== REPORT ==

<ROS 1일차>

1. ROS 보고서 작성

	robit 18 기 인턴
학번	2023741059
이름	홍지현

1. 로봇 소프트웨어 플랫폼

1. 기존 로봇 개발 방식과 한계

- 하드웨어 설계, 제어부터 제어기, 네비게이션 등 모든 것을 개발해야 함
- API 마다의 인터페이스가 다르기에 어떤 데이터를 주고 받고 코딩을 구성하는 등 처음부터 적용하는데 하나하나 학습이 필요함
- 하드웨어에 의존적인 소프트웨어 로봇이 바뀌게 되면 소프트를 다시 해야함
- 소프트웨어를 작성하는데 하드웨어에 대한 지식이 필요함
- 디버깅을 하기 위해 디버깅 툴을 작성하거나 디버깅 API를 코드에 삽입해야함
- OS에 의존적이어서 OS변경이 어렵거나 불가능함
- 멀티 PC, 로봇을 구성하는 경우 통신 구축, 검증에 많은 기간을 소비함

2. 로봇 소프트웨어 플랫폼의 필요성

하드웨어 모듈 + 운영체제(플랫폼) + 애플리케이션 + 유저 => 하나의 생태계 생성

플랫폼이 가져온 변화 -> 휴대폰이나 컴퓨터 등

- 하드웨어 인터페이스 통합
- 하드웨어 추상화, 규격화, 모듈화
- 가격↓, 성능↑
- 하드웨어, 미들웨어(OS), 애플리케이션 분리
- 사용자 수요에 맞는 서비스에 집중
- 유저 증가-> 구매와 피드백->사용자 증가 => 새로운 생태계의 선환 구조 형성

로봇도 플랫폼 도입!!

로봇 플랫폼의 장점

- 높은 프로그램 재 사용성
- 통신 기반 프로그램
- 개발 도구 지원

- 활성화된 커뮤니티
- 생태계 조성

3. ROS 목적

전세계 사람들이 로봇 개발의 편리성을 높이게 하기 위함, + PC나 휴대폰의 역사를 로봇에 반영시키기위한 생태계형성

2. 로봇 운영체제 ROS

1. About ROS

- Node간 메시지 교환 방식으로 프로그램을 잘게 나누어 공동 개발 가능
- 도구 지원 : 명령어 도구, 시각화 도구, GUI 도구, 시뮬레이션 도구
- 로보틱스에서 사용하는 데이터의 규격화
- 모델링, 센싱, 인식, 내비게이션, 메니퓰레이션 기능 지원
- 크케 형성된 WiFi(생태계)

2. ROS 특징

메타 운영체제란

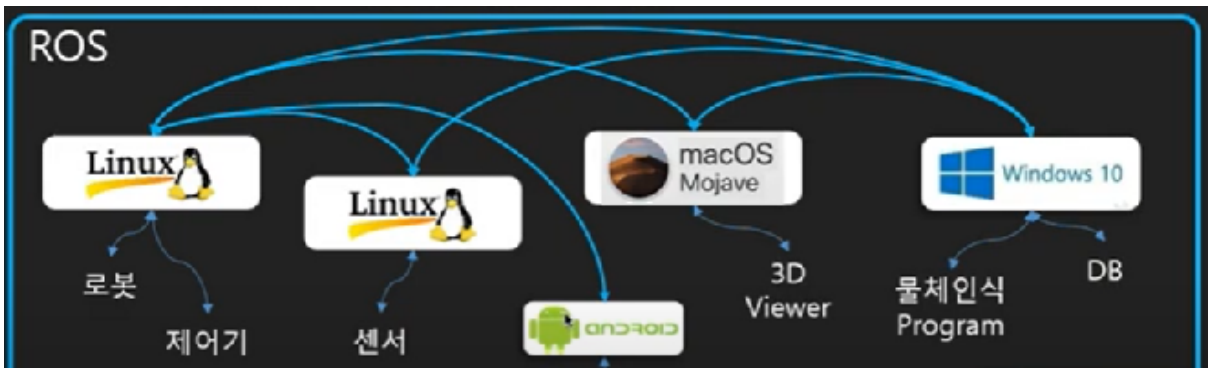
- 분산 컴퓨팅에서 프로그램과 컴퓨팅 자원을 가상화 레이어로 연결하여 분산된 컴퓨팅 자원을 스케쥴링, 관리, 여러 처리 등을 실행하는 시스템
- 전통적 OS(윈도우, 리눅스 개념)이 아님
- 기존 OS를 활용해 개별 컴퓨팅 자원을 관리, 동작시키며, 메타운영체제는 각 컴퓨팅 자원을 관리, 동작시키는 상위 시스템 개념
- OS가 설치된 여러 컴퓨팅를 하나의 시스템을 동작하도록 하는 프로그램과 라이브러링의 집합

메타운영체제 ROS

- ROS = 메타운영체제

- 윈도우 리눅스 등 에 프로그램과 라이브러리 형태로 설치해 사용
- 여러 PC, 로봇을 효과적으로 통합 가능
- 윈도우 + 리눅스 등 다른 OS가 설치된 PC도 하나의 시스템으로 쉽게 통합 가능
- 로봇 개발에 필수 기능을 담은 멀티 플랫폼 라이브러리와 멀티 플랫폼 빌드 툴 제공
- 다양한 형태의 로봇을 통합하는데 효과적

이기종 디비이스간 통신을 통한 통합 가능



로봇 여러 기능 통합이 가능하다

분산된 로봇 자원간 통합 -> 네트워크만 연결된다면 통신 기반이기 때문에 다른장소에서도 가능

통신 라이브러리 , 툴 제공

- 메시지 파싱 기능 -> 노드간 메시지를 전달하기 위한 인터페이스, 노드의 캡슐화와 재사용성 촉진
- 메시지 기록 및 재생 -> 메시지를 기록하고 재생하는 기능 제공, 반복적인 실험, 알고리즘 개발에 매우 유용
- 다양한 프로그래밍 언어에서 사용 가능한 메시지 라이브러리 -> C++, 자바, 파이썬 등 다양한 프로그래밍 언어 제공
- 분산 매개 변수 시스템 -> 시스템에서 사용하는 변수를 모든 기기가 실시간으로 공유, 수정
- 로봇 표준 메시지 정의 -> 카메라 이미지, IMU, 오도메트리, 경로 등 표준 데이터 정의, 모듈화로 로봇 교체가 매우 쉬워지고 협업을 유도하고 효율성을 향상시킴

3. ROS 버전

자신이 사용할 패키지가 해당 버전을 지원하는지 반드시 확인해야 함

3. ROS 개발환경 구축

1. ROS 설치

리눅스 20.04-> noetic

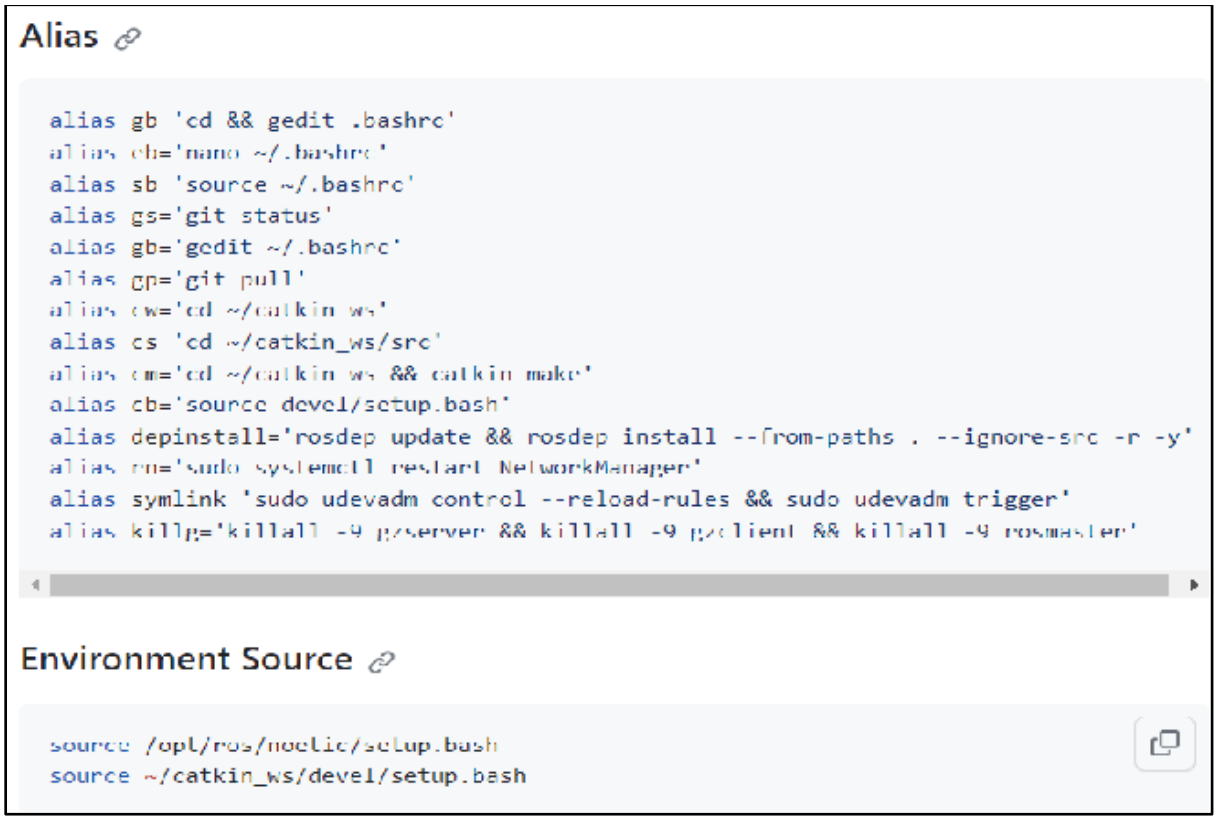
```
git clone https://github.com/mjlee111/ros1_ubuntu_installer.git

cd ros1_ubuntu_installer

sudo chmod 755 ros_noetic_install.sh

bash ros_noetic_install.sh
```

2. ROS 개발환경



sudo gedit ~/.bashrc

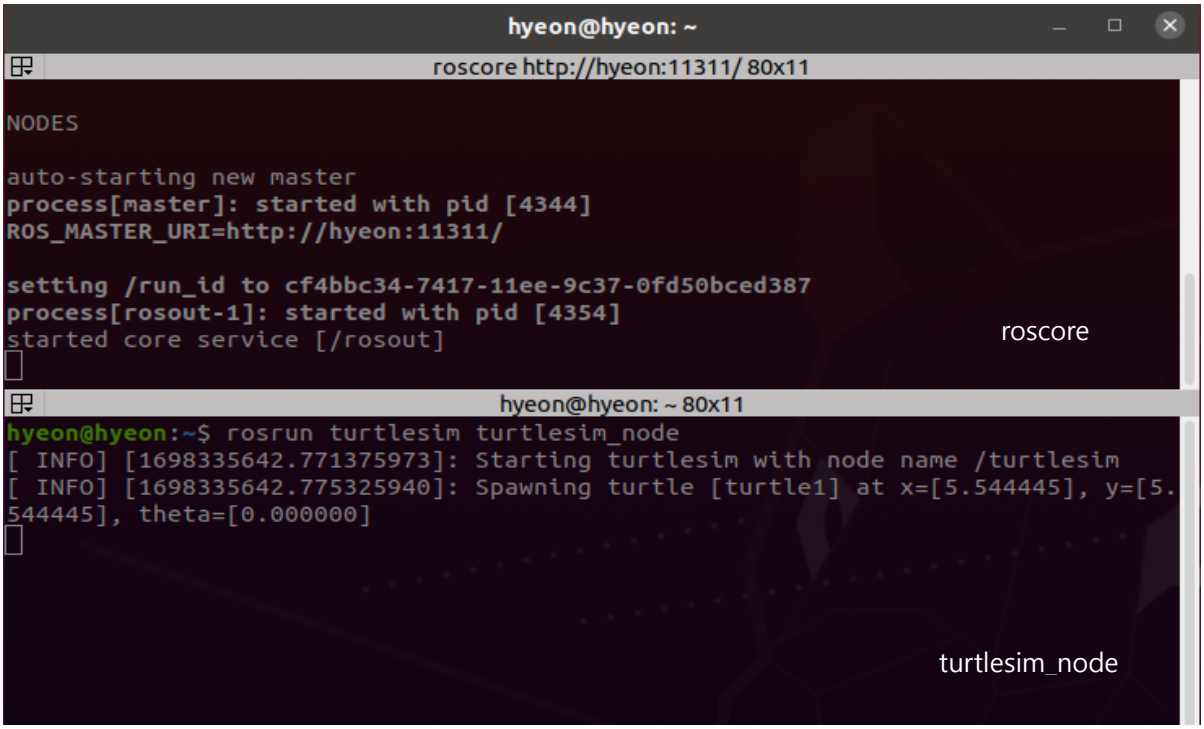
.bashrc 환경 변수 파일에 들어가 다음 텍스트를 입력, 저장 후 닫기

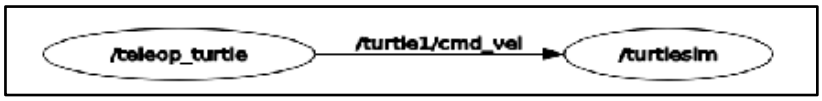
.bashrc : 리눅스를 부팅하여 구동될 때, .bashrc에 적힌 내용이 적용되도록 함.

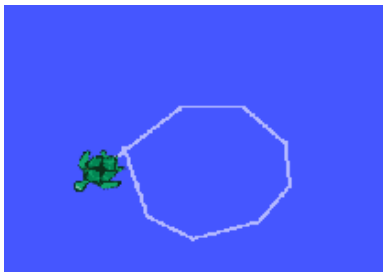
2. ROS 동작 테스트

roscore

rosrun turtlesim turtlesim_node







rosrun turtlesim turtle_teleop_key

rostopic list

rosrun rqt_graph rqt_graph

4. ROS 중요 컨셉

1. ROS 용어 정리

1. 마스터

마스터는 노드와 노드 사이의 연결과 메시지 통신을 위한 네임서버와 같은 역할을 한다.
roscore 명령어로 실행 가능하며, 마스터 없이는 노드 간의 접속, 토픽과 서비스 같은 메시지 통신을 할 수 없다.
ROS_MASTER_URI 변수에 기재된 URI 주소와 포트를 가진다.
기본 값으로 URI 주소는 현재 로컬 IP를 가지며 포트는 11311을 사용한다.

2. 노드

노드는 ROS에서 실행되는 최소 단위의 프로세서를 지칭한다.
ROS에서는 하나의 목적에 하나의 노드를 작성하길 권하며, 재사용이 쉽게 구성하여 개발하기를 권한다.
예를 들어 모바일 로봇의 경우, 로봇을 구동하기 위하여 각 프로그램을 세분화시킨다.
즉, 센서 드라이브, 센서 데이터를 이용한 변환, 장애물 판단, 모터 구동, 엔코더 입력, 내비게이션 등 세분화된 작은 노드들을 이용한다.
노드는 구동과 함께 마스터에 노드 이름과 publisher, subscriber, service server, service client에서 사용하는 토픽 및 서비스 이름, 메시지 형태, URI 주소와 포트를 등록한다. 이 정보들을 기반으로 각 노드는 노드끼리 토픽과 서비스를 이용하여 메시지를 주고받을 수 있다.

3. 패키지

패키지는 ROS를 구성하는 기본 단위이다. ROS의 응용프로그램은 패키지 단위로 개발되며 패키지는 최소한 하나 이상의 노드를 포함하거나 다른 패키지의 노드를 실행하기 위한 설정 파일들을 포함하게 된다. 더불어 각종 프로세스를 구동하기 위한 ROS 의존성 라이브러리, 데이터셋, 설정 파일 등 패키지에 필요한 모든 파일을 포함하고 있다.

4. 메타패키지

메타 패키지는 공통된 목적을 지닌 패키지들의 집합을 말한다. 예를 들어 Navigation 메타 패키지는 AMCL, DWA, EKF, map_server 등 10여 개의 패키지로 구성되어 있다.

5. 메시지

노드는 메시지(message)를 통해 노드 간의 데이터를 주고받는다. 메시지는 integer, floating, point, boolean과 같은 변수 형태이다. 또한 메시지 안에 메시지를 품고 있는 간단한 데이터 구조나 메시지들이 나열된 배열과 같은 구조도 사용할 수 있다.
단방향 메시지 송수신 방식의 토픽과 양방향 메시지 요청/응답 방식의 서비스를 이용한다.

6. 토픽

토픽(topic)은 "이야깃거리"이다. 퍼블리셔(publisher) 노드가 하나의 이야깃거리에 대해서 토픽으로 마스터에 등록한 후, 이야깃거리에 대한 이야기를 메시지 형태로 퍼블리시한다.
이 이야깃거리를 수신받기를 원하는 서브스크라이버 노드는 마스터에 등록된 토픽의 이름에 해당하는 퍼블리셔 노드의 정보를 받는다. 이 정보를 기반으로 서브스크라이버 노드는 퍼블리셔 노드와 직접 연결하여 메시지를 토픽으로 송수신하게 된다.

7. 퍼블리시 및 퍼블리셔

퍼블리시(publish)는 토픽의 내용에 해당하는 메시지 형태의 데이터를 송신하는 것을 말한다. 퍼블리셔(publisher) 노드는 퍼블리시를 수행하기 위하여 토픽을 포함한 자신의 정보들을 마스터에 등록하고, 서브스크라이브를 원하는 서브스크라이버 노드에 메시지를 보낸다. 퍼블리셔는 이를 실행하는 개체로써 노드에서 선언한다. 퍼블리셔는 하나의 노드에서 복수로 선언할 수 있다.

8. 서브스크라이버 및 서브스크라이버

서브스크라이브(subscribe)는 토픽의 내용에 해당하는 메시지 형태의 데이터를 수신하는 것을 말한다. 서브스크라이버(subscriber) 노드는 서브스크라이브를 수행하기 위하여 토픽을 포함한 자신의 정보들을 마스터에 등록하고, 구독하고자 하는 토픽을 퍼블리시하는 퍼블리셔 노드의 정보를 마스터로부터 받는다. 이 정보를 기반으로 서브스크라이버 노드는 퍼블리셔 노드와 직접 접속하여 메시지를 받는다. 서브스크라이버는 이를 실행하는 개체로서 노드에서 선언한다. 서브스크라이버는 하나의 노드에서 복수로 선언할 수 있다.

퍼블리시와 서브스크라이브 개념의 토픽 통신 방식은 비동기 방식이라 필요에 따라서 주어진 데이터를 전송하고 받기에 매우 훌륭한 방법이다. 또한 한 번의 접속으로지속적인 메시지를 송수신하기 때문에 지속해서 메시지를 발송해야 하는 센서 데이터에 적합하여 많이 사용된다. 하지만 때에 따라서는 요청과 응답이 함께 사용되는 동기 방식의 메시지 교환 방식도 필요하다. 이에 따라, ROS에서는 서비스(service)라는 이름으로 메시지 동기 방식을 제공한다. 서비스는 요청이 있을 때 응답하는 서비스 서버와 요청하고 응답받는 서비스 클라이언트로 나누니다. 서비스는 토픽과는 달리 일회성 메시지 통신이다. 서비스의 요청과 응답이 완료되면 연결된 두 노드의 접속은 끊긴다.

9. 서비스

서비스(service) 메시지 통신은 특정 목적의 작업에 해당되는 서비스를 요청하는 서비스 클라이언트와 서비스 응답을 담당하는 서비스 서버 간의 동기적 양방향 서비스 메시지 통신을 말한다.

10. 서비스 서버

서비스 서버(service server)는 요청을 입력으로 받고, 응답을 출력으로 하는 서비스 메시지 통신의 서버 역할을 말한다. 요청과 응답은 모두 메시지로 되어 있으며, 서비스 요청을 받으면 지정된 서비스를 수행한 다음에 그 결과를 서비스 클라이언트에 전달한다. 서비스 서버는 정해진 명령을 받아 수행하는 노드에 사용된다.

11. 파라미터

ROS에서의 파라미터(parameter)는 노드에서 사용되는 파라미터를 말한다. 흔히 윈도우 프로그램에서 *ini 설정 파일과 같다고 생각하면 된다. 파라미터는 디폴트(default)로 설정값들이 지정되어 있고, 필요에 따라 외부에서 읽거나 쓸 수 있다. 특히 외부에서 쓰기 기능을 이용하여 상황에 따라 설정값을 실시간으로 바꿀수 있기 때문에 매우 유용하다. 예를 들어 외부 장치와 연결되는 PC의 USB 포트나 카메라 캘리브레이션 값, 모터 속도나 명령어들의 최댓값과 최솟값 등의 설정을 지정할 수 있다.

12. 파라미터 서버

파라미터 서버(parameter server)는 패키지에서 파라미터를 사용할 때, 각 파라미터를 등록하는 서버를 말한다. 파라미터 서버는 마스터의 한 기능이기도 하다.

Topic

단방향, 연속성을 가진 통신방법중 하나

n : n 통신이 가능하다.

메세지를 보내는 노드 : Publisher node

메세지를 받는 노드 : Subscriber node

Service

센서 데이터는 대부분 Topic통신을 사용

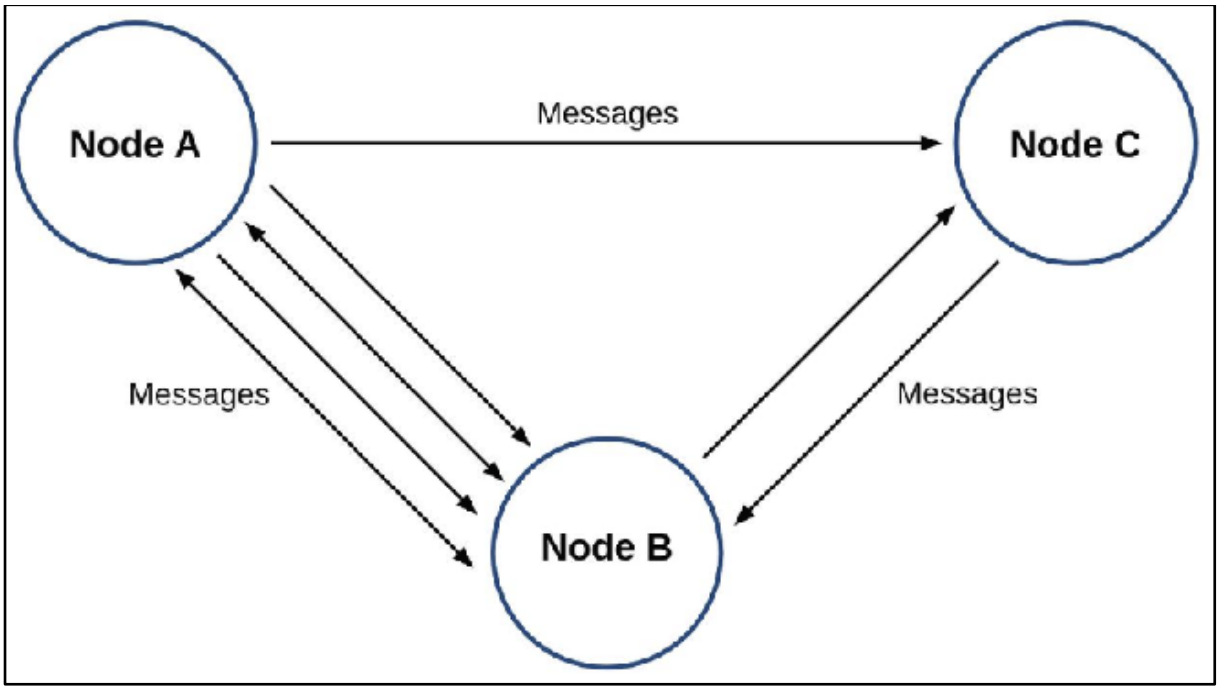
그렇지만 가끔 **양방향 , 일회성 통신**이 필요할 때도 있음

그럴때 사용하는게 service

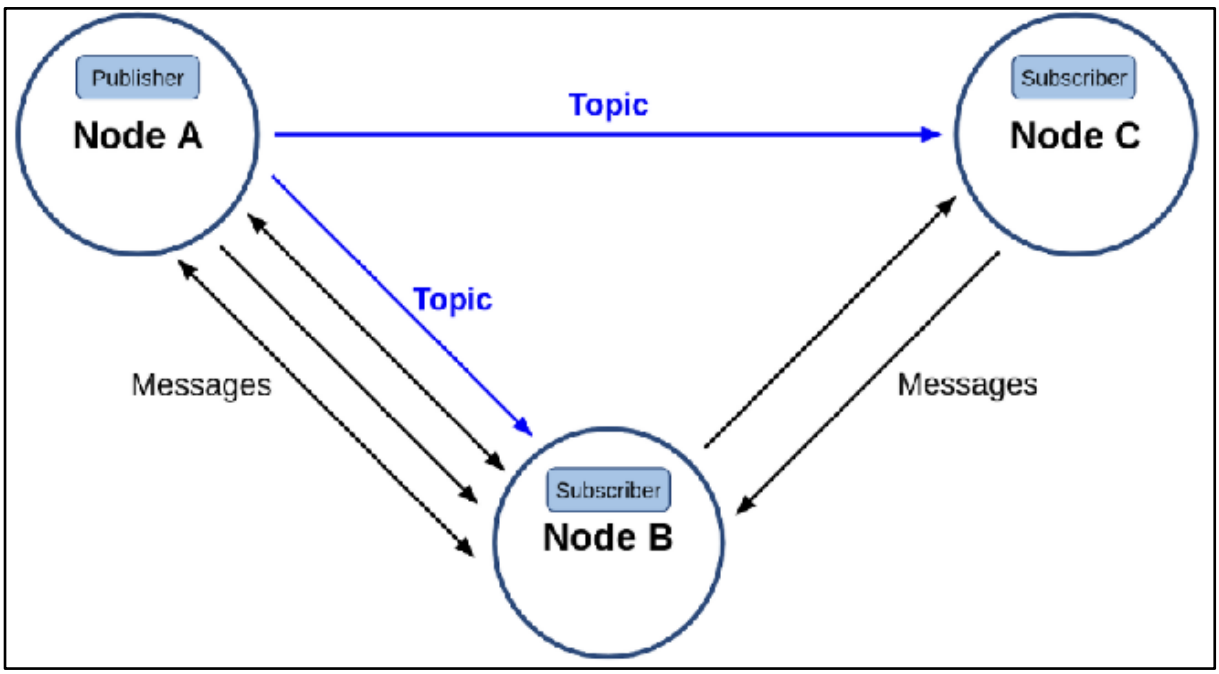
서버 <-> 클라이언트

2. 메시지 통신

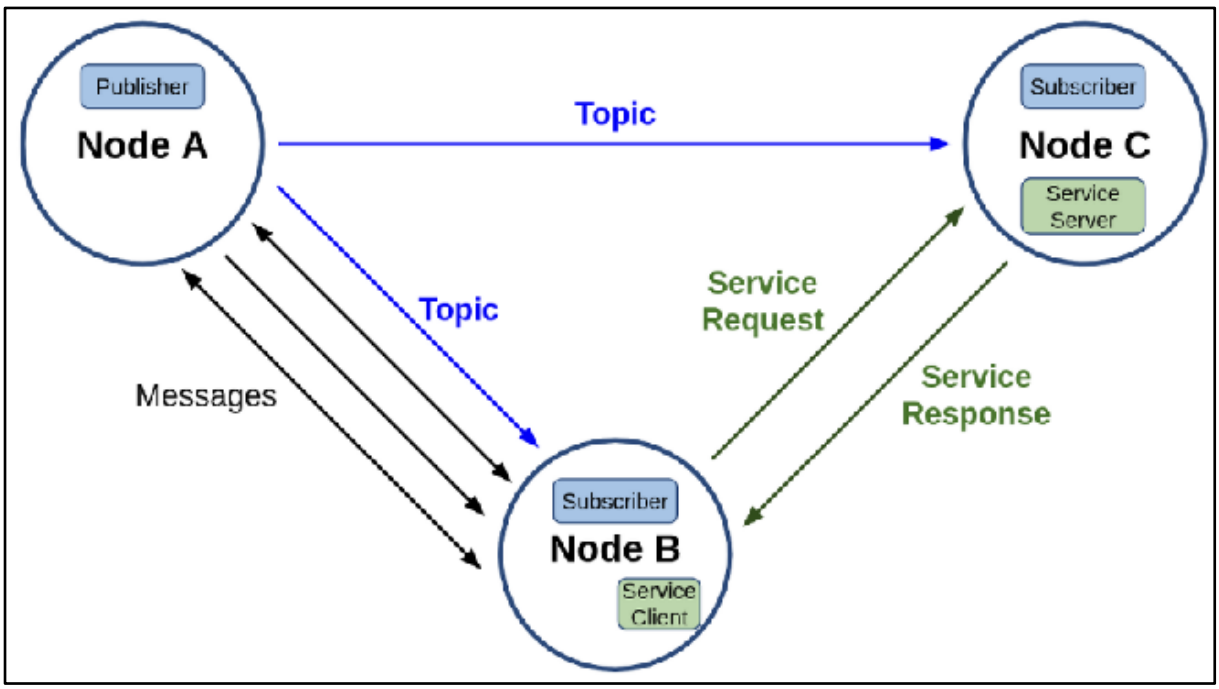
노드(node)는 아래의 그림처럼 Node A, Node B, Node C라는 노드가 있을 때 각각의 노드들은 서로 유기적으로 Message로 연결되어 사용된다. 수행하고자 하는 테스크가 많아질수록 메세지로 연결되는 노드가 늘어나며 시스템이 확장될 수 있게된다. 노드들 간의 메세지가 토픽, 서비스, 액션, 파라미터이다.



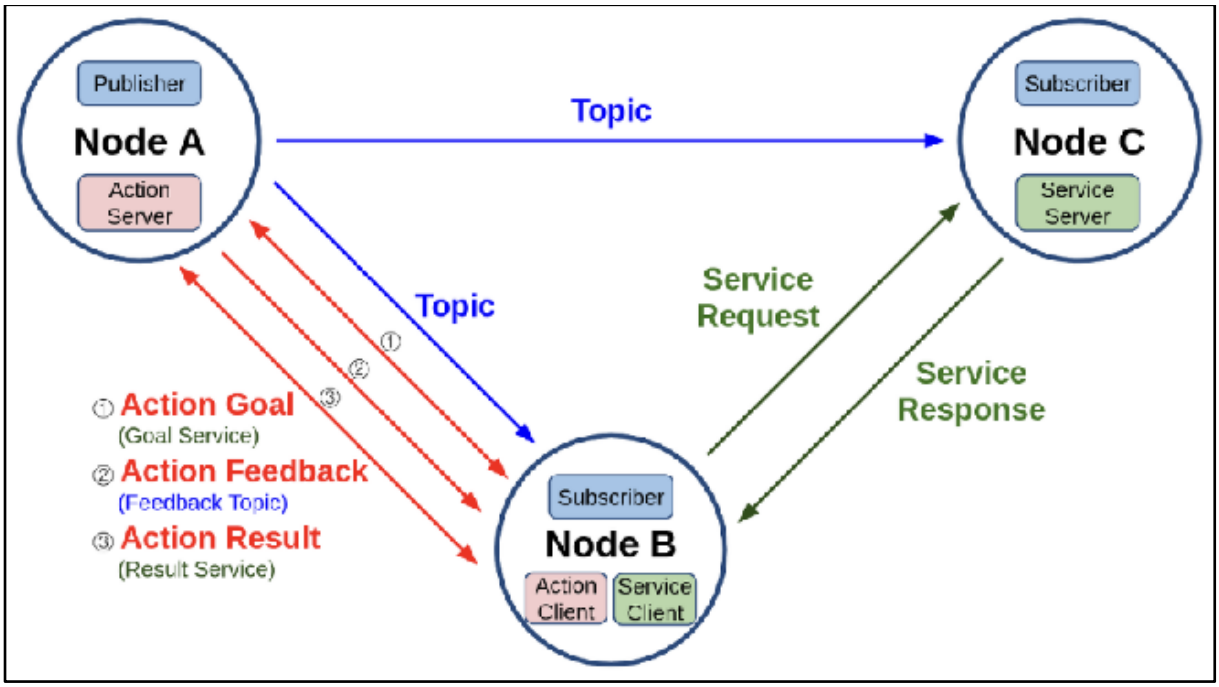
토픽(Topic)은 아래와 같이 `Node A - Node B`, `Node A - Node C`처럼 비동기식 단방향 메시지 송수신 방식으로 msg 메시지 형태의 메시지를 발간하는 Publisher와 메시지를 구독하는 Subscriber 간의 통신이라고 볼 수 있다. 이는 1:N, N:1, N:N 통신도 가능하며 ROS 메시지 통신에서 가장 널리 사용되는 통신 방법이다.



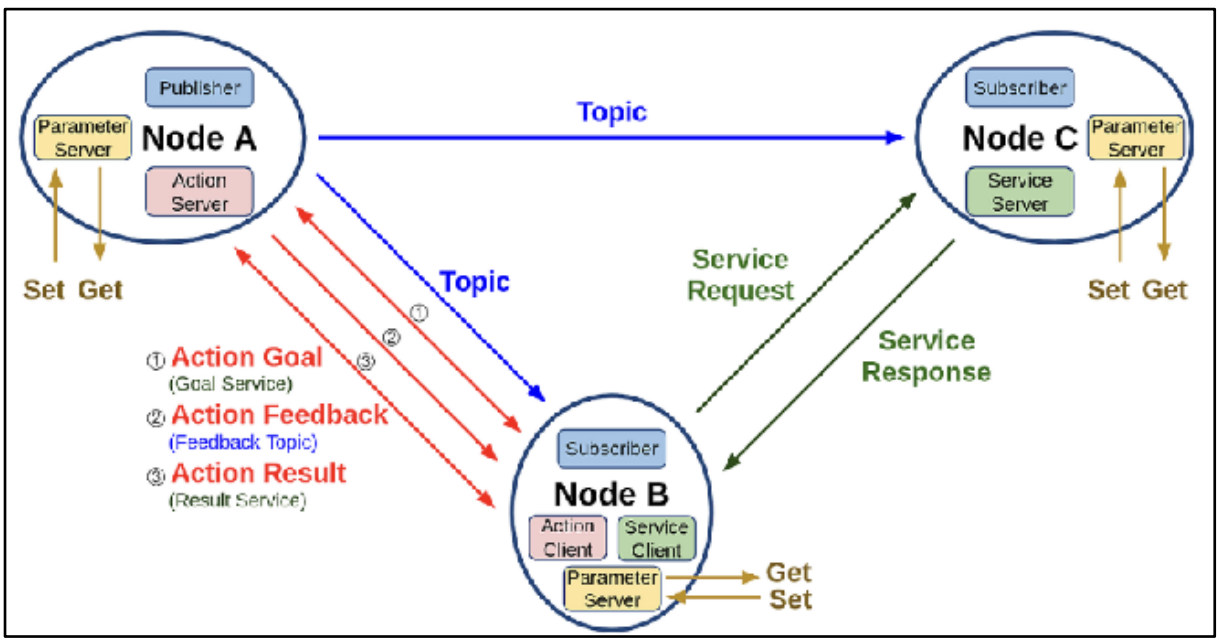
서비스(Service)는 아래의 그림과 같이 `Node B - Node C`처럼 동기식 양방향 메시지 송수신 방식으로 서비스의 요청(Request)을 하는 쪽을 Service client라고 하며 서비스의 응답(Response)을 하는 쪽을 Service server라고 한다. 결국 서비스는 특정 요청을 하는 클라이언트 단과 요청받은 일을 수행 후에 결과값을 전달하는 서버 단과의 통신이라고 볼 수 있다. 서비스 요청 및 응답 (Request/Response) 또한 위에서 언급한 msg 메시지의 변형으로 srv 메시지라고 한다.



액션(Action)은 `Node A - Node B`처럼 비동기식+동기식 양방향 메시지 송수신 방식으로 액션 목표 Goal를 지정하는 Action client과 액션 목표를 받아 특정 태스크를 수행하면서 중간 결괏값에 해당되는 액션 피드백(Feedback)과 최종 결괏값에 해당되는 액션 결과(Result)를 전송하는 Action server 간의 통신이라고 볼 수 있다. 액션의 구현 방식을 더 자세히 살펴보면 그림 5와 같이 토픽(topic)과 서비스(service)의 혼합이라고 볼 수 있는데 액션 목표 및 액션 결과를 전달하는 방식은 서비스와 같으며 액션 피드백은 토픽과 같은 메시지 전송 방식이다. 액션 목표/피드백/결과(Goal/Feedback/Result) 메시지 또한 위에서 언급한 msg 메시지의 변형으로 action 메시지라고 한다.



파라미터(Parameter)는 각 노드에 파라미터 관련 Parameter server를 실행시켜 외부의 Parameter client 간의 통신으로 파라미터를 변경하는 것으로 서비스와 동일하다고 볼 수 있다. 단 노드 내 매개변수 또는 글로벌 매개변수를 서비스 메시지 통신 방법을 사용하여 노드 내부 또는 외부에서 쉽게 지정(Set) 하거나 변경할 수 있고, 쉽게 가져(Get)와서 사용할 수 있게 하는 점에서 목적이 다르다고 볼 수 있다.



5. ROS 명령어

1.Package 관련

명령어	설명
catkin_create_pkg<my_package_name> <dependecies>	새로운 패키지를 생성한다.
rospack list	시스템에 설치되어 있는 패키지를 확인한다.
rospack find <package_name>	설치된 패키지의 위치를 확인한다.
rospack depends <package_name>	패키지와 의존성있는 패키지들을 확인한다.
roscd <location_name>	패키지가 존재하는 디렉터리로 이동한다.

2.Topic 관련

명령어	설명
rostopic list rostopic type <topic_name> rostopic info <topic_name> rostopic echo <topic_name>	시스템에서 사용중인 토픽을 확인한다. 토픽의 타입을 확인한다. 토픽의 타입과 더불어 현재 해당 토픽을 구독하고 발행하는 노드의 정보를 확인한다. 전달중인 토픽의 메시지 내용을 확인한다.

3.Message 관련

명령어	설명
rosmsg show <topic_type>	토픽 안의 메시지 데이터의 타입을 확인한다.

4.Node관련

명령어	설명
rosnode list rosnode info <node_name> rqt_graph	시스템에 실행중인 node의 리스트를 확인한다. 특정 node의 정보를 확인하다. (Subscription, Publication, etc...) 노드들 간의 연결된 메시지 관계를 시각화해서 확인한다.

5.실행 관련

명령어	설명
rosrun <package_name> <code_file_name> roslaunch <package_name> <launch_file.launch>	작성한 소스코드를 실행시킨다. rosrun 명령을 쉽게 사용하기 위한 명령으로 docker-compse와 비슷한 개념이다. 여러개의 node를 한번에 실행시킬 때 사용된다. 이때 반드시 패키지에 포함되어 있는 launch 파일이어야 한다.

6.Bag 관련

명령어	설명
rosbag record -O <output_file_name> <topic_name>	topic에서 발생되는 데이터를 output_file_name에 기록한다.
rosbag record -a	현재 시스템에 발생중인 모든 토픽에 대한 데이터들을 기록한다.
rosbag record <topic_1> ... <topic_n>	입력된 토픽들의 데이터를 기록한다.
rosbag info <*.bag>	입력된 토픽들의 데이터를 기록한다.
rosbag play <output_file_name.bag>	기록된 토픽의 데이터를 발생시킨다.
rosbag play -r 2 <output_file_name.bag>	기록된 토픽의 데이터를 2배속으로 재생한다.
rosrun image_view video_recorder image:='/usb_cam/image_raw' _filename:='track2.avi' _fps:=30	카메라의 토픽을 받아 영상 파일로 저장한다.

7.기타 관련

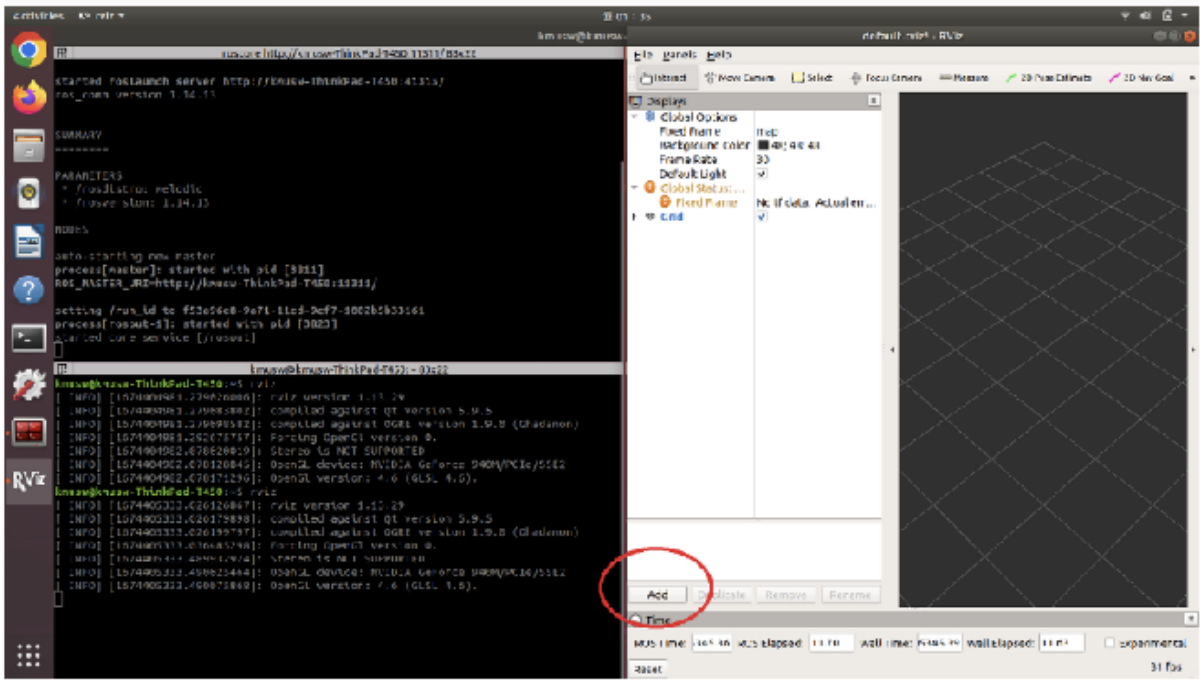
명령어	설명
rosls <location_name> rosed <file_name>	Linux의 ls와 동일하게 파일들의 목록을 출력한다. 파일을 편집한다.

6. ROS 도구

1.RViz

RViz(ROS Visvualiztion Tool)는 ROS 도구 중 하나이다. 센서 데이터의 시각화되어 있다. (레이저 거리 센서(LDS) 센서의 거리 데이터, Depth Camera의 포인트 클라우드 데이터, 카메라의 영상 데이터, IMU 센서의 관성 데이터 등..) Rviz를 이용하면 센서 및 로봇 관련 데이터 시각화가 매우 간단하다. 아래의 명령어를 누르면 쉽게 들어갈 수 있다. rviz

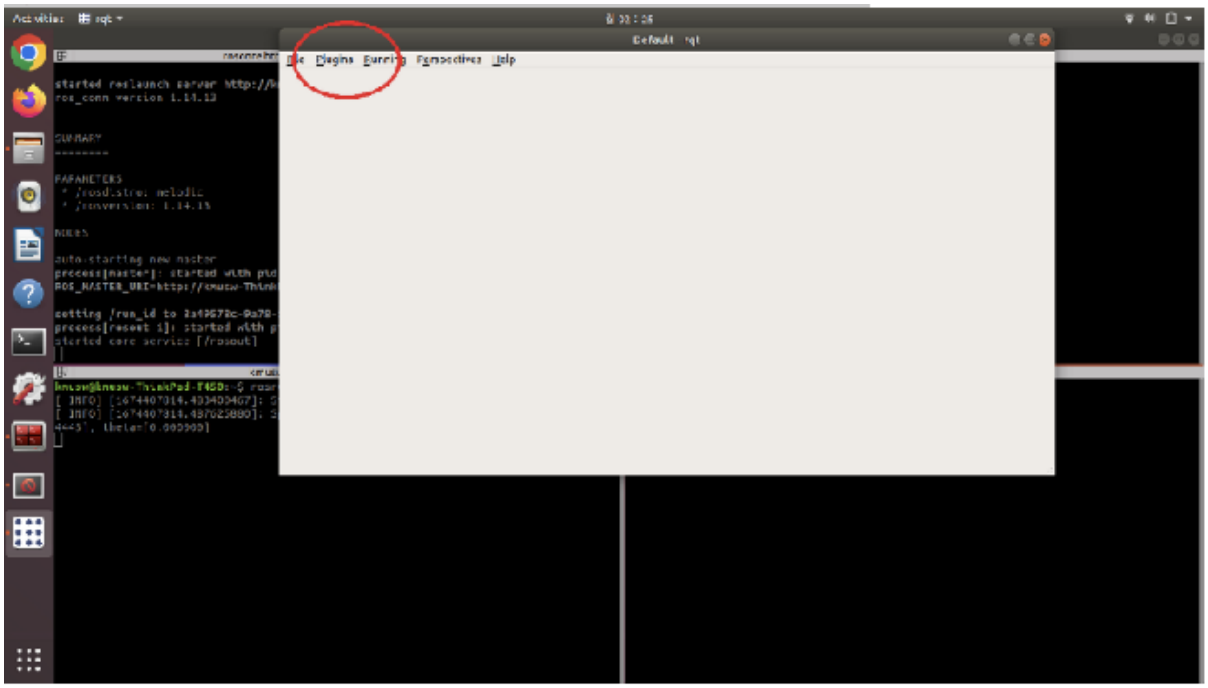
ADD를 누르면 센서와 장치정보들을 사용가능하다.



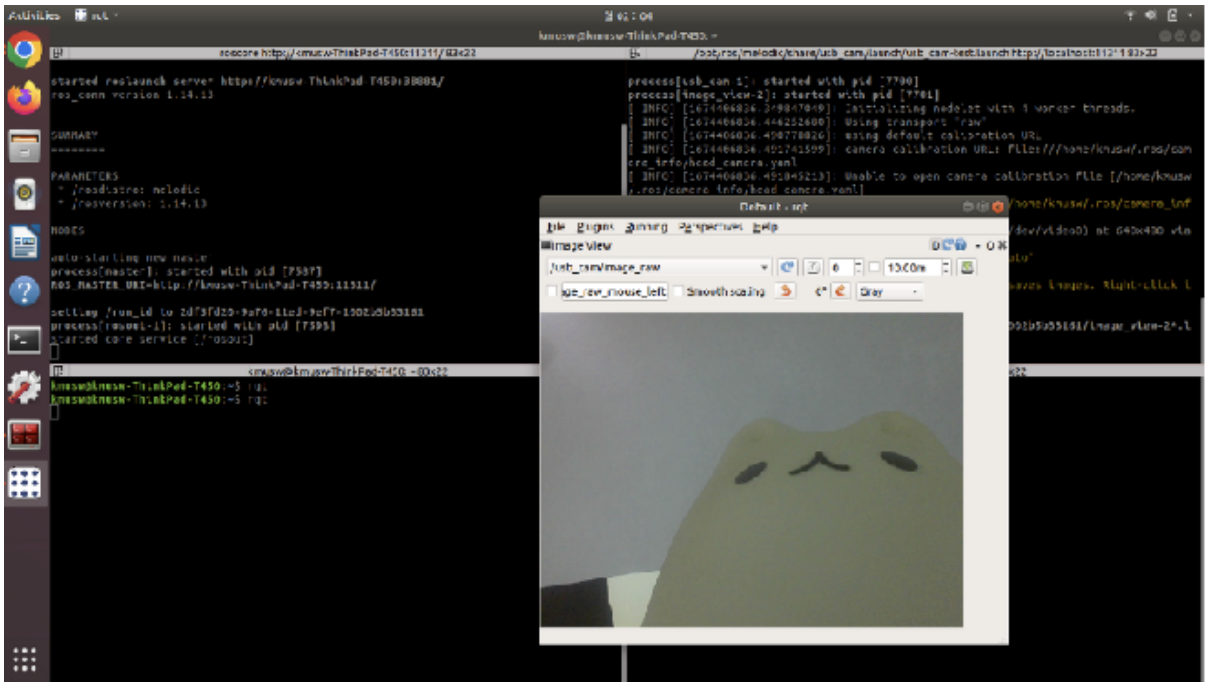
2. rqt

로봇으로부터 얻을 수 있는 데이터를 쉽게 확인하고 관리할 수 있도록 하는 rqt이다. GUI개발에 있어서 공통으로 필요한 부분들을 API형태로 제공을 한다.

Plugin에 들어가서 선택적으로 골라서 실행이 가능하다.



3. rqt_image_view



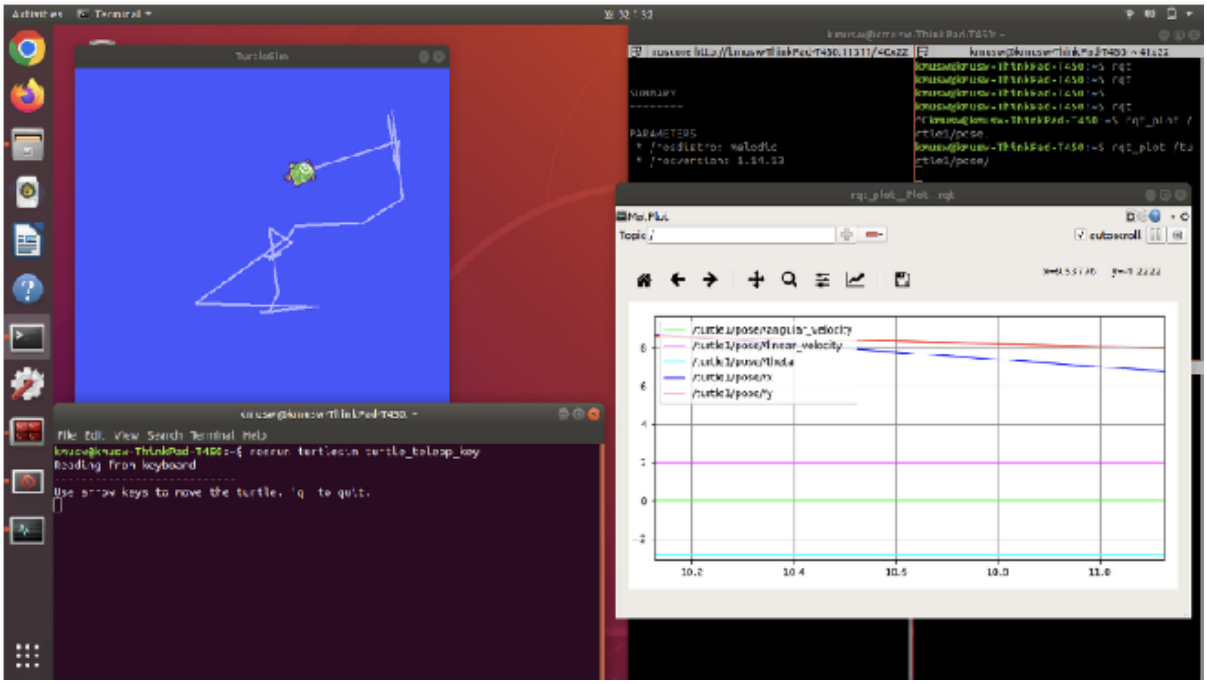
3. rqt_plot

$ roscore

$ rosrun turtlesim turtlesim_node

$ rosrun turtlesim turtle_teleop_key

$ rqt_plot /turtle/pose/



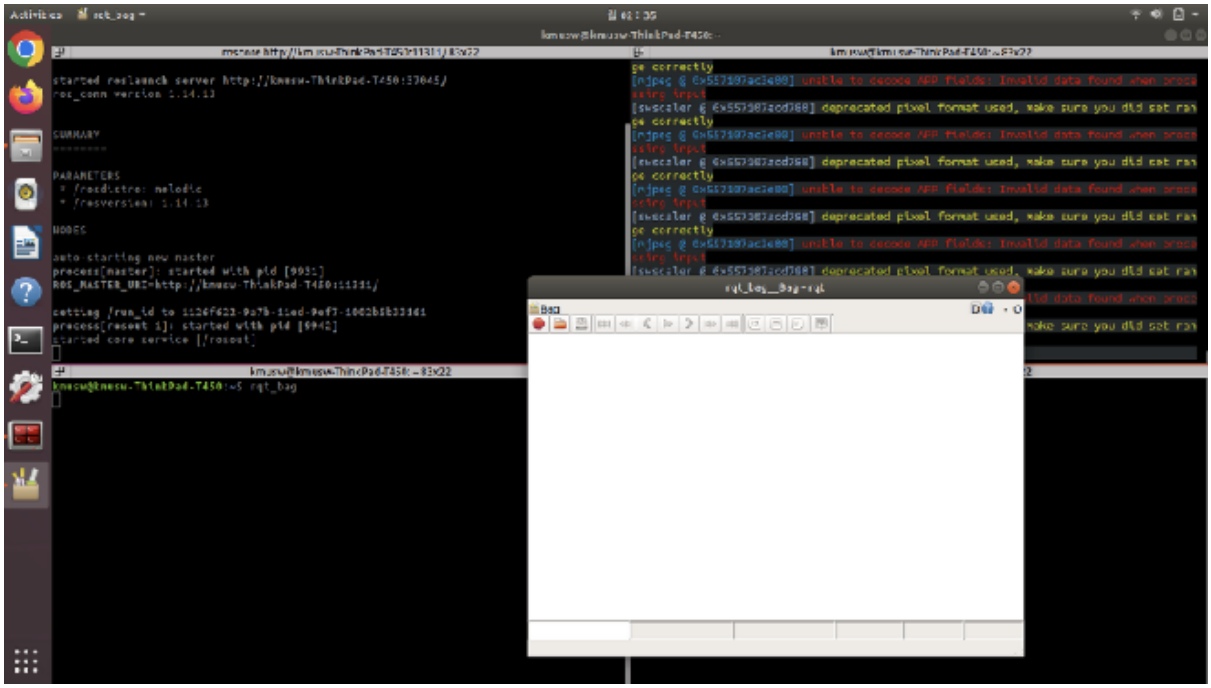
4. rqt_bag

$ roscore

$ rosrun usb_cam usb_cam_node

$ rqt_bag

영상이나 음성 재생하기



5. Gezebo

- 로봇 개발에 필요한 3차원 시뮬레이션을 위한 로봇, 센서, 환경 모델 등을 지원하고 물리엔진을 탑재하여 실제와 근사한 결과를 얻을 수 있는 3차원 시뮬레이터이다.

- DAPPA Robotics Challenge의 공식 시뮬레이터

- ROS와의 호환성이 매우 좋다