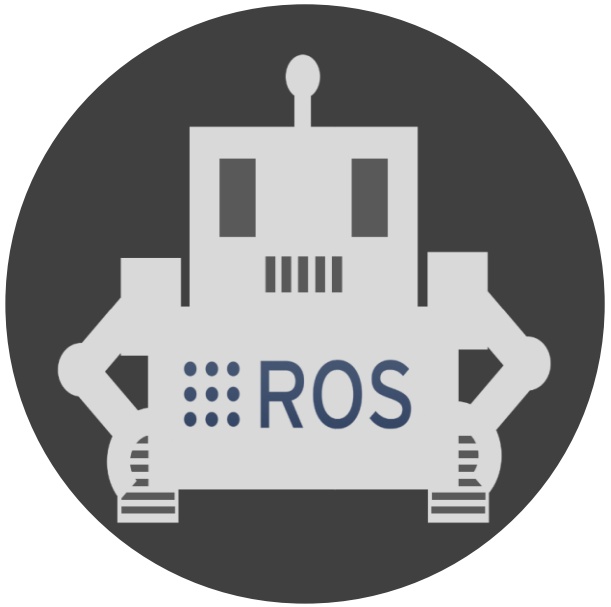
ROS 기초 강의

Chapter 4. ROS Topic

구선생 로보틱스



Update 20240620

강의 자료 다운로드

ROS 기초 강의 강의노트

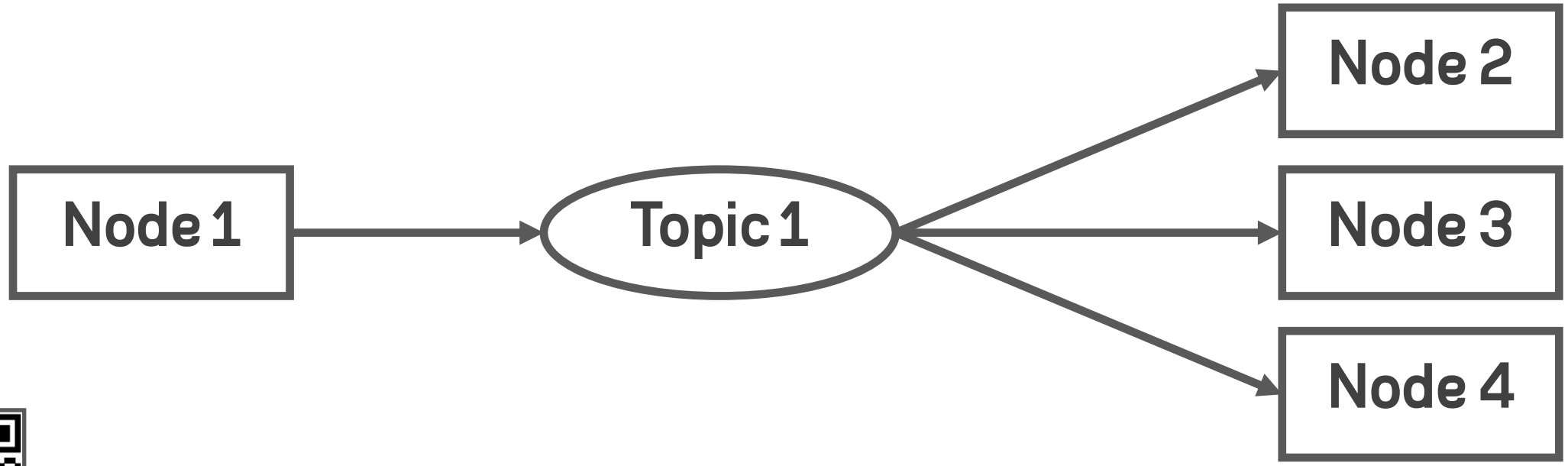
https://drive.google.com/drive/folders/1rRwS2j98HNyj5Is_yVXEGj30ILvMPtrz?usp=drive_link

1. ROS Topic 기초
2. ROS Publisher Node
3. ROS Subscriber Node
4. 커스텀 메시지

ROS Topic 기초

ROS Topic 이란?

ROS에서 Node와 Node 사이에
연속적으로 데이터를 주고 받을 수 있는 방법 중 하나



동영상 강의 - Topic 기초
https://youtu.be/4RjtS8yp7X4?si=UPYMtRI6wnCDHmXv

ROS Topic 기초

ROS Topic 명령어

- Topic 리스트 보기

```
$ rostopic list
```

- Topic 데이터 확인

```
$ rostopic echo <토픽_이름>
```

- Topic 상세 정보 확인

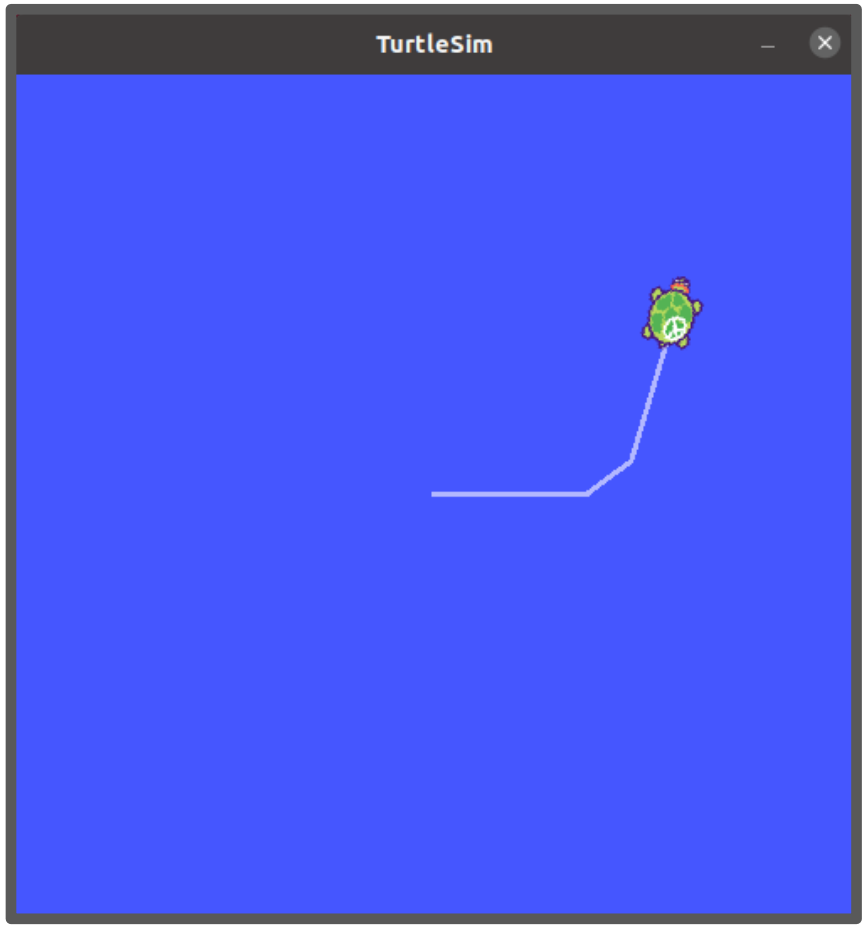
```
$ rostopic info <토픽_이름>
```

기타 명령어는 아래 위키 참고

http://wiki.ros.org/rostopic

ROS Topic 기초

ROS Topic 명령어



- rostopic list 의 결과

```
ubuntu@ubuntu:~/catkin_ws$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

- rostopic info /turtle1/cmd_vel 의 결과

```
ubuntu@ubuntu:~/catkin_ws$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers:
 * /teleop_turtle (http://ubuntu:43459/)

Subscribers:
 * /turtlesim (http://ubuntu:42873/)
```

해당 토픽의 타입과 Publisher와 Subscriber가 누구인지 확인 가능

ROS Topic 기초

ROS Topic 명령어

- Rostopic echo /turtle1/cmd_vel의 결과

```
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0
```

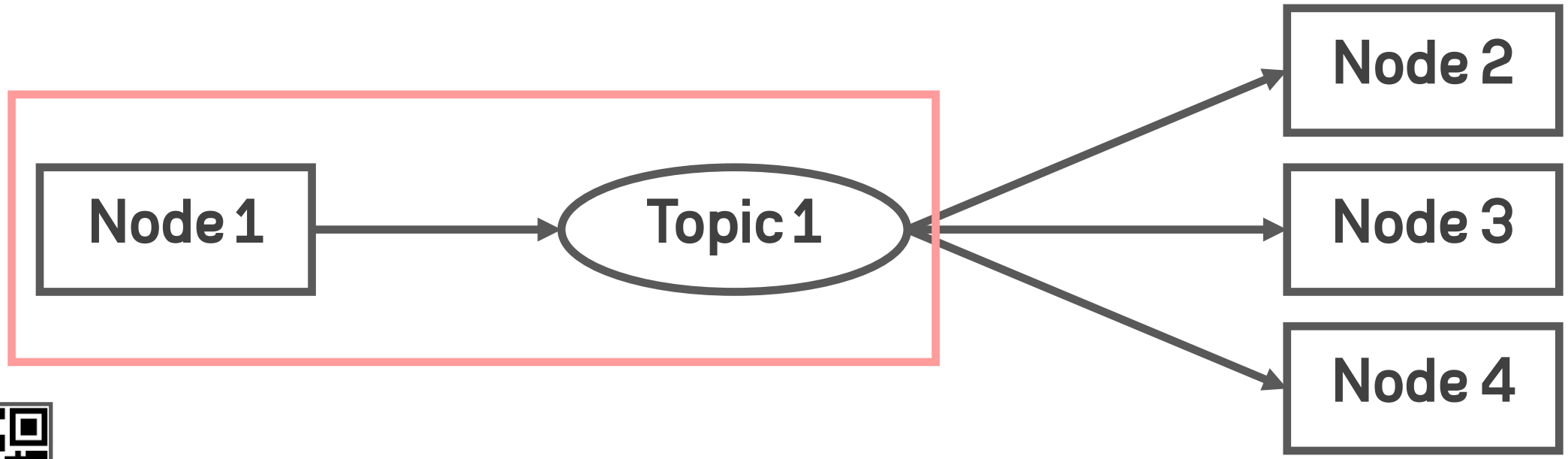
해당 토픽의 값을 확인 할 수 있다.

1. ROS Topic 기초
2. **ROS Publisher Node**
3. ROS Subscriber Node
4. 커스텀 메시지

ROS Publisher Node

Publisher Node 란?

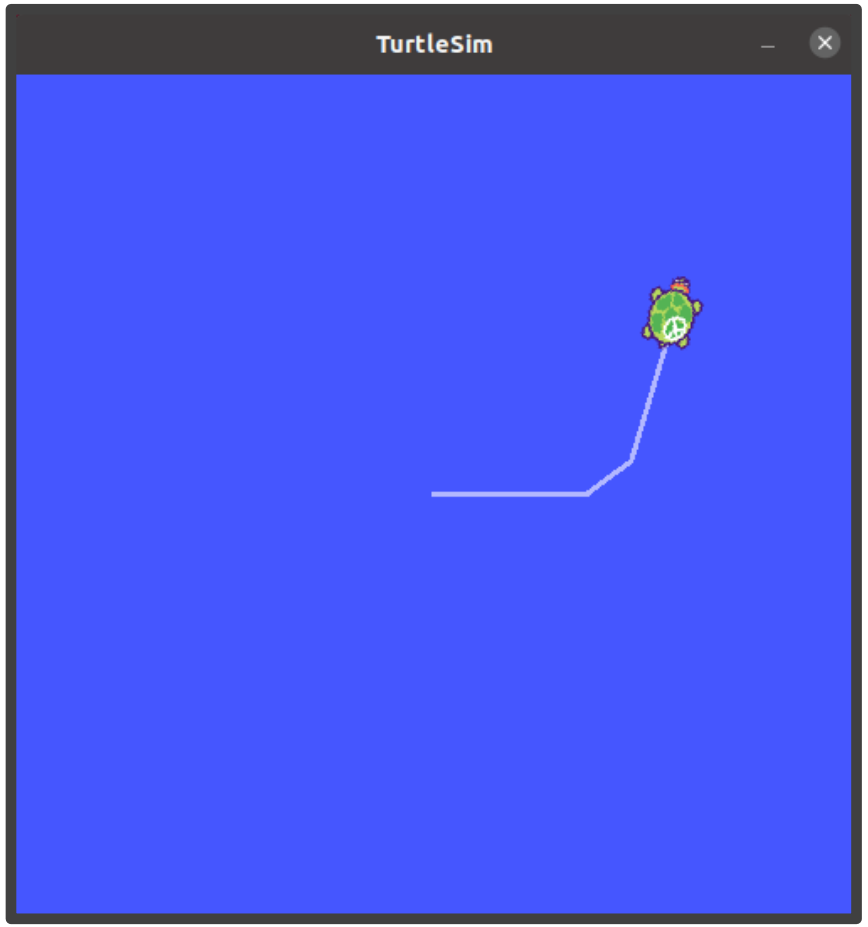
ROS 에서 Topic을 뿌려주는 Node를 의미



동영상 강의 - Publisher Node 생성
https://youtu.be/V3ntb4I4mRQ?si=kW85y0XeQqFbehjF

ROS Publisher Node

Publisher Node 란?




```
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0
```

teleop_turtle 노드는
turtle1/cmd_vel 토픽을 퍼블리시 하고 있다.

ROS Publisher Node

Publisher Node 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_publisher_tutorial roscpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg basic_publisher_tutorial roscpp
Created file basic_publisher_tutorial/package.xml
Created file basic_publisher_tutorial/CMakeLists.txt
Created folder basic_publisher_tutorial/include/basic_publisher_tutorial
Created folder basic_publisher_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/basic_publisher_tutorial. Please
adjust the values in package.xml.
```

소스코드 - basic_publish_tutorial
https://github.com/PigeonSensei/pigeon_ros_tutorial/tree/master/basic/basic_publish_tutorial

ROS Publisher Node

Publisher Node 생성

2) 소스코드 파일 생성

```
$ nano basic_publisher.cpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ cd basic_publisher_tutorial/
ubuntu@ubuntu:~/catkin_ws/src/basic_publisher_tutorial$ cd src
ubuntu@ubuntu:~/catkin_ws/src/basic_publisher_tutorial/src$ nano basic_publisher.cpp
```

∵ 소스코드는 패키지의 src 경로에서 생성되어야 한다.

ROS Publisher Node

Publisher Node 생성

3) 소스코드 작성

```cpp
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>

int main(int argc, char **argv)
{
  ros::init(argc, argv, "basic_publish_node");

  ROS_INFO("basic_publish_node Open");
  ros::NodeHandle n;

  geometry_msgs::Twist cmd_vel;
  ros::Publisher publisher_cmd_vel = n.advertise<geometry_msgs::Twist>("cmd_vel", 1000);

  ros::Rate loop_rate(60);

  while (ros::ok())
  {

    cmd_vel.linear.x = 10;
    cmd_vel.angular.z = 10;
    publisher_cmd_vel.publish(cmd_vel);

    ros::spinOnce();
    loop_rate.sleep();
  }

  ROS_INFO("basic_publish_node Close");

  return 0;
}
```

ROS Publisher Node

Publisher Node 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```

```
ubuntu@ubuntu:~/catkin_ws/src/basic_publisher_tutorial/src$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/basic_publisher_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_publisher_tutorial$ nano CMakeLists.txt
```

∵ CMakeLists.txt는 패키지 경로에 있다.

ROS Publisher Node

Publisher Node 생성

5) CMakeLists.txt의 find_package 내용 추가

```
10  find_package(catkin REQUIRED COMPONENTS
11    roscpp
12    geometry_msgs
13  )
```

ROS Publisher Node

Publisher Node 생성

6) CMakeLists.txt add_executable, target_link_librariese 추가

```
117  include_directories(
118  #  include
119    ${catkin_INCLUDE_DIRS}
120  )
121
122  add_executable(basic_publish_node src/basic_publish.cpp)
123  target_link_libraries(basic_publish_node ${catkin_LIBRARIES})
```

ROS Publisher Node

Publisher Node 생성

7) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

```
13    <build_depend>roscpp</build_depend>
14    <build_depend>geometry_msgs</build_depend>
15
16    <build_export_depend>roscpp</build_export_depend>
17    <build_export_depend>geometry_msgs</build_export_depend>
18
19    <exec_depend>roscpp</exec_depend>
20    <exec_depend>geometry_msgs</exec_depend>
```

ROS Publisher Node

Publisher Node 생성

- Pulisher Node 실행 결과

```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_publish_tutorial basic_publish_node
[ INFO] [1690954952.726861324]: basic_publish_node Open
```

- topic list 확인 결과

```
ubuntu@ubuntu:~/catkin_ws$ rostopic list
/cmd_vel
/rosout
/rosout_agg
```

소스코드에서 정의한
cmd_vel 이름의 토픽 확인 가능

cmd_vel 의 값
확인

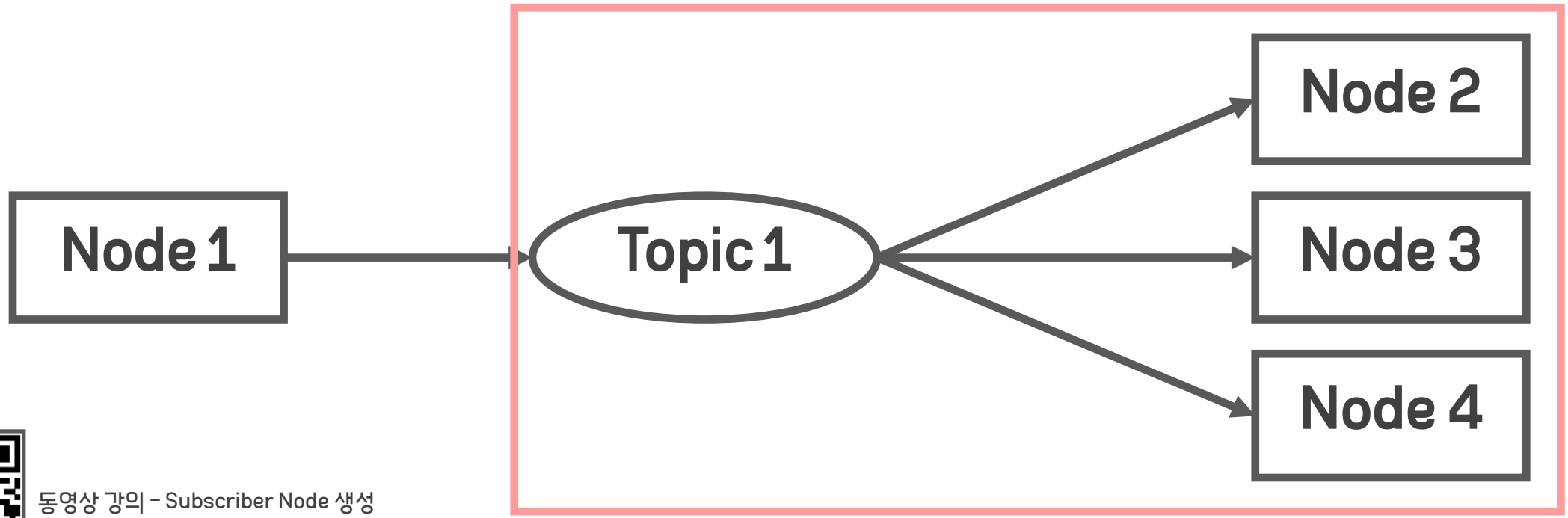
```
linear:
  x: 10.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 10.0
```

1. ROS Topic 기초
2. ROS Publisher Node
3. **ROS Subscriber Node**
4. 커스텀 메시지

ROS Subscriber Node

Subscriber Node 란?

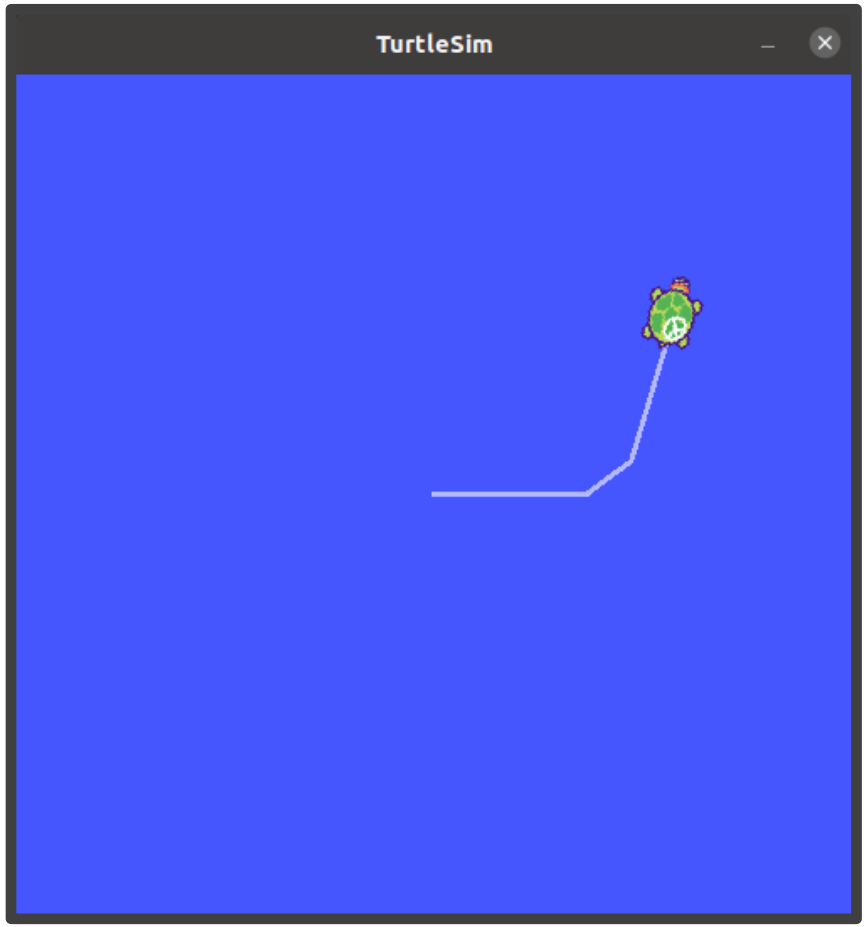
ROS 에서 Topic을 받는 Node를 의미



동영상 강의 - Subscriber Node 생성
https://youtu.be/28KRjlj-G3Y?si=4YYOp1ybd_Ocl3jA

ROS Subscriber Node

Subscriber Node 란?




```
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0
```

turtlesim 노드는
turtle1/cmd_vel 토픽의 값을 읽어서 움직인다.

ROS Subscriber Node

Subscriber Node 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_subscribe_tutorial roscpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg basic_subscribe_tutorial roscpp
Created file basic_subscribe_tutorial/package.xml
Created file basic_subscribe_tutorial/CMakeLists.txt
Created folder basic_subscribe_tutorial/include/basic_subscribe_tutorial
Created folder basic_subscribe_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/basic_subscribe_tutorial. Please
 adjust the values in package.xml.
```

소스코드 - basic_subscribe_tutorial
https://github.com/PigeonSensei/pigeon_ros_tutorial/tree/master/basic/basic_subscribe_tutorial

ROS Subscriber Node

Subscriber Node 생성

2) 소스코드 파일 생성

```
$ nano basic_subscribe.cpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ cd basic_subscribe_tutorial/
ubuntu@ubuntu:~/catkin_ws/src/basic_subscribe_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_subscribe_tutorial$ cd src
ubuntu@ubuntu:~/catkin_ws/src/basic_subscribe_tutorial/src$ nano basic_subscribe.cpp
```

∵ 소스코드는 패키지의 src 경로에서 생성되어야 한다.

ROS Subscriber Node

Subscriber Node 생성

3) 소스코드 작성

```
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>

void CmdVelCallback(const geometry_msgs::Twist::ConstPtr& cmd_vel)
{
  ROS_INFO("subscribe cmd_vel : linear.x = %.3f , angular.z = %.3f", cmd_vel->linear.x, cmd_vel->angular.z);
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "basic_subscribe_node");
  ros::NodeHandle n;

  ROS_INFO("basic_subscribe_node Open");

  ros::Subscriber subscriber_cmd_vel;
  subscriber_cmd_vel = n.subscribe("cmd_vel", 1000, CmdVelCallback);

  ros::Rate loop_rate(60);

  while (ros::ok())
  {
    ros::spinOnce();
    loop_rate.sleep();
  }

  ROS_INFO("basic_subscribe_node Close");

  return 0;
}
```

ROS Subscriber Node

Subscriber Node 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```

```
ubuntu@ubuntu:~/catkin_ws/src/basic_subscribe_tutorial/src$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/basic_subscribe_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_subscribe_tutorial$ nano CMakeLists.txt
```

∵ CMakeLists.txt는 패키지 경로에 있다.

ROS Subscriber Node

Subscriber Node 생성

5) CMakeLists.txt의 find_package 내용 추가

```
10  find_package(catkin REQUIRED COMPONENTS
11    roscpp
12    geometry_msgs
13  )
```

ROS Subscriber Node

Subscriber Node 생성

6) CMakeLists.txt add_executable, target_link_librariese 추가

```
117  include_directories(
118  #  include
119     ${catkin_INCLUDE_DIRS}
120  )
121
122  add_executable(basic_subscribe_node src/basic_subscribe.cpp)
123  target_link_libraries(basic_subscribe_node ${catkin_LIBRARIES})
```

ROS Subscriber Node

Subscriber Node 생성

7) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

```
13    <build_depend>roscpp</build_depend>
14    <build_depend>geometry_msgs</build_depend>
15
16    <build_export_depend>roscpp</build_export_depend>
17    <build_export_depend>geometry_msgs</build_export_depend>
18
19    <exec_depend>roscpp</exec_depend>
20    <exec_depend>geometry_msgs</exec_depend>
```

ROS Subscriber Node

Subscriber Node 생성

- Subscriber Node 실행 결과

```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_subscribe_tutorial basic_subscribe_node
[ INFO] [1690955596.492275937]: basic_subscribe_node Open
```

Subscriber Node 단독으로 실행 시 아무런 반응이 없다.
cmd_vel 토픽을 Publish하는 Node가 없기 때문이다.

ROS Subscriber Node

Subscriber Node 생성

- cmd_vel Topic을 받은 Subscriber Node 실행 결과

```
[ INFO] [1690955688.072158963]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.088616210]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.105487222]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.121851814]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.138425359]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.155226967]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.172012042]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.188398233]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.205346422]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
[ INFO] [1690955688.221934570]: subscribe cmd_vel : linear.x = 10.000 , angular.z = 10.000
```

cmd_vel 토픽을 Publish하는 Node가 실행 된 경우
cmd_vel Topic이 수신되어 CallBack 함수를 호출한다

1. ROS Topic 기초
2. ROS Publisher Node
3. ROS Subscriber Node
4. 커스텀 메시지

커스텀 메시지

커스텀 메시지란?

- ROS에서 제공하는 기본 메시지 타입

- geometry_msgs/Twist
- sensor_msgs/Imu
- nav_msgs/Odometry
- tf/transform_broadcaster
- 등등..

- 커스텀 메시지 타입

- motor_driver_msgs/MotorCommand
- encoder_msgs/EncoderCount
- imu_msgs/Angle
- 등등..

ROS 에서는 모듈화를 위해 기본적으로 제공하는 메시지 타입의 사용을 권장하고 있지만
커스텀 메시지 타입의 제작이 필요한 경우도 있음

동영상 강의 - 커스텀 메시지 생성
https://youtu.be/FuZp8WsGR8g?si=LtcShIw8Ze7EneUR

커스텀 메시지

커스텀 메시지에서 사용가능한 타입

Primitive Type	Serialization	C++
bool (1)	unsigned 8-bit int	uint8_t (2)
int8	signed 8-bit int	int8_t
uint8	unsigned 8-bit int	uint8_t
int16	signed 16-bit int	int16_t
uint16	unsigned 16-bit int	uint16_t
int32	signed 32-bit int	int32_t
uint32	unsigned 32-bit int	uint32_t
int64	signed 64-bit int	int64_t
uint64	unsigned 64-bit int	uint64_t
float32	32-bit IEEE float	float
float64	64-bit IEEE float	double
string	ascii string (4)	std::string
time	secs/nsecs unsigned 32-bit ints	ros::Time
duration	secs/nsecs signed 32-bit ints	ros::Duration

- 사용 예시

```
1  string first_name
2  string last_name
3  uint8 age
4  uint32 score
5
```

더 자세한 내용은 아래 사이트 참고

http://wiki.ros.org/msg

커스텀 메시지

커스텀 메시지 패키지 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_msg_tutorial roscpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg msg_tutorial roscpp
Created file msg_tutorial/package.xml
Created file msg_tutorial/CMakeLists.txt
Created folder msg_tutorial/include/msg_tutorial
Created folder msg_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/msg_tutorial. Please adjust the
values in package.xml.
```

∵ 메시지 패키지는 모듈화를 위해 독립된 패키지로 생성하는 것을 권장

소스코드 - tutorial_msgs
https://github.com/PigeonSensei/pigeon_ros_tutorial/tree/master/others/tutorial_msgs

커스텀 메시지

커스텀 메시지 패키지 생성

2) 메시지 파일 생성

```
$ nano TutorialMsg.msg
```

```
ubuntu@ubuntu:~/catkin_ws/src$ cd msg_tutorial/
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial$ mkdir msg
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial$ ls
CMakeLists.txt  include  msg  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial$ cd msg/
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial/msg$ nano TutorialMsg.msg
```

∴ 메시지 파일은 패키지의 msg 경로에서 생성되어야 한다.
msg 경로가 없다면 생성한다.

커스텀 메시지

커스텀 메시지 패키지 생성

3) 메시지 파일 작성

```
bool Bool
int8 Int8
uint8 UInt8
int16 Int16
uint16 UInt16
int32 Int32
uint32 UInt32
int64 Int64
uint64 UInt64
float32 Float32
float64 Float64
string String
time Time
duration Duration
```

커스텀 메시지

커스텀 메시지 패키지 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```

```
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial/msg$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial$ ls
CMakeLists.txt  include  msg  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/msg_tutorial$ nano CMakeLists.txt
```

∴ CMakeLists.txt는 패키지 경로에 있다.

커스텀 메시지

커스텀 메시지 패키지 생성

5) CMakeLists.txt의 find_package 내용 추가

```
10  find_package(catkin REQUIRED COMPONENTS
11    roscpp
12    message_generation
13    std_msgs
14  )
```

커스텀 메시지

커스텀 메시지 패키지 생성

6) CMakeLists.txt add_message_files, generate_messages 추가

```
50  add_message_files(
51    FILES
52    TutorialMsg.msg
53  )
```

⋮

```
70  generate_messages(
71    DEPENDENCIES
72    std_msgs  # Or other packages containing msgs
73  )
```

커스텀 메시지

커스텀 메시지 패키지 생성

7) CMakeLists.txt catkin_package의 CATKIN_DEPENDS 수정

```
104  catkin_package(
105  #  INCLUDE_DIRS include
106  # LIBRARIES
107   CATKIN_DEPENDS roscpp message_generation std_msgs message_runtime
108  #  DEPENDS system_lib
109  )
```

커스텀 메시지

커스텀 메시지 패키지 생성

8) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

```
13    <build_depend>roscpp</build_depend>
14    <build_depend>message_generation</build_depend>
15    <build_depend>std_msgs</build_depend>
16
17    <build_export_depend>roscpp</build_export_depend>
18    <build_export_depend>message_generation</build_export_depend>
19    <build_export_depend>std_msgs</build_export_depend>
20
21    <exec_depend>roscpp</exec_depend>
22    <exec_depend>message_generation</exec_depend>
23    <exec_depend>std_msgs</exec_depend>
24    <exec_depend>message_runtime</exec_depend>
25
```

∵ message_runtime 추가 작성

커스텀 메시지

커스텀 메시지 패키지 생성

9) 컴파일 후 라이브러리 생성 확인

```
[100%] Linking CXX executable /home/ubuntu/catkin_ws/devel/lib/msg_tutorial/msg_tutorial_node
[100%] Built target msg_tutorial_node
Base path: /home/ubuntu/catkin_ws
Source space: /home/ubuntu/catkin_ws/src
Build space: /home/ubuntu/catkin_ws/build
Devel space: /home/ubuntu/catkin_ws/devel
Install space: /home/ubuntu/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/ubuntu/catkin_ws/build"
####
####
#### Running command: "make -j2 -l2" in "/home/ubuntu/catkin_ws/build"
####
22:44:35: The process "/opt/ros/noetic/bin/catkin_make" exited normally.
22:44:35: Elapsed time: 00:06.
```

```
ubuntu@ubuntu:~/catkin_ws/devel/include/tutorial_msgs$ ls
TutorialMsg.h
```

컴파일이 완료되고 해당 위치에 메시지 파일 이름의 헤더파일이 생성 되어야 사용가능 하다

커스텀 메시지

커스텀 메시지 Publisher Node 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_msg_tutorial roscpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg basic_msg_tutorial roscpp
Created file basic_msg_tutorial/package.xml
Created file basic_msg_tutorial/CMakeLists.txt
Created folder basic_msg_tutorial/include/basic_msg_tutorial
Created folder basic_msg_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/basic_msg_tutorial. Please adjus
t the values in package.xml.
```

커스텀 메시지

커스텀 메시지 Publisher Node 생성

2) 소스코드 파일 생성

```
$ nano basic_msg_publisher.cpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ cd basic_msg_tutorial/
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ cd src
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial/src$ nano basic_msg_publish.cpp
```

∵ 소스코드는 패키지의 src 경로에서 생성되어야 한다.

커스텀 메시지

커스텀 메시지 Publisher Node 생성

3) 소스코드 작성

```cpp
#include <ros/ros.h>
#include "tutorial_msgs/TutorialMsg.h"

int main(int argc, char **argv)
{
  ros::init(argc, argv, "basic_msg_publish_node");

  ROS_INFO("basic_msg_publish_node Open");
  ros::NodeHandle n;

  tutorial_msgs::TutorialMsg tutorial_msg;
  ros::Publisher publisher_tutorial_msg = n.advertise<tutorial_msgs::TutorialMsg>("tutorial_msg", 1000);

  ros::Rate loop_rate(60);

  while (ros::ok())
  {
    tutorial_msg.Bool = true;
    tutorial_msg.Int8 = 127;
    tutorial_msg.UInt8 = 255;
    tutorial_msg.Int16 = 32767;
    tutorial_msg.UInt16 = 65535;
    tutorial_msg.Int32 = 2147483647;
    tutorial_msg.UInt32 = 4294967295;
    tutorial_msg.Int64 = 9223372036854775807LL;
    tutorial_msg.UInt64 = 18446744073709551615ULL;
    tutorial_msg.Float32 = 123.456f;
    tutorial_msg.Float64 = 123.456;
    tutorial_msg.String = "Tutorial";
    tutorial_msg.Time = ros::Time::now();
    tutorial_msg.Duration = ros::Duration(5.0);

    publisher_tutorial_msg.publish(tutorial_msg);

    ros::spinOnce();
    loop_rate.sleep();
  }

  ROS_INFO("basic_msg_publish_node Close");

  return 0;
}
```

커스텀 메시지

커스텀 메시지 Publisher Node 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```

```
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial/src$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ nano CMakeLists.txt
```

∵ CMakeLists.txt는 패키지 경로에 있다.

커스텀 메시지

커스텀 메시지 Publisher Node 생성

5) CMakeLists.txt의 find_package 내용 추가

```
10  find_package(catkin REQUIRED COMPONENTS
11    roscpp
12    tutorial_msgs
13  )
```

∵ tutorial_msgs는 커스텀 메시지 패키지 이름이다.

커스텀 메시지

커스텀 메시지 Publisher Node 생성

6) CMakeLists.txt add_executable, target_link_librariese, add_dependencies 추가

```
121  add_executable(basic_msg_publish_node src/basic_msg_publish.cpp)
122  target_link_libraries(basic_msg_publish_node ${catkin_LIBRARIES})
123  add_dependencies(basic_msg_publish_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
```

커스텀 메시지

커스텀 메시지 Publisher Node 생성

7) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

```
13    <build_depend>roscpp</build_depend>
14    <build_depend>tutorial_msgs</build_depend>
15
16    <build_export_depend>roscpp</build_export_depend>
17    <build_export_depend>tutorial_msgs</build_export_depend>
18
19    <exec_depend>roscpp</exec_depend>
20    <exec_depend>tutorial_msgs</exec_depend>
21
```

커스텀 메시지

커스텀 메시지 Publisher Node 생성

- 커스텀 메시지 Publisher Node 실행 결과

```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_msg_tutorial basic_msg_publish_node
[ INFO] [1690957475.801757879]: basic_msg_publish_node Open
```

- Topic list

```
ubuntu@ubuntu:~/catkin_ws$ rostopic list
/rosout
/rosout_agg
/tutorial_msg
```

- Topic 확인

```
ubuntu@ubuntu:~/catkin_ws$ rostopic echo /tutorial_msg
ERROR: Cannot load message class for [tutorial_msgs/TutorialMsg]. Are your mess
ages built?
```

ㄴ 터미널 환경을 워크스페이스로 맞춰야한다.

```
ubuntu@ubuntu:~/catkin_ws$ source devel/setup.bash
```

```
Bool: True
Int8: 127
UInt8: 255
Int16: 32767
UInt16: 65535
Int32: 2147483647
UInt32: 4294967295
Int64: 9223372036854775807
UInt64: 18446744073709551615
Float32: 123.45600128173828
Float64: 123.456
String: "Tutorial"
Time:
  secs: 1690957581
  nsecs: 953200196
Duration:
  secs: 5
  nsecs:         0
```

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_msg_tutorial roscpp
```

```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg basic_msg_tutorial roscpp
Created file basic_msg_tutorial/package.xml
Created file basic_msg_tutorial/CMakeLists.txt
Created folder basic_msg_tutorial/include/basic_msg_tutorial
Created folder basic_msg_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/basic_msg_tutorial. Please adjus
t the values in package.xml.
```

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

2) 소스코드 파일 생성

```
$ nano basic_msg_subscribe.cpp
```

```
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ cd src
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial/src$ nano basic_msg_subscribe.cpp
```

∵ 소스코드는 패키지의 src 경로에서 생성되어야 한다.

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

3) 소스코드 작성

```cpp
#include <ros/ros.h>
#include "tutorial_msgs/TutorialMsg.h"

void TutorialMsgCallback(const tutorial_msgs::TutorialMsg::ConstPtr& tutorial_msg)
{
  ROS_INFO("tutorial_msg.Bool = %s", tutorial_msg->Bool ? "true" : "false");
  ROS_INFO("tutorial_msg.Int8 = %d", tutorial_msg->Int8);
  ROS_INFO("tutorial_msg.UInt8 = %u", tutorial_msg->UInt8);
  ROS_INFO("tutorial_msg.Int16 = %d", tutorial_msg->Int16);
  ROS_INFO("tutorial_msg.UInt16 = %u", tutorial_msg->UInt16);
  ROS_INFO("tutorial_msg.Int32 = %d", tutorial_msg->Int32);
  ROS_INFO("tutorial_msg.UInt32 = %lu", (unsigned long)tutorial_msg->UInt32);
  ROS_INFO("tutorial_msg.Int64 = %lld", (long long int)tutorial_msg->Int64);
  ROS_INFO("tutorial_msg.UIn64 = %llu", (unsigned long long int)tutorial_msg->UInt64);
  ROS_INFO("tutorial_msg.Float32 = %.2f", tutorial_msg->Float32);
  ROS_INFO("tutorial_msg.Float64 = %.2lf", tutorial_msg->Float64);
  ROS_INFO("tutorial_msg.String = %s", tutorial_msg->String.c_str());
  ROS_INFO("tutorial_msg.Time = %f", tutorial_msg->Time.toSec());
  ROS_INFO("tutorial_msg.Duration = %f", tutorial_msg->Duration.toSec());
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "basic_msg_subscribe_node");
  ros::NodeHandle n;

  ROS_INFO("basic_msg_subscribe_node Open");

  ros::Subscriber subscriber_tutorial_msg;
  subscriber_tutorial_msg = n.subscribe("tutorial_msg", 1000, TutorialMsgCallback);

  ros::Rate loop_rate(60);

  while (ros::ok())
  {
    ros::spinOnce();
    loop_rate.sleep();
  }

  ROS_INFO("basic_msg_subscribe_node Close");

  return 0;
}
```

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```

```
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial/src$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_msg_tutorial$ nano CMakeLists.txt
```

∴ CMakeLists.txt는 패키지 경로에 있다.

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

5) CMakeLists.txt의 find_package 내용 추가

```
10  find_package(catkin REQUIRED COMPONENTS
11    roscpp
12    tutorial_msgs
13  )
```

∵ tutorial_msgs는 커스텀 메시지 패키지 이름이다.

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

6) CMakeLists.txt add_executable, target_link_librariese, add_dependencies 추가

```
125  add_executable(basic_msg_subscribe_node src/basic_msg_subscribe.cpp)
126  target_link_libraries(basic_msg_subscribe_node ${catkin_LIBRARIES})
127  add_dependencies(basic_msg_subscribe_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
```

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

7) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

```
13    <build_depend>roscpp</build_depend>
14    <build_depend>tutorial_msgs</build_depend>
15
16    <build_export_depend>roscpp</build_export_depend>
17    <build_export_depend>tutorial_msgs</build_export_depend>
18
19    <exec_depend>roscpp</exec_depend>
20    <exec_depend>tutorial_msgs</exec_depend>
21
```

커스텀 메시지

커스텀 메시지 Subscriber Node 생성

- 커스텀 메시지 Subscriber Node 실행 결과

```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_msg_tutorial basic_msg_subscribe_node
[ INFO] [1690958610.548687860]: basic_msg_subscribe_node Open
[ INFO] [1690958610.855255442]: tutorial_msg.Bool = true
[ INFO] [1690958610.855515836]: tutorial_msg.Int8 = 127
[ INFO] [1690958610.855689561]: tutorial_msg.UInt8 = 255
[ INFO] [1690958610.855873897]: tutorial_msg.Int16 = 32767
[ INFO] [1690958610.856049750]: tutorial_msg.UInt16 = 65535
[ INFO] [1690958610.856220623]: tutorial_msg.Int32 = 2147483647
[ INFO] [1690958610.856299022]: tutorial_msg.UInt32 = 4294967295
[ INFO] [1690958610.856441250]: tutorial_msg.Int64 = 9223372036854775807
[ INFO] [1690958610.856879641]: tutorial_msg.UIn64 = 18446744073709551615
[ INFO] [1690958610.857407982]: tutorial_msg.Float32 = 123.46
[ INFO] [1690958610.858247765]: tutorial_msg.Float64 = 123.46
[ INFO] [1690958610.858765023]: tutorial_msg.String = Tutorial
[ INFO] [1690958610.859112190]: tutorial_msg.Time = 1690958610.852707
[ INFO] [1690958610.859244414]: tutorial_msg.Duration = 5.000000
```

커스텀 메시지 Publisher Node가 실행되어야
Callback 함수를 호출하여 위와 같은 결과를 보여준다

감사합니다

구선생 로보틱스