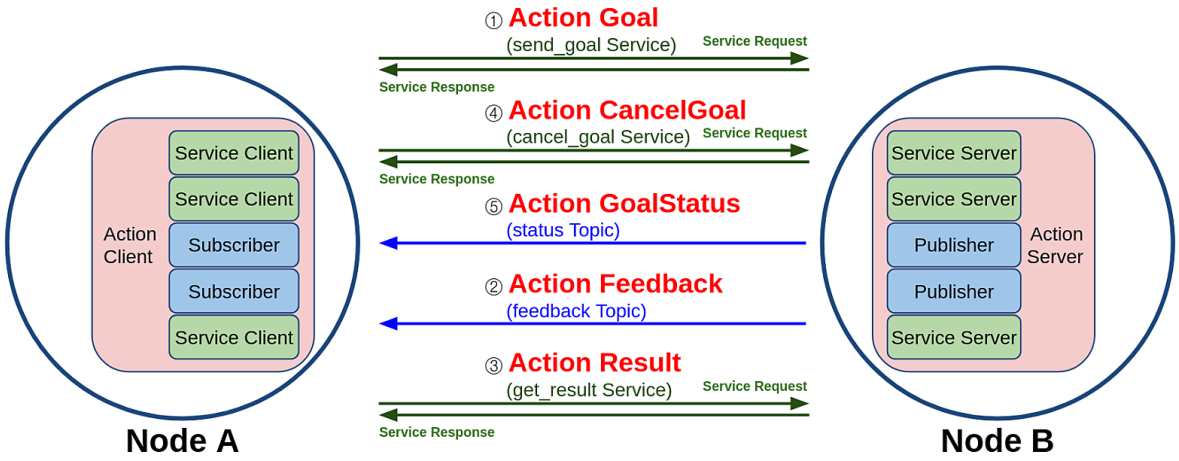
# Week4

## 13장 ROS 2 액션

액션은 비동기식, 동기식 양방향 메시지 송수신 방식으로 액션 목표를 지정하는 Action Client와 액션 목표를 받아 특정 태스크를 수행하면서 중간 결과값을 전송하는 액션 피드백 그리고 최종 결과값을 담은 액션 결과를 전송하는 Action Server간의 통신이다.




ROS 2에서는 Action안에 3개의 Service와 2개의 토픽이 들어있는 구성을 하고 있고 Action goal, Feedback, Result는 msg, srv의 변형인 action 인터페이스를 사용한다.

### 액션 서버 및 클라이언트

turtlesim 노드와 teleop_turtle 노드에서 액션을 살펴보겠다.

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run turtlesim turtle_teleop_key
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.
```

teleap_turtle 노드에서 G/B/V/C/D/E/R/T 키를 누르는 것이 거북이의 각도를 회전하는 rotate_absolute 액션이다.

F를 누르면 액션 목표를 취소하여 동작이 바로 멈추게 된다.

```
$ ros2 run turtlesim turtlesim_node
"G 키를 눌렀을 때"
[INFO] [1690432374.355104756] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1690432374.357228200] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[INFO] [1690432378.461297440] [turtlesim]: Rotation goal completed successfully
```

ros2 node info를 하면 turtlesim 노드에는 RotateAbsolute 액션을 사용하는 rotate_absolute 액션 서버를 가지고 있고 teleap_turtle 노드에는 마찬가지의 액션을 사용하는 rotate_absolute  액션 클라이언트를 가지고 있다.

ros2 action list -t 를 사용하면 현재 실행중인 액션을 메시지타입과 함께 출력한다.

```
$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
```

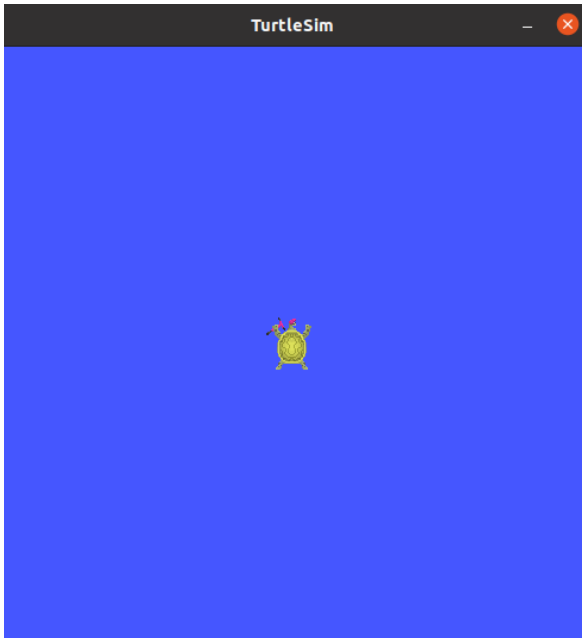
ros2 action info <액션이름> 을 사용하면 액션의 클라이언트와 서버의 개수, 위치한 노드 등을 알 수 있다.

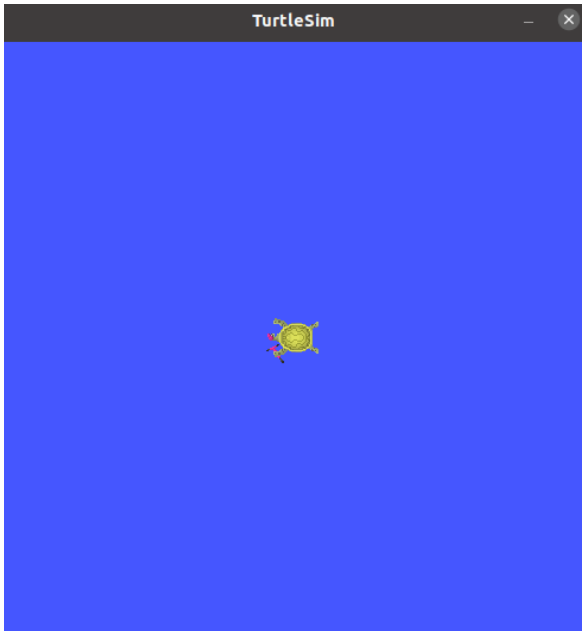
```
$ ros2 action info /turtle1/rotate_absolute
Action: /turtle1/rotate_absolute
Action clients: 1
    /teleop_turtle
Action servers: 1
    /turtlesim
```

action send_goal <액션이름> <메시지타입> “설정인수” 를 사용하면 명령창에서 직접 액션 목표를 보낼 수 있다.  —feedback 옵션을 사용하면 feedback 결과를 출력받을 수 있다.

```
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 3.14}" --feedback
Waiting for an action server to become available...
Sending goal:
     theta: 3.14

Goal accepted with ID: 204bed16566f45479904cdb548054525

Feedback:
    remaining: 0.7768199443817139

.......

Feedback:
    remaining: 0.008818387985229492

Result:
    delta: -0.7680015563964844

Goal finished with status: SUCCEEDED
```

명령창에서 3.14rad으로 돌리라는 액션 Goal을 주면 180도 위치로 돌아가는 것을 볼 수 있다. 또한 중간에 Feedback 결과나 액션 완료후 Result값을 출력해주는 것을 볼 수 있다.




## 14장 ROS 2 인터페이스

ROS 2 노드 간에 데이터를 주고 받기 위해 토픽, 서비스, 액션을 사용하는데, 이때의 메시지 형태를 ROS 2 인터페이스라고 한다. 토픽, 서비스, 액션은 각각 msg, srv, action interface를 사용하고 정수, 부동 소숫점, 불리언과 같은 단순 자료형을 기본으로 메시지안에 메시지를 담은 구조나 메시지가 나열된 배열과 같은 구조도 있다. ROS 2 에서는 새로이 추가된 IDL(Interface Definition Language)가 있다.

예를 들어 Vector3.msg는 다음과 같이 float64 자료형으로 x,y,z 가 선언 되어있다.

```
float64 x
float64 y
float64 z
```

인터페이스에서 사용하는 자료형과 중요 언어별 자료형은 이름에서 약간의 차이가 있으니 관련 디자인 문서를 참고하자.

c++

Generated C++ interfaces

Distilled design documents related to the ROS 2 effort

2 https://design.ros2.org/articles/generated_interfaces_cpp.html

python

Generated Python interfaces

Distilled design documents related to the ROS 2 effort

https://design.ros2.org/articles/generated_interfaces_python.html

## 메시지 인터페이스

ROS 2에서는 인터페이스 형태를 확인하기 위해 ros2 interface show를 사용한다.

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

Vector3  linear
Vector3  angular
```

```
$ 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
```

Twist 라는 메시지를 확인하면 Vector3 라는 타입의 메시지 2개로 구성되어 있고 다시 Vector3 라는 메시지를 확인하면 float64 자료형 인수 3개로 구성됨을 볼 수 있다.

## 서비스 인터페이스

```
$ ros2 interface show turtlesim/srv/Spawn.srv
float32 x
float32 y
float32 theta
string name # Optional.  A unique name will be created and returned if this is empty
---
string name
```

Spawn 이라는 서비스를 확인하면 새로운 거북이를 만들때 위치값 x,y 각도 theta, 이름 name을 지정하고, 그리고 거북이를 만든 후에 이름인 name을 반환 받는다.

## 액션 인터페이스

```
$ ros2 interface show turtlesim/action/RotateAbsolute.action
# The desired heading in radians
float32 theta
---
# The angular displacement in radians to the starting position
float32 delta
---
# The remaining rotation in radians
float32 remaining
```

RotateAbsolute 라는 액션을 확인하면 거북이를 돌려야할 각도 theta를 Goal로 지정하고 거북이를 돌리는 것을 완료한 후, 처음 각도와의 차이 delta를 Result로 반환한다. 그리고 중간 과정에 Feedback 값으로 남은 각도값은 계속 반환해 준다.

## 15장 ROS 2 토픽/서비스/액션 정리 및 비교

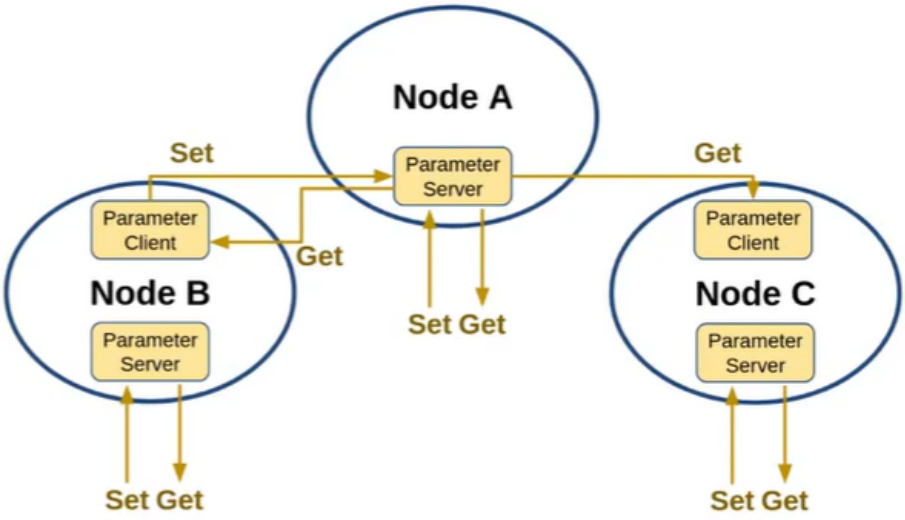
토픽/서비스/액션을 간단히 정리하면 다음과 같다.

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

더 자세한 비교는 책의 표를 참조하면 좋을 것 같다.

## 16장 ROS 2 파라미터

ROS 2에서의 파라미터(Parameter)는 각 노드가 가진 Parameter server를 통해 외부의 Parameter client 와 통신하여 파라미터를 변경하는 것을 의미한다. 파라미터를 통해 노드 내외부에서 노드 내 매개변수를 쉽게 지정, 변경할 수 있고, 쉽게 가져와서 사용할 수 있게 해준다.




파라미터 목록 확인은 ros2 param list를 사용하면 된다. 사용하게 되면 각 노드에 있는 파라미터를 확인할 수 있다.

```
$ ros2 param list
/teleop_turtle:
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  background_r
  use_sim_time
```

파라미터 내용 확인은 ros2 param describe <노드명> <파라미터명> 을 사용하면 된다.

```
$ ros2 param describe /turtlesim background_b
Parameter name: background_b
  Type: integer
  Description: Blue channel of the background color
  Constraints:
    Min value: 0
    Max value: 255
    Step: 1
```

background_b 파라미터를 확인하면 정수 integer 타입의 최대 최소 값이 지정되있고 step이라는 변수도 지정되어 있다.

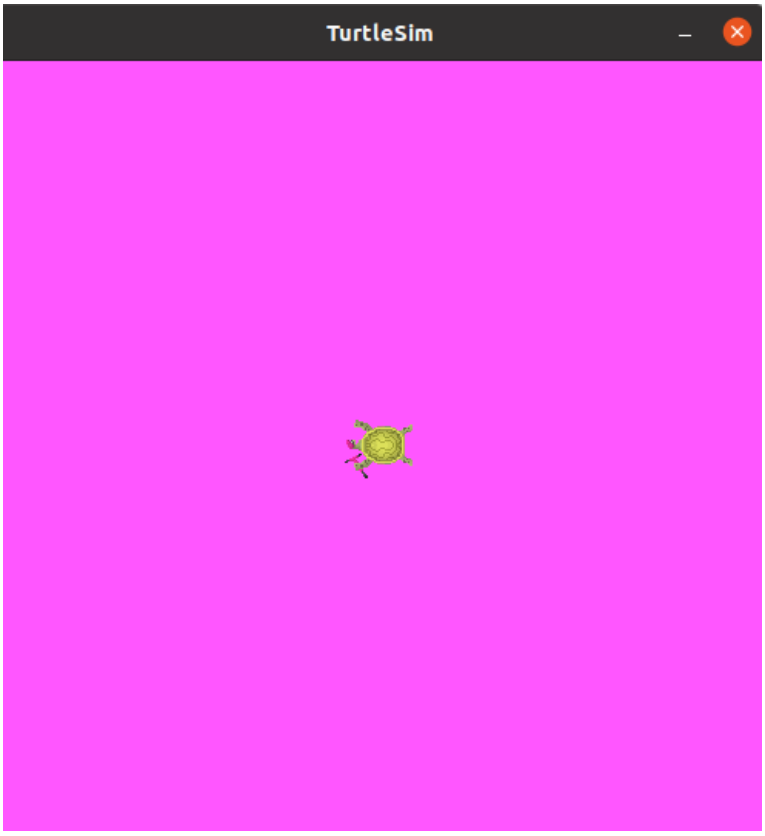
현재 파라미터의 값을 읽는 것은 ros2 param get <노드명> <파라미터명> 을 사용하면 된다.

```
$ ros2 param get /turtlesim background_r
Integer value is: 69
```

파라미터값을 명령창에서 지정하는 것은 ros2 param set <노드명> <파라미터명> <지정값> 을 사용하면 된다.

```
$ ros2 param set /turtlesim background_r 255
ameter successfulSet parameter successful
```

background_r 값을 255로 지정하면 turtlesim의 배경색이 변하는 것을 볼 수 있다.




현제 파라미터의 값을 저장하고 싶다면 ros2 param dump <노드명> 을 사용하면 노드의 parameter 값들을 저장한다.

```
ros2 param dump /turtlesim
Saving to:  ./turtlesim.yaml
```

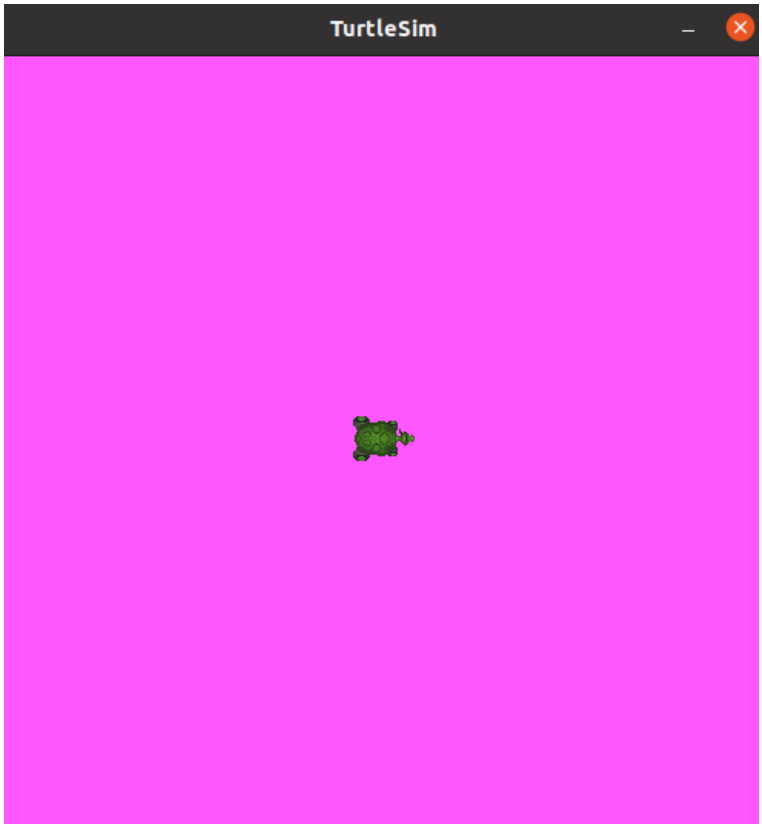
저장하면 해당 노드의 파일이름으로 yaml 파일이 만들어지고 yaml 파일을 확인하면 파라미터 값이 저장되어 있음을 볼 수 있다.

```
/turtlesim:
  ros__parameters:
    background_b: 255
    background_g: 86
    background_r: 255
    use_sim_time: false
```

노드 실행 시 저장된 파라미터 값을 사용하려면 `--ros-args --params-file` 옵션과 함께 파라미터가 저장된 yaml 파일의 이름을 입력하면 된다.

```
$ ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
[INFO] [1690434616.897686681] [turtlesim]: Starting turtlesim with node name /turtlesim
```

```
[INFO] [1690434616.900058814] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

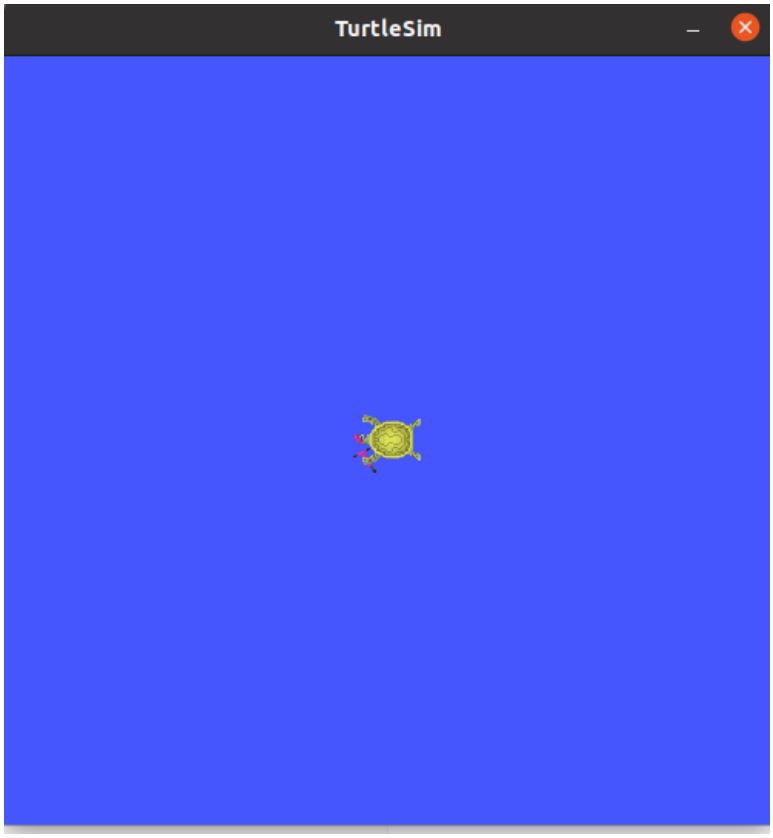



그러면 아까와 같은 색의 배경으로 turtlesim이 시작된다.

특정 파라미터를 삭제하려면 ros2 param delete <노드명> <파라미터명> 을 사용하면 된다.

```
$ ros2 param delete /turtlesim background_r
Deleted parameter successfully
```

turtlesim 노드의 background_r을 삭제하면 파라미터가 삭제된 영향으로 배경색이 변하는 것을 볼 수 있고, ros2 param list를 확인했을때 파라미터가 사라져 있음을 확인할 수 있다.




```
$ ros2 param list
/teleop_turtle:
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  use_sim_time
```