Week5

태그	
내용	

17장 ROS2 도구와 CLI 명령어

ROS를 사용하는 가장 큰 목적 중 하나는 다양한 개발 도구입니다. ROS 도구는 4가지로 분류할 수 있습니다.
1. CLI 형태의 Command-Line Tools
2. GUI 형태의 RQT
3. 3차원 시각화툴인 RViz
4. 3차원 시뮬레이터 Gazebo

CLI 기반 Command-Line Tools

- 명령어 기반의 도구를 사용하여 ROS의 거의 모든 기능을 사용할 수 있습니다.
- 개발 환경, 빌드, 테스트 도구인 colcon과 데이터 녹화, 재생, 관리 도구인 ros2bag을 포함하여 20여 가지 도구를 제공합니다.

ros2cli + [verbs]	**[arguments]**	**기능**
ros2 run	<package> <executable>	특정 패키지의 특정 **노드 실행** (1개의 노드) * executable에 따라 복수 노드도 실행 가능
ros2 launch	<package> <launch-file>	특정 패키지의 특정 **런치 파일 실행** (0개~복수개의 노드)

다음은 CLI(Command-Line Interface) 기반의 Command-Line Tools 예제입니다.

```
$ ros2 run turtlesim turtlesim_node

$ ros2 launch demo_nodes_cpp talker_listener.launch.py
```

ROS 2 정보 명령어

ros2cli + [verbs]	**[sub-verbs]**	**기능**
ros2 pkg	create executables list prefix xml	새로운 ROS 2 패키지 생성 지정 패키지의 실행 파일 목록 출력 사용 가능한 패키지 목록 출력 지정 패키지의 저장 위치 출력 지정 패키지의 패키지 정보 파일(xml) 출력

ros2cli + [verbs]	[sub-verbs]	기능
ros2 node	info list	실행 중인 노드 중 지정한 노드의 정보 출력 실행 중인 모든 노드의 목록 출력
ros2 topic	bw delay echo find hz info list pub type	지정 토픽의 대역폭 측정 지정 토픽의 지연시간 측정 지정 토픽의 데이터 출력 지정 타입을 사용하는 토픽 이름 출력 지정 토픽의 주기 측정 지정 토픽의 정보 출력 사용 가능한 토픽 목록 출력 지정 토픽의 토픽 발행 지정 토픽의 토픽 타입 출력
ros2 service	call find list type	지정 서비스의 서비스 요청 전달 지정 서비스 타입의 서비스 출력 사용 가능한 서비스 목록 출력 지정 서비스의 타입 출력
ros2 action	info list send_goal	지정 액션의 정보 출력 사용 가능한 액션 목록 출력 지정 액션의 액션 목표 전송
ros2 interface	list package packages proto show	사용 가능한 모든 인터페이스 목록 출력 특정 패키지에서 사용 가능한 인터페이스 목록 출력 인터페이스 패키지들의 목록 출력 지정 패키지의 프로토타입 출력 지정 인터페이스의 데이터 형태 출력
ros2 param	delete describe dump get list set	지정 파라미터 삭제 지정 파라미터 정보 출력 지정 파라미터 저장 지정 파라미터 읽기 사용 가능한 파라미터 목록 출력 지정 파라미터 쓰기
ros2 bag	info play record	저장된 rosbag 정보 출력 rosbag 기록 rosbag 재생

ROS 2 정보 명령어의 실행 예제 입니다.

```
$ ros2 pkg executables turtlesim

$ ros2 node info /turtlesim

$ ros2 topic echo /turtle1/cmd_vel

$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen "{r: 255, g: 255, b: 255, width: 10}"

$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.5708}"

$ ros2 interface show turtlesim/srv/Spawn.srv

$ ros2 param set /turtlesim background_r 148

$ ros2 bag record /turtle1/cmd_vel
```

ROS 2 기능 보조 명령어

ros2cli + [verbs]	[sub-verbs] (options)	기능
ros2 extensions	(-a) (-v)	ros2cli의 extension 목록 출력
ros2 extension_points	(-a) (-v)	ros2cli의 extension point 목록 출력
ros2 daemon	start status stop	daemon 시작 daemon 상태 보기 daemon 정지
ros2 multicast	receive send	multicast 수신 multicast 전송
ros2 doctor	hello (-r) (-rf) (-iw)	ROS 설정 및 네트워크, 패키지 버전, rmw 미들웨어 등과 같은 잠재적 문제를 확인하는 도구
ros2 wtf	hello (-r) (-rf) (-iw)	doctor와 동일함 (ros2 doctor의 alias) (WTF: Where's The Fire)
ros2 lifecycle	get list nodes set	라이프사이클 정보 출력 지정 노드의 사용 가능한 상태천이 목록 출력 라이프사이클을 사용하는 노드 목록 출력 라이프사이클 상태 전환 트리거
ros2 component	list load standalone types unload	실행 중인 컨테이너와 컴포넌트 목록 출력 지정 컨테이너 노드의 특정 컴포넌트 실행 표준 컨테이너 노드로 특정 컴포넌트 실행 사용 가능한 컴포넌트들의 목록 출력 지정 컴포넌트의 실행 중지
ros2 security	create_key create_keystore create_permission generate_artifacts generate_policy list_keys	보안키 생성 보안키 저장소 생성 보안 허가 파일 생성 보안 정책 파일를 이용하여 보안키 및 보안 허가 파일 생성 보안 정책 파일(policy.xml) 생성 보안키 목록

ROS 2 기능 보조 명령어의 예제 입니다.

```
$ ros2 extensions -a

$ ros2 extension_points -a

$ ros2 daemon start

$ ros2 multicast send

$ ros2 doctor -r

$ ros2 wtf -r

$ ros2 lifecycle list lc_talker

$ ros2 component load /ComponentManager composition composition::Talker
```

```
$ ros2 security create_key demo_keys /talker_listener/listener
```

GUI기반 RQT

- 그래픽 인터페이스 개발을 위한 Qt 기반 프레임워크를 제공합니다.
- 노드와 노드 사이의 연결 정보를 표시하는 rqt_graph와 속도, 전압, 또는 시간이 지남에 따라 변화하는 데이터를 플로팅하는 rqt_plot을 포함하여 30여 가지 기능을 제공합니다.

RViz

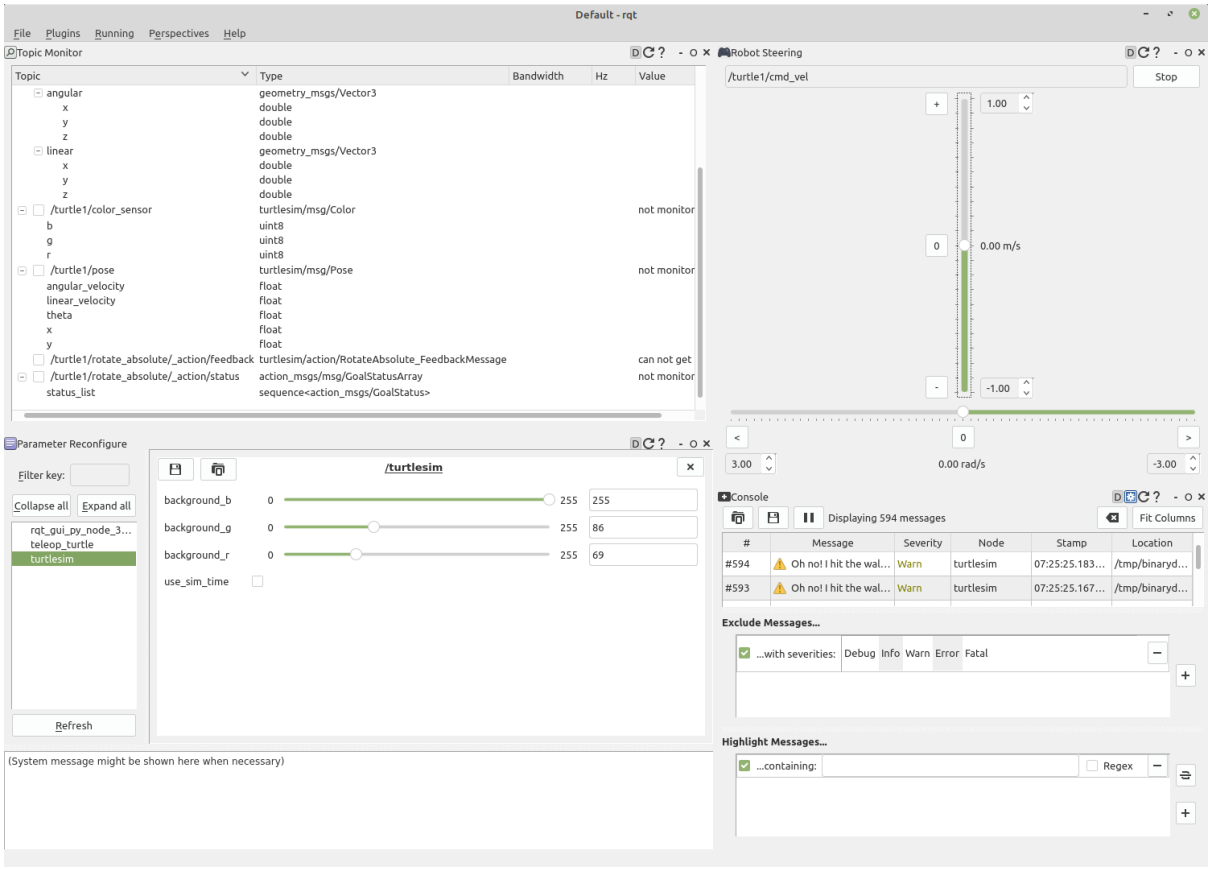
- 3차원 시각화 도구
- 레이저, 카메라 등의 센서 데이터를 시각화합니다.
- 로봇 외형과 계획된 동작을 표현합니다.

Gazebo

- 3D 시뮬레이터입니다.
- 물리 엔진을 탑재하며, 로봇, 센서, 환경 모델 등을 지원합니다.
- ROS와의 높은 호환성을 가지고 있어 다른 시뮬레이터와 비교하여 우수합니다.

18장 ROS2 GUI 개발을 위한 RQt

RQt는 플러그인 형태로 다양한 도구와 인터페이스를 구현할 수 있는 그래픽 사용자 인터페이스(GUI, Graphical User Interface) 프레임워크입니다. 또한, ROS의 종합 GUI 툴박스로 다양한 목적의 GUI 툴을 모아두고 있습니다.



ROS 환경에서 사용할 수 있는 GUI 개발에 있어서 공통으로 필요한 API를 제공합니다. 이를 통해 ROS와 연동하는 GUI 툴을 쉽게 개발할 수 있으며, RQt 플러그인 형태로 개발이 가능합니다. 이러한 개발 방식을 통해 각 플러그인은 RQt에서 통합하여 사용할 수 있게 됩니다. Linux, macOS, Windows에서 모두 사용 가능합니다.

RQt 플러그인의 종류

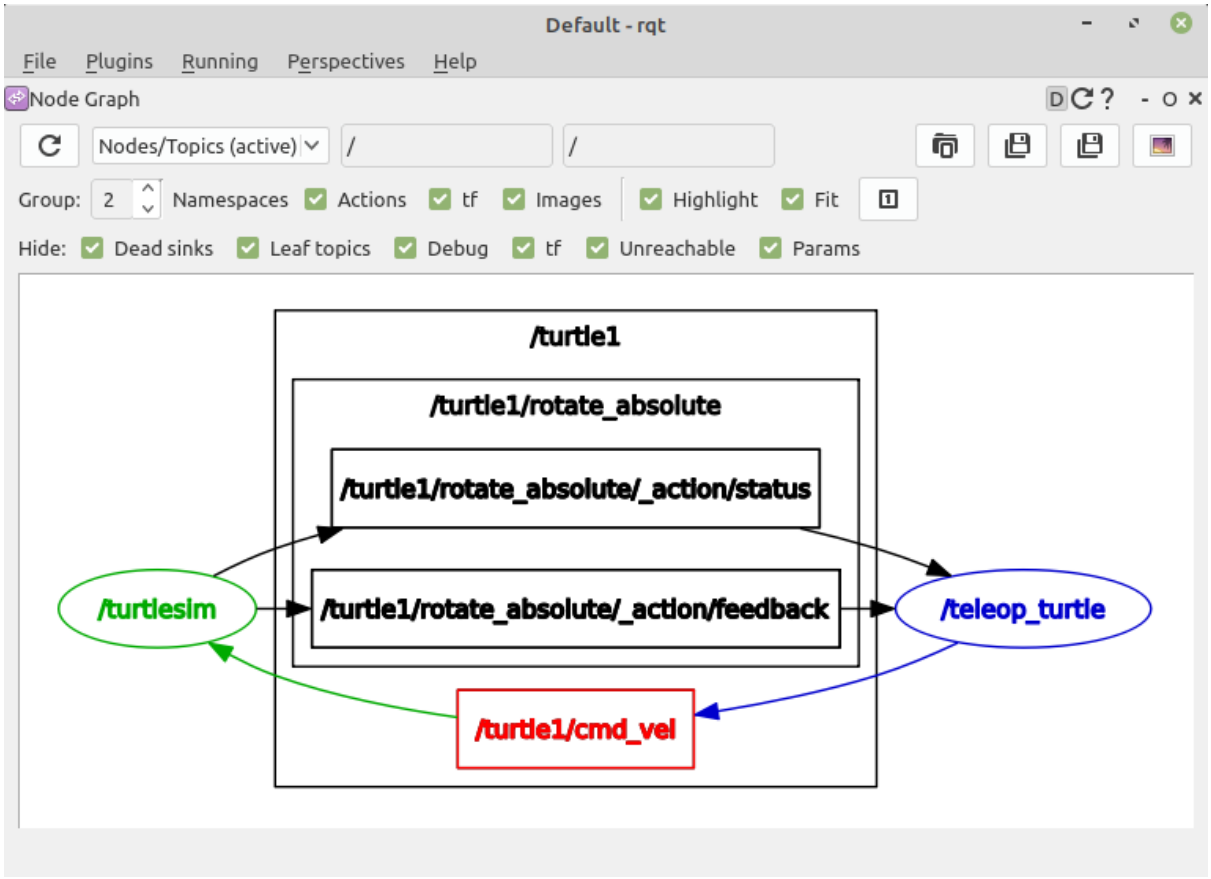
RQt 플러그인은 총 10 종류에 30여 개의 플러그인으로 구성되어 있습니다.

1. 액션 (Actions)
    - Action Type Browser: Action 타입의 데이터 구조를 확인
2. 구성 (Configuration)
    - Dynamic Reconfigure: 노드들에서 제공하는 파라미터 값 확인 및 변경
    - Launch: roslaunch 의 GUI 버전
3. 내성 (Introspection)
    - Node Graph: 실행 중인 노드들의 관계 및 토픽을 확인 가능한 그래프 뷰
    - Package Graph: 노드의 의존 관계를 표시하는 그래프 뷰
    - Process Monitor: 실행 중인 노드들의 CPU 사용률, 메모리 사용율, 스레드 수 등을 확인
4. 로깅 (Logging)

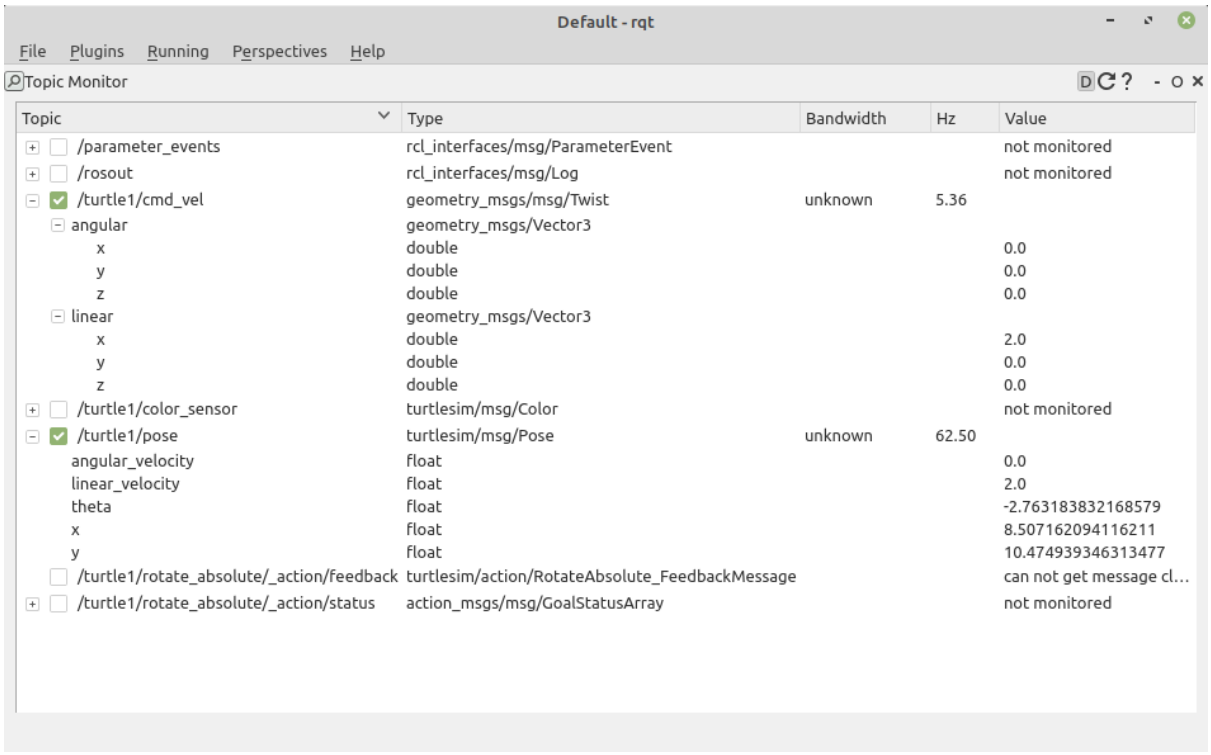
- Bag: ROS 데이터 로깅
- Console: 노드들에서 발생되는 경고(Warning), 에러(Error) 등의 메시지를 확인
- Logger Level: ROS의 Debug, Info, Warn, Error, Fatal 로거 정보를 선택하여 표시

5. 다양한 툴 (Miscellaneous Tools)
    - Python Console: 파이썬 콘솔 화면
    - Shell: 쉘(shell)을 구동
    - Web: 웹 브라우저를 구동
6. 로봇 (Robot)
    - 사용하는 로봇에 따라 계기판(dashboard) 등의 플러그인을 이곳에 추가
7. 로봇툴 (Robot Tools)
    - Controller Manager: 컨트롤러 관리에 필요한 플러그인
    - Diagnostic Viewer: 로봇 디바이스의 경고 및 에러 확인
    - Moveit! Monitor: 로봇 매니퓰레이터 툴인 Moveit! 데이터 확인
    - Robot Steering: 로봇에게 병진 속도와 회전 속도를 토픽으로 발행하는 GUI 툴
    - Runtime Monitor: 실시간으로 노드들에서 발생되는 에러 및 경고를 확인
8. 서비스 (Services)
    - Service Caller: 실행 중인 서비스 서버에 접속하여 서비스를 요청
    - Service Type Browser: 서비스 타입의 데이터 구조를 확인
9. 토픽 (Topics)
    - Message Publisher: 메시지 발행
    - Message Type Browser: 메시지 타입의 데이터 구조 확인
    - Topic Monitor: 토픽 목록 확인 및 사용자가 선택한 토픽의 정보를 확인
10. 시각화 (Visualization)
    - Image View: 카메라의 영상 데이터를 확인
    - Navigation Viewer: 로봇 네비게이션의 위치 및 목표지점 확인
    - Plot: 2차원 데이터 플롯 GUI 플러그인, 2차원 데이터의 도식화
    - Pose View: 현재 TF의 위치 및 모델의 위치 표시
    - RViz: 3차원 시각화 툴인 RViz 플러그인
    - TF Tree: tf 관계를 트리로 나타내는 그래프 뷰

Node Graph

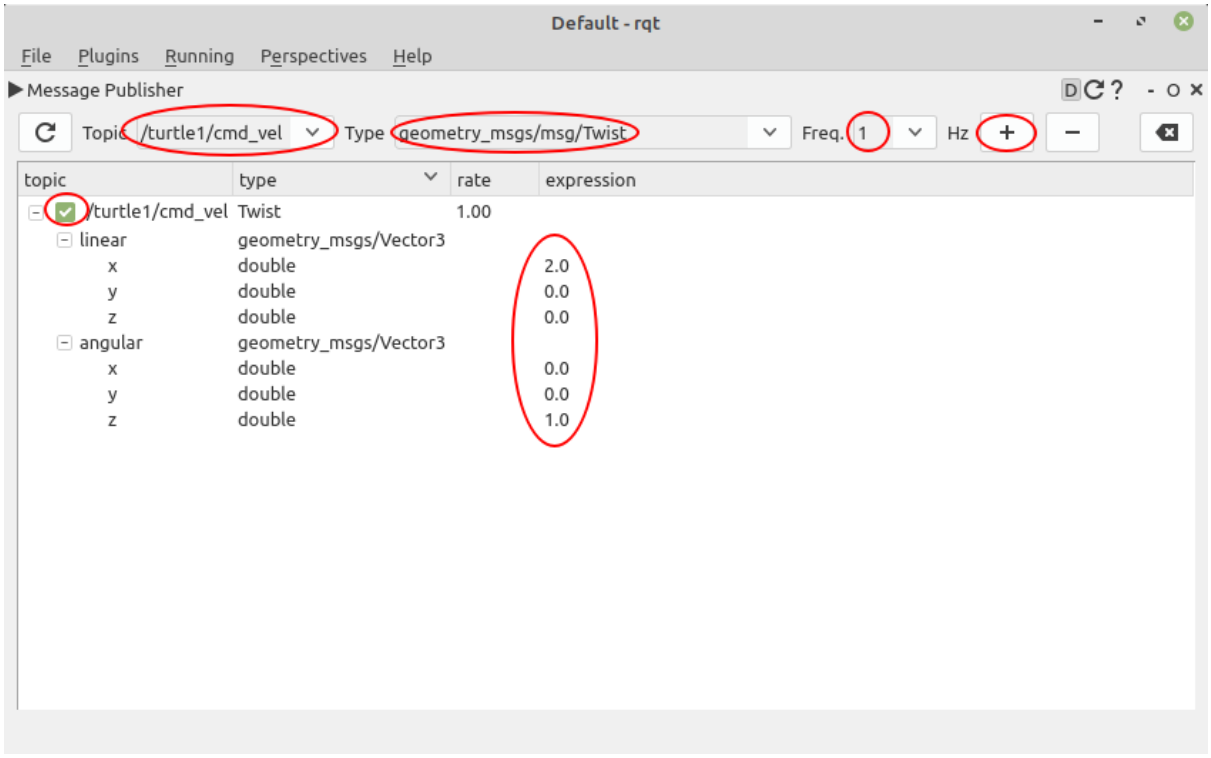


Topic Monitor

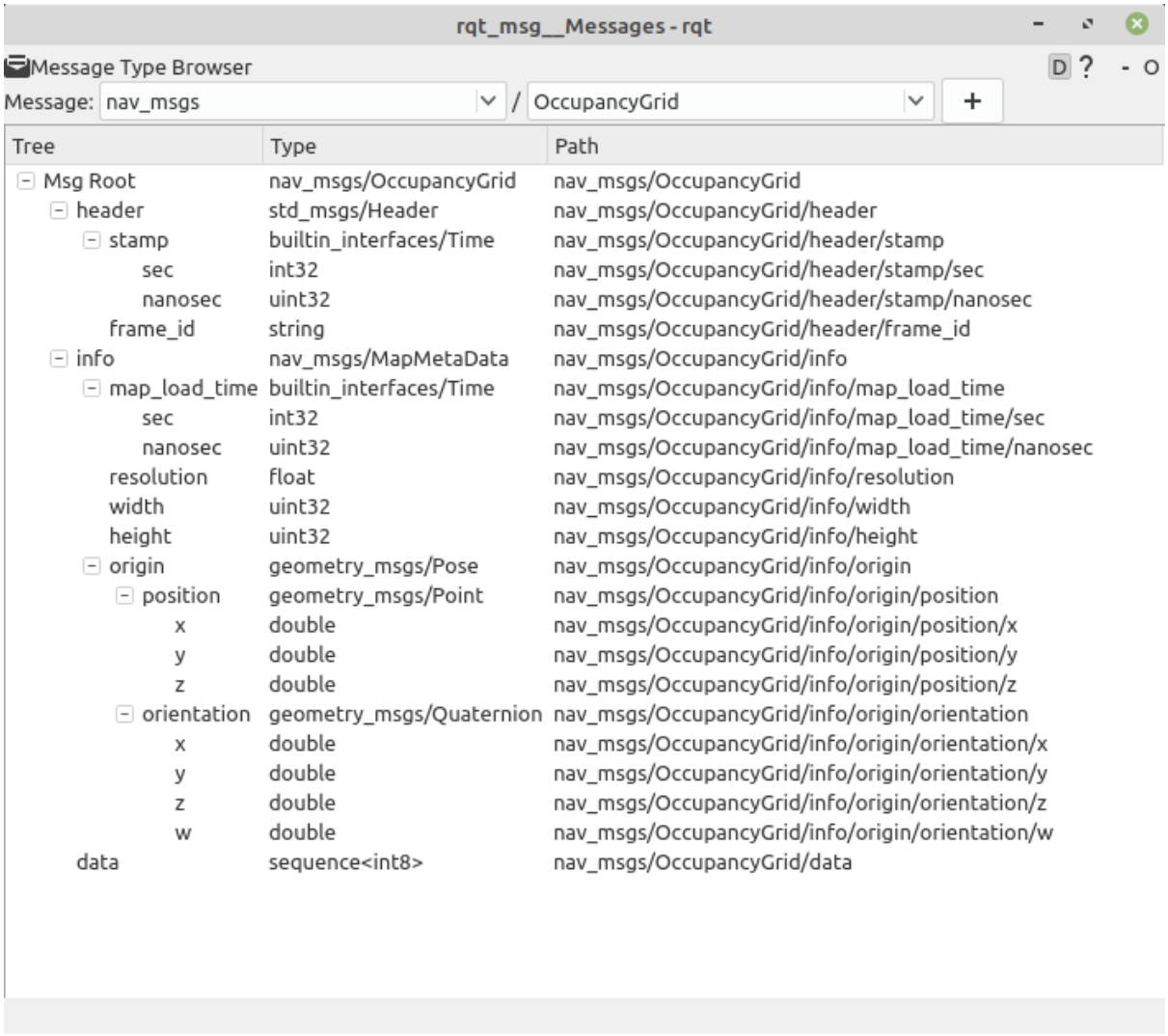


Topic	Type	Bandwidth	Hz	Value
/parameter_events	rcl_interfaces/msg/ParameterEvent			not monitored
/rosout	rcl_interfaces/msg/Log			not monitored
/turtle1/cmd_vel	geometry_msgs/msg/Twist	unknown	5.36	
angular	geometry_msgs/Vector3			
x	double			0.0
y	double			0.0
z	double			0.0
linear	geometry_msgs/Vector3			
x	double			2.0
y	double			0.0
z	double			0.0
/turtle1/color_sensor	turtlesim/msg/Color			not monitored
/turtle1/pose	turtlesim/msg/Pose	unknown	62.50	
angular_velocity	float			0.0
linear_velocity	float			2.0
theta	float			-2.763183832168579
x	float			8.507162094116211
y	float			10.474939346313477
/turtle1/rotate_absolute/_action/feedback	turtlesim/action/RotateAbsolute_FeedbackMessage			can not get message cl…
/turtle1/rotate_absolute/_action/status	action_msgs/msg/GoalStatusArray			not monitored

Message Publisher

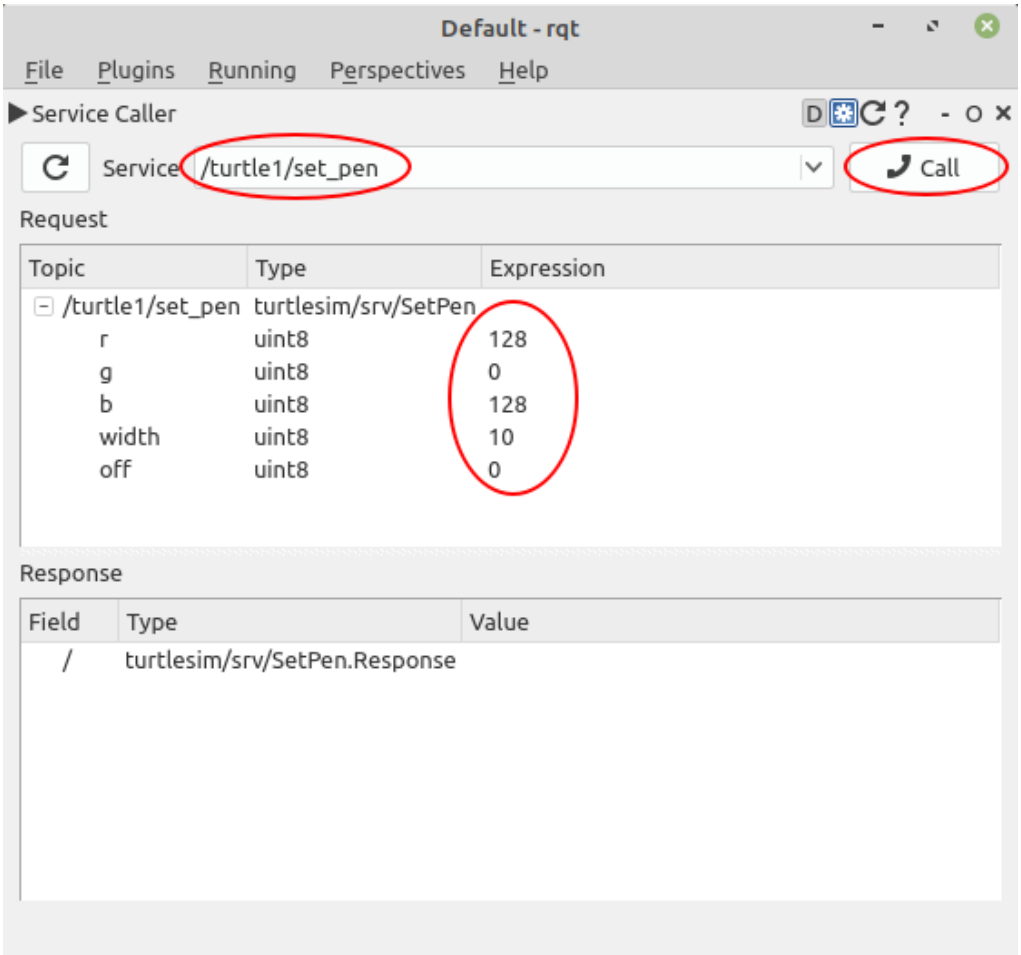


Message Type Browser



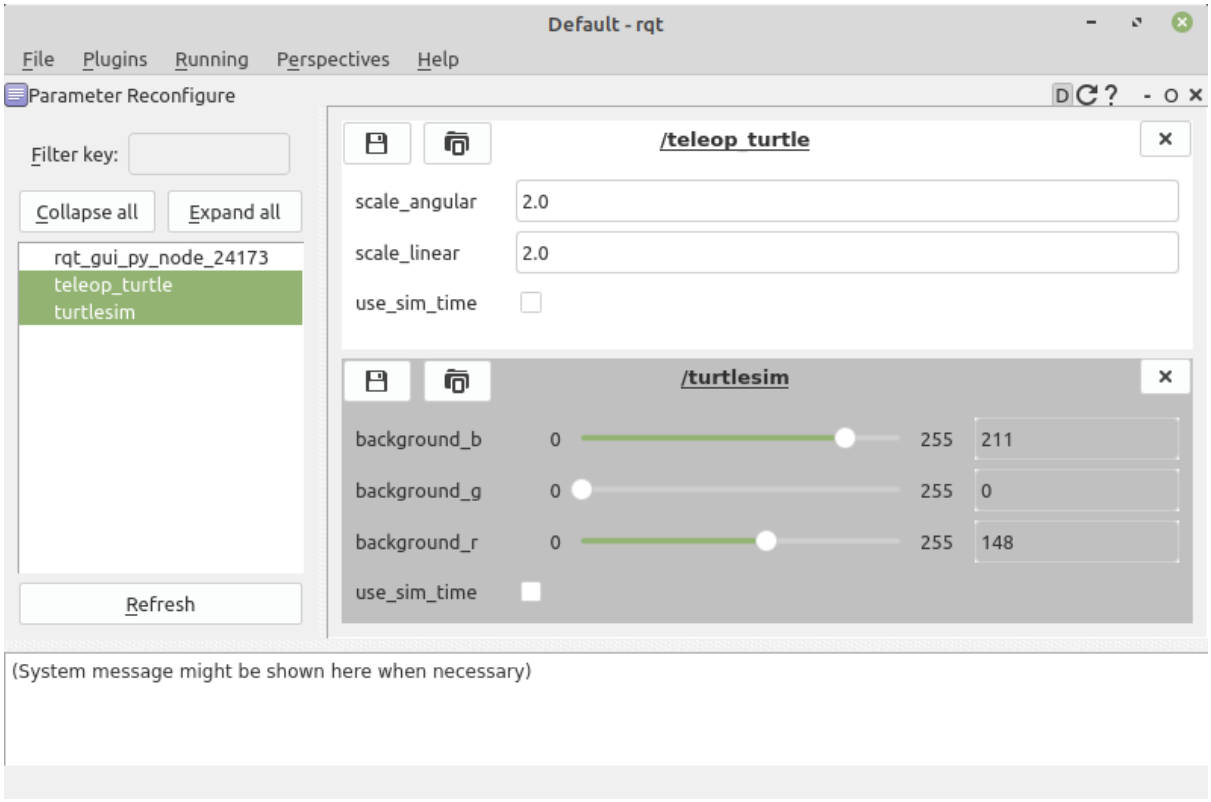
Service Caller

이 RQt 플러그인은 현재 개발 환경에서 실행 중인 서비스 서버에 접속하여 서비스를 요청하는 기능을 제공합니다.



Parameter Reconfigure

Parameter Reconfigure (Dynamic Reconfigure)는 노드들에서 제공하는 파라미터 값을 확인하고 변경할 수 있는 RQt 플러그인입니다. `ros2 param` 명령어의 GUI 버전으로 생각할 수 있습니다.



Plot

Plot 플러그인은 2차원 데이터 플롯 기능을 갖춘 RQt 플러그인입니다. 이 플러그인을 사용하면 2차원 데이터를 시각화할 수 있습니다.

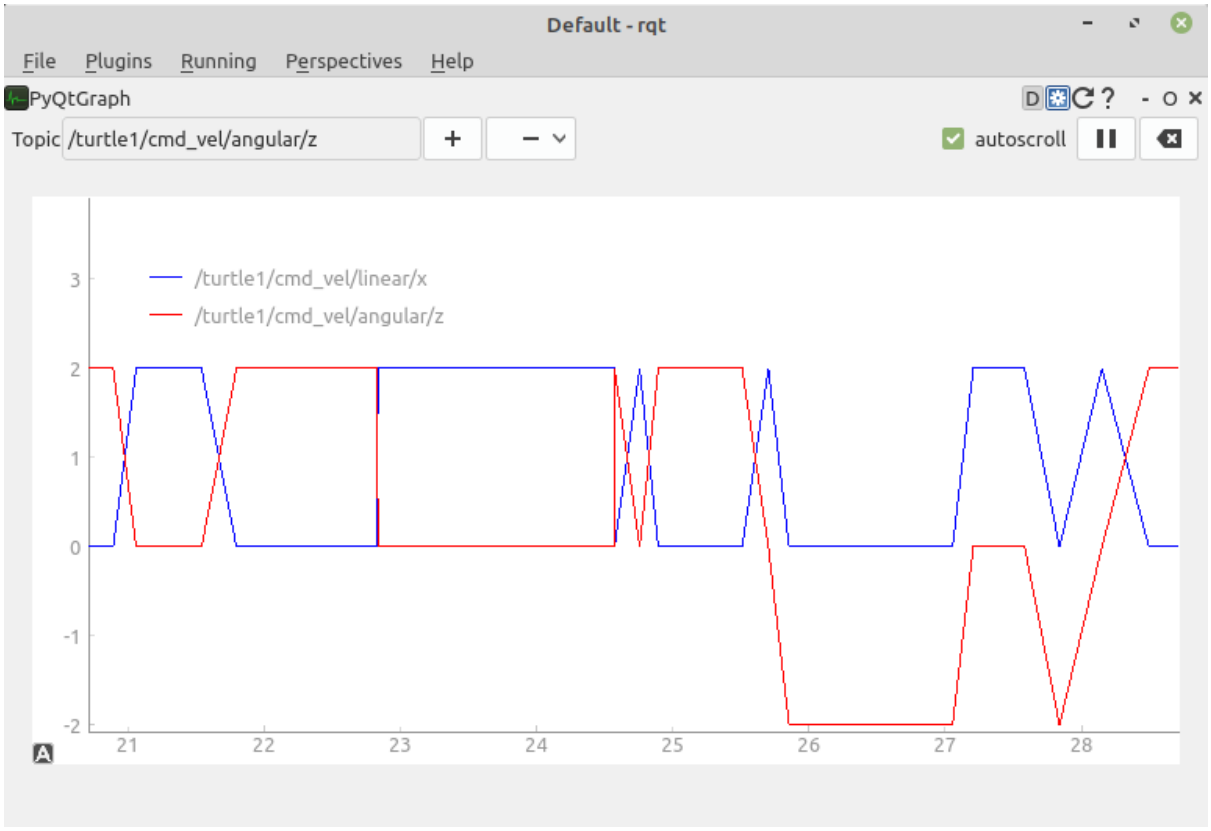


Image View

Image View 플러그인은 카메라의 영상 데이터를 확인할 수 있는 RQt 플러그인입니다.

Console

Console 플러그인은 노드들에서 발생되는 정보(Info), 경고(Warning), 에러(Error) 등의 `rosout` 데이터를 확인할 수 있는 RQt 플러그인입니다.

19장 ROS2 의 표준단위

ROS 커뮤니티에서는 ROS 프로그래밍에 사용하는 표준 단위로 세계적으로 가장 널리 사용되고 있는 국제단위계인 `SI 단위(SI unit)`와 국제단위계의 일곱 개 기본 단위를 조합해 만들어진 `SI 유도 단위(SI derived unit)`을 표준 단위로 채택합니다.

SI 단위는 총 7개이며, SI 유도 단위에는 20개가 있습니다. 로봇공학에서 주로 사용하는 단위는 아래 표와 같습니다. 예를 들어, 메시지를 전송할 때에는 데이터의 종류에 따라 배터리의 전압 단위는 volt (V)를 사용하며, 각의 크기는 radian (rad)을 사용하며, 길이는 meter (m)를 사용해야 합니다. 그리고 시간과 관련한 조합 또한 이러한 규칙을 따릅니다. 예를 들어, 병진 속도는 m/s 단위를, 회전 속도는 rad/s를 사용해야 합니다. 처음 사용하는 분들은 사용에 어색할 수 있지만, 사용하다 보면 딱 정해진 단위로 프로그램 간에 고민 없이 통일할 수 있습니다. 또한, 커뮤니티에서 공개된 다양한 패키지를 사용할 때도 단위로 인한 잘못된 사용이나 버그, 불필요한 변환 연산을 줄일 수 있습니다.

물리량	단위 (SI unit)	물리량	단위 (SI derived unit)
length (길이)	meter (m)	angle (평면각)	radian (rad)
mass (질량)	kilogram (kg)	frequency (주파수)	hertz (Hz)
time (시간)	second (s)	force (힘)	newton (N)
current (전류)	ampere (A)	power (일률)	watt (W)
		voltage (전압)	volt (V)
		temperature (온도)	celsius (°C)
		magnetism (자기장)	tesla (T)[출처] 017 ROS 2의 물리량 표준 단위 (오픈소스 소프트웨어 & 하드웨어: 로봇 기술 공유 카페 (오로카)) | 작성자 표윤석

20장 ROS2 좌표 표현

ROS 2의 좌표 표현 통일의 필요성

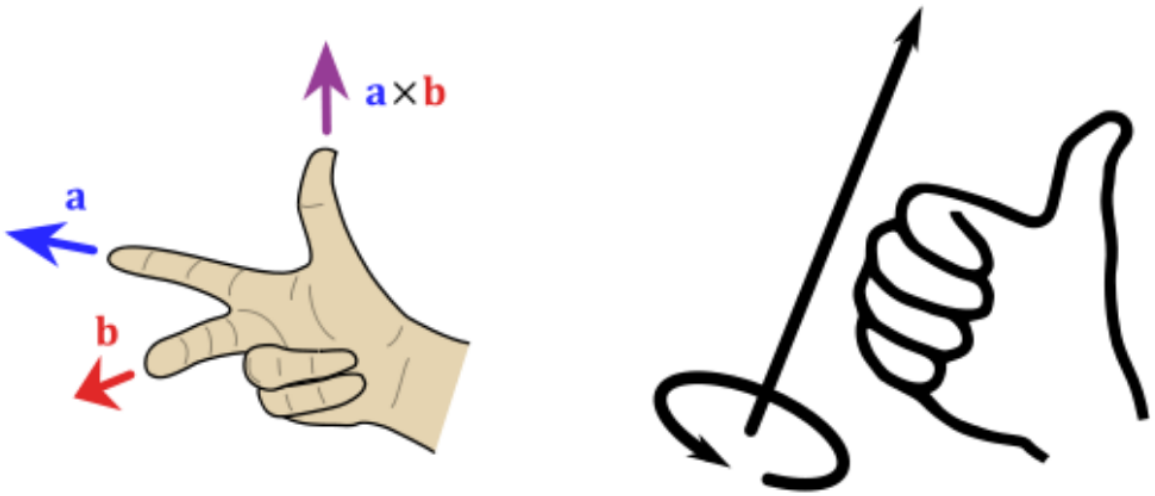
예를 들어 로봇의 센서로 널리 사용되는 카메라의 경우, 컴퓨터 비전 분야에서는 z forward, x right, y down을 기본 좌표계로 사용합니다. 하지만 로봇은 x forward, y left, z up을 기본 좌표계로 사용합니다. 이러한 상황에서는 두 좌표계를 사용함에 따른 논리적 문제에 빠지지 않기 위해 한쪽 좌표계를 기준으로 맞추어야 합니다.

로봇에서 사용하는 LiDAR, IMU, Torque 센서들도 제조사별로 서로 다른 좌표계를 사용할 수 있습니다. 좌표 변환 API가 있더라도 기본 출력을 미리 알아보기 위해서는 각 센서의 좌표를 로봇 좌표에 맞추어 적절한 좌표 변환이 필요합니다. 예를 들어, IMU가 NED 타입(x north, y east, z down, relative to magnetic north) 이냐 ENU 타입(x-east, y-north, z-up, relative to magnetic north)이냐를 잘 살펴봐야 합니다.

좌표 표현도 통일되지 않으면 불편을 겪게 하고 소프트웨어 버그를 유발할 수 있습니다. 이에 ROS 커뮤니티에서는 ROS 개발 초기(2010년)부터 이러한 문제를 줄이기 위해 표준 단위에 대한 규칙과 함께 좌표 표현에 대한 규칙을 제공하여 ROS 프로그래밍에 대한 가이드라인을 만들고 있습니다.

ROS 2 좌표 표현의 기본 규칙

ROS 커뮤니티에서는 모든 좌표계를 삼차원 벡터 표기 규칙을 이해하기 위한 일반적인 기억법인 오른손 법칙 [4]에 따라 표현합니다. 회전 축의 경우, 기본적으로 검지, 중지, 엄지 손가락을 축으로 사용하며, 오른손의 손가락을 감는 방향이 정회전 + 방향입니다.

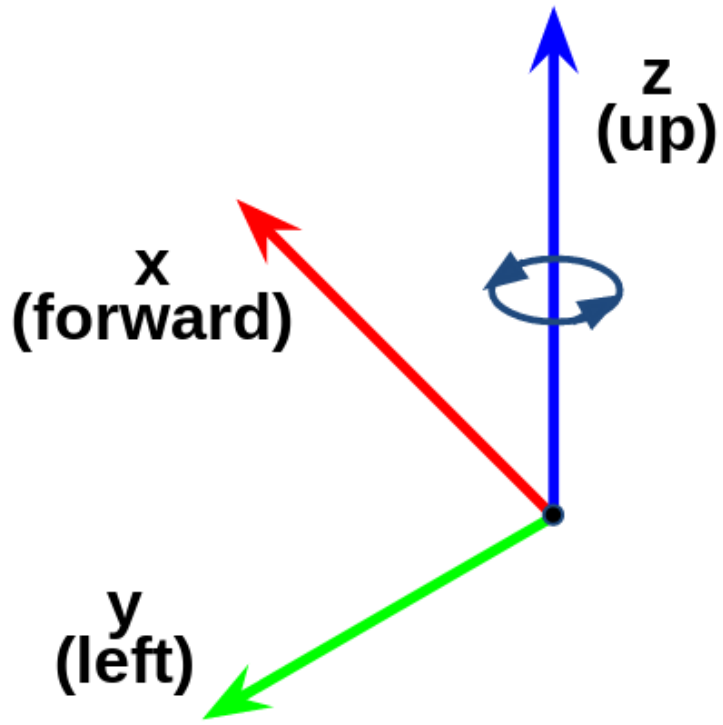


예를 들어, 표준 단위와 함께 설명하면, 회전 각은 라디안(rad) 단위를 사용하며, 로봇이 제자리에서 시계 12시 방향에서 9시 방향으로 회전하였다면 로봇은 정방향 +1.5708 rad 만큼 회전하였다고 이야기합니다.

ROS 2의 좌표 표현의 축 방향 (Axis Orientation) 규칙

기본 3축

ROS 커뮤니티에서는 그림 2와 같이 축 방향 (Axis Orientation)으로 x forward, y left, z up 을 사용합니다. 시각화 툴 RViz나 3차원 시뮬레이터 Gazebo에서는 이러한 기본 3축의 표현을 RGB의 원색으로 나타내며, 순서대로 Red는 x 축, Green은 y 축, Blue는 z 축을 의미합니다. 이를 통해 사용자들은 3축의 방향을 명확하게 인지할 수 있도록 도와줍니다.



ENU 좌표

지리적 위치(geographic locations)의 단거리 데카르트 표현의 경우 ENU(east north up, [5]) 규칙을 사용합니다. 실내 로봇에서는 잘 다루지 않는 좌표이지만, 비교적 큰 맵을 다루는 드론이나 실외 자율 주행 로봇에서 사용하는 좌표입니다.

접미사 프레임 사용 (Suffix Frames)

위에서 언급한 x forward, y left, z up의 기본 3축 및 ENU 좌표에서 벗어나는 경우, 접미사 프레임을 사용하여 기본 좌표계와 구별하여 사용합니다. 자주 사용되는 접미사로는 `_optical` 접미사와 `_ned` 접미사가 있으며, 필요시 좌표 변환을 통해 사용합니다.

`_optical` 접미사

컴퓨터 비전 분야에서는 보통 z 축을 정면, x 축을 오른쪽, y 축을 아래쪽으로 사용하는 카메라 좌표계를 사용합니다. 이 경우, 카메라 센서의 메시지에 `_optical` 접미사를 추가하여 구분합니다. 이때, 카메라 좌표계와 로봇 좌표계의 차이로 인해, x 축을 정면, y 축을 왼쪽, z 축을 위로 사용하는 로봇 좌표계와의 간에는 변환 행렬(TF)이 필요합니다.

`_ned` 접미사

실외에서 동작하는 시스템의 경우, 사용하는 센서 및 지도에 따라 ENU가 아닌 NED (north east down, [6]) 좌표계를 사용해야 할 때가 있습니다. 이때에는 `_ned` 접미사를 붙여 구분합니다.

ROS 2의 좌표 표현의 회전 표현(Rotation Representation)

회전은 목적 및 계산 방법에 따라 같은 좌표계에서도 다양하게 표현될 수 있습니다. ROS 커뮤니티에서는 아래와 같은 대표적인 회전 표현 방식 중 1~3번을 권장합니다.

1. 쿼터니언 (quaternion)
2. 회전 매트릭스 (rotation matrix)
3. 고정축 roll, pitch, yaw (fixed axis roll, pitch, yaw about X, Y, Z axes respectively)
4. 오일러 각도 yaw, pitch, roll (euler angles yaw, pitch, and roll about Z, Y, X axes respectively)