ROS 개요

선행사항

- 우분투 설치(18.04 LTS 권장)
- Linux 터미널 환경에 대한 기본적 이해 및 명령어 숙지(ls, mkdir, cd, chmod, pipeline 및 standard io등)
- https://www.cs.cmu.edu/~15131/f17/ 참고

ROS 설치 1단계: 패키지 설치

- 우분투는 debian 계열 linux 배포판임. Debian 계열은 apt라는 패키지 관리자를 제공,
-
- 밑에 코드를 터미널에 한줄한줄 복붙

  - sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
  - sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
  - sudo apt update
  - sudo apt install ros-melodic-desktop-full

ROS 설치 2단계: rosdep 초기화

- rosdep은 ros package들의 dependency를 관리하는 툴이고 이를 초기화하는 과정
- 자세히 몰라도됨, dependency는 어떤 패키지가 필요로 하는 다른 패키지
- 예를 들어 ROS에서 c++로 작성된 모든 패키지는 roscpp라는 패키지를 dependency로 갖음
- 밑에 코드를 터미널에 한줄한줄 복붙

  - sudo rosdep init
  - rosdep update

ROS 설치 3단계: 환경변수 설정

- 환경변수가 무엇인지를 설명하는것은 너무 오래 걸리므로 반드시 terminal 및 shell 환경에서의 environment variable이 무엇인지 공부하고 검색하기 바람 (linux 터미널 기초임)
- ROS에서는 environment variable을 통해 실행파일의 위치와 명령어 등을 선언
- /opt/ros/melodic/setup.bash 파일을 통해 이를 한번에 선언(source, export,alias 명령어 및 shell script, bash,zsh,가 뭔지 모른다면 검색/공부, 이것도 linux터미널 기초)
- source /opt/ros/melodic/setup.bash를 통해 ROS 환경변수 일괄적으로 선언 가능. 단 터미널 실행시마다 매번 실행해야함(환경변수는 터미널을 재시작하면 초기화되기 때문)
- 따라서 .bashrc에 source 명령어를 추가해 자동으로 선언되게 하는게 합리적 (아래의 명령어가 이해되지 않는다면 반드시 구글에 검색해서 완벽하게 이해하기 바람. 모르고 넘어가면 나중에 언젠간 문제됨 반드시.)
  - echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
  - source ~/.bashrc

ROS 설치 4단계: rosinstall 설정

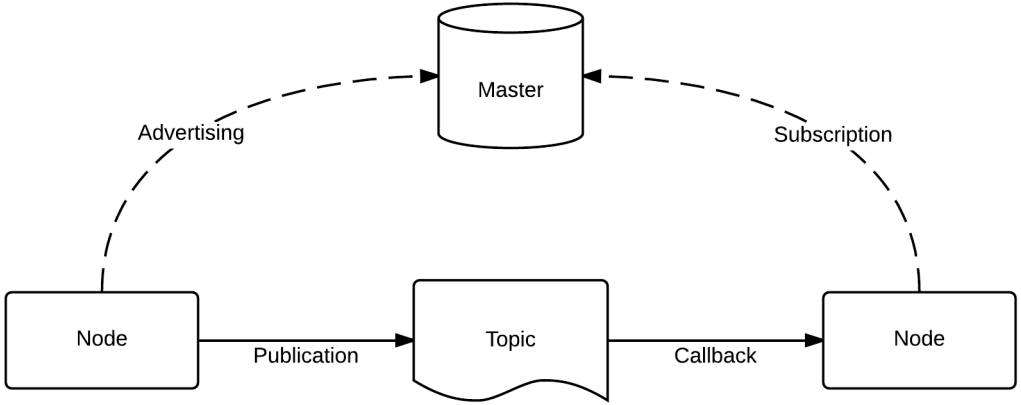
- sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential

  이거 그냥 입력 뭔진 몰라도됨

ROS 설치 5단계: catkin workspace 설정

- catkin은 cmake 기반 ROS를 위한 build tool임. 뭔말인지 모르겠으면 그냥 컴파일러라 생각 (실제론 아님)
- 아래 코드 입력
- mkdir -p ~/catkin_ws/src
- cd ~/catkin_ws/
- catkin_make
- catkin_ws는 앞으로 사용할 catkin을 위한 작업공간임. 이 중 src는 소스코드가 들어갈 곳임.
- catkin_make를 하면 cmake list 파일 외에 devel, build 폴더등이 자동으로 생성되는데 유저가 건드릴 일은 없는 공간임.
- devel 폴더의 setup.bash를 설치3단계처럼 .bashrc에 source해줘야 함. (이게 무슨 말인지 모르겠으면 linux 기초부터 다시) 그래야 이 폴더에서 작업한 내용을 ROS가 인식할 수 있음.

ROS의 기초 개념



- ROS는 node들로 구성되어있음.
- node는 각각의 프로그램임.
- 이를 조율해주는게 roscore node(master)
- 각 node는 Topic이라는 메시지를 서로 주고받으며 통신함
- Topic을 발행하면 publish, 수신하면 subscribe라 부름
- Topic은 수신자와 발신자를 정해놓지 않음. publisher node는 누가 이 메시지를 받는지 고려하지 않고 지멋대로 마구 뿌림. 마치 라디오 방송국과 같이 발신자는 무차별적으로 신호를 뿌리고 수신자가 알아서 메시지를 받음.
- 그걸 알아서 subscribe 하고 정보를 process하는게 subscriber node의 역할.
- 예를 들어 로봇 노드와 컨트롤러 노드가 있다 치자.
- 컨트롤러 노드는 속도 topic을 publish, 로봇 노드는 그 속도 정보가 포함된 토픽을 subscribe해서 그 속도대로 움직임.

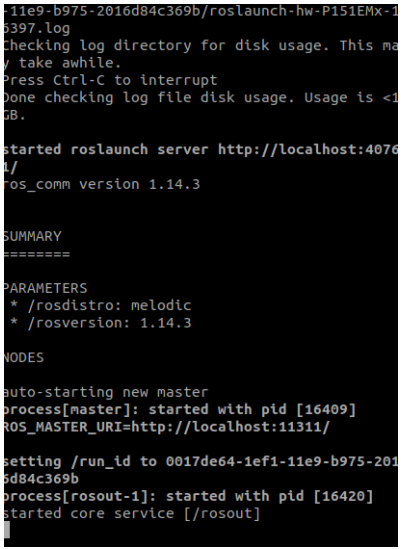
ROS 기초개념2

- 각 node의 소스코드가 어떻게 작성되고 build되는지는 그 양이 매우 많으므로 별도로 공부해야함.
- https://www.cse.sc.edu/~jokane/agitr/
- http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

- 위 두 링크를 참조하면서 node의 작성법을 익혀야 함. cpp와 oop에 익숙하지 않다면 매우 오래걸리기 때문에 시간을 들여 배워야함. 직접 소스코드를 작성해보고 다른 소스코드를 뜯어보고 자신의 코드를 재사용 가능하게 object화 하기 위해서는 랭귀지에 대한 전반적인 이해가 필요.
- 오래 걸리기 때문에 일단은 node의 작성은 넘어가도록 함.

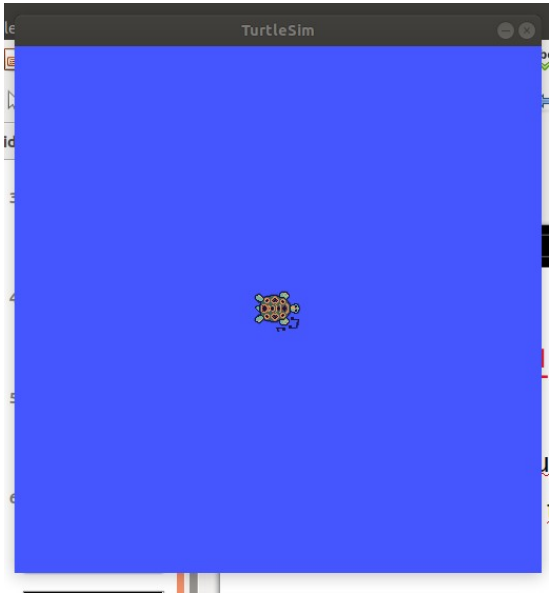
ROS 사용해보기 - turtlesim

- terminal을 네개 띄우고
- 첫번째 터미널에는 roscore 입력, 아래와 같은 창이 떠야한다.



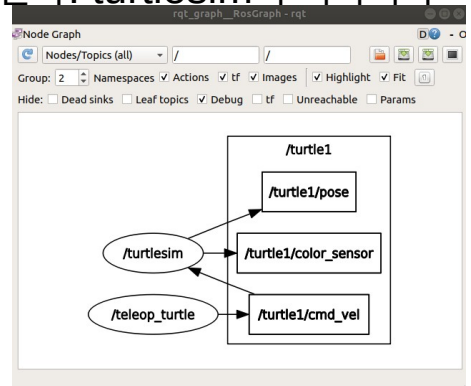
ROS 사용해보기 - turtlesim

- 두번째 창에는 rosrun turtlesim turtlesim_node 입력
- turtlesim 패키지의 turtlesim_node 라는 node를 실행하라는 의미임
- 이런 창이 뜬다.



ROS 사용해보기 - turtlesim

- rosrun turtlesim turtle_teleop_key 입력
- Use arrow keys to move the turtle.라는 메시지가 뜨면 키보드 화살표를 눌러본다. turtlesim의 거북이가 움직이는 것을 확인할 수 있다.
- 이제 새 창에서 rqt_graph를 입력해보자.
- 이 도표가 현재 노드의 상태이다. 이렇게 안뜨면 설정을 그림처럼 바꿔보자
- teleop_turtle 노드는 turtle1/cmd_vel이라는 이름의 토픽을 publish하고
- 이 topic을 turtlesim 노드가 subscribe하고 있으며 다시 turtlesim 노드는
- turtle1/pose와 color_sensor라는 토픽을 publish하나 아무도 subscribe하고있지 않음을 볼 수 있다.

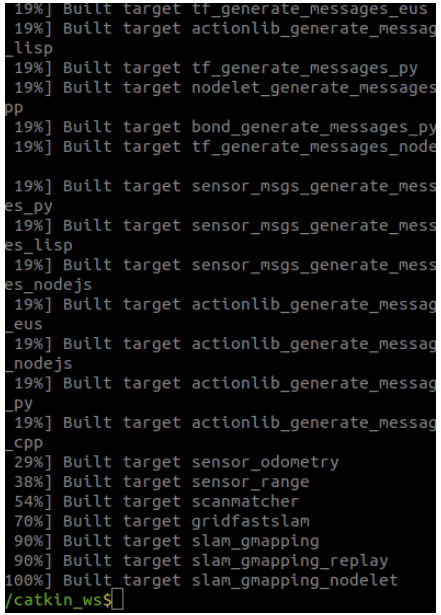


ROS 패키지 설치

- ROS의 공개 패키지를 설치하는 방법은 2가지이다. 첫째는 github등에서 직접 소스코드를 받아서 직접 make하는 것이고
- 두번째는 debian의 apt기능을 사용하는 것이다. 후자를 통해 ROS의 navigation 메타패키지를 설치해본다.
- 메타패키지는 여러 패키지를 모아둔 패키지이다. 가령 navigation 패키지에는 amcl, base_local_planner 등 십여가지의 패키지가 포함되어있다.
- sudo apt install ros-melodic-navigation

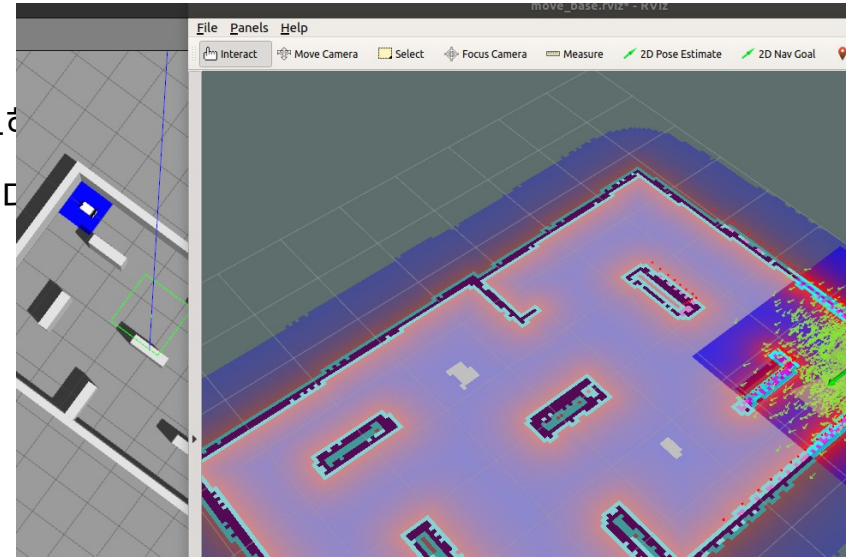
ROS 패키지 설치

- 이번에는 직접 source 파일로부터 패키지를 설치하는 법을 배워보겠다.
- https://github.com/woong164/RobomasterAIChallenge/tree/master/Ros 이 폴더의 모든 파일을
- 아까 만들어둔 ~/catkin_ws/src에 복사한 후
- catkin_ws로 이동하여 터미널에 catkin_make를 입력한다.
- 좌측과 같이 나오면 컴파일이 잘 된것이다.



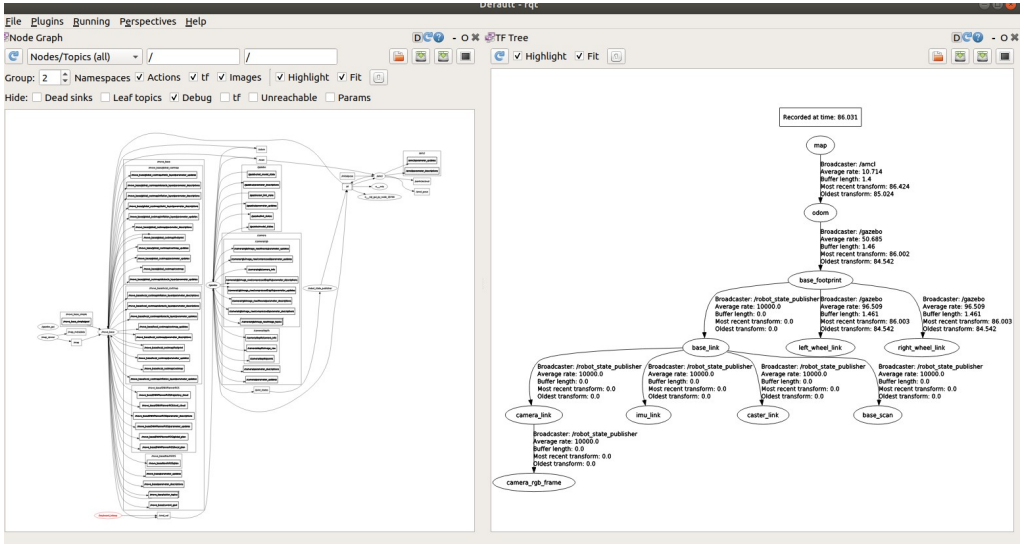
ROS launch 사용

- roslaunch roboin_simulation robomaster_navigation.launch 를 입력
- roslaunch는 launch 파일을 통해 ros의 노드들을 실행한다는 의미이다. roslaunch는 여러개의 노드와 설정을 한 파일로 묶어둔 것으로 이 파일을 통해 한번에 여러 노드들을 실행할 수 있다.
- 사진과 같이 두개의 gui창이 뜨는데 이 중 RVIZ란 창을 보자.
- 창 상단의 2d nav goal 이라는 버튼을 누르고 지도상에 드래그
- 로봇이 이동할 목표지점을 입력해주면 움직이는 모습을 볼 수 있
- 만약 창이 정상적으로 뜨지 않는다면 터미널에 ctrl c를
- 입력해서 모든 프로그램을 종료하고 다시 roslaunch한다.

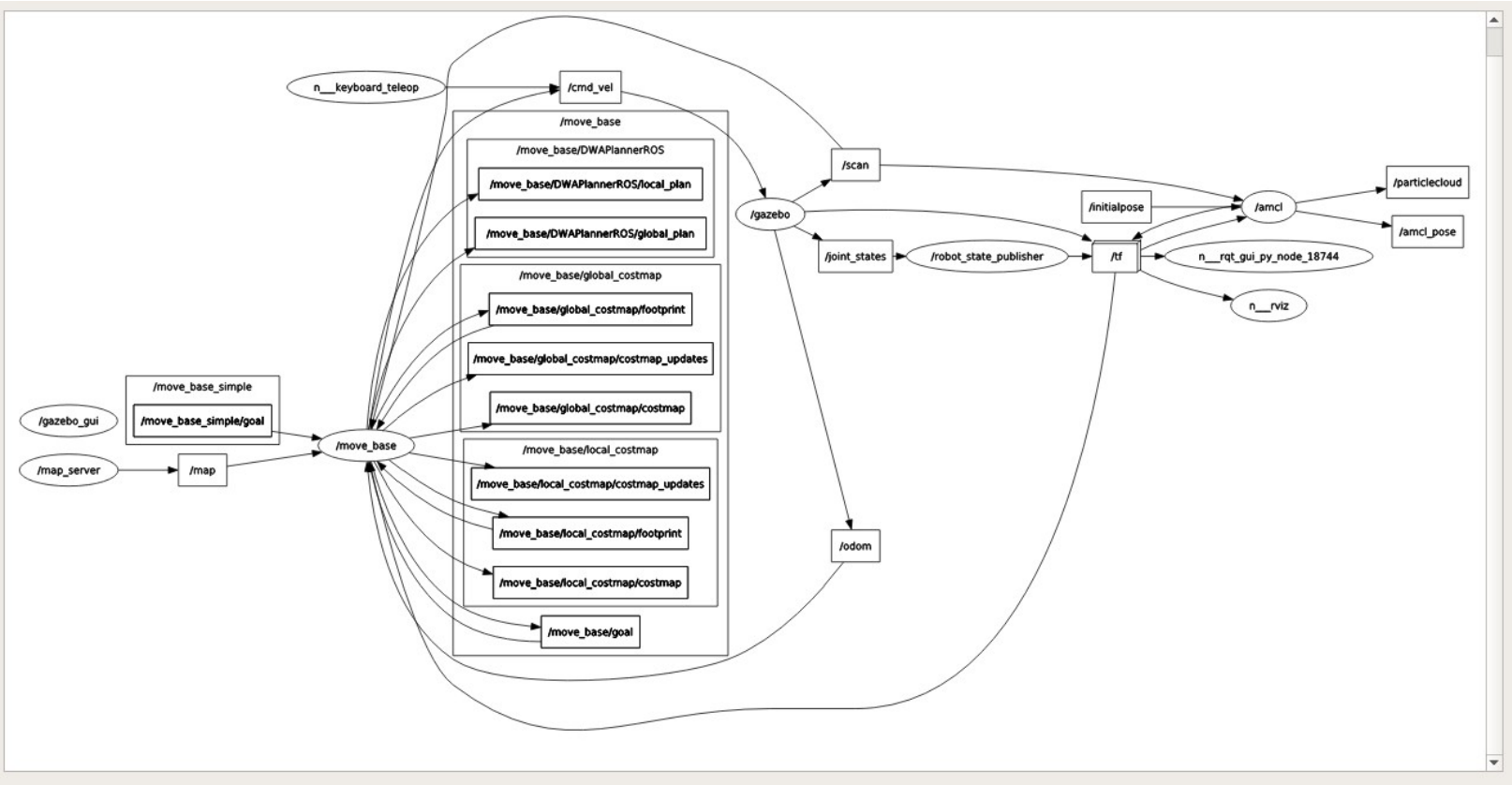


ROS node 분석하기

- 방금 전에 뜬 2개의 gui중 3d 시뮬레이션은 gazebo라는 3차원 시뮬레이션 프로그램이다. 다른 하나는 RVIZ인데 ros에서 발행되는 topic을 시각화해주는 툴이다.
- 이제 다른 콘솔에 rqt를 입력해보자. 그리고 상단의 plugins - introspection - node graph와 plugins-visualization-tf tree를 켜서 사진처럼 만들어보자.
-



ROS rqt

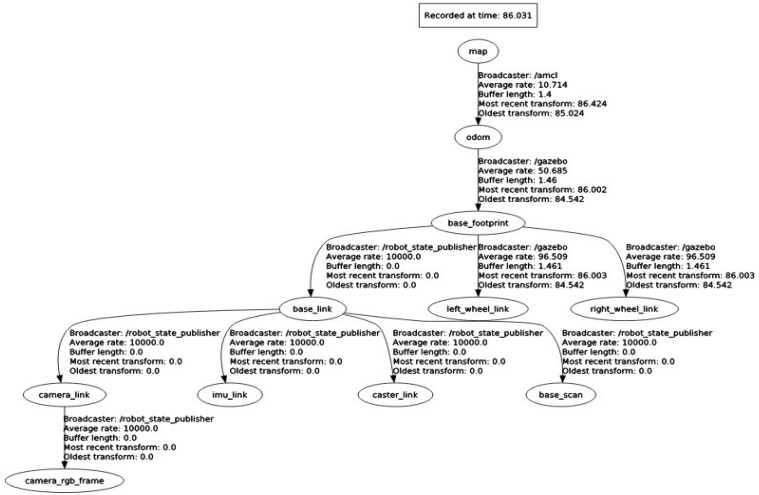

n__keyboard_teleop
/cmd_vel
/move_base
/move_base/DWAPlannerROS
/move_base/DWAPlannerROS/local_plan
/move_base/DWAPlannerROS/global_plan
/move_base/global_costmap
/move_base/global_costmap/footprint
/move_base/global_costmap/costmap_updates
/move_base/global_costmap/costmap
/move_base/local_costmap
/move_base/local_costmap/costmap_updates
/move_base/local_costmap/footprint
/move_base/local_costmap/costmap
/move_base/goal
/move_base_simple
/move_base_simple/goal
/gazebo_gui
/map_server
/map
/move_base
/gazebo
/scan
/joint_states
/robot_state_publisher
/odom
/initialpose
/tf
/amcl
/particlecloud
/amcl_pose
n__rqt_gui_py_node_18744
n__rviz

ROS rqt

- map server 노드가 map 토픽을 publish한 것을 move_base가 subscribe한 것을 볼 수 있다.
- gazebo node는 시뮬레이션의 로봇의 위치정보를 odom 토픽을 통해 publish하며
- joint_states 토픽을 robot_state_publisher 노드에 줘 간접적으로 tf(이후 설명함) topic을 publish하고있다.
- 또한 시뮬레이트 된 로봇의 lidar scan값을 scan 토픽으로 publish하고 이를 amcl 노드가 subscribe한다.
- amcl노드는 이 스캔값과 map정보를 통해 자신의 위치를 결정하고 이를 tf topic으로 발행한다.(그림에는 map topic을 받지 않는데 실제로는 topic이 아니라 service라는 방법으로 정보를 받기 때문에 그래프에 안보인다.)
- 이렇게 결정된 자신의 위치 등의 정보를 포함한 tf 토픽을 move_base노드는 받아 운동 계획을 세우고 cmd_vel 토픽을 publish한다.
- cmd_vel 토픽은 로봇의 속도 명령에 관한 정보가 담겨있는데 이를 다시 gazebo 노드가 받아 시뮬레이션의 로봇을 움직인다.

ROS TF

- TF는 좌측 그림과 같이 로봇의 각 부품의 좌표계를 hierarchy를 통해 나열한 것이다.
- 각 좌표간의 상대좌표를 통해 절대 좌표를 자동으로 변환해준다.
- 이 정보가 tf 토픽으로 매번 발행되는 것이다.



Further study

- https://www.cse.sc.edu/~jokane/agitr/ 를 통해 앞에서 생략한 로스의 프로그래밍을 공부한다.
- roslaunch, service, actionlib, TF, gazebo, RVIZ, RQT, catkin 등 필요한 툴을 익힌다.
- ROS 패키지의 사용법을 모르겠으면 ros wiki나 관련 documentation에 검색해보거나 직접 헤더파일을 열어서 해당 클래스를 이해한다.
- 에러가 발생하면 구글에 해당 에러를 긁어서 검색해본다. 검색할 때는 가능하면 한글을 멀리하고 영어를 가까이해야함
- tensorflow, open ai, open cv등의 외부 라이브러리도 로스와 통합해본다.