4️⃣

Week 4

1. ROS2 action

- 완료하는 데 상당한 시간이 걸리거나 작업이 어려 단계로 구성되는 작업이 있음
- 작업을 수행하기 위해 노드에 요청을 보내고, 작업 진행상황을 추적하고, 완료되면 응답을 받음 ⇒ 액션

크게 액션 목표, 피드백, 결과로 나눌 수 있음

- 목표 : 노드가 수행하는 작업. 예를 들어 로봇팔을 다루는 경우 목표는 특정 위치 x로 이동하는것.
- 피드백 : 작업을 수행하는 동안 노드가 주는 중간응답. 로봇팔은 현재 y 위치에 있다는 피드백을 제공할 수 있음.
- 결과 : 작업이 끝났을때 노드가 반환하는것. (x 위치에 도달함)

2. Message interface

geometry_msgs/msgs/Twist 형태의 예시

- geometry_msgs/msgs/Twist 메시지 형태는 float64 자료형.
- linear.x linear.y linear.z angular.x angular.y angular.z 총 6개.
- 위 메시지들을 Twist.msg, Vector3.msg 처럼 직접 코드를 보는 방법이 있고
- ros2 interface show 명령어를 이용하는 방법이 있음

```
$ ros2 interface show geometry_msgs/msg/Twist
Vector3 linear
Vector3 angular
```

```
$ ros2 interface show geometry_msgs/msg/Vector3
float64 x
float64 y
float64 z
```

3. 토픽/서비스/액션 개념 복습

- 토픽: 비동기식 단방향 메세지 송신
  - Publisher
  - Subscriber
  - 다자간 송신 가능
- 서비스: 동기식 양방향 메세시 송수신
  - Service Client 에서 Request 전송
  - Service Server 에서 Response 회신
  - 다자간 송수신 불가
- 액션: 비동기식+동기식 양방향 메시지 송수신
  - Action Client 에서 Action Goal 전송
  - Action Server 에서 Feedback 또는 Result 회신
  - 다자간 송수신 불가

4. 파라미터

- 서비스와 비슷하지만 RPC를 목적으로 하는 서비스와 달리 파라미터는 노드의 매개변수 설정을 목적으로 함
- ROS에서 노드는 기본적으로 파라미터 서버를 포함하고 있음

Q: 별도의 토픽 기반 UI나 컨트롤러를 통해 제어할수도 있는데 굳이 파라미터를 사용하는 이유?

ex) 모바일로봇의 속도를 제어하는 노드

- 토픽 기반 콜백함수를 사용하여 제어
  - speed 라는 토픽에 새로운 메시지를 보내는 방식으로 제어
  - 토픽을 subscribe하는 모든 노드에서 콜백함수가 호출되어 속도를 update

- 파라미터 사용
  - 속도를 제어하는 노드에 speed라는 파라미터를 설정
  - 속도값이 자주 변경되지 않을 때 유용

⇒ 실시간으로 정밀한 제어가 필요할 경우에는 토픽 기반으로 정보를 다루는것이 더 적합함

⇒ ㅇ max값, min값과 같이 임계값이나 알고리즘의 매개변수 등을 파라미터로 선언하는것이 일반적