Week6

태그	
내용	

21장. ROS2의 시간

다수의 노드로 구성된 로봇의 소프트웨어 동작에서 시작은 매우 중요합니다. ROS 프레임워크는 시간과 관련된 라이브러리를 제공하고 있으므로 적극적으로 활용합니다.

시간의 추상화

ROS2에서는 시간을 추상화하여 3가지 형태로 제공합니다.

1. RCL_ROS_TIME
2. RCL_SYSTEM_TIME
3. RCL_STEADY_TIME

ROS Time

이것은 일반적으로 시뮬레이션 환경에서 시간을 다루기 위해 사용됩니다. use_sim_time을 사용하여 설정된 노드는 clock 토픽을 수신할 때까지 시간을 0으로 초기화합니다. 이 값이 True인 경우 설정된 노드가 clock 토픽을 수신할 때까지 시간을 0으로 초기화합니다.

System Time

Host system의 시간을 사용하는 것을 의미합니다. 예외적으로 host와 네트워크 시간이 동기화되면서 System time이 반대로 가는 것을 유의해야 합니다. 특정 서버의 시간을 기준으로 네트워크 상에 Host 머신들의 시간을 동기화하는 방법은 다음 명령어를 사용하는 것입니다.

```
sudo ntpdate ntp.ubuntu.com
```

Steady Time

하드웨어 타임아웃은 시간을 의미하며, 단조 증가하는 특성을 가집니다.

Time API

ROS2에서 제공하는 시간 API에는 time, duration, rate가 있습니다.

1. Time 클래스는 시간을 다룰 수 있는 연산자를 제공합니다. 결과를 다양한 시간 형식으로 변환할 수 있으며, now 함수를 통해 현재 시간을 얻을 수 있습니다.
2. Duration 클래스는 시간의 기간에 대한 연산자를 제공합니다. 이전 시간은 음수로 표기됩니다.
3. Rate 클래스는 반복문에서 특정 주기를 유지시켜주는 API를 제공합니다.

22장. ROS2의 파일 시스템

ROS2 파일 시스템에 대해 설명하기 위해서는 기본 폴더 구조 및 파일 구성, 설치 폴더, 사용자 패키지에 대해 살펴볼 필요가 있습니다.

패키지와 메타패키지

패키지는 ROS2 응용프로그램의 논리적인 단위로, 하나 이상의 노드를 포함합니다. ROS2는 전 세계의 개발자들이 로봇 개발에 필요한 소프트웨어 패키지를 계속해서 만들어서 공개하고 있습니다.

패키지는 공통의 목적으로 묶일 경우 메타패키지로 관리됩니다. 각 패키지는 package.xml 파일을 포함하며, 이는 패키지 정보를 담고 있는 메타데이터입니다.

패키지 바이너리 설치

ROS2 패키지는 apt-get을 이용하여 다양한 패키지를 설치할 수 있습니다. 설치된 파일은 /opt/ros/humble에 저장됩니다.

소스코드 빌드 설치

소스 코드를 다운로드하고 빌드하여 설치하려면 git clone을 사용할 수 있습니다.

기본설치폴더

ROS2를 설치하면 /opt/ros/humble 디렉토리 아래에 기본적으로 다음과 같은 하위 디렉토리들이 설치됩니다.

/bin 실행 가능한 바이너리 파일

/cmake 빌드 설정 파일

/include 헤더 파일

/lib 라이브러리 파일

/opt 기타 의존 패키지

/share 패키지의 빌드, 환경 설정 파일

local_setup.* 환경 설정 파일

setup.* 환경 설정 파일

23장. ROS2 빌드 시스템과 빌드 툴

빌드 시스템은 단일 패키지를 빌드합니다. 빌드 툴과의 차이점은 전체를 대상으로 한다는 것입니다.

빌드 시스템

ROS1에서는 catkin 빌드 시스템을 사용하지만, ROS 2는 ament를 사용합니다. 이 두 빌드 시스템 모두 CMake를 기반으로 동작합니다.

빌드 툴

빌드 툴은 전체 패키지를 대상으로 하므로 하나의 패키지가 아닌 전체를 대상으로 합니다. 따라서 별도의 도구가 필요합니다. ROS2에서는 colcon(Collective Construction)이라는 빌드 툴을 사용합니다.

패키지 생성

ros2 pkg create 명령어를 통해 패키지의 기본 내용을 쉽게 생성할 수 있다.

```
$ ros2 pkg create (패키지명) --build-type (빌드타입) --dependencies (패키지1) ...
```

빌드 타입은 ament_cmake와 ament_python 중에서 선택하여 개발 언어를 정할 수 있습니다.

패키지 빌드

ROS2에서 특정 패키지 또는 전체 패키지를 빌드할 때는 colcon 빌드 도구를 사용합니다. colcon은 Python으로 작성된 통합 빌드 도구입니다. `-package-select` 옵션을 사용하여 특정 패키지를 빌드할 수 있습니다.

24장. ROS2의 패키지 파일

오늘 다룰 패키지 파일은 패키지 설정 파일 `package.xml`, 빌드 설정 파일 `CMakeLists.txt`, 파이썬 패키지 설정 파일 `setup.py`, 파이썬 패키지 환경 설정 파일 `setup.cfg`, RQt 플러그인

설정 파일 `plugin.xml`, 패키지 변경 로그 파일 `CHANGELOG.rst`, 라이선스 파일 `LICENSE`, 패키지 설명 파일 `README.md`입니다. 이들에 대해 자세히 알아보도록 하겠습니다.

패키지 설정 파일 (package.xml)

모든 ROS 패키지는 패키지 설정 파일(package.xml)을 반드시 포함합니다. 이 파일은 ROS 패키지의 필수 구성 요소로, 패키지 이름, 작성자, 라이선스, 의존성 패키지 등 패키지 정보를 기술합니다. 이 파일은 XML 형식으로 작성되며, 파일 이름은 `package.xml`입니다. 이 파일은 빌드 툴과 의존성 패키지를 모두 기술하고 있기 때문에, 빌드, 패키지 설치 및 사용에 있어서 매우 중요합니다. 패키지 설정 파일에 기술된 각 아이템에 대한 간단한 설명은 아래와 같습니다.

`<?xml>` 문서 문법을 정의하는 문구로 아래의 내용은 xml 버전 1.0을 따르고 있다는 것을 알립니다.

`<package>` 이 구문부터 맨 끝의 </package>까지가 ROS 패키지 설정 부분입니다. 세부사항으로 format="3" 이라고 패키지 설정 파일의 버전을 기재합니다. ROS 2는 3을 사용하면 됩니다.

`<name>` 패키지의 이름입니다. 패키지를 생성할 때 입력한 패키지 이름이 사용됩니다. 다른 옵션도 마찬가지지만 이는 사용자가 원할 때 언제든지 변경할 수 있습니다.

`<version>` 패키지의 버전입니다. 자유롭게 지정할 수 있지만, 나중에 패키지를 바이너리 패키지로 공개한다면 버전 관리에 사용되므로 신중할 필요가 있습니다.

`<description>` 패키지의 간단한 설명입니다. 보통 2~3 문장으로 기술합니다.

`<maintainer>` 패키지 관리자의 이름과 이메일 주소를 기재합니다.

`<license>` 라이선스를 기재합니다. Apache 2.0, BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3, Proprietary 등을 기재하면 됩니다.

`<url>` 패키지를 설명하는 웹 페이지 또는 버그 관리, 소스 코드 저장소 등의 주소를 기재합니다. 이 종류에 따라 type에 website, bugtracker, repository를 대입하면 됩니다.

`<author>` 패키지 개발에 참여한 개발자의 이름과 이메일 주소를 적습니다. 복수의 개발자가 참여한 경우에는 바로 다음 줄에 <author> 태그를 이용하여 추가로 넣어주면 됩니다.

`<buildtool_depend>` 빌드 툴의 의존성을 기술합니다.

`<build_depend>` 패키지를 빌드할 때 필요한 의존 패키지 이름을 적습니다.

`<exec_depend>` 패키지를 실행할 때 필요한 의존 패키지 이름을 적습니다.

`<test_depend>` 패키지를 테스트할 때 필요한 의존 패키지 이름을 적습니다.

CMakeList.txt

ROS에서 CMake를 이용하는 이유는 ROS 패키지를 멀티 플랫폼에서 빌드할 수 있게 하기 위함입니다. ROS 2의 빌드 시스템인 ament에서는 C++ 프로그래밍 언어를 사용한 패키지나 RQt Plugin의 경우 CMake(Cross Platform Make)를 이용합니다. 패키지 폴더의 `CMakeLists.txt`라는 파일에 빌드 환경을 기술하여 사용하고 있습니다. 이 빌드 설정 파일에서는 실행 파일 생성, 의존성 패키지 우선 빌드, 링크 생성 등을 설정합니다.

setup.py

`ROS 2 Python` 패키지에서는 `CMakeLists.txt` 파일 대신 `setup.py` 파일을 사용합니다. 이 파일은 `ROS 2 C++` 패키지의 `CMakeLists.txt`와 `package.xml`의 역할을 합니다. `setup.py` 파일은 setuptools을 사용하여 다양한 배포를 위한 설정을 하게 됩니다. 그러나 `package.xml` 파일은 `ROS` 패키지의 필수 구성 요소이므로, 비슷한 내용을 포함하더라도 패키지에 반드시 포함되어야 합니다.

`name`: 패키지 이름을 기입합니다.

`version`: 패키지 버전을 기입합니다.

`packages`: 의존하는 패키지를 하나씩 나열해도 되지만, `find_packages()`를 사용하면 자동으로 의존하는 패키지를 찾아줍니다.

`data_files`: 이 패키지에서 사용되는 파일들을 함께 배포하기 위해 기입합니다.

- `ROS`에서는 주로 `resource` 폴더 내에 있는 `ament_index`를 위한 패키지의 이름의 빈 파일이나 `package.xml`, `.launch.py`, `.yaml` 등을 기입합니다.

`install_requires`: 이 패키지를 설치할 때 함께 설치할 패키지를 기입합니다.

- `ROS`에서는 `pip`로 설치하지 않으므로 `setuptools`, `launch`만을 기입합니다.

`tests_require`: 이 패키지를 테스트하는데 필요한 패키지를 기입합니다. `ROS`에서는 `pytest`를 사용합니다.

`zip_safe`: 설치할 때 zip 파일로 아카이브할지 여부를 설정합니다.

`keywords`: 이 패키지의 키워드를 기입합니다. Python Package Index (PyPI) [8] 에서 검색하여 이 패키지를 찾을 수 있도록 합니다.

`author`, **`author_email`**, **`maintainer`**, **`maintainer_email`**: 저작자와 관리자의 이름과 이메일을 기입합니다.

`description`: 패키지 설명을 기입합니다.

`license`: 라이선스 종류를 기입합니다.

`entry_points`: 플랫폼 별로 콘솔 스크립트를 설치하도록 콘솔 스크립트 이름과 호출 함수를 기입합니다.