ROS 2 6주

21장 ROS 2의 시간

시계와 시간

- ROS 2는 여러 노드들이 서로 통신하며 다양한 정보들을 주고 받기 때문에 해당 정보들이 퍼블리시된 정확한 시간이 필수

  → 퍼블리시되는 토픽에 주요 데이터뿐만 아니라 해당 토픽이 퍼블리시되는 시간을 함께 포함시킬 수 있도록 함

- 기본 시계 : System Clock
  - rclcpp : std::chorono, rclpy : time 모듈을 캡슐화하여 사용
  - 국제 표준시인 협정 시계시(UTC, Coordinated Universal Time)으로 표시되어 전 세계 어디서든 사용 가능

시간 추상화(Time Abstractions)

- 기본 시계 이외에 타임머신처럼 작용하는 시계 사용 가능
  - 과거 시점으로 시간을 돌려줌
  - 시간을 더 빠르게 흘러가게 함
  - 시간을 멈춤

  → 과거에 기록한 데이터를 다룰 때나 로봇 시뮬레이션에서 사용

  → 개발된 알고리즘을 보다 효율적으로 디버깅

system time

System Clock을 사용한 시간

server pc와 remote pc간의 데이터 통신을 원활히 하기 위해서는 시간을 동기화시켜야만함

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

ROS Time

시뮬레이션 환경에서 시간을 조절하기 위해 사용

노드가 생성되기 전에 use_sim_time을 통해 사용

use_sim_time이 True로 설정된 노드는 /clock 토픽을 서브스크라이브할 때까지 시간을 0으로 초기화

```
$ ros2 run time_rclcpp_example time_example --ros-args -p use_sim_time:=True
```

Steady Time

Hardware timeouts를 사용한 시간

다른 시간들과 달리 무조건 단조증가

Time API

ROS 2에서 제공하는 시간과 관련된 API는 time, duration, rate가 있음

1) Time

시간을 다룰 수 있는 오퍼레이터 제공

결과를 seconds 혹은 nanoseconds 단위로 반환

2) Duration

기간을 다룰 수 있는 오퍼레이터를 제공

seconds/nanoseconds 단위로 반환

이전시간은 음수로 표시

TIme과의 연산 가능

3) Rate

반복문에서 특정 주기를 유지시켜 주는 API를 제공

콜백함수를 사용하는 Timer API를 제공하고 있기 때문에 이것을 사용하여 주기를 맞춰주면 좋음

22장 ROS 2의 파일 시스템

패키지와 메타패키지

- 패키지 : 소프트웨어 구성을 위한 기본 단위, 노드를 하나 이상 포함하거나 다른 노드를 실행하기 위한 런치와 같은 실행 및 설정 파일들을 포함
- 메타 패키지 : 공통된 목적을 지닌 패키지들을 모아둔 패키지의 집합 단위

바이너리 설치와 소스코드 설치

ROS 패키지 설치 방법

1. 바이너리 형태로 제공되어 빌드과정 없이 바로 실행
2. 소스코드를 직접 내려 받은 후 사용자가 빌드해 사용 (수정이나 소스코드 내용을 확인할 경우)

1) 바이너리 설치

해당 패키지 명으로 설치, 설치된 파일은 /opt/ros/foxy에 저장

```
$ sudo apt install ros-foxy-teleop-twist-joy
```

2) 소스코드 설치

사용자의 작업 폴더(ex. ~/robot_ws/src)에 git clone 명령어를 통해 원격의 리포지터리를 복사한 후 빌드

```
$ cd ~/robot_ws/src
$ git clone https://github.com/ros2/teleop_twist_joy.git
$ cd ~/robot_ws/
$ colcon build --symlink-install --packages-select teleop_twist_joy
```

기본 설치 폴더와 사용자 작업 폴더

1. 기본 설치 폴더
   - 기본 설치 폴더 경로 : /opt/ros/foxy
2. 사용자 작업 폴더

   사용자 작업 폴더 경로 : /home/kimms19/robot_ws

23장 ROS 2의 빌드 시스템과 빌드 툴

- 빌드 시스템 : 단일 패키지 대상
  - 단일 패키지에서 동작하는 것

    - 단일 패키지의 의존성을 해결하고 실행 가능한 파일을 생
- 빌드 툴 : 시스템 전체 대상
    - 각 패키지에 기술되어 있는 종속성 그래프를 해석
    - 토폴로지 순서로 각 패키지에 대한 특정 빌드 시스템을 호출
    - 개발 환경을 설정하고 빌드 시스템을 호출
    - 빌드 된 패키지를 사용하도록 실행 환경 구성
    - rosbuild, catkin_make, catkin_make_isolated, colon 등
- 단일 패키지 vs 시스템 전체 : 의존성 차이

1. 빌드 시스템
    - ament
        - ament_cmake : catkin의 업그레이드 버전으로 CMake의 빌드 설정 파일인 CMakeList.txt에 기술된 빌드 설정을 기반으로 빌드를 수
2. 빌드 툴

colcon : 작업 흐름을 향상시키는 CLI 타입의 명령어 도구

- catkin_make : ROS 1 대표 빌드 툴
- catkin_make_isolated : 하나의 CMake으로 복수의 패키지를 빌드, 격리 빌드를 지원함으로써 모든 패키지를 별도로 빌드 → 설치용 폴더를 분리하거나 병합할 수 있게 됨
- catkin_tools : Python으로 구성된 패키지도 관리할 수 있게 해주는 툴
- ament_tools : ROS 2의 ament_cmake 및 ament_python, 순수 CMake 패키지를 모두 지원하는 툴, ROS 2 Bouncy 버전 이전까지 사용됨
- colcon : ROS 1과 ROS 2 모두를 지원하기 위하여 통합된 빌드 툴, ROS 2 Bouncy 이후 ROS 2의 기본 빌드 툴로 사용중

3. 패키지 생성

```
$ ros2 pkg create [패키지이름] --build-type [빌드 타입] --dependencies [의존하는패키지1] [의존하는패키지n]
```

```
$ ros2 pkg create test_pkg_rclcpp --build-type ament_cmake
```

```
$ cd ~/robot_ws/src
```

```
$ ros2 pkg create my_first_ros_rclcpp_pkg --build-type ament_cmake --dependencies rclcpp std_msgs
```

4. 패키지 빌드

빌드 명령어

```
$ cd ~/robot_ws && colcon build --symlink-install
```

```
$ cd ~/robot_ws && colcon build --symlink-install --packages-select [패키지 이름]
```

5. 빌드 시스템에 필요한 부가 기능

1) vcstool(비전 컨트롤 시스템 툴)

Version Control System의 약자

여러 리포지토리 작업을 보다 쉽게 관리할 수 있도록 설계된 버전 관리 시스템

2) rosdep(의존성 관리 툴)

package.xml에 기술된 의존성 정보를 가지고 의존성 패키지들을 설치

3) bloom(바이너리 패키지 관리 툴)

바이너리 패키지 배포 및 관리를 위한 툴

바이너리 패키지를 관리하기 위한 메타 데이터 생성

개발이 끝나고 배포하고 유지 관리할 때 사용

24장 ROS 2의 패키지 파일

1. 패키지 설정 파일(package.xml)

   패키지의 정보를 기술(패키지 이름, 저작자, 라이선스 등에 대해)

   각 패키지당 무조건 1개의 패키지 설정 파일을 포함하는 패키지의 필수 파일

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_first_ros_rclcpp_pkg</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="pyo@robotis.com">pyo</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>rclcpp</depend>
  <depend>std_msgs</depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

2. 빌드 설정 파일(CMakeList.txt)

   이 빌드 설정 파일에 실행 파일 생성, 의존성 패키지 우선 빌드, 링크 생성

```
cmake_minimum_required(VERSION 3.5)
project(my_first_ros_rclcpp_pkg)

# Default to C99
if(NOT CMAKE_C_STANDARD)
  set(CMAKE_C_STANDARD 99)
endif()

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
```

```
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()
```

3. 파이썬 패키지 설정 파일 (setup.py)

```
from setuptools import setup

package_name = 'my_first_ros_rclpy_pkg'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='pyo',
    maintainer_email='pyo@robotis.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        ],
    },
)
```

4. 파이썬 패키지 설정 파일(setup.cfg)

   ros 2 python 패키지에서만 사용하는 배포를 위한 구성 파일

5. RQt 플러그인 설정 파일(plugin.xml)
6. 패키지 변경로그 파일(CHANGELOG.rst)

   패키지 업데이트 내역 기술

   개발 이력을 추적할 때 도움

   개발된 패키지를 바이너리 패키지로 공개하는 절차를 수행할 경우 필수로 포함시켜야함

7. 라이선스파일(LICENSE)

   패키지의 코드에 사용된 라이선스 기술

8. 패키지 설명 파일 (README.md)

   패키지의 부가 설명을 기술