frame: 기준 좌표계



Transform

기본설명

static



URDF

A tree of...

Links

connected by

Joints

TF

A tree of...

Frames

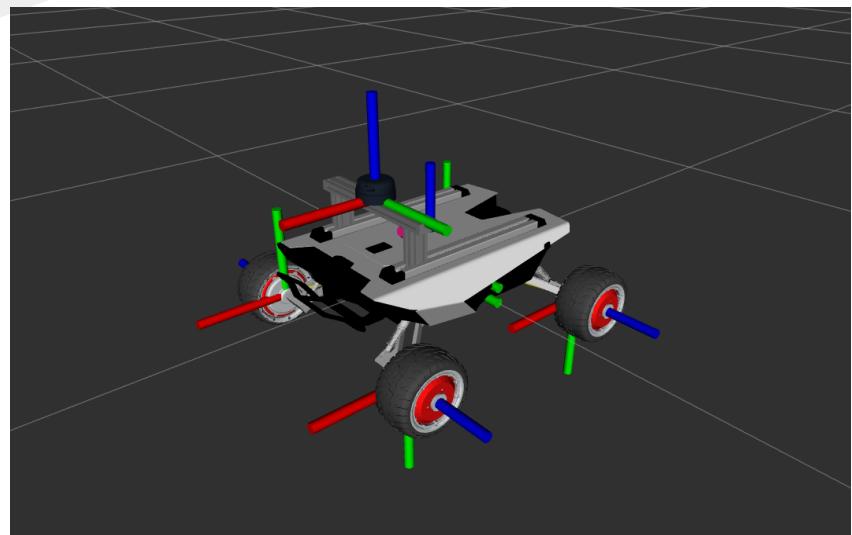
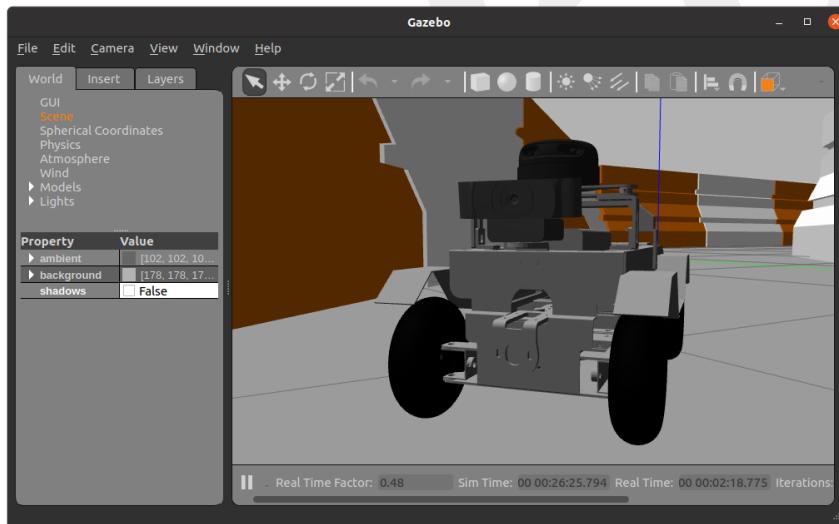
connected by

Transforms

URDF

URDF 란

- 설명: URDF는 Universal Robot Description Format의 약자로, 로봇의 물리적 구조를 정의하기 위한 XML 형식
- 목적: 로봇의 각 부위와 이들 간의 관계를 정의하여 시뮬레이션, 시각화 및 제어에 사용됩니다.
- ROS에서의 역할: ROS에서 로봇의 모델을 정의하고 이를 기반으로 시뮬레이션 및 실제 로봇 제어를 수행할 때 사용됩니다.





URDF

URDF 파일 구조

- ❖ URDF 파일은 여러 개의 태그로 구성되며, 주요 태그는 다음과 같습니다.
 - **<robot>**: URDF 파일의 루트 태그로, 로봇 전체를 감싸는 태그
 - **<link>**: 로봇의 각 부위를 정의
 - **<joint>**: 로봇 부위 간의 연결을 정의합니다



URDF

URDF 파일 구조

1. <robot> 태그

- 로봇의 링크, 조인트를 포함하여 여러가지 태그를 포함합니다.

xml 코드 복사

```
<robot name="my_robot">
    .
    <!-- 로봇의 링크와 조인트가 여기에 정의됩니다 -->
</robot>
```

URDF 파일 구조

2. <link> 태그

- **기본 구조:** 로봇의 각 부위를 나타내며, 각 부위는 시각적 모델, 충돌 모델, 관성 정보 등을 포함할 수 있습니다.

xml

코드 복사

```
<link name="base_link">
  <visual>
    <geometry>
      <box size="1 1 1"/>
    </geometry>
  </visual>
</link>
```



URDF

URDF 파일 구조

3. <joint> 태그

- **기본 구조:** 두 링크를 연결하고, 연결된 링크 간의 상대적인 움직임을 정의합니다.

xml 코드 복사

```
<joint name="base_to_link1" type="continuous">
  <parent link="base_link"/>
  <child link="link1"/>
  <origin xyz="0 0 1" rpy="0 0 0"/>
</joint>
```

URDF

URDF 작성 예

```
Release Notes: 1.89.0  bunker_pro.urdf.xacro •  
bunker_bringup > urdf > bunker_pro.urdf.xacro  
  2   <robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="urdf_tutorial">  
  3     <link name="base_link">  
  4       <visual>  
  5         <origin xyz="0 0 0"/>  
  6         <geometry>  
  7           <box size="1.04 0.85 0.45"/>  
  8         </geometry>  
  9         <material name="white"/>  
 10       </visual>  
 11       <collision>  
 12         <origin xyz="0 0 0"/>  
 13         <geometry>  
 14           <box size="1.04 0.85 0.45"/>  
 15         </geometry>  
 16       </collision>  
 17     </link>  
 18  
 19     <!-- LEFT WHEEL LINK -->  
 20  
 21     <link name="left_wheel">  
 22       <visual>  
 23         <geometry>  
 24           <cylinder radius="0.06" length="0.06"/>  
 25         </geometry>  
 26         <material name="black"/>  
 27     </link>
```



URDF

URDF vs TF

1. 목적과 사용 분야:

- **URDF**: 로봇의 물리적 구조를 정의하는데 중점을 둡니다. 로봇의 각 부위와 이들 간의 관계를 기술하여 시뮬레이션, 시각화, 제어에 사용됩니다.
- **TF**: 로봇의 각 부위 간의 좌표 변환을 실시간으로 관리합니다. 로봇의 센서 데이터 처리, 로봇 팔의 이동 경로 계산 등에 사용됩니다.

1. 형식과 구성 요소:

- **URDF**: XML 형식으로 링크와 조인트를 정의합니다. 각 링크의 시각적 모델, 충돌 모델, 관성 정보 등을 포함합니다.
- **TF**: ROS의 TF 라이브러리 및 메시지 시스템을 사용합니다. 좌표 변환과 관련된 정보를 트리 구조로 관리합니다.



URDF

URDF vs TF

3. 데이터의 동적 관리:

- **URDF**: 주로 정적 데이터(로봇의 물리적 구조)를 정의하며, 런타임 동안 변하지 않습니다.
- **TF**: 동적 데이터(시간에 따라 변하는 좌표 변환)를 관리하며, 런타임 동안 지속적으로 업데이트됩니다.

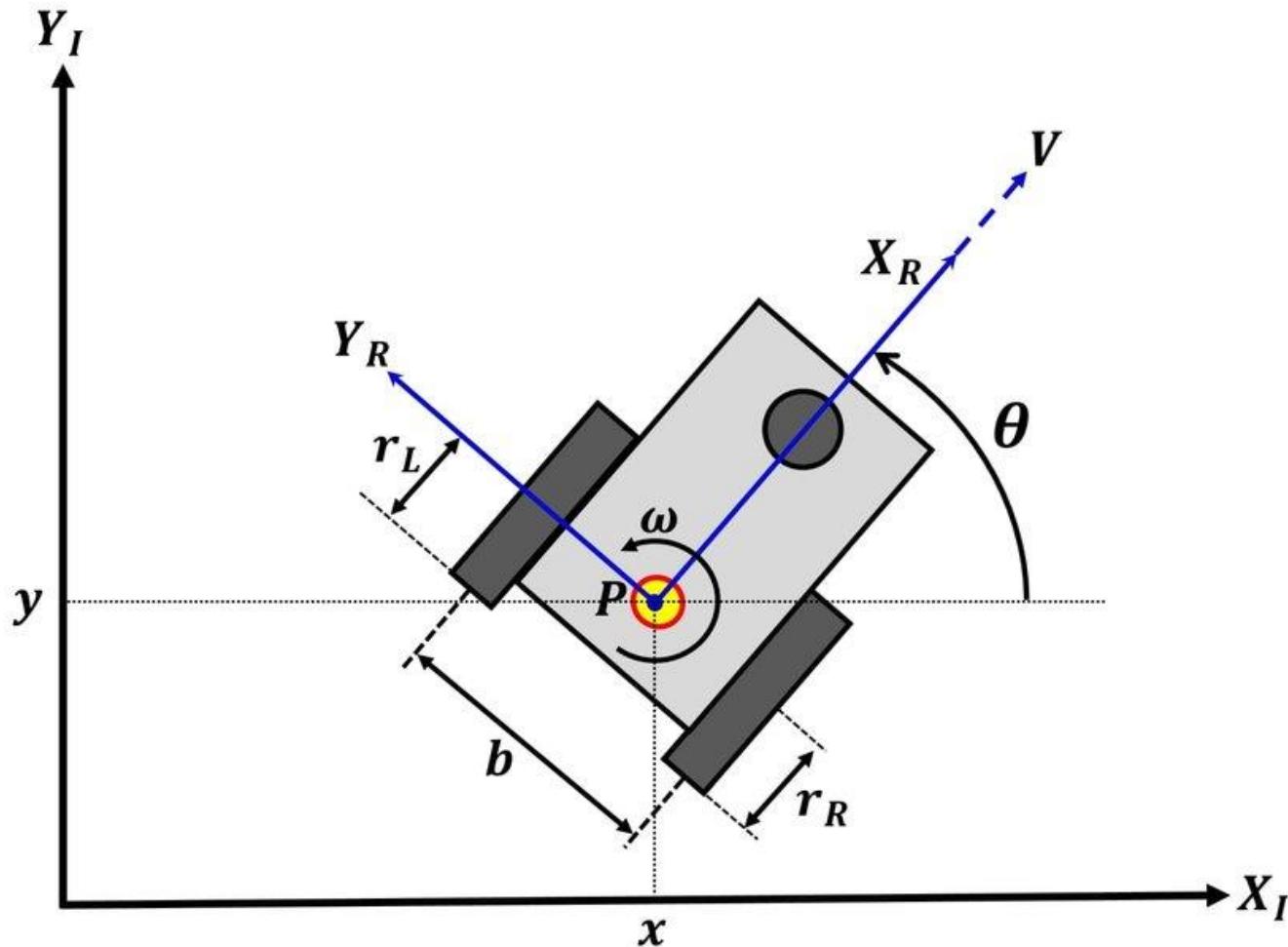
4. 형식과 구성 요소:

- **URDF**: RViz, Gazebo 등의 시각화 및 시뮬레이션 도구에서 로봇 모델을 정의하는데 사용됩니다.
- **TF**: RViz에서 좌표 변환을 시각화하거나, ROS 시스템에서 로봇의 동작을 추적하고 제어하는데 사용됩니다.

ROS2 구성

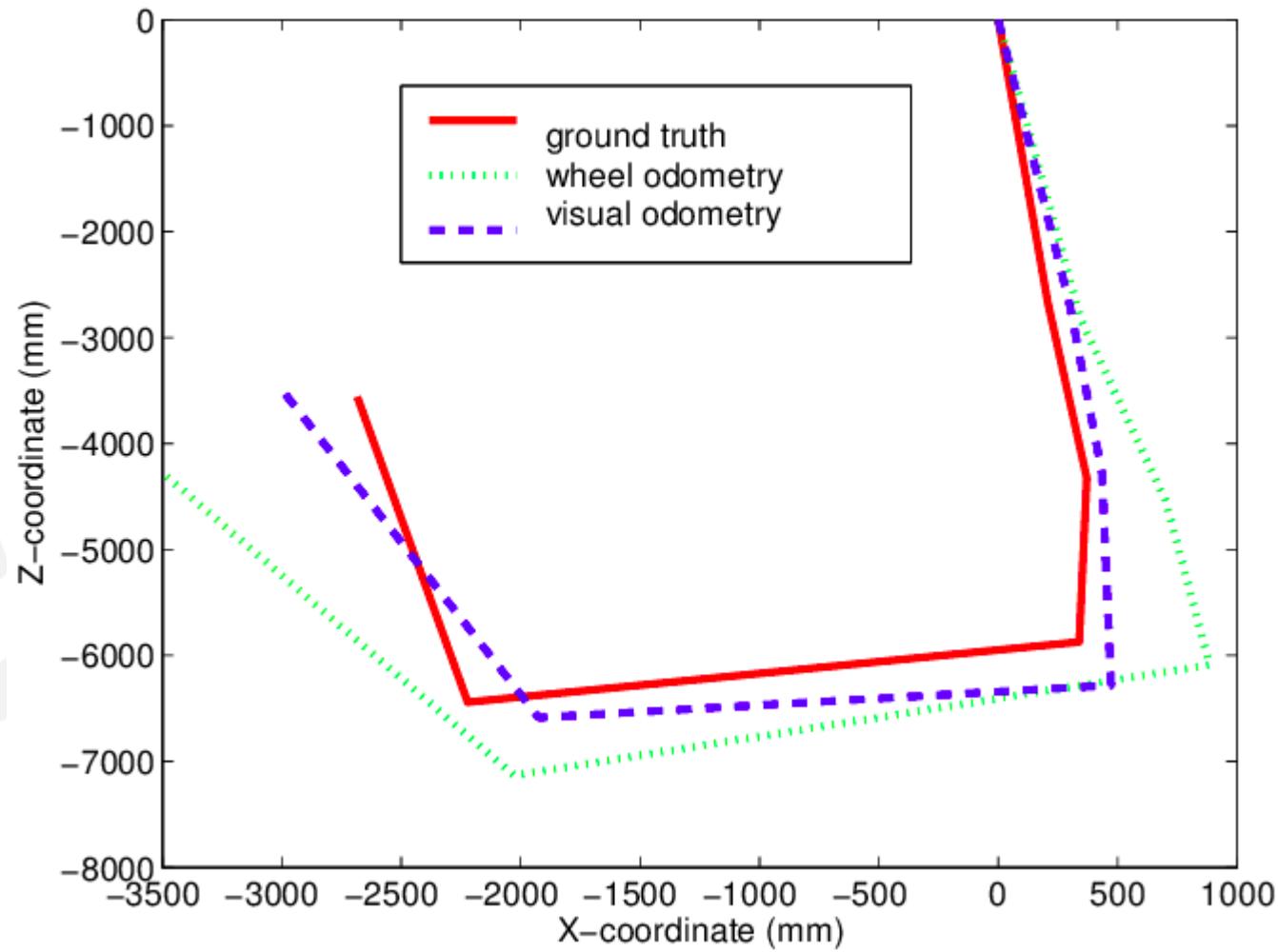
Odometry 란

odom: 로봇의 Odometry(오도메트리) 정보를 기준으로 정의된 좌표계



ROS2 구성

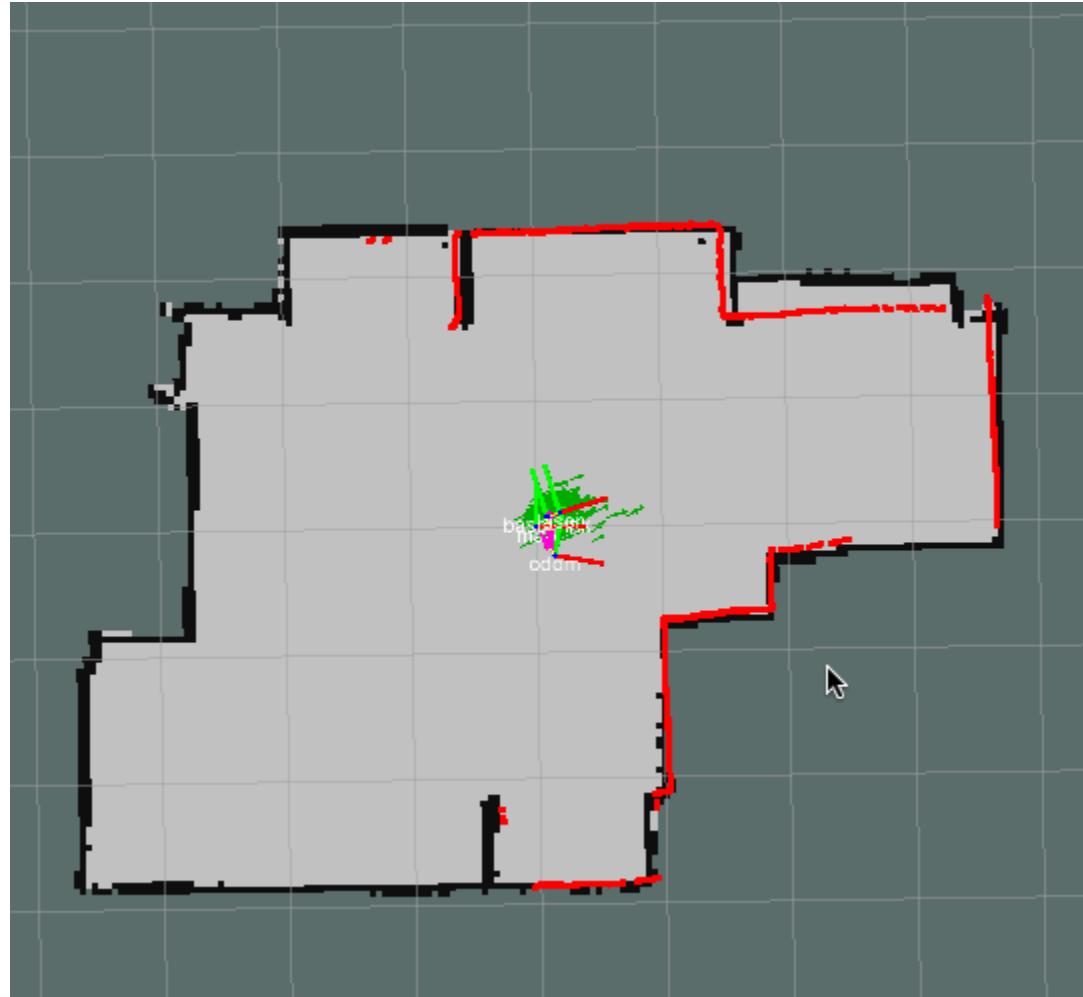
Odometry 비교



ROS2 구성

Odometry 계산표시

robot localization and AMCL



ROS2 구성

Odometry 란

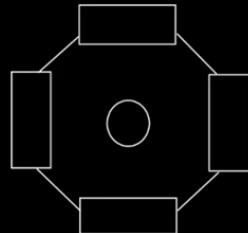
link: <https://www.youtube.com/watch?v=ZW7T6EFyYnc&t=19s>

What is Odometry?

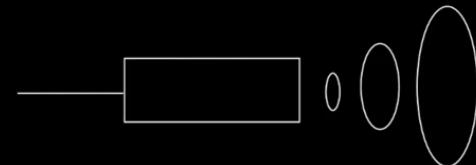
Visual



Encoder



Distance

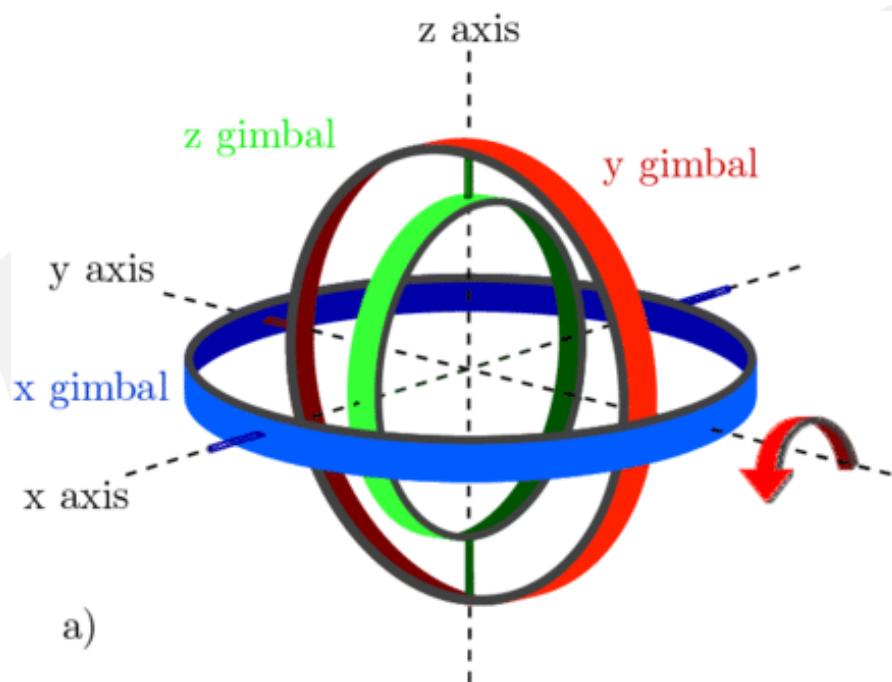
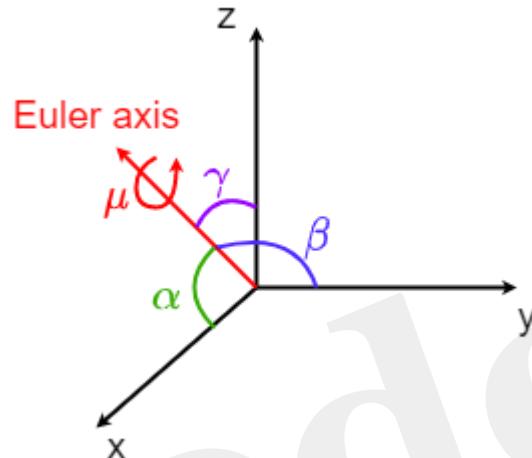


ROS2 구성

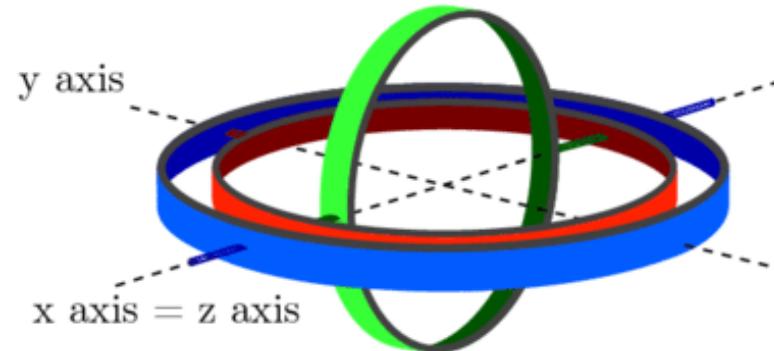
tf2 참고

Euler vs **Quaternion**

>Gimbal lock



a)



b)

ROS2 구성

tf2 참고

Quaternion

> 쿼터니언은 본래 아일랜드의 수학자 해밀턴(William R. Hamilton)이 1843년에 복소수를 확장시켜 만든 수체계(number system)다. 복소수가 2차원 평면상의 점을 표현할 수 있다는 사실로부터 3차원 공간상의 점을 표현하는 방법을 찾다가 만들었다고 한다

쿼터니언은 x, y, z, w **4개의 성분**으로 구성됩니다.

각 성분은 x, y, z 벡터와 스칼라입니다.

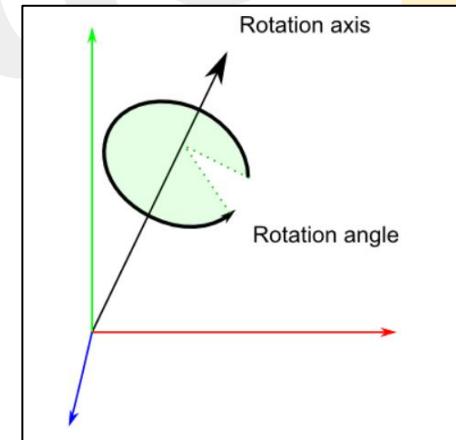
쿼터니언은 모든 축을 한 번에 계산하기 때문에 짐벌락 문제가 생기지 않으며

방향과 회전을 모두 표현할 수 있습니다

$$R_x(\phi) = \text{Roll}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}$$

$$R_y(\theta) = \text{Pitch}(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

$$R_z(\psi) = \text{Yaw}(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

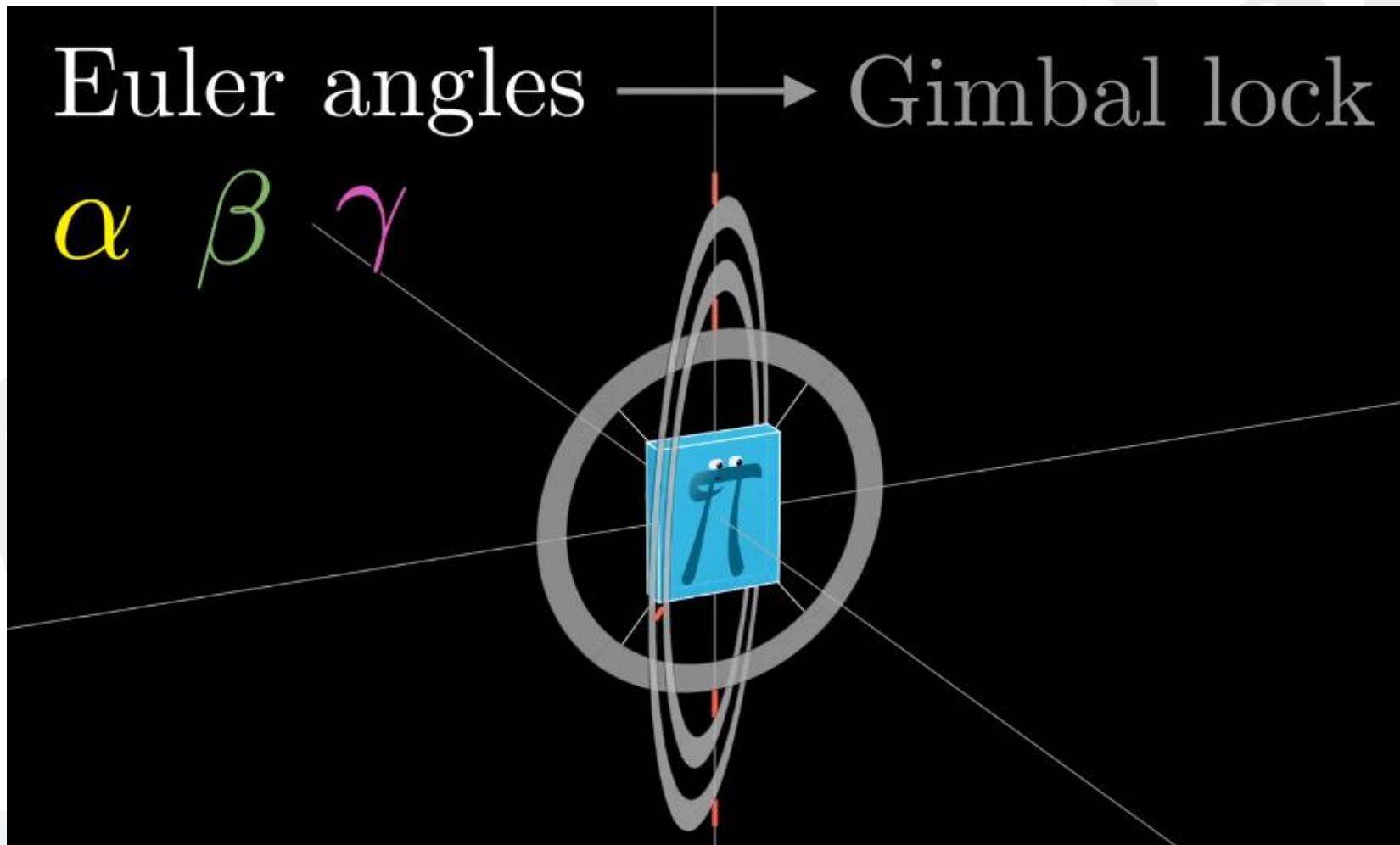


$$\left\{ \begin{array}{l} q_0 = \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ q_1 = \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ q_2 = \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ q_3 = \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{array} \right.$$

ROS2 구성

tf2 참고

link: <https://www.youtube.com/watch?v=zjMulxRvygQ>



RQT

rqt 정의

RQt는 ROS + Qt의 합성어, 일종의 Framework

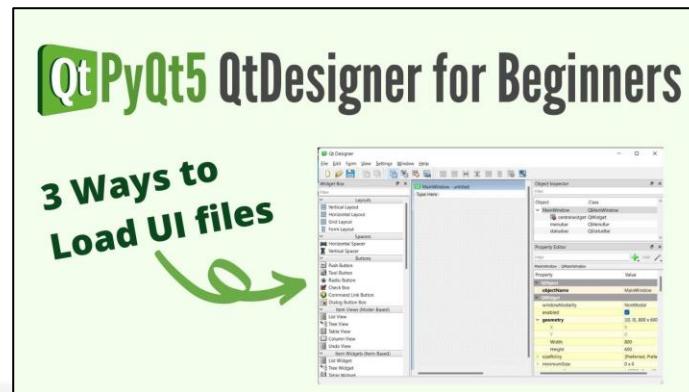
용도

ROS 환경에서 데이터를 쉽게 확인, 관리할 수 있는 모니터링 도구

특징

ROS 환경에서 사용할 수 있는 GUI 개발에 있어서 공통으로 필요한 부분들을 API형태로 제공

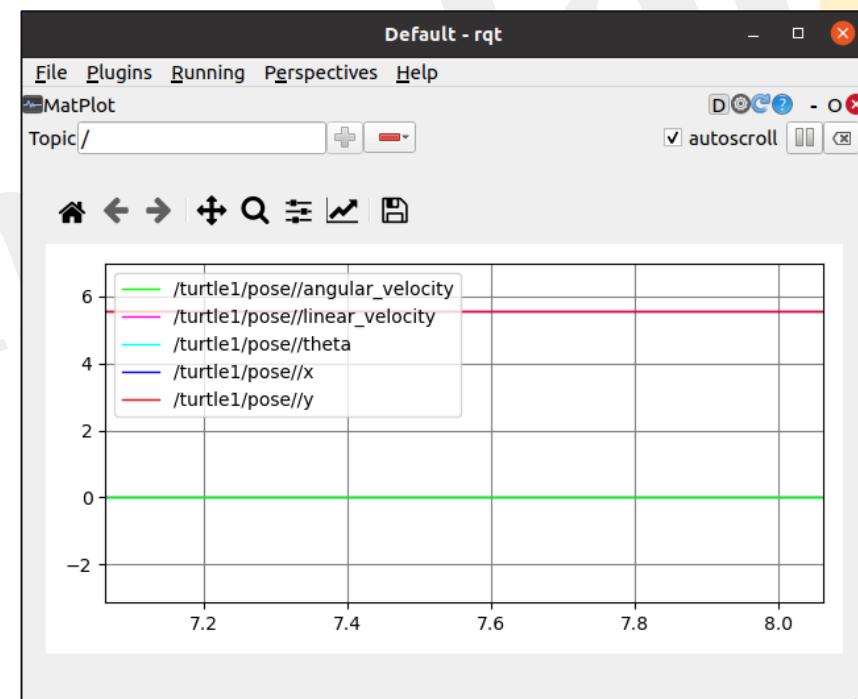
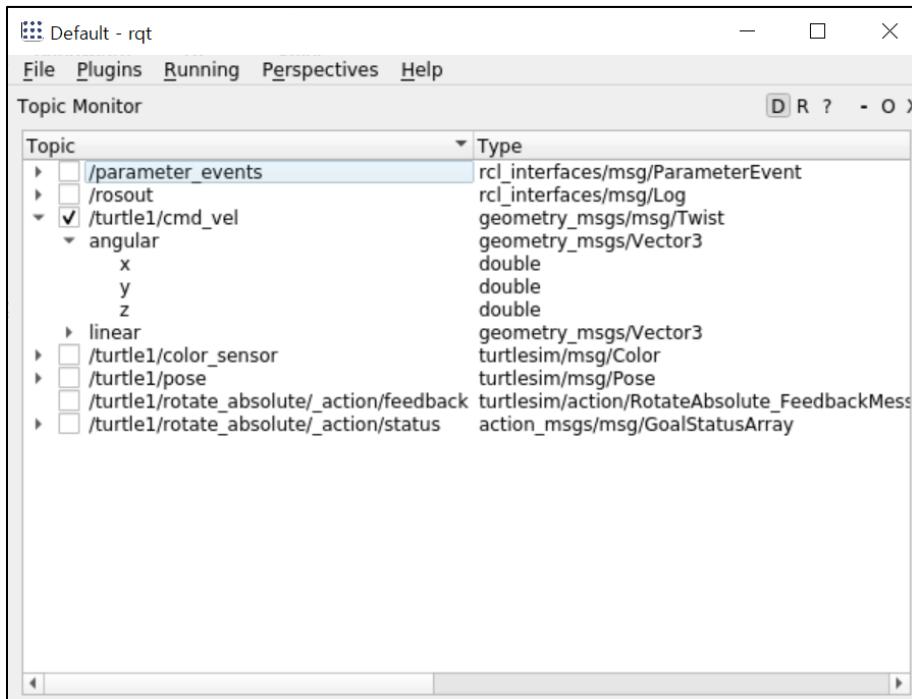
Qt의 특징인 크로스 플랫폼과 다양한 프로그래밍 언어 지원이라는 특징을 그대로 반영.



RQT

rqt 실행

```
$ rqt
```

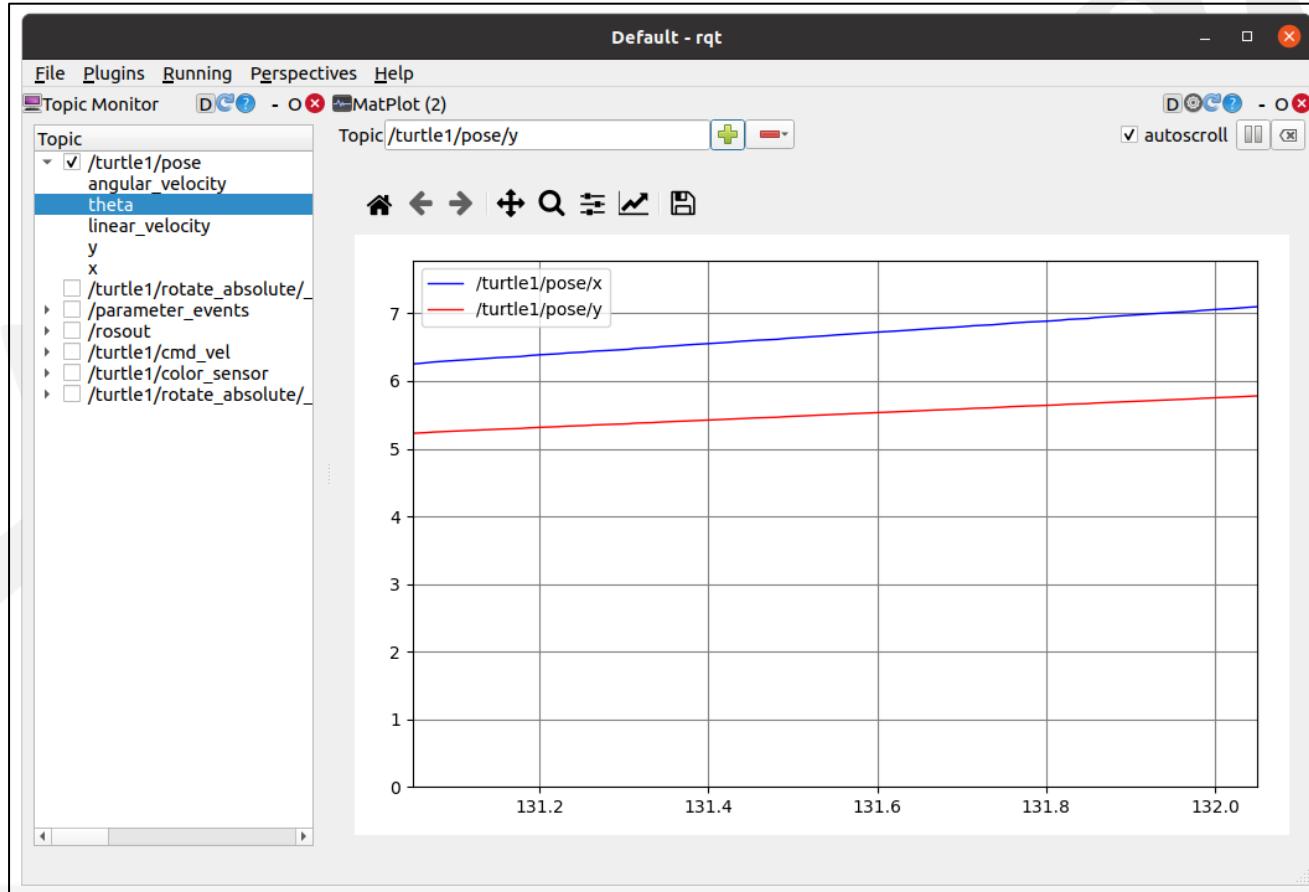


RQT

rqt 실행

ex) turtle pose x/y >

Topic: /turtle1/pose/y

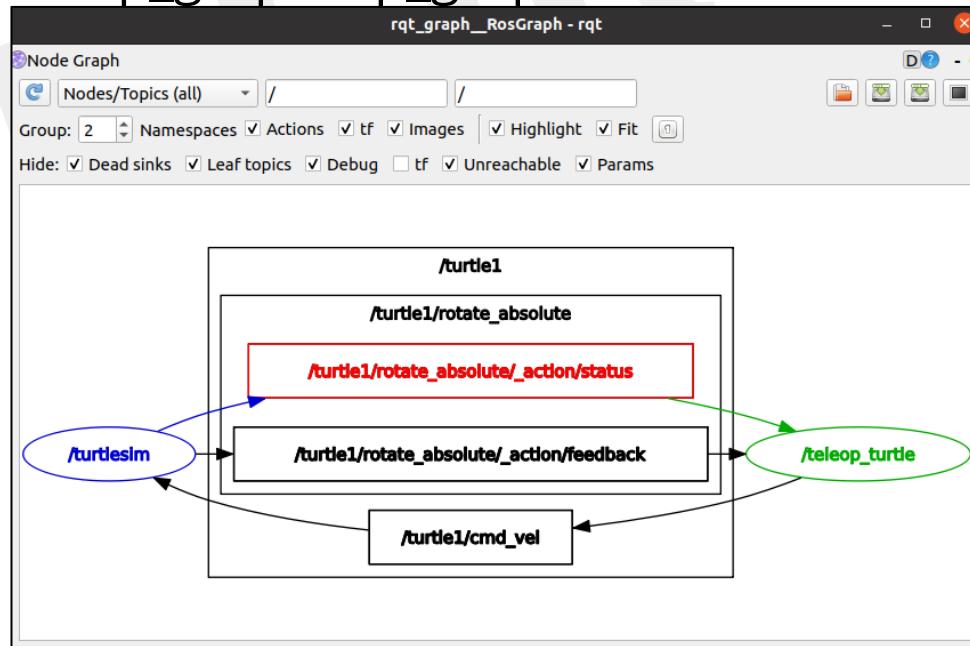


rqt_graph

rqt_graph 란

rqt_graph는 로봇 시스템의 통신 인프라를 시각화하기 위해 ROS(로봇 운영 체제) 프레임워크에서 사용되는 강력한 그래픽 도구입니다. ROS 네트워크 내에서 서로 다른 노드와 주제 간의 관계를 실시간으로 표현

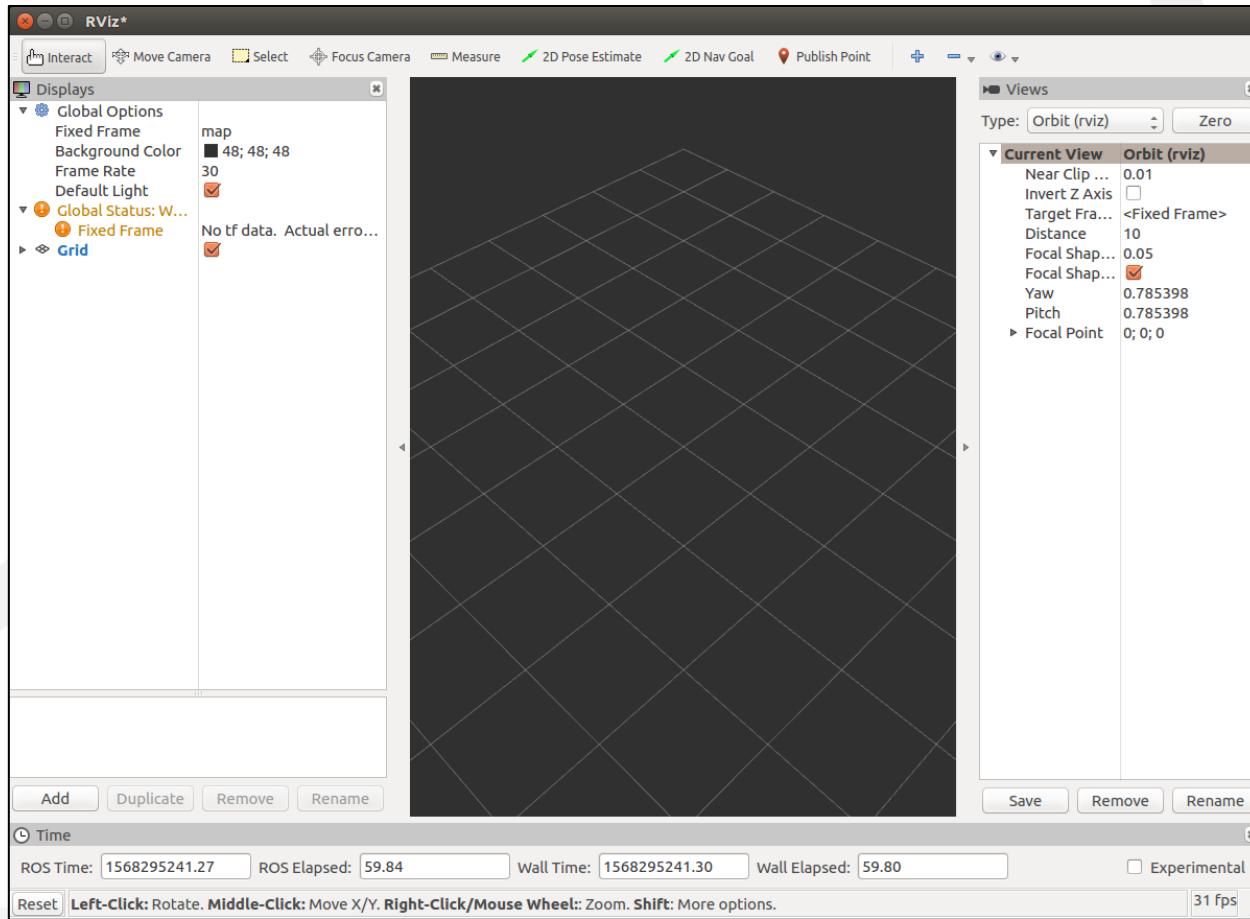
명령어: ros2 run rqt_graph rqt_graph



rviz 소개

rviz 란

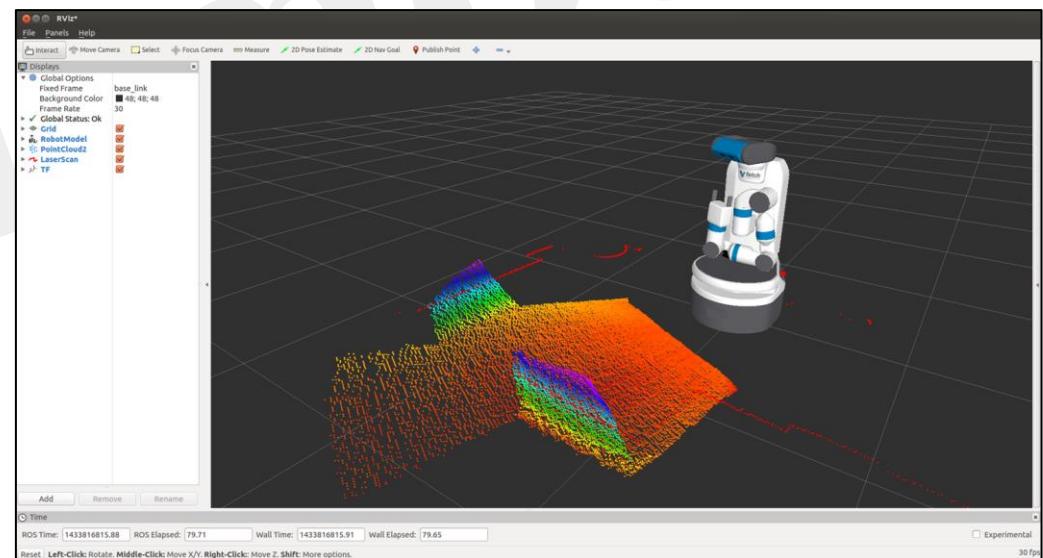
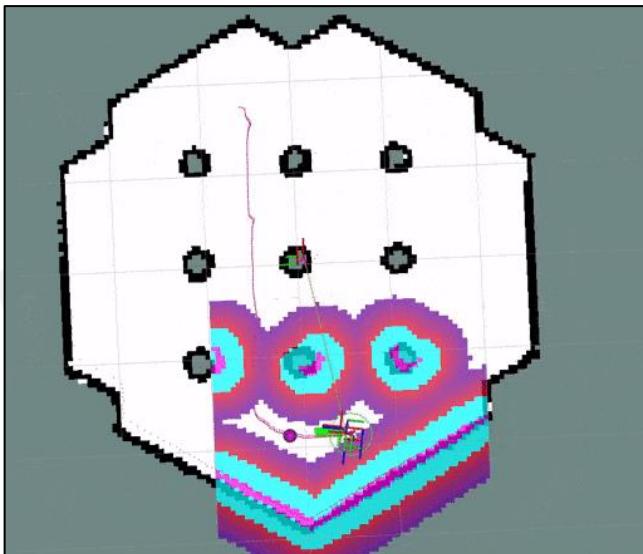
ROS Visualization Tool의 약자로 ROS 도구 중 하나다. 센서 데이터를 시각화하는 기능을 수행



rviz 소개

rviz 란

RViz는 ROS Visualization Tool의 약자로 ROS 도구 중 하나.
센서 데이터를 시각화, 레이저 거리 센서(LDS)의 거리 데이터, 카메라의
영상 데이터 로봇 외형의 표시, 원격 제어
ex) navigation

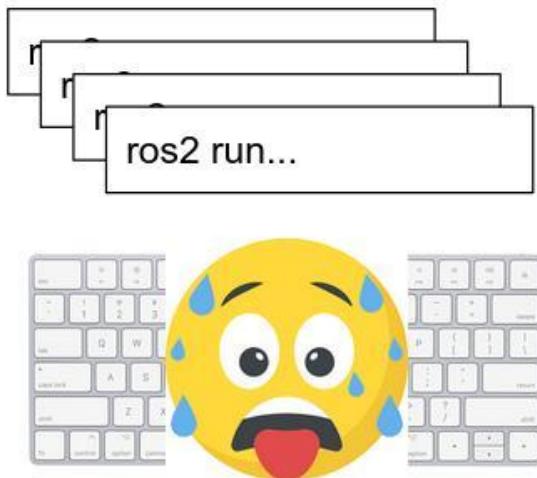


launch 파일

정의/설명

여러 개의 노드를 한번에 실행하는 .launch 확장자를 가진 파일.

Why Use a Launch File?





launch 파일

작성방법

- python, yaml, xml

```
#!/usr/bin/env python3

import os

from launch import LaunchDescription
from launch.actions import ExecuteProcess, IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration

from launch_ros.actions import Node

# this is the function launch system will look for
def generate_launch_description():

    # create and return launch description object
    return LaunchDescription([
        ExecuteProcess(
            cmd=["ros2", "run", "rviz2", "rviz2"], output="screen"
        ),
    ]
)
```



launch 파일

부가기능

namespace

Node 클래스를 사용할 때 namespace를 지정하면 모든 노드와 노드에 포함된 토픽, 서비스, 액션 이름을 일괄 변경 가능하다.

IncludeLaunchDescription

하나의 런치 파일로 동일 패키지의 노드 실행뿐만 아니라 다른 패키지의 노드 실행도 묶어 할 수 있어 유용하다.



launch 파일

사용방법

문법: \$ ros2 launch '패키지명' '런치파일명'

예시: \$

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel'),
            ]
        )
    ])
```



DDS

DDS 란

DDS(Data Distribution Service)는 OMG(<http://www.omgwiki.org/dds/>)에서 국제 표준으로 정한 실시간 데이터 분배 미들웨어

<특징>

- Publish/ Subscribe 구조 : 데이터를 Pub-Sub하는 구조.
- 성능 : pub/sub모델 채택으로 request/reply 모델에 비해 자연시간이 적고 단위 시간당 데이터 처리량이 높다.

이에 리얼타임 퍼포먼스가 가능 환경.

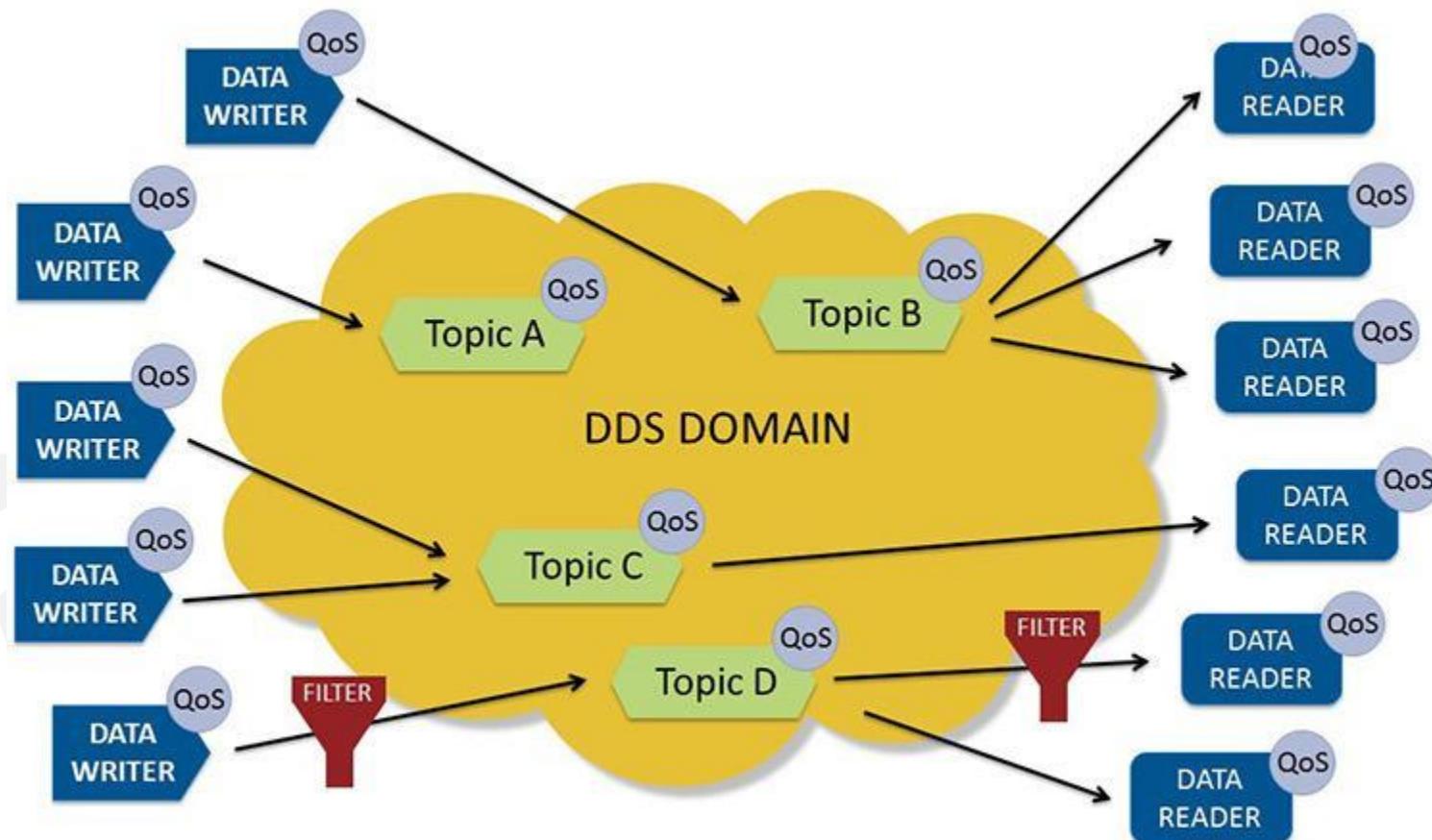
- Automatic Discovery of remote participants(Dynamic Discovery):

Remote participants 연결 용이. Discovery Protocol을 사용하여 퍼블리셔들이 서브스크라이버들의 존재를 파악하고 각 노드들이 서로 매칭합니다. >ROS2에서는 ROS Core가 사라짐

- QoS 파라미터 : reliability, persistence, redundancy, lifespan, transport settings, resources등, 통신에 있어 여러 설정값 변경 가능

DDS

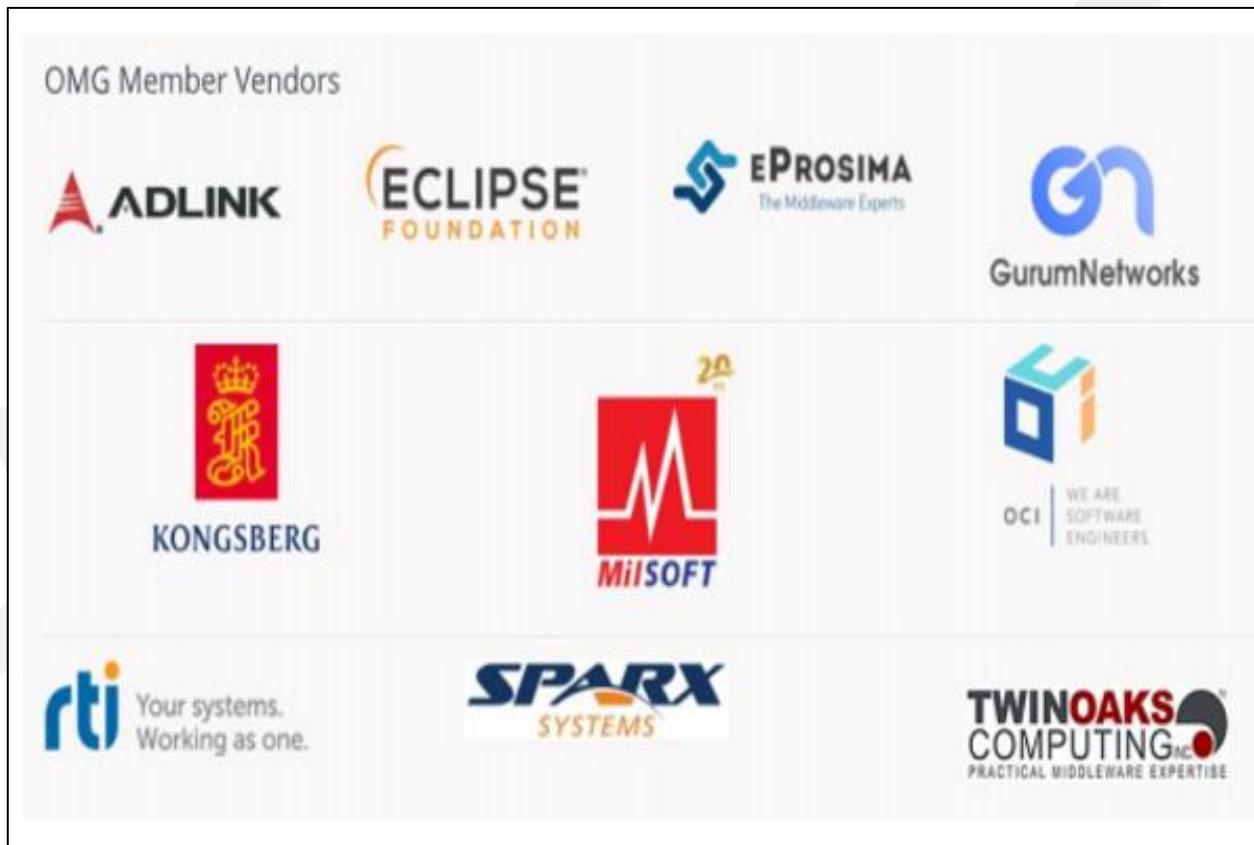
ROS2 - DDS 구조



DDS

DDS 공급사

ex) 국내업체: 구름DDS



DDS

DDS 공급사

ex) rti DDS

The screenshot shows a Firefox browser window displaying the RTI website at <https://www.rti.com/products/dds-standard>. The page has a dark blue background with a network graph pattern. The main heading is "DDS: An Open Standard for Real-Time Applications". Below it, a paragraph explains that DDS is a middleware protocol and API standard for real-time systems. A sidebar on the right contains links for "Contact Sales", "Video: What is DDS?", "Products", "Connectivity Selection Tool", "Professional Services", "Getting Started", "Resource Center", "RTI Community", and "OMG".

Free Trial

日本語 中文 Contact Us

DDS: An Open Standard for Real-Time Applications

The Object Management Group Data Distribution Service is a middleware protocol and API standard that provides data connectivity, extreme reliability and a scalable architecture to meet real-time system requirements.

RTI Connext® software includes the world's leading implementation of the Object Management Group (OMG) Data Distribution Service (DDSTM) standard. DDS is the only open standard for messaging that supports the unique needs of both enterprise and real-time systems. Its open interfaces and

Contact Sales

Video: What is DDS?

Products

Connectivity Selection Tool

Professional Services

Getting Started

Resource Center

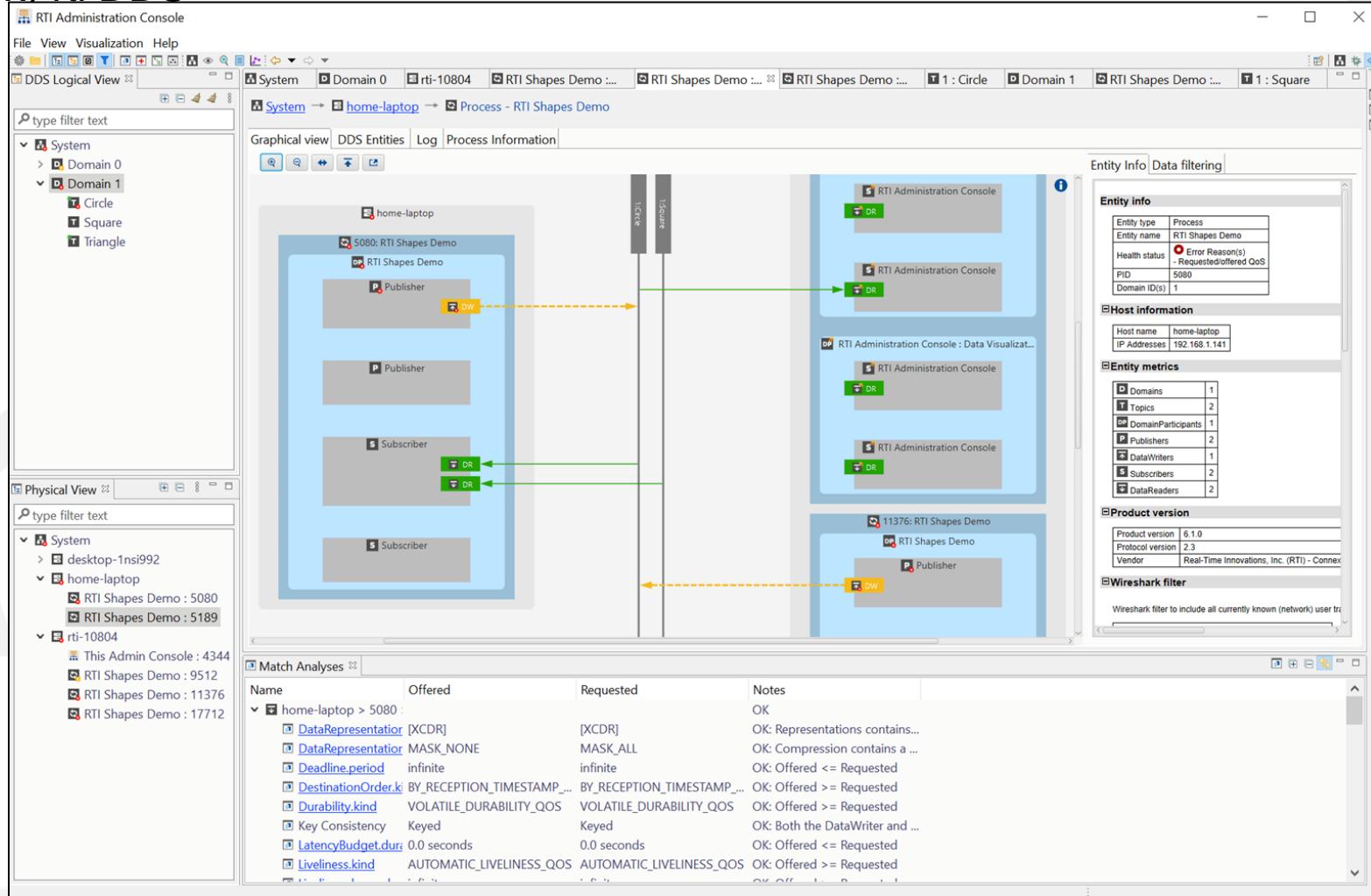
RTI Community

OMG

DDS

DDS 공급사

ex) rti DDS



DDS 연동

DDS 영상데모





ROS2 동작확인

기본 패키지 (`demo_nodes_py`)

```
$ ros2 run demo_nodes_py talker  
$ ros2 run demo_nodes_py listener
```

```
ybbaek@ubuntu: ~/ros2_ws  
ybbaek@ubuntu:~/ros2_ws$ ros2 run demo_nodes_py talker  
[INFO] [1687178589.286086124] [talker]: Publishing: "Hello World: 0"  
[INFO] [1687178590.264473533] [talker]: Publishing: "Hello World: 1"  
[INFO] [1687178591.264370675] [talker]: Publishing: "Hello World: 2"  
[INFO] [1687178592.265381075] [talker]: Publishing: "Hello World: 3"  
[INFO] [1687178593.264470278] [talker]: Publishing: "Hello World: 4"  
[INFO] [1687178594.265665136] [talker]: Publishing: "Hello World: 5"  
[INFO] [1687178595.264292165] [talker]: Publishing: "Hello World: 6"  
  
ybbaek@ubuntu: ~/ros2_ws$ ros2 run demo_nodes_py listener  
ybbaek@ubuntu:~/ros2_ws$  
ybbaek@ubuntu:~/ros2_ws$ ros2 run demo_nodes_py listener  
[INFO] [1687178590.280535667] [listener]: I heard: [Hello World: 1]  
[INFO] [1687178591.265569304] [listener]: I heard: [Hello World: 2]  
[INFO] [1687178592.267171576] [listener]: I heard: [Hello World: 3]  
[INFO] [1687178593.267110221] [listener]: I heard: [Hello World: 4]  
[INFO] [1687178594.268632899] [listener]: I heard: [Hello World: 5]  
[INFO] [1687178595.265545708] [listener]: I heard: [Hello World: 6]
```

ROS2 동작확인

토픽발행

```
$ ros2 topic pub /chatter std_msgs/msg/String "data: Hello world"  
$ ros2 topic echo /chatter
```

The screenshot shows a terminal window with two command-line sessions. The top session is publishing messages to the '/chatter' topic, and the bottom session is echoing those messages back.

```
ybbaek@ubuntu: ~/ros2_ws  
ybbaek@ubuntu: ~/ros2_ws 87x15  
ybbaek@ubuntu:~/ros2_ws$ ros2 topic pub /chatter std_msgs/msg/String "data: Hello world"  
"  
publisher: beginning loop  
publishing #1: std_msgs.msg.String(data='Hello world')  
  
publishing #2: std_msgs.msg.String(data='Hello world')  
  
publishing #3: std_msgs.msg.String(data='Hello world')  
  
publishing #4: std_msgs.msg.String(data='Hello world')  
  
publishing #5: std_msgs.msg.String(data='Hello world')  
  
[  
  
ybbaek@ubuntu:~/ros2_ws$ ros2 topic echo /chatter  
data: Hello world  
---  
data: Hello world  
---
```



Turtlesim

실습

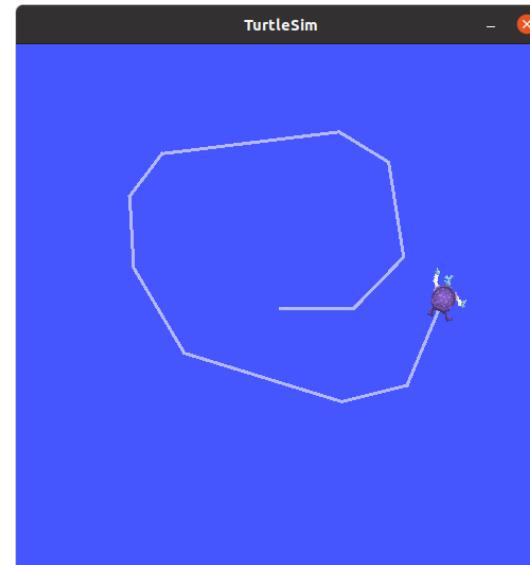
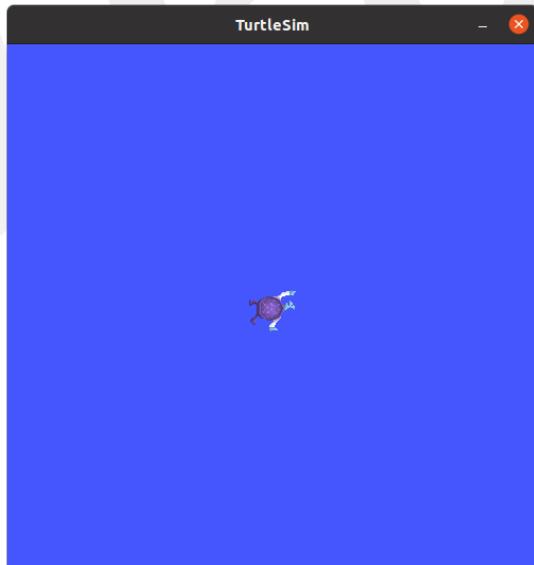


ROS2 구성

turtlesim

cmd: \$ ros2 pkg executables turtlesim

```
byb76@rlhp: ~/add_ros2_ws
byb76@rlhp: ~/add_ros2_ws$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
(ROS 2 foxy) byb76@rlhp:~/add_ros2_ws$ █
```



turtlesim

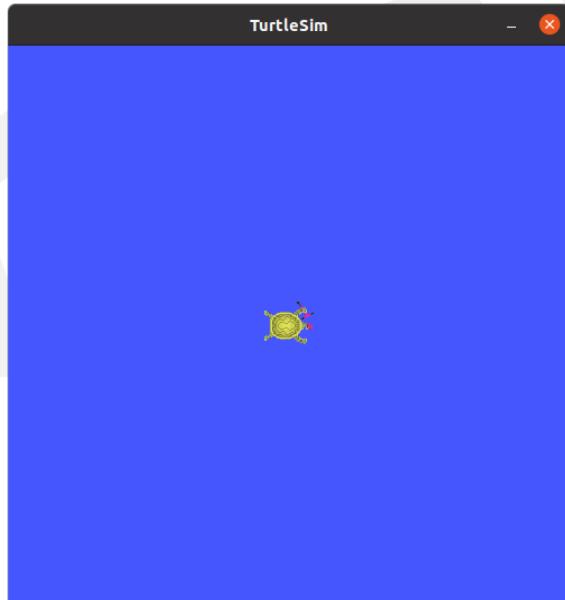
turtlesimulator 실행

2개의 노드실행

>turtlesim_node and turtle_teleop_key.

1. cmd: ros2 run turtlesim turtlesim_node
2. cmd: ros2 run turtlesim turtle_teleop_key

별도의 2개의 터미널에서 실행



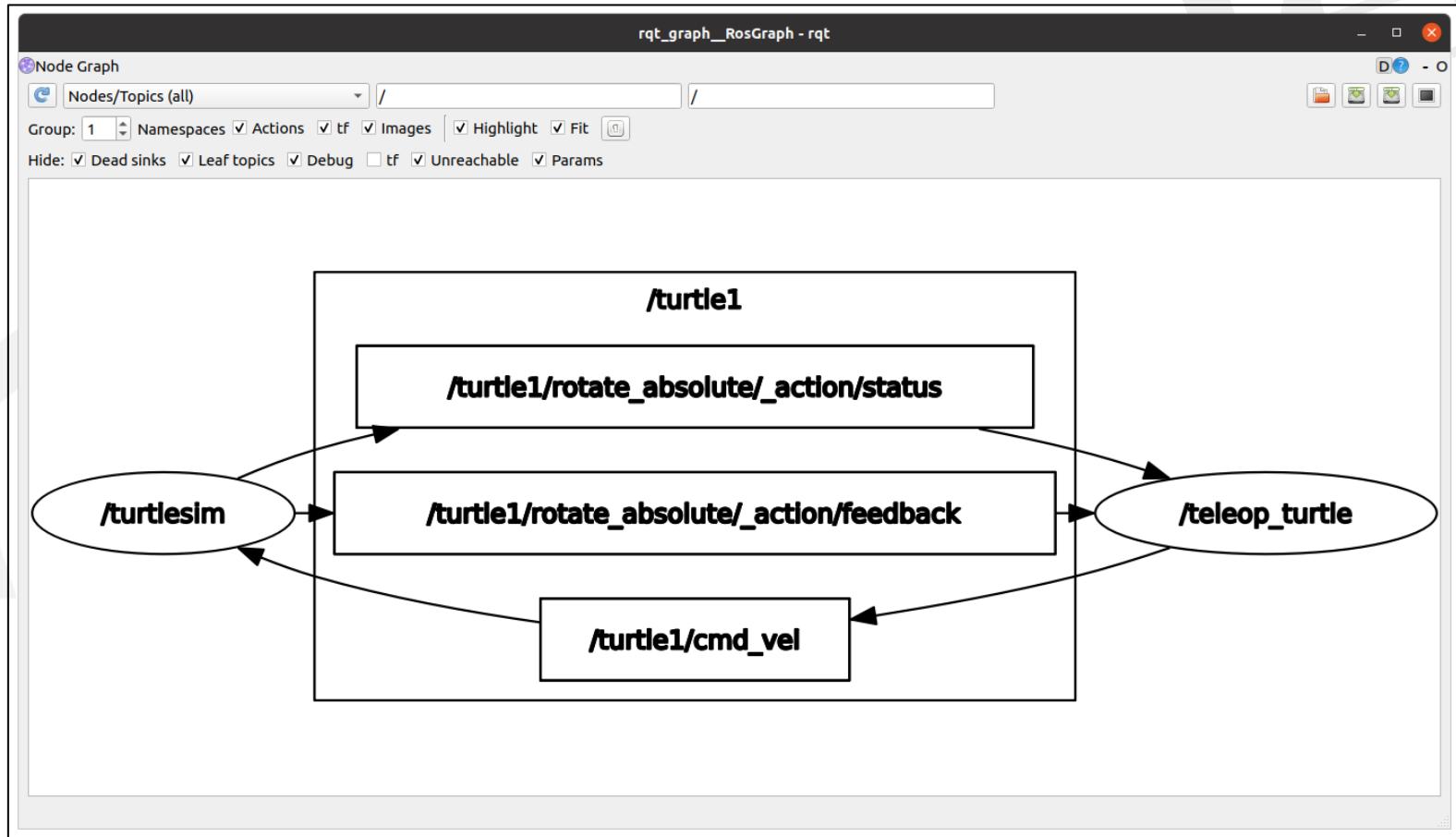
```
byb76@rlhp:~/add_ws
byb76@rlhp:~/add_ws 74x11
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 run turtlesim turtlesim_node
1674289083.879969 [0] turtlesim : using network interface wlp2s0 (udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
[INFO] [1674289083.931410507] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1674289083.942301217] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
byb76@rlhp:~/add_ws$ ros2 run turtlesim turtle
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 run turtlesim turtle_teleop_key
1674289745.651858 [0] turtle tel: using network interface wlp2s0 (udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
Reading from keyboard
-----
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
'Q' to quit.
```

turtlesim

구조확인

rqt_graph: node, topic (방향)

명령어: ros2 run rqt_graph rqt_graph





turtlesim

topic 확인

\$ros2 topic list

```
byb76@rlhp: ~/add_ws
byb76@rlhp: ~/add_ws 65x12
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 topic list
1674290505.468772 [0]          ros2: using network interface wlp2s0
(udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
/turtle1/cmd_vel
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
(ROS 2 foxy) byb76@rlhp:~/add_ws$ 
```



turtlesim

topic 확인

```
$ ros2 topic list -t
```

The screenshot shows a terminal window with three tabs at the top, all belonging to the user 'ybbaek' on an 'ubuntu' system. The active tab is the third one, which contains the command and its output.

```
ybbaek@ubuntu: ~      ybbaek@ubuntu: ~      ybbaek@ubuntu: ~/ro...
ybbaek@ubuntu:~/ros2_ws$ ros2 topic list -t
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
ybbaek@ubuntu:~/ros2_ws$
```



turtlesim

topic 확인

```
$ ros2 interface show geometry_msgs/msg/Twist
```

The screenshot shows a terminal window with three tabs open, all belonging to the user `ybbaeck@ubuntu`. The current tab is highlighted and displays the command `ros2 interface show geometry_msgs/msg/Twist` and its documentation. The previous tab shows the same command. The third tab is partially visible.

```
ybbaeck@ubuntu:~/ros2_ws$ ros2 interface show geometry_msgs/msg/Twist
# This expresses velocity in free space broken into its linear and angular parts.

Vector3 linear
Vector3 angular
ybbaeck@ubuntu:~/ros2_ws$
```



turtlesim

topic 확인

```
$ ros2 interface show geometry_msgs/msg/Twist
```

The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "rlmodel@rlmodel: ~/ros2_ws/src/ROS2_edu". There are two tabs open: one for "rlmodel@rlmodel: ~/ros2_ws" and another for "rlmodel@rlmodel: ~/ros2_ws/src/ROS2_edu", which is currently active. The terminal displays the following command and its output:

```
rlmodel@rlmodel:~/ros2_ws/src/ROS2_edu$ ros2 interface show geometry_msgs/msg/Vector3
# This represents a vector in free space.

# This is semantically different than a point.
# A vector is always anchored at the origin.
# When a transform is applied to a vector, only the rotational component is applied.

float64 x
float64 y
float64 z
rlmodel@rlmodel:~/ros2_ws/src/ROS2_edu$
```

turtlesim

topic publish

토픽 발행: CLI 방법 / 터미널 창에 명령어 입력

<예>

```
ros2 topic pub /turtle1/cmd_vel geometry_msgs/msg/Twist "linear:
```

```
    x: 0.5
```

```
    y: 0.0
```

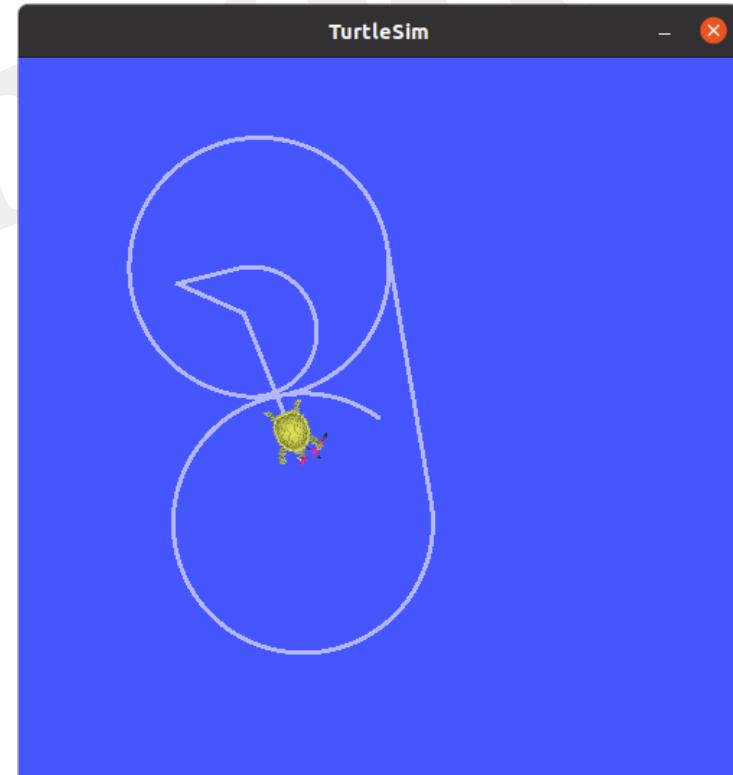
```
    z: 0.0
```

```
angular:
```

```
    x: 0.0
```

```
    y: 0.0
```

```
    z: 0.5"
```



turtlesim

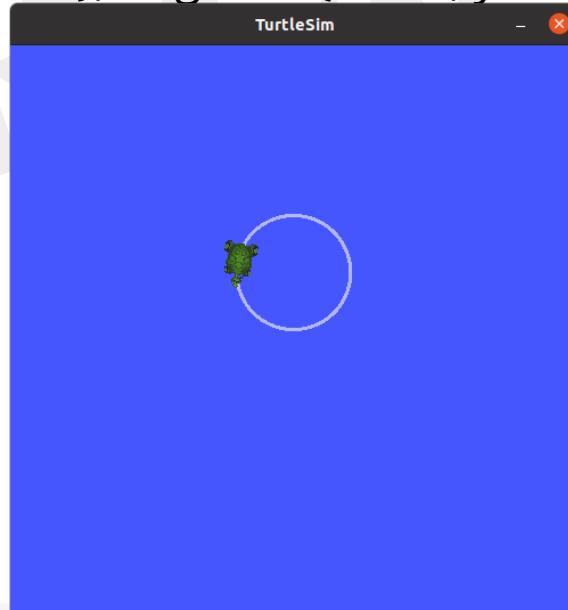
topic publish

토픽 발행: once, rate

<예>

```
$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist  
"{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

```
$ ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist  
"{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```



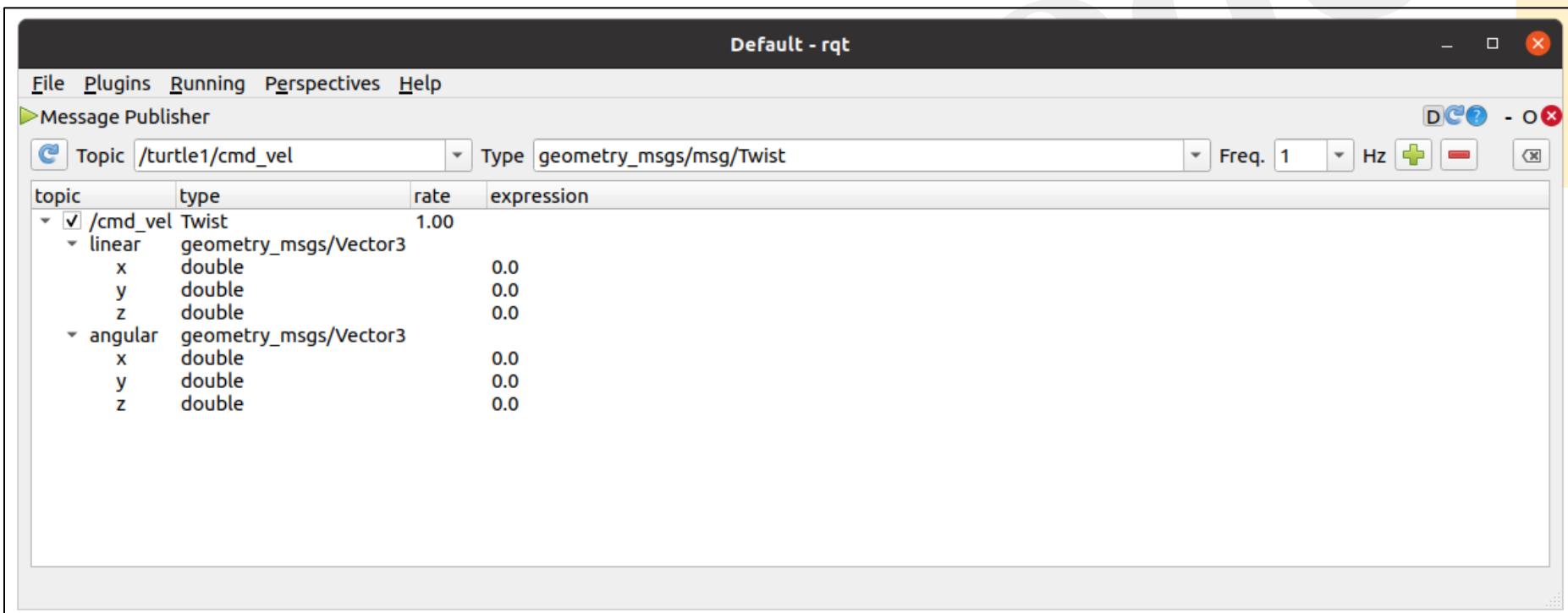
turtlesim

topic 발행

rqt method

\$rqt

>Plugins -> Topics -> Message Publisher ->

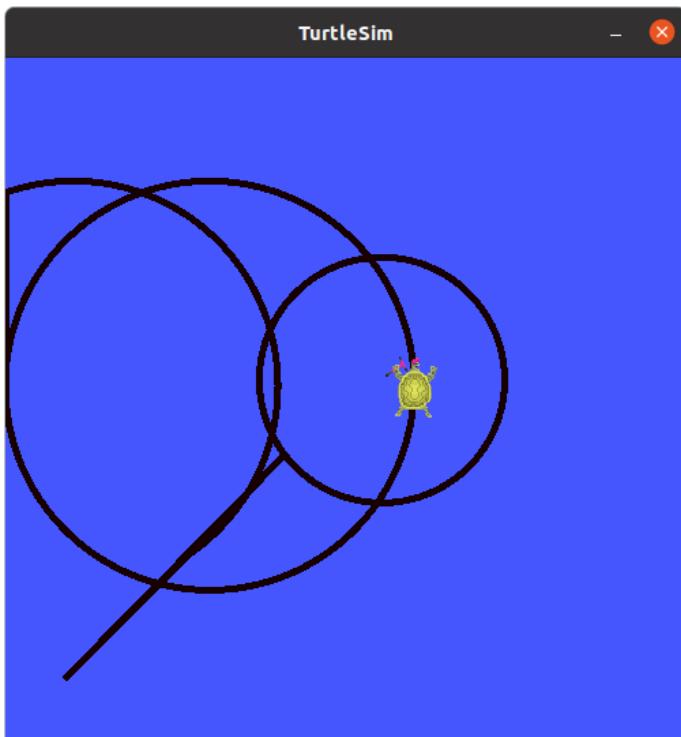


turtlesim

topic 확인

Terminal 창을 통해 CLI로 확인 방법

\$ros2 topic echo /turtle1/pose



```
byb76@rlhp: ~/add_ws
byb76@rlhp: ~/add_ws 68x14

---
x: 6.499609470367432
y: 6.7892303466796875
theta: 1.886519432067871
linear_velocity: 1.0
angular_velocity: 0.30000001192092896
---
x: 6.494568824768066
y: 6.804415225982666
theta: 1.8913193941116333
linear_velocity: 1.0
angular_velocity: 0.30000001192092896
---
```

turtlesim

topic 확인

rqt로 확인

Default - rqt

File Plugins Running Perspectives Help

Topic Monitor

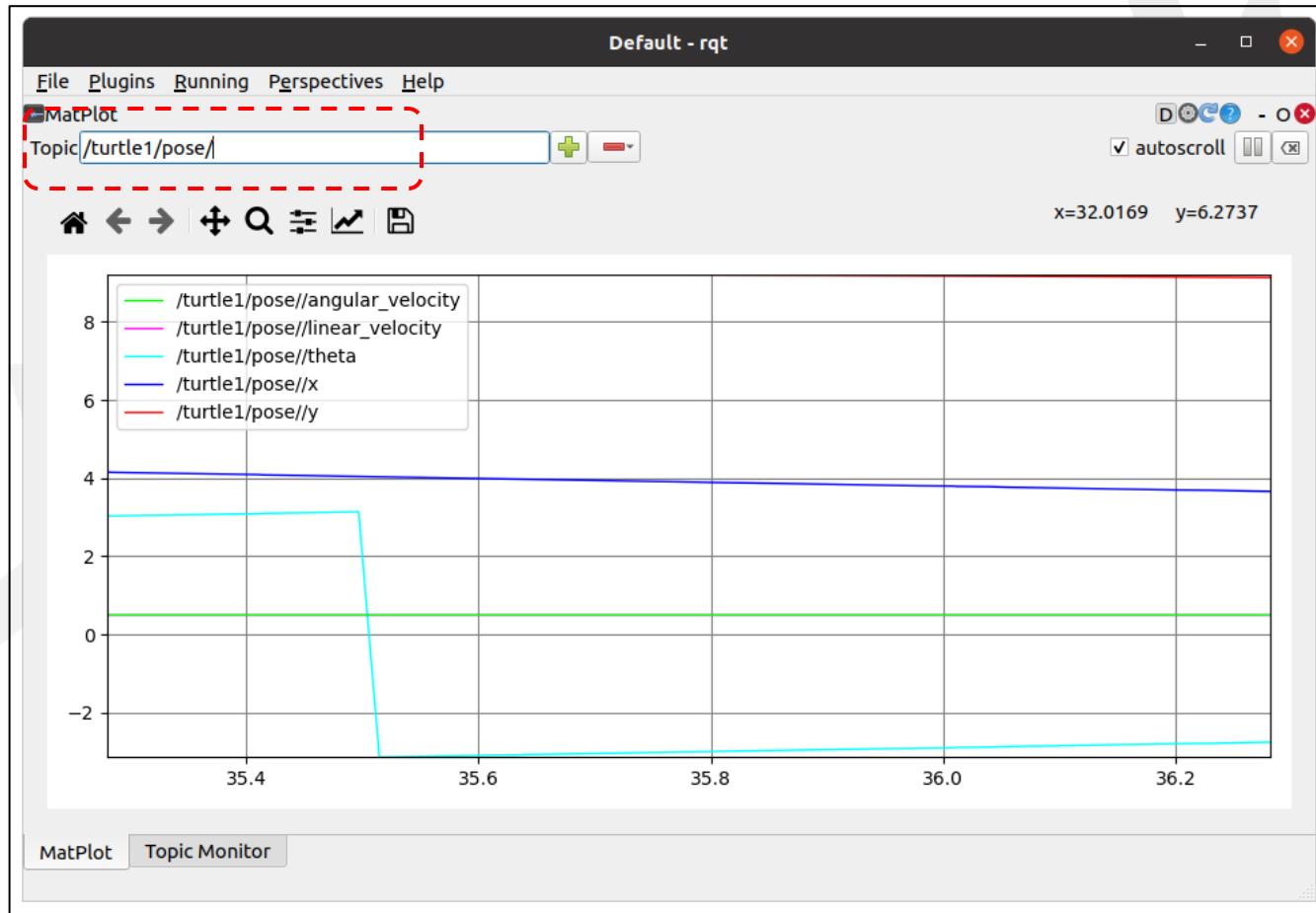
Topic	Type	Bandwidth	Hz	Value
✓ /cmd_vel	geometry_msgs/msg/Twist			not monitored
✓ /parameter_events	rcl_interfaces/msg/ParameterEvent			not monitored
✓ /rosout	rcl_interfaces/msg/Log			not monitored
✓ /turtle1/cmd_vel	geometry_msgs/msg/Twist			not monitored
✓ /turtle1/color_sensor	turtlesim/msg/Color			not monitored
✓ /turtle1/pose	turtlesim/msg/Pose	unknown	62.49	0.30000001192092896
angular_velocity	float			1.0
linear_velocity	float			-1.7559419870376587
theta	float			0.07985839992761612
x	float			6.352381229400635
y	float			can not get message class for type
✓ /turtle1/rotate_absolute/_action/feedback	turtlesim/action/RotateAbsolute_FeedbackMessage			not monitored
✓ /turtle1/rotate_absolute/_action/status	action_msgs/msg/GoalStatusArray			

turtlesim

topic 확인

rqt: Plugins -> Visualization -> Plot

>/turtle1/pose/

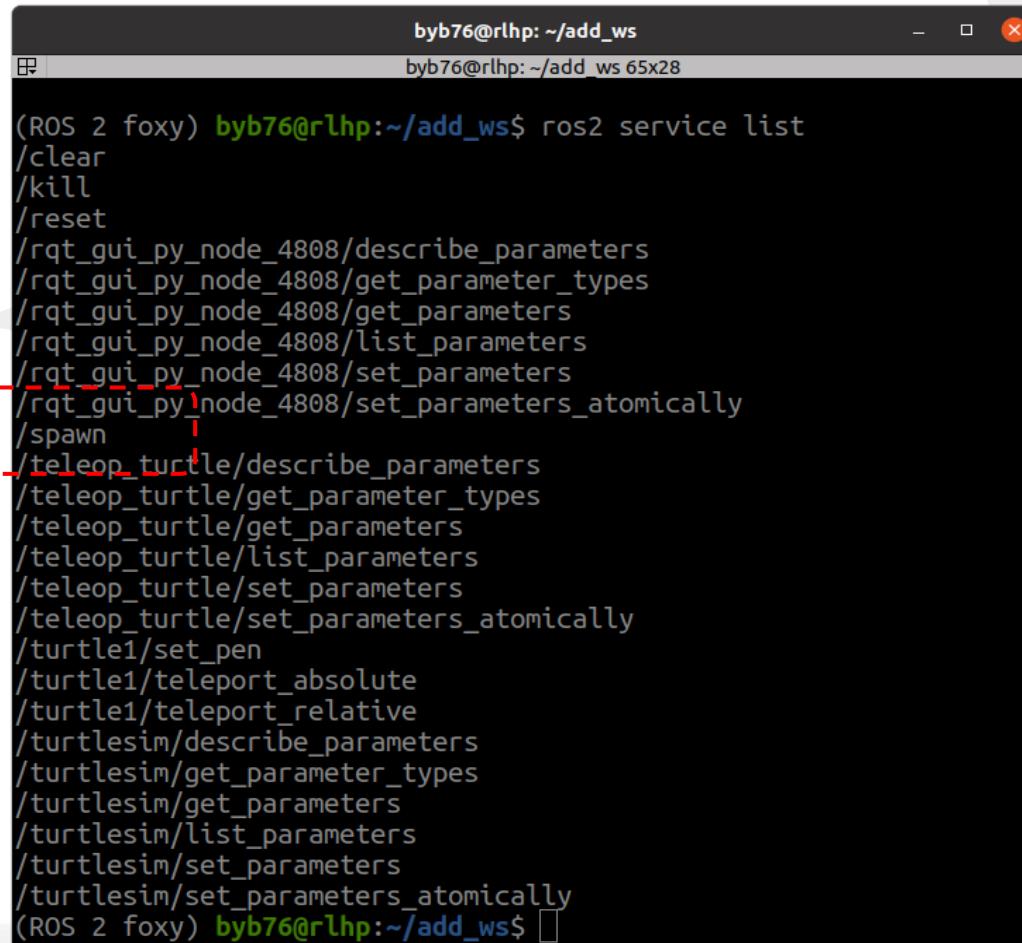


turtlesim

service 확인

>서비스 리스트 확인

>\$ros2 service list (-t option show msg type)



```
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 service list
/clear
/kill
/reset
/rqt_gui_py_node_4808/describe_parameters
/rqt_gui_py_node_4808/get_parameter_types
/rqt_gui_py_node_4808/get_parameters
/rqt_gui_py_node_4808/list_parameters
/rqt_gui_py_node_4808/set_parameters
/rqt_gui_py_node_4808/set_parameters_atomically
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
(ROS 2 foxy) byb76@rlhp:~/add_ws$
```

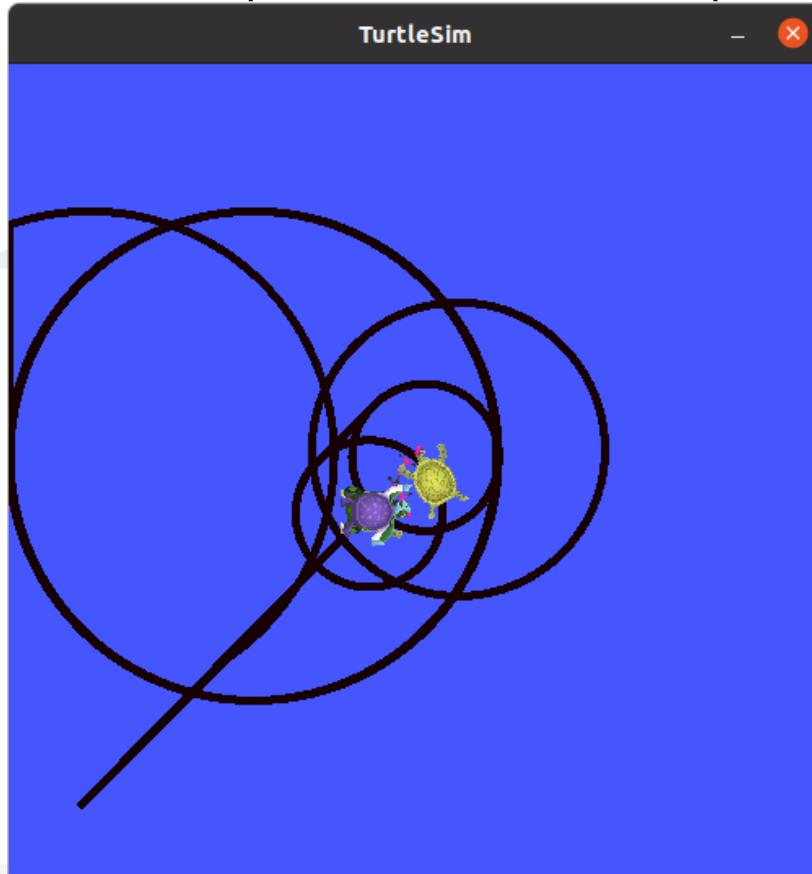
turtlesim

service 확인

spawn service 실행

```
$ ros2 service call /spawn turtlesim/srv/Spawn
```

```
$ ros2 service call -r 0.5 /spawn turtlesim/srv/Spawn "{x: 5,y: 5,theta: 0}"
```



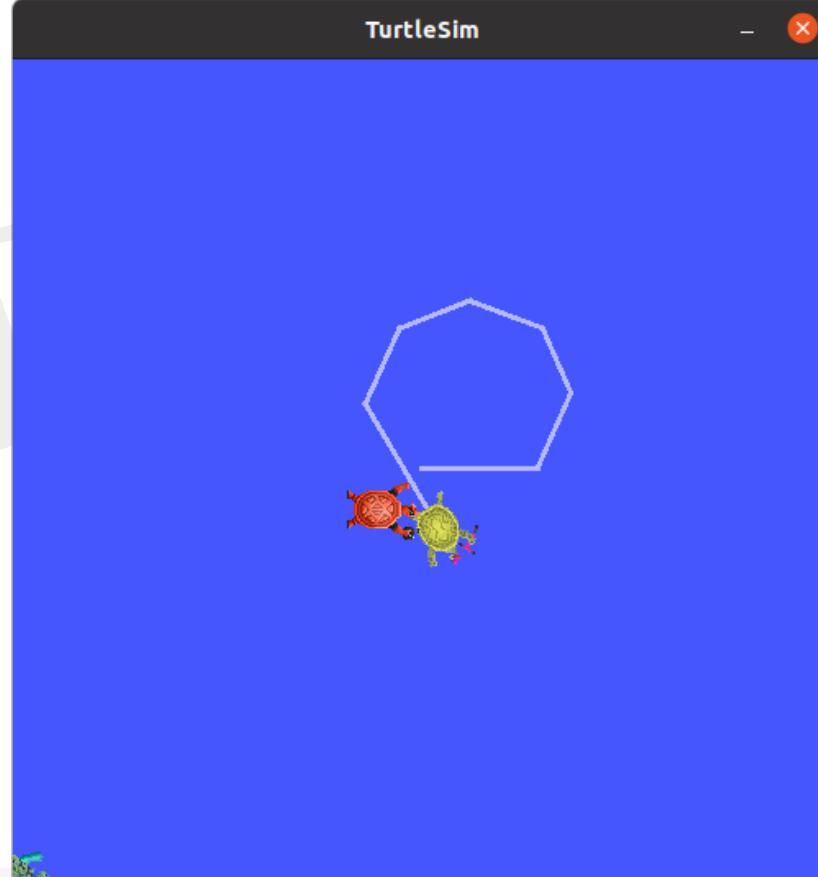
turtlesim

service 확인

spawn service 실행

```
$ros2 service call /spawn turtlesim/srv/Spawn
```

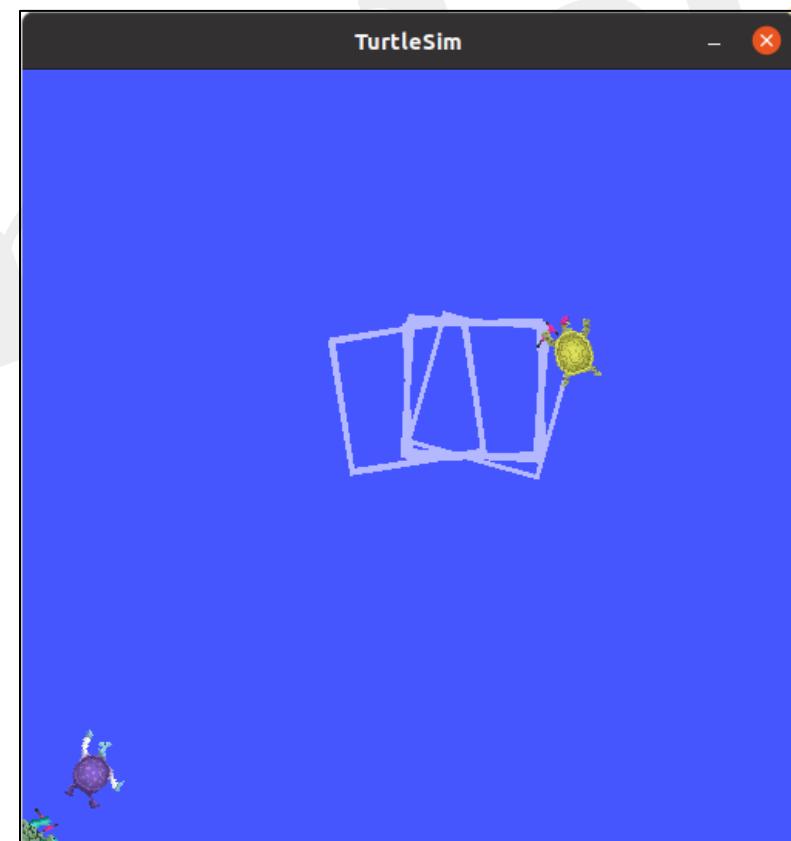
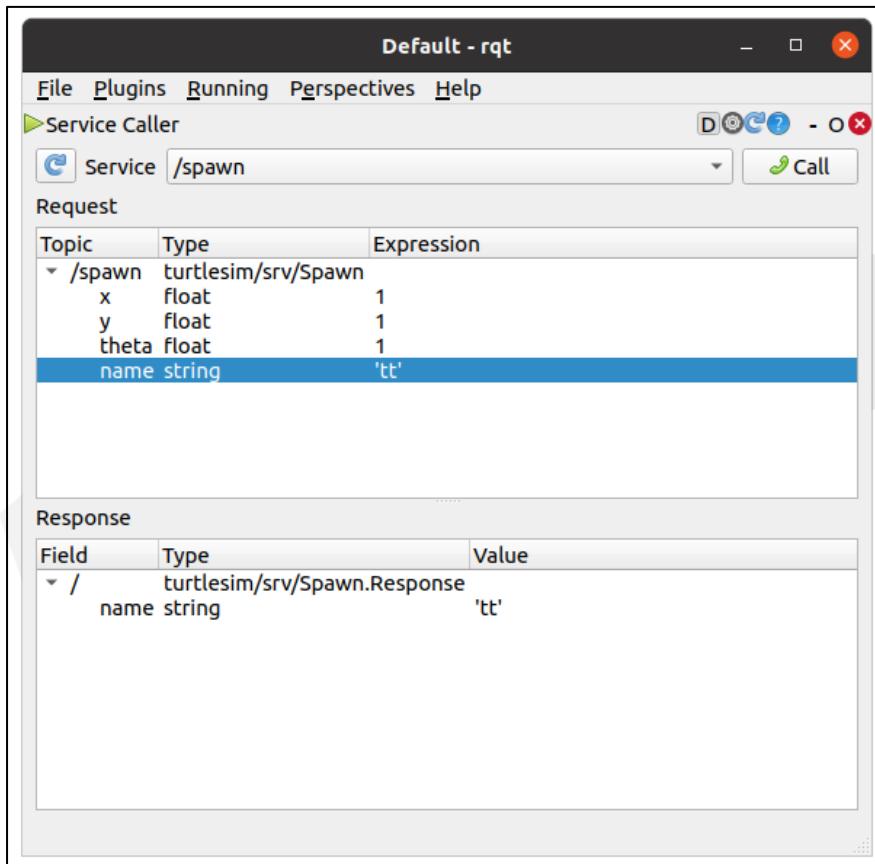
```
$ros2 service call -r 0.5 /spawn turtlesim/srv/Spawn "{x: 5,y: 5,theta: 0}"
```



turtlesim

service 확인

spawn service 실행 with rqt

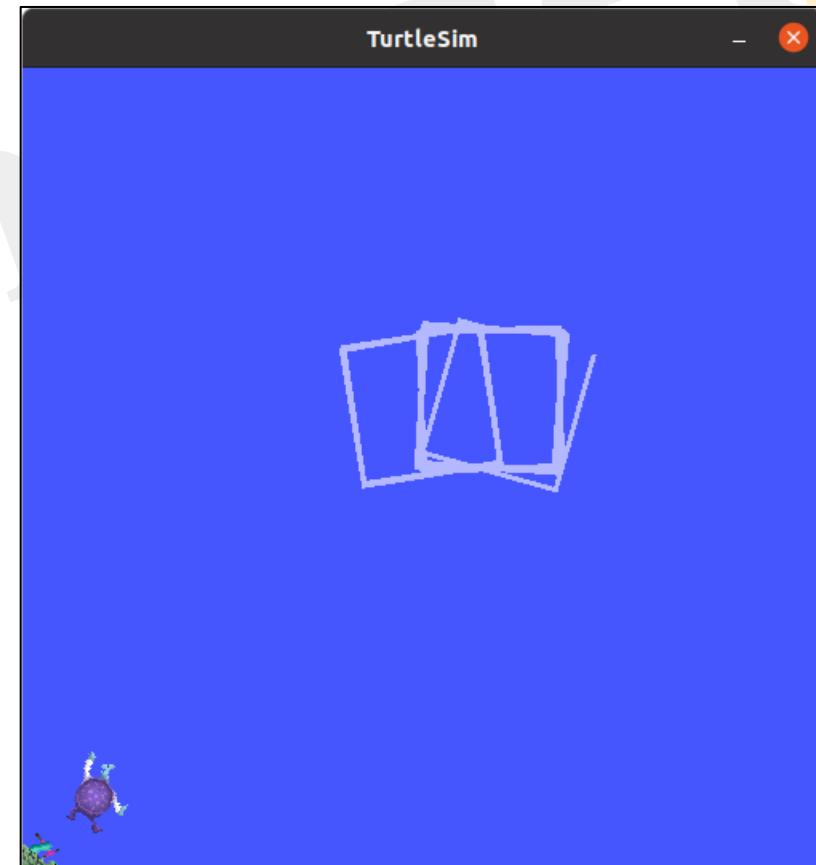
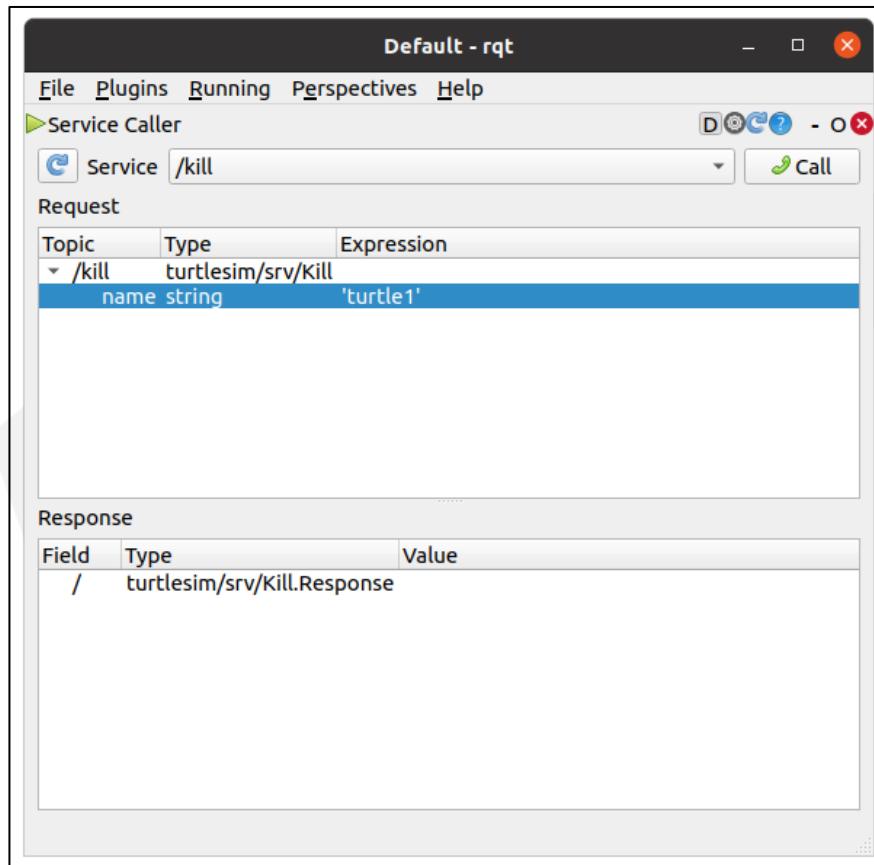


turtlesim

service 확인

kill service 실행 with rqt

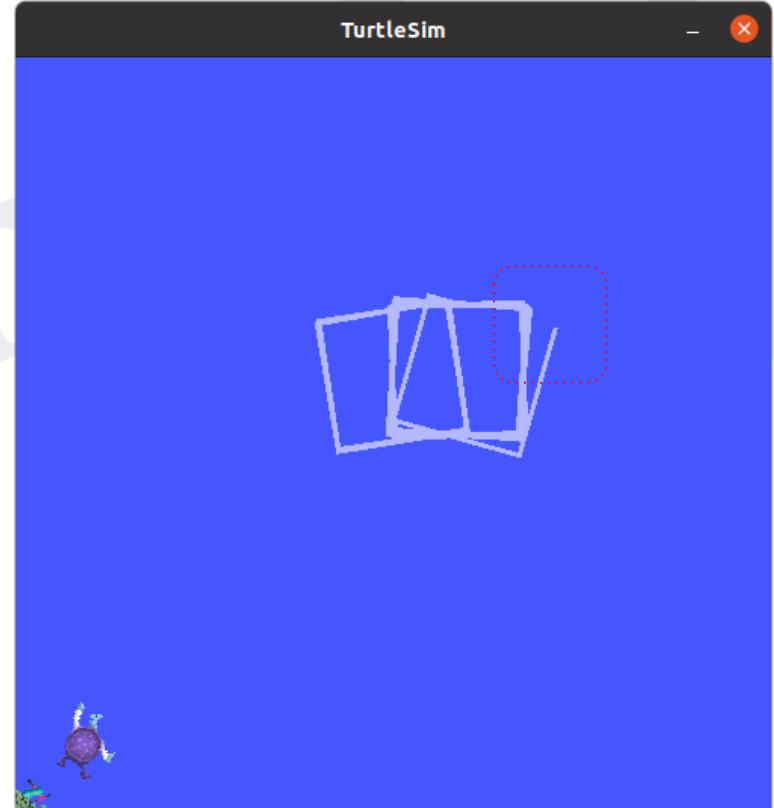
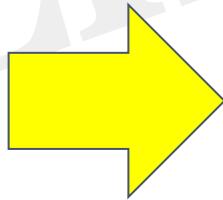
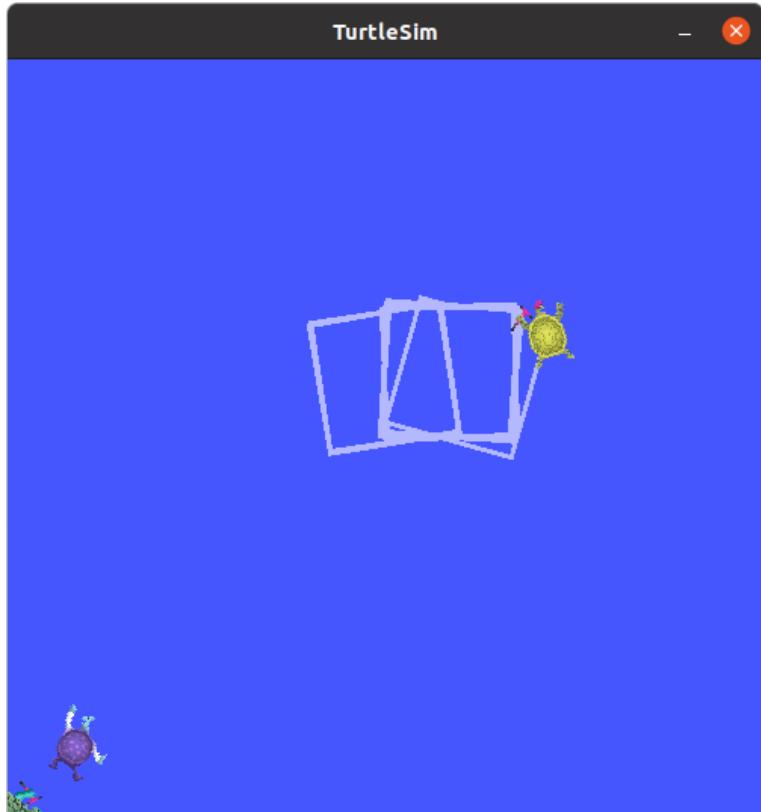
ros2 service call /kill turtlesim/srv/Kill "name: 'turtle1'"



turtlesim

service 확인

kill service 실행 with rqt

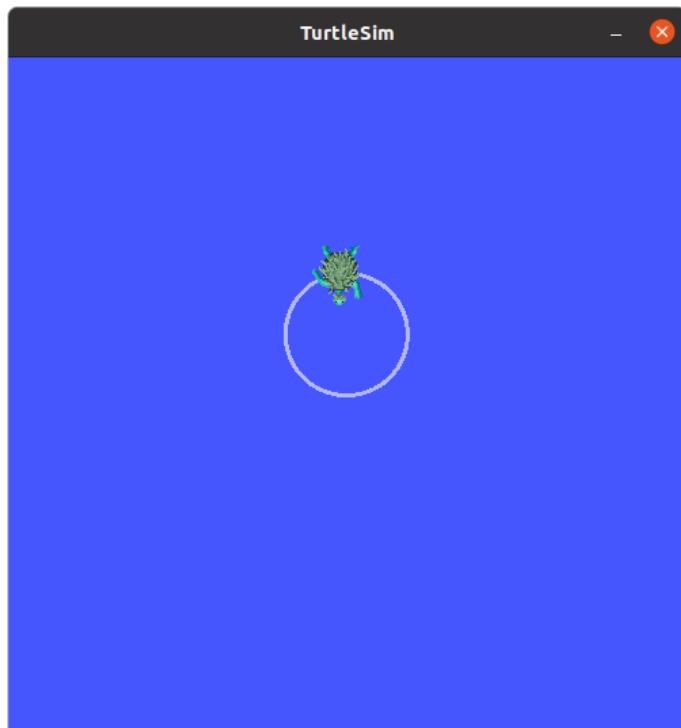


turtlesim

action 확인

send_goal 실행

```
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute {'theta: -1.57'} --feedback
```



```
byb76@rlhp: ~/add_ws
(ROS 2 foxy) byb76@rlhp:~/add_ws$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute {'theta: -1.57'} --feedback
1674303291.478640 [0]      ros2: using network interface wlp2s0 (udp
/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
Waiting for an action server to become available...
Sending goal:
    theta: -1.57

Goal accepted with ID: 19c88841ce0d4498bccb2640609d152d

Feedback:
    remaining: 0.012405037879943848

Result:
    delta: 0.0

Goal finished with status: SUCCEEDED
(ROS 2 foxy) byb76@rlhp:~/add_ws$
```



turtlesim

parameter 확인

\$ros2 param list

```
ybb@ubuntu: ~/ros2_ws$ ros2 param list
/turtlesim:
background_b
background_g
background_r
use_sim_time
ybb@ubuntu: ~/ros2_ws$
```



turtlesim

parameter 확인

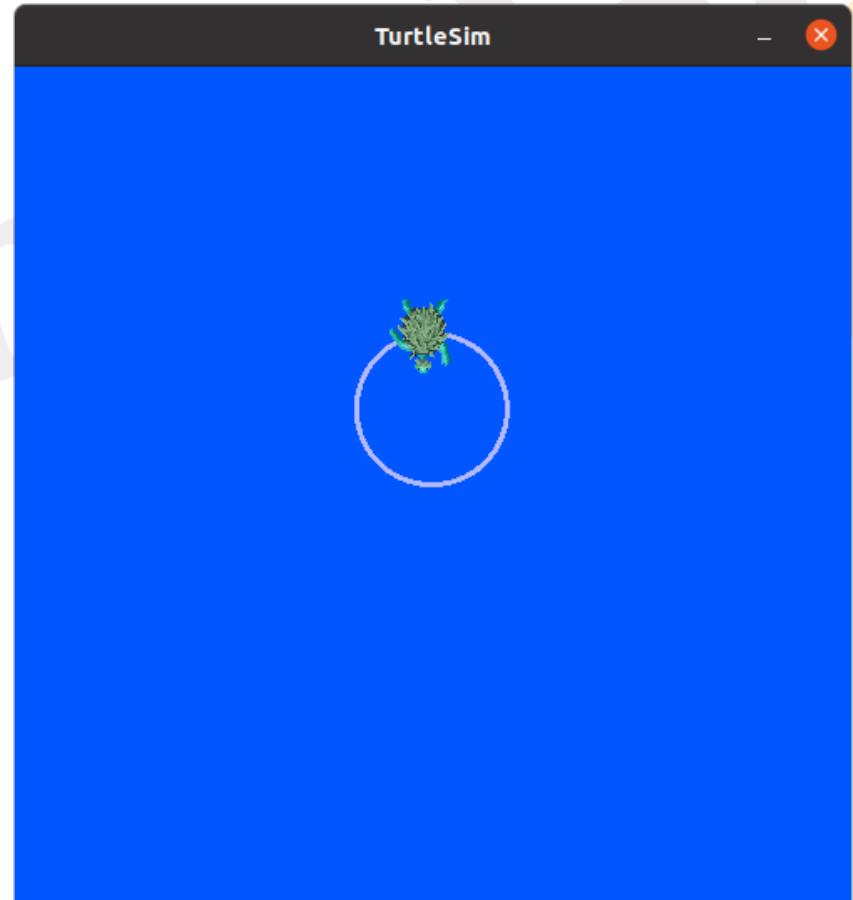
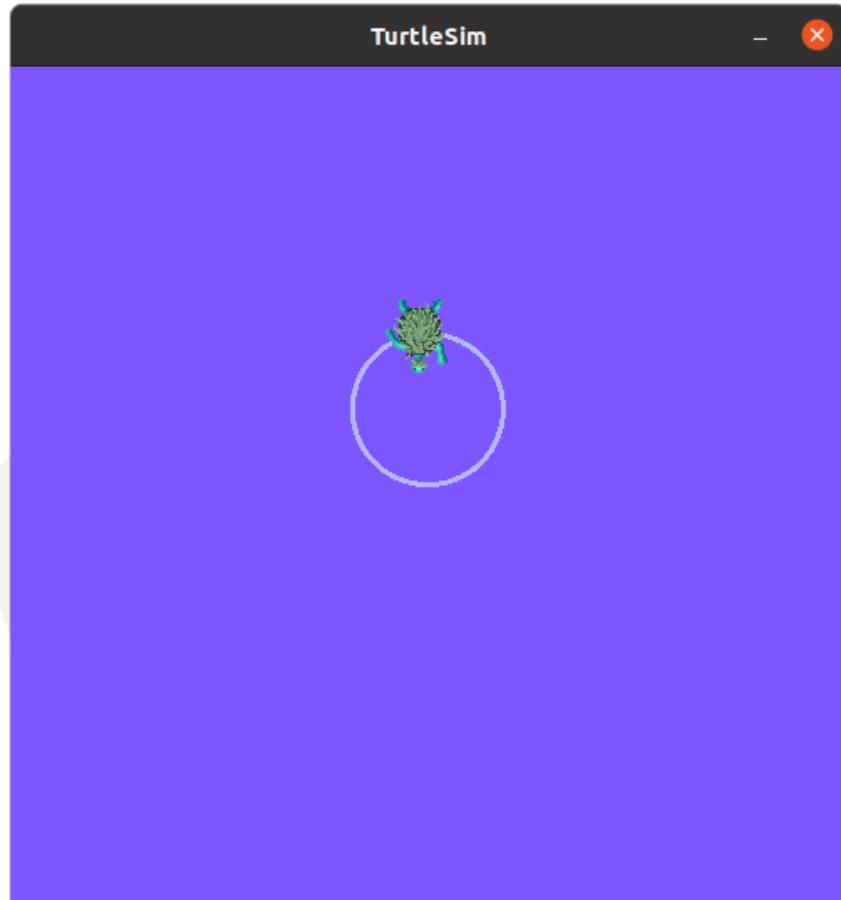
```
$ ros2 param describe /turtlesim background_r
```

```
ybbaeek@ubuntu: ~/ros2_ws$ ros2 param describe /turtlesim background_r
Parameter name: background_r
Type: integer
Description: Red channel of the background color
Constraints:
  Min value: 0
  Max value: 255
  Step: 1
ybbaeek@ubuntu: ~/ros2_ws$
```

turtlesim

parameter set - 쓰기

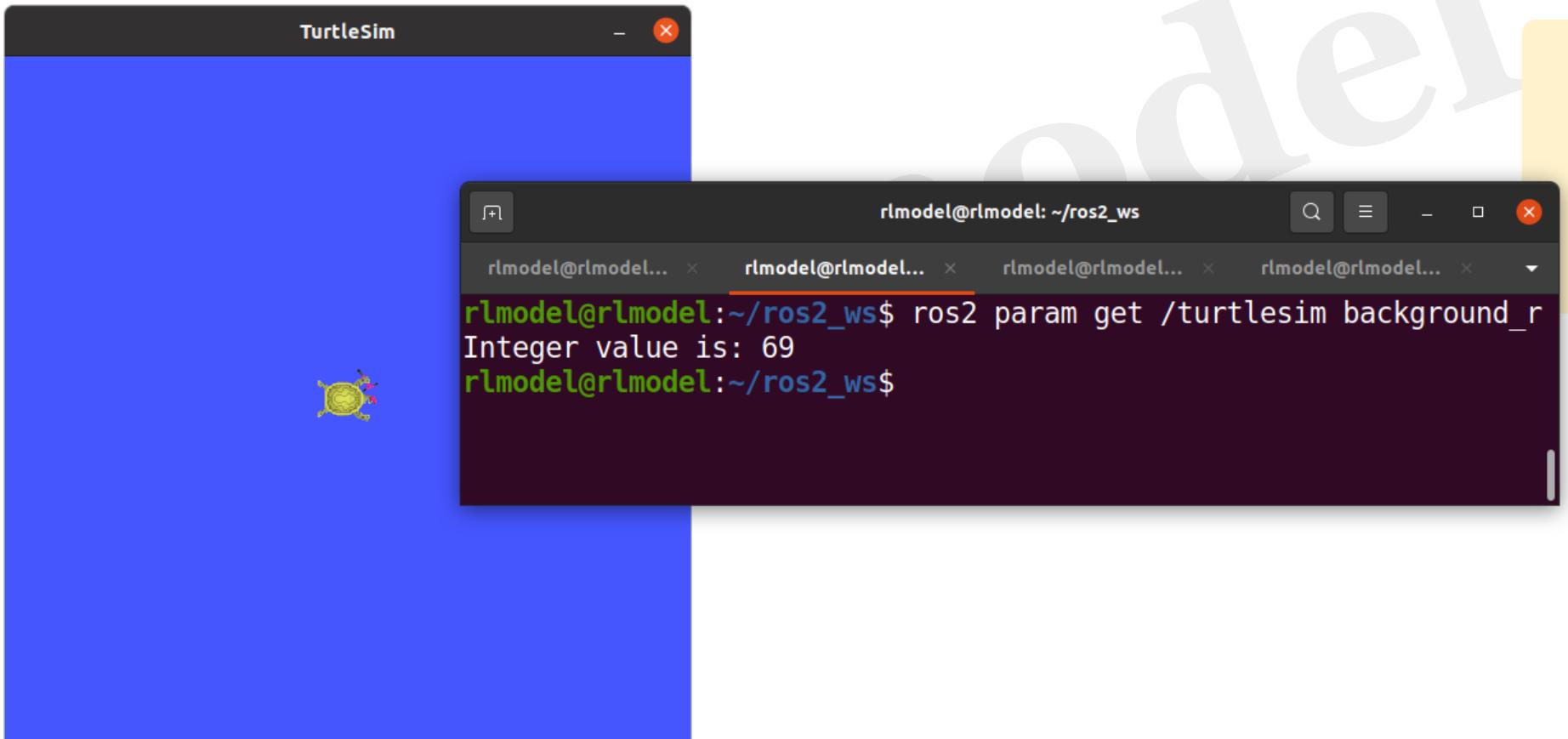
```
$ros2 param set /turtlesim background_r 1
```



turtlesim

parameter get - 읽기

```
$ ros2 param get /turtlesim background_r
```





turtlesim

parameter 저장

```
$ ros2 param dump /turtlesim
```

The screenshot shows a terminal window with a dark background and light-colored text. The title bar indicates the user is on an Ubuntu system with the path ~/ros2_ws. The terminal has three tabs open, all showing the same content. The current tab is highlighted with an orange underline. The text in the terminal is as follows:

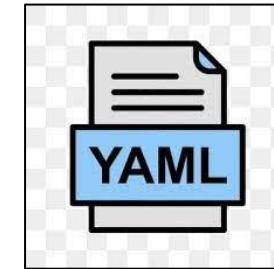
```
ybbaek@ubuntu:~/ros2_ws$ ros2 param dump /turtlesim
Saving to: ./turtlesim.yaml
ybbaek@ubuntu:~/ros2_ws$
```



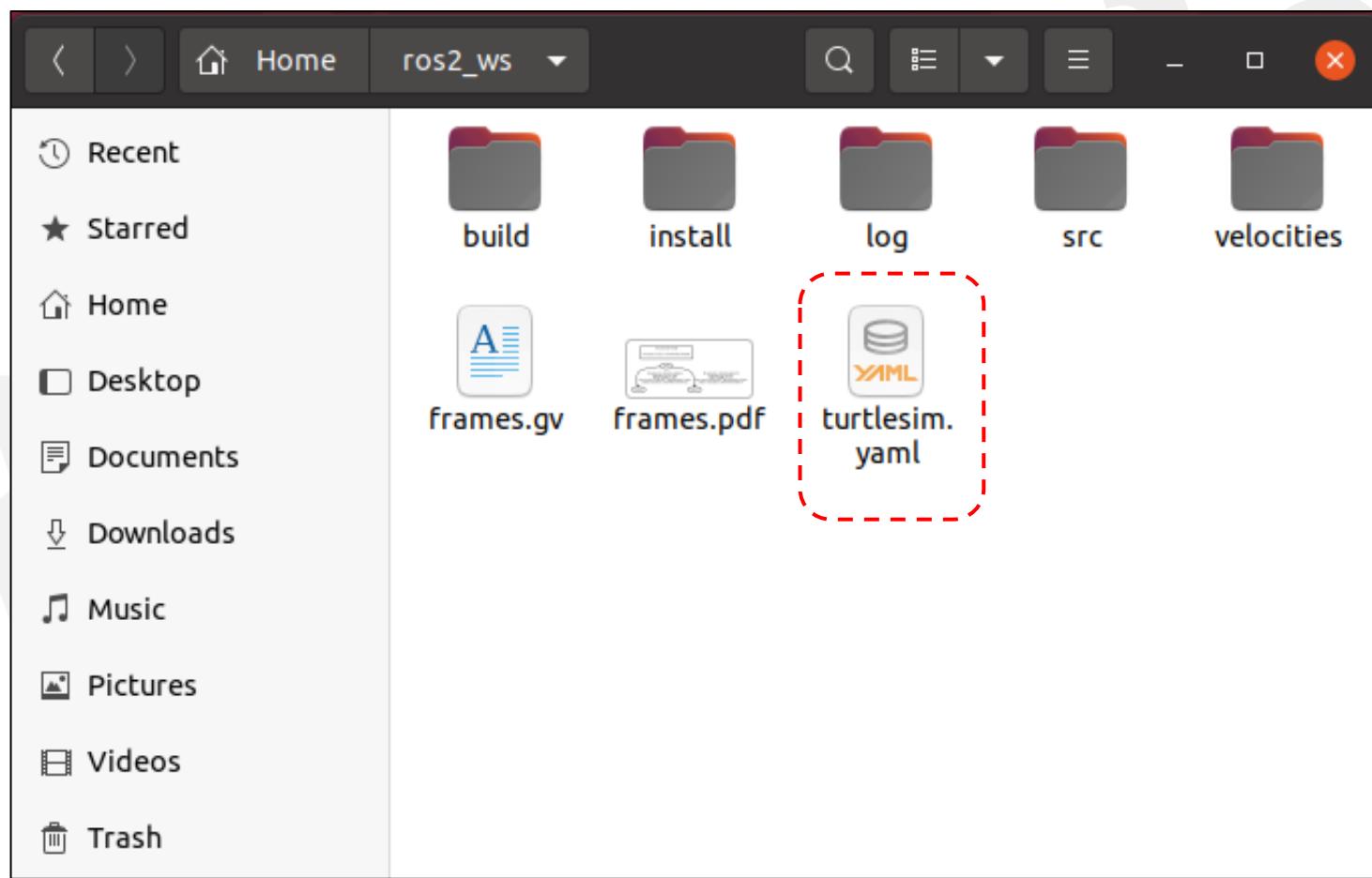
turtlesim

parameter 저장

\$ros2 param dump /turtlesim



Appendix 참고



turtlesim

parameter 저장

```
$ros2 param dump /turtlesim
```

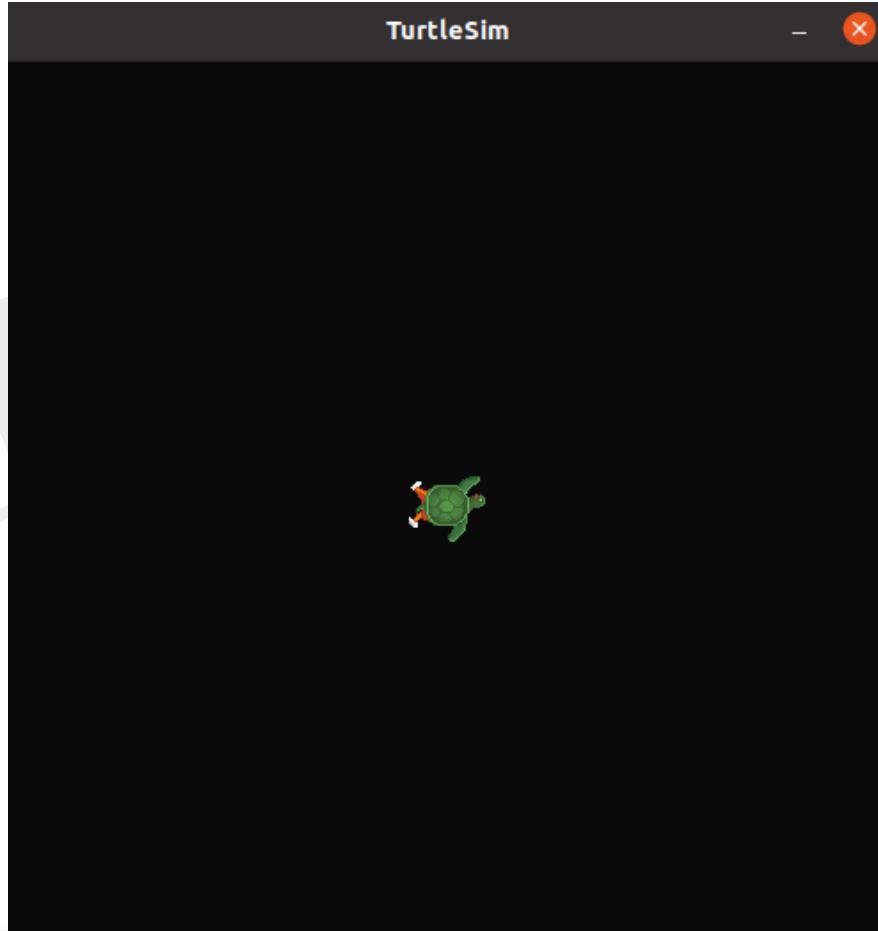
```
1 /turtlesim:  
2   ros__parameters:  
3     background_b: 10  
4     background_g: 10  
5     background_r: 10  
6     use_sim_time: false
```

YAML ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

turtlesim

parameter 로딩

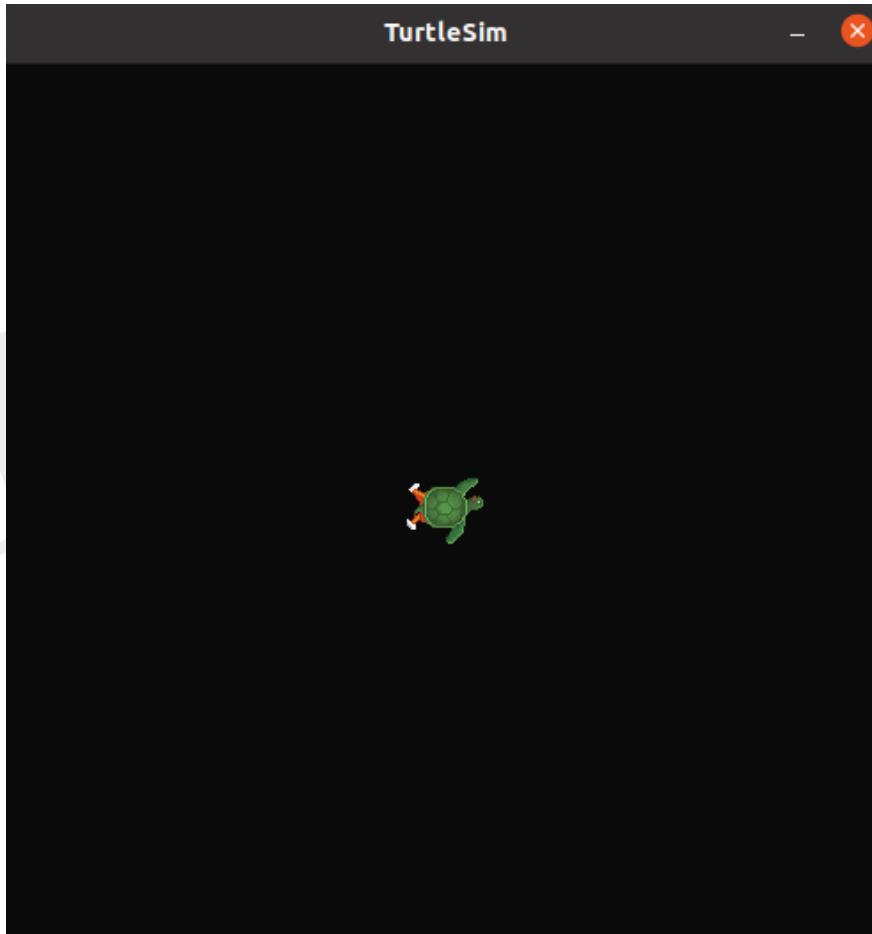
```
$ ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```



turtlesim

parameter 로딩

```
$ ros2 param load turtlesim ./turtlesim.yaml
```

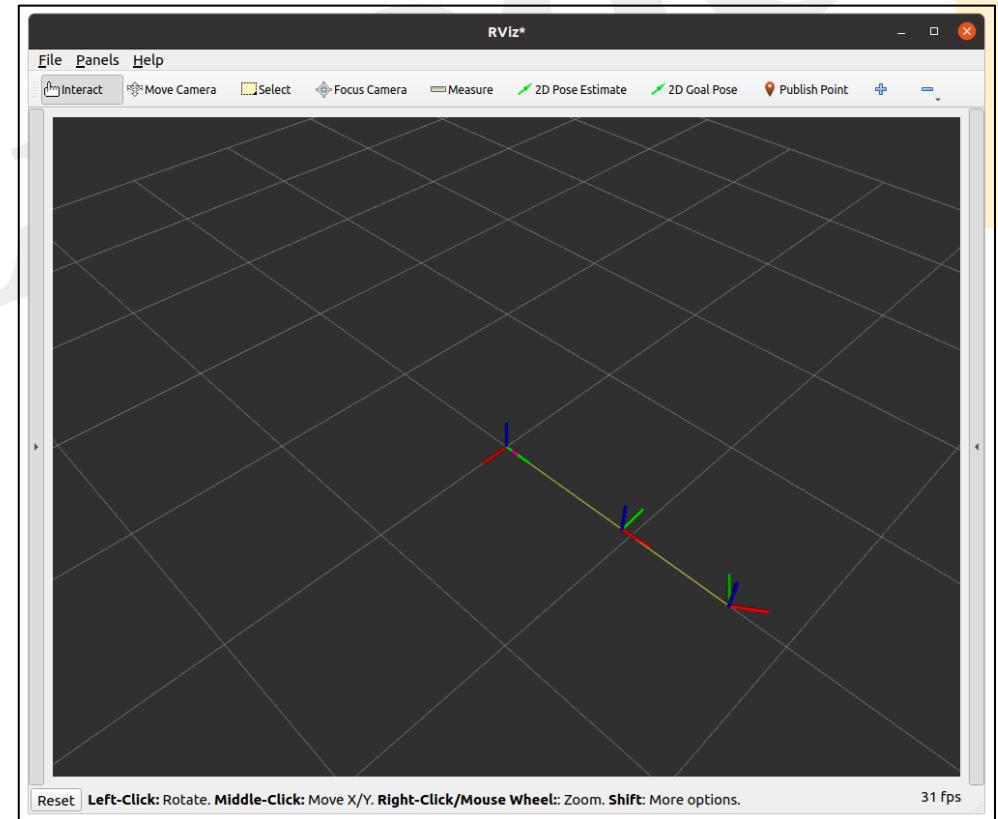
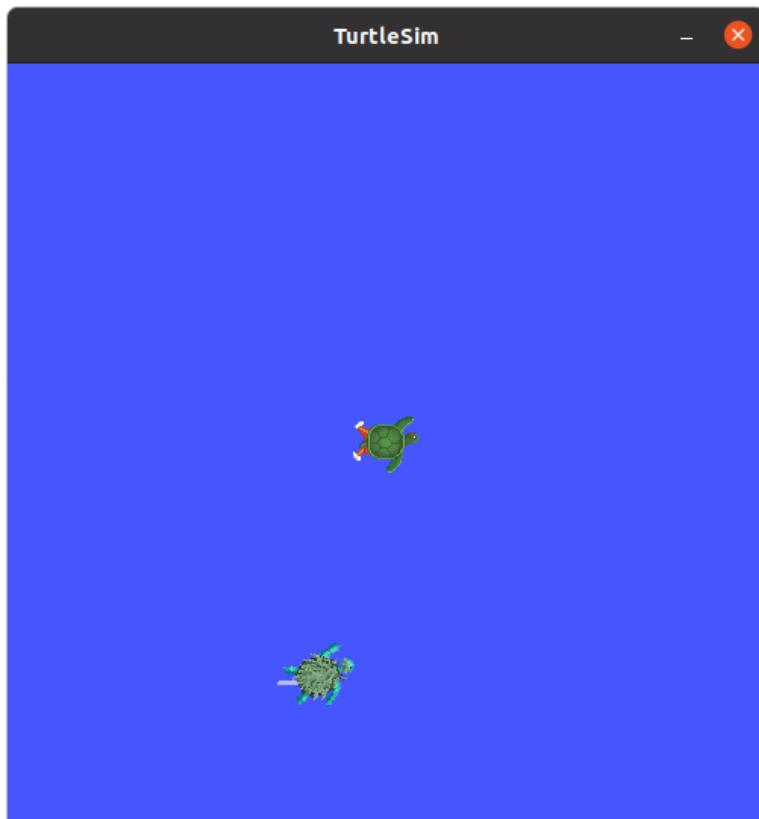


turtlesim

tf2 - turtlesim (rviz2)

설치: sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools
ros-foxy-tf-transformations

실행: ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py

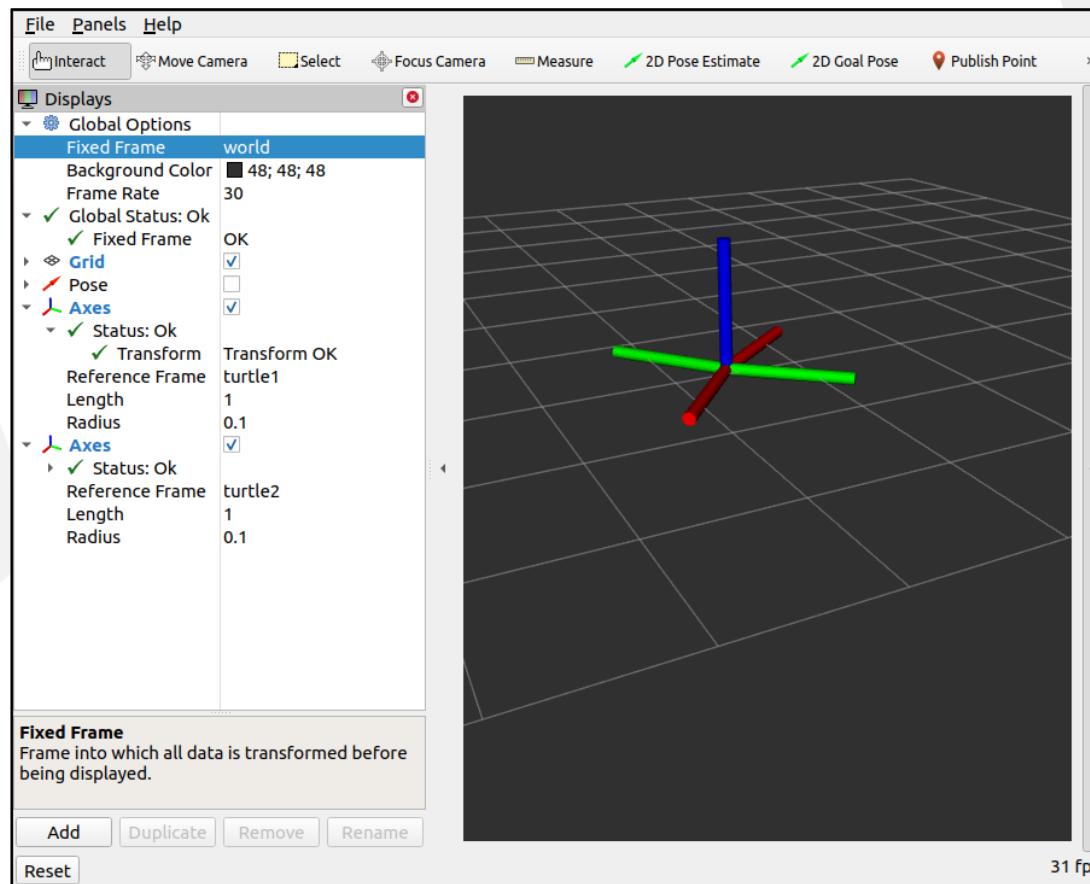


turtlesim

tf2 - turtlesim

실행1: turtlesim turtle_teleop_key

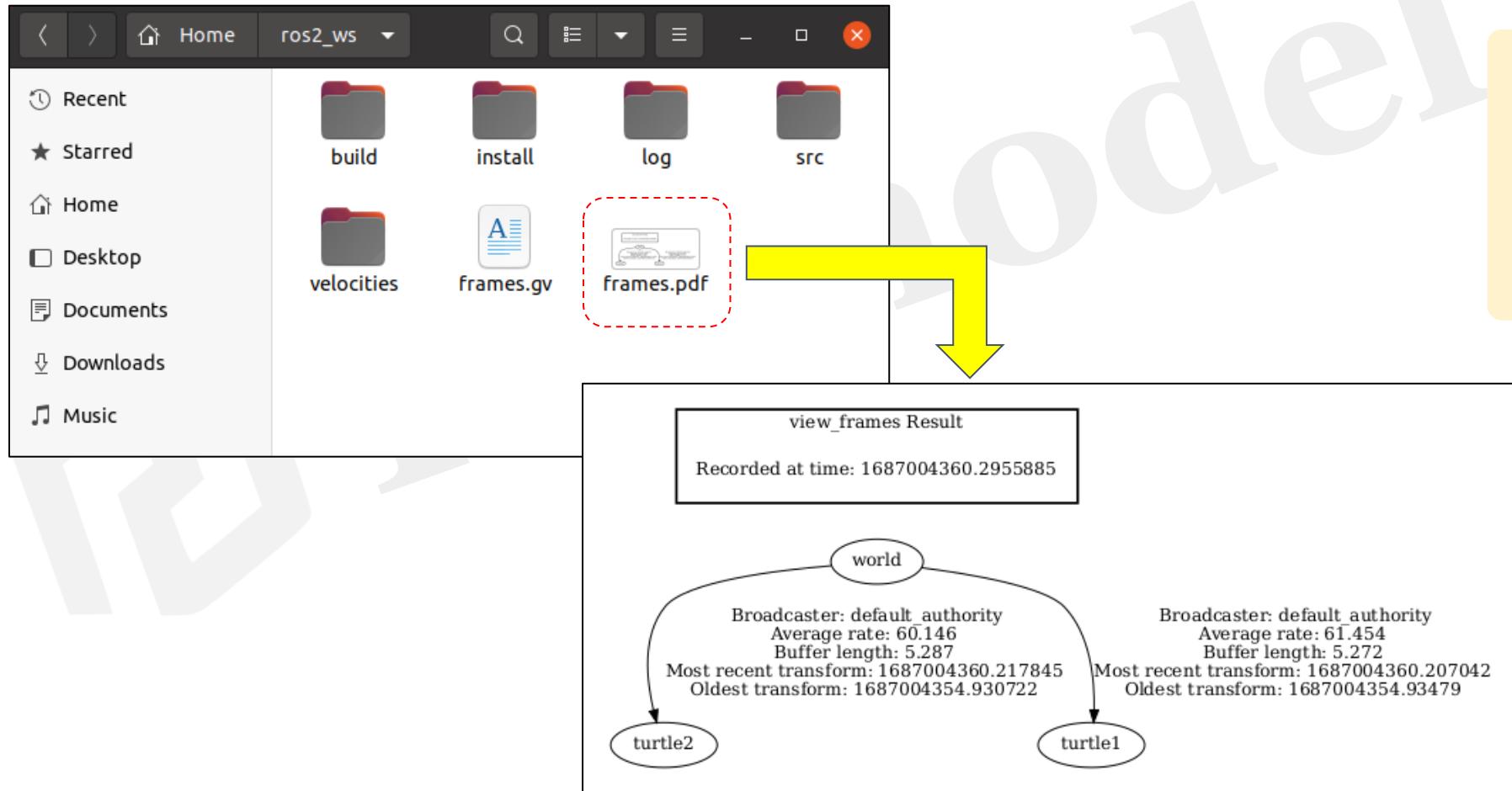
실행2: rviz Axes 추가



turtlesim

frame 확인

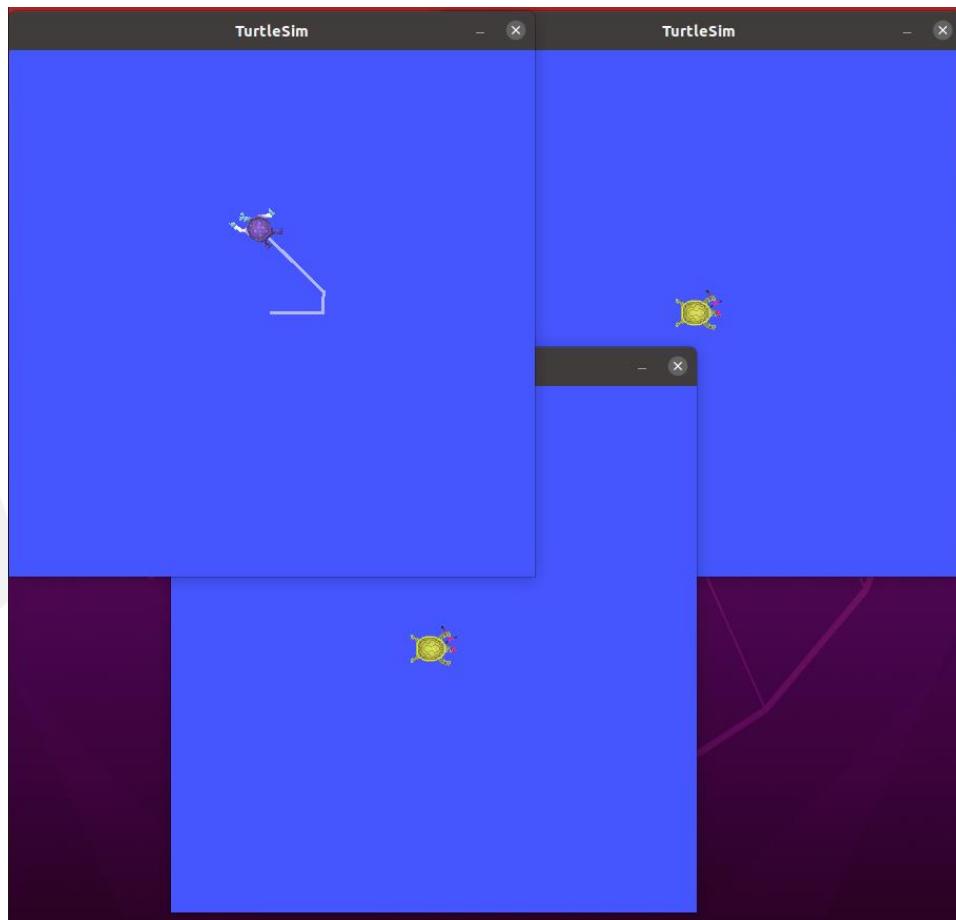
실행: ros2 run tf2_tools view_frames.py



turtlesim

launch file 실행

실행: \$ ros2 launch turtlesim multisim.launch.py



turtlesim

bag 명령어

bag 기능

시스템의 topics가 published한 데이터를 기록하기 위한 도구.

이 도구는 여러 topics에 걸쳐 전달된 데이터를 축적 및 데이터베이스에 저장합니다. 다음, 데이터를 재생하여 테스트 및 실험 결과를 재현할 수 있음



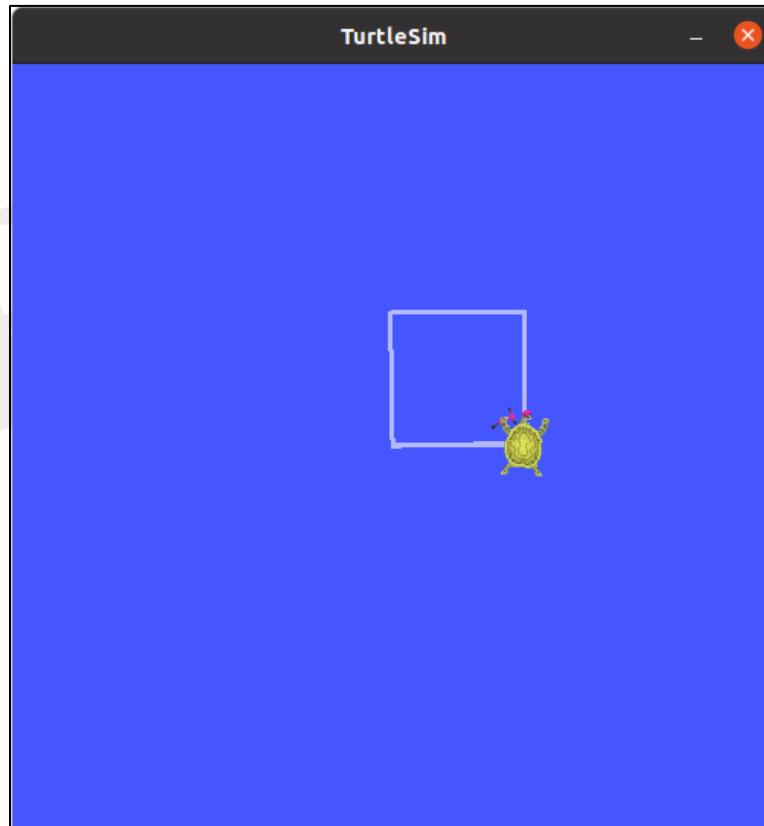
turtlesim

bag 명령어

example record/play

```
$ ros2 run turtlesim turtlesim_node
```

```
$ r ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

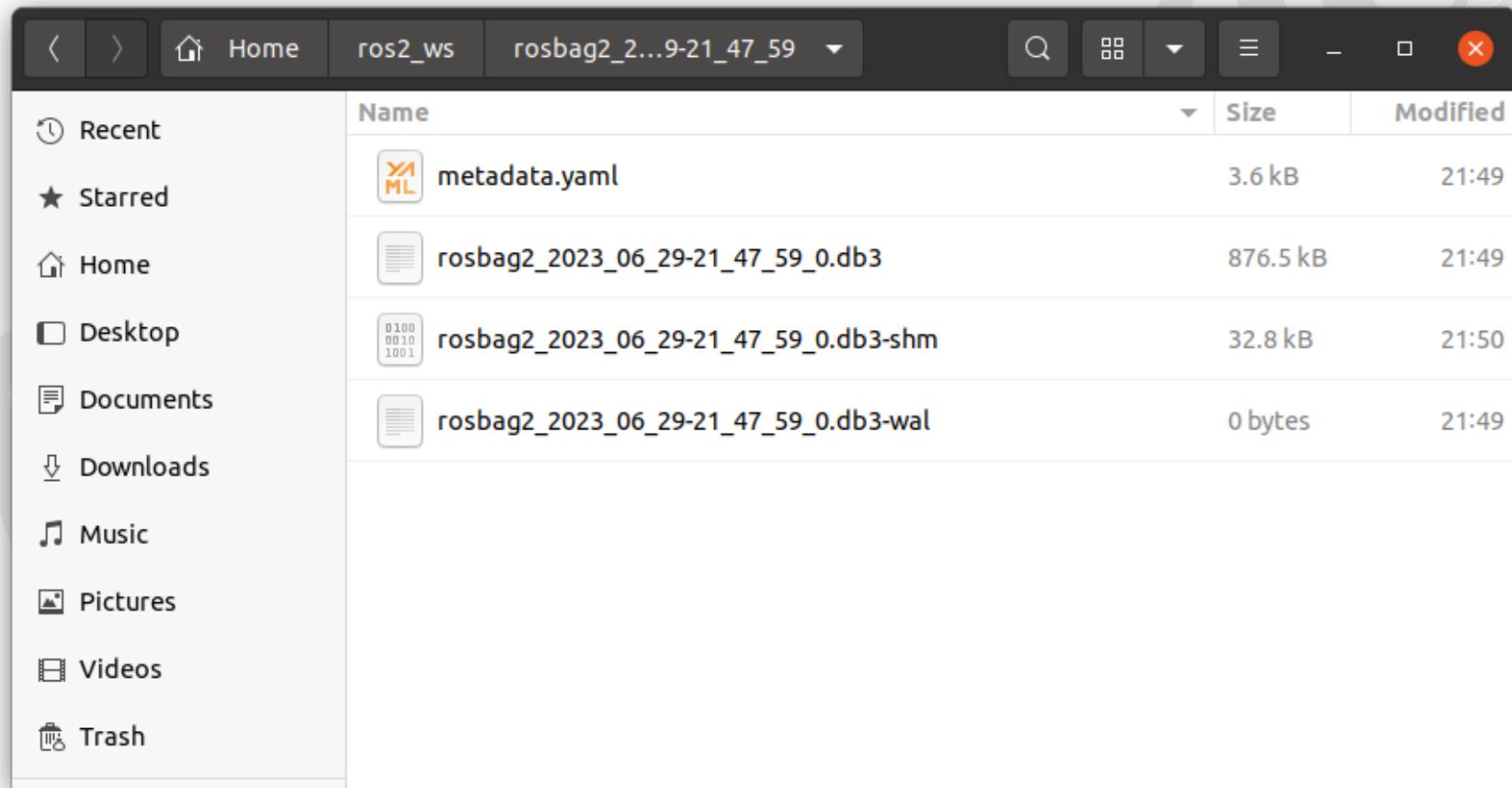


turtlesim

bag 명령어

bag check > saved file list

\$ ros2 bag record -a

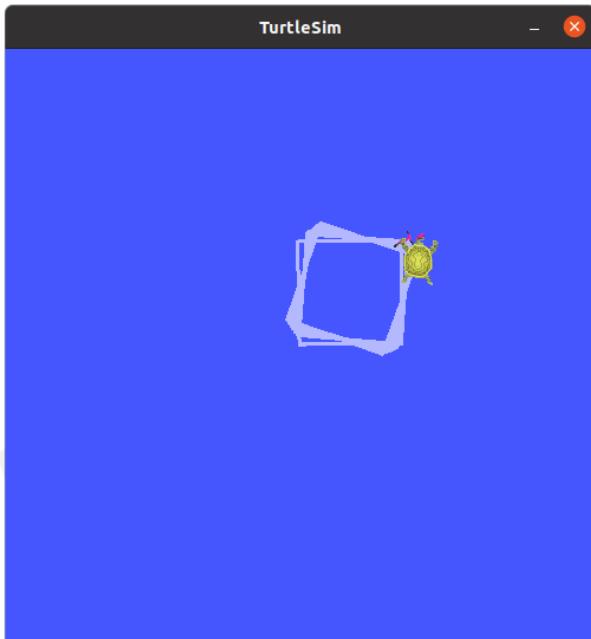


turtlesim

bag 명령어

bag 실행 - topic replay

```
$ ros2 bag play rosbag2_xxx
```



```
rlmodel@rlmodel:~/ros2_ws$ ros2 bag play rosbag2_2023_06_29-21_47_59_0.db3
[INFO] [1688043021.868253324] [rosbag2_storage]: Opened database 'rosbag2_2023_06_29-21_47_59_0.db3' for READONLY.
```

A screenshot of a terminal window titled "rlmodel@rlmodel: ~/ros2_ws". The terminal shows the command "ros2 bag play rosbag2_2023_06_29-21_47_59_0.db3" being run. The output of the command is displayed below the command line, indicating that the database was opened successfully for reading.

turtlesim



bag 명령어

bag 실행

```
$ ros2 bag play rosbag2_xxx
```

```
rlmodel@rlmodel:~/ros2_ws$ ros2 bag play rosbag2_2023_06_29-21_47_59/
rosbag2_2023_06_29-21_47_59_0.db3
[INFO] [1688043021.868253324] [rosbag2_storage]: Opened database 'ros
bag2_2023_06_29-21_47_59/rosbag2_2023_06_29-21_47_59_0.db3' for READ_
ONLY.
```

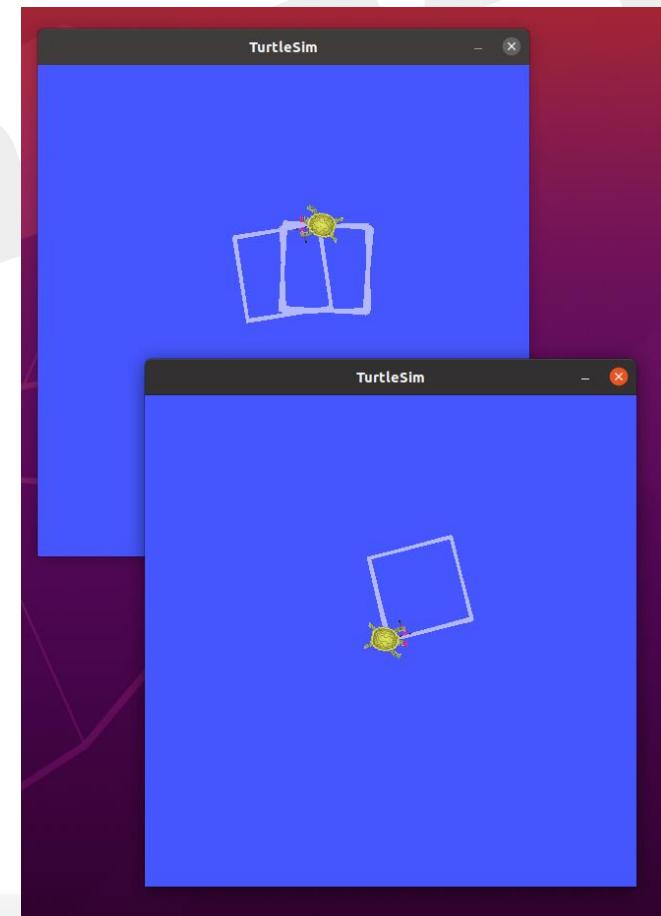
turtlesim

bag 명령어

bag 실행

```
$ ros2 run turtlesim turtlesim_node __node:=new_turtle  
$ ros2 bag play rosbag2_xxx
```

```
rlmodel@rlmodel:~/ros2_ws$ ros2 bag play rosbag2_2023_06_29-21_47_59/  
rosbag2_2023_06_29-21_47_59_0.db3  
[INFO] [1688043021.868253324] [rosbag2_storage]: Opened database 'ros  
bag2_2023_06_29-21_47_59/rosbag2_2023_06_29-21_47_59_0.db3' for READ_  
ONLY.
```

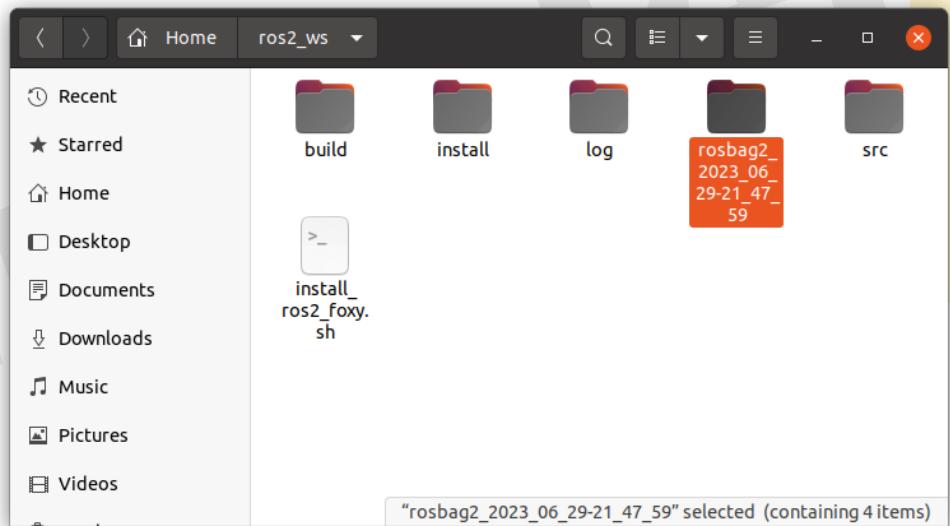
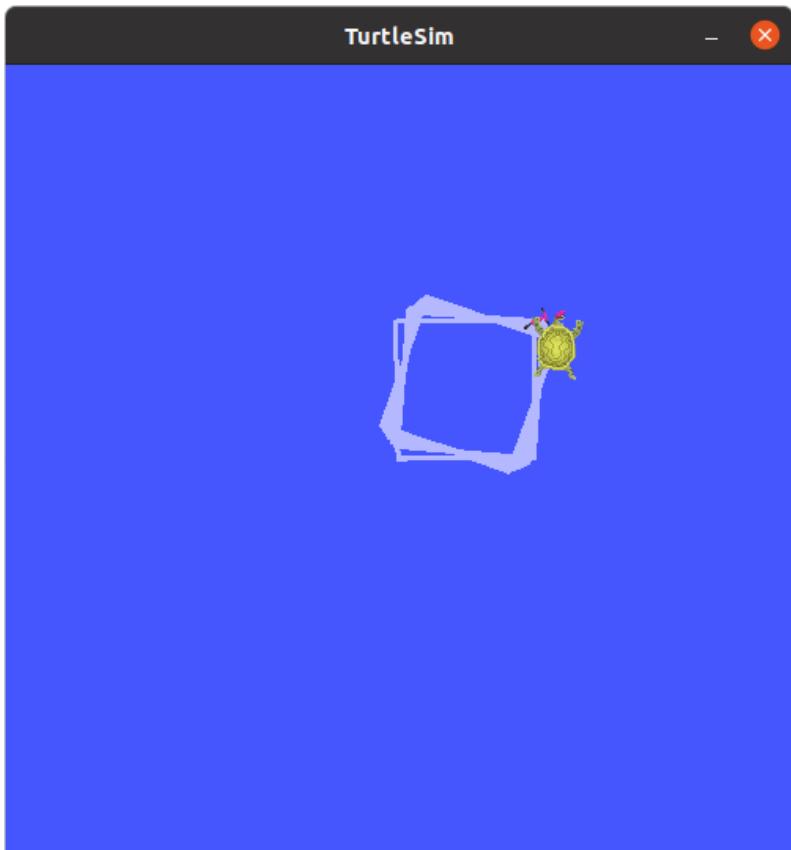


turtlesim

bag 명령어

bag 실행

\$ ros2 bag record -a

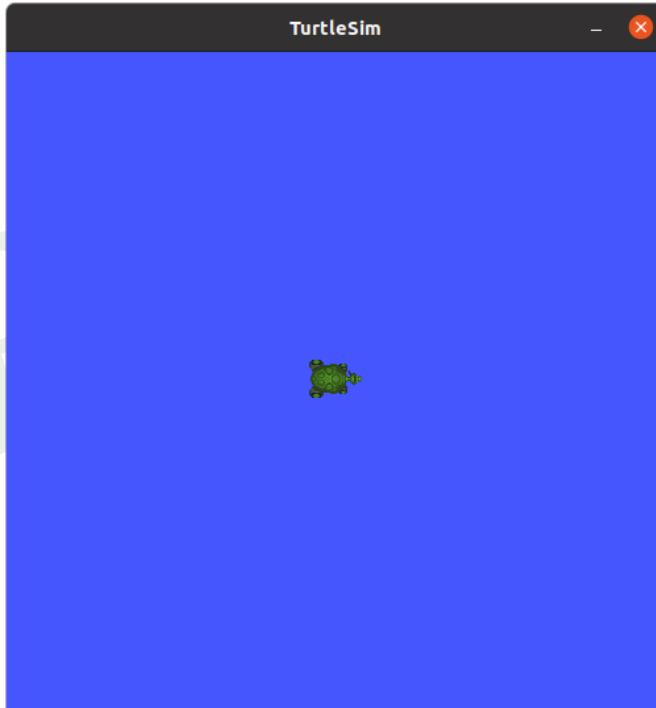


turtlesim

doctor 명령어

ros2 status check

\$ ros2 doctor

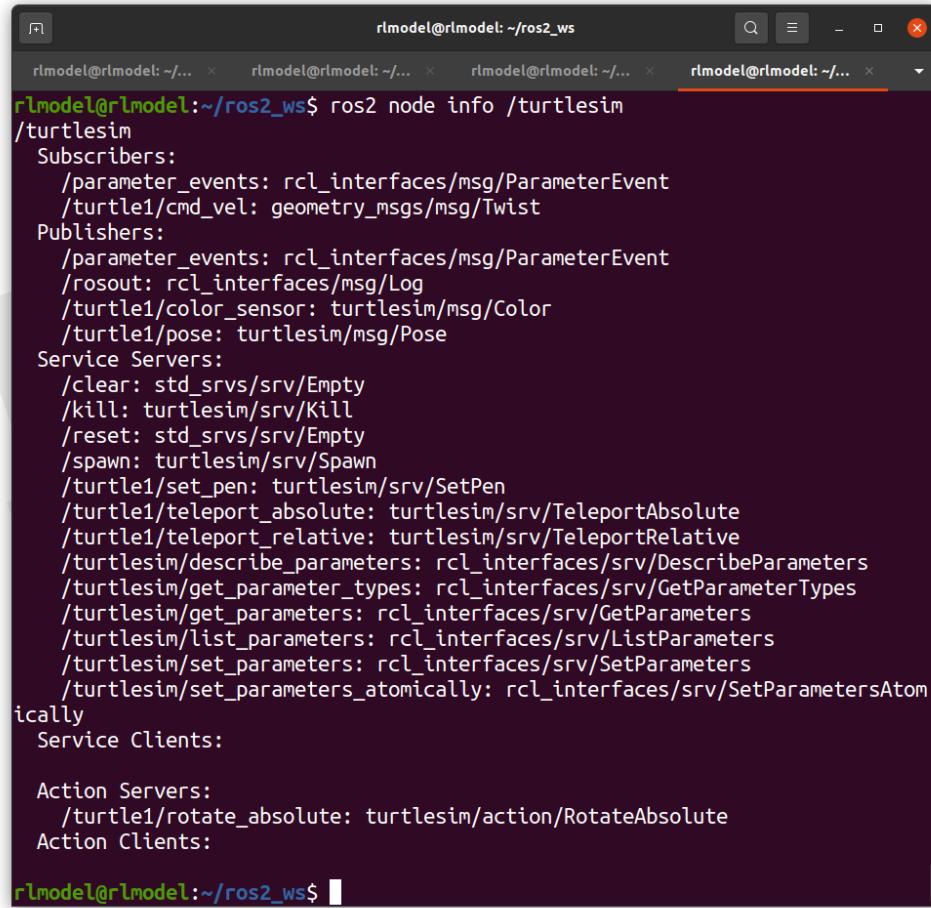


turtlesim

doctor 명령어

ros2 status check

\$ ros2 node info /turtlesim



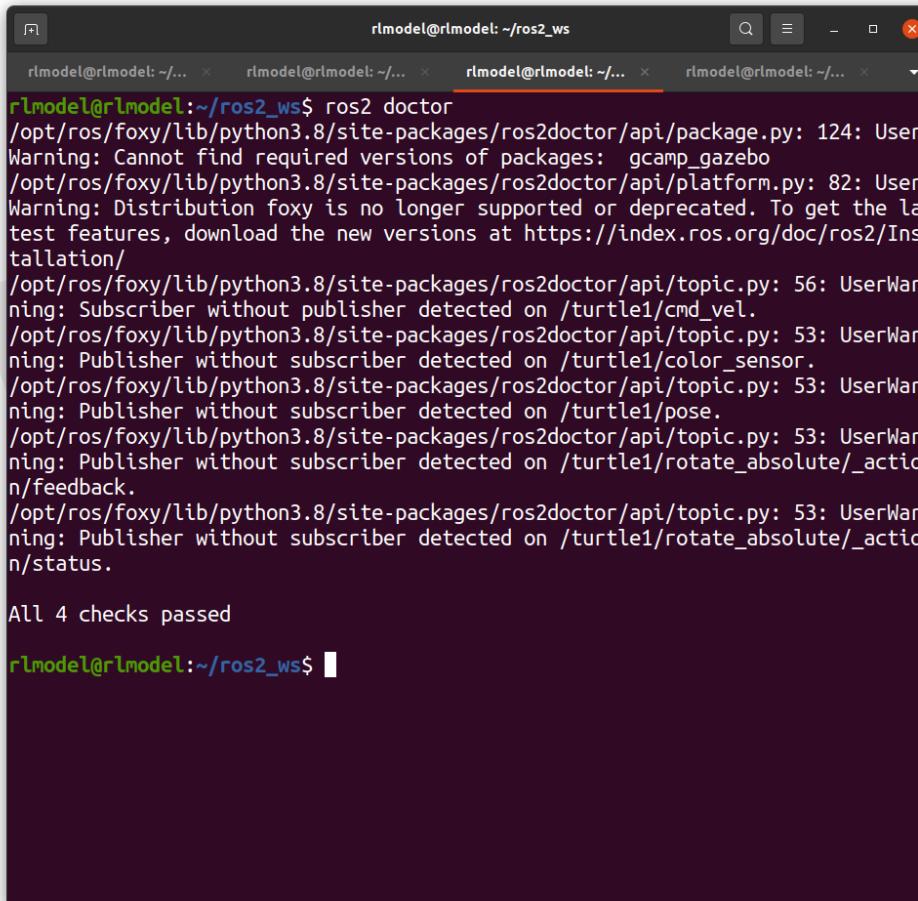
```
rlmodel@rlmodel:~/ros2_ws$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

turtlesim

doctor 명령어

ros2 status check

\$ ros2 doctor



```
rlmodel@rlmodel:~/ros2_ws$ ros2 doctor
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/package.py: 124: User
Warning: Cannot find required versions of packages: gcamp_gazebo
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/platform.py: 82: User
Warning: Distribution foxy is no longer supported or deprecated. To get the la
test features, download the new versions at https://index.ros.org/doc/ros2/Ins
tallation/
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/topic.py: 56: UserWar
ning: Subscriber without publisher detected on /turtle1/cmd_vel.
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/topic.py: 53: UserWar
ning: Publisher without subscriber detected on /turtle1/color_sensor.
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/topic.py: 53: UserWar
ning: Publisher without subscriber detected on /turtle1/pose.
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/topic.py: 53: UserWar
ning: Publisher without subscriber detected on /turtle1/rotate_absolute/_actio
n/feedback.
/opt/ros/foxy/lib/python3.8/site-packages/rosversion/api/topic.py: 53: UserWar
ning: Publisher without subscriber detected on /turtle1/rotate_absolute/_actio
n/status.

All 4 checks passed

rlmodel@rlmodel:~/ros2_ws$
```



turtlesim

doctor 명령어

ros2 status check

```
$ ros2 run turtlesim turtle_teleop_key
```

```
$ ros2 doctor
```

The screenshot shows a terminal window with four tabs, all titled 'rlmodel@rlmodel: ~'. The fourth tab is active. The terminal displays the following text:

```
rlmodel@rlmodel:~/ros2_ws$ ros2 doctor
rlmodel@rlmodel:~/ros2_ws$ ros2 doctor
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/package.py: 124: UserWarning: Cannot find required versions of packages: gcamp_gazebo
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/platform.py: 82: UserWarning: Distribution foxy is no longer supported or deprecated. To get the latest features, download the new versions at https://index.ros.org/doc/ros2/Installation/
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/topic.py: 53: UserWarning: Publisher without subscriber detected on /diffbot/cmd_vel.
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/topic.py: 53: UserWarning: Publisher without subscriber detected on /turtle1/color_sensor.
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/topic.py: 53: UserWarning: Publisher without subscriber detected on /turtle1/pose.

All 4 checks passed
```

turtlesim

doctor 명령어

ros2 status check

\$ rqt

> topic subscribe

- turtle1/color_sensor
- turtle1/pose

Default - rqt

The screenshot shows the 'Topic Monitor' perspective of the rqt interface. It lists the following topics:

Topic	Type	Bandwidth	Hz	Value
/parameter_events	rcl_interfaces/msg/ParameterEvent			not monitored
/rosout	rcl_interfaces/msg/Log			not monitored
/turtle1/cmd_vel	geometry_msgs/msg/Twist			not monitored
/turtle1/rotate_absolute/_action/feedback	turtlesim/action/RotateAbsolute_FeedbackMessage			can not get m...
/turtle1/rotate_absolute/_action/status	action_msgs/msg/GoalStatusArray			not monitored
/turtle1/color_sensor	turtlesim/msg/Color	unknown	62.50	
/turtle1/pose	turtlesim/msg/Pose	unknown	62.49	



turtlesim

doctor 명령어

ros2 status check

\$ ros2 doctor

```
rlmodel@rlmodel: ~/ros2_ws$ ros2 doctor
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/package.py: 124: Use
Warning: Cannot find required versions of packages: gcamp_gazebo
/opt/ros/foxy/lib/python3.8/site-packages/ros2doctor/api/platform.py: 82: Use
Warning: Distribution foxy is no longer supported or deprecated. To get the
latest features, download the new versions at https://index.ros.org/doc/ros2/
Installation/

All 4 checks passed

rlmodel@rlmodel:~/ros2_ws$
```

turtlesim

doctor 명령어

```
ros2 status check  
$ ros2 doctor --report
```

```
rlmodel@rlmodel: ~/ros2_ws$ ros2 doctor --report

NETWORK CONFIGURATION
ether      : 24:f5:aa:db:d8:a9
device     : enp3s0
flags      : 4099<UP,BROADCAST,MULTICAST>
mtu        : 1500
inet       : 127.0.0.1
inet4      : ['127.0.0.1']
inet6      : ['::1']
netmask    : 255.0.0.0
device     : lo
flags      : 73<UP,LOOPBACK,RUNNING>
mtu        : 65536
inet       : 192.168.0.63
inet4      : ['192.168.0.63']
ether      : 6c:29:95:58:a1:bf
inet6      : ['fe80::96d1:cc36:8e6f:f41d']
netmask    : 255.255.255.0
```

NETWORK
CONFIGURATION

...

PLATFORM
INFORMATION

...

RMW MIDDLEWARE

...

ROS 2 INFORMATION

...

TOPIC LIST



turtlesim

launch file 실습

link: <https://docs.ros.org/en/foxy/Tutorials/Intermediate/Launch/Creating-Launch-Files.html>

The screenshot shows the ROS 2 Documentation for the Foxy version. The top navigation bar includes a home icon, the text 'ROS 2 Documentation: Foxy', a search bar, and a 'Edit on GitHub' button. Below the navigation is a cartoon illustration of two foxes sitting at a table with a bowl of food. The main content area has a breadcrumb trail: Home » Tutorials » Intermediate » Launch » Creating a launch file. A note at the top states: 'You're reading the documentation for an older, but still supported, version of ROS 2. For information on the latest version, please have a look at Iron.' The main title is 'Creating a launch file'. Below it, the goal is 'Create a launch file to run a complex ROS 2 system.', the tutorial level is 'Intermediate', and the estimated time is '10 minutes'. A 'Contents' section lists the following outline:

- Prerequisites
- Background
- Tasks
 - 1 Setup
 - 2 Write the launch file
 - 3 ros2 launch
 - 4 Introspect the system with rqt_graph
- Summary



turtlesim

launch file 생성

```
$ mkdir launch
```

```
$ gedit first_turtlesim_launch.py
```

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
    ]
)
)
```



turtlesim

launch file 생성

```
$ gedit second_turtlesim_launch.py
```

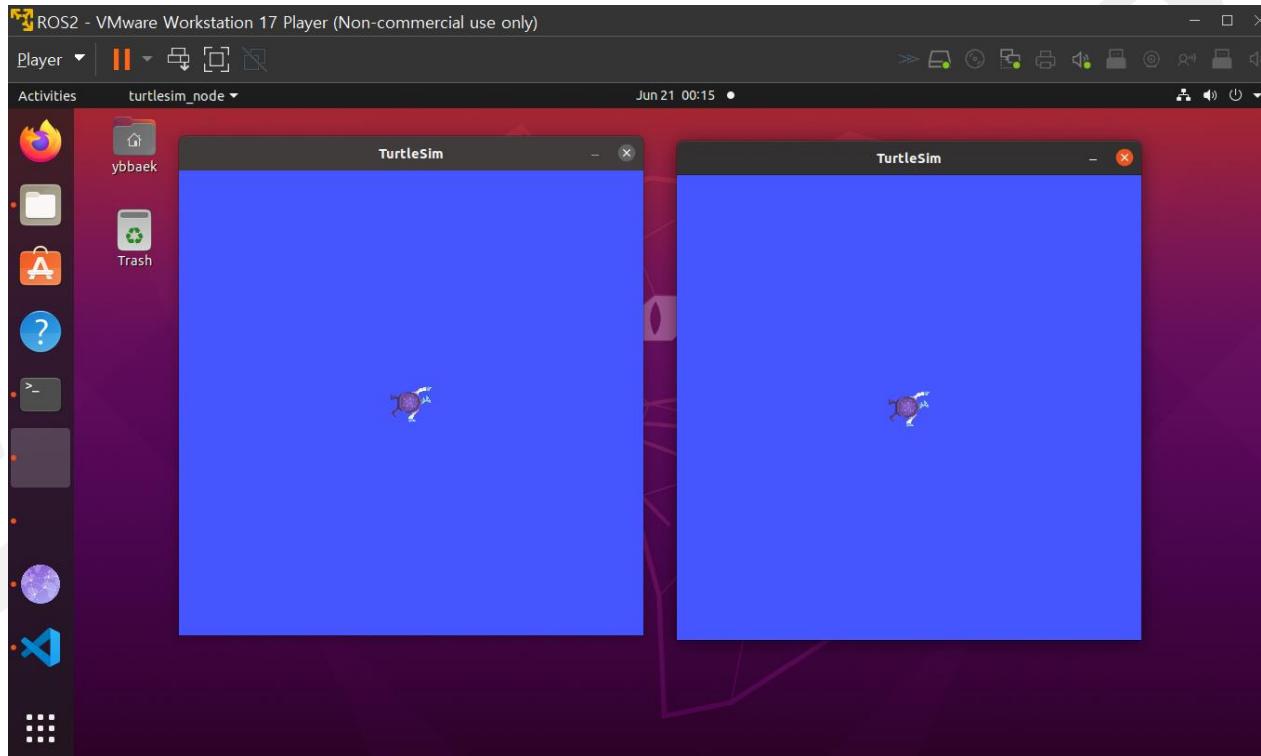
```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel')
            ]
        )
    ])
```

turtlesim

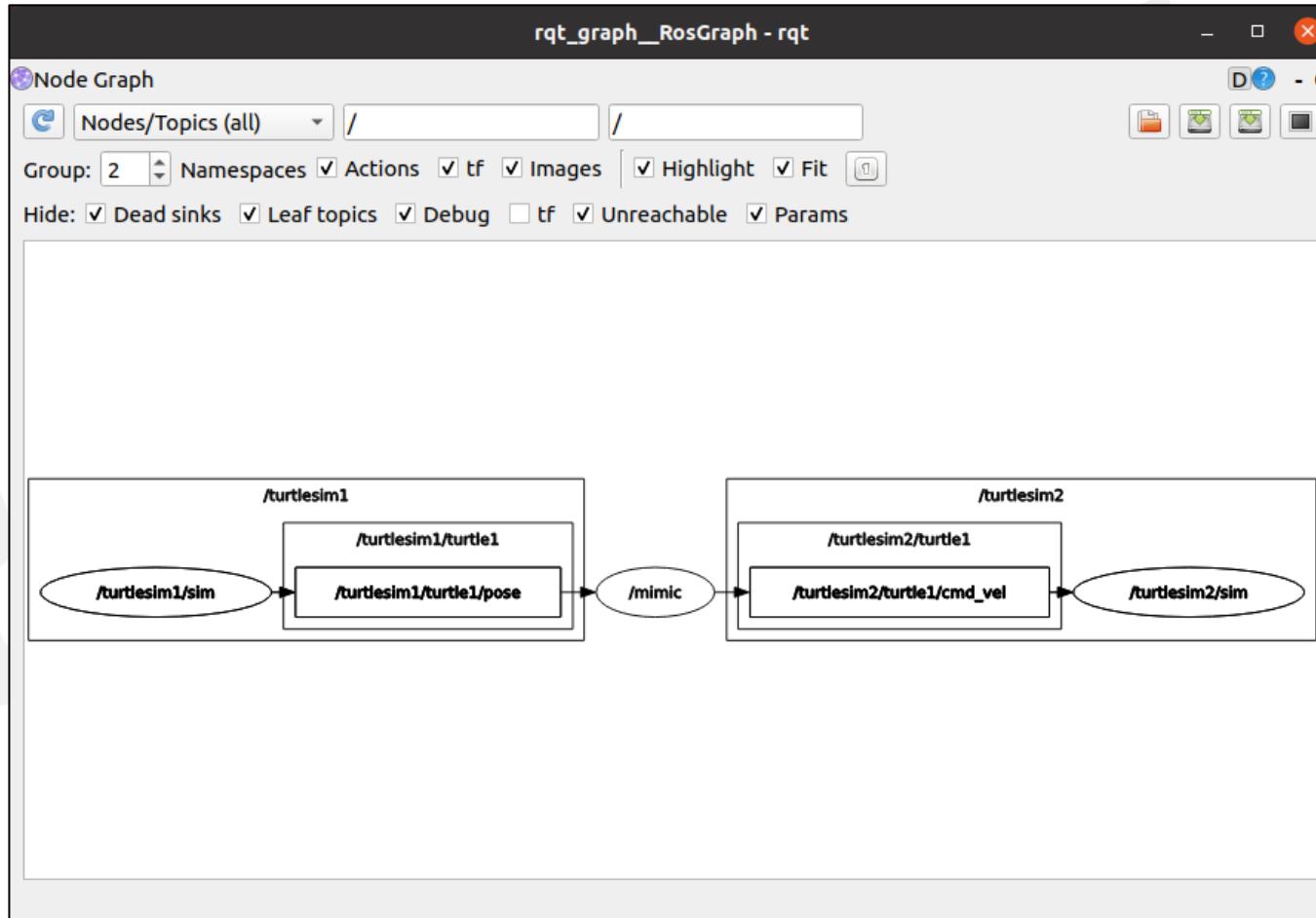
launch file 실행

```
$ mkdir launch  
$ cd launch  
$ ros2 launch first_turtlesim_launch.py
```



turtlesim

launch file 실행 rqt_graph 확인



turtlesim

launch file 실행

연동 확인

```
> ros2 topic pub -r 1 /turtlesim1/turtle1/cmd_vel geometry_msgs/msg/Twist  
"linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: -1.8}"
```





turtlesim

default logger level

levels: Fatal, Error, Warn, Info, Debug

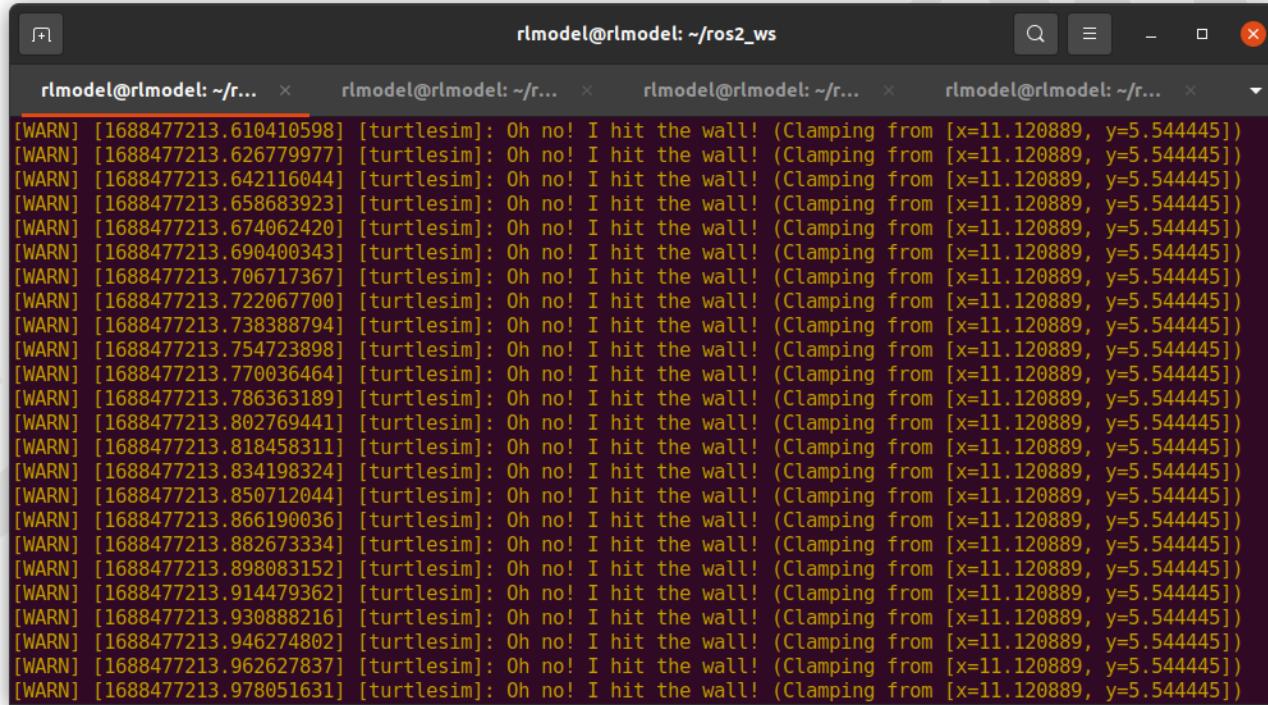
- **Fatal** messages indicate the system is going to terminate to try to protect itself from detriment.
- **Error** messages indicate significant issues that won't necessarily damage the system, but are preventing it from functioning properly.
- **Warn** messages indicate unexpected activity or non-ideal results that might represent a deeper issue, but don't harm functionality outright.
- **Info** messages indicate event and status updates that serve as a visual verification that the system is running as expected.
- **Debug** messages detail the entire step-by-step process of the system execution.

turtlesim

default logger level

levels: Fatal, Error, Warn, Info, Debug

- `ros2 run turtlesim turtlesim_node --ros-args --log-level WARN`

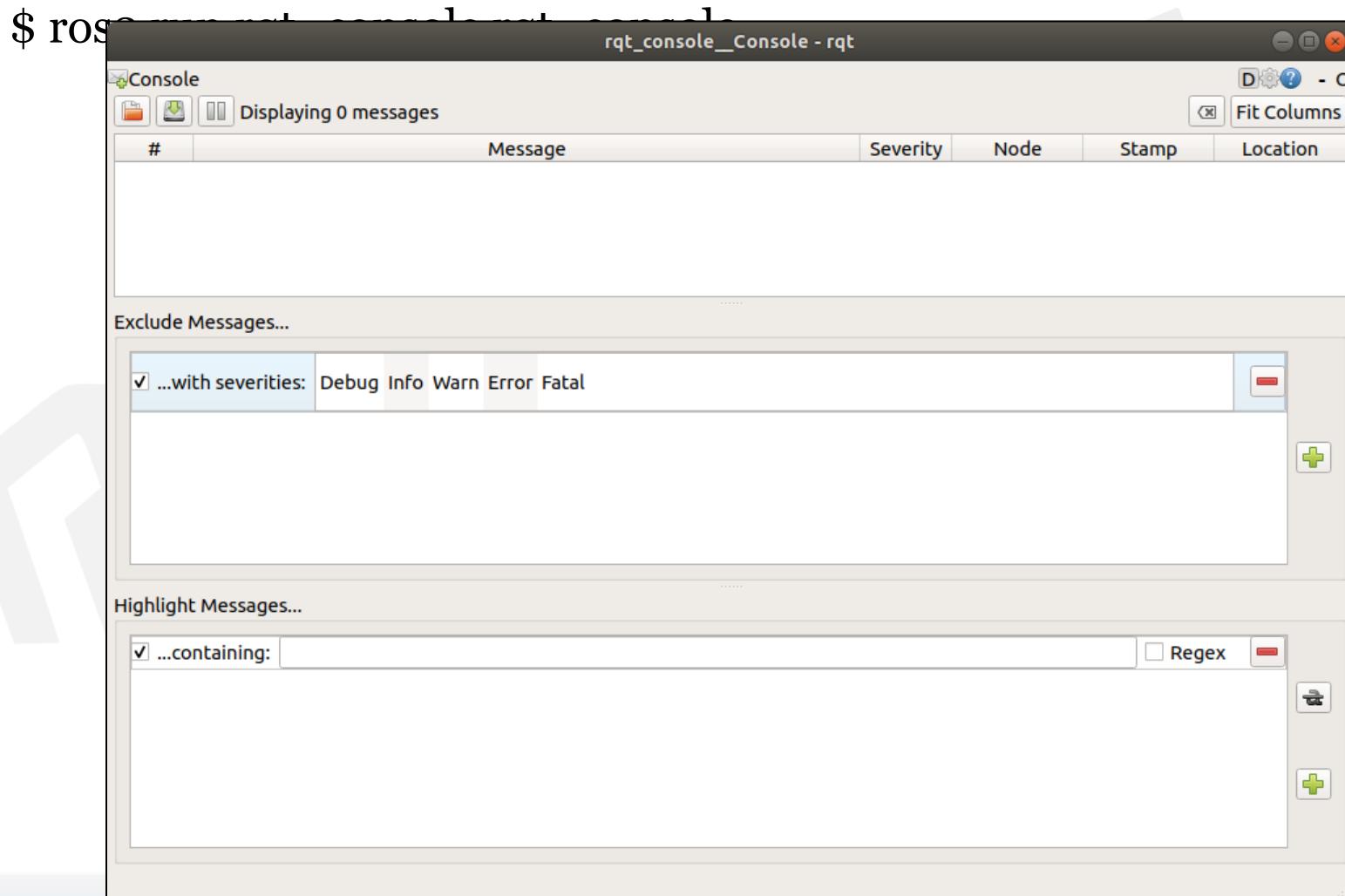


The screenshot shows a terminal window titled "rlmodel@rlmodel: ~/ros2_ws" with four tabs. The active tab displays ROS 2 log messages from the "turtlesim" node. The logs consist of approximately 30 identical entries, each containing a timestamp, a log level indicator ([WARN]), a timestamp again, and a message: "Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])."

```
[WARN] [1688477213.610410598] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.626779977] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.642116044] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.658683923] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.674062420] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.690400343] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.706717367] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.722067700] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.738388794] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.754723898] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.770036464] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.786363189] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.802769441] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.818458311] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.834198324] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.850712044] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.866190036] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.882673334] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.898083152] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.914479362] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.930888216] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.946274802] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.962627837] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
[WARN] [1688477213.978051631] [turtlesim]: Oh no! I hit the wall! (Clamping from [x=11.120889, y=5.544445])
```

turtlesim

rqt_console



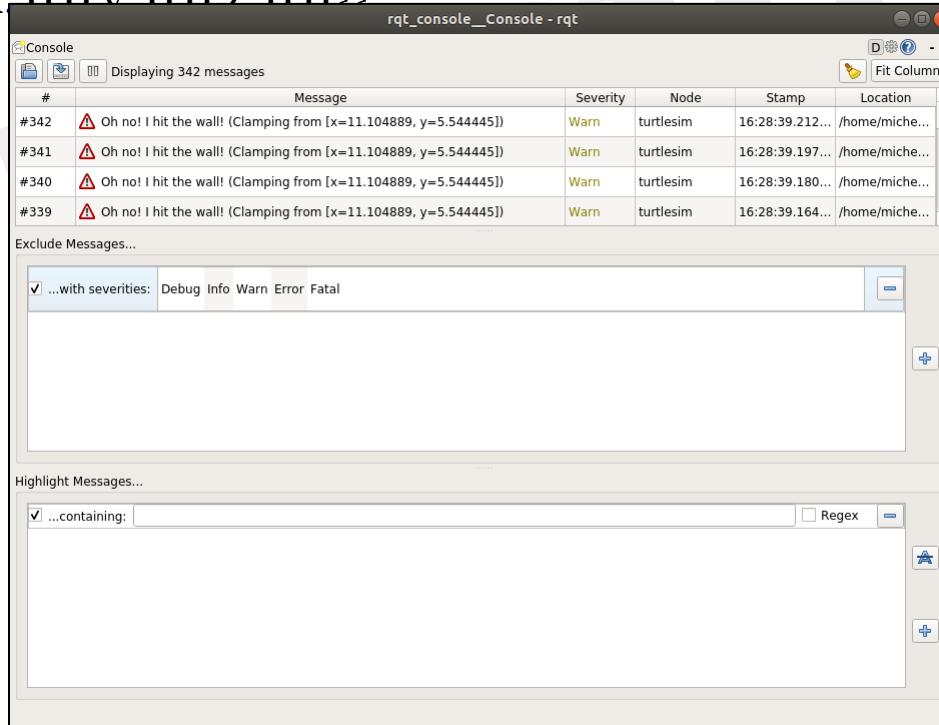
turtlesim

rqt_console

```
ros2 run rqt_console rqt_console
```

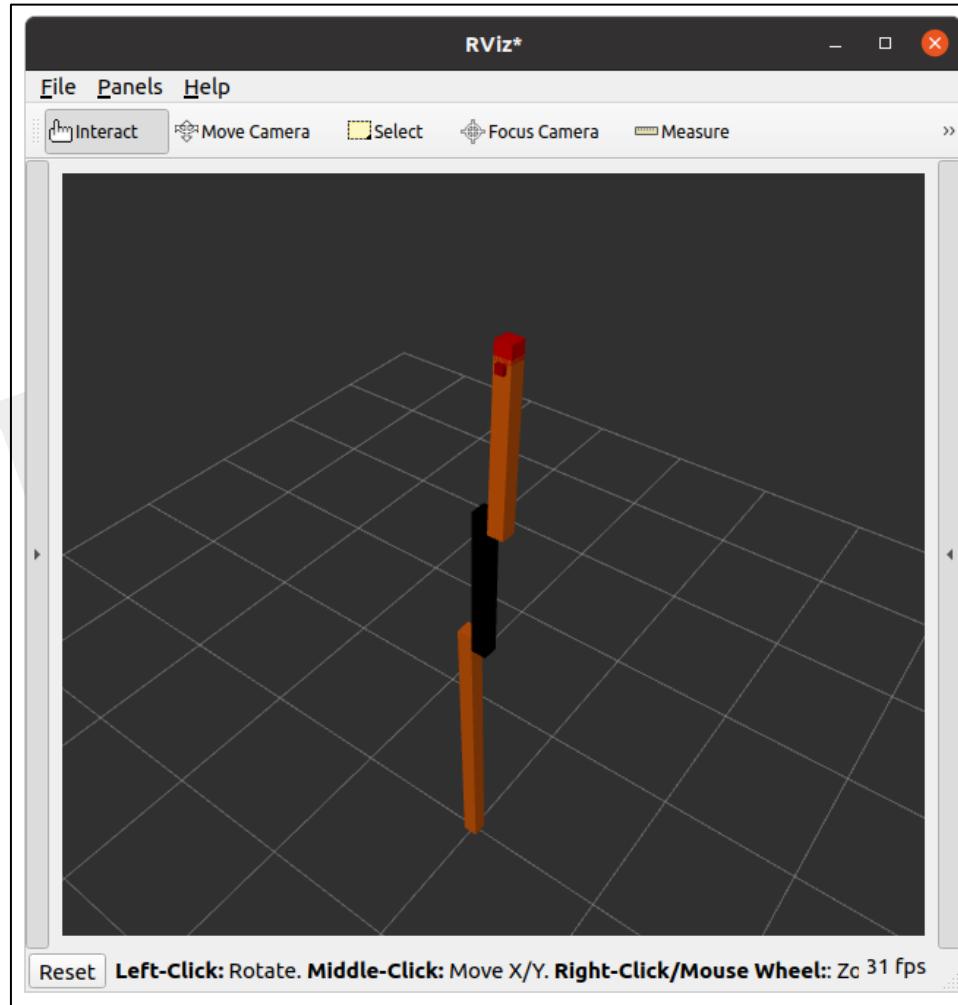
```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```



dummy robot

default executable pkg

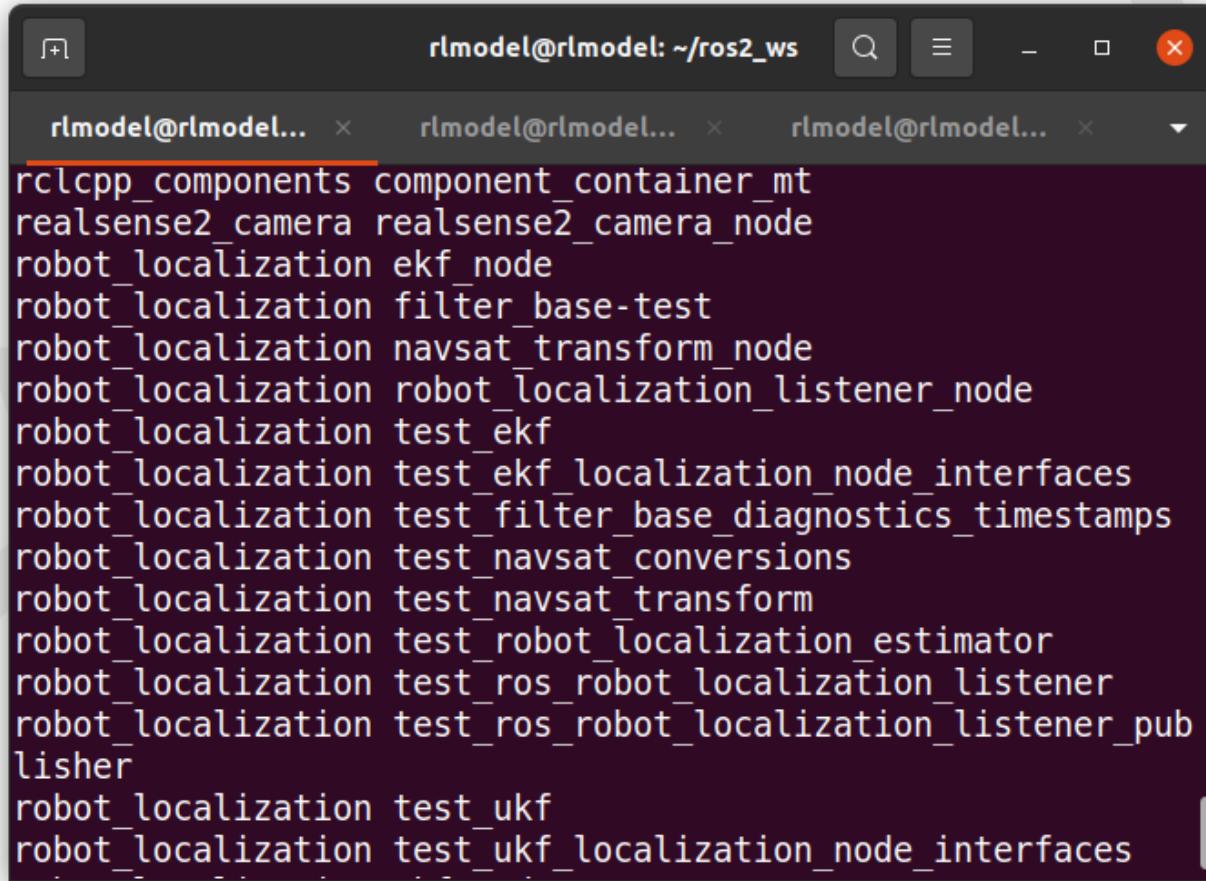


dummy robot

dummy simulation robot 실행

package check

```
$ ros2 pkg executables
```

A terminal window titled "rlmodel@rlmodel: ~/ros2_ws" showing the output of the command "ros2 pkg executables". The window has three tabs: "rclcpp_components component_container_mt", "realsense2_camera realsense2_camera_node", and "robot_localization ekf_node".

```
rclcpp_components component_container_mt
realsense2_camera realsense2_camera_node
robot_localization ekf_node
robot_localization filter_base-test
robot_localization navsat_transform_node
robot_localization robot_localization_listener_node
robot_localization test_ekf
robot_localization test_ekf_localization_node_interfaces
robot_localization test_filter_base_diagnostics_timestamps
robot_localization test_navsat_conversions
robot_localization test_navsat_transform
robot_localization test_robot_localization_estimator
robot_localization test_ros_robot_localization_listener
robot_localization test_ros_robot_localization_publisher
robot_localization test_ukf
robot_localization test_ukf_localization_node_interfaces
```



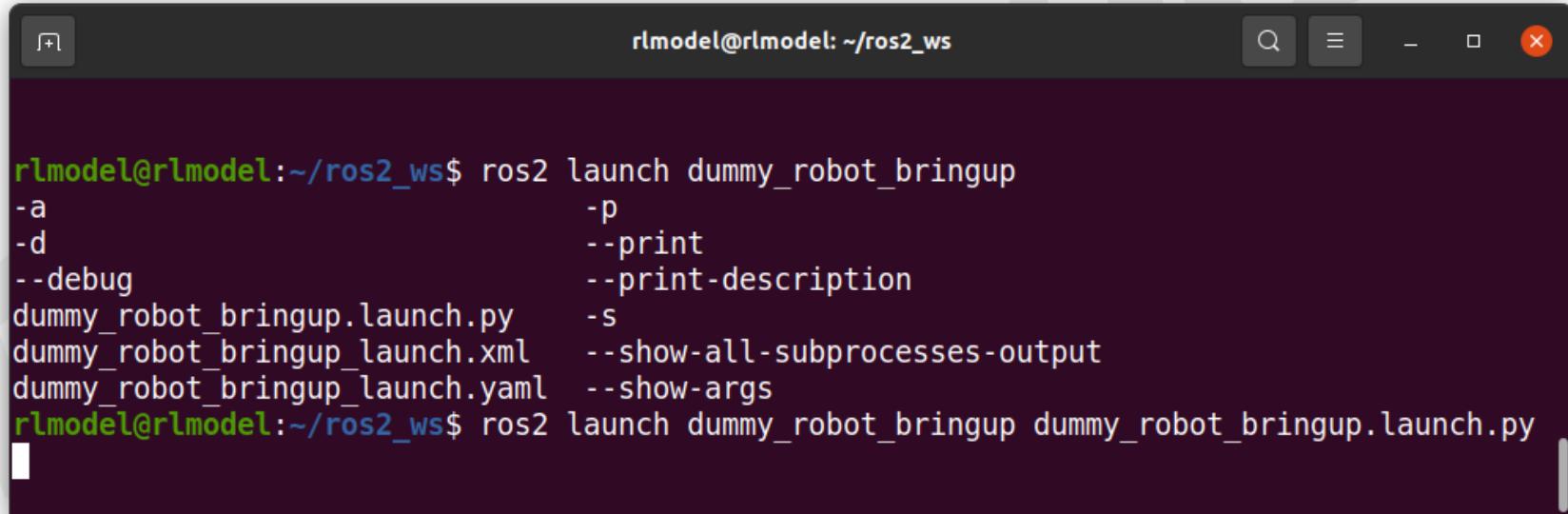
dummy robot

dummy simulation robot 실행

>run dummy_robot

```
$ source ~/ros2_ws/install/setup.bash
```

```
$ ros2 launch dummy_robot_bringup dummy_robot_bringup.launch.py
```



The screenshot shows a terminal window with a dark theme. The title bar reads "rlmodel@rlmodel: ~/ros2_ws". The terminal content displays the following ROS 2 commands:

```
rlmodel@rlmodel:~/ros2_ws$ ros2 launch dummy_robot_bringup
-a
-d
--debug
dummy_robot_bringup.launch.py
dummy_robot_bringup_launch.xml
dummy_robot_bringup_launch.yaml
-p
--print
--print-description
-s
--show-all-subprocesses-output
--show-args
rlmodel@rlmodel:~/ros2_ws$ ros2 launch dummy_robot_bringup dummy_robot_bringup.launch.py
```

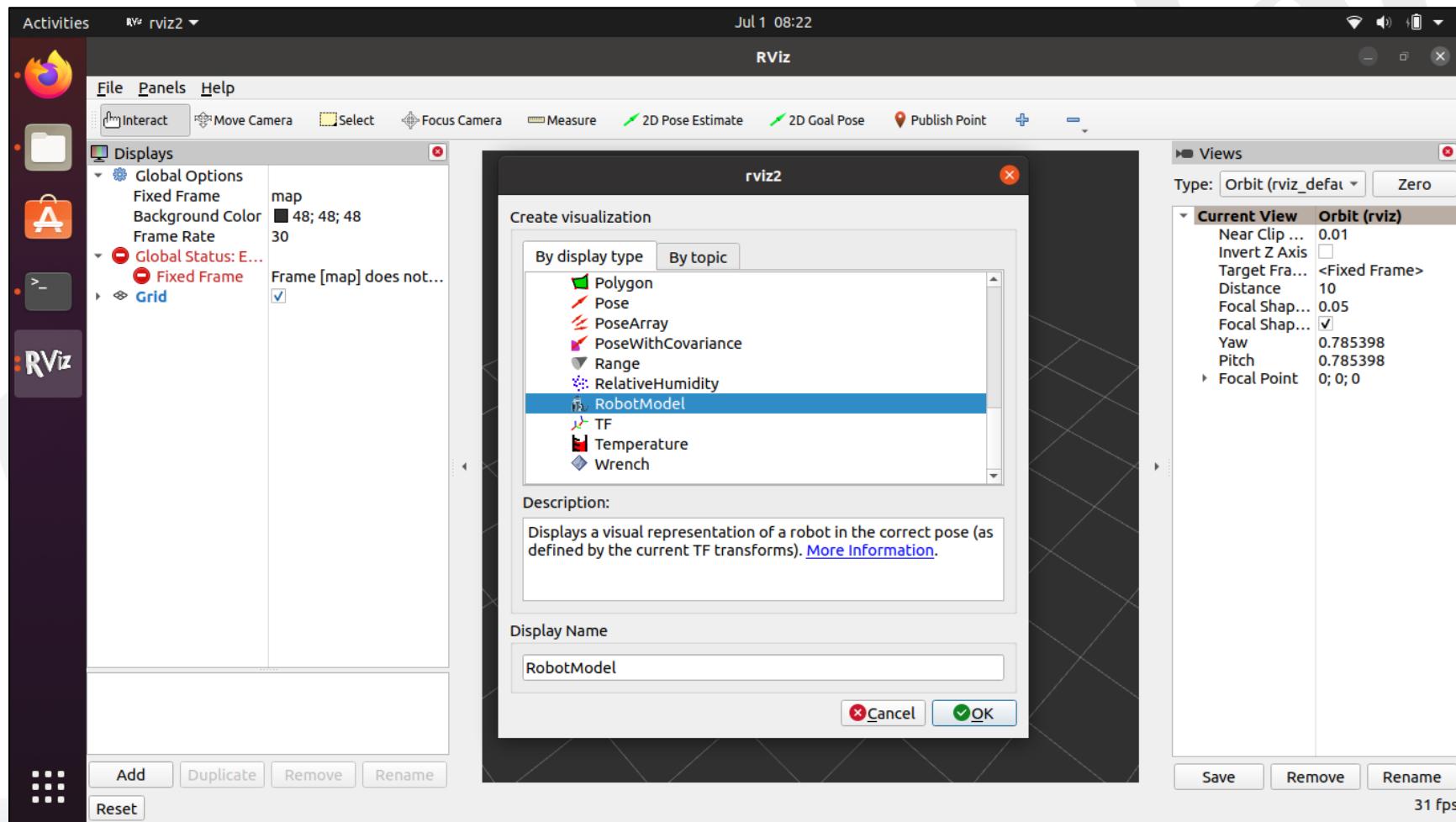


dummy robot

dummy simulation robot 실행

>run rviz

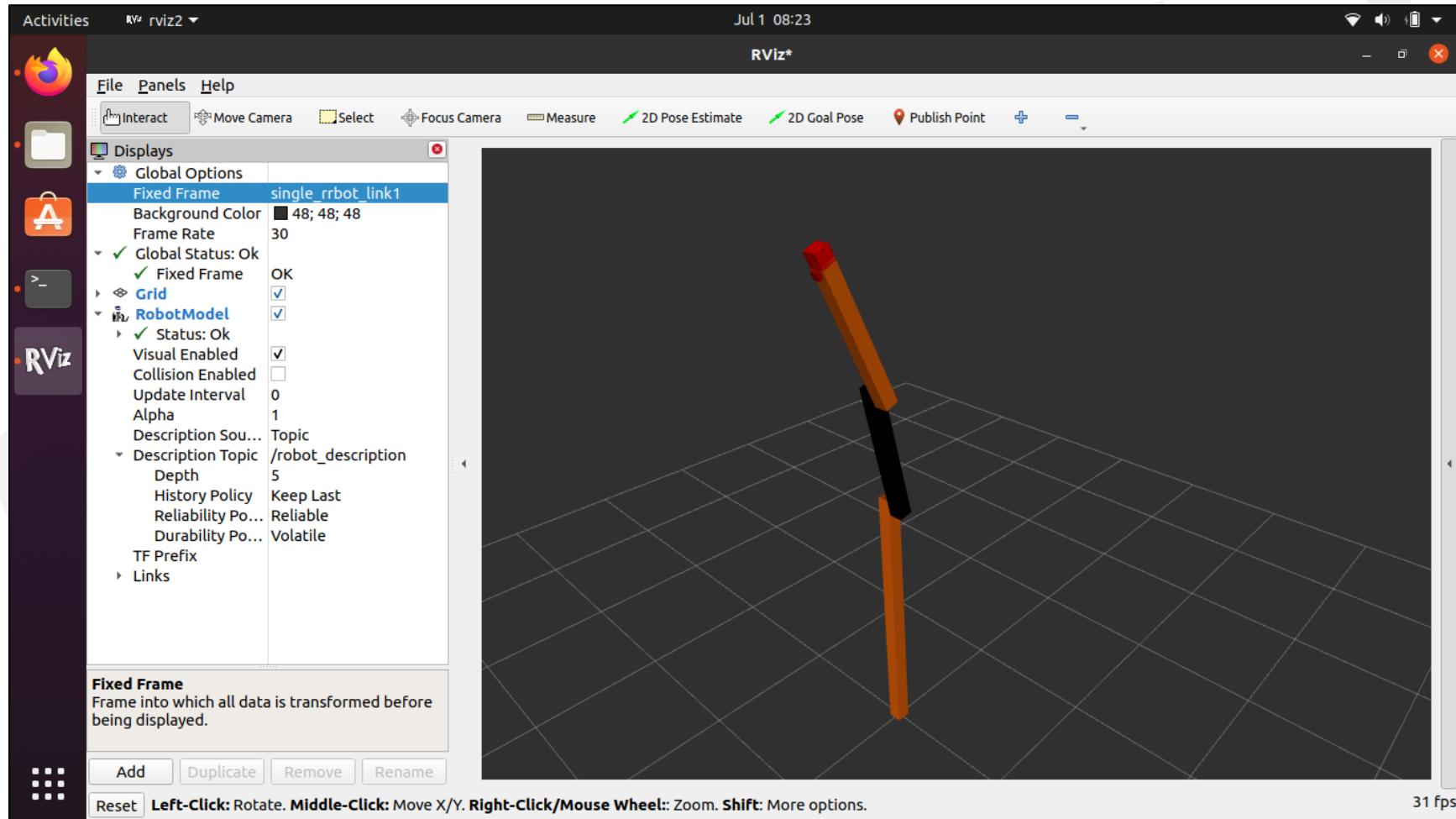
>add - RobotModel



dummy robot

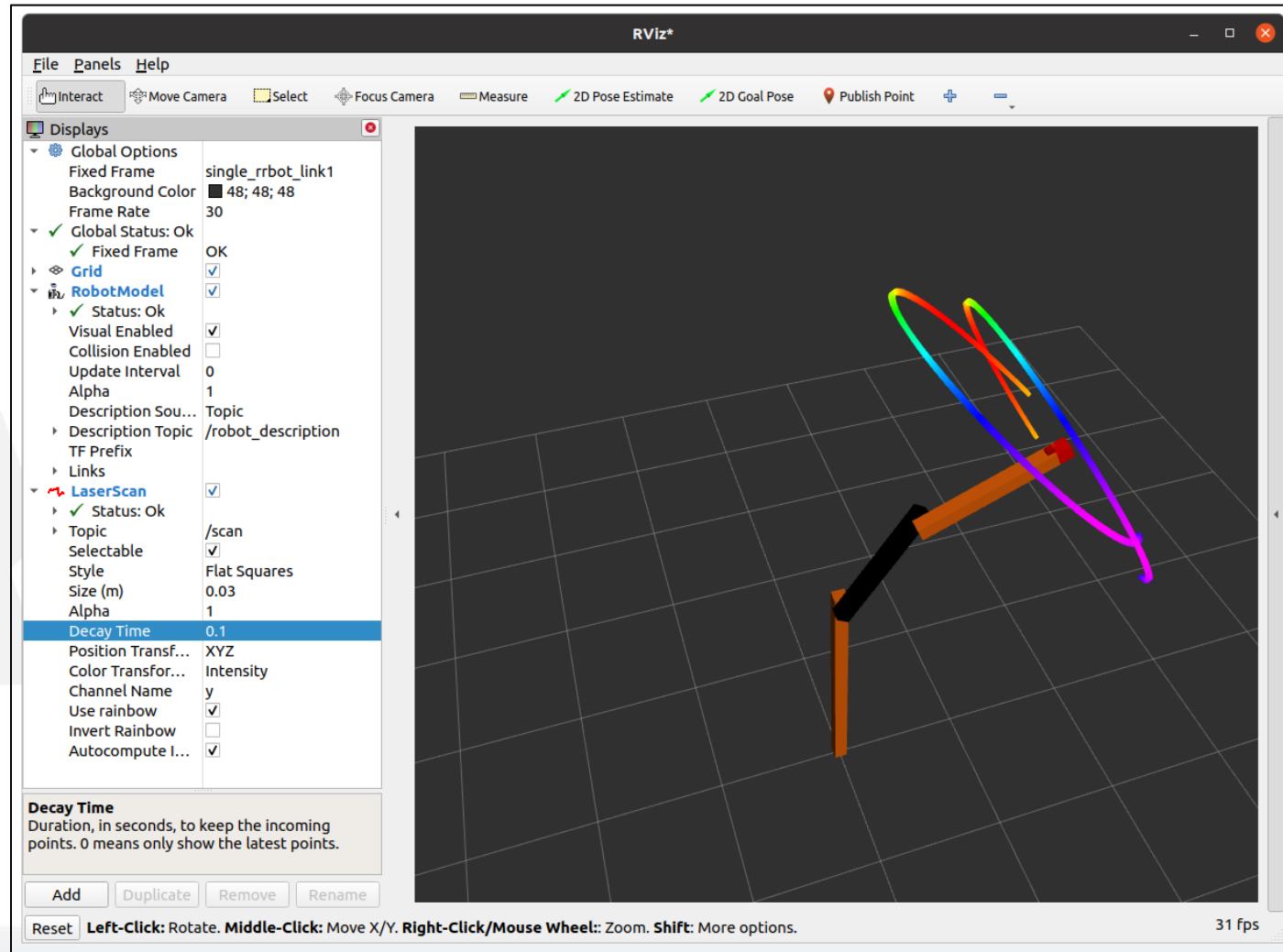
dummy simulation robot 실행

>check TF, Frame id



dummy robot

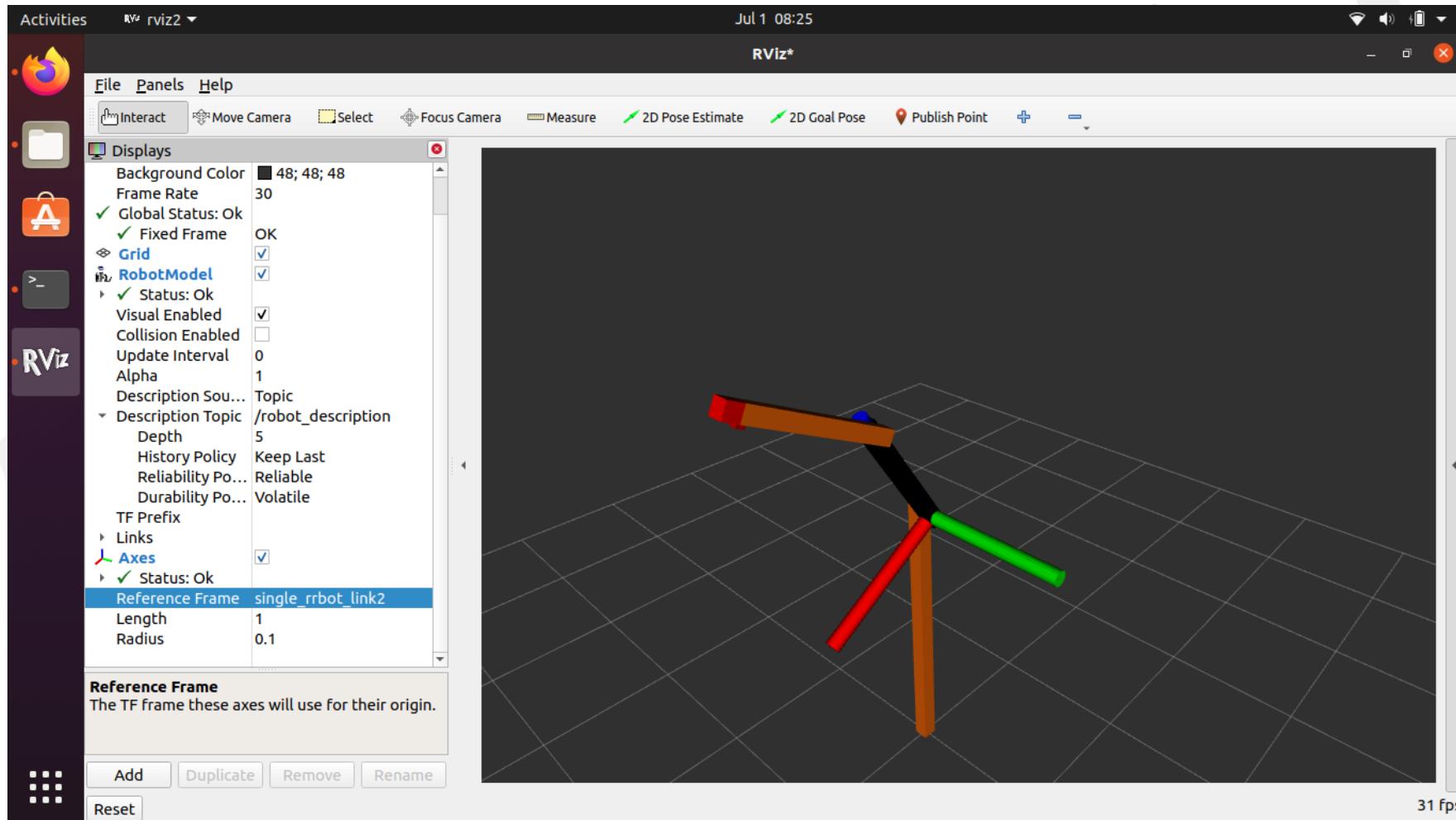
dummy simulation robot 실행 >check /scan topic



dummy robot

dummy simulation robot 실행

>check TF, Frame id





Gazebo

3D simulation



Simulation

Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces.



Gazebo 구성

공식문서 (old)

link: <https://classic.gazebosim.org/>

The screenshot shows the homepage of the Gazebo classic website. At the top, there is a navigation bar with links to Tutorials, Download, Blog, Media, and Projects. A search bar is located on the right side of the header. The main content area features a large Gazebo logo with a 3D cube icon. Below the logo, the tagline "Robot simulation made easy." is displayed. Two prominent buttons are present: a blue "Download (11.0.0)" button and a grey "View on GitHub" button. A yellow callout box contains the text: "A new version of Gazebo (formerly known as Ignition) is now available. Please visit <https://gazebosim.org> to learn more." Below this, there are links for "GET STARTED", "FEATURES", and "STATUS". On the right side of the page, there are sections for "SDFormat : Robot and simulation model format" and "Ignition : Libraries for robot applications", each with its respective icon. Social media icons for Google+, YouTube, and Twitter are also visible. The bottom section contains three columns: "Why Gazebo?", "The Latest", and "Useful Links".

Why Gazebo?

Robot simulation is an essential tool in every roboticist's toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant

The Latest

Gazebo 11.0.0 release
2019-01-30

Download (11.0.0)

[Changelog | Migration Guide](#)

Release Highlights

- SDFormat 1.7 frame semantics, see the [ROSCon](#)

Useful Links

Answers
Find answers and ask questions.

Community
Join for discussions and announcements.

Simulation Models
Robots, objects, and other simulation models.

Gazebo

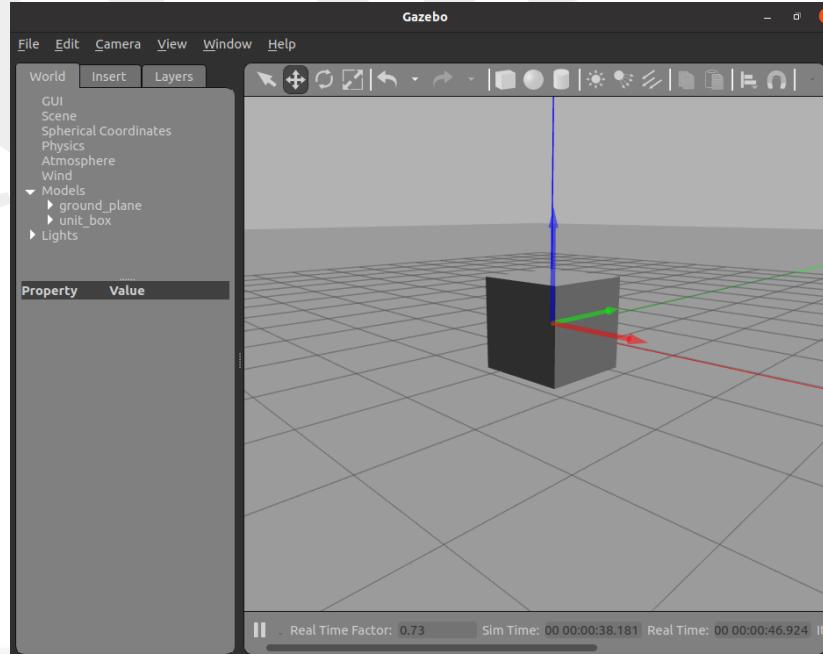
gazebo 소개

개요: Gazebo는 ROS와 별개의 프로그램, 오픈소스 가상 시뮬레이션 툴.

URDF, SDF, Xacro 등의 XML 문서로 로봇을 가상 시뮬레이션에 삽입한 후, 로봇과 사용자가 작성한 프로그램이 메시지를 주고 받으며 동작하는 모습을 확인할 수 있음.

설치: sudo apt install gazebo11

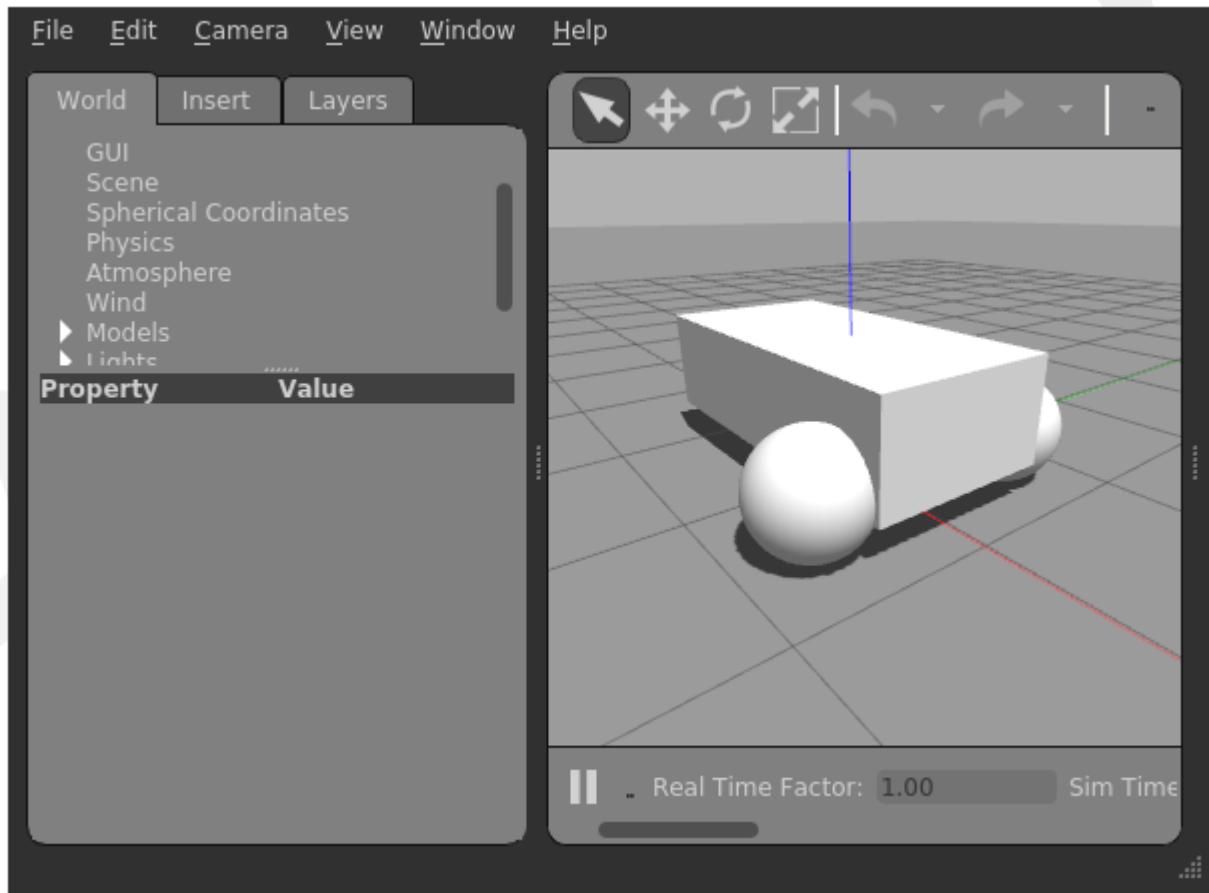
실행: \$ gazebo



Gazebo 구성

실행

```
gazebo --verbose /opt/ros/foxy/share/gazebo_plugins/worlds/gazebo_ros_diff_drive_demo.world  
ros2 topic pub /demo/cmd_demo geometry_msgs/Twist '{linear: {x: 1.0}}' -1
```





Gazebo 구성

설치

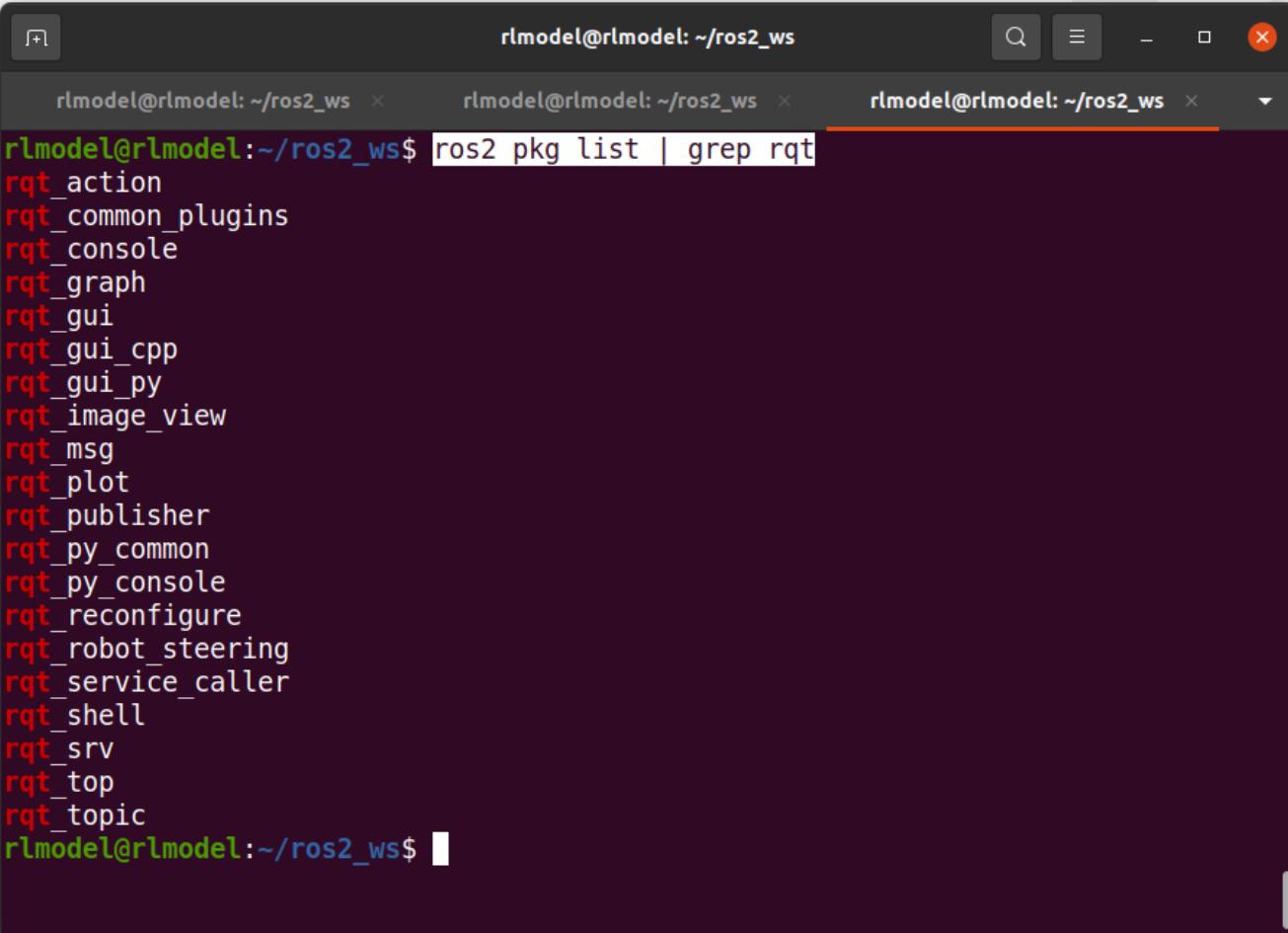
```
$ sudo apt-get update  
$ sudo apt-get install gazebo11  
$ sudo apt-get install libgazebo11-dev  
$ gazebo --version
```

```
$ sudo apt install ros-foxy-gazebo-ros-pkgs  
$ sudo apt install ros-foxy-ros-core ros-foxy-geometry2
```

Gazebo play

rqt_robot_control

check pkg: ros2 pkg list | grep rqt



```
rlmodel@rlmodel:~/ros2_ws$ ros2 pkg list | grep rqt
rqt_action
rqt_common_plugins
rqt_console
rqt_graph
rqt_gui
rqt_gui_cpp
rqt_gui_py
rqt_image_view
rqt_msg
rqt_plot
rqt_publisher
rqt_py_common
rqt_py_console
rqt_reconfigure
rqt_robot_steering
rqt_service_caller
rqt_shell
rqt_srv
rqt_top
rqt_topic
rlmodel@rlmodel:~/ros2_ws$
```

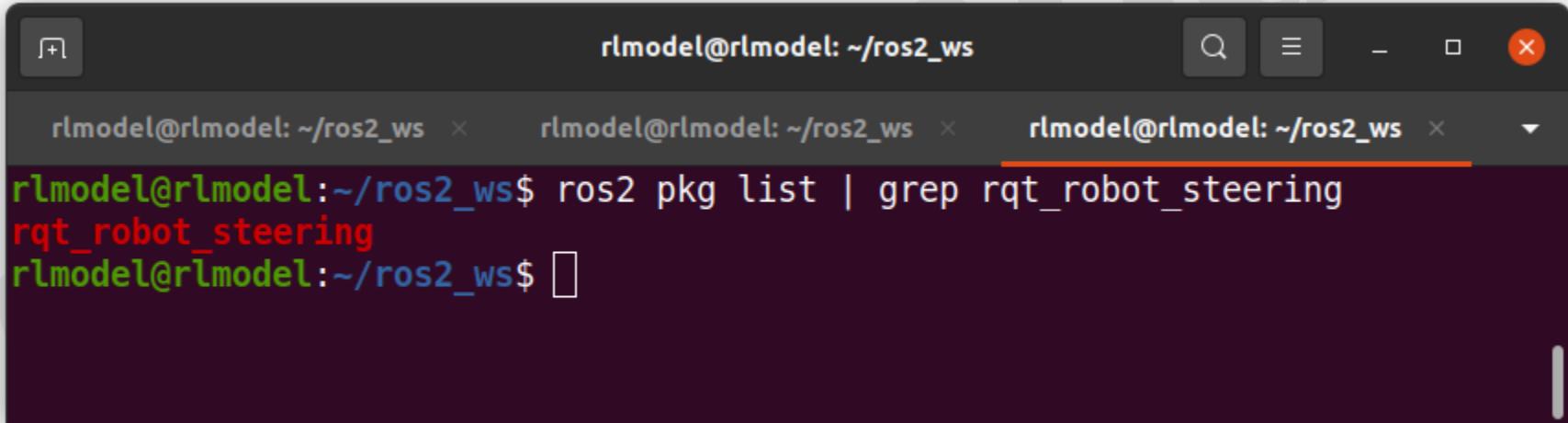
Gazebo play

rqt_robot_control

check pkg: rqt_robot_steering

install: sudo apt-get install ros-**kinetic**-rqt-robot-steering (x)

install: sudo apt-get install ros-**foxy**-rqt-robot-steering (o)



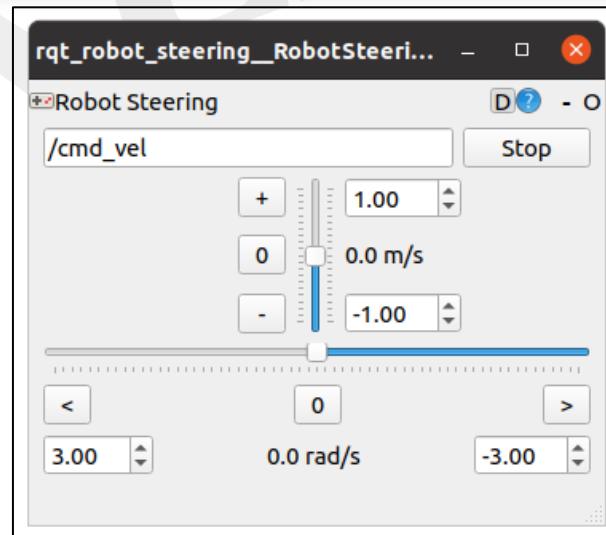
```
rlmodel@rlmodel: ~/ros2_ws$ ros2 pkg list | grep rqt_robot_steering
rqt_robot_steering
rlmodel@rlmodel:~/ros2_ws$
```

Gazebo play

rqt_robot_control

run: rqt_robot_steering

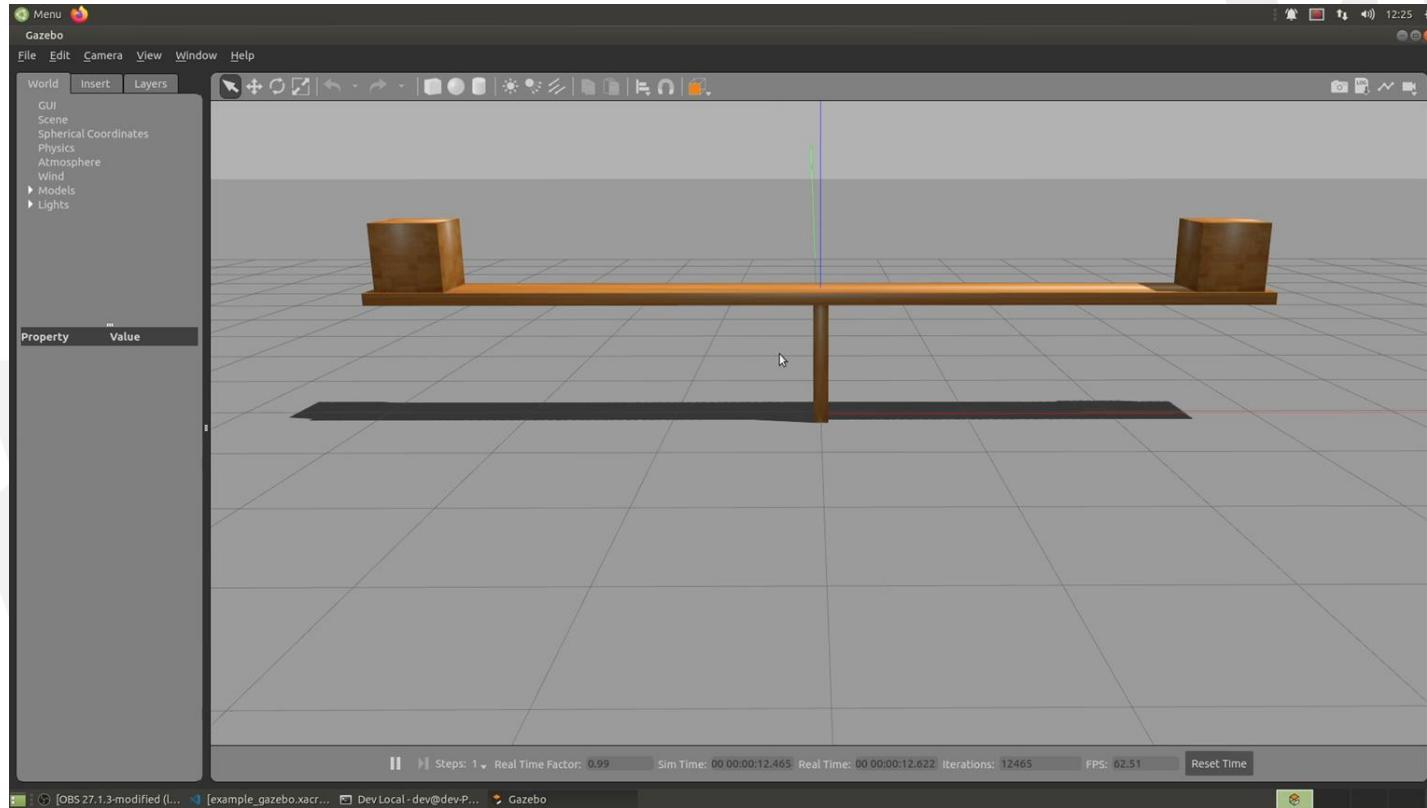
```
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws$ ros2 run rqt_robot_steering rqt_robot_steering
```



Gazebo 예제

Basic 예제

```
$ gazebo /usr/share/gazebo-11/worlds/seesaw.world
```





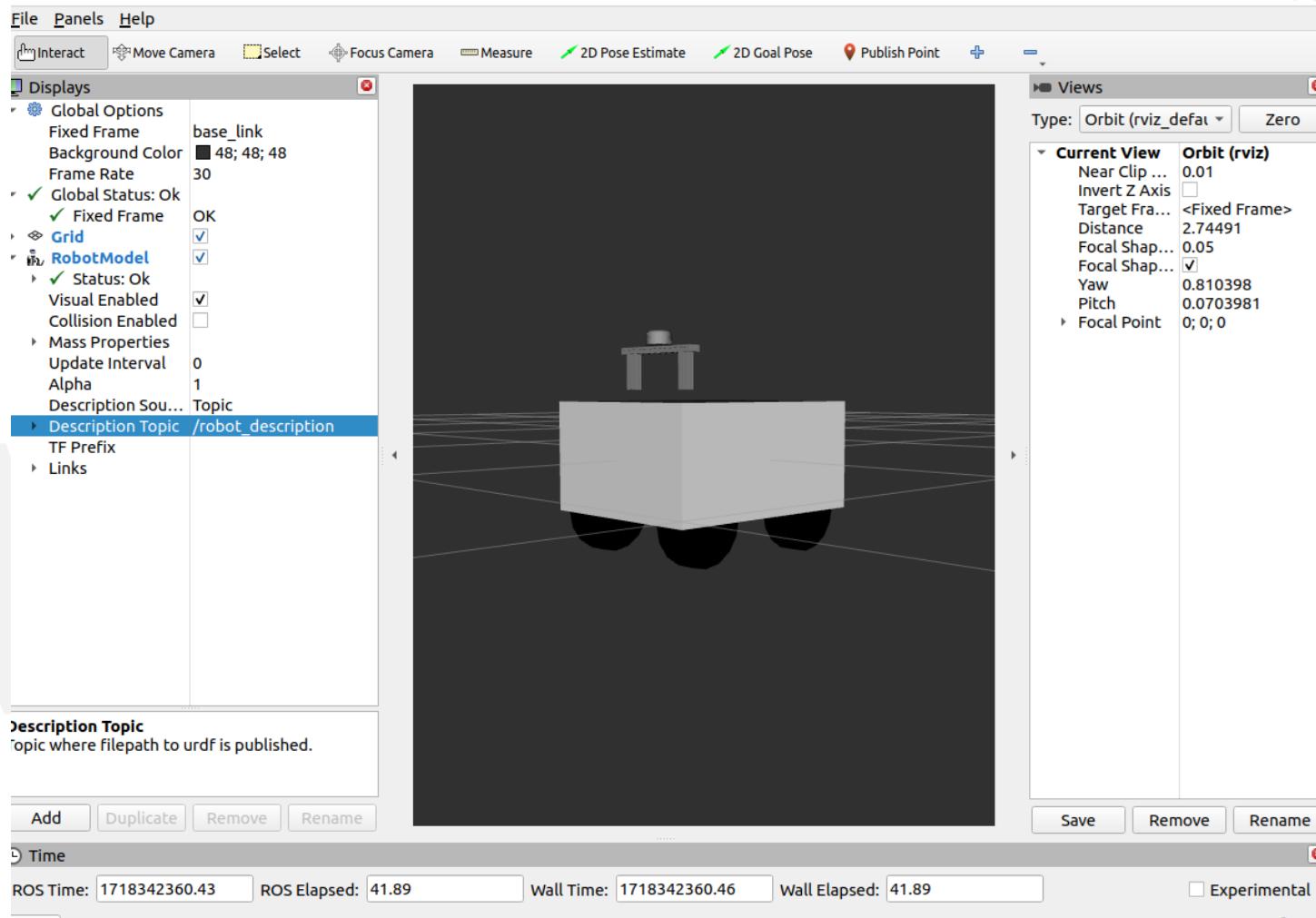
Gazebo 예제

URDF 예제

```
1  <?xml version="1.0"?>
2  ~<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="urdf_tutorial">
3
4      <!-- COLOR -->
5      <material name="white">
6          | <color rgba="1 1 1 1" />
7      </material>
8
9      <material name="blue">
10         | <color rgba="0 0 1 1"/>
11     </material>
12
13     <material name="black">
14         | <color rgba="0 0 0 1"/>
15     </material>
16
17
18
19     <link name="base_link">
20         <visual>
21             | <origin xyz="0 0 0"/>
22             | <geometry>
23                 |   <box size="1.04 0.85 0.45"/>
24             | </geometry>
25             | <material name="white"/>
26         </visual>
27         <collision>
28             | <origin xyz="0 0 0"/>
29             | <geometry>
30                 |   <box size="1.04 0.85 0.45"/>
31             | </geometry>
32         </collision>
33     </link>
34
35     <!-- LEFT WHEEL LINK -->
36
37     <link name="left_wheel">
38         <visual>
39             | <geometry>
40                 |   <cylinder radius="0.06" length="0.06"/>
41             | </geometry>
42             | <material name="black"/>
43         </visual>
44         <collision>
45             | <geometry>
46                 |   <cylinder radius="0.06" length="0.06"/>
47             | </geometry>
48         </collision>
```

Gazebo 예제

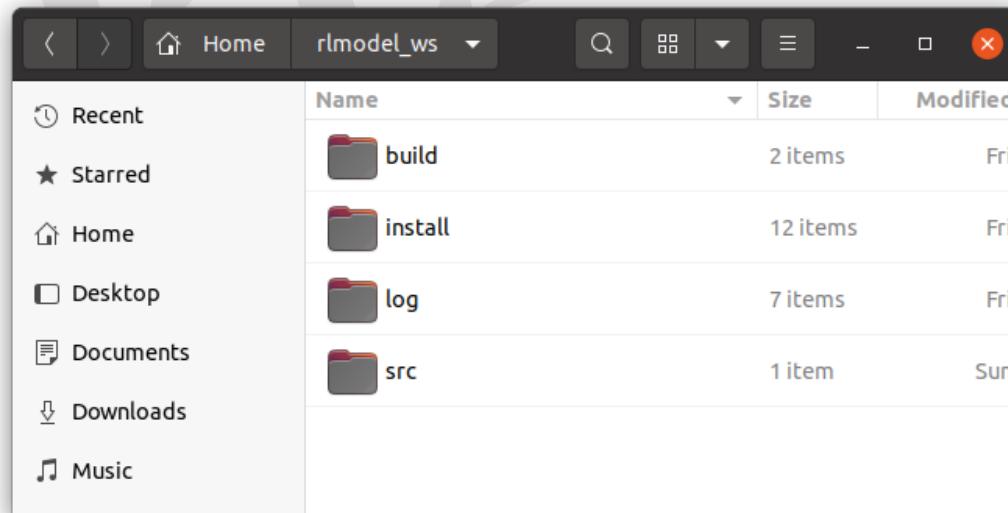
URDF 예제



ROS2 실습

workspace 생성

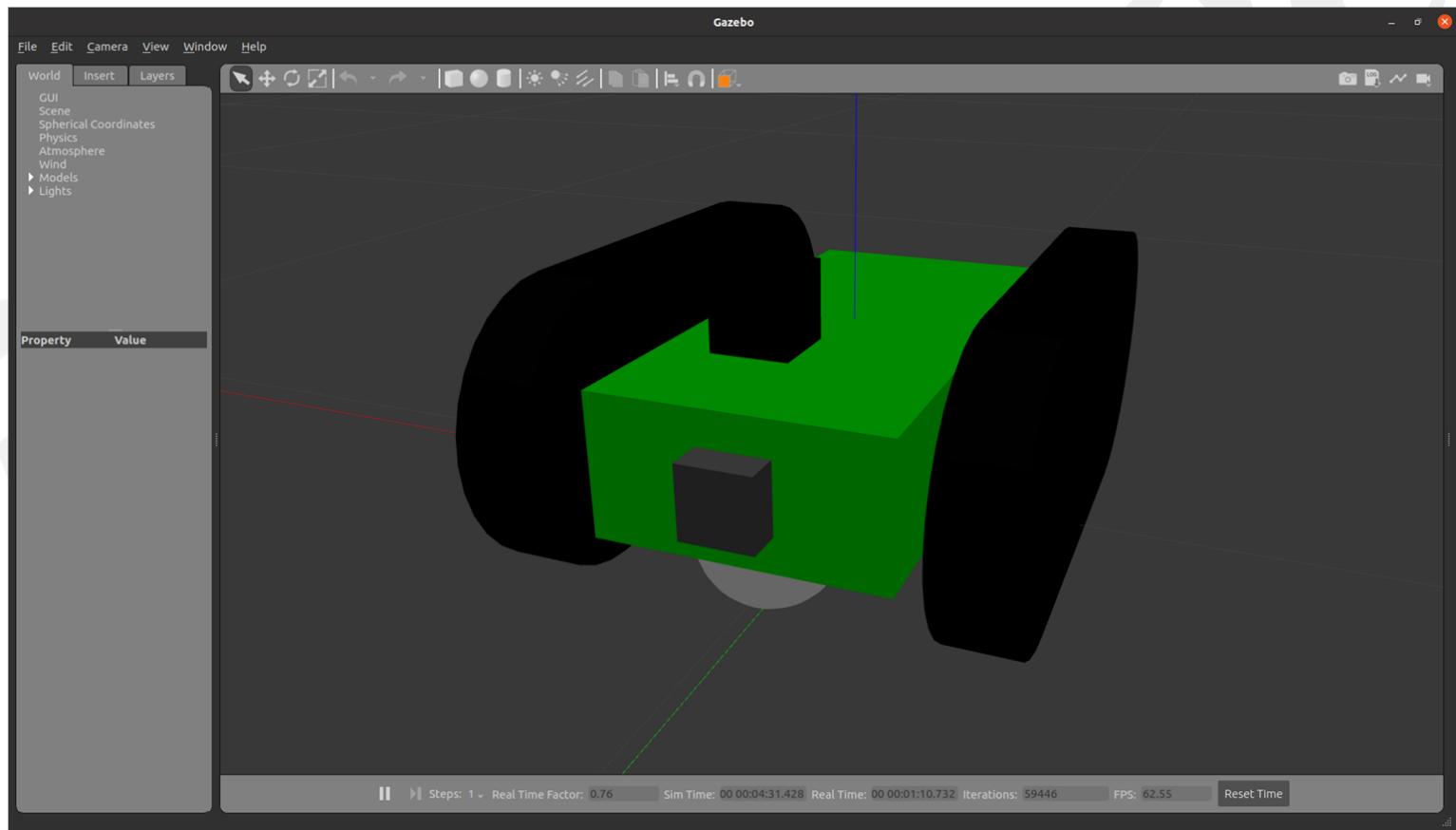
- \$ mkdir -p ~/ros2_ws/src
- \$ sudo apt install python3-rosdep
- \$ sudo rosdep init
- \$ rosdep update
- \$ rosdep install -i --from-path src --rosdistro foxy -y
- \$ colcon build
- \$. install/local_setup.bash





SKI_bot

gazebo maze world example

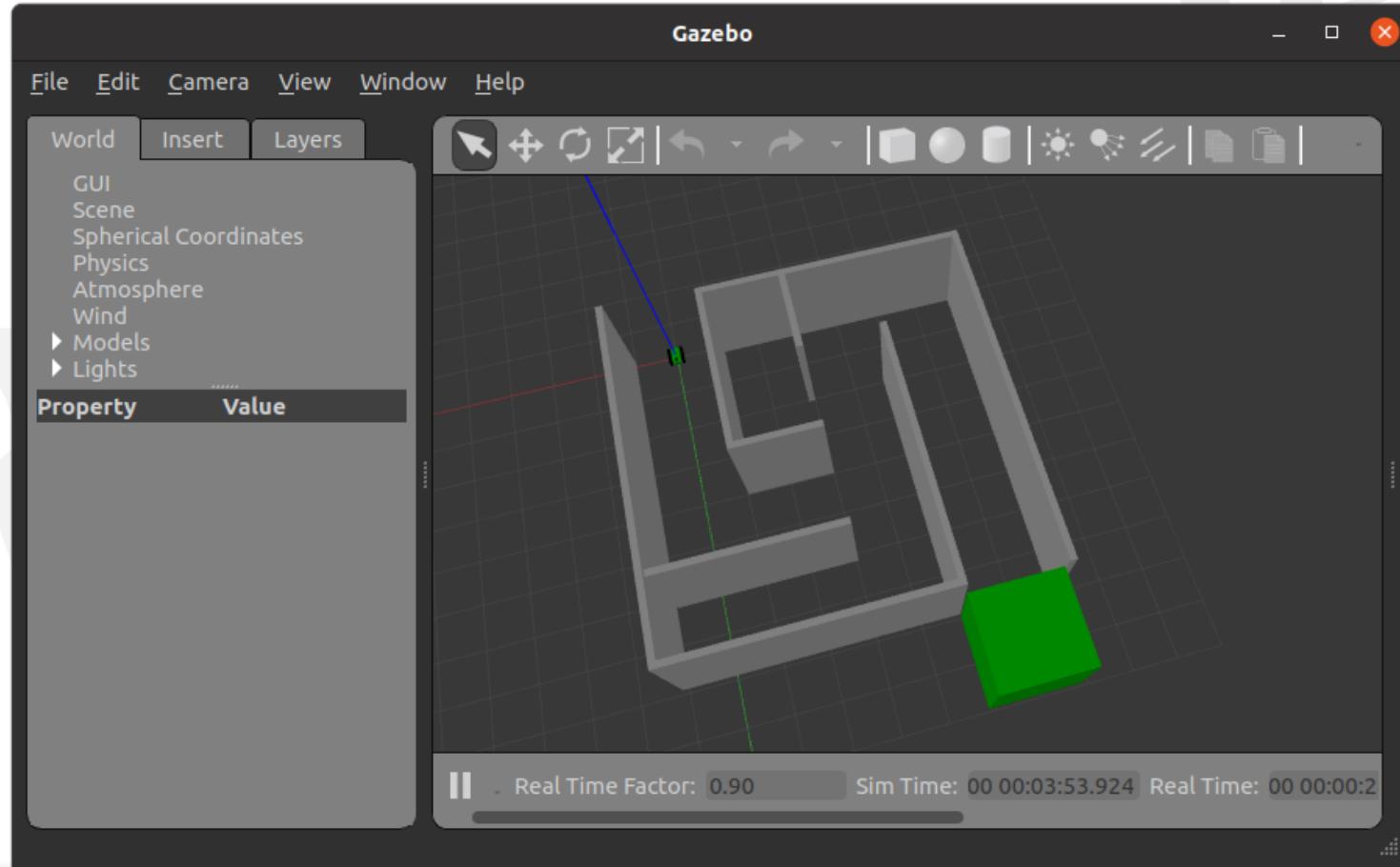


Clone

ski_bot

github: https://github.com/RLmodel/gcamp_gazebo

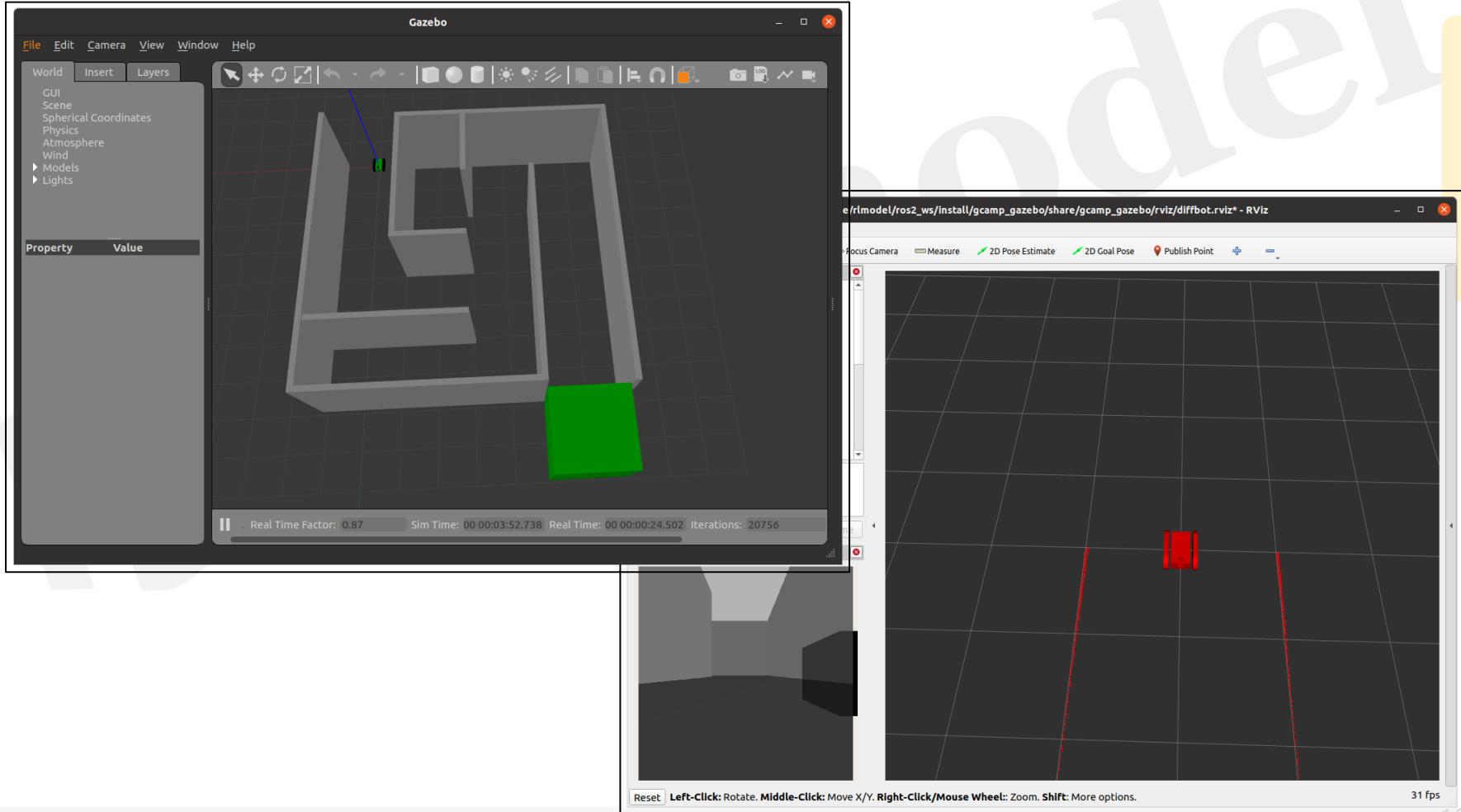
World: maze_world



Gazebo example

UGV robot - skidbot run

```
$ ros2 launch gcamp_gazebo maze_world.launch.py
```

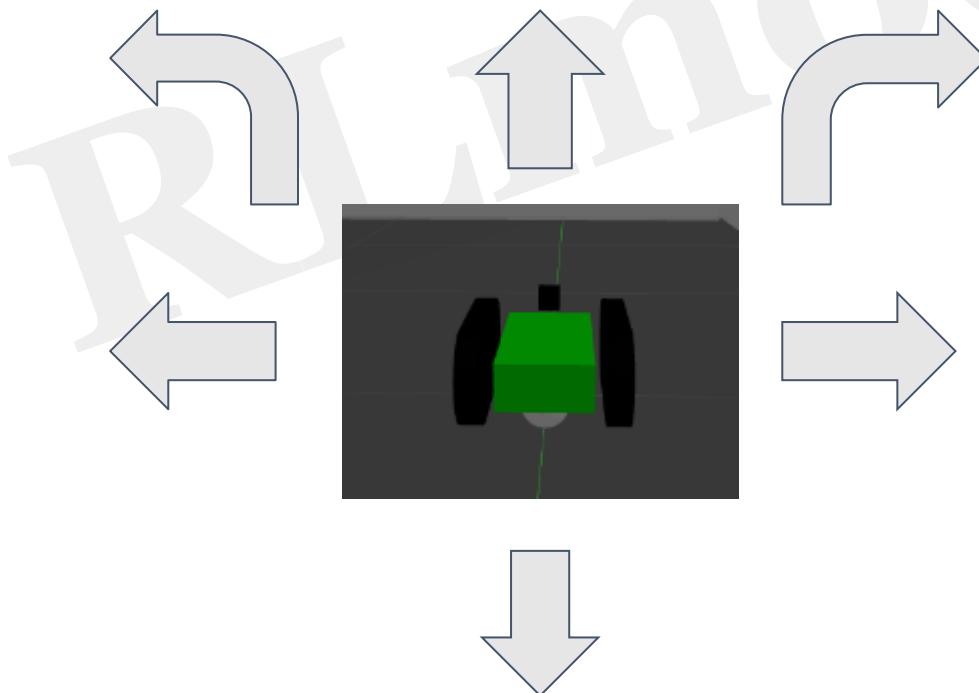


Gazebo example

UGV robot - skidbot control

exercise 1: move robot, check topic, view lidar data

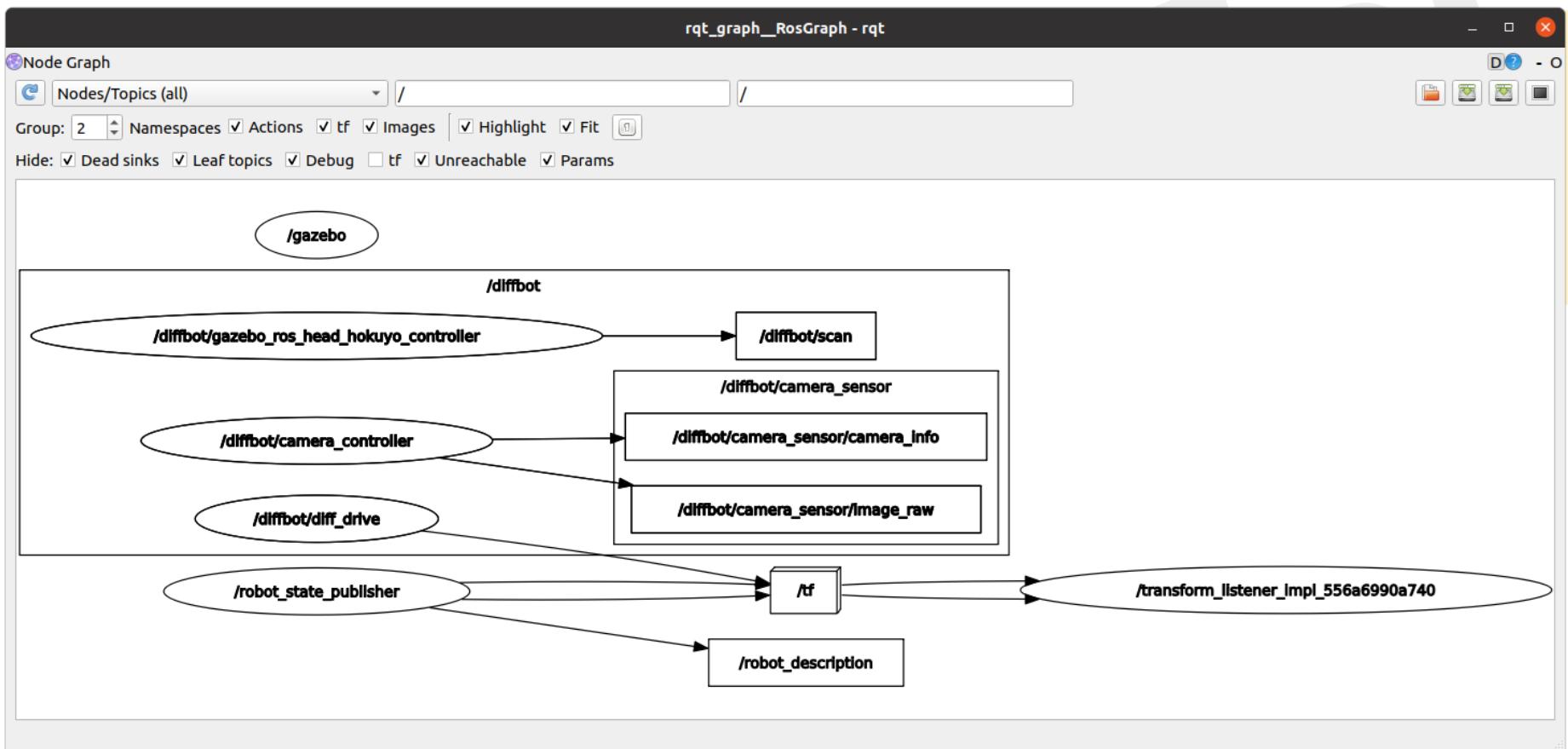
exercise 2: escape maze, replay using bag file



Gazebo example

UGV robot - skidbot control

view rqt_graph



Gazebo example

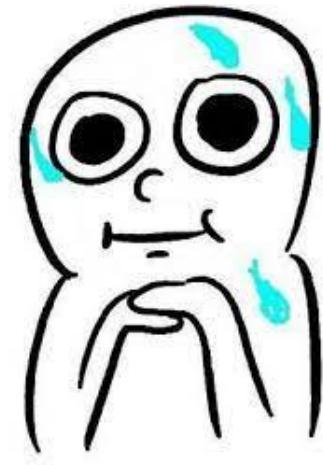
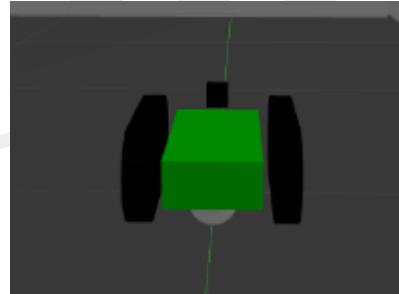
UGV robot - skidbot control

run teleop node

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

```
> rqt
```

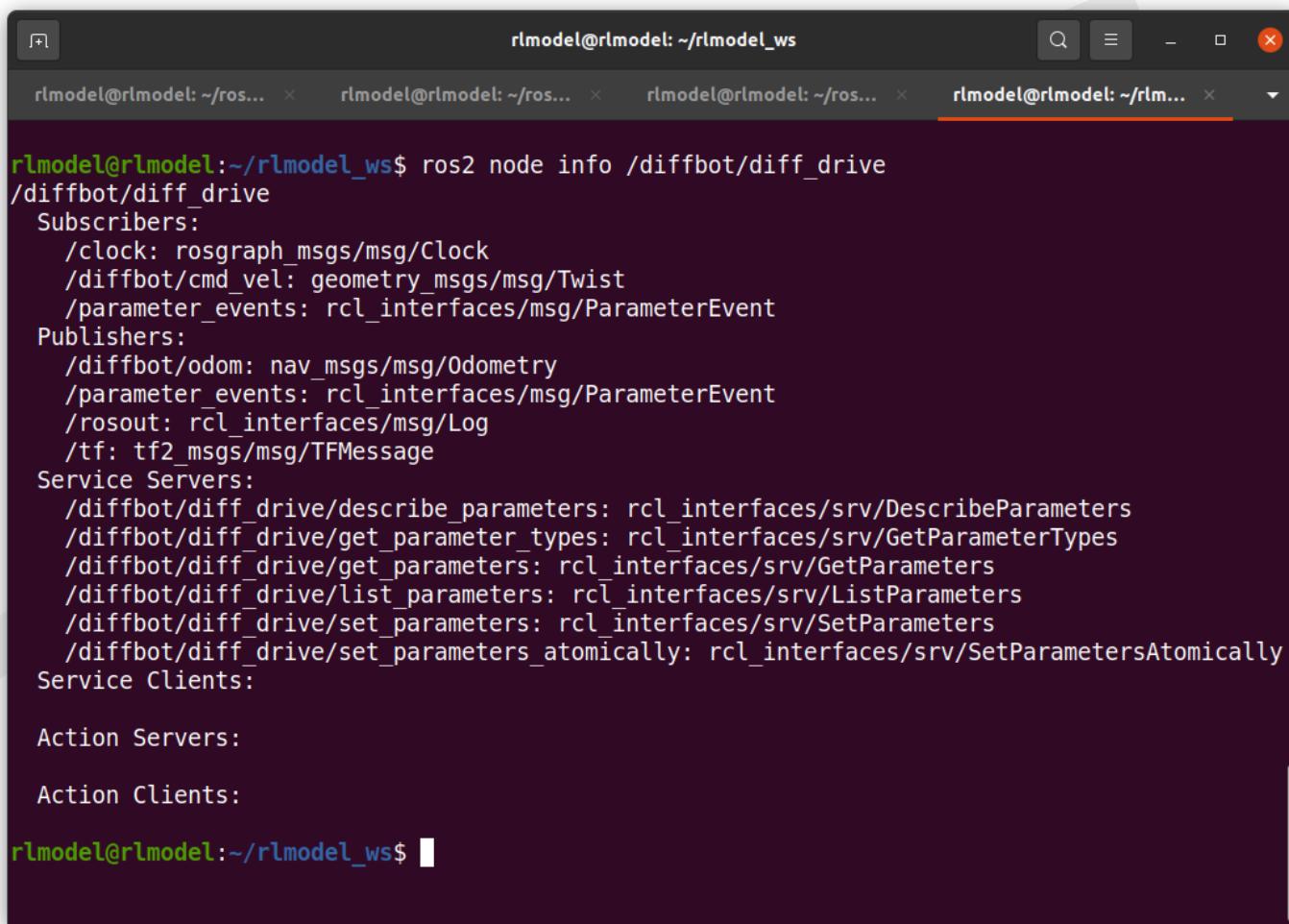
```
> rqt_robot_steering
```



Gazebo example

UGV robot - skidbot control

check node info



```
rlmodel@rlmodel:~/rlmodel_ws$ ros2 node info /diffbot/diff_drive
/diffbot/diff_drive
Subscribers:
/clock: rostopic msgs/msg/Clock
/diffbot/cmd_vel: geometry_msgs/msg/Twist
/parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
/diffbot/odom: nav_msgs/msg/Odometry
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/tf: tf2_msgs/msg/TFMessage
Service Servers:
/diffbot/diff_drive/describe_parameters: rcl_interfaces/srv/DescribeParameters
/diffbot/diff_drive/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/diffbot/diff_drive/get_parameters: rcl_interfaces/srv/GetParameters
/diffbot/diff_drive/list_parameters: rcl_interfaces/srv/ListParameters
/diffbot/diff_drive/set_parameters: rcl_interfaces/srv/SetParameters
/diffbot/diff_drive/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
Action Clients:

rlmodel@rlmodel:~/rlmodel_ws$
```

Gazebo example

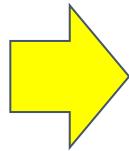
UGV robot - skidbot control

run teleop

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

```
$ ros2 topic list
```

```
rlmodel@rlmodel:~/rlmodel_ws$ ros2 topic list
/clicked_point
/clock
/cmd_vel
/diffbot/camera_sensor/camera_info
/diffbot/camera_sensor/image_raw
/diffbot/cmd_vel
/diffbot/odom
/diffbot/scan
/goal_pose
/initialpose
/joint_states
/parameter_events
/robot_description
/rosout
/tf
/tf_static
rlmodel@rlmodel:~/rlmodel_ws$
```

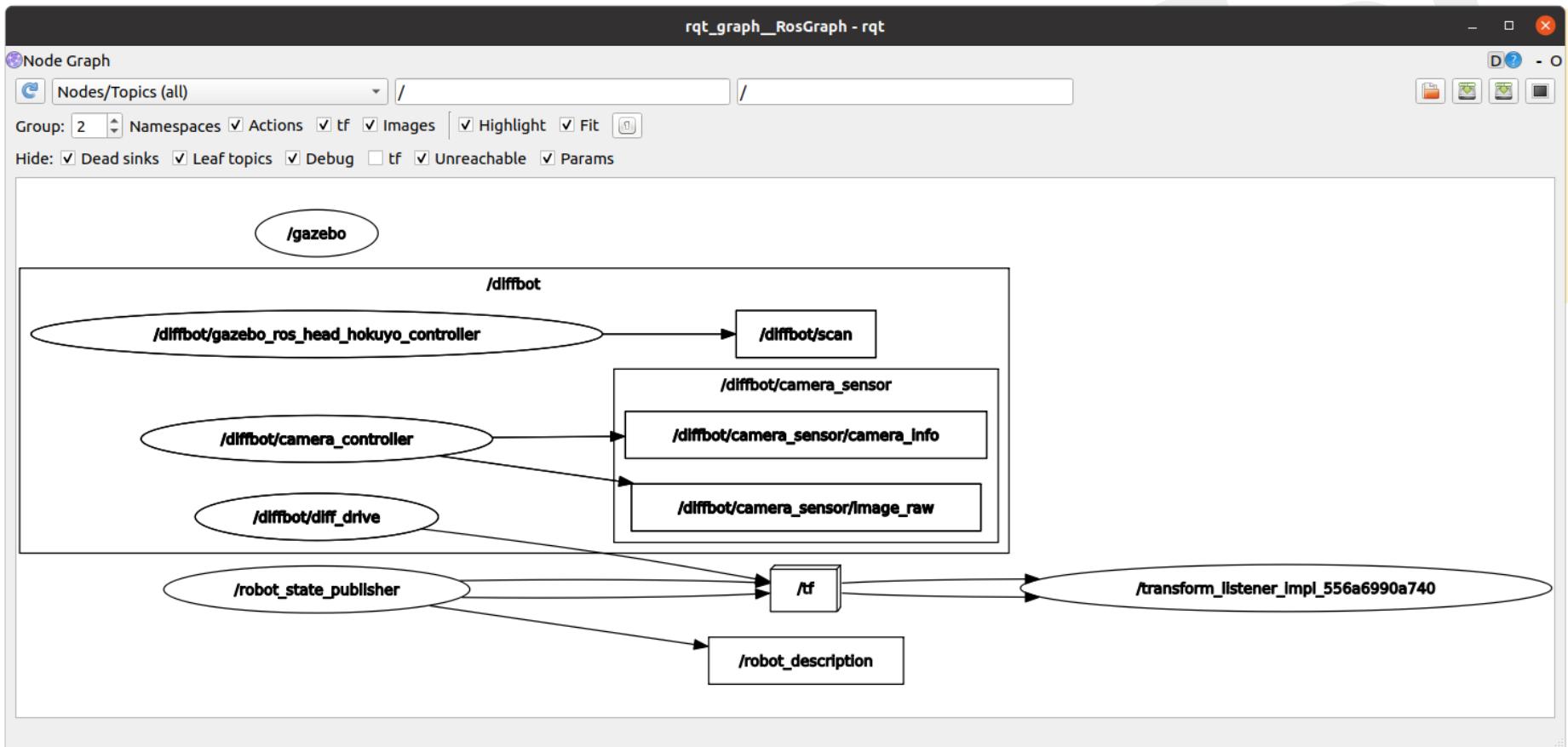


```
rlmodel@rlmodel:~/rlmodel_ws$ ros2 topic list
/clicked_point
/clock
/diffbot/camera_sensor/camera_info
/diffbot/camera_sensor/image_raw
/diffbot/cmd_vel
/diffbot/odom
/diffbot/scan
/goal_pose
/initialpose
/joint_states
/parameter_events
/robot_description
/rosout
/tf
/tf_static
rlmodel@rlmodel:~/rlmodel_ws$
```

Gazebo example

UGV robot - skidbot control

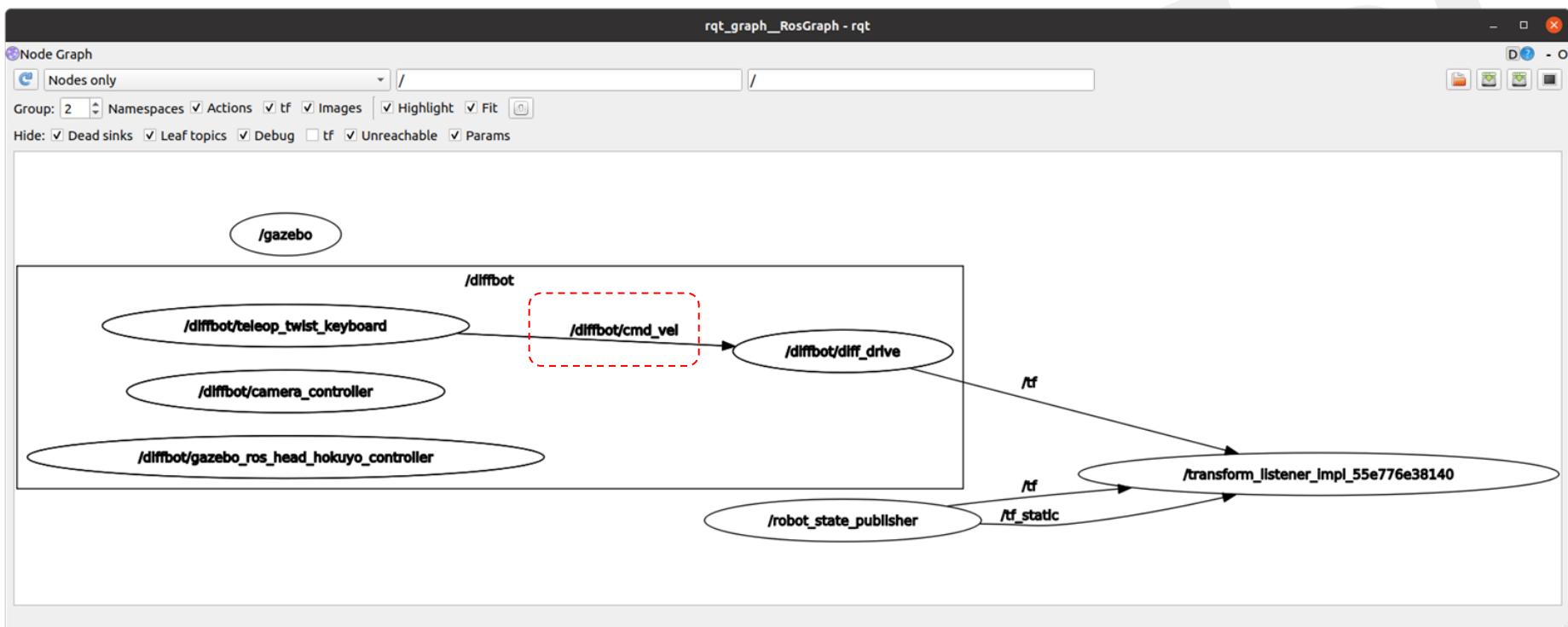
run -> /diffbot/cmd_vel



Gazebo example

UGV robot - skidbot control

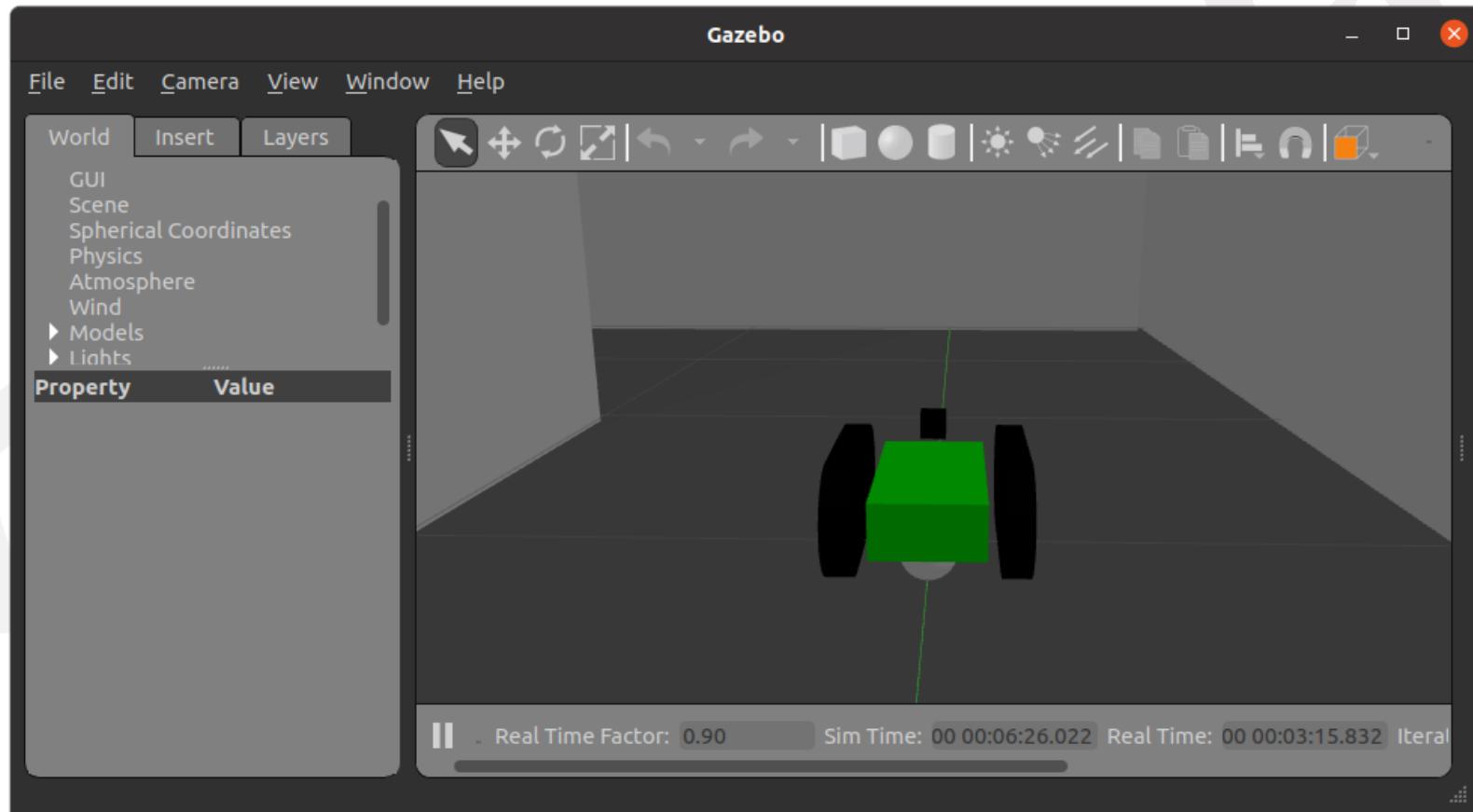
run -> /diffbot/cmd_vel



Gazebo example

UGV robot - skidbot run

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -r  
__ns:=/diffbot
```



Gazebo example

UGV robot - skid bot control

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -r  
_ns:=/skidbot
```

```
rlmodel@rlmodel: ~/ros2_ws$ ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -r _ns:=/skidbot

This node takes keypresses from the keyboard and publishes them as Twist messages. It works best with a US keyboard layout.

Moving around:
  u    i    o
  j    k    l
  m    ,    .

For Holonomic mode (strafing), hold down the shift key:
  U    I    O
  J    K    L
  M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

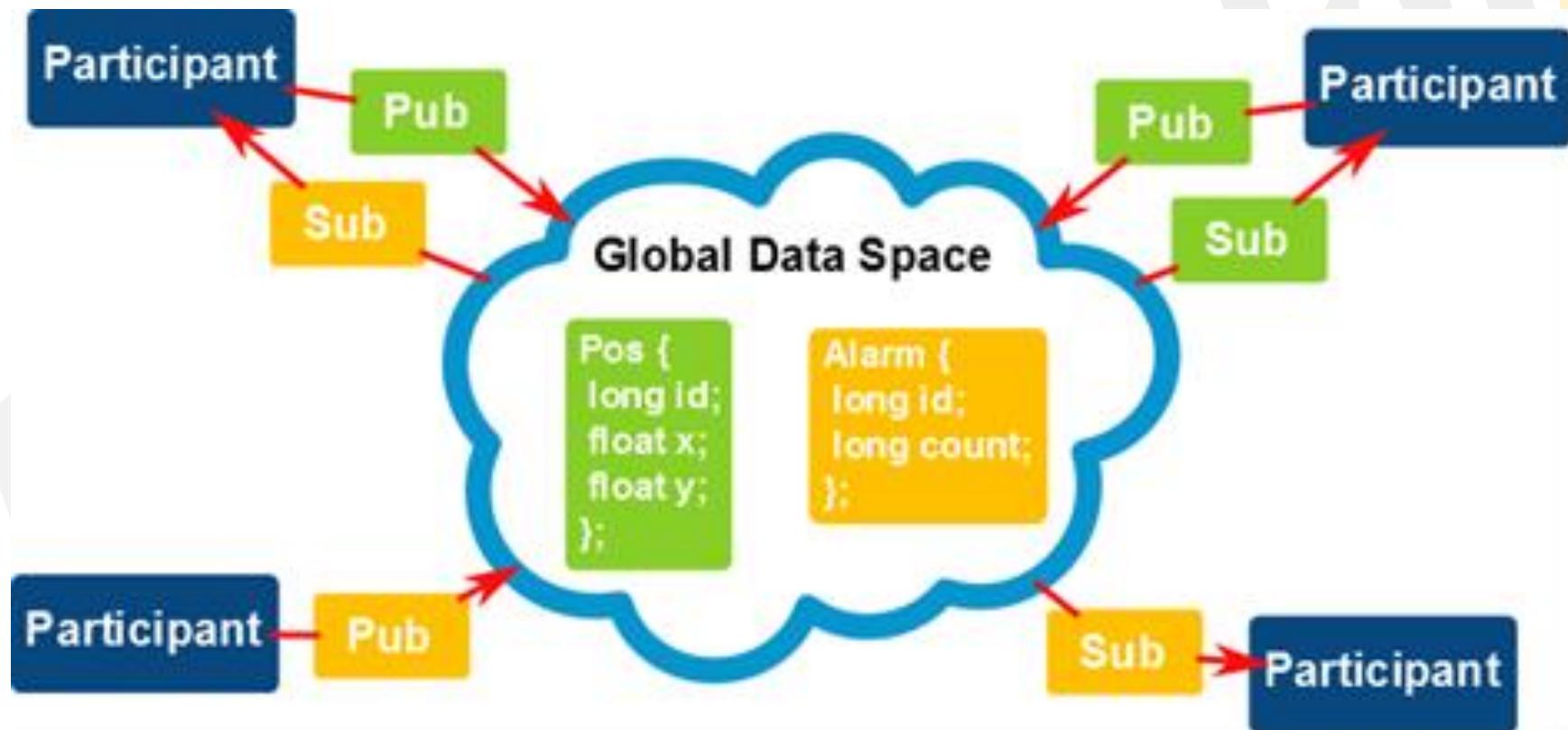
Moving around:

u	i	o
j	k	l
m	,	.

DDS config

Domain ID

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -r  
__ns:=/skidbot
```



Gazebo

Clearpath Husky

Link:

<https://clearpathrobotics.com/tutorials/husky/simulating-husky.html>

The screenshot shows a dark-themed website for Clearpath Robotics. On the left, a sidebar menu lists categories: HUSKY OVERVIEW, HUSKY PACKAGES, SETTING UP HUSKY, USING HUSKY, and Simulating Husky (which is highlighted with a yellow bar). The main content area has a header "SIMULATING HUSKY". Below it, a paragraph explains the purpose of the simulator. A section titled "LAUNCH GAZEBO" contains a command-line instruction: "ros2 launch husky_gazebo husky_playpen.launch.py". At the bottom, there's a screenshot of the Gazebo simulation interface showing a Husky robot in a 3D environment with obstacles.

Husky UGV Tutorials

CLEARPATH
ROBOTICS™

1.0.0

Search

HUSKY OVERVIEW

- Introduction

HUSKY PACKAGES

- Husky Common Packages

SETTING UP HUSKY

- Upgrading ROS1 to ROS2
- Installing Husky Software
- Setting Up Husky's Network
- Joystick Controller Pairing

USING HUSKY

- Driving Husky

Simulating Husky

- Launch Gazebo
- Launch rviz

[Home](#) » Simulating Husky

SIMULATING HUSKY

Whether you actually have a physical Husky robot or not, the Husky simulator is a great way to get started with ROS2 robot development. In this tutorial, we will go through the basics of starting Gazebo and Rviz and how to drive your Husky around.

LAUNCH GAZEBO

```
ros2 launch husky_gazebo husky_playpen.launch.py
```

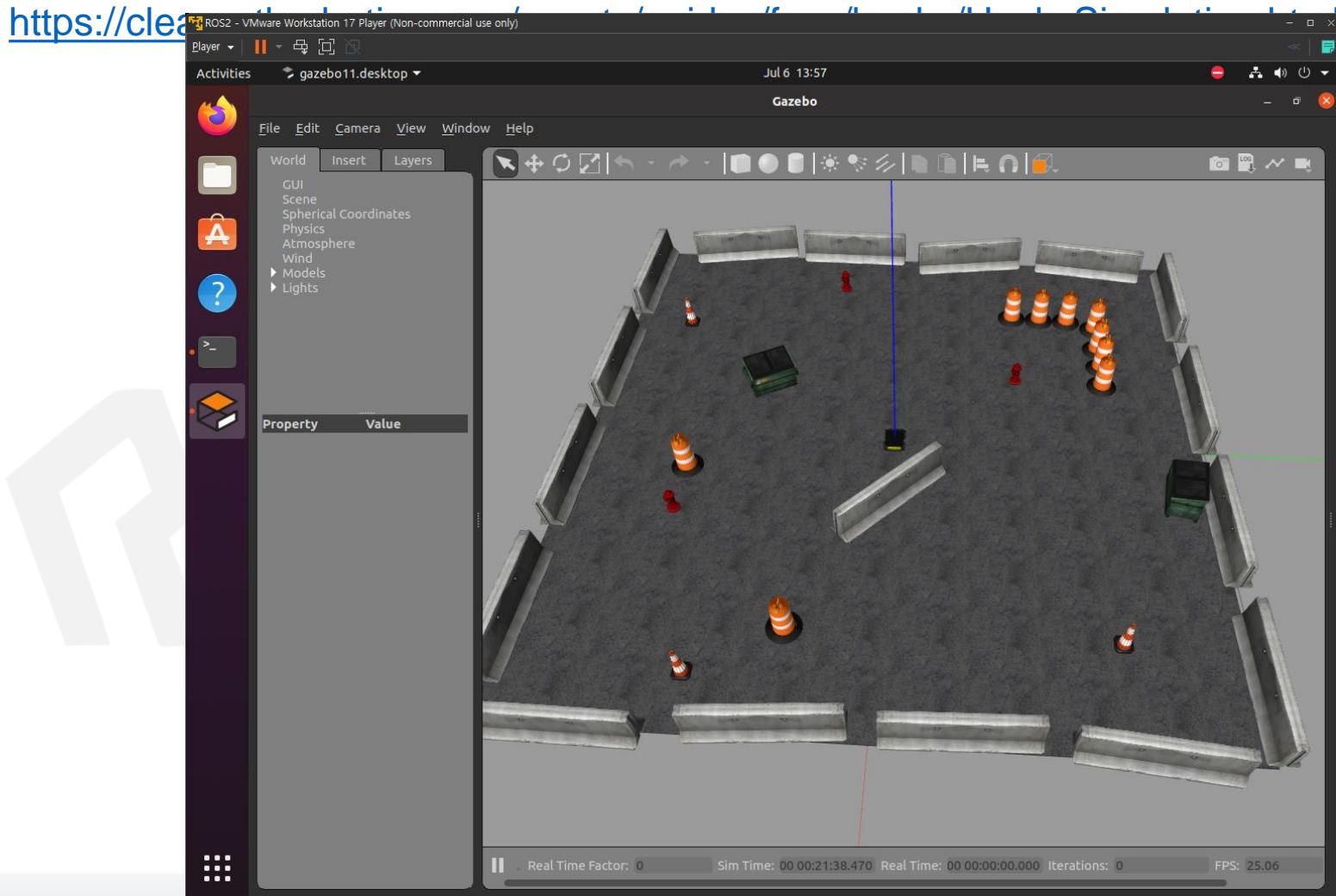
You should see the following window appear, or something like it. You can adjust the camera angle by clicking and dragging while holding CTRL, ALT, or the shift key:

The Gazebo simulation window displays a 3D environment for a Husky robot. The robot is a red and black model positioned in the center of a circular track. The track is bounded by concrete barriers and several orange traffic cones. There are also some green shipping containers scattered around the track. The Gazebo interface includes a toolbar at the top and a properties panel on the left side.

Gazebo

Clearpath Husky

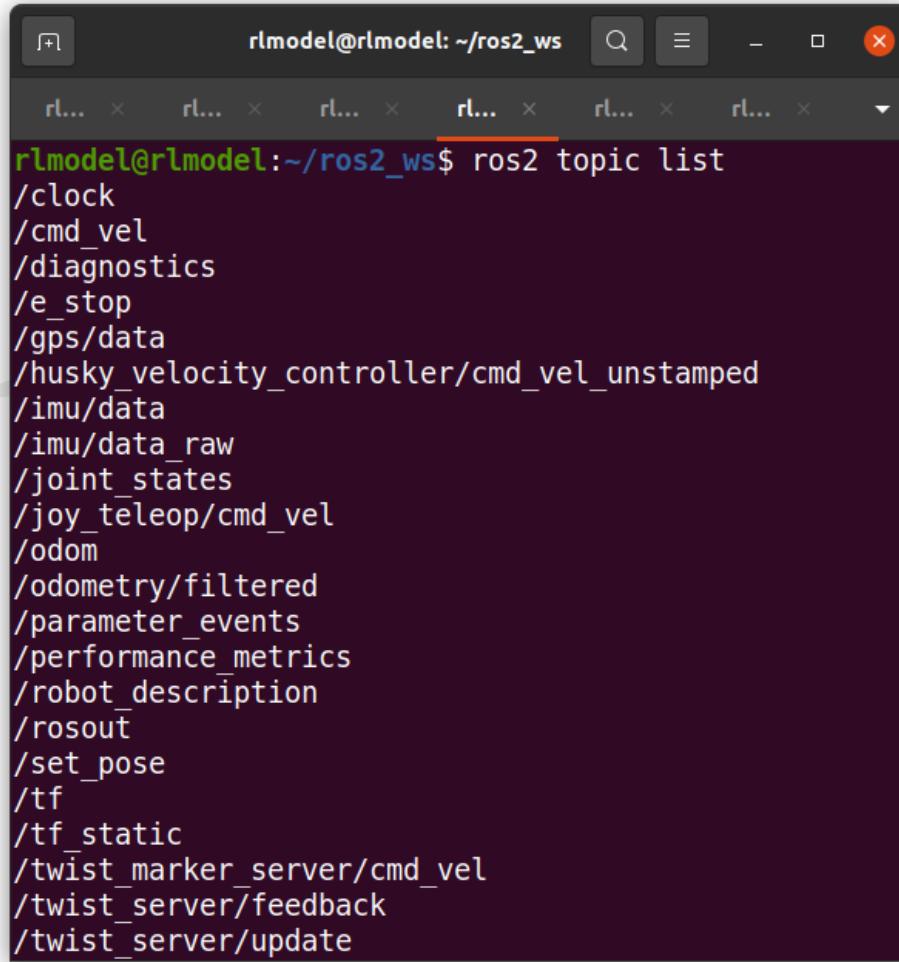
Link:



Gazebo

Clearpath Husky

check topics: IMU, GPS, etc



A terminal window titled "rlmodel@rlmodel: ~/ros2_ws" displays the output of the command "ros2 topic list". The list includes various ROS 2 topics such as /clock, /cmd_vel, /diagnostics, /e_stop, /gps/data, /husky_velocity_controller/cmd_vel_unstamped, /imu/data, /imu/data_raw, /joint_states, /joy_teleop/cmd_vel, /odom, /odometry/filtered, /parameter_events, /performance_metrics, /robot_description, /rosout, /set_pose, /tf, /tf_static, /twist_marker_server/cmd_vel, /twist_server/feedback, and /twist_server/update.

```
rlmodel@rlmodel:~/ros2_ws$ ros2 topic list
/clock
/cmd_vel
/diagnostics
/e_stop
/gps/data
/husky_velocity_controller/cmd_vel_unstamped
 imu/data
 imu/data_raw
/joint_states
/joy_teleop/cmd_vel
/odom
/odometry/filtered
/parameter_events
/performance_metrics
/robot_description
/rosout
/set_pose
/tf
/tf_static
/twist_marker_server/cmd_vel
/twist_server/feedback
/twist_server/update
```

Clearpath Husky

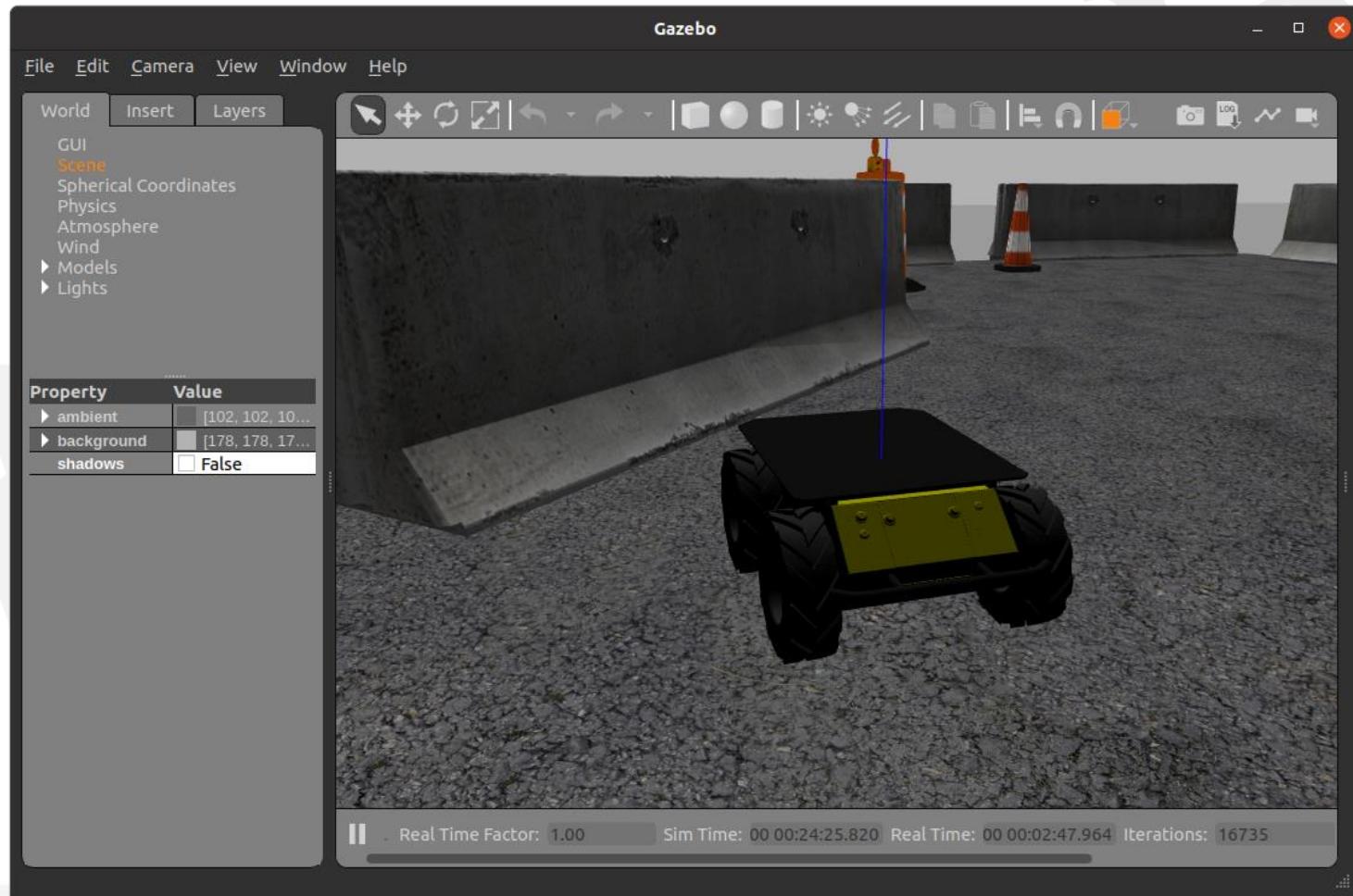
rqt_graph: IMU, GPS, ekf, .. etc



Gazebo

Clearpath Husky

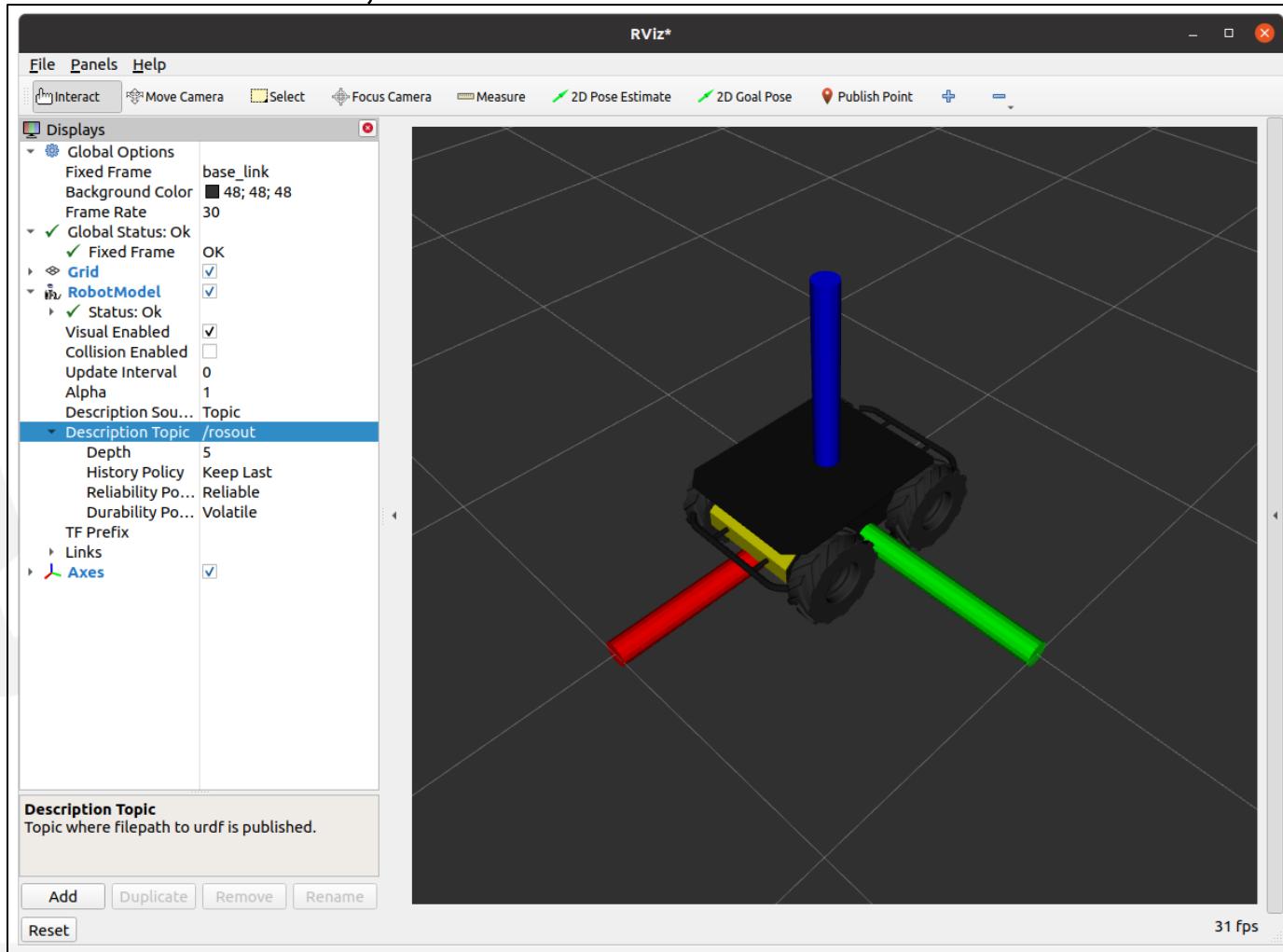
```
cmd: ros2 topic pub --once /husky_velocity_controller/cmd_vel_unstamped  
geometry_msgs/msg/Twist '{linear: {x: -0.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}'
```



Gazebo

Clearpath Husky

rviz: Robot model , tf Axis



Gazebo

Clearpath Husky

Link: <https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/>

EXTERNAL DIMENSIONS

990 x 670 x 390 mm
(39 x 26.4 x 14.6 in)

INTERNAL DIMENSIONS

296 x 411 x 155 mm
(11.7 x 16.2 x 6.1 in)

WEIGHT

50 kg
(110 lbs)

MAX PAYLOAD

75 kg
(165 lbs)



Gazebo

Clearpath Husky

GPS Waypoint navigation

Link 1: https://www.youtube.com/watch?v=1Ys1_akUD-s



Gazebo

Clearpath Husky

GPS Waypoint navigation

Link 2: <https://www.youtube.com/watch?v=hP6NiGzOLml>





프로그램 실습

ROS2 Jupyter notebook

zmk5/jupyter-ros2

Jupyter widget helpers for ros2, the Next-Generation of the Robot Operating System

0 Contributors 7 Issues 18 Stars 5 Forks

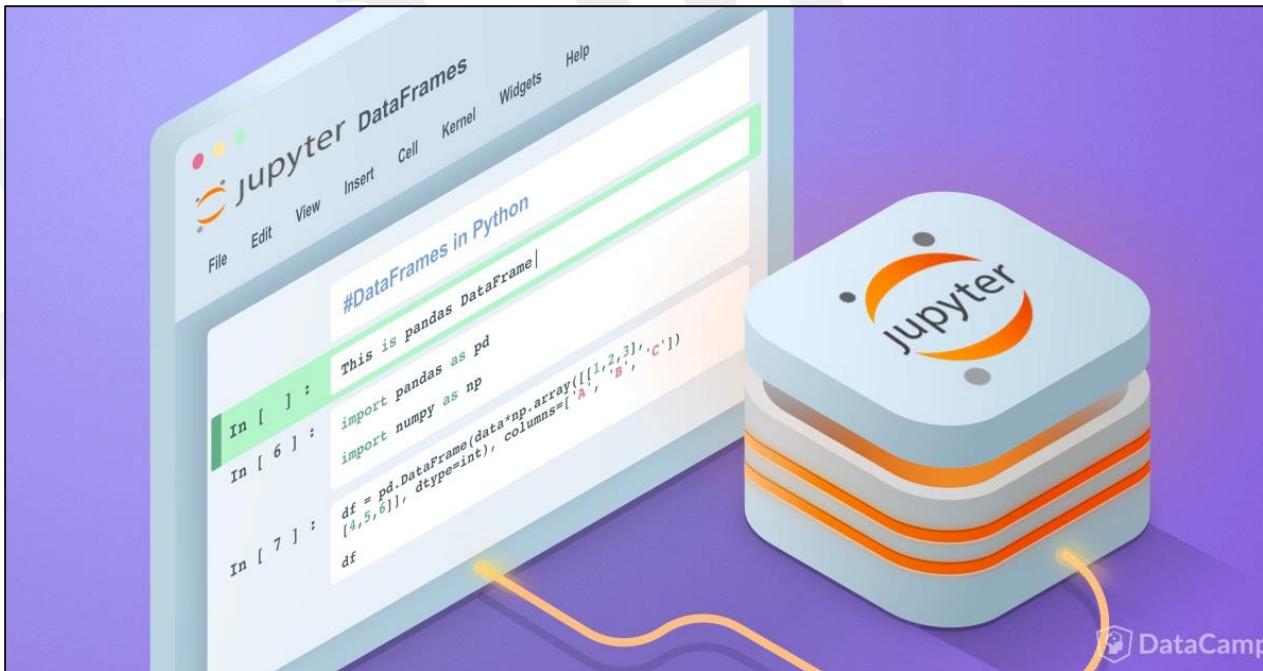


ROS2 Python 실습

Jupyter notebook

설치:

```
>sudo apt install python3-pip  
>pip3 install --upgrade pip  
>pip3 install jupyter ipywidewegets pyyaml bqplot  
>sudo apt install jupyter-core
```



ROS2 Python 실습

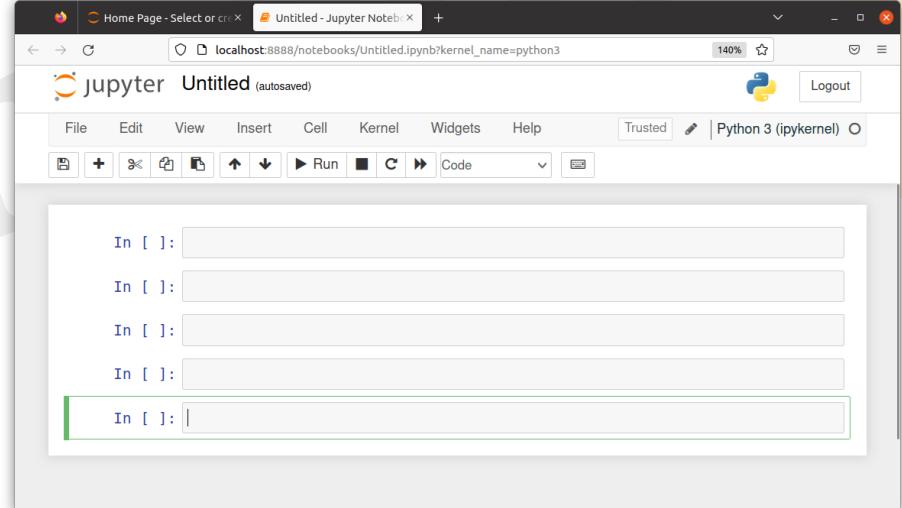
Jupyter notebook 실행

>jupyter notebook

>mark down 언어지원

```
source /home/byb76/.ros_menu/plugins_bashrc/dds_bashrc 1
Source Eclipse Cyclone DDS successfully!
(ROS 2 foxy) byb76@rlhp:~/ros2test/src/ROS2_edu/my_turtlesim_pkg/Jupyter$ jupyter notebook
[I 11:24:09.403 NotebookApp] Serving notebooks from local directory: /home/byb76/ros2test/src/ROS2_edu/my_turtlesim_pkg/Jupyter
[I 11:24:09.404 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 11:24:09.404 NotebookApp] http://localhost:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
[I 11:24:09.404 NotebookApp] or http://127.0.0.1:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
[I 11:24:09.404 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip configuration).
[C 11:24:09.472 NotebookApp]

To access the notebook, open this file in a browser:
  file:///home/byb76/.local/share/jupyter/runtime/nbserver-19369-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
  or http://127.0.0.1:8888/?token=c426d4d4d0d158cec7a262949d847ac49a449fc31cdf404c
[I 11:24:32.422 NotebookApp] Creating new notebook in
[I 11:24:34.649 NotebookApp] Kernel started: 5c962139-1b34-4037-a826-51e90c0253f9, name: python3
[I 11:26:34.570 NotebookApp] Saving file at /Untitled.ipynb
/home/byb76/.local/lib/python3.8/site-packages/nbformat/_int_.py:129: MissingIDFieldWarning: Code cell is missing an id field, this will become a hard error in future nbformat versions. You may want to use 'normalize()' on your notebooks before validations (available since nbformat 5.1.4). Previous versions of nbformat are fixing this issue transparently, and will stop doing so in the future.
  validate(nb)
/home/byb76/.local/lib/python3.8/site-packages/notebook/services/contents/manager.py:353: MissingIDFieldWarning: Code cell is missing an id field, this will become a hard error in future nbformat versions. You may want to use 'normalize()' on your notebooks before validations (available since nbformat 5.1.4). Previous version
```



ROS2 Python 실습

mark down 테스트

>기본문법 예시

>See Appendix

```
# title
## *title2
### **title3
* item1
* item2
1. test1
2. test2
> comment
...
code block
...
```

title
***title2**
****title3**

- item1
- item2

1. test1
2. test2

comment "" code block ""

ROS2 Python 실습

topic 수신

준비: \$Source /opt/ros/foxy/setup.bash

순서: python code 작성 > 노드 생성

확인: \$ros2 node list

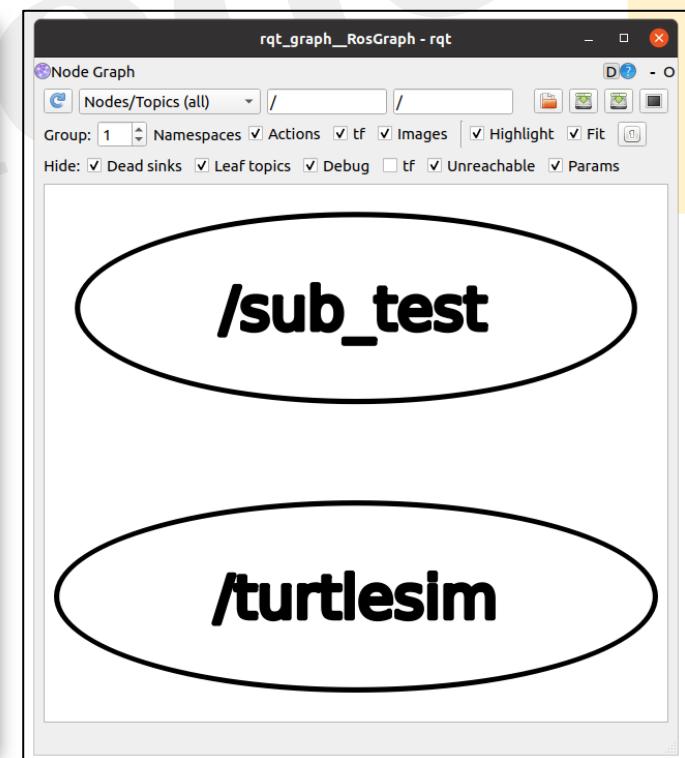
The screenshot shows a Jupyter Notebook interface with the title "jupyter topic_sub_test". The toolbar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Below the toolbar is a toolbar with icons for file operations like new, open, save, and run. The notebook has two code cells:

```
In [4]: import rclpy as rp
from turtlesim.msg import Pose
```

```
In [5]: rp.init()
test_node = rp.create_node('sub_test')
```

Cell In [5] has a red background and displays the output of the code execution:

```
1674529369.490302 [0]    python3: using network interface wlp2s0 (udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
```



ROS2 Python 실습

topic 코드

> 토픽 수신 (subscription)

rp.spin_once(test_node)

The screenshot shows a Jupyter Notebook interface with the title "topic_sub_test". The notebook contains the following code:

```
In [4]: import rclpy as rp
from turtlesim.msg import Pose

In [5]: rp.init()
test_node = rp.create_node('sub_test')
1674529369.490302 [0]    python3: using network interface wlp2s0 (udp/19
2.168.0.19) selected arbitrarily from: wlp2s0, docker0

In [6]: def callback(data):
    print("turtle pose",data)
    print("X: ", data.x)
    print("Y: ", data.y)
    print("Heading: ",data.theta)

In [7]: test_node.create_subscription(Pose, '/turtle1/pose', callback, 10)
Out[7]: <rclpy.subscription.Subscription at 0x7f5c6c4fdac0>

In [8]: rp.spin_once(test_node)
turtle pose turtlesim.msg.Pose(x=4.840552806854248, y=5.830162525177002,
theta=-0.7792094945907593, linear_velocity=0.0, angular_velocity=0.0)
X:  4.840552806854248
Y:  5.830162525177002
Heading: -0.7792094945907593
```

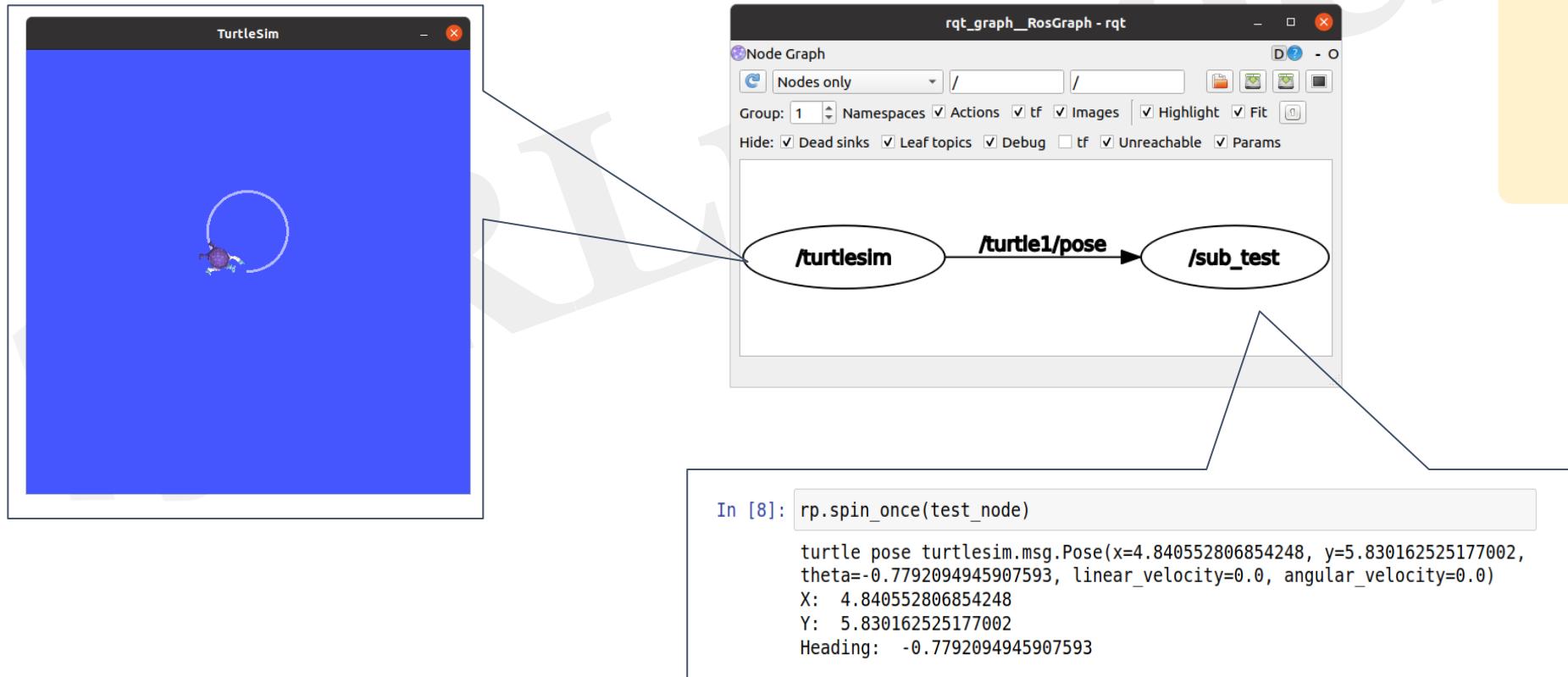
수신된 토픽 내용
확인

ROS2 Python 실습

topic 코드

> 토픽 구독

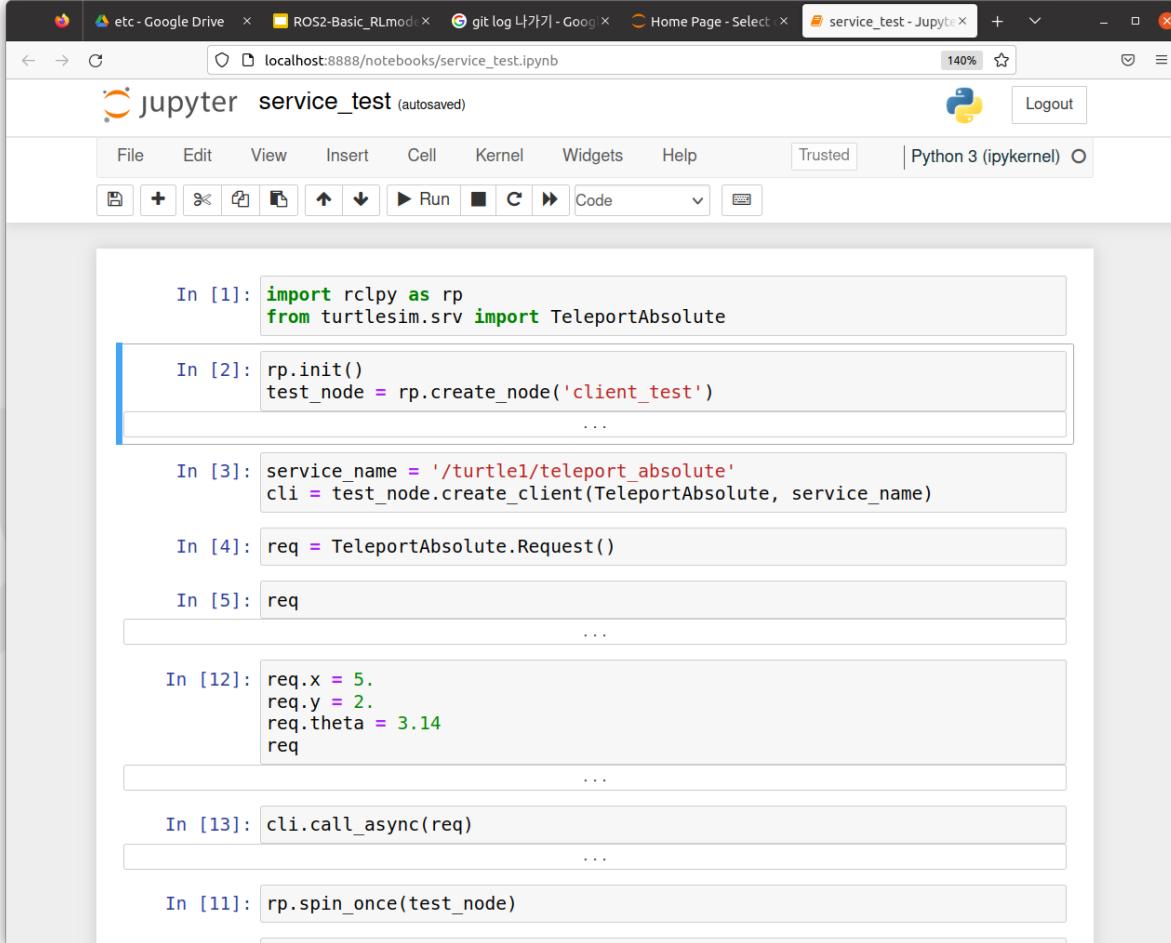
```
rp.spin_once(test_node)
```



ROS2 Python 실습

service 코드

순서: node 생성 > client 생성 > client call >



The screenshot shows a Jupyter Notebook interface with the title "service_test" (autosaved). The notebook contains the following code:

```
In [1]: import rclpy as rp
from turtlesim.srv import TeleportAbsolute

In [2]: rp.init()
test_node = rp.create_node('client_test')
...
In [3]: service_name = '/turtle1/teleport_absolute'
cli = test_node.create_client(TeleportAbsolute, service_name)

In [4]: req = TeleportAbsolute.Request()

In [5]: req
...
In [12]: req.x = 5.
req.y = 2.
req.theta = 3.14
req
...
In [13]: cli.call_async(req)
...
In [11]: rp.spin_once(test_node)
```



프로그램 실습

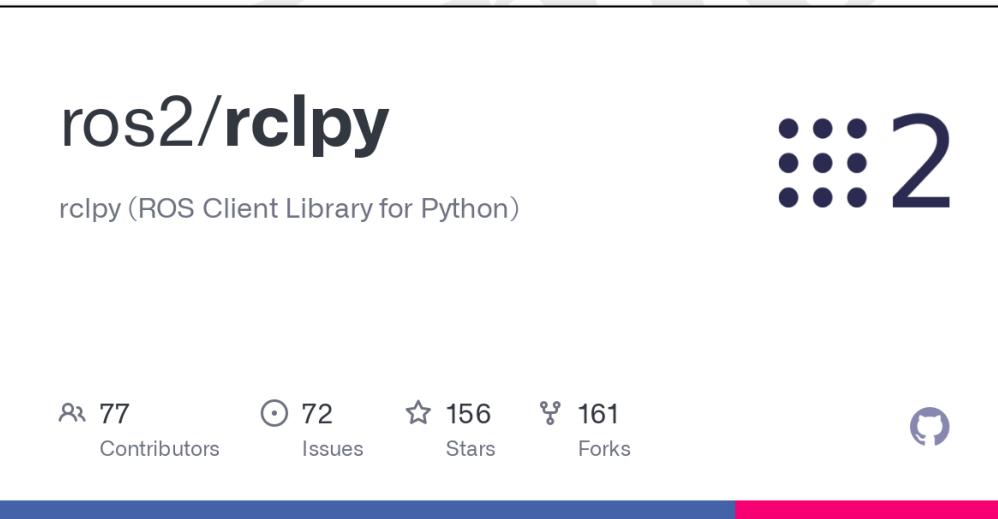
ROS2 Python

ros2/rclpy

rclpy (ROS Client Library for Python)

⋮ 2

77 Contributors 72 Issues 156 Stars 161 Forks



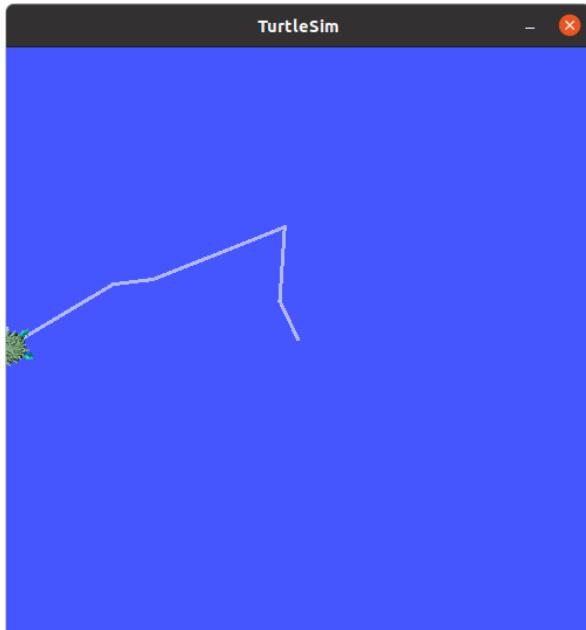
ROS2 실습

workspace 생성

생성: mkdir -p ~/ros2_ws/src

워크페이스 이름: ros2_ws

link: https://github.com/yunbum/ROS2_edu.git



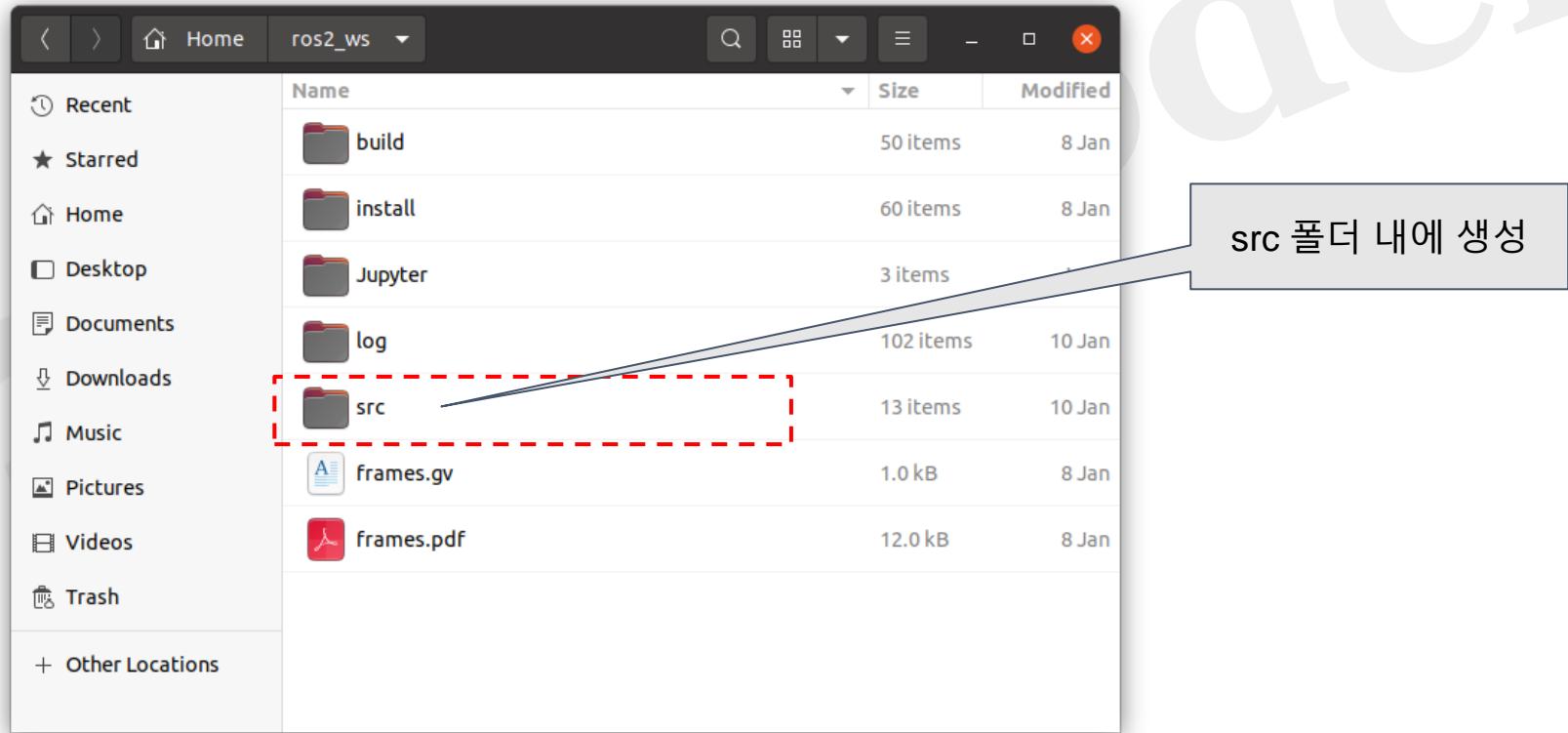
```
byb76@rlhp: ~/add_ros2_ws
```

```
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
```

ROS2 실습

Package 생성

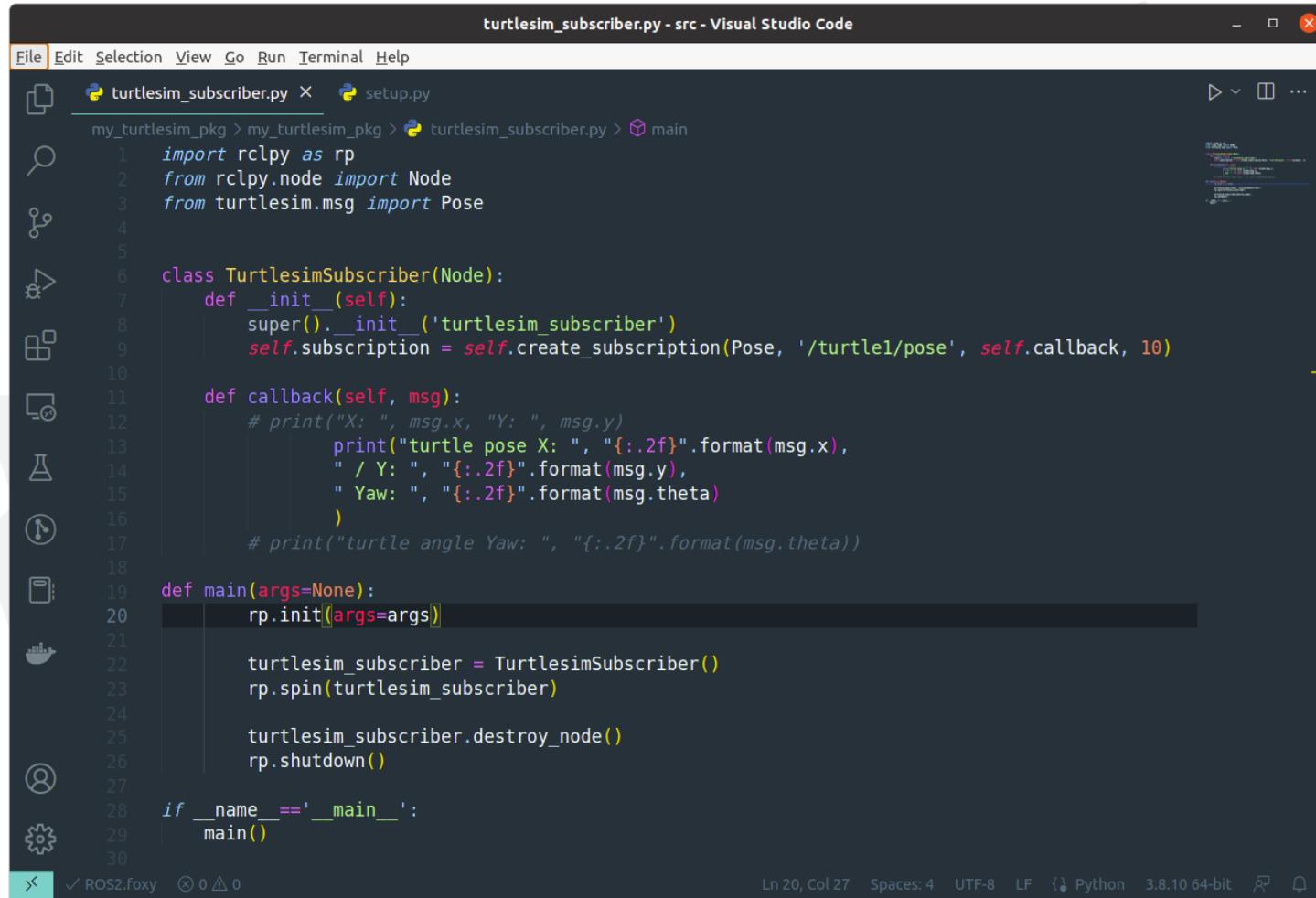
- cd ~/ros2_ws/src
- ros2 pkg create --build-type ament_python --node-name turtlesim_node my_turtlesim_pkg



ROS2 실습

Python code 생성

- turtlesim_subscriber.py



The screenshot shows a Visual Studio Code window with the title "turtlesim_subscriber.py - src - Visual Studio Code". The code editor displays Python code for a ROS2 node. The code defines a class `TurtlesimSubscriber` that inherits from `Node`. It creates a subscription to the `/turtle1/pose` topic and defines a callback function `callback` to handle incoming messages. The main function initializes the node, creates an instance of `TurtlesimSubscriber`, and starts the node. The code uses ROS2 message types from the `turtlesim.msg` package.

```
File Edit Selection View Go Run Terminal Help
turtlesim_subscriber.py × setup.py
my_turtlesim_pkg > my_turtlesim_pkg > turtlesim_subscriber.py > main
1 import rclpy as rp
2 from rclpy.node import Node
3 from turtlesim.msg import Pose
4
5
6 class TurtlesimSubscriber(Node):
7     def __init__(self):
8         super().__init__('turtlesim_subscriber')
9         self.subscription = self.create_subscription(Pose, '/turtle1/pose', self.callback, 10)
10
11     def callback(self, msg):
12         # print("X: ", msg.x, "Y: ", msg.y)
13         #     print("turtle pose X: ", "{:.2f}".format(msg.x),
14         #           " / Y: ", "{:.2f}".format(msg.y),
15         #           " Yaw: ", "{:.2f}".format(msg.theta)
16         #           )
17         #     print("turtle angle Yaw: ", "{:.2f}".format(msg.theta))
18
19     def main(args=None):
20         rp.init(args=args)
21
22         turtlesim_subscriber = TurtlesimSubscriber()
23         rp.spin(turtlesim_subscriber)
24
25         turtlesim_subscriber.destroy_node()
26         rp.shutdown()
27
28     if __name__ == '__main__':
29         main()
30
```

Ln 20, Col 27 Spaces: 4 UTF-8 LF { Python 3.8.10 64-bit



ROS2 실습

Python code 생성

- turtlesim_subscriber.py

Node class 정보

link: <https://docs.ros2.org/latest/api/rclpy/api/node.html>

rclpy

Navigation

About

Examples

API

- Initialization, Shutdown, and Spinning
- Node
- Topics
- Services
- Actions
- Timer
- Parameters
- Logging
- Context
- Execution and Callbacks
- Utilities
- Quality of Service

Node

```
class rclpy.node.Node(node_name, *, context=None, cli_args=None,  
                      namespace=None, use_global_arguments=True, enable_rosout=True,  
                      start_parameter_services=True, parameter_overrides=None,  
                      allow_undeclared_parameters=False,  
                      automatically_declare_parameters_from_overrides=False)
```

Create a Node.

- Parameters:**
- **node_name (str)** - A name to give to this node. Validated by `validate_node_name()`.
 - **context (Optional[Context])** - The context to be associated with, or None for the default global context.
 - **cli_args (Optional[List[str]])** - A list of strings of command line args to be used only by this node. These arguments are used to extract remappings used by the node and other ROS specific settings, as well as user defined non-ROS arguments.
 - **namespace (Optional[str])** - The namespace to which relative topic and service names will be prefixed. Validated by `validate_namespace()`.
 - **use_global_arguments (bool)** - False if the node should ignore process-wide command line args.



ROS2 실습

dependency 확인

- ```
cd if you're still in the ``src`` directory with the ``ros_tutorials`` clone
```
- cd ..
  - rosdep install -i --from-path src --rosdistro foxy -y

## colcon build 실행

- colcon build
- 

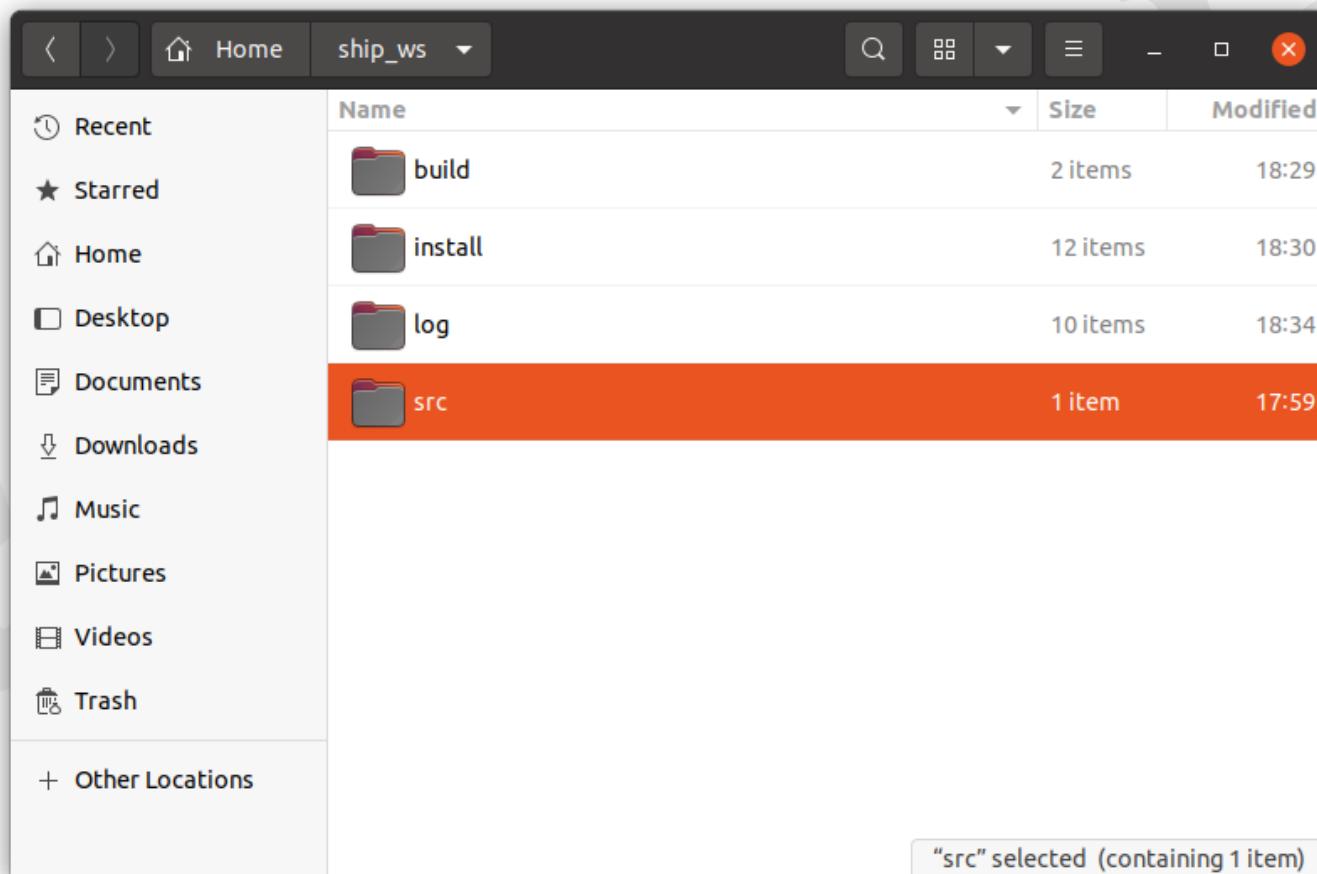
<결과>

- build, install, log 폴더 생성

# ROS2 실습

## build 결과 확인

- build, install, log 파일 생성





# ROS2 실습

## package 실행

- . install/localsetup.bash
- ros2 run my\_turtlesim\_pkg turtle\_subscriber

The screenshot shows a terminal window with the following content:

```
byb76@rlhp: ~/ros2test
(ROS 2 foxy) byb76@rlhp:~/ros2test$ ros2 run my_turtlesim_pkg turtlesim_node
Hi from my_turtlesim_pkg.
(ROS 2 foxy) byb76@rlhp:~/ros2test$
```

The terminal window has a dark background and light-colored text. The title bar reads "byb76@rlhp: ~/ros2test". The window includes standard Linux-style window controls (minimize, maximize, close) in the top right corner.

# ROS2 실습

# turtlesim topic subscriber 노드 생성

## 파이썬 파일 생성: turtlesim\_subscriber.py

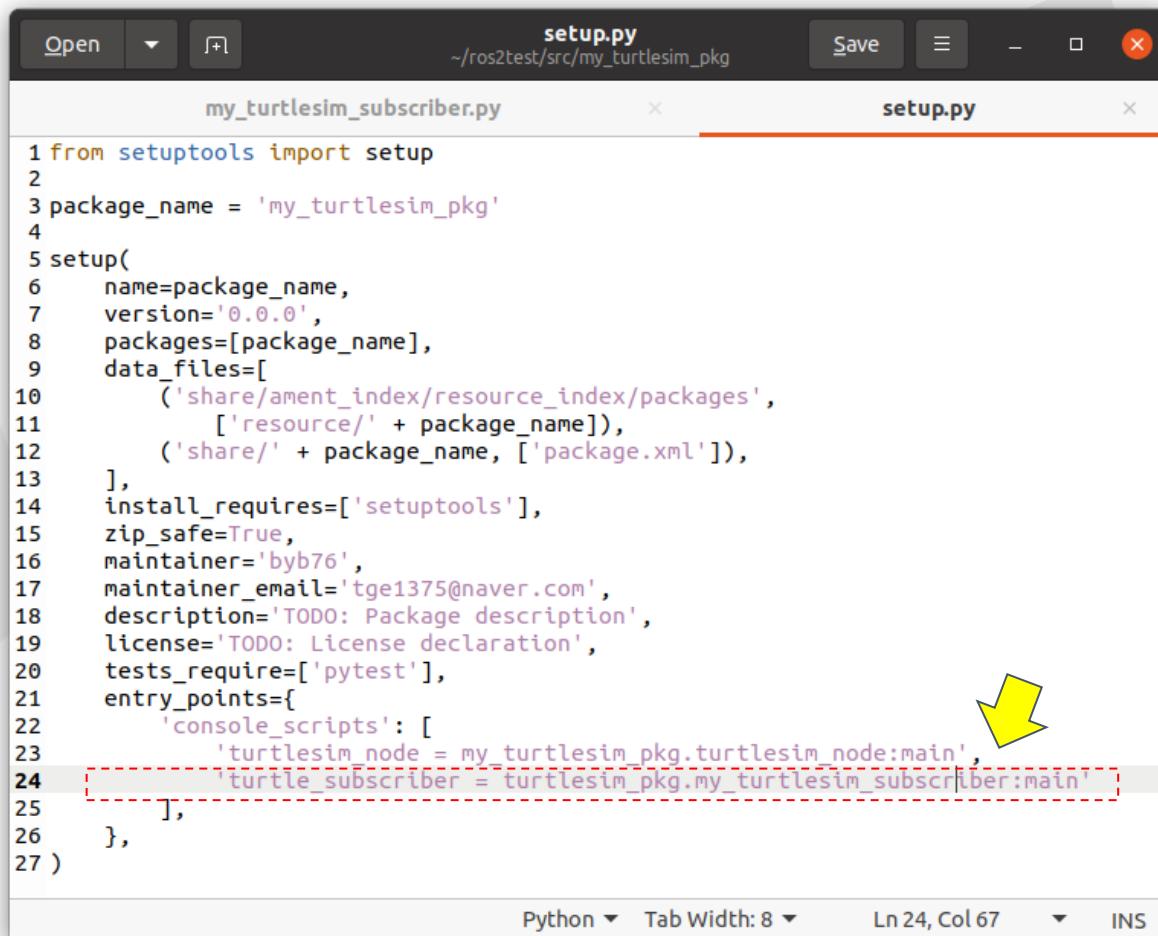
The screenshot shows the Visual Studio Code interface with the title bar "turtlesim\_subscriber.py - src - Visual Studio Code". The left sidebar contains icons for file operations like Open, Save, Find, and Run. The main editor area displays the Python code for a ROS 2 subscriber node. The code imports rclpy and Node from rclpy.node, and Pose from turtlesim.msg. It defines a class TurtlesimSubscriber that inherits from Node. The \_\_init\_\_ method initializes the node with the name 'turtlesim\_subscriber' and creates a subscription to the '/turtle1/pose' topic with a callback function. The callback prints the X, Y, and yaw values of the turtle pose. The main function initializes the rclpy library, creates an instance of TurtlesimSubscriber, and spins it. Finally, it destroys the node and shuts down the rclpy library. An if \_\_name\_\_ == '\_\_main\_\_': block calls the main function. The status bar at the bottom indicates the file is ROS2\_foxy, has 0 errors, and is in Python mode.

```
1 import rclpy as rp
2 from rclpy.node import Node
3 from turtlesim.msg import Pose
4
5
6 class TurtlesimSubscriber(Node):
7 def __init__(self):
8 super().__init__('turtlesim_subscriber')
9 self.subscription = self.create_subscription(Pose, '/turtle1/pose', self.callback, 10)
10
11 def callback(self, msg):
12 # print("X: ", msg.x, "Y: ", msg.y)
13 print("turtle pose X: ", "{:.2f}".format(msg.x),
14 " / Y: ", "{:.2f}".format(msg.y),
15 " Yaw: ", "{:.2f}".format(msg.theta))
16
17 # print("turtle angle Yaw: ", "{:.2f}".format(msg.theta))
18
19 def main(args=None):
20 rp.init(args=args)
21
22 turtlesim_subscriber = TurtlesimSubscriber()
23 rp.spin(turtlesim_subscriber)
24
25 turtlesim_subscriber.destroy_node()
26 rp.shutdown()
27
28 if __name__ == '__main__':
29 main()
```

# ROS2 실습

## turtlesim topic subscriber 노드 생성

setup.py 업데이트: entrypoint 추가 (, 주의)



```
setup.py
~/ros2test/src/my_turtlesim_pkg
Save
my_turtlesim_subscriber.py
setup.py

1 from setuptools import setup
2
3 package_name = 'my_turtlesim_pkg'
4
5 setup(
6 name=package_name,
7 version='0.0.0',
8 packages=[package_name],
9 data_files=[
10 ('share/ament_index/resource_index/packages',
11 ['resource/' + package_name]),
12 ('share/' + package_name, ['package.xml']),
13],
14 install_requires=['setuptools'],
15 zip_safe=True,
16 maintainer='byb76',
17 maintainer_email='tge1375@naver.com',
18 description='TODO: Package description',
19 license='TODO: License declaration',
20 tests_require=['pytest'],
21 entry_points={
22 'console_scripts': [
23 'turtlesim_node = my_turtlesim_pkg.turtlesim_node:main',
24 'turtle_subscriber = turtlesim_pkg.my_turtlesim_subscriber:main'
25],
26 },
27)
```

The screenshot shows a code editor with two tabs: "my\_turtlesim\_subscriber.py" and "setup.py". The "setup.py" tab is active, displaying Python code for a ROS2 package named "my\_turtlesim\_pkg". A yellow arrow points to the "entry\_points" section, specifically to the line "entry\_points={ 'console\_scripts': [ 'turtlesim\_node = my\_turtlesim\_pkg.turtlesim\_node:main', 'turtle\_subscriber = turtlesim\_pkg.my\_turtlesim\_subscriber:main' ] }". This line defines two command-line scripts: "turtlesim\_node" and "turtle\_subscriber". The "turtlesim\_node" script is associated with the "turtlesim\_node" function in the "my\_turtlesim\_pkg.turtlesim\_node" module, and the "turtle\_subscriber" script is associated with the "my\_turtlesim\_subscriber" function in the "turtlesim\_pkg.my\_turtlesim\_subscriber" module.

# ROS2 실습

## package build

```
$ colcon build
```

```
$. install/localsetup.bash
```



```
byb76@rlhp: ~/ros2test$ colcon build
Starting >>> my_turtlesim_pkg
Finished <<< my_turtlesim_pkg [2.13s]

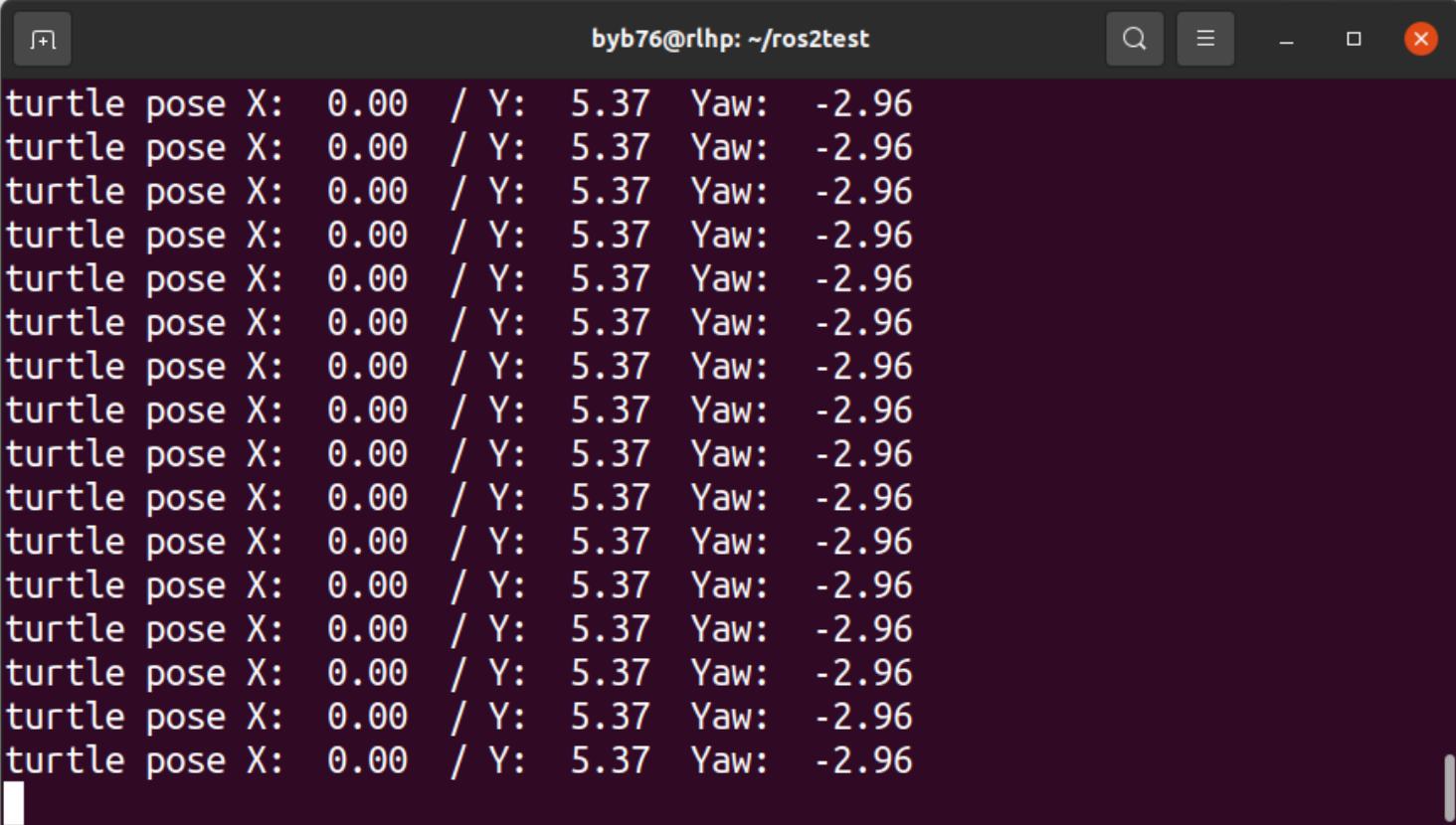
Summary: 1 package finished [2.56s]
(ROS 2 foxy) byb76@rlhp:~/ros2test$
```

# ROS2 실습

## package 실행

```
$ros2 run my_turtlesim_pkg turtle_subscriber
```

>위 이름은 setup.py entrypoint 이름과 일치

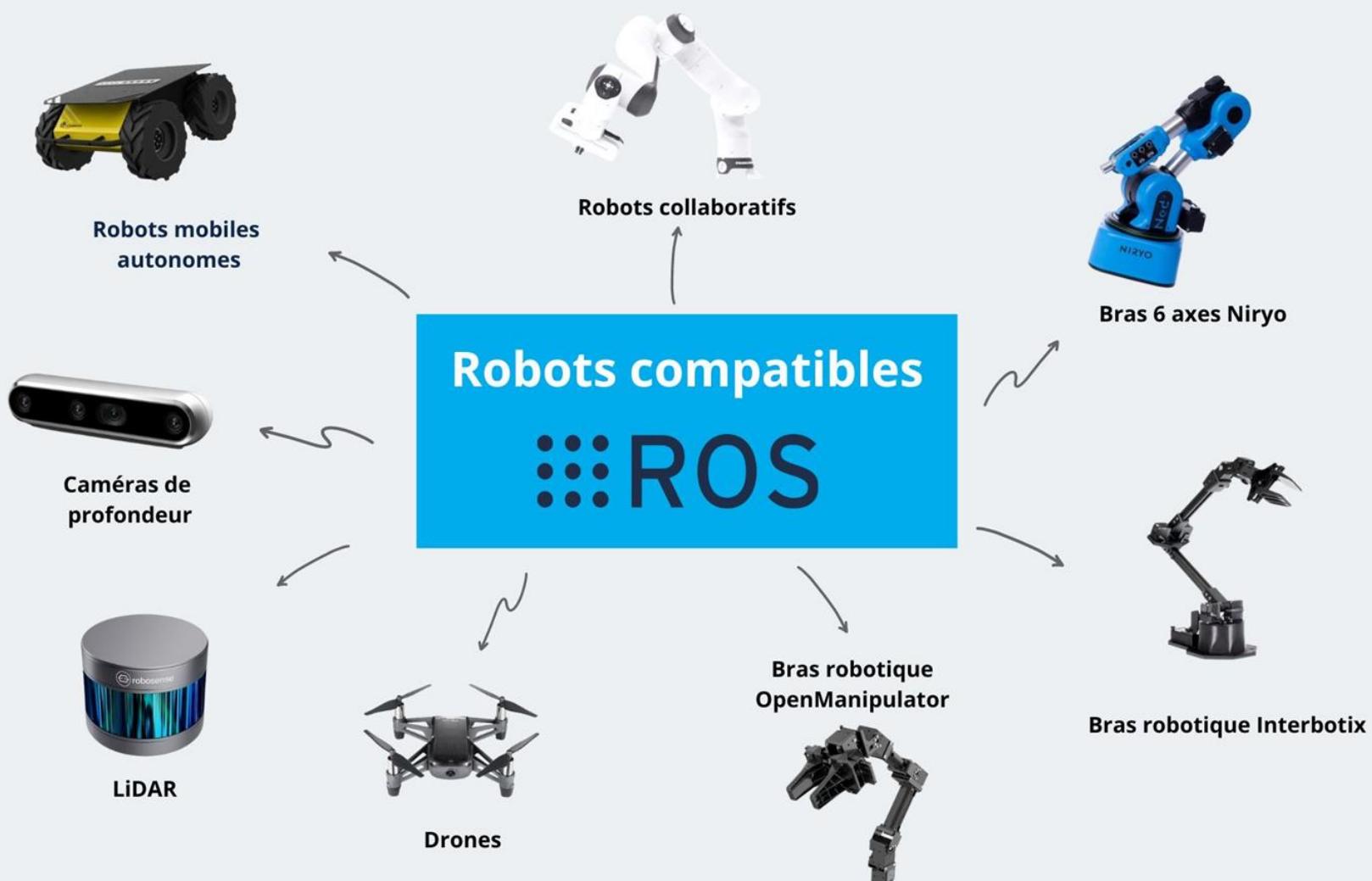


The screenshot shows a terminal window titled "byb76@rlhp: ~/ros2test". The window contains a series of identical lines of text, each representing the pose of a turtle in a simulation. The text is as follows:

```
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
turtle pose X: 0.00 / Y: 5.37 Yaw: -2.96
```

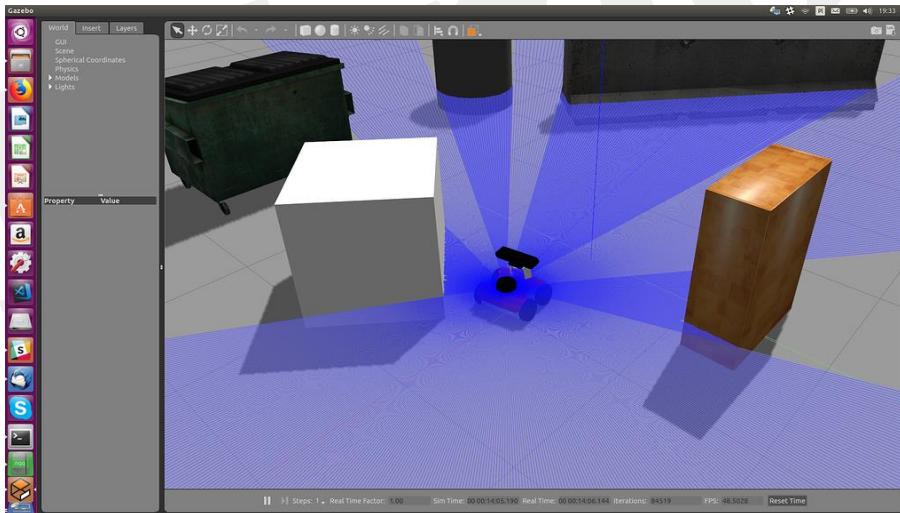


# ROS2 HW 실습



# Lidar sensor

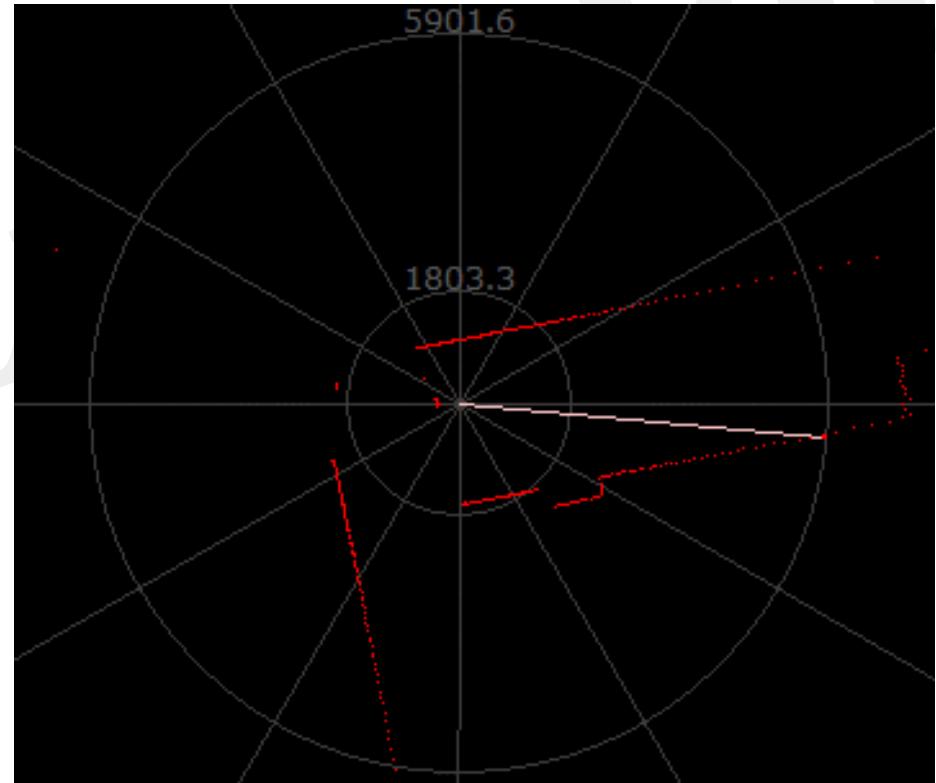
## 실습



# ROS2 HW

## Lidar

demo - rplidar a2





# ROS2 HW

## RPLidar

link: [https://github.com/Slamtec/rplidar\\_ros](https://github.com/Slamtec/rplidar_ros)

The screenshot shows the GitHub repository page for `Slamtec/rplidar_ros`. The repository is public and has 6 branches and 6 tags. The main code tab is selected, showing a list of files and their commit history. Key commits include support for scan frequency configuration, first release of the ROS package, and updates to the SDK and README. The repository has 334 stars, 18 watchers, and 399 forks. It includes sections for About, Releases, Packages, and Contributors.

**Code** Issues 67 Pull requests 20 Actions Projects Security Insights

Go to file Code

**About**

No description, website, or topics provided.

Readme BSD-2-Clause license 334 stars 18 watching 399 forks

**Releases**

6 tags

**Packages**

No packages published

**Contributors** 11

WubinXia scan frequency configuration support  
7b011f1 on Oct 17, 2022 61 commits

| File           | Description                                                              | Time Ago     |
|----------------|--------------------------------------------------------------------------|--------------|
| launch         | scan frequency configuration support                                     | 2 months ago |
| rviz           | First release of RPLIDAR ROS package                                     | 8 years ago  |
| scripts        | update to RPLIDAR SDK 1.5.2                                              | 6 years ago  |
| sdk            | support RPLIDAR S2E                                                      | 7 months ago |
| src            | scan frequency configuration support                                     | 2 months ago |
| CHANGELOG.rst  | release v2.0.0                                                           | last year    |
| CMakeLists.txt | release v2.0.0                                                           | last year    |
| LICENSE        | [release] rplidar_ros release 1.7.0                                      | 4 years ago  |
| README.md      | Update README.md                                                         | 3 months ago |
| package.xml    | upgrade sdk 1.10.0                                                       | 4 years ago  |
| rplidar_A1.png | change rplidar_A1.png                                                    | 6 years ago  |
| rplidar_A2.png | add new picture for install and fixed the default param of inverted a... | 6 years ago  |

README.md



# Lidar

Demo - rplidar a2

link: [https://github.com/Slamtec/rplidar\\_ros](https://github.com/Slamtec/rplidar_ros)

```
$ git clone -b ros2 https://github.com/Slamtec/rplidar_ros.git
$ colcon build --package-select rplidar_ros
$. intall/setup.bash
$ ros2 launch rpliar_ros view_rplidar.launch.py
```

```
rlmodel@rlmodel: ~/ros2_ws
In file included from /home/rlmodel/ros2_ws/src/rplidar_ros/src/standalone_rplidar.cpp
:16:
/home/rlmodel/ros2_ws/src/rplidar_ros/include/rplidar_node.hpp:76:14: warning: 'float
rplidar_ros::getAngle(const rplidar_response_measurement_node_hq_t&)' defined but not
used [-Wunused-function]
 76 | static float getAngle(const rplidar_response_measurement_node_hq_t & node)
 |
...
Finished <<< rplidar_ros [13.2s]

Summary: 1 package finished [13.8s]
 1 package had stderr output: rplidar_ros
rlmodel@rlmodel:~/ros2_ws$
```



# Lidar

## Demo - chmod

```
$ ls /dev/ttyUSB*
$ sudo chmod 777 /dev/ttyUSB0 (USB0, USB1...)
```

The screenshot shows a terminal window with a dark theme. The title bar reads "rlmodel@rlmodel: ~/ros2\_ws". The terminal content is as follows:

```
rlmodel@rlmodel:~/ros2_ws$ ls -al /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 Jun 30 10:21 /dev/ttyUSB0
rlmodel@rlmodel:~/ros2_ws$ sudo chmod 777 /dev/ttyUSB0
[sudo] password for rlmodel:
rlmodel@rlmodel:~/ros2_ws$ ls -al /dev/ttyUSB0
crwxrwxrwx 1 root dialout 188, 0 Jun 30 10:21 /dev/ttyUSB0
rlmodel@rlmodel:~/ros2_ws$
```



# Lidar

## Demo - rplidar a2

```
$ros2 launch rplidar_ros view_rplidar.launch.py
```

```
ok
rlmodel@rlmodel:~/ros2_ws$ ros2 launch rplidar_ros view_rplidar.launch.py
[INFO] [launch]: All log files can be found below /home/rlmodel/.ros/log/2023-06-30-09-5
5-20-681997-rlmodel-4257
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [rplidar_composition-1]: process started with pid [4259]
[INFO] [rviz2-2]: process started with pid [4261]
[rplidar_composition-1] [INFO] [1688086520.962289624] [rplidar_composition]: RPLIDAR run
ning on ROS 2 package rplidar_ros. SDK Version: '1.12.0'
[rplidar_composition-1] [ERROR] [1688086520.962472021] [rplidar_composition]: Error, can
not bind to the specified serial port '/dev/ttyUSB0'.
[rviz2-2] [INFO] [1688086521.701884739] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] [INFO] [1688086521.702042398] [rviz2]: OpenGL version: 4.6 (GLSL 4.6)
[rviz2-2] [INFO] [1688086521.760985360] [rviz2]: Stereo is NOT SUPPORTED
```



# Lidar

## Demo - rplidar a2

```
$ros2 launch rplidar_ros view_rplidar.launch.py
```

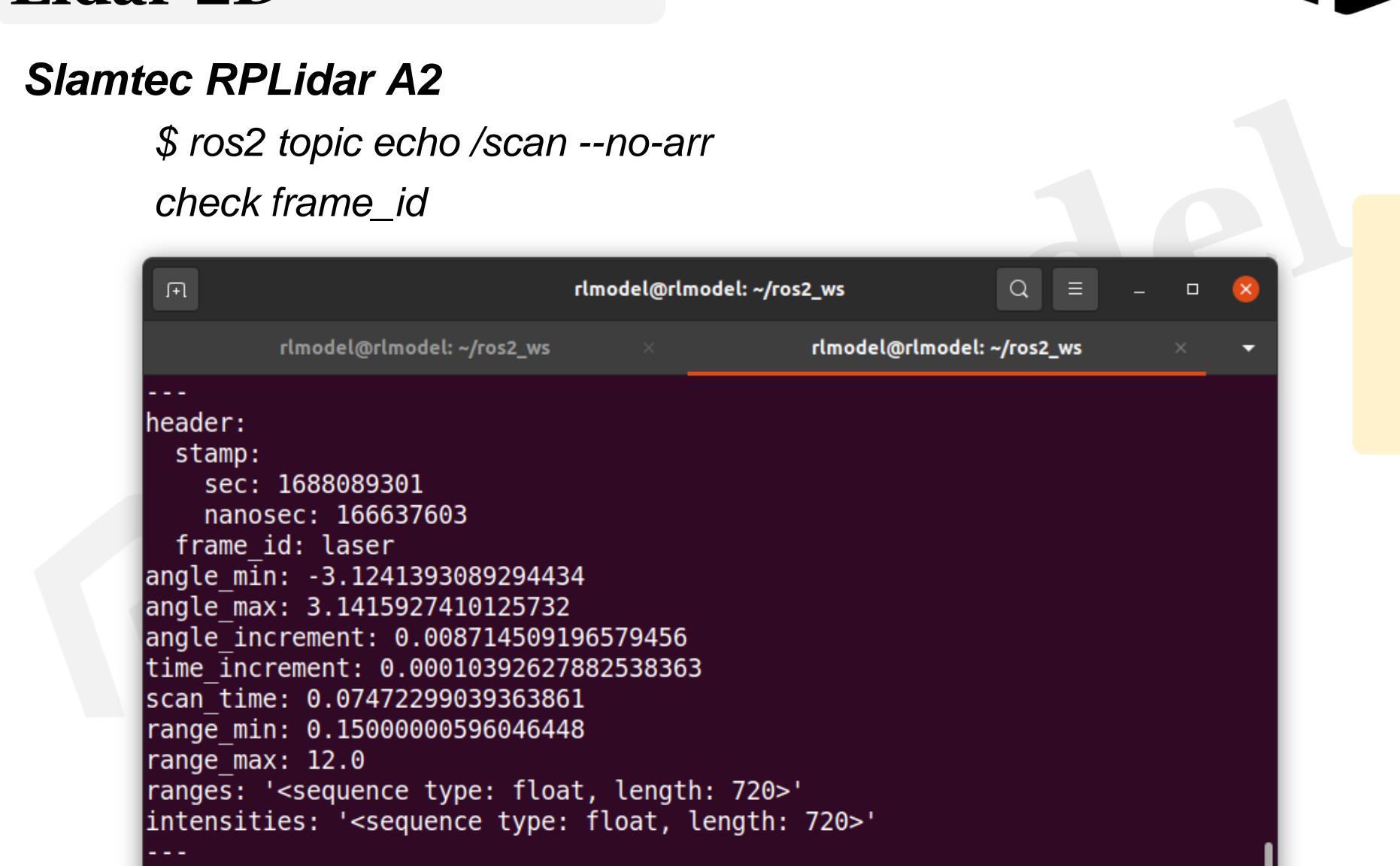
The screenshot shows a terminal window titled "rlmodel@rlmodel: ~/ros2\_ws". It displays the output of the command "ros2 topic list", which lists various ROS 2 topics. The topics listed are: /clicked\_point, /initialpose, /move\_base\_simple/goal, /parameter\_events, /rosout, /scan, /tf, and /tf\_static.

```
rlmodel@rlmodel:~/ros2_ws$ ros2 topic list
/clicked_point
/initialpose
/move_base_simple/goal
/parameter_events
/rosout
/scan
/tf
/tf_static
rlmodel@rlmodel:~/ros2_ws$
```

# Lidar 2D

## Slamtec RPLidar A2

```
$ ros2 topic echo /scan --no-arr
check frame_id
```



```
rlmodel@rlmodel: ~/ros2_ws
```

```
rlmodel@rlmodel: ~/ros2_ws
```

```
rlmodel@rlmodel: ~/ros2_ws
```

```

```

```
header:
```

```
 stamp:
```

```
 sec: 1688089301
```

```
 nanosec: 166637603
```

```
 frame_id: laser
```

```
angle_min: -3.1241393089294434
```

```
angle_max: 3.1415927410125732
```

```
angle_increment: 0.008714509196579456
```

```
time_increment: 0.00010392627882538363
```

```
scan_time: 0.07472299039363861
```

```
range_min: 0.15000000596046448
```

```
range_max: 12.0
```

```
ranges: '<sequence type: float, length: 720>'
```

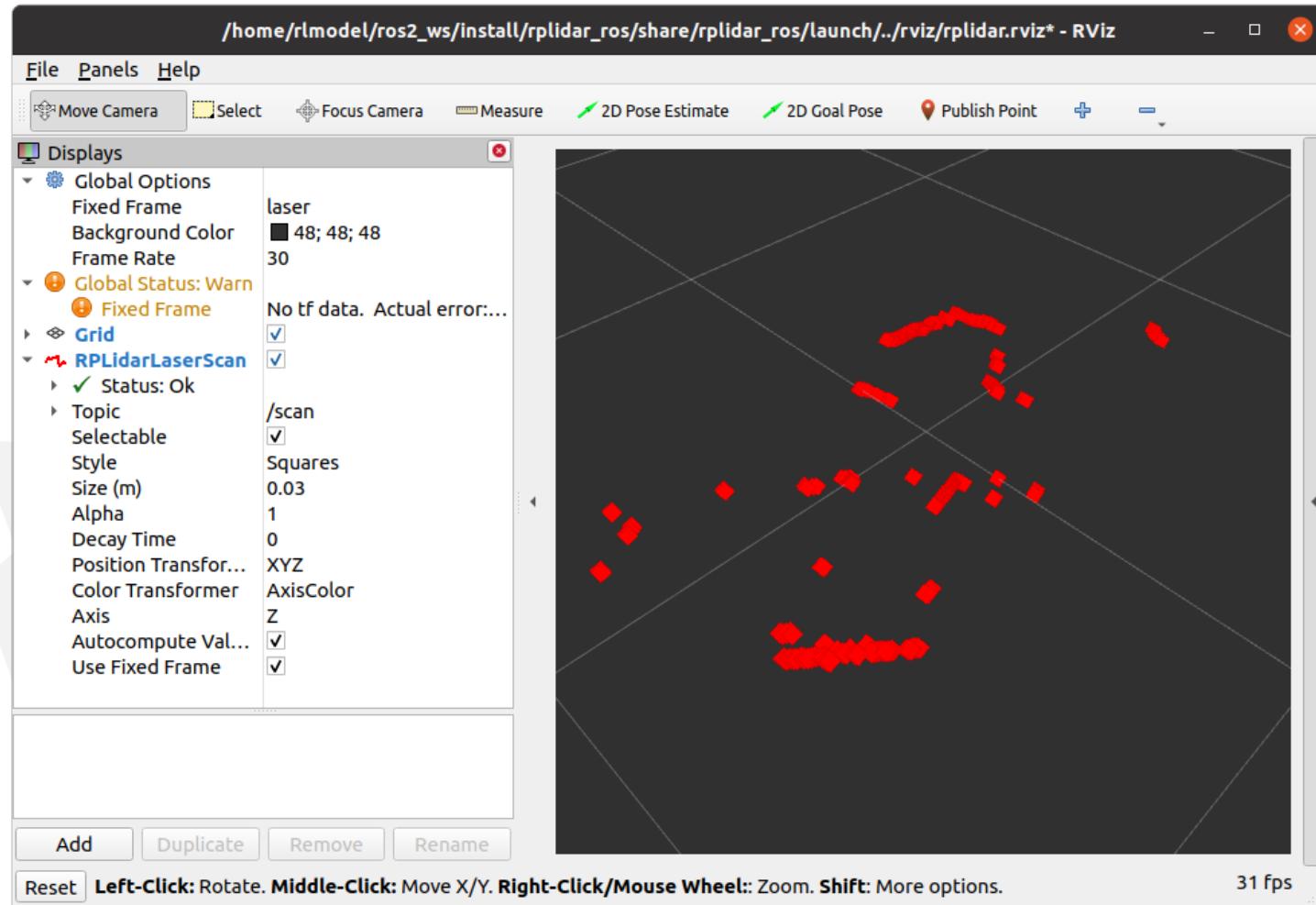
```
intensities: '<sequence type: float, length: 720>'
```

```

```

# Lidar 2D

## Demo - rplidar a2





# Lidar

## Demo - rplidar service

check service: \$ ros2 service list

The screenshot shows a terminal window with a dark background and light-colored text. The title bar of the terminal window reads "rlmodel@rlmodel: ~/ros2\_ws". The window contains the output of the command "ros2 service list", which lists various ROS services. The services listed are:

```
/rplidar_composition/get_parameters
/rplidar_composition/list_parameters
/rplidar_composition/set_parameters
/rplidar_composition/set_parameters_atomically
/rviz/describe_parameters
/rviz/get_parameter_types
/rviz/get_parameters
/rviz/list_parameters
/rviz/set_parameters
/rviz/set_parameters_atomically
/start_motor
/stop_motor
```

The prompt at the bottom of the terminal window is "rlmodel@rlmodel:~/ros2\_ws\$".



# Lidar

## Demo - rplidar service

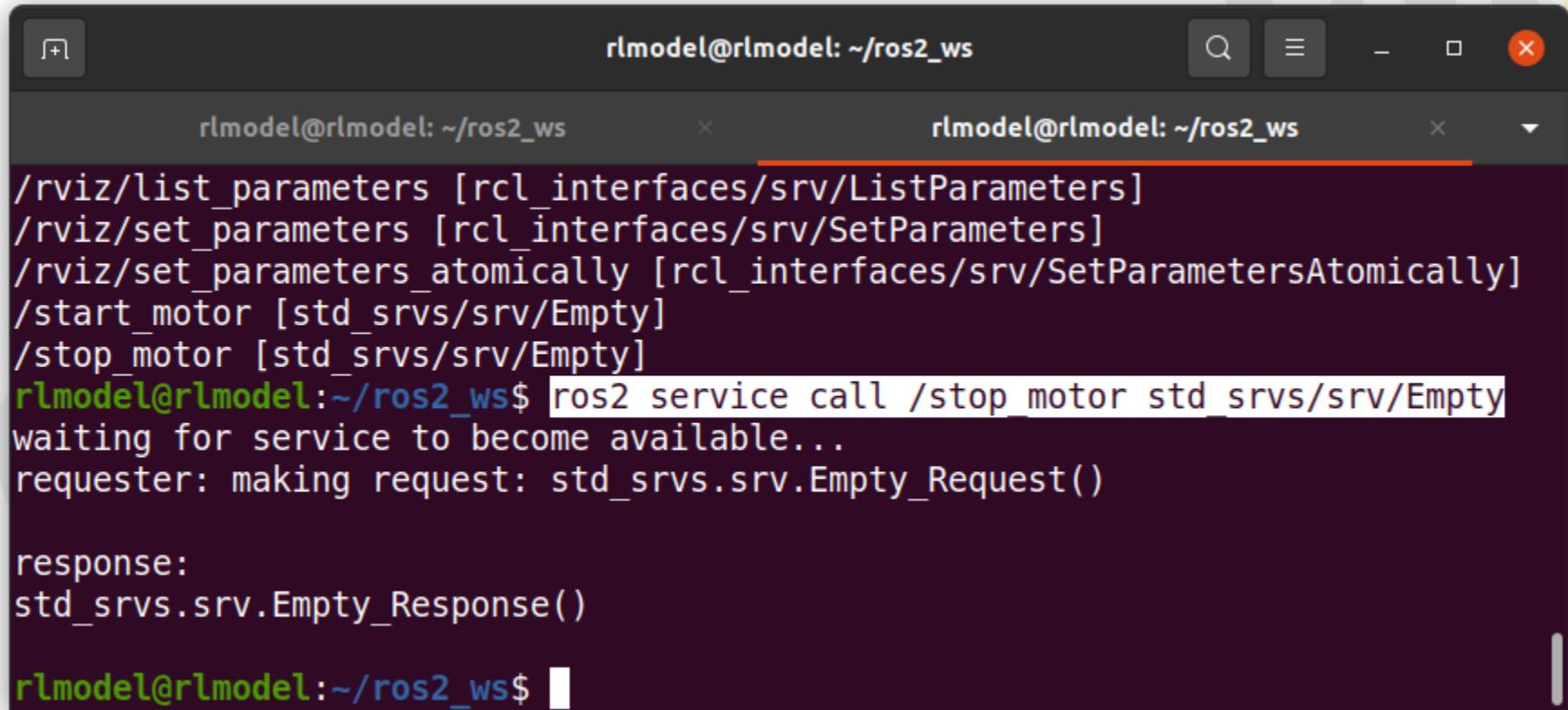
check service: \$ ros2 service list -t

```
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws
s]
/rplidar_composition/set_parameters_atomically [rcl_interfaces/srv/SetParametersAtomically]
/rviz/describe_parameters [rcl_interfaces/srv/DescribeParameters]
/rviz/get_parameter_types [rcl_interfaces/srv/GetParameterTypes]
/rviz/get_parameters [rcl_interfaces/srv/GetParameters]
/rviz/list_parameters [rcl_interfaces/srv/ListParameters]
/rviz/set_parameters [rcl_interfaces/srv/SetParameters]
/rviz/set_parameters_atomically [rcl_interfaces/srv/SetParametersAtomically]
/start_motor [std_srvs/srv/Empty]
/stop_motor [std_srvs/srv/Empty]
rlmodel@rlmodel:~/ros2_ws$
```

# Lidar

## Demo - rplidar service stop\_motor

```
$ ros2 service call /stop_motor std_srvs/srv/Empty
```



A terminal window titled "rlmodel@rlmodel: ~/ros2\_ws" showing the output of a ROS 2 service call. The window has three tabs, with the middle one active. The text shows various service names and their descriptions, followed by the command being run and its progress.

```
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws
rlmodel@rlmodel: ~/ros2_ws

/rviz/list_parameters [rcl_interfaces/srv/ListParameters]
/rviz/set_parameters [rcl_interfaces/srv/SetParameters]
/rviz/set_parameters_atomically [rcl_interfaces/srv/SetParametersAtomically]
/start_motor [std_srvs/srv/Empty]
/stop_motor [std_srvs/srv/Empty]
rlmodel@rlmodel:~/ros2_ws$ ros2 service call /stop_motor std_srvs/srv/Empty
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()

rlmodel@rlmodel:~/ros2_ws$
```



# Lidar

## Demo - rplidar service stop motor

```
$ ros2 service call /stop_motor std_srvs/srv/Empty {}
```

```
byb76@rlhp: ~/ros2_ws
byb76@rlhp: ~/ros2_ws 73x19
(ROS 2 foxy) byb76@rlhp:~/ros2_ws$ ros2 service call /stop_motor std_srvs/srv/Empty {}
1673182032.650391 [0] ros2: using network interface wlp2s0 (udp/192.168.0.19) selected arbitrarily from: wlp2s0, docker0
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()

(ROS 2 foxy) byb76@rlhp:~/ros2_ws$
```



# Lidar

## Demo - rplidar service start\_motor

```
$ ros2 service call /stop_motor std_srvs/srv/Empty
```

The screenshot shows a terminal window titled "rlmodel@rlmodel: ~/ros2\_ws". It contains two tabs, both labeled "rlmodel@rlmodel: ~/ros2\_ws". The bottom tab is active and displays the following text:

```
requester: making request: std_srvs.srv.Empty_Request()
response:
std_srvs.srv.Empty_Response()

rlmodel@rlmodel:~/ros2_ws$ ros2 service call /start_motor std_srvs/srv/Empty
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()

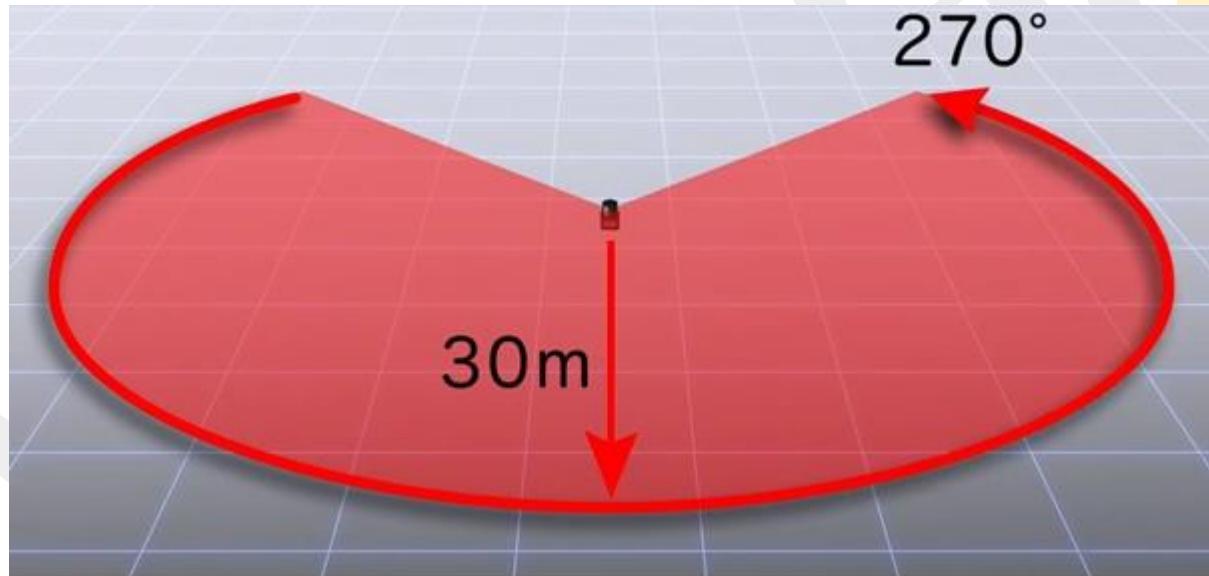
rlmodel@rlmodel:~/ros2_ws$
```

# Lidar 2D

## scan topic

laser scan data

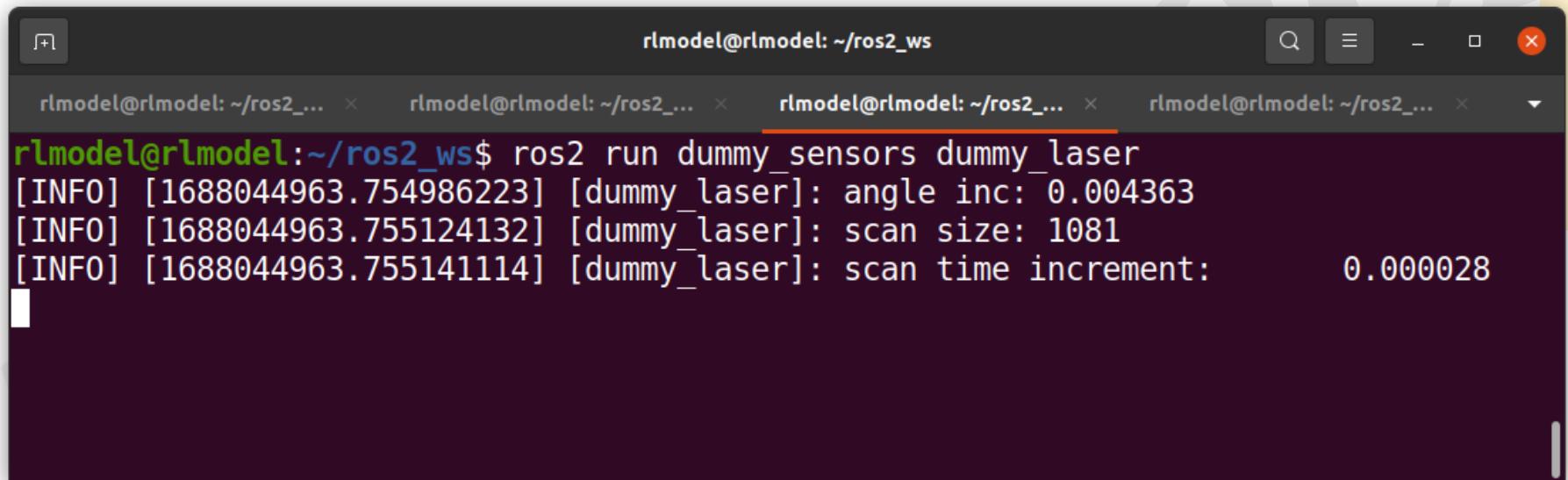
>simulation example





# Lidar 2D

scan topic  
dummy\_sensor package



A terminal window titled "rlmodel@rlmodel: ~/ros2\_ws" is shown. The window has four tabs, with the third one being the active tab. The command "ros2 run dummy\_sensors dummy\_laser" is run in the terminal, and its output is displayed:

```
rlmodel@rlmodel:~/ros2_ws$ ros2 run dummy_sensors dummy_laser
[INFO] [1688044963.754986223] [dummy_laser]: angle inc: 0.004363
[INFO] [1688044963.755124132] [dummy_laser]: scan size: 1081
[INFO] [1688044963.755141114] [dummy_laser]: scan time increment: 0.000028
```

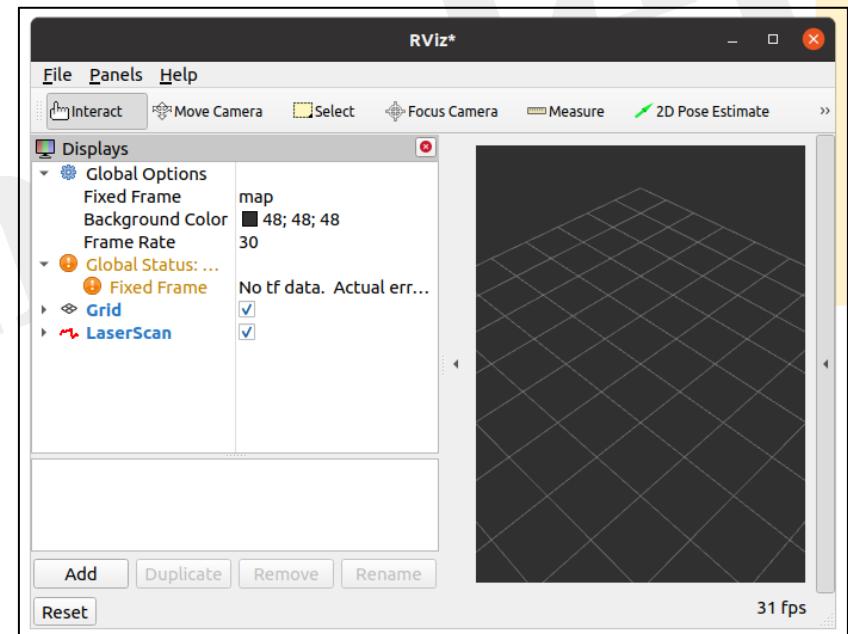
# Lidar 2D

## scan topic

dummy\_sensor package - rviz  
frame: single\_rrbot\_hokuyo\_link

```
rlmodel@rlmodel: ~/ros2_ws
```

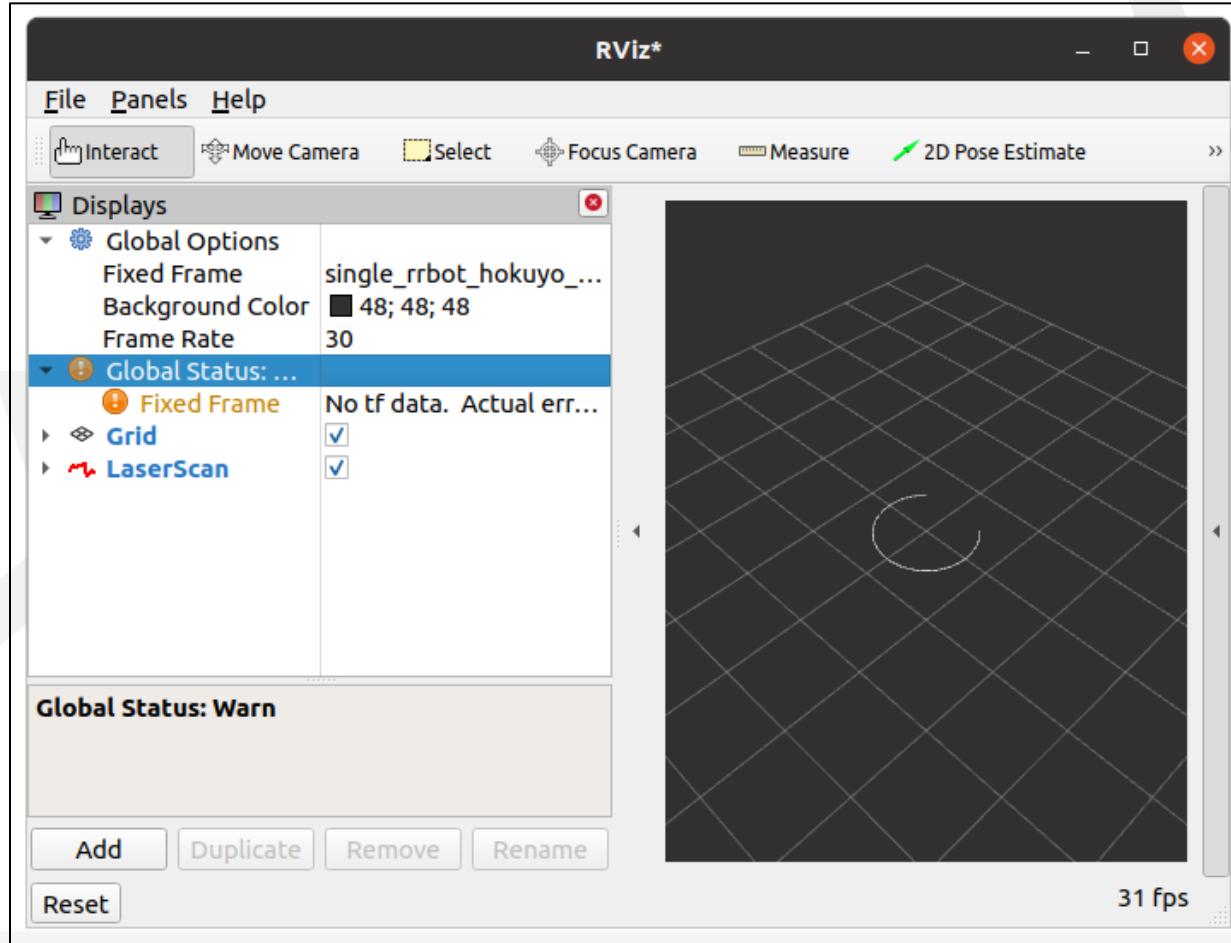
```
ng message: frame 'single_rrbot_hokuyo_link' at time 168804
5056.122 for reason 'Unknown'
[INFO] [1688045056.503279037] [rviz]: Message Filter dropping message: frame 'single_rrbot_hokuyo_link' at time 168804
5056.155 for reason 'Unknown'
[INFO] [1688045056.534249827] [rviz]: Message Filter dropping message: frame 'single_rrbot_hokuyo_link' at time 168804
5056.188 for reason 'Unknown'
[INFO] [1688045056.566900612] [rviz]: Message Filter dropping message: frame 'single_rrbot_hokuyo_link' at time 168804
5056.222 for reason 'Unknown'
```



# Lidar 2D

## scan topic

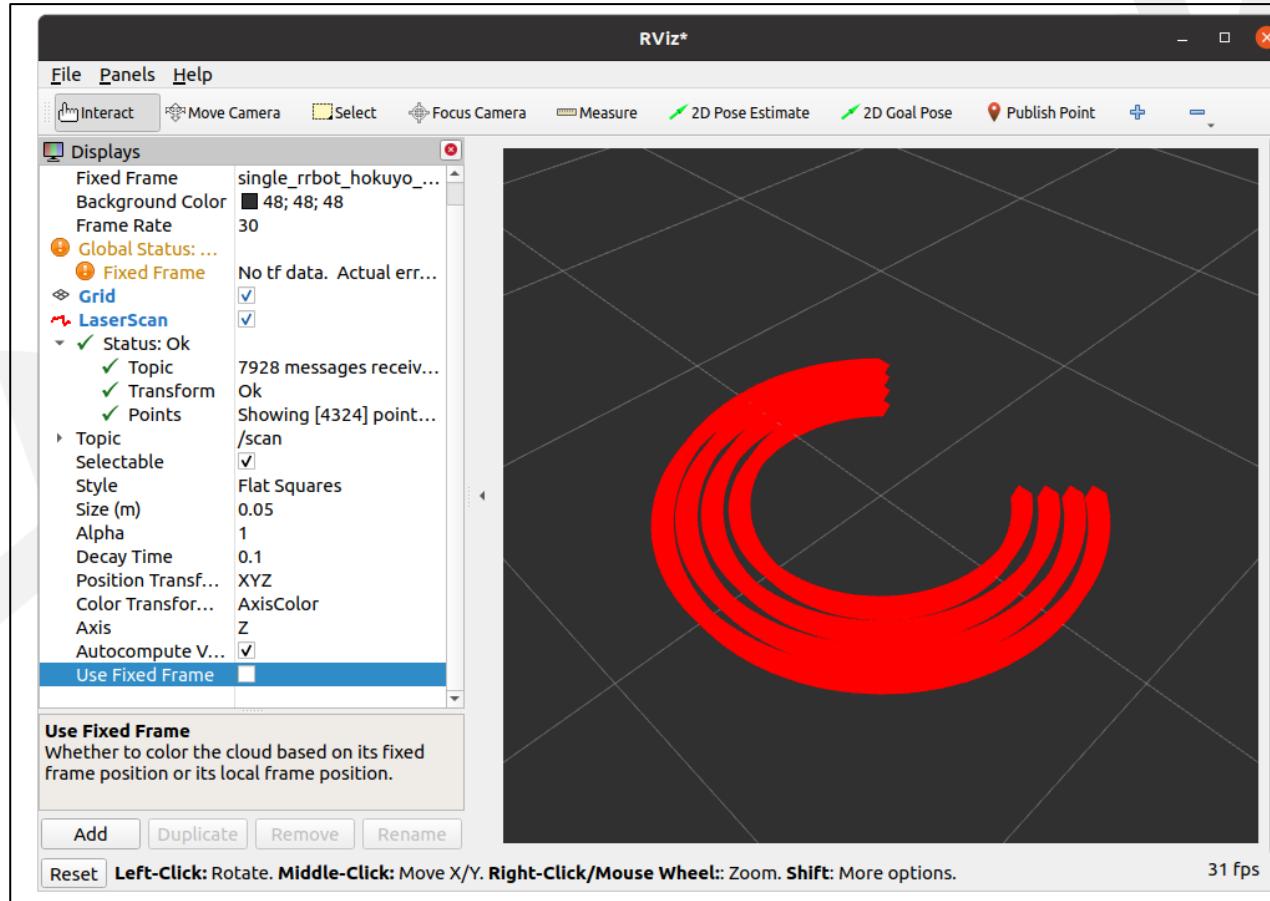
dummy\_sensor package - rviz  
frame: single\_rrbot\_hokuyo\_link



# Lidar 2D

## scan topic

dummy\_sensor package - rviz  
frame: single\_rrbot\_hokuyo\_link

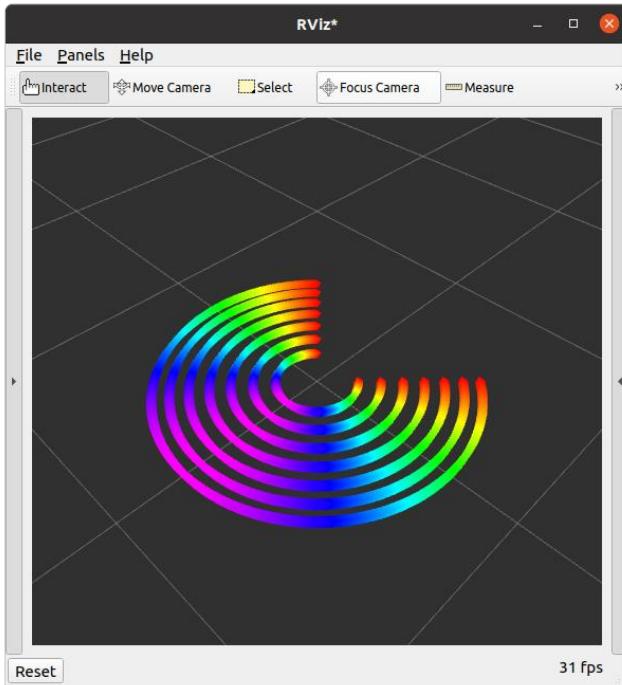


# Lidar 2D

## topic bandwidth

지정된 토픽 메시지의 송수신되는 토픽의 초당 대역폭

\$ ros2 topic bw /scan

A terminal window titled 'rlmodel@rlmodel: ~/ros2\_ws' showing the output of the command 'ros2 topic bw /scan'. The output shows three sets of statistics for different message counts: 30 messages at 134.04 KB/s, 60 messages at 133.05 KB/s, and 90 messages at 132.74 KB/s. Each set includes the mean message size and its minimum and maximum values.

```
rlmodel@rlmodel:~/ros2_ws$ ros2 topic bw /scan
Subscribed to [/scan]
134.04 KB/s from 30 messages
 Message size mean: 4.40 KB min: 4.40 KB max: 4.40 KB
133.05 KB/s from 60 messages
 Message size mean: 4.40 KB min: 4.40 KB max: 4.40 KB
132.74 KB/s from 90 messages
 Message size mean: 4.40 KB min: 4.40 KB max: 4.40 KB
```

# Lidar 2D

## topic hertz

토픽의 전송 주기

\$ ros2 topic hz /scan

A terminal window titled 'rlmodel@rlmodel: ~/ros2\_ws'. The window contains several tabs, with the fifth tab from the left active and highlighted in orange. The terminal is displaying the command 'ros2 topic hz /scan' followed by its execution results. The output shows four sets of statistics for the lidar topic:

```
rlmodel@rlmodel:~/ros2_ws$ ros2 topic hz /scan
average rate: 29.982
 min: 0.033s max: 0.034s std dev: 0.00021s window: 31
average rate: 29.987
 min: 0.033s max: 0.034s std dev: 0.00021s window: 61
average rate: 29.994
 min: 0.032s max: 0.034s std dev: 0.00023s window: 92
average rate: 30.000
 min: 0.032s max: 0.034s std dev: 0.00022s window: 123
```

# Lidar 2D

## topic delay

메시지 발행 시간과 구독 시간의 차를 계산하여 지연 시간을 확인

```
$ ros2 topic delay /scan
```

A terminal window titled 'rlmodel@rlmodel: ~/ros2\_ws' is shown. It contains the command 'ros2 topic delay /scan' followed by its output. The output shows four sets of statistics for the lidar topic:

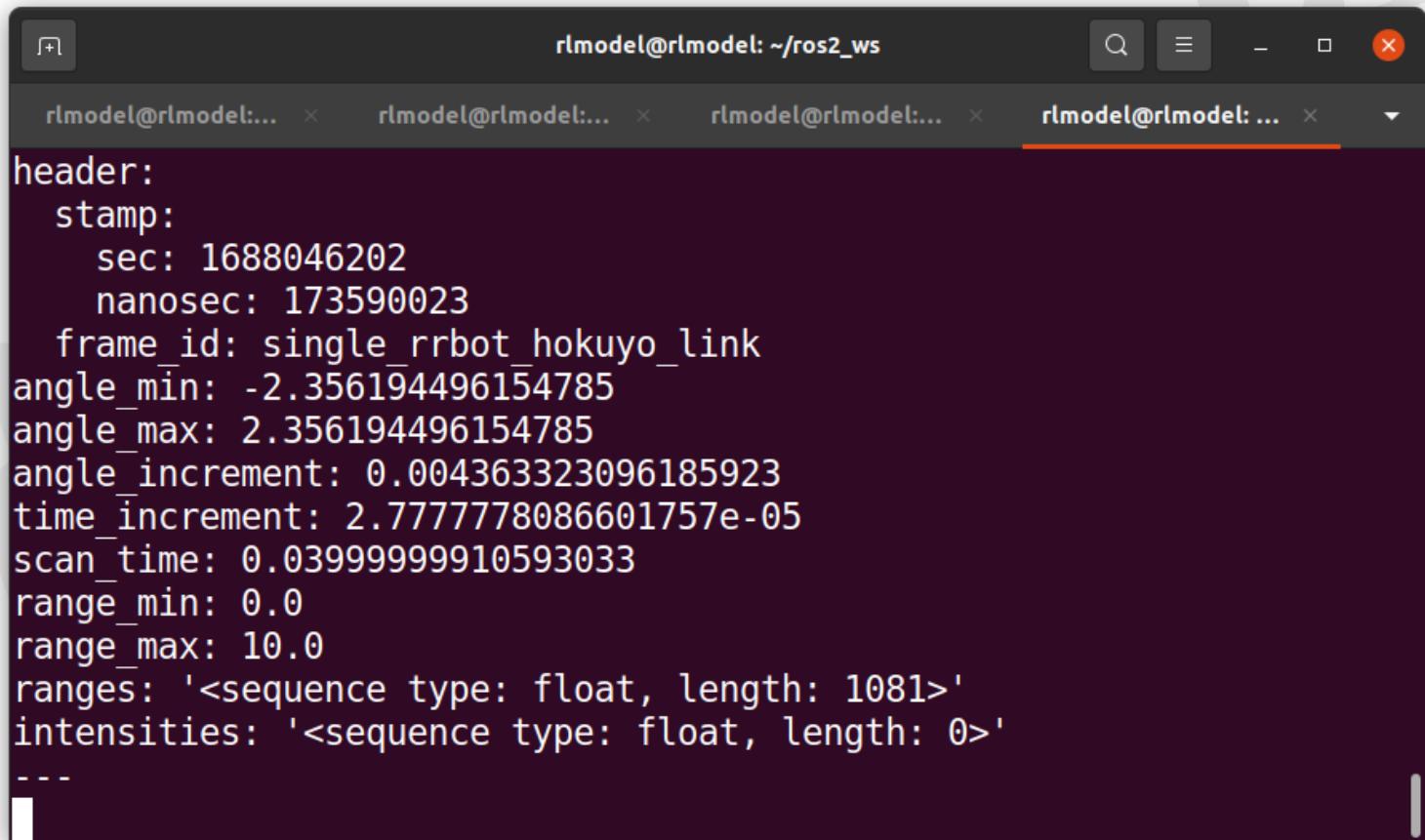
```
rlmodel@rlmodel:~/ros2_ws$ ros2 topic delay /scan
average delay: 0.001
 min: 0.000s max: 0.002s std dev: 0.00041s window: 29
average delay: 0.001
 min: 0.000s max: 0.002s std dev: 0.00041s window: 59
average delay: 0.001
 min: 0.000s max: 0.002s std dev: 0.00039s window: 89
average delay: 0.001
 min: 0.000s max: 0.002s std dev: 0.00037s window: 119
```

# Lidar 2D

## scan topic

fram ID check

```
$ ros2 topic echo /scan --no-arr
```



The screenshot shows a terminal window with four tabs open, all labeled "rlmodel@rlmodel: ...". The fourth tab is active and contains the following text:

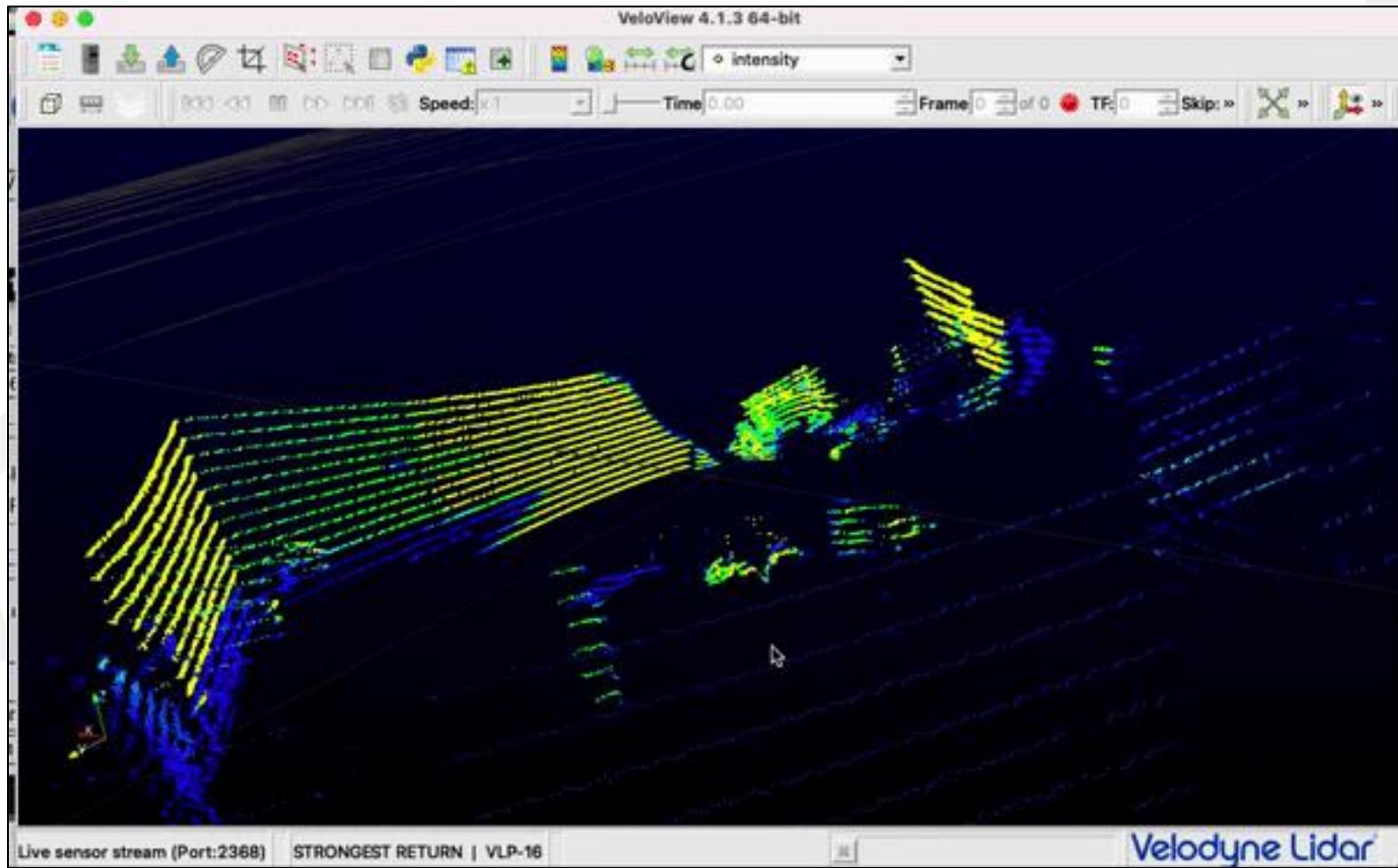
```
header:
 stamp:
 sec: 1688046202
 nanosec: 173590023
 frame_id: single_rrbot_hokuyo_link
angle_min: -2.356194496154785
angle_max: 2.356194496154785
angle_increment: 0.004363323096185923
time_increment: 2.7777778086601757e-05
scan_time: 0.03999999910593033
range_min: 0.0
range_max: 10.0
ranges: '<sequence type: float, length: 1081>'
intensities: '<sequence type: float, length: 0>'

```

# Lidar 3D

scan topic

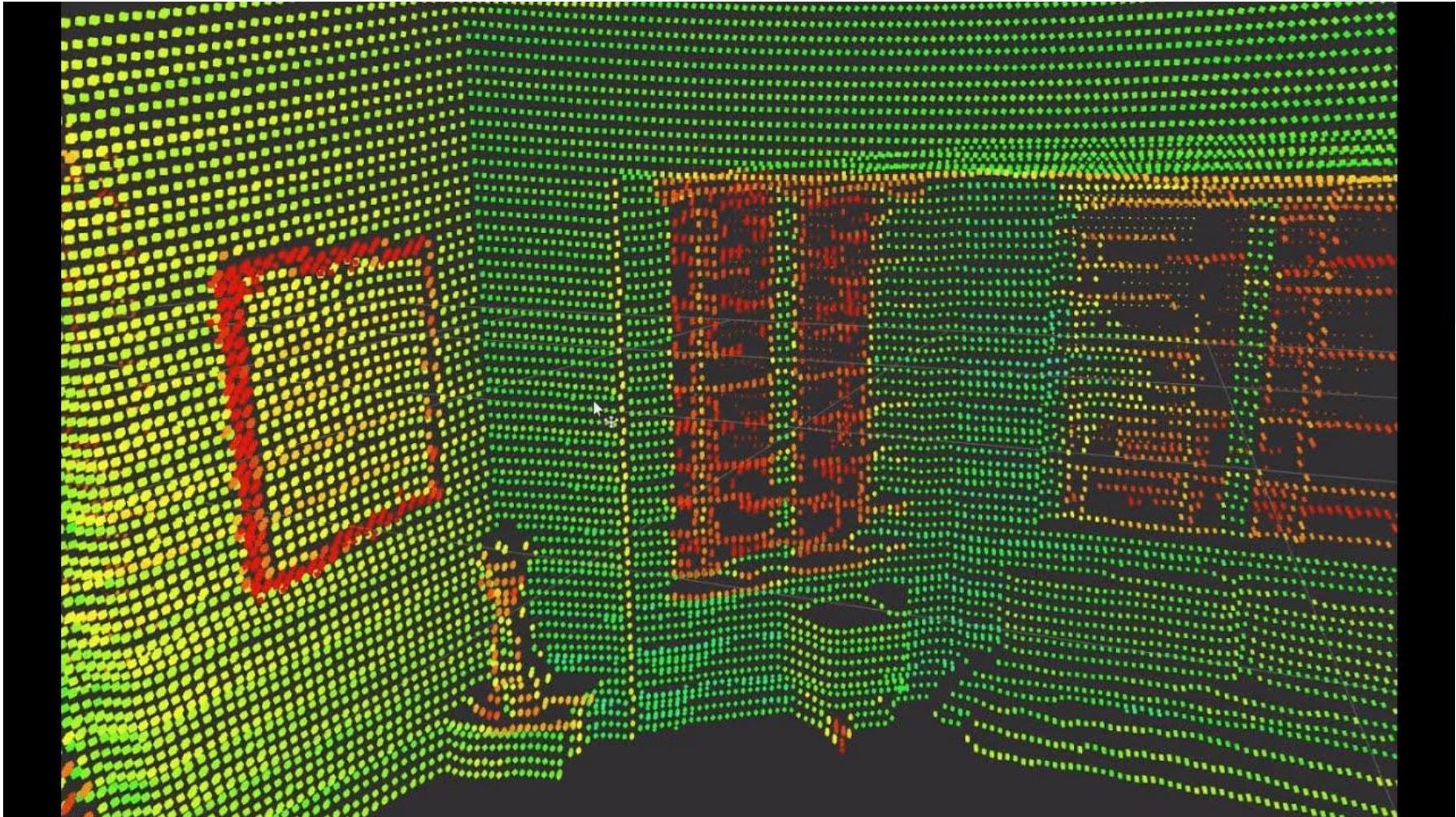
velodyne veloview



# Lidar 3D

scan topic

rviz display





# Camera

## 실습

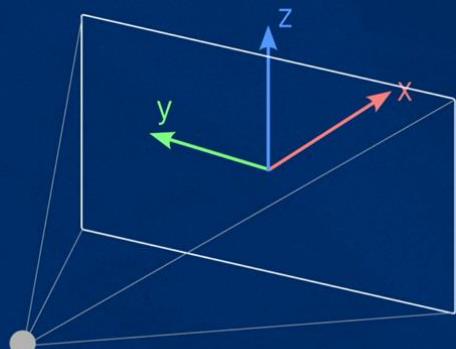


# ROS2 HW

## CAMERA

demo - webcam mage

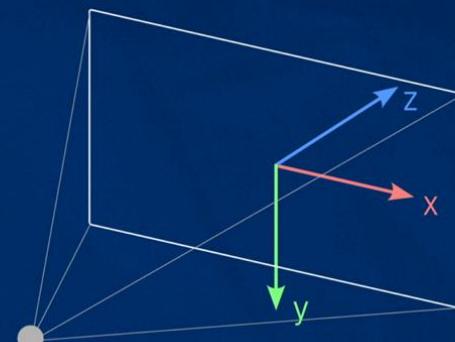
ROS Body Standard



`camera_link`

*See ROS Rep 103 for details...*

Image Standard



`camera_link_optical`

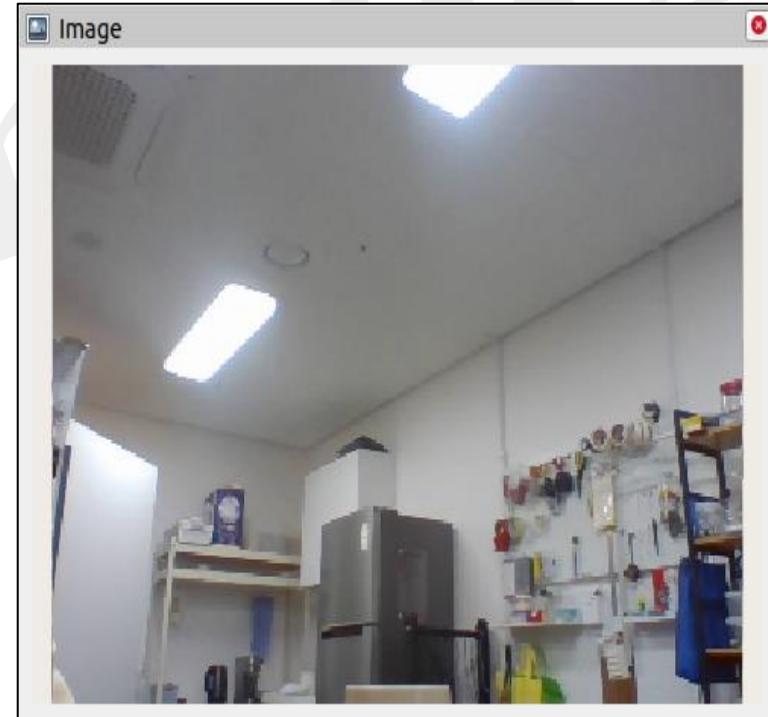
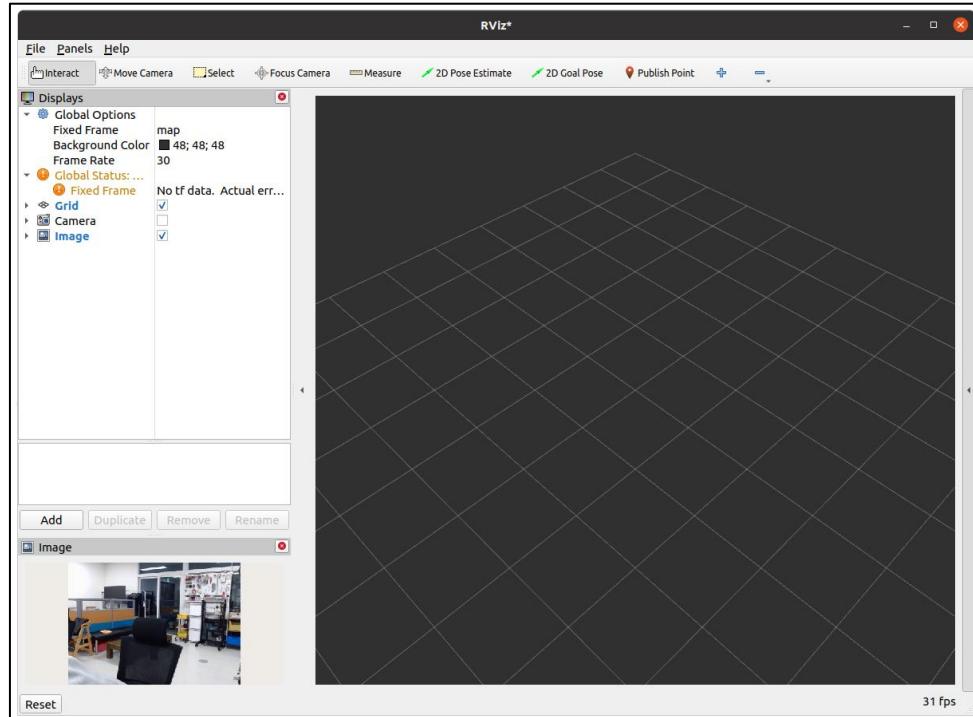
Use as TF frame for all Images etc.

# ROS2 HW

## CAMERA

명령어: \$ ros2 run image\_tools cam2image

확인: rviz2 -> add -> By topic -> /image/Image

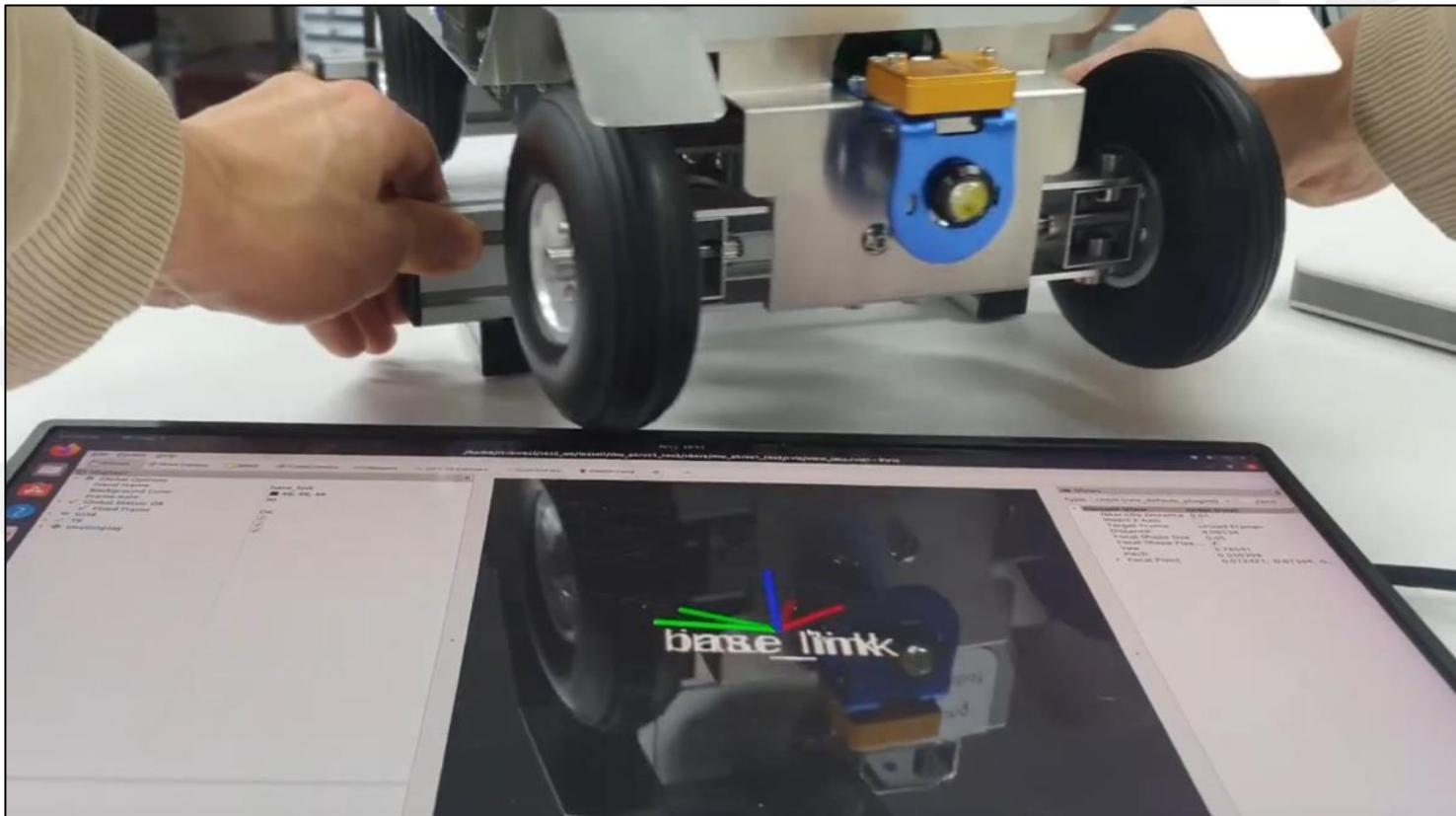


# ROS2 HW

## IMU

demo: MW-AHRSv1

> AHRS(Attitude Heading Reference System)

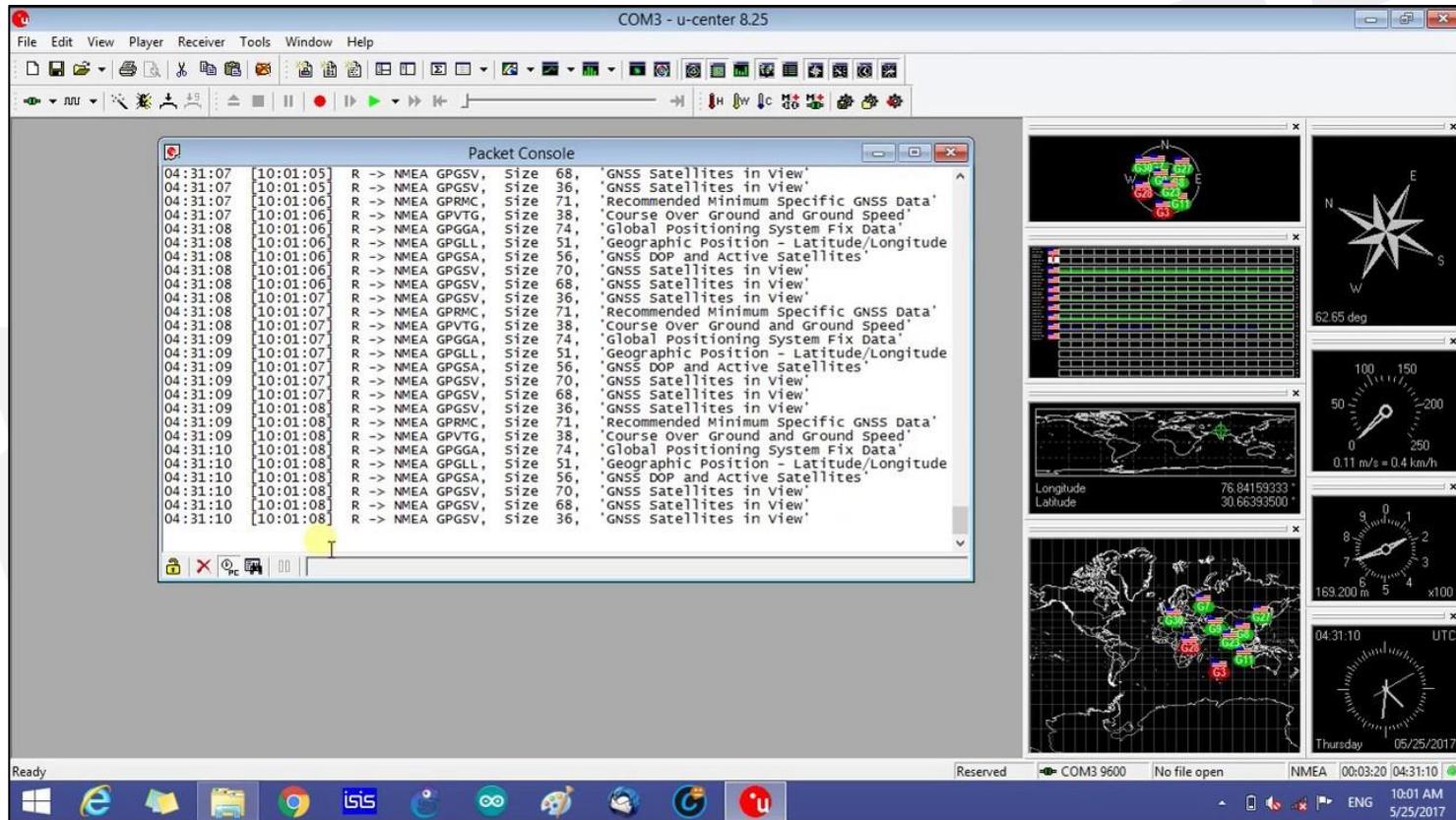


# ROS2 HW

GPS

demo - Ublox F9P, RTK, Ntrip client

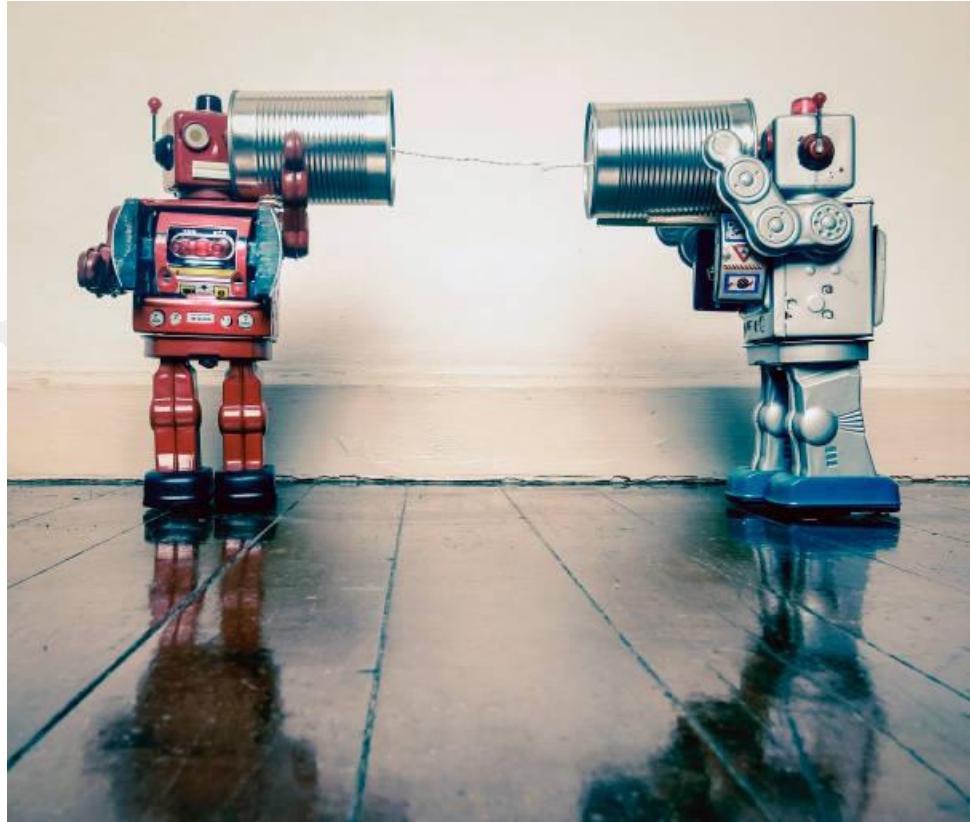
sw: u-center



# ROS2 HW

## CAN (Controller Area Network)

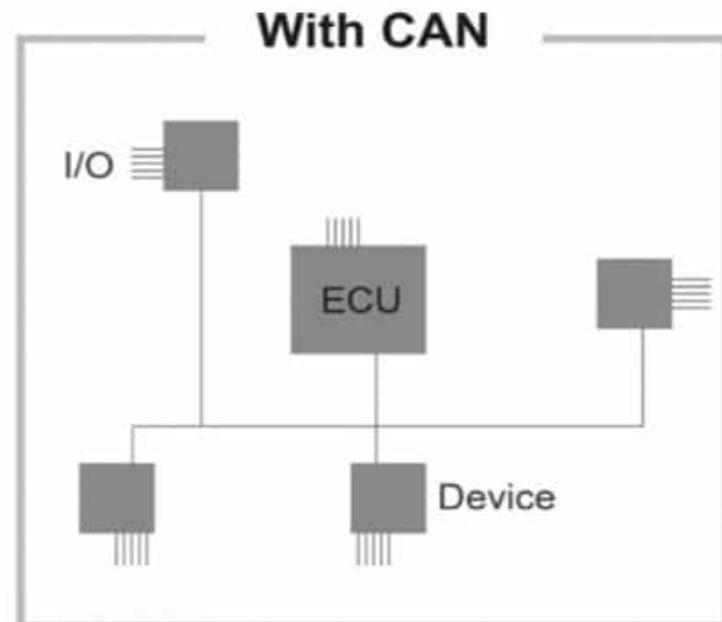
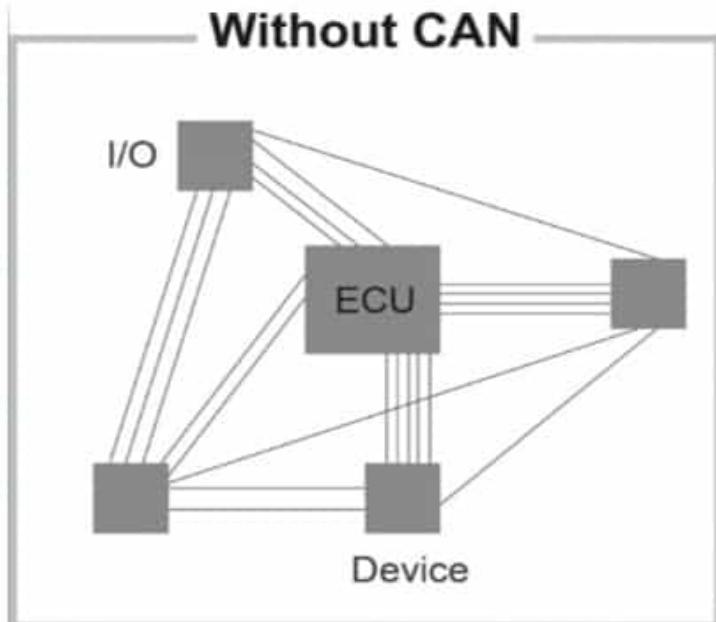
demo - can communication



# ROS2 HW

## CAN

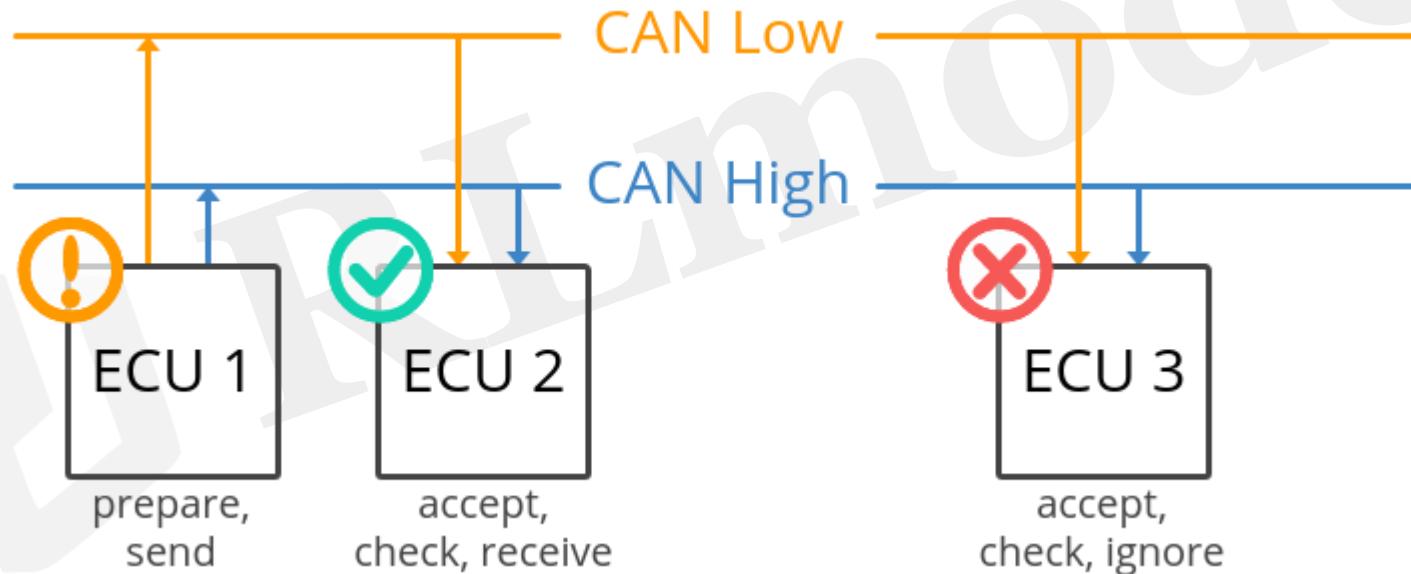
demo - can communication



# ROS2 HW

## CAN

demo - can communication





# CAN command

## CAN device check (USB)

>lsusb



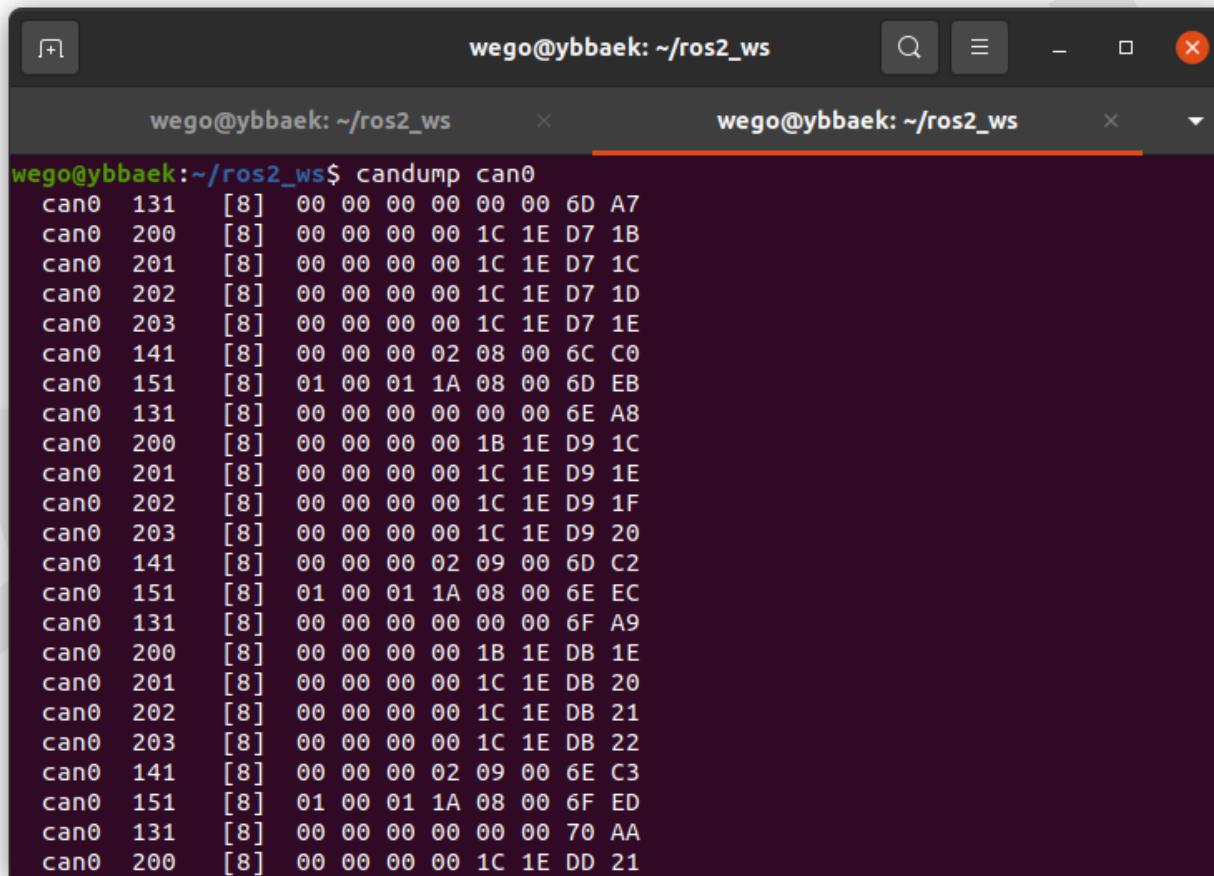
```
rlmodel@rlmodel: ~/rlmodel
rlmodel@rlmodel: ~/rlmodel
rlmodel@rlmodel: ~/rlmodel

rlmodel@rlmodel:~/rlmodel$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 8087:0a2b Intel Corp.
Bus 001 Device 002: ID 046d:c534 Logitech, Inc. Unifying Receiver
Bus 001 Device 011: ID 0c72:000c PEAK System PCAN-USB
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
rlmodel@rlmodel:~/rlmodel$
```

# CAN command

## Ubuntu cmd

data check: \$ candump can0



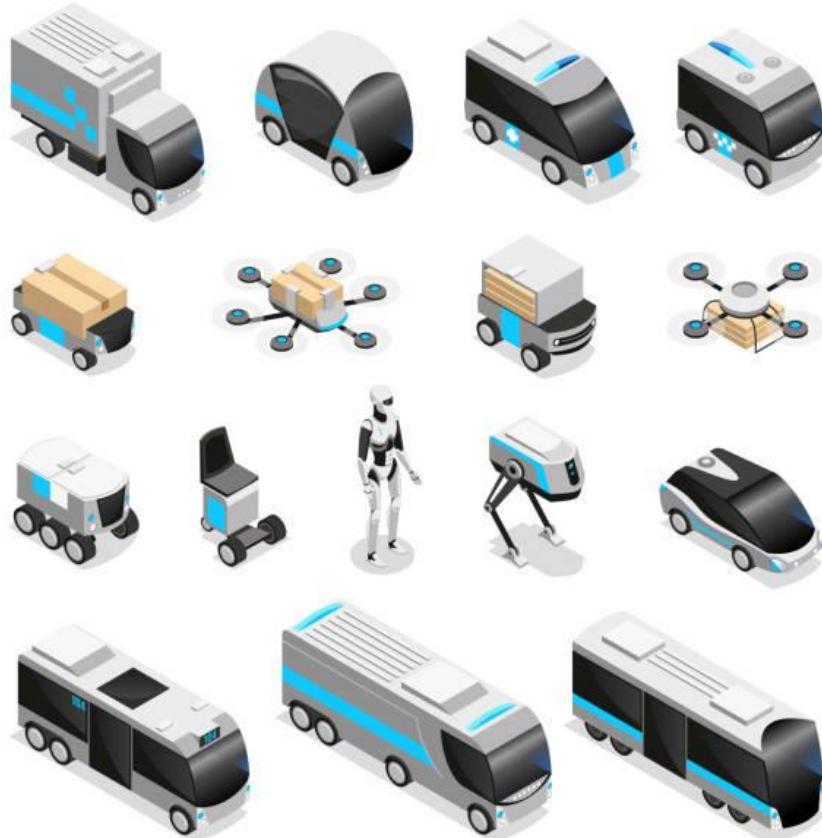
The image shows a terminal window on an Ubuntu system. The title bar says "wego@ybaek: ~/ros2\_ws". The terminal displays the output of the command "candump can0", which lists numerous CAN frames received on the can0 interface. Each frame is represented by a timestamp, a source address (can0), a destination address, and a payload of 8 bytes (hex values). The payloads show various patterns, such as 00 00 00 00 00 00 6D A7, 00 00 00 00 1C 1E D7 1B, etc.

```
wego@ybaek:~/ros2_ws$ candump can0
can0 131 [8] 00 00 00 00 00 00 6D A7
can0 200 [8] 00 00 00 00 1C 1E D7 1B
can0 201 [8] 00 00 00 00 1C 1E D7 1C
can0 202 [8] 00 00 00 00 1C 1E D7 1D
can0 203 [8] 00 00 00 00 1C 1E D7 1E
can0 141 [8] 00 00 00 02 08 00 6C C0
can0 151 [8] 01 00 01 1A 08 00 6D EB
can0 131 [8] 00 00 00 00 00 00 6E A8
can0 200 [8] 00 00 00 00 1B 1E D9 1C
can0 201 [8] 00 00 00 00 1C 1E D9 1E
can0 202 [8] 00 00 00 00 1C 1E D9 1F
can0 203 [8] 00 00 00 00 1C 1E D9 20
can0 141 [8] 00 00 00 02 09 00 6D C2
can0 151 [8] 01 00 01 1A 08 00 6E EC
can0 131 [8] 00 00 00 00 00 00 6F A9
can0 200 [8] 00 00 00 00 1B 1E DB 1E
can0 201 [8] 00 00 00 00 1C 1E DB 20
can0 202 [8] 00 00 00 00 1C 1E DB 21
can0 203 [8] 00 00 00 00 1C 1E DB 22
can0 141 [8] 00 00 00 02 09 00 6E C3
can0 151 [8] 01 00 01 1A 08 00 6F ED
can0 131 [8] 00 00 00 00 00 00 70 AA
can0 200 [8] 00 00 00 00 1C 1E DD 21
```



# Appendix

# 실제 응용사례

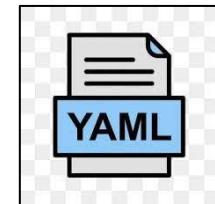




ref

<https://ros2-industrial-workshop.readthedocs.io/en/latest/index.html>

<https://youtu.be/qHr82IPJ-18>



# yaml 파일

## 정의

Yaml의 뜻은 'Yaml Ain't Markup Language' 라는 뜻으로 'YAML은 마크업 언어가 아니다'라는 재귀적 이름

```
1 # This is a comment
2 title: This is some YAML
3 publisher: ConvertSimple Books
4 pages: 250
5 chapters: 21
6 time_to_read: 12 hours
7 descriptors:
8 - fun
9 - entertaining
10 - exciting
11 contributors:
12 author: Mark Templeton
13 editor: Cindy Johnson
```



# yaml 파일

## 비교/특징

- 웹에서 데이터 통신을 위해 JSON을 많이 사용하고, reference를 정의할때 복잡한 object 구조를 표현하기 위해 YAML이 적합

Xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
 <apiVersion>v1</apiVersion>
 <kind>Pod</kind>
 <metadata>
 <name>hello-pod</name>
 <labels>
 <app>hello</app>
 </labels>
 </metadata>
 <spec>
 <containers>
 <name>hello-container</name>
 <image>tmkube/hello</image>
 <ports>
 <containerPort>8000</containerPort>
 </ports>
 </containers>
 </spec>
</root>
```

Json

```
{
 "apiVersion": "v1",
 "kind": "Pod",
 "metadata": {
 "name": "hello-pod",
 "labels": {
 "app": "hello"
 }
 },
 "spec": {
 "containers": [
 {
 "name": "hello-container",
 "image": "tmkube/hello",
 "ports": [
 {
 "containerPort": 8000
 }
]
 }
]
 }
}
```

Yaml

```
apiVersion: v1
kind: Pod
metadata:
 name: hello-pod
 labels:
 app: hello
spec:
 containers:
 - name: hello-container
 image: tmkube/hello
 ports:
 - containerPort: 8000
```



# yaml 파일

## 문법 /

- yaml 파일의 자료형은 스칼라(Scalar), 시퀀스(Sequence), 맵핑(Mapping)으로 구성
- 스칼라(Scalar) 자료형은 문자열(String)이나 숫자(Number)입니다. 문자열은 큰 따옴표("")나 작은 따옴('')로 구분하기도 합니다.
- 맵핑(Mapping) 자료형은 해시(Hash) 또는 딕셔너리(Dictionary) 형식입니다. 콜론(:)으로 키(key)와 값(value)를 구분하며 콜론 뒤에는 공백을 반드시 넣어야합니다. 중괄호 { }를 통해서도 작성 가능합니다. 노드는 2개의 공백이나 4개의 공백으로 구분됩니다.
- 시퀀스(Sequence) 자료형은 배열(array) 또는 리스트(list) 형식입니다. 대시(-)를 통해 표시하며, 대괄호 [ ]를 통해서도 작성 가능합니다.



# Linux 기본 명령어

LINUX  
COMMANDS

CAT  
NETSTAT  
CAT CHMOD  
**SUDO**  
MKDIR  
SSH  
MV  
PASSWD  
LS  
NLLOOKUP



# 리눅스

## Ubuntu 기본명령어

- ls - 현재 위치의 파일 목록 조회
- cd - 디렉토리 이동
- touch - 0바이트 파일 생성, 파일의 날짜와 시간을 수정
- mkdir - 디렉토리 생성
- cp - 파일 복사
- mv - 파일 이동
- rm - 파일 삭제
- cat - 파일의 내용을 화면에 출력, 리다이렉션 기호('>')를 사용하여 새로운 파일 생성
- redirection - 화면의 출력 결과를 파일로 저장
- alias - 자주 사용하는 명령어들을 별명으로 정의하여 쉽게 사용할 수 있도록 설정



# 리눅스

## Ubuntu 기본명령어

### 1. ls (List segments) : 현재 위치의 파일 목록 조회

- ls -l : 파일의 상세정보
- ls -a : 숨김 파일 표시
- ls -t : 파일들을 생성시간순(제일 최신 것부터)으로 표시
- ls -rt : 파일들을 생성시간순(제일 오래된 것부터)으로 표시
- ls -f : 파일 표시 시 마지막 유형에 나타내는 파일명을 끝에 표시  
('/' : 디렉터리, '\*' : 실행파일, '@' : 링크 등등,,,)

### 2. cd (Change directory) : 디렉터리 이동

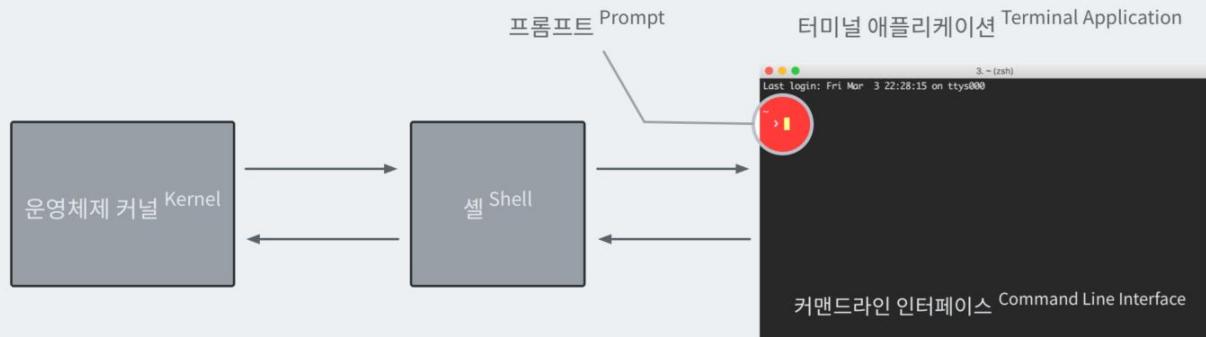
- cd [디렉터리 경로] : 이동하려는 디렉터리로 이동 (경로 입력 시 '[, ]'부분은 빼고 입력!)
- cd ~ : 홈 디렉터리로 이동
- cd / : 최상위 디렉터리로 이동
- cd .. : 현재 디렉터리
- cd ... : 상위 디렉터리로 이동
- cd - : 이전 경로로 이동

# 리눅스

## 터미널(Terminal)

### 리눅스 터미널 프로그램

- 터미널은 GUI에서 실행되는 이전 물리적 터미널의 소프트웨어 표현이다.
- 사용자가 명령을 입력하고 텍스트를 인쇄할 수 있는 인터페이스를 제공한다.
- Linux 서버에 SSH로 연결할 때 로컬 컴퓨터에서 실행하고 명령을 입력하는 프로그램은 터미널이다.

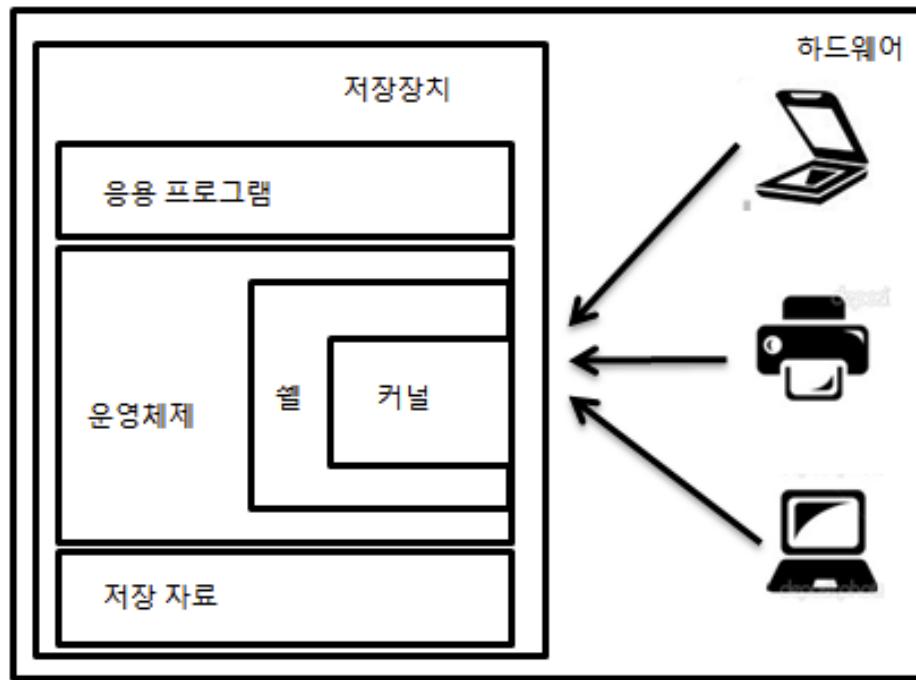


# 리눅스

## 셸(shell)이란

### 1. 기본기능

- 셸(shell)은 터미널 창에 입력한 명령을 해석하여 운영 체제에서 사용자가 원하는 작업을 이해할 수 있도록 하는 프로그램

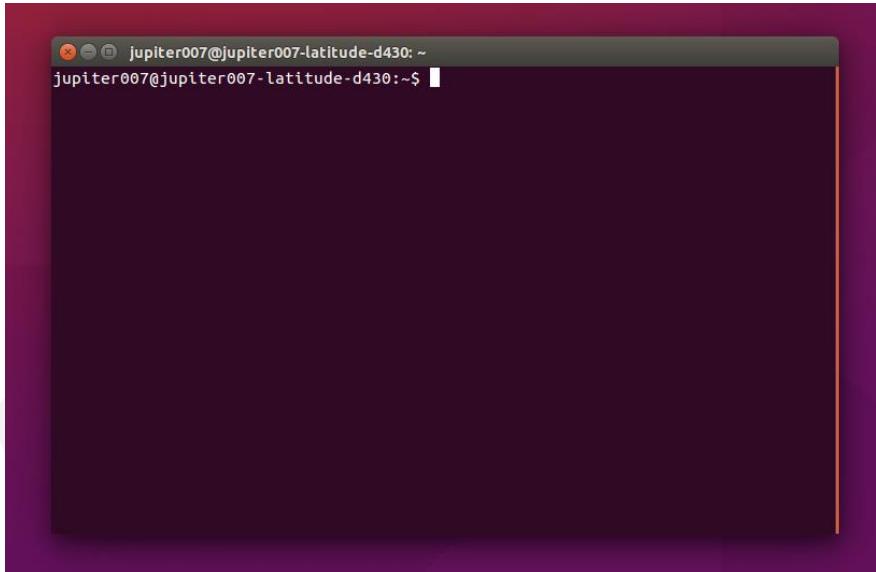


# 리눅스

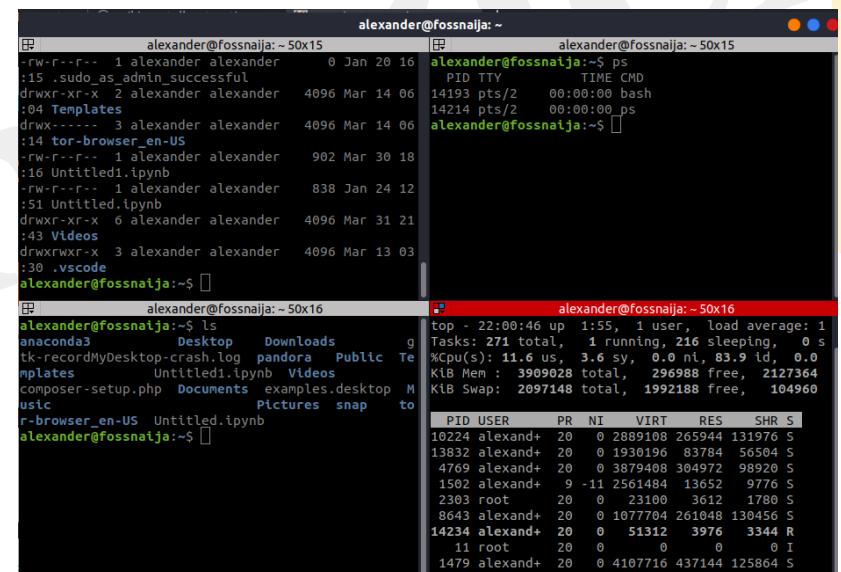
## 터미널(Terminal)

기본/Terminator

실행: Alt + Ctrl + T



기본 터미널

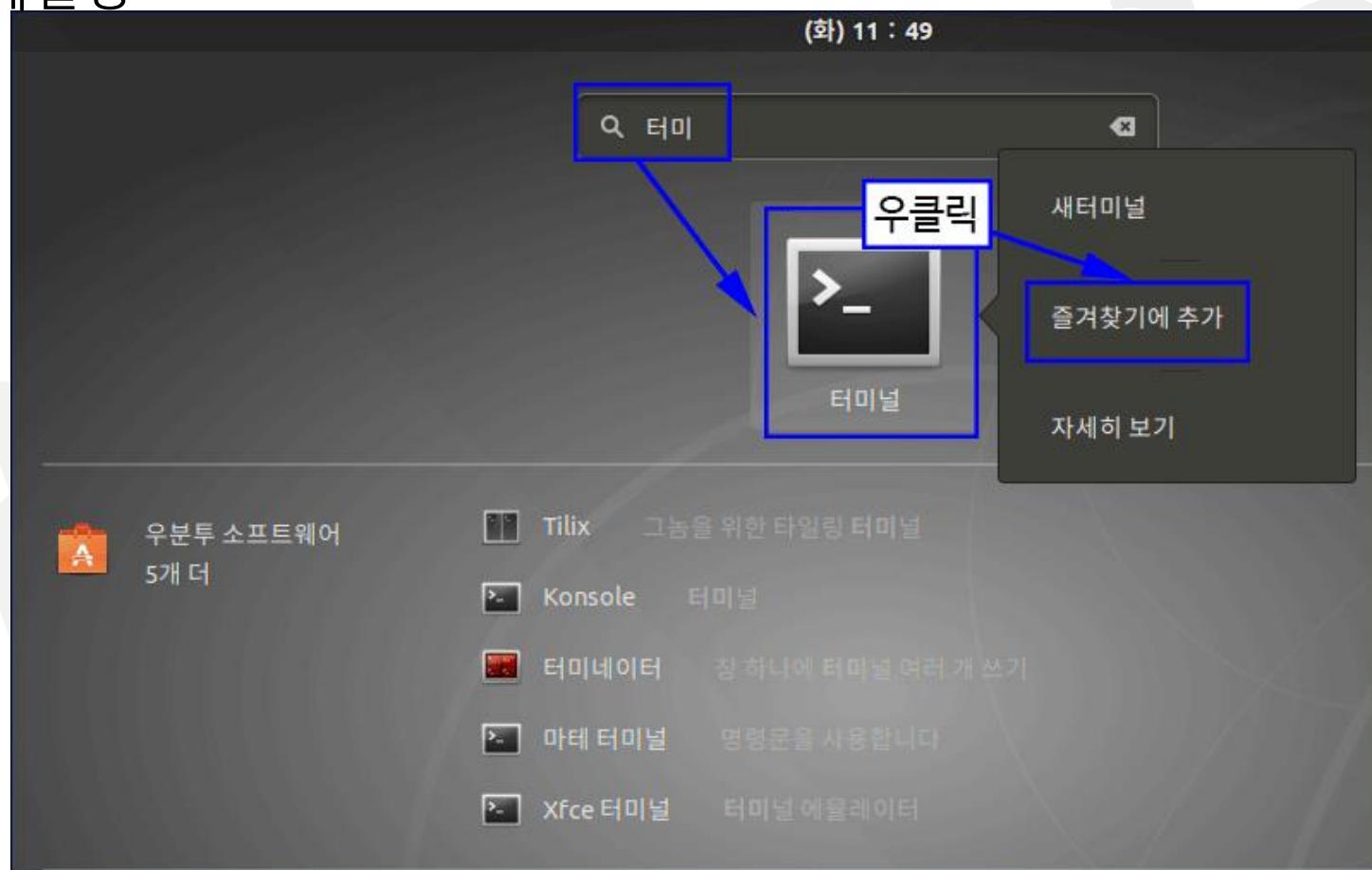


Terminator

# 리눅스

## 터미널(Terminal)

### 상세설정





# 리눅스

## 셸(shell)이란

### 1. 상세기능

- 사용자와 커널 사이에서 명령을 해석해 전달 프로그램
- 쉘은 자체 내에 프로그래밍 기능이 있어서 프로그램을 작성할 수 있음.  
따라서 쉘 프로그래밍 기능을 이용하면 여러 명령을 사용해 반복적으로  
수행하는 작업을 하나의 프로그램으로 제작 할 수 있습니다.  
쉘 프로그래밍을 쉘 스크립트라고 함
- 사용자 환경 설정의 기능 - 초기화 파일 기능을 이용해 사용자의 환경을  
설정할 수 있습니다.



# 리눅스

## bash

### 설명: 현재 리눅스의 표준 셸

- 우분투(Ubuntu) 기본 셸 > bash
- bash 는 리눅스 뿐만 아니라 GNU 운영체제, 맥 OS 등 다양한 운영체제에서 사용 중입니다.

### bash 의 특징

- Alias 기능 (명령어 단축 기능)
- History 기능 (키보드 화살표)
- 연산기능
- Job Control 기능
- 자동 이름완성 기능 (tab)
- 프롬프트 제어 기능
- 명령어 편집기능 등



# 리눅스

## CLI 란 (Command Line Interface)

### 1. 개념

커맨드라인 인터페이스(Command-line Interface, CLI)란 사용자가 텍스트로 명령어를 입력하고 다시 텍스트로 결과를 화면에 출력해주는 인터페이스를 가진 컴퓨팅 인터페이스를 의미

오로지 문자열로만 이루어진 인터페이스를 의미



# 리눅스

## bash 셸 환경변수

변수	내용	변수	내용
HOME	현재 사용자의 홈 디렉터리	PATH	실행 파일을 찾는 디렉터리 경로
LANG	기본 지원되는 언어	PWD	사용자의 현재 작업 디렉터리
TERM	로그인 터미널 타입	SHELL	사용자의 로그인 셸
USER	사용자의 이름	DISPLAY	X윈도에서 프로그램 실행 시 출력되는 창
COLUMNS	현재 터미널의 컬럼 수	LINES	현재 터미널 라인 수
PS1	1차 명령 프롬프트 변수	PS2	2차 프롬프트 변수
BASH	bash 셸의 경로	BASH_VERSION	bash 버전
HISTFILE	히스토리 파일의 절대 경로	HISTSIZE	히스토리 파일에 저장되는 명령어(줄)의 개수
HISTFILESIZE	히스토리 파일의 크기	HOSTNAME	시스템의 호스트명
MAIL	도착한 메일이 저장되는 경로	LOGNAME	로그인 이름
TMOUT	사용자가 로그인 한 후 일정 시간 동안 작업을 하지 않을 경우에 로그아웃시키는 시간으로 단위는 초		
UID	사용자의 UID	OSTYPE	운영체제 타입



# 리눅스

## bash 셸 환경변수

### /etc/profile

시스템 전역 쉘 변수

시스템 사용자 전체에 적용할 쉘 환경 변수를 담고 있으며 보통

경로 환경변수(PATH),

사용자 환경변수(USER),

로그인 사용자(USERNAME),

사용자 메일박스(MAIL),

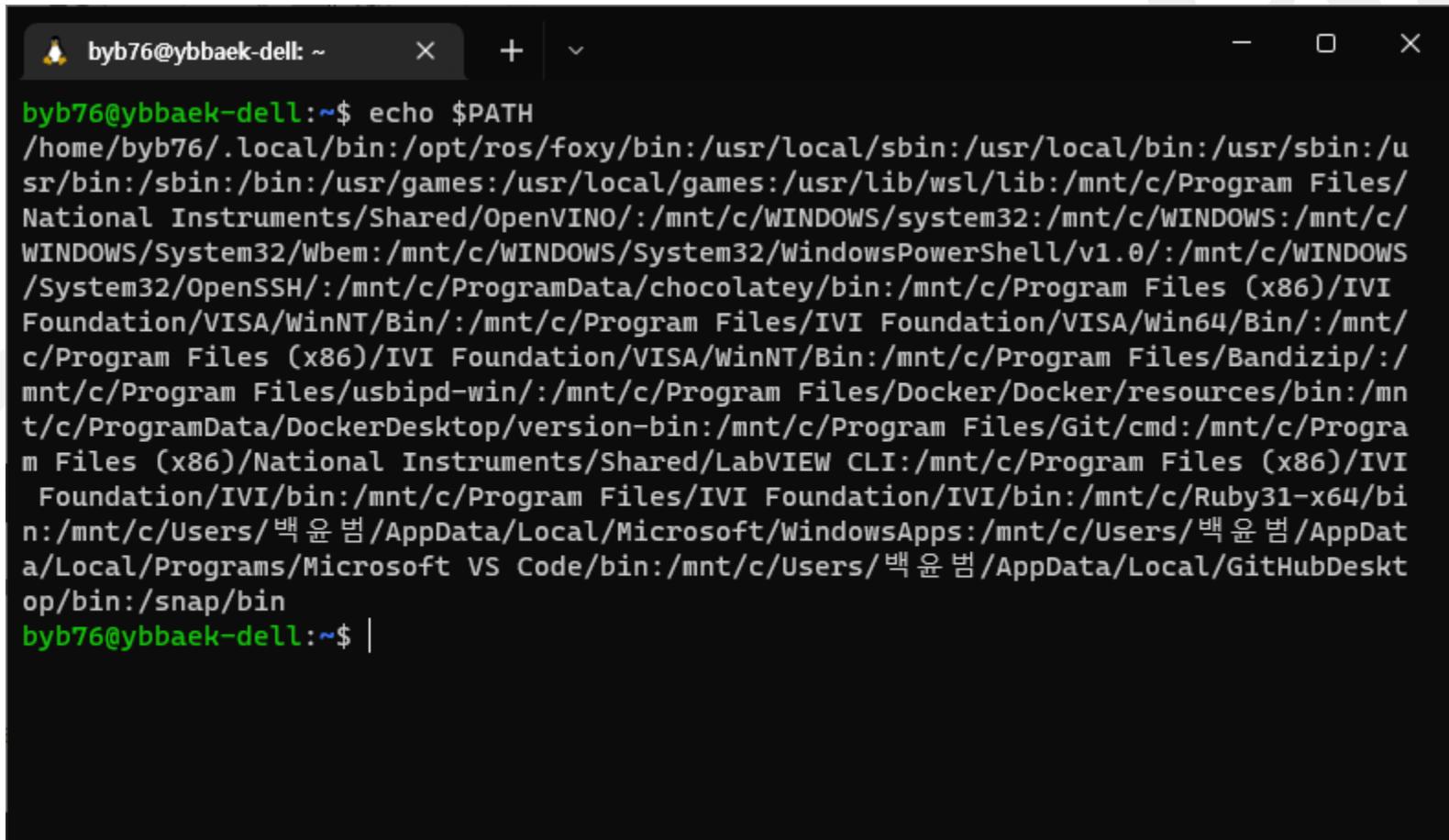
호스트 이름(HOSTNAME),

히스토리(HISTORY)

# 리눅스

## bash 셸 환경변수

- path



```
byb76@ybbaek-dell: ~$ echo $PATH
/home/byb76/.local/bin:/opt/ros/foxy/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/u
sr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/
National Instruments/Shared/OpenVINO/:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/
WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS
/System32/OpenSSH/:/mnt/c/ProgramData/chocolatey/bin:/mnt/c/Program Files (x86)/IVI
Foundation/VISA/WinNT/Bin/:/mnt/c/Program Files/IVI Foundation/VISA/Win64/Bin/:/mnt/
c/Program Files (x86)/IVI Foundation/VISA/WinNT/Bin:/mnt/c/Program Files/Bandizip/:/
mnt/c/Program Files/usbipd-win/:/mnt/c/Program Files/Docker/Docker/resources/bin:/mn
t/c/ProgramData/DockerDesktop/version-bin:/mnt/c/Program Files/Git/cmd:/mnt/c/Progra
m Files (x86)/National Instruments/Shared/LabVIEW CLI:/mnt/c/Program Files (x86)/IVI
Foundation/IVI/bin:/mnt/c/Program Files/IVI Foundation/IVI/bin:/mnt/c/Ruby31-x64/bi
n:/mnt/c/Users/백윤범/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users/백윤범/AppDat
a/Local/Programs/Microsoft VS Code/bin:/mnt/c/Users/백윤범/AppData/Local/GitHubDeskt
op/bin:/snap/bin
byb76@ybbaek-dell: ~$ |
```

# 리눅스

## 셀 확인방법

- echo \$SHELL

```
byb76@ybbaek-dell:~/ros2_foxy$ echo $SHELL
/bin/bash
byb76@ybbaek-dell:~/ros2_foxy$ |
```



**BASH**  
THE BOURNE-AGAIN SHELL



# 리눅스

## **bashrc**

- Bash Bourn Again Shell widely used shell
  - >Next 5 configuration file

/etc/profile  
/etc/bashrc  
~/.bash\_profile  
~/.bashrc  
~/.bash\_logout

Global configuration: /etc/profile, /etc/bashrc  
Local configuration: ~/.bashrc, ~/.bash\_profile

rc mean: run command, run configuration



# 리눅스

## bashrc

- .bashrc
- 일반적으로 전역적 파일은 /etc 디렉토리에 위치  
~/.bashrc > 별칭(alias) 과 bash가 수행될 때 실행되는 함수를  
제어하는 지역적인 시스템 설정과 관련된 파일이다.  
숨김파일은 .으로 시작

## /etc/profile

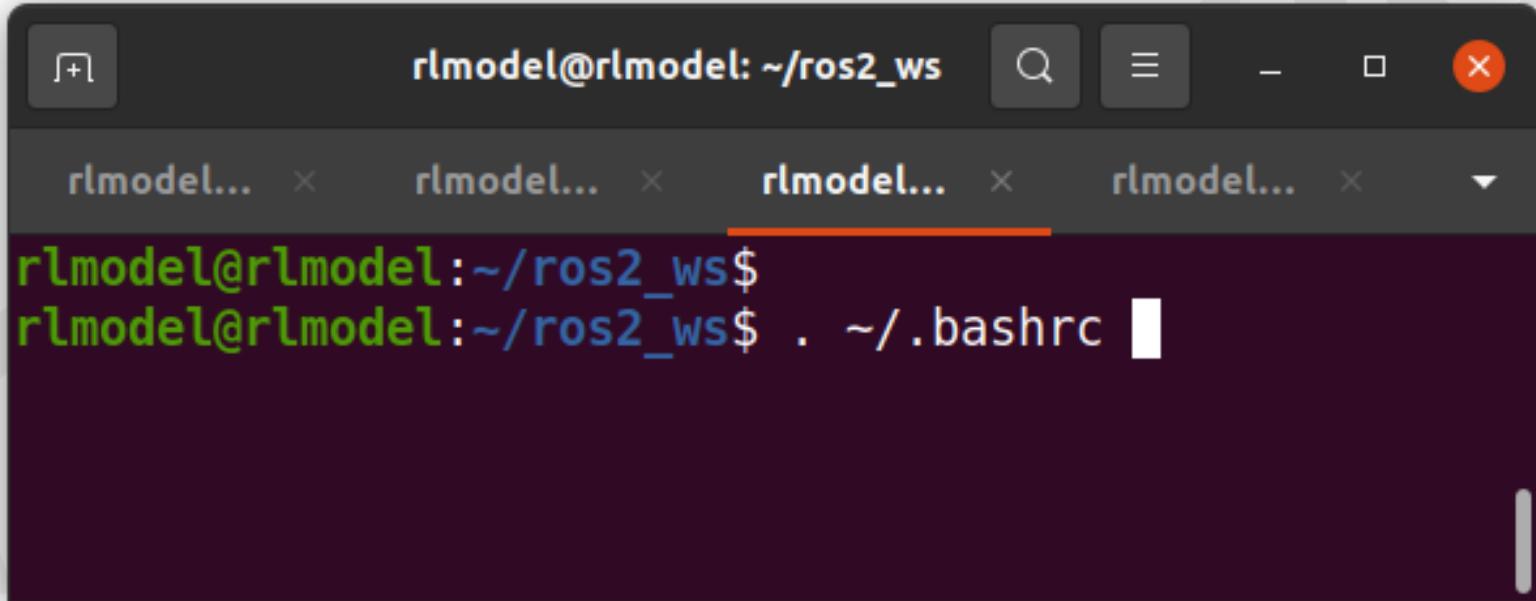
>은 환경변수와 bash 가 수행될 때 실행되는 프로그램을 제어하는  
전역적인 시스템 설정과 관련된 파일

# 리눅스

## bashrc

.bashrc

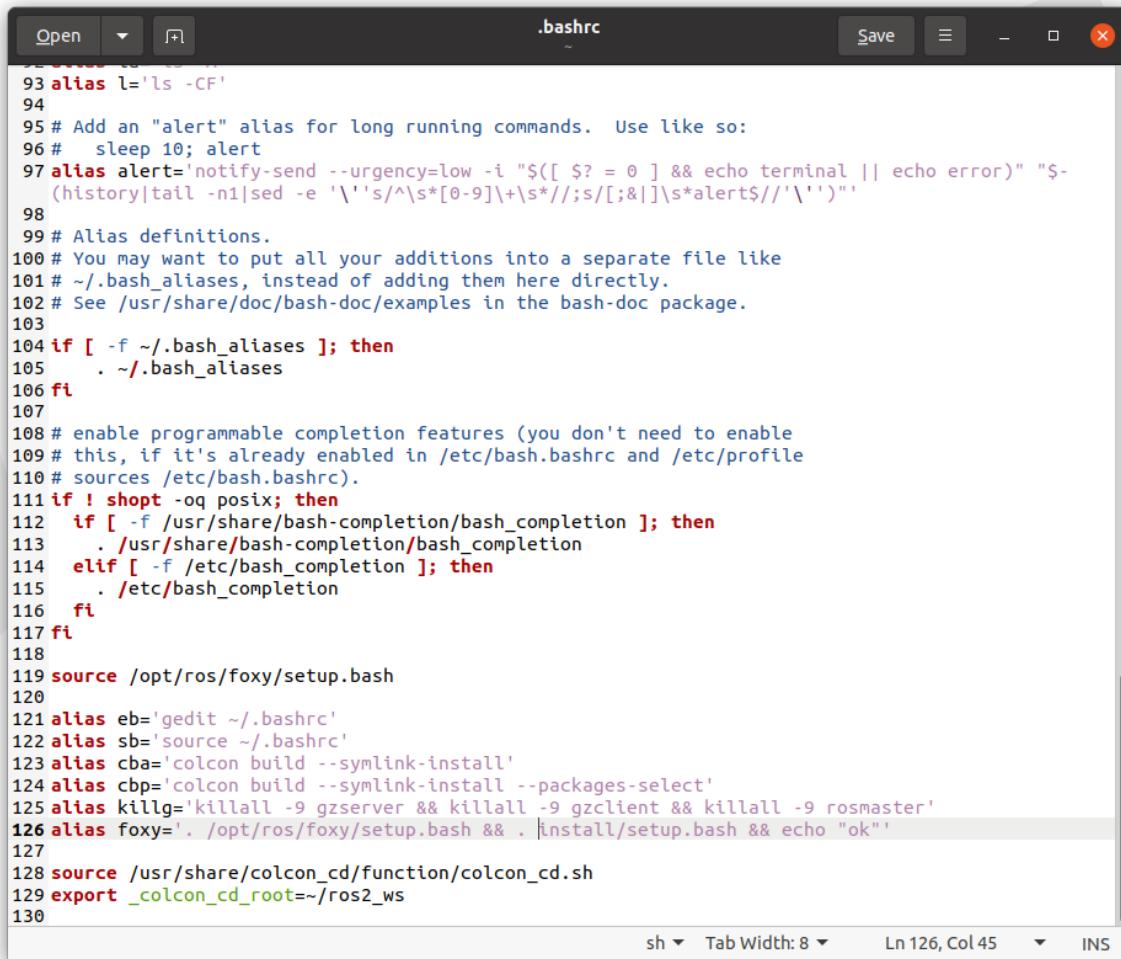
>Run: . ~./.bashrc



A screenshot of a terminal window titled "rlmodel@rlmodel: ~/ros2\_ws". The window has four tabs, all labeled "rlmodel...". The active tab shows the command "rlmodel@rlmodel:~/ros2\_ws\$ . ~/.bashrc" being typed. The terminal has a dark background and light-colored text.

## bashrc

>alias: short key



The screenshot shows a terminal window with the title ".bashrc". The window contains the content of the .bashrc file, which is a shell script. The script includes several alias definitions, such as 'ls' and 'alert', and source statements for completion files and a ROS setup script. The code editor interface includes tabs for 'Open', 'Save', and 'Close', and status bars at the bottom showing 'Tab Width: 8', 'Ln 126, Col 45', and 'INS'.

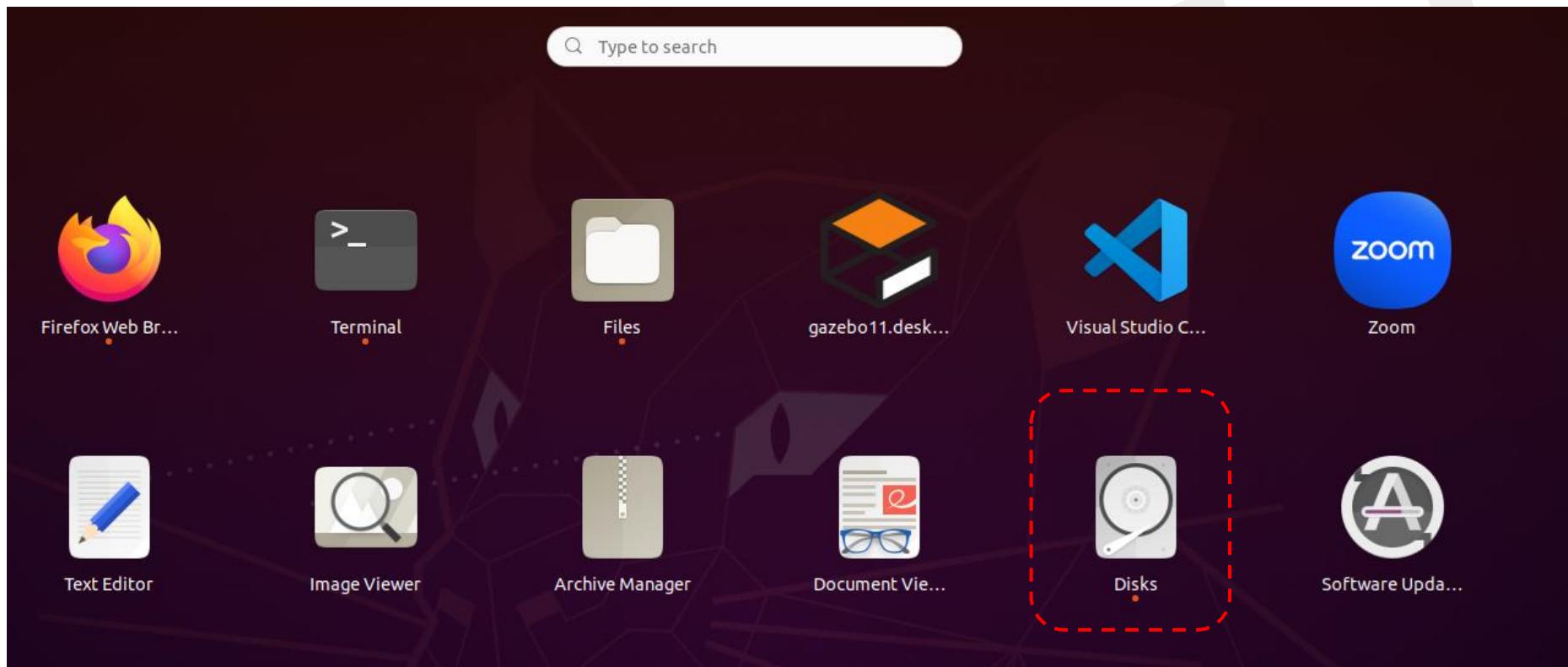
```
93 alias l='ls -CF'
94
95 # Add an "alert" alias for long running commands. Use like so:
96 # sleep 10; alert
97 alias alert='notify-send --urgency=low -i "$([$? = 0] && echo terminal || echo error)" "$_
 (history|tail -n1|sed -e '\''$s/^\\s*[0-9]\\+\\s*//;s/[;&]$/\\s*alert$/\\''")"
98
99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [-f ~/.bash_aliases]; then
105 . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112 if [-f /usr/share/bash-completion/bash_completion]; then
113 . /usr/share/bash-completion/bash_completion
114 elif [-f /etc/bash_completion]; then
115 . /etc/bash_completion
116 fi
117 fi
118
119 source /opt/ros/foxy/setup.bash
120
121 alias eb='gedit ~/.bashrc'
122 alias sb='source ~/.bashrc'
123 alias cba='colcon build --symlink-install'
124 alias cbp='colcon build --symlink-install --packages-select'
125 alias killg='killall -9 gzserver && killall -9 gzclient && killall -9 rosmaster'
126 alias foxy='. /opt/ros/foxy/setup.bash && . /install/setup.bash && echo "ok"'
127
128 source /usr/share/colcon_cd/function/colcon_cd.sh
129 export _colcon_cd_root=~/ros2_ws
130
```



# 리눅스

## HDD disk usage

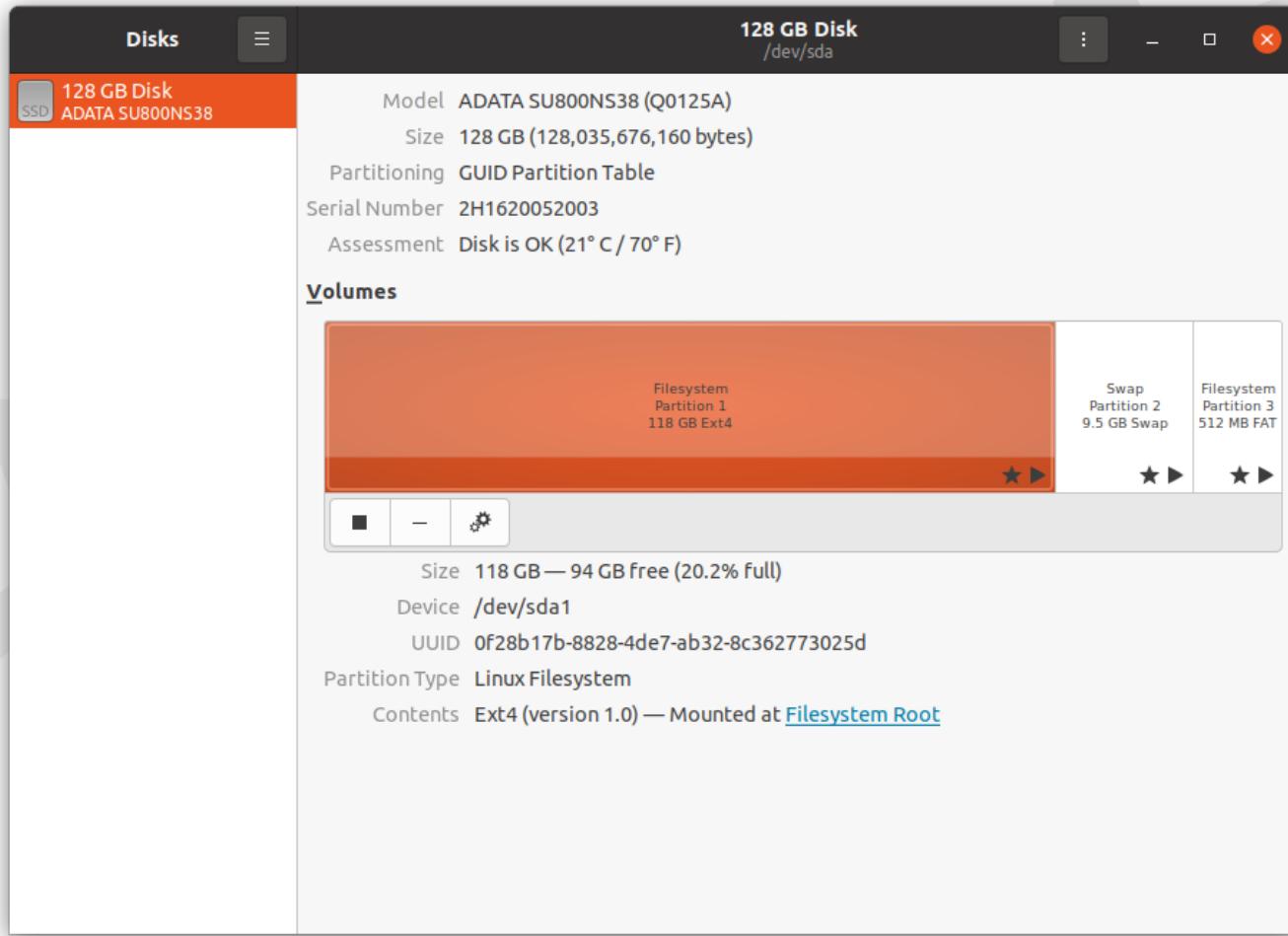
>DISK



# 리눅스

## HDD disk usage

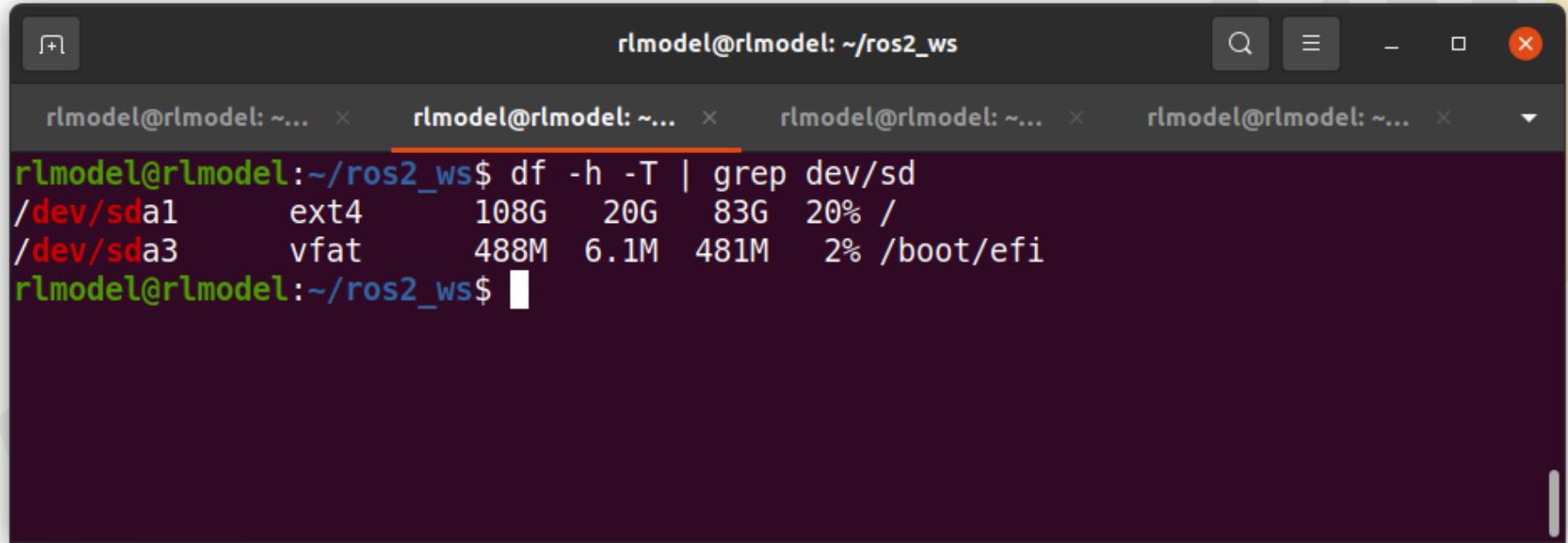
>DISK



# 리눅스

## HDD disk usage

>DISK: df -h -T | grep dev/sd



A screenshot of a Linux terminal window titled "rlmodel@rlmodel: ~/ros2\_ws". The window contains four tabs, all showing the same command output. The command run is "df -h -T | grep dev/sd". The output shows two partitions: "/dev/sda1" (ext4, 108G total, 20G used) and "/dev/sda3" (vfat, 488M total, 6.1M used). The terminal has a dark background and light-colored text.

```
rlmodel@rlmodel:~/ros2_ws$ df -h -T | grep dev/sd
/dev/sda1 ext4 108G 20G 83G 20% /
/dev/sda3 vfat 488M 6.1M 481M 2% /boot/efi
rlmodel@rlmodel:~/ros2_ws$
```



# ROS2 info

## ROS2 Industrial

<https://rosindustrial.org/events/2023/5/ros-industrial-consortium-americas-2023-annual-meeting>

The screenshot shows the ROS-Industrial website with the URL <https://rosindustrial.org/events/2023/5/ros-industrial-consortium-americas-2023-annual-meeting> in the browser's address bar. The page features a blue header with the ROS-Industrial logo and navigation links for About, Blog, Consortium, Developer, Events, Tutorials, and Videos. Below the header, there are sections for the event, including a "Event Program Available Here!" link and a "Printable Agenda Here!" link. A collage of images shows industrial robots and people at a conference. To the right, there is a sidebar with information about the ROS-Industrial project, upcoming events (including the ROS-Industrial Conference 2023 and ROS-Industrial Developers Meeting), and ROS-Industrial Training.

**ROS-Industrial Consortium Americas 2023 Annual Meeting**

Wednesday, May 24, 2023, 6:30 PM – Thursday, May 25, 2023, 5:00 PM

TCF Center  
1 Washington Boulevard, Detroit, MI, 48226, United States ([map](#))

[Google Calendar](#) · [ICS](#)

**Event Program Available Here!**

**Printable Agenda Here!**

The ROS-Industrial Consortium Americas will host its annual meeting in conjunction with the Automate 2023 show in Detroit. The annual Members' Meeting is a chance to understand the latest developments relative to ROS-Industrial, both within the Americas and around the world. There will be a chance to

**ROS-INDUSTRIAL**

ROS-Industrial is an open source project that extends the advanced capabilities of the [Robot Operating System](#) (ROS) software to manufacturing.

**Upcoming Events**

**ROS-Industrial Conference 2023**  
Jul 4, 2023 – Jul 5, 2023

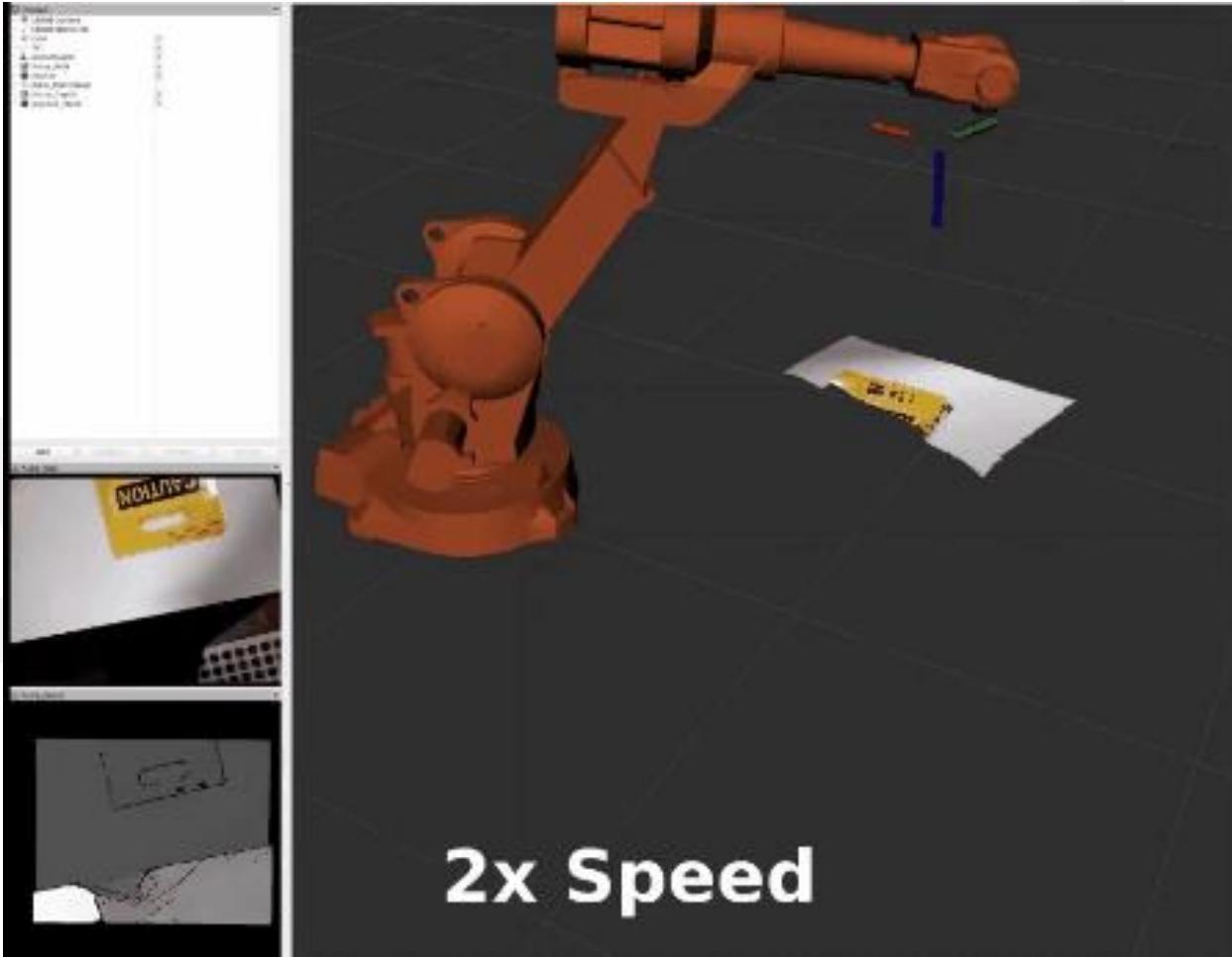
**ROS-Industrial Developers Meeting (Jul) - 2023 (Americas)**  
Jul 11, 2023

**ROS-Industrial Training (Americas) July 2023**

# ROS2 info

## ROS2 Industrial

<https://rosindustrial.org/events/2023/5/ros-industrial-consortium-americas-2023-annual-meeting>





# DDS 특징

## RTPS

### 1. RTPS(Real Time Publish Subscribe)란

DDS의 표준 wire-protocol로 선정.

산업자동화 protocol. (산업용 디바이스에 많이 사용되고 있는 검증된 방식)

QoS 설정: best-effort와 reliable publish subscribe 통신 선택 가능.

Fault tolerance: single points failure에도 네트워크 구성 가능.

Extensibility: 프로토콜을 새로운 서비스로 확장시킬 수 있음.

Plug-and-play: 새로운 application이나 service가 자동으로 discover되고 연결됨.

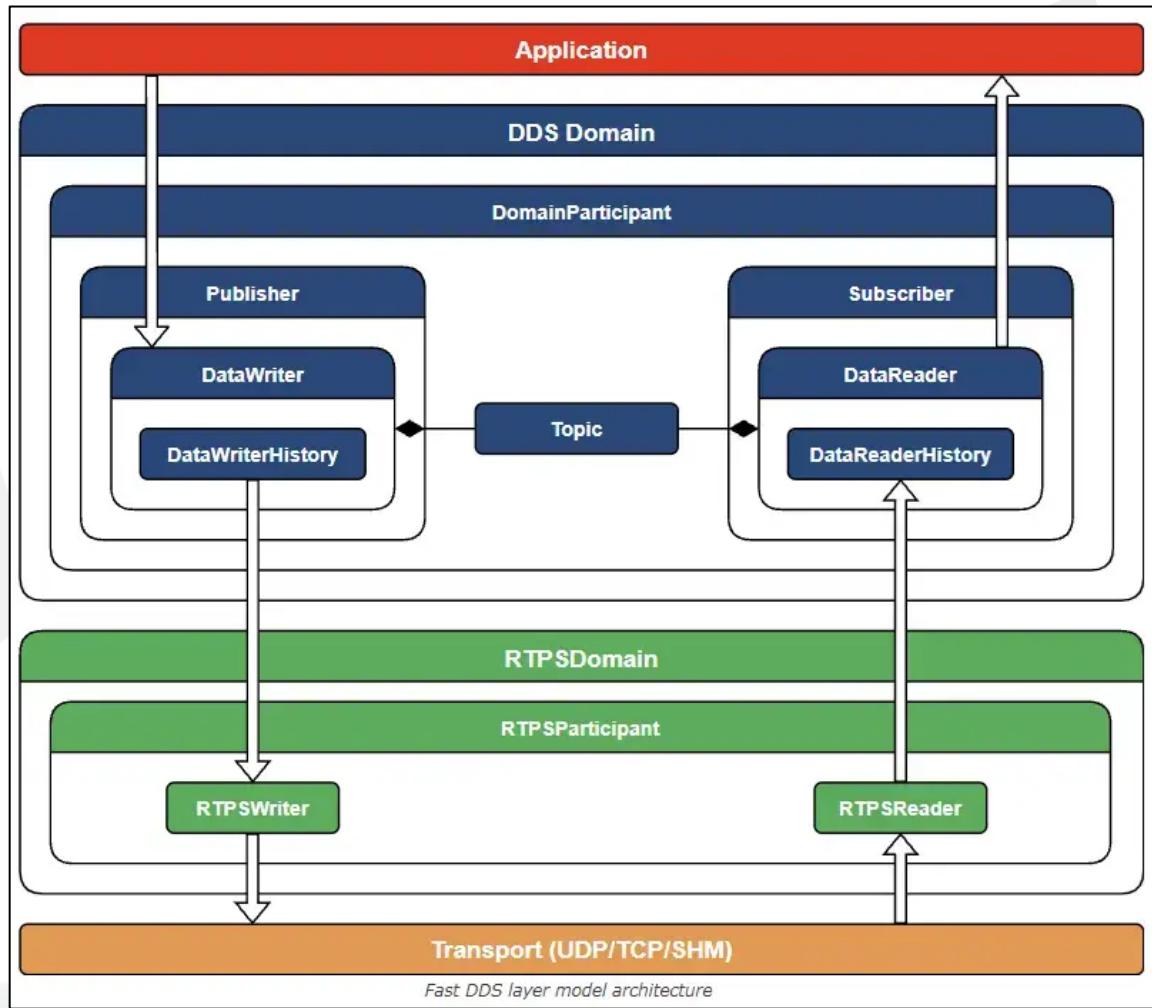
Scalability: 매우 큰 네트워크로 시스템을 확장할 수 있음.

Type-safety: programming 에러가 remote 노드들을 손상시키지 못하게 함.

# DDS 특징

## RTPS 구조 (Realtime Publisher Subscribe)

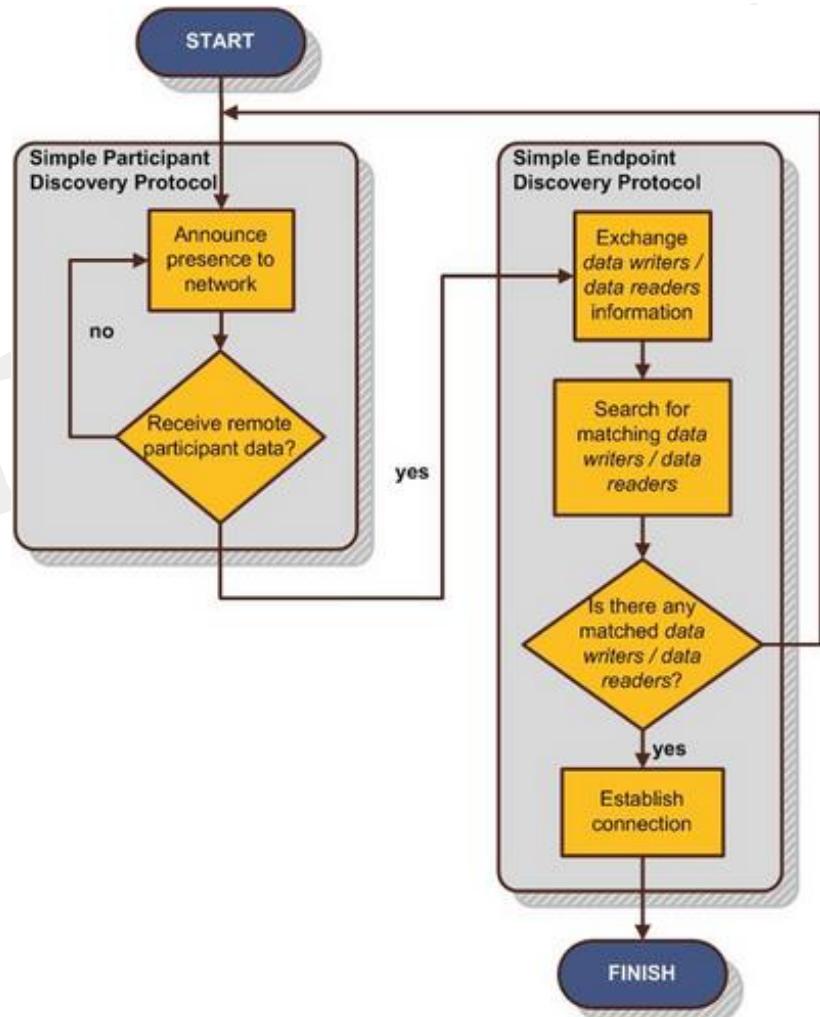
### Block Diagram



# DDS 특징

## PDP, EDP

- Auto discovery
- PDP (Participant Discovery Protocol)
- EDP (Endpoint Discovery Protocol)



# DDS 특징

## IDL Interface Definition Language

### 1. 설명

소프트웨어 컴포넌트의 인터페이스를 묘사하기 위한 명세 언어.

IDL은 어느 한 언어에 국한되지 않는 언어중립적인 방법으로 인터페이스를 표현함으로써, 같은 언어를 사용하지 않는 소프트웨어 컴포넌트 사이의 통신을 가능하게 한다.

```
struct ShapeType {
 string<128> color; // @key
 //@ID 0
 long x; // @ID 1
 long y; // @ID 2
 long shapesize; // @ID 3
};

// @Extensibility EXTENSIBLE_EXTENSIBILITY

enum ShapeFillKind {
 @default_literal
 SOLID_FILL = 0,
 TRANSPARENT_FILL = 1,
 HORIZONTAL_HATCH_FILL = 2,
 VERTICAL_HATCH_FILL = 3
};

Type tree | Equivalent IDL
```

Description Panel  
System > Host : 10.0.2.15 > Process : 2160 > DP : 64 >  
Subscriber > DR { Example ComplexType }

Type Name:	ComplexType
Type ID:	
Typecode Serialized Size:	3,934
Minimum Serialized Size:	276
Maximum Serialized Size:	49,940
Maximum Key Serialized Size:	0

IDL Representation

```
struct NestedType{
 short sensor_group_id; // @key
 string<64> sensor_group_name;
 float temperature;
 short temperature_units;
 float light_exposure;
 float pressure;
 long c_seq;
 sequence<octet,8192> payload;
};

// @Extensibility EXTENSIBLE_EXTENSIBILITY

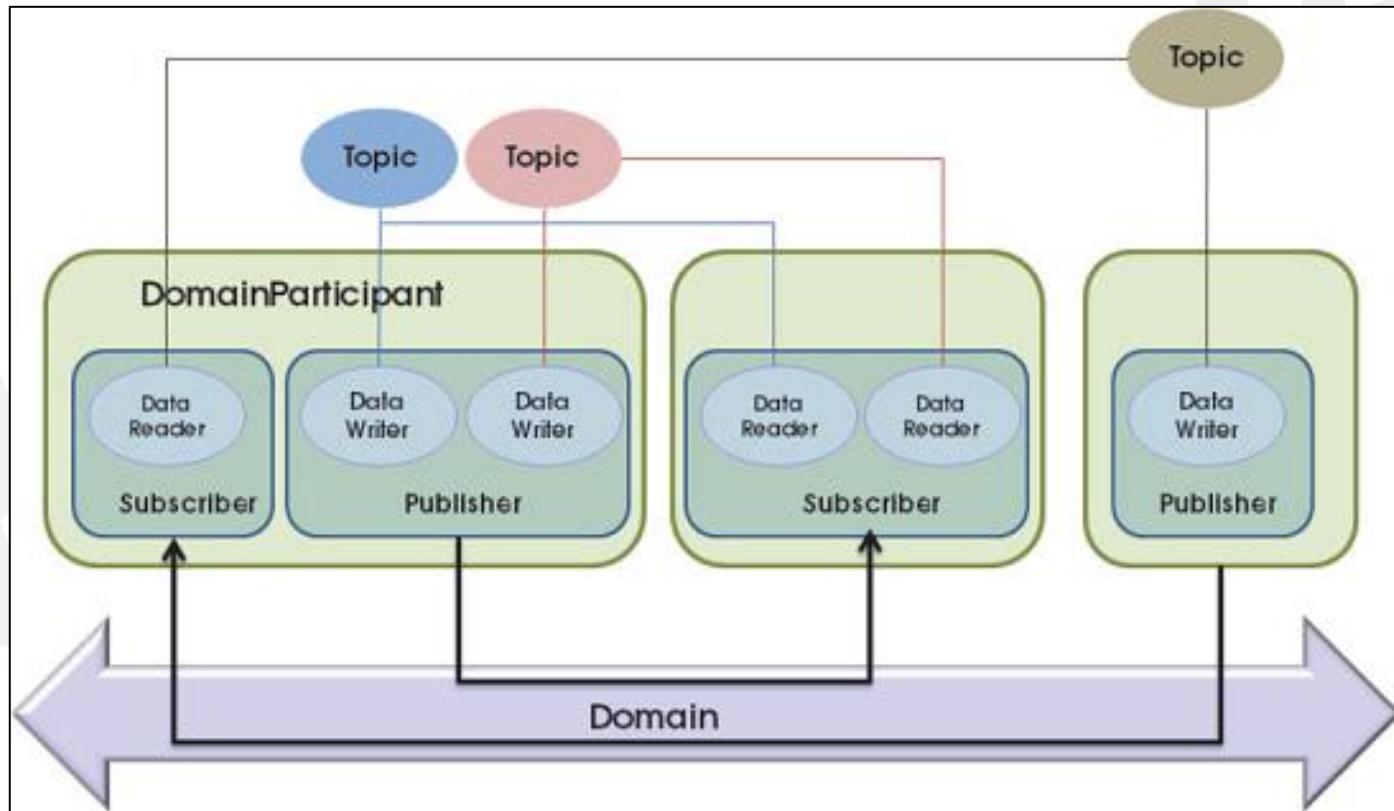
enum NestedEnum {
 ENUM00 = 0,
 ENUM01 = 1,
 ENUM02 = 2,
 ENUM03 = 3,
 ENUM04 = 4,
 ENUM05 = 5,
 ENUM06 = 6,
 ENUM07 = 7,
 ENUM08 = 8,
 ENUM09 = 9
};

// @Extensibility EXTENSIBLE_EXTENSIBILITY
```

# DDS 특징

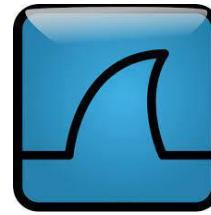
## Domain ID

1. 자동으로 연결할 대상을 찾아주는 Auto-Discovery 기능





# Wireshark



## 특징/장점

기능: 자유 및 오픈 소스 패킷 분석 프로그램이다.

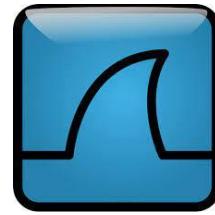
네트워크의 문제, 분석, 소프트웨어 및 통신 프로토콜 개발, 교육에 쓰임

The screenshot shows the Wireshark interface with the following details:

- Title Bar:** NVIDIA nForce MCP Networking Adapter Driver | Wireshark 1.6.5 (SVN Rev 40429 from /trunk...)
- Menu Bar:** File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, Help
- Toolbar:** Includes icons for opening files, saving, zooming, and capturing.
- Filter Bar:** Contains a "Filter:" input field, an "Expression..." button, a "Clear" button, and an "Apply" button.
- Packet List:** A table showing captured packets. The first few rows are:
  - No. 2443 Time 21.830746 Source 175.143.151.240 Destination 192.168.1.64 Protocol UDP Length 9:
  - 2444 21.830820 192.168.1.64 02dd8511.bb.sky.com BitTorr... 12:
  - 2445 21.834295 02dd8511.bb.sky.com 192.168.1.64 UDP 7:
  - 2446 21.834478 192.168.1.64 02dd8511.bb.sky.com UDP 130:
  - 2447 21.836583 02dbfb66.bb.sky.com 192.168.1.64 UDP 7:
  - 2448 21.836767 192.168.1.64 02dbfb66.bb.sky.com UDP 130:
  - 2449 21.837124 02dbfb66.bb.sky.com 192.168.1.64 TCP 60:
  - 2450 21.837186 192.168.1.64 02dbfb66.bb.sky.com TCP 54:
  - 2451 21.837313 192.168.1.64 02dbfb66.bb.sky.com BitTorr... 12:
  - 2452 21.870317 111.122.113.202 192.168.1.64 TCP 60:
- Details Panel:** Shows expanded details for the selected packet (Frame 6).
  - Frame 6: 1163 bytes on wire (9304 bits), 1163 bytes captured (9304 bits)
  - Ethernet II, Src: Msi\_74:82:e6 (00:16:17:74:82:e6), Dst: Actionte\_d8:a3:88
  - Internet Protocol Version 4, Src: 192.168.1.64 (192.168.1.64), Dst: sea09s
  - Transmission Control Protocol, Src Port: 55684 (55684), Dst Port: https (443)
- Hex Editor:** Shows the raw hex and ASCII representation of the selected frame.
- Status Bar:** File: "C:\Users\Chris\AppData\Local\Temp\...", Packets: 9476 Displayed..., Profile: Default



# Wireshark



## RTPS protocol

### Data packet check

\*Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Expression... +

No. Time Source Destination Protocol Length Info

1074	41.528624609	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1075	41.528639227	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1076	41.528679007	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1077	41.528735794	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1078	41.528756087	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1079	41.528757316	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1080	41.528779897	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1081	41.528779794	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1082	41.528815853	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1083	41.528826291	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, ACKNACK
1084	41.528837366	127.0.0.1	127.0.0.1	RTPS	766	INFO_TS, INFO_DST, DATA(w), HEARTBEAT
1085	41.528839021	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1086	41.528846454	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1087	41.530966208	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK
1088	41.531096939	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1089	41.531123289	127.0.0.1	127.0.0.1	RTPS	110	INFO_DST, HEARTBEAT
1090	41.531126372	127.0.0.1	127.0.0.1	RTPS	106	INFO_DST, ACKNACK

▶ Frame 1084: 766 bytes on wire (6128 bits), 766 bytes captured (6128 bits) on interface 0  
▶ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
▶ User Datagram Protocol, Src Port: 57724, Dst Port: 7412  
▶ Real-Time Publish-Subscribe Wire Protocol  
    Magic: RTPS  
    Protocol version: 2.1  
    vendorId: 01.01 (Real-Time Innovations, Inc. - Connext DDS)  
    guidPrefix: 0a1e01610000173200000001  
    Default port mapping: domainId=0, participantIdx=1, nature=UNICAST\_METATRAFFIC  
    submessageId: INFO\_TS (0x09)  
        Flags: 0x01, Endianness bit  
        octetsToNextHeader: 8  
        Timestamp: Dec 14, 2016 18:48:17.146668000 UTC  
    submessageId: INFO\_DST (0x0e)  
        Flags: 0x01, Endianness bit  
        octetsToNextHeader: 12  
        guidPrefix: 0a1e0161000017d900000001  
        submessageId: DATA (0x15)  
        submessageId: HFARTBEAT (0x07)

0070 00 00 00 03 00 00 5a 00 10 00 0a 1e 01 61 00 00 .Z. ....a..  
0080 17 32 00 00 00 01 80 00 00 03 05 00 10 00 0a 00 .2. ....  
0090 00 00 50 69 6e 67 54 6f 70 69 63 00 00 00 07 00 ..PingTo pic....  
00a0 10 00 09 00 00 50 69 6e 67 54 79 70 65 00 00 .....P1 ngType..  
00b0 00 00 1a 00 0c 00 01 00 00 00 00 00 00 00 99 99 .....  
00c0 99 19 0b 80 04 00 00 00 00 00 10 00 04 00 01 00 .....  
00d0 00 00 13 00 04 00 ff ff ff ff 06 00 04 00 00 00 .....  
Text item (text), 20 bytes

Packets: 1937 · Displayed: 493 (25.5%) · Profile: Default

# VMware

## HW settings

Host / VMware: Disconnect, Connect

