

스마트 모빌리티 프로그래밍

## Ch 22. 시스템 통합, 기본 주행 기능 점검, 자율주행제어



**김 영 탁**

영남대학교 기계IT대학 정보통신공학과  
(Tel : +82-53-810-3940; E-mail : yse09@ynu.ac.kr)

# Outline

- ◆ 스마트 카 참조 플랫폼
- ◆ 운영체제, 파이썬, ROS 설치 및 시스템 통합
- ◆ 기본 주행 기능 시험
- ◆ SLAM 구동, Map 작성 및 보정
- ◆ 자율 주행 제어 시험

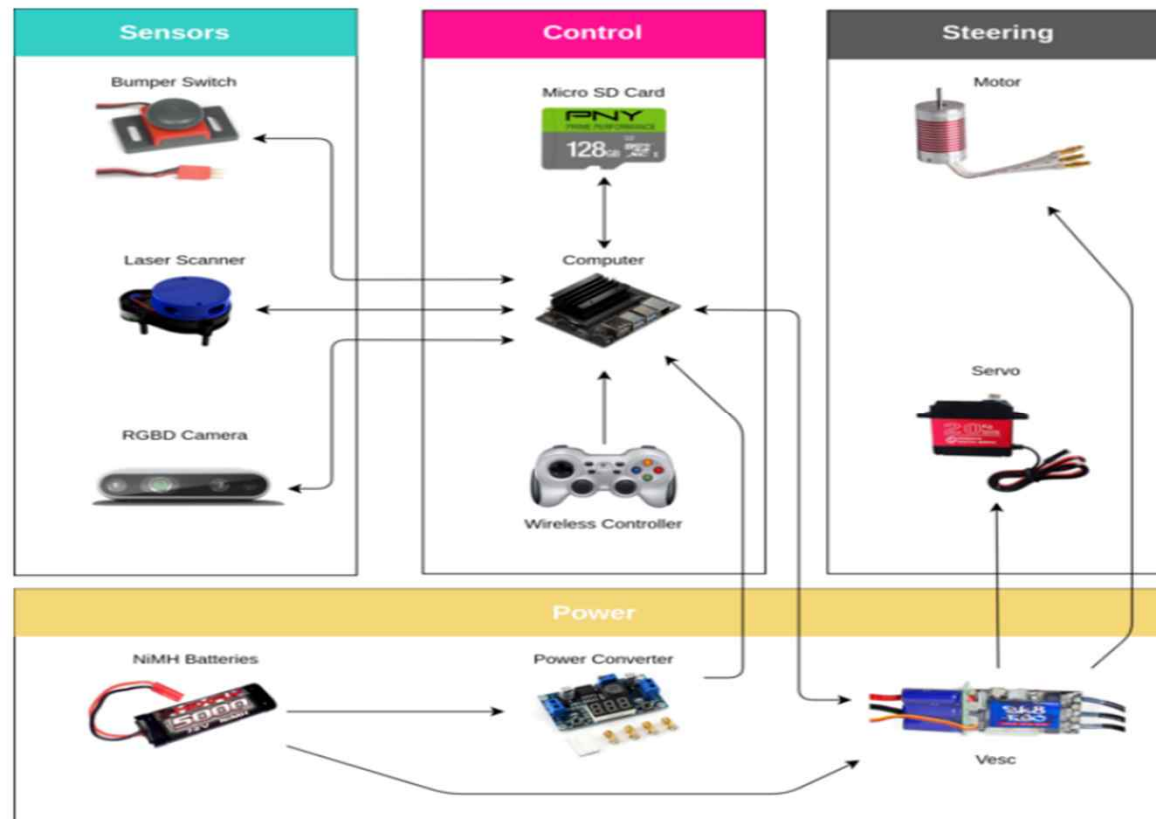


## 스마트 카 참조 모델

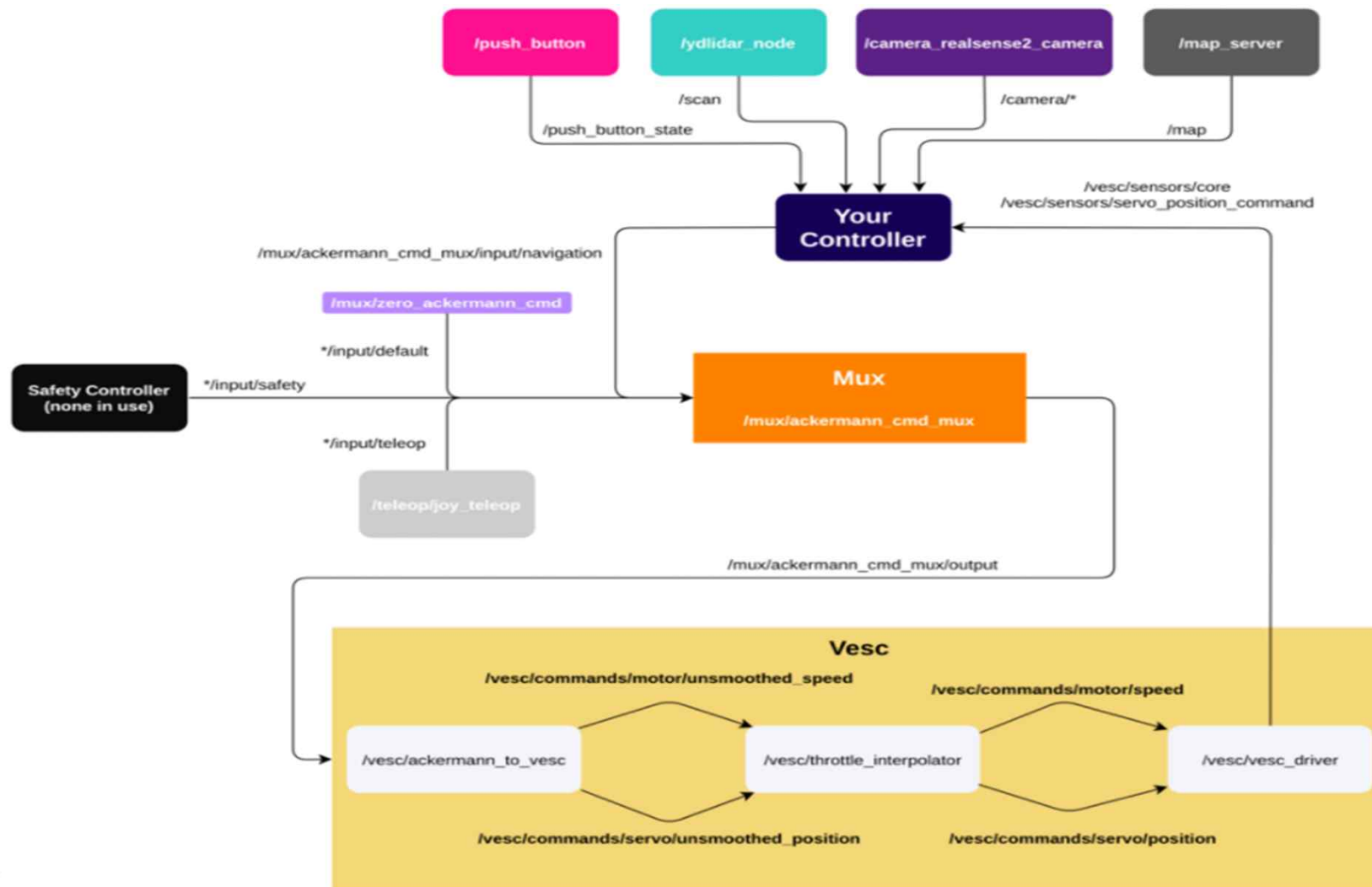
# 스마트 카 참조 모델 (1) - MuSHR

## ◆ MuSHR – The UW (Univ. of Washington) Open Racecar Project

- <https://mushr.io/tutorials/overview/>



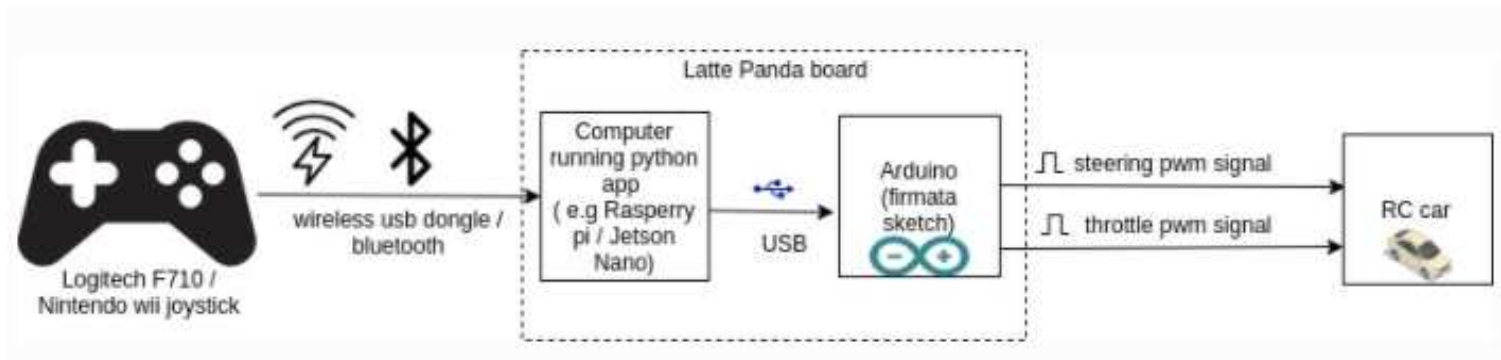
# 스마트 카 참조 모델 (1) - MuSHR Functional Software Block Diagram



## 스마트 카 참조 모델 (2) - Donkey Car

### ◆ Donkey Car

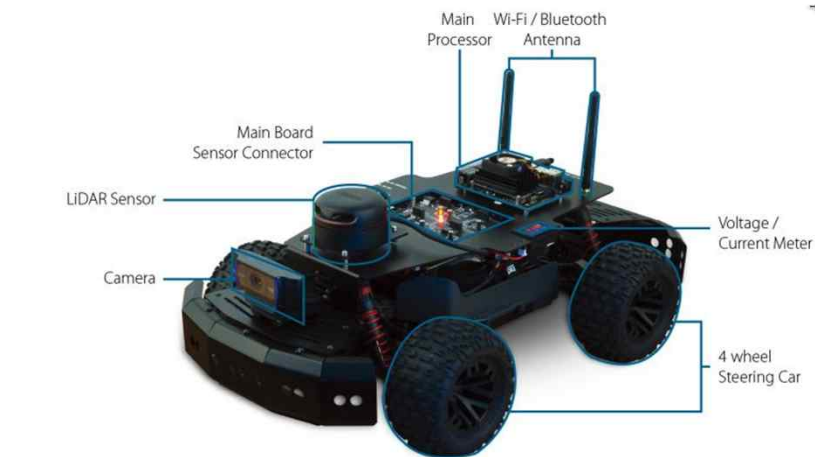
- source: <https://docs.donkeycar.com/>
- steering servo
- throttle motor



# 스마트 카 참조 모델 (3) - 한백전자 HBE RoboCAR

## ◆ AIoT AutoCar Prime Series

- [Source: https://hanback.com/ko/archives/9740](https://hanback.com/ko/archives/9740)



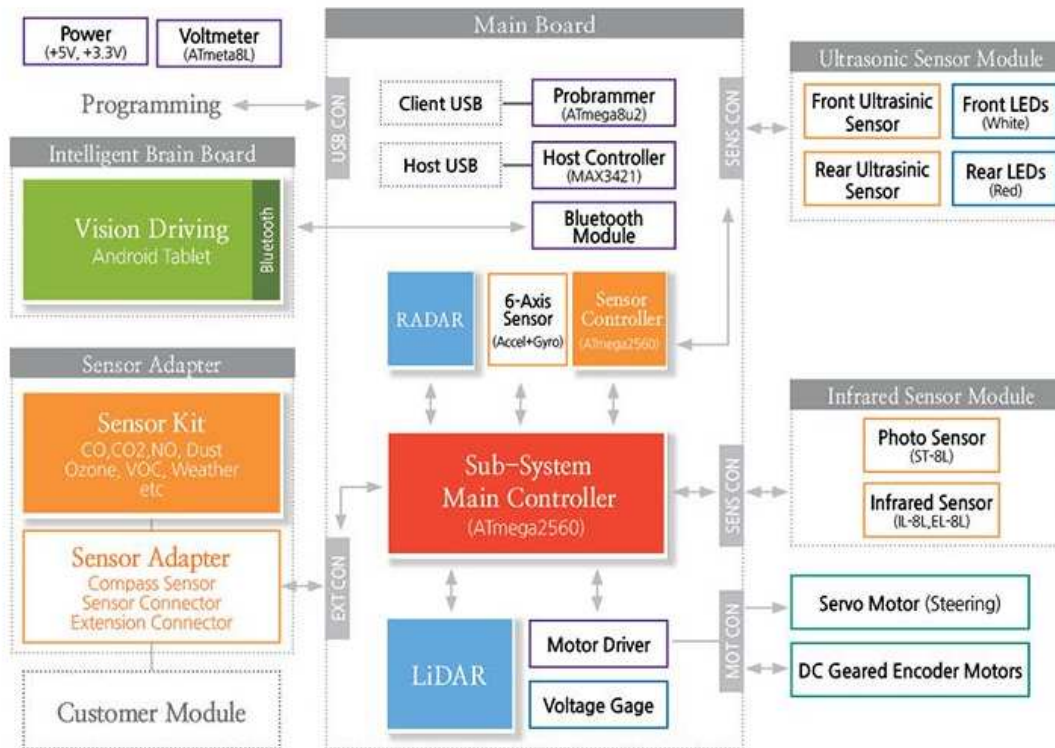
Body		
List	Specification	
Battery	11.1V/7000mA	
Motor	DC Geared Motor ~ DC 12V, Max. 12kg-cm, 1540rpm	
Steering	Servo Motor ~ Stall Torque(6.8V) : 21.5kg/cm ~ Speed : 0.16 sec/60°(9V), 0.14 sec/60°(6.8V)	
Camera	Servo Motor ~ Stall Torque : 9.4 kgf-cm (4.8 V), 11 kgf-cm (6 V) ~ Operating Speed : 0.17 s/60°(4.8 V), 0.14 s/60°(6 V) Servo Brackets 2EA Camera Guide	
PAN/TILT Part		
Size	340 X 600 X 220 (mm)	
Weight	6kg	
Wheels	4Wheels	



## 스마트 카 참조 모델 (3) - 한백전자 SmartCar

### ◆ 기능 블록도

- ATmega2560 main controller, ATmega128 serial controller
- Nvidia Jetson Nano for AI/ML processing of vision



ch 22 - 8



## 스마트 카 참조 모델 (4) - 휴인스 AI Lab-Car

### ◆ Huins AI Lab-Car

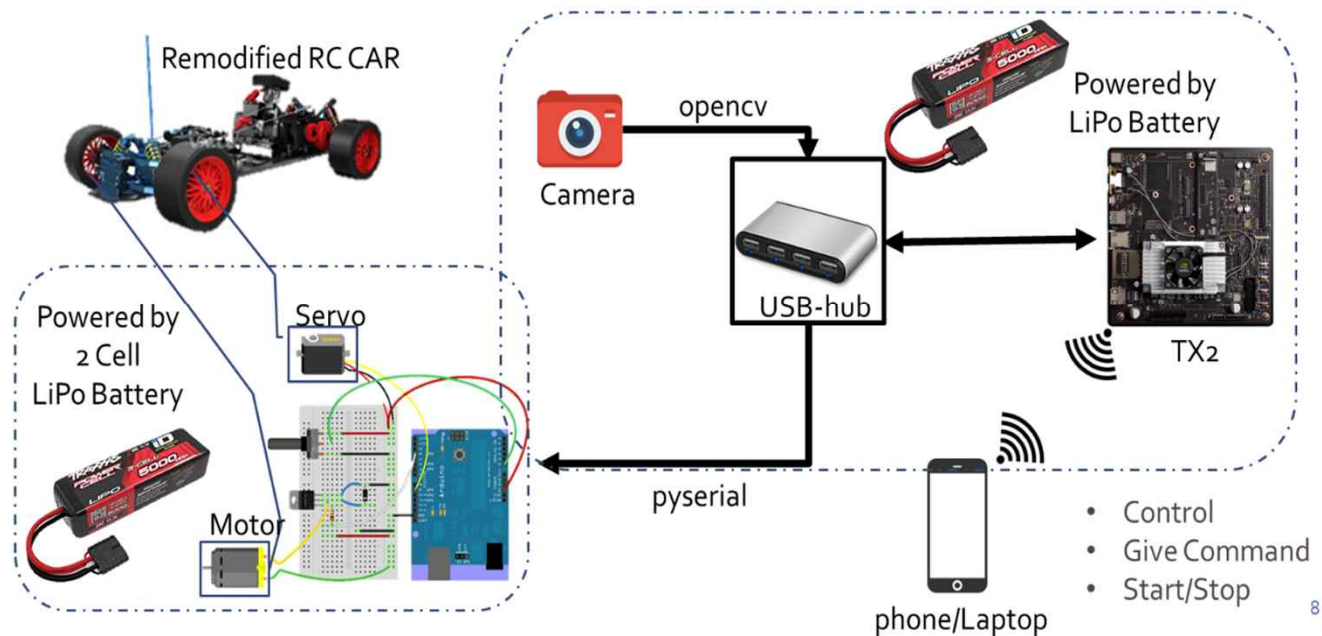
- source:  
[http://www.huins.com/new/sub/goods\\_view.php?ca\\_id=70&ca\\_id2=7020&it\\_id=1598408261&n=2&sn=1](http://www.huins.com/new/sub/goods_view.php?ca_id=70&ca_id2=7020&it_id=1598408261&n=2&sn=1)



## 스마트 카 참조 모델 (4) - 휴인스 AI Lab-Car

### ◆ 시스템 구성도

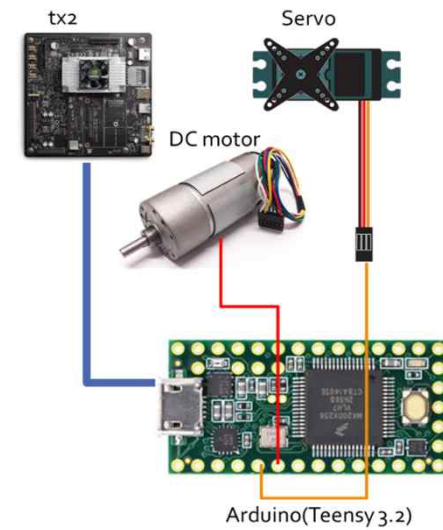
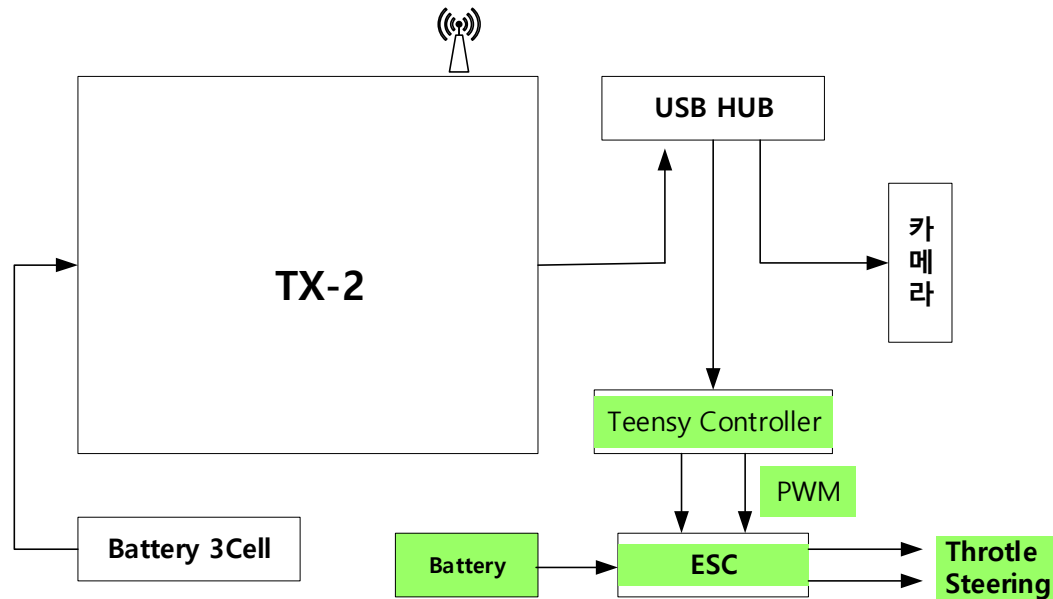
- Nvidia Jetson TX2 board 기반 vision processing
- Servo motor for steering
- DC motor for throttling



## 스마트 카 참조 모델 (4) - 휴인스 AI Lab-Car

### ◆ AI Lab-Car Steering 및 주행 구동부

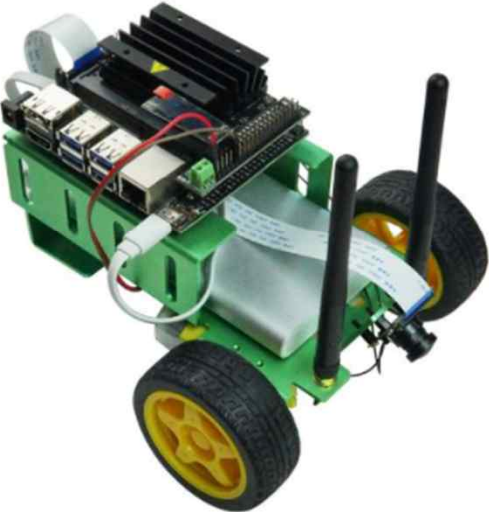
- Steering을 위한 servo motor
- 주행 (throttling)을 위한 DC motor



## 스마트 카 참조 모델 (5) – JetBot Smart Car Kit

### ◆ Seeedstudio JetBot Smart Car Powered by NVIDIA Jetson Nano

- <https://www.seeedstudio.com/Seeedstudio-JetBot-Smart-Car-Powered-by-NVIDIA-Jetson-Nano-p-4054.html>



Seeedstudio JetBot Smart Car  
Powered by NVIDIA Jetson Nano

SKU 110090221

The JetBot kit is based on the NVIDIA Jetson Nano AI computer and includes the complete robot chassis, wheels, and 8MP camera.

**\$248.00**  
Out Of Stock

Subscribe to back in stock notification

**SUBSCRIBE**

## 스마트 카 참조 모델 (6) – Racing Robot Kit Donkey Car

### ◆ XiaoR Geek AI Racing Robot Kit Donkey Car

- <https://www.amazon.com/XiaoR-Geek-Starter-Platform-Upgraded/dp/B07TZ449VP>



Visit the XiaoR Geek Store

XiaoR Geek AI Racing Robot Kit Donkey Car  
Starter Kit Powered by Jetson Nano Deep  
Learning Self Driving Platform for Small Scale  
Cars Jetson Nano AI Smart Robot  
Car(Upgraded Version)

★★★★☆ 8 ratings

Currently unavailable.

We don't know when or if this item will be back in stock.

Color: Without Jetson Nano

 1 option from \$379.00	 1 option from \$239.00	 \$190.00
------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------

Brand	XiaoR Geek
Item Dimensions LxWxH	7.09 x 11.81 x 7.09 inches
Color	Without Jetson Nano
Material	Rubber
Size	Large

#### About this item

- ◆ XR-F2 is an AI line patrol robot car based on machine intelligence and deep learning framework Tensorflow. Use NVIDIA Jetson as the main controller, rich learning materials, you can use peripheral sensors and accessories, and lay a



## **Realtime Operating System 기능 구성**

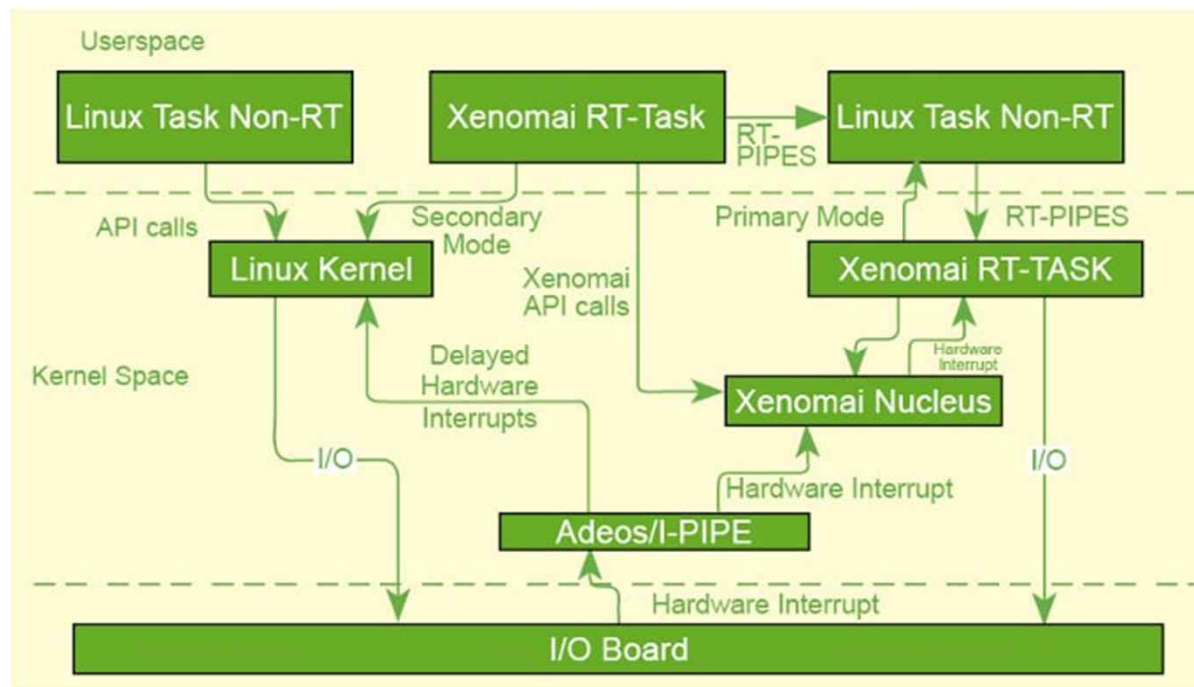
# 실시간 운영체제 기능 구성을 위한 Xenomai

## ◆ Xenomai

- <https://en.wikipedia.org/wiki/Xenomai>
- Xenomai is a real-time development [software framework](#) cooperating with the [Linux kernel](#) to provide [pervasive](#), interface-agnostic, hard [real-time computing](#) support to [user space application software](#) seamlessly integrated into the [Linux](#) environment.
- The Xenomai project was launched in August 2001. In 2003, it merged with the Real-Time Application Interface ([RTAI](#)) project to produce RTAI/fusion, a real-time free software platform for Linux on Xenomai's abstract [real-time operating system](#) (RTOS) core. Eventually, the RTAI/fusion effort became independent from [RTAI](#) in 2005 as the Xenomai project.
- Xenomai is based on an abstract RTOS core, usable for building any kind of real-time interface, over a nucleus which exports a set of generic RTOS services. Any number of RTOS personalities called “skins” can then be built over the nucleus, providing their own specific interface to the applications, by using the services of a single generic core to implement it.

# Xenomai

## ◆ Xenomai/Linux기반 Realtime Task 실행 구조



(참고자료: <https://www.opensourceforu.com/2015/10/the-xenomai-project-a-linux-based-rtos/>)



## 차량 내부 및 외부 통신망

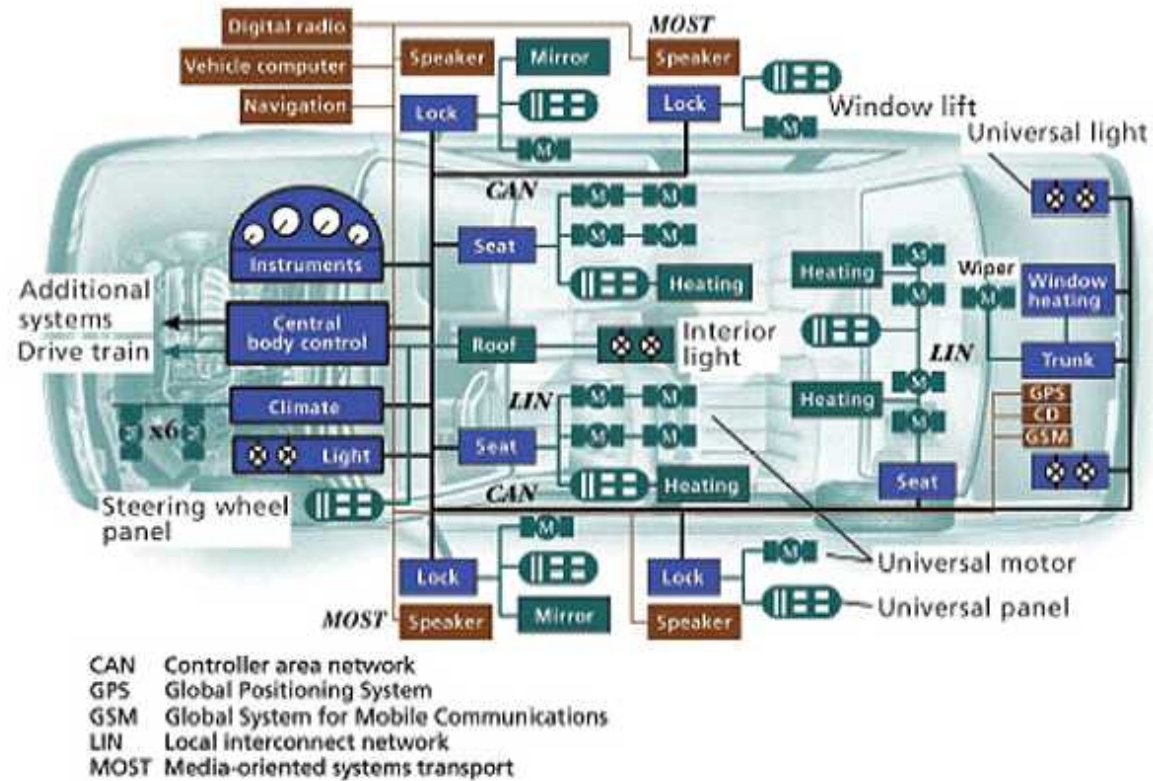
# Sensors and Actuators in a Car

## ◆ Sensors and Actuators in a Car



([http://www.aa1car.com/library/can\\_systems.htm](http://www.aa1car.com/library/can_systems.htm))

# CAN (Controller Area Network)



(Source: <http://hollisbrothersauto.com/your-car-can-network-with-controller-area-networks-can/>)

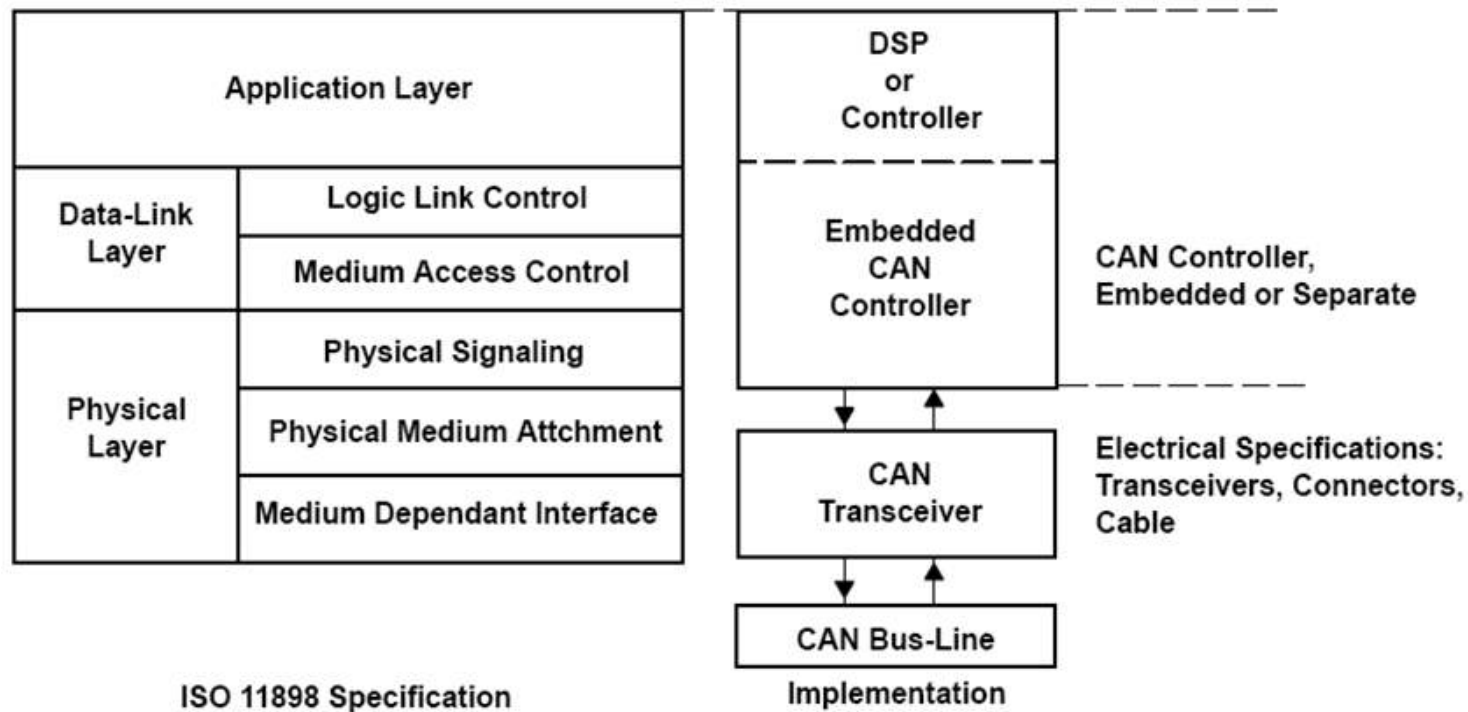
# Controller Area Network (CAN)

## ◆ What is CAN ?

- developed by BOSCH for real-time communications among automobile electrical controller devices (Ex : ABS, engine management, speed-change management)
- CSMA/CD+AMP  
(carrier-sense multiple-access protocol with collision detection and arbitration on message priority)
- Simple 2-wire differential bus
- Bit rates up to 1Mbit/s
- 3 types of CAN version
- Four Frame Types
- Five Error Types
- Fault Confinement

# Layered Architecture of CAN

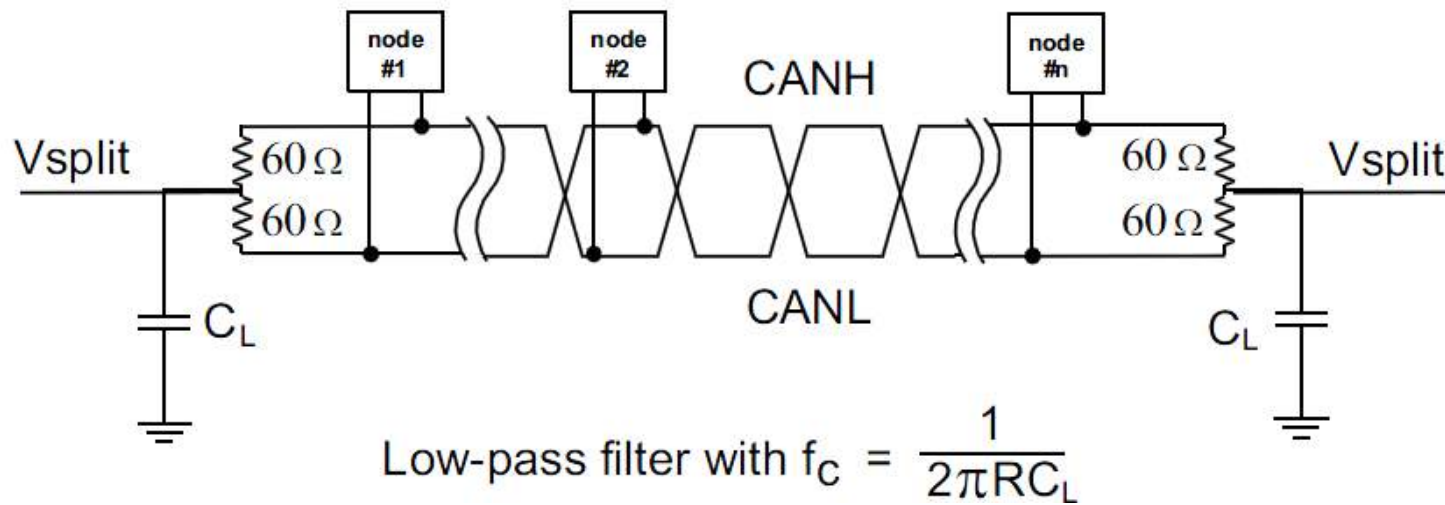
## ◆ Layered ISO 11898 Standard Architecture



ISO 11898 Specification

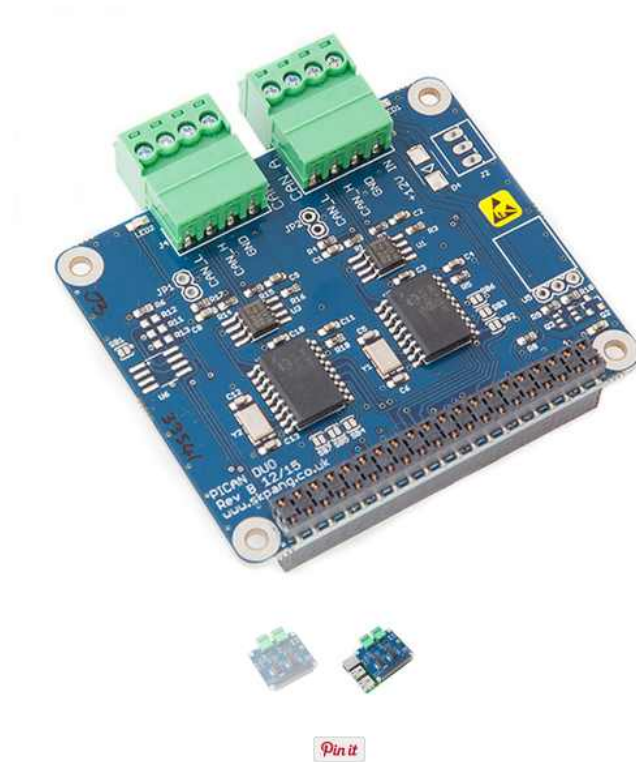
Implementation

# Split Termination of CAN Bus



# PiCAN2 Duo CAN-Bus Board for Raspberry Pi 2

Home > CAN Interfaces > PiCAN2 Duo CAN-Bus Board for Raspberry Pi 2



SK PANG ELECTRONICS

## PiCAN2 Duo CAN-Bus Board for Raspberry Pi 2

\$68.95

SKU:

piCAN2-DUO

Shipping:

Calculated at checkout

Quantity:

1

ADD TO CART

ADD TO WISHLIST



# LIN (Local Interconnect Network)

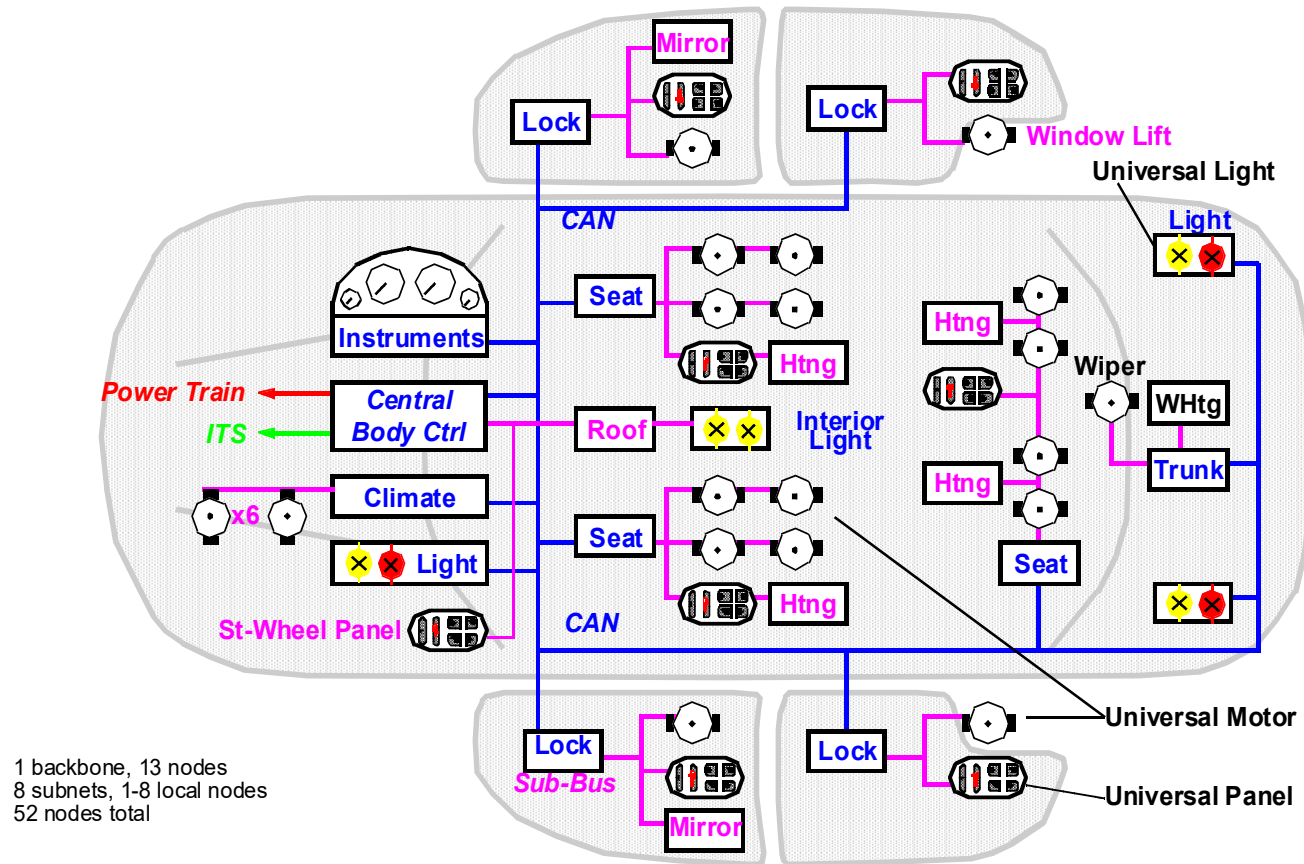
## ◆ LIN Overview

- LIN (Local Interconnect Network) is a [serial network protocol](#) used for communication between components in vehicles. The need for a cheap serial network arose as the technologies and the facilities implemented in the car grew, while the [CAN bus](#) was too expensive to implement for every component in the car.
- European car manufacturers started using different serial communication topologies, which led to compatibility problems.
- In the late 1990s, the LIN Consortium was founded by five automakers ([BMW](#), [Volkswagen Group](#), [Audi Group](#), [Volvo Cars](#), [Mercedes-Benz](#)), with the technologies supplied (networking and hardware expertise) from Volcano Automotive Group and [Motorola](#).
- The first fully implemented version of the new LIN specification (LIN version 1.3) was published in November 2002. In September 2003, version 2.0 was introduced to expand capabilities and make provisions for additional diagnostics features. LIN may be used also over the vehicle's battery [power-line](#) with a special DC-LIN transceiver.





# Automotive Body Network



## CAN vs. LIN

Features	LIN	CAN
Medium access control	single master	multiple masters
Typical bus speed	2.4, 9.6 and 19.2kbps	62,5...1000kbps
Multicast message routing	6 bit identifier	11/29 bit identifier
Typical size of network	2...16 nodes	4...20 nodes
Data byte per frame	0... 8	2...8
Transmission time for 4 data bytes	6 ms at 20kbps	0.8 ms at 125kbps
Error detection (data field)	8-bit checksum	15-bit CRC
Physical layer	single-wire, 12 V	twisted-pair, 5 V
Quartz/ceramic resonator	master only	Yes
Relative cost per network connection	x 0.5	x 1.0

# EtherCAT

## ◆ EtherCAT이란?

- Introduced by Beckhoff in 2003, and currently managed by EtherCAT Technology Group - <https://www.ethercat.org/default.htm>
- Like EtherNet/IP, EtherCAT is built on the Ethernet physical layer. But instead of using TCP/IP for transport and routing of messages, EtherCAT uses a “processing-on-the-fly” (also referred to as “communicating-on-the-fly”) approach
- In this approach, the EtherCAT master sends a telegram (data packet) that passes through each node, or slave, (typically a drive or I/O device).
- A unique feature of EtherCAT is that the networked slaves can read, or extract, only the relevant information they need from the telegram and can add data to the telegram before it travels to the next slave. The telegram travels through all the connected slaves and then returns to the master.
- Reading and writing to/from the telegram is enabled by a special [ASIC](#) on each EtherCAT slave. This hardware-based approach means that each slave introduces minimum latency (delay) to the process, and collisions are not possible. Network commands on an EtherCAT network can be processed at speeds that rival those of analog-based systems.
- EtherCAT provides deterministic, real-time communication and is well-suited for synchronized, multi-axis motion control “out of the box,” without the need for additional hardware to achieve synchronization between multiple axes.

# EtherCat Master install

◆ 현재 EtherCat은 ubuntu 12.04 LTS 버전에서 동작가능

◆ Get the EtherLab Bundle 2.1

- <https://www.etherlab.org/en/components.php>

## Downloads

Release	Date
<b>EtherLab® Bundle 2.1</b> (current), including all components. [ <a href="#">tar.bz2</a> , 99M] [ <a href="#">sig</a> ] [ <a href="#">md5</a> ] <ul style="list-style-type: none"><li>• Linux kernel 3.4.69-rt85</li><li>• EtherCAT Master 1.5.2</li><li>• EtherLab 2.1.0</li><li>• PdServ 1.1.0</li><li>• Testmanager 3.6.2</li><li>• Data-Logging-Service 1.3.0</li></ul>	2013-12-11

- 호환 가능한 리눅스 커널 버전과 EtherCAT Master, Slave 패키지 포함되어 있는 압축 파일 다운로드

# EtherCat Master install

## ◆ Bundle Package 구성

```
root@antl:~/etherlab-bundle-2.1# ls
2_kernel  3_ethercat  5_etherlab  6_testmanager  7_dls  8_examples  etherlab-technology-2.0-2013-02-05.pdf
```

## ◆ Kernel build & install

- Xenomai 커널 빌드 방법과 동일

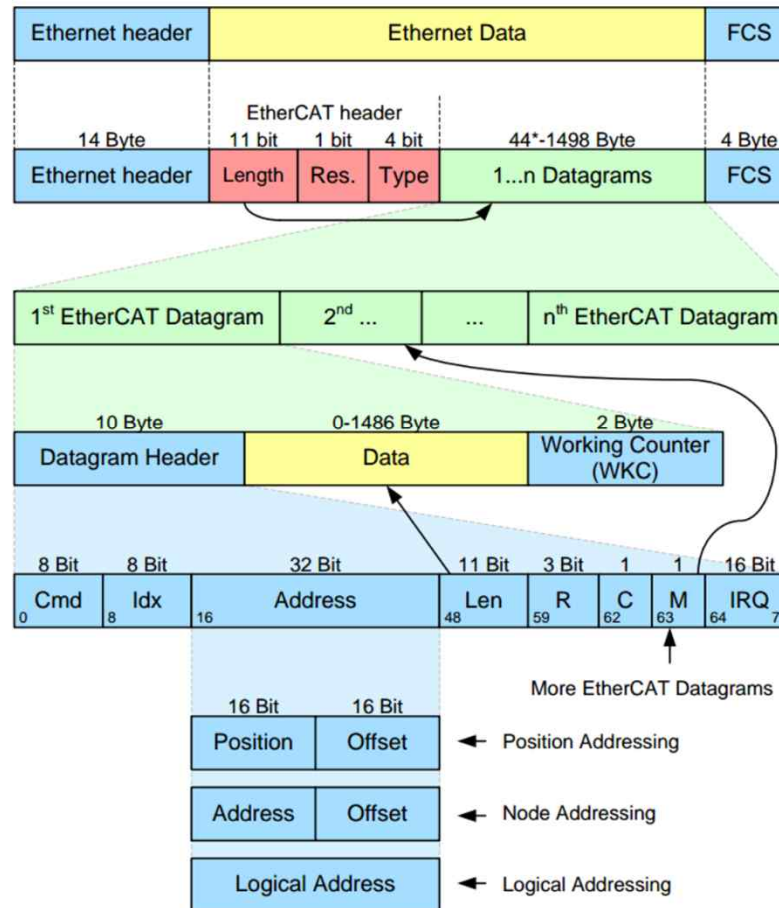
## ◆ Download EtherCat master

- mkdir /usr/local/etherlab
- mkdir /usr/local/etherlab/src
- cd /usr/local/etherlab/src
- wget <http://www.etherlab.org/download/ethercat/ethercat-1.5.2.tar.bz2>
- tar -xjf ethercat-1.5.2.tar.bz2



# EtherCAT datagram structure

\* add 1-32 padding bytes if Ethernet frame is shorter than 64 Bytes (Ethernet Header+Ethernet Data+FCS)



Datagram header	Command	8 bit
	Index	8 bit
	Address	32 bit
	Length	11 bit
	Reserved	3 bit
	Circulating	1 bit
	NEXT	1 bit
	IRQ	16 bit

# Comparison of EtherCAT Masters in GNU/Linux

Table 3.2: *EtherLab versus SOEM.*

	EtherLab	SOEM
Learning Curve (Installation, Configuration and API)	Steep	Low
Documentation	Excellent (since v.1.5)	Poor
Mailing List Support	Yes	Yes
Portability	No	Yes
Integrability of EtherCAT (EM Type)	Full (Type A)	Some EtherCAT features not implemented (Type B)
user-space / kernel-space	kernel-space with user-space API	user-space
Licence	LGPLv2	GPLv2

# EtherCat Master phases and transitions

## ◆ Orphaned phase

- In this phase the master waits for its Ethernet device(s) to connect. There is no bus configuration available.

## ◆ Idle phase

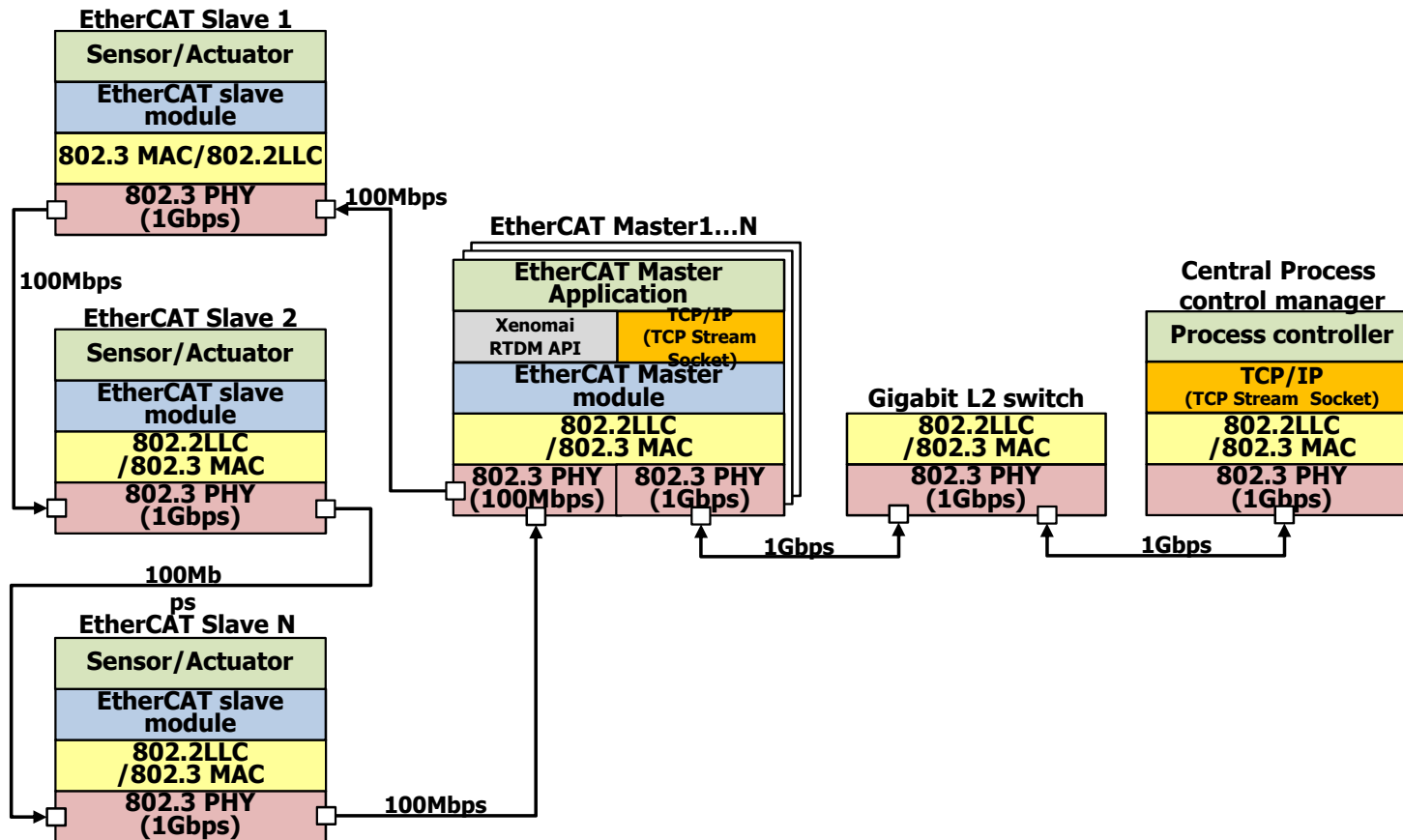
- In this phase the master has accepted all available Ethernet devices, but is not requested by any application yet. Thus, the master merely runs its state machine. This involves automatically scanning the bus for slaves and executing pending operations from the user-space interface (for example SDO access). Again, there is no Process Data exchange since the bus communication isn't configured yet.

## ◆ Operation phase

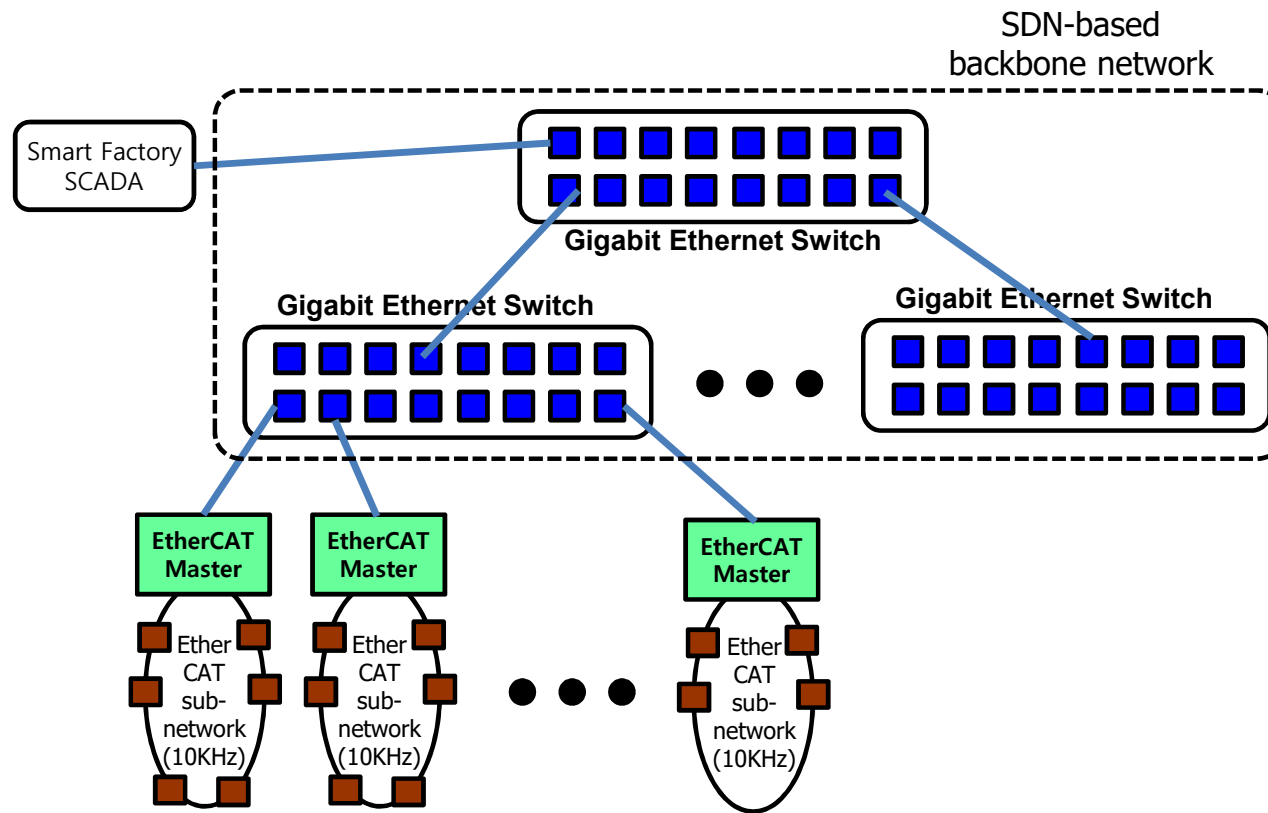
- In this phase the master is requested by an application which provides a bus configuration and exchanges Process Data.



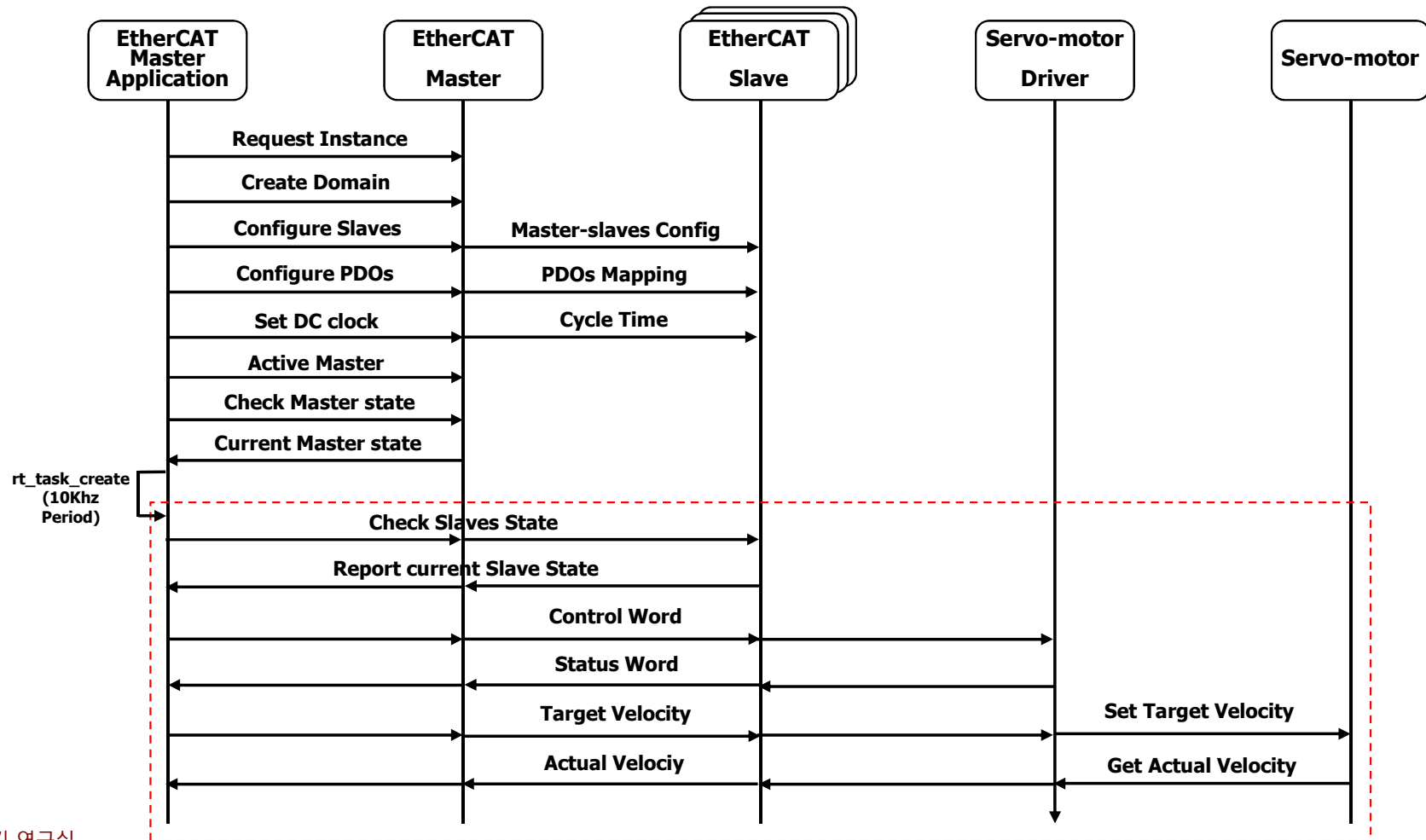
# EtherCAT 구성도



# EtherCAT Master 기반 계층적 네트워킹



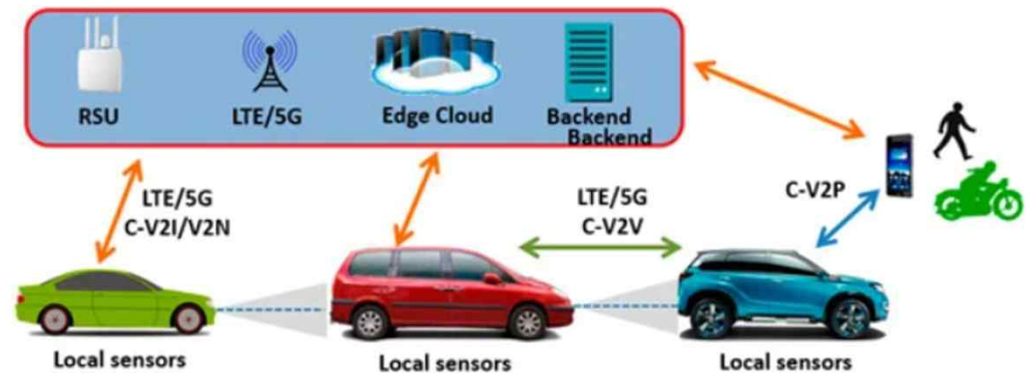
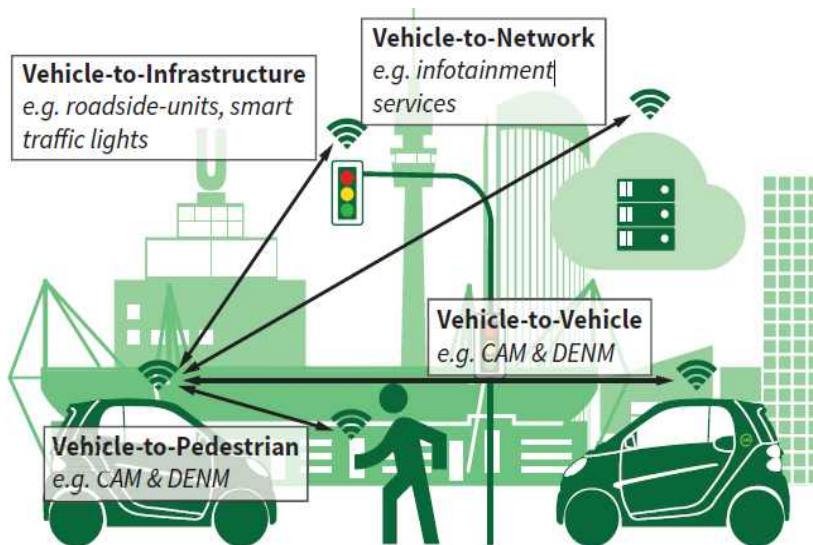
# Real-time User Space Application Sequence diagram using Servo-motor



# C-V2X Communication

## ◆ C-V2X(Cellular Vehicle to Everything)

- The 3GPP(Third Generation Partnership Project) cellular vehicle to everything(C-V2X) standard was released in 2017(rel. 14).
- The 3GPP specifies four types of C-V2X applications.



CAM: Cooperative Aware Message  
DENM: Decentralized Environmental Notification Messages

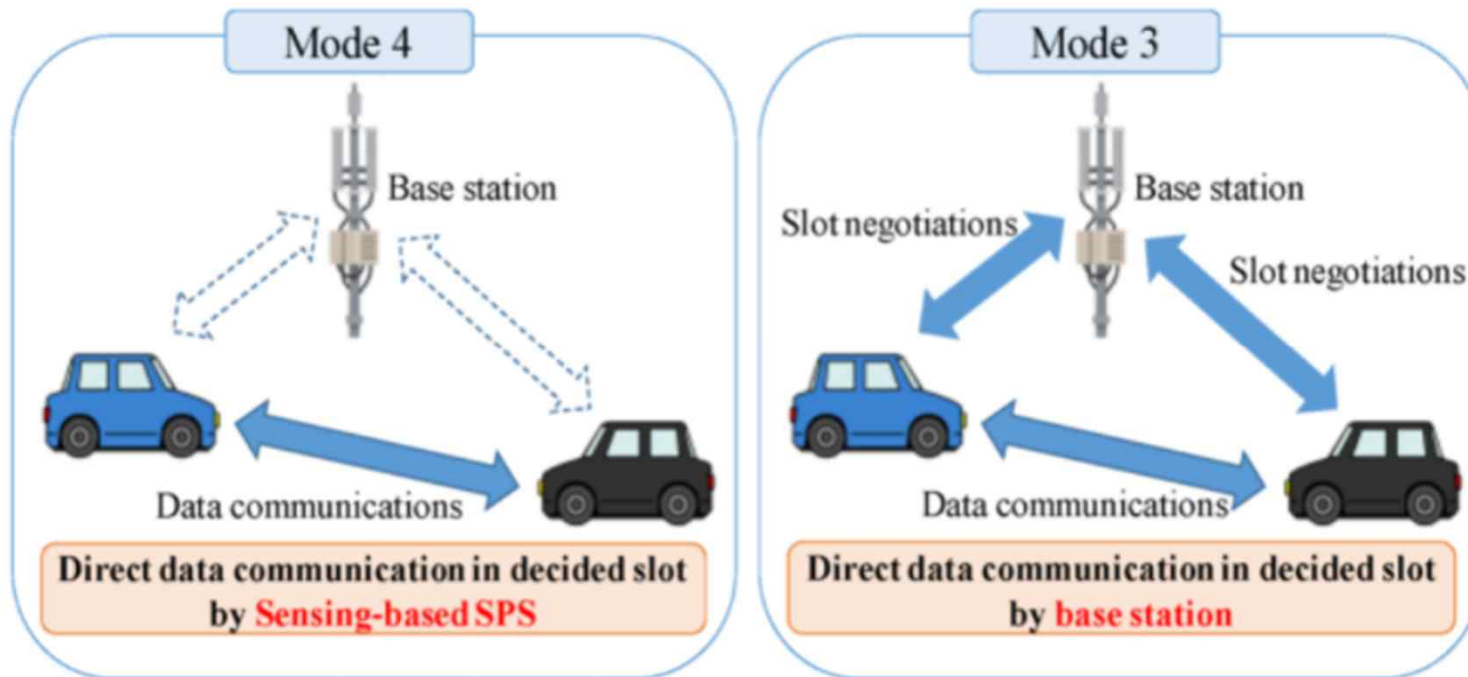
# C-V2X

## ◆ C-V2X

- LTE-V2V adopts single carrier frequency division multiple access (SC-FDMA) at PHY and MAC layers
- resource blocks (RBs) are allocated in pairs, corresponding to 180KHz bandwidth (12 subcarriers with 15KHz space) and 1 ms duration (14 OFDM symbols, of which 9 carry data, 4 are used for channel estimation, and one for timing and possible tx-rx switch)
- minimum allocation time interval is 1 ms; transmission time interval (TTI)
- resource blocks (RBs) are also grouped in the frequency domain into sub-channels, which are all of a given size
- each sub-channel can carry at most one data packet, although one data packet can span over more than one sub-channel
- Data is transmitted in Transport Blocks (TBs)
- a TB contains a full packet to be transmitted, e.g., a beacon or a CAM (cooperative awareness message)/BSM(basic safety message)
- each TB is transmitted with a side-link control information (SCI) that occupies 2 RBs in the same sub-frame, and represents the signaling overhead in C-V2X Mode 4
- the SCI includes information such as the modulation and coding scheme used to transmit the TB, and the RBs used to transmit the TB

## C-V2X Mode 3, Mode 4

### ◆ C-V2X Mode 3, Mode 4



자율주행 모형카에 운영체제 (OS) 및 파이썬 설치,  
구동부 기능 제어 시험

## 운영체제 (OS) 설치

### ◆ Raspberry Pi

- Raspberry Pi OS (Raspbian)
- Ubuntu 20.04 for ROS

### ◆ Jetson Nano

- JetPack SDK that includes Jetson Linux Driver Package with Linux OS, CUDA-X accelerated libraries and APIs for Deep Learning

### ◆ 플랫폼에 올바른 운영체제 (OS) 설치

- 지원되는 운영체제 버전을 확인



# 파이썬 및 확장 모듈 설치

## ◆ Python 기본 설치

- AI/DL 모듈들과의 호환성이 검증된 모듈 설치
- Python 3.9.9

## ◆ Python 확장모듈

- NumPy
- Matplotlib.plot
- PyTorch



# Install JetPack(1)

## ◆ Jetson Nano Image 파일 설치

- <https://developer.nvidia.com/embedded/jetpack> 접속 후 기기 종류에 맞는 image 다운로드
- Jetson Nano 2GB의 예

### Installing JetPack

#### SD Card Image Method

[JETSON XAVIER NX DEVELOPER KIT >](#)

[JETSON NANO DEVELOPER KITS >](#)

For Jetson Nano Developer Kit:

[Download the SD Card Image](#)

Follow the steps at [Getting Started with Jetson Nano Developer Kit](#).

For Jetson Nano 2GB Developer Kit:

[Download the SD Card Image](#)

Follow the steps at [Getting Started with Jetson Nano 2GB Developer Kit](#).

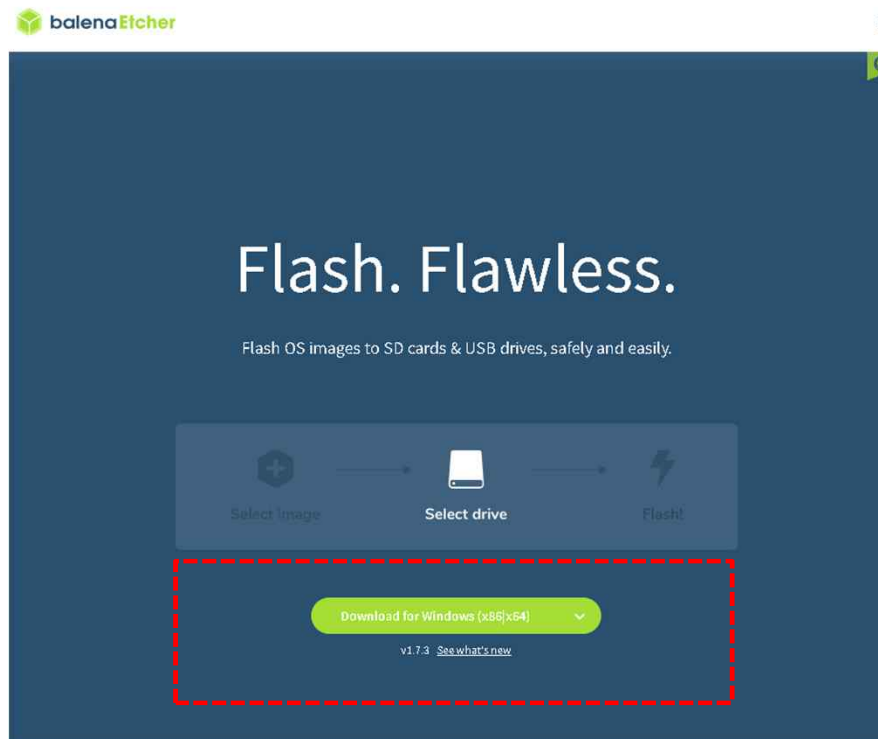
#### NVIDIA SDK Manager Method

[FOR ANY JETSON DEVELOPER KIT >](#)

# Install JetPack(2)

## ◆ Image write(1)

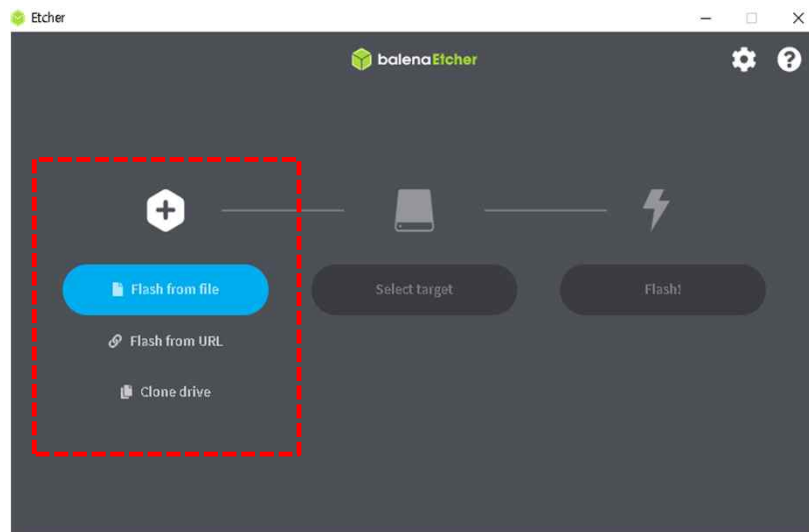
- <https://www.balena.io/etcher/>
- 설치한 image 파일을 write하기 위해 Etcher install



# Install JetPack(3)

## ◆ Image write(2)

- Flash from file 에서 설치한 image 파일 선택
- Image 파일을 write 하고자 하는 sd카드 선택
- Flash



# JetPack SDK(1)

## ◆ CUDA

- `nvcc -V`

## ◆ 만약 `nvcc` 실행에 문제가 있는 경우

- `sudo gedit ~/.profile`
- 설치된 CUDA 경로 확인 후 아래 두 줄 입력

```
export PATH=/usr/local/cuda-10.2/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-10.2/lib64:$LD_LIBRARY_PATH
```

## ◆ `nvcc -V`

```
root@antl-desktop:/usr/local/cuda-10.2# nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Sun_Feb_28_22:34:44_PST_2021
Cuda compilation tools, release 10.2, V10.2.300
Build cuda_10.2_r440.TC440_70.29663091_0
```

## JetPack SDK(2)

### ◆ openCV

- pkg-config --modversion opencv

### ◆ 버전 확인 불가능한 경우 아래 명령어 입력

- pkg-config --modversion opencv3
- pkg-config --modversion opencv4

```
root@antl-desktop:/usr/local/cuda-10.2# pkg-config --modversion opencv4  
4.1.1
```

# JetPack SDK(3)

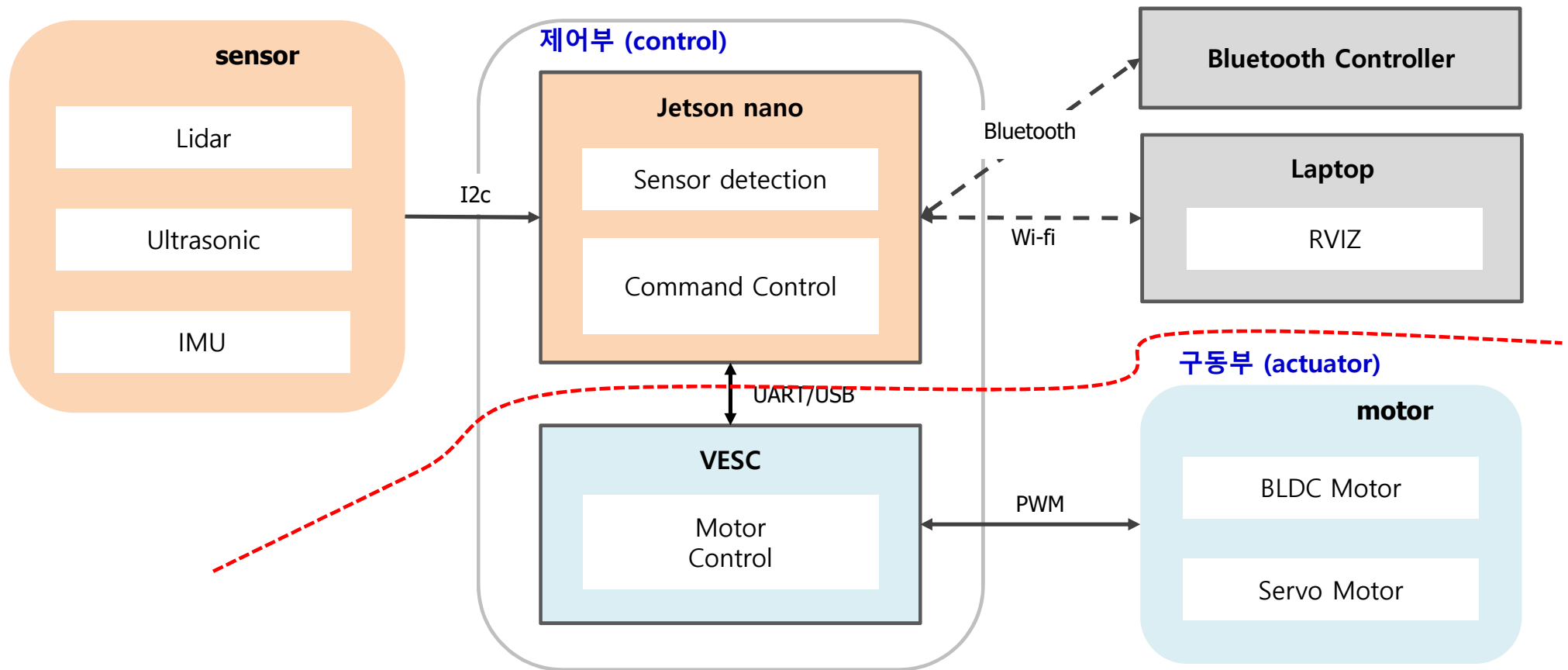
## ◆ cuDNN

- `cat /usr/include/cudnn_version.h | grep CUDNN_MAJOR -A 2`

```
root@antl-desktop: /usr/local/cuda-10.2# cat /usr/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 8
#define CUDNN_MINOR 2
#define CUDNN_PATCHLEVEL 1
--
#define CUDNN_VERSION (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)
#endif /* CUDNN_VERSION_H */
```

- **cuDNN의 버전이 MAJOR (8), MINOR (2), PATCHLEVEL (1)로 표현되어 출력됨**

## 자율 주행 자동차 플랫폼의 기능 블록도 (예시)





## 제어부와 구동부의 통신 기능 확인

### ◆ 제어부 (Raspberry Pi 또는 Jetson Nano)와 구동부 (VESC)의 통신 연결

- serial port
- USB

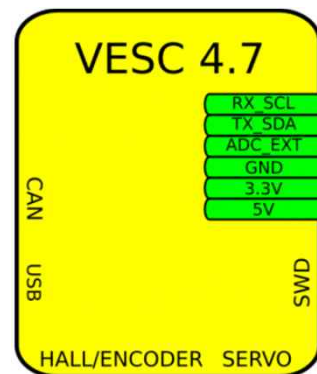
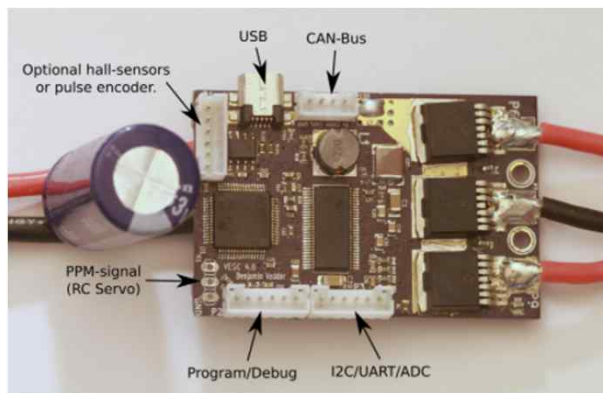
### ◆ 제어부와 구동부의 통신 프로토콜 확인

- ESC (electronic speed controller)
- VESC (Vedder's Electronic Speed Controller)
- ROS 기반의 통신 기능으로 구현
- 직접 Serial 통신 (UART)를 사용하여 명령 전달

# OpenRobot의 VESC 접속을 위한 Custom App

## ◆ VESC와의 통신 및 제어

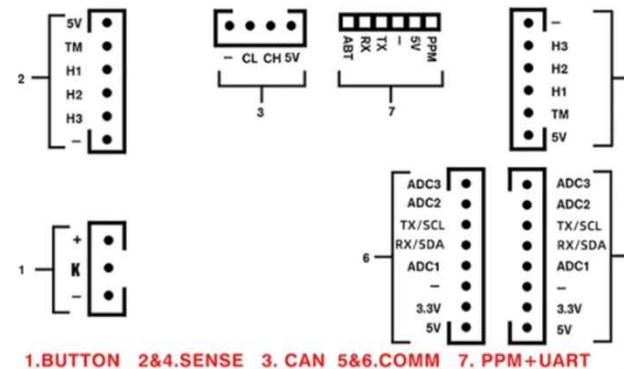
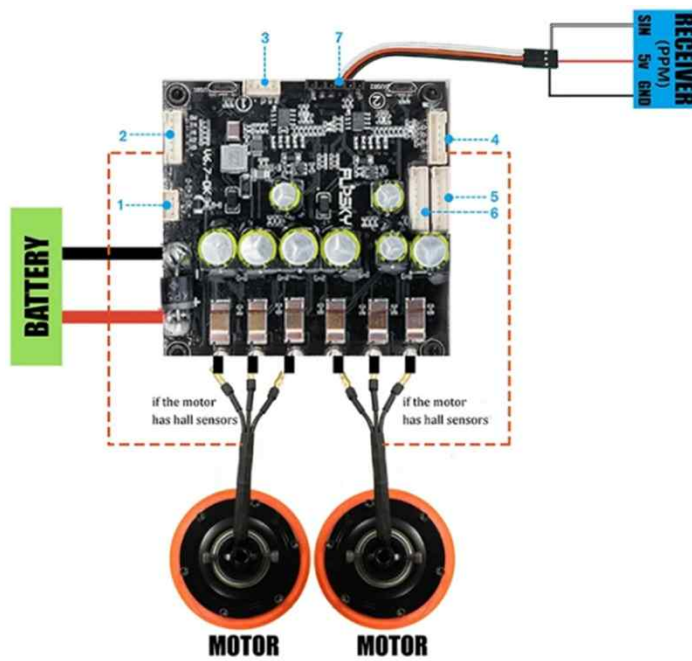
- VESC – Open Source ESC, <http://vedder.se/2015/01/vesc-open-source-esc/>
- Communicating with the VESC using UART, <http://vedder.se/2015/10/communicating-with-the-vesc-using-uart/>
- VESC – Writing Custom Applications, <http://vedder.se/2015/08/vesc-writing-custom-applications/>
- <https://dongilc.gitbook.io/openrobot-inc/tutorials/control-with-vesc-driver>



# Dual FSESC6.6 6.7 Plus based on VESC6

## ◆ Dual FSESC6.6 6.7 Plus

- used in Jetson Race Car/J



1.BUTTON 2&4.SENSE 3. CAN 5&6.COMM 7. PPM+UART

# VESC 통신 채널












## ◆ VESC 통신 채널

통신채널	사용	통신속도
USB	VESC-Tool, ROS	기본 115.2Kbps 최대 3Mbps
UART	VESC_Tool, ROS, Bluetooth	기본 115.2Kbps 최대 921.6Kbps
CAN	VESC to VESC	기본 500Kbps 최대 1Mbps
I2C (Software)	Nunchuk, IMU	기본 1Mbit/sec 최대 2Mbit/sec
SPI (Software)	DRV8301, AS5047p	Bit-banging (didn't measure yet)
SPI (Hardware)	Arduino (OPenRobot App only)	기본 4Mhz 최대 8Mhz(Clock Speed)

# MuSHR VESC

## ◆ MuSHR VESC

- <https://github.com/prl-mushr/vesc>

	schmittlema Adjust integral term on pid		c792756 on 12 May 2021	 59 commits
	vesc	Added readme, and updated metapackage vesc to contain vesc_main	3 years ago	
	vesc_ackermann	use clipped value for servo command to estimate odometry, thanks to ...	6 years ago	
	vesc_configs	Adjust integral term on pid	9 months ago	
	vesc_driver	reverse speed bugfix	9 months ago	
	vesc_main	Merge branch 'master' of <a href="https://github.com/prl-mushr/vesc">https://github.com/prl-mushr/vesc</a>	10 months ago	
	vesc_msgs	add vesc_ackermann package for translating between vesc and ackerma...	6 years ago	
	.gitignore	VESC motor controller driver boilerplate, no functionality yet	6 years ago	
	LICENSE.md	Added license	3 years ago	
	README.md	update readme	16 months ago	

# PyVESC

## ◆ PyVESC

- <https://pyvesc.readthedocs.io/en/latest/>
- Benjamin Vedder의 [VESC - Open Source ESC](http://vedder.se/2015/01/vesc-open-source-esc/) project (<http://vedder.se/2015/01/vesc-open-source-esc/>)에 사용할 수 있는 파이썬 기반 VESC 접속 프로그램

## ◆ PyVESC 설치

> pip install pyvesc

## ◆ PyVESC GitHub

- <https://github.com/LiamBindle/PyVESC>

# PyVESC 예제 프로그램

## ◆ SetDutyCycle 예제 프로그램

```
# make a SetDutyCycle message
my_msg = pyvesc.SetDutyCycle(1e5)
print(my_msg.duty_cycle) # prints value of my_msg.duty_cycle
my_packet = pyvesc.encode(my_msg)
# my_packet (type: bytes) can now be sent over your UART connection
```

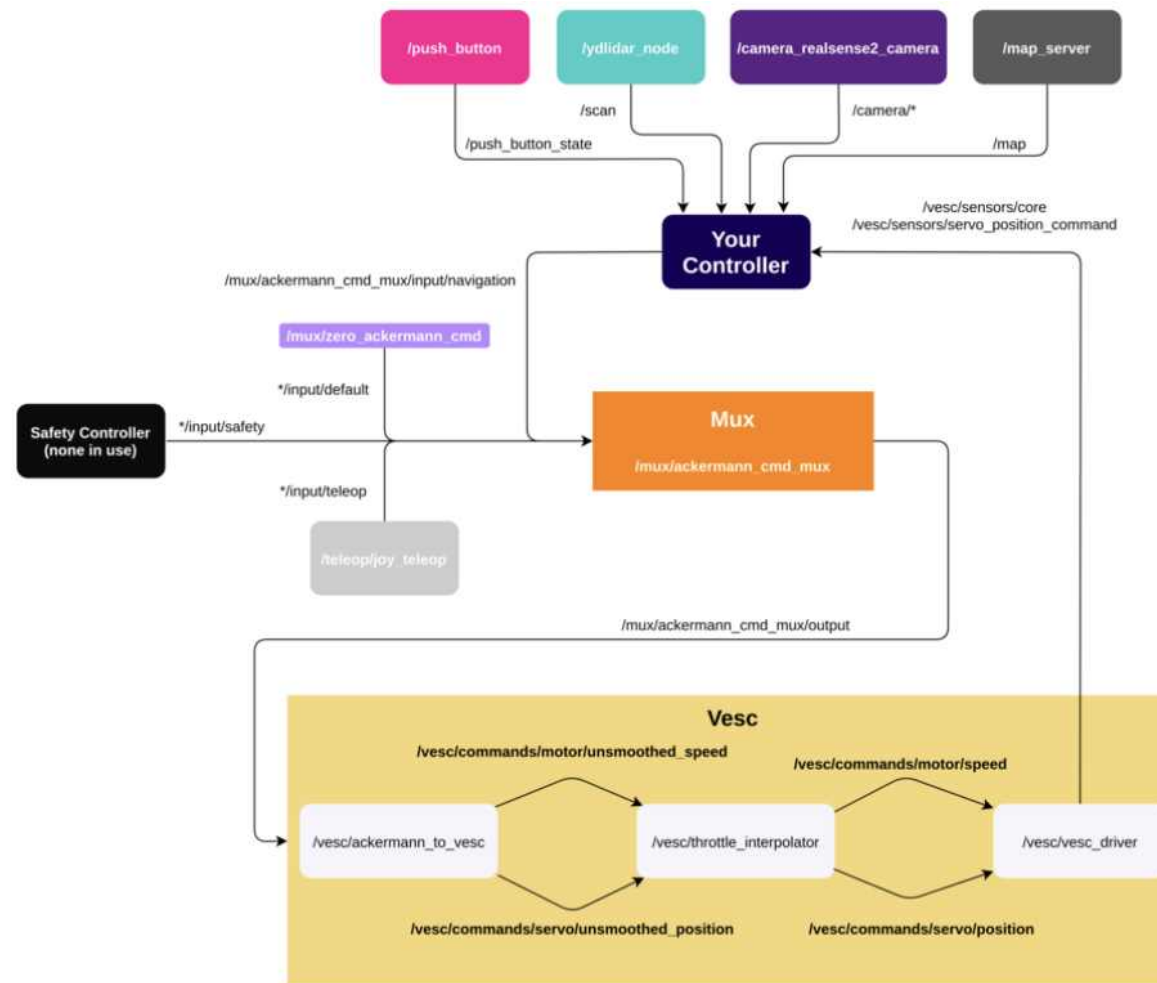
## ◆ UART 통신 채널의 버퍼로 부터 메시지 디코딩 및 수신

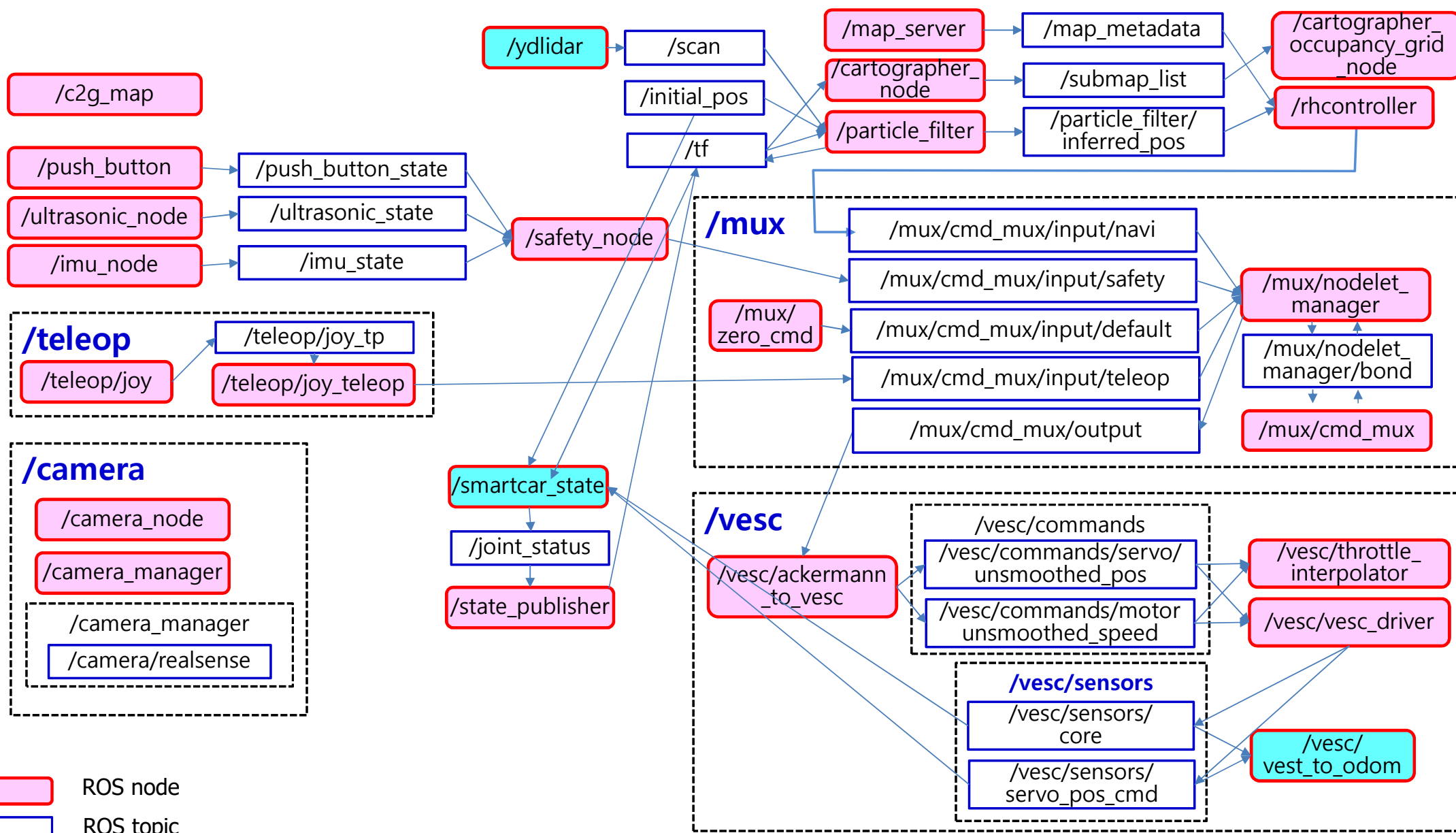
```
# buff is bytes filled from your UART connection
my_msg, consumed = pyvesc.decode(buff)
buff = buff[consumed:] # remove consumed bytes from the buffer
if my_msg:
    print(my_msg.duty_cycle) # prints value of my_msg.duty_cycle
```

자율 주행을 위한 **ROS** 노드 구성 및 실행



# 자율주행을 위한 ROS Node 구성 (MuSHR 예)





# cartographer

## ◆ install(1)

- 빌드를 위해 필요한 tool 다운로드

```
$ sudo apt-get update  
$ sudo apt-get install -y python3-wstool python3-rosdep ninja-build stow
```

- cartographer\_ros 작업공간 생성

```
$ mkdir catkin_ws  
$ cd catkin_ws  
$ wstool init src  
$ wstool merge -t src https://raw.githubusercontent.com/cartographer-project/cartographer_  
ros/master/cartographer_ros.rosinstall  
$ wstool update -t src
```

# cartographer

## ◆ install(2)

- cartographer\_ros의 종속성 설치

```
$ sudo rosdep init  
$ rosdep update  
$ rosdep install --from-paths src --ignore-src --rosdistro=${ROS_DISTRO} -y
```

- cartographer은 abseil-cpp라이브러리를 사용하기에 다음을 설치

```
$ src/cartographer/scripts/install_abseil.sh  
$ sudo apt-get remove ros-${ROS_DISTRO}-abseil-cpp
```

- 빌드 & install

```
$ catkin_make_isolated --install --use-ninja
```

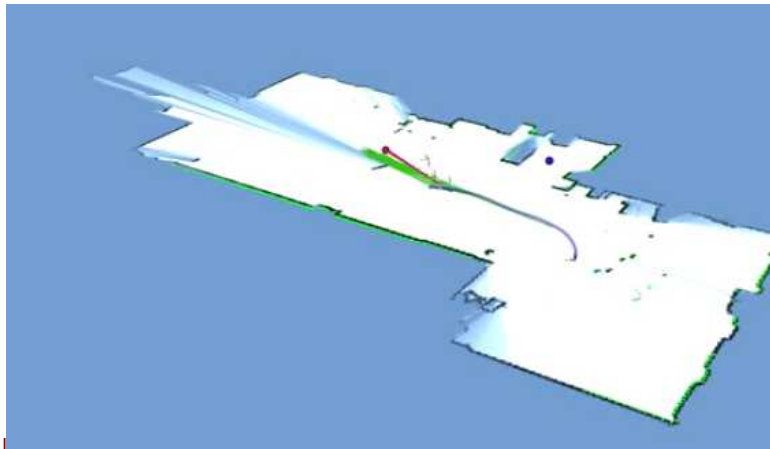
# cartographer

## ◆ demo

- 2D 데모 다운로드 및 실행:

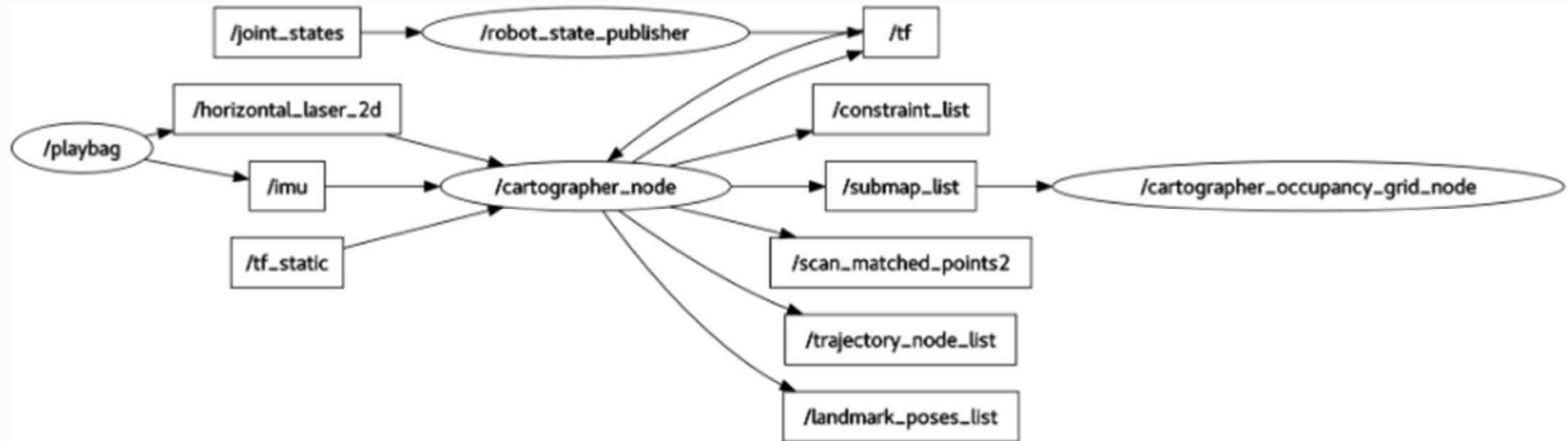
```
$ wget -P ~/Downloads https://storage.googleapis.com/cartographer-public-data/bags/backpack_2d/cartographer_paper_deutsches_museum.bag
```

```
$ roslaunch cartographer_ros demo_backpack_2d.launch  
  bag_filename:=${HOME}/Downloads/cartographer_paper_deutsches_museum.  
  bag
```



# cartographer

## ◆ Cartographer 상태도



# cartographer

## ◆ launch

```
<launch>
<!-- Urdf 설정 -->
<param name="robot_description"
  textfile="$(find cartographer_ros)/urdf/backpack_2d.urdf" />

<!-- /tf 추가 설정 -->
<node name="robot_state_publisher" pkg="robot_state_publisher"
  type="robot_state_publisher" />

<!-- cartographer node -->
<node name="cartographer_node" pkg="cartographer_ros"
  type="cartographer_node" args="
    configuration_directory $(find cartographer_ros)/configuration_files
    configuration_basename backpack_2d.lua"
  output="screen">
  <!-- Multi echo laser scan 1개 이상일때 -->
  <remap from="echoes" to="horizontal_laser_2d" />
  <!-- Multi echo laser scan 1개 이상일때 -->
  <remap from="echoes_1" to="echo_laser_2d_1" />
  <remap from="echoes_2" to="echo_laser_2d_2" />
  <!-- -->
  <!-- 만약 1개 이상의 pointcloud일때 -->
  <remap from="points2_1" to="horizontal_laser_3d" />
  <remap from="points2_2" to="vertical_laser_3d" />
  <!-- -->
</node>

<!-- Cartographer에서 나오는 map 포맷을 nav_msgs/OccupancyGrid 로 바꿔주는 노드 -->
<node name="cartographer_occupancy_grid_node" pkg="cartographer_ros"
  type="cartographer_occupancy_grid_node" args="-resolution 0.05" />
</launch>
```

1. configuration\_directory + basename  
=> cartographer를 설정해주는 lua파일 위치
2. 각 센서의 <remap>들  
=> \_n이 붙으면 여러 센서를 사용한다는 것  
odom, imu등도 추가 가능
3. Gmapping에서는 nav\_msgs/OccupancyGrid  
라는 msg type을 사용한다. 반면  
cartographer는 submap\_list를 사용하는데  
이때 이를 하나로 엮어 주어야 한다. 이때  
이를 수행하는 노드가  
cartographer\_occupancy\_grid\_node이다.

# cartographer

## ◆ Backpack\_2d.lua

```
15 include "map_builder.lua"
16 include "trajectory_builder.lua"
17
18 options = {
19   map_builder = MAP_BUILDER,
20   trajectory_builder = TRAJECTORY_BUILDER,
21   map_frame = "map",
22   tracking_frame = "base_link",
23   published_frame = "base_link",
24   odom_frame = "odom",
25   provide_odom_frame = true,
26   publish_frame_projected_to_2d = false,
27   use_pose_extrapolator = true,
28   use_odometry = false,
29   use_nav_sat = false,
30   use_landmarks = false,
31   num_laser_scans = 0,
32   num_multi_echo_laser_scans = 1,
33   num_subdivisions_per_laser_scan = 10,
34   num_point_clouds = 0,
35   lookup_transform_timeout_sec = 0.2,
36   submap_publish_period_sec = 0.3,
37   pose_publish_period_sec = 5e-3,
38   trajectory_publish_period_sec = 30e-3,
39   rangefinder_sampling_ratio = 1.,
40   odometry_sampling_ratio = 1.,
41   fixed_frame_pose_sampling_ratio = 1.,
42   imu_sampling_ratio = 1.,
43   landmarks_sampling_ratio = 1.,
44 }
45
46 MAP_BUILDER.use_trajectory_builder_2d = true
47 TRAJECTORY_BUILDER_2D.num_accumulated_range_data = 10
48
49 return options
```

- Launch 파일에서 불러오는 Cartographer 설정파일이다. default 값으로 사용해도 상관없지만, 컴퓨터 성능, 센서 성능에 따라서 조절해 주어야 함
- num\_subdivisions\_per\_laser\_scan 값을 올릴수록 스캔의 정확도가 올라가지만, 값이 커질수록 처리 속도가 느려짐
- =>laser scan 센서의 frequency가 떨어지고 로봇의 속도가 빠를 때 값을 올려줌



# cartographer

## ◆ Localization launch

```
17 <launch>
18   <param name="/use_sim_time" value="true" />
19
20   <param name="robot_description"
21     textfile="$(find cartographer_ros)/urdf/backpack_2d.urdf" />
22
23   <node name="robot_state_publisher" pkg="robot_state_publisher"
24     type="robot_state_publisher" />
25
26   <node name="cartographer_node" pkg="cartographer_ros"
27     type="cartographer_node" args="
28       -configuration_directory $(find cartographer_ros)/configuration_files
29       -configuration_basename backpack_2d_localization.lua
30       -load_state_filename $(arg load_state_filename)"
31     output="screen">
32     <remap from="echoes" to="horizontal_laser_2d" />
33   </node>
34
35   <node name="cartographer_occupancy_grid_node" pkg="cartographer_ros"
36     type="cartographer_occupancy_grid_node" args="-resolution 0.05" />
37
38   <node name="rviz" pkg="rviz" type="rviz" required="true"
39     args="-d $(find cartographer_ros)/configuration_files/demo_2d.rviz" />
40   <node name="playbag" pkg="roscpp" type="play"
41     args="--clock $(arg bag_filename)" />
42 </launch>
```

- 일반 launch 파일과 거의 동일하다
- 단, 시작할 때 localization 설정이 들어간 lua 파일과 이전에 매핑한 state를 load\_state\_filename을 파라미터로 받아와야 함
- State file은 이전 실행에서 매핑을 완료한 후 /write\_state 서비스로 생기는 pbstream 파일임

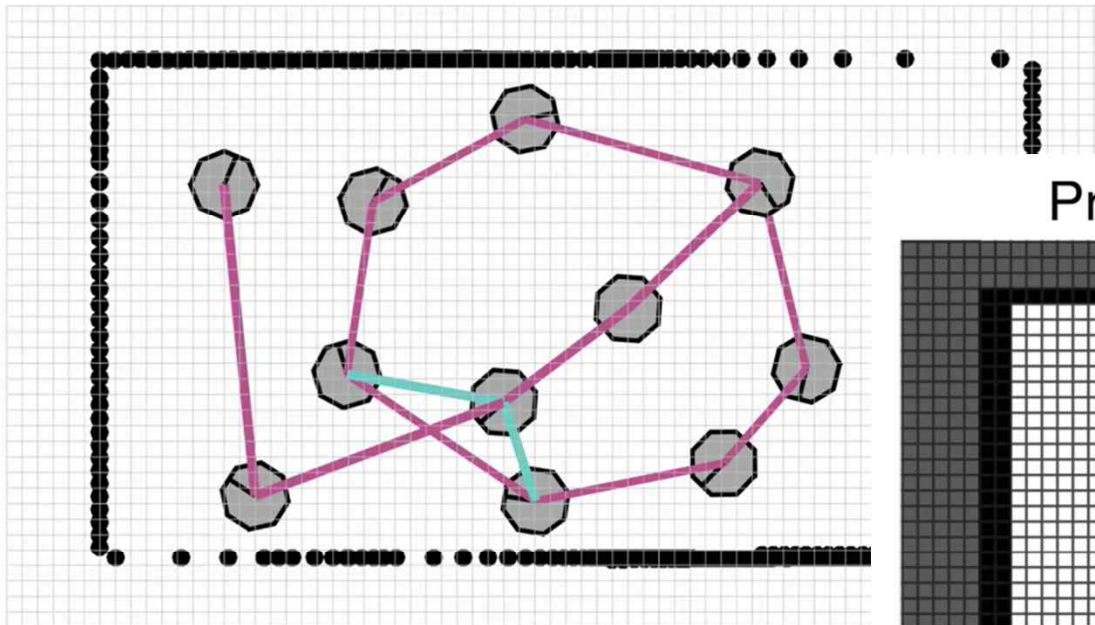
# cartographer

## ◆ 추가적으로 실행 가능한 노드

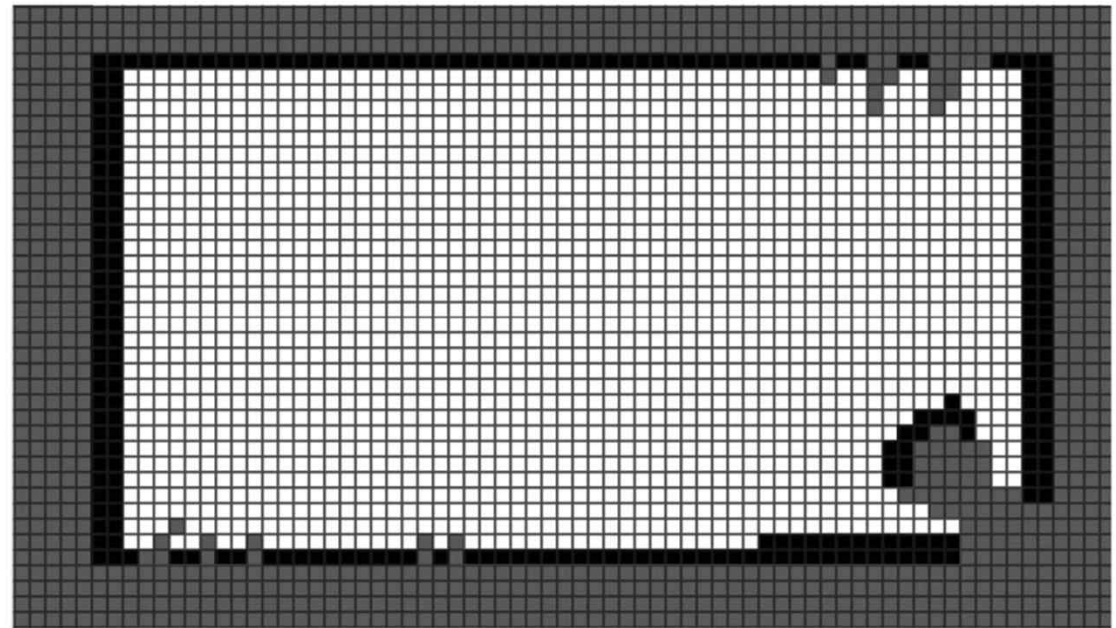
1. `cartographer_offline_node` :
  - Rosbag이 이미 생성되어 있을 때, map을 조금 더 빠르게 생성할 때 사용.
2. `Cartographer_assets_writer` :
  - rosbag과 생성된 state로 3차원 맵과 같은 포맷을 만들 때 사용하는 노드 (pbstream file + rosbag)
3. `cartographer_rosbag_validate` :
  - rosbag 데이터를 훑어서 cartographer에서 실행해도 문제가 없는 rosbag인지 체크해주는 노드
  - *rosbag*이란 ROS에서 제공하는 로깅 툴로써, 센서에 들어오는 모든 메시지들을 시간 순서에 맞게 저장하고 재생할 수 있게 해준다.

# SLAM 구동, Map 작성

Binary Occupancy Grid











Probabilistic Occupancy Grid



# ROS Interfaces for VESC Motor Drivers

## ◆ ROS Interfaces for VESC Motor Drivers

- <https://github.com/sbgisen/vesc>
- vesc : a meta package to manage the other packages
- vesc\_driver : the main library to drive VESCs
- vesc\_hw\_interface wraps vesc\_driver as a hardware interface to use ros\_control systems
- vesc\_msgs defines messages to publish VESC status

 .github	Add GitHub templates	8 months ago
 vesc	Fix license notations	2 years ago
 vesc_driver	[vesc_driver] Fix bugs depending on an implementation-defined behavior	2 months ago
 vesc_hw_interface	Merge pull request #29 from sbgisen/feature/use_ros_time_in_servo	7 months ago
 vesc_msgs	[vesc_driver] Correct velocity units with the # of motor pole pairs	8 months ago
 .gitignore	Update .gitignore	2 years ago
 LICENSE	fix License	2 years ago
 README.md	Update README	2 years ago

## 자율 주행 기능 시험

## 직진, 후진 기능 시험

### ◆ Raspberry Pi 또는 Jetson Nano에서 파이썬 프로그램으로 VESC를 접속

- 자율 주행 자동차 플랫폼에서 지원하는 접속 기능 사용
- 파이썬 프로그램으로 원격 통신 기능 구성

### ◆ VESC를 통하여 주행 기능 시험

- 조향을 위한 Servo Motor의 구동 및 회전 각도 점검
- 주행을 위한 BLDC Motor의 구동, 전진 및 후진 회전 속도 점검

# SLAM 및 Path Finding 기능 시험

## ◆ LiDAR 기반의 SLAM 기능 시험

- Map 생성 기능 시험
- Localization 기능 시험

## ◆ Path Finding 기능

- 경로 탐색 기능 시험
- 경로에 따른 주행 기능 시험

## ◆ 장애물 회피 기능 시험

- 경로상의 장애물 식별 및 회피 기능

## 참고문헌

- [1] <https://docs.px4.io/master/ko/>.
- [2] Sebastian Thrun, Wolfram Burgard and Dieter Fox, Probabilistic Robotics, MIT press, August 2005.
- [3] 시각 관성 주행거리 측정(VIO), [https://docs.px4.io/master/ko/computer\\_vision/visual\\_inertial\\_odometry.html](https://docs.px4.io/master/ko/computer_vision/visual_inertial_odometry.html).
- [4] Pixhawk와 ROS를 이용한 자율주행 드론, <https://dnddnjs.gitbooks.io/drone-autonomous-flight/content/>.
- [5] Drone Programming With Python Course | 3 Hours | Including x4 Projects | Computer Vision, <https://www.youtube.com/watch?v=LmEcyQnfpDA>.
- [6] <https://google-cartographer.readthedocs.io/en/latest/>
- [7] <https://ichi.pro/ko/google-cartographer-mich-rplidarwa-raspberry-pileul-sayonghan-2d-maeping-186147659465925>
- [8] [https://github.com/cartographer-project/cartographer\\_ros](https://github.com/cartographer-project/cartographer_ros)
- [9] <https://elecs.tistory.com/296>
- [10] <https://linklab-uva.github.io/autonomoustracing/assets/files/SLAM.pdf>
- [11] 오로카 cartographer 강연 20201201 – YouTube
- [12] [https://google-cartographer-ros.readthedocs.io/en/latest/assets\\_writer.html](https://google-cartographer-ros.readthedocs.io/en/latest/assets_writer.html)
- [13] 강화학습을 이용한 자율주행 구현, [https://github.com/NOHYC/autonomous\\_driving\\_car\\_project](https://github.com/NOHYC/autonomous_driving_car_project).
- [14] Mini FSESC4.20 50A base on VESC® 4.12 with Aluminum Anodized Heat Sink, <https://flipsky.net/products/mini-fsesc4-20-50a-base-on-vesc-widely-used-in-eskateboard-escooter-ebike>.
- [15] 미니 FSESC4.20 50A 베이스, VESC®알루미늄 알루미늄 방열판 Flipsky 4.12, Ali Express, KRW 118,211, [https://ko.aliexpress.com/item/4000438827676.html?gatewayAdapt=glo2kor&spm=a2g0o.search0302.0.0.4c2fb920WxB9Ld&algo\\_pvid=4f45efd9-8eb9-41f3-ac5d-29d1bd1c3155&algo\\_exp\\_id=4f45efd9-8eb9-41f3-ac5d-29d1bd1c3155-4](https://ko.aliexpress.com/item/4000438827676.html?gatewayAdapt=glo2kor&spm=a2g0o.search0302.0.0.4c2fb920WxB9Ld&algo_pvid=4f45efd9-8eb9-41f3-ac5d-29d1bd1c3155&algo_exp_id=4f45efd9-8eb9-41f3-ac5d-29d1bd1c3155-4)



## 참고문헌

- [16] Get Your Motor Running! - VESC - Jetson RACECAR, <https://www.youtube.com/watch?v=fiaiA-o83c4>
- [17] JetsonHacks, Jetson Race Car/J, <https://www.jetsonhacks.com/racecar-j/>.
- [18] Understanding A\* Path Algorithms and Implementation with Python, <https://towardsdatascience.com/understanding-a-path-algorithms-and-implementation-with-python-4d8458d6ccc7>.
- [19] Python Implementation of A\* Algorithm, <https://github.com/ademakdogan/Implementation-of-A-Algorithm-Visualization-via-Pyp5js>.
- [20] Path Planning with A\* and RRT | Autonomous Navigation, Part 4, <https://www.youtube.com/watch?v=QR3U1dgc5RE>.
- [21] RRT STAR | RRT\* | Path Planning Algorithm | Python Code, <https://www.youtube.com/watch?v=M5Q6Fywd36w>.
- [22] RRT, RRT\* & Random Trees, <https://www.youtube.com/watch?v=Ob3BIJkQJEw>.
- [23] Python implementation of Rapidly-exploring random tree (RRT) path-planning algorithm, <https://gist.github.com/Fnjin/58e5eaa27a3dc004c3526ea82a92de80>.
- [24] The RRT path planning algorithm simulated with python | part 1, <https://www.youtube.com/watch?v=TzfNzqjJ2VQ>.
- [25] The RRT path planning algorithm simulated with python | part 2, <https://www.youtube.com/watch?v=JpKkfWxbqgg>.
- [26] The RRT path planning algorithm simulated with python | part 3, <https://www.youtube.com/watch?v=BHG8VKwEPuw>.
- [27] The RRT path planning algorithm simulated with python | part 4, <https://www.youtube.com/watch?v=vAoDnjgIVKU>.
- [28] RRT\* FND - motion planning in dynamic environments, <https://www.youtube.com/watch?v=hXTnWN8NiKE>.
- [29] Easy pathfinding in python [almost without math], [https://www.youtube.com/watch?v=8SigT\\_jhz4I](https://www.youtube.com/watch?v=8SigT_jhz4I).
- [30] Motion Planning Algorithms (RRT, RRT\*, PRM) - [MIT 6.881 Final Project], [https://www.youtube.com/watch?v=gP6MRe\\_IHFo](https://www.youtube.com/watch?v=gP6MRe_IHFo).
- [31] Dynamic RRT (demo), [https://www.youtube.com/watch?v=TA0\\_0Zb7cjE](https://www.youtube.com/watch?v=TA0_0Zb7cjE).

