5 장 ROS2 의 중요 콘셉트

5.1 Why ROS 2?

- •시장 출시 시간 단축
- •생산을 위한 설계
- •멀티 플랫폼 (리눅스, 윈도우, macOS)
- •다중 도메인 (다양한 로봇 응용 분야)
- •벤더 선택 가능
- •공개 표준 기반 (산업 표준 기반의 통신 방법)
- •자유 재량 허용 범위가 넓은 오픈소스 라이센스 해택
- •글로벌 커뮤니티
- •산업 지원
- •ROS1 과의 상호 운용성 확보

6 장 ROS1 과 2 의 차이점으로 알아보는 ROS2 의 특징

6.1 ROS2 개발의 필요성

□ ROS1 의 제한 사항

- •단일 로봇
- •워크스테이션급 컴퓨터
- •리눅스 환경
- •실시간 제어 지원하지 않음 ※NASA 에서 실시간 제어를 위해 ROS1 을 수정해서 사용하였다.
- •안정된 네트워크 환경이 요구됨
- •주로 대학이나 연구소와 같은 아카데미 연구 용도

□ 새로운 요구 사항

- •두 대 이상의 로봇
- •임베디드 시스템에서의 ROS 사용
- •실시간 제어
- •불안정한 네트워크 환경에서도 동작할 수 있는 유연함
- •멀티 플랫폼 (리눅스, 윈도우, macOS)
- •최신 기술 지원 (Zeroconf, Protocol Buffers, ZeroMQ, WebSockets, DDS 등)
- •상업용 제품 지원

6.2 ROS2 의 특징

□ Platforms

- •WSL2 를 이용하여 Windows 에서도 가능케되었다.

□ Real-time

- •선별된 하드웨어, 리얼타임 지원 운영체제, DDS 의 RTPS(Real time Publish Subscribe)와 같은 통신 프로토콜, 리얼타임 코드 사용을 전제로 한다.

□ Security

- •상용 로봇에 ROS 를 도입할 수 없게 만드는 첫 번째 이유

□ Communication

- •통신 미들웨어 DDS 를 사용하는데 ROS1 에서 각 노드들의 정보를 관리하였던 ROS Master 가 없어도 여러 DDS 프로그램 간에 통신이 가능하다.
- •또한 QoS 를 설정할 수 있어 신뢰도를 높이거나, 통신 속도를 최우선하여 사용할 수도 있다.

□ Middleware interface

- •벤더들의 미들웨어를 유저가 원하는 사용 목적에 맞게 선택하여 사용할 수 있도록 RMW 형태로 지원하고 있다.

□ Node manager(Discovery)

•ROS1 의 경우 roscore 를 실행해야 ROS Master, ROS Parameter Server, rosout logging node 가 실행됐지만, ROS2 에서는 3 가지 프로그램이 각각 독립적 수행으로 바뀌어서 실행할 필요가 없어졌다.

□ Languages

•ROS1 과 동일 (C++, 파이썬)

□ Build system

•catkin(ROS1) --> ament(ROS2)

•ament 는 CMake 를 사용하지 않는 파이썬 패키지 관리도 가능하다.

□ Build tools

•colcon 을 이용해서 패키지 작성, 테스트, 빌드 등을 지원하며, ROS2 기반 프로그래밍을 할 때 빼놓을 수 없는 툴이다.

□ Build options

•Multiple workspace: 복수의 독립된 워크스페이스를 사용할 수 있어서 작업 목적 및 패키지 종류별로 이를 관리할 수 있다.

•No non-isolated build: 설치용 폴더를 분리하거나 병합할 수 있다.

•No devel space: ROS2 에서는 패키지를 빌드한 후 설치해야 패키지를 사용할 수 있도록 바뀌었다. 단, 편리한 사용성도 고려하여 colcon 사용 시에 "--symlink-install"과 같은 옵션을 제공하여 심볼릭 링크 설치가 가능하도록 했다.

□ Version control system

□ Client library, Life cycle, Multiple nodes, Threading model

□ Messages (Topic, Service, Action)

□ Command Line Interface, Launch, Graph API, Embedded Systems

7 장 ROS2 와 DDS

7.1 ROS2 와 DDS

- TCPROS(ROS1) --> DDS 의 RTPS(ROS2)
- 실시간 데이터 전송을 보장하고 임베디드 시스템에도 사용할 수 있게 되었다.
- QoS 를 이용한 신뢰도 상승과 통신 속도 최우선시

7.2 DDS 란?

- 데이터 분산 시스템의 줄임말로 OMG 에서 표준을 정하고자 만든 트레이드 마크이다. (실체는 데이터 통신을 위한 미들웨어)

7.3 DDS 의 특징

- 산업 표준
- 운영체제 독립
- 언어 독립
- UDP 기반의 전송 방식
- 데이터 중심적 기능
- 동적 검색
- 확장 가능한 아키텍처
- 상호 운용성
- 서비스 품질(QoS)
- 보안

7.4 ROS 에서의 사용법

- 기본적인 퍼블리셔 노드와 서브스크라이브 노드 실행
- RMW 변경 방법
- RMW 의 상호 운용성 테스트
- Domain 변경 방법
- QoS 테스트

8 장 DDS 의 QoS

- DDS 의 서비스 품질 (Quality of Service)
- History: 데이터를 몇 개나 보관할지를 결정
- Reliability: 데이터 전송에 있어 속도를 우선시 하는지 신뢰성을 우선시 하는지를 결정
- Durability: 데이터를 수신하는 서브스크라이버가 생성되기 전의 데이터를 사용할 것인지에 대한 옵션
- Deadline: 정해진 주기 안에 데이터가 발신 및 수신되지 않을 경우 EventCallback 을 실행시키는 옵션
- Lifespan: 정해진 주기 안에서 수신되는 데이터만 유효 판정하고 그렇지 않은 데이터는 삭제하는 옵션
- Liveliness: 정해진 주기 안에서 노드 혹은 토픽의 생사를 확인하는 옵션