ROS2 Ubuntu Cheat Sheet (edit 2023.07.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

turtlesim

Ubuntu 버전 확인
```
$ lsb release -d
```

turtlesim 패키지 설치
```
$ sudo apt update
$ sudo apt install ros-foxy-turtlesim
$ cd /opt/ros/foxy/include ¦ ls
```

turtlesim 실행 파일 목록
```
$ ros2 pkg executables turtlesim
```

turtlesim 패키지 노드 실행
```
$ ros2 run <package_name> <executable_name>
$ ros2 run turtlesim turtlesim_node
$ ros2 run turtlesim turtle_teleop_key
```
터미널 창 선택 후, 키보드 화살표 키로 거북이 움직임
중지하고 싶은 창에서 [Ctrl + C] 클릭시 멈춤

rqt

rqt 실행
```
$ rqt
$ rqt_graph
```

node

노드 목록 확인
```
$ ros2 node list
```

노드 정보 확인
```
$ ros2 node info <node_name>
$ ros2 node info /turtlesim
```

노드 리매핑
```
$ ros2 run turtlesim turtlesim_node --ros-args --remap __node:=sohi_turtle
```

topic

토픽 목록 확인
```
$ ros2 topic list
$ ros2 topic list -t
```

토픽 정보 확인
```
$ ros2 topic info <topic_name>
$ ros2 topic info /turtle1/cmd_vel
```

토픽 타입 확인
```
$ ros2 topic type <topic_name>
$ ros2 topic type /turtle1/cmd_vel
```

토픽 내용 확인
```
$ ros2 topic echo <topic_name>
$ ros2 topic echo /turtle1/cmd_vel
```

토픽 대역폭 확인 (송수신 메시지 크기)
```
$ ros2 topic bw <topic_name>
$ ros2 topic bw /turtle1/cmd_vel
```

토픽 전송 주기 확인 (topic을 publish하는 주기)
```
$ ros2 topic hz <topic_name>
$ ros2 topic hz /turtle1/cmd_vel
```

토픽 지연 시간 확인 (header stamp 메시지 사용시 가능)
```
$ ros2 topic delay <topic_name>
$ ros2 topic delay /turtle1/cmd_vel
```

publish

토픽 퍼블리시(publish)
```
$ ros2 topic pub <topic_name> <msg_type> '<args>'
  --once : 하나의 메시지를 게시한 다음 종료
  --rate 1 : 전송 주기 1hz마다 publish
$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}'
$ ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}'
```
x-z값 입력시 콜론(:) 뒤에 한 칸 공백 꼭 있어야 함.

service

서비스 목록 확인
```
$ ros2 service list
$ ros2 service list -t
```

서비스 형태 확인
```
$ ros2 service type <service_name>
$ ros2 service type /clear
```

서비스 찾기
```
$ ros2 service find <type_name>
$ ros2 service find std_srvs/srv/Empty
```

서비스 요청
```
$ ros2 service call <service_name> <service_type> <arguments>
$ ros2 service call /clear std_srvs/srv/Empty
$ ros2 service call /spawn turtlesim/srv/Spawn '{x: 2, y: 2, theta: 0.2, name: 'sohi'}'
$ ros2 service call /kill turtlesim/srv/Kill '{name: 'sohi'}'
$ ros2 service call /reset std_srvs/srv/Empty
$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{r: 255, g: 255, b: 255, width: 10}'
```

parameter

파라미터 목록 확인
```
$ ros2 param list
```

파라미터 내용 확인
```
$ ros2 param describe /turtlesim background_b
```

파라미터 값 얻기
```
$ ros2 param get <node_name> <parameter_name>
$ ros2 param get /turtlesim background_g
```

파라미터 값 설정
```
$ ros2 param set <node_name> <parameter_name> <value>
$ ros2 param set /turtlesim background_r 150
```

파라미터 저장
```
$ ros2 param dump <node_name>
$ ros2 param dump /turtlesim
```

파라미터 로드
```
$ ros2 param load <node_name> <parameter_file>
$ ros2 param load /turtlesim ./turtlesim.yaml
```

파라미터 삭제
```
$ ros2 param delete <node_name> <parameter_name>
$ ros2 param delete /turtlesim background_g
```

ROS2 CLI tools Cheat Sheet (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

turtlesim

turtlesim 패키지 설치 여부 확인
```
$ ros2 pkg list
```

turtlesim 패키지 설치
```
$ sudo apt update
$ sudo apt install ros-foxy-turtlesim
$ cd /opt/ros/foxy/include ¦ ls
```

turtlesim 실행 파일 목록
```
$ ros2 pkg executables turtlesim
```

turtlesim 패키지 노드 실행
```
$ ros2 run <package_name> <executable_name>
$ ros2 run turtlesim turtlesim_node
$ ros2 run turtlesim turtle_teleop_key
```
터미널 창 선택 후, 키보드 화살표 키로 거북이 움직임
중지하고 싶은 창에서 [Ctrl + C] 클릭시 멈춤

rqt

rqt 실행
```
$ rqt
$ rqt_graph
```

node

노드 목록 확인
```
$ ros2 node list
```

노드 정보 확인
```
$ ros2 node info <node_name>
$ ros2 node info /turtlesim
```

노드 리매핑
```
$ ros2 run turtlesim turtlesim_node --ros-args --
remap __node:=sohi_turtle
```

topic

토픽 목록 확인
```
$ ros2 topic list
$ ros2 topic list -t
```

토픽 정보 확인
```
$ ros2 topic info <topic_name>
$ ros2 topic info /turtle1/cmd_vel
```

토픽 타입 확인
```
$ ros2 topic type <topic_name>
$ ros2 topic type /turtle1/cmd_vel
```

토픽 내용 확인
```
$ ros2 topic echo <topic_name>
$ ros2 topic echo /turtle1/cmd_vel
```

토픽 대역폭 확인 (송수신 메시지 크기)
```
$ ros2 topic bw <topic_name>
$ ros2 topic bw /turtle1/cmd_vel
```

토픽 전송 주기 확인 (topic을 publish하는 주기)
```
$ ros2 topic hz <topic_name>
$ ros2 topic hz /turtle1/cmd_vel
```

토픽 지연 시간 확인 (header stamp 메시지 사용시 가능)
```
$ ros2 topic delay <topic_name>
$ ros2 topic delay /turtle1/cmd_vel
```

publish

토픽 퍼블리시(publish)
```
$ ros2 topic pub <topic_name> <msg_type> '<args>'
  --once : 하나의 메시지를 게시한 다음 종료
  --rate 1 : 전송 주기 1hz마다 publish
$ ros2 topic pub --once /turtle1/cmd_vel
geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z:
0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}'
$ ros2 topic pub --rate 1 /turtle1/cmd_vel
geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z:
0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}'
```
x-z값 입력시 콜론(:) 뒤에 한 칸 공백 꼭 있어야 함.

service

서비스 목록 확인
```
$ ros2 service list
$ ros2 service list -t
```

서비스 타입 확인
```
$ ros2 service type <service_name>
$ ros2 service type /clear
```

서비스 타입에 해당하는 서비스 찾기
```
$ ros2 service find <type_name>
$ ros2 service find std_srvs/srv/Empty
```

서비스 요청
```
$ ros2 service call <service_name> <service_type>
  <arguments>
$ ros2 service call /clear std_srvs/srv/Empty
$ ros2 service call /spawn turtlesim/srv/Spawn '{x:
2, y: 2, theta: 0.2, name: 'sohi'}'
$ ros2 service call /kill turtlesim/srv/Kill
'{name: 'sohi'}'
$ ros2 service call /reset std_srvs/srv/Empty
$ ros2 service call /turtle1/set_pen
turtlesim/srv/SetPen '{r: 255, g: 255, b: 255, width:
10}'
```

parameter

파라미터 목록 확인
```
$ ros2 param list
```

파라미터 내용 확인
```
$ ros2 param describe /turtlesim background_b
```

파라미터 값 얻기
```
$ ros2 param get <node_name> <parameter_name>
$ ros2 param get /turtlesim background_g
```

파라미터 값 설정
```
$ ros2 param set <node_name> <parameter_name>
  <value>
$ ros2 param set /turtlesim background_r 150
```

파라미터 저장
```
$ ros2 param dump <node_name>
$ ros2 param dump /turtlesim
```

파라미터 로드
```
$ ros2 param load <node_name> <parameter_file>
$ ros2 param load /turtlesim ./turtlesim.yaml
```

파라미터 삭제
```
$ ros2 param delete <node_name> <parameter_name>
$ ros2 param delete /turtlesim background_r
```

ROS2 CLI tools Cheat Sheet (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

turtlesim

turtlesim 패키지 노드 실행(1개 노드)

```
$ ros2 run turtlesim turtlesim_node
$ ros2 run turtlesim turtle_teleop_key
  터미널 창 선택 후, 키보드 화살표 키로 거북이 움직임
  키보드 G¦B¦V¦C¦D¦E¦R¦T 키로 거북이 회전, F 키로 취소
```

node

노드 확인

```
$ ros2 node list
$ ros2 node info /turtlesim
$ ros2 node info /teleop_turtle
```

노드 시작시 파라미터 파일 로드

```
$ ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>
$ ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```

action

액션 목록(list) 확인

```
$ ros2 action list
$ ros2 action list –t
```

액션 정보(information) 확인

```
$ ros2 action info <action_name>
$ ros2 action info /turtle1/rotate_absolute
```

액션 목표(goal) 설정

```
$ ros2 action send_goal <action_name> <action_type> <values>
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute '{theta: 1.57}'
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute '{theta: -1.57}' --feedback
```

interface

인터페이스 목록 확인

```
$ ros2 interface list
```

패키지에서 사용 가능한 인터페이스 확인

```
$ ros2 interface package <package_name>
$ ros2 interface package std_msgs
```

인터페이스를 제공하는 패키지 목록 확인

```
$ ros2 interface packages
$ ros2 interface packages --only-msgs
```

인터페이스의 기본 타입 표시

```
$ ros2 interface proto <interface_name>
$ ros2 interface proto geometry_msgs/msgs/Twist
```

토픽 인터페이스 확인

```
$ ros2 interface show <msg type>
$ ros2 interface show geometry_msgs/msg/Twist
```

서비스 인터페이스 확인

```
$ ros2 interface show <type_name>.srv
$ ros2 interface show std_srvs/srv/Empty.srv
$ ros2 interface show turtlesim/srv/Spawn
```

액션 인터페이스 확인

```
$ ros2 interface show <type_name>
$ ros2 interface show turtlesim/action/RotateAbsolute
```

launch

런치 실행(2개 이상 노드)

복수의 노드를 함께 실행할 경우

```
$ ros2 launch <package_name> <launch_file_name>
$ ros2 launch demo_nodes_cpp add_two_ints.launch.py
```

list

목록 확인

```
$ ros2 node list
$ ros2 topic list
$ ros2 service list
$ ros2 action list
$ ros2 param list
```

bag

폴더 생성

```
$ mkdir bag_files
$ cd bag_files
```

토픽 선택

```
$ ros2 topic list
$ ros2 topic echo /turtle1/cmd_vel
```

Bag 데이터 확인

```
$ ros2 bag info <bag_file_name>
$ ros2 bag info test.bag
```

Bag 데이터 기록

```
$ ros2 bag record <topic_name>
$ ros2 bag record /turtle1/cmd_vel
  turtlesim 움직이기, Ctrl+C로 레코딩 멈추기
  -a는 모든 토픽 기록

$ ros2 bag record -o <bag_file_name> <topic_name>
$ ros2 bag record –o test.bag /turtle1/cmd_vel /turtle1/pose
```

Bag 데이터 재생

```
$ ros2 bag play <bag_file_name>
$ ros2 bag play test.bag
```

interface file

메시지 인터페이스(msg)

필드 타입과 필드 이름으로 구성

```
fieldtype1 fieldname1
fieldtype2 fieldname2
fieldtype3 fieldname3
```

서비스 인터페이스(srv)

요청(request) 메시지와 응답(response) 메시지로 구성
대시(-)로 구분

```
fieldtype1 fieldname
---
fieldtype1 fieldname
```

액션 인터페이스(action)

액션 목표(goal), 액션 결과(result),
액션 피드백(feedback)으로 구성, 대시(-)로 구분

```
fieldtype1 fieldname
---
fieldtype1 fieldname
---
fieldtype1 fieldname
```

https://github.com/ros2/common_interfaces/tree/foxy
reference : https://docs.ros.org/en/foxy/Tutorials.html

ROS2 Hello Programming (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

Workspaces

워크 스페이스(Workspaces)
원하는 작업 수행을 위한 code를 작성하는 공간
워크 스페이스 위치 : ~/Workspaces/<ws_name>

디렉토리 생성
```
$ mkdir <dir_name>
$ mkdir <dir_name1> <dir_name2> <dir_name3>
  dir_name을 여러개 쓰면, 여러 폴더를 동시에 생성함
  -p : 계층형 디렉토리를 생성함
```

워크 스페이스(ros2_ws)와 소스폴더(src) 생성
```
$ mkdir -p ~/Workspaces/ros2_ws/src
```

Package

패키지(Package)
파이썬 모듈을 계층적(디렉터리 구조)으로 관리함.

패키지 생성
```
$ ros2 pkg create <package_name> --build-type
  <build_type> --dependencies <package1> <package2>
  --build-type : ament_python 파이썬으로 코딩한 파일 빌드
  --dependencies : 의존하는 패키지들(import한 패키지 작성)
    https://docs.ros2.org/foxy/api/rclpy/
```

디렉토리 이동
```
$ cd <dir_name>
```

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create hello_pkg --build-type
ament_python --dependencies rclpy
```

hello_ros2.py

노드 프로그램 파일
```
$ cd ~/Workspaces/ros2_ws/src/hello_pkg/hello_pkg
$ code ./hello_ros2.py
```

```
import rclpy
from rclpy.node import Node

def main(args=None):
    rclpy.init(args=args)
    node = Node('hello_node')
    node.get_logger().info('Hello ROS2')
    rclpy.spin(node)
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

setup.py

파이썬 패키지 설정 파일
ROS2 파이썬 패키지에서 배포를 위한 파일
```
$ cd ~/Workspaces/ros2_ws/src/hello_pkg
$ code setup.py
```

entry_points 옵션의 console_scripts 키를 사용한 실행 파일의 설정을 함. 예) hello_script 콘솔 스크립트는 hello_pkg.hello_ros2 모듈의 main 함수를 호출함

```
  entry_points={
    'console_scripts': [
      'hello_script = hello_pkg.hello_ros2:main'
    ],
  },
```

colcon build

빌드(Build)
소스 코드 파일을 컴퓨터에서 실행할 수 있는 독립적인 형태로 변환함
```
$ cd ~/Workspaces/<ws_name>
  Home폴더에 위 경로로 폴더를 생성한 후 이동함.
$ colcon build
  각 패키지에 기술되어 있는 종속성 그래프를 해석하고 토폴로지 순서로 각 패키지에 대한 특정 빌드 시스템을 호출함.
$ source install/setup.bash
  설정 스크립트를 적용함. 패키지 빌드 후 setup.bash를 실행해야 ROS 2 패키지를 찾거나 노드를 실행하는 것이 가능함.
```

빌드 옵션
```
$ colcon build --symlink-install
  --packages-select <package_name> <package_name>
   --packages-up-to <package_name>
```

--symlink-install : 파일 복사 대신 심볼릭 링크 형태로 저장 심볼릭 링크(symbolic link) : 링크를 연결하여 원본 파일을 직접 사용하는 것과 같은 효과를 내는 링크
--packages-select 옵션 : 특정 패키지만 선택해서 빌드 특정 패키지 및 의존성 패키지를 함께 빌드
--packages-up-to 옵션 : 특정 패키지의 첫 빌드 후에는 환경설정 파일을 불러와서 실행 가능한 패키지의 노드 설정을 해줘야 빌드 된 노드 실행 가능함

파이썬 파일 실행
```
$ cd ~/Workspaces/ros2_ws/src/hello_pkg/hello_pkg
$ python3 hello_ros2.py
```

빌드 후 노드 실행(ros2 run)
```
$ cd ~/Workspaces/ros2_ws
$ colcon build
$ source install/setup.bash
$ ros2 pkg executables hello_pkg
$ ros2 run hello_pkg hello_script
```

setup script

설정 스크립트
새로운 패키지를 빌드 시 설정 스크립트를 터미널에서 실행해야 함.

local_setup.bash : 워크스페이스의 모든 패키지에 대한 환경을 설정함
setup.bash : 워크스페이스와 ROS가 설치된 개발환경에 대한 환경을 설정함

bash : interactive shell, user input에 의해 script가 실행되는 shell mode
~/.bashrc : bash로 실행될 때 먼저 자동으로 실행되는 파일. 새로운 터미널을 열 때마다 파일에 적힌 명령어가 수행됨

~/.bashrc 파일에 다음 내용 추가하면 터미널 열 때 명령어 수행이 되어, 빌드 후 source install/setup.bash를 하지 않아도 됨. ~/.bashrc에 다음 내용을 저장한 후 사용하는 것을 추천함.

```
export my_ws=~/Workspaces/ros2_ws
source /opt/ros/foxy/setup.bash
source $my_ws/install/setup.bash

alias eb='code ~/.bashrc'
alias nb='nano ~/.bashrc'
alias sb='source ~/.bashrc'
alias cw='cd '$my_ws
alias cs='cd '$my_ws'/src'
alias cb='cd '$my_ws' && colcon build --symlink-
install && source ~/.bashrc'
```

 https://github.com/sohicode/ros2_basics/

SOHI CODE

패키지 생성 Package

```
  $ cd ~/Workspaces/ros2_ws/src
  $ ros2 pkg create oop_pkg --build-type ament_python
--dependencies rclpy
```

노드 프로그램 파일 oop_ros2.py

```
  $ cd ~/Workspaces/ros2_ws/src/oop_pkg/oop_pkg
  $ code ./oop_ros2.py
```

```
#! /usr/bin/env python3
import rclpy
from rclpy.node import Node

class OopNode(Node):

    def __init__(self):
        super().__init__('oop_node')
        self.counter_ = 0
        self.get_logger().info('Hello World!')
        self.create_timer(0.5, self.timer_cb)

    def timer_cb(self):
        self.counter_ += 1
        self.get_logger().info('Hello ' +
str(self.counter_))

def main(args = None):
    rclpy.init(args=args)
    node = OopNode()
    rclpy.spin(node)
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

설정 스크립트 setup.py

ROS2 파이썬 패키지에서 배포를 위한 파일
~/Workspaces/ros2_ws/src/oop_pkg/setup.py

```
from setuptools import setup
package_name = 'oop_pkg'

setup(
    name=package_name,
    version='0.1.0',
    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='sohi',
    maintainer_email='sohicode@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'oop_script = oop_pkg.oop_ros2:main'
        ],
    },
)
```

패키지 설정 파일 package.xml

ROS 패키지의 정보를 기술하는 파일
패키지 생성시 --dependencies에 기술한 패키지는 <depend>에 자동 기록됨.
생성시 의존성 패키지 기록 안한 경우엔 패키지 설정 파일에 추가해야 함.

~/Workspaces/ros2_ws/src/oop_pkg/package.xml

```
<?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>oop_pkg</name>
  <version>0.0.0</version>
  <description>TODO: Package
description</description>
  <maintainer
email='sohicode@gmail.com'>sohi</maintainer>
  <license>TODO: License declaration</license>

  <depend>rclpy</depend>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

파이썬 패키지 환경설정 파일 setup.cfg

빌드 시 지정 폴더에 실행 파일이 생성됨.
~/Workspaces/ros2_ws/src/oop_pkg/setup.cfg

$base = /home/<user_name>/<ws_name>/install/oop_pkg/
워크스페이스의 install폴더에 <package_name>폴더를 $base로 지정하고, $base/lib/<package_name> 폴더에 script파일 저장함

```
[develop]
script-dir=$base/lib/oop_pkg
[install]
install-scripts=$base/lib/oop_pkg
```

빌드 build

설정스크립트에 기록한 별칭 cw, cb, sb를 활용

```
  $ cw
  $ cb
  $ sb
```

```
  $ cd ~/Workspaces/ros2_ws
  $ colcon build --symlink-install --packages-select
oop_pkg
  $ source ~/Workspaces/ros2_ws/install/setup.bash
```

노드 실행 run

```
  $ ros2 pkg executables oop_pkg
  $ ros2 run oop_pkg oop_script
```

ROS2 Simple Topic Programming (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

패키지 생성 — package

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create simple_topic_pkg --build-type
  ament_python --dependencies rclpy std_msgs
```

package.xml 중 일부

```
<depend>rclpy</depend>
<depend>std_msgs</depend>
```

메시지 인터페이스 — String.msg

라이브러리 위치: /opt/ros/foxy/include/std_msgs/msg/
참고사이트: https://github.com/ros2/common_interfaces/blob/foxy/std_msgs/msg/String.msg

```
# This was originally provided as an example message.
# It is deprecated as of Foxy
string data
```

토픽 발행 프로그램 파일 — publisher.py

$my_ws/src/simple_topic_pkg/simple_topic_pkg/publisher.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class  MyPublisher(Node):

    def __init__(self):
        super().__init__('pub_node')
        self.counter = 0
        self.pub = self.create_publisher(String,
'my_topic', 10)
        self.timer = self.create_timer(1, self.pub_cb)
        self.get_logger().info('Publisher Node
Running...')

    def pub_cb(self):
        self.counter += 1
        msg = String()
        msg.data = 'hi: ' + str(self.counter)
        self.pub.publish(msg)
        self.get_logger().info('Published message: '
+ msg.data)


def main(args=None):
    rclpy.init(args=args)
    node = MyPublisher()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

토픽 수신 프로그램 파일 — subscriber.py

$my_ws/src/simple_topic_pkg/simple_topic_pkg/subscriber.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class MySubscriber(Node):

    def __init__(self):
        super().__init__('sub_node')
        self.sub = self.create_subscription(String,
'my_topic', self.sub_cb, 10)
        self.get_logger().info('Subscriber Node
Running...')

    def sub_cb(self, msg):
        self.get_logger().info('Received message: ' +
msg.data)

def main(args=None):
    rclpy.init(args=args)
    node = MySubscriber()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

파이썬 패키지 설정 파일 — setup.py

```
$ cd ~/Workspaces/ros2_ws/src/topic_pkg
$ code setup.py
  ROS2 파이썬 패키지에서 배포를 위한 설정 파일
```

```
entry_points={
  'console_scripts': [
    'publisher_script = simple_topic_pkg.publisher:main',
    'subscriber_script = simple_topic_pkg.subscriber:main'
  ],
},
```

빌드 — build

```
$ cd ~/Workspaces/ros2_ws
$ colcon build
$ source ./install/setup.bash
```

노드 실행 — run

```
$ ros2 pkg executables simple_topic_pkg
$ ros2 run simple_topic_pkg subscriber_script
$ ros2 run simple_topic_pkg publisher_script
  두 개의 창에 각각 실행함
```


패키지 생성 — package

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create rpm_topic_pkg --build-type
  ament_python --dependencies rclpy std_msgs
```

메시지 인터페이스 — Float32.msg

참고사이트: https://github.com/ros2/common_interfaces/blob/foxy/std_msgs/msg/Float32.msg

```
float32 data
```

rpm값 토픽 발행 프로그램 파일 — rpm_pub.py

$my_ws/src/rpm_topic_pkg/rpm_topic_pkg/rpm_pub.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32

RPM = 10

class  RpmPublisher(Node):

    def __init__(self):
        super().__init__('rpm_pub_node')
        self.pub = self.create_publisher(Float32,
'rpm_topic', 10)
        self.timer = self.create_timer(2,
self.rpm_pub_cb)
        self.get_logger().info('RPM Publisher Node
Running...')

    def rpm_pub_cb(self):
        msg = Float32()
        msg.data = float(RPM)
        self.pub.publish(msg)
        self.get_logger().info('Published message: '
+ str(msg.data))

def main(args=None):
    rclpy.init(args=args)
    node = RpmPublisher()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

파이썬 패키지 설정 파일 — setup.py

$my_ws/src/rpm_topic_pkg

```
'console_scripts': [
  'rpm_pub_script = rpm_topic_pkg.rpm_pub:main',
  'speed_calc_script = rpm_topic_pkg.speed_calc:main'
],
```

rpm 토픽 수신 & — speed_calc.py

rpm으로 계산한 speed 토픽 발생 프로그램 파일

$my_ws/src/rpm_topic_pkg/rpm_topic_pkg/speed_calc.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32

WHEEL_RADIUS = 12.5 / 100

class SpeedCalculator(Node):

    def __init__(self):
        super().__init__('speed_calc_node')
        self.sub = self.create_subscription(Float32,
'rpm_topic', self.speed_calc_cb, 10)
        self.pub = self.create_publisher(Float32,
'speed_topic', 10)
        self.get_logger().info('Speed Calculator Node
Started...')

    def speed_calc_cb(self, rpm_msg):
        self.get_logger().info('Received rpm message:
' + str(rpm_msg.data))
        speed = rpm_msg.data * WHEEL_RADIUS * 2 *
3.14159 / 60 # speed in m/s
        speed_msg = Float32()
        speed_msg.data = float(speed)
        self.pub.publish(speed_msg)
        self.get_logger().info('Published speed
message: ' + str(speed_msg.data))


def main(args=None):
    rclpy.init(args=args)
    node = SpeedCalculator()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()


if __name__ == '__main__':
    main()
```

빌드 — build

```
$ cd ~/Workspaces/ros2_ws
$ colcon build
$ source ./install/setup.bash
```

노드 실행 — run

```
$ ros2 pkg executables rpm_topic_pkg
$ ros2 run rpm_topic_pkg speed_calc_script
$ ros2 run rpm_topic_pkg rpm_pub_script
```

두 개의 창에 각각 실행함

 https://github.com/sohicode/ros2_basics/ SOHI CODE

토픽을 발행함 - rpm

rpm_pub.py

$my_ws/src/rpm_topic_pkg/rpm_topic_pkg/rpm_pub.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32

RPM = 10

class  RpmPublisher(Node):

    def __init__(self):
        super().__init__('rpm_pub_node')
        self.pub = self.create_publisher(Float32,
'rpm_topic', 10)
        self.timer = self.create_timer(2,
self.rpm_pub_cb)
        self.get_logger().info('RPM Publisher Node
Running...')

    def rpm_pub_cb(self):
        msg = Float32()
        msg.data = float(RPM)
        self.pub.publish(msg)
        self.get_logger().info('Published message: '
+ str(msg.data))


def main(args=None):
    rclpy.init(args=args)
    node = RpmPublisher()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

파이썬 패키지 설정 파일

setup.py

$my_ws/src/rpm_topic_pkg

```
'console_scripts': [
  'rpm_pub_script = rpm_topic_pkg.rpm_pub:main',
  'speed_calc_script = rpm_topic_pkg.speed_calc:main'
],
```

파라미터값 확인

```
    $ ros2 param get /speed_calc_node
wheel_radius_param
    $ ros2 topic echo /speed
```

파라미터값 설정

```
    $ ros2 param set /speed_calc_node
wheel_radius_param 0.5
    $ ros2 topic echo /speed
```

CLI

토픽을 수신함 - rpm
토픽을 발행함 - rpm으로 계산한 speed

speed_calc.py

$my_ws/src/rpm_topic_pkg/rpm_topic_pkg/speed_calc.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32

WHEEL_RADIUS_DEFAULT = 12.5 / 100 # centimeters to
meters
class SpeedCalculator(Node):

    def __init__(self):
        super().__init__('speed_calc_node')
        self.declare_parameter('wheel_radius_param',
WHEEL_RADIUS_DEFAULT)
        self.sub = self.create_subscription(Float32,
'rpm_topic',

self.speed_calc_cb, 10)
        self.pub = self.create_publisher(Float32,
'speed_topic', 10)
        self.get_logger().info('Speed Calculator Node
Started...')

    def speed_calc_cb(self, rpm_msg):
        self.get_logger().info('Received rpm message:
' + str(rpm_msg.data))
        wheel_radius =
self.get_parameter('wheel_radius_param').get_paramete
r_value().double_value
        speed = rpm_msg.data * wheel_radius * 2 *
3.14159 / 60  # speed in m/s
        speed_msg = Float32()
        speed_msg.data = float(speed)
        self.pub.publish(speed_msg)
        self.get_logger().info('Published speed
message: ' + str(speed_msg.data))

def main(args=None):
    rclpy.init(args=args)
    node = SpeedCalculator()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

빌드

```
  $ cd ~/Workspaces/ros2_ws
  $ colcon build
  $ source ./install/setup.bash
```

노드 실행

```
  $ ros2 pkg executables rpm_topic_pkg
  $ ros2 run rpm_topic_pkg speed_calc_script
  $ ros2 run rpm_topic_pkg rpm_pub_script
  두 개의 창에 각각 실행함
```

build & run

ROS2 Simple Service Programming (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

패키지 생성 — package

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create simple_service_pkg --build-type
  ament_python --dependencies rclpy std_msgs
```

서비스 인터페이스 — SetBool.srv

std_msgs/srv/SetBool

```
# Request
bool data # e.g. for hardware enabling / disabling
---
# Response
bool success   # indicate successful run of triggered
service
string message # informational, e.g. for error
messages
```

서비스 서버 — service_server.py

src/simple_service_pkg/simple_service_pkg/service_server.py

```
import rclpy
from rclpy.node import Node
from std_srvs.srv import SetBool

class  PowerServer(Node):
    def __init__(self):
        super().__init__('service_server_node')
        self.srv = self.create_service(SetBool,
'power_service', self.power_cb)
        self.get_logger().info('Service Server
Running...')

    def power_cb(self, request, response):
        self.get_logger().info('Request Received... ')

        if request.data:
            response.success = True
            response.message = 'Power On'
        elif not request.data:
            response.success = True
            response.message = 'Power Off'
        else:
            response.success = False
            response.message = 'Error'
        print(request)
        print(response)
        return response

def main(args=None):
    rclpy.init(args=args)
    node = PowerServer()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

서비스 클라이언트 — service_client.py

src/simple_service_pkg/simple_service_pkg/service_client.py

```
import rclpy
from rclpy.node import Node
from std_srvs.srv import SetBool

class PowerClient(Node):

    def __init__(self):
        super().__init__('service_client_node')
        self.client = self.create_client(SetBool,
'power_service')
        self.req = SetBool.Request()
        self.get_logger().info('Service Client Start')

    def send_request(self, user_input):
        self.req.data = (user_input.lower() == 'on')
        self.client.wait_for_service()
        self.future = self.client.call_async(self.req)
        rclpy.spin_until_future_complete(self,
self.future)
        self.result = self.future.result()
        return self.result

def main(args=None):
    rclpy.init(args=args)
    node = PowerClient()

    try:
        #pass
        user_input = input('Enter an power "on" or
"off" : ')
        res = node.send_request(user_input)
        node.get_logger().info('Server returned: ' +
res.message)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

setup.py

```
'console_scripts': [
    'service_server =
simple_service_pkg.service_server:main',
    'service_client =
simple_service_pkg.service_client:main'
],
```

빌드 — build & run

```
$ cd ~/Workspaces/ros2_ws
$ colcon build --symlink-install --packages-select
simple_ service_pkg
$ source
~/Workspaces/ros2_ws/install/local_setup.bash
```

노드 실행(ros2 run)

```
$ ros2 pkg executables simple_ service_pkg
$ ros2 run simple_ service_pkg service_server
$ ros2 run simple_ service_pkg service_client
```

ROS2 Interface Programming (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

Package

인터페이스 패키지 만들기

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create interface_pkg --build-type
  ament_cmake
$ cd interface_pkg
$ mkdir msg srv action
```

빌드

```
$ cd ~/Workspaces/ros2_ws
$ colcon build
$ source ./install/setup.bash
```

Rect.msg

메시지 인터페이스

src/interface_pkg/msg/Rect.msg

```
int64 width
int64 height
```

OddEvenCheck.srv

서비스 인터페이스

src/interface_pkg/srv/OddEvenCheck.srv

```
# Request
int64 number
---
# Response
string decision
```

package.xml

패키지 설정 파일

src/interface_pkg/package.xml

```
<?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>interface_pkg</name>
  <version>0.1.0</version>
  <description>
    ROS 2 interface package
  </description>
  <maintainer
email='sohicode@gmail.com'>sohi</maintainer>
  <license>Apache 2.0</license>
  <author email='sohicode@gmail.com'>sohi</author>

  <buildtool_depend>ament_cmake</buildtool_depend>
  <buildtool_depend>rosidl_default_generators
  </buildtool_depend>
  <exec_depend>rosidl_default_runtime</exec_depend>
  <member_of_group>rosidl_interface_packages
  </member_of_group>

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

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

CMakeLists.txt

빌드 설정 파일

src/interface_pkg/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(interface_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)
# uncomment the following section in order to fill in
# further dependencies manually.
# find_package(<dependency> REQUIRED)

find_package(rosidl_default_generators REQUIRED)
# find_package(builtin_interfaces REQUIRED)

set(msg_files
  "msg/Rect.msg"
)
set(srv_files
  "srv/OddEvenCheck.srv"
)
rosidl_generate_interfaces(${PROJECT_NAME}
  ${msg_files}
  ${srv_files}
#  DEPENDENCIES builtin_interfaces
)

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_export_dependencies(rosidl_default_runtime)
ament_package()
```

interface CLI

사용 가능 인터페이스 확인

```
$ ros2 interface package interface_pkg
```

인터페이스 확인

```
$ ros2 interface show interface_pkg/msg/Rect
$ ros2 interface show
interface_pkg/srv/OddEvenCheck
```

https://github.com/ros2/common_interfaces/tree/foxy
https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Single-Package-Define-And-Use-Interface.html

ROS2 Odd Even Service Programming (sohi edit 2023.08.)

 https://github.com/sohicode/ros2_basics/

SOHI CODE

서비스 패키지 만들기 — Package

```
  $ cd ~/Workspaces/ros2_ws/src
  $ ros2 pkg create oe_service_pkg --build-type
ament_python
    --dependencies rclpy std_srvs interface_pkg
```

서비스 인터페이스 — OddEvenCheck.srv

src/interface_pkg/srv/OddEvenCheck.srv

```
# Request
int64 number
---
# Response
string decision
```

서비스 서버 - 짝홀 판단값 — odd_even_server.py

src/oe_service_pkg/oe_service_pkg/odd_even_server.py

```
import rclpy
from rclpy.node import Node
from interface_pkg.srv import OddEvenCheck

class  OddEvenCheckServer(Node):

    def __init__(self):
        super().__init__('odd_even_server_node')
        self.srv = self.create_service(OddEvenCheck,
'odd_even_check', self.odd_even_cb)
        self.get_logger().info('Odd Even Check Service Server
Running...')

    def odd_even_cb(self, request, response):
        self.get_logger().info('Request Received... ')
        if request.number % 2 == 0:
            response.decision = 'Even'
        elif request.number % 2 == 1:
            response.decision = 'Odd'
        else:
            response.decision = 'Error'

        print(request)
        print(response)
        return response


def main(args=None):
    rclpy.init(args=args)
    node = OddEvenCheckServer()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

서비스 클라이언트 숫자 보내고 짝홀 판단값을 받음 — odd_even_client.py

src/oe_service_pkg/oe_service_pkg/odd_even_client.py

```
import rclpy
from rclpy.node import Node
from service_pkg.srv import OddEvenCheck

class OddEvenCheckClient(Node):
    def __init__(self):
        super().__init__('odd_even_client_node')
        self.client = self.create_client(OddEvenCheck,
'odd_even_check')
        self.req = OddEvenCheck.Request()
        self.get_logger().info('Service Client Start')

    def send_request(self, num):
        self.req.number = int(num)
        self.client.wait_for_service()
        self.future = self.client.call_async(self.req)
        rclpy.spin_until_future_complete(self,
self.future)
        self.result = self.future.result()
        return self.result

def main(args=None):
    rclpy.init(args=args)
    node = OddEvenCheckClient()

    try:
        #pass
        user_input = input('Enter an Integer: ')
        res = node.send_request(user_input)
        node.get_logger().info('Server returned: ' +
res.decision)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt')
    finally:
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

setup.py

```
'console_scripts': [
            'oe_server =
oe_service_pkg.odd_even_server:main',
            'oe_client =
oe_service_pkg.odd_even_client:main'
        ],
```

빌드 — build & run

```
  $ cd ~/Workspaces/ros2_ws
  $ colcon build --symlink-install --packages-select
oe_service_pkg
  $ source ./install/setup.bash
```

노드 실행(ros2 run)

```
  $ ros2 pkg executables oe_service_pkg
  $ ros2 run oe_service_pkg oe_server
  $ ros2 run oe_service_pkg oe_client
```

ROS2 Launch Programming
Cheat Sheet (edit 2022.09.)

SOHI CODE

rpm_speed.launch.py

ROS2 Launch 파일
하나 이상의 정해진 노드를 실행할 수 있음
노드를 실행할 때 패키지의 매개변수나 노드 이름 변경,
노드 네임스페이스 설정, 환경변수 변경 등의 옵션을 설정함.

src/topic_pkg/launch/rpm_speed.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.actions import ExecuteProcess

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='topic_pkg',
            executable='rpm_pub',
            name='rpm_pub_node'
        ),
        Node(
            package='topic_pkg',
            executable='speed_calc',
            name='speed_calc_node',
            parameters=[
                {'wheel_radius': 0.5}
            ]
        ),
        Node(
            package='topic_pkg',
            executable='speed_sub',
            name='speed_sub_node'
        ),
        ExecuteProcess(
            cmd=['ros2', 'topic', 'list'],
            output='screen'
        )
    ])
```

setup.py

파이썬 패키지 설정 파일(setup.py)
ROS2 파이썬 패키지에서 배포를 위한 파일

src/topic_pkg/setup.py

```
from setuptools import setup
import glob
import os

package_name = 'my_py_pkg'

setup(
    name=package_name,
    version='0.1.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        ('share/' + package_name + '/launch',
glob.glob(os.path.join('launch', '*.launch.py'))),
        ('share/' + package_name + '/param',
glob.glob(os.path.join('param', '*.yaml'))),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='sohi',
    maintainer_email='sohicode@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'publish = topic_pkg.publish:main',
            'subscriber = topic_pkg.subscriber:main',
            'rpm_pub = topic_pkg.rpm_pub:main'
            'speed_calc = topic_pkg.speed_calc:main'
            'speed_sub = topic_pkg.speed_sub:main'
        ],
    },
)
```

build & run

빌드

```
$ cd ~/Workspaces/ros2_ws
$ colcon build --symlink-install --packages-select
topic_pkg
$ source ./install/setup.bash
```

런치 실행(ros2 launch)

```
$ ros2 launch topic_pkg rpm_speed.launch.py
```

확인

```
$ ros2 node list
$ ros2 param list
$ ros2 param get /speed_calc_node wheel_radius
```

ROS2 Interface Programming

Cheat Sheet (edit 2022.09.) reference : https://github.com/robotpilot/ros2-seminar-examples/

SOHI CODE

인터페이스 패키지 만들기 — Package

```
$ cd ~/Workspaces/ros2_ws/src
$ ros2 pkg create --build-type ament_cmake interface_pkg
$ cd interface_pkg
$ mkdir msg srv action
```

토픽 인터페이스 — ArithmeticArgument.msg

src/interface_pkg/msg/ArithmeticArgument.msg

```
# Messages
builtin_interfaces/Time stamp
float32 argument_a
float32 argument_b
```

서비스 인터페이스 — ArithmeticOperator.srv

src/interface_pkg/srv/ArithmeticOperator.srv

```
# Constants
int8 PLUS = 1
int8 MINUS = 2
int8 MULTIPLY = 3
int8 DIVISION = 4

# Request
int8 arithmetic_operator
---
# Response
float32 arithmetic_result
```

액션 인터페이스 — ArithmeticChecker.action

src/interface_pkg/action/ArithmeticChecker.action

```
# Goal
float32 goal_sum
---
# Result
string[] all_formula
float32 total_sum
---
# Feedback
string[] formula
```

빌드 — build & run

```
$ cd ~/Workspaces/ros2_ws
$ colcon build --symlink-install --packages-select interface_pkg
$ source ~/Workspaces/ros2_ws/install/local_setup.bash
```

패키지 설정 파일 — package.xml

src/interface_pkg/package.xml

```
<?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>interface_pkg</name>
 <version>0.0.0</version>
 <description>
  ROS 2 interface package
 </description>
 <maintainer email='sohicode@gmail.com'>sohi</maintainer>
 <license>Apache 2.0</license>
 <author email='sohicode@gmail.com'>sohi</author>
<buildtool_depend>ament_cmake</buildtool_depend>

<buildtool_depend>rosidl_default_generators</buildtool_depend>
 <exec_depend>builtin_interfaces</exec_depend>
 <exec_depend>rosidl_default_runtime</exec_depend>

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

빌드 설정 파일 — CMakeLists.txt

src/interface_pkg/CMakeLists.txt

```
################################################
# Declare ROS messages, services and actions
################################################
set(msg_files
 'msg/ArithmeticArgument.msg'
)

set(srv_files
 'srv/ArithmeticOperator.srv'
)

set(action_files
 'action/ArithmeticChecker.action'
)

rosidl_generate_interfaces(${PROJECT_NAME}
 ${msg_files}
 ${srv_files}
 ${action_files}
 DEPENDENCIES builtin_interfaces
)
```