# ROS2: Simple Publisher and Subscriber

운영체제의 실제

안인규 (Inkyu An)

# ROS2: Talker and Listener example

- The example used here is a simple “talker” and “listener” system
- one node publishes data and the other subscribes to the topic so it can receive that data.

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify “package.xml”, “setup.py”, “setup.cfg”
5. Build and run

# ROS2: 1. Create a package

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Navigate into the "ros2_ws/src" directory created in a previous class
- Run the package creation command:
  - ros2 pkg create --build-type ament_python --license Apache-2.0 py_pubsub

Pakcage 이름

Python용 빌드 시스템

Software license (Apache-2.0: open source lib)

# ROS2: 2. Write the pub. node

1. Create a package

.cfg"

- Generate the python file into "ros2_ws/src/py_pubsub":
  - publisher_member_function.py

Code it!

```python
import rclpy
from rclpy.node import Node

from std_msgs.msg import String


class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1


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

    minimal_publisher = MinimalPublisher()

    rclpy.spin(minimal_publisher)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_publisher.destroy_node()
    rclpy.shutdown()


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

# ROS2: 2. Write the pub. node

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Examine the code

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String
```

  - rclpy: ROS2 client Library for Python (Python 언어로 ROS2 노드를 작성할 수 있게 해주는 interface) → Node 생성, Topic pub/sub, Service req/res, parameter, timer, action, ...
  - std_msgs: ROS2에서 가장 기본적인 message type package → Bool, Byte, Char, Int8, Int16, Float32, Float64, ...
    (e.g., ros2 topic info /turtle1/cmd_vel)

```
$ ros2 topic info /turtle1/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 1
Subscription count: 2
```

# ROS2: 2. Write the pub. node

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Examine the code

```
class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1
```

Node를 상속

Node name

Topic publisher object 생성
1. Message type: String
2. Topic name: 'topic'
3. Message buffer: 10

Timer object 생성
1. timer period: 0.5초 마다 타이머 실행
2. callback function: 타이머 실행 시 호출할 함수

Publish topic
- Message는 반드시 초기화 된 type과 일치해야 함

Log system에 해당 message 출력

# ROS2: 2. Write the pub. node

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Examine the code

```
def main(args=None):
    rclpy.init(args=args)

    minimal_publisher = MinimalPublisher()

    rclpy.spin(minimal_publisher)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_publisher.destroy_node()
    rclpy.shutdown()


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

Node 초기화 (ROS2 program에서 반드시 실행)
- DDS 연결/Node 실행 준비/CLI argument 파싱 등

Publisher class 생성

- 노드를 event loop에 넣고, callback을 기다림 (e.g., Timer)
- busy-wait (OS에서 spin)이 아닌, DDS 미들웨어가 제공하는 blocking wait을 사용

Publisher, Subscriber, Timer 등의 ROS2 entities를 해제

ROS2 종료 (event loop에서 빠져나옴)

# ROS2: 2. Write the sub. node

1. Create a package
2. Write the publisher (talker) node

le
"setup.cfg"

- Generate the python file into "ros2_ws/src/py_pubsub":
  - subscriber_member_function.py

Code it!

```python
import rclpy
from rclpy.node import Node

from std_msgs.msg import String


class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            String,
            'topic',
            self.listener_callback,
            10)
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


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

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()


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

# ROS2: 2. Write the sub. node

1. Create a package
2. Write the publisher (talker) node

le
"setup.cfg"

- Generate the python file into "ros2_ws/src/py_pubsub":
  - subscriber_member_function.py

Code it!

publisher_member_function.py와 유사

```python
import rclpy
from rclpy.node import Node

from std_msgs.msg import String


class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            String,
            'topic',
            self.listener_callback,
            10)
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


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

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()


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

# ROS2: 2. Write the sub. node

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Examine the code

```
class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            String,
            'topic',
            self.listener_callback,
            10)
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)
```

Node를 상속

Node name

Topic subscription object 생성
1. Message type: String
2. Topic name: 'topic'
3. Callback function: topic 수신 시 호출 함수
4. Message buffer: 10

Log system에 해당 message 출력

# ROS2: 2. Modify “package.xml”, “setup.py”, “setup.cfg”

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify “package.xml”, “setup.py”, “setup.cfg”
5. Build and run

- Add dependencies (package.xml)
  - Add the meta information

```
<description>Examples of minimal publisher/subscriber using rclpy</description>
<maintainer email="you@email.com">Your Name</maintainer>
<license>Apache License 2.0</license>
```

  - Add the dependencies

```
<exec_depend>rclpy</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

# ROS2: 2. Modify “package.xml”, “setup.py”, “setup.cfg”

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify “package.xml”, “setup.py”, “setup.cfg”
5. Build and run

- Add an entry point (setup.py)
  - Add the meta information (match them to our “package.xml”)

```
maintainer='YourName',
maintainer_email='you@email.com',
description='Examples of minimal publisher/subscriber using rclpy',
license='Apache License 2.0',
```

  - Add an entry point: 명령어와 Python code를 연결 (executable node)

```
entry_points={
        'console_scripts': [
                'talker = py_pubsub.publisher_member_function:main',
                'listener = py_pubsub.subscriber_member_function:main',
        ],
},
```

# ROS2: 2. Modify “package.xml”, “setup.py”, “setup.cfg”

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify “package.xml”, “setup.py”, “setup.cfg”
5. Build and run

- Check setup.cfg
  - The contents of the setup.cfg file should be correctly populated automatically, like so:

```
[develop]
script_dir=$base/lib/py_pubsub
[install]
install_scripts=$base/lib/py_pubsub
```

  - This is simply telling setuptools to put our executables in lib, because ros2 run will look for them there

# ROS2: 2. Build and run

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Check for missing dependencies before building
  - Navigate into the root directory of our workspace (ros2_ws):
  - *rosdep install -i --from-path src --rosdistro humble -y*

package.xml 파일 안에 dependency를 확인하여 설치 안된 package의 설치 명령 (apt, pip 등)을 실행

- Build our new package:
  - *colcon build --packages-select py_pubsub*

py_pupsub package만 build

# ROS2: 2. Build and run

1. Create a package
2. Write the publisher (talker) node
3. Write the subscriber (listener) node
4. Modify "package.xml", "setup.py", "setup.cfg"
5. Build and run

- Run our new package
  - *source install/setup.bash*
  - *ros2 run py_pubsub talker*
  - *ros2 run py_pubsub listener*

```
$ ros2 run py_pubsub talker
[info] [minimal_publisher]: publishing: "hello world: 0"
[info] [minimal_publisher]: publishing: "hello world: 1"
[info] [minimal_publisher]: publishing: "hello world: 2"
[info] [minimal_publisher]: publishing: "hello world: 3"
[info] [minimal_publisher]: publishing: "hello world: 4"
...
```

```
$ ros2 run py_pubsub listener
[INFO] [minimal_subscriber]: I heard: "Hello World: 10"
[INFO] [minimal_subscriber]: I heard: "Hello World: 11"
[INFO] [minimal_subscriber]: I heard: "Hello World: 12"
[INFO] [minimal_subscriber]: I heard: "Hello World: 13"
[INFO] [minimal_subscriber]: I heard: "Hello World: 14"
```