

# 01. Hello World

AI ROBOT

Exported on 05/29/2021

# Table of Contents

<b>1 개발환경 .....</b>	<b>5</b>
1.1 개발환경을 맞추는 이유.....	5
<b>2 Sublime Text.....</b>	<b>6</b>
2.1 프로그램 IDE로 Sublime Text를 사용하겠습니다.....	6
2.2 설치 .....	6
2.3 설치 확인 .....	7
2.4 Package Controller 설치 .....	7
<b>3 화면 분할 가능한 터미널 .....</b>	<b>9</b>
3.1 tilix .....	9
3.2 최초 터미널에서 tilix라고 실행하면 나타나는 에러 .....	9
3.3 환경설정에서 run command as a login shell 체크.....	10
3.4 favorite에 등록 .....	10
3.5 원하는 테마 적용 .....	11
3.6 우분투 설정에서 CTRL+ALT+D키 단축키를 삭제 .....	11
3.7 tilix에서.....	12
<b>4 ROS1 melodic 실행.....</b>	<b>13</b>
4.1 ROS1 melodic .....	13
4.2 먼저 melodic을 설치하기 전에 .....	14
4.3 Melodic 설치 .....	14
4.4 melodic 설치를 위한 pkg 리스트 추가 및 key 추가 .....	14
4.5 melodic desktop full version 설치.....	15
4.6 bashrc 설정 .....	15
4.7 일단 .....	15
4.8 다시 설치 가이드 페이지에서 .....	16
4.9 catkin tools 설치.....	17
4.10 ROS1용 워크스페이스는 catkin_ws로 한다.....	17
4.11 catkin init (주의) catkin_ws에서 수행 .....	17
4.12 catkin build .....	18

4.13 빌드를 수행하고 나면 워크스페이스에 devel 폴더가 생성됨 .....	18
4.14 devel 폴더에는 setup.bash 파일이 있음 .....	18
4.15 bashrc의 ROS1 부분에 이 파일 경로를 추가.....	18
<b>5   로봇을 개발? 한다? .....</b>	<b>19</b>
5.1 기계 설계? - 자기 분야가 명쾌해 보인다 .....	19
5.2 회로 설계? - 기계에 숨겨지긴 하지만 역시 명쾌하다.....	20
5.3 전장설계? - 뭐 역시~ .....	20
5.4 기계설계/기계조립/회로설계/전장작업... 이제 코딩? .....	21
5.5 로봇 SW? 개발? - 아두이노급이라면.....	21
5.6 혹은 간단한 모터 정도 구동한다면.....	21
5.7 뭐... 그런 모터 여러개와 함께 기구학 해석도 어쩌면.....	22
5.8 알고리즘 제어기를 포함한 로봇팔 하나 구동하는건? SW를 어떻게? .....	23
5.9 시스템이 더 복잡해지면??? .....	24
5.10 이쯤 되면 대책이 필요하다~.....	24
5.11 개발계획은 언제나 아름답다~ .....	25
5.12 현실은 항상 XXX.....	25
5.13 ROS) SW 및 알고리즘 개발을 기계/하드웨어의 이슈와 독립적으로 개발 검증할 수는 없을 까? .....	26
5.14 내가 혼자 다 다루는데는 한계가 있다~ .....	26
5.15 그 와중에 하고 싶은 것도 많다.....	27
5.16 딥러닝+영상인식+SLAM+자율주행 wow~ .....	27
5.17 이걸 다 언제하나? .....	28
5.18 ROS) 이럴때 대부분 맡겨두고 난 집중하고 싶은 것에 집중할 수 있다면? .....	29
5.19 만약 제어 알고리즘 엔지니어라면 .....	29
5.20 실제 ROS는 로봇을 운용해야 할 경우 활용도가 높다 .....	30
<b>6   Robot Operating System .....</b>	<b>31</b>
6.1 ROS는 2000년 그 유명한 앤드류昂 교수의 랩에서 시작했다 .....	31
6.2 ROS의 구성 .....	31
6.3 주행로봇에서의 Node와 Topic의 예.....	32
6.4 ROS2의 특징 .....	32
6.5 ROS1과 ROS2의 빌드툴.....	33

6.6 ROS 사용예 - 우아한 형제들 .....	33
6.7 AWS RoboMaker에서 개발환경 구성 .....	34
6.8 배민 배달 로봇 .....	34
6.9 ROS는 System Architecture에 대한 부담을 덜어준다 .....	35
6.10 RViz .....	36
6.11 Inverse Kinematics .....	36
6.12 물리엔진이 탑재된 Gazebo .....	37
6.13 이 모든 기능이 나를 위해 존재한다 .....	37
<b>7 ROS History .....</b>	<b>38</b>
7.1 ROS1 History .....	38
7.2 ROS1 .....	38
7.3 ROS2 History .....	39
<b>8 이제 배우고 즐길 때이다 .....</b>	<b>40</b>

# 1 개발환경

## 1.1 개발환경을 맞추는 이유

- 다들 선호하는 다양한 툴들이 있을 겁니다.
- 요즘 나타나는 다양한 툴들의 강력하고 유용한 기능은 정말 멋집니다
- 그러나 수업때 그런 다양한 툴들을 모두 대상으로 할 수 없고
- 또 툴에 대한 질문에 대응하는 것도 어려울 수 있습니다.
- 그래서 본 수업에서는 수업의 원활한 진행을 위해 개발환경을 맞춥니다.
- 그러나 따로 선호하는 환경을 사용하셔도 상관없습니다.

## 2 Sublime Text

### 2.1 프로그램 IDE로 Sublime Text를 사용하겠습니다.

- <https://www.sublimetext.com/>
- 쉽고 가볍고 사용하기 간편합니다.

### 2.2 설치

apt

Install the GPG key:

```
wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
```

Ensure apt is set up to work with https sources:

```
sudo apt-get install apt-transport-https
```

Select the channel to use:

Stable

```
echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee  
/etc/apt/sources.list.d/sublime-text.list
```

Dev

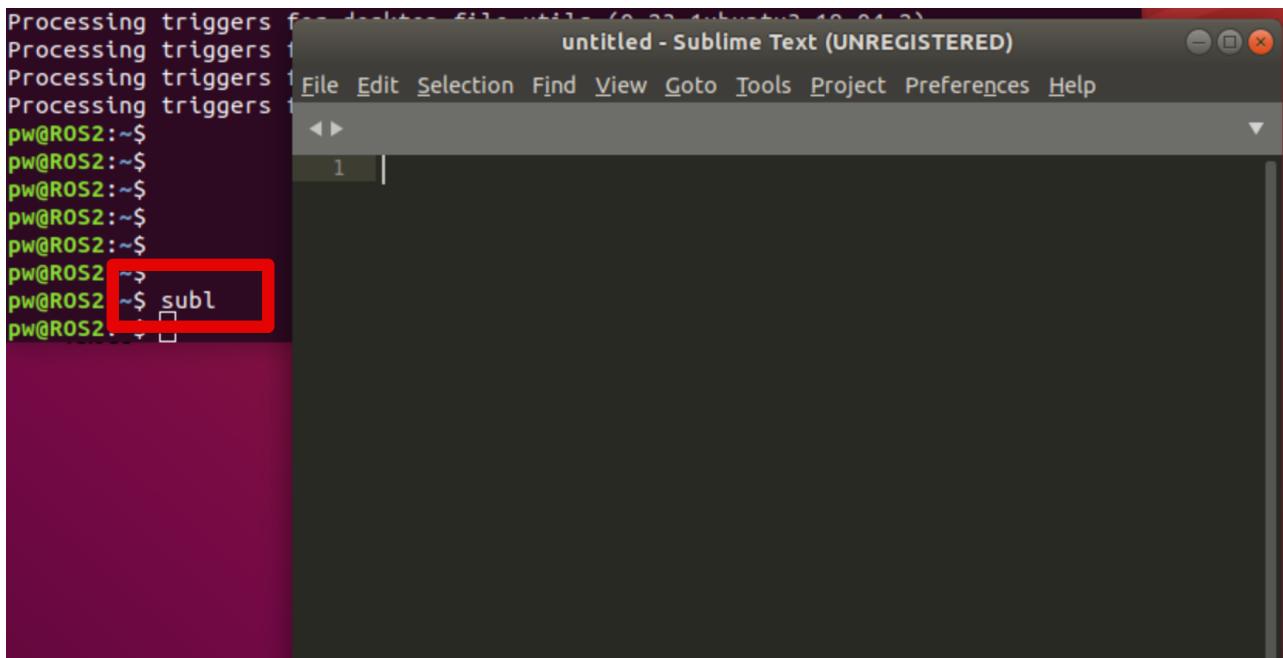
```
echo "deb https://download.sublimetext.com/ apt/dev/" | sudo tee  
/etc/apt/sources.list.d/sublime-text.list
```

Update apt sources and install Sublime Text

```
sudo apt-get update  
sudo apt-get install sublime-text
```

- [https://www.sublimetext.com/docs/3/linux\\_repositories.html](https://www.sublimetext.com/docs/3/linux_repositories.html)
- 위 사이트에서 위에 표시된 부분을 터미널로 복사해서 실행합니다.

## 2.3 설치 확인

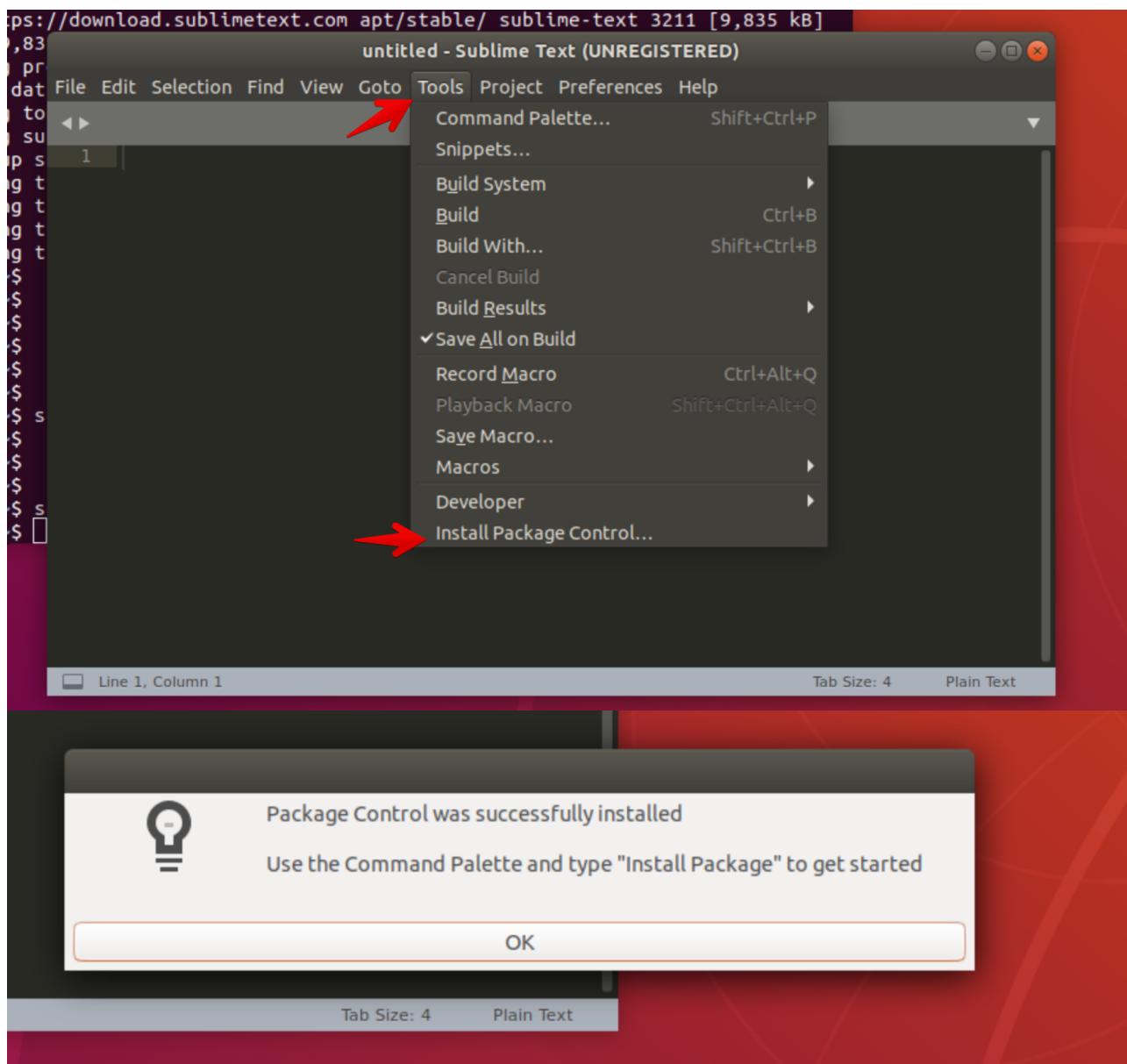


A screenshot of a terminal window titled "untitled - Sublime Text (UNREGISTERED)". The window shows a history of commands starting with "Processing triggers" and then "pw@ROS2:~\$". A red box highlights the command "subl" being typed at the prompt.

- 터미널에서 subl 실행

## 2.4 Package Controller 설치

- <https://packagecontrol.io/installation>

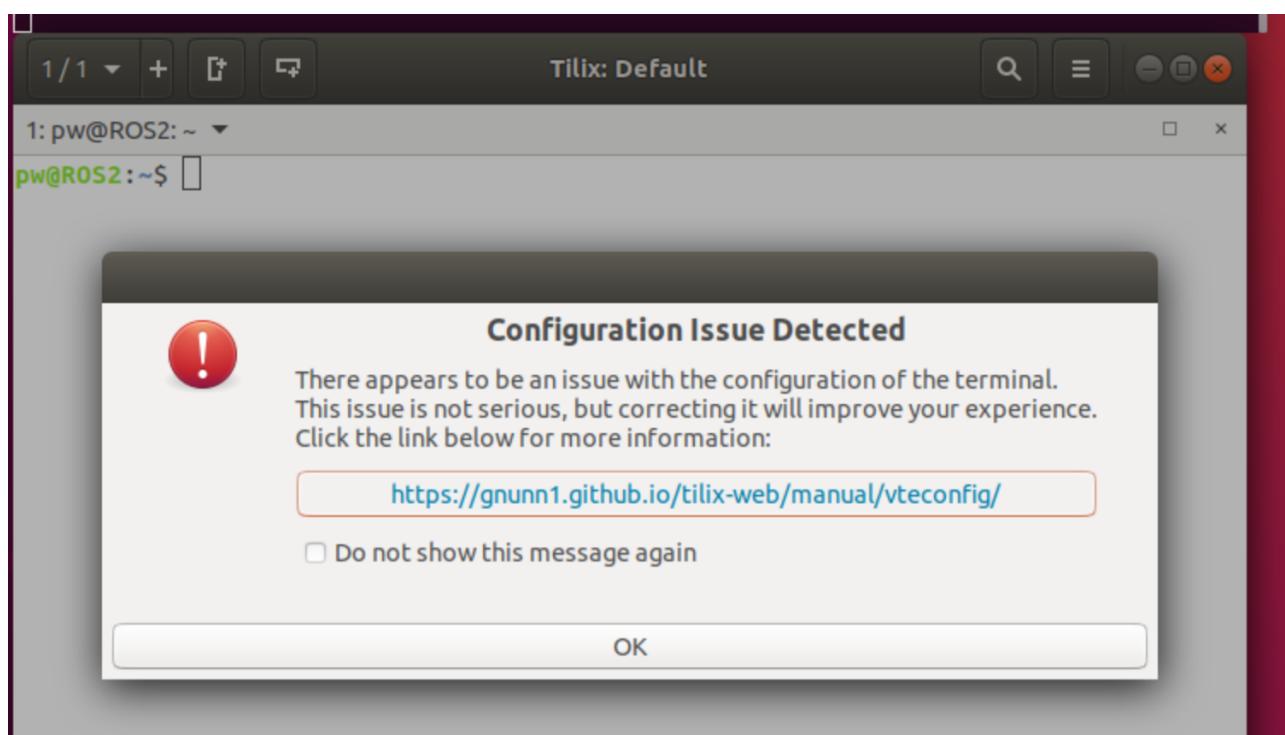


## 3 화면 분할 가능한 터미널

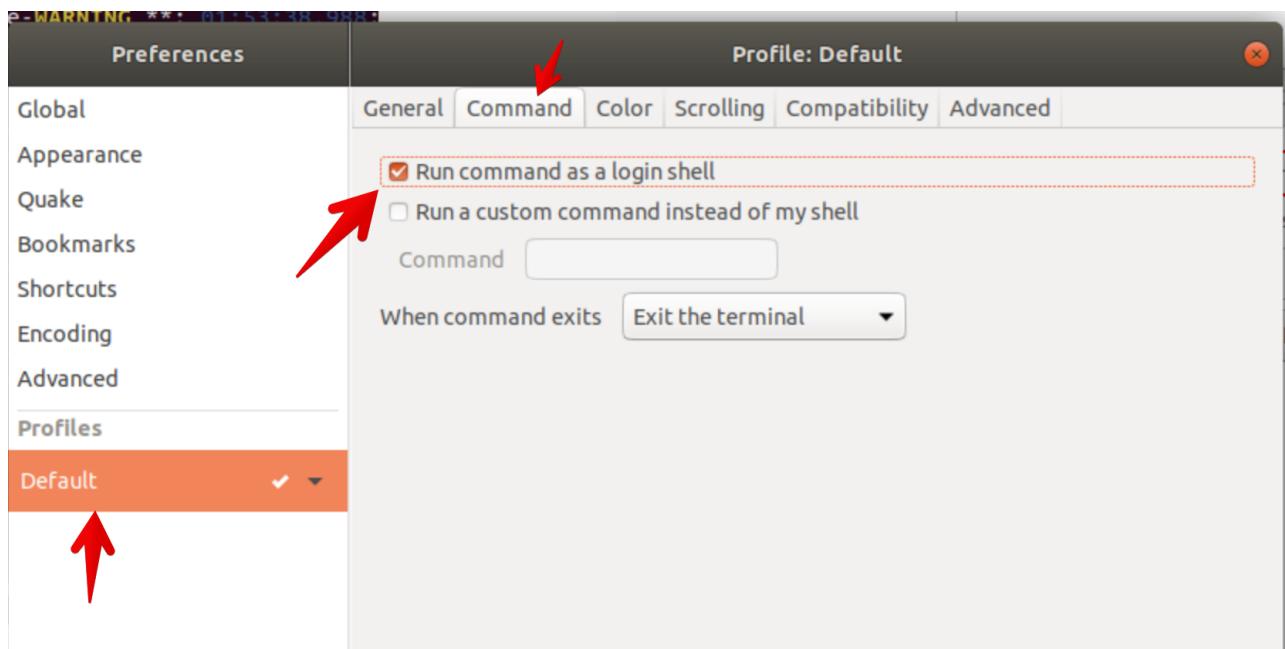
### 3.1 tilix

```
pw@KUSZ:~$ sudo apt install tilix
[sudo] password for pw:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  efibootmgr gir1.2-geocodeglib-1.0 libfwup1 libllvm8 libwayland-egl1-mesa
  ubuntu-web-launchers
Use 'sudo apt autoremove' to remove them.
• sudo apt install tilix
```

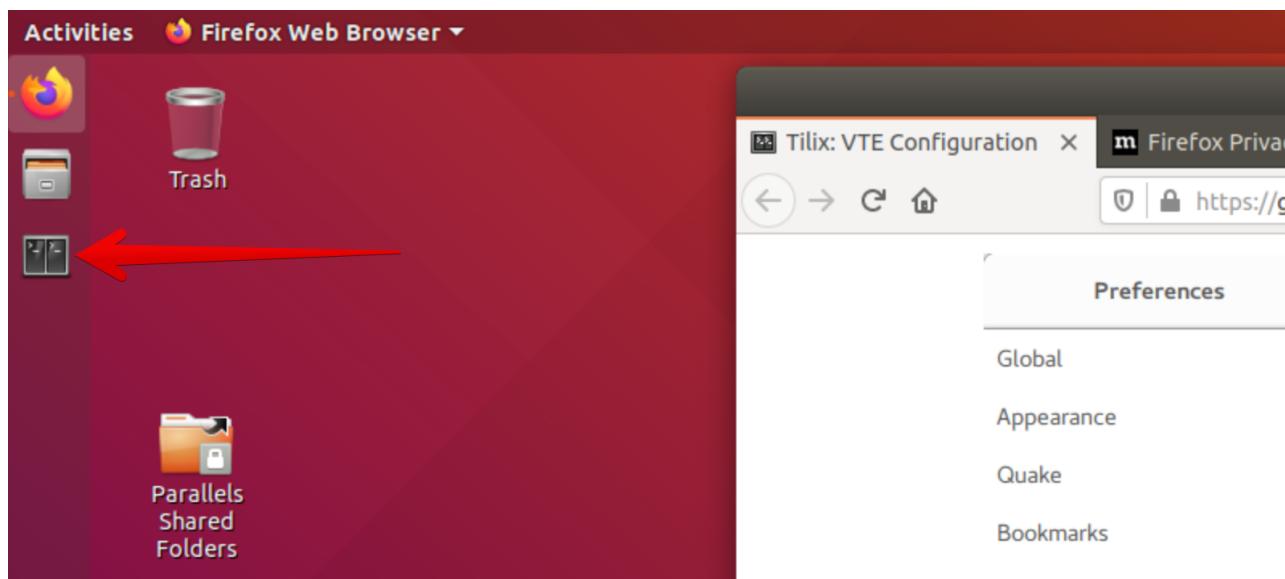
### 3.2 최초 터미널에서 tilix라고 실행하면 나타나는 에러



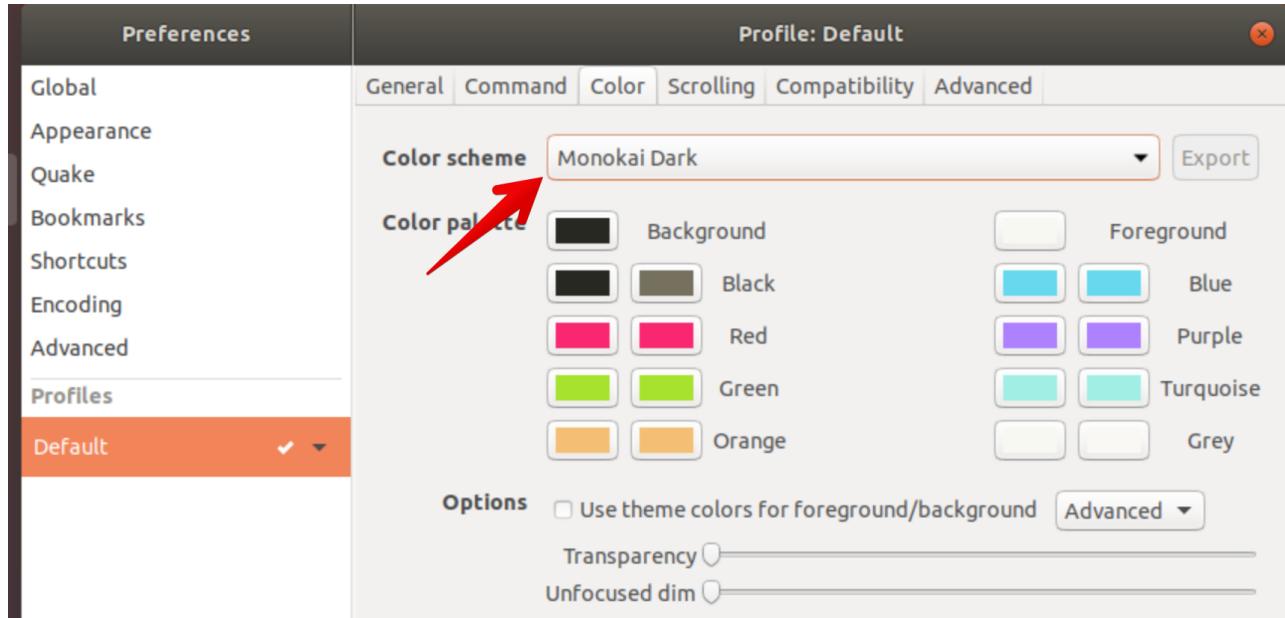
### 3.3 환경설정에서 run command as a login shell 체크



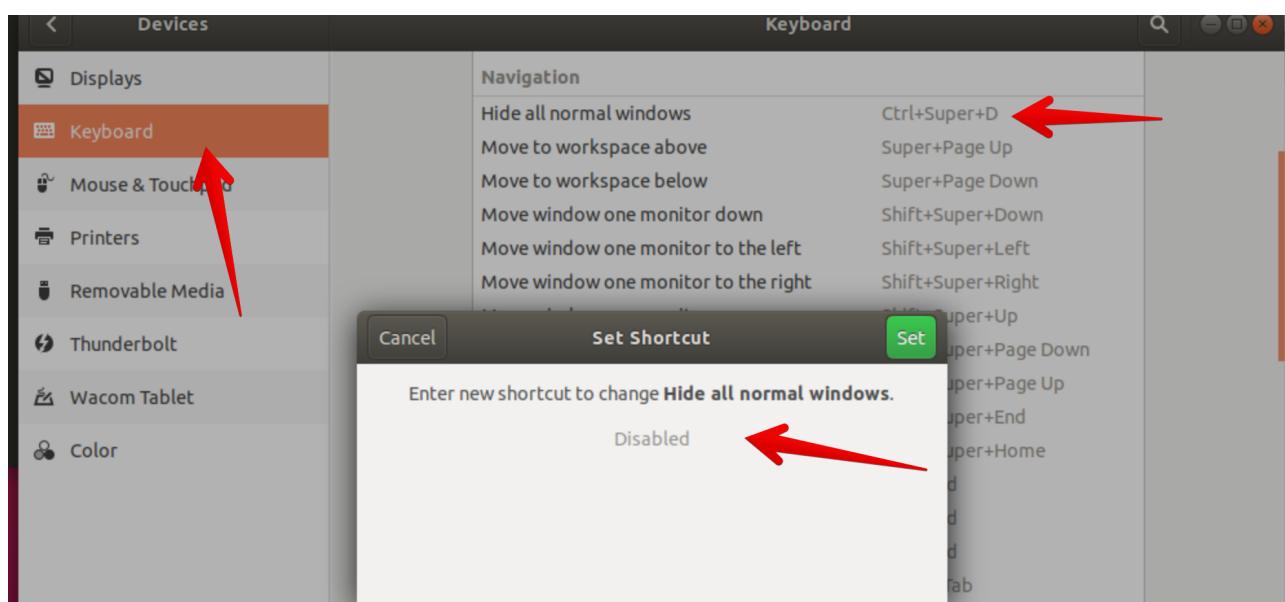
### 3.4 favorite에 등록



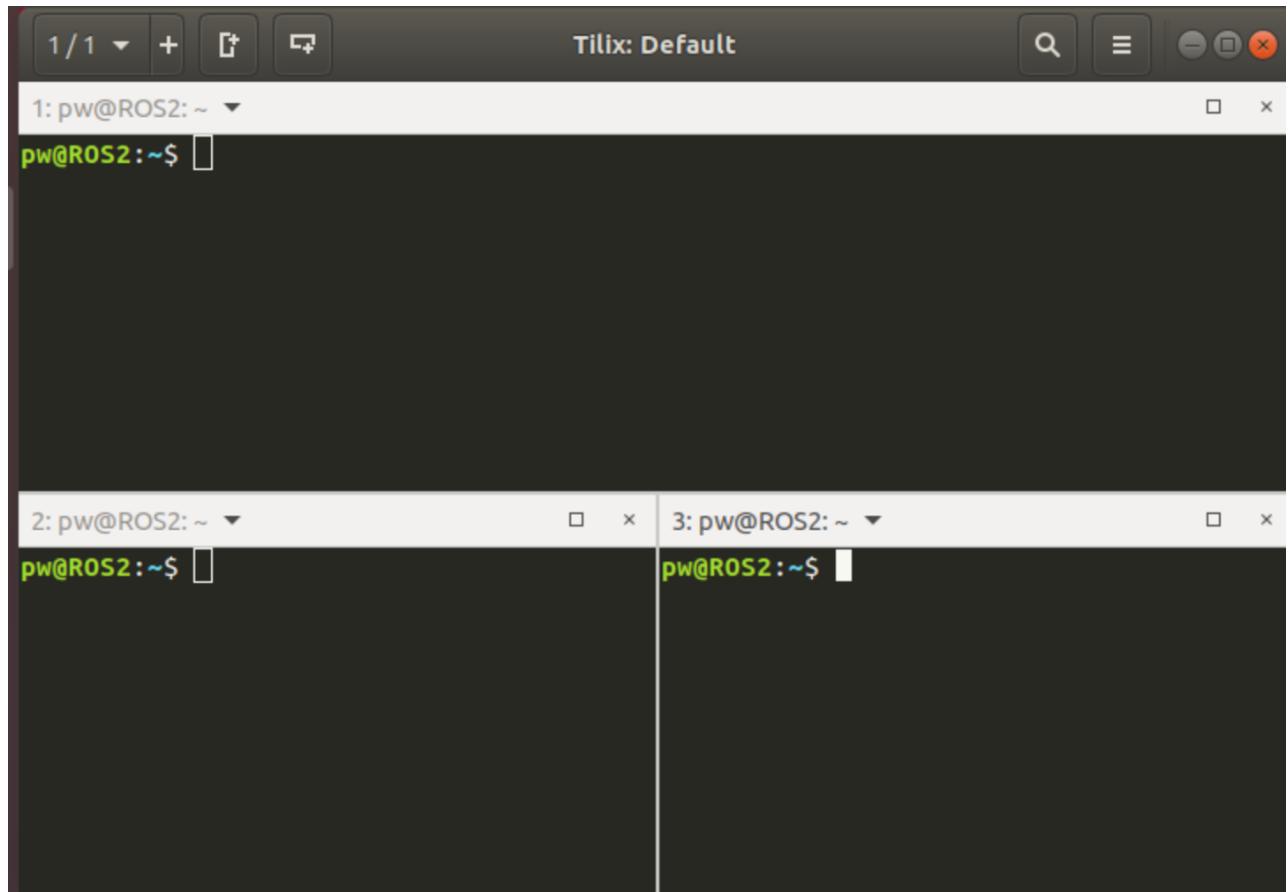
### 3.5 원하는 테마 적용



### 3.6 우분투 설정에서 CTRL+ALT+D키 단축키를 삭제



### 3.7 tilix에서



- CTRL+ALT+D는 아래쪽에, CTRL+ALT+R은 오른쪽에 화면 분할로 터미널 생성

## 4 ROS1 melodic 실행

### 4.1 ROS1 melodic



## 4.2 먼저 melodic을 설치하기 전에

```

~/.bashrc - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
.bashrc
112 if [ -f /usr/share/bash-completion/bash_completion ]; then
113     . /usr/share/bash-completion/bash_completion
114 elif [ -f /etc/bash_completion ]; then
115     . /etc/bash_completion
116 fi
117 fi
118
119 # ROS2
120 # echo "ROS2 activate"
121 # source /opt/ros/dashing/setup.bash

```

- ROS2를 설치하고 설정했던 bashrc 부분을 주석처리하고 터미널을 끄고 다시 실행한다

## 4.3 Melodic 설치

- <http://wiki.ros.org/melodic/Installation/Ubuntu>
- 위 경로에서 다음 가이드에 따라 하나씩 실행한다

## 4.4 melodic 설치를 위한 pkg 리스트 추가 및 key 추가

### 1.2 Setup your sources.list

Setup your computer to accept software from packages.ros.org.

```

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.
d/roslatest.list'

```

Mirrors Source Debs are also available

### 1.3 Set up your keys

```

sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B
172B4F42ED6FBAB17C654

```

If you experience issues connecting to the keyserver, you can try substituting hkp://pgp.mit.edu:80 or hkp://keyserver.ubuntu.com:80 in the previous command.

## 4.5 melodic desktop full version 설치

### 1.4 Installation

First, make sure your Debian package index is up-to-date:

```
sudo apt update
```



There are many different libraries and tools in ROS. We provided four default configurations to get you started. You can also install ROS packages individually.

In case of problems with the next step, you can use following repositories instead of the ones mentioned above [ros-shadow-fixed](#)

**Desktop-Full Install: (Recommended)** : ROS, [rqt](#), [rviz](#), robot-generic libraries, 2D/3D simulators and 2D/3D perception

```
sudo apt install ros-melodic-desktop-full
```



or [click here](#)

## 4.6 bashrc 설정

```
118
119 # ROS1
120 # echo "ROS1 activated"
121 # source /opt/ros/melodic/setup.bash
122
123 # ROS2
124 # echo "ROS2 activated"
125 # source /opt/ros/dashing/setup.bash
```



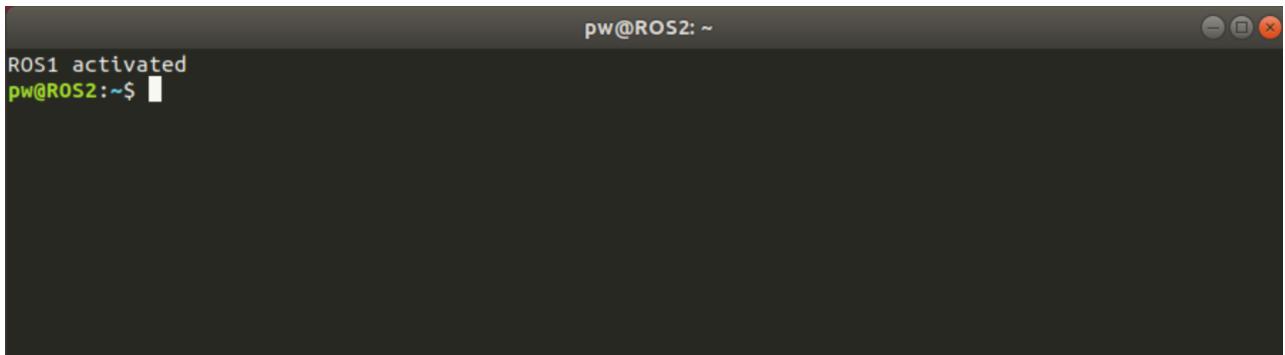
- 이렇게 설정해 두고, ROS1을 사용할때는 ROS1 부분을 주석해제, ROS2일때는 그 부분을 주석해제하는 방식으로 사용한다
- 언젠가는 ROS2만 사용하게 되겠지만, 지금은 불편해도 이렇게 진행하자

## 4.7 일단

```
118
119 # ROS1
120 echo "ROS1 activated"
121 source /opt/ros/melodic/setup.bash
122
123 # ROS2
124 # echo "ROS2 activated"
125 # source /opt/ros/dashing/setup.bash
```



- 이렇게 ROS1 설정을 활성화해서 터미널을 다시 시작하자



- 이렇게 나타나야 한다

## 4.8 다시 설치 가이드 페이지에서

### 1.6 Dependencies for building packages

Up to now you have installed what you need to run the core ROS packages. To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, `rosinstall` is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command.

To install this tool and other dependencies for building ROS packages, run:

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

#### 1.6.1 Initialize rosdep

Before you can use many ROS tools, you will need to initialize `rosdep`. `rosdep` enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS. If you have not yet installed `rosdep`, do so as follows.

```
sudo apt install python-rosdep
```

With the following, you can initialize `rosdep`.

```
sudo rosdep init  
rosdep update
```

- `rosdep` 까지 설치하고 초기화한다

## 4.9 catkin tools 설치

```
pw@ROS2:~$ sudo apt-get install python-catkin-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  efibootmgr gir1.2-geocodeglib-1.0 libfwup1 libllvm8 ubuntu-web-launchers
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python-osrf-pycommon
The following NEW packages will be installed:
  python-catkin-tools python-osrf-pycommon
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
```

- sudo apt-get install python-catkin-tools
- ROS1에서는 빌드툴로 catkin tools를 사용하도록 한다

## 4.10 ROS1용 워크스페이스는 catkin\_ws로 한다

```
pw@ROS2:~$ mkdir -p catkin_ws/src
pw@ROS2:~$ cd catkin_ws/
pw@ROS2:~/catkin_ws$
```

## 4.11 catkin init (주의) catkin\_ws에서 수행

```
pw@ROS2:~/catkin_ws$ catkin init
Initializing catkin workspace in `/home/pw/catkin_ws`.

-----
Profile:           default
Extending:        [env] /opt/ros/melodic
Workspace:        /home/pw/catkin_ws

-----
Build Space:      [missing] /home/pw/catkin_ws/build
Devel Space:      [missing] /home/pw/catkin_ws/devel
Install Space:    [unused] /home/pw/catkin_ws/install
Log Space:        [missing] /home/pw/catkin_ws/logs
Source Space:     [exists] /home/pw/catkin_ws/src
DESTDIR:          [unused] None
```

## 4.12 catkin build

```
pw@ROS2:~/catkin_ws$ catkin build
-----
Profile:           default
Extending:        [env] /opt/ros/melodic
Workspace:        /home/pw/catkin_ws

Build Space:      [exists] /home/pw/catkin_ws/build
Devel Space:     [exists] /home/pw/catkin_ws/devel
Install Space:   [unused] /home/pw/catkin_ws/install
Log Space:        [missing] /home/pw/catkin_ws/logs
Source Space:    [exists] /home/pw/catkin_ws/src
DESTDIR:         [unused] None
-----
Devel Space Layout: linked
Install Space Layout: None
```

## 4.13 빌드를 수행하고 나면 워크스페이스에 devel 폴더가 생성됨

```
pw@ROS2:~/catkin_ws$ ls
build  devel  logs  src
pw@ROS2:~/catkin_ws$
```



## 4.14 devel 폴더에는 setup.bash 파일이 있음

```
pw@ROS2:~/catkin_ws$ ls ./devel/
cmake.lock  etc  local_setup.bash  local_setup.zsh  setup.sh      setup.zsh
env.sh      lib  local_setup.sh    setup.bash       _setup_util.py  share
pw@ROS2:~/catkin_ws$
```

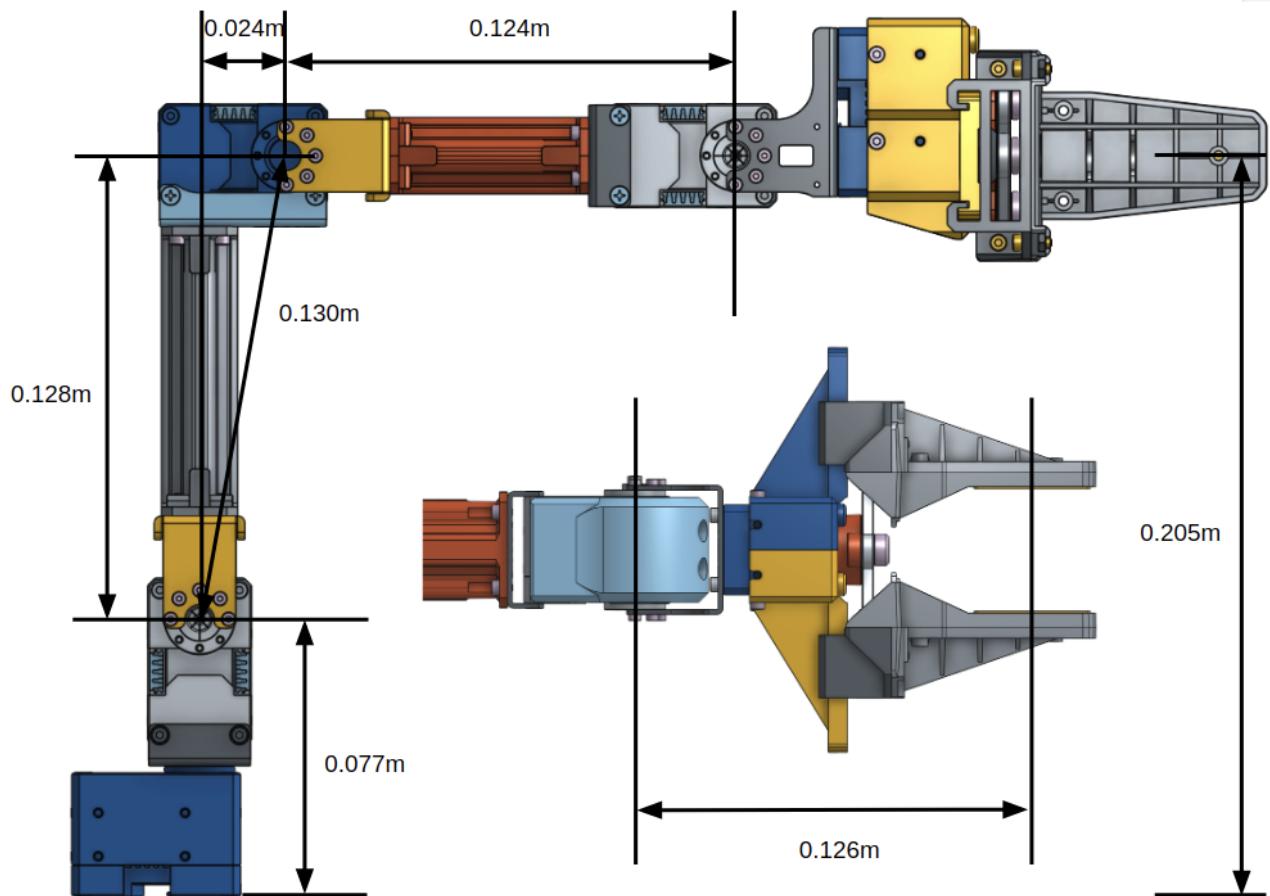
## 4.15 bashrc의 ROS1 부분에 이 파일 경로를 추가

```
115     . /etc/bash_completion
116   fi
117 fi
118
119 # ROS1
120 echo "ROS1 activated"
121 source /opt/ros/melodic/setup.bash
122 source ~/catkin_ws/devel/setup.bash
123
124 # ROS2
125 # echo "ROS2 activated"
126 # source /opt/ros/dashing/setup.bash|
```

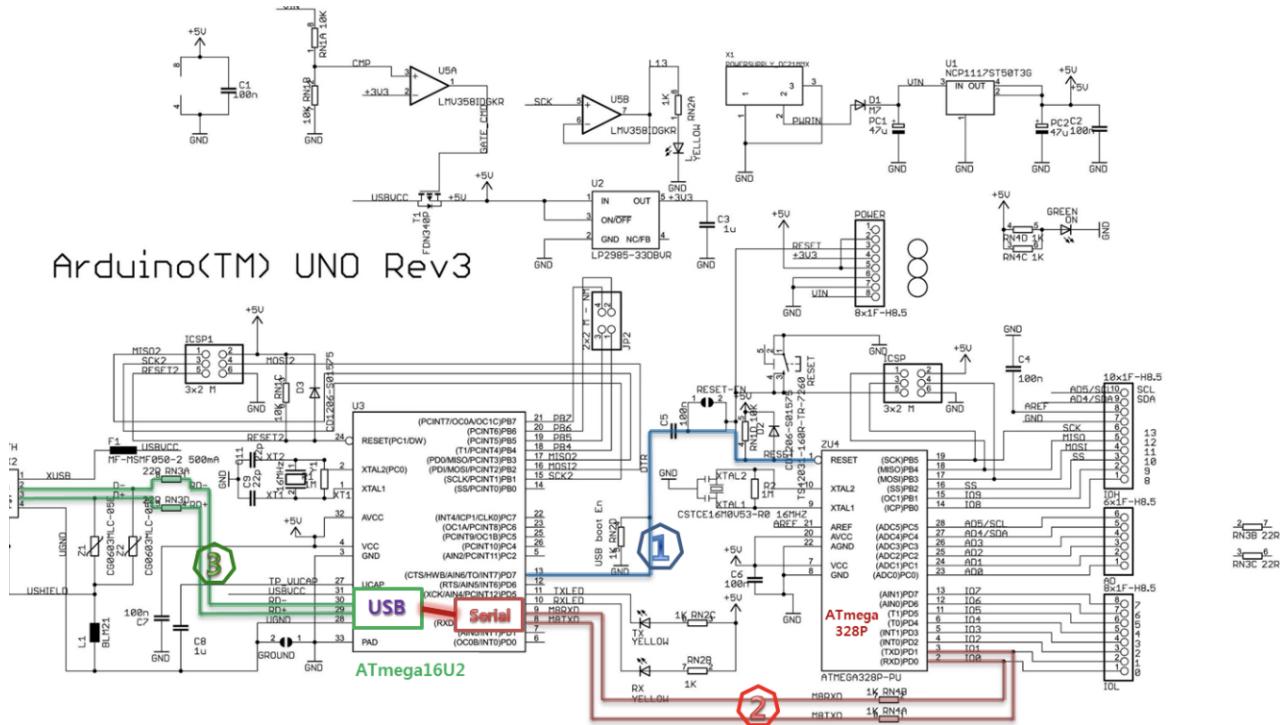


## 5 로봇을 개발? 한다?

### 5.1 기계 설계? - 자기 분야가 명쾌해 보인다



## 5.2 회로 설계? - 기계에 숨겨지긴 하지만 역시 명쾌하다

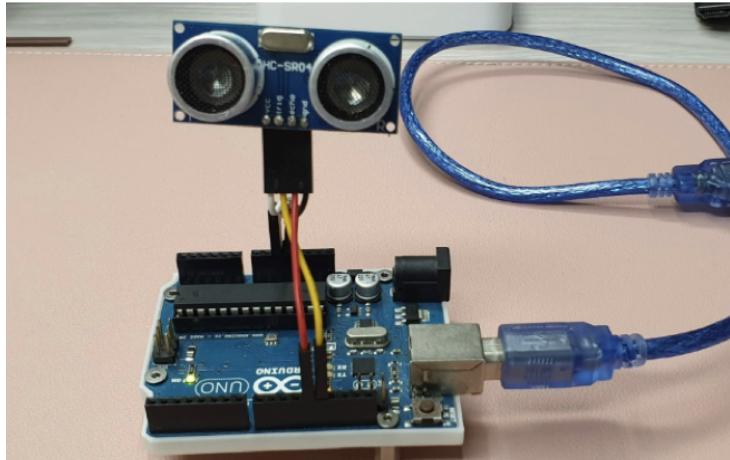


## 5.3 전장설계? - 뭐 역사~



## 5.4 기계설계/기계조립/회로설계/전장작업... 이제 코딩?

## 5.5 로봇 SW? 개발? - 아두이노급이라면...



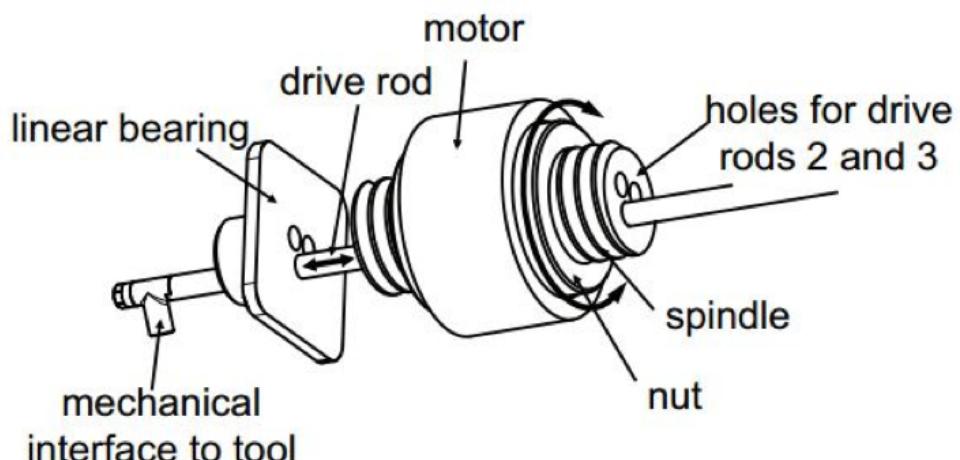
```

1#define trigPin 13
2#define echoPin 12
3
4void setup() {
5    // put your setup code here, to run once:
6    Serial.begin (115200);
7    pinMode(trigPin, OUTPUT);
8    pinMode(echoPin, INPUT);
9}
10
11void loop() {
12    // put your main code here, to run repeatedly:
13    long duration, distance;
14    digitalWrite(trigPin, LOW);
15    delayMicroseconds(2);
16    digitalWrite(trigPin, HIGH);
17    delayMicroseconds(10);
18    digitalWrite(trigPin, LOW);
19    duration = pulseIn(echoPin, HIGH);
20
21    distance = duration * 17 / 1000;
22    Serial.println(distance);
23
24
25}

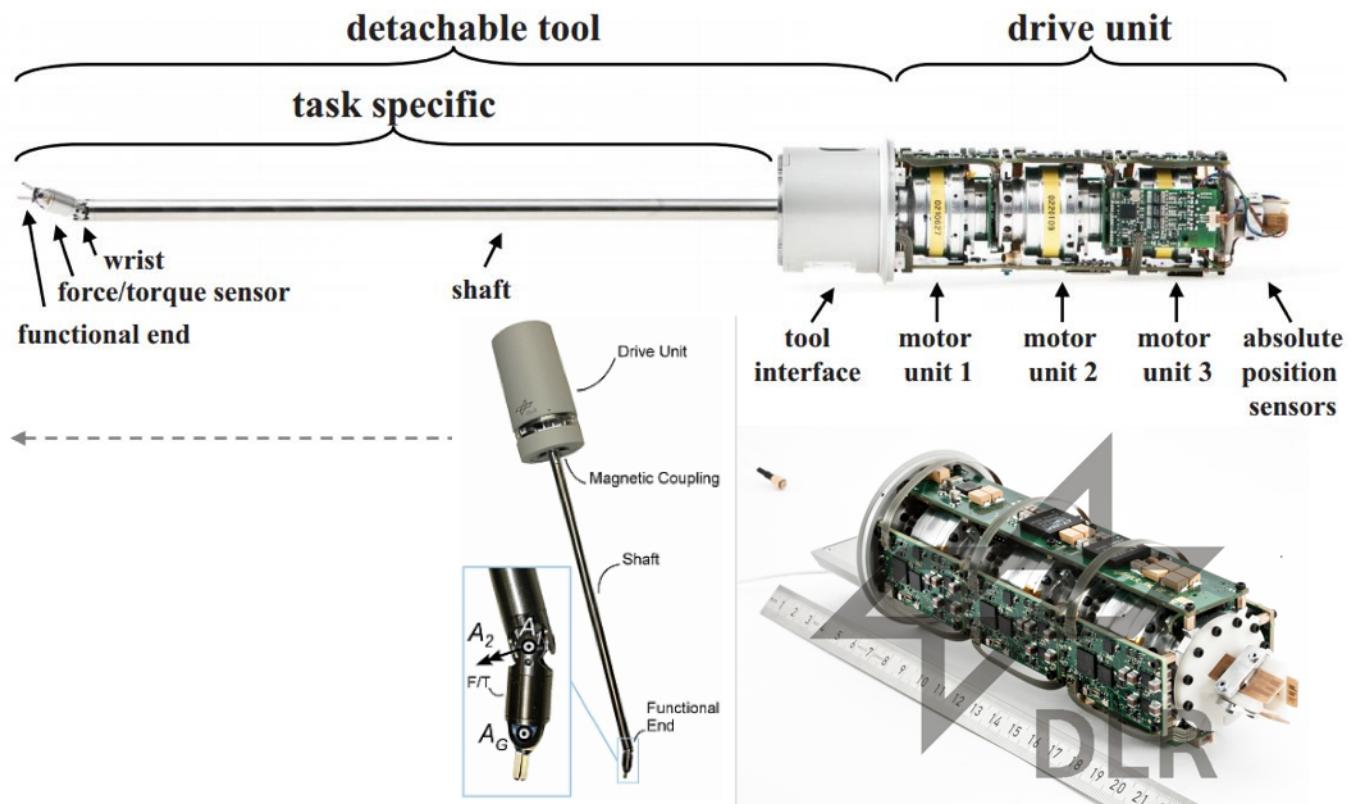
```

- 그냥 간단한 코딩으로 센서나 모터 쯤 들려 볼 수 있게 된다.

## 5.6 혹은 간단한 모터 정도 구동한다면...



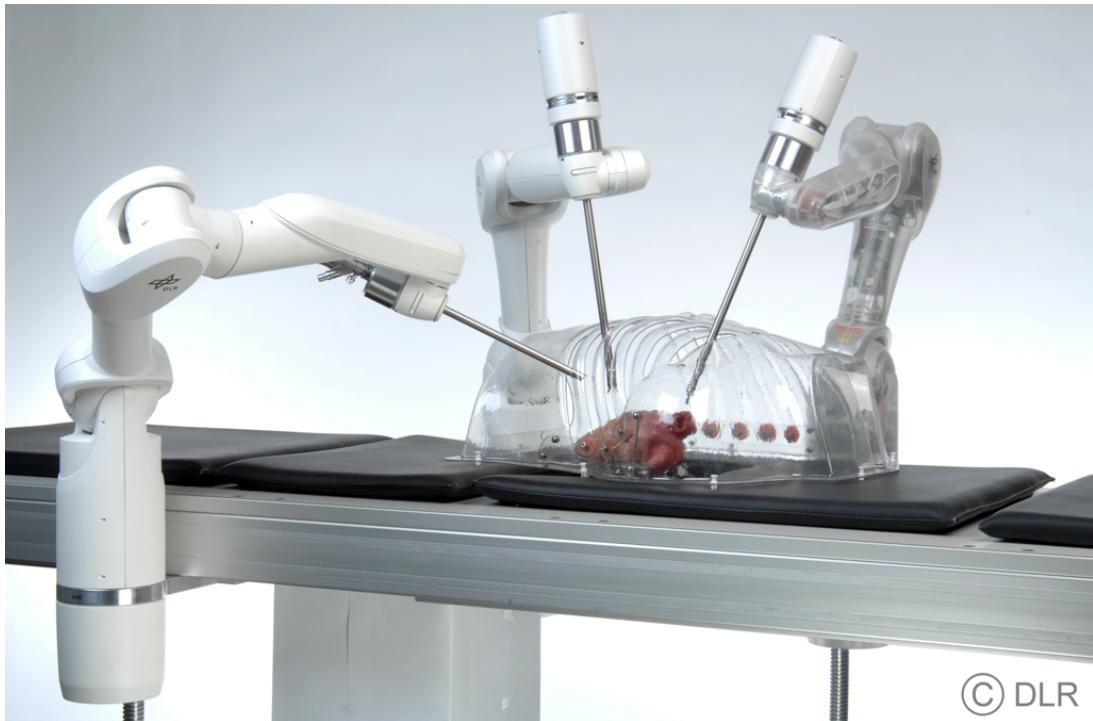
5.7 뭐... 그런 모터 여럿과 함께 기구학 해석도 어쩌면...



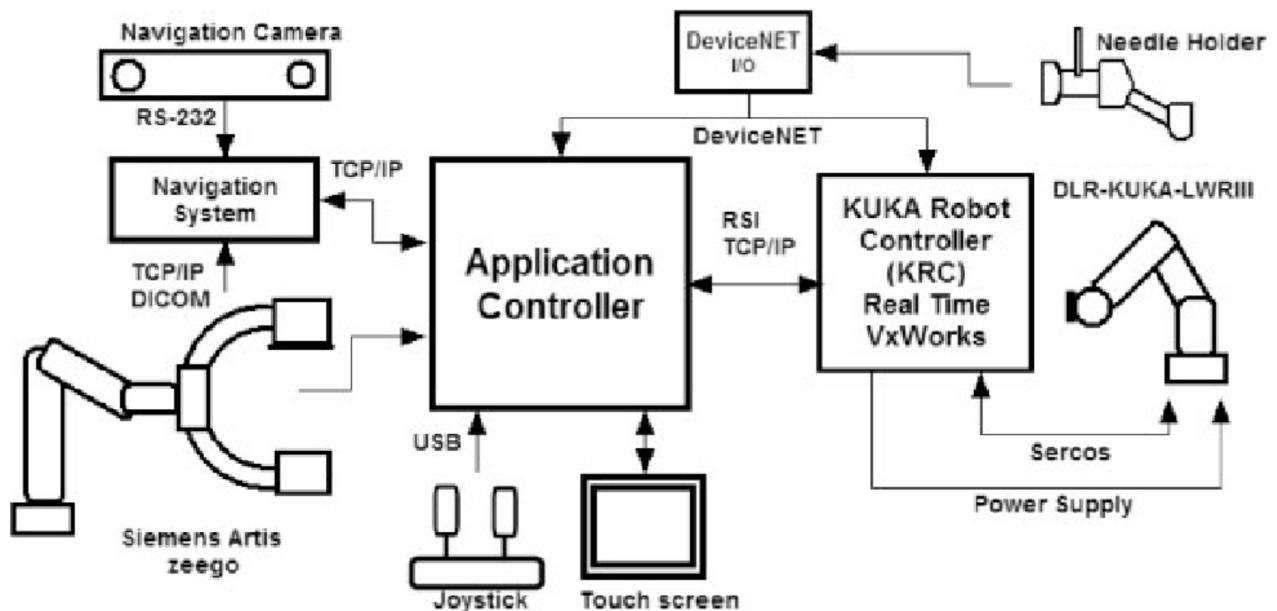
5.8 알고리즘 제어기를 포함한 로봇팔 하나 구동하는건? SW를 어떻게?



## 5.9 시스템이 더 복잡해지면???



## 5.10 이쯤 되면 대책이 필요하다~



## 5.11 개발계획은 언제나 아름답다~

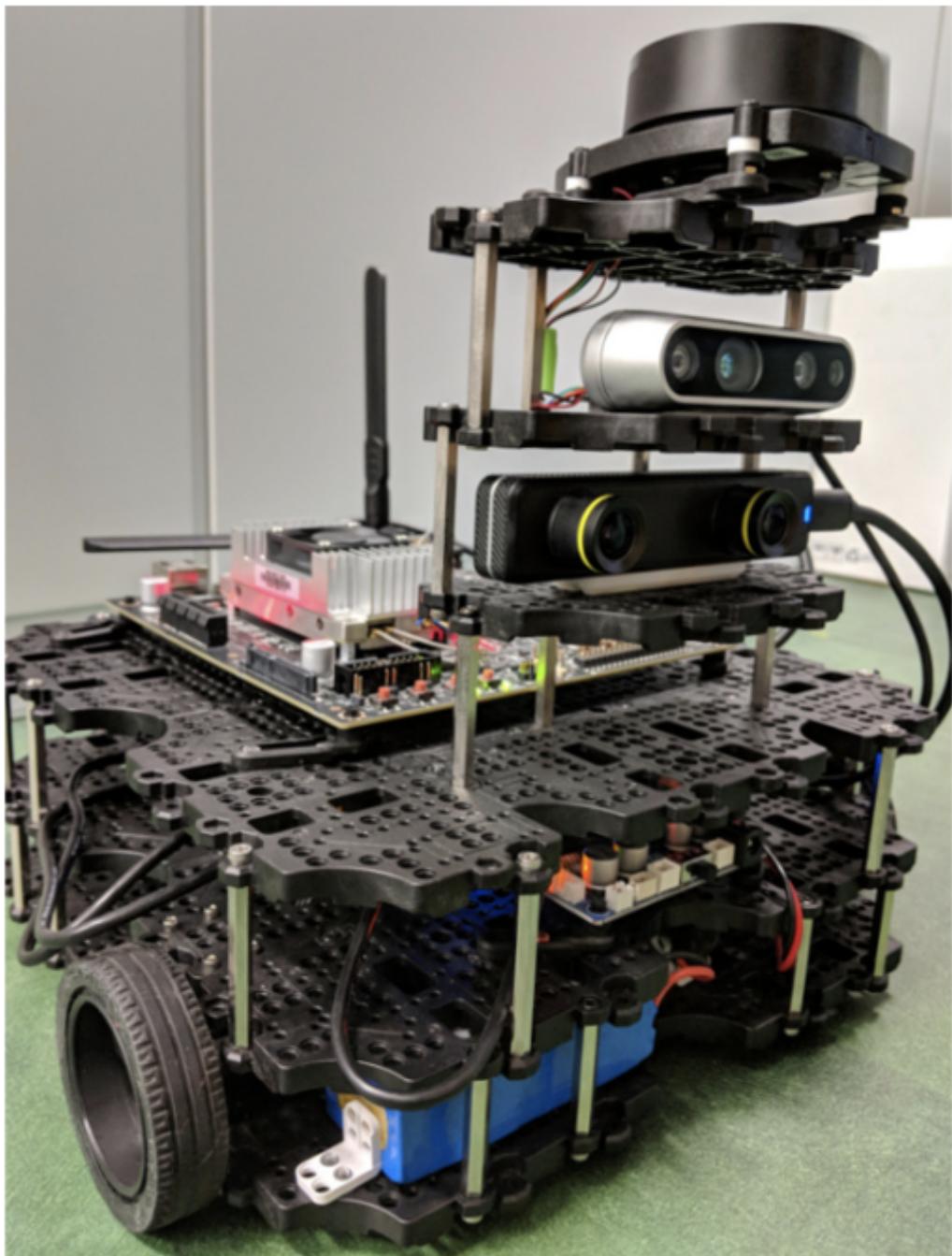


## 5.12 현실은 항상 XXX

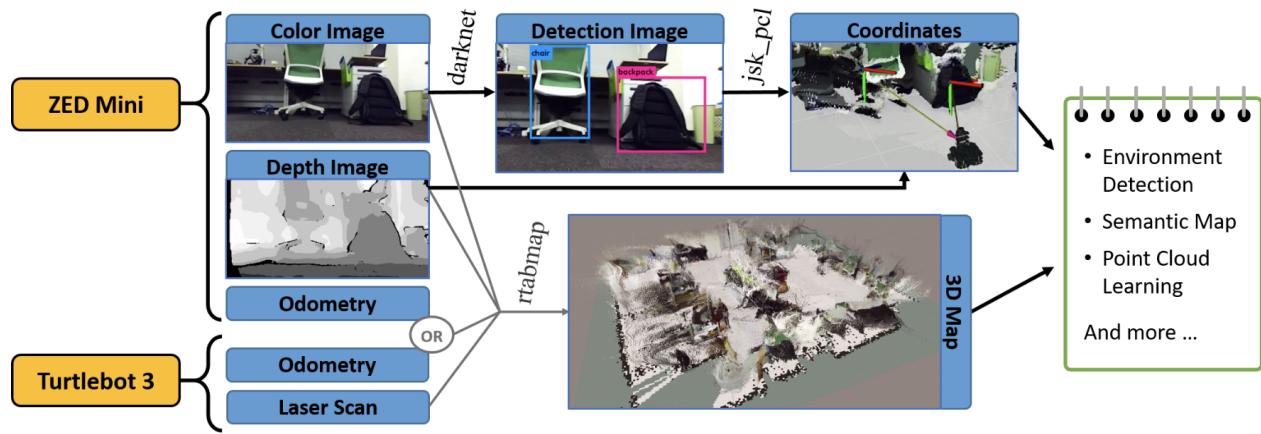


5.13 ROS) SW 및 알고리즘 개발을 기계/하드웨어의 이슈와 독립적으로 개발 검증할 수는 없을까?

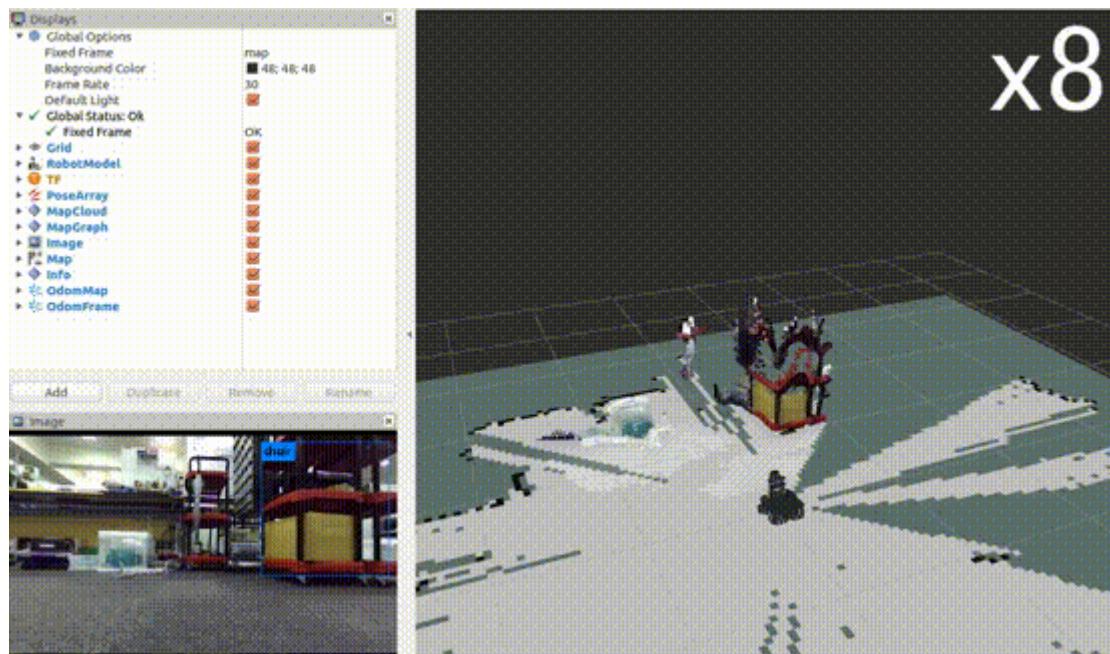
5.14 내가 혼자 다 다루는데는 한계가 있다~



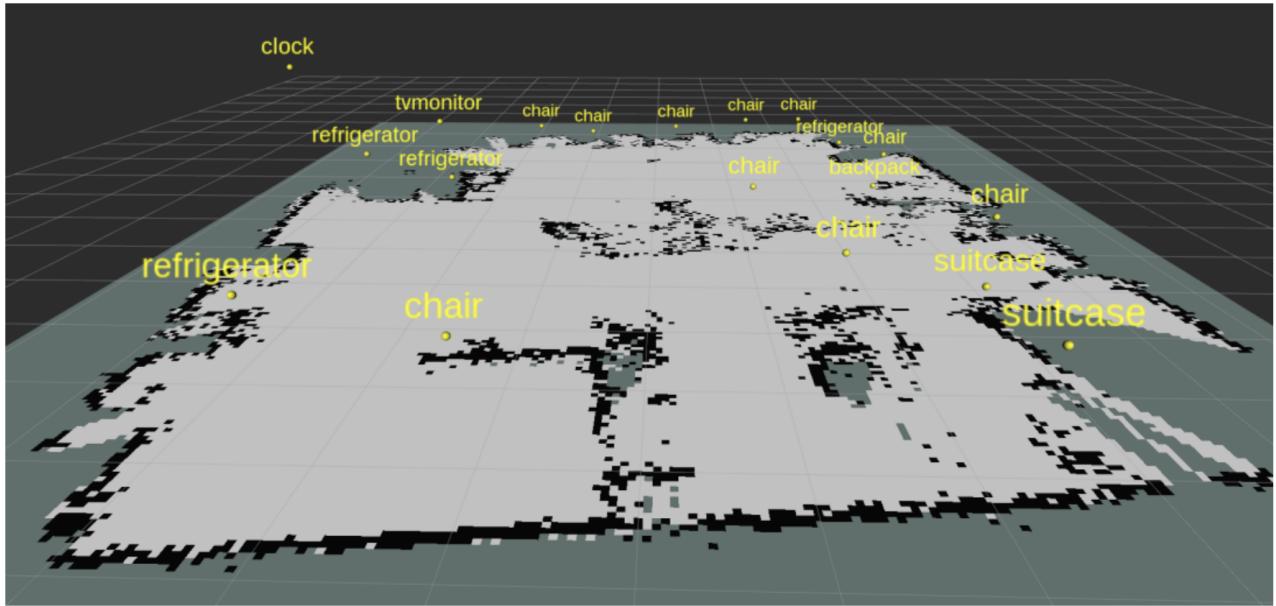
## 5.15 그 와중에 하고 싶은 것도 많다



## 5.16 딥러닝+영상인식+SLAM+자율주행 wow~

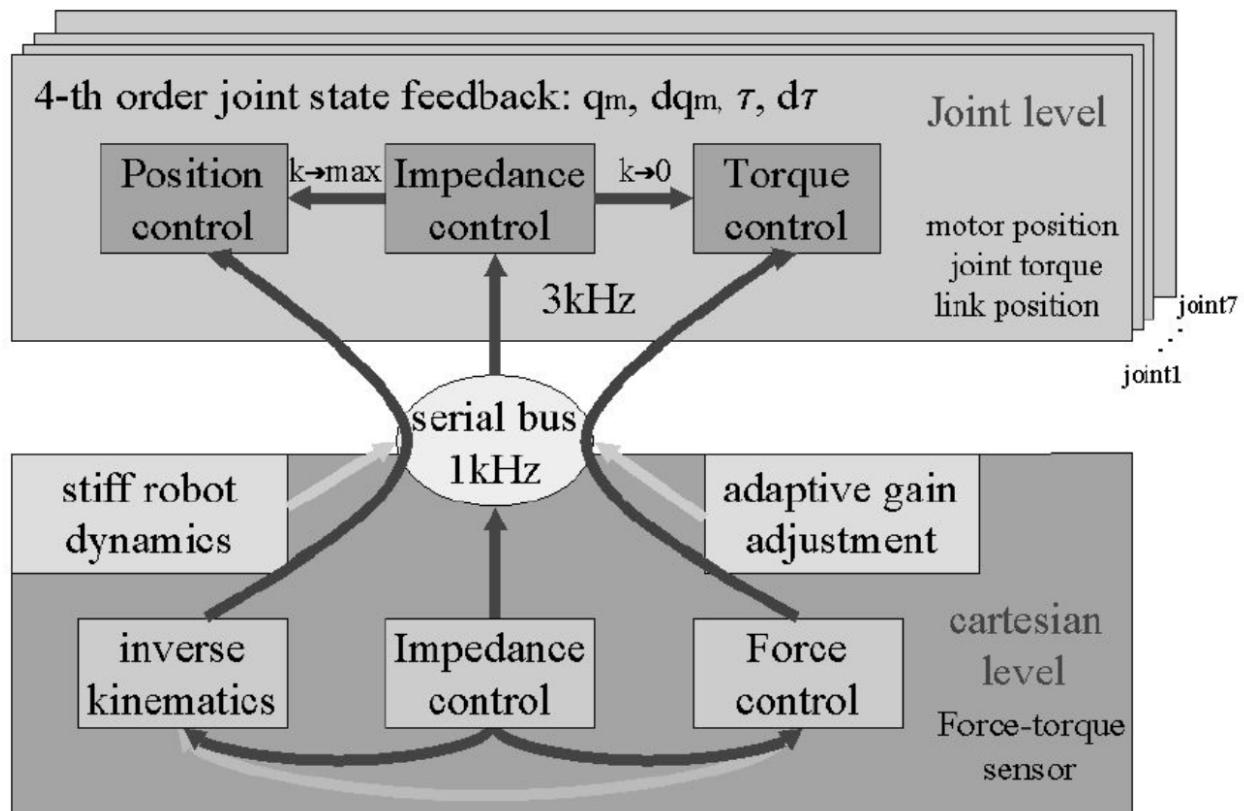


## 5.17 이걸 다 언제하나?



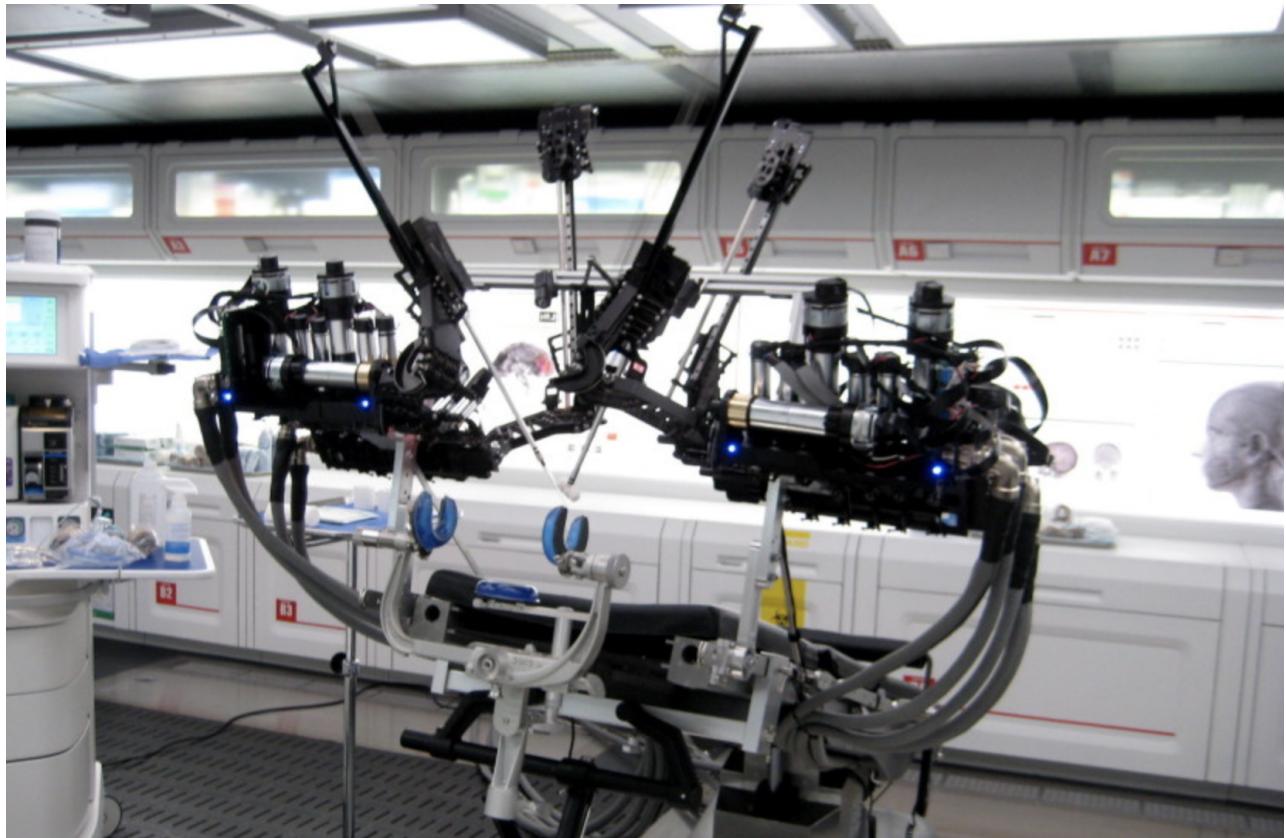
5.18 ROS) 이럴때 대부분 맡겨두고 난 집중하고 싶은 것에 집중할 수 있다면?

5.19 만약 제어 알고리즘 엔지니어라면



[https://www.youtube.com/watch?v=FRstM7b9OmQ&feature=emb\\_title](https://www.youtube.com/watch?v=FRstM7b9OmQ&feature=emb_title)

## 5.20 실제 ROS는 로봇을 운용해야 할 경우 활용도가 높다



- Surgical Robot RAVEN
- <https://www.youtube.com/watch?v=nPk90YCpqFg>
- <https://www.youtube.com/watch?v=Rg2xIu38PLg>

## 6 Robot Operating System

6.1 ROS는 2000년 그 유명한 앤드류昂 교수의 랩에서 시작했다

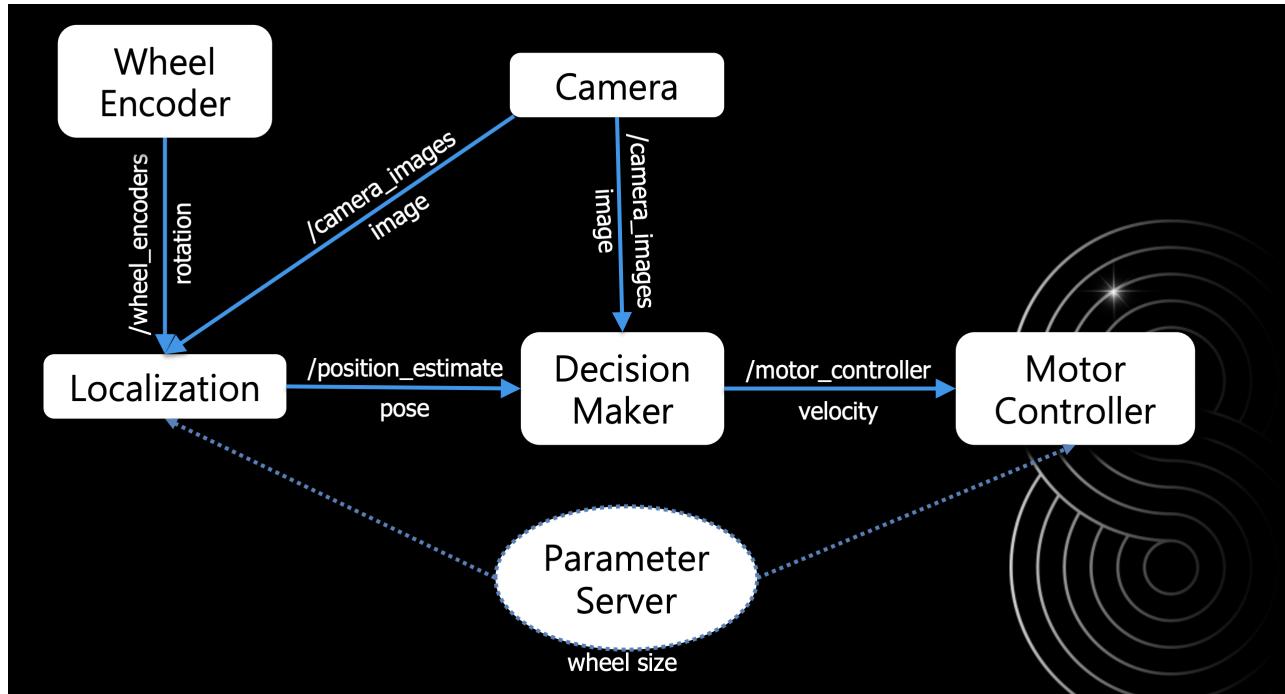


- 그 후 구글의 관심을 받았다

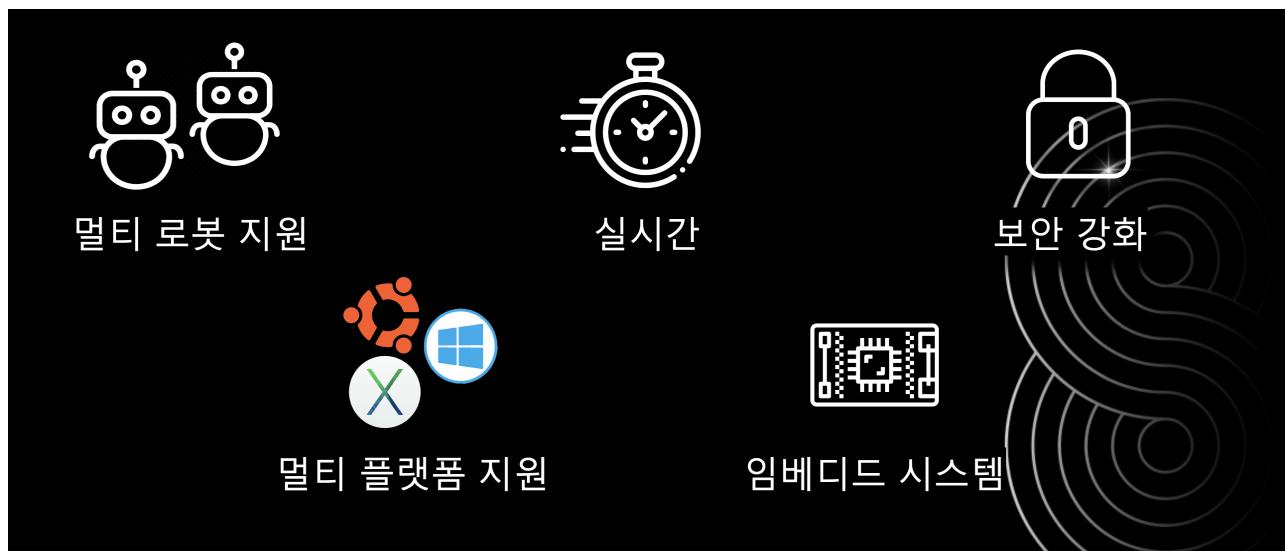
## 6.2 ROS의 구성



### 6.3 주행로봇에서의 Node와 Topic의 예



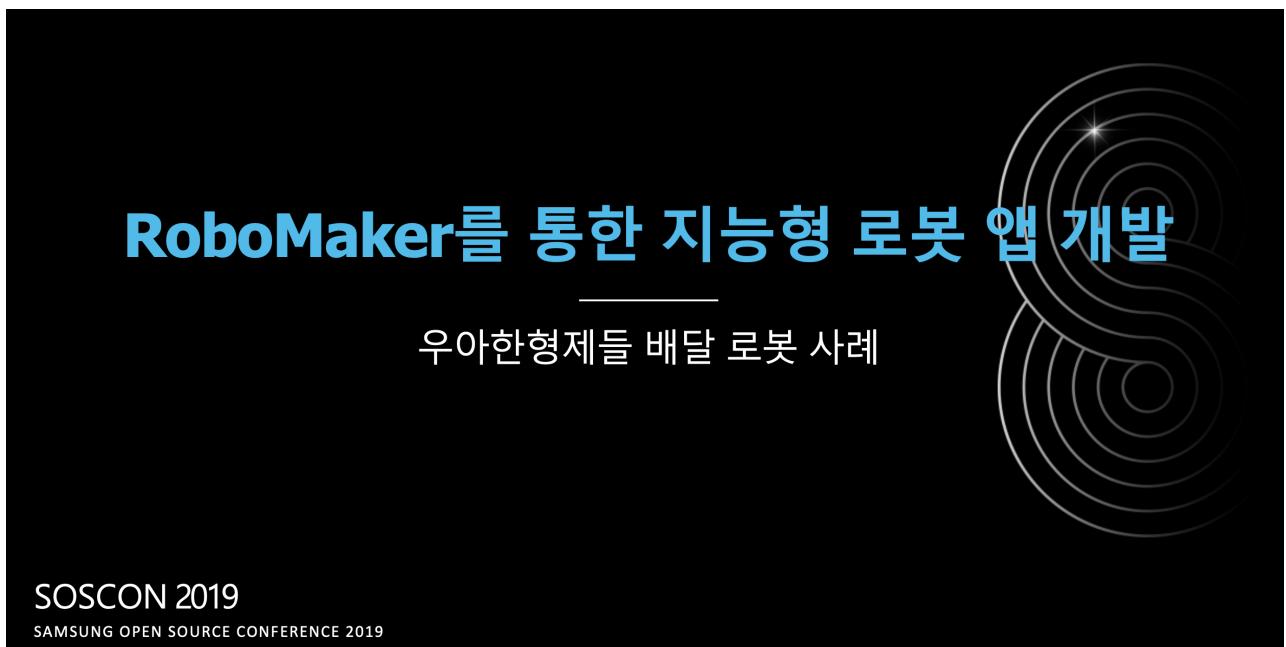
### 6.4 ROS2의 특징



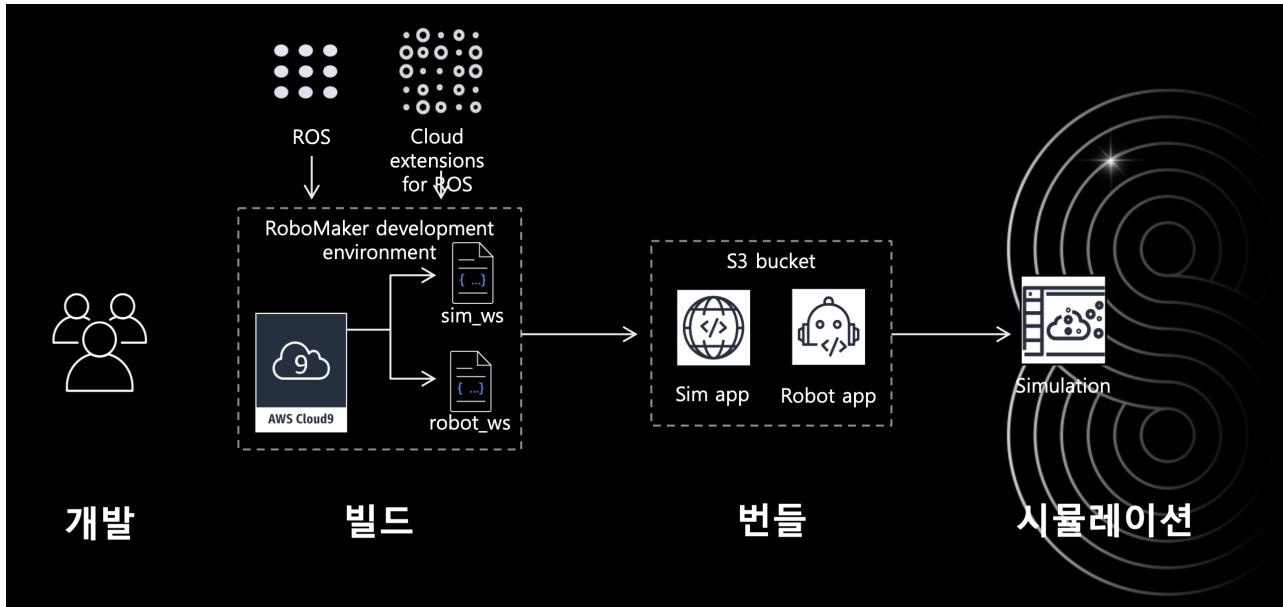
## 6.5 ROS1과 ROS2의 빌드툴



## 6.6 ROS 사용예 - 우아한 형제들



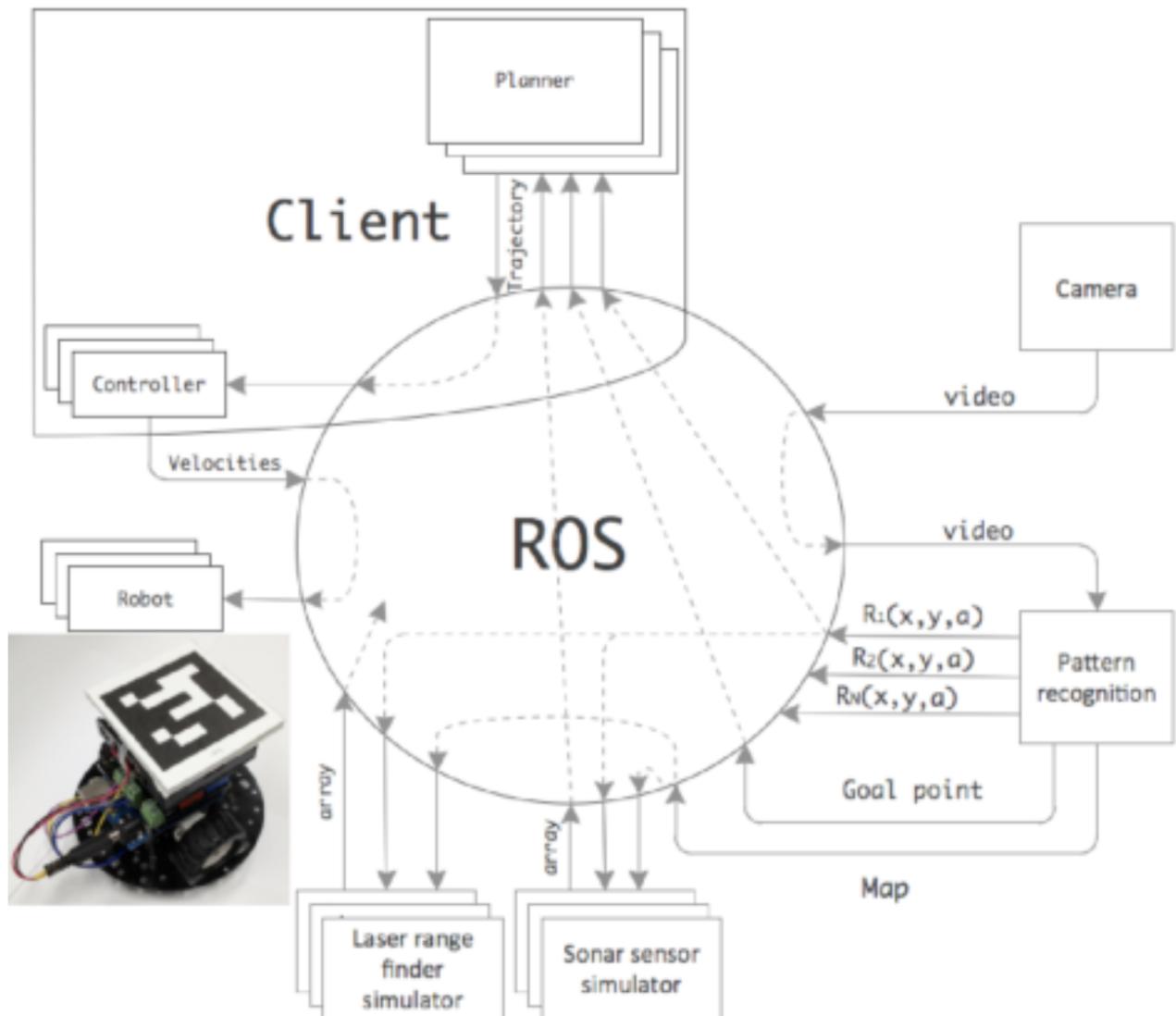
## 6.7 AWS RoboMaker에서 개발환경 구성



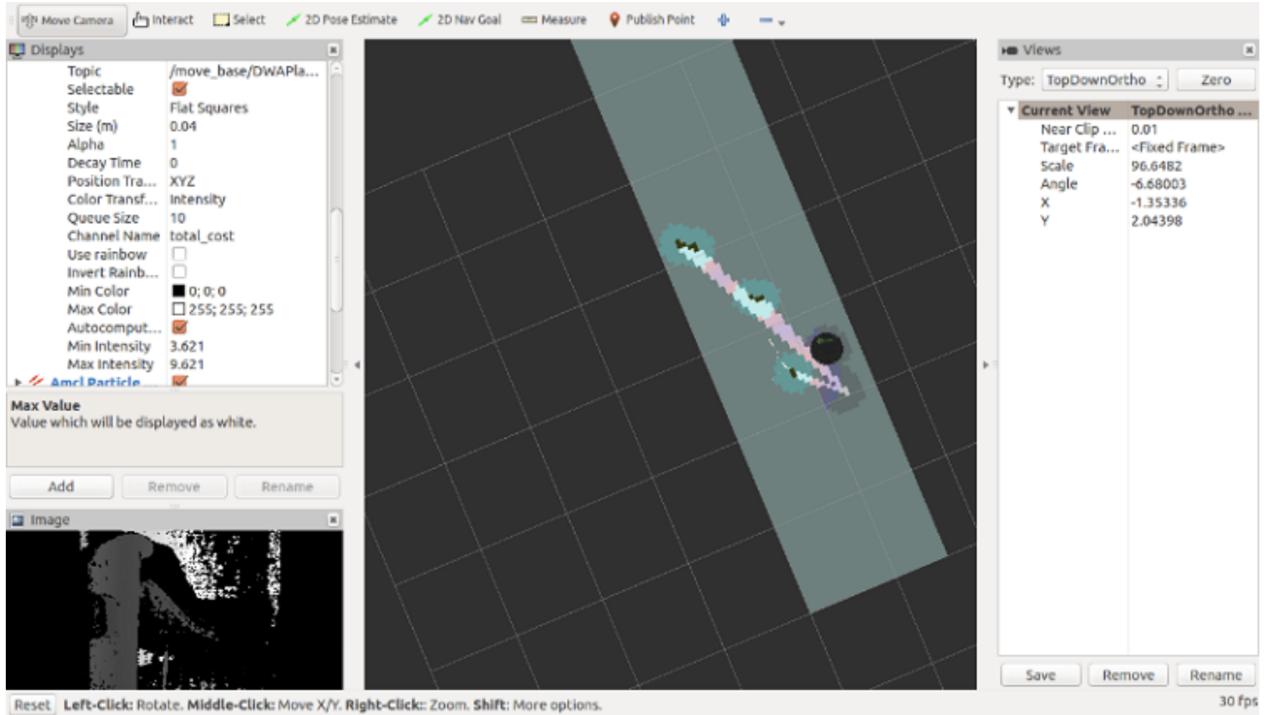
## 6.8 배민 배달 로봇



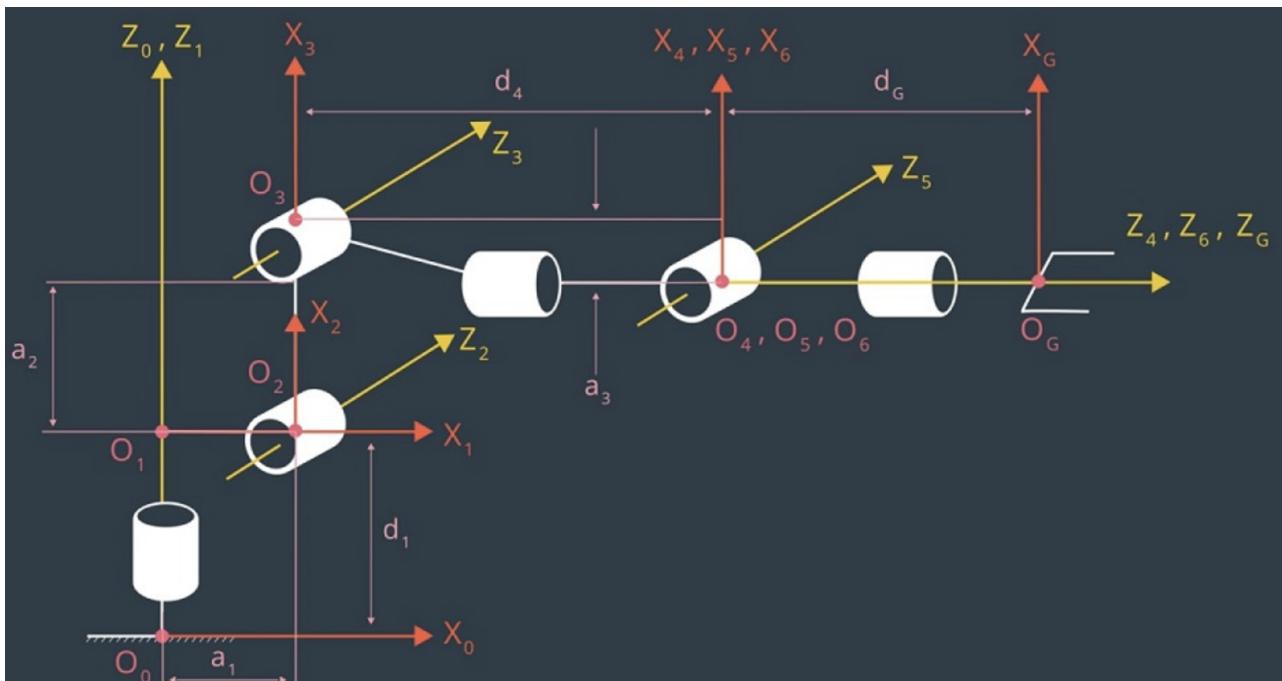
## 6.9 ROS는 System Architecture에 대한 부담을 덜어준다



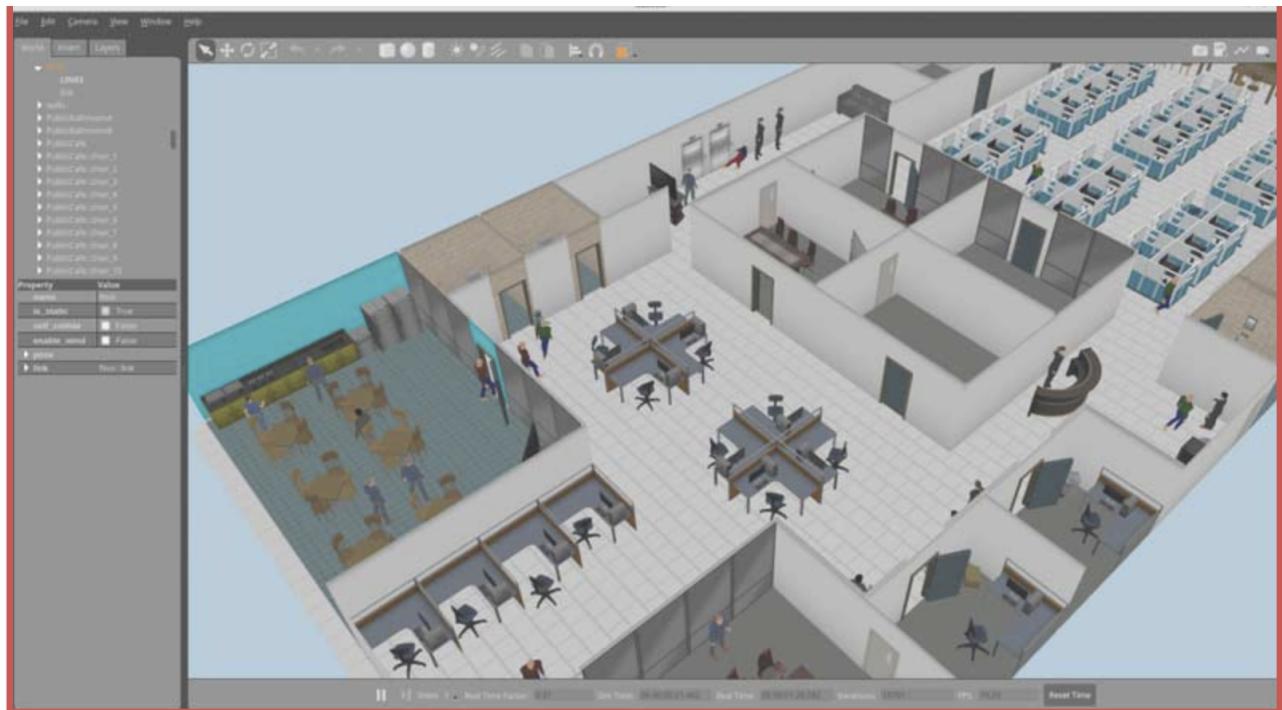
## 6.10 RViz



## 6.11 Inverse Kinematics



## 6.12 물리엔진이 탑재된 Gazebo



6.13 이 모든 기능이 나를 위해 존재한다...

## 7 ROS History

### 7.1 ROS1 History

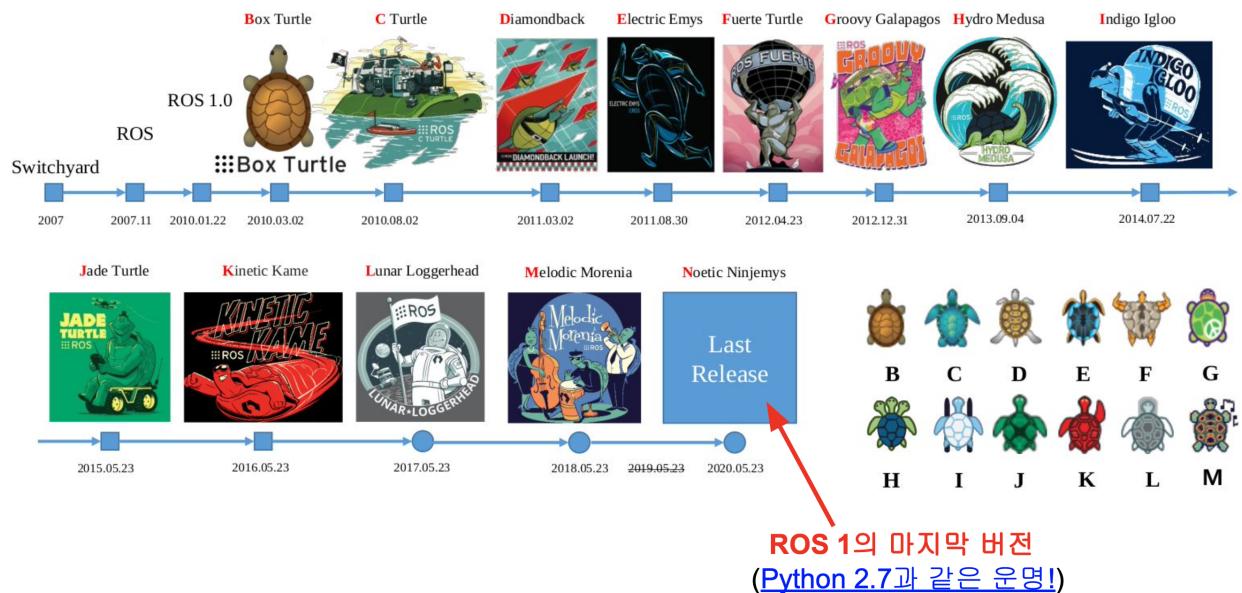
2020.05.23 - **Noetic** Ninjemys 릴리즈  
 2019.10.31 - ROSCon2019 컨퍼런스 개최 (중국)  
 2019.09.25 - ROSCon JP 2019 컨퍼런스 개최 (일본)  
 2019.06.11 - ROSCon FR 2019 컨퍼런스 개최 (프랑스)  
 2018.09.29 - ROSCon2018 컨퍼런스 개최 (스페인)  
 2018.09.14 - ROSCon JP 2018 컨퍼런스 개최 (일본)  
 2018.05.23 - **Melodic** Morenia 릴리즈  
**2017.12.08 - ROS 2.0 릴리즈**  
 2017.09.21 - ROSCon2017 컨퍼런스 개최 (캐나다)  
 2017.05.23 - **Lunar** Loggerhead 릴리즈  
 2017.05.16 - Open Robotics 으로 이름 변경 (구 OSRF)  
 2016.10.08 - ROSCon2016 컨퍼런스 개최 (한국)  
 2016.05.23 - **Kinetic** Kame 릴리즈  
 2015.10.03 - ROSCon2015 컨퍼런스 개최 (독일)  
 2015.05.23 - **Jade** Turtle 릴리즈  
 2014.09.12 - ROSCon2014 컨퍼런스 개최 (미국)  
 2014.07.22 - **Indigo** Igloo 릴리즈  
 2014.06.06 - ROS Kong 2014 개최 (홍콩)  
 2013.09.04 - **Hydro** Medusa 릴리즈  
 2013.05.11 - ROSCon2013 컨퍼런스 개최 (독일)  
 2013.02.11 - Open Source Robotics Foundation 개발, 관리

- 1) <https://www.ros.org/>
- 2) <https://www.openrobotics.org>

2012.12.31 - **Groovy** Galapagos 릴리즈  
 2012.05.19 - ROSCon2012 컨퍼런스 개최 (미국)  
 2012.04.23 - **Fuerte** 릴리즈  
 2011.08.30 - **Electric** Emys 릴리즈  
 2011.03.02 - **Diamondback** 릴리즈  
 2010.08.02 - **C** Turtle 릴리즈  
 2010.03.02 - **Box** Turtle 릴리즈  
 2010.01.22 - **ROS 1.0** 개발  
 2007.11.01 - Willow Garage 가 “ROS”라는 이름으로 개발 시작  
 2007.05.01 - Switchyard, Morgan Quigley, Stanford AI LAB, 스탠포드 대학  
 2000 - Player/Stage Project, Brian Gerkey, 남캘리포니아 대학



### 7.2 ROS1



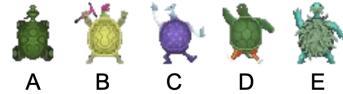
**ROS 1의 마지막 버전**  
(Python 2.7과 같은 운명!)

## 7.3 ROS2 History

2020.05.23 - ROS 2 **Foxy** Fitzroy release (LTS, 3+ years support)  
 2019.11.22 - ROS 2 **Eloquent** Elusor release  
 2019.05.31 - ROS 2 **Dashing** Diademata release (First LTS, 2 years support)  
 2018.12.14 - ROS 2 **Crystal** Clemmys release  
 2018.07.02 - ROS 2 **Bouncy** Bolson release  
 2017.12.08 - ROS 2 **Ardent** Apalone release (1st version)



- 1) <https://index.ros.org/doc/ros2/>
- 2) <http://design.ros2.org/>
- 3) <https://github.com/ros-infrastructure/>
- 4) <https://index.ros.org/doc/ros2 Releases/>
- 5) [https://github.com/ros/ros\\_tutorials](https://github.com/ros/ros_tutorials)



## 8 이제 배우고 즐길 때이다