

LIMO Start Manual

WeGo & LIMO

목 차

1. LIMO
2. LIMO 기능 소개
3. LIMO Mode Switching
4. LIMO 하드웨어 사용 방법
5. LIMO 소프트웨어 사용 방법

01

LIMO

01 LIMO

- 소개

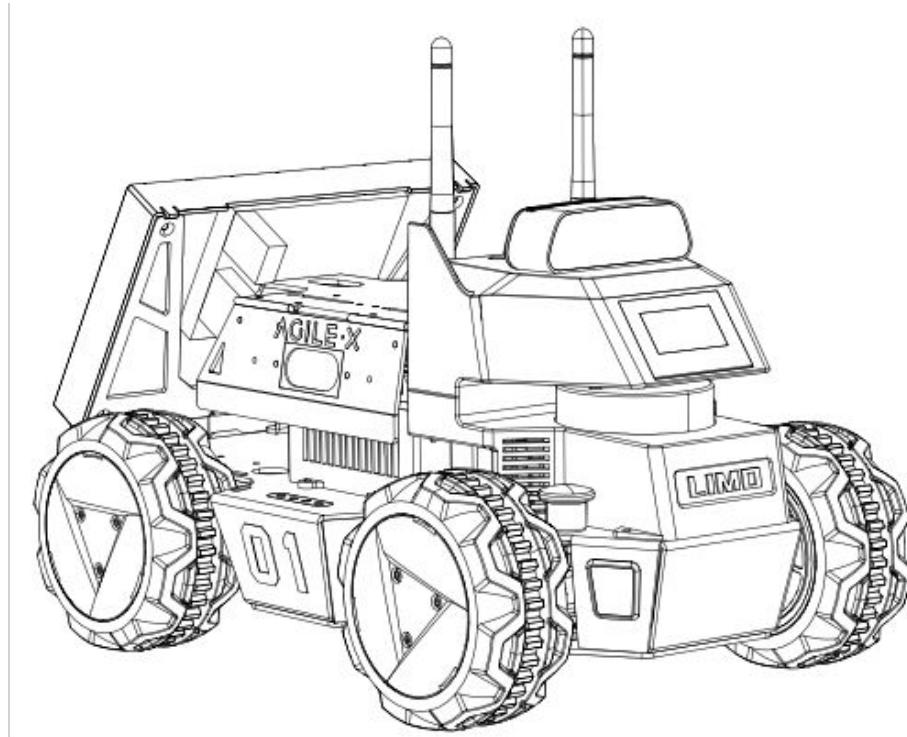
- LIMO는 4가지 바퀴 타입을 제공하는 세계 최초의 ROS 개발 플랫폼입니다.
- LIMO는 로봇 교육 및 연구 개발에 적합한 플랫폼입니다.
- LIMO는 4-Wheel Drive Mode, Ackermann Mode, Track-type Mode, Mecanum wheel Mode의 4가지 움직임 모델을 제공합니다.
- LIMO는 Jetson Nano, EAI X2L LiDAR, Depth Camera, IMU가 장착되어있습니다.
- LIMO는 SLAM mapping, Path Planning, Autonomous Navigation with Obstacle Avoidance 등의 기능을 제공합니다.

01 LIMO

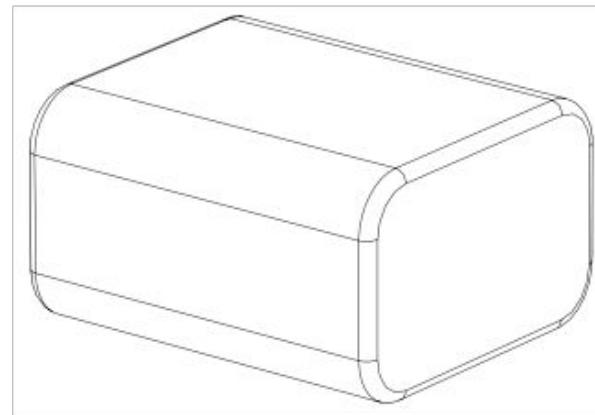
- 구성품
 - LIMO high-end body
 - Battery
 - Charger
 - Mecanum wheel x 4
 - Track x 2
 - Cross screwdriver
 - Screw (M3x12mm 37, M3x5mm 20)

01 LIMO

- LIMO high-end body (off-road wheel 4개 포함, 배터리 1개 내부 장착)



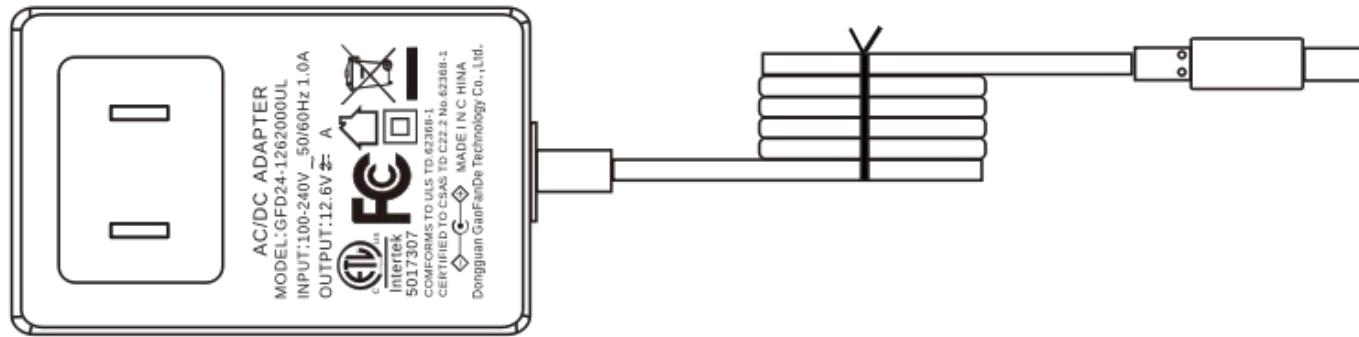
LIMO high-end body *1 (install off-road wheel *4)



Battery *1

01 LIMO

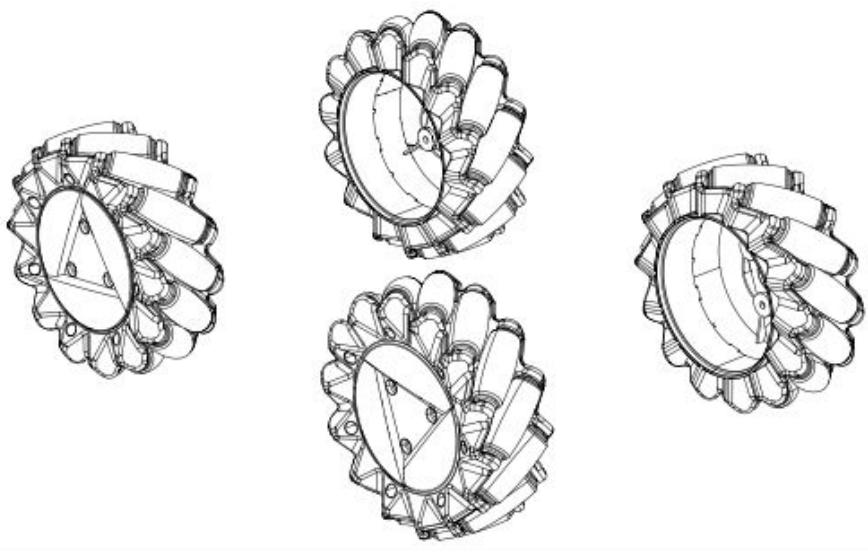
- Charger



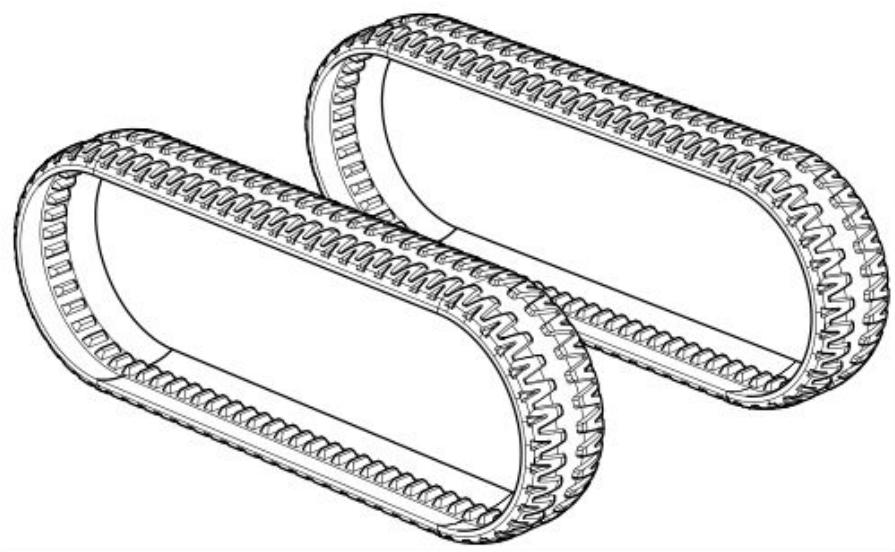
Charger *1

01 LIMO

- Mecanum wheel 47H
- Track 27H



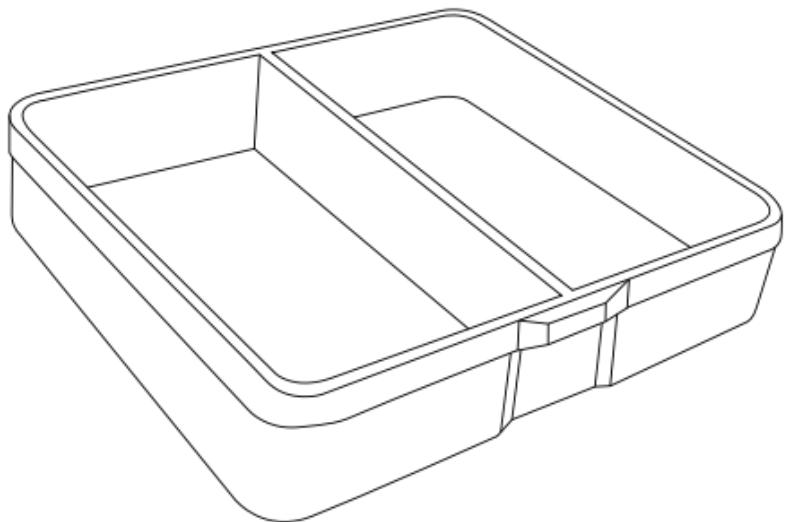
Mecanum wheel *4



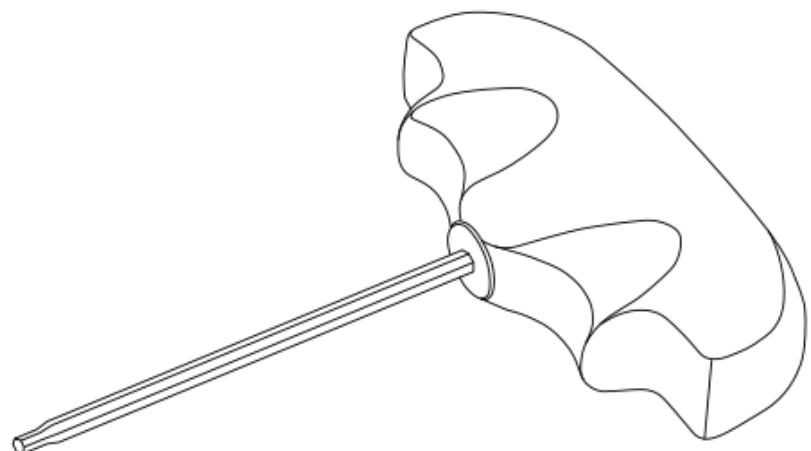
Track *2

01 LIMO

- **Screw Box 17H (M3x12mm 37H, M3x5mm 20H)**
- **Allen Driver 17H**



Screw box *1 , include:
M3x12mm , 20pcs ; M3x5mm , 20pcs



allen driver *1

01 LIMO

• Hardware Specifications

항목	파라미터	값
Mechanical	전체 크기	322 x 220 x 251 mm
	축거 (앞, 뒷 바퀴 사이의 거리)	200 mm
	윤거 (앞 바퀴 틀 사이의 거리)	175 mm
	무게	4.8 kg
	최대 부하	1kg (4-Wheel differential) 4kg (Ackermann mode) 4kg (Mecanum wheel)
	최저 지상고 (바퀴 접지면과 플랫폼 하부 사이의 거리)	24mm
Drive type	Hub motor (4x14.4W)	

01 LIMO

- Hardware Specifications

항목	파라미터	값
Performance	무부하 최고 속도	1 m/s
	Ackermann 최소 회전 반경	0.4 m
	동작 온도	-10 ~ +40 degrees Celsius
	최대 등반 각도	40도 (Track mode AI)
System	Power interface	DC (5.5 x 2.1mm)
	OS	Ubuntu 18.04
	CPU	ARM 64-bit 4-core @ 1.43GHz (Cortex-A57)
	GPU	128-core nVidia Maxwell @921MHz
	배터리	5200mAh 12V
	구동 시간	40분
	대기 시간	2시간
	통신 인터페이스	Wifi, Bluetooth

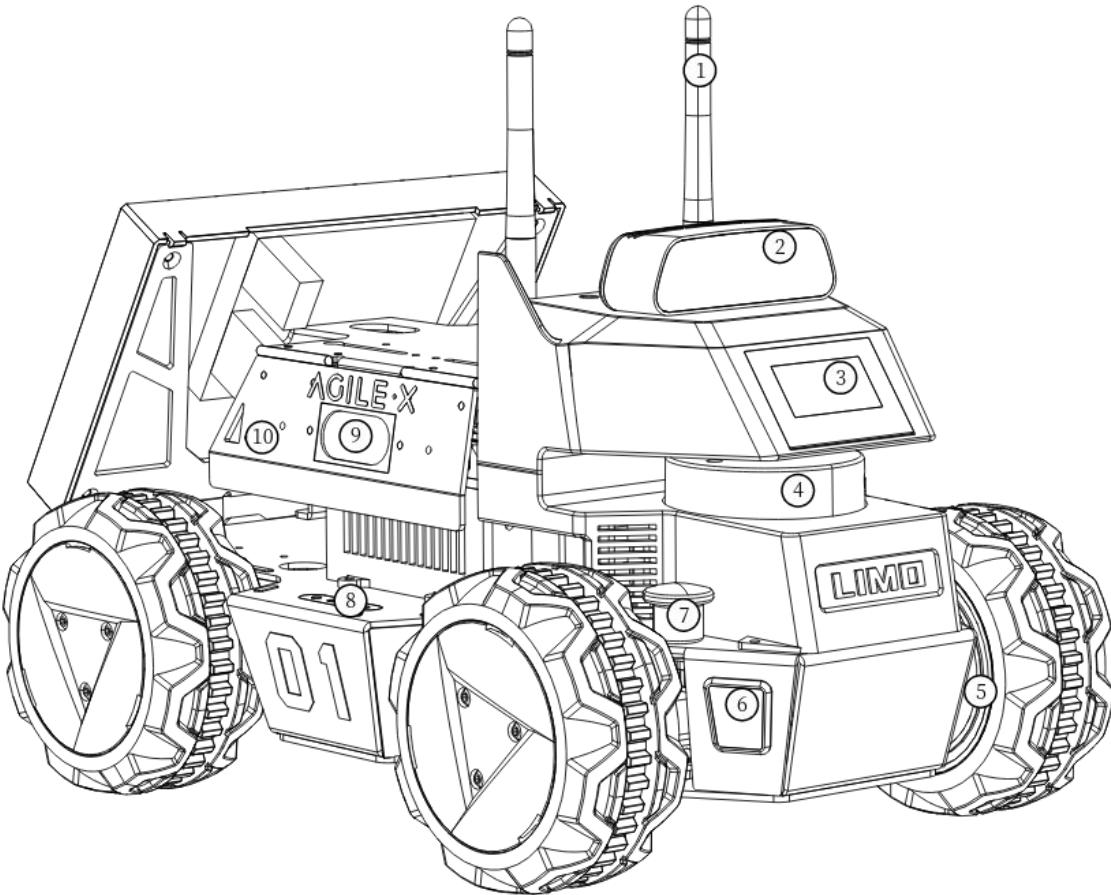
01 LIMO

- Hardware Specifications

항목	파라미터	값
Sensor	LiDAR	EAI X2L
	Depth Camera	DaBai / RealSense D435
	IPC	Nvidia Jetson Nano (4G)
	IMU	MPU6050
	Voice module	iFlytek Voice Assistant / Google Assistant
	Speaker	Left and right dual channels (2x2W)
	USB-Hub	Type-C x1, USB2.0 x2
	Front Display	1.54 inch 128x64 white OLED display screen
	Rear Display	7 inch 1024x600 IPS touch screen
Control	Conctrmode	Mobile APP, Command Control
	Mobile APP	Bluetooth, maximum distance 10m

01 LIMO

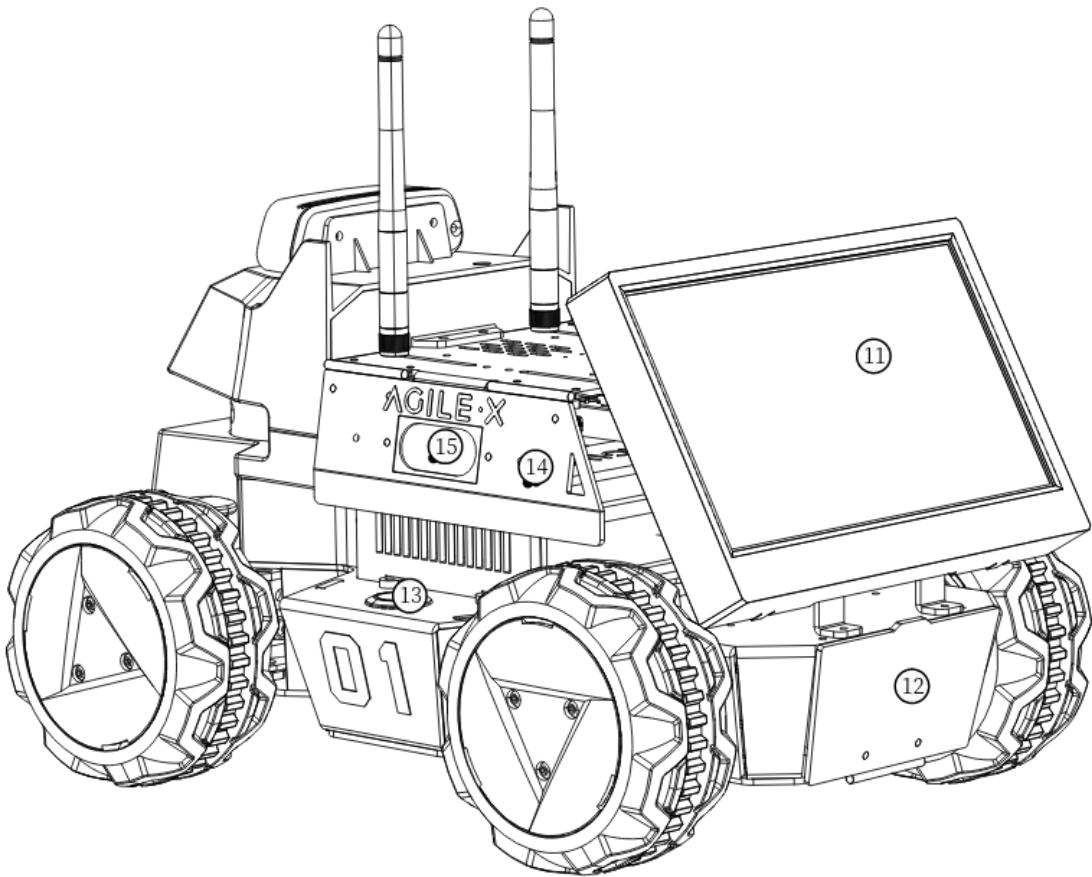
• 각 부분 명칭



1	WiFi/Bluetooth antenna
2	Depth Camera
3	Front Display
4	EAI X2L LiDAR
5	Hub motor
6	RGB light
7	Four-wheel differential/Ackermann mode switching latch
8	Power display
9	Left speaker
10	Left seagull door

01 LIMO

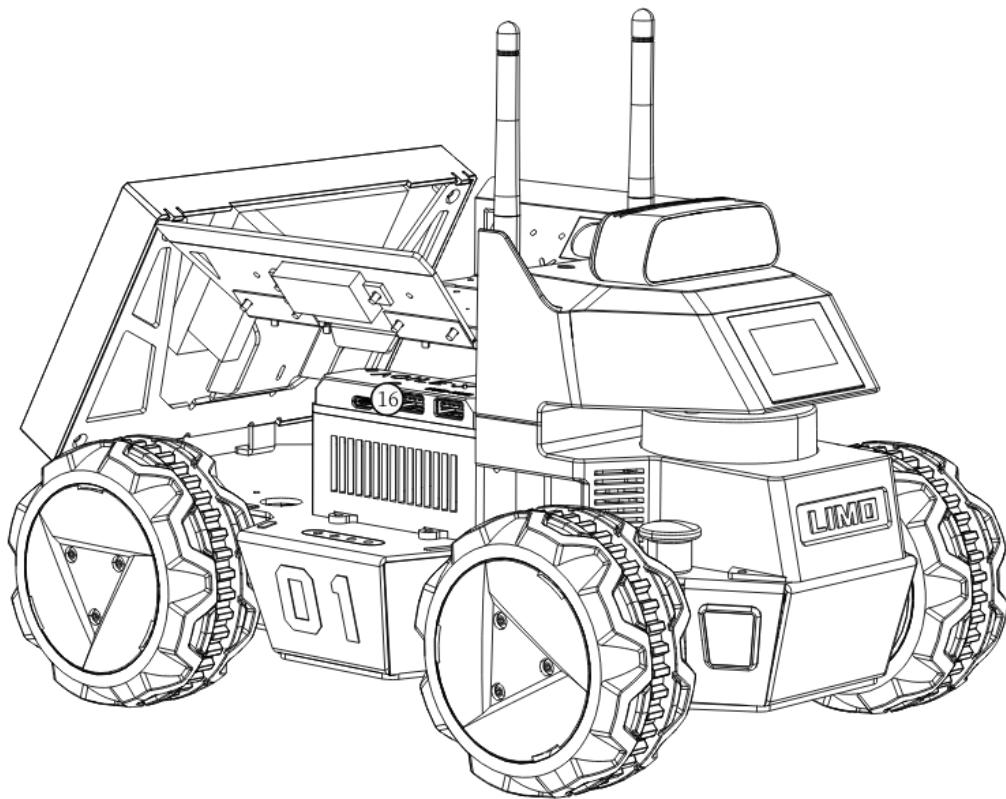
- 각 부분 명칭



11	Rear display
12	Battery door
13	Switch
14	Right seagull door
15	Right speaker

01 LIMO

- 각 부분 명칭

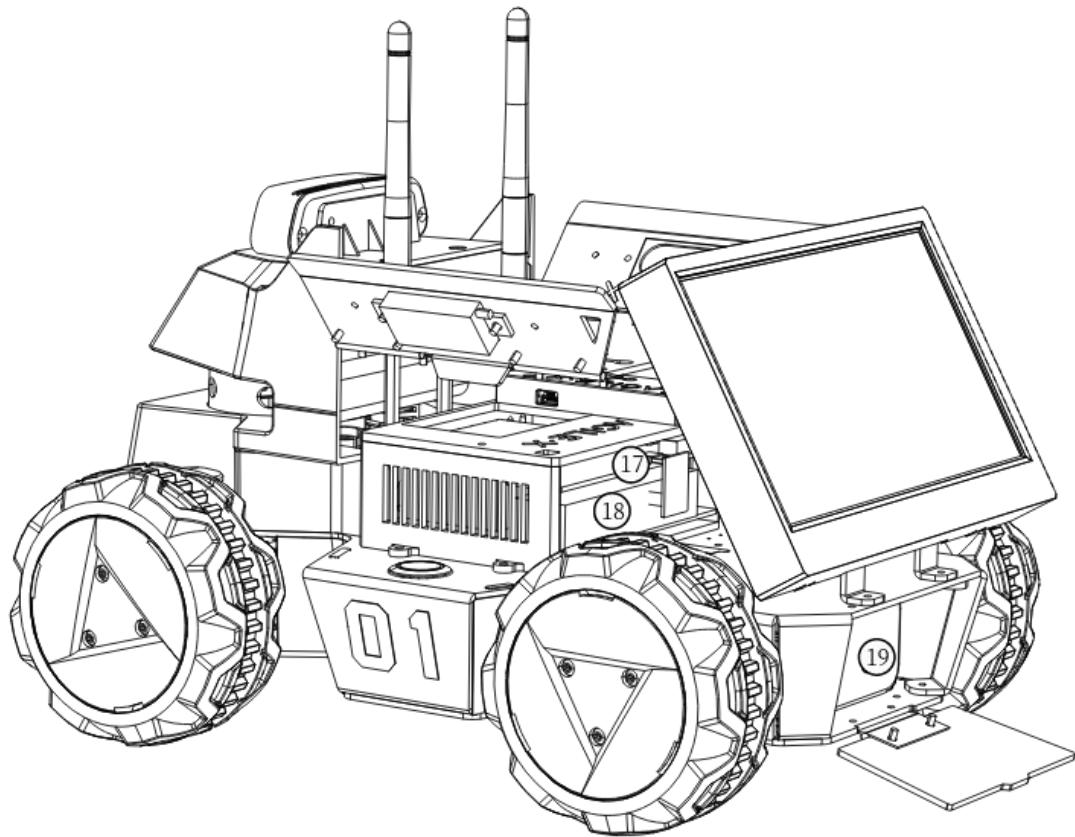


16

USB-HUB

01 LIMO

- 각 부분 명칭



17	Voide module
18	IPC Nvidia Jetson Nano (4G)
19	Battery

02

LIMO
기능 소개

02 LIMO 기능 소개

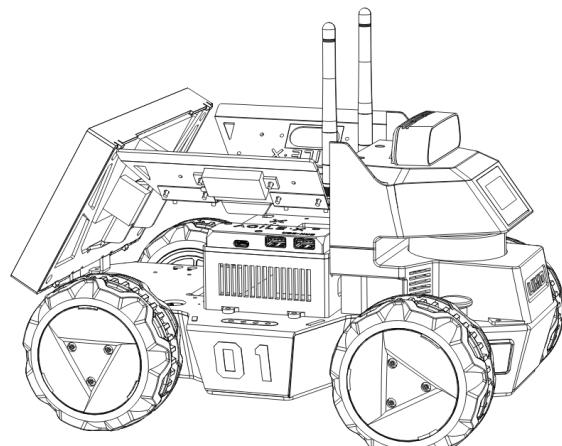
- LIMO는 내부 공간을 적절히 활용하기 위해, 4개의 Hub motor를 사용하고 있으며. 간편하게 Ackermann, 4-Wheel Differential, Track, Mecanum으로 움직임 모드를 변환할 수 있습니다.
- Ackermann Drive mode
 - 차량과 유사한 형태로 회전하며, 전방 조향, 후방 구동을 담당
 - 전방 바퀴의 조향에 따라 다른 크기의 원 궤적을 그리며 구동
 - 회전 시, 양쪽 바퀴의 각도가 바깥쪽 바퀴의 각도보다 약 2~4도 정도 크게 만들어서, 차량이 원활하게 회전할 수 있도록 합니다.
- Four-wheel differential mode
 - 제자리 회전이 가능한 형태이지만, 이로 인해 바퀴의 마모가 심해질 수 있음
 - 제자리 회전을 자주 사용하지 않는 것이 좋음

02 LIMO 기능 소개

- LIMO는 내부 공간을 적절히 활용하기 위해, 4개의 Hub motor를 사용하고 있으며. 간편하게 Ackermann, 4-Wheel Differential, Track, Mecanum으로 움직임 모드를 변환할 수 있습니다.
- Track mode
 - 오프로드에서 좋은 성능을 보이며, 최대 40도의 기울기 및 작은 계단 등반이 가능
- Mecanum wheel mode
 - Omni-directional motion이 가능해지며, 전후좌우로 움직일 수 있으며, 제자리 회전 등의 여러 동작이 가능한 형태

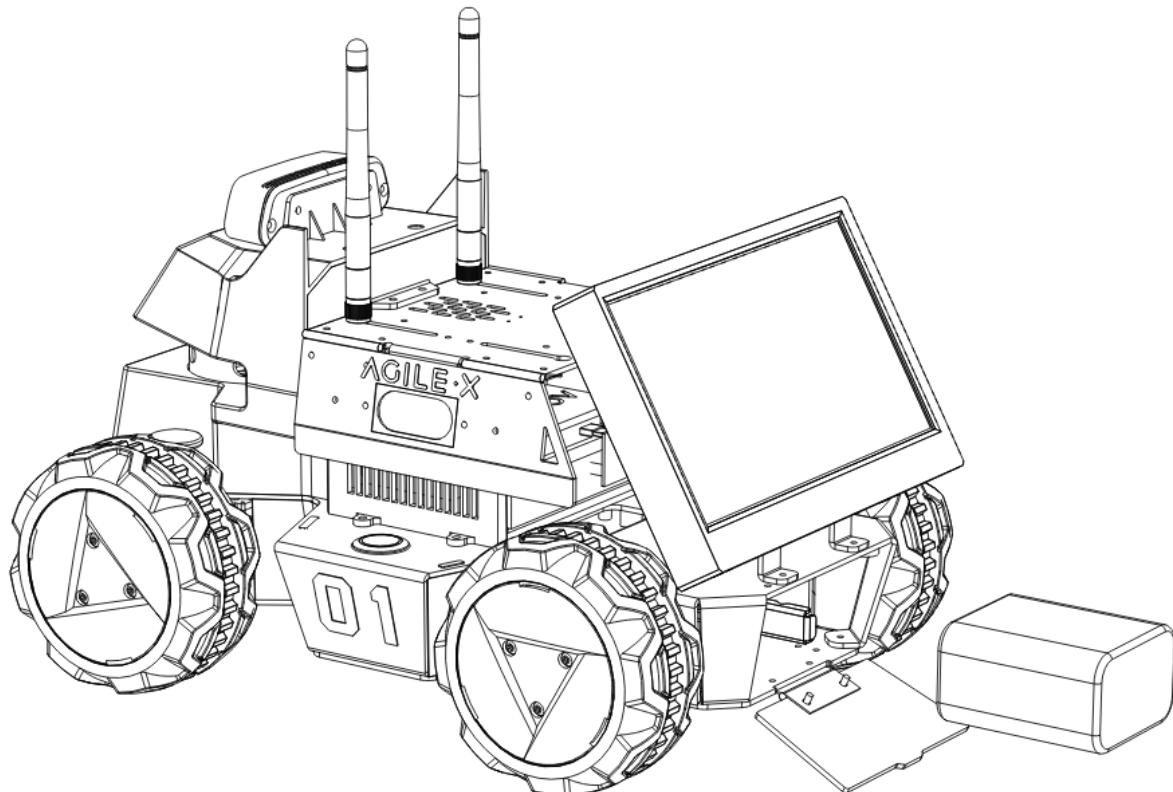
02 LIMO 기능 소개

- LIMO LED 소개
 - LIMO 전방에 있는 LED를 의미하며, 5개의 색상으로 현재의 상태를 표시
 - 적색 점멸 - 배터리 부족
 - 적색 - Software가 정지
 - 녹색 - Ackermann Mode
 - 황색 - Four-wheel differential or Track mode
 - 청색 - Mecanum wheel mode
- LIMO 우측의 문을 열어서, USB Type C 그리고 2개의 USB 2.0을 연결할 수 있습니다.



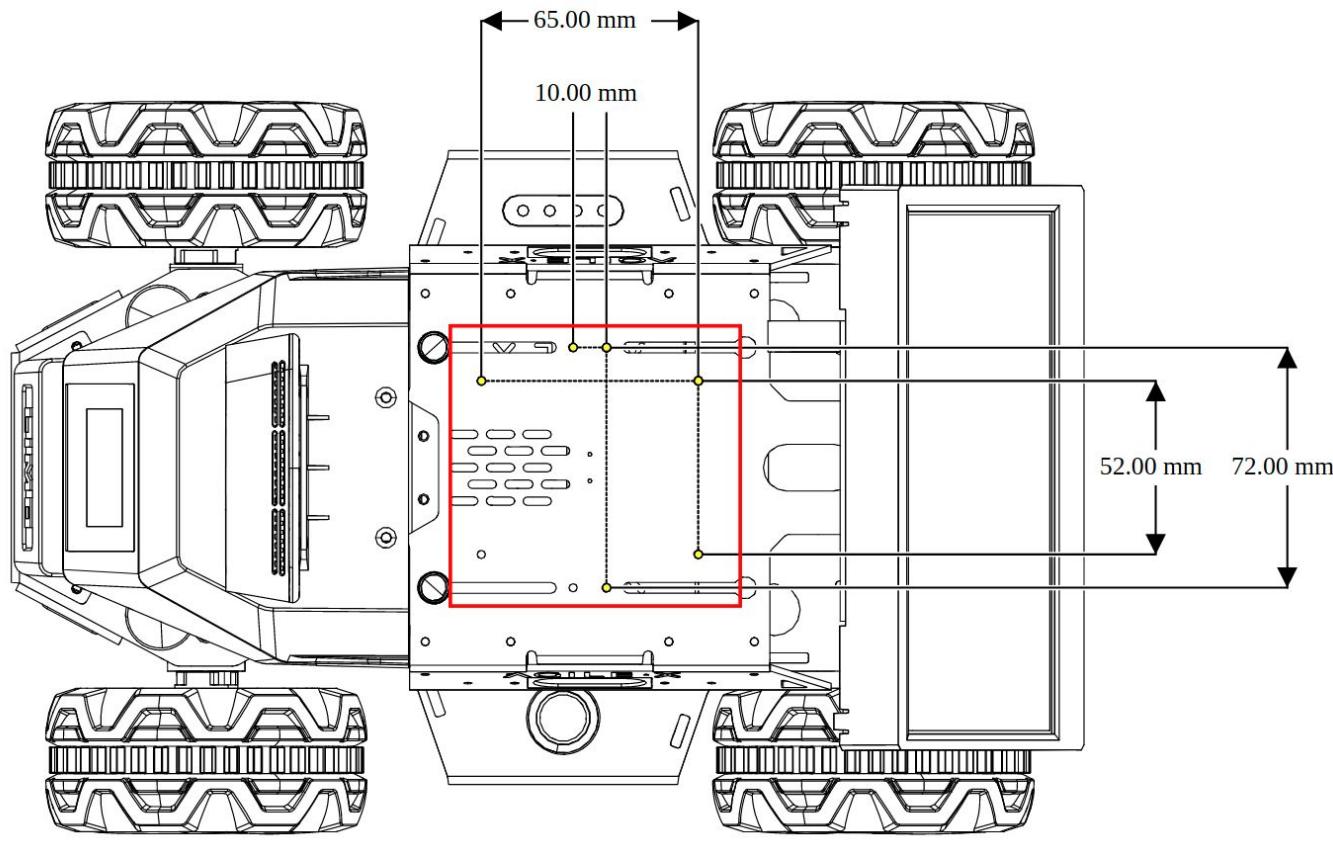
02 LIMO 기능 소개

- LIMO 배터리
 - 배터리는 후방에 위치하며, 여러개가 있을 경우, 제거 후 교체가 가능합니다.



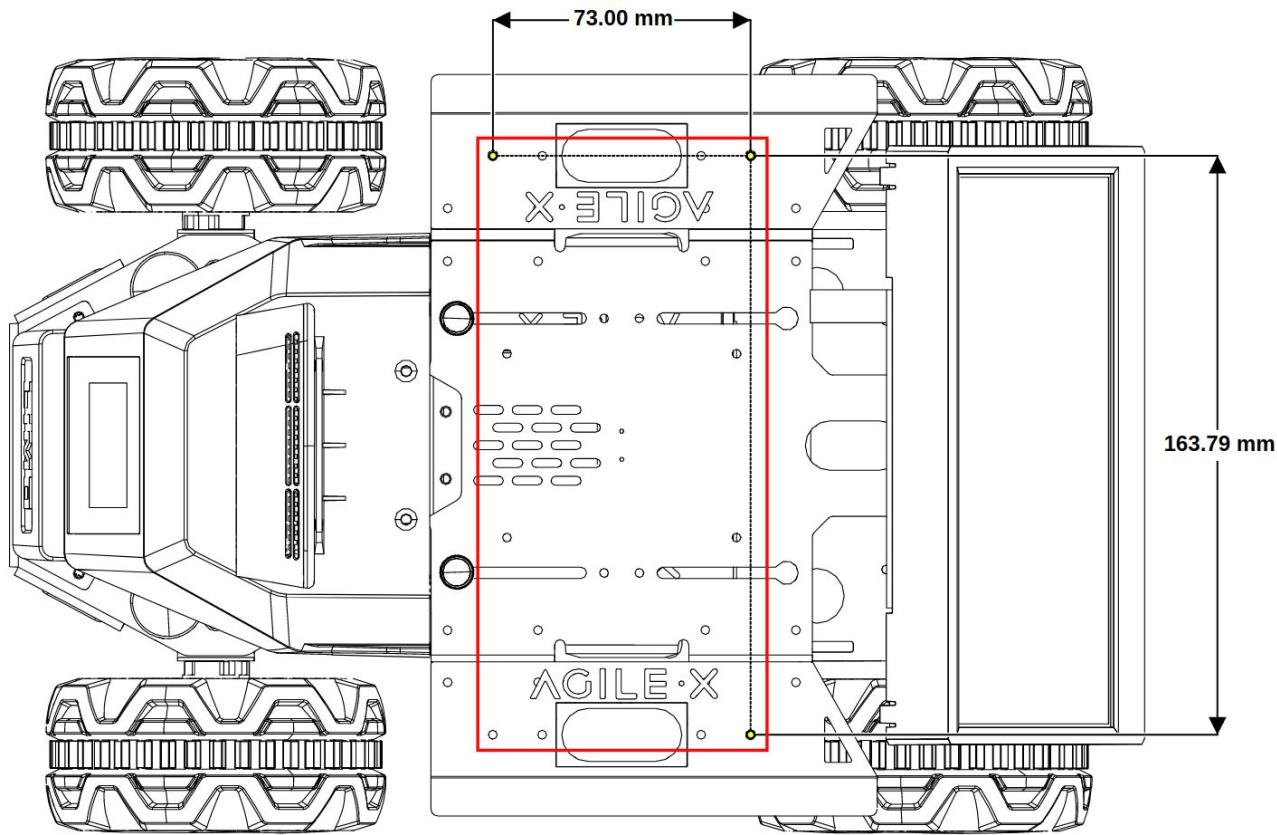
02 LIMO 기능 소개

- 확장을 위한 나사 구멍 존재
 - LIMO 상단에 확장을 위한 나사 구멍이 존재하며, 이를 위한 나사도 포함되어 있습니다.



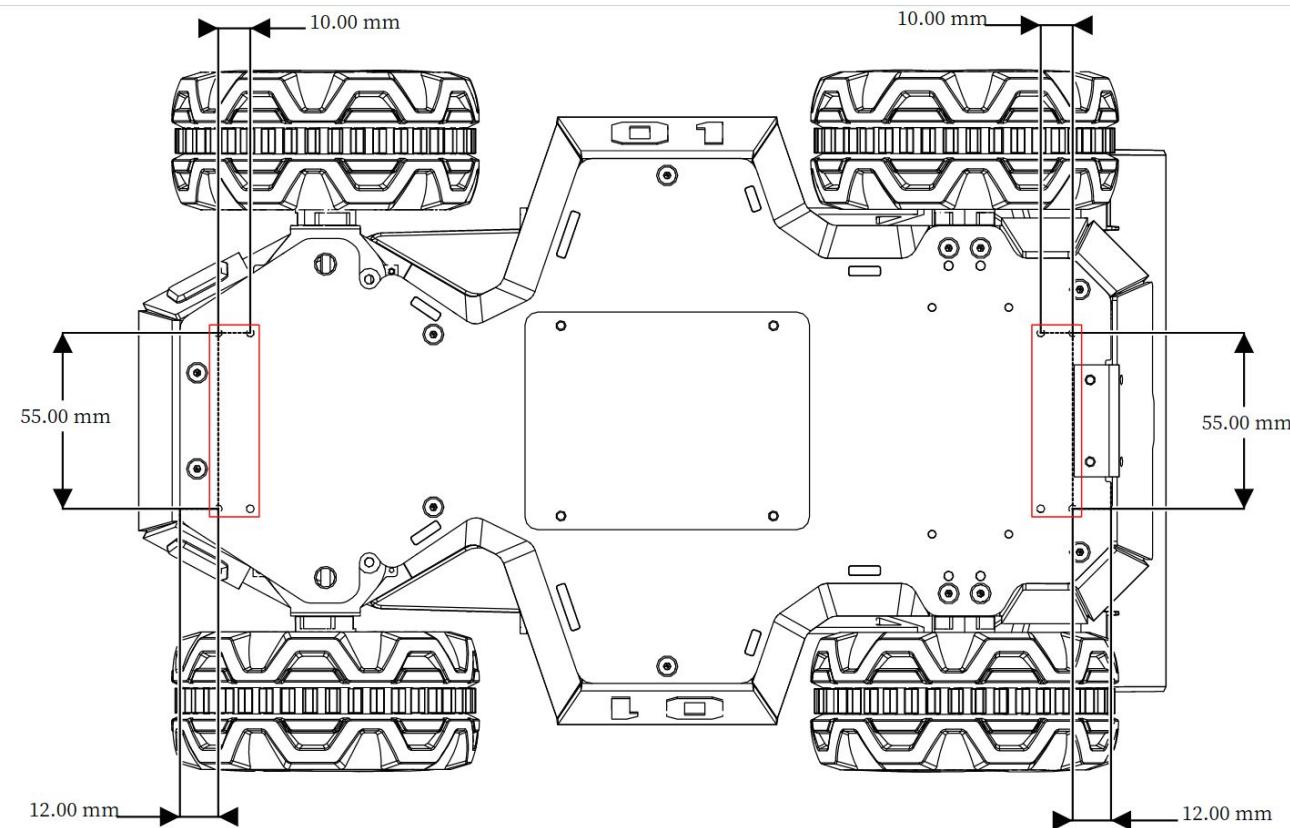
02 LIMO 기능 소개

- 확장을 위한 나사 구멍 존재
 - LIMO 상단에 확장을 위한 나사 구멍이 존재하며, 이를 위한 나사도 포함되어 있습니다.



02 LIMO 기능 소개

- 확장을 위한 나사 구멍 존재
 - LIMO 상단에 확장을 위한 나사 구멍이 존재하며, 이를 위한 나사도 포함되어 있습니다.



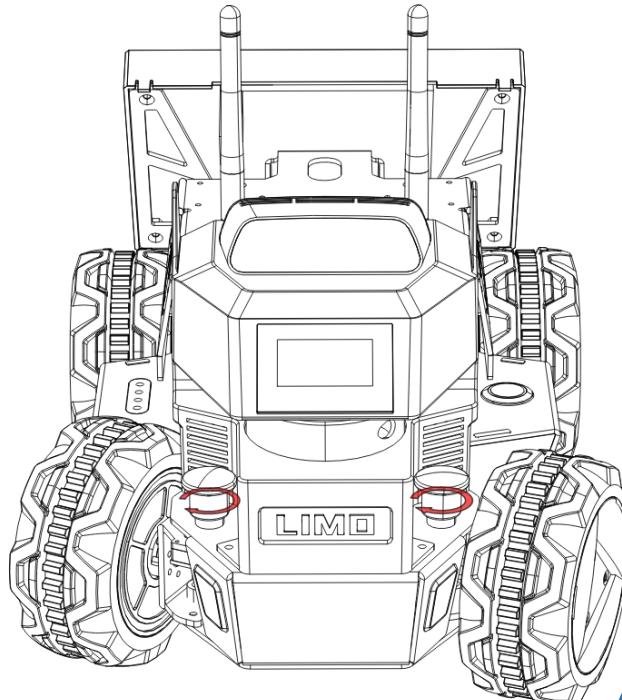
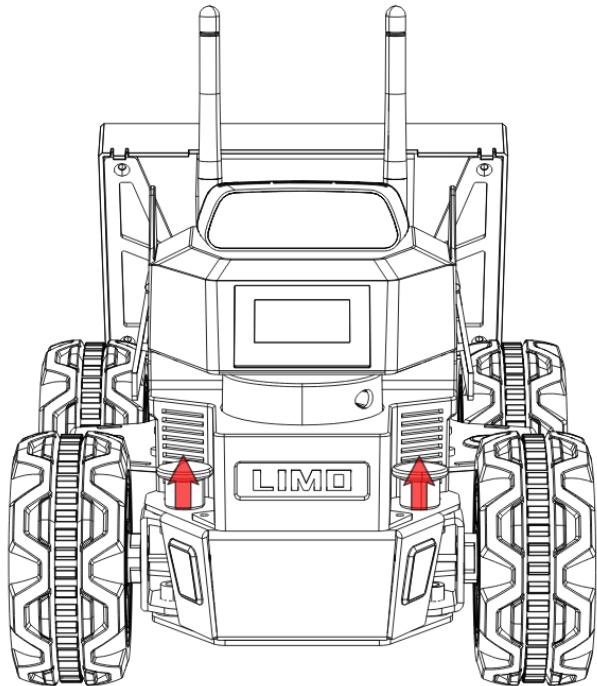
03

LIMO Mode Switching

03 LIMO Mode Switching

- Ackermann mode

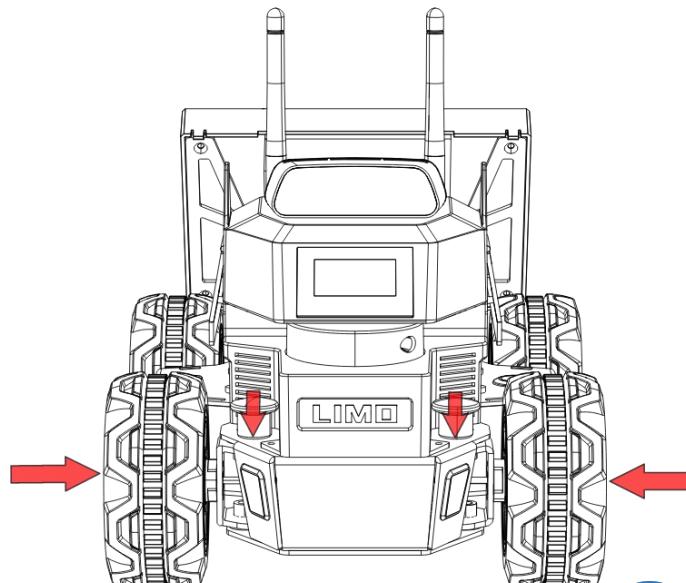
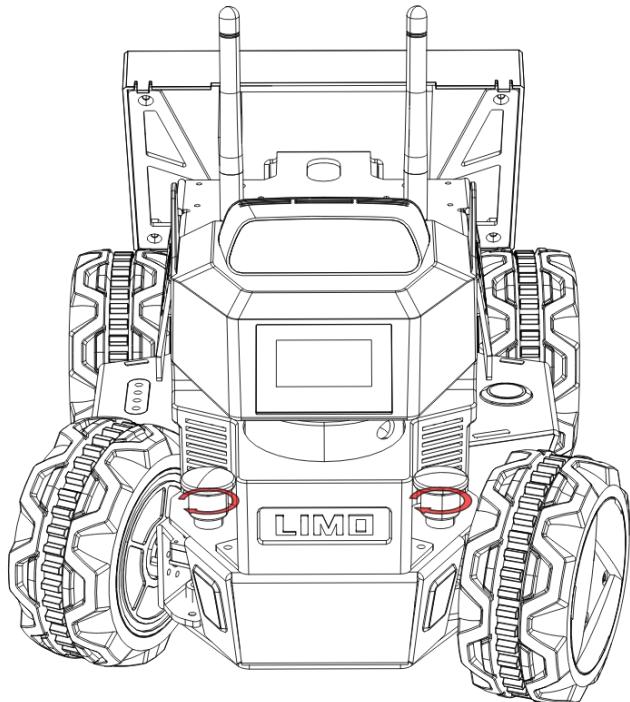
- 전방에 위치한 빨간색 걸쇠()를 올린 후, 약 30도 정도 회전 시켜서 내려가지 않도록 합니다.
- 차량 정면에 있는 LED가 녹색으로 변경되며 변경이 성공합니다.



03 LIMO Mode Switching

- Differential mode

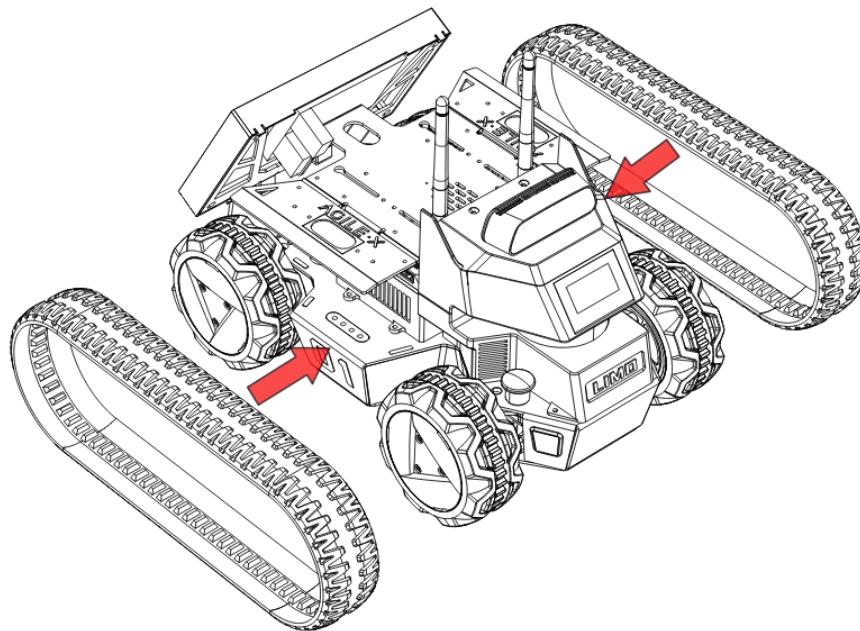
- 양쪽 바퀴를 정면을 보도록 정면을 맞춥니다.
- 전방에 위치한 빨간색 걸쇠()를 당겨 올린 후, 약 30도 정도 회전 시켜서 아래로 내려진 상태로 고정시킵니다.
- 차량 정면에 있는 LED가 황색으로 변경되며 변경이 성공합니다.



03 LIMO Mode Switching

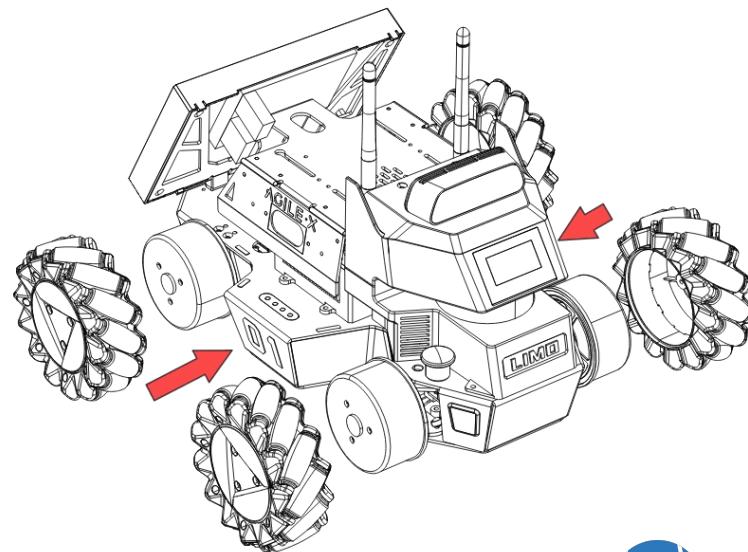
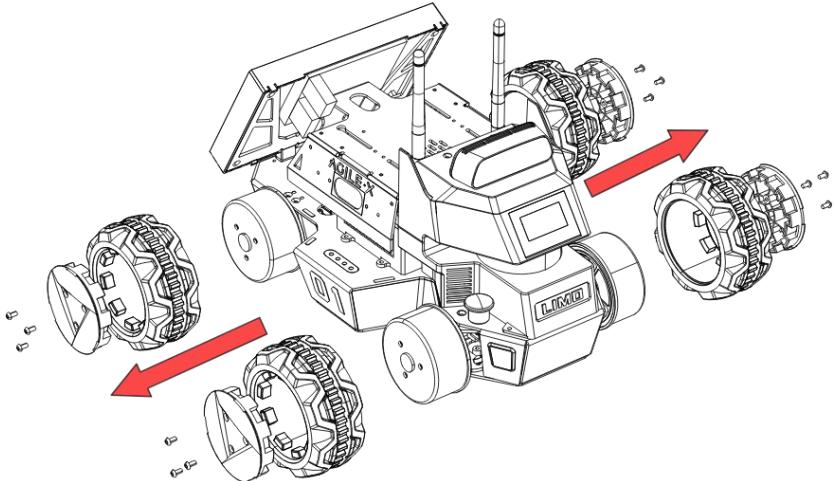
- Track mode

- 걸쇠를 아래로 내려 Differential mode로 변경합니다.
- 구성품에 포함된 Track을 연결합니다.
- 굽힘을 방지하기 위해, Track mode에서는 양쪽 문을 열어둔 채로 사용합니다.



03 LIMO Mode Switching

- Mecanum wheel mode
 - 우선 바퀴의 뚜껑과 타이어를 제거합니다.
 - Hub motor만을 남겨둔 상태에서 메카눔 휠의 작은 롤러가 차량 몸쪽으로 향하게 장착합니다.
 - M3 * 5 나사를 이용하여 고정합니다.
 - 조종기 or APP을 이용하여 Mecanum 모드로 변경합니다.



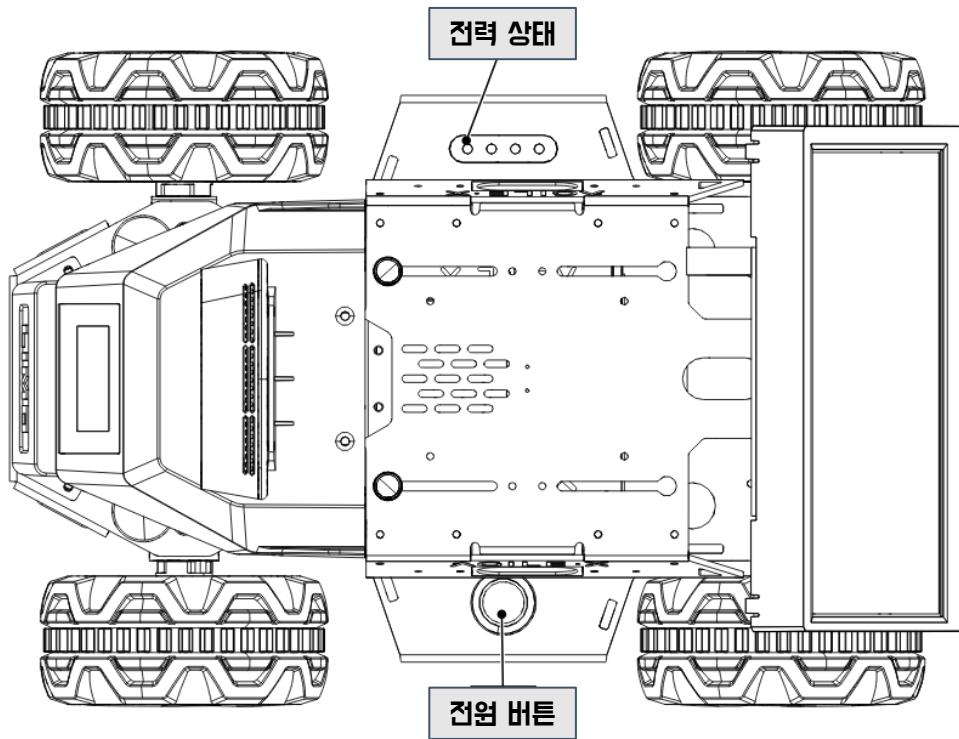
04

**LIMO
하드웨어
사용 방법**

04 LIMO 하드웨어 사용 방법

- LIMO 전원 켜기

- 잔축의 전원 버튼을 길게 누릅니다. (짧게 누를 경우, 프로그램 일시 정지)
- 오른쪽의 전력 상태를 확인하고, 부족할 경우 충전하거나, 다른 배터리로 교체합니다.



04 LIMO 하드웨어 사용 방법

- LIMO 조종 (조종기가 따로 있는 경우)
 - SWA : 기능 없음
 - SWB : 가운데일 경우 수동 조작. 상단일 경우 코드를 통해 제어 가능



04 LIMO 하드웨어 사용 방법

- LIMO 조종 (조종기가 따로 있는 경우)
 - SWC : 기능 없음
 - SWD : Differential or Mecanum 변경 가능



04 LIMO 하드웨어 사용 방법

- LIMO 조종 (조종기가 따로 있는 경우)
 - Analog Stick Left : 전후진 가능 (Mecanum 모드에서 좌우 이동도 가능)
 - Analog Stick Right : Ackermann Mode 및 전방바퀴 조향, Diff or Mecanum 및 회전



04 LIMO 하드웨어 사용 방법

- LIMO 조종
 - 아래의 QR코드를 인식하여 조종 어플을 다운로드 받습니다.



IOS



Android

04 LIMO 하드웨어 사용 방법

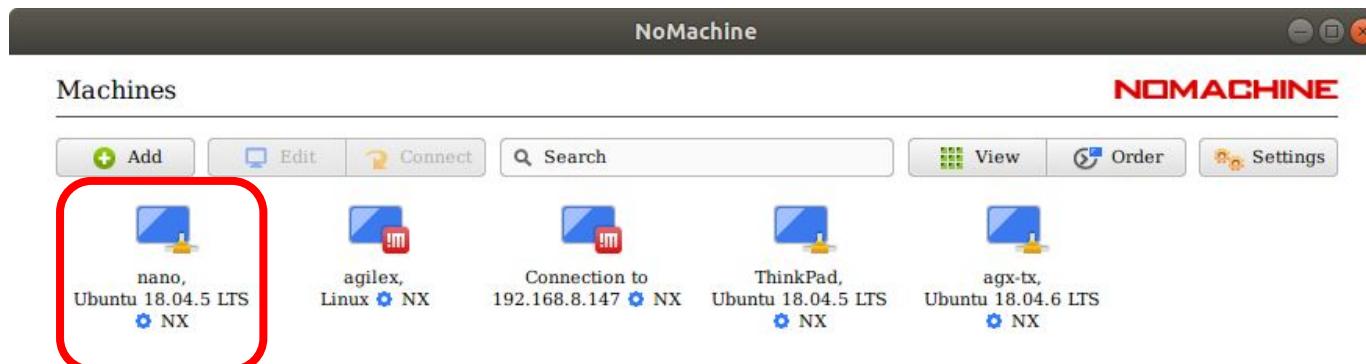
- LIMO 조종

- Application을 실행하고, Bluetooth로 LIMO를 연결합니다. 차량 앞부분에 붙어있는 숫자와 맞는 Bluetooth에 연결합니다.
- 왼쪽 레버를 이용하여 LIMO를 앞뒤로 움직일 수 있습니다.
- 오른쪽 레버를 이용하여 LIMO를 좌우로 회전시킬 수 있습니다.
- LIMO의 걸쇠를 변경한 후, 위쪽의 조종기 모드를 변경하여 조종 모드를 변경할 수 있습니다.
- 우측 상단의 설정에 진입하여 아래의 내용을 변경할 수 있습니다.
 - 언어 선택
 - 왼쪽 레버 감도 (리모 이동 속도 변경 가능)
 - 오른쪽 레버 감도 (리모 회전 속도 변경 가능)
 - 바퀴 각도 Calibration (Ackermann mode에서 0도 설정 변경 가능)
 - Bluetooth (연결 관련 셋팅 확인 가능)

04 LIMO 하드웨어 사용 방법

- LIMO 원격 접속

- NoMachine 다운로드 및 설치
- <https://www.nomachine.com/download>
- 위 링크에 접속하여 사용하는 OS에 맞는 버전을 다운로드 및 설치합니다.
- LIMO를 원격 접속하는 PC와 동일한 WiFi에 접속시킵니다.
- 원격 접속 PC에서 NoMachine을 실행하여, nano Ubuntu 18.04.5 LTS가 뜨는지 확인합니다. (없을 경우, NoMachine을 종료한 후, 다시 실행합니다)
- 해당 아이콘을 더블클릭하여 접속합니다. (ID: agilex, PW: agx)



04 LIMO 하드웨어 사용 방법

- LIMO 배터리

- LIMO는 12V 배터리를 장착하고 있습니다. (5000mAH)
- 평균 전압 - 11.1V. 최대 전압 - 12.6V. 최소 전압 - 8.25V. 최대 방전 전류 - 10A
- 배터리가 방전된 이후에는 충전하지 마십시오
- LIMO의 저전력 알람이 확인되면 꼭 충전하십시오
- LIMO는 전원이 꺼져있을 때도 대기 전력을 약간 사용합니다. 오랜 기간 LIMO를 사용하지 않을 경우, LIMO의 배터리를 분리해놓으십시오
- 배터리를 섭씨 -10 ~ 40도 범위에서만 사용하십시오

04 LIMO 하드웨어 사용 방법

- LIMO 충전

- LIMO는 12.6V, 2A 충전기를 포함하고 있습니다.
- LIMO 충전기에는 LED가 있으며, 충전 상태를 확인할 수 있습니다.
- LIMO 충전을 위해서는 LIMO에서 배터리를 분리한 상태에서 충전하십시오
- LIMO 배터리의 커넥터와 충전기를 연결한 후, 충전기의 LED의 전원이 들어오는지 확인하십시오
- 완전히 충전되면, 배터리를 충전기와 분리한 후, 충전기의 전원을 끄십시오
- LED의 상태는 다음과 같습니다.
 - 적색 : 충전 중
 - 녹색 점멸 : 거의 충전 된 상태
 - 녹색 : 충전 완료

04 LIMO 하드웨어 사용 방법

- LIMO 충전
 - 충전 관련 주의 사항
 - 정품 충전기 이외의 충전기를 사용하지 마십시오.
 - 섭씨 0도 이하의 온도에서 충전하지 마십시오
 - LIMO를 사용 중에 충전하지 마십시오
 - 충전기의 LED가 녹색으로 변경되었을 때, 완충된 상태이지만, 배터리를 오래 사용하기 위해, 30분정도 추가로 충전하는 것이 좋습니다.
 - 8.25V에서 완충되는데 2.5시간이 소요됩니다. 완충된 상태의 전압은 12.6V입니다.

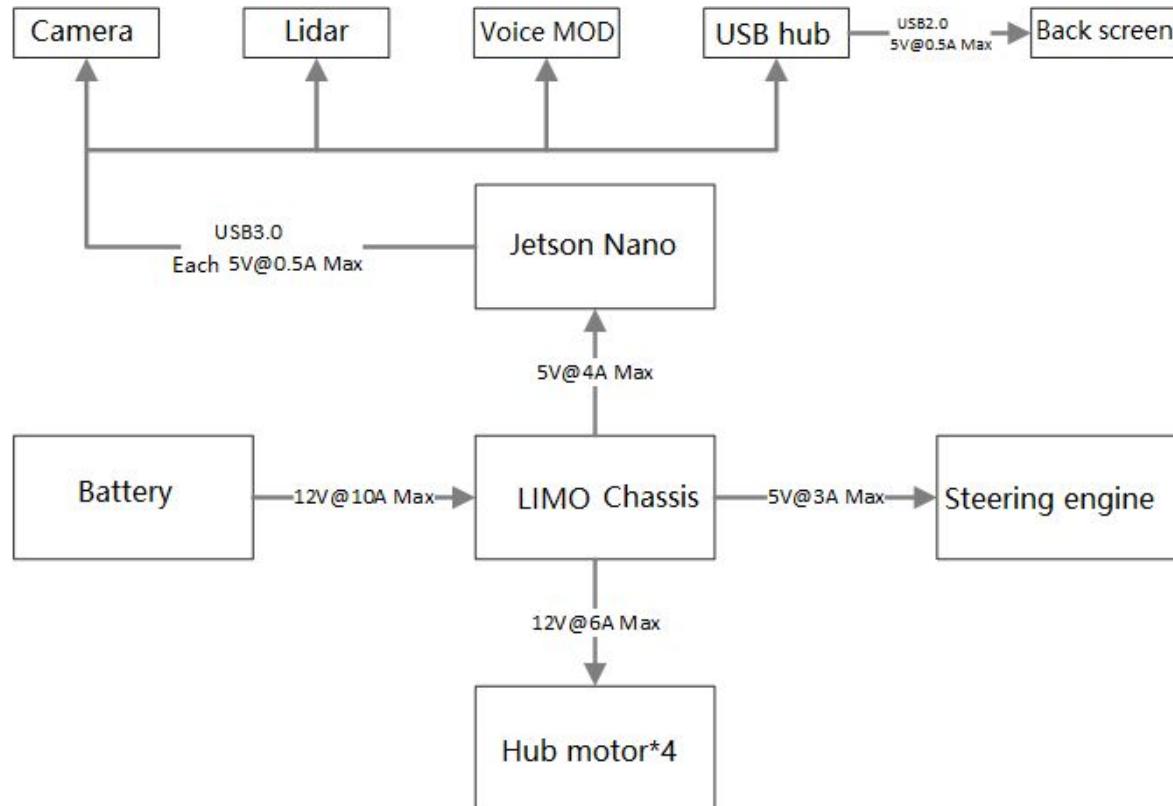
04 LIMO 하드웨어 사용 방법

- LIMO 사용
 - LIMO 사용 주의 사항
 - LIMO의 적정 사용 온도는 섭씨 -10 ~ 40도입니다. 이 외의 온도에서 사용하지 마십시오.
 - LIMO의 적정 사용 습도는 30 ~ 80%입니다. 이 외의 습도에서 사용하지 마십시오.
 - 화기 근처에서 LIMO를 사용하지 마십시오.
 - LIMO는 방수가 아닙니다. 눈, 비, 그 외의 물에 닿지 않게 주의하십시오.
 - 1000M 이하의 고도에서 사용하는 것을 권장합니다.
 - 낮과 밤의 온도 차이가 섭씨 25도 이상 나지 않는 것을 권장합니다.
 - 사용 시 문제 발생 시, 사용 매뉴얼 또는 기술 지원 문의를 주시길 바랍니다.
 - 기술 지원과 협력 없이는 제품의 구조를 변경하지 마십시오.

04 LIMO 하드웨어 사용 방법

- LIMO Power Supply 구조

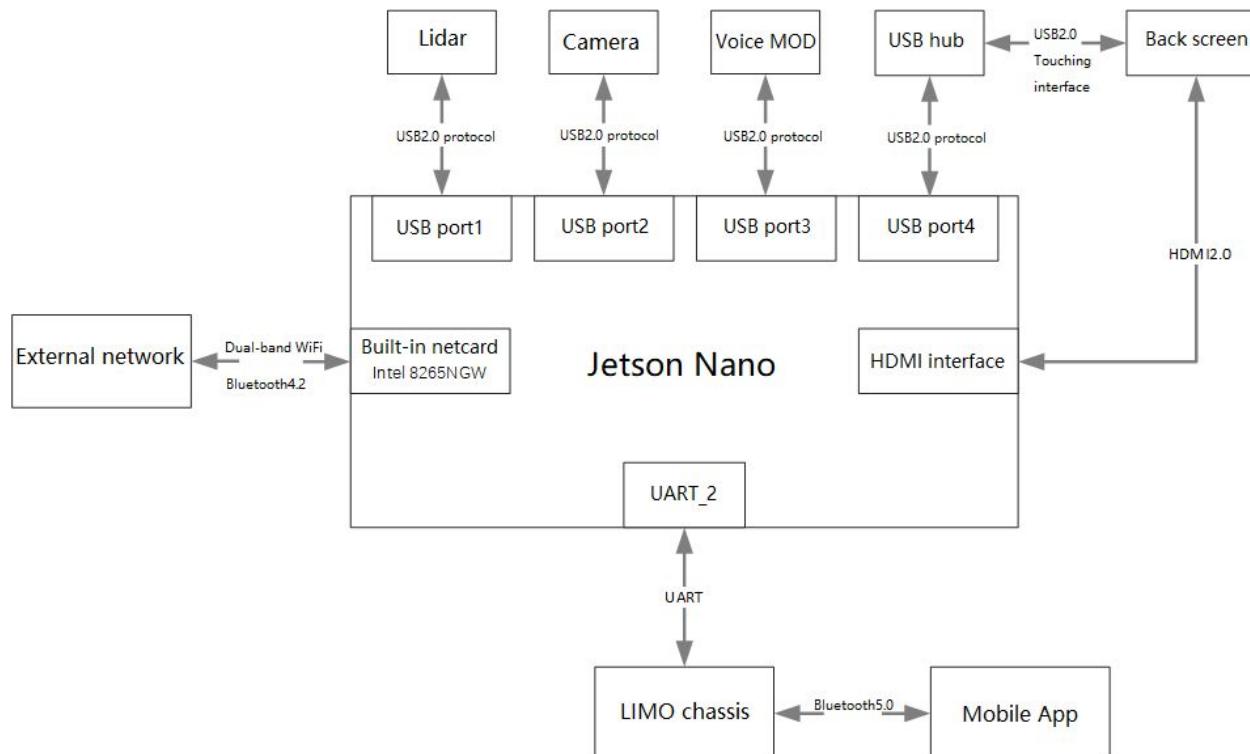
- LIMO의 배터리는 최대 10A까지 전달할 수 있습니다. 10A 이상의 전류가 흐를 경우, 과전류 보호 상태로 진입합니다.
- USB 3개는 최대 전류 0.5A까지 사용 가능합니다.



04 LIMO 하드웨어 사용 방법

LIMO 통신 구조

- LIMO 본체는 Bluetooth 5.0 모듈을 장착하고 있습니다. (원격 조종)
- LIMO와 Jetson Nano는 UART로 연결되어 있습니다.
- USB Hub는 2개의 USB와 1개의 Type C를 제공합니다.
- 후면 스크린은 터치 사용을 위해 USB로 연결되어 있습니다.



05

**LIMO
소프트웨어
사용방법**

05 LIMO 소프트웨어 사용 방법

- 차량 Driver

- 차량 Driver는 C++과 Python의 두 가지 타입으로 제공됩니다.
- 각 버전은 모두 LIMO를 제어할 수 있습니다.

05 LIMO 소프트웨어 사용 방법

- C++ 차량 Driver

- `~/agilex_ws/src/limo_ros/limo_base` 폴더에서 확인할 수 있습니다.

- 내부에는 4개의 폴더가 있으며, 각각 다음의 기능을 제공합니다.

- include - Driver가 사용하는 Library
 - launch - Driver 실행 파일
 - msg - Driver에 필요한 Message 파일
 - src - Driver source code

- 아래의 command를 실행 전, 다른 터미널이 모두 종료되어 있는지 확인하십시오. 종료는 `Ctrl + C` 를 터미널에서 입력하면 됩니다.

- 터미널을 열고, 아래의 명령어를 입력하여, 차량을 실행합니다.

- `$ roslaunch limo_base limo_base.launch`

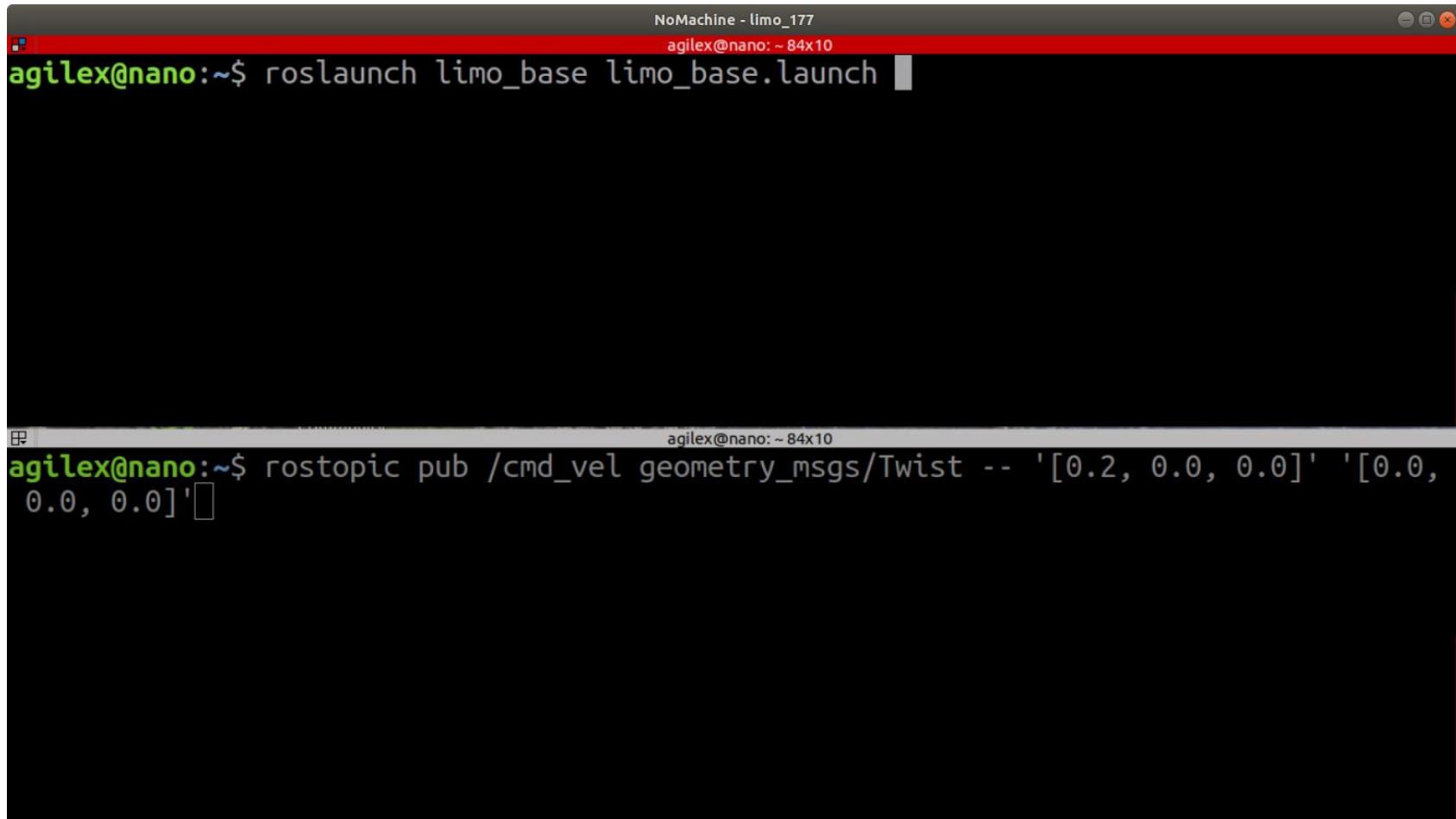
- 새로운 터미널을 열고, 아래의 명령어를 입력하여, 차량을 움직입니다. (짧게 전진)

- `$ rostopic pub /cmd_vel geometry_msgs/Twist -- '[0.2, 0.0, 0.0]' '[0.0, 0.0, 0.0]'`

```
└── limo_base
    ├── CMakeLists.txt
    ├── include
    │   ├── limo_driver.h
    │   ├── limo_protocol.h
    │   └── serial_port.h
    ├── launch
    │   └── limo_base.launch
    ├── msg
    │   └── LimoStatus.msg
    ├── package.xml
    └── src
        ├── limo_base_node.cpp
        ├── limo_driver.cpp
        └── serial_port.cpp
```

05 LIMO 소프트웨어 사용 방법

- 실행 환경



The screenshot shows a terminal window titled "NoMachine - limo_177" with the command "agilex@nano:~\$ rosrun limo_base limo_base.launch" entered. Below it, another terminal window shows the command "agilex@nano:~\$ rostopic pub /cmd_vel geometry_msgs/Twist -- '[0.2, 0.0, 0.0]' '[0.0, 0.0, 0.0]'". Both windows have a red header bar.

```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ rosrun limo_base limo_base.launch

agilex@nano:~$ rostopic pub /cmd_vel geometry_msgs/Twist -- '[0.2, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

The screenshot shows two terminal windows. The top window is titled 'NoMachine - limo_177' and displays the output of running the 'limo_base.launch' file. It shows the master starting, nodes like 'rosout-1' and 'limo_base_node-2' launching with their respective process IDs (9077, 9090, 9093), and the serial port '/dev/ttyTHS1' being opened. The bottom window is titled 'agilex@nano: ~ 84x10' and shows a user publishing a Twist message to the '/cmd_vel' topic. The message is defined by the geometry_msgs/Twist message type, with linear and angular velocities set to [0.2, 0.0, 0.0] and [0.0, 0.0, 0.0] respectively. The user is instructed to press ctrl-C to terminate the publishing process.

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_base/launch/limo_base.launch http://localhost:11311 84x10
auto-starting new master
process[master]: started with pid [9077]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 665d1404-bc60-11ec-8347-1418c3db156e
process[rosout-1]: started with pid [9090]
started core service [/rosout]
process[limo_base_node-2]: started with pid [9093]
[ INFO] [1649988278.930622741]: open the serial port: /dev/ttyTHS1

agilex@nano:~$ rostopic pub /cmd_vel geometry_msgs/Twist -- '[0.2, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
publishing and latching message. Press ctrl-C to terminate
```

05 LIMO 소프트웨어 사용 방법

- C++ 차량 Driver

- Driver Source Code의 다양한 함수입니다.
 - `connect()` → 차량과 연결
 - `readData()` → 차량에서 나오는 데이터를 받아오기
 - `processRxData()` → Serial Data를 받아오기
 - `parseFrame()` → Serial Data를 처리 (파싱)
 - `sendFrame()` → Serial Data를 전송
 - `setMotionCommand()` → LIMO의 Control mode 설정
 - `enableCommandedMode()` → Control Mode를 활성화
 - `publishOdometry()` → Odometer Data를 Publish (ROS)
 - `publishLimoState()` → LIMO의 상태 정보를 Publish (ROS)
 - `publishIMUData()` → IMU Data를 Publish (ROS)
 - `processErrorCode()` → Error Detection 진행
 - `twistCmdCallback()` → 속도 제어 데이터 Publish (ROS)
 - `normalizeAngle()` → 각도 범위를 넘어가지 않도록 일반화
 - `degToRad()` → Degree를 Radian으로 변경
 - `convertInnerAngleToCentral()` → Inner Angle을 Central Angle로 변환
 - `convertCentralAngleToInner()` → Central Angle을 Inner Angle로 변환

05 LIMO 소프트웨어 사용 방법

- Python 차량 Driver

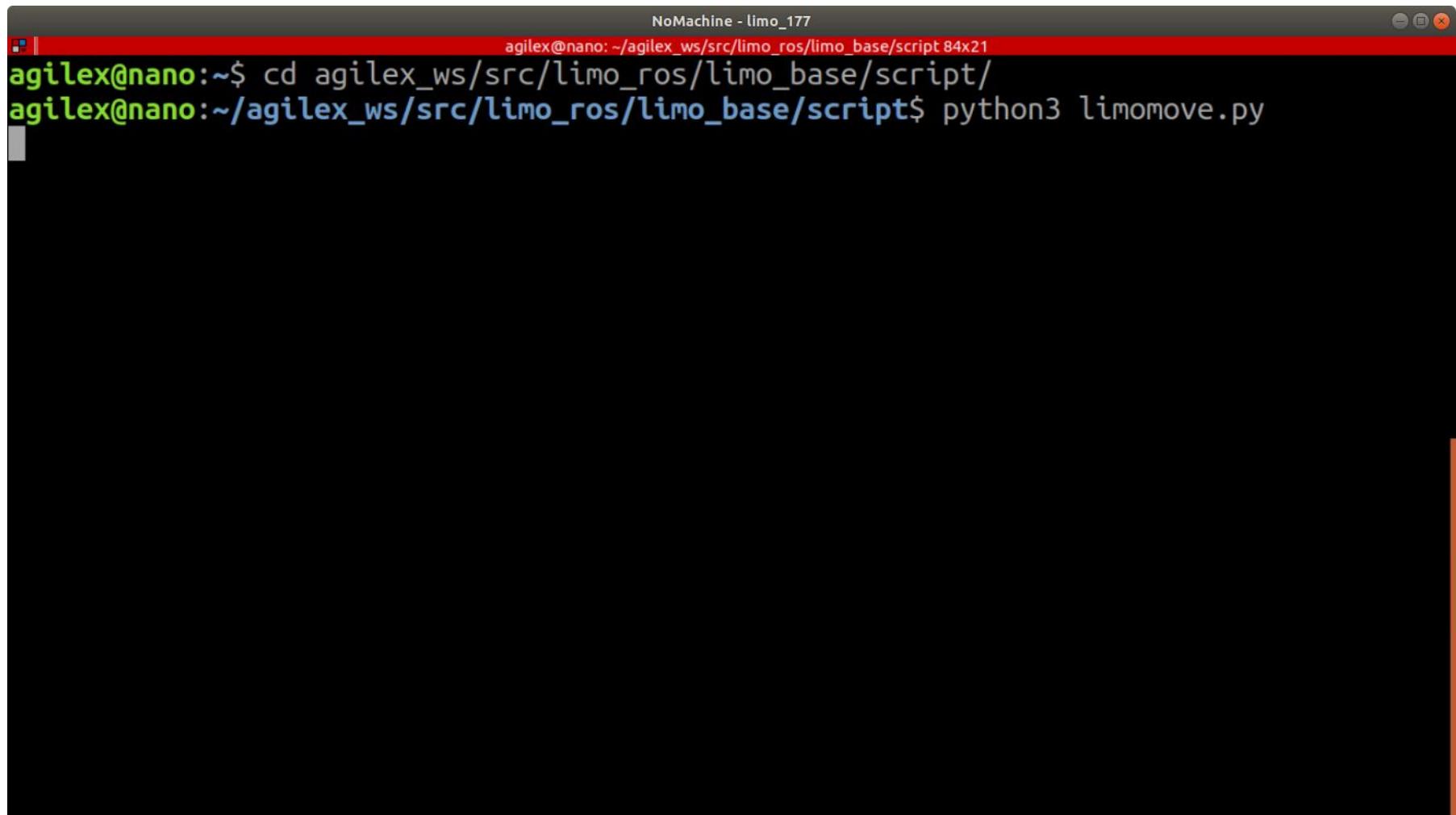
- LIMO의 Driver는 pypi에 업로드되어 있습니다.
- pip를 이용하여 pylimo를 다운로드할 수 있습니다. 내용물은 다음과 같습니다.
- Python 버전의 코드는 상대적으로 간결합니다.
 - init.py → 사용할 파일 정의
 - limomsg.py → 필요 메시지를 다루는 코드
 - limo.py → 메인 프로그램. LIMO 제어를 위해 사용
- LIMO 제어를 위한 코드는
~/agilex_ws/src/limo_ros/limo_base/script/limomove.py 입니다.

```
└── __init__.py
└── limomsg.py
└── limo.py
└── __pycache__
    ├── __init__.cpython-36.pyc
    ├── limo.cpython-36.pyc
    └── limomsg.cpython-36.pyc
```

- 모든 터미널을 종료하고 아래의 명령어를 입력하여 LIMO를 제어할 수 있습니다.
 - \$ cd agilex_ws/src/limo_ros/limo_base/script
 - \$ python3 limomove.py
- 위 명령을 입력하면 LIMO가 약간 앞으로 전진한 후 멈춥니다.

05 LIMO 소프트웨어 사용 방법

- 실행 환경 및 결과



A screenshot of a terminal window titled "NoMachine - limo_177". The terminal is running on a Linux system with the command line interface. The user, "agilex", is in their home directory (~) and has navigated to the "limo_base/script" directory within the "limo_ros" source code. They have run the command "python3 limomove.py". The terminal window has a red header bar and a black background.

```
NoMachine - limo_177
agilex@nano:~/agilex_ws/src/limo_ros/limo_base/script$ python3 limomove.py
```

05 LIMO 소프트웨어 사용 방법

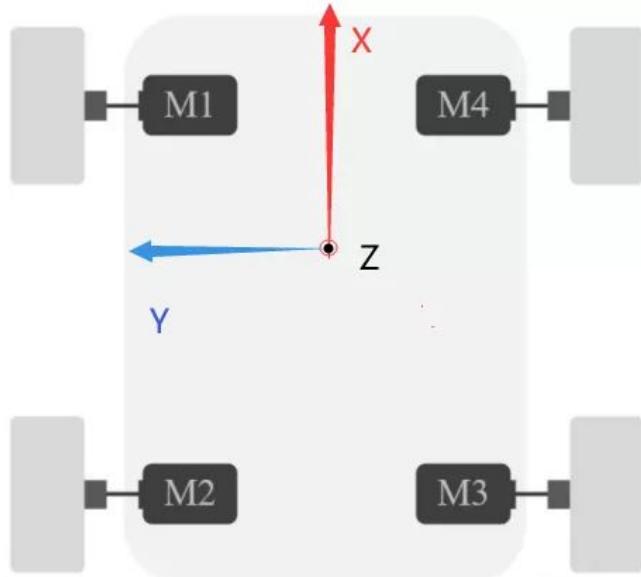
- Python 차량 Driver

- Python Driver 내부의 함수

- `EnableCommand()` → 제어 활성화
 - `SetMotionCommand()` → Motion Command 설정
 - `GetLinearVelocity()` → Linear Velocity 받아오기
 - `GetAngularVelocity()` → Angular Velocity 받아오기
 - `GetSteeringAngle()` → Steering Angle 받아오기
 - `GetLateralVelocity()` → Lateral Velocity 받아오기
 - `GetControlMode()` → Control Mode 받아오기
 - `GetBatteryVoltage()` → Battery 상태 받아오기
 - `GetErrorCode()` → Error Code 받아오기
 - `GetRightWheelOdom()` → 오른쪽 바퀴의 Odometer 받아오기
 - `GetLeftWheelOdom()` → 왼쪽 바퀴의 Odometer 받아오기
 - `GetIMUAccelData()` → IMU 가속도 받아오기
 - `GetIMUGyroData()` → IMU 회전속도 받아오기
 - `GetIMUYawData()` → IMU Yaw 각도 받아오기
 - `GetIMUPitchData()` → IMU Pitch 각도 받아오기
 - `GetIMURollData()` → IMU Roll 각도 받아오기

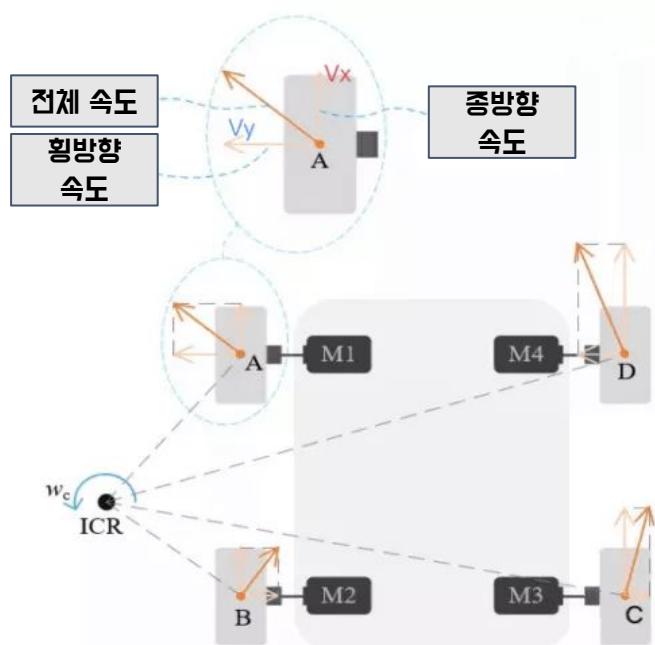
05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Four-wheel differential motion mode
 - 4wD mode의 경우, 각 바퀴의 속도를 조절하여, 전후진 및 제자리 회전 등이 가능합니다.
 - 빨간색 화살표가 +X 방향이며, 파란색 화살표가 +Y 방향, +Z 방향은 로봇에서 나오는 방향입니다.
 - 각 잣표의 원점은 로봇의 무게중심입니다.
 - 로봇의 모든 바퀴의 속도가 같을 때, 전후진이 되며, 다를 때 회전이 됩니다.



05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Four-wheel differential motion mode
 - 로봇이 회전할 때, 회전에 대한 중심이 있으며, 이를 point ICR로 표현합니다.
 - 각 바퀴에서의 상대 속도는 ICR 점과 각 바퀴의 중심을 이은 선에 수직하게 작용합니다.
 - 타이어는 종방향으로만 회전할 수 있으나, 속도는 횡방향에서도 존재합니다.



05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Four-wheel differential motion mode
 - 네 바퀴의 횡방향 속도가 다르지만, 이들의 합을 통해 횡방향 움직임이 생성되고 네 바퀴의 종방향 속도 또한 다르지만, 이들의 합을 통해 종방향 움직임이 생성됩니다.
 - 결과로 생성된 움직임은 점 ICR을 기준으로 한 원 운동을하게 됩니다.
 - 로봇이 회전할 때, 왼쪽 바퀴들 그리고 오른쪽 바퀴들의 속도는 각각 동일합니다.
 - 회전 반대 방향에 있는 바퀴들은 작은 회전 반경을 가지며, 따라서 선속도도 작습니다.
 - 회전 바깥 방향에 있는 바퀴들은 큰 회전 반경을 가지며, 따라서 선속도도 크게 작용합니다.
 - 오른쪽의 바퀴들의 속도가 더 크다면, 로봇은 왼쪽으로 회전하며, 반대의 경우 반대로 회전합니다.
 - 왼쪽 바퀴들과 오른쪽 바퀴들의 속도를 반대 방향으로 작용하면 제자리에서 회전할 수 있습니다.

05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Four-wheel differential motion mode
 - LIMO를 Four-wheel differential mode로 제어하려면 우선 차량 앞에 있는 걸쇠를 내려간 상태로 유지합니다.
 - 전방의 LED가 노란색으로 출력되는 것을 확인합니다.
 - 열려 있는 모든 터미널을 닫고 새로운 터미널을 2개 열어 아래의 명령어를 각각 실행하여 LIMO를 키보드로 제어할 수 있습니다.
 - \$ roslaunch limo_base limo_base.launch
 - \$ roslaunch limo_bringup limo_teleop_keyboard.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with two stacked command-line sessions. The top session is titled 'NoMachine - limo_177' and has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_base limo_base.launch`. The bottom session is titled 'agilex@nano: ~ 84x10' and also has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch`.

```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_base limo_base.launch

agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_base/launch/limo_base.launch http://localhost:11311 84x10
auto-starting new master
process[master]: started with pid [9904]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 2f647cca-bc61-11ec-86c2-1418c3db156e
process[rosout-1]: started with pid [9920]
started core service [/rosout]
process[limo_base_node-2]: started with pid [9928]
[ INFO] [1649988616.215791187]: open the serial port: /dev/ttyTHS1

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

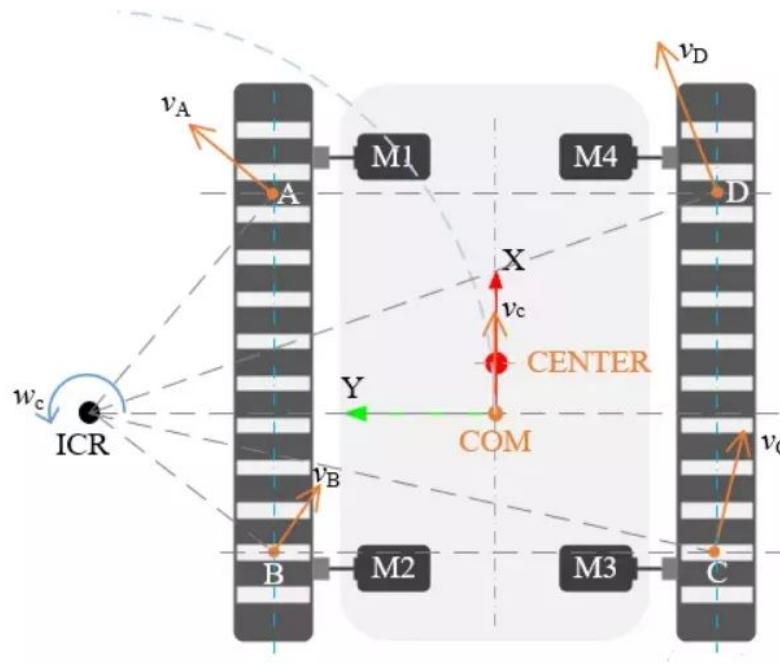
currently:      speed 0.5      turn 1.0
```

05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Track motion mode
 - 한 쪽에 장착된 Track은 무한한 개수의 작은 바퀴가 달려 있다고 생각할 수 있습니다.
 - 그리고 한 쪽의 무한한 개수의 작은 바퀴들의 속도는 모두 동일합니다.
 - 그러므로, Track motion mode의 조향은 four-wheel differential mode와 동일합니다.
 - 차이가 있다면, 전단력과 압력의 분포가 달라지게 됩니다.
 - 이 차이로 인해 속도 제어 시 약간의 차이가 발생합니다.

05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Track motion mode
 - 바퀴들의 속도가 일정하지 않을 때, 다음과 같은 모델을 확인할 수 있습니다.
 - ICR은 마찬가지로 회전 중심이며, CENTER는 실제 로봇의 기하학적 중심위치이며, COM은 로봇의 무게중심입니다.
 - 회전할 때, 회전하는 양쪽 Track의 속도가, 바깥 쪽 Track보다 작을 것입니다.



05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics

- Track motion mode

- Four-Wheel Mode에서 Track을 바로 장착하면 사용할 수 있습니다.
 - Track은 뒷바퀴에 먼저 장착 후, 앞바퀴에 장착하는 것을 권장합니다.
 - Track Mode 시에는 차량 외관에 상처가 나는 것을 방지하기 위해, 차량 옆 문을 열고 사용하는 것을 권장합니다.
 - 교체가 종료되면 열려 있는 모든 터미널을 종료하고, 터미널을 2개 열어 아래의 명령어를 각각 실행시킵니다. 이를 통해서 LIMO를 키보드로 조작할 수 있습니다.
 - \$ roslaunch limo_base limo_base.launch
 - \$ roslaunch limo_bringup limo_teleop_keyboard.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with two stacked command-line sessions. The top session is titled 'NoMachine - limo_177' and has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_base limo_base.launch`. The bottom session is titled 'agilex@nano: ~ 84x10' and also has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch`.

```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_base limo_base.launch

agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_base/launch/limo_base.launch http://localhost:11311 84x10
auto-starting new master
process[master]: started with pid [9904]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 2f647cca-bc61-11ec-86c2-1418c3db156e
process[rosout-1]: started with pid [9920]
started core service [/rosout]
process[limo_base_node-2]: started with pid [9928]
[ INFO] [1649988616.215791187]: open the serial port: /dev/ttyTHS1

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

05 LIMO 소프트웨어 사용 방법

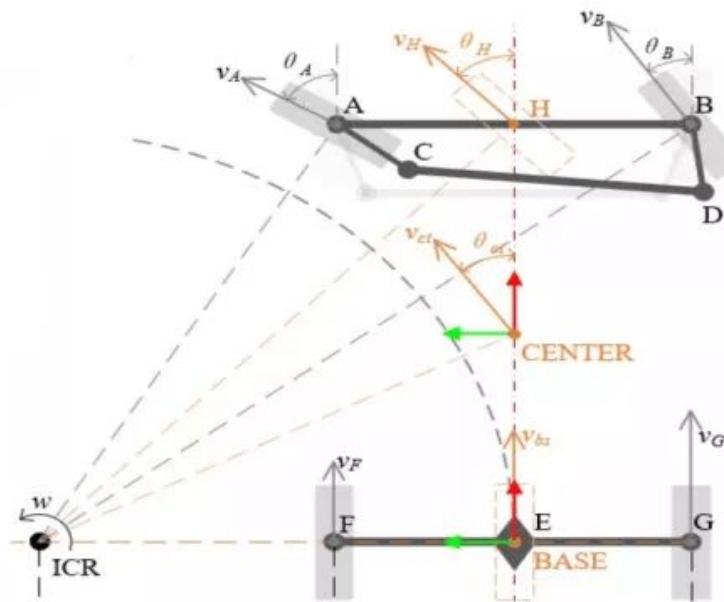
- Chassis Kinematics
 - Ackermann motion mode
 - Ackermann Steering 구조는 현대의 자동차의 조향 방식과 동일합니다.
 - 이 방식을 통해 차량이 조향할 때, 좌우 스티어링 휠의 조향 반경이 다르기 때문에 좌우 스티어링 휠의 조향 각도가 다른 문제를 해결할 수 있습니다.

05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics

- Ackermann motion mode

- 전후진은 Differential motion과 동일합니다.
 - 두 앞바퀴의 틀어진 방향을 기준으로 회전반경을 계산하기 위해서 액커만 기하학을 사용합니다.
 - CENTER 점은 로봇의 기하학적 중심이며, BASE 점은 뒷 축의 중심점입니다.
 - 앞 바퀴의 틀어진 각도가 같지 않으며, 두 앞바퀴의 틀어진 각도의 차이가 액커만 각입니다.

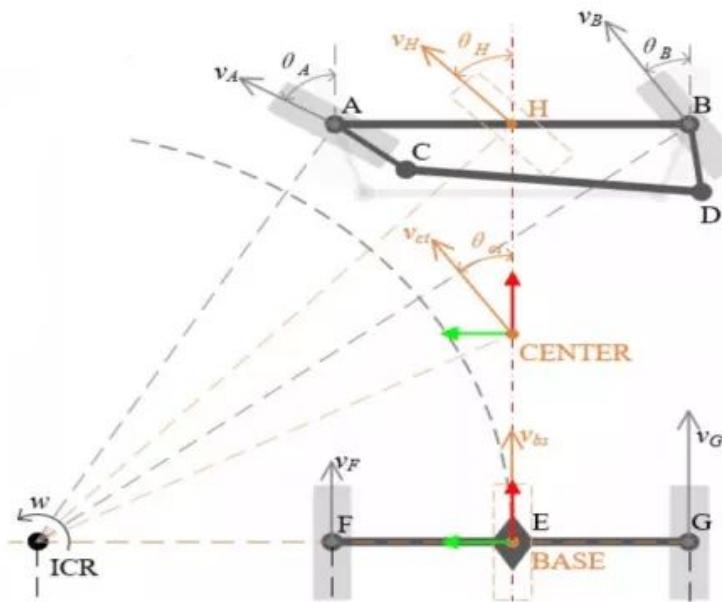


05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics

- Ackermann motion mode

- 애커만 모델은 일반적으로 타는 자전거와 같이 단순화될 수 있습니다.
 - 그림의 오렌지색으로 표현된 부분은 로봇과 동일한 자전거 모델입니다.
 - Differential mode와 달리 애커만 모델은 회전 반경의 제한이 있습니다.
 - 따라서, 제자리 회전이 불가능합니다.



05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Ackermann motion mode
 - 애커만 모드를 사용하기 위해서는 양쪽의 걸쇠를 위로 걸쳐줍니다.
 - 앞에 있는 LED의 색이 녹색으로 변하면 정상적으로 변경된 것입니다.
 - 변경이 완료된 후, 열려 있는 모든 터미널을 닫고, 터미널을 2개를 열고 아래의 명령어를 각각 실행합니다.
 - \$ roslaunch limo_base limo_base.launch
 - \$ roslaunch limo_bringup limo_teleop_keyboard.launch
 - 차량이 앞으로 전진하지 못할 경우, **Steering Gear Calibration01** 필요합니다.

05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with two stacked command-line sessions. The top session is titled 'NoMachine - limo_177' and has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_base limo_base.launch`. The bottom session is also titled 'agilex@nano: ~ 84x10' and displays the command: `agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch`.

```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_base limo_base.launch

agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_base/launch/limo_base.launch http://localhost:11311 84x10
auto-starting new master
process[master]: started with pid [9904]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 2f647cca-bc61-11ec-86c2-1418c3db156e
process[rosout-1]: started with pid [9920]
started core service [/rosout]
process[limo_base_node-2]: started with pid [9928]
[ INFO] [1649988616.215791187]: open the serial port: /dev/ttyTHS1

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

05 LIMO 소프트웨어 사용 방법

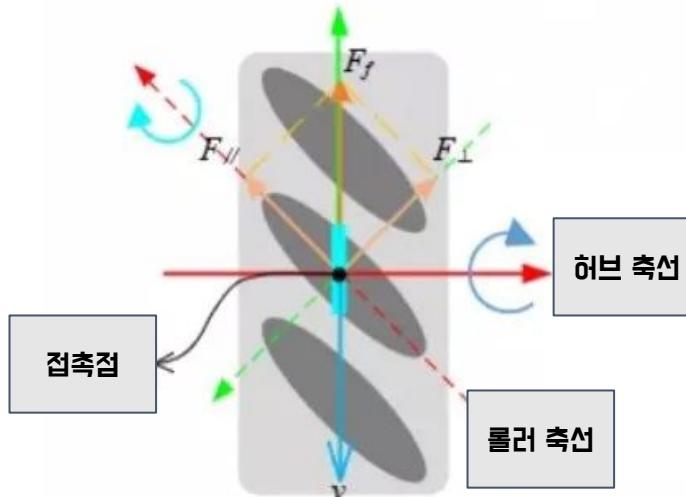
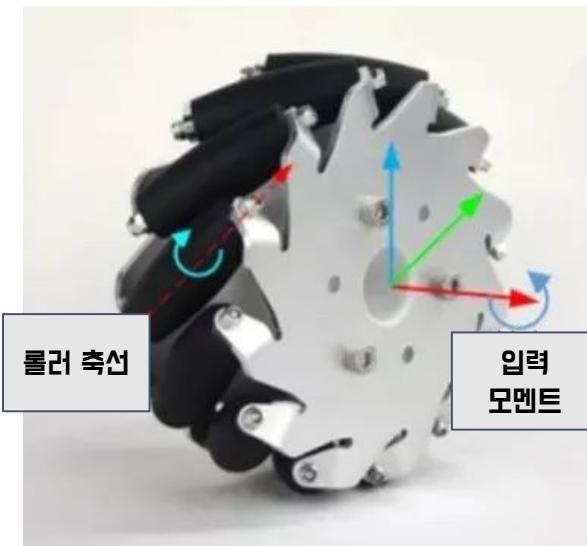
- Chassis Kinematics
 - Mecanum motion mode
 - 메카넘 휠은 허브와 롤러로 이루어져있는 휠입니다.
 - 허브는 전체 휠을 지지하며, 롤러는 허브에 장착되어 있는 움직이는 작은 바퀴입니다.
 - 현재 시장에서 판매하는 메카넘 휠의 허브 축과 롤러 샤프트 사이의 각도는 30도, 45도, 60도 등으로 나뉩니다.

05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics

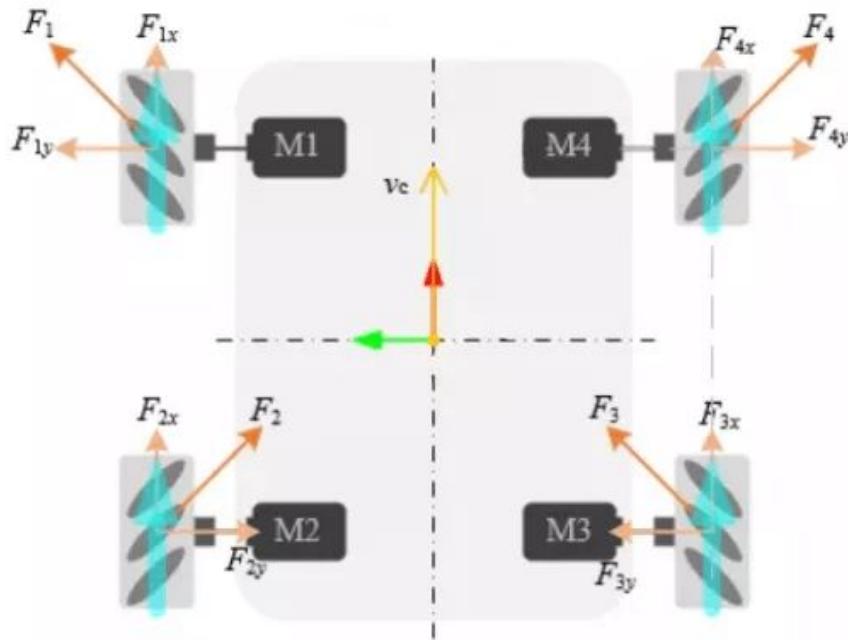
- Mecanum motion mode

- 빨간색 화살표가 +X 방향, 녹색 화살표가 +Y 방향, 파란색 화살표가 +Z 방향
 - 롤러의 잣표 축은 점선으로 표현되며, 허브의 잣표 축은 실선으로 표현됩니다.
 - 노란색 화살표는 메카님 휠과 롤러의 힘 분석을 나타냅니다.
 - 파란색 화살표는 속도의 방향을 나타냅니다.



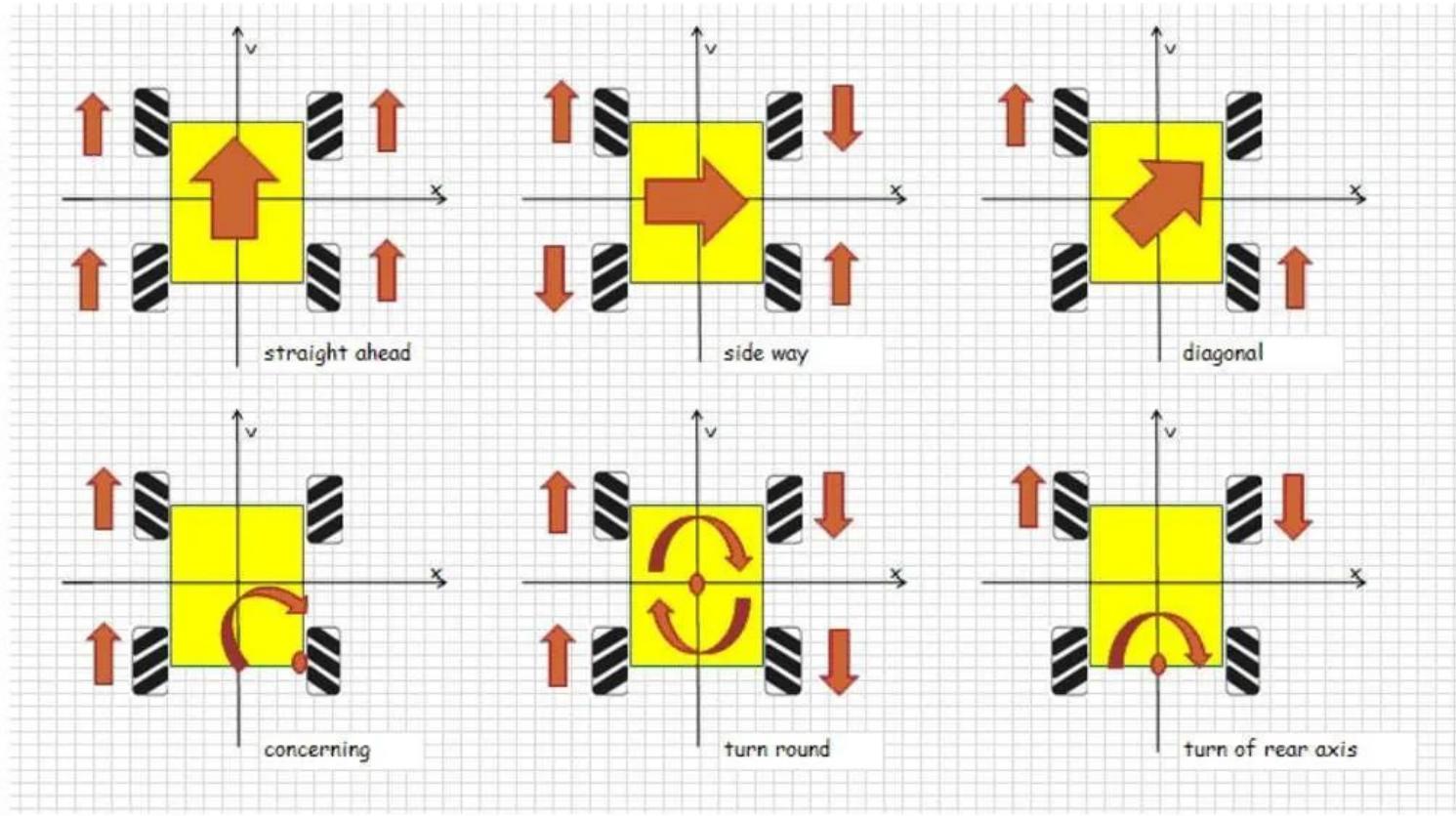
05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Mecanum motion mode
 - 파란색 화살표는 바퀴의 회전 방향을 의미하며, 오렌지색 화살표는 메카넘 휠의 힘 분석을 의미합니다.
 - 이러한 힘들의 합을 통해, 로봇은 앞쪽을 향하는 힘을 받게 됩니다.
 - 로봇의 각 바퀴의 마찰력을 조합하면, 로봇을 어떤 방향으로도 이동할 수 있습니다.



05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - Mecanum motion mode
 - 아래의 모양처럼 제어하면 차량을 어디로든 움직일 수 있습니다.



05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics

- 우선 허브 커버와 타이어를 제거하여 허브 모터만을 남깁니다.
- 각 메카넘 휠의 롤러가 로봇 몸쪽을 향하도록 합니다.
- M3 나사 5개를 이용하여 메카넘 휠을 장착합니다.
- 조종기 또는 앱에서 메카넘 휠 모드로 변경합니다.
- 이후 실행된 모든 터미널을 닫고 두 개의 터미널을 열어 아래의 명령어를 각각 실행하면 LIMO를 키보드로 제어할 수 있습니다.
 - \$ roslaunch limo_base limo_base.launch
 - \$ roslaunch limo_bringup limo_teleop_keyboard.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with two stacked command-line sessions. The top session is titled 'NoMachine - limo_177' and has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_base limo_base.launch`. The bottom session is titled 'agilex@nano: ~ 84x10' and also has a terminal size of 'agilex@nano: ~ 84x10'. It displays the command: `agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch`.

```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_base limo_base.launch

agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_teletop_keyboard.launch
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_base/launch/limo_base.launch http://localhost:11311 84x10
auto-starting new master
process[master]: started with pid [9904]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 2f647cca-bc61-11ec-86c2-1418c3db156e
process[rosout-1]: started with pid [9920]
started core service [/rosout]
process[limo_base_node-2]: started with pid [9928]
[ INFO] [1649988616.215791187]: open the serial port: /dev/ttyTHS1

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

05 LIMO 소프트웨어 사용 방법

- Chassis Kinematics
 - 각각의 모드는 장단점이 있으며, 상황에 맞게 골라서 사용할 수 있습니다.

Mode	Four-wheel differential	Track	Ackermann	Mecanum
장점	움직임 성능이 좋음 제어하기가 쉬움	높은 내구성 오프로드 주행성 향상 복잡한 지형 탐색이 가능	차량과 동일한 모델 자율주행차에 적합	기동성이 좋음
단점	조향 시 과도한 슬립이 발생 바퀴 마모가 큼	슬라이딩 조향 저항이 큼 트랙 마모가 심함	제한된 회전 반경 바퀴 마모 작음	지면 요구 사항이 큼 연속적이지 못한 롤러 움직임 중 진동이 크게 발생

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성

- LiDAR 소개 및 사용 방법

- LIMO에 장착된 LiDAR는 YDLiDAR X2L이며, 이는 360도 2차원 거리 센서입니다.
 - 삼각측량에 기반한 거리 측정 방식을 사용하며, 이에 관련된 광학적, 전자적 알고리즘을 이용하여 높은 주기와 높은 정확도로 거리를 측정합니다.
 - 거리 측정 시, 물리적인 구조가 360도 회전을 하며, 지속적으로 각도에 해당하는 정보를 얻어옵니다.

Items	• 순차적으로 총 360도의 거리를 측정한 흰 출력으로 포인트클라우드를 생성하여 전달합니다.				
측정 빈도	-	3000	-	Hz	초당 3000번의 거리를 측정
측정 빈도 (1바퀴 기준)	5	6	8	Hz	PWM 신호로 제어할 수 있으며, 6Hz를 권장
측정 거리	0.12	-	8	m	실내 환경, 80% 이상의 반사도를 가진 물체
측정 각도	-	0 ~ 360	-	Degree	-
절대 오차	-	2	-	cm	1m 이내 측정 시
상대 오차	-	3.5%	-	-	1~6m 사이 측정 시
각도 해상도	0.6 (5Hz)	0.72 (6Hz)	0.96 (8Hz)	Degree	측정 빈도에 따라 다름



WeGo

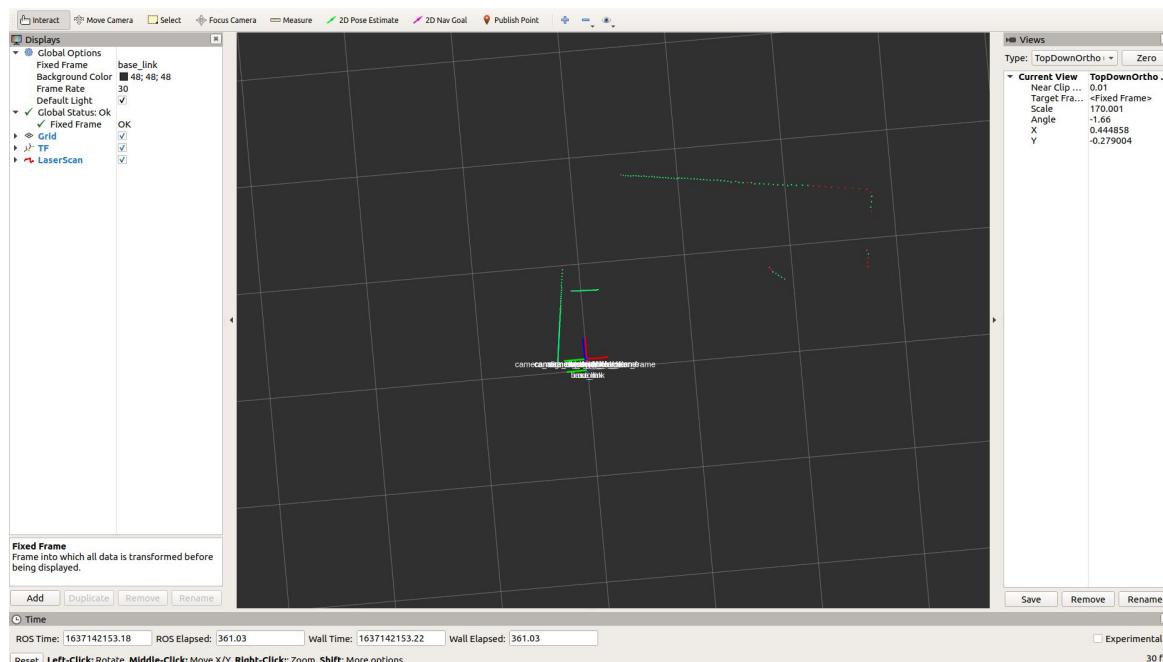
05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성
 - LiDAR 소개 및 사용 방법
 - 터미널을 모두 닫고 새로운 터미널을 열어 아래 명령어를 입력합니다.
 - \$ rosrun limo_bringup limo_start.launch pub_odom_tf:=false
 - 성공적으로 실행되면, 터미널에 아래와 같은 화면이 출력됩니다.



05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성
 - LiDAR 소개 및 사용 방법
 - 이전 페이지에서 실행한 터미널을 그대로 둔 채로 새로운 터미널을 실행하고 아래 명령어를 입력합니다.
 - \$ roslaunch limo Bringup lidar_rviz.launch
 - 성공적으로 실행되면 Rviz가 실행되며, 녹색 점이 LiDAR 데이터입니다.
 - 실행된 상태로 조종기/APP으로 LIMO를 조종하면 실시간으로 바뀌는 데이터를 볼 수 있습니다.



05 LIMO 소프트웨어 사용 방법

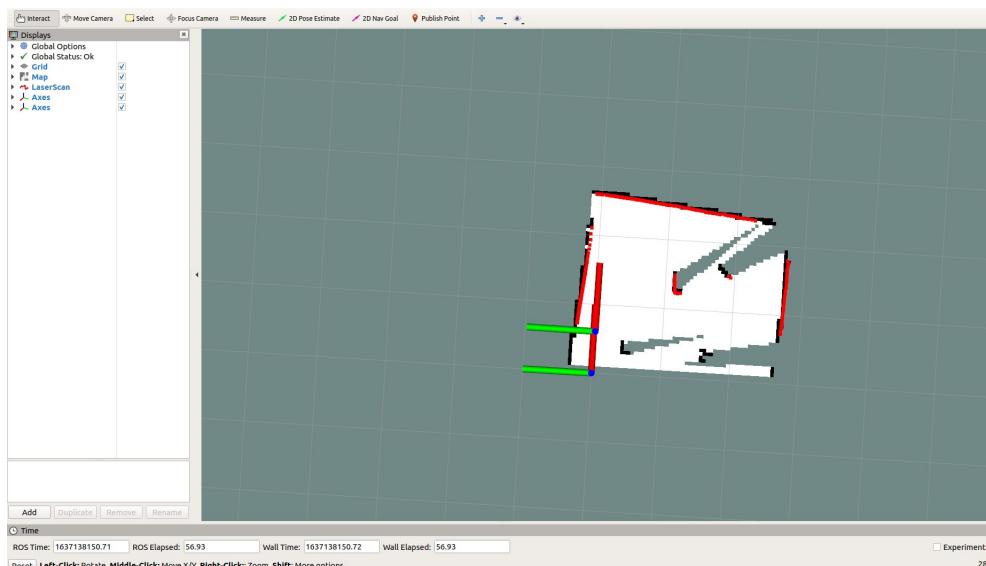
- LiDAR 기반 지도 생성
 - GMapping 기반의 지도 작성
 - GMapping은 Filter 기반의 SLAM 중 일반적으로 사용되는 알고리즘
 - GMapping은 효과적으로 Wheel odometer 정보를 사용
 - GMapping에는 고속 회전하는 LiDAR가 필요가 없음
 - 좁은 영역의 지도 생성 시, 필요한 계산량 역시 적으며, 정확도는 높음
 - 여기에서는 ROS 기반의 GMapping 패키지를 사용하여 LIMO 주변의 지도를 생성함

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성

- GMapping 기반의 지도 작성

- 열려 있는 모든 터미널을 닫고, 아래의 명령어를 새로운 터미널을 열어 각각 입력
 - `$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=false`
 - `$ roslaunch limo_bringup limo_gmapping.launch`
 - 올바르게 실행이 되면 아래와 같이 rviz가 실행됨
 - 지도 생성 시, LIMO의 속도는 느리면 느릴수록 좋습니다.
 - 속도가 과하게 빠를 경우, 매팅 성능에 영향을 미칠 수 있습니다.



05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with two stacked command-line sessions. The top session is titled 'NoMachine - limo_177' and has a user prompt 'agilex@nano:~\$'. It displays the command: 'roslaunch limo_bringup limo_start.launch pub_odom_tf:=false'. The bottom session is titled 'agilex@nano: ~ 84x10' and also has a user prompt 'agilex@nano:~\$'. It displays the command: 'roslaunch limo_bringup limo_gmapping.launch'. Both sessions are running on a Linux system named 'nano'.

```
NoMachine - limo_177
agilex@nano:~$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=false

agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_gmapping.launch
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

The screenshot shows two terminal windows side-by-side. The left window, titled 'NoMachine - limo_177', displays log messages from the YDLidar SDK initialization. It includes details about the SDK version (1.4.7), successful connection to the LiDAR, and the start of scanning at a fixed size of 500 and a sample rate of 4K. The right window, also titled 'NoMachine - limo_177' with a red header bar, shows log messages from the gmapping node. It tracks the registration of laser frames, starting with frame 0 and reaching frame 25, while maintaining a constant laser pose of 0.105 0 0 and an m_count of 1.

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x10
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.4.7
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 500
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ......

update frame 0
update ld=0 ad=0
Laser Pose= 0.105 0 0
m_count 0
Registering First Scan
update frame 25
update ld=0 ad=0
Laser Pose= 0.105 0 0
m_count 1
```

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성
 - GMapping 기반의 지도 작성
 - LIMO를 수동으로 조작하여 지도 생성이 완료되었을 경우. 나머지 터미널을 그대로 유지한 상태로 새로운 터미널을 열어 아래의 명령어를 입력하면 지도를 저장할 수 있습니다.
 - \$ cd ~/agilex_ws/src/limo_ros/limo_bringup/maps/
 - \$ rosrun map_server map_saver -f map1
 - 위의 map1은 저장될 지도의 이름이며. 원하는 이름을 입력하여 다른 이름으로 저장할 수도 있습니다. 폴더 내에 존재하는 다른 파일과 이름이 동일할 경우. 자동으로 덮어 씁니다.

05 LIMO 소프트웨어 사용 방법

- 실행 환경

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x10
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.4.7
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 500
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ......

/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_gmapping.launch http://localhost:11311 84x4
Average Scan Matching Score=155.831
neff= 29.828
Registering Scans:Done

agilex@nano:~/agilex_ws/src/limo_ros/limo_bringup/maps 84x5
agilex@nano:~/agilex_ws/src/limo_ros/limo_bringup/maps$ rosrun map_server map_saver
-f map1
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x10
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.4.7
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 500
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ......

/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_gmapping.launch http://localhost:11311 84x4
update ld=8.32667e-17 ad=0
Laser Pose= 0.105 -3.0662e-13 -2.92019e-12
m_count 46

agilex@nano:~/agilex_ws/src/limo_ros/limo_bringup/maps 84x5
[ INFO] [1649989081.015506060]: Writing map occupancy data to map1.pgm
[ INFO] [1649989081.366157292]: Writing map occupancy data to map1.yaml
[ INFO] [1649989081.366456784]: Done

agilex@nano:~/agilex_ws/src/limo_ros/limo_bringup/maps$
```

05 LIMO 소프트웨어 사용 방법

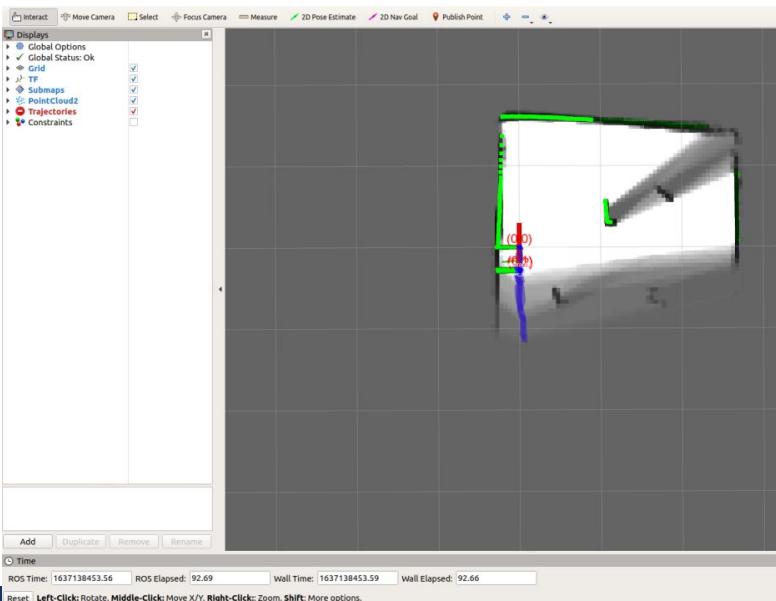
- LiDAR 기반 지도 생성
 - Cartographer mapping
 - Cartographer는 Google에서 개발한 친척화 기반의 SLAM 알고리즘입니다.
 - Cartographer의 최종 목표는 적은 계산 자원을 사용하여 실시간 SLAM을 수행하는 것입니다.
 - 알고리즘은 크게 두 가지 부분으로 구성됩니다.
 - 첫 번째 부분은 Local SLAM이라고 불리며, LaserScan을 통해서 Submap을 생성하고 관리하는 역할을 합니다. 이렇게 생성된 Submap은 Grid map의 일종입니다.
 - 두 번째 부분은 Global SLAM이라고 불리며, Loop Closure를 이용하여 Closed-Loop를 검출하여 누적된 오차를 제거합니다.
 - Submap이 생성되었을 때, 새로운 LaserScan은 Submap에 적용되지 않습니다.
 - 알고리즘은 Submap을 Closed-loop detection에 넣어 사용합니다.

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성

- Cartographer mapping

- 열려 있는 모든 터미널을 닫고, 아래의 명령어를 새로운 터미널을 열어 각각 입력
 - \$ roslaunch limo Bringup limo_start.launch pub_odom_tf:=false
 - \$ roslaunch limo Bringup limo_cartographer.launch
 - 성공적으로 실행되면 아래와 같은 Rviz가 실행됩니다.
 - 지도 생성 시, LIMO의 속도는 느리면 느릴수록 좋습니다.
 - 속도가 과하게 빠를 경우, 매팅 성능에 영향을 미칠 수 있습니다.



05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with two stacked command-line sessions. The top session is titled 'NoMachine - limo_177' and has a title bar 'agilex@nano: ~ 84x10'. It contains the command: `agilex@nano:~$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=false`. The bottom session also has a title bar 'agilex@nano: ~ 84x10' and contains the command: `agilex@nano:~$ roslaunch limo_bringup limo_cartographer.launch`.

```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=false

agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_cartographer.launch
```

05 LIMO 소프트웨어 사용 방법

- 실행 결과

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x10
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.4.7
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 500
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ......

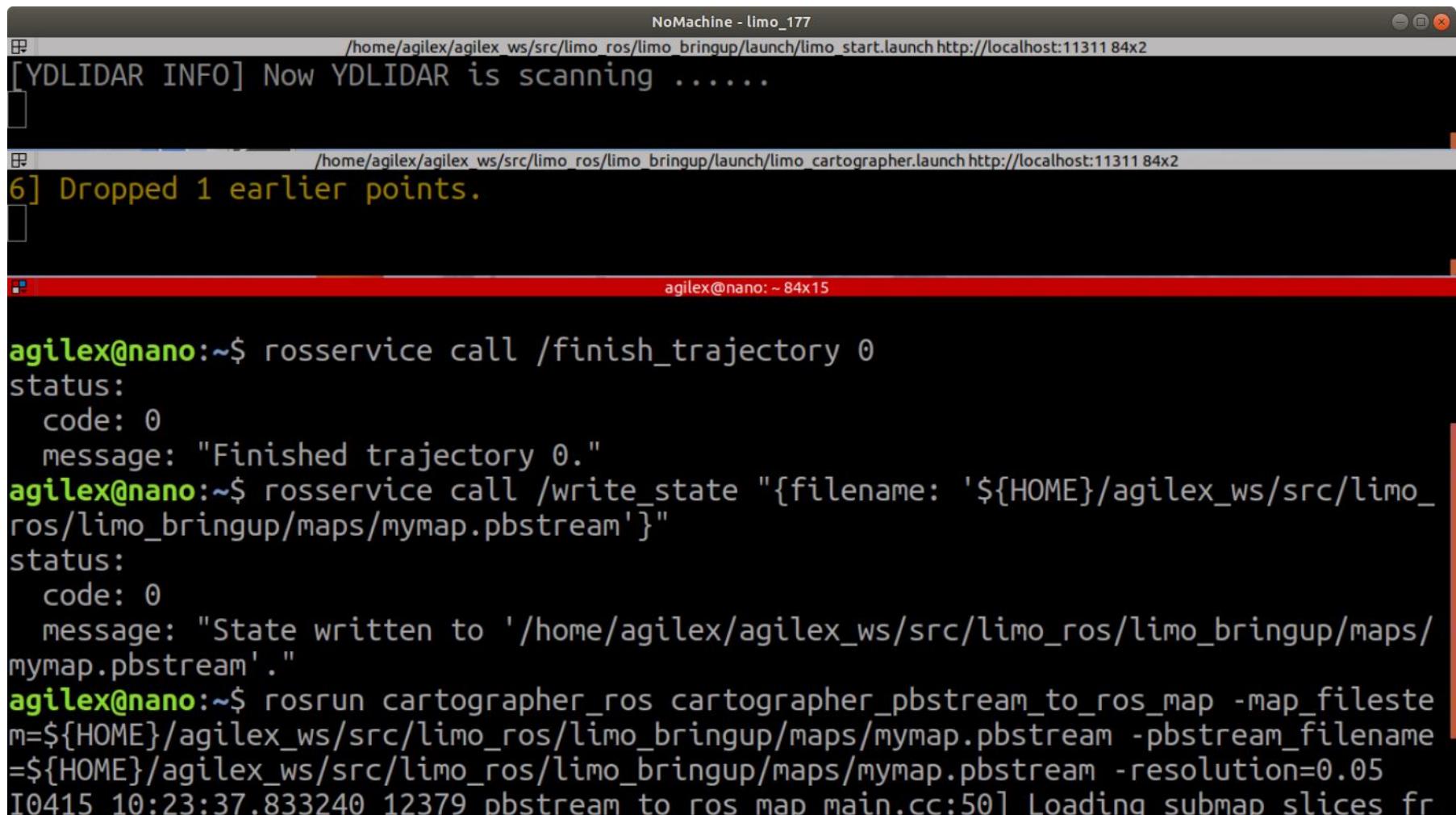
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_cartographer.launch http://localhost:11311 84x10
6] Dropped 1 earlier points.
[ WARN] [1649989179.640601561]: W0415 10:19:39.000000 11926 range_data_collator.cc:7
6] Dropped 1 earlier points.
[ WARN] [1649989179.760617351]: W0415 10:19:39.000000 11926 range_data_collator.cc:7
6] Dropped 1 earlier points.
[ WARN] [1649989180.129949603]: W0415 10:19:40.000000 11926 range_data_collator.cc:7
6] Dropped 1 earlier points.
[ WARN] [1649989180.261130548]: W0415 10:19:40.000000 11926 range_data_collator.cc:7
6] Dropped 1 earlier points.
```

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 지도 생성
 - Cartographer mapping
 - LIMO를 수동으로 조작하여 지도 생성이 완료되었을 경우, 나머지 터미널을 그대로 유지한 상태로 새로운 터미널을 열어 아래와 같이 입력합니다.
 - \$ rosservice call /finish_trajectory 0
 - \$ rosservice call /write_state "{filename: '\${HOME}/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream'}"
 - \$ rosrun cartographer_ros cartographer_pbstream_to_ros_map -map_filestem=\${HOME}/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream -pbstream_filename=\${HOME}/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream -resolution=0.05
 - 내용이 완료되면 pgm 및 yaml 파일로 된 지도가 생성됩니다.

05 LIMO 소프트웨어 사용 방법

- 실행 환경 및 결과



```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x2
[YDLIDAR INFO] Now YDLIDAR is scanning ......

/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_cartographer.launch http://localhost:11311 84x2
6] Dropped 1 earlier points.

agilex@nano: ~ 84x15

agilex@nano:~$ rosservice call /finish_trajectory 0
status:
  code: 0
  message: "Finished trajectory 0."
agilex@nano:~$ rosservice call /write_state "{filename: '${HOME}/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream'}"
status:
  code: 0
  message: "State written to '/home/agilex/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream'."
agilex@nano:~$ rosrun cartographer_ros cartographer_pbstream_to_ros_map -map_fileste
m=${HOME}/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream -pbstream_filename
=${HOME}/agilex_ws/src/limo_ros/limo_bringup/maps/mymap.pbstream -resolution=0.05
I0415 10:23:37.833240 12379 pbstream_to_ros_map_main.cc:50] Loading submap slices fr
```

05 LIMO 소프트웨어 사용 방법

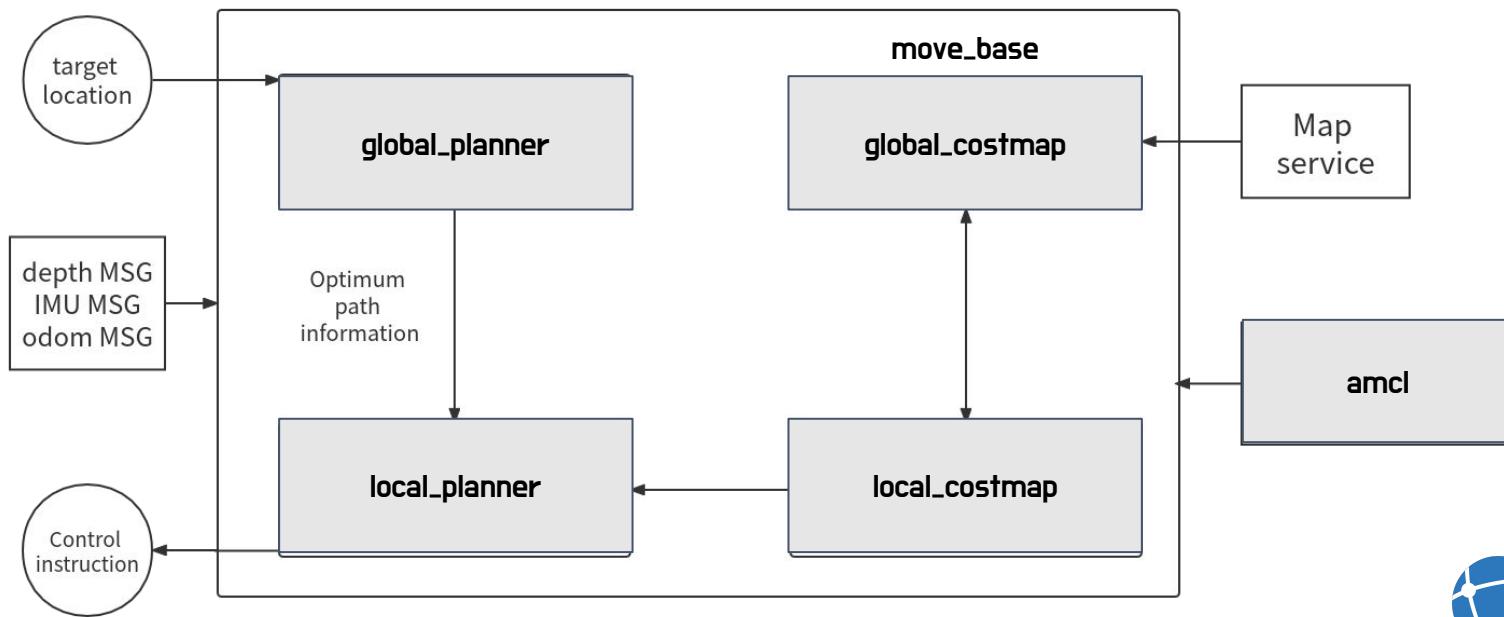
- LiDAR 기반 지도 생성
 - Cartographer mapping
 - 지도 생성 수행 시, 다음과 같은 경고가 터미널에 출력될 수 있습니다.
 - 이는 과도한 속도 때문이거나 데이터 처리 속도의 자연으로 발생한 문제입니다.
 - 알고리즘 자체에 문제가 되지 않으므로 그대로 사용하셔도 됩니다.

```
process[cartographer_node-1]: started with pid [10493]
process[cartographer_occupancy_grid_node-2]: started with pid [10494]
process[rviz-3]: started with pid [10495]
[ WARN] [1638342058.059636324]: W1201 15:00:58.000000 10493 range_data_collator.cc:76] Dropped 5 earlier points.
[ WARN] [1638342059.645684251]: W1201 15:00:59.000000 10493 range_data_collator.cc:76] Dropped 2 earlier points.
[ WARN] [1638342082.102137014]: W1201 15:01:22.000000 10493 range_data_collator.cc:76] Dropped 1 earlier points.
[ WARN] [1638342086.339528880]: W1201 15:01:26.000000 10493 range_data_collator.cc:76] Dropped 1 earlier points.
[ WARN] [1638342100.216579145]: W1201 15:01:40.000000 10493 range_data_collator.cc:76] Dropped 1 earlier points.
```

05 LIMO 소프트웨어 사용 방법

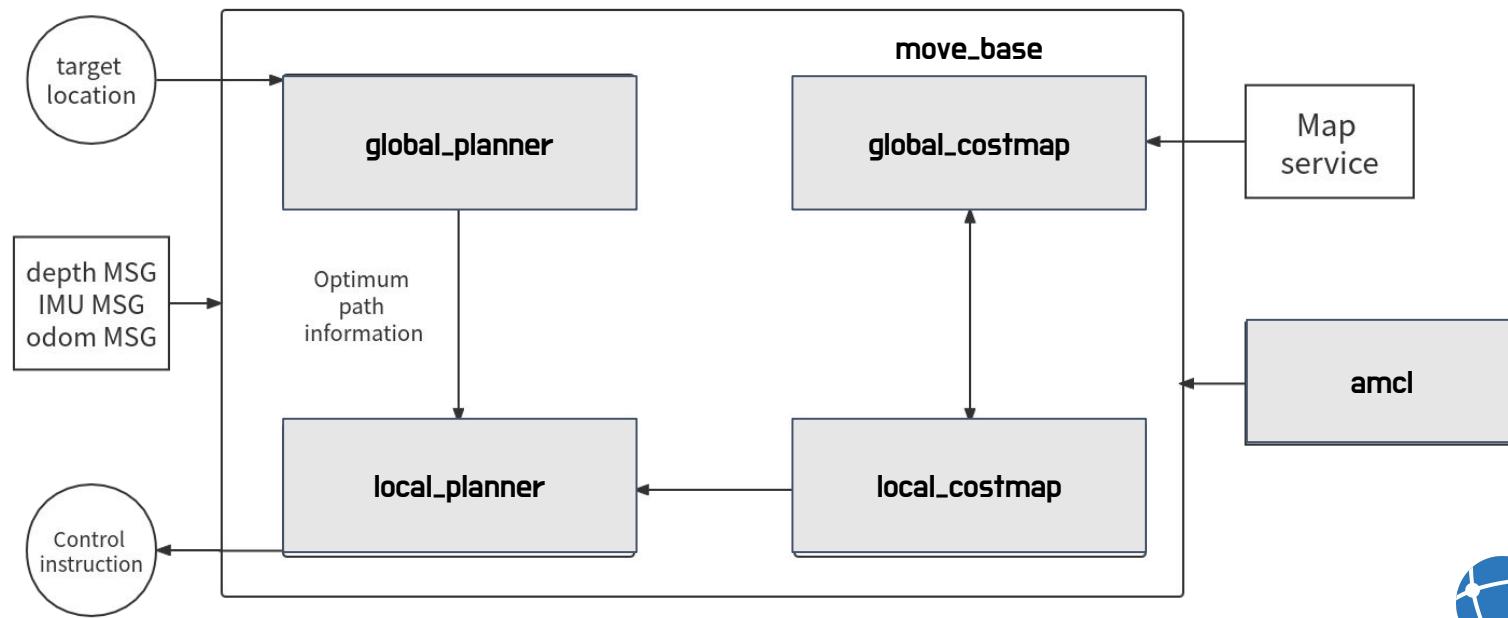
- LiDAR 기반 Navigation (자율 주행)

- 이전에 생성한 지도를 기반으로 자율주행
- Navigation의 핵심은 지도 상의 로봇의 위치를 파악하는 것과 경로를 생성하는 것.
두 가지이며, ROS는 이에 대해 각각 패키지를 제공합니다.
- move_base : Robot 자율주행에서 최적 경로 생성을 수행
- amcl : 2차원 Grid-map에서 로봇의 위치를 추정



05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)
 - 로봇에서 전달해야 할 값은 필요한 센서 데이터와 목표 장소입니다.
 - move_base 패키지가 자율 주행을 직접적으로 처리합니다.
 - 자율 주행을 정확히 수행하기 위해서는 로봇의 위치를 정확하게 알아야하며, 이는 amcl 패키지에서 수행합니다.



05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)
 - move_base package
 - move_base는 경로 계획을 위한 ROS 패키지입니다.
 - move_base는 Global Path Planning과 Local real-time Planning으로 이루어집니다.
 - Global Path Planning은 현재 위치에서 주어진 목표 장소까지를 연결하는 경로를 그립니다.
 - Dijkstra 또는 A* 등의 Global Path Search 알고리즘이 일반적으로 사용됩니다.
 - 실제 환경에서 로봇이 Global Path를 정확하게 따라갈 수 없으므로, 로봇이 따라갈 수 있는 경로를 실시간으로 생성하는 것이 Local Path Planning의 역할입니다.
 - 지도의 정보와 센서에서 얻어진 장애물 정보 등을 활용하여 경로를 생성하며, Global Path를 가능한 따라가도록 설계되어 있습니다.

05 LIMO 소프트웨어 사용 방법

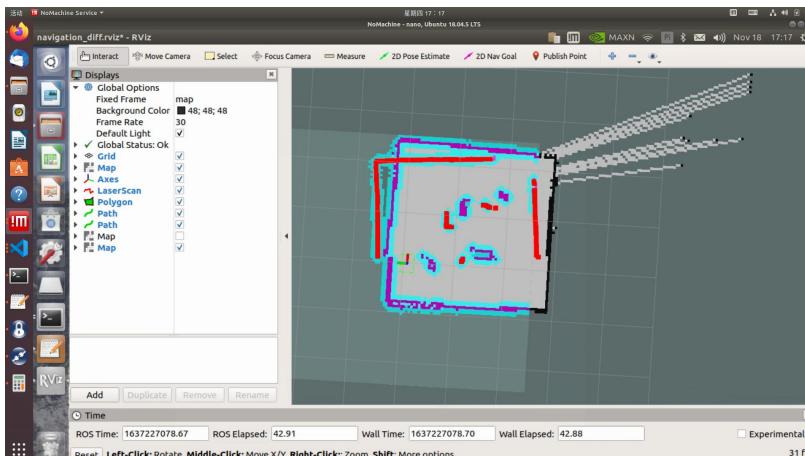
- LiDAR 기반 Navigation (자율 주행)
 - amcl package
 - 자동으로 위치를 추정하는 것은 지도 상의 로봇의 위치를 어떤 상태에서도 계산할 수 있다는 것을 의미합니다.
 - ROS는 2차원에서 움직이는 로봇의 위치를 Particle filter를 기반으로 확률적으로 추정하는 adaptive monte carlo localization를 제공합니다.

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)
 - DWA Planner
 - DWA는 Dynamic Window Approaches를 뜻합니다.
 - 이 알고리즘은 피하거나 이동할 다수의 경로를 생성하며, 이를 이용하여 다양한 평가를 거친 후 최적의 경로를 1개만 선택합니다. (경로와 장애물이 충돌하는지, 시간이 얼마나 걸리는지 등)
 - 그리고 경로를 따라가기 위한 선속도와 회전 속도를 계산하여 운전 중 움직이는 물체와의 충돌을 회피할 수 있도록 합니다.
 - TEB Planner
 - TEB은 Time Elastic Band Local Planner를 뜻합니다.
 - 이 방법은 Global Path Planner에서 생성된 경로에 추가적인 수정을 수행하여 로봇의 움직임을 최적화 하는 Local Path Planning입니다.
 - 경로 최적화 과정에서, 알고리즘은 전체 경로 길이, 경로 수행 시간, 장애물과의 거리, 중간 경유점 통과, 로봇의 동역학적 고려, 기하학적 제약 등을 일부 또는 전부 포함하는 최적화 목표를 가집니다.
 - TEB 방식은 특히 움직임에 따른 시간과 공간의 달라지는 제약조건을 고려합니다.

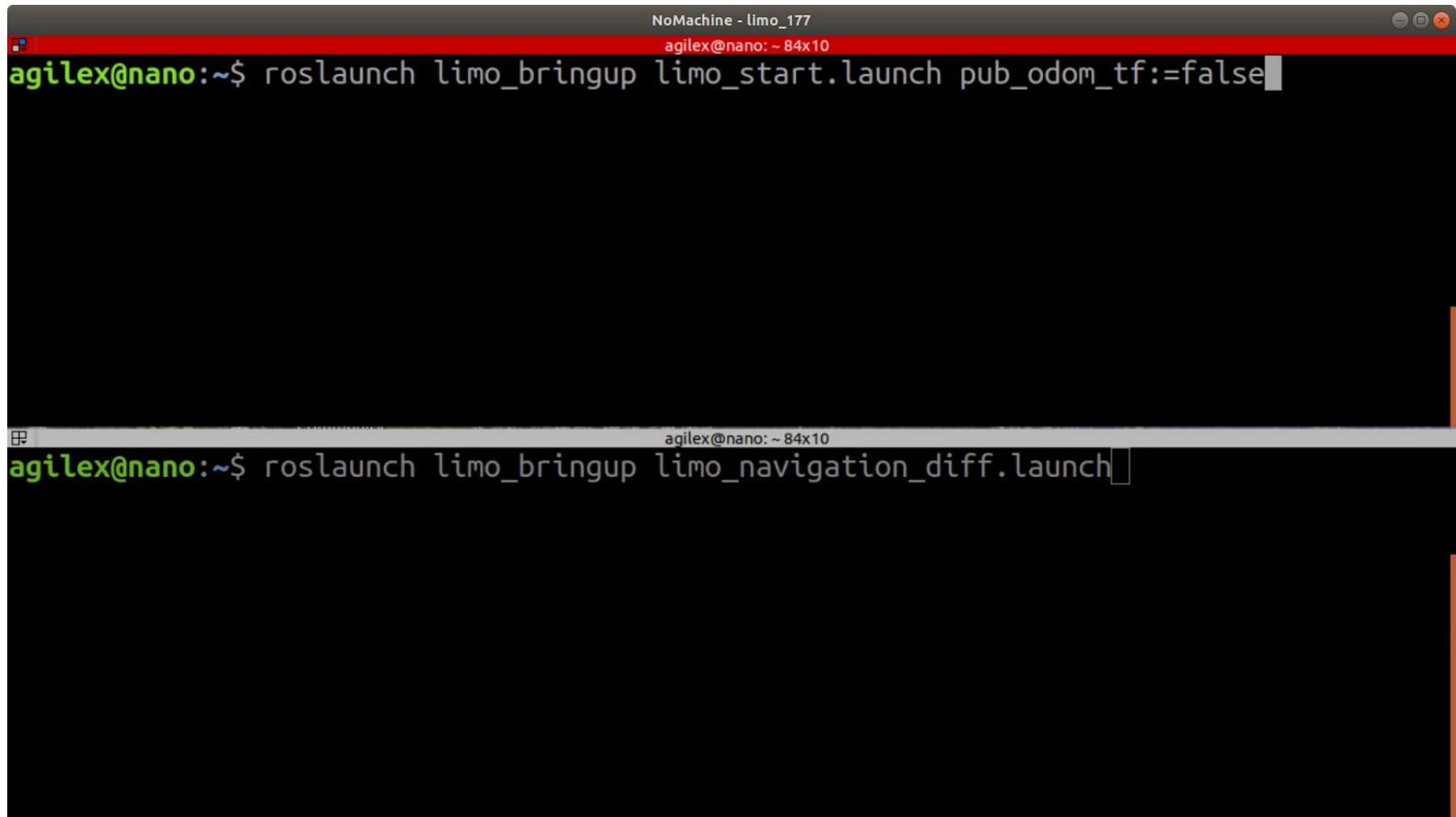
05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)
 - Four-wheel differential mode, mecanum mode, track mode에서 동일하게 동작합니다. 터미널을 모두 닫고 아래의 명령어를 새로운 터미널에 각각 입력합니다.
 - \$ roslaunch limo Bringup limo_start.launch pub_odom_tf:=false
 - \$ roslaunch limo Bringup limo_navigation_diff.launch
 - 만약 Ackermann mode인 경우 위의 명령어가 아닌 아래의 명령어를 각각 입력합니다.
 - \$ roslaunch limo Bringup limo_start.launch pub_odom_tf:=false
 - \$ roslaunch limo Bringup limo_navigation_ackerman.launch
 - 실행이 완료되면, Rviz가 다음과 같이 실행됩니다.



05 LIMO 소프트웨어 사용 방법

- 실행 환경 (Not Ackermann mode)



```
NoMachine - limo_177
agilex@nano: ~ 84x10
agilex@nano:~$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=false

agilex@nano:~$ roslaunch limo_bringup limo_navigation_diff.launch
```

05 LIMO 소프트웨어 사용 방법

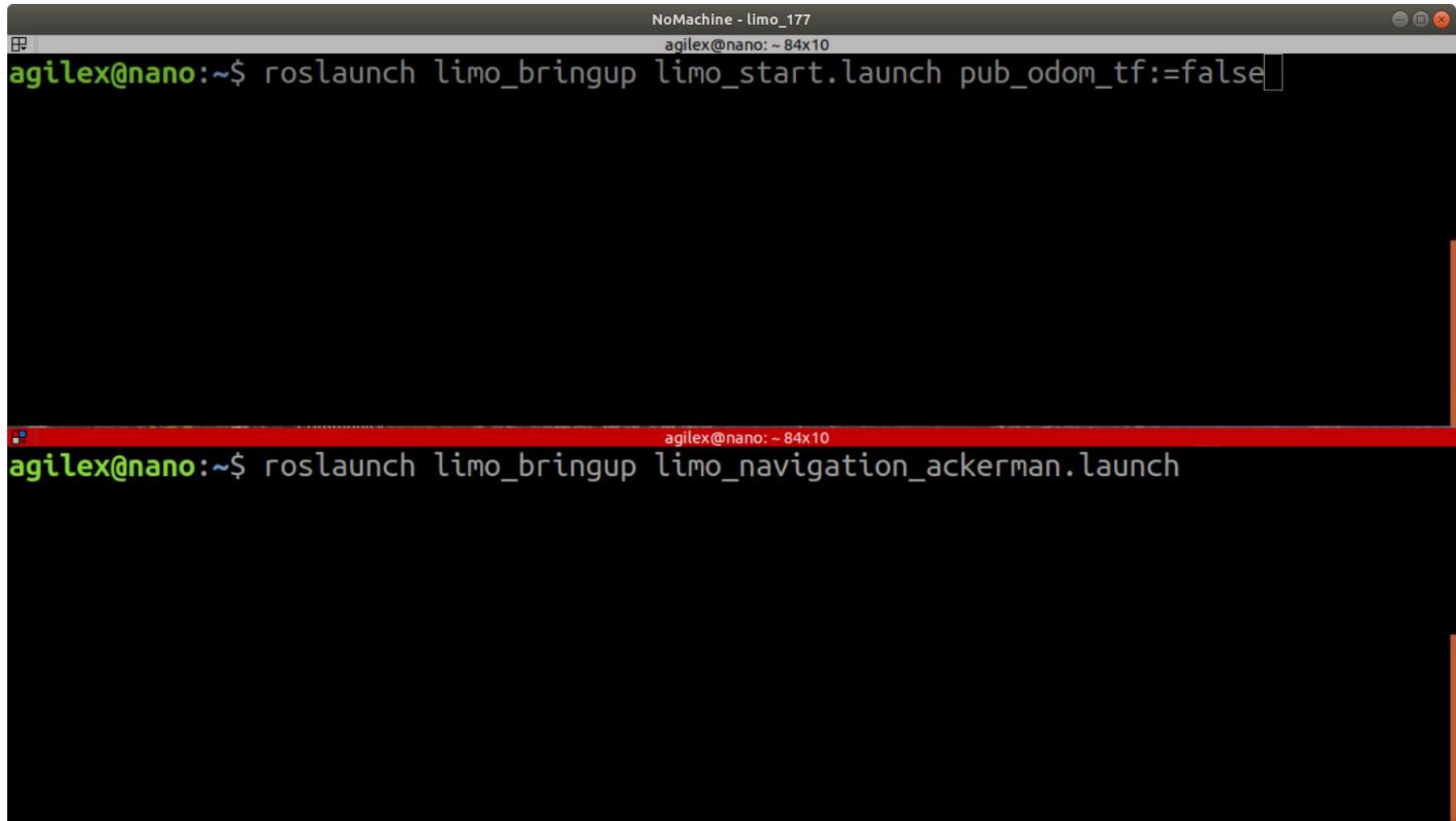
- 실행 결과 (Not Ackermann mode)

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x10
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.4.7
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 490
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ......

lannerROS
[ INFO] [1649989590.506376814]: Sim period is set to 0.05
[ WARN] [1649989590.560828304]: Parameter pdist_scale is deprecated. Please use the
name path_distance_bias instead.
[ WARN] [1649989590.590528464]: Parameter gdist_scale is deprecated. Please use the
name goal_distance_bias instead.
[ INFO] [1649989591.759540111]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1649989591.827602338]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1649989592.091863148]: odom received!
```

05 LIMO 소프트웨어 사용 방법

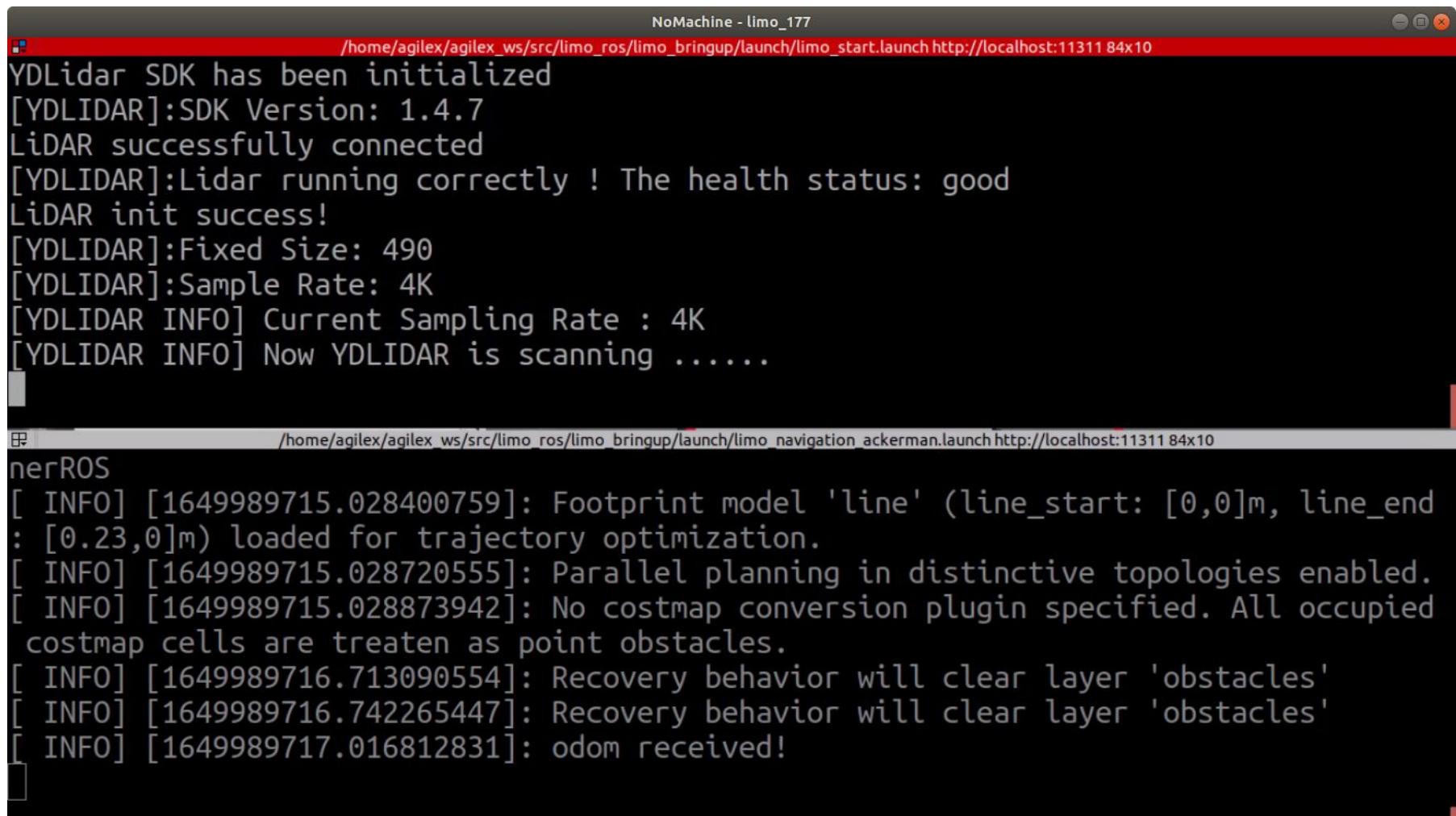
- 실행 환경 (Ackermann mode)



The screenshot shows two terminal windows side-by-side. The top window has a dark gray header bar with the text "NoMachine - limo_177" and "agilex@nano: ~ 84x10". The main body of the window is black and contains the command: "agilex@nano:~\$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=false". The bottom window has a red header bar with the text "agilex@nano: ~ 84x10". The main body of the window is black and contains the command: "agilex@nano:~\$ roslaunch limo_bringup limo_navigation_ackerman.launch". Both windows have standard Linux-style window controls (minimize, maximize, close) at the top right.

05 LIMO 소프트웨어 사용 방법

- 실행 결과 (Ackermann mode)



The screenshot shows two terminal windows side-by-side. The top window is titled 'NoMachine - limo_177' and displays logs related to the YDLidar SDK initialization. The bottom window is titled 'nerROS' and displays logs related to ROS navigation stack configuration.

```
NoMachine - limo_177
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311 84x10
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.4.7
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 490
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ......

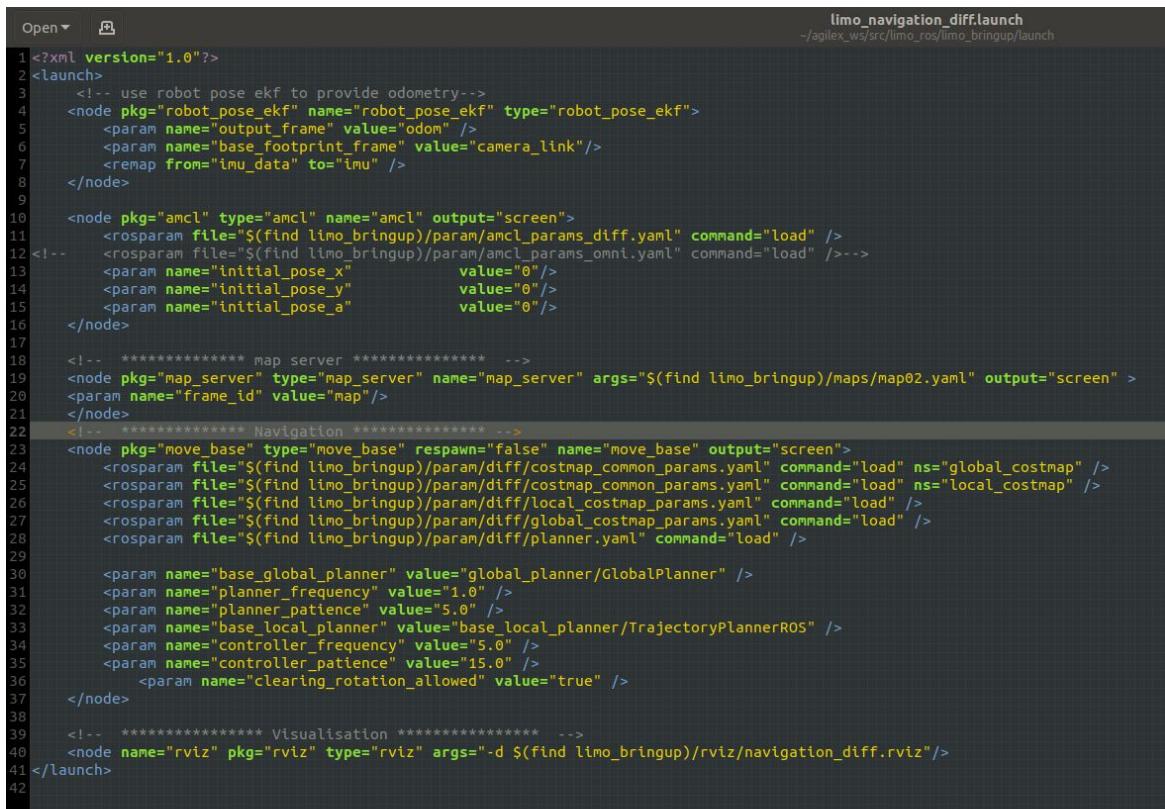
nerROS
[ INFO] [1649989715.028400759]: Footprint model 'line' (line_start: [0,0]m, line_end: [0.23,0]m) loaded for trajectory optimization.
[ INFO] [1649989715.028720555]: Parallel planning in distinctive topologies enabled.
[ INFO] [1649989715.028873942]: No costmap conversion plugin specified. All occupied costmap cells are treaten as point obstacles.
[ INFO] [1649989716.713090554]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1649989716.742265447]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1649989717.016812831]: odom received!
```

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)

- 만약 지도를 다른 지도로 변경하고 싶다면 limo_navigation_diff.launch 또는 limo_navigation_ackerman.launch를 열어 파라미터를 수정합니다. (map02를 수정)

```
<node pkg="map_server" ... args="$(find limo_bringup)/maps/map02.yaml ...>
```



```
Open ▾ limo_navigation_diff.launch
~/agilex_ws/src/limo_ros/limo_bringup/launch

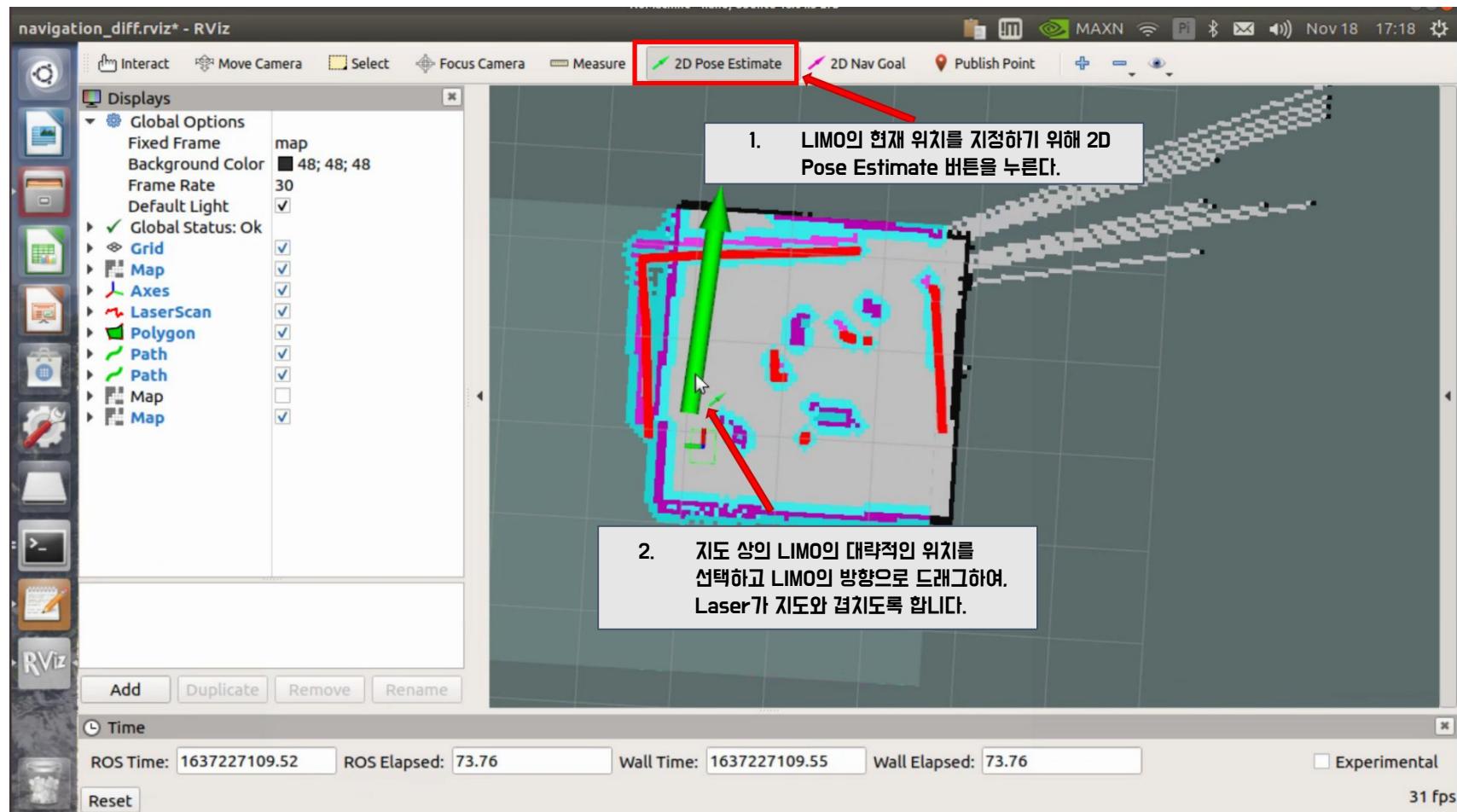
1 <?xml version="1.0"?>
2 <launch>
3   <!-- use robot pose ekf to provide odometry-->
4   <node pkg="robot_pose_ekf" name="robot_pose_ekf" type="robot_pose_ekf">
5     <param name="output_frame" value="odom" />
6     <param name="base_footprint_frame" value="camera_link" />
7     <remap from="imu_data" to="imu" />
8   </node>
9
10  <node pkg="amcl" type="amcl" name="amcl" output="screen">
11    <rosparam file="$(find limo_bringup)/param/amcl_params_diff.yaml" command="load" />
12  <!-- <rosparam file="$(find limo_bringup)/param/amcl_params_omni.yaml" command="load" /-->
13    <param name="initial_pose_x" value="0"/>
14    <param name="initial_pose_y" value="0"/>
15    <param name="initial_pose_a" value="0"/>
16  </node>
17
18  <!-- ***** map server ***** -->
19  <node pkg="map_server" type="map_server" name="map_server" args="$(find limo_bringup)/maps/map02.yaml" output="screen" >
20    <param name="frame_id" value="map"/>
21  </node>
22  <!-- ***** Navigation ***** -->
23  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
24    <rosparam file="$(find limo_bringup)/param/diff/costmap_common_params.yaml" command="load" ns="global_costmap" />
25    <rosparam file="$(find limo_bringup)/param/diff/costmap_common_params.yaml" command="load" ns="local_costmap" />
26    <rosparam file="$(find limo_bringup)/param/diff/local_costmap_params.yaml" command="load" />
27    <rosparam file="$(find limo_bringup)/param/diff/global_costmap_params.yaml" command="load" />
28    <rosparam file="$(find limo_bringup)/param/diff/planner.yaml" command="load" />
29
30    <param name="base_global_planner" value="global_planner/GlobalPlanner" />
31    <param name="planner_frequency" value="1.0" />
32    <param name="planner_patience" value="5.0" />
33    <param name="base_local_planner" value="base_local_planner/TrajectoryPlannerROS" />
34    <param name="controller_frequency" value="5.0" />
35    <param name="controller_patience" value="15.0" />
36    <param name="clearing_rotation_allowed" value="true" />
37  </node>
38
39  <!-- ***** Visualisation ***** -->
40  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find limo_bringup)/rviz/navigation_diff.rviz"/>
41 </launch>
```

05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)
 - Navigation이 실행되면, 지도와 현재 LaserScan 데이터가 일치하지 않는 것을 확인할 수 있습니다. 이는 수동으로 보정할 수 있습니다.
 - Rviz에 표시되는 2D Pose Estimate 버튼을 클릭하여, LIMO의 대략적인 위치를 전달하고, 조종기/APP을 이용하여 LIMO를 전후진 및 회전하면 자동으로 LaserScan 데이터와 지도가 일치합니다.

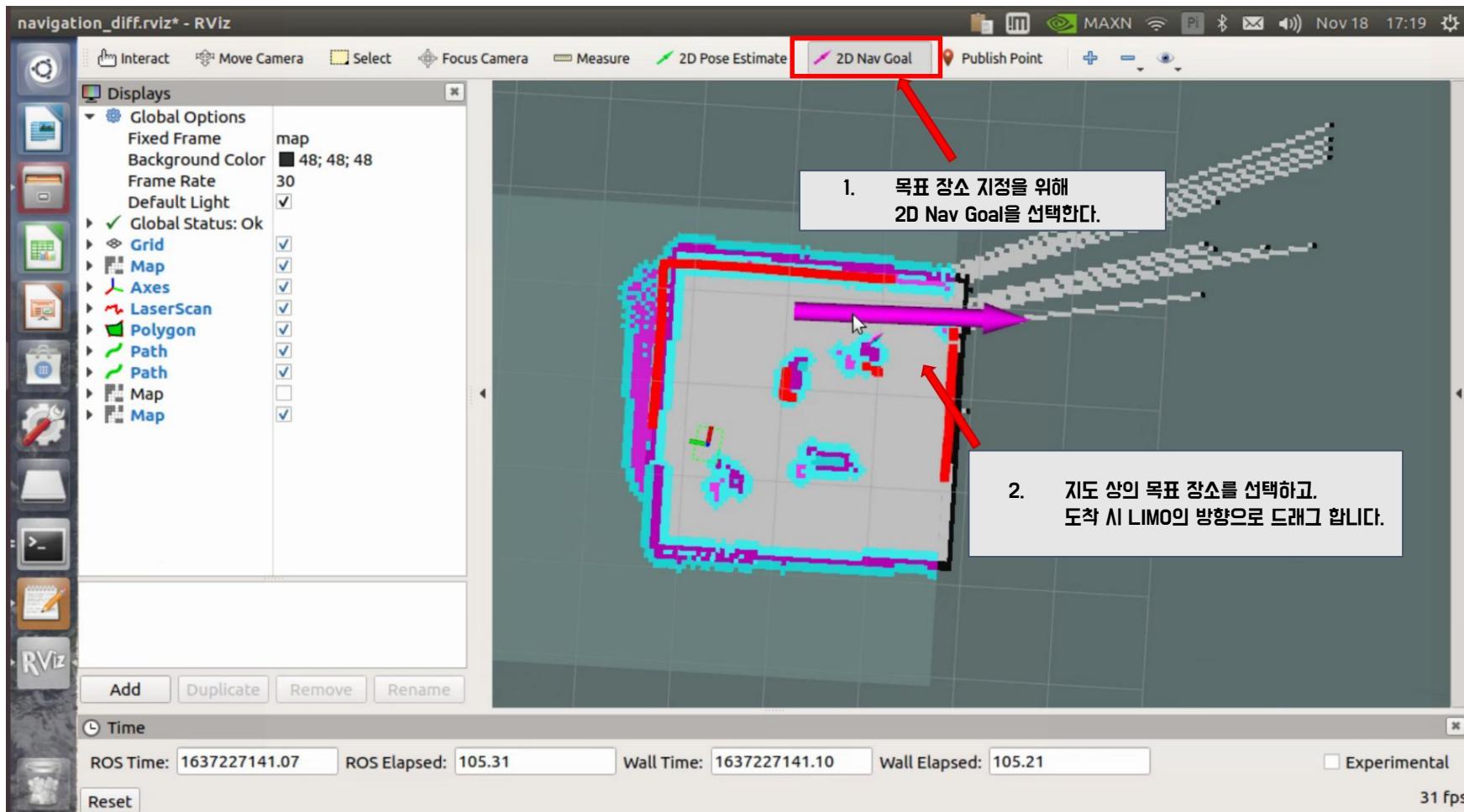
05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)



05 LIMO 소프트웨어 사용 방법

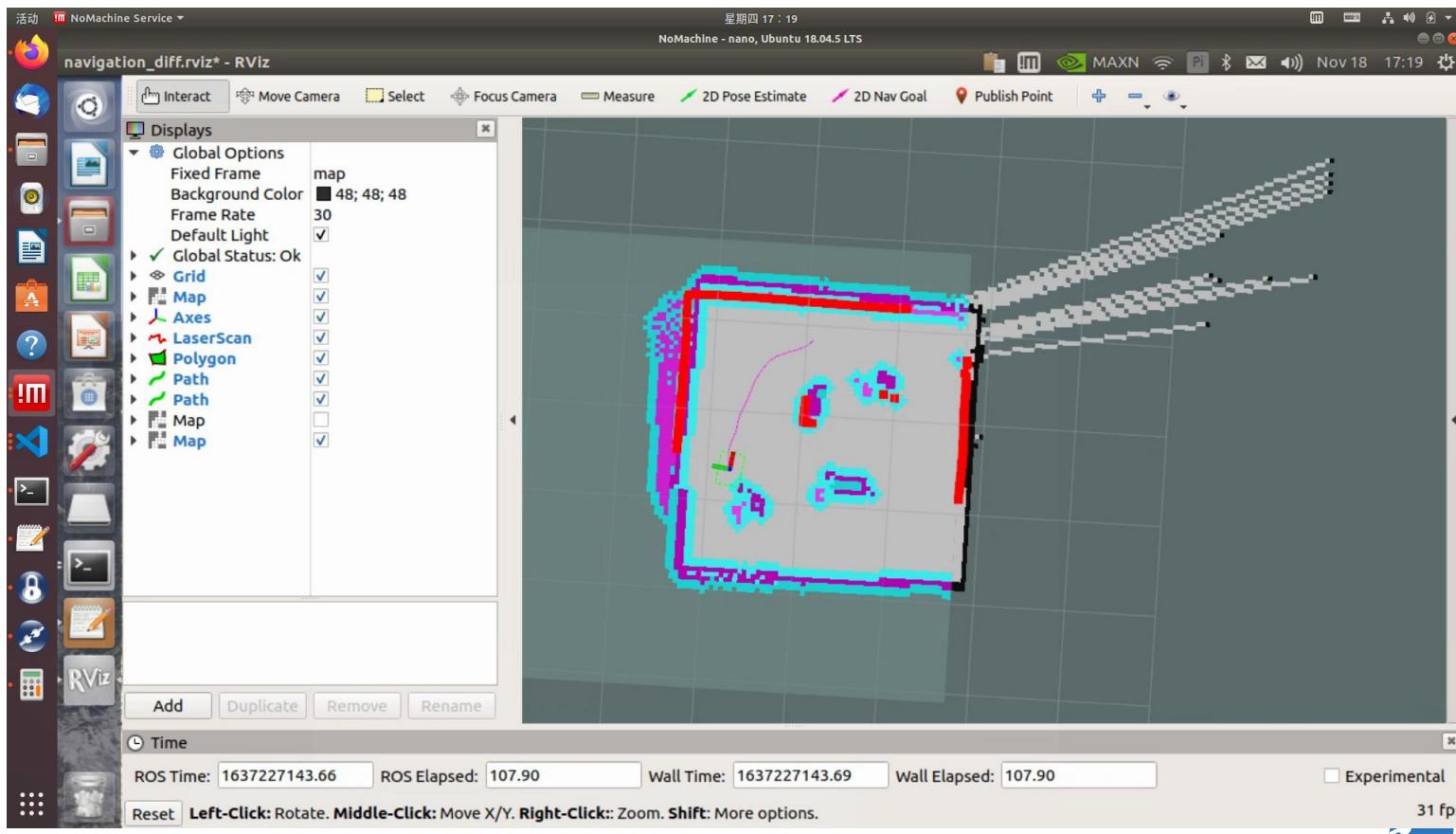
- LiDAR 기반 Navigation (자율 주행)



05 LIMO 소프트웨어 사용 방법

- LiDAR 기반 Navigation (자율 주행)

- 다음과 같이 보라색으로 경로가 생성된 후, 조종기의 SWB를 자동으로 변경하거나, APP의 ON/OFF를 OFF로 설정하면, LIMO가 목표 장소로 자율주행합니다.



05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - LIMO는 Intel RealSense D435가 장착된 버전과 ORBBEC Dabai가 장착된 두 가지 버전이 있습니다.
 - 두 맵스 카메라 모두 vision + LiDAR 기반의 지도 작성과 자율 주행에 사용할 수 있습니다.

05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - ORBBEC Dabai
 - ORBBEC Dabai는 양안 Structured light 기반 Depth 카메라입니다.
 - 좌 우측의 적외선 카메라를 1대씩 가지고 있으며, 적외선 프로젝터와 Depth 처리 프로세서를 포함하고 있습니다.
 - 적외선 프로젝터는 Structured light pattern을 발사합니다.
 - 좌우측의 적외선 카메라는 각각 발사된 Structured light image를 받고, Depth 처리 프로세서에서 Depth를 계산합니다.

05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping

- ORBBEC Dabai

파라미터	파라미터 값
좌우측 적외선 카메라 사이의 거리	40mm
Depth 측정 범위	0.3-3m
소모 전력	평균 전력 <2W. 적외선 발사 시 <5W (3ms간). 평균 대기 전력 <0.7W
Depth 해상도	640x400@30FPS; 320x200@30FPS
RGB 해상도	1920x1080@30FPS; 1280x720@30FPS; 640x480@30FPS
Depth 정확도	6mm@1m (81% FOV area participates in accuracy calculation*)
Depth FOV	H 67.9° V 45.3°
RGB FOV	H 71° V43.7° @1920X1080
Delay	30-45ms
Data transmission	USB2.0 or above
Operating temperature	10°C ~ 40°C
Applicable scene	Indoor / outdoor (적용하는 환경 및 알고리즘에 따라 다름)
Dustproof and waterproof	방진

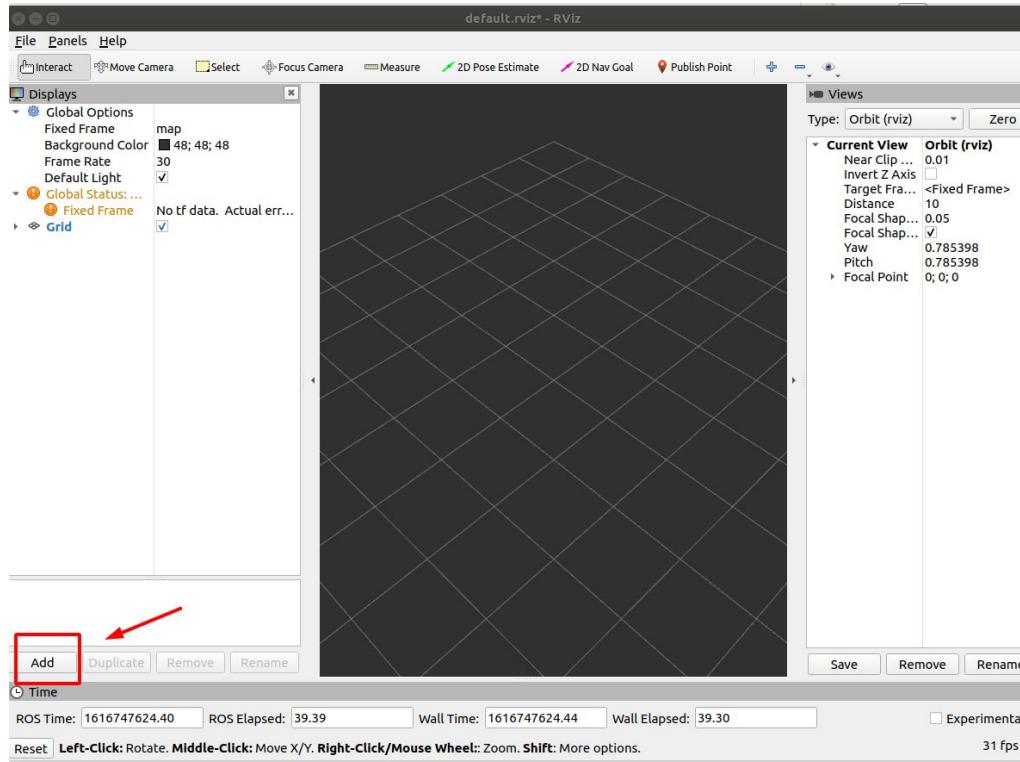
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - ORBBEC Dabai
 - 열려 있는 모든 터미널을 다 종료한 후, 아래의 명령어를 새로운 터미널에 입력합니다.
 - \$ roslaunch astra_camera dabai_u3.launch
 - 다음과 같은 경고가 실행 중에 발생합니다. 무시하셔도 됩니다.

```
attempt to claim already-claimed interface 1
[ WARN] [1638252908.984689488]: Unable to set scanning_mode to 0
[ WARN] [1638252908.986331647]: Unable to set auto_focus to 1
[ WARN] [1638252908.987202340]: Unable to set iris_absolute to 0
[ WARN] [1638252908.988015164]: Unable to set pantilt to -648000, 648000
[ WARN] [1638252909.216563391]: Camera calibration file /home/agilex/.ros/camera_info/camera.yaml not found.
```

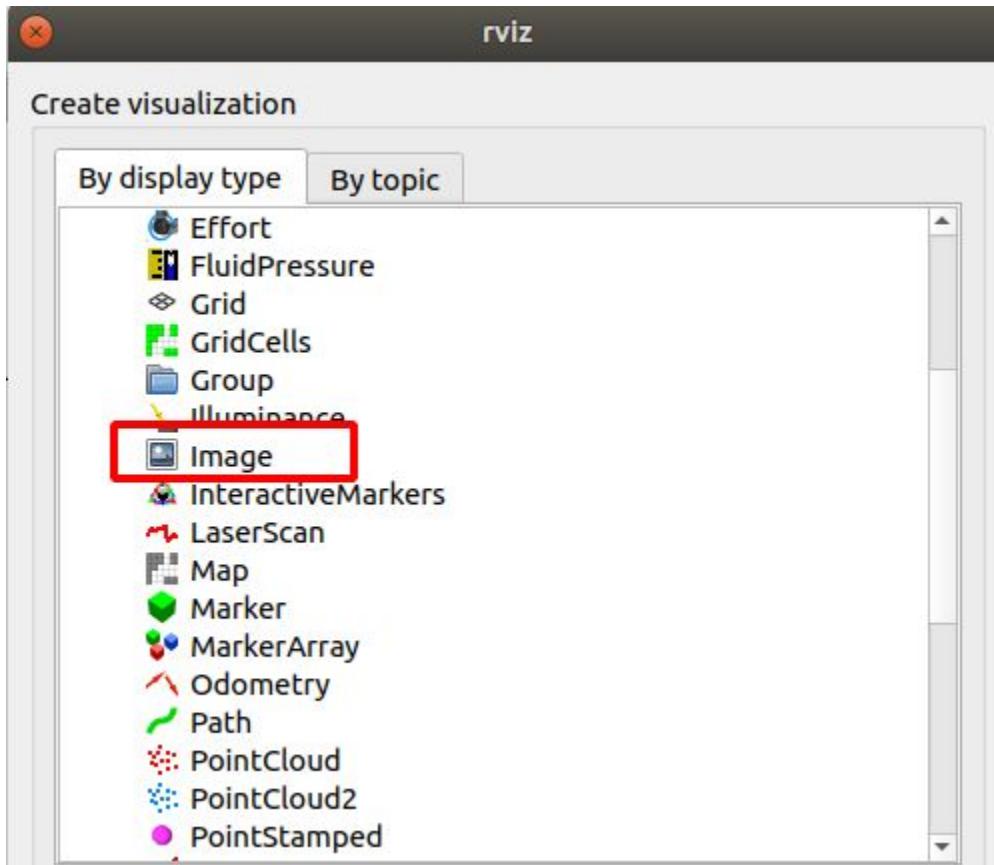
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - Depth Camera가 실행이 되면 새로운 터미널을 열어 다음과 같이 입력합니다.
 - \$ rviz
 - 아래와 같은 창에 버튼을 순서대로 입력하여 이미지를 확인합니다.



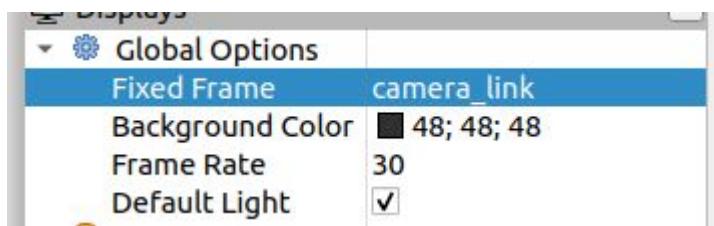
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기



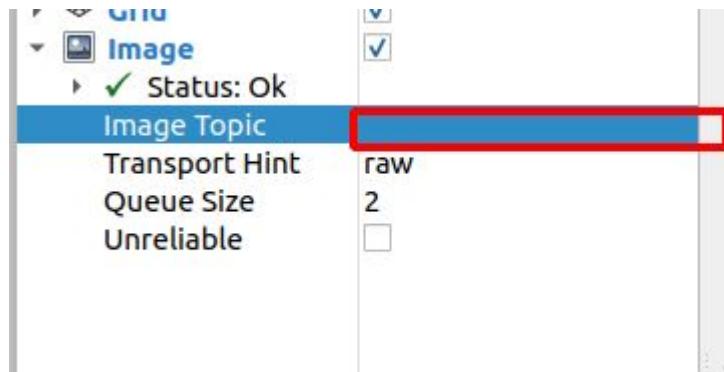
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - 좌측 상단의 Global Options의 Fixed Frame을 camera_link로 변경합니다.



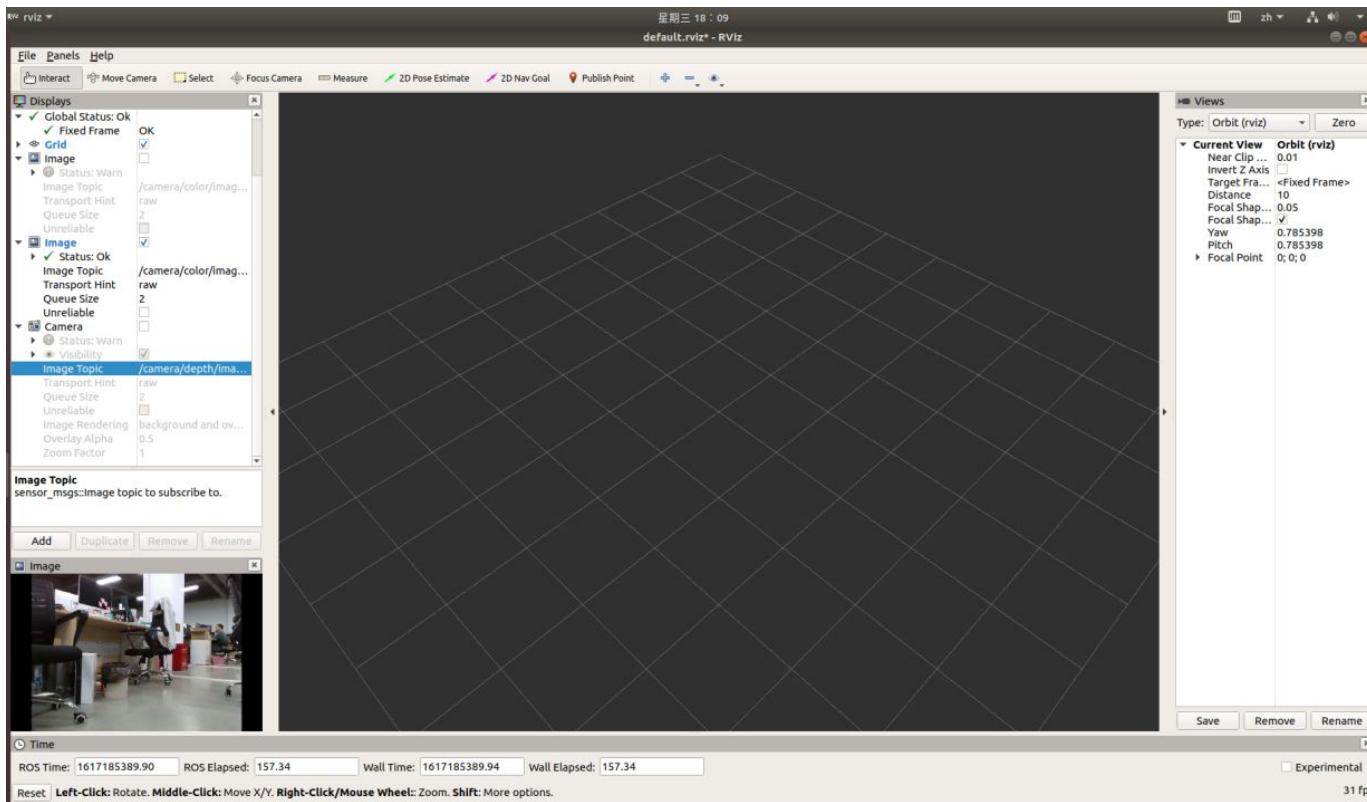
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - 추가된 Image의 Image Topic 부분을 RGB 이미지로 맞춥니다.
`/camera/rgb/image_rect_color`



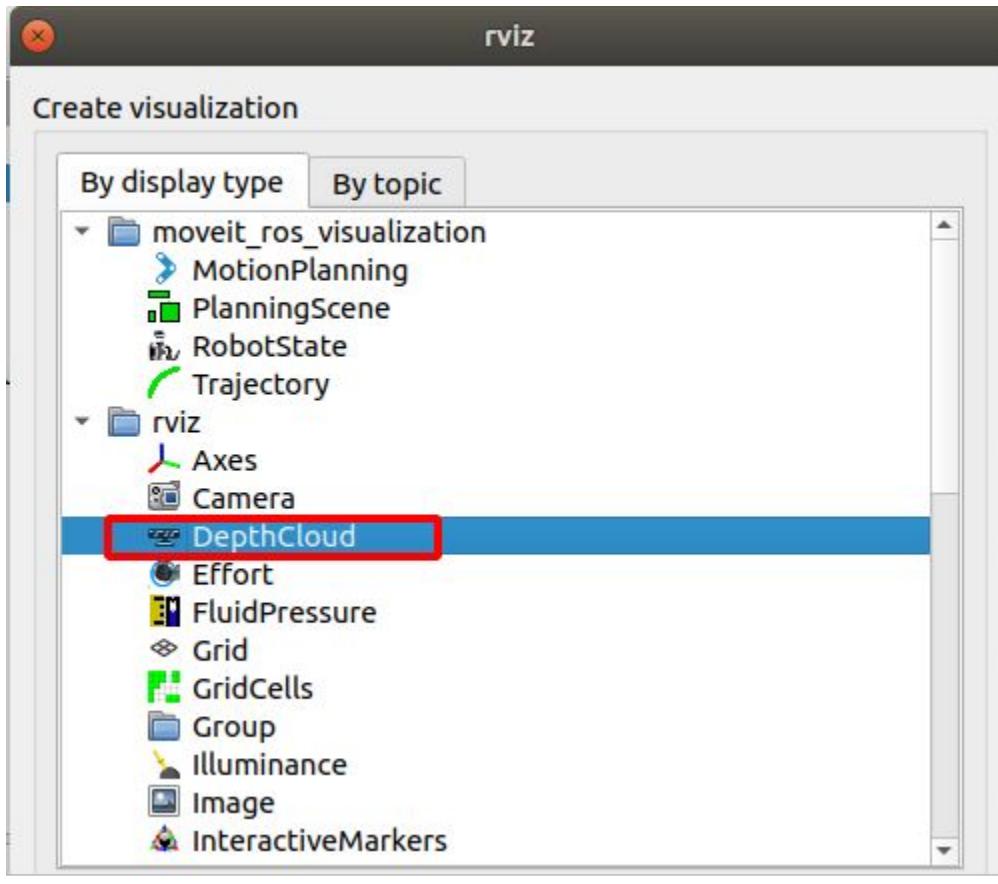
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - 다음과 같이 RGB 이미지를 확인할 수 있습니다.



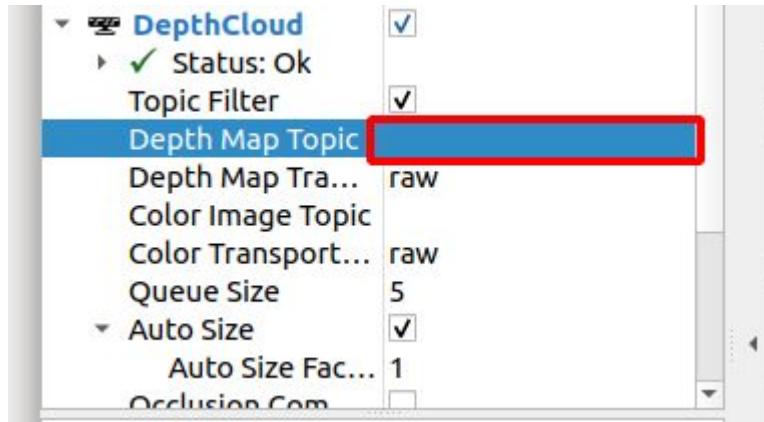
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - 포인트 클라우드 확인을 위해서는 DepthCloud를 추가합니다.



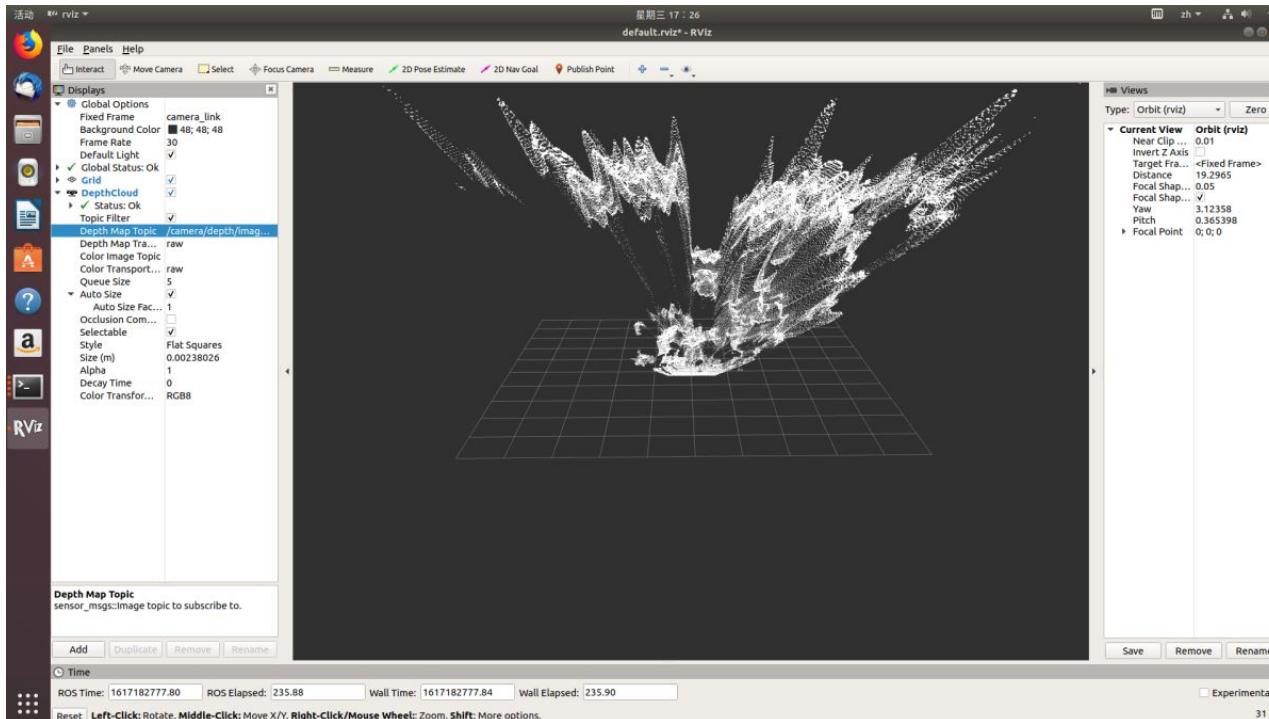
05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - 마찬가지로 Depth Map Topic을 Depth로 맞춰줍니다.
 - `/camera/depth/image_rect`



05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - Depth Camera 정보 확인하기
 - 다음과 같은 Depth Cloud를 확인할 수 있습니다.



05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - RTABMap 알고리즘
 - RTABMap 알고리즘은 센서 데이터를 기반으로 한 위치 추정 및 지도 생성을 수행합니다.
 - 넓은 영역에서의 실시간 closed-loop detection을 해결하는 것을 목표로 하고 있습니다.
 - Closed-loop detection은 전체 맵의 수많은 점들 중 한정된 숫자의 점들만을 사용합니다.
 - 열려 있는 모든 터미널을 닫고 새로운 터미널을 열어 아래의 명령어를 각각 실행합니다.
 - \$ roslaunch limo Bringup limo_start.launch pub_odom_tf:=true
 - \$ roslaunch astra_camera dabai_u3.launch
 - \$ roslaunch limo Bringup limo_rtabmap_orbbec.launch
 - \$ roslaunch limo Bringup rtabmap_rviz.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경

```
agilex@nano:~$ rosrun limo Bringup limo_start.launch pub_odom_tf:=true
agilex@nano:~$ rosrun astra_camera dabai_u3.launch
agilex@nano:~$ rosrun limo Bringup limo_rtabmap_orbbec.launch
agilex@nano:~$ rosrun limo Bringup rtabmap_rviz.launch
```

05 LIMO 소프트웨어 사용 방법

• 실행 결과

```
NoMachine - limo_177
[home/agilex/agilex ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311]
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 500
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ..
....
```



```
[home/agilex/agilex ws/src/limo_ros/limo_bringup/launch/limo_rtabmap_orbbec.launch http://localhost:11311]
.0128s pub=0.0001s (local map=1, WM=1)
[ INFO] [1649990109.935485389]: rtabmap (26): Rate=1.00s, Limit=0.000s, Conversion =0.0075s, RTAB-Map=0.3549s, Maps update=0
.0041s pub=0.0001s (local map=1, WM=1)
[ INFO] [1649990110.742483004]: rtabmap (27): Rate=1.00s, Limit=0.000s, Conversion =0.0123s, RTAB-Map=0.1656s, Maps update=0
.0145s pub=0.0001s (local map=1, WM=1)
```



```
[home/agilex/agilex ws/src/ros_astra_camera/launch/dabai_u3.launch http://localhost:11311]
libration URL: file:///home/agilex/.ros/camera_info/depth_Astra_Orbbec.yaml
[ INFO] [1649990074.242335451]: Unable to open camera calibration file [/home/agilex/.ros/camera_info/depth_Astra_Orbbec.yaml]
[ WARN] [1649990074.242567231]: Camera calibration file /home/agilex/.ros/camera_info/depth_Astra_Orbbec.yaml not found.
```



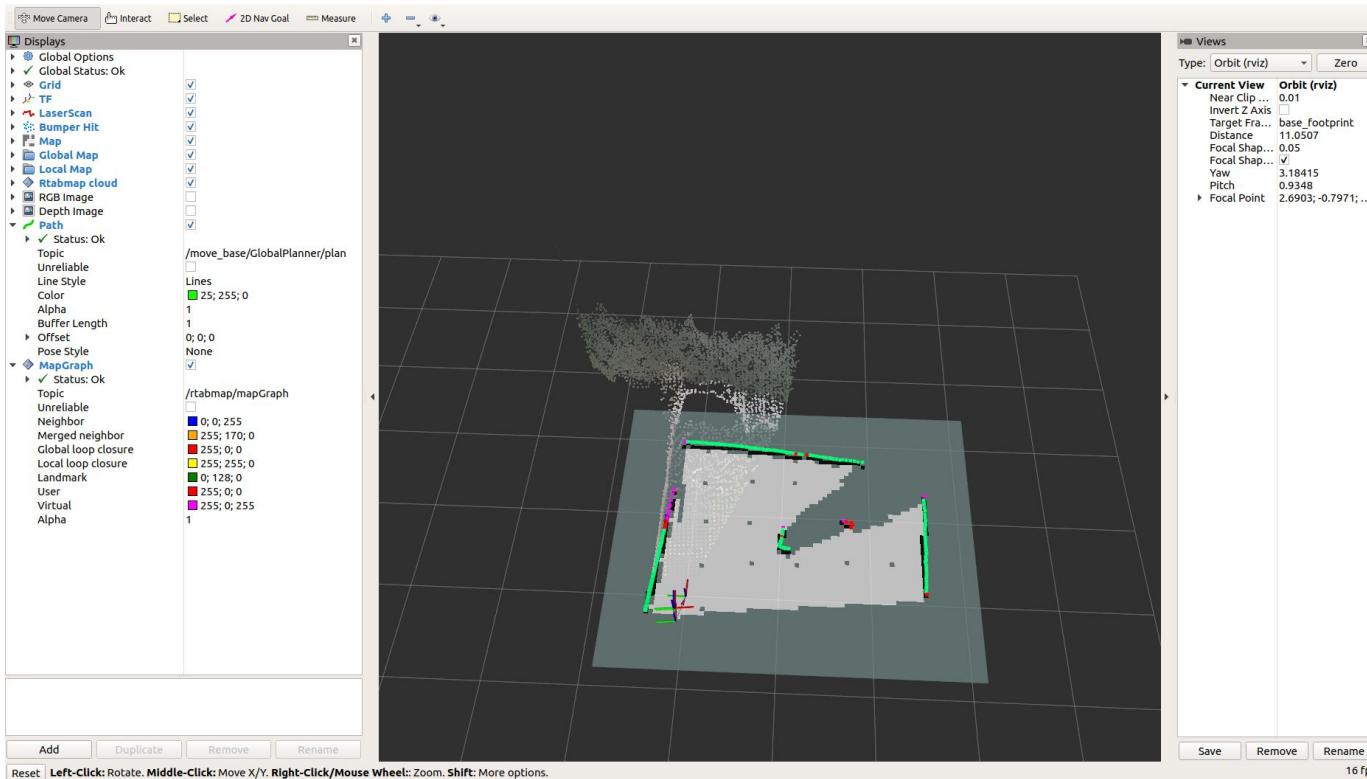
```
[home/agilex/agilex ws/src/limo_ros/limo_bringup/launch/rtabmap_rviz.launch http://localhost:11311]
process[rviz-1]: started with pid [25744]
QObject::connect: Cannot queue arguments of type ' QVector<int> '
(Make sure ' QVector<int>' is registered using qRegisterMetaType())
QObject::connect: Cannot queue arguments of type ' QVector<int> '
(Make sure ' QVector<int>' is registered using qRegisterMetaType())
```

05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping

- RTABMap 알고리즘

- 다음과 같은 Rviz가 실행되면 정상적으로 Mapping이 실행되고 있는 중입니다.



05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping

- RTABMap 알고리즘

- 지도 작성이 종료되면 터미널에서 Ctrl + C 를 입력하여 종료시킵니다.

생성된 지도는 자동적으로 .ros 폴더에 rtabmap.db 파일로 저장됩니다.

- RTABMap Navigation (자율주행)

- 실행된 모든 터미널을 닫고 아래의 명령어를 터미널에 각각 실행합니다.

(보라색 부분은 모드에 따라 선택해서 실행)

- \$ roslaunch limo_bringup limo_start.launch pub_odom_tf:=true
 - \$ roslaunch astra_camera dabai_u3.launch
 - \$ roslaunch limo_bringup limo_rtabmap_orbbec.launch localization:=true
 - \$ roslaunch limo_bringup limo_navigation_rtabmap.launch
 - \$ roslaunch limo_bringup limo_navigation_rtabmap_ackerman.launch
 - \$ roslaunch limo_bringup rtabmap_rviz.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with five tabs, each displaying a ROS command being run on a nano machine. The tabs are arranged in two rows: the top row has two tabs, and the bottom row has three tabs.

- Top Left Tab:** agilex@nano: ~ 41x10
agilex@nano:~\$ rosrun limo_bringup limo_start.launch pub_odom_tf:=true
- Top Right Tab:** agilex@nano: ~ 41x4
agilex@nano:~\$ rosrun astra_camera dabai_u3.launch
- Bottom Left Tab:** agilex@nano: ~ 41x10
agilex@nano:~\$ rosrun limo_bringup rtabmap_rviz.launch
- Bottom Middle Tab:** agilex@nano: ~ 41x10
agilex@nano:~\$ rosrun limo_bringup limo_rtabmap_orbbec.launch localization:=true
- Bottom Right Tab:** agilex@nano: ~ 41x10
agilex@nano:~\$ rosrun limo_bringup limo_navigation_rtabmap.launch

05 LIMO 소프트웨어 사용 방법

• 실행 결과

```
NoMachine - limo_177
[INFO] [1649990687.947123166]: Camera calibration file /home/agilex/.ros/camera_info/depth_Astra_Orbbec.yaml not found.

/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_start.launch http://localhost:11311
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success!
[YDLIDAR]:Fixed Size: 490
[YDLIDAR]:Sample Rate: 4K
[YDLIDAR INFO] Current Sampling Rate : 4K
[YDLIDAR INFO] Now YDLIDAR is scanning ..
.....
[INFO] [1649990731.034490984]: rtabmap (98): Rate=1.00s, Limit=0.000s, Conversion =0.0353s, RTAB-Map=0.5766s, Maps update=0.0054s pub=0.0001s (local map=42, WM=42)
[INFO] [1649990731.827876794]: rtabmap (99): Rate=1.00s, Limit=0.000s, Conversion =0.0048s, RTAB-Map=0.3508s, Maps update=0.0050s pub=0.0001s (local map=42, WM=42)

ROS_MASTER_URI=http://localhost:11311
process[rviz-1]: started with pid [21099]

[INFO] [1649990716.879807341]: Recovery behavior will clear layer 'obstacles'
[INFO] [1649990717.303098664]: Recovery behavior will clear layer 'obstacles'
[INFO] [1649990719.267912735]: odom received!

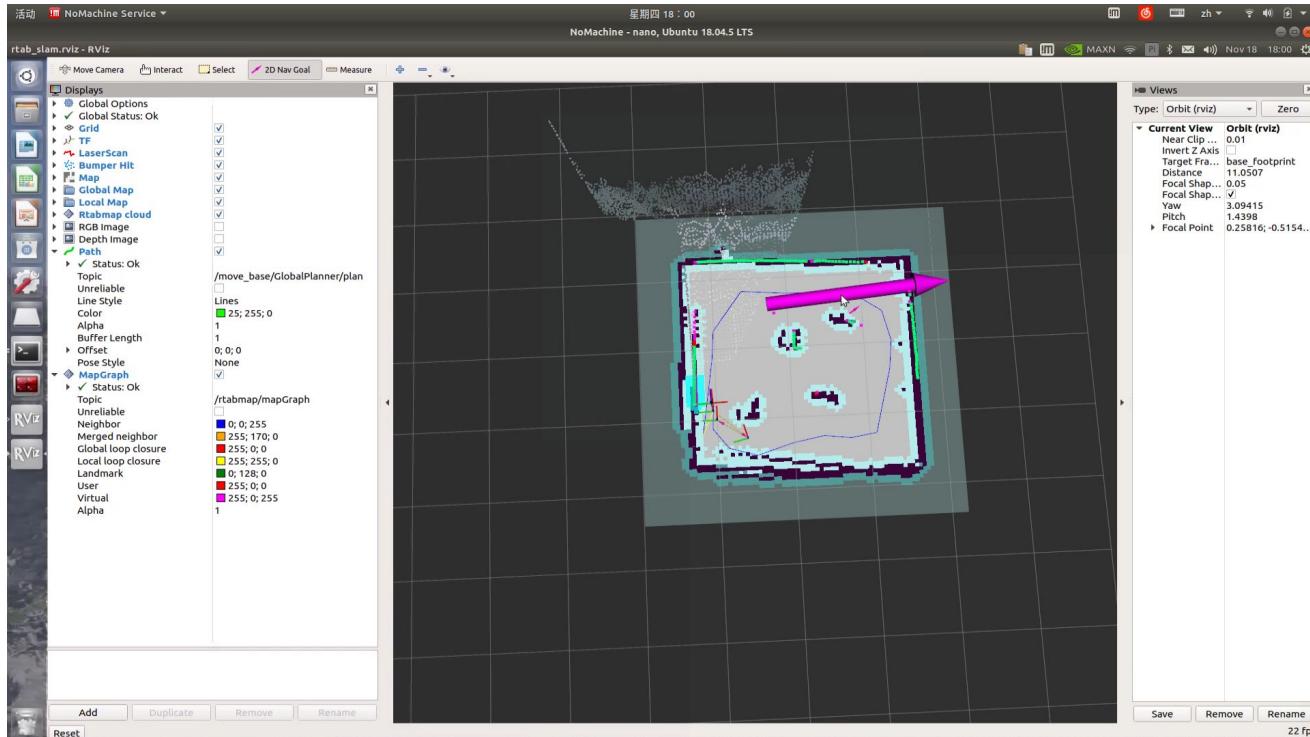
/home/agilex/agilex_ws/src/limo_ros/limo_bringup/launch/limo_navigation_rtabmap.launch
rot_stopped_vel is deprecated (and will not load properly). Use theta_stopped_vel instead.
[INFO] [1649990716.879807341]: Recovery behavior will clear layer 'obstacles'
[INFO] [1649990717.303098664]: Recovery behavior will clear layer 'obstacles'
[INFO] [1649990719.267912735]: odom received!
```

05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping

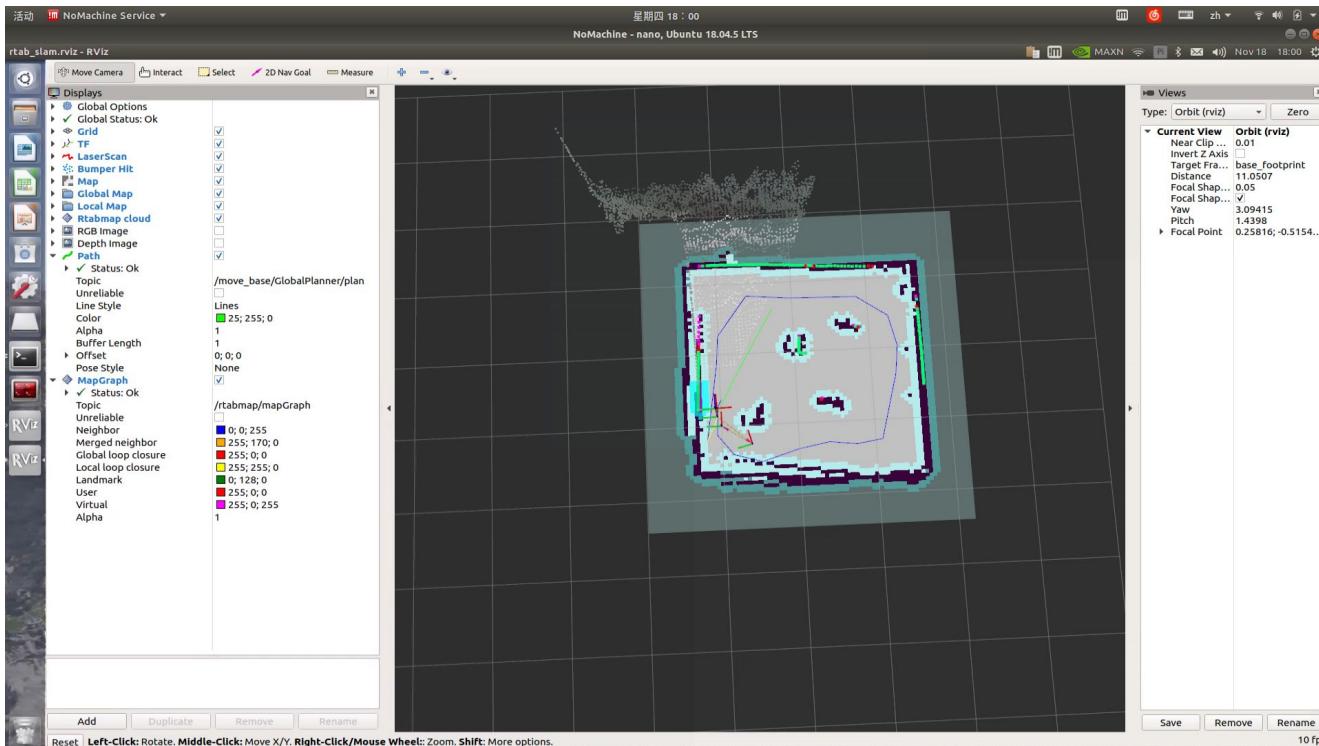
- RTABMap Navigation (자율주행)

- 아래와 같이 실행되면 정상적으로 실행된 것이며, Visual Positioning을 사용하기 때문에 초기 위치를 정해줄 필요가 없습니다. 바로 2D Nav Goal을 이용해서 목적 장소를 전달하면 됩니다.



05 LIMO 소프트웨어 사용 방법

- Depth Camera + LiDAR Mapping
 - RTABMap Navigation (자율주행)
 - 다음과 같이 녹색 경로가 생성되고, 조종기/APP을 자동모드로 설정하면 LIMO가 목표장소로 이동합니다.

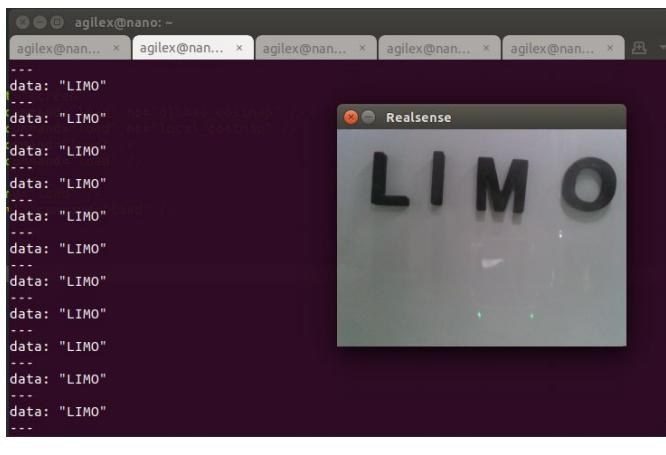


05 LIMO 소프트웨어 사용 방법

- Vision Module

- Recognize Text

- RGB 이미지를 입력으로 받은 후, GrayScale로 변환 후 이진화합니다.
 - pytesseract text recognition library를 이용하여 영어 또는 숫자를 인식합니다.
 - 최종 결과는 detect_word_reslut 토픽에 전달됩니다.
 - 실행을 위해서는 모든 터미널을 닫고 아래의 명령어를 각 터미널에 순서대로 입력합니다.
 - \$ roscore
 - \$ rosrun vision detect_node.py
 - \$ rostopic echo /detect_word_reslut



05 LIMO 소프트웨어 사용 방법

- 실행 환경

The screenshot shows a terminal window with three stacked command-line sessions:

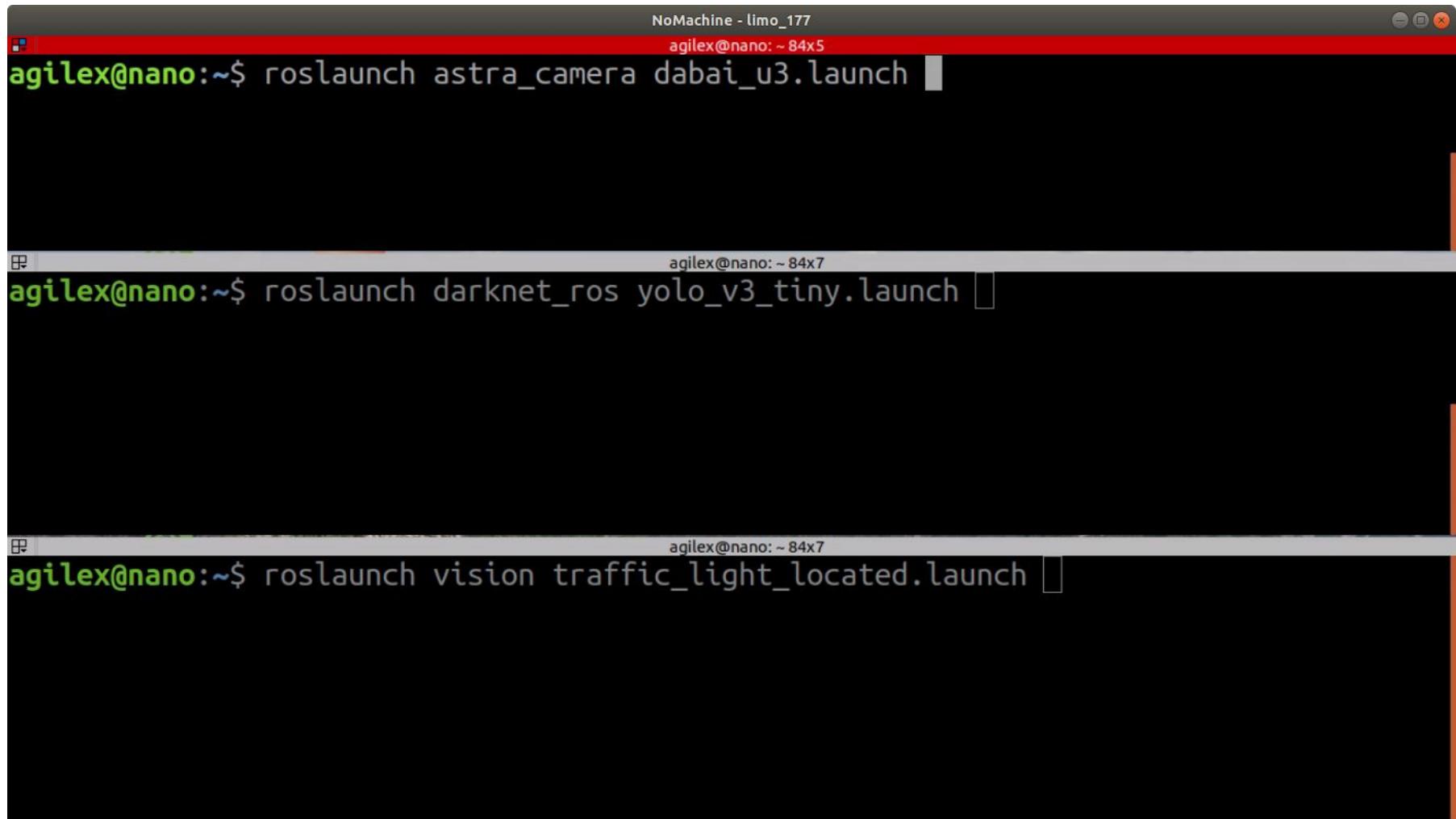
- Top Session:** NoMachine - limo_177
agilex@nano:~\$ roscore
- Middle Session:** agilex@nano: ~ 84x4
agilex@nano:~\$ rosrun vision detect_node.py
- Bottom Session:** agilex@nano: ~ 84x10
agilex@nano:~\$ rostopic echo /detect_word_reslut

05 LIMO 소프트웨어 사용 방법

- Vision Module
 - Identifying Traffic lights
 - Darknet_ros를 기반으로 신호등을 검출한 후, Traffic light의 위치를 확인
 - Depth camera의 정보를 이용하여 Traffic light의 Depth도 확인 가능
 - Traffic Light의 상태는 알 수 없음 (노란색, 파란색, 빨간색 등 알 수 없음)
 - 열려 있는 모든 터미널을 닫고, 아래의 명령어를 새로운 터미널에 순서대로 입력합니다.
 - \$ roslaunch astra_camera dabai_u3.launch
 - \$ roslaunch darknet_ros yolo_v3_tiny.launch
 - \$ roslaunch vision traffic_light_located.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경



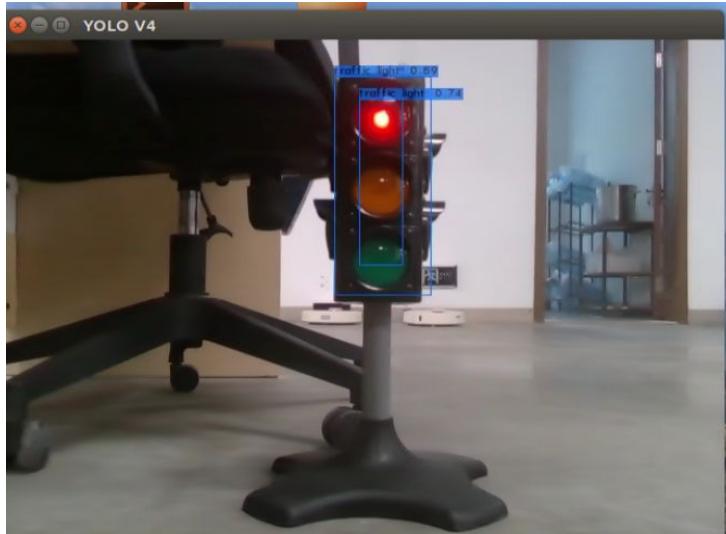
```
NoMachine - limo_177
agilex@nano: ~ 84x5
agilex@nano:~$ roslaunch astra_camera dabai_u3.launch

agilex@nano: ~ 84x7
agilex@nano:~$ roslaunch darknet_ros yolo_v3_tiny.launch

agilex@nano: ~ 84x7
agilex@nano:~$ roslaunch vision traffic_light_located.launch
```

05 LIMO 소프트웨어 사용 방법

- Vision Module
 - Identifying Traffic lights



```
/home/agilex/agilex_ws/src/src/vision/launch/traffic_light_located.launch http://localhost:  
/home/agilex/agilex_ws/... x /home/agilex/agilex_ws/... x /home/agilex/agilex_ws/... x  
red:2533 green:0 yellow:41  
red  
traffic light in camera:x:336 y:133  
red:1182 green:0 yellow:32  
red  
traffic light in camera:x:337 y:136  
red:2343 green:0 yellow:41  
red  
traffic light in camera:x:335 y:133  
red:1196 green:0 yellow:32  
red  
traffic light in camera:x:337 y:135  
red:2165 green:0 yellow:41  
red  
traffic light in camera:x:336 y:132  
red:1135 green:0 yellow:32  
red  
traffic light in camera:x:337 y:135  
red:2699 green:81 yellow:42  
red  
traffic light in camera:x:336 y:132  
red:1096 green:11 yellow:32  
red
```

A terminal window displaying the output of a ROS launch file for traffic light detection. The window title is "/home/agilex/agilex_ws/src/src/vision/launch/traffic_light_located.launch http://localhost:". The terminal lists numerous detections of traffic lights in the camera feed, providing coordinates and color values (red, green, yellow) for each detection.

05 LIMO 소프트웨어 사용 방법

- Vision Module

- Lifting barrier control

- LIMO Lifter와 같이 동봉된 수신기를 LIMO에 장착한 후 실행해야 합니다.
 - LIMO는 부착된 QR코드를 인식하여 거리가 0.3m 이하일 경우.

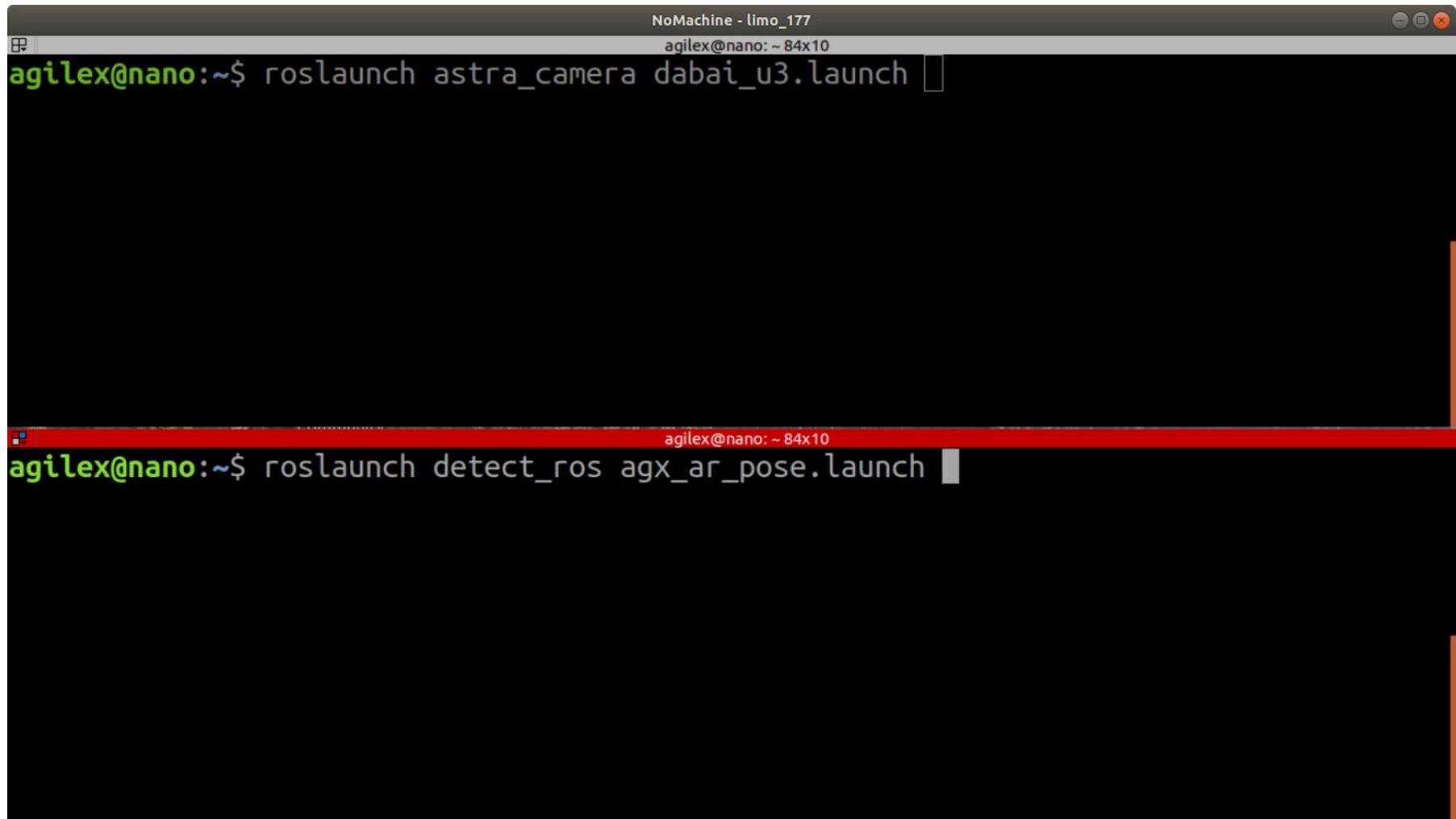
/chatter_updown 토픽에 barrier의 lift up 과 down을 전달하여 Barrier를 제어

- 열려 있는 모든 터미널을 닫고 새로운 터미널을 열어 아래의 명령어를 각각 입력
(파란색 부분은 해당하는 한줄만 입력)

- \$ roslaunch astra_camera dabai_u3.launch
 - \$ roslaunch realsense2_camera rs_camera.launch
 - \$ roslaunch detect_ros agx_ar_pose.launch
or \$ roslaunch detect_ros agx_ar_pose_dabai.launch

05 LIMO 소프트웨어 사용 방법

- 실행 환경



The screenshot shows a terminal window titled "NoMachine - limo_177" with a user session "agilex@nano: ~ 84x10". It displays two separate command-line sessions:

- The top session shows the command: `agilex@nano:~$ roslaunch astra_camera dabai_u3.launch`. This command launches a ROS node for an Astra camera using the dabai_u3 configuration.
- The bottom session shows the command: `agilex@nano:~$ roslaunch detect_ros agx_ar_pose.launch`. This command launches a ROS node for detecting AR markers using the agx_ar_pose configuration.

05 LIMO 소프트웨어 사용 방법

- Vision Module

- Lifting barrier control

- 아래의 내용이 출력되고, pub num : 1 이 터미널에 출력될 때, lifting barrier가 올라옵니다.
 - 그리고 LIMO는 3초간 Lifting barrier를 통과할 수 있습니다.

```
[ INFO] [1641976996.569838723]: get data!
[INFO] [1641976996.570155]: 2
[ INFO] [1641976996.694799865]: get data!
[INFO] [1641976996.695078]: 2
[ INFO] [1641976996.819796086]: get data!
[INFO] [1641976996.820138]: 2
[ INFO] [1641976996.944912622]: get data!
[INFO] [1641976996.945188]: 2
[ INFO] [1641976997.069627807]: get data!
[INFO] [1641976997.069886]: 2
[ INFO] [1641976997.194774435]: get data!
[INFO] [1641976997.195041]: 2
[ INFO] [1641976997.319595716]: get data!
[INFO] [1641976997.319861]: 2
[ INFO] [1641976997.444807196]: get data!
[INFO] [1641976997.445089]: 2
[ INFO] [1641976997.569841162]: pub num :1
[ INFO] [1641976997.569870049]: get data!
[INFO] [1641976997.570186]: 1
[ INFO] [1641977000.569984839]: pub num :2
[ INFO] [1641977000.570053222]: pub num :3
[ INFO] [1641977000.570086183]: 1 no date!
[ INFO] [1641977000.570221570]: 1 no date!
```

05 LIMO 소프트웨어 사용 방법

- Voice module (성능이 좋지 못함)
 - Speech converted to text
 - 목소리는 wav 파일로 나노의 외부 사운드 카드를 통해 녹음되며, voice library pocketsphinx를 통해 음성 인식이 진행됩니다.
 - 영어에서만 정상적으로 동작합니다.
 - 실행 중인 모든 터미널을 종료하고 새로운 터미널을 열어 아래의 명령어를 순서대로 입력합니다.
 - \$ rosrun voice demo_record_voice.py
 - 터미널에 recording이라는 문자가 표시되면 녹음이 시작되며 3초 후 Done이 표시됩니다.
 - 이 후 아래의 명령어를 실행하여 음성 인식을 진행합니다.
 - \$ rosrun voice demo_voice2word.py output.wav
 - 터미널에 인식된 음성이 출력됩니다.

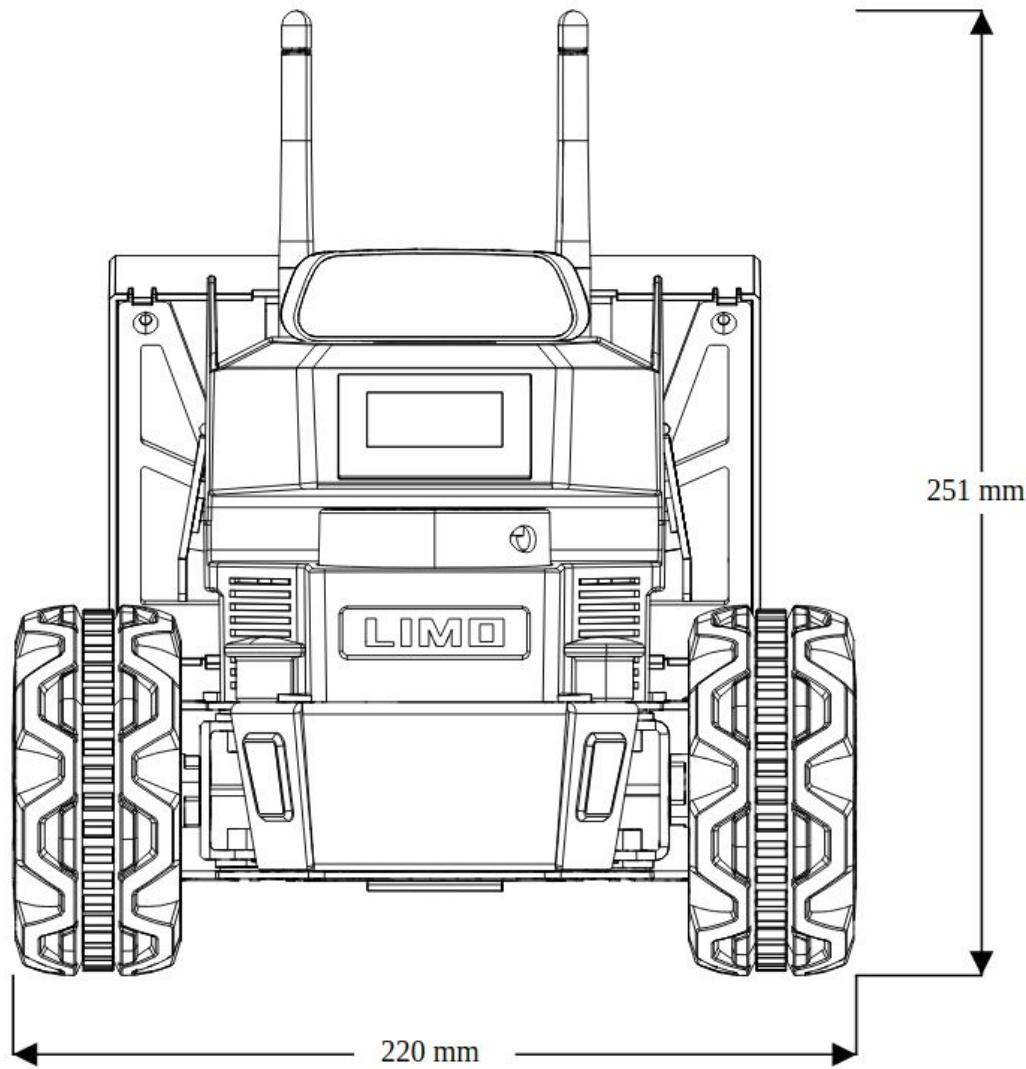
05 LIMO 소프트웨어 사용 방법

- **Voice module (성능이 좋지 못함)**
 - **Voice control**
 - Ahead. Back. Right. Left를 LIMO에게 말해서 LIMO를 제어합니다.
 - 실행 중인 터미널을 모두 닫고 새로운 터미널을 열어 아래의 명령어를 입력합니다.
 - \$ roslaunch limo_base limo_base.launch
 - \$ rosrun voice voice_ctrl_node.py
 - 아래의 명령어 실행 후, 1을 입력하여 recording 모드에 진입합니다.

Appendix

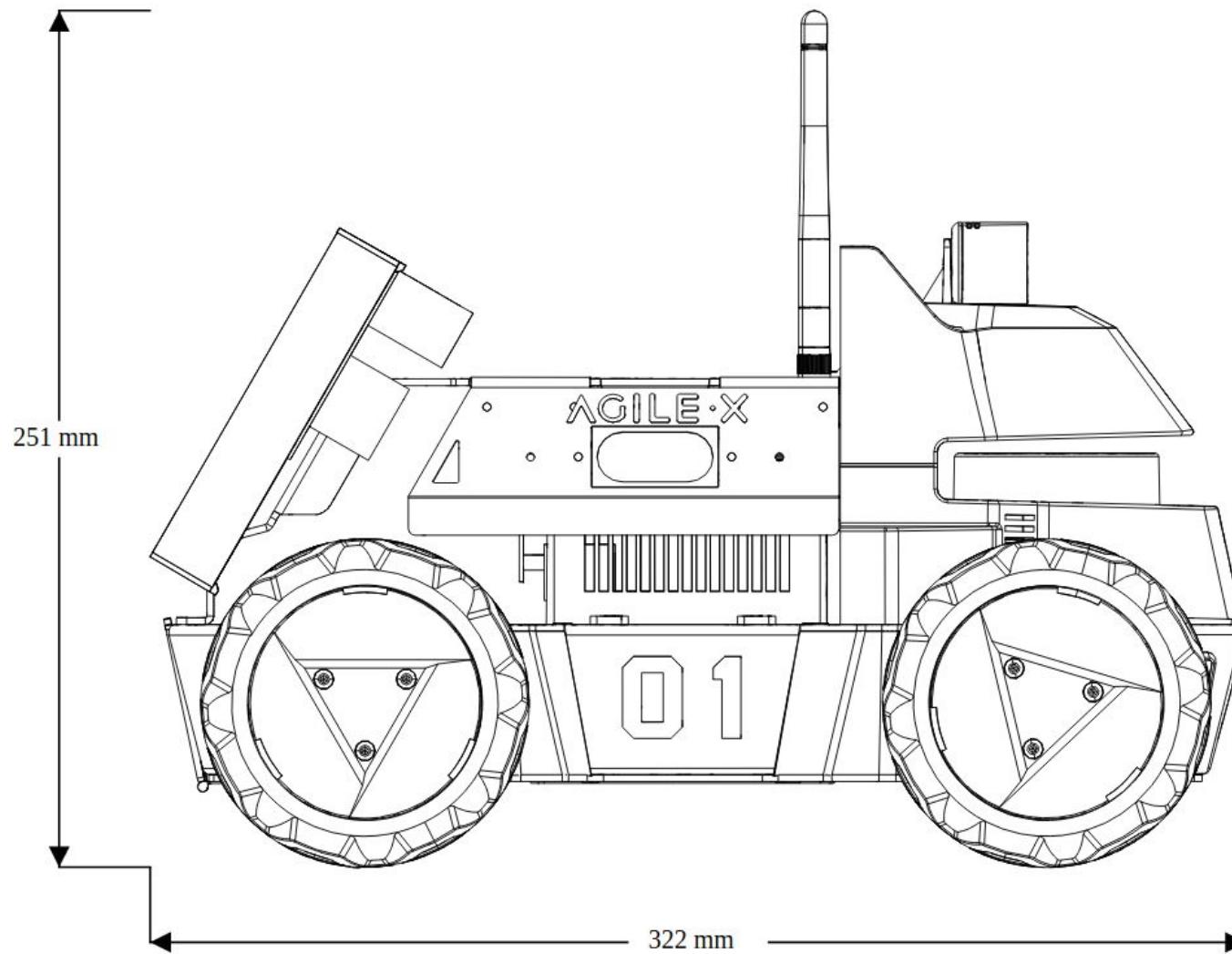
Appendix

- Appendix 1. 세 가지 시점



Appendix

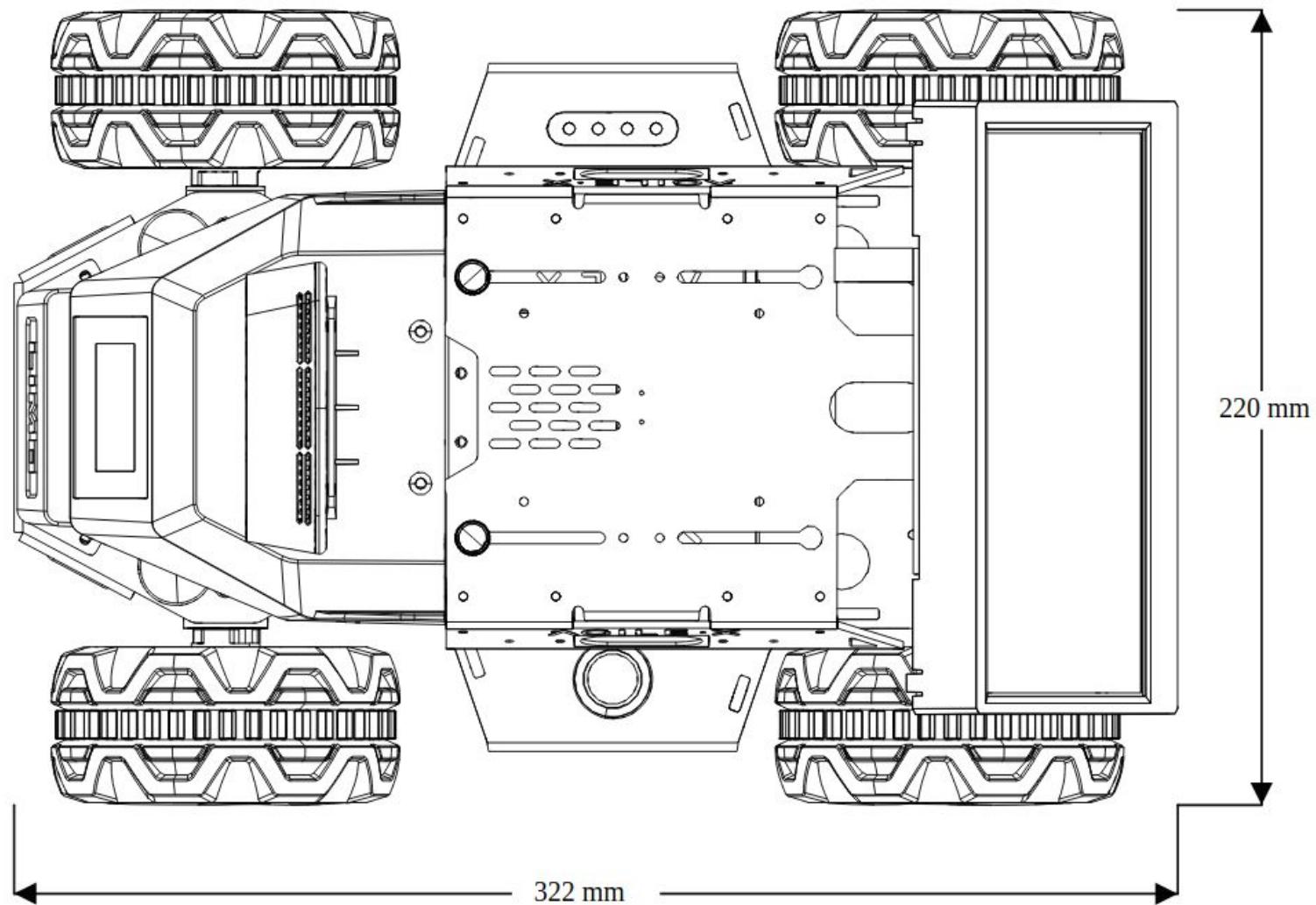
- Appendix 1. 세 가지 시점



WeGo

Appendix

- Appendix 1. 세 가지 시점



Appendix

- Appendix
 - 더욱 상세한 내용은 아래의 Github을 통해 확인할 수 있습니다.
 - [https://github.com/agilexrobotics/limo-doc/blob/master/Limo%20user%20manual\(EN\).md](https://github.com/agilexrobotics/limo-doc/blob/master/Limo%20user%20manual(EN).md)



WeGo Robotics

Tel. 031 – 229 – 3553



Fax. 031 – 229 – 3554



제품 문의: go.sales@wego-robotics.com

기술 문의: go.support@wego-robotics.com