



**ML-0**



**ML-2**



# Hardware User Guide

Solid-State LiDAR ML

Release v1.1.0

2022-02-25



# Table of Contents

---

<b>1. Mechanical Interface .....</b>	<b>03</b>
1.1. Included Components .....	04
1.2. Exterior Mechanical Dimensions – ML-0 Sensor .....	05
1.3. Exterior Mechanical Dimensions – ML-2 Sensor .....	06
<b>2. Electrical Interface .....</b>	<b>07</b>
2.1. Cable Connection .....	08
2.2. IP/Port Information .....	10
2.3. Power Adapter Information .....	11
<b>3. Drivers .....</b>	<b>12</b>
3.1. Ethernet .....	13
<b>4. Updating Firmware .....</b>	<b>14</b>

# Chapter 1

## Mechanical Interface



Smart Optical Sensors

## 1.1. Included Components

ML은 다음과 같은 아이템들을 포함하여 제공합니다.

- ML-0 or ML-2 sensor
- Sensor to Power cable/connector (4 meters)
- Sensor to Ethernet cable/connector (4 meters)
- Sensor AC/DC Power adapter/cable

**<Sensor to Power cable/connector>**



**<Sensor to Ethernet cable/connector>**



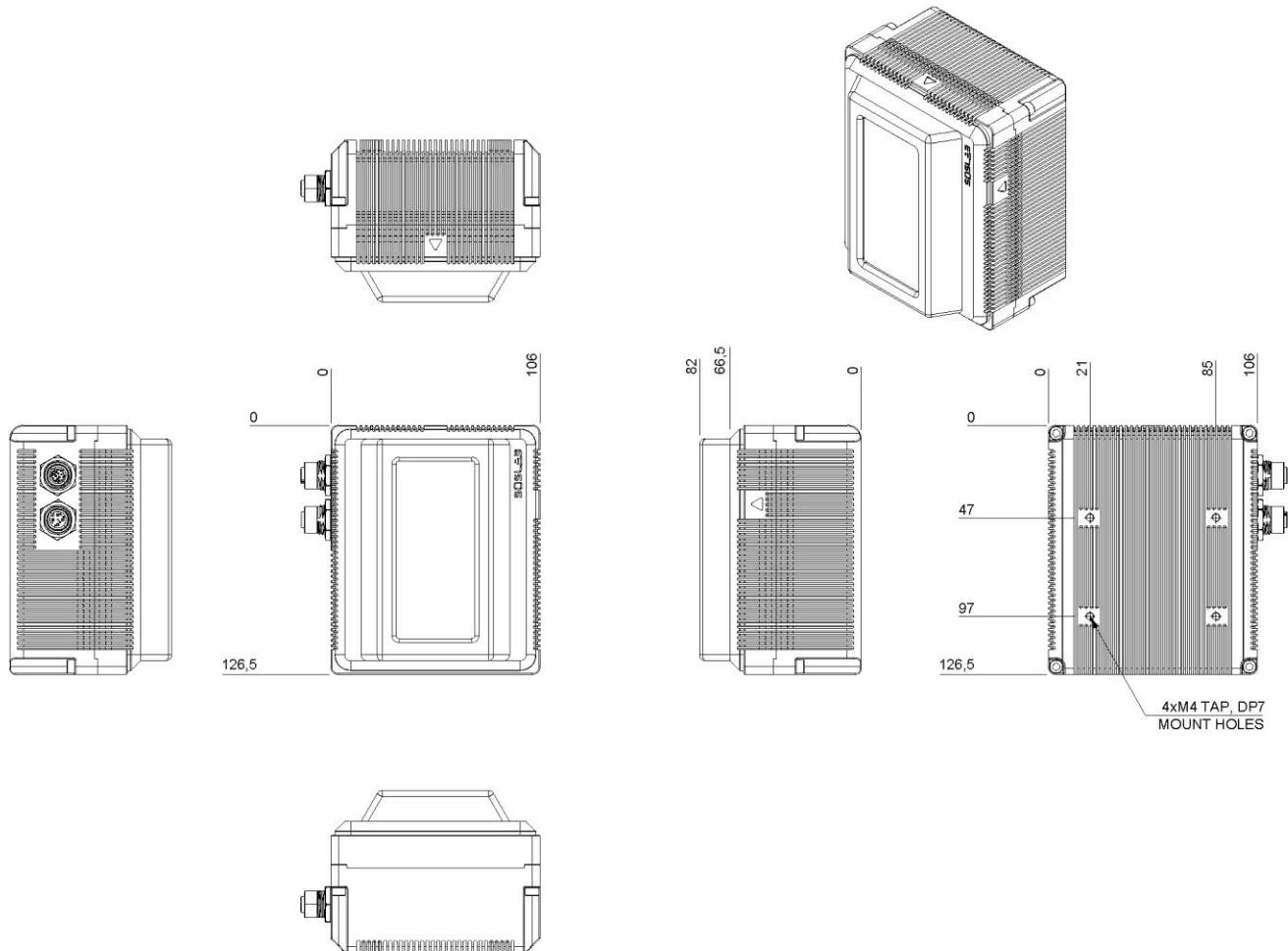
**<Sensor to AC/DC Power adapter>**



**<Sensor to AC/DC Power cable>**

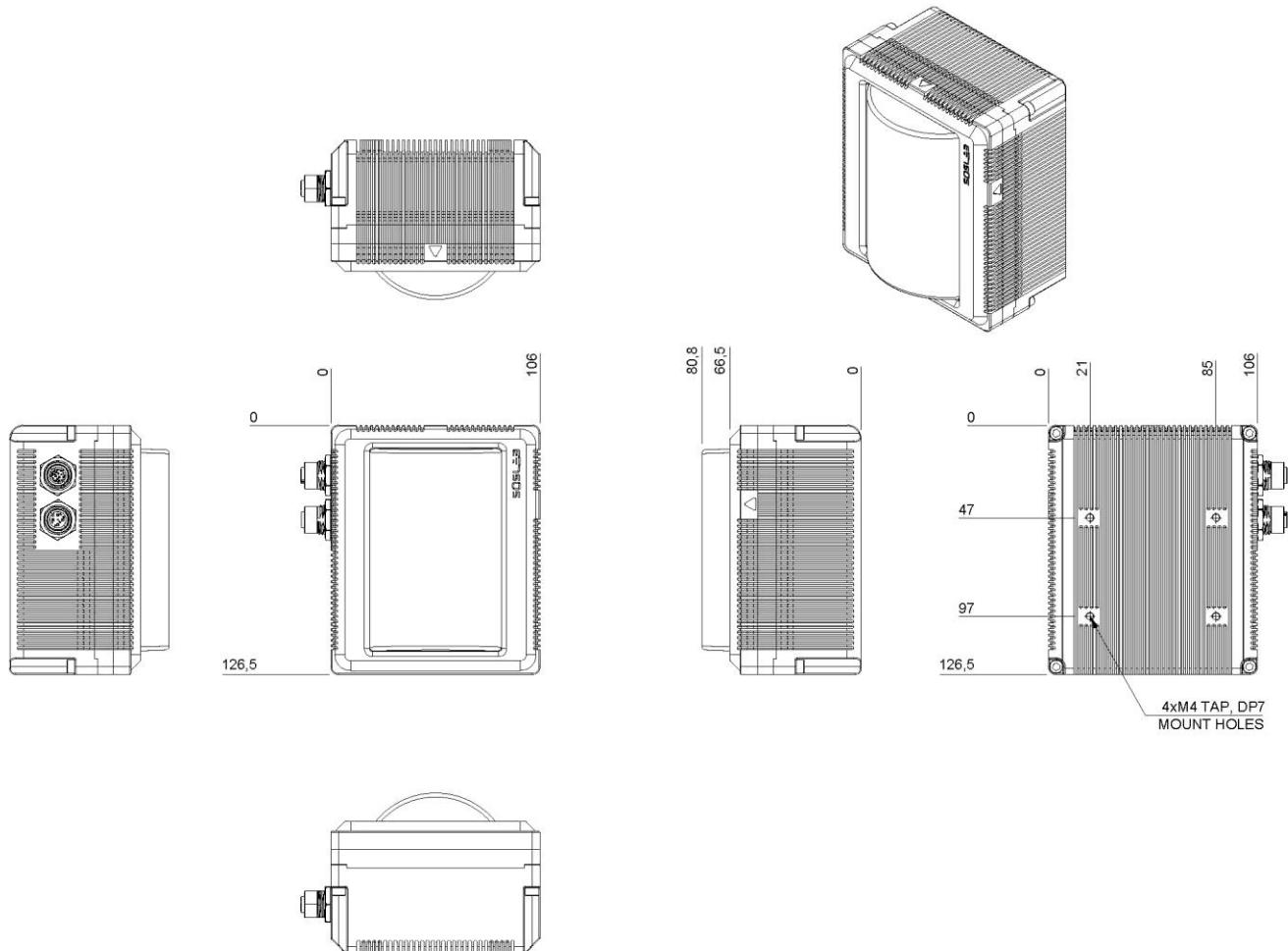


## 1.2. Exterior Mechanical Dimensions – ML-0 Sensor



The sensor has 4 x M4 mounting holes

## 1.3. Exterior Mechanical Dimensions – ML-2 Sensor



The sensor has 4 x M4 mounting holes

# **Chapter 2**

# **Electrical Interface**



## 2.1. Cable Connection

케이블을 연결하는 순서는 다음과 같습니다.

- 제공된 센서와 이더넷 케이블을 연결합니다.
- 제공된 센서와 파워 케이블을 연결합니다.
- 이더넷 케이블과 PC를 연결합니다.
- 파워 케이블과 파워 어댑터를 연결합니다.

<ML-0 케이블 연결 위치>

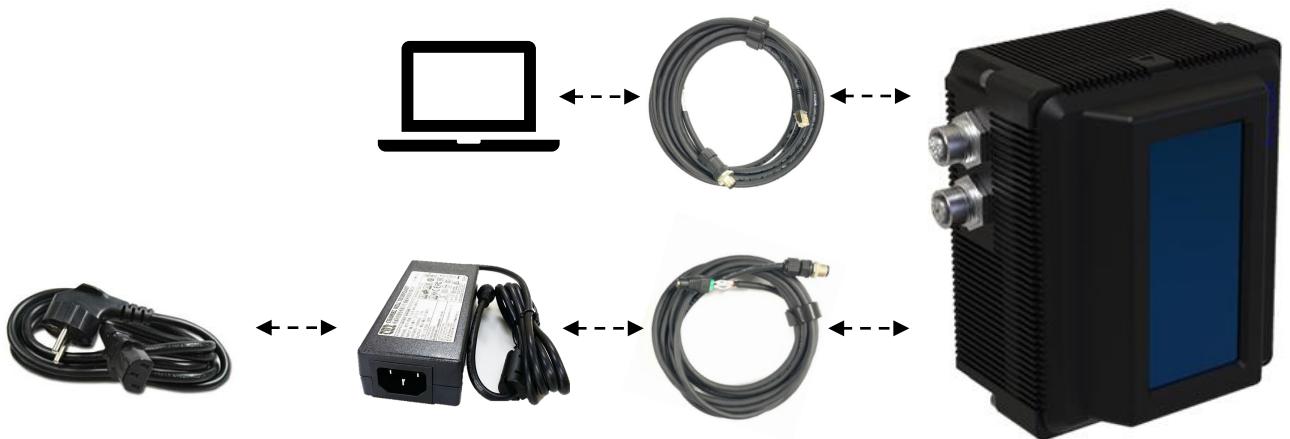


<ML-2 케이블 연결 위치>



## 2.1. Cable Connection

<ML-0 케이블 연결 순서>



<ML-2 케이블 연결 순서>



## 2.2. IP/Port Information

제공된 센서의 기본 IP/Port 정보는 다음과 같습니다.

- Local IP : 192.168.1.15
- Local Port : 2000
- Device IP : 192.168.1.10
- Device Port : 2000

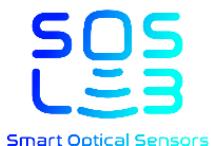
## 2.3. Power Adapter Information

제공된 파워 어댑터의 정보는 다음과 같습니다.

- 정격입력 : 100-240V~ 50/60Hz 1.7A
- 정격출력: +12.0V 5.0A 60.0W
- 위 사항과 동일한 규격을 가진 파워 어댑터를 사용하지 않아 발생한 문제에 대해서는 제품의 신뢰성을 보증하지 않습니다.

# **Chapter 3**

## **Drivers**



### 3.1. Ethernet

“2.1. Cable Connection”의 순서대로 케이블을 연결하면, ML LiDAR Manager를 통해 데이터를 확인하거나 Window/Ubuntu/ROS에서 ML의 API를 활용할 수 있습니다. ML LiDAR Manager와 ML의 API를 활용하는 것에 대한 자세한 내용은 **ML LiDAR SW User Guide**를 참고하시길 바랍니다.

---

**Note:** 기가비트 이더넷에 센서를 연결하지 않을 경우 센서와의 통신이 불가합니다. 반드시 기가비트 이더넷에 센서를 연결해주세요.

---

# Chapter 4

## Updating Firmware

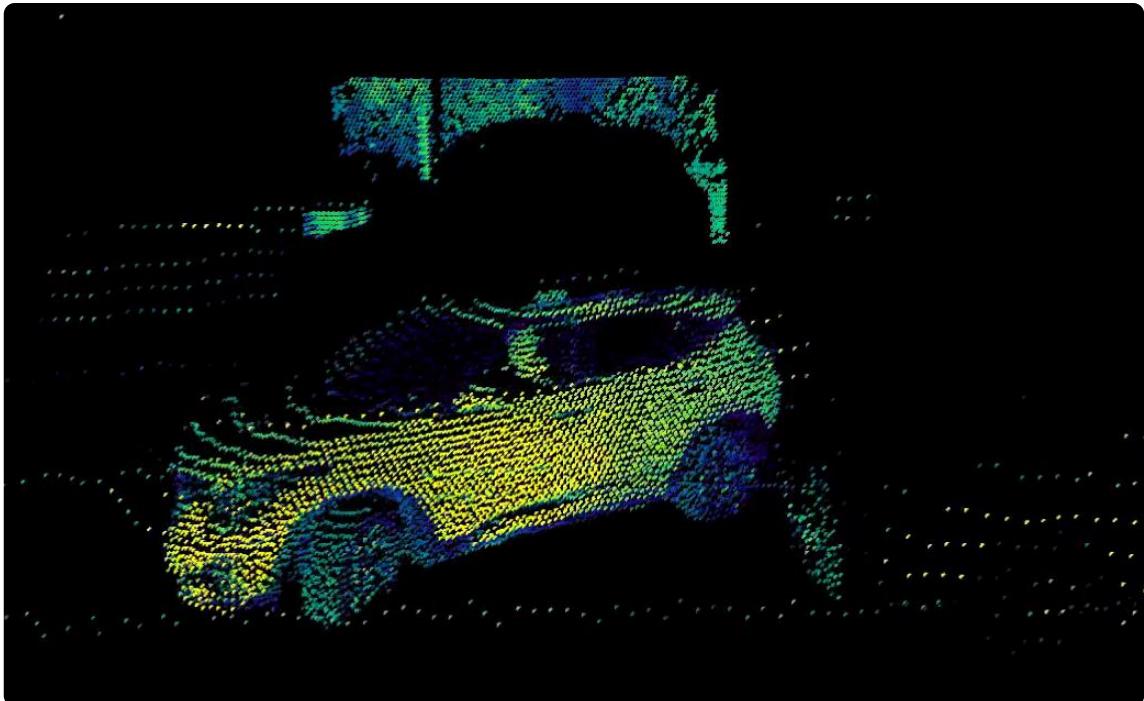


## 4. Updating Firmware



제공된 제품의 펌웨어를 항상 최신 버전으로 유지하는 것을 권장합니다.

펌웨어 업데이트에 대한 자세한 내용은 **ML LiDAR SW User Guide**를 참고하시길 바랍니다.



# Software User Guide

Solid-State LiDAR ML

Release v1.1.0

2022-02-25



# Table of Contents

---

<b>1. ML LiDAR Manager .....</b>	<b>20</b>
1.1. LiDAR Manager Setup .....	21
1.1.1. Installation .....	21
1.1.2. TCP/IP Setting .....	21
1.1.3. LiDAR Connection .....	23
1.2. LiDAR Manager User Interface .....	26
1.2.1. Point Cloud Viewer .....	26
1.2.2 Image Viewer .....	29
1.2.3 Subplot Viewer .....	31
1.3. LiDAR Device Setup .....	32
1.3.1. Read/Write Device Configuration .....	32
1.3.2. Firmware Update .....	37
1.4. LiDAR Data record / Play mode .....	38
1.4.1. LiDAR Data Recording .....	38
1.4.1. LiDAR Data Conversion .....	40
1.4.3. Play Mode .....	41

# Table of Contents

---

<b>2. ML API (Window) .....</b>	<b>42</b>
2.1. ML API Build for Windows .....	43
2.2. Example Code for Window .....	48
2.2.1. Connect ML Device .....	48
2.2.2. Setting Enable/Parameter of Data Processing Modules .....	49
2.2.3. Get Raw Data .....	50
2.2.4. Convert Raw Data(Depth, Ambient, Intensity) to Image .....	51
2.2.5. Save Raw Point Cloud .....	54
2.2.6. Data Recording example .....	55
2.2.7. Multi-LiDAR Example .....	56
<b>3. ML API (Ubuntu/ROS) .....</b>	<b>60</b>
3.1. ML API Build for Ubuntu/ROS .....	61
3.2. Example Code for Ubuntu/ROS .....	69
3.2.1. Connect ML Device .....	69
3.2.2. Connect ML Device using ROS .....	70
3.2.3. Setting Enable/Parameter of Data Processing Modules .....	71
3.2.4. Get Raw Data .....	72
3.2.5. Data Recording example .....	73
3.2.6. Multi-LiDAR Example .....	74
3.2.7. Publish Depth, Intensity, Ambient Image for Rviz .....	78
3.2.8. Publish Point Cloud for Rviz .....	81
3.2.9. Visualization with Rviz .....	82

# Table of Contents

---

<b>Appendix .....</b>	<b>84</b>
A.1. TCP Protocol Packet Structure .....	85
A.2. UDP Protocol Packet Structure .....	86
A.3. Device Information .....	88
A.3. LiDAR API – Typedefs .....	90
A.4. LiDAR API – Classes .....	91

# Chapter 1

## ML LiDAR Manager



# 1.1. LiDAR Manager Setup

본 문서는 ML LiDAR Manager 소프트웨어 사용에 관한 상세 설명을 제공합니다.

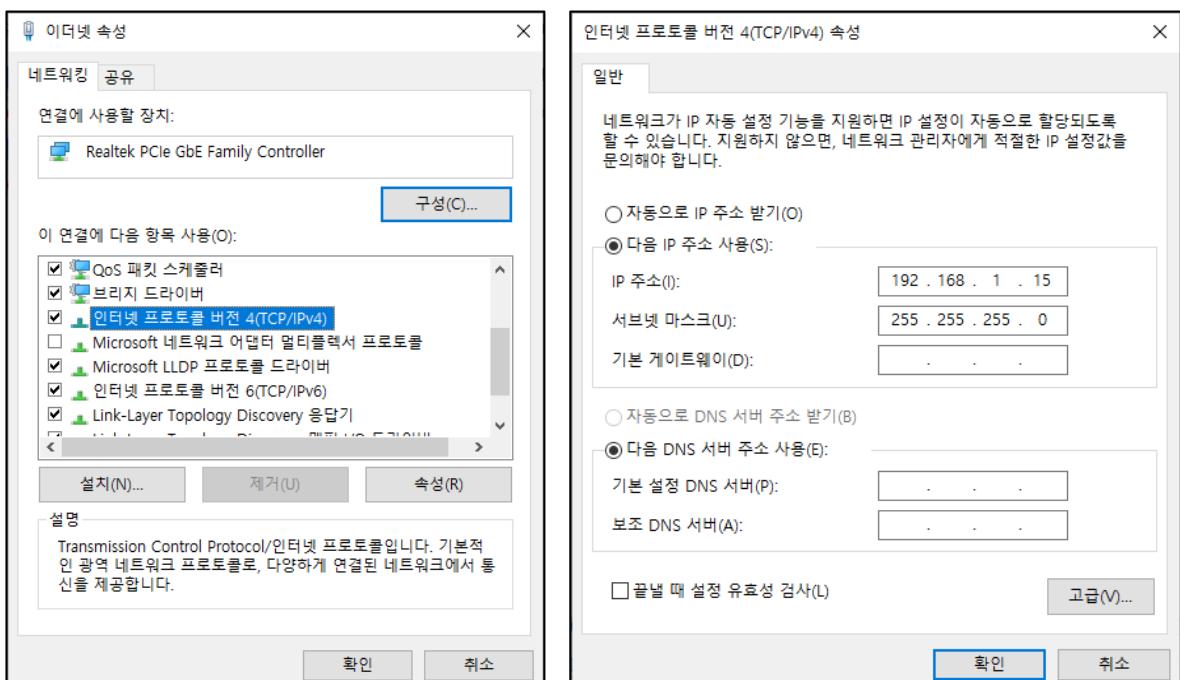
## 1.1.1. Installation

ML LiDAR Manager 소프트웨어는 제공된 ML\_Manager.zip 파일의 압축 해제 후, 폴더 내에 있는 lidar\_ml\_manager.exe 실행 파일을 이용해 별도의 설치 없이 실행 가능합니다.

## 1.1.2. TCP/IP Setting

ML LiDAR와 PC 간의 통신을 위해 TCP/IP 설정을 진행 합니다.

- 1) ML 전원 케이블을 연결하고, 네트워크 케이블로 PC와 LiDAR를 연결합니다.
- 2) 제어판 > 네트워크 및 인터넷 > 네트워크 및 공유 센터를 실행합니다.
- 3) 활성화 된 이더넷을 클릭 후 속성 창을 엽니다.
- 4) 인터넷 프로토콜 버전 4(TCP/IPv4) 선택 후 속성 버튼을 클릭합니다.
- 5) 속성 창에서 “다음 IP 주소 사용” 옵션을 클릭합니다.
- 6) **IP 주소(192.168.1.15)와 서브넷 마스크(255.255.255.0)**를 아래 그림과 같이 설정 후 확인 버튼을 클릭합니다.



이더넷 속성 창 / 인터넷 프로토콜 버전4(TCP/IPv4) 속성 창 설정

## 1.1. LiDAR Manager Setup

전원/네트워크 케이블 연결과 IP 설정이 완료되면 아래의 과정을 통해 PC와 ML LiDAR가 정상적으로 연결되었는지 확인이 가능합니다.

- 1) 윈도우 검색창에 ‘cmd’ 명령어를 입력하여 명령 프롬프트를 실행합니다.
- 2) 명령어에 ‘**ping 192.168.1.10 -t**’를 입력합니다.
- 3) PC와 ML이 정상적으로 연결되었다면 아래와 같은 메시지가 출력됩니다.



```
C:\선택 VS2015 x64 네이티브 도구 명령 프롬프트
C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>ping 192.168.1.10

Ping 192.168.1.10 32바이트 데이터 사용:
192.168.1.10의 응답: 바이트=32 시간<1ms TTL=255

192.168.1.10에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms

C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>
```

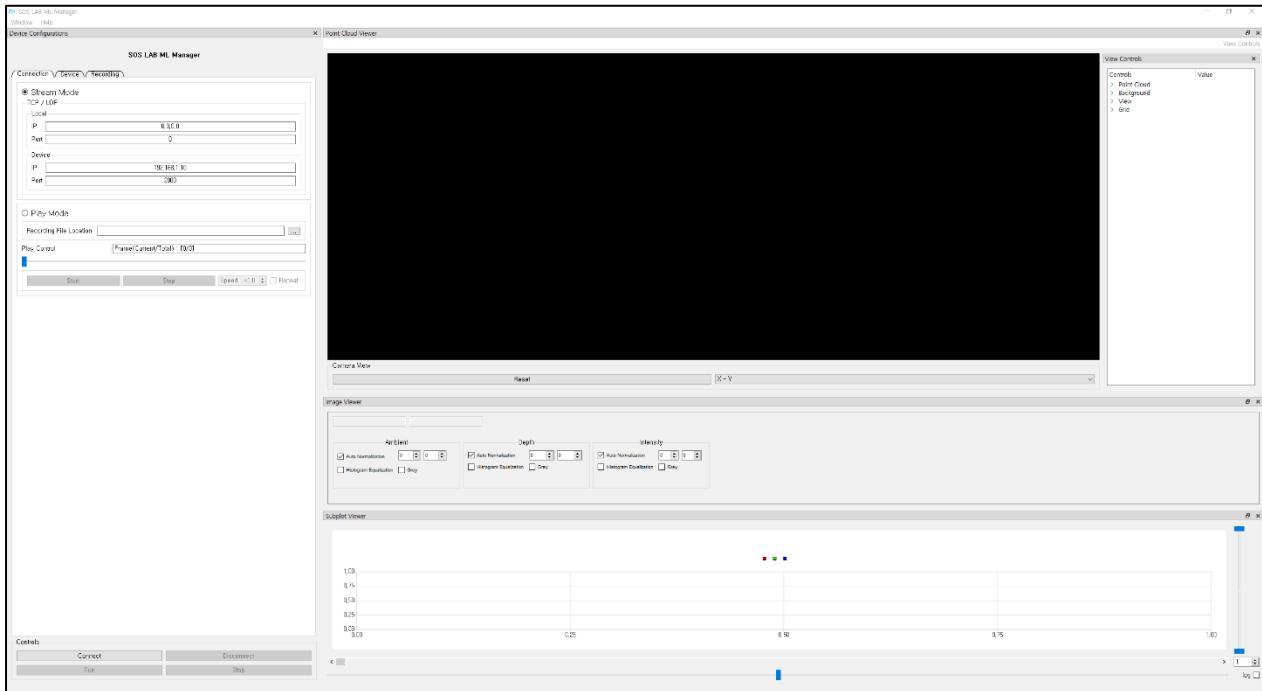
명령 프롬프트 창 및 ping 테스트 성공 결과 예시

테스트 결과 정상적으로 연결이 안되었다면 전원/네트워크 케이블과 이더넷 활성화 여부를 점검 후 다시 테스트를 진행합니다. 지속적으로 문제가 발생할 경우에는 고객 대응팀([sales@soslab.co](mailto:sales@soslab.co))으로 문의 주시기 바랍니다.

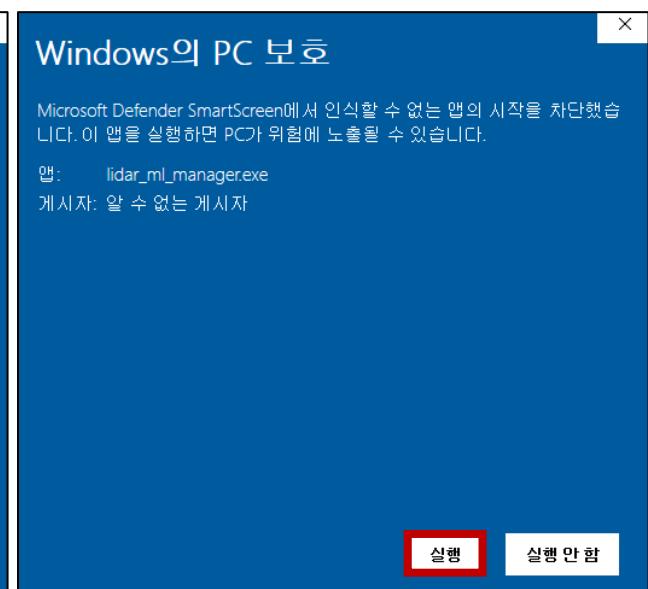
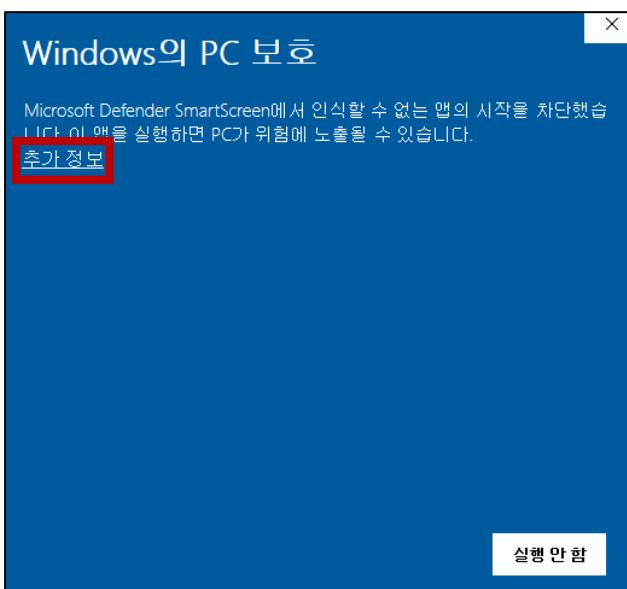
# 1.1. LiDAR Manager Setup

## 1.1.3. LiDAR Connection

`soslidar_ml_manager.exe`를 실행하면 아래의 그림과 같은 실행 화면이 나타납니다.



프로그램 최초 실행 시 아래와 같은 경고 알람이 발생할 수 있으며, 추가 정보 클릭 후 프로그램을 실행하면 됩니다.



# 1.1. LiDAR Manager Setup

Device Configuration에는 Connection(1.1), Device(1.3), Recording(1.4) 탭이 존재합니다.

Connection 탭에는 LiDAR 장치와 연결하는 Stream mode와 Recording 파일을 재생하는 Play mode(1.1.3)를 선택할 수 있습니다.

기본 값으로 제공되는 Stream mode를 위해서 PC(Local) 및 ML LiDAR(Device)의 IP/Port 입력 창이 있으며, 연결을 위해 아래와 같이 PC(Local)와 ML LiDAR(Device)의 IP/Port 정보를 입력합니다.

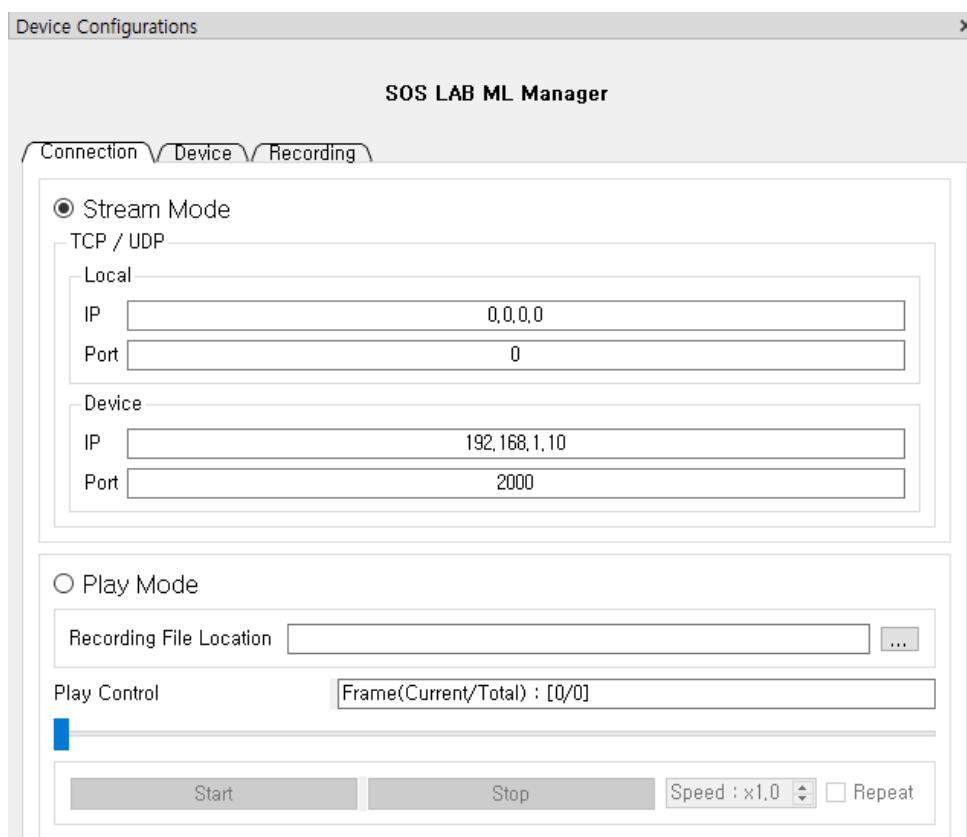
## [Local – PC]

- IP: 0.0.0.0
- Port: 0

## [Device – LiDAR]

- IP: 192.168.1.10
- Port: 2000

Play mode는 LiDAR Manager를 통해 녹화한 파일을 재생하는 기능을 제공합니다. Recording File Location 입력창의 [...] 을 클릭하여 확장자가 .bin으로 녹화된 파일을 선택합니다. Play mode에 대한 자세한 내용은 1.1.3항목에 있습니다.



Local 및 Device의 IP/Port 설정 예시

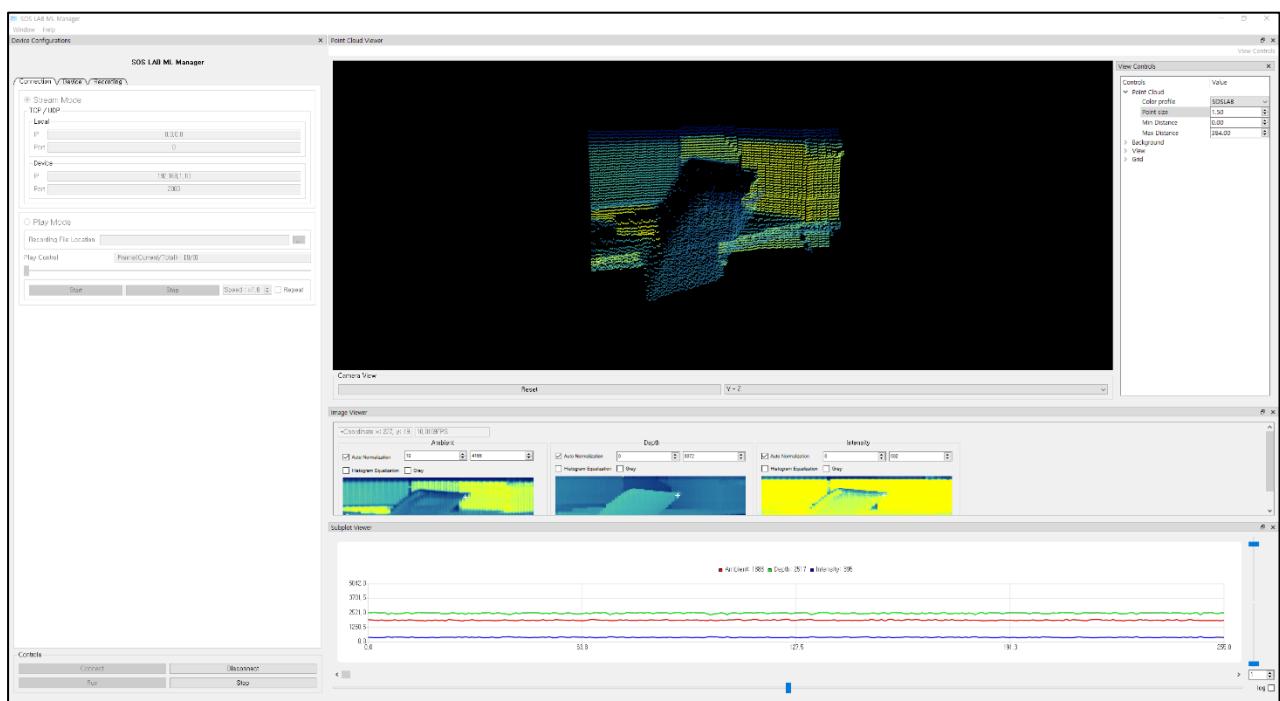
# 1.1. LiDAR Manager Setup

실행화면 왼쪽 하단 Controls 박스의 ‘Connect’ 버튼을 클릭하면 PC와 ML이 연결됩니다. 해당 장치와 최종 연결 시 아래와 같은 Windows 보안 경고 창이 나타납니다. 두 개의 check box를 아래 그림과 같은 상태로 변경한 뒤, ‘액세스 허용(A)’을 클릭합니다.



Windows 보안 경고 화면

PC와 ML LiDAR 연결 후, 실행화면 왼쪽 하단의 ‘Run’ 버튼을 클릭하면 아래 그림과 같이 Point-cloud viewer, Image Viewer, Subplot Viewer에 데이터가 출력됩니다.

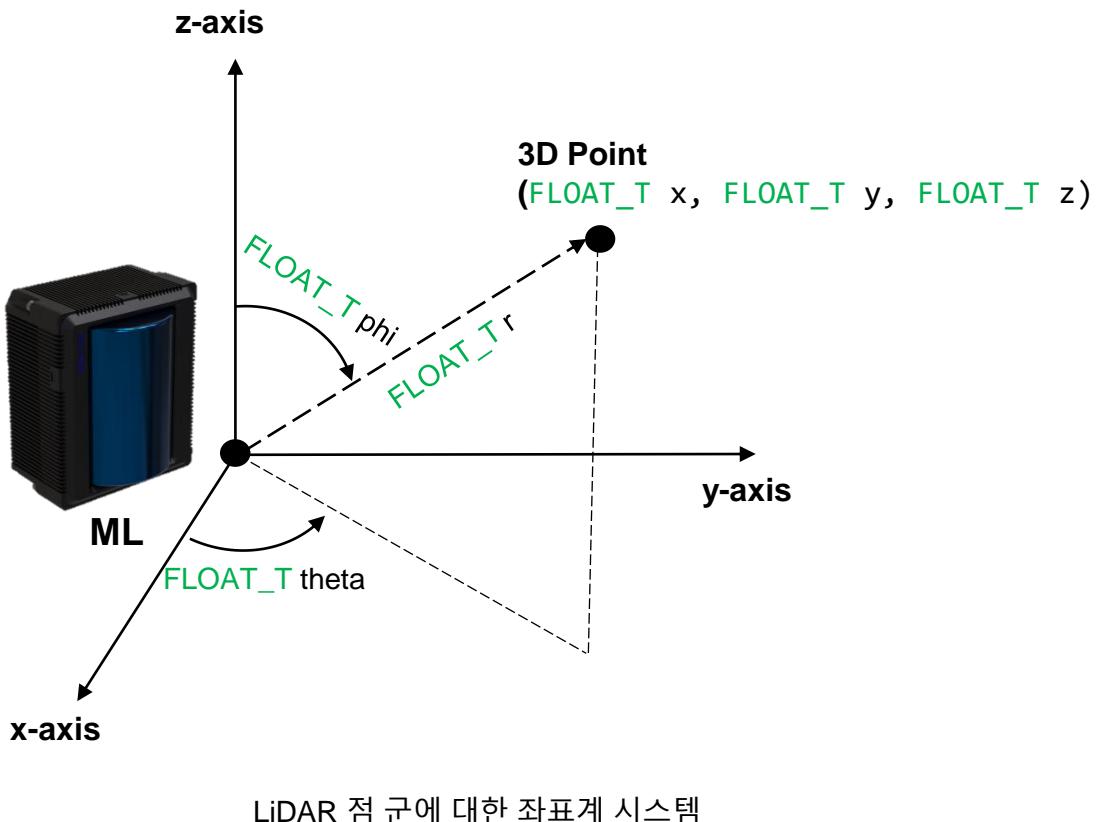


lidar\_ml\_manager 연결 완료 동작 화면

## 1.2. LiDAR Manager User Interface

### 1.2.1. Point Cloud Viewer

Point Cloud Viewer 창은 LiDAR 점 군(Point Cloud) 데이터를 가시화하기 위한 Main window 창, 가시화 값을 조절할 수 있는 View control 창으로 구성됩니다. 점 군 데이터는 아래와 같은 좌표계 시스템에서  $(x, y, z, i)$  값으로 표현합니다. 여기서  $(x, y, z)$ 는 한 점에 대한 각 축에서의 위치를 의미하고,  $(i)$ 는 해당 점에 대한 반사강도(intensity) 값을 의미합니다.



## 1.2. LiDAR Manager User Interface

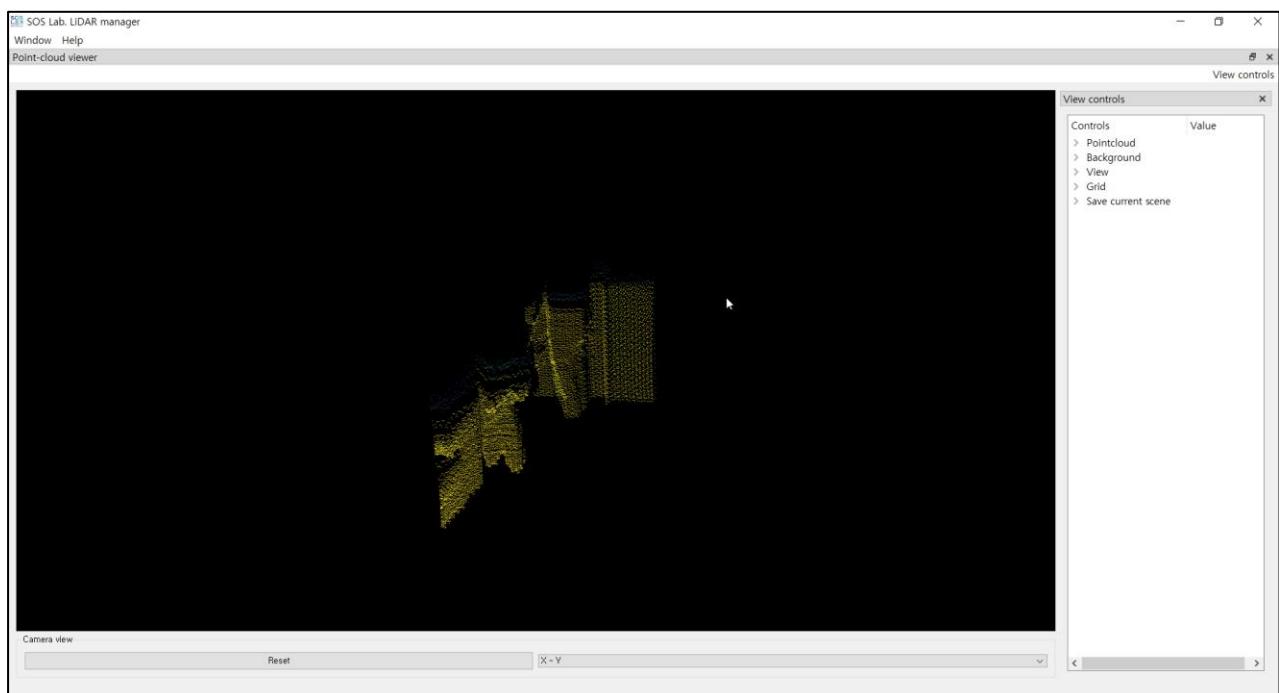
### A. Main window

Main window는 아래 그림과 같이 LiDAR에서 획득하고 있는 점 군 데이터를 3차원으로 가시화 합니다.

- <마우스 좌 클릭 + 이동>: 점 군에 대한 시점 회전
- <마우스 휠 버튼 드래그>: 점 군의 확대/축소
- <마우스 우클릭 + 이동>: 점 군에 대한 중심 이동

Main window 아래의 Camera view에는 점 군 데이터의 중심을 바라보도록 시점을 위치하는 Reset 버튼과 좌표계를 구성하는 3개의 평면에 수직인 시점을 이동할 수 있는 버튼이 있습니다.

- <Reset>: 점 군의 중앙에 카메라가 위치하도록 시점 이동
- <X-Y>, <Y-Z>, <X-Z>: 해당 평면과 카메라가 수직으로 위치하도록 시점 이동

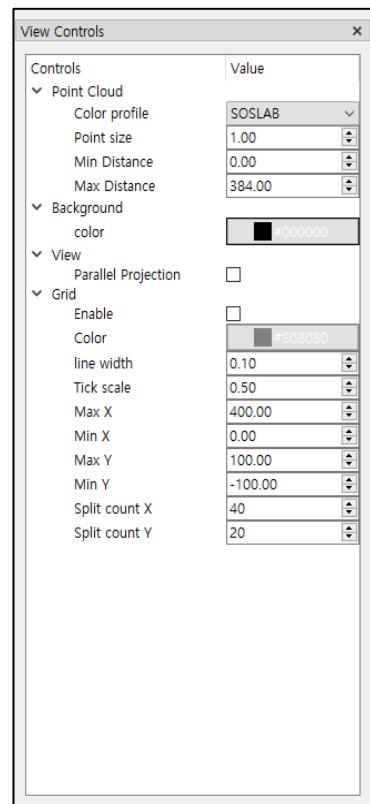


Point Cloud Main window 창

# 1.2. LiDAR Manager User Interface

## B. View controls

View control은 Main Window에서 가시화되는 점 군 및 배경에 대한 설정 값을 변경할 수 있는 창으로 아래와 같은 항목들로 구성되어 있습니다.



### Point Cloud

- Color profile: 점 군의 색상 변경
- Point size: 점 군의 크기 변경
- Min Distance: 최소 가시화 범위
- Max Distance: 최대 가시화 범위

### Background

- Color: Main Window의 배경 색상 변경

### View

- Parallel Projection: Main Window의 3차원 가시화 방법 변경  
(Default: Perspective Projection)

### Grid

- Enable: Grid 가시화 On/Off
- Color: Grid 색상 변경
- line width: Grid 선 두께 변경
- Tick scale: Grid 글자 크기 조절
- Max/Min X: Grid의 X축 최대/최소 거리 설정
- Max/Min Y: Grid의 Y축 최대/최소 거리 설정
- Split count X/Y: Grid 분할 개수

## 1.2. LiDAR Manager User Interface

### 1.2.2. Image Viewer

Image Viewer 창은 LiDAR에서 측정한 Ambient, Depth, Intensity 이미지를 가시화합니다.

데이터 종류	데이터 구조	설명
<b>Point Cloud</b>	$(x, y, z, i)$	라이다 점 군 데이터의 3차원 위치 정보와 반사 강도 정보를 3차원 좌표계 공간에 표현한 것
<b>Ambient Image</b>	$(u, v, a)$	라이다 신호 이외에 주변광(ambient light)에 의한 수신 신호의 세기를 라이다 좌표계에 대응하는 이미지 좌표계에 표현한 것
<b>Depth Image</b>	$(u, v, d)$	라이다 좌표계에서 각 점에 대해 측정된 거리(depth) 값을 라이다 좌표계에 대응하는 이미지 좌표계에 표현한 것
<b>Intensity Image</b>	$(u, v, i)$	라이다 좌표계의 각 점에 대해 측정된 레이저 신호의 반사강도(intensity) 값을 라이다 좌표계에 대응하는 이미지 좌표계에 표현한 것

## 1.2. LiDAR Manager User Interface

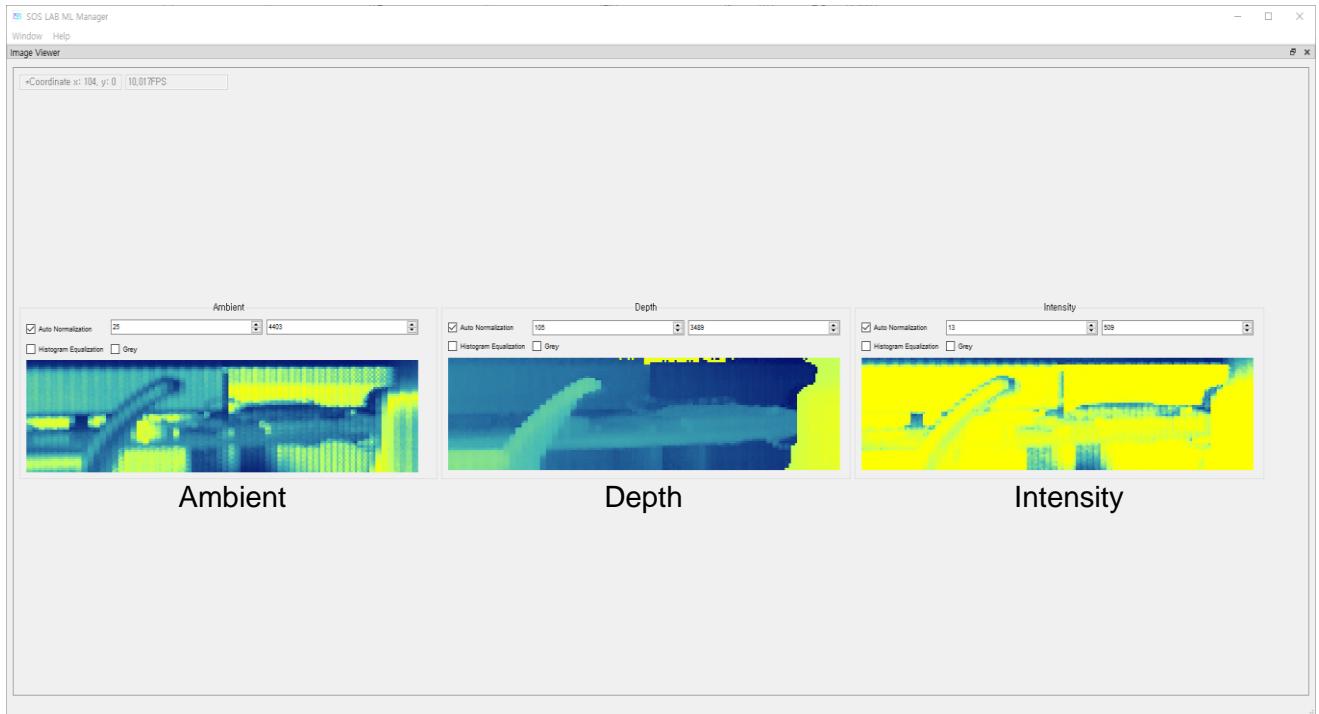


Image Viewer 창

Image Viewer의 상단 왼쪽은 현재 이미지에서 선택된 픽셀의 정보를 나타내며, 이미지 왼쪽 클릭을 통해 원하는 픽셀을 변경할 수 있으며, 오른쪽 클릭을 통해 픽셀 위치를 초기화 합니다. 해당 픽셀에 대한 정보는 아래의 Subplot Viewer에서 시간에 따른 Depth, Intensity, Ambient 정보로써 확인할 수 있습니다. 상단 오른쪽은 현재 ML Device의 속도(fps)를 나타냅니다.

이미지 가시화 결과는 좌측부터 ambient, depth, intensity 이미지를 나타내며 아래의 기능들을 통해 이미지를 조절할 수 있습니다.

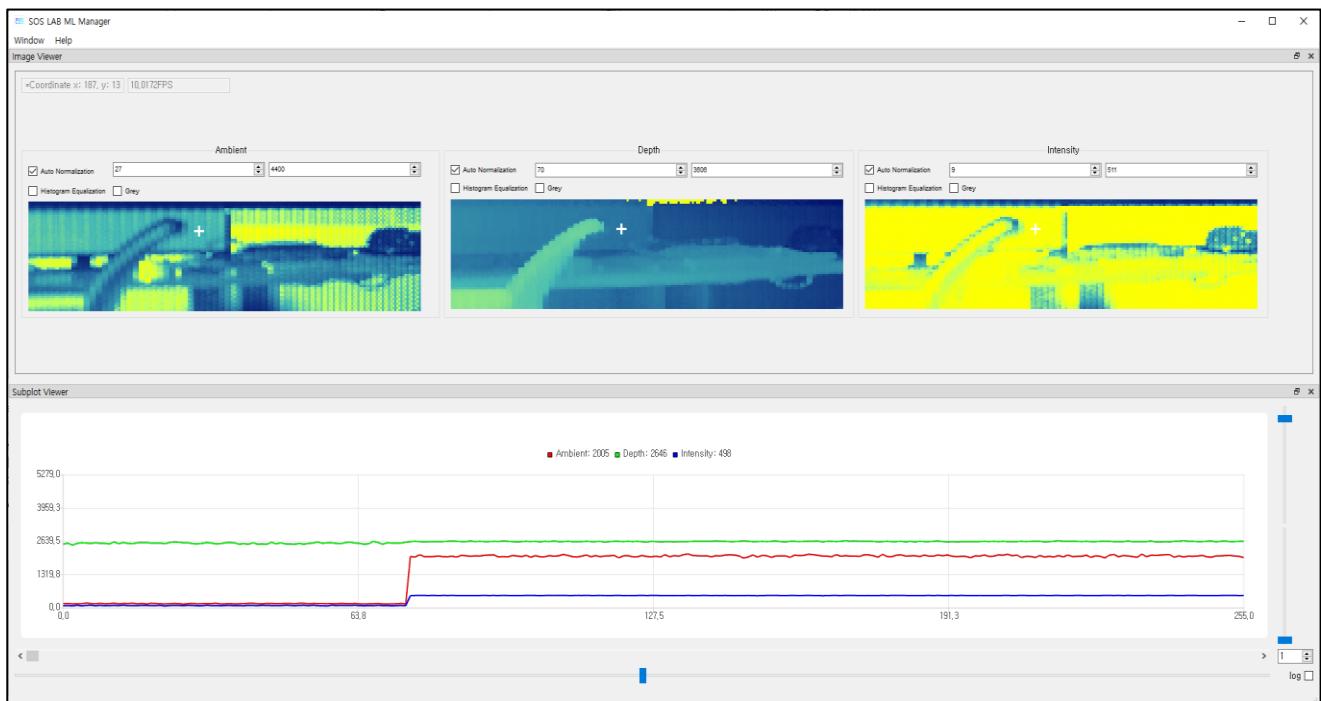
- <Ctrl+마우스 휠>: 영상 가시화 크기 조절
- Auto normalization
  - On : 이미지에서 최소, 최대값을 기준으로 이미지 정규화
  - Off : 사용자가 지정한 최소, 최대값을 기준으로 이미지 정규화
- Grey: 이미지를 흑백 이미지로 변경
- Histogram Equalization: 이미지의 명암을 극대화 하는 히스토그램 균일화

## 1.2. LiDAR Manager User Interface

### 1.2.3. Subplot Viewer

Subplot Viewer 창은 Image Viewer에서 선택된 픽셀에 대한 Depth/Intensity/Histogram 정보를 출력합니다.

상단의 **Ambient**, **Depth**, **Intensity**는 선택된 픽셀의 Ambient, Depth, Intensity 값을 나타내며, 아래는 선택된 픽셀의 시간에 따른 데이터 값을 가시화합니다.



Subplot Viewer 창

# 1.3. LiDAR Device Setup



## 1.3.1. Read/Write Device Configuration

Device 탭은 현재 상태의 ML의 IP/Port, Firmware 등의 장치 정보를 확인 및 변경할 수 있는 기능을 제공합니다.

---

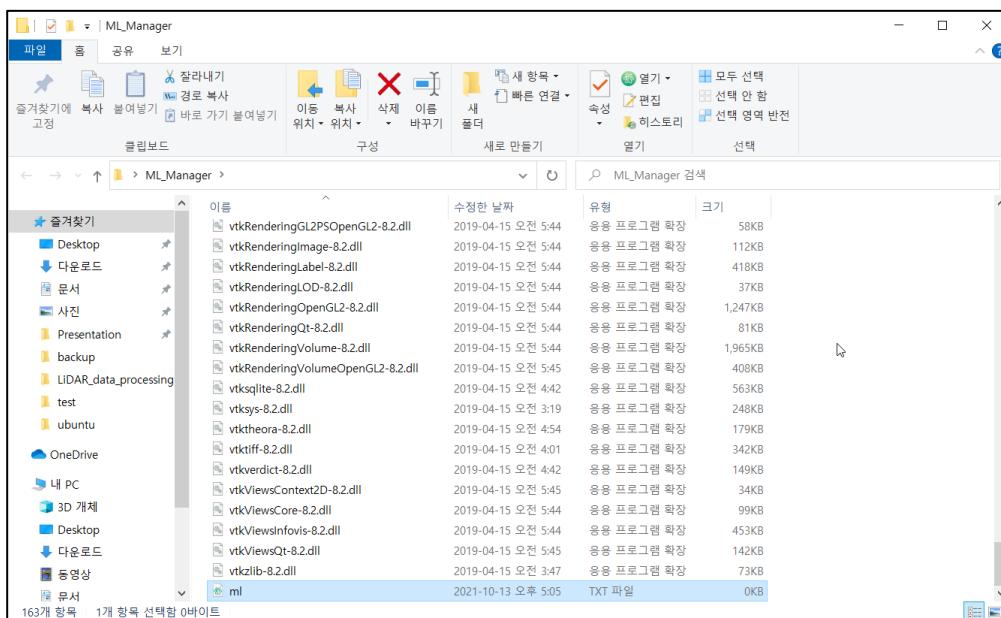
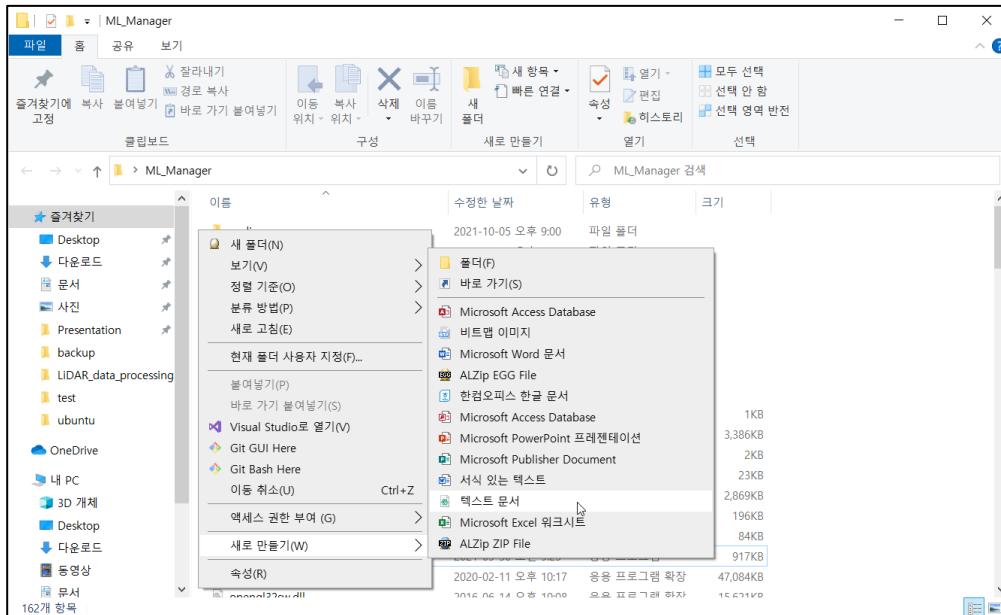
※ **Caution:** 장치 설정의 변경이 필요한 경우 고객 대응 팀([sales@soslab.co](mailto:sales@soslab.co))으로 문의 주시기 바랍니다. 사용자의 임의 장치 변경으로 인한 고장의 경우 무상 A/S 지원이 어려운 점 유의하시기 바랍니다.

---

# 1.3. LiDAR Device Setup

## A. ML Device 정보 읽어 오기

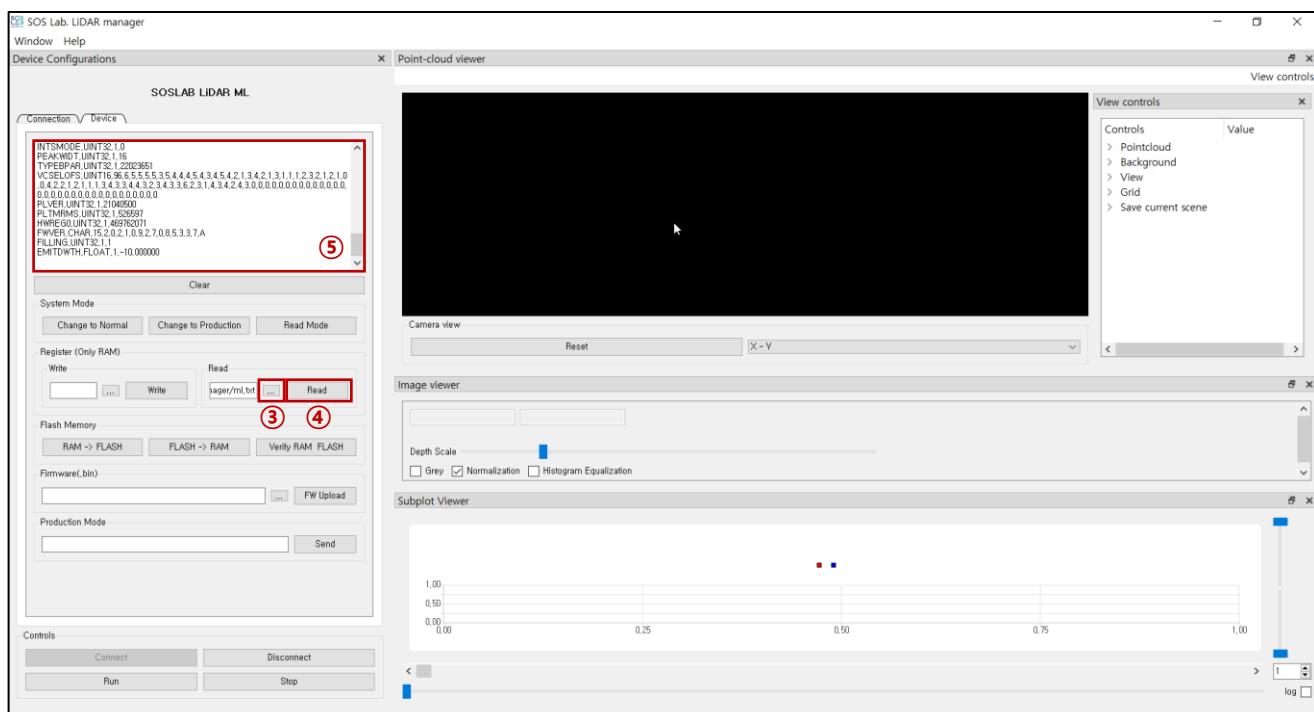
- 1) Stream mode를 선택한 뒤, 네트워크 케이블로 PC와 ML LiDAR를 연결하고 “Connect” 버튼을 클릭합니다.
- 2) lidar\_ml\_manager.exe가 있는 경로에 “ml.txt” 텍스트 파일을 생성합니다.



“ml.txt” 텍스트 파일 생성

# 1.3. LiDAR Device Setup

- 3) Device 창에서 Register (Only RAM)의 Read 구역의 “...” 버튼을 클릭하여, 생성한 “ml.txt”에 대해 파일을 선택합니다.
- 4) “ml.txt”에 대한 경로가 설정되었으면 “Read” 버튼을 클릭합니다. 파일 경로에 한글이 포함된 경우 오류가 발생할 수 있습니다.
- 5) 상단 출력창에 ML Device 정보들이 출력되면 성공적으로 장치 정보를 읽어와 “ml.txt” 파일에 저장한 것입니다. 이 외에 다른 값이 출력되면 오류가 발생한 것입니다.
- 6) Read된 ML Device 정보는 이름/데이터 타입/값으로 작성되어 있으며, 장치 설정의 세부 사항은 Appendix – A.3. Device Information에서 확인할 수 있습니다.

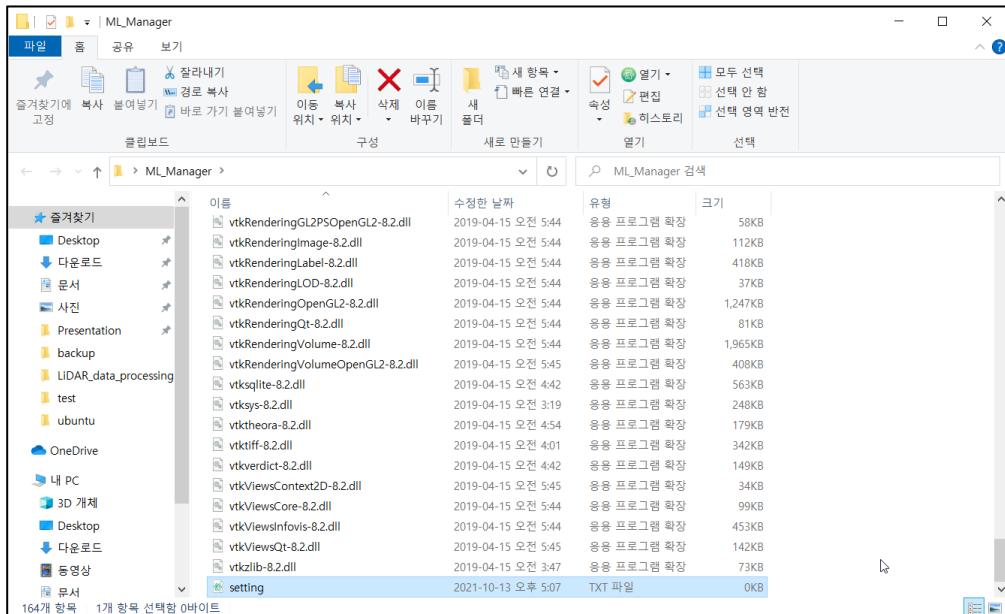


ML Device 정보 확인 – 파일 출력

# 1.3. LiDAR Device Setup

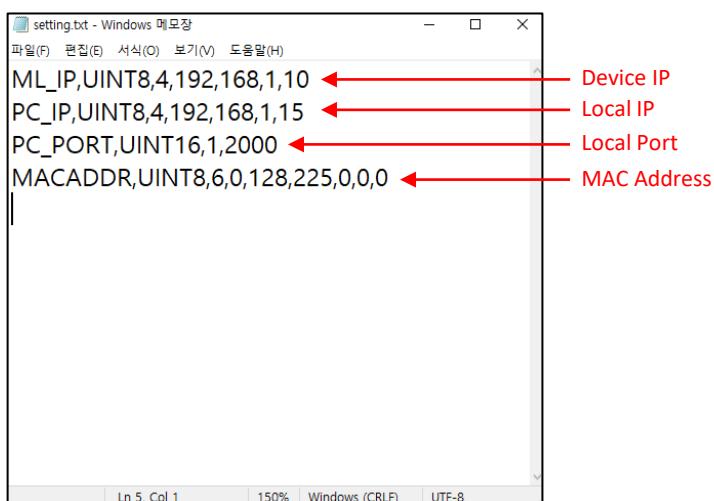
## B. ML Device 정보 쓰기

- 1) lidar\_ml\_manager.exe가 있는 경로에 "setting.txt" 텍스트 파일을 생성합니다.



"setting.txt" 텍스트 파일 생성

- 2) 생성된 "setting.txt" 텍스트 파일을 열어 수정할 항목들을 아래와 같이 작성합니다.

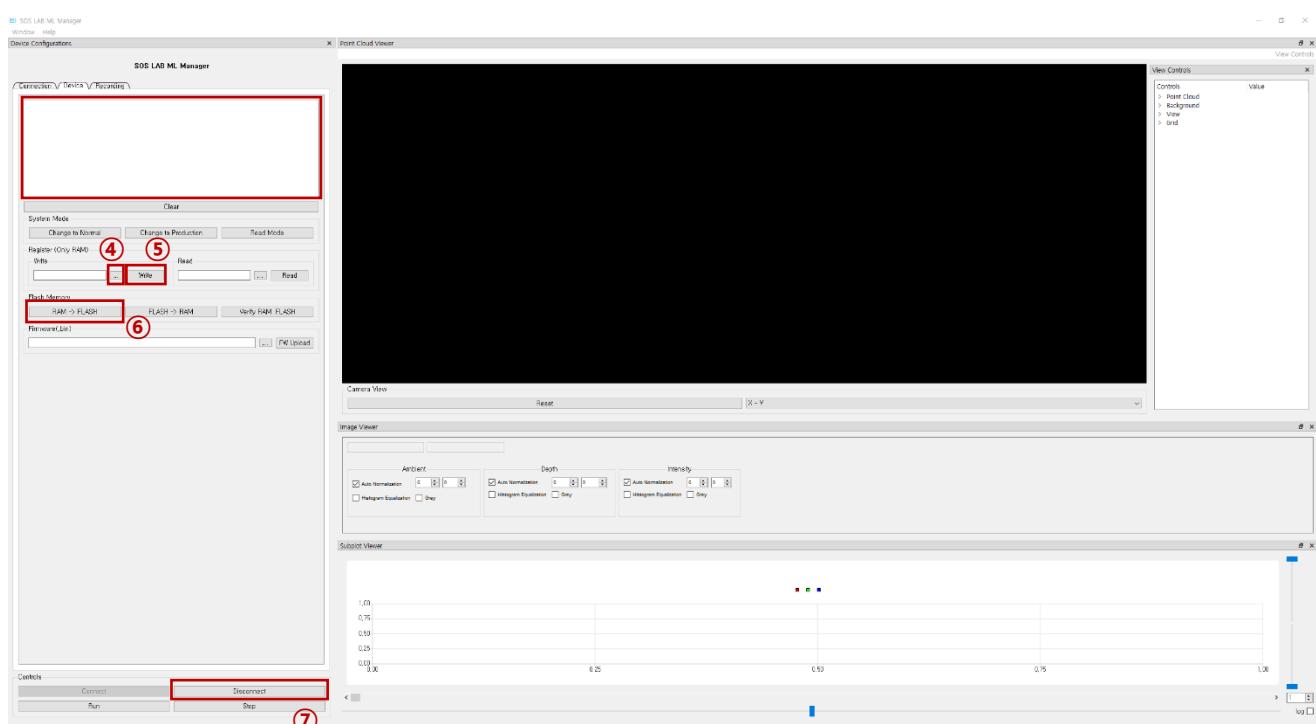


"setting.txt" 텍스트 파일 작성 예시

- ※ 수정할 항목을 변경할 때, 이름/쉼표 등의 문자가 변경되지 않도록 해야 합니다.  
※ 작성 후 마지막에 Enter를 클릭하여 마지막 줄을 공백으로 만들어야 합니다.

# 1.3. LiDAR Device Setup

- 3) Stream mode를 선택한 뒤, 네트워크 케이블로 PC와 ML LiDAR를 연결하고 “Connect” 버튼을 클릭합니다.
- 4) Device 창에서 Register (Only RAM)의 Write 구역의 “...” 버튼을 클릭하여, 생성한 “setting.txt”에 대해 파일을 선택합니다. 파일 경로에 한글이 포함된 경우 오류가 발생할 수 있습니다.
- 5) “setting.txt”에 대한 경로가 설정되었으면 “Write” 버튼을 클릭합니다. 성공적으로 동작할 경우 출력 창에 “0::RO::”가 출력됩니다.
- 6) Flash Memory 아래의 “RAM -> FLASH” 버튼을 클릭합니다. 성공적으로 동작할 경우 출력 창에 “0::MO”가 출력되고 변경된 세팅 값이 ML에 저장됩니다. 이 외에 다른 값이 출력되면 오류가 발생한 것입니다.
- 7) 왼쪽 하단의 “Disconnect” 버튼을 클릭하여, ML과 연결을 끊고 전원을 꺼냅니다.
- 8) IP/Port 등 같이 변경된 경우 변경된 정보를 Device의 IP와 Port에 입력한 뒤, 다시 “Connect” 버튼을 클릭하여 ML과 재연결을 합니다.
- 9) “Run” 버튼을 클릭하면 Point-cloud viewer, Image Viewer, Subplot Viewer에서 데이터가 정상 출력되는지 확인합니다.



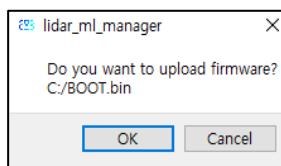
ML Device 정보 변경

# 1.3. LiDAR Device Setup

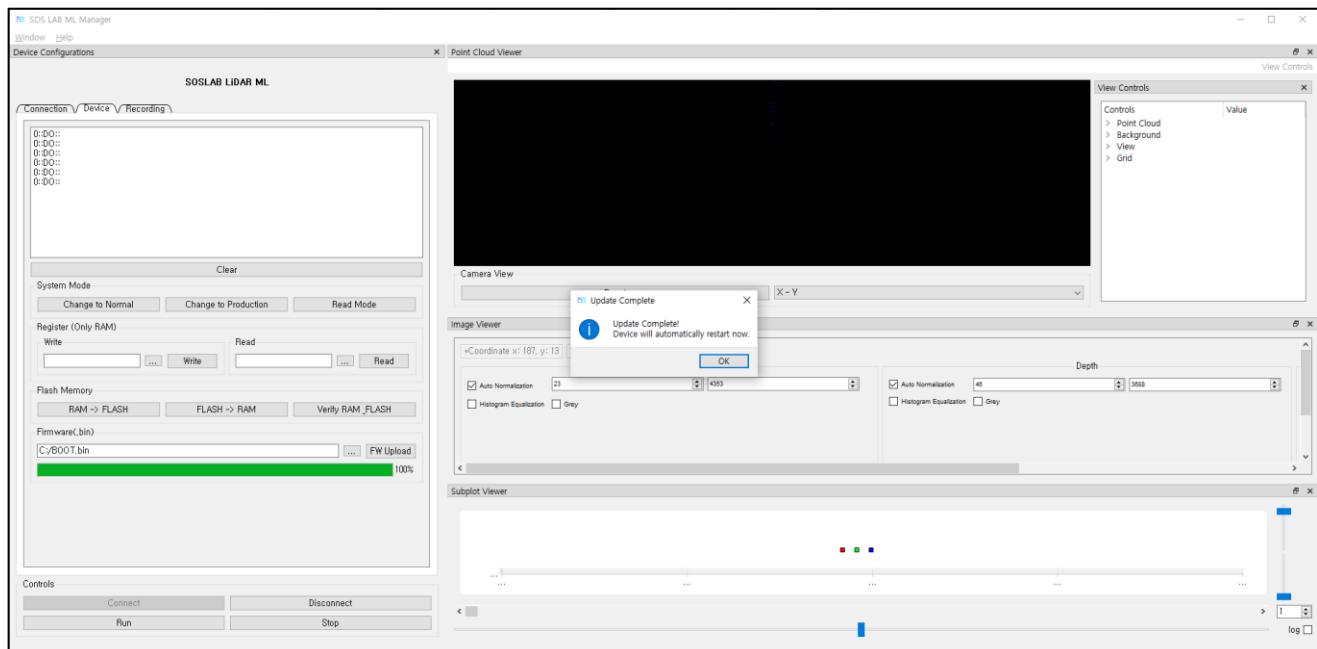
## 1.3.2. Firmware Update

Device 탭의 Firmware(.bin) 항목에서 ML의 Firmware를 업데이트 할 수 있습니다.

- 1) 네트워크 케이블로 PC와 ML LiDAR를 연결 한 뒤, “Connect” 버튼을 클릭합니다.
- 2) Firmware(.bin) 항목의 [...] 버튼을 클릭하여 Firmware 파일을 선택합니다.
- 3) Firmware(.bin) 항목의 [FW Upload] 버튼을 클릭하여 Firmware 업데이트를 진행합니다.
- 4) 아래 그림의 창이 열리면, Firmware 파일 경로를 확인하고 OK 버튼을 클릭합니다.



- 5) Firmware 업데이트가 정상적으로 완료되고 아래 그림의 창이 열리면 ML 전원을 끈 뒤, 다시 연결하면 Firmware 업데이트가 완료됩니다.



ML Firmware update 완료 화면

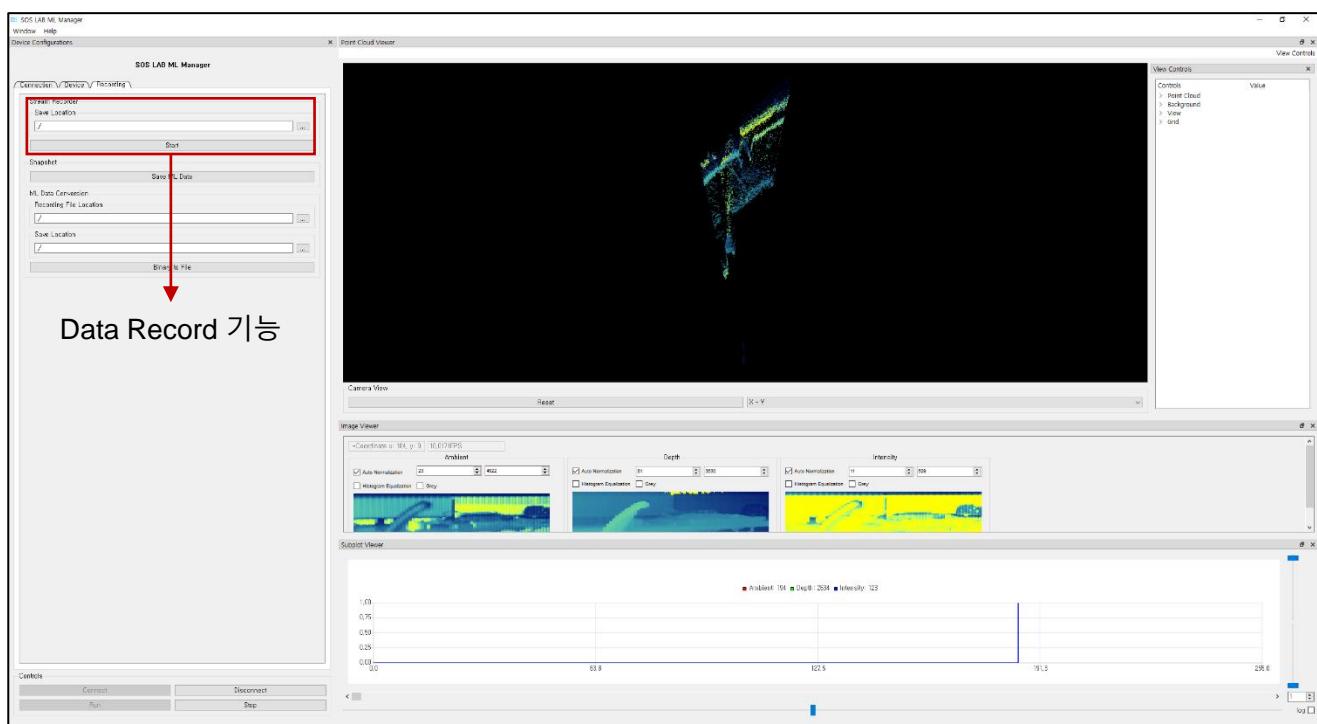
# 1.4. LiDAR Record / Play Mode

ML Manager에서는 ML 데이터를 녹화/재생하는 기능을 제공합니다.

## 1.4.1. LiDAR Data Record

Device configuration창의 Recording탭에서 ML 데이터 실시간 녹화 기능과 캡쳐 기능을 사용할 수 있습니다. 실시간 녹화 기능의 사용 방법은 아래와 같습니다.

- 1) Stream mode를 선택한 뒤, 네트워크 케이블로 PC와 ML LiDAR를 연결하고 “Connect”, “Run” 버튼을 클릭합니다.
- 2) Recording탭으로 이동한 후, Stream Recorder 박스에 있는 Save location에  버튼을 클릭하여 저장 위치를 선택합니다.
- 3) “Start” 버튼을 클릭하여 LiDAR 데이터 Recording을 시작합니다.
- 4) “Stop” 버튼을 클릭하여 Recording을 멈춥니다.
- 5) 사전에 지정한 위치에 저장된 파일이 생성됩니다.
  - 파일명 : “월-일-시-분-초.bin”



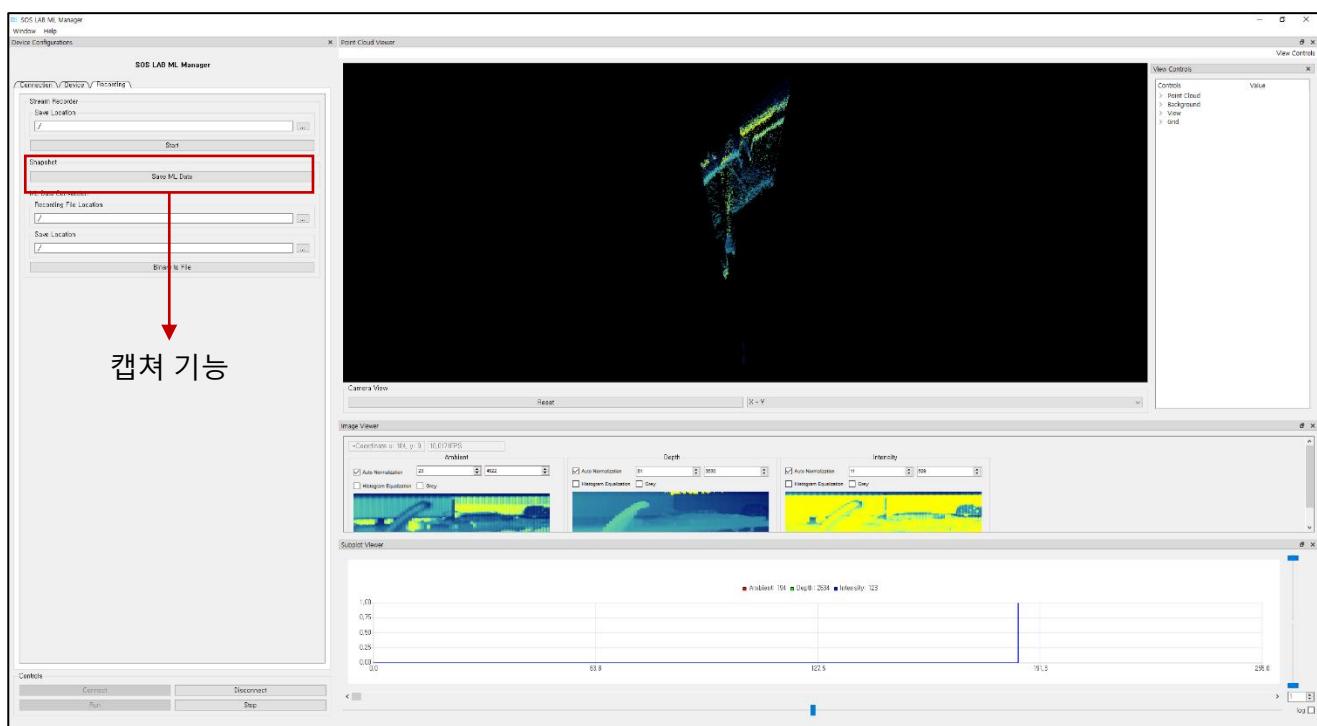
LiDAR Data Recording탭 화면

# 1.4. LiDAR Record / Play Mode

## 1.4.1. LiDAR Data Record

캡쳐 기능은 한 Frame의 LiDAR 데이터를 pcd, png파일로 저장할 수 있습니다. 사용 방법은 아래와 같습니다.

- 1) Stream mode를 선택한 뒤, 네트워크 케이블로 PC와 ML LiDAR를 연결하고 “Connect” 버튼, “Run” 버튼을 클릭합니다.
- 2) Recording탭으로 이동한 후, Snapshot 박스에 있는 “Save ML Data” 버튼을 클릭합니다.
- 3) 데이터를 저장할 위치와 이름을 선택한 후 저장을 하면 해당 위치에 1개의 pcd파일과 3개의 png파일이 생성됩니다.

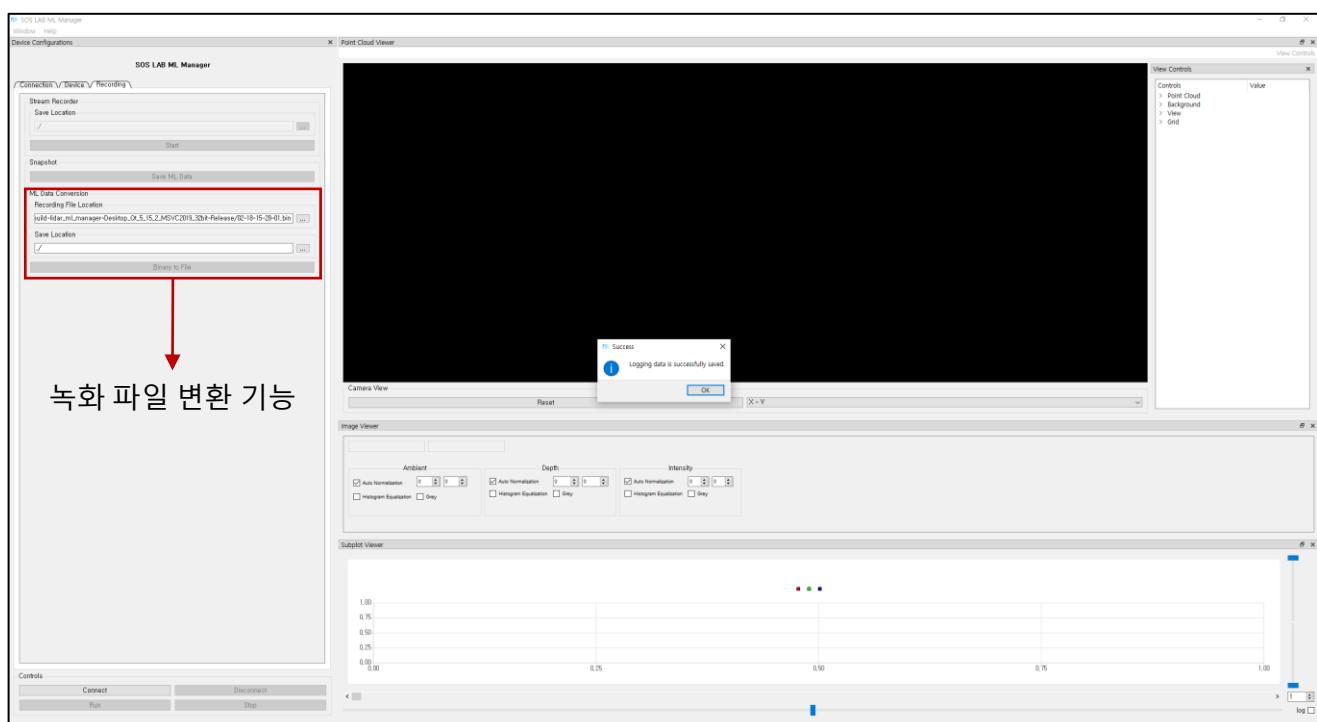


# 1.4. LiDAR Record / Play Mode

## 1.4.2. LiDAR Data Conversion

Recording 탭의 ML Data Conversion 박스는 녹화 기능을 통해 저장된 binary 파일을 pcd 파일로 변환해주는 기능을 제공합니다. 사용 방법은 아래와 같습니다.

- 1) Recording 탭으로 이동한 후, ML Data Conversion 박스에 있는 Recording File Location 항목의 [...] 버튼을 클릭하여 녹화 파일을 선택합니다.
- 2) Recording File Location 항목 아래 Save Location 항목의 [...] 버튼을 클릭하여 저장 폴더 위치를 선택합니다.
- 3) “Binary to File” 버튼을 클릭하여 녹화 파일을 pcd 파일로 변환합니다.  
※ 녹화 파일 용량에 따라서 시간이 소요될 수 있습니다.
- 4) 아래 그림과 같은 창이 열리며 녹화 파일 변환이 완료됩니다.



LiDAR Data Conversion 완료 화면

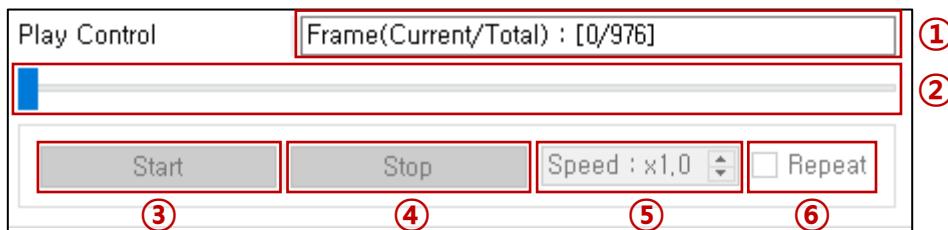
# 1.4. LiDAR Record / Play Mode

## 1.4.3. Play mode

Connection 탭의 Play mode는 녹화 기능을 통해 저장된 binar파일을 재생하는 기능을 제공합니다. 사용 방법은 아래와 같습니다.

- 1) Play mode를 선택한 뒤, Recording File Location 항목의 [...] 버튼을 클릭하여 녹화 파일은 선택합니다.
- 2) 하단의 “Connect” 버튼을 클릭하여 녹화 파일을 Load 합니다.  
※ 녹화 파일 용량에 따라서 시간이 소요될 수 있습니다.
- 3) “Run” 버튼이 활성화 되면 클릭한 뒤, Play mode 박스의 “Start” 버튼을 클릭하여 녹화 파일을 재생합니다.

Play mode의 control은 다음과 같은 항목으로 구성됩니다.

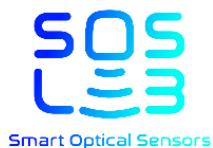


Play mode Control 화면

1. 상태 창 : 현재 Frame과 총 Frame을 알려줍니다.
2. Frame bar : 파란색 상자를 클릭하여 움직이면서 현재 Frame을 변경합니다.
3. Start 버튼 : 녹화 파일을 재생합니다.
4. Stop 버튼 : 녹화 파일 재생을 멈춥니다.
5. Speed box : 재생 속도를 변경합니다. (최소 0.1 배속, 최대 10배속)
6. Repeat 체크 박스 : 녹화 파일 재생을 반복합니다.

# Chapter 2

## ML LiDAR API (Windows)



## 2.1. ML API Build for Windows

ML SDK는 ML API와 API를 활용한 4개의 예제 코드를 제공합니다.

API는 libsoslab\_core, libsoslab\_ml 두 개이며, 32bit/64bit 환경에서 사용 가능한 lib/dll 파일로 구성되며 이를 활용한 예제 코드는 아래와 같이 구성됩니다.

- test\_core : libsoslab\_core API를 활용한 예제
- test\_ml : libsoslab\_core API를 활용한 ML 연결 예제
- test\_multi\_ml : 다중 ML LiDAR 연결 예제
- test\_record : ML 데이터 녹화 / 변환 예제

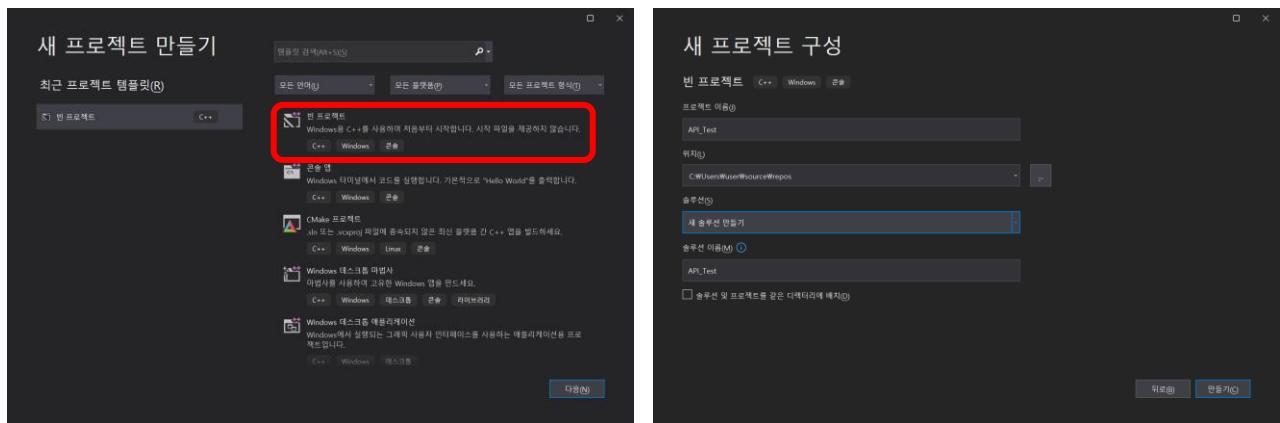
test\_core 예제 코드와 test\_record 예제 코드는 libsoslab API를 사용합니다. test\_ml 예제 코드와 test\_multi\_ml 예제 코드는 ML 데이터 가시화를 위하여 외부 라이브러리를 사용합니다. 예제 코드를 위한 외부 라이브러리는 아래 version을 사용하길 권장합니다.

- [test\_ml 예제 코드] OpenCV : 4.2.0
- [test\_multi\_ml 예제 코드] PCL : 1.9.1
- [test\_multi\_ml 예제 코드] Boost : 1.62
- [test\_multi\_ml 예제 코드] VTK : 8.2.0

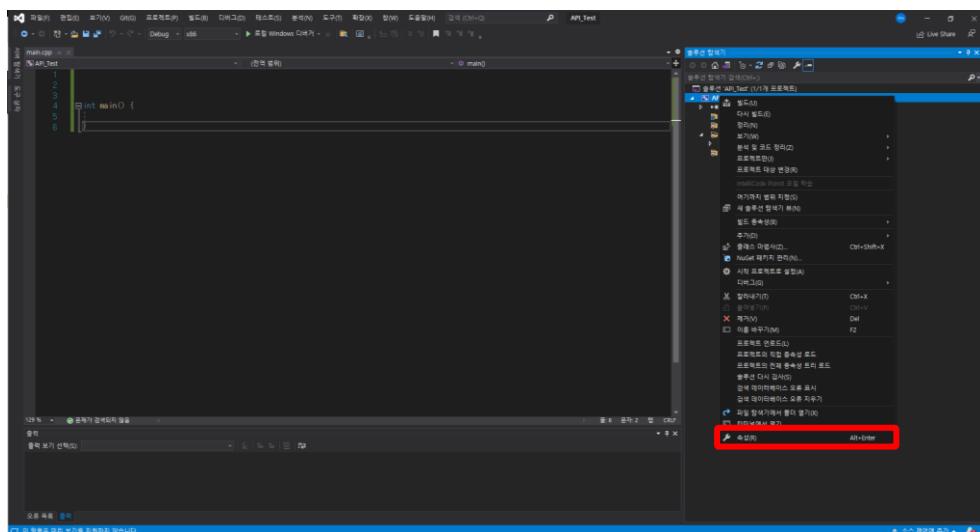
## 2.1. ML API Build for Windows

Visual Studio의 새로운 프로젝트를 생성하여 ML API 연동하는 방법은 다음과 같습니다.

(1) Visual Studio를 열어 API 사용을 위한 새로운 빈 프로젝트를 생성합니다.

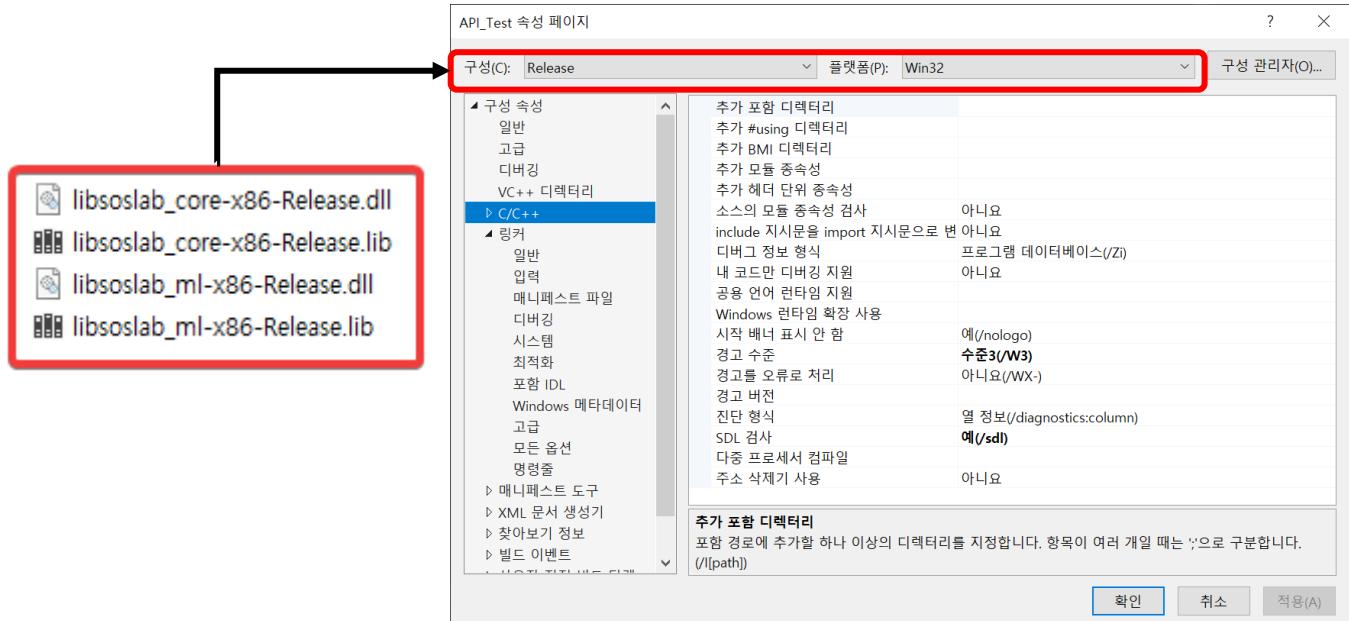


(2) Build Setting을 위하여 Project에 main.cpp 생성 후, 아래 그림과 같이 프로젝터 속성을 클릭 또는 단축키(Alt + Enter)를 입력하여 엽니다.



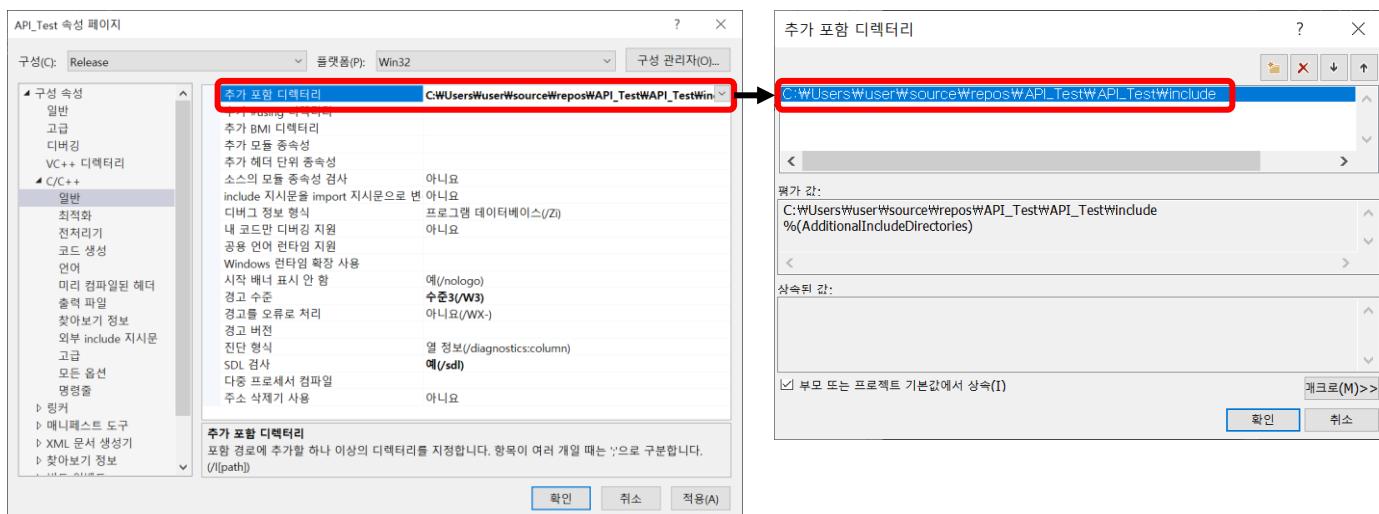
## 2.1. ML API Build for Windows

(3) 프로젝트 속성 페이지에서 구성과 플랫폼을 ML API build 세팅과 동일하게 변경합니다.  
ex) ML\_API가 32bit, Release일 경우, 프로젝터 구성은 Release, 플랫폼은 Win32 또는 x86으로 변경합니다.



### ML API 빌드 환경과 Visual Studio 프로젝트 빌드 환경 일치

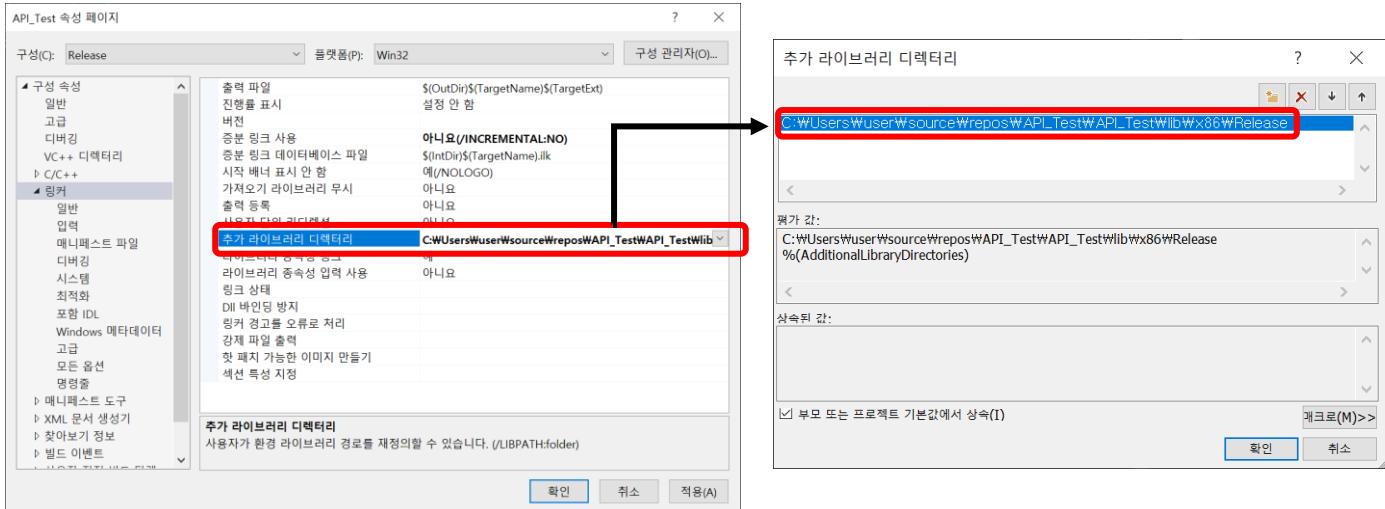
(4) ML API header 파일 경로 설정을 위해 속성 페이지 왼쪽 C/C++ - 일반 항목에서 추가 포함 디렉토리 항목에 ML API header 경로를 입력합니다.



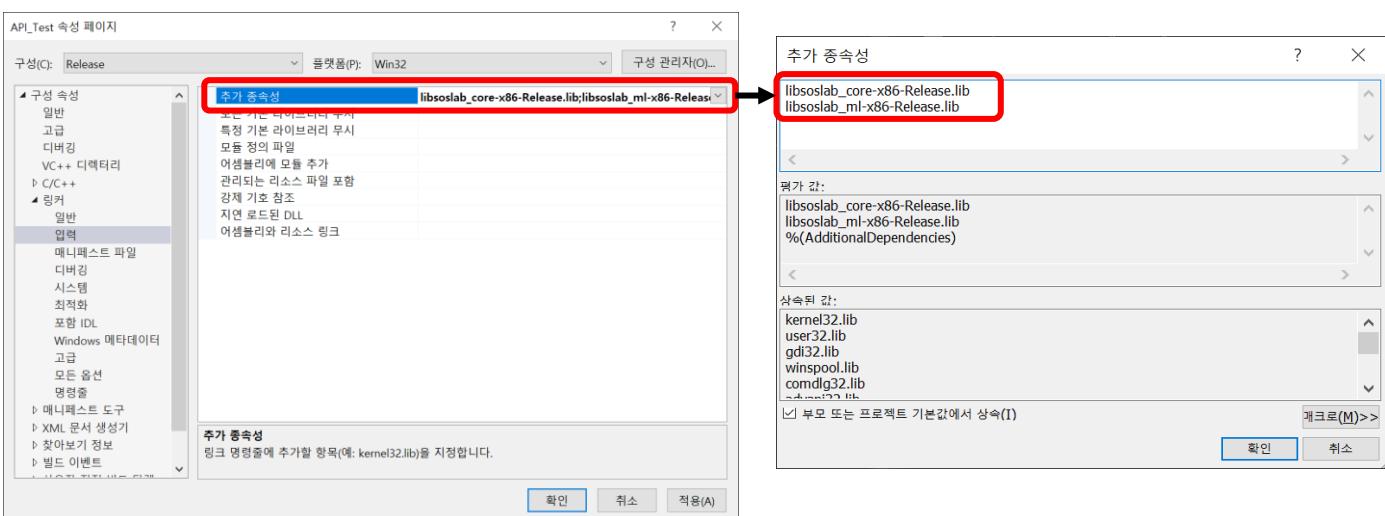
### ML API header 파일 경로 설정

## 2.1. ML API Build for Windows

(4) API 정적 라이브러리(.lib) 경로 설정을 위하여 속성 페이지 왼쪽 링커 – 일반 항목에서 추가 라이브러리 디렉터리 항목에 API 정적 라이브러리(.lib) 경로를 입력합니다. 그리고 링커 – 입력 항목에서 추가 종속성 항목에 정적 라이브러리 파일명을 입력합니다.



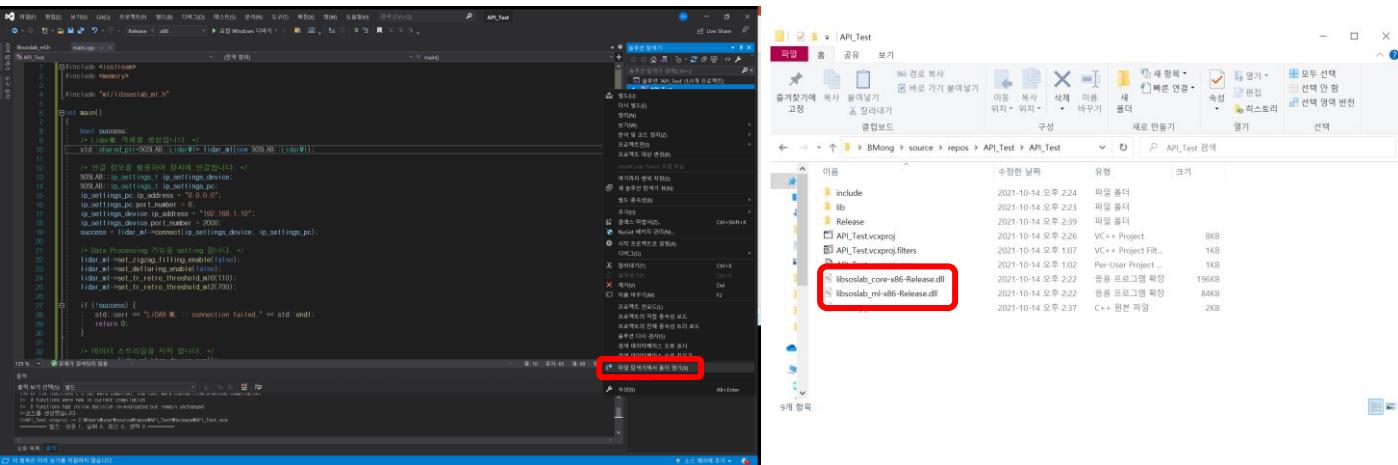
### ML API 정적 라이브러리 경로 설정



### ML API 정적 라이브러리 추가

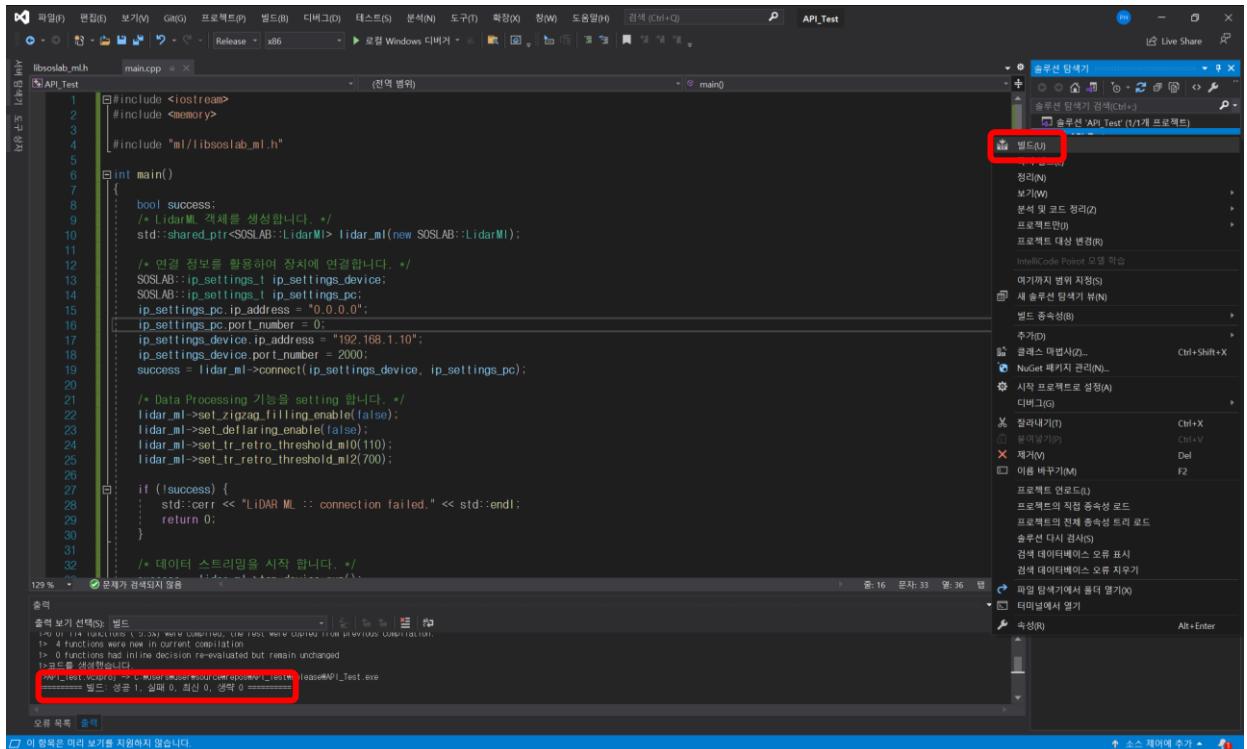
## 2.1. ML API Build for Windows

(5) 프로젝트 아이콘에 오른쪽 버튼을 클릭하여 파일 탐색기에서 폴더 열기 항목을 통해 현재 프로젝트 폴더를 연 후, ML API 동적 라이브러리를 프로젝트 폴더로 이동합니다.



### ML API 동적 라이브러리 위치 이동

(6) build 테스트를 위하여 제공된 예제 코드를 빌드하여 출력창에 성공 1, 실패 0이 출력되면 build가 완료됩니다.



### Example Code with ML API build

## 2.2. Example Code for Windows

### 2.2.1. Connect ML Device

```
1. bool success;
2. std::shared_ptr<SOSLAB::LidarML> lidar_ml(new SOSLAB::LidarML);

3. SOSLAB::ip_settings_t ip_settings_device;
4. SOSLAB::ip_settings_t ip_settings_pc;
5. ip_settings_pc.ip_address = "0.0.0.0";
6. ip_settings_pc.port_number = 0;
7. ip_settings_device.ip_address = "192.168.1.10";
8. ip_settings_device.port_number = 2000;
9. success = lidar_ml->connect(ip_settings_device, ip_settings_pc);
10. if (!success) {
11.     std::cerr << "LiDAR ML :: connection failed." << std::endl;
12.     return 0;
13. }

14. success = lidar_ml->tcp_device_run();
15. if (!success) {
16.     std::cerr << "LiDAR ML :: start failed." << std::endl;
17.     return 0;
18. }
19. std::cout << "LiDAR ML :: Streaming started!" << std::endl;
```

ML device와 PC의 통신을 위한 ML API 예제 코드입니다. pc IP 주소와 port 번호는 예제 코드와 동일하게, ML device의 IP 주소와 Port 번호는 제공된 정보와 같이 사용하시면 됩니다.

Line	Description
2	Local(PC)와 ML Device와의 통신을 위한 SOSLAB::LidarML 변수를 lidar_ml로서 선언합니다.
3-8	SOSLAB::ip_settings_t 변수를 Local(PC)과 ML device에 대하여 두 개를 선언합니다. 예제 코드에서는 ip_setting_device, ip_setting_pc라는 이름으로 선언했습니다. 선언된 변수에 IP 주소, Port 번호와 ML device의 IP 주소, Port 번호를 구조체의 ip_address, port_number 변수 값에 대입합니다.
9-13	정의된 Local, ML ip_settings_t 변수를 LidarML Class의 connect 함수의 입력 변수로 하여 ML device와 연결합니다. 입력 변수 순서는 ML device ip setting, local(pc) ip setting 순입니다. 이후, connect 함수의 반환 값으로 연결 성공 여부를 bool 변수형으로 얻습니다.
14-19	ML device와의 연결이 완료되면 LidarML Class의 tcp_device_run()을 통해 scanning 을 시작합니다. 함수의 반환값으로 scanning 시작 여부를 bool 변수형으로 얻습니다.

## 2.2. Example Code for Windows

### 2.2.2. Setting Enable/Parameter of Data Processing Modules

```
1. lidar_ml->set_zigzag_filling_enable(true);  
2. lidar_ml->set_deflaring_enable(false);  
3. lidar_ml->set_real_ambient_enable(false);  
4. lidar_ml->set_line_filling_enable(true);  
5. lidar_ml->set_tr_retro_threshold_ml0(110);  
6. lidar_ml->set_tr_retro_threshold_ml2(700);
```

ML API에는 Raw data 성능 향상을 위한 data processing module이 기본적으로 포함되어 있습니다. 해당 기능을 사용하지 않거나, parameter를 변경하기 위해서 사용되는 함수에 대한 예제입니다. setting을 위한 함수는 scanning 시작하는 함수(tcp\_device\_run(), start()) 전에 호출하시면 됩니다. 각 함수의 기본값은 appendix를 참조하시면 됩니다.

Line	Description
1	ML device에서 Data Processing Module의 zigzag filling 기능을 ON(true)/OFF(false) 하는 set_zigzag_filling_enable 함수를 호출합니다
2	ML device에서 Data Processing Module의 deflaring 기능을 ON(true)/OFF(false) 하는 set_deflaring_enable 함수를 호출합니다.
3	ML device에서 Data Processing Module의 real ambient 기능을 ON(true)/OFF(false) 하는 set_real_ambient_enable 함수를 호출합니다.
4	ML2 device에서 Data Processing Module의 line filling 기능을 ON(true)/OFF(false) 하는 set_line_filling_enable 함수를 호출합니다.
5	deflaring 기능의 parameter 값을 변경하는 함수를 호출합니다. ML0 device에서는 set_tr_retro_threshold_ml0 함수를 호출합니다. 변경할 parameter 값을 입력으로 받습니다.
6	deflaring 기능의 parameter 값을 변경하는 함수를 호출합니다. ML2 device에서는 set_tr_retro_threshold_ml2 함수를 호출합니다. 변경할 parameter 값을 입력으로 받습니다.

## 2.2. Example Code for Windows

### 2.2.3. Get Raw Data

```
1. while (keyinput_checker(cv::waitKey(1))) {  
2.     SOSLAB::LidarMl::scene_t scene;  
3.     if (lidar_ml->get_scene(scene)) {  
4.         std::vector<uint16_t> ambient = scene.grey_image;  
5.         std::vector<uint16_t> intensity = scene.intensity_image;  
6.         std::vector<uint32_t> depth = scene.depth_image;  
7.         std::vector<SOSLAB::point_t> pointcloud = scene.pointcloud;  
8.     }  
9. }
```

ML device으로부터 raw data를 획득하는 방법에 대한 예제 코드입니다. raw data는 `scene_t`로 획득되며, 총 4개의 1차원 데이터 배열(Ambient, intensity, depth, pointcloud)로 구성되어 있습니다. 자세한 `scene_t` 구조체 구조에 대한 설명은 appendix를 참조하시면 됩니다.

Line	Description
2	Raw data를 저장할 <code>SOSLAB::LidarMl::scene_t</code> 변수를 <code>scene</code> 이름으로 선언합니다.
3	<code>scene_t</code> 변수를 입력으로 받는 <code>SOSLAB::LidarMl</code> class의 <code>get_scene</code> 함수를 통해서 <code>scene</code> 변수에 raw data를 획득합니다.
4-6	raw data를 저장한 <code>scene_t</code> <code>scene</code> 구조체로부터 <code>ambient</code> , <code>intensity</code> , <code>depth</code> , <code>pointcloud</code> 데이터를 획득합니다. 각 데이터의 구조체 변수명과 변수형은 appendix에 참조하시면 됩니다.

## 2.2. Example Code for Windows

### 2.2.4. Convert Raw Data(Depth, Ambient, Intensity) to Image

```
1. std::string depth_viz_name = "Depth Image";
2. cv::Mat depth_image_raw(scene.rows, scene.cols, CV_32SC1,
   scene.depth_image.data());
3. cv::Mat depth_8b;
4. double depth_scale = 32.0;
5. depth_image_raw.convertTo(depth_8b, CV_8U, 1.0 / depth_scale);
6. cv::Mat depth_rgb;
7. cv::applyColorMap(255 - depth_8b, depth_rgb, cv::COLORMAP_VIRIDIS);
8. cv::imshow(depth_viz_name, depth_rgb);
```

1차원 배열로 형태의 Ambient, Depth, Intensity 데이터를 이미지로 변환하여 가시화하는 예제 코드입니다. 이미지 가시화는 OpenCV library를 활용하였습니다.

Line	Description
2	scene_t 구조체의 depth 데이터를 이미지로 가시화하기 위해 opencv의 cv::Mat 형식으로 변환하는 과정입니다. cv::Mat 초기화 입력 값으로 Raw data의 height(scene.rows), width(scene.cols) 그리고 depth 데이터 type과의 일치(CV_32SC1)시키고 Raw data 값을 넣습니다.
3-5	이미지 가시화를 위하여 uint32_t(CV_32SC1) 변수 타입을 uchar(CV_8U) 타입으로 변환하면서 depth scale을 조절합니다.
6-8	1 channel 이미지인 depth_8b 변수에 3 channel Colormap(cv::COLORMAP_VIRIDIS)으로 변환 후, imshow 함수를 통해 raw depth data를 가시화합니다.

## 2.2. Example Code for Windows

### 2.2.4. Convert Raw Data(Depth, Ambient, Intensity) to Image

```
1. std::string grey_viz_name = "Grey Image";
2. cv::Mat grey_image_raw(scene.rows, scene.cols, CV_16SC1,
   scene.grey_image.data());

3. cv::Mat grey_image_norm;
4. cv::normalize(grey_image_raw, grey_image_norm, 0, 255,
   cv::NORM_MINMAX)

5. cv::Mat grey_rgb;
6. grey_image_norm.convertTo(grey_rgb, CV_8U);
7. cv::applyColorMap(grey_rgb, grey_rgb, cv::COLORMAP_VIRIDIS);
8. cv::imshow(grey_viz_name, grey_rgb);
```

Line	Description
2	scene_t 구조체의 Ambient 데이터를 이미지로 가시화하는 과정입니다. cv::Mat 형식으로 변환하기 위하여 데이터 type을 uint16_t(CV_16SC1)으로 설정합니다. 그리고 이미지 크기, 데이터 정의하는 과정은 Depth 데이터 과정과 동일하게 진행합니다.
3-5	이미지 가시화를 위하여 ambient data의 최소, 최대 값 기준으로 0~255값으로 정규화 합니다.
6-8	Depth 이미지 가시화와 동일하게 진행합니다.

```
1. std::string intensity_viz_name = "Intensity Image";
2. cv::Mat intensity_image_raw(scene.rows, scene.cols, CV_16SC1,
   scene.intensity_image.data());

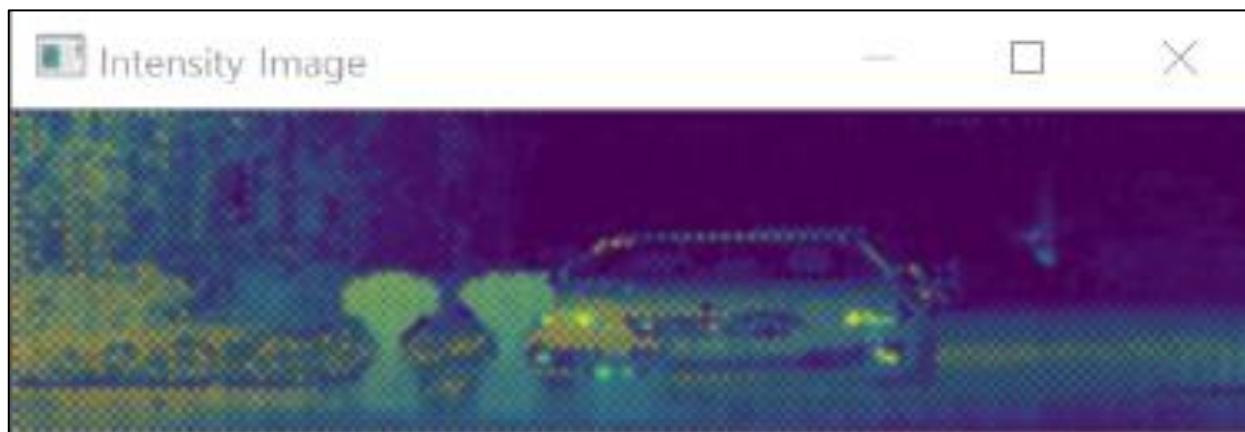
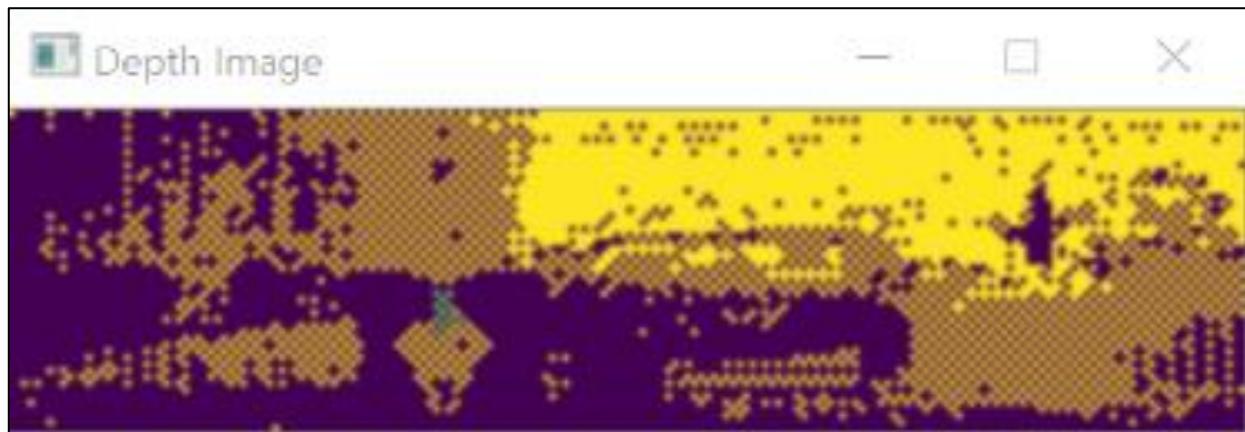
3. cv::Mat intensity_image_norm;
4. cv::normalize(intensity_image_raw, intensity_image_norm, 0, 255,
   cv::NORM_MINMAX)

5. cv::Mat intensity_rgb;
6. intensity_image_norm.convertTo(intensity_rgb, CV_8U);
7. cv::applyColorMap(intensity_rgb, intensity_rgb, cv::COLORMAP_VIRIDIS);
8. cv::imshow(intensity_viz_name, intensity_rgb);
```

Line	Description
2-8	scene_t 구조체의 intensity 데이터를 이미지로 가시화하기 위해 opencv의 cv::Mat 형식으로 변환하는 과정입니다. Ambient 데이터 가시화 동일하게 진행됩니다

## 2.2. Example Code for Windows

### 2.2.4. Convert Raw Data(Depth, Ambient, Intensity) to Image



Opencv Library를 활용한 Ambient, Depth, Intensity 이미지 가시화

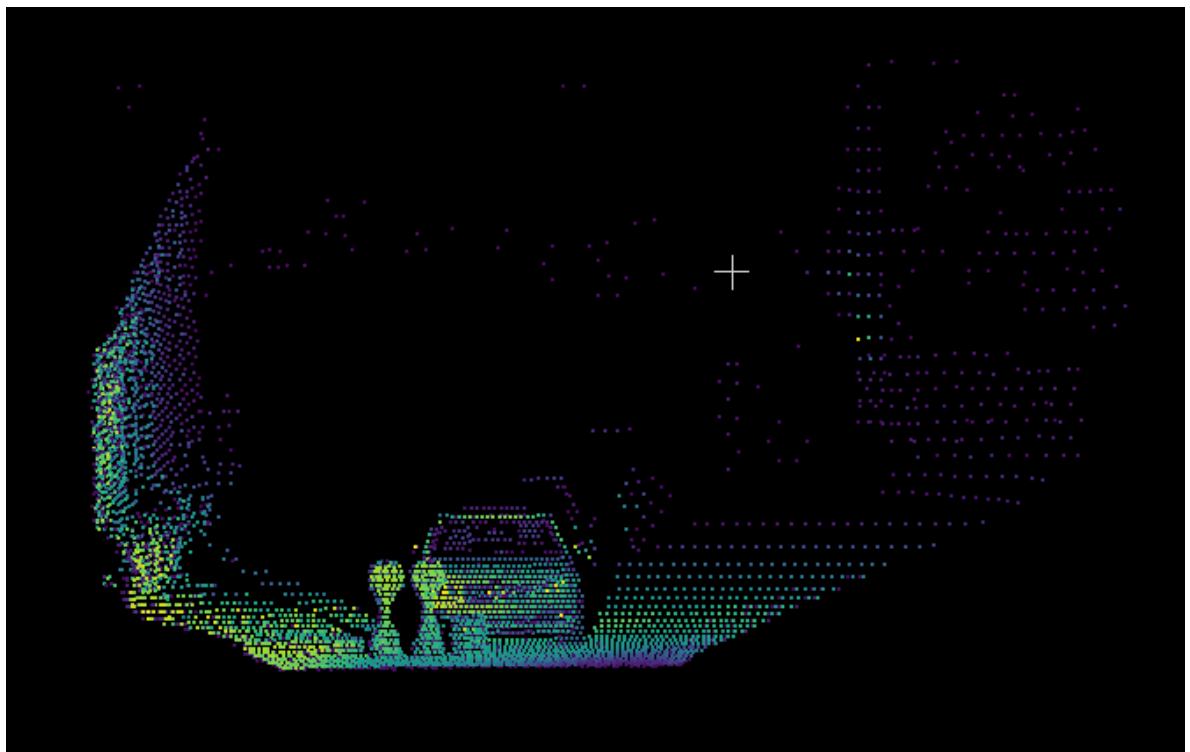
## 2.2. Example Code for Windows

### 2.2.5. Save Raw Point Cloud

```
1. std::string title = std::to_string(scene.timestamp[0]) + "_pc.txt"
2. std::ofstream write_pointcloud(title);
3. for (int i = 0; i < scene.pointcloud.size(); i++) {
4.     double x = scene.pointcloud[i].xyz.x;
5.     double y = scene.pointcloud[i].xyz.y;
6.     double z = scene.pointcloud[i].xyz.z;
7.     double intensity = scene.intensity_image[i];
8.     write_pointcloud << x << "," << y << "," << z << ","
9.     << intensity << std::endl;
10. }
10. write_pointcloud.close();
```

1차원 배열 형태의 raw point cloud 데이터의 txt 파일로 저장하는 예제 코드입니다.

Line	Description
1	raw data 획득 시간(scene.timestamp[0])을 파일명으로 정의합니다.
3-8	raw point cloud를 저장하기 위하여 SOSLAB::scene_t 구조체에서 std::vector<SOSLAB::point_t> 배열로 저장된 point cloud data의 위치 정보(x,y,z)와 intensity_image로부터 반사 강도 값을 획득하고 std::ofstream을 통해서 저장합니다.



CloudCompare 프로그램을 사용하여 저장된 point cloud(x, y, z, intensity) 가시화

## 2.2. Example Code for Windows

### 2.2.6. Data Recording example

```
1. std::string save_directory = "./log/"
2. #ifdef _WIN32
3.     if (_access(save_directory.c_str(), 0)) {
4.         if (_mkdir(save_directory.c_str())) return false;
5.     }
6. #endif // _WIN32
7. #ifdef __linux__
8.     mkdir(save_directory.c_str(), S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
9. #endif
10. Bool retval = lidar_ml->start_recording(save_directory);
11. int frame_number = 0;
12. int logging_data_size = 10;
13. while (frame_number < logging_data_size) {
14.     SOSLAB::LidarML::scene_t scene;
15.     if (lidar_ml->get_scene(scene)) {
16.         std::size_t height = scene.rows;
17.         std::size_t width = scene.cols;
18.         frame_number++;
19.     }
20. }
21. lidar_ml->stop_recording();
```

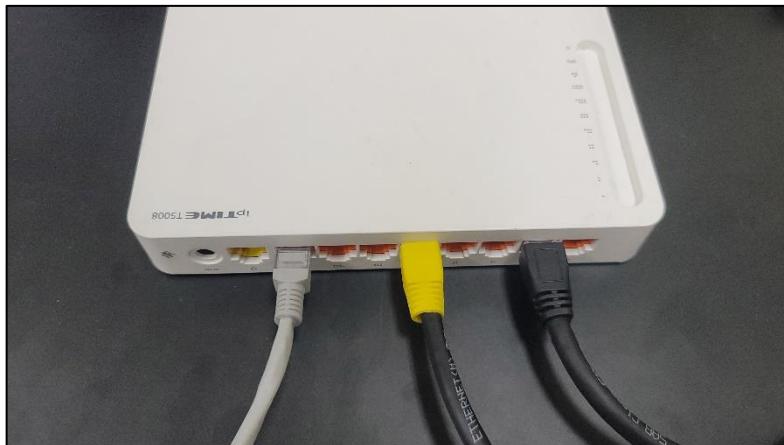
Line	Description
1-9	녹화 파일의 저장 위치(save_directory)를 설정하고 해당 폴더가 없으면 생성합니다.
10	SOSLAB::LidarML class의 start_recording(save_directory) 함수를 호출하여 녹화를 시작합니다.
11-20	scene_t 변수를 입력으로 받는 SOSLAB::LidarML class의 get_scene 함수를 통해서 scene 변수에 raw data를 획득합니다.
21	SOSLAB::LidarML class의 stop_recording() 함수를 호출하여 녹화를 중지합니다.

## 2.2. Example Code for Windows

### 2.2.7. Multi-LiDAR Example

다중 ML 가시화 예제 코드는 다중 ML의 네트워크 케이블을 공유기에 연결한 뒤, 공유기와 PC 사이의 Lan 케이블을 연결하여 다중 ML과 PC 연결을 합니다.

다중 ML 가시화를 위해서 ML의 Device setting의 ML\_IP와 PC\_PORT 그리고 MACADDR 항목 값을 ML마다 다르게 변경합니다. Device setting 변경 방법은 1.3.1 항목에 자세히 작성되어 있습니다.



2개의 ML 연결을 위한 공유기 연결 예시

The image shows two windows of the Windows Notepad application. The left window is titled "first\_ml\_setting.txt" and the right window is titled "second\_ml\_setting.txt". Both windows display text configurations for LiDAR devices.

**Content of first\_ml\_setting.txt:**

```
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
ML_IP,UINT8,4,192,168,1,10  
PC_IP,UINT8,4,192,168,1,15  
PC_PORT,UINT16,1,2000  
MACADDR,UINT8,6,0,128,225,0,0,1
```

**Content of second\_ml\_setting.txt:**

```
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
ML_IP,UINT8,4,192,168,1,11  
PC_IP,UINT8,4,192,168,1,15  
PC_PORT,UINT16,1,2001  
MACADDR,UINT8,6,0,128,225,0,0,2
```

1번 ML Device Setting

2번 ML Device Setting

2개의 ML 연결을 위한 device setting 예시

## 2.2. Example Code for Windows

### 2.2.7. Multi-LiDAR Example

```
1.  bool success = true;
2.  int num_lidar = 1;
3.  std::vector<std::shared_ptr<SOSLAB::LidarML>> lidar_ml_list(new
SOSLAB::LidarML);
4.  int standard_port = 2000;
5.  for (int i = 0; i < num_lidar; i++) {
6.      SOSLAB::ip_settings_t ip_settings_pc;
7.      SOSLAB::ip_settings_t ip_settings_device;
8.      ip_settings_pc.ip_address = "192.168.1.15";
9.      ip_settings_pc.port_number = standard_port + i;
10.     ip_settings_device.ip_address = "192.168.1.1" + to_string(i);
11.     ip_settings_device.port_number = standard_port;
12.     success = lidar_ml_list[i]->connect(ip_settings_device,
ip_settings_pc);
13.     if (!success) {
14.         std::cerr << "LiDAR ML #" + std::to_string(i) + " :: connection failed" << std::endl;
15.     }
16.     for (int i = 0; i < num_lidar; i++) {
17.         success &= lidar_ml_list[i]->tcp_device_run();
18.         if (!success) {
19.             std::cerr << "LiDAR ML #" + std::to_string(i) + " :: start failed" << std::endl;
20.             return 0;
21.         }
22.     }
23.  }
```

Line	Description
2	Local(PC)와 ML Device와의 통신을 위한 SOSLAB::LidarML 객체를 연결할 ML 개수 (num_lidar)에 따라서 초기화 합니다. (default num_lidar = 1)
4-15	예제 코드에서는 초기 값 Device IP(192.168.1.10), PC port(2000)에서 ML 개수에 따라 Device IP, PC port의 마지막 값이 1씩 증가시켜 PC와 연결합니다. 연결 방법은 2.3.1과 동일합니다. - Example : 2개의 ML 연결 #1 ML Device port = "192.168.1.10", PC port = "2000" #2 ML Device port = "192.168.1.11", PC port = "2001"
16-18	모든 ML과 PC를 연결한 뒤, ML 개수에 맞게 LidarML Class의 tcp_device_run()을 호출하여 scanning을 시작합니다.

## 2.2. Example Code for Windows

### 2.2.7. Multi-LiDAR Example

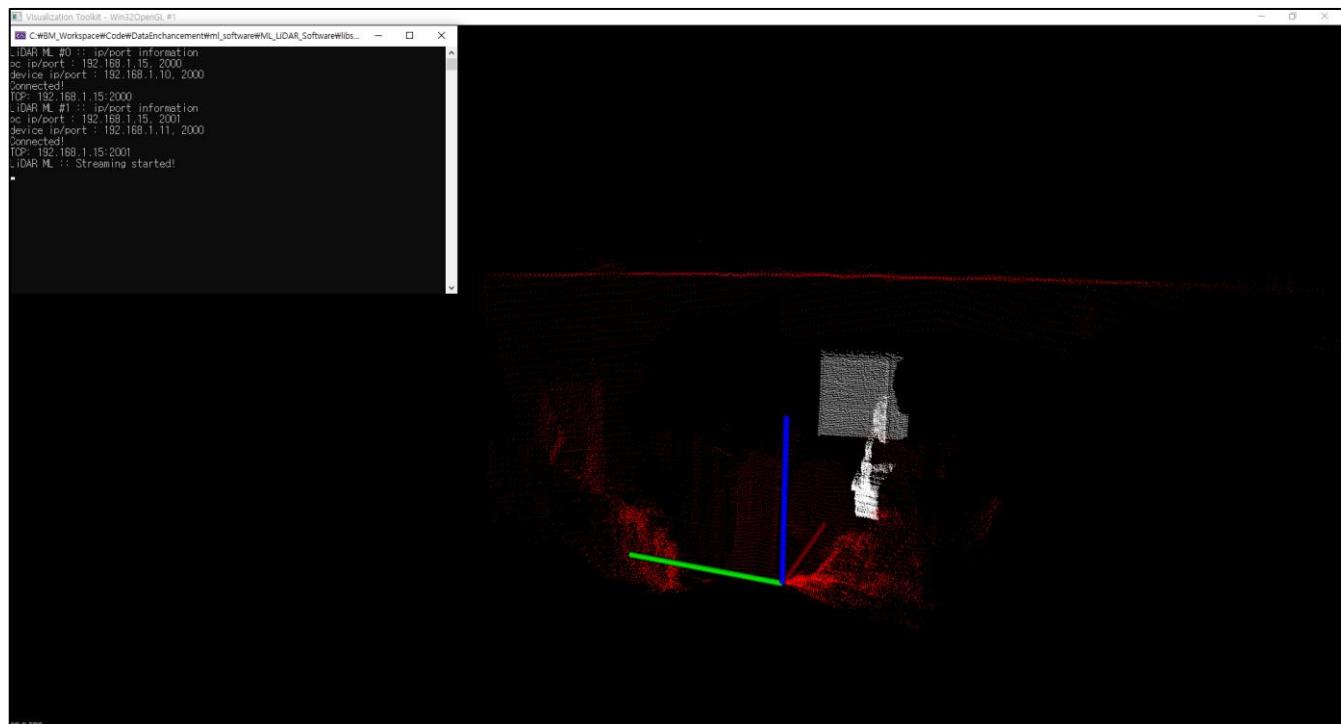
```
1.     int points_size = 0;
2.     while (!pclviz_->wasStopped()) {
3.         SOSLAB::LidarML::scene_t scene;
4.         for (int i = 0; i < num_lidar; i++) {
5.             if (lidar_ml_list[i]->get_scene(scene)) {
6.                 std::size_t height = scene.rows;
7.                 std::size_t width = scene.cols;
8.                 if (initialize) size_list[i+1] = height * width;
9.                 if (!scene.pointcloud.empty() && streaming_start) {
10.                     for (int j = 0; j < size_list[i+1]; j++) {
11.                         PointT point;
12.                         point.x = scene.pointcloud[j].xyz.x / 1000.0f;
13.                         point.y = scene.pointcloud[j].xyz.y / 1000.0f;
14.                         point.z = scene.pointcloud[j].xyz.z / 1000.0f;
15.                         point.r = uint8_t(color_vector[i][0]);
16.                         point.g = uint8_t(color_vector[i][1]);
17.                         point.b = uint8_t(color_vector[i][2]);
18.                         cloud_->points[j+ clouds_start_points[i]] = point;
19.                     }
20.                 }
21.             }
22.         }
23.         if (!pclviz_->updatePointCloud(cloud_, "cloud")) {
24.             pclviz_->addPointCloud(cloud_, "cloud");
25.             pclviz_->spinOnce(1);
26.         }
}
```

Line	Description
2-3	PCL 가시화 창이 끝날 때까지 ML 데이터를 획득하는 Loop를 시작합니다.
4-5	SOSLAB::LidarML 객체를 저장하는 lidar_ml_list의 각 요소에 접근하여 get_scene 함수를 호출하여 각 LiDAR의 데이터를 scene 변수에 저장합니다.
6-14	2.2.5 항목과 동일한 방법으로 scene 변수의 Point 정보 획득하여 PointT 변수의 값으로 설정합니다.
15-17	연결된 LiDAR 순서에 따라서 다른 색상을 설정합니다. 1번 : 흰색, 2번 : 빨강, 3번 : 초록, 4번 : 파랑
18	PointT 변수에 저장된 Point를 가시화를 위한 cloud_ 변수에 저장합니다.
23-25	Loop를 돌면서 저장된 모든 LiDAR의 Point Cloud 정보를 저장하는 cloud_ 변수를 PCL 가시화 객체에 등록합니다.

## 2.2. Example Code for Windows

### 2.2.7. Multi-LiDAR Example

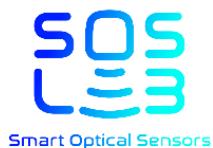
Multi-LiDAR 예제의 결과는 아래 그림과 같이 다중 LiDAR의 점군을 PCL VTK 모듈을 사용하여 가시화합니다. 아래 그림은 2개의 ML을 공유기를 통해 PC에 연결한 Point Cloud를 1번 LiDAR는 흰색, 2번 LiDAR는 빨간색으로 가시화됩니다.



2개의 ML 연결을 위한 device setting 예시

# Chapter 3

## ML LiDAR API (Ubuntu/ROS)



### 3.1. ML API Build for Ubuntu/ROS

ML SDK는 Ubuntu 18.04버전과 ROS는 melodic을 사용하였으며, ML API와 API를 활용한 4개의 예제 코드를 제공합니다.

API는 libsoslab\_core, libsoslab\_ml 두 개이며, 64bit 환경에서 사용 가능한 so파일로 구성되며 이를 활용한 예제 코드는 아래와 같이 구성됩니다.

- test\_core : libsoslab\_core API를 활용한 예제
- test\_ml : libsoslab\_core API를 활용한 ML 연결 예제
- test\_multi\_ml : 다중 ML LiDAR 연결 예제
- test\_record : ML 데이터 녹화 / 변환 예제

test\_core 예제 코드와 test\_record 예제 코드는 libsoslab API를 사용합니다. test\_ml 예제 코드와 test\_multi\_ml 예제 코드는 ML 데이터 가시화를 위하여 외부 라이브러리를 사용합니다. 예제 코드를 위한 외부 라이브러리는 아래 version을 사용하길 권장합니다.

- [test\_ml 예제 코드] OpenCV : 3.4.6
- [test\_multi\_ml 예제 코드] libpcl : 1.8.1
- [test\_multi\_ml 예제 코드] Boost : 1.65
- [test\_multi\_ml 예제 코드] VTK : 6.3.0

test\_multi\_ml에 필요한 PCL, Boost, VTK는 ROS를 설치하면서 함께 설치가 되며, 아래 script를 통해서도 설치가 가능합니다.

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install -y software-properties-common git
$ sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl -y
$ sudo apt-get update
$ sudo apt-get install -y libpcl-dev
```

### 3.1. ML API Build for Ubuntu/ROS

Ubuntu 18.04에서 OpenCV 설치를 위한 Script는 다음과 같습니다.

#### [OpenCV Package를 위한 사전 설치]

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential cmake
$ sudo apt-get install pkg-config
$ sudo apt-get install libjpeg-dev libtiff5-dev libpng-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
libxvidcore-dev libx264-dev libxine2-dev
$ sudo apt-get install libv4l-dev v4l-utils
$ sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-
base1.0-dev
$ sudo apt-get install libgtk2.0-dev
$ sudo apt-get install mesa-utils libgl1-mesa-dri libgtkg12.0-dev
libgtkglext1-dev
$ sudo apt-get install libatlas-base-dev gfortran libeigen3-dev
```

### 3.1. ML API Build for Ubuntu/ROS

Ubuntu 18.04에서 OpenCV 설치를 위한 Script는 다음과 같습니다.

#### [OpenCV 설치]

```
$ cd ~  
$ mkdir opencv && cd opencv  
$ wget -O opencv.zip  
https://github.com/opencv/opencv/archive/3.4.6.zip  
$ unzip 3.4.6.zip  
$ cd opencv-3.4.6  
$ mkdir build && cd build  
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \  
-D CMAKE_INSTALL_PREFIX=/usr/local \  
-D WITH_TBB=OFF \  
-D WITH_IPP=OFF \  
-D WITH_1394=OFF \  
-D BUILD_WITH_DEBUG_INFO=OFF \  
-D BUILD_DOCS=OFF \  
-D INSTALL_C_EXAMPLES=ON \  
-D BUILD_EXAMPLES=OFF \  
-D BUILD_TESTS=OFF \  
-D BUILD_PERF_TESTS=OFF \  
-D WITH_QT=OFF \  
-D WITH_GTK=ON \  
-D WITH_OPENGL=ON \  
-D WITH_V4L=ON \  
-D WITH_FFMPEG=ON \  
-D WITH_XINE=ON \  
-D OPENCV_GENERATE_PKGCONFIG=ON ../  
$ time make -j4  
$ sudo make install  
  
$ cd ~  
$ sudo nano .bashrc  
- bashrc 파일 맨 아래 줄에 아래 내용 추가  
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib  
$ source ~/.bashrc  
$ sudo ldconfig
```

### 3.1. ML API Build for Ubuntu/ROS

ML API 예제 코드에 사용된 ROS 설치는 아래와 같습니다.

(1) Ubuntu 18.04 설치 후, Terminal을 실행하여 아래 command를 순서대로 입력하여 ROS를 설치합니다.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'  
$ sudo apt install curl  
$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc |  
  sudo apt-key add -  
$ sudo apt update  
$ sudo apt install ros-melodic-desktop
```

(2) ROS 설치 후, ROS 환경 설정 파일(setup.bash)을 자동으로 Ubuntu에 추가하기 위하여 아래와 같은 command를 추가로 입력합니다.

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc  
$ source ~/.bashrc
```

### 3.1. ML API Build for Ubuntu/ROS

(2-1) 환경 설정이 완료되면, 아래 command를 각각 새로운 Terminal에 입력하여 ROS가 정상적으로 설치가 됐는지 확인합니다.

[Terminal #1]

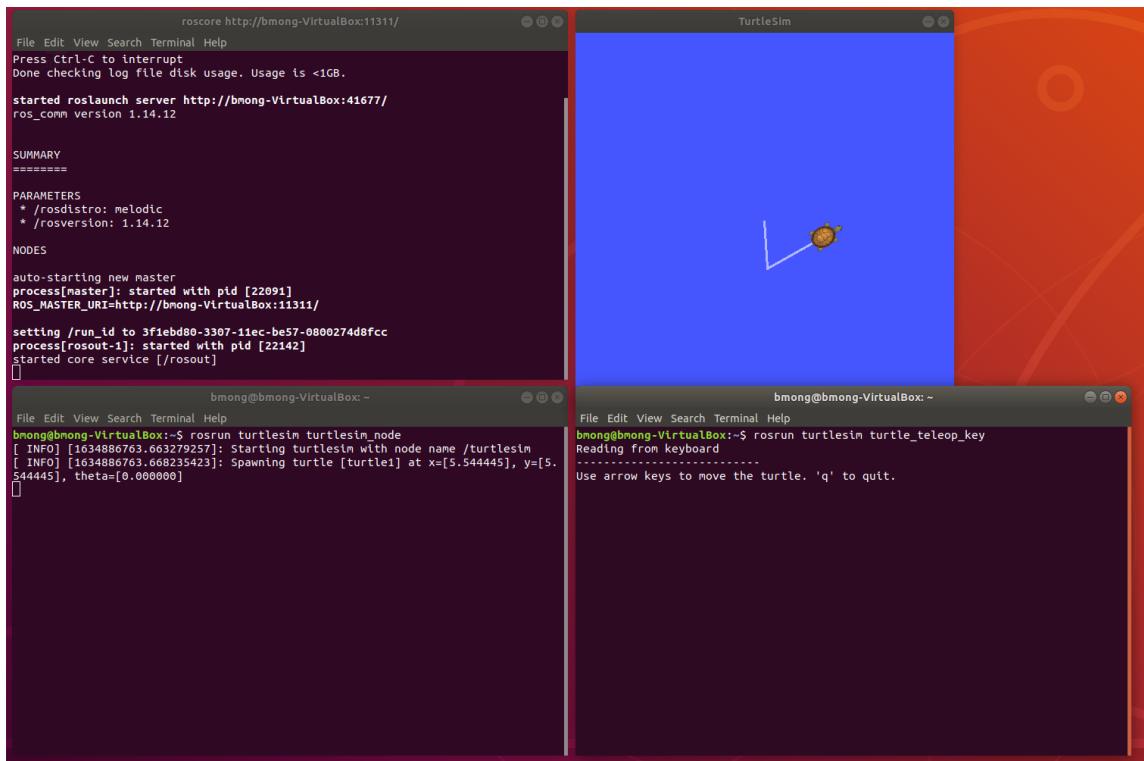
```
$ roscore
```

[Terminal #2]

```
$ rosrun turtlesim turtlesim_node
```

[Terminal #3]

```
$ rosrun turtlesim turtle_teleop_key
```



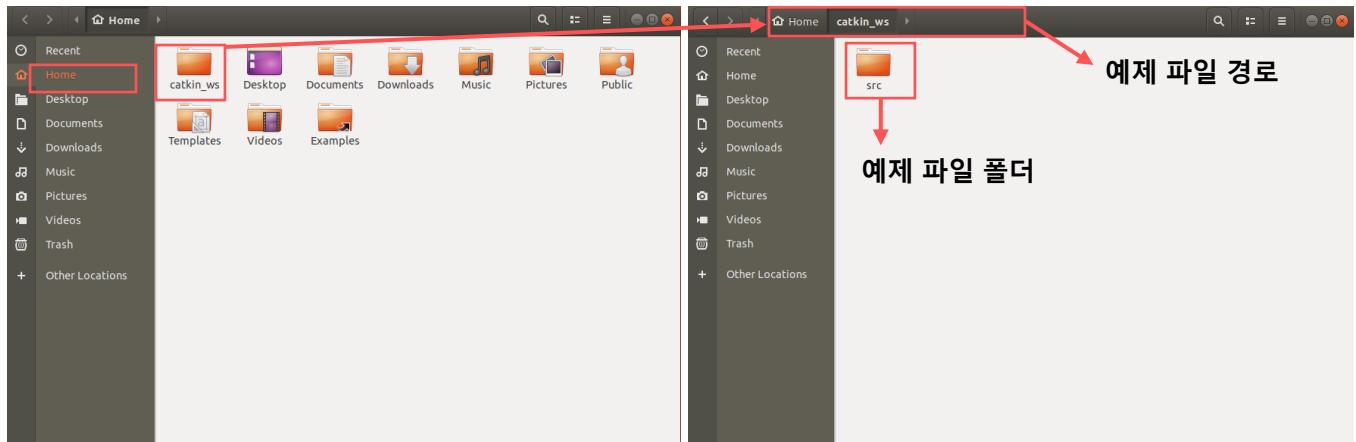
정상적인 ROS-melodic 설치 결과

### 3.1. ML API Build for Ubuntu/ROS

(3) 예제 코드에서 사용되는 Point Cloud 가시화를 위하여 pcl\_ros package를 설치합니다.

```
$ sudo apt install ros-melodic-pcl-ros
```

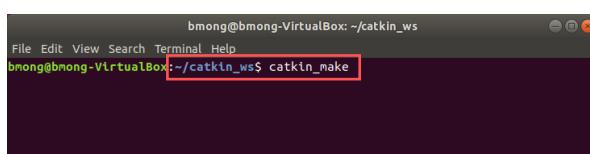
(4) 예제 코드를 아래 그림과 같이 Home/catkin\_ws 위치로 이동합니다.



#### 예제 코드 위치 변경

(5) 아래 그림과 같이 Terminal0| catkin\_ws 폴더 위치에서 “catkin\_make”를 입력하여 ML package를 생성합니다.

```
$ catkin_make
```



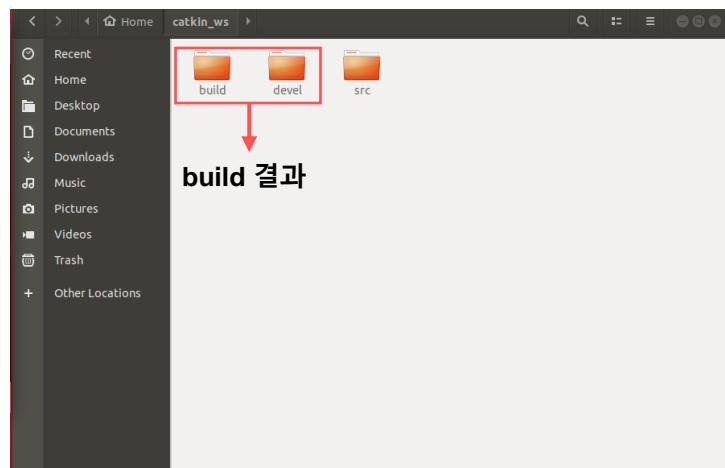
```
bmong@bmong-VirtualBox: ~/catkin_ws
File Edit View Search Terminal Help
bmong@bmong-VirtualBox:~/catkin_ws$ catkin_make
```

```
bmong@bmong-VirtualBox: ~/catkin_ws
File Edit View Search Terminal Help
#define DEG2RAD((x)*0.017453293)

In file included from /home/bmong/catkin_ws/src/ml/include/libssolab_ml.h:12:0,
                 from /home/bmong/catkin_ws/src/ml/src/main.cpp:6:
/home/bmong/catkin_ws/src/ml/include/ssolab_typedef.h:49:0: warning: "RAD2DEG" redefined
#define RAD2DEG          (180.0 / M_PI)

In file included from /usr/include/pcl-1.8/pcl/PCLHeader.h:11:0,
                 from /usr/include/pcl-1.8/pcl/point_cloud.h:48,
                 from /opt/ros/melodic/include/pcl_ros/include/point_cloud.h:5,
                 from /home/bmong/catkin_ws/src/ml/src/main.cpp:5:
/usr/include/pcl-1.8/pcl/macros.h:143:0: note: this is the location of the previous definition
#define RAD2DEG(x) ((x)*57.29578)

In file included from /home/bmong/catkin_ws/src/ml/src/main.cpp:5:0:
/opt/ros/melodic/include/pcl_ros/include/point_cloud.h:303:27: warning: variable template only available with -std=c++14 or -std=gnu++14
constexpr static bool pcl_uses_boost = true;
[100%] LINKING CXX executable /home/bmong/catkin_ws/devel/lib/ml/ml
[100%] Built target ml
bmong@bmong-VirtualBox:~/catkin_ws$
```



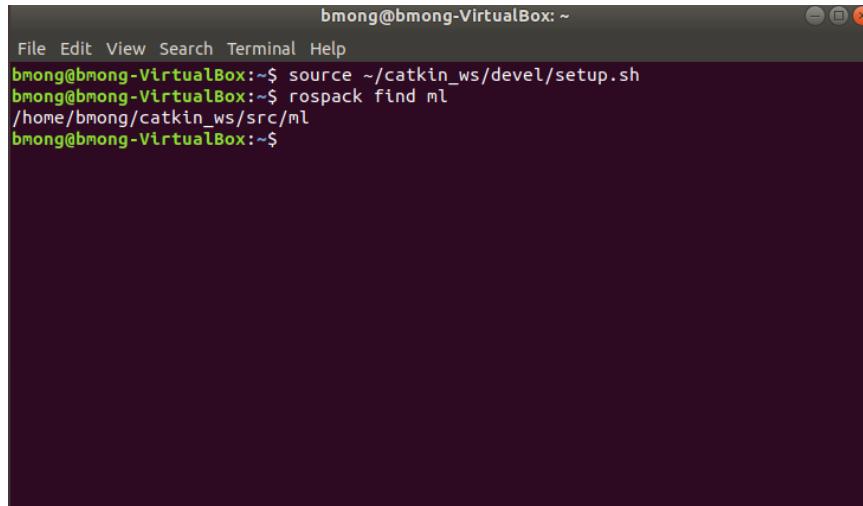
예제 코드 build 성공

catkin\_make를 활용한 예제 코드 build

### 3.1. ML API Build for Ubuntu/ROS

(6) build를 통해 생성된 ML package를 ROS 환경에 추가하기 위하여 아래 command를 입력하여 ROS 환경에 ML package 추가 및 확인할 수 있습니다.

```
$ source ~/catkin_ws/devel/setup.sh  
$ rospack find ml
```

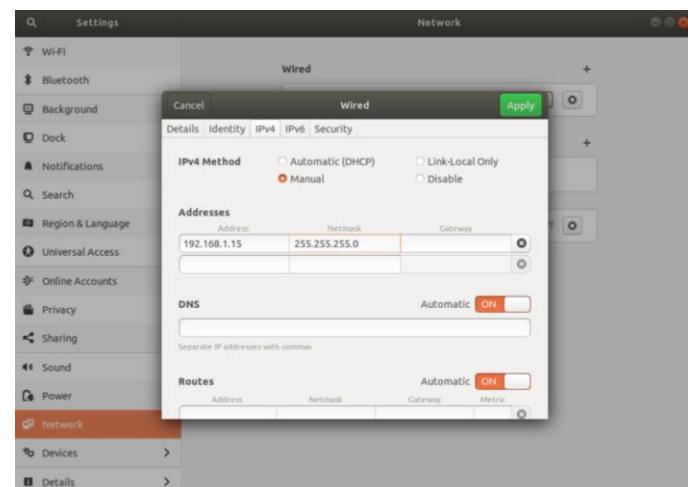
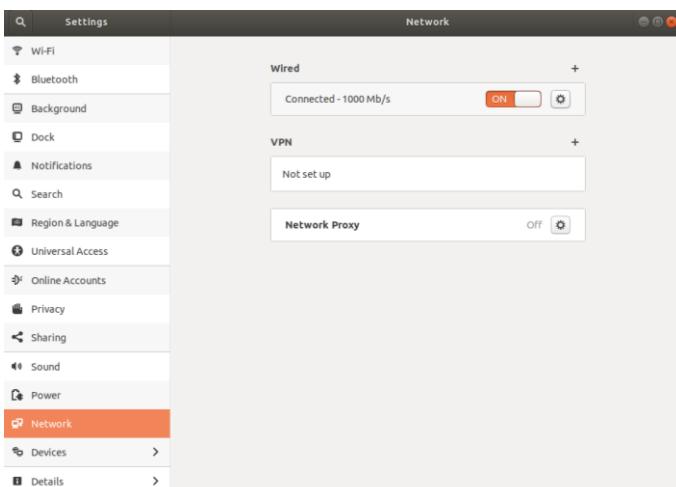


```
bmong@bmong-VirtualBox: ~  
File Edit View Search Terminal Help  
bmong@bmong-VirtualBox:~$ source ~/catkin_ws/devel/setup.sh  
bmong@bmong-VirtualBox:~$ rospack find ml  
/home/bmong/catkin_ws/src/ml  
bmong@bmong-VirtualBox:~$
```

#### ML package를 ROS 환경에 추가 및 결과 확인

(7) ML device와 pc와의 연결을 위하여 setting창의 network 항목에서 아래 설정과 같이 network의 IPv4를 setting합니다.

- IPv4 – Manual
- Address : 192.168.1.15
- NetMask : 255.255.255.0

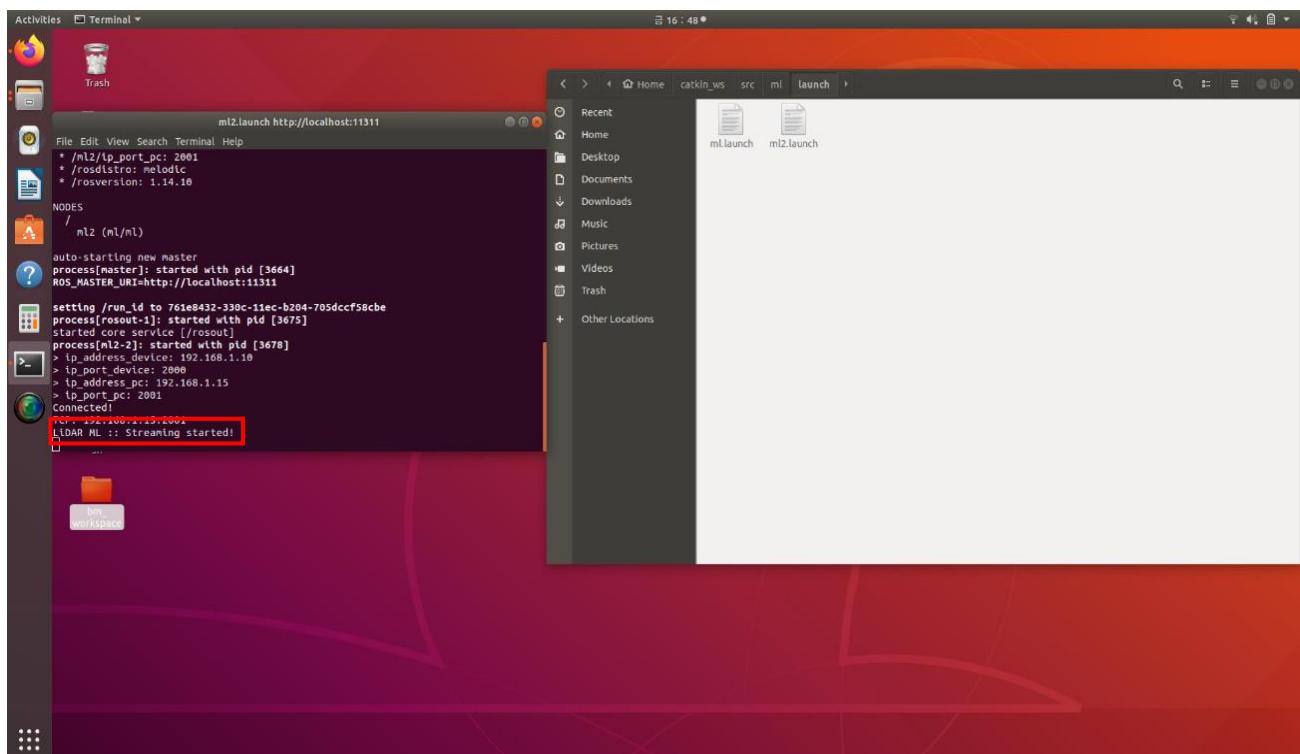


#### IPv4 Network Setting

### 3.1. ML API Build for Ubuntu/ROS

(8) Network 설정 이후, ML device에 맞게 아래 command를 입력하여 ML device와 연결합니다. 정상적으로 ML Device와 연결이 되면 아래 그림과 같이 Terminal 창에 Lidar ML :: Streaming started! 출력을 확인할 수 있습니다.

```
$ cd ~/catkin_ws/src/ml/launch  
$ roslaunch ml.launch
```



Ubuntu 환경에서 ROS를 통해 ML Device 연결

## 3.2. Example Code for Ubuntu/ROS

### 3.2.1. Connect ML Device

```
1. bool success;
2. std::shared_ptr<SOSLAB::LidarML> lidar_ml(new SOSLAB::LidarML);

3. SOSLAB::ip_settings_t ip_settings_device;
4. SOSLAB::ip_settings_t ip_settings_pc;
5. ip_settings_pc.ip_address = "0.0.0.0";
6. ip_settings_pc.port_number = 0;
7. ip_settings_device.ip_address = "192.168.1.10";
8. ip_settings_device.port_number = 2000;
9. success = lidar_ml->connect(ip_settings_device, ip_settings_pc);
10. if (!success) {
11.     std::cerr << "LiDAR ML :: connection failed." << std::endl;
12.     return 0;
13. }

14. success = lidar_ml->tcp_device_run();
15. if (!success) {
16.     std::cerr << "LiDAR ML :: start failed." << std::endl;
17.     return 0;
18. }
19. std::cout << "LiDAR ML :: Streaming started!" << std::endl;
```

ML device와 PC의 통신을 위한 ML API 예제 코드입니다. pc IP 주소와 port 번호는 예제 코드와 동일하게, ML device의 IP 주소와 Port 번호는 제공된 정보와 같이 사용하시면 됩니다.

Line	Description
2	Local(PC)와 ML Device와의 통신을 위한 SOSLAB::LidarML 변수를 lidar_ml로서 선언합니다.
3-8	SOSLAB::ip_settings_t 변수를 Local(PC)과 ML device에 대하여 두 개를 선언합니다. 예제 코드에서는 ip_setting_device, ip_setting_pc라는 이름으로 선언했습니다. 선언된 변수에 IP 주소, Port 번호와 ML device의 IP 주소, Port 번호를 구조체의 ip_address, port_number 변수 값에 대입합니다.
9-13	정의된 Local, ML ip_settings_t 변수를 LidarML Class의 connect 함수의 입력 변수로 하여 ML device와 연결합니다. 입력 변수 순서는 ML device ip setting, local(pc) ip setting 순입니다. 이후, connect 함수의 반환 값으로 연결 성공 여부를 bool 변수형으로 얻습니다.
14-19	ML device와의 연결이 완료되면 LidarML Class의 tcp_device_run()을 통해 scanning 을 시작합니다. 함수의 반환값으로 scanning 시작 여부를 bool 변수형으로 얻습니다.

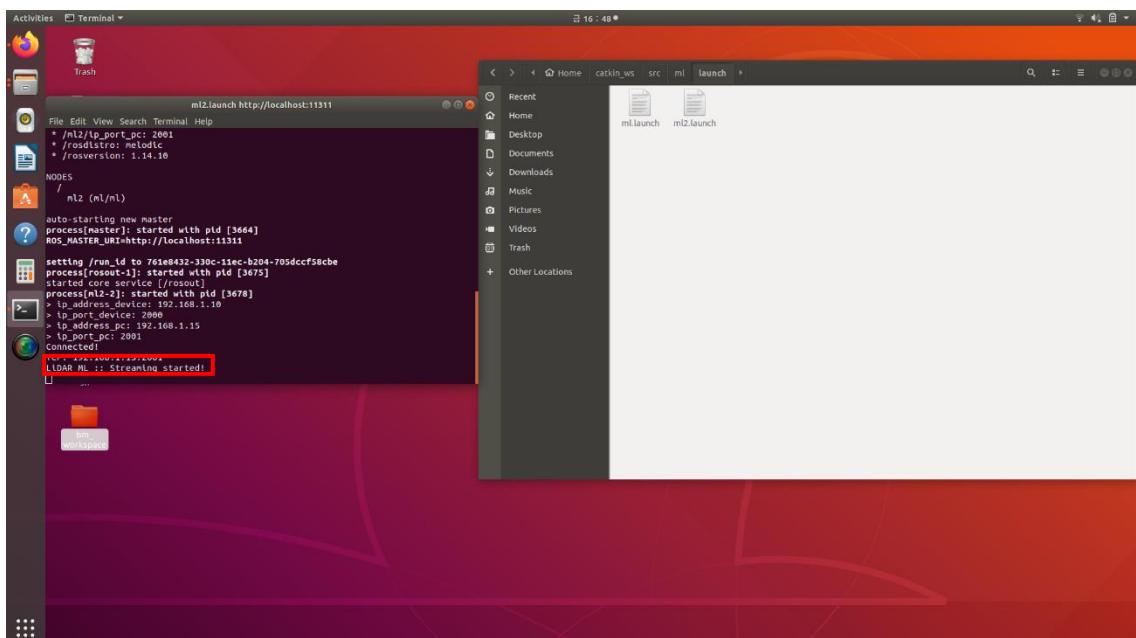
## 3.2. Example Code for Ubuntu/ROS

### 3.2.2 Connect ML Device using ROS

```
$ cd ~/catkin_ws/src
$ catkin_make
$ source ~/catkin_ws/devel/setup.sh
$ cd ~/catkin_ws/src/ml/launch
[ML0 Device]
$ roslaunch ml.launch
[ML2 Device]
$ roslaunch ml2.launch
```

ROS를 이용하여 ML device와 PC의 통신을 위한 예제입니다. launch 파일에는 실행할 package(ml)과 각 device에 맞는 ip 주소 정보가 포함되어 있습니다.

Line	Description
1-2	제공된 src 폴더로 Terminal 위치를 이동한 후, catkin_make를 통하여 build합니다.
3	build된 ML package를 ROS 환경에 추가합니다.
4-5	Terminal 위치를 launch 파일이 있는 폴더로 이동한 후, 연결할 ML device에 맞는 launch파일을 실행합니다.



Ubuntu 환경에서 ROS를 통한 ML Device 연결

## 3.2. Example Code for Ubuntu/ROS

### 3.2.3 Setting Enable/Parameter of Data Processing Modules

```
1. lidar_ml->set_zigzag_filling_enable(true);  
2. lidar_ml->set_deflaring_enable(false);  
3. lidar_ml->set_real_ambient_enable(false);  
4. lidar_ml->set_line_filling_enable(true);  
5. lidar_ml->set_tr_retro_threshold_ml0(110);  
6. lidar_ml->set_tr_retro_threshold_ml2(700);
```

ML API에는 Raw data 성능 향상을 위한 data processing module이 기본적으로 포함되어 있습니다. 해당 기능을 사용하지 않거나, parameter를 변경하기 위해서 사용되는 함수에 대한 예제입니다. setting을 위한 함수는 scanning 시작하는 함수(tcp\_device\_run(), start()) 전에 호출하시면 됩니다. 각 함수의 기본값은 appendix를 참조하시면 됩니다.

Line	Description
1	ML device에서 Data Processing Module의 zigzag filling 기능을 ON(true)/OFF(false) 하는 set_zigzag_filling_enable 함수를 호출합니다
2	ML device에서 Data Processing Module의 deflaring 기능을 ON(true)/OFF(false) 하는 set_deflaring_enable 함수를 호출합니다.
3	ML device에서 Data Processing Module의 real ambient 기능을 ON(true)/OFF(false) 하는 set_real_ambient_enable 함수를 호출합니다.
4	ML2 device에서 Data Processing Module의 line filling 기능을 ON(true)/OFF(false) 하는 set_line_filling_enable 함수를 호출합니다.
5	deflaring 기능의 parameter 값을 변경하는 함수를 호출합니다. ML0 device에서는 set_tr_retro_threshold_ml0 함수를 호출합니다. 변경할 parameter 값을 입력으로 받습니다.
6	deflaring 기능의 parameter 값을 변경하는 함수를 호출합니다. ML2 device에서는 set_tr_retro_threshold_ml2 함수를 호출합니다. 변경할 parameter 값을 입력으로 받습니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.4. Get Raw Data

```
1. while (ros::ok()) {  
2.     SOSLAB::LidarMI::scene_t scene;  
3.     if (lidar_mi->get_scene(scene)) {  
4.         std::vector<uint16_t> ambient = scene.grey_image;  
5.         std::vector<uint16_t> intensity = scene.intensity_image;  
6.         std::vector<uint32_t> depth = scene.depth_image;  
7.         std::vector<SOSLAB::point_t> pointcloud = scene.pointcloud;  
8.     }  
9. }
```

ML device으로부터 raw data를 획득하는 방법에 대한 예제 코드입니다. raw data는 `scene_t`로 획득되며, 총 4개의 1차원 데이터 배열(Ambient, intensity, depth, pointcloud)로 구성되어 있습니다. 자세한 `scene_t` 구조체 구조에 대한 설명은 appendix를 참조하시면 됩니다.

Line	Description
2	Raw data를 저장할 <code>SOSLAB::LidarMI::scene_t</code> 변수를 <code>scene</code> 이름으로 선언합니다.
3	<code>scene_t</code> 변수를 입력으로 받는 <code>SOSLAB::LidarMI</code> class의 <code>get_scene</code> 함수를 통해 <code>scene</code> 변수에 raw data를 획득합니다.
4-6	raw data를 저장한 <code>scene_t</code> <code>scene</code> 구조체로부터 <code>ambient</code> , <code>intensity</code> , <code>depth</code> , <code>pointcloud</code> 데이터를 획득합니다. 각 데이터의 구조체 변수명과 변수형은 appendix에 참조하시면 됩니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.5. Data Recording example

```
1. std::string save_directory = "./log/"
2. #ifdef _WIN32
3.     if (_access(save_directory.c_str(), 0)) {
4.         if (_mkdir(save_directory.c_str())) return false;
5.     }
6. #endif // _WIN32
7. #ifdef __linux__
8.     mkdir(save_directory.c_str(), S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
9. #endif
10. Bool retval = lidar_ml->start_recording(save_directory);
11. int frame_number = 0;
12. int logging_data_size = 10;
13. while (frame_number < logging_data_size) {
14.     SOSLAB::LidarML::scene_t scene;
15.     if (lidar_ml->get_scene(scene)) {
16.         std::size_t height = scene.rows;
17.         std::size_t width = scene.cols;
18.         frame_number++;
19.     }
20. }
21. lidar_ml->stop_recording();
```

Line	Description
1-9	녹화 파일의 저장 위치(save_directory)를 설정하고 해당 폴더가 없으면 생성합니다.
10	SOSLAB::LidarML class의 start_recording(save_directory) 함수를 호출하여 녹화를 시작합니다.
11-20	scene_t 변수를 입력으로 받는 SOSLAB::LidarML class의 get_scene 함수를 통해서 scene 변수에 raw data를 획득합니다.
21	SOSLAB::LidarML class의 stop_recording() 함수를 호출하여 녹화를 중지합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.6. Multi-LiDAR Example

다중 ML 가시화 예제 코드는 다중 ML의 네트워크 케이블을 공유기에 연결한 뒤, 공유기와 PC 사이의 Lan 케이블을 연결하여 다중 ML과 PC 연결을 합니다.

다중 ML 가시화를 위해서 ML의 Device setting의 ML\_IP와 PC\_PORT 그리고 MACADDR 항목 값을 ML마다 다르게 변경합니다. Device setting 변경 방법은 1.3.1 항목에 자세히 작성되어 있습니다.



2개의 ML 연결을 위한 공유기 연결 예시

The image shows two windows of the Windows Notepad application. The left window is titled "first\_ml\_setting.txt" and the right window is titled "second\_ml\_setting.txt". Both windows display text configurations for ML devices.

**Content of first\_ml\_setting.txt:**

```
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
ML_IP,UINT8,4,192,168,1,10  
PC_IP,UINT8,4,192,168,1,15  
PC_PORT,UINT16,1,2000  
MACADDR,UINT8,6,0,128,225,0,0,1
```

**Content of second\_ml\_setting.txt:**

```
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
ML_IP,UINT8,4,192,168,1,11  
PC_IP,UINT8,4,192,168,1,15  
PC_PORT,UINT16,1,2001  
MACADDR,UINT8,6,0,128,225,0,0,2
```

1번 ML Device Setting      2번 ML Device Setting

2개의 ML 연결을 위한 device setting 예시

## 3.2. Example Code for Ubuntu/ROS

### 3.2.6. Multi-LiDAR Example

```
1.  bool success = true;
2.  int num_lidar = 1;
3.  std::vector<std::shared_ptr<SOSLAB::LidarML>> lidar_ml_list(new
SOSLAB::LidarML);
4.  int standard_port = 2000;
5.  for (int i = 0; i < num_lidar; i++) {
6.      SOSLAB::ip_settings_t ip_settings_pc;
7.      SOSLAB::ip_settings_t ip_settings_device;
8.      ip_settings_pc.ip_address = "192.168.1.15";
9.      ip_settings_pc.port_number = standard_port + i;
10.     ip_settings_device.ip_address = "192.168.1.1" + to_string(i);
11.     ip_settings_device.port_number = standard_port;
12.     success = lidar_ml_list[i]->connect(ip_settings_device,
ip_settings_pc);
13.     if (!success) {
14.         std::cerr << "LiDAR ML #" + std::to_string(i) + " :: connection failed" << std::endl;
15.     }
16.     for (int i = 0; i < num_lidar; i++) {
17.         success &= lidar_ml_list[i]->tcp_device_run();
18.         if (!success) {
19.             std::cerr << "LiDAR ML #" + std::to_string(i) + " :: start failed" << std::endl;
20.             return 0;
21.         }
22.     }
23.  }
```

Line	Description
2	Local(PC)와 ML Device와의 통신을 위한 SOSLAB::LidarML 객체를 연결할 ML 개수 (num_lidar)에 따라서 초기화 합니다. (default num_lidar = 1)
4-15	예제 코드에서는 초기 값 Device IP(192.168.1.10), PC port(2000)에서 ML 개수에 따라 Device IP, PC port의 마지막 값이 1씩 증가시켜 PC와 연결합니다. 연결 방법은 2.3.1과 동일합니다. - Example : 2개의 ML 연결 #1 ML Device port = "192.168.1.10", PC port = "2000" #2 ML Device port = "192.168.1.11", PC port = "2001"
16-18	모든 ML과 PC를 연결한 뒤, ML 개수에 맞게 LidarML Class의 tcp_device_run()을 호출하여 scanning을 시작합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.6. Multi-LiDAR Example

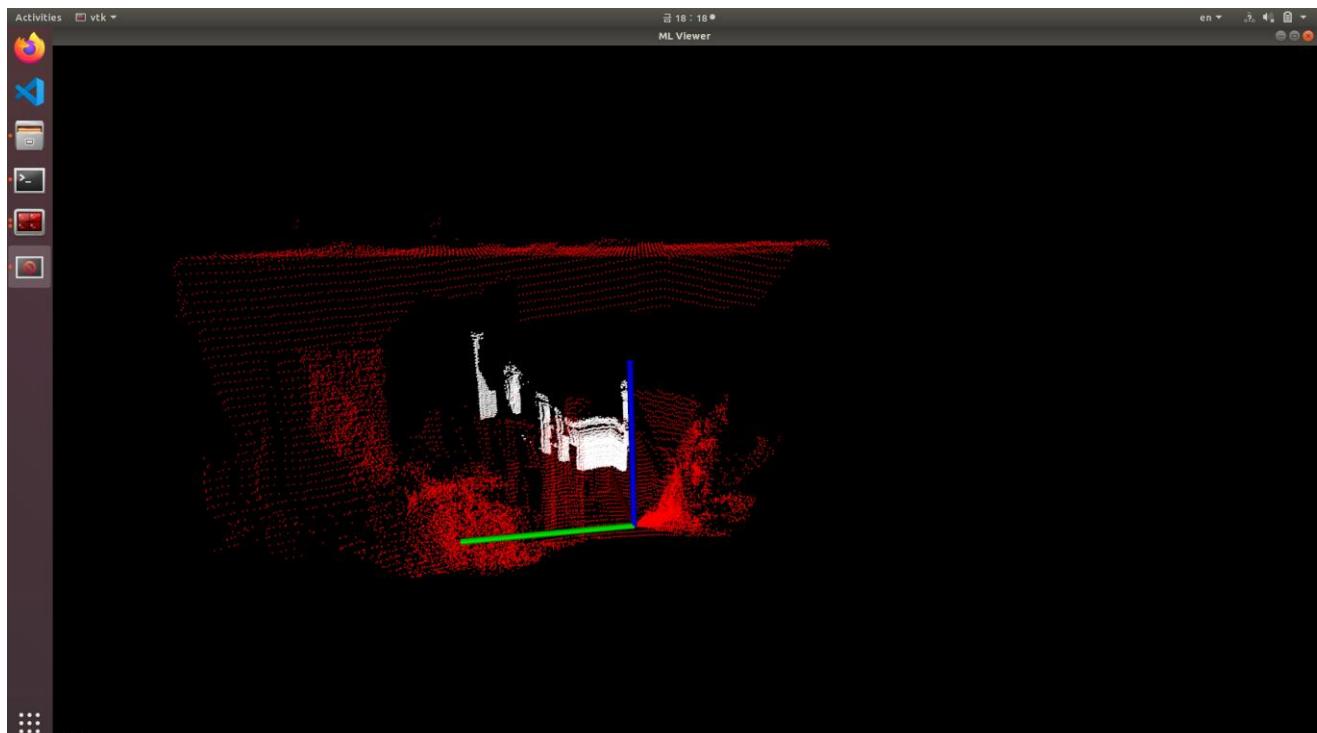
```
1.     int points_size = 0;
2.     while (!pclviz_->wasStopped()) {
3.         SOSLAB::LidarML::scene_t scene;
4.         for (int i = 0; i < num_lidar; i++) {
5.             if (lidar_ml_list[i]->get_scene(scene)) {
6.                 std::size_t height = scene.rows;
7.                 std::size_t width = scene.cols;
8.                 if (initialize) size_list[i+1] = height * width;
9.                 if (!scene.pointcloud.empty() && streaming_start) {
10.                     for (int j = 0; j < size_list[i+1]; j++) {
11.                         PointT point;
12.                         point.x = scene.pointcloud[j].xyz.x / 1000.0f;
13.                         point.y = scene.pointcloud[j].xyz.y / 1000.0f;
14.                         point.z = scene.pointcloud[j].xyz.z / 1000.0f;
15.                         point.r = uint8_t(color_vector[i][0]);
16.                         point.g = uint8_t(color_vector[i][1]);
17.                         point.b = uint8_t(color_vector[i][2]);
18.                         cloud_->points[j+ clouds_start_points[i]] = point;
19.                     }
20.                 }
21.             }
22.         }
23.         if (!pclviz_->updatePointCloud(cloud_, "cloud")) {
24.             pclviz_->addPointCloud(cloud_, "cloud");
25.             pclviz_->spinOnce(1);
26.         }
}
```

Line	Description
2-3	PCL 가시화 창이 끝날 때까지 ML 데이터를 획득하는 Loop를 시작합니다.
4-5	SOSLAB::LidarML 객체를 저장하는 lidar_ml_list의 각 요소에 접근하여 get_scene 함수를 호출하여 각 LiDAR의 데이터를 scene 변수에 저장합니다.
6-14	2.2.5 항목과 동일한 방법으로 scene 변수의 Point 정보 획득하여 PointT 변수의 값으로 설정합니다.
15-17	연결된 LiDAR 순서에 따라서 다른 색상을 설정합니다. 1번 : 흰색, 2번 : 빨강, 3번 : 초록, 4번 : 파랑
18	PointT 변수에 저장된 Point를 가시화를 위한 cloud_ 변수에 저장합니다.
23-25	Loop를 돌면서 저장된 모든 LiDAR의 Point Cloud 정보를 저장하는 cloud_ 변수를 PCL 가시화 객체에 등록합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.6. Multi-LiDAR Example

Multi-LiDAR 예제의 결과는 아래 그림과 같이 다중 LiDAR의 점군을 PCL VTK 모듈을 사용하여 가시화합니다. 아래 그림은 2개의 ML을 공유기를 통해 PC에 연결한 Point Cloud를 1번 LiDAR는 흰색, 2번 LiDAR는 빨간색으로 가시화됩니다.



2개의 ML 연결을 위한 device setting 예시

## 3.2. Example Code for Ubuntu/ROS

### 3.2.5. Publish Depth, Intensity, Ambient Image for Rviz

```
1. ros::NodeHandle nh("~");
2. image_transport::ImageTransport it(nh);
3. image_transport::Publisher pub_grey = it.advertise("grey", 1);
4. sensor_msg::ImagePtr msg_grey;

5. cv::Mat grey_image
6. grey_image(scene.ros, scene.cols, CV_16UC1, scene.grey_image.data());
7. cv::normalize(grey_image, grey_image, 0, 255, cv::NORM_MINMAX);
8. msg_grey = cv_bridge::CvImage(std_msgs::Header(), "mono16",
    grey_image).toImageMsg();
9. pub_grey.publish(msg_grey);
```

Rviz를 통한 가시화를 위하여 ML devic으로부터 획득한 Depth, Intensity, Ambient 데이터를 이미지 변환 및 publisher에 대한 예제 코드입니다. Rviz는 ROS에서 사용하는 가시화 Tool로서 Publisher Node라는 객체로부터 Message 형태로 저장된 정보(이미지, Point Cloud)들을 읽어 가시화할 수 있습니다. Publisher Node는 사용자가 사전에 이름(topic)을 지정할 수 있습니다. **Image** 가시화를 위하여 **OpenCV Library**를 사용합니다.

Line	Description
1-4	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 <b>image transport</b> , <b>ImagePtr</b> 등의 객체를 생성합니다. publisher는 grey라는 topic으로 Ambient 데이터를 제공합니다.
5-6	<b>scene_t</b> 구조체의 Ambient 데이터를 이미지로 가시화하기 위해 opencv의 <b>cv::Mat</b> 형식으로 변환하는 과정입니다. <b>cv::Mat</b> 초기화 입력 값으로 Raw data의 height( <b>scene.rows</b> ), width( <b>scene.cols</b> ) 그리고 Ambient 데이터 type과의 일치( <b>CV_16UC1</b> )시키고 Raw data 값을 4번째 인수로 입력합니다.
7	이미지 가시화를 위하여 최소, 최대값을 기준으로 정규화 합니다.
8	Opencv library의 <b>cv::Mat</b> 객체를 Publisher Node의 Message 형태로 저장하기 위하여 <b>ImagePtr</b> 로 변환하는 과정입니다.
9	topic이 “grey”로 지정된 <b>Publisher</b> 객체의 <b>publish</b> 함수에 <b>ImagePtr</b> 변수를 입력 인수로 사용하여 <b>publish</b> 를 완료합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.5. Publish Depth, Intensity, Ambient Image for Rviz

```
1. ros::NodeHandle nh("~");
2. image_transport::ImageTransport it(nh);
3. image_transport::Publisher pub_intensity = it.advertise("intensity",
   1);
4. sensor_msg::ImagePtr msg_intensity;

5. cv::Mat intensity_image
6. intensity_image(scene.ros, scene.cols, CV_16UC1, scene.
   intensity_image.data());
7. cv::normalize(intensity_image, intensity_image, 0, 255,
   cv::NORM_MINMAX);
8. msg_intensity = cv_bridge::CvImage(std_msgs::Header(), "mono16",
   intensity_image).toImageMsg();
9. pub_intensity.publish(msg_intensity);
```

Intensity 데이터는 Ambient와 변수 형태가 동일(`std::vector<uint16_t>`)하기 때문에 Ambient와 publish과정이 동일합니다.

Line	Description
1-4	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 <code>image transport</code> , <code>ImagePtr</code> 등의 객체를 생성합니다. <code>publisher</code> 는 <code>intensity</code> 라는 topic으로 Intensity 데이터를 제공합니다.
5-6	<code>scene_t</code> 구조체의 <code>intensity</code> 데이터를 이미지로 가시화하기 위해 opencv의 <code>cv::Mat</code> 형식으로 변환하는 과정입니다. <code>cv::Mat</code> 초기화 입력 값으로 Raw data의 <code>height</code> ( <code>scene.rows</code> ), <code>width</code> ( <code>scene.cols</code> ) 그리고 Intensity 데이터 type과의 일치( <code>CV_16UC1</code> )시키고 Raw data 값을 4번째 인수로 입력합니다.
7	이미지 가시화를 위하여 최소, 최대값을 기준으로 정규화 합니다.
8	Opencv library의 <code>cv::Mat</code> 객체를 Publisher Node의 Message 형태로 저장하기 위하여 <code>ImagePtr</code> 로 변환하는 과정입니다.
9	topic이 “intensity”로 지정된 <code>Publisher</code> 객체의 <code>publish</code> 함수에 <code>ImagePtr</code> 변수를 입력 인수로 사용하여 <code>publish</code> 를 완료합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.5. Publish Depth, Intensity, Ambient Image for Rviz

```
1. ros::NodeHandle nh("~");
2. image_transport::ImageTransport it(nh);
3. image_transport::Publisher pub_depth = it.advertise("depth", 1);
4. sensor_msg::ImagePtr msg_depth;

5. cv::Mat depth_image
6. depth_image(scene.ros, scene.cols, CV_16UC1, scene.
   depth_image.data());
7. depth_image.convertTo(depth_image, CV_16UC1, 1.0/ 1.0);
8. msg_depth = cv_bridge::CvImage(std_msgs::Header(), "mono16",
   depth_image).toImageMsg();
9. pub_intneisyt.publish(msg_depth);
```

depth 데이터는 Ambient/Intensity와 변수 형태(`std::vector<uint32_t>`)가 다르기 때문에 이미지 변환 과정을 다르게 처리합니다.

Line	Description
1-4	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 <code>image transport</code> , <code>ImagePtr</code> 등의 객체를 생성합니다. publisher는 depth라는 topic으로 Depth 데이터를 제공합니다.
5-6	<code>scene_t</code> 구조체의 depth 데이터를 이미지로 가시화하기 위해 opencv의 <code>cv::Mat</code> 형식으로 변환하는 과정입니다. <code>cv::Mat</code> 초기화 입력 값으로 Raw data의 height( <code>scene.rows</code> ), width( <code>scene.cols</code> ) 그리고 depth 데이터 type과의 일치( <code>CV_32SC1</code> )시키고 Raw data 값을 4번째 인수로 입력합니다.
7	이미지 가시화를 <code>CV_16UC1(uint16_t)</code> 변수 형태로 변환 합니다.
8	Opencv library의 <code>cv::Mat</code> 객체를 Publisher Node의 Message 형태로 저장하기 위하여 <code>ImagePtr</code> 로 변환하는 과정입니다.
9	topic이 "depth"로 지정된 <code>Publisher</code> 객체의 <code>publish</code> 함수에 <code>ImagePtr</code> 변수를 입력 인수로 사용하여 <code>publish</code> 를 완료합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.6. Publish Point Cloud for Rviz

```
1. typedef pcl::PointCloud<pcl::PointXYZRGB> PointCloud_T
2. static const uint32_t DEFAULT_PUBLISHER_QUEUE_SIZE = 10
3. static const char* DEFAULT_FRAME_ID = "map"

4. ros::NodeHandle nh("~");
5. ros::Publisher pub_lidar = nh.advertise<PointCloud_T>("pointcloud",
   uint32_t(DEFAULT_PUBLISHER_QUEUE_SIZE));

6. PointCloud_T::Ptr msg_pointcloud(new PointCloud_T);
7. msg_pointcloud->header.frame_id = DEFAULT_FRAME_ID
8. msg_pointcloud->width = scene.cols;
9. msg_pointcloud->height = scene.rows;
10. msg_pointcloud->points.resize(scene.pointcloud.size())
11. for (int i = 0; i < scene.pointcloud.size(); i++) {
12.     msg_pointcloud->points[i].x = scene.pointcloud[i].xyz.x;
13.     msg_pointcloud->points[i].y = scene.pointcloud[i].xyz.y;
14.     msg_pointcloud->points[i].z = scene.pointcloud[i].xyz.z;
15. }

16. pcl_conversions::toPCL(ros::Time::now(), msg_pointcloud-
   >header.stamp);
17. pub_lidar.publish(msg_pointcloud);
```

Rviz를 통해 `scene_t`의 pointcloud 데이터를 가시화하기 위한 publisher에 대한 예제 코드입니다. Rviz는 ROS에서 사용하는 가시화 Tool로서 Publisher Node라는 객체로부터 Message 형태로 저장된 정보(이미지, Point Cloud)들을 읽어 가시화할 수 있습니다. Publisher Node는 사용자가 사전에 이름(topic)을 지정할 수 있습니다. **Point Cloud 가시화를 위하여 PCL Library를 사용합니다.**

Line	Description
4-5	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 <code>image transport</code> , <code>sensor_msg</code> 객체를 생성합니다. publisher는 pointcloud라는 topic으로 Point Cloud 데이터를 제공합니다.
6-10	Message 변수를 정의하여 데이터 크기, 이름을 초기화합니다.
11-15	<code>scene_t</code> 의 pointcloud 데이터를 <code>msg_pointcloud</code> 변수로 복사합니다.
16-17	Point Cloud가 scanning된 timestamp를 저장한 뒤, topic이 "pointcloud"로 지정된 Publisher 객체의 publish 함수에 <code>PointCloud_T::Ptr</code> 변수를 입력 인수로 사용하여 publish를 완료합니다.

## 3.2. Example Code for Ubuntu/ROS

### 3.2.7. Visualization with Rviz

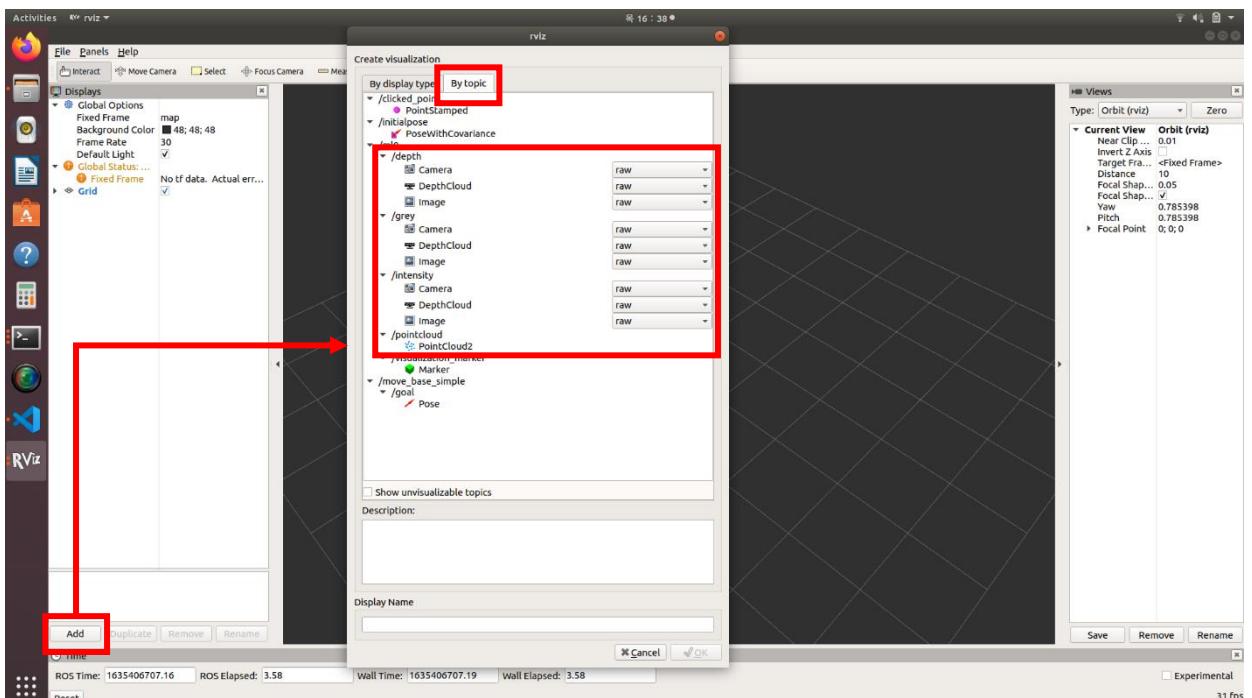
```
[Terminal #1]
$ cd ~/catkin_ws/src
$ catkin_make
$ source ~/catkin_ws/devel/setup.sh
$ cd ~/catkin_ws/src/ml/launch
[ML0 Device]
$ roslaunch ml.launch
[ML2 Device]
$ roslaunch ml2.launch

[Terminal #2]
$ rviz
```

3.2.5-3.2.6 예제 코드에 publish한 raw data를 Rviz를 통해 가시화하는 예제입니다.

Line	Description
1-6	3.2.2 예제와 동일하게 ML Device와 연결하는 command입니다.
7	새로운 Terminal에서 rviz command를 입력하여 Rviz 프로그램을 실행합니다.

Rviz command를 입력한 뒤, 아래 그림과 같이 왼쪽 하단의 Add 버튼을 클릭하면 아래 그림과 같은 창을 확인할 수 있습니다. 해당 창의 상단 “By topic” 메뉴를 클릭하면 publish되고 있는 전체 topic을 확인할 수 있습니다.

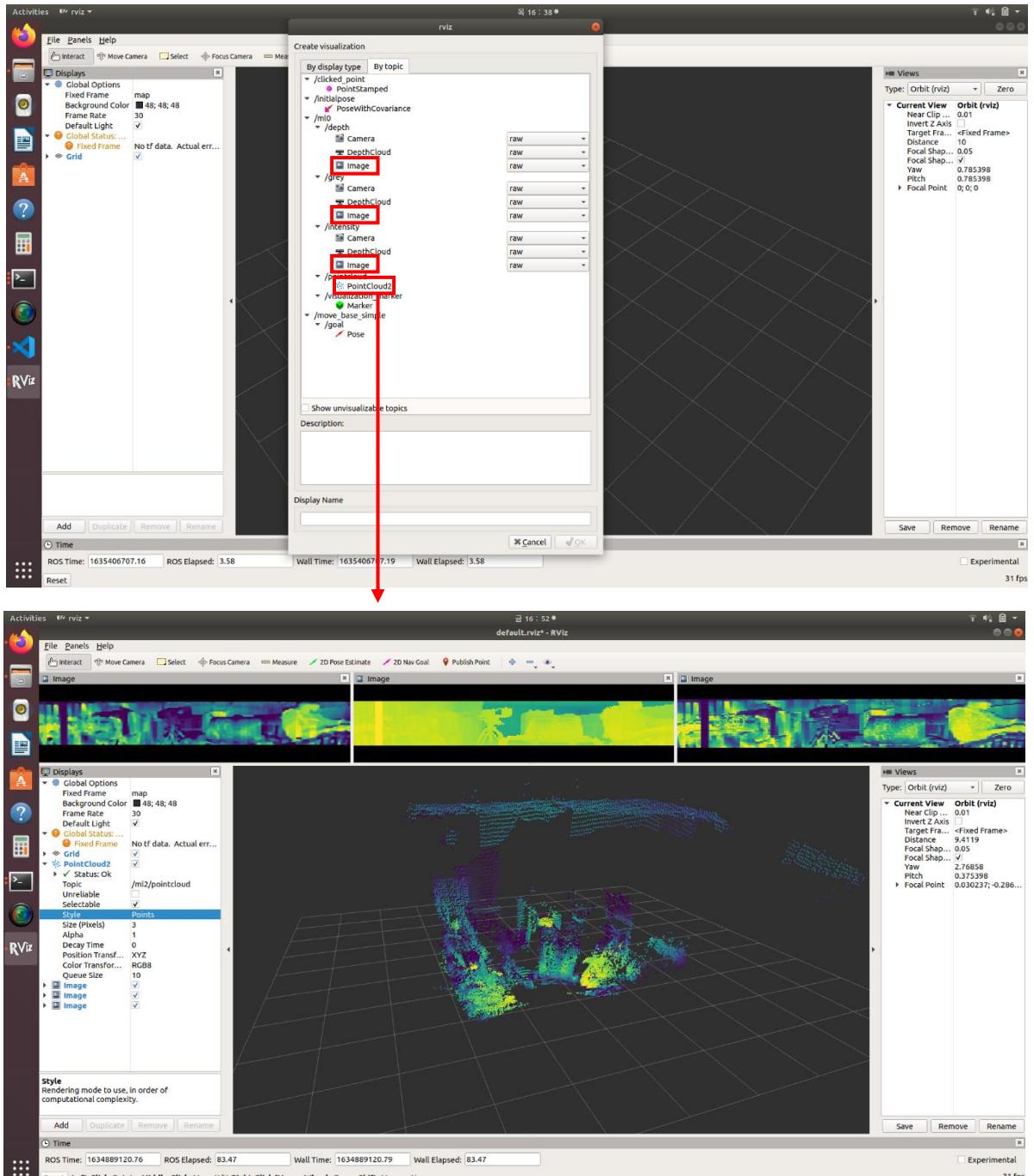


Rviz 프로그램 실행 화면

## 3.2. Example Code for Ubuntu/ROS

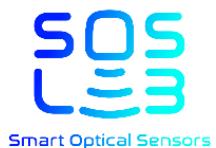
### 3.2.7. Visualization with Rviz

Depth, Image, Ambient 데이터를 가시화하기 위해서 각 topic 메뉴의 “Image”를 선택한 뒤 Ok 버튼을 클릭하시면 해당 데이터의 이미지를 확인 가능하며, Point Cloud 데이터는 “/pointcloud” topic의 “PointCloud2”를 선택한 뒤 Ok 버튼을 클릭하시면 가시화할 수 있습니다. 아래 그림은 Depth, Image, Ambient, Point Cloud 데이터를 가시화한 결과입니다.



Rviz를 사용한 Raw Data(Depth, Intensity, Ambient, Point Cloud) 가시화

# Appendix



## A.1. TCP Protocol Packet Structure

TCP 통신에서는 대화식 문자열 구조로 구성되어 있습니다. PC쪽에서 설정된 명령어를 전송하면 ML device에서 응답을 하며 ML device에서 추가적인 설정이 필요할 경우, 질문 형태의 응답이 PC로 전송됩니다. 패킷 구조는 아래와 같습니다.

**TCP Protocol 패킷 구조**

Byte	Name	Sub Name	Type	Range	Description
0	DL	DL[0]	Number	0~65531	DATA영역의 길이 (DL의 LSB)
1		DL[1]			DATA영역의 길이 (DL의 MSB)
3	CMD	MAIN CMD	Character	ASCII코드	<p>[MAIN CMD/SUB CMD]  <b>* PC → Device</b></p> <ul style="list-style-type: none"> <li>- S/N: System Mode Change to Normal, (DL == 0)</li> <li>- S/P: System Mode Change to Production, (DL == 0)</li> <li>- S/R: System Mode Read, (DL==0)</li> <li>- R/W: Register Write</li> <li>- R/R: Register Read, (DL == 0)</li> <li>- M/U: Memory Upload, (DL == 0)</li> <li>- M/D: Memory Download, (DL == 0)</li> <li>- M/V: Memory Verify,(DL == 0)</li> <li>- F/U: Firmware Upload</li> <li>- P/C: Production Mode Command</li> <li>- D/R: Device Run, (DL == 0)</li> <li>- D/S: Device Stop, (DL == 0)</li> </ul> <p><b>* Device → PC</b></p> <ul style="list-style-type: none"> <li>- E/D: Error Device</li> <li>- E/P: Error Packet</li> <li>- S/O: System Mode Change Success, (DL == 0)</li> <li>- S/X: System Mode Change Fail</li> <li>- S/N: System Mode is Normal</li> <li>- S/P: System Mode is Production</li> <li>- S/F: System Mode is Firmware Upload</li> <li>- S/E: System Mode is Error</li> <li>- R/O: Register Write Success, (DL == 0)</li> <li>- R/X: Register Write Fail</li> <li>- R/S: Register Send</li> <li>- M/O: Memory Command Success, (DL == 0)</li> <li>- M/X: Memory Command Fail</li> <li>- P/R: Production Mode Report</li> <li>- P/X: Production Mode Fail</li> <li>- F/O: Firmware Upload Success, (DL == 0)</li> <li>- F/A: Firmware Upload Packet Ack, (DL == 0)</li> <li>- F/X: Firmware Upload Fail</li> <li>- D/O: Device Command Success, (DL == 0)</li> <li>- D/X: Device Command Fail</li> </ul>
4		SUB CMD			
5~N	DATA	DATA	String or Binary	ASCII코드 or Binary	펌웨어를 제외한 모든 정보들은 문자열 형태로 전달

## A.2. UDP Protocol Packet Structure

UDP 통신의 기본 패킷 구조 아래 table과 같습니다.

UDP Protocol 기본 패킷 구조				
Byte	Name	Type	Range	Description
0~7	TS	UINT64	-	Timestamp. Unit: 10[nsec]. Little Endian(Byte 0: Lowest Byte)
8	PT	Character	ASCII코드	Packet Type - 'R': Raw Data – Single Echo - 'M': Raw Data – Multi Echo - 'H': Histogram Data - 'S': Sensing Data
9	FC	UINT8	-	Frame Count
10~11	WH	UINT16	-	WIDTH x HEIGHT ([15:9]Height, [8:0] Width)
12~13	CC	UINT16	-	Column Count (except "Sensing Data")
14~15	DL	UINT16	-	Data Length
16~N	DATA	-	-	It depends on "Packet Type"

Raw data 통신에 사용되는 패킷은 single echo(default) 모드와 multi echo 모드로 구성됩니다. 아래 table은 single echo 모드에서의 패킷 구조입니다.

UDP Protocol 패킷 구조(single echo)					
Data Byte	Name	Sub Name	Type	Range	Description
0~7	TS	-	UINT64	-	Timestamp. Unit: 10[nsec]. Little Endian(Byte 0: Lowest Byte)
8	PT	-	Character	'R' / 'r'	Raw Data – Single Echo ('R': mm / 'r': bin)
9	FC	-	UINT8	-	Frame Count
10~11	WH	-	UINT16	-	Width x Height ([15:9]Height, [8:0] Width)
12~13	CC	-	UINT16	-	Column ([15:9] Column Info, [8:0] Column Number)
14~15	DL	-	UINT16	700	Data Length (50[CH]: 700)
16~19	ROW[0]	THETA	FLOAT	-3.14 ~ 3.14	Polar Angle. Unit: Radian
20~23		PHI	FLOAT	-3.14 ~ 3.14	Azimuthal Angle. Unit: Radian
24~25			UINT16		Grey Image
26~29		DATA	UINT32	0 ~ 262143	[31:18] Intensity [17:0] Radial Distance. Unit (mm / bin)
...	...	...	...	...	...
702~705	ROW[49]	THETA	FLOAT	-3.14 ~ 3.14	Polar Angle. Unit: Radian
706~709		PHI	FLOAT	-3.14 ~ 3.14	Azimuthal Angle. Unit: Radian
710~711			UINT16		Grey Image
712~715		DATA	UINT32	0 ~ 262143	[31:18] Intensity [17:0] Radial Distance. Unit: (mm / bin)

## A.2. UDP Protocol Packet Structure

UDP 통신에서 multi-echo모드에서 사용되는 패킷 구조입니다.

UDP Protocol 패킷 구조(multi-echo)

Data Byte	Name	Sub Name	Type	Range	Description
0~7	TS	-	UINT64	-	Timestamp. Unit: 10[nsec]. Little Endian(Byte 0: Lowest Byte)
8	PT	-	Character	'M' / 'm'	Raw Data – Multi Echo ('R': mm / 'r': bin)
9	FC	-	UINT8	-	Frame Count
10~11	WH	-	UINT16	-	Width x Height ([15:9]Height, [8:0] Width)
12~13	CC	-	UINT16	-	Column ([15:9] Column Info, [8:0] Column Number)
14~15	DL	-	UINT16	900	Data Length (50[CH]: 900)
16~19	ROW[0]	THETA	FLOAT	-3.14 ~ 3.14	Polar Angle. Unit: Radian
20~23		PHI	FLOAT	-3.14 ~ 3.14	Azimuthal Angle. Unit: Radian
24~25			UINT16		Grey Image
26~29		DATA0	UINT32	0 ~ 262143	[31:18] Intensity, [17:0] Radial Distance. Unit: (mm / bin)
30~33		DATA1	UINT32	0 ~ 262143	[31:18] Intensity, [17:0] Radial Distance. Unit: (mm / bin)
...	...	...	...	...	...
898~901	ROW[49]	THETA	FLOAT	-3.14 ~ 3.14	Polar Angle. Unit: Radian
902~905		PHI	FLOAT	-3.14 ~ 3.14	Azimuthal Angle. Unit: Radian
906~909			UINT16		Grey Image
910~911		DATA0	UINT32	0 ~ 262143	[31:18] Intensity, [17:0] Radial Distance. Unit: (mm / bin)
912~915		DATA1	UINT32	0 ~ 262143	[31:18] Intensity, [17:0] Radial Distance. Unit: (mm / bin)

## A.3. Device Information

아래는 Read된 ML Device 정보이며, 이름/데이터 타입/값으로 작성되어 있습니다.

ml.txt
SN,CHAR,16,M,L,S,E,R,I,A,L,N,U,M,B,E,R,0,0
RXVDP,FLOAT,1,0.996471
RXVDN,FLOAT,1,3.300000
RXHV,FLOAT,1,27.486275
RXHVCTRL,UINT32,1,1
MACADDR,UINT8,6,0,128,225,0,0,0
ML_IP,UINT8,4,192,168,1,10
ML_PORT,UINT16,1,2000
PC_IP,UINT8,4,192,168,1,15
PC_PORT,UINT16,1,2000
GATEWAY,UINT8,4,192,168,1,1
NETMASK,UINT8,4,255,255,255,0
STREAM,UINT32,1,0
THRESCOE,FLOAT,4,80.910004,0.429500,54.349998,15.000000
LENSCOE,FLOAT,12,558.652771,558.652771,199.500000,49.500000,0.000000,0.000000,-0.843500,1.099000,-8.880000, ...
GLBOFS,UINT16,1,20
PRI,UINT32,1,847
INTSREJ,UINT32,1,1
TGSAMPL,UINT32,1,120
STARTCOL,UINT32,1,104
ENDCOL,UINT32,1,295
VCSELPAR,UINT32,1,832
PULSEINH,UINT32,1,0
PULSEEDG,UINT32,1,1
PULSEDEL,UINT32,1,587
PULSEDUT,UINT32,1,2
VCSELCS,UINT32,1,0
VCSELTYP,UINT32,1,1
JITTEREN,UINT32,1,1
SPADOFSL,UINT8,100,14,16,14,15,12,15,13,15,13,13,14,13,9,12,10,12,10,14,9,10,8,10,9,10,4,7,0,2,3,6,5,5,8,7,9,7,8,6,8,5, ...
SPADOFSR,UINT8,100,20,18,21,17,19,17,18,15,15,15,16,15,15,15,16,12,13,11,11,9,9,6,6,2,9,6,9,8,11,7,9,8,11,6,7,5,7,5,8,7, ...
DUALMODE,UINT32,1,1
COEGRPA,UINT32,1,2271560481
COEGRPB,UINT32,1,19088743
FIROFS,UINT32,1,7
COGTAP,UINT32,1,15
STARTROW,UINT32,1,0
ENDROW,UINT32,1,99
INTSMODE,UINT32,1,0
PEAKWIDT,UINT32,1,16
TYPEBPAR,UINT32,1,22023651
VCSELOFS,UINT16,96,6,5,5,5,3,5,4,4,4,5,4,3,4,5,4,2,1,3,4,2,1,3,1,1,1,2,3,2,1,2,1,0,0,4,2,2,1,2,1,1,1,3,4,3,3,4,4,3,2,3,4,3,3, ...
PLVER,UINT32,1,21040500
PLTMRMMS,UINT32,1,526597
HWREG0,UINT32,1,469762071
FWVER,CHAR,15,2,0,2,1,0,9,2,7,0,8,5,3,3,7,A
FILLING,UINT32,1,1
EMITDWTH,FLOAT,1,-10.000000

## A.3. Device Information

아래는 Read된 ML Device 정보이며, 이름/데이터 타입/값으로 작성되어 있습니다.

이름	데이터 타입	설명
SN	CHAR	ML 시리얼 번호
RXVDP	FLOAT	ML RX VDP 값
RXVDN	FLOAT	ML RX VDN 값
RXHV	FLOAT	ML RX HV 값
RXHVCTRL	UINT32	RX HV 컨트롤 여부
MACADDR	UINT8	ML MAC Address
ML_IP	UINT8	ML IP 주소
ML_PORT	UINT16	ML Port
PC_IP	UINT8	PC IP 주소
PC_PORT	UINT16	PC Port
GATEWAY	UINT8	PC Gateway
NETMASK	UINT8	PC Netmask
STREAM	UINT32	ML Steam 여부
THRESCOE	FLOAT	ML Adaptive Threshold 파라미터
LENSCOE	FLOAT	ML 렌즈 파라미터
GLBOFS	UINT16	ML Global Offset 값
PRI	UINT32	ML PRI 값
INTSREJ	UINT32	
TGSAMPL	UINT32	Sample 개수
STARTCOL	UINT32	ML 시작 Column
ENDCOL	UINT32	ML 끝 Column
VCSELPAR	UINT32	VCSEL
PULSEINH	UINT32	Pulse
PULSEEDG	UINT32	Pulse
PULSEDEL	UINT32	Pulse Delay
PULSEDUT	UINT32	Pulse Duty
VCSELCS	UINT32	VCSEL
VCSELTYP	UINT32	VCSEL 타입
JITTEREN	UINT32	Jitter 여부
SPADOFSL	UINT8	SPAD
SPADOFSR	UINT8	SPAD
DUALMODE	UINT32	Dual Mode 여부
COEGRPA	UINT32	
COEGRPB	UINT32	
FIROFS	UINT32	FIR
COGTAP	UINT32	COG Tap Number
STARTROW	UINT32	ML 시작 Row
ENDROW	UINT32	ML 끝 Row
INTSMODE	UINT32	Intensity Mode
PEAKWIDT	UINT32	Peak Width
TYPEBPAR	UINT32	
VCSELOFS	UINT16	VCSEL
PLVER	UINT32	
PLTMRMS	UINT32	
HWREG0	UINT32	
FWVER	CHAR	
FILLING	UINT32	Filling 알고리즘 적용 여부
EMITDWTH	FLOAT	

## A.4. LiDAR API - Typedefs

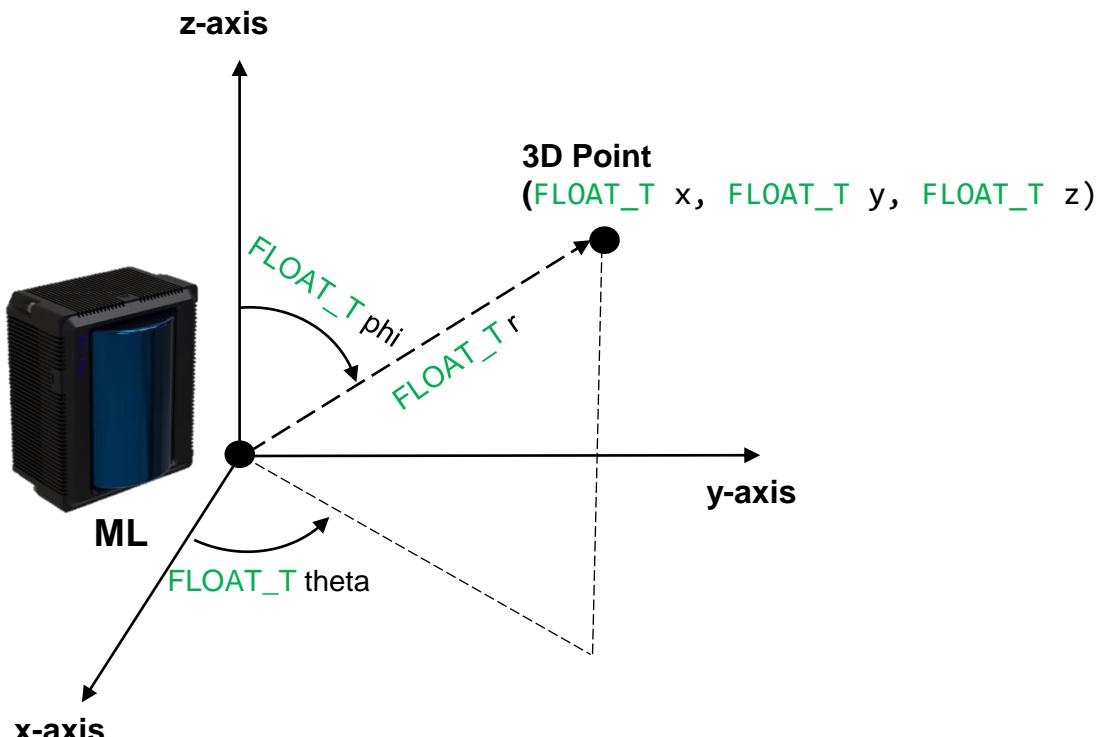
### Typedefs

이름	SOSLAB::INT_T
선언	<code>typedef int32_t SOSLAB::INT_T</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	integer 변수
이름	SOSLAB::UINT_T
선언	<code>typedef uint32_t SOSLAB::UINT_T</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	unsigned int 변수형
이름	SOSLAB::FLOAT_T
선언	<code>typedef double SOSLAB::FLOAT_T</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	double 변수형
이름	SOSLAB::hex_array_t
선언	<code>typedef std::vector&lt;uint8_t&gt; SOSLAB::hex_array_t</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	16진수 배열 변수형
이름	SOSLAB::pointcloud_t
선언	<code>typedef std::vector&lt;point_t&gt; SOSLAB::pointcloud_t</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	SOSLAB LiDAR Point Cloud 변수형

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::spherical_point_t
선언	<code>typedef struct _SPHERICAL_POINT_T spherical_point_t</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	single 3D Point에 대응하는 spherical coordinate system 구조체
member variables	member variables 설명
FLOAT_T r	spherical coordinate system의 Range 값(Depth)
FLOAT_T theta	spherical coordinate system의 theta 값
FLOAT_T phi	spherical coordinate system의 phi 값

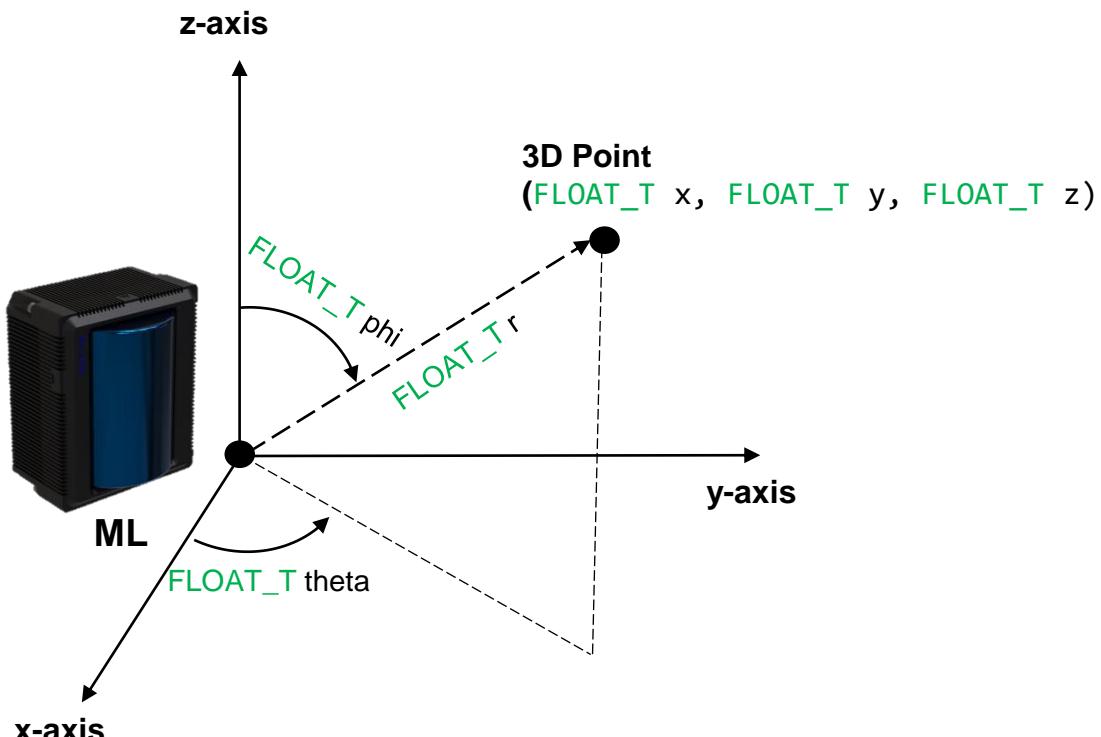


single 3D point의 spherical, cartesian coordinate system

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::cartesian_point_t
선언	<code>typedef struct _CARTESIAN_POINT_T cartesian_point_t</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	single 3D Point에 대응하는 cartesian coordinate system 구조체
member variables	member variables 설명
FLOAT_T x	cartesian coordinate system의 x 값
FLOAT_T y	cartesian coordinate system의 y 값
FLOAT_T z	cartesian coordinate system의 z 값



single 3D point의 spherical, cartesian coordinate system

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::point_t
선언	typedef struct _POINT_T point_t
header file	#include "soslab_typedef.h"
설명	single 3D Point에 대응하는 두 가지 coordinate system 구조체
member variables	member variables 설명
std::size_t index	Point index – 사용 안함
std::size_t timestamp_1us	Timestamp [1 micro second 단위] – 사용 안함
spherical_point_t rtp	point의 spherical coordinate system 정보
cartesian_point_t xyz	point의 cartesian coordinate system 정보
FLOAT_T pw	single point에 대응하는 intensity 값

변수형	SOSLAB::ip_setting_t
선언	typedef struct IP_ADDRESS_T ip_setting_t
header file	#include "soslab_typedef.h"
설명	IP 주소와 Port 정보를 저장하는 구조체
member variables	member variables 설명
std::string ip_address	IP 주소
int port_number	Port 번호

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::LidarML::scene_t
선언	<code>typedef struct _ML1_SCENE_T point_t</code>
header file	<code>#include "soslab_typedef.h"</code>
설명	single 3D Point에 대응하는 두 가지 coordinate system 구조체
member variables	member variables 설명
<code>std::vector&lt;uint64_t&gt;</code> timestamp	Timestamp [1 micro second 단위]
<code>uint8_t frame_id</code>	Raw sequence data의 index 값
<code>std::size_t rows</code>	Raw Data의 height 값
<code>std::size_t cols</code>	Raw Data의 width 값
<code>FLOAT_T pw</code>	single point에 대응하는 intensity 값
<code>std::vector&lt;uint16_t&gt;</code> grey_image	Ambient 데이터 배열 [size : cols x rows] (Ambient : 측정 된 모든 빛을 표현한 데이터)
<code>std::vector&lt;uint32_t&gt;</code> depth_image	Depth 데이터 배열 [size : cols x rows] (Depth : 측정된 거리 데이터 [단위 : mm])
<code>std::vector&lt;uint16_t&gt;</code> intensity_image	Intensity 데이터 배열 [size : cols x rows] (Intensity : 측정된 Lidar 신호 강도)
<code>std::vector&lt;point_t&gt;</code> pointcloud	Point cloud 데이터 배열 [size : cols x rows] (Point cloud : 3차원 point의 집합 데이터)
<code>std::vector&lt;uint32_t&gt;</code> depth_image2	Multi echo에서 2번째로 측정된 Depth 데이터 배열 [size : cols x rows]
<code>std::vector&lt;uint16_t&gt;</code> intensity_image2	Multi echo에서 2번째로 측정된 Intensity 데이터 배열 [size : cols x rows]
<code>std::vector&lt;point_t&gt;</code> pointcloud2	Multi echo에서 2번째로 측정된 Point cloud 데이터 배열 [size : cols x rows]

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::LidarMl
선언	<code>class LidarMl</code>
header file	#include "ml/libsoslab_ml.h"
설명	ML Device와 연결, 데이터 획득 등을 관리하는 Class
<b>member functions</b>	
<code>bool connect(const ip_settings_t ml, const ip_settings_t local);</code>	
설명 : Local(PC)에서 ML device로 접속하는 함수 input : ML IP 주소와 Port 번호(ml)와 Local IP 주소와 Port 번호(local) output : 접속 상태에 대해 bool 변수형으로 반환(접속되면 true 반환)	
<code>bool disconnect();</code>	
설명 : Local(PC)에서 ML device로 접속을 해제하는 함수 output : 접속 해제 상태에 대해 bool 변수형으로 반환(해제되면 true 반환)	
<code>bool start();</code>	
설명 : ML Device Scan을 시작하는 함수 output : Scan 시작 유무에 대해 bool 변수형으로 반환(시작되면 true 반환)	
<code>bool tcp_device_run();</code>	
설명 : ML Device Scan을 시작하는 함수 output : Scan 시작 유무에 대해 bool 변수형으로 반환(시작되면 true 반환)	
<code>bool stop();</code>	
설명 : ML Device Scan을 중단하는 함수 output : Scan 중단 유무에 대해 bool 변수형으로 반환(중단되면 true 반환)	
<code>bool tcp_device_stop();</code>	
설명 : ML Device Scan을 중단하는 함수 output : Scan 중단 유무에 대해 bool 변수형으로 반환(중단되면 true 반환)	
<code>bool get_scene(scene_t&amp; scene);</code>	
설명 : 입력 변수 scene으로 Raw Data를 획득하는 함수 input : scene_t 변수형 output : ML Device Raw Data를 저장한 scene_t 변수	

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::LidarML
선언	<code>class LidarML</code>
header file	<code>#include "ml/libsoslab_ml.h"</code>
설명	ML Device와 연결, 데이터 획득 등을 관리하는 Class
<b>member functions</b>	
<code>void set_zigzag_filling_enable(bool enable)</code>	
설명	: ML device를 위한 Data Processing Module의 zigzag filling 기능 사용 여부를 세팅하는 함수입니다. (default : <code>true</code> )
input	: zigzag filling 기능 On( <code>true</code> )/Off( <code>false</code> )를 입력으로 받습니다.
<code>void set_deflaring_enable(bool enable)</code>	
설명	: ML device를 위한 Data Processing Module의 deflaring 기능 사용 여부를 세팅하는 함수입니다. (default : <code>false</code> )
input	: deflaring 기능 On( <code>true</code> )/Off( <code>false</code> )를 입력으로 받습니다.
<code>void set_real_ambient_enable(bool enable)</code>	
설명	: ML device를 위한 Data Processing Module의 real ambient 기능 사용 여부를 세팅하는 함수입니다. (default : <code>false</code> )
input	: real ambient 기능 On( <code>true</code> )/Off( <code>false</code> )를 입력으로 받습니다.
<code>void set_line_filling_enable(bool enable)</code>	
설명	: ML2 device를 위한 Data Processing Module의 line filling 기능 사용 여부를 세팅하는 함수입니다. (default : <code>true</code> )
input	: line filling 기능 On( <code>true</code> )/Off( <code>false</code> )를 입력으로 받습니다.
<code>void set_tr_retro_threshold_ml0(bool enable)</code>	
설명	: ML0 device를 위한 deflaring 기능의 parameter를 세팅하는 함수입니다. (defalut : 110)
input	: <code>int</code> 변수형의 parameter 값을 입력으로 받습니다.
<code>void set_tr_retro_threshold_ml2(bool enable)</code>	
설명	: ML2 device를 위한 deflaring 기능의 parameter를 세팅하는 함수입니다. (defalut : 700)
input	: <code>int</code> 변수형의 parameter 값을 입력으로 받습니다.

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::LidarML
선언	<code>class LidarML</code>
header file	<code>#include "ml/libsoslab_ml.h"</code>
설명	ML Device와 연결, 데이터 획득 등을 관리하는 Class
<b>member functions</b>	
<code>void set_line_filling_left_offset(int val)</code>	
설명 : ML2 device를 위한 line filling 기능의 parameter를 세팅하는 함수입니다. (defalut : 0) input : <code>int</code> 변수형의 parameter 값을 입력으로 받습니다.	
<code>void set_line_filling_right_offset(int val)</code>	
설명 : ML2 device를 위한 line filling 기능의 parameter를 세팅하는 함수입니다. (defalut : 5) input : <code>int</code> 변수형의 parameter 값을 입력으로 받습니다.	

## A.5. LiDAR API - Classes

### Classes

변수형	SOSLAB::LidarML
선언	class LidarML
header file	#include "ml/libsosslab_ml.h"
설명	ML Device와 연결, 데이터 획득 등을 관리하는 Class

#### member functions

`void start_recording(std::string dir_path)`

설명 : ML 데이터 녹화 기능을 시작하는 함수입니다.

Input : 녹화 파일이 저장될 위치를 입력으로 받습니다.

`void stop_recording()`

설명 : ML 데이터 녹화 기능을 정지하는 함수입니다.

`void connect_file(const std::string filepath)`

설명 : ML 데이터 녹화 파일을 Loading하는 함수입니다.

Input : 녹화 파일이 저장된 위치를 입력으로 받습니다.

`void binary2file(std::string save_directory)`

설명 : ML 데이터 녹화 파일을 pcd 파일로 변환하는 함수입니다.

Input : pcd파일을 저장할 위치를 입력으로 받습니다.

`bool get_scene(scene_t& scene, uint64_t index);`

설명 : connect\_file를 통해 녹화 파일을 loading한 뒤, 입력 변수 scene과 frame 번호를 통해 해당 frame의 Data를 획득하는 함수입니다.

Input 1) Raw Data를 저장할 변수 (`scene_t& scene`)

Input 2) Load할 frame (`uint64_t index`)

output : ML Device Raw Data를 저장한 `scene_t` 변수

`void get_logging_file_size(uint64_t& size)`

설명 : 설명 : connect\_file를 통해 녹화 파일을 loading한 뒤, 총 Frame 수를 반환합니다.

Input : `uint64_t` 변수를 입력으로 받아 총 Frame 수를 반환합니다.