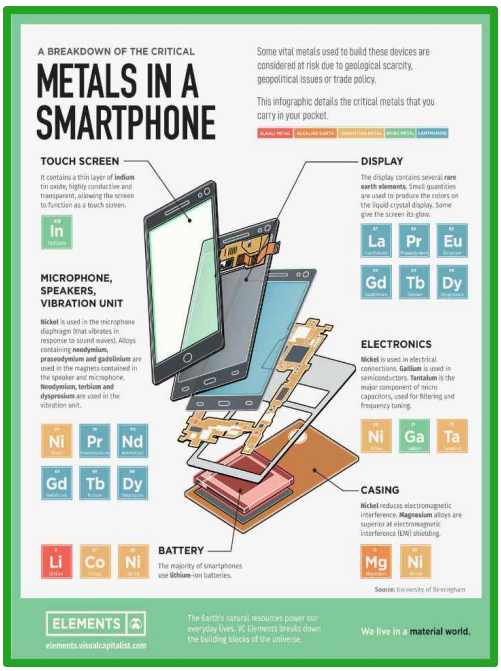
<RO:BIT ROS 보고서>

18기 지능팀 인턴 선현빈

1. 로봇 소프트웨어 플랫폼의 필요성

로봇의 대중화를 위한 조건

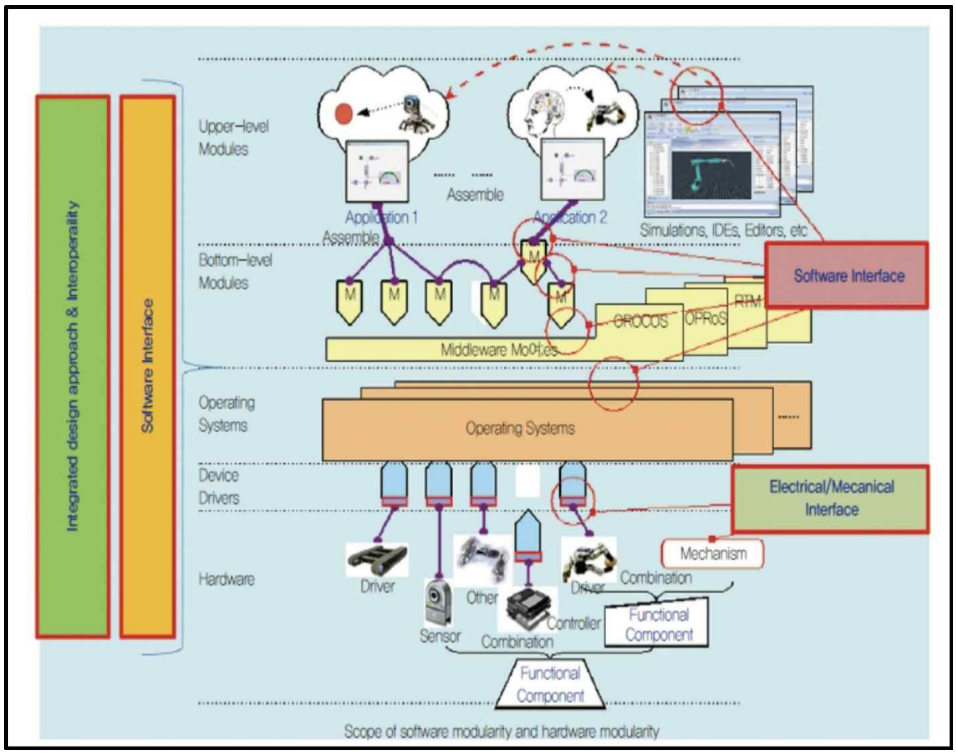
다양하고도 수많은 전자 장치와 기계, 로봇 등이 탄생하고, 급속도로 발전하고 있는 현 시대에 가장 대중화되어 있는 기기는 스마트폰(Personal Phone)과 PC(Personal Computer)라고 할 수 있으며, '대중화'라는 측면에서 바라볼 때 두 기기와 비견될 만한 기기는 없다고 해도 과언이 아니다. 이러한 PC와 스마트폰은 공통적으로 수많은 모듈로 구성되어 있으며, OS와 APP가 존재하고, 산업적 생태계가 존재한다.



[스마트폰은 스크린, 디스플레이, 배터리 등 다양한 하드웨어 모듈로 구성되어 있다]

세 가지 특징 중에서도 하드웨어 모듈화 및 규격화를 가능케 하는 운영 체제가, 나머지 두 특징을 활성화할 수 있었던 핵심이다. 특히 PC는 대중화되는 과정에서 하드웨어 모듈과 운영 체제, 응용 프로그램(서비스), 그리고 유저로

구성되는 생태계를 구축하였고, 그 바탕에는 PC 전반에 사용되는 OS의 통합이 있었다. 이에 따라서 산업적 생태계가 조성되었기에 PC가 대중화될 수 있었다. 스마트폰 역시 PC와 마찬가지로, PC가 대중화되기 위해 지나쳤던 과정을 그대로 답습하여 대중화되었으며, 둘 모두 보이지 않는 생태계 속의 분업이 이루어지고 있다. 이러한 이유로 로봇 분야도 PC와 스마트폰처럼 Personal Robot의 시장을 활성화하고 대중화하기 위해 같은 단계를 밟아갈 것으로 전망된다.

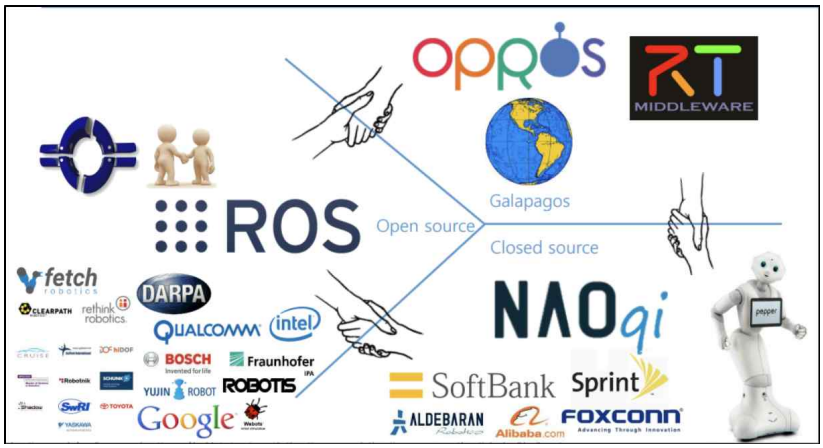


[서비스 로봇의 모듈화-하드웨어와 소프트웨어 각각이 모듈화되어 로봇이 구성되고 있다(2014년 자료)]

로봇 소프트웨어 플랫폼

로봇 소프트웨어 플랫폼은 다양한 산업과 환경에서 사용할 목적으로 개발되는 무수히 많은 종류의 로봇들의 시스템 알고리즘과 응용 프로그램을 쉽게 개발하고 동작시키기 위한 아키텍쳐이자 플랫폼이다. 특히 목적에 부합하고, 나아가 스마트폰 시장의 IOS, 안드로이드나 PC의 리눅스, 윈도우 등과 같이 시장 지배를 위해 센싱, 네비게이션, SLAM, 하드웨어 추상화, 패키지 관리 및

라이브러리 등 보다 다양한 기능을 갖춘 플랫폼이 등장하고 있다. 로봇 소프트웨어 플랫폼은 오픈 소스 기반, Closed 소스 기반, Galapagos의 세 가지로 분류할 수 있다. 오픈 소스 기반으로 ROS가 대표적이며, 나머지는 각각 NAOqi, OPROS, 미들웨어 등이 대표적이다.



[주요 로봇 운영 체제]

로봇 소프트웨어 플랫폼의 필요성과 미래

로봇 소프트웨어 플랫폼은 하드웨어를 추상화-규격화-모듈화함으로써 로봇 분야에서 하드웨어와 운영 체제, 응용 프로그램을 각각 분리하는 것이 가능케 한다. 이를 만족하면서도 사용자의 니즈에 맞추어야 하고, 수많은 플랫폼들과 경쟁해야 하기에, 가격이 낮아지면서 성능은 향상되는 현상이 일어난다. 이는 유저의 증가를 불러오며, 구매와 피드백이 증가하게 되어 산업적 생태계의 선순환을 가져온다. 특히, ROS와 같은 소프트웨어 플랫폼을 통해 타 분야에서 로봇 분야로 진입하고자 하는 사람에게 요구되는 선행 지식의 양이 큰 폭으로 감소하기에 개발자, 유저 모두 급증하여 로봇 분야의 급속 발전의 발판이 될 수 있다.

2. ROS란 무엇인가

ROS의 정의

ROS는 현재 가장 많은 유저가 사용하고 있는 로봇 소프트웨어 플랫폼으로,

안드로이드와 같은 오픈소스 기반이자 로봇을 위한 '메타 운영 체제'이다. 이는 엄밀히 따져서 Robot Operating System이라는 이름의 ROS가 운영 체제는 아님을 의미하기도 하며, ROS가 큰 범주에서 소프트웨어 프레임 워크이자, 미들웨어임을 시사한다.

*) 메타 운영 체제란? : 메타 운영 체제는 정확히 정의되어 있는 용어는 아니나, 운영체제 위에서 별도의 시스템으로써 기능하여 임의의 동작을 위해 실행되는 것으로 응용 프로그램과 분산 컴퓨팅 자원을 활용해 스케쥴링, 로드, 감시, 에러 처리 등을 실행하는 시스템을 의미한다. 따라서 기존 운영 체제를 이용한다는 점에서, 본질적으로 OS보다는 프레임워크, Tool Box의 성격을 같는다.

ROS의 발전 배경 및 현황

ROS는 2000년대 스탠포드 대학 내 로봇 연구실에서 박사 과정을 밟고 있던 Eric Berger와 Keenan Wyrobek이 로봇 기술에 관심을 갖는 많은 학생이 그 난이도때문에 포기하는 것을 보며, 누구나 로봇 분야에 쉽게 진입할 수 있게 하기 위해 기술을 만드는 것에서 시작했다. ROS의 프로토타입 개발 도중에 구글 초기 검색 엔진 기술에 기여한 Scott Hassan이 후원하며, 두 사람을 '윌로우 개러지(로봇 기업)'으로 스카우트 한다. 이후, 윌로우 개러지가 따로 분리한 프로젝트 중 오픈소스 로봇 재단이 있으며, 이곳에서 ROS를 비롯한 로봇 오픈소스 기술을 후원 및 관리하고 있다. 현재는 'open robotics'로 재단명을 변경했다. 오픈 로보틱스는 ROS와 이그니션(Gazebo), 미들웨어 프레임워크 Open-RMF를 핵심기술로 보유하고 있으며, DARPA, NASA, 아마존, 엔비디아, 도요타, 싱가포르 정부, MS 등 여러 다국적 기업 및 정부와 협력하고 있으며, 그 호환성을 바탕으로 로봇 프레임워크 시장 점유율을 넓혀가고 있는 해당 분야의 선두주자이다.

[open robotics & ROS logo]

ROS의 특징

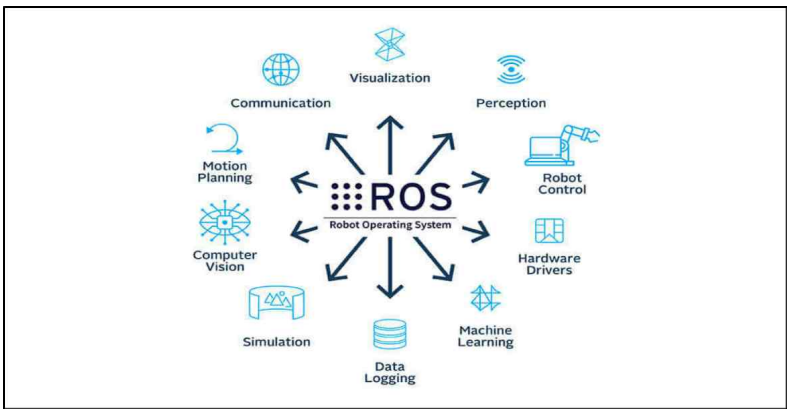
ROS는 앞서, 발전 배경에서 언급한 바와 같이, 다양한 직군의 많은 사람들이 로봇 산업에 쉽게 진입하게 하여 로봇 산업 전반을 발전시키고, 생태계를 구축하는 것에 목적을 두고 있다. 따라서 ROS는 가장 큰 이점 중 하나로 '이기종 디바이스 간의 통신 지원'을 갖고 있다. 이는 아두이노나 안드로이드, 우분투, IOS, ROS 등 서로 다른 운영 체제 및 응용 프로그램을 사용하고 있는 로봇간의 통신이 가능하다는 것과 일맥상통한다. 추가적으로 OS를 탑재할 수 없는 MCU의 경우에 시리얼 통신이나 블루투스 등에 대한 라이브러리도 제공하고 있으므로, 로봇 생태계를 구축한다는 목적에 부합하는 범용성을 지니고 있다. 따라서 ROS는 통신 기반의 프로그램으로 분류할 수 있다.

이외의 특징은 아래와 동일하다.

- 프로그램 재사용성 : ROS는 오픈 소스 기반의 소프트웨어 플랫폼이므로, 사용자는 자신이 개발하고자 하는 특정 부분에만 집중하고, 나머지 기능은 관련 패키지를 다운로드하여 사용할 수 있다. 이와 같은 맥락으로, 자신이 개발한 프로그램을 다른 사람들이 사용할 수 있도록 공유하는 것도 가능하다.

- 원격 제어에 유리 : 노드 패키지화를 거쳐 최소 실행 단위로 프로그램이 나뉘어져, 노드 간의 데이터 송수신을 진행하므로, 디버깅에 유리하고 하드웨어 의존적이지 않고, 네트워크에서 통신을 제공하므로 원격 제어에 유리하다.

- 개발 도구 및 요소 지원 : 시뮬레이터, 디버깅, 시각화 도구 등 매우 다양하고 로봇 개발에 사용되는 도구를 제공하므로 센싱 데이터 등에 대한 직관적인 정보 이해가 가능하다.



[ROS가 사용되는 범위]

- 분산 매개 변수 시스템 : 시스템 사용 변수를 글로벌 키 값으로 작성 및 공유, 수정하여 외부에서 실시간으로 반영하는 것이 가능하다.

3. ROS 버전 선택 및 개발 환경 구축

ROS 버전 선택

- **리눅스** : 5년간 기술 지원이 약속되어 있는 최신 LTS 버전의 우분투를 지원한다. 2년마다 매년 4월에 LTS 버전을 릴리즈한다. 릴리즈 3개월 이후에 사용하는 것을 권한다.(현재 최신 버전 : ubuntu 22.04)

- **ROS** : 2년마다 매년 5월에 LTS 버전을 릴리즈한다. 릴리즈 3개월 이후에 사용하는 것을 추천한다.(현재 최신 버전 : noetic)

- **Gazebo** : "gazebosim.org"에서 ROS 호환성 정보 검토 후 사용한다.

ROS 설치 과정

- **자동 설치 방법**

- **한 줄 설치 코드**

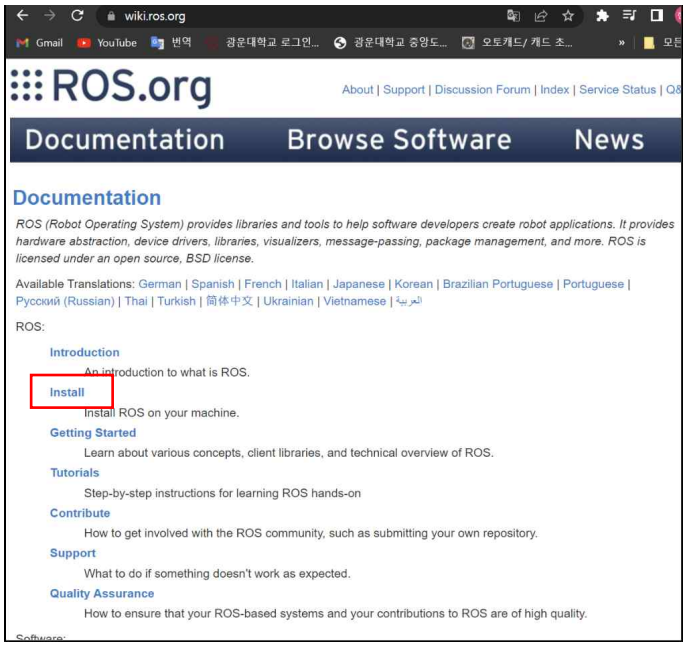
```
wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_kinetic.sh
&& chmod 755 ./install_ros_kinetic.sh && bash ./install_ros_kinetic.sh
```

[ROS 한 줄 설치 코드(ver.kinetic)]

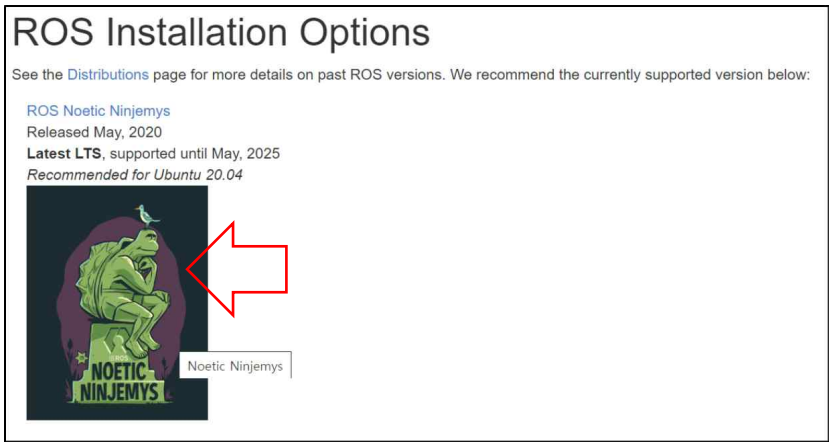
- **수동 설치 방법**

a. ROS.org, 로스 위키에 접속한다.

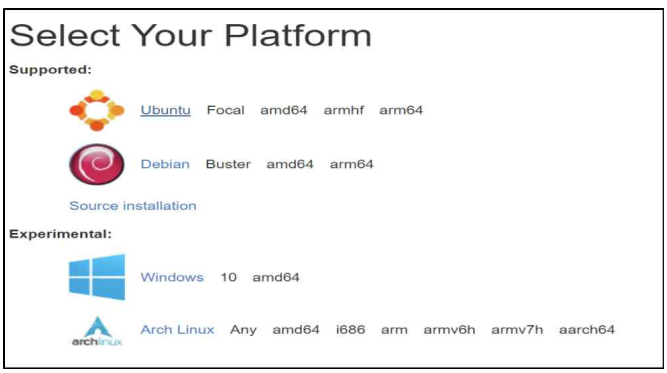
b. Install을 클릭한다.



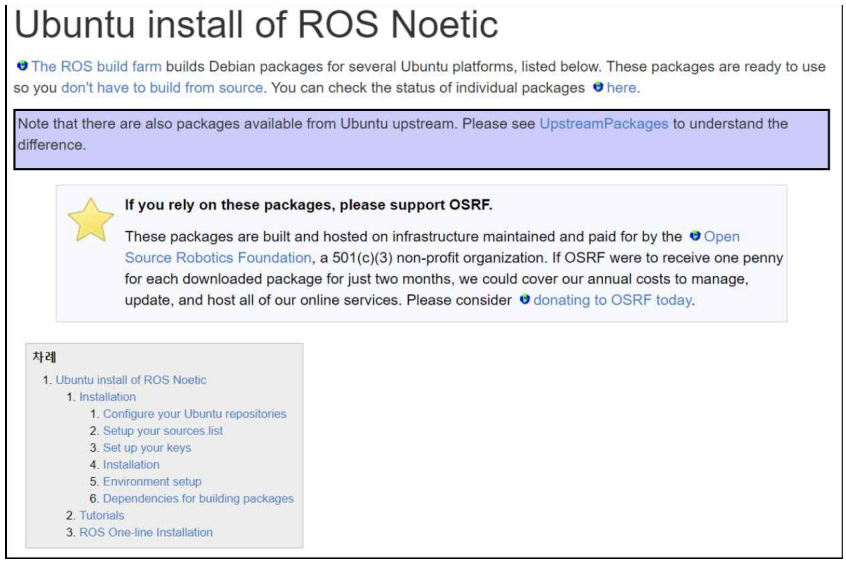
c. Noetic을 선택한다.



d. 사용하고 있는 OS를 선택한다.



e. 매뉴얼에 따라 설치한다.



Ubuntu install of ROS Noetic

The ROS build farm builds Debian packages for several Ubuntu platforms, listed below. These packages are ready to use so you don't have to build from source. You can check the status of individual packages here.

Note that there are also packages available from Ubuntu upstream. Please see UpstreamPackages to understand the difference.

If you rely on these packages, please support OSRF.

These packages are built and hosted on infrastructure maintained and paid for by the Open Source Robotics Foundation, a 501(c)(3) non-profit organization. If OSRF were to receive one penny for each downloaded package for just two months, we could cover our annual costs to manage, update, and host all of our online services. Please consider donating to OSRF today.

차례

ROS 환경 설정

환경 설정 wiki 사이트 : https://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

a. 환경 변수 설정 여부를 확인($ printenv | grep ROS)

b. $ source /opt/ros/noetic/setup.bash

c. <ROS> 작업 공간 생성

$ mkdir -p ~/catkin_ws/src

$ cd ~/catkin_ws/

$ catkin_make

$ source devel/setup.bash

$ echo$ROS_PACKAGE_PATH/home/youruser/catkin_ws/src:/opt/ros/noetic/share

d. alias <단축명령어> =''로 단축 명령 설정 등 기타 환경 설정

ROS 동작 테스트

ROS에 내장되어 있는 turtlesim 패키지로 ROS의 완전한 설치 및 동작 여부를 테스트할 수 있다.

1. 첫 번째 터미널 창에 roscore 입력
2. 두 번째 터미널 창에 rosrun turtlesim turtlesimm_node 입력(node 실행)
3. 세 번째 터미널 창에 rosrun turtlesim turtle teleop_key 입력

위의 세 가지 명령어를 각각 별도의 터미널 창을 생성하여 입력하여 준 뒤, teleop_key 창에서 방향키를 조작한다.

[turtlesim_node]

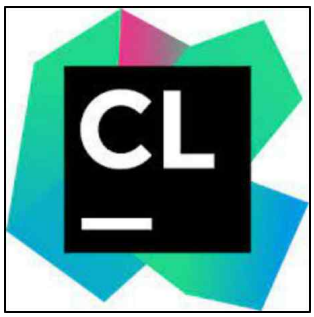
- 추가적으로 cm을 입력하여 컴파일을 확인한다.

ROS IDEs 추천

- > https://wiki.ros.org/IDEs 에서 사용 가능한 IDE를 확인할 수 있다.

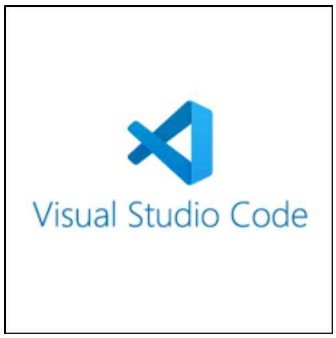
1. QtCreator, Qt Creator Plugin for ROS : GUI 기반으로 rqt를 사용할 때 유용하다. 그리고 CMakeLists.txt 파일을 그대로 사용할 수 있다.

2. Eclipse : 사용자가 많은 만큼, 관련 자료가 많이 존재한다. 하지만 Java 기반으로 개발되어 리소스가 무거운 경우가 있고, 복잡한 IDE라서 진입하기 어렵다.



3. CLion : 기본적으로 CMake를 지원하고, Python 코드 작업을 허용하는 C/C++ 작업환경 플랫폼이다. ROS 플러그인이 존재한다. 코드 자동 완성 기능으로 효율적인 작업이 가능하지만, 메모리 사용이 증가하는 경우가 있다.

4. Vim : ROS 개발을 위한 다양한 기능이 제공되는 vim-ros 플러그인을 제공. 텍스트 편집기로 효율성 및 확장성이 좋으며, 다양한 플랫폼과 콘솔, GUI 모드를 지원한다. 플러그인 호환성 문제가 발생할 수 있고, 초기 설정이 복잡하다.



5. VSCode : 소모하는 용량이 적고 빠른 속도의 편집기이며, 다양한 프로그래밍 언어를 지원한다. 또한 확장 프로그램이 많으며, Git 지원이 내장되어 있다.

4. ROS의 중요 개념

ROS의 기본 개념 정리

기본	Node	메시지로 데이터 통신이 가능한 최소 단위이자, 하나의 로봇, 센서, 프로그램이다. ROS에서는 노드 단위로 작업을 하게 되며, 노드와 패키지를 나누는 것은 개발자에게 달려있다.
	Package	하나 이상의 노드와 노드 실행을 위한 설정 파일, 의존성 라이브러리 등을 갖고 있다. 패키지의 묶음은 메타 패키지라 하여 따로 분리한다.

	Message	메시지를 통해 노드간의 데이터를 주고 받게 된다. 메시지는 int, float, point, boolean과 같은 변수 형태이며, 메시지가 메시지를 품고 있는 데이터 구조, 배열 구조의 메세지 등도 사용할 수 있다.
	Master	$roscore을 통해 마스터를 구동한다. XMLRPC(XML-Remote Procedure Call)이라는 아주 간단한 서버를 구현하여, 노드를 연결해, 메시지 통신을 위한 네임 서버의 역할을 한다.
	Launch	launch파일을 추가하여, 여러 노드를 동시에 실행시킬 수 있다. 그리고 파라미터, 노드 명 등을 변경시킬 수도 있다.
	Parameter	파라미터는 어떤 변수를 네트워크에 지정하여, 외부 노드에서 변경시키는 등 외부에서 변화를 줄 수 있도록 전역적으로 설정하는 것이다.. ROS Master의 기능 중 일부라고 볼 수 있으며, rosparam 관련 명령어로 조작한다.
토픽	Topic	단방향이자, 연속적을 갖는 메시지 통신 방법. 목적에 따라서 1대1, 1대N, N대1, N대N 통신 모두 가능하다. 메세지 통신의 대부분은 Topic을 사용하게 된다.
	Publisher	토픽에서 메세지를 송신하는 노드
	Subscriber	토픽에서 메세지를 수신받는 노드
서비스	Service	양방향이자, 일회성인 메시지 통신 방식이다. 다시 하기 위해서는 재접속을 해야 한다.
	Service server	서비스 클라이언트의 요청에 대한 응답을 하는 대상
	Service client	서비스 서버에게 서비스를 요청하며, 서비스의 응답을 받는 대상
액션	Action	액션은 로봇 통신의 복잡도가 높거나 중간에 피드백이 필요할 때 사용한다.

	Action server	전달 받은 액션 목표에 대한 액션 피드백과 최종 액션 결과를 클라이언트에세 송신한다.
	Action client	액션 서버에게 액션 목표를 전달하고, 서버로부터 액션 목표를 수신받는다.

ROS 표준 단위 및 메시지 데이터 타입

표준 단위 : ROS에서는 회전축에 대해서는 오른손 법칙을 기준으로 하며, x축은 forward, y축은 left, z축은 up 방향을 가리킨다. 다른 물리량의 표준 단위에 대해서는 SI 단위계를 사용한다.

- ROS 프로그래밍 규칙에 대해서는 ROS 위키의 ROS C++ style guide를 참조할 수 있다.

- ROS에서 사용하는 변수는 int형, uint형, float형, string형, bool형을 비롯해 time형이나 duration과 같은 처음 접하는 변수형도 사용한다. 자세한 사항은 아래의 표나 ROS위키 msg(메시지) 항목을 참조할 수 있다.

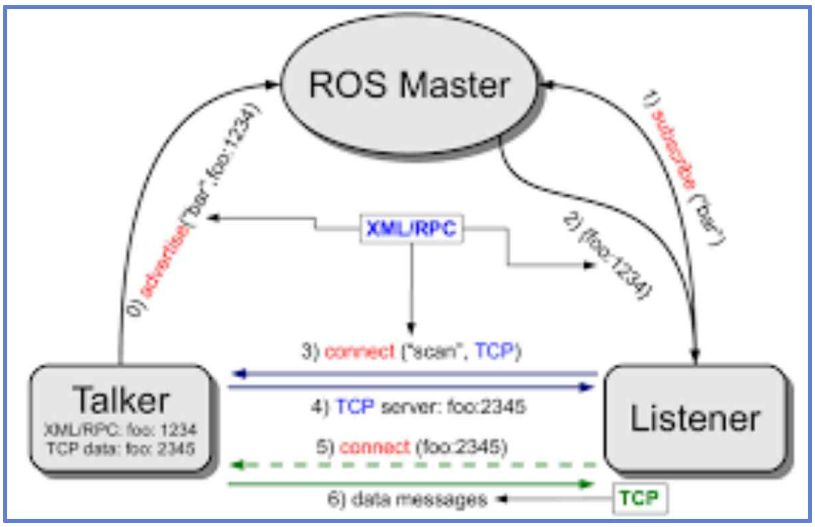
Primitive Type	Serialization	C++	Python
bool	unsigned 8-bit int	uint8_t	bool
int8	signed 8-bit int	int8_t	int
uint8	unsigned 8-bit int	uint8_t	int
int16	signed 16-bit int	int16_t	int
uint16	unsigned 16-bit int	uint16_t	int
int32	signed 32-bit int	int32_t	int
uint32	unsigned 32-bit int	uint32_t	int
int64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

Array Type	Serialization	C++	Python
fixed-length	no extra serialization	boost::array, std::vector	tuple
variable-length	uint32 length prefix	std::vector	tuple
uint8[]			bytes
bool[]		std::vector<uint8_t>	list of bool

[ROS에서 사용하는 데이터 타입에 대한 표]

메시지 통신

- 메시지 통신은 기본적으로 노드 간의 정보 교류를 마스터가 관리하며, 모든 정보는 '메시지'를 통해 이루어진다.



[토픽의 통신]

- 순서 : 퍼블리셔 노드가 구동하면 마스터가 정보를 전달받고, 서브스크라이버 노드가 구동되어 자신의 노드명, 토픽명, 메시지형, IP 정보. 포트 번호 등을 마스터에게 전달한다. 이후에, 마스터가 정보의 일치를 판단하여 각 노드에게 맞는 정보를 전달한다.

- 정보를 전송받은 두 노드는 먼저, 서브스크라이버가 퍼블리셔에게 접속 요청을 한다.

- 퍼블리셔 노드가 접속에 응답하면 서로 정보를 전송하고 두 노드가 직접 연결된다.

- 이후, 퍼블리셔 노드가 메시지를 발행하여 서브스크라이버 노드가 구독하면 통신 과정이 완성된다.

네임

네임은 노드, 메시지(토픽, 서비스, 액션, 파라미터)가 가지는 고유의 식별자이며, 노드나 메세지의 이름을 변경시키고 싶을 때 사용한다.

- global : 문자 없이 네임을 바로 작성하거나 네임 앞에 슬래쉬(/)를 붙인다.

- private: 네임 앞에 틸트(~)를 붙인다.

Node	Relative (default)	Global	Private
/node1	bar → /bar	/bar → /bar	~bar → /node1/bar
/wg/node2	bar → /wg/bar	/bar → /bar	~bar → /wg/node2/bar
/wg/node3	foo/bar → /wg/foo/bar	/foo/bar → /foo/bar	~foo/bar → /wg/node3/foo/bar

[네임 작성 예시]

파라미터

[파라미터 및 메시지 통신 과정의 구조도(모든 방법을 표시)]

파라미터는 ROS Master의 기능 중 일부라고 볼 수 있으며, getParam(매개변수 읽기), setParam(매개변수 초기설정) 등을 통해 사용한다. 파라미터는 int, float, boolean, string, list 등으로 설정할 수 있다.

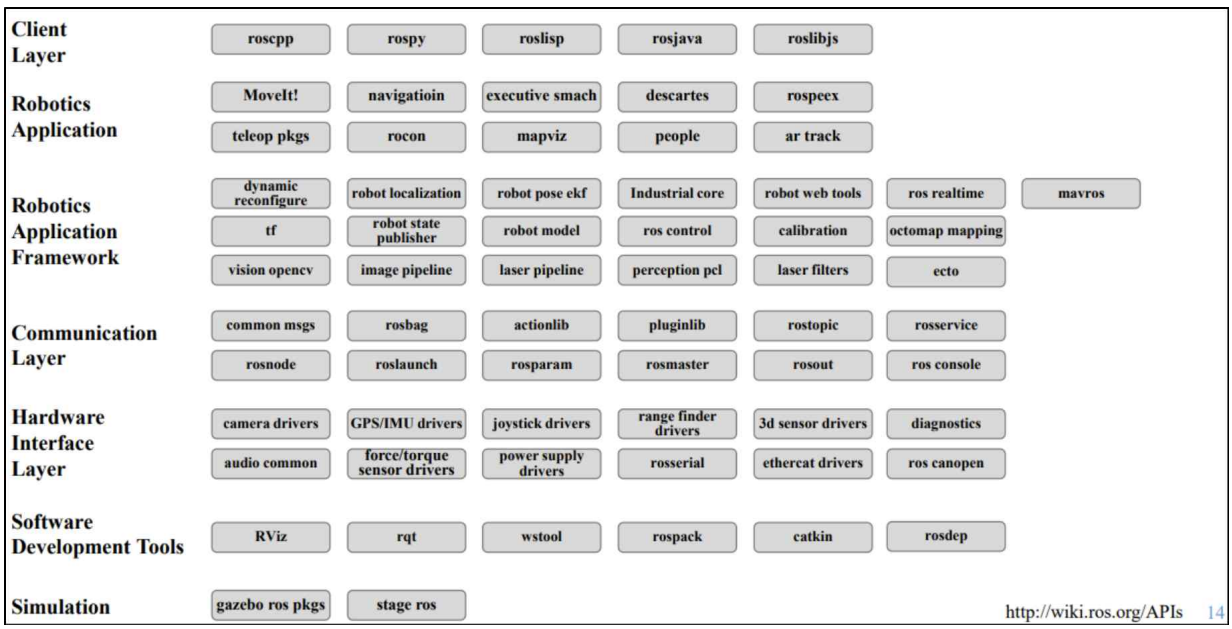
5. ROS Commands List

ROS 명령어 정리(https://wiki.ros.org/ROS/CommandLineTools)

bag 파일 관련	rosbag	ROS 주제 기록 및 재생을 위한 tool set
파일 관련	roscd	이름만으로 패키지 등의 위치로 이동
		roscd 위치 이름[/subdirectory]
	rosls	ROS 패키지의 파일 목록을 확인
	rosed	ROS 패키지의 파일을 편집
	roscp	ROS 패키지의 파일을 복사
	rosclean	파일 시스템 리소스를 정리
중요	roscore	Master, Parameter server, rosout 등을 실행
	roslaunch	여러개의 노드를 실행
		roslaunch 패키지명 런치파일
	rosrun	하나의 노드를 실행
		rosrun 패키지명 노드명
catkin 관련	catkin_create_pkg	ROS 패키지 형태의 디렉토리 및 파일을 자동으로 생성
	catkin_create_qt_pkg	ROS qt패키지 형태의 디렉토리 및 파일을 자동으로 생성
	catkin_make	catkin 빌드 시스템 기반으로 빌드를 실행
	catkin_prepar	릴리즈 시에 사용되는 log 정리 및 version

	e_release	tagging
	catkin_genera te_changlog	릴리즈 시에, CHANGELOG.rst 파일 생성 및 업데이트
	catkin_init_w orkspace	catkin 빌드 시스템의 작업 디렉토리 초기화
	catkin_find	catkin에서의 검색 기능
정보 관련	rostopic	topic의 정보를 확인
	rosservice	서비스의 정보 확인
	rosnode	노드 정보 확인
	rosparam	파라미터 정보 확인 및 수정
	rosmsg	메시지 정보 확인
	rossrv	서비스 정보 확인
	rosversion	패키지 및 릴리즈 버전 정보 화긴
	roswtf	시스템 검사
패키지 관련	rosinstall	추가 패키지 설치
	rosdep	패키지의 의존성 파일 설치
	rospack	패키지 관련 정보 열람

6. ROS의 구성 요소



[ROS의 구성 요소]

- Client Layer

  - 간단히 말하자면, 다양한 프로그래밍 언어에 대한 지원에 해당하는 요소로, roscpp, rospy, roslisp, rosjava, roslibjs 등을 지원한다.

  - 각각 c++, 파이썬, LISP, 자바, 자바스크립트이다.

- Robotics Application

  - 내비게이션, 매니퓰레이션 등의 기능을 포함하고 있다.

  - MoveIt!, navigation 등이 대표적인 기능이다.

- Robotics Application Framework

  - maping, realtime, laser filter, tf 등 각종 유용한 프로그램 지원에 대한 툴이 들어가 있다.

  - TF, 좌표변환(tranform)은 로봇의 각 조인트, 즉 관절들의 상대 좌표를 변환하여 트리 형태로 여러 조인트들 사이의 관계도를 표시하는 기능 및 프로그램이다.

- Communication Layer

  - 토픽, 서비스, 메시지 등 메시지 통신과 관련한 요소가 들어가 있다.

- Hardware Interface Layer

  - 카메라, GPS, 모터 등에 대한 하드웨어 관련 인터페이스를 갖추고 있다.

- Software Development Tools

  - 시각화, GUI, 컴파일 등 소프트웨어 Develop에 대한 툴을 구성 요소로 갖추고 있다.

- Simulation

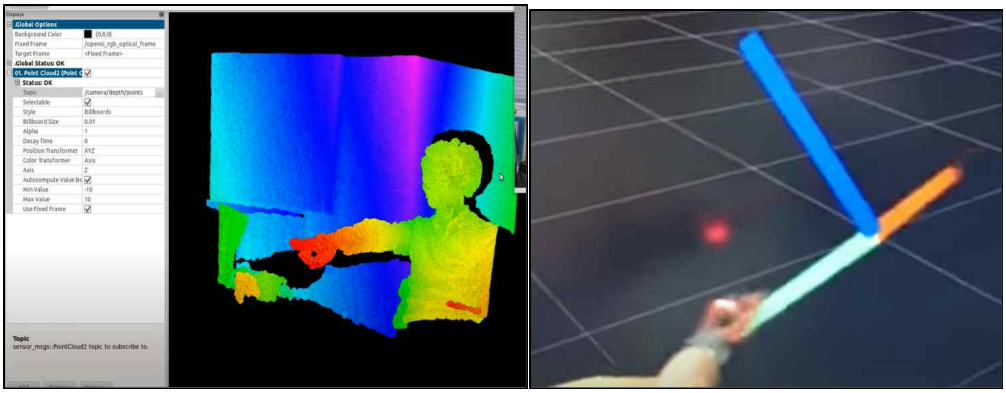
  - 2차원의 stage ros와 3차원의 gazebo를 지원하고 있다.

7. ROS의 주요 개발 도구

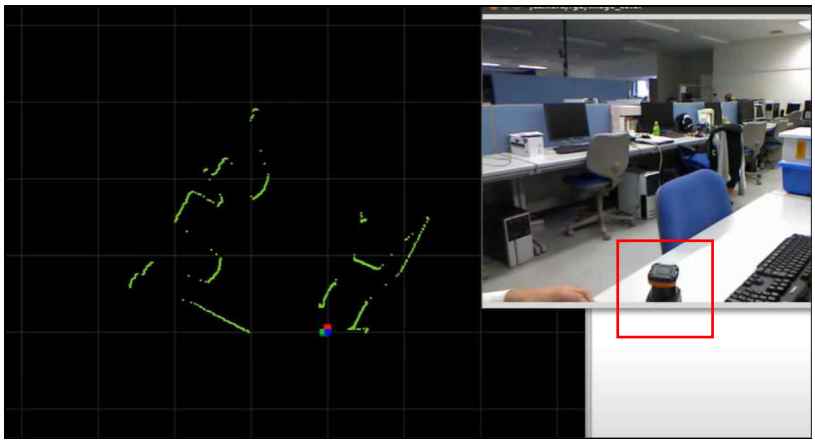
Rviz

ROS의 3차원 시각화 도구로, 센서 데이터를 시각화하거나, 로봇 외형 및 동작 표현, 내비게이션, 매니퓰레이션, 원격 제어와 같은 기능을 포함하고 있다.

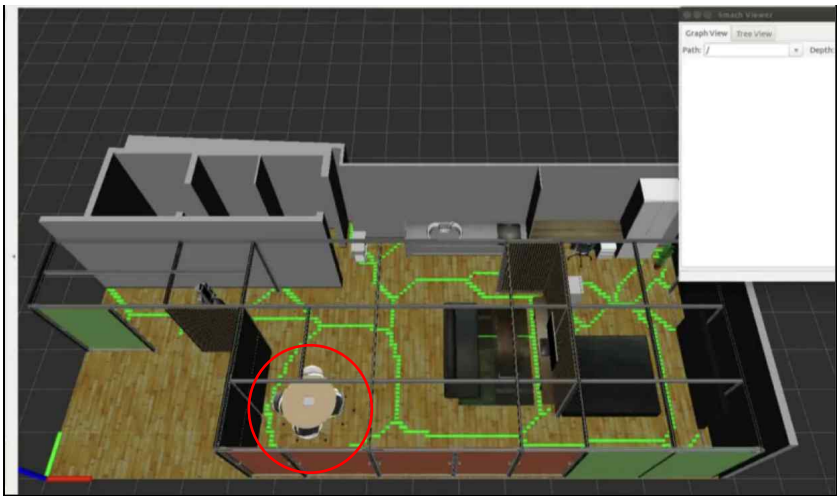
센서 데이터 시각화의 예) 레이저 거리 센서의 거리 데이터 표기, Depth 카메라의 데이터를 포인트 클라우드 형태로 시각화, 카메라의 영상 데이터나 IMU 센서의 관성 데이터 등을 시각화.



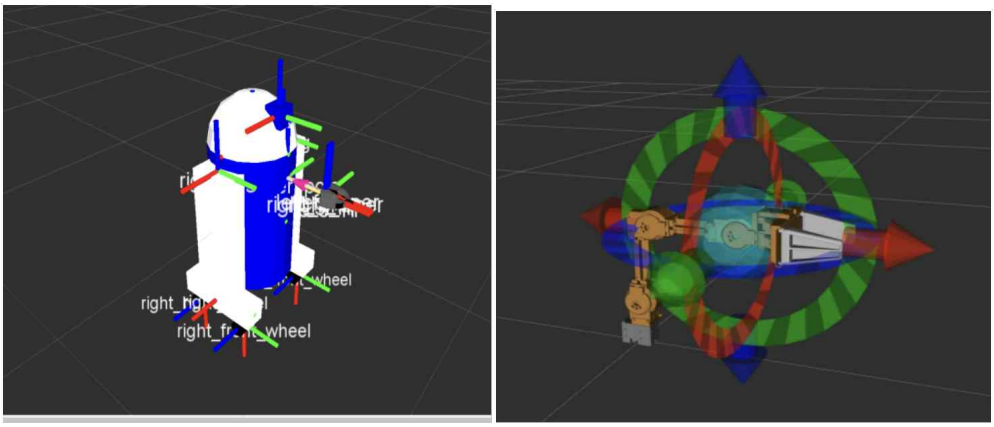
[Kinect 포인트 클라우드 데이터(좌), IMU센서의 관성 데이터(우) 시각화]



[레이저 거리 센서의 거리 값을 시각화한 화면]



[환경, 로봇, 경로를 시각화한 화면(녹색 선이 경로, 붉은 원 안의 물체가 로봇이다)]



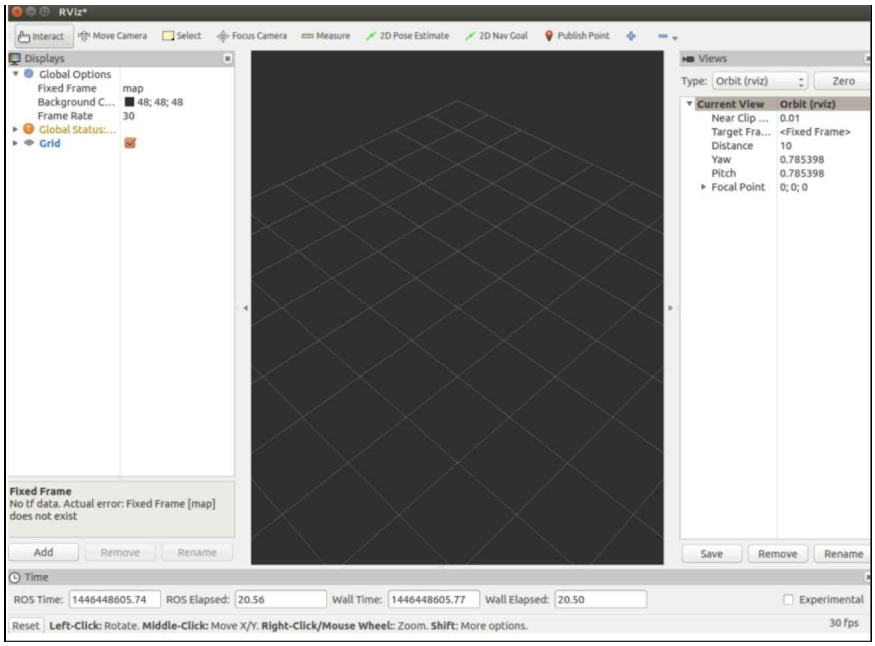
[R2-D2 로봇 모델(좌), 인터렉티브 마커를 활용한 IK 목표 위치와 경로 표시(우)]

RViz 설치 및 실행 과정

a. **설치** : $ sudo apt install ros-kinetic-rviz(ros kinetic 기준)

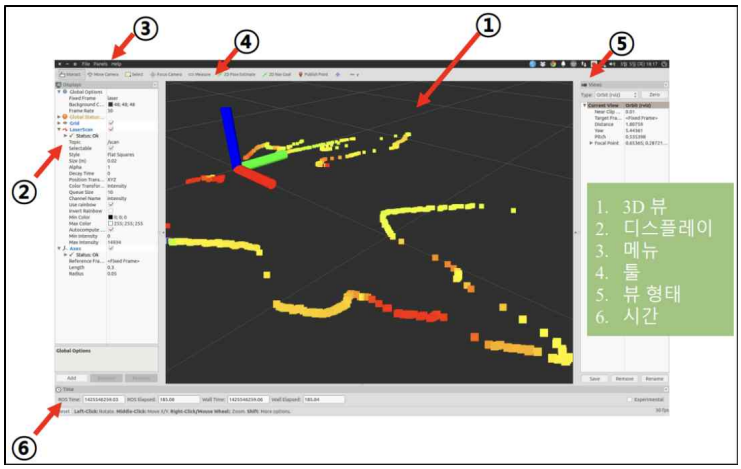
b. **실행** : $ rosrun rviz rviz 또는 $ rviz

c. **초기 화면**



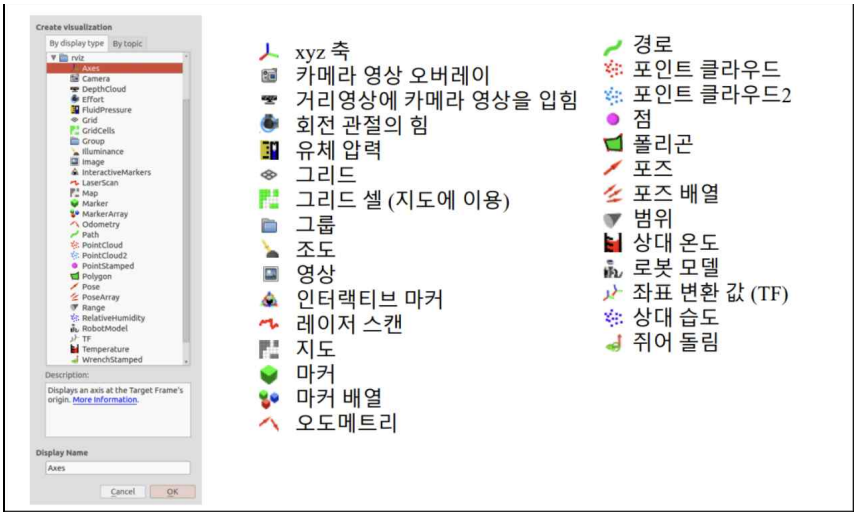
[RViz를 처음 실행했을 때의 화면(미설정)]

RViz 화면 구성



[레이저 거리 센서의 데이터를 시각화한 화면]

RViz 디스플레이 종류



[디스플레이의 ADD를 클릭하면 디스플레이 종류를 선택할 수 있다]

Rqt

rqt는 rqt_bag, rqt_plot, rqt_graph 등을 플러그인으로 하는 ROS의 종합 GUI 툴로, Qt로 개발되어 있어서 사용자들이 자유롭게 플러그인을 개발하여 추가할 수 있다. 아래는 대표적인 플러그인들이다.

- rqt_graph : 노드와 그들 사이의 연결 정보 표시

- rqt_plot : 인코더, 전압, 또는 시간이 지남에 따라 변화하는 숫자를 플로팅

- rqt_bag : 데이터를 메시지 형태로 기록하고 재생

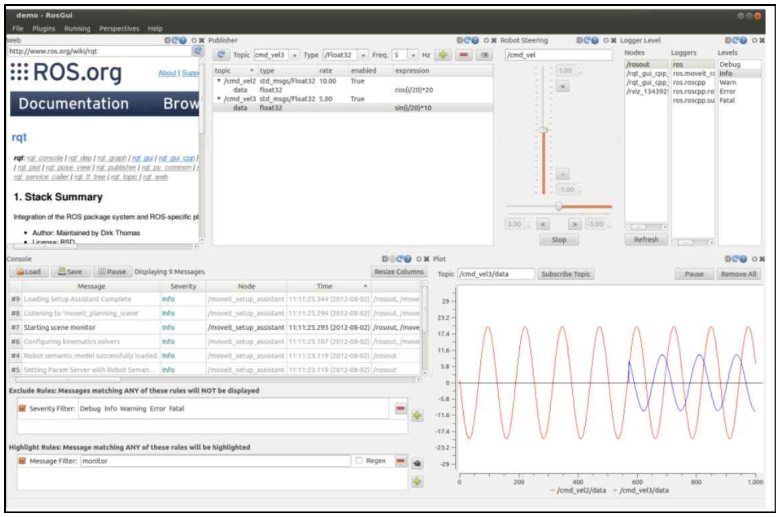
- rqt_image_view : 카메라의 영상 데이터를 확인

rqt 설치 : $ sudo apt install ros-kinetic-rqt ros-kinetic-rqt-common-plugins

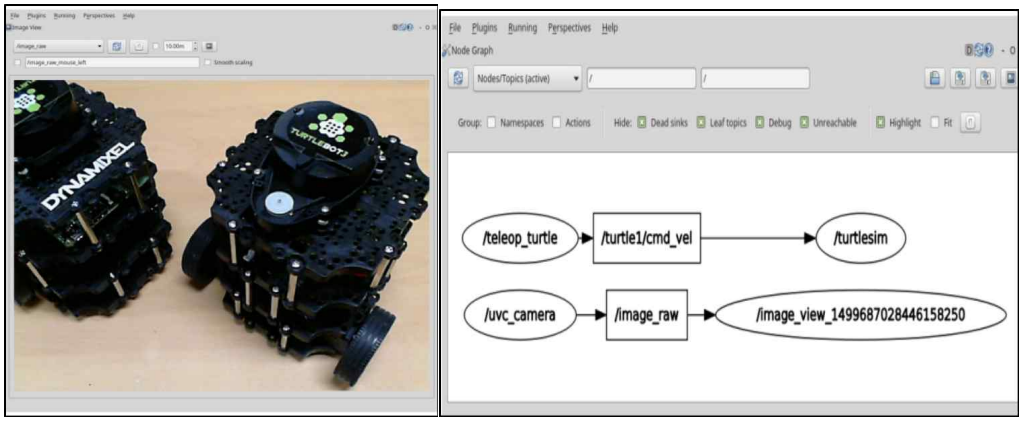
rqt 실행 : $ rqt

[rqt 초기 화면]

rqt 사용 예시



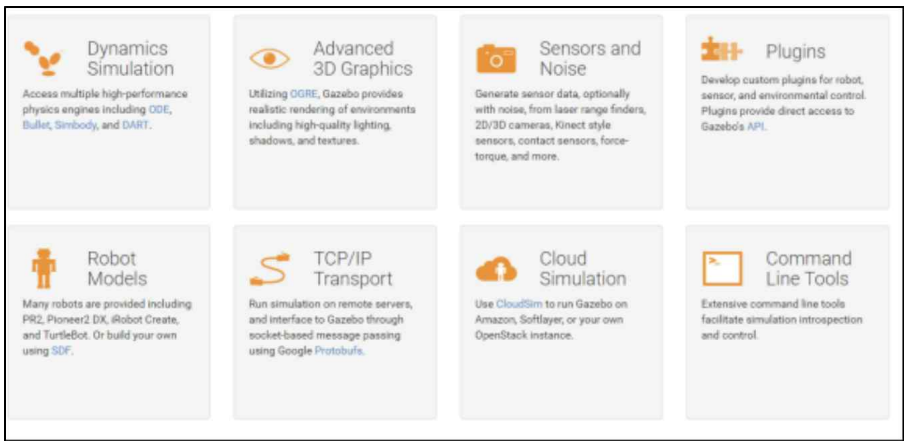
[example 1]



[example 2]

Gazebo

가제보는 로봇 개발을 위한 3차원 시뮬레이터에 대한 로봇, 센서, 환경 모델을 지원하고, 무엇보다도 물리 엔진이 탑재되어 있기 때문에 실제와 유사한 결과를 시뮬레이션할 수 있다.



[가제보의 기능들]

1. roslaunch 사용법

기본적으로 rosrun이 하나의 노드를 실행하는 명령어라면, roslaunch는 하나 이상의 정해진 노드를 실행할 수 있는 명령어이다.주 기능이 아닌, 다른 옵션을 부여하여 실행할 때에는 파라미터나 노드의 이름 변경, 환경 변수 변경, ROS_ROOT 및 ROS_PACKAGE_PATH 설정 등의 다양한 기능을 수행할 수 있다.

roslaunch는 '*.launch'라는 파일을 사용하여 실행 노드를 설정하고, 이는 XML 기반이다. 추가적으로 태그별 옵션도 제공하며, 아래와 같다.

pkg	패키지 이름
type	실행할 노드 이름 또는 실행 파일
name	노드의 실행 이름. 일반적으로는 type과 동일
args	노드에 인자를 전달
respawn	기본은 False. True로 설정 시 노드가 꺼졌을 때 자동으로 다시 실행됨
output	screen으로 설정 시 스크린(터미널)에 입출력값 출력

[param tag 종류]