ROS2 17 ~ 20장

17.1. ROS 도구

- CLI(Command Line Interface) 형태의 Command-Line Tools
- GUI 형태의 RQt
- 3차원 시각화툴 Rviz
- 3차원 시뮬레이터 Gazebo

17.1.1. CLI 기반 Command-Line Tools

- 명령어 기반의 툴로 로봇 엑세스 및 거의 모든 ROS 기능을 다룬다.
- 개발환경 및 빌드, 테스트 툴(colcon)
- 데이터를 기록, 재생, 관리하는 툴(ros2bag)
- 그 외 20여 가지

17.1.2. GUI 기반 RQt

- 그래픽 인터페이스 개발을 위한 Qt 기반 프레임워크 제공
- 노드와 그들 사이의 연결 정보 표시(rqt_graph)
- 속도, 전압 등 시간이 지남에 따라 변화하는 데이터를 플로팅(rqt_plot)
- 그 외 30여가지

17.1.3. RViz

- 3차원 시각화툴
- 레이저, 카메라 등의 센서 데이터를 시각화
- 로봇 외형과 계획된 동작을 표현

17.1.4. Gazebo

- 3차원 시뮬레이터
- 물리 엔진을 탑재, 로봇, 센서, 환경 모델 등을 지원
- 타 시뮬레이터 대비 ROS와의 높은 호환성

17.2. ROS2 CLI 명령어

17.3. ROS2 CLI 사용법

```
$ ros2 [verbs] [sub-verbs] [options] [arguments]
```

17.4. ROS2 CLI 종류와 각 sub-verbs의 기능

17.4.1. ROS2 실행 명령어

```
$ ros2 run <package> <excutable>

$ ros2 launch <package> <launch-file>
```

17.4.2. ROS2 정보 명령어

```
$ ros2 pkg [create]             // 새로운 ROS2 패키지 생성
           [executables]        // 지정 패키지의 실행 파일 목록 출력
           [list]               // 사용 가능한 패키지 목록 출력
           [prefix]             // 지정 패키지의 저장 위치 출력
           [xml]                // 지정 패키지의 패키지 정보 파일(xml) 출력
```

```
$ ros2 node [info]              // 실행 중인 노드 중 지정한 노드의 정보 출력
            [list]              // 실행 중인 모든 노드의 목록 출력
```

```
$ ros2 topic [bw]               // 지정 토픽의 대역폭 측정
             [delay]            // 지정 토픽의 지연시간 측정
             [echo]             // 지정 토픽의 데이터 출력
             [find]             // 지정 타입을 사용하는 토픽 이름 출력
             [hz]               // 지정 토픽의 주기 측정
             [info]             // 지정 토픽의 정보 출력
             [list]             // 사용 가능한 토픽 목록 출력
             [pub]              // 지정 토픽의 토픽 퍼블리시
             [type]             // 지정 토픽의 토픽 타입 출력
```

```
$ ros2 service [call]           // 지정 서비스의 서비스 요청 전달
               [find]           // 지정 서비스 타입의 서비스 출력
               [list]           // 사용 가능한 서비스 목록 출력
               [type]           // 지정 서비스의 타입 출력
```

```
$ ros2 action [info]            // 지정 액션의 정보 출력
              [list]            // 사용 가능한 액션 목록 출력
              [send_goal]       // 지정 액션의 액션 목표 전송
```

```
$ ros2 interface [list]         // 사용 가능한 모든 인터페이스 목록 출력
                 [package]      // 특정 패키지에서 사용 가능한 인터페이스 목록 출력
                 [packages]     // 인터페이스 패키지들의 목록 출력
                 [proto]        // 지정 패키지의 프로토타입 출력
                 [show]         // 지정 인터페이스의 데이터 형태 출력
```

```
$ ros2 param [delete]           // 지정 파라미터 삭제
             [describe]         // 지정 파라미터 정보 출력
             [dump]             // 지정 파라미터 저장
             [get]              // 지정 파라미터 읽기
             [list]             // 사용 가능한 파라미터 목록 출력
             [set]              // 지정 파라미터 쓰기
```

```
$ ros2 bag [info]               // 저장된 rosbag 정보 출력
           [play]               // rosbag 기록
           [record]             // rosbag 재생
```

17.4.3. ROS2 기능 보조 명령어

```
$ ros2 extensions -a            // ros2cli의 extension 목록 출력
                  -v
```

```
$ ros2 extension_points -a      // ros2cli의 extension point 목록 출력
                        -v
```

```
$ ros2 daemon start             // daemon 시작
              status            // daemon 상태 보기
              stop              // daemon 정지
```

```
$ ros2 multicast receive        // multicast 수신
                 send           // multicast 전송
```

```
$ ros2 doctor hello             // ROS 설정 및 네트워크, 패키지 버전, rmw 미들웨어 등과 같은
              -r                // 잠재적 문제를 확인하는 도구
              -rf
              -iw
```

```
$ ros2 wtf hello                // doctor와 동일함
           -r                   // (ros2 doctor의 alias)
           -rf                  // (WTF:Where's The File)
           -iw
```

```
$ ros2 lifecycle get            // 라이프사이클 정보 출력
                 list           // 지정 노드의 사용 가능한 상태천이 목록 출력
                 nodes          // 라이프사이클을 사용하는 노드 목록 출력
                 set            // 라이프사이클 상태 전환 트리거
```

```
$ ros2 component list           // 실행 중인 컨테이너와 컴포넌트 목록 출력
                 load           // 지정 컨테이너 노드의 특정 컴포넌트 실행
                 standalone     // 표준 컨테이너 노드로 특정 컴포넌트 실행
                 types          // 사용 가능한 컴포넌트들의 목록 출력
                 unload         // 지정 컴포넌트의 실행 중지
```

```
$ ros2 security create_key      // 보안키 생성
                create_keystore // 보안키 저장소 생성
                create_permission // 보안 허가 파일 생성
                generate_artifacts // 보안 정책 파일을 이용하여 보안키 및 보안 허가 파일 생성
                generate_policy // 보안 정책 파일(policy.xml) 생성
                list_keys       // 보안키 목록 출력
```

17.5. 지속 개발되고 있는 ROS2 CLI

- ROS2 CLI 툴 개발 참고자료

https://ubuntu.com/blog/creating-a-ros-2-cli-command-and-verb

- ROS2 CLI 관련 정보

> https://github.com/ros2/ros2cli

- Canonical - CLI Cheats Sheet 자료

https://github.com/ubuntu-robotics/ros2_cheats_sheet/blob/master/cli/cli_cheats_sheet.pdf

18.1. ROS의 종합 GUI 툴 RQt

19.2 ROS2 표준 단위

- 국제단위계 SI 단위와 SI 유도 단위가 ROS2의 표준 단위

물리량	단위(SI unit)	물리량	단위(SI unit)
Length	Meter	Angle	Radian
Mass	Kilogram	Frequency	Hertz
Time	Second	Force	Newton
Current	Ampere	Power	Watt
		Voltage	Volt
		Temperature	Celsius
		Magnetism	Tesla

20.2. 좌표 표현의 기본 규칙

- https://www.ros.org/reps/rep-0103.html
- 모든 좌표계는 오른손 법칙에 따라 표현

Coordinate Frame Conventions

All coordinate frames should follow these conventions.

Chirality

All systems are right handed. This means they comply with the right hand rule [4].

20.3. 좌표 표현의 축 방향(Axis Orientation) 규칙

1. 기본 3축

- x축 : Red, y축 : Green, z축 : Blue

2. ENU 좌표

- 큰 맵을 다루는 드론, 실외 자율주행 로봇에서 사용하는 좌표

Axis Orientation

In relation to a body the standard is:

- x forward
- y left
- z up

For short-range Cartesian representations of geographic locations, use the east north up [5] (ENU) convention:

- X east
- Y north
- Z up

To avoid precision problems with large float32 values, it is recommended to choose a nearby origin such as your system's starting position.

3. Suffix Frames(접미사 프레임)

- _optical 접미사 : 컴퓨터 비전 분야에서 사용
- _ned 접미사 : 실외에서 동작하는 시스템의 경우, 사용하는 센서 및 지도에 따라 ENU가 아닌 NED 좌표계를 사용할 때가 있다.

Suffix Frames

In the case of cameras, there is often a second frame defined with a "_optical" suffix. This uses a slightly different convention:

- z forward
- x right
- y down

For outdoor systems where it is desirable to work under the north east down [6] (NED) convention, define an appropriately transformed secondary frame with the "_ned" suffix:

- X north
- Y east
- Z down

20.4. 좌표 표현의 회전 표현(Rotation Representation) 규칙

1. 쿼터니언(Quaternion)

- 간결한 표현방식으로 가장 널리 사용됨(x, y, z, w)
- 특이점 없음(No singularities)

2. 회전 매트릭스(Rotation matrix)

- 특이점 없음(No singularities)

3. 고정축 roll, pitch, yaw(fixed axis roll, pitch, yaw about X, Y, Z axes respectively

- 각속도에 사용

4. 오일러 각도 yaw, pitch, roll(euler angles yaw, pitch and roll about Z, Y, X axes respectively)

- 전역 좌표계에서 회전이 발생하기 때문에 한 축의 회전이 다른 축의 회전과 겹치는 문제로 인해 사용을 권장하지 않는다.

Rotation Representation

There are many ways to represent rotations. The preferred order is listed below, along with rationale.

1. quaternion
    - Compact representation
    - No singularities
2. rotation matrix
    - No singularities
3. fixed axis roll, pitch, yaw about X, Y, Z axes respectively
    - No ambiguity on order
    - Used for angular velocities
4. euler angles yaw, pitch, and roll about Z, Y, X axes respectively
    - Euler angles are generally discouraged due to having 24 'valid' conventions with different domains using different conventions by default.