ROS 2 3주차

9장 패키지 설치와 노드 실행

9.1 Turtlesim 패키지 설치

```
$ sudo apt update
$ sudo apt install ros-foxy-turtlesim
```

9.2 Turtlesim 이란?

ROS를 처음 접한 유저들에게 튜토리얼로 제공하기 위해 제작된 ROS 패키지

9.3 Turtlesim 패키지와 노드

Node : 최소 단위의 실행 가능한 프로세스

Package : 하나 이상의 노드 또는 노드 실행을 위한 정보 등을 묶어 놓은 것

Metapackage : 메타패키지

- 자신의 개발환경에 포함되어 있는 패키지 리스트 확인

```
$ ros2 pkg list
```

- turtlesim 패키지에 포함된 노드

```
$ ros2 pkg executables turtlesim //ros2 pkg executables <패키지명>
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

draw_squre : 사각형 모양으로 turtle을 움직이게 하는 노드

mimic : 유저가 지정한 토픽으로 동일 움직임의 turtlesim_node를 복수 개 실행시킬 수 있는 노드

turtle_teleop_key : turtlesim_node를 움직이게 하는 속도 값을 퍼블리시 하는 노드

turtlesim_node : turtle_teleop_key로부터 속도 값을 토픽으로 받아 움직이게 하는 간단한 2D 시뮬레이터 노드

9.4 Turtlesim 패키지의 노드 실행

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run turtlesim turtle_teleop_key
```

9.5 노드, 토픽, 서비스, 액션의 조회

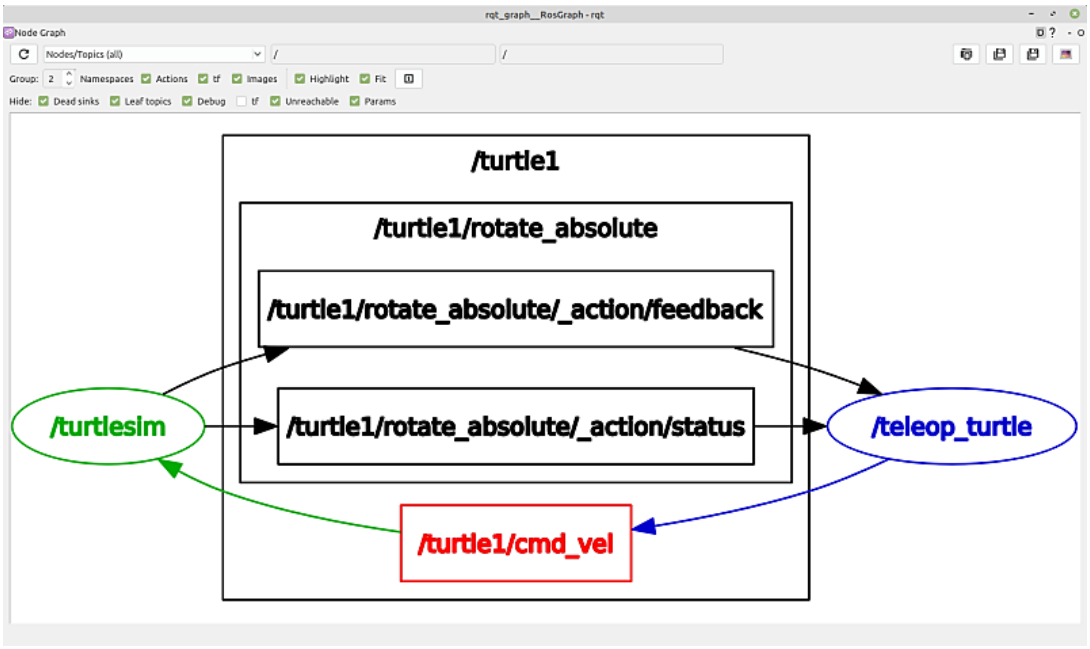
```
$ ros2 node list
/turtlesim
/teloep_turtle
```

```
$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

```
$ ros2 service list
/clear
/kill
/reset
/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
```

9.6 rqt_graph로 보는 노드와 토픽의 그래프 뷰

```
$ rqt_graph
```



10장 ROS 2 노드와 데이터 통신

10.1 노드와 메시지 통신

각 노드의 역할을 목적에 맞추어 세분화시켜 각 노드들 간의 의존성은 줄이고 독립성을 높여 다른 목적의 작업에서도 일부 노드를 재사용할 수 있도록함

Message : 주고받는 데이터

-Integer, Floating point, Boolean, String, 배열

Message Communication : 데이터를 주고받는 방식

-Topic, Service, Action, Parameter

10.2 노드 실행(ros2 run)

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run turtlesim turtle_teleop_key
```

ros2 run : 하나의 노드 실행

ros2 launch : 하나 이상의 노드 실행

rqt, rqt_graph, rviz2 : 지정된 실행 명령어

10.3 노드 목록(ros2 node list)

- 현재 개발환경에서 동작 중인 노드 목록을 보기 위한 명령어

```
$ ros2 node list
/rqt_gui_py_node_28168
/teleop_turtle
/turtlesim
```

- 동일 노드를 복수 개 실행시킬 때(노드명을 변경하여 실행)

```
$ ros2 run turtlesim turtlesim_node __node:=new_turtle
```

노드 목록을 확인하는 명령어로 다시 확인하면 new_turtle이 새롭게 추가되었음을 알 수 있음

```
$ ros2 node list
/rqt_gui_py_node_29017
/teleop_turtle
/new_turtle
/turtlesim
```

10.4 노드 정보(ros2 node info)

```
$ ros2 node info /<노드명>
```

지정된 노드의 Publishers, Subscriber, Service, Action, Parameter 정보 확인

11장 ROS 2 토픽

11.1 토픽

- 비동기식 단방향 메시지 송수신 방식
- Publisher와 Subscriber 간의 통신
- Publisher기능과 동시에 Subscriber 역할도 동시에 수행 가능
- 비동기성과 연속성을 가지므로 항시 정보를 주고 받아야하는 센서 값과 같은 경우에 주로 사용됨

11.2 토픽 목록 확인(ros2 topic list)

- 토픽 정보 확인

```
$ ros node info /<토픽이름>
```

- 현재 동작 중인 노드들의 토픽 정보

```
$ ros2 topic list -t //-t 옵션은 각 메시지의 형태도 표시
```

11.3 토픽 정보 확인(ros2 topic info)

- rqt_graph 외에 CLI를 이용한 정보 확인

```
$ ros2 topic info /<토픽이름>
```

11.4 토픽 내용 확인(ros2 topic echo)

메시지 내용을 실시간으로 표시

```
$ ros2 topic echo /<토픽이름>
```

11.5 토픽 대역폭 확인(ros2 topic bw)

송수신 받는 토픽 메시지의 크기

대역폭은 사용하는 메시지 형태 및 주기에 따라 달라짐

```
$ ros2 topic bw /<토픽이름>
```

11.6 토픽 주기 확인(ros2 topic hz)

토픽의 전송 주기 확인

```
$ ros2 topic hz /<토픽이름>
```

11.7 토픽 지연 시간 확인(ros2 topic delay)

토픽은 RMW 및 네트워크 장비를 거치기 때문에 지연 시간이 반드시 존재

메시지 내에 header stamp 메시지를 사용하고 있다면 명령어를 통해 지연 시간 확인 가능

```
$ ros2 topic delay /<토픽이름>
```

11.8 토픽 퍼블리시<ros2 topic pub)

ros2 topic pub명령어에 토픽 이름, 메시지 타입, 메시지 내용을 기술

```
$ ros2 topic pub <topic_name> <msg_type> "<args>"
```

- 퍼블리시 한 번 수행 : —once

```
$ 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}}"
```

- 지속적은 퍼블리시 : —rate 1 (1Hz)

```
$ ros2 topic pub --rate1 /turtle1/cmd_vel geometry_msgs/msg/Twist {Linear: {x:2.0, y:0.0, z:0.0}, angualr:{x:0.0, y:0.0, z:1.8}}"
```

11.9 bag 기록(ros2 bag record)

rosbag : 퍼블리시되는 토픽을 파일 형태로 저장하고 필요할 때 저장된 토픽을 다시 불러와 동일한 주기로 재생하는 기능

```
ros2 bag record <topic_name1> <topic_name2> <topic_name3>
```

- 원하는 파일 이름이 있을 경우

```
ros2 bag record -o <파일 이름> <토픽 이름>
```

Ctrl+c : 기록 종료

11.10 bag 정보(ros2 bag info)

```
$ ros2 bag info <파일 이름>
```

11.11 bag 재생(ros2 bag play)

```
$ ros2 bag play <파일 이름>
```

12장 ROS 2 서비스

12.1 서비스

동기식 양방향 메시지 송수신 방식

Service Client : 서비스 요청 ← →Service Server : 요청받은 서비스 수행 후 서비스 응답

msg 인터페이스의 변형인 srv 인터페이스 사용

동일한 서비스 서버에 대해 복수의 클라이언트 가질 수 있음

12.2 서비스 목록 확인(ros2 service list)

```
$ ros2 run turtlesim turtlesim_node
```

실행 중인 노드들의 서비스 목록 확인

```
$ ros2 service list
```

12.3 서비스 형태 확인(ros2 service type)

```
ros2 service type <서비스 이름>
```

서비스 목록 확인 명령어에 -t를 붙이면 형태를 함께 볼 수 있음

```
$ ros2 service list -t
```

12.4 서비스 찾기(ros2 service find)

특정 형태의 서비스를 사용하는 서비스명 확인

```
$ ros2 service find std_srvs/srv/Empty
```

12.5 서비스 요청(ros2 service call)

매개 변수로 서비스명, 서비스 혀앧, 서비스 요청 내용을 기술

```
ros2 service call <service_name> <service_type> "<arguments>"
```

1) /clear 서비스로 turtlesim 창의 이동 궤적을 지우기

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

2) /kill

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

죽이고자하는 거분이 이름을 서비스 요청의 내용으로 입력, 위의 예에서는 turtle1

3) /reset

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

4) /set_pen

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

width : 궤적의 크기

5) /spawn

지정된 위치 및 자세로 지정한 이름의 거북이를 추가

이름을 옵션으로 지정하지 않으면 turtle 뒤의 숫자가 자동으로 지정, 동일한 이름 사용 불가

- 기본으로 지정된 turtle1을 /kill 서비스를 이용해 없애고 닌자 거북이 leonardo 를 생성

```
$ ros2 service call /kill turtlesim/srv/Kill "name : 'turtle1'"
$ ros2 service call /spawn turtlesim/srv/Spawn "{x:5.5, y:9,theta:1.57,name:'leonardo'}"
```