Solid-State LiDAR ML-X

User Guide

Version History	Last updated	Description
Release v2.0	2023-07-25	Multi-echo On/Off 기능 추가

Table of Contents

Chapter 1

Hardware Configuration

1.1. Mechanical Interface

1.1.1 Included Components

ML-X는 다음과 같은 아이템을 포함하여 제공합니다.

- ML-X 80°, 80° 전용 통합 (Power/Ethernet/Time Sync) 케이블
- ML-X 120°, 120° 전용 통합 (Power/Ethernet/Time Sync) 케이블
- Sensor AC/DC Power Adapter/Power Cable(80°, 120° 공통)

(a) ML-X 120°

(b) ML-X 80°

(c) 120° 전용 통합 케이블

(d) 80° 전용 통합 케이블

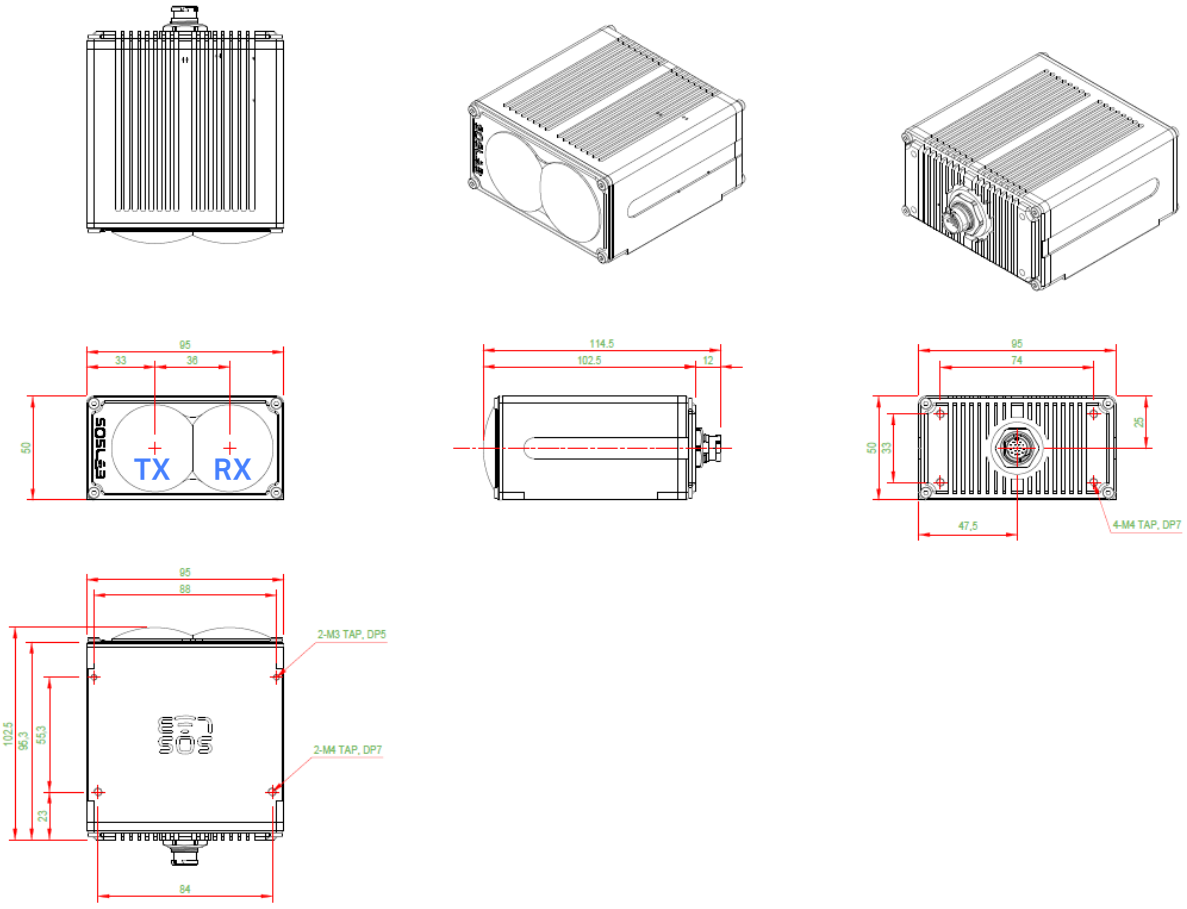
(e) AC/DC Power Adapter

(f) AC/DC Power Cable

1.1. Mechanical Interface

1.1.2 Exterior Mechanical Dimensions

ML-X 120° Dimensions

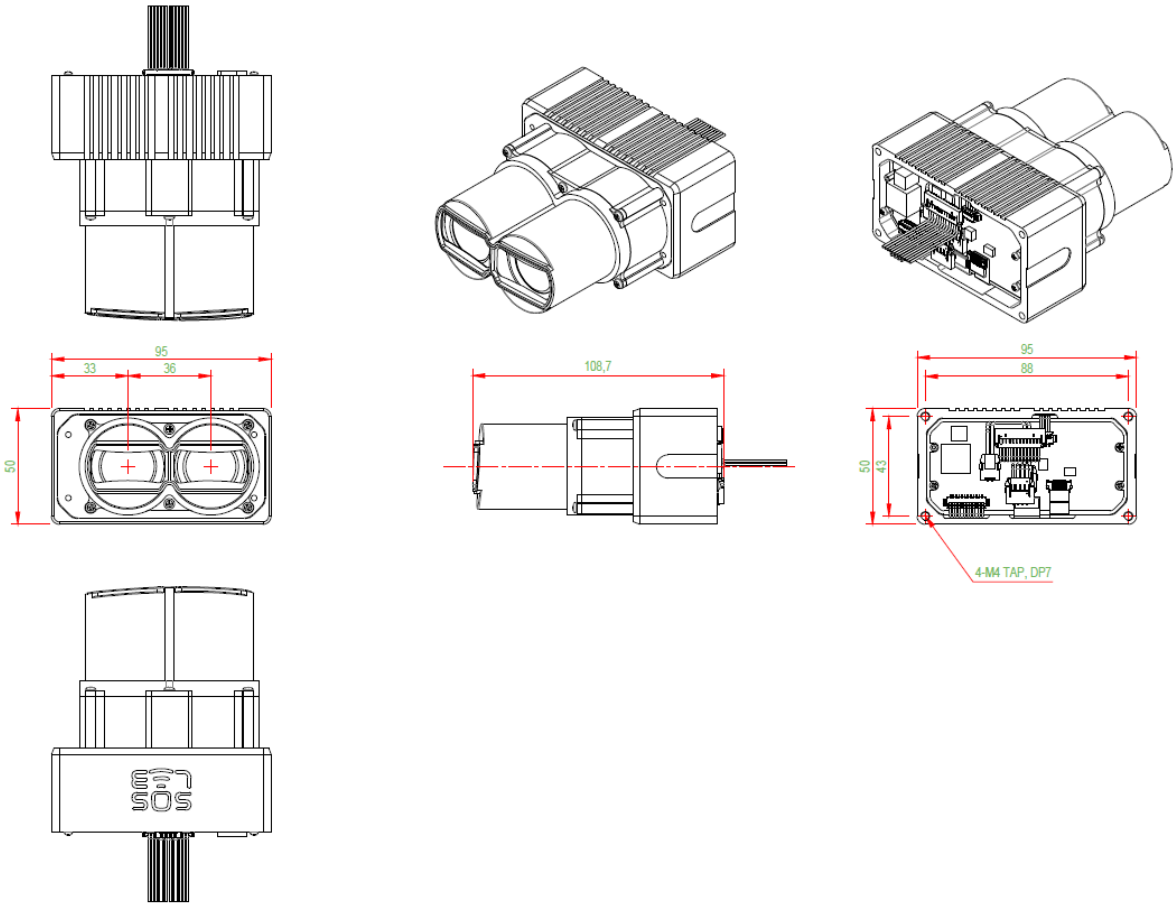


<The sensor has 4×M4 mounting holes>

1.1. Mechanical Interface

1.1.2 Exterior Mechanical Dimensions

ML-X 80° Dimensions



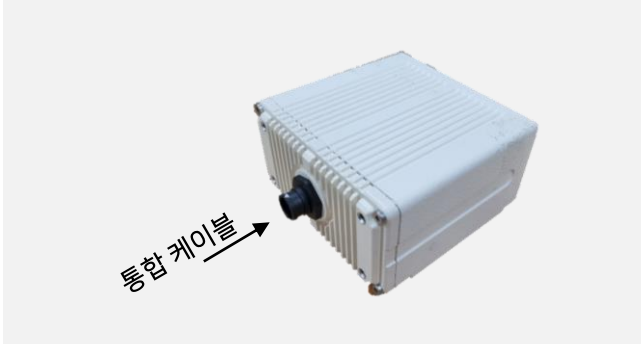
<The sensor has 4×M4 mounting holes>

1.2. Electrical Interface

1.2.1 Cable Connection

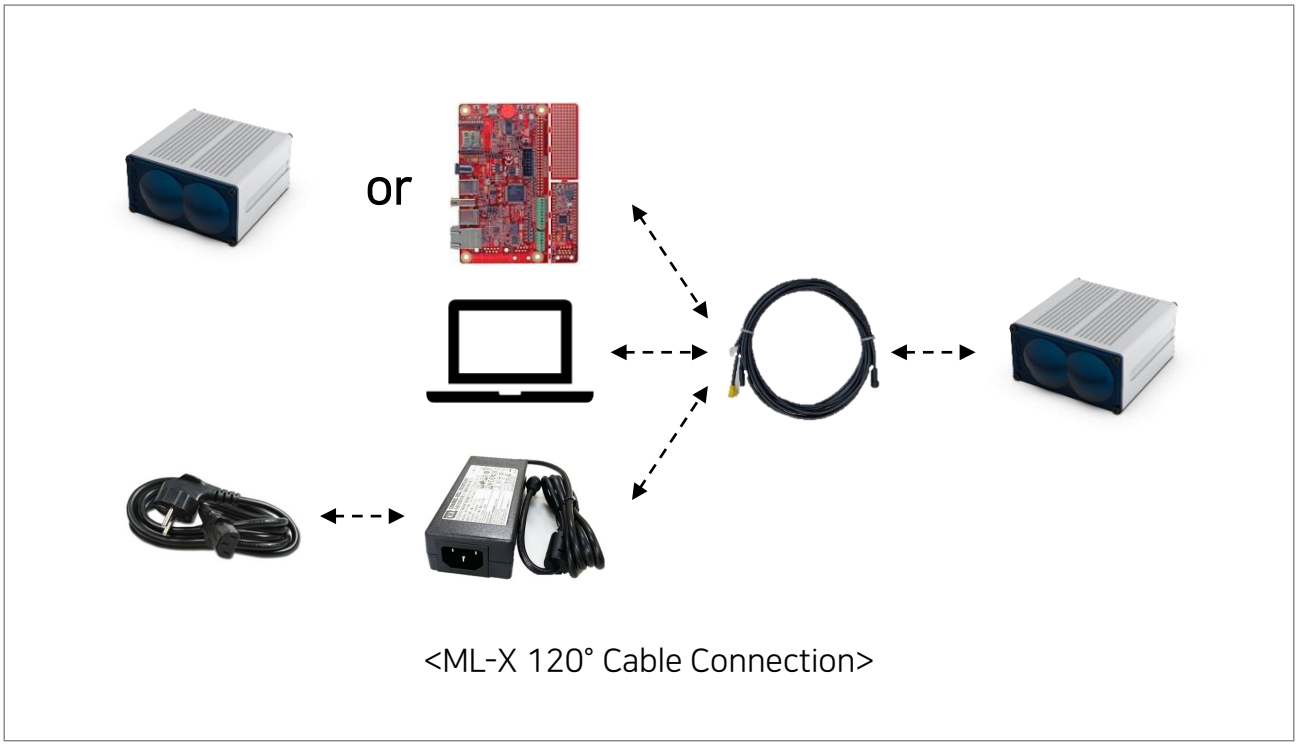
케이블을 연결하는 순서는 다음과 같습니다.

1. 제공된 센서와 통합 케이블을 연결합니다.
2. 이더넷 케이블과 PC를 연결합니다.
3. Timing Synchronization을 사용할 경우, 제공된 센서와 외부 Device를 연결합니다.
4. 파워케이블과 파워 어댑터를 연결합니다.



<ML-X 120 ° Cable to Connector>

<ML-X 80 ° Cable to Connector>



<ML-X 120° Cable Connection>

1.2.2 IP/Port Information

제공된 센서의 기본 IP/Port 정보는 다음과 같습니다.

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

1.2.3 Power Connector

센서의 Power 연결은 제공된 파워 어댑터를 사용하는 방법과 External Power Cable을 사용하는 방법이 있습니다.

1.2.3.1 Power Adapter Information

제공된 파워 어댑터의 정보는 다음과 같습니다.

- 정격입력 : 100-240V 50/60Hz 1.7A
- 정격출력 : +12.0V 5.0A 60.0W

1.2.3.2 External Power Cable

ML-X External Power Cable은 12V~24V의 전압을 공급하며 4개의 Wire로 구성됩니다.

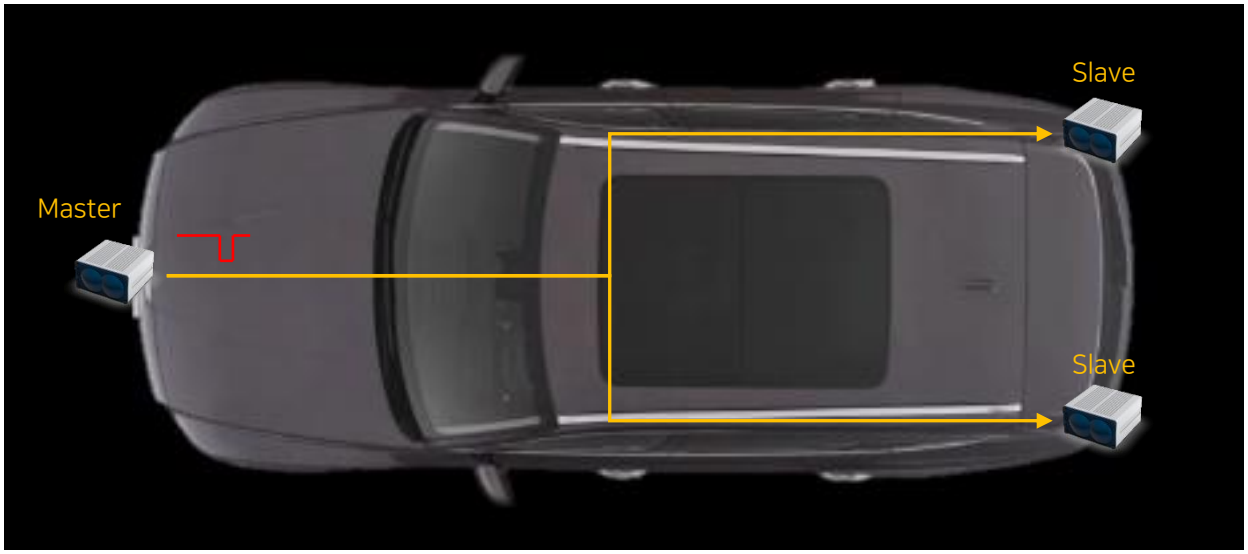
Power Cable Wire	DC 5.5 전원 잭
하얀색: EX_GND	(-) 극
하얀색+DOT: EX_GND	
주황색: EX_VIN	(+) 극
주황색+DOT: EN_VIN	

4개의 Wire는 아래와 같이 DC 5.5 전원 잭과 연결하여 전원으로 사용할 수 있습니다.

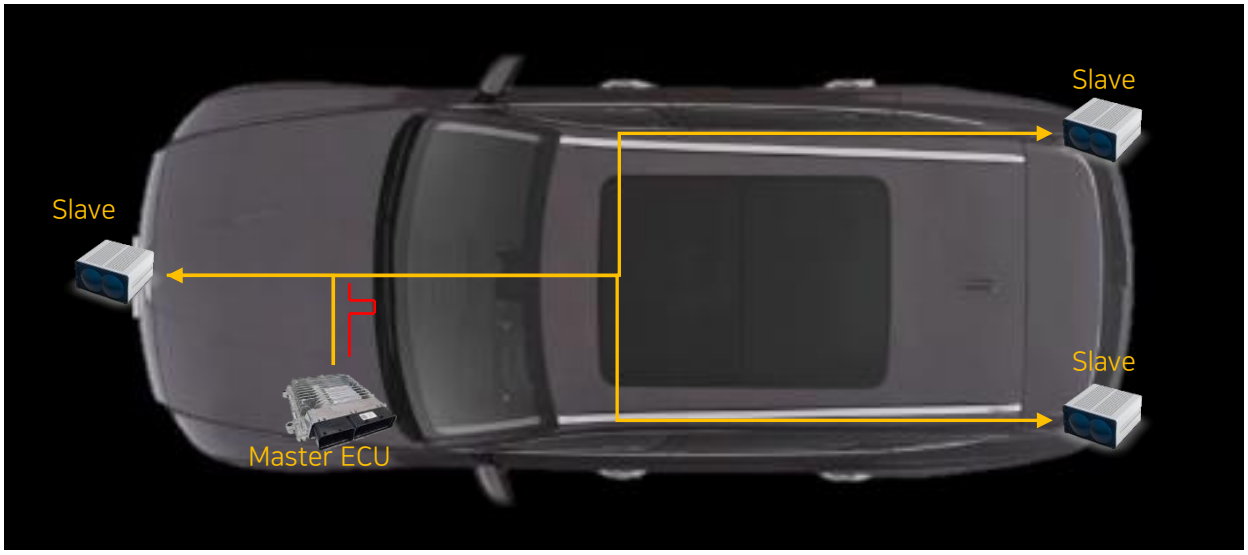
1.2.4 Timing Synchronization

ML-X는 입/출력 신호를 사용하여 프레임 동기화 기능을 제공하며, 아래 그림과 같이 다양한 형태의 Application을 구성할 수 있습니다.

1) ML-X들을 Master/Slave로 구성하여 프레임 동기화를 구현할 수 있습니다.



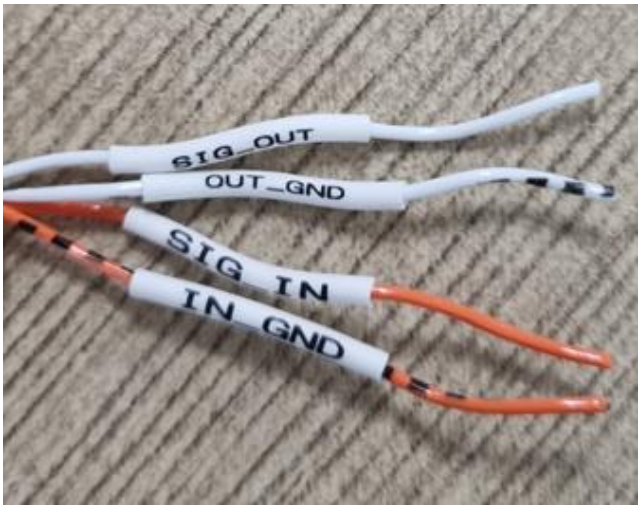
2) 외부 Device(ex. ECU)의 신호를 받아서 ML-X가 Slave로 동기화될 수 있습니다.



1.2. Electrical Interface

1.2.4.1 External SYNC IN/OUT Cable

ML-X External Cable은 SYNC IN/OUT 포트를 제공하고, 해당 포트를 사용하여 외부에 회로를 구성할 수 있습니다.

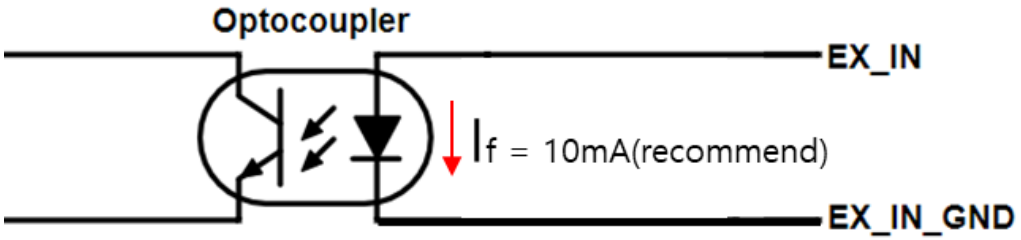


- SIG_IN (주황색, EX_IN)
- IN_GND (주황색+DOT, EX_IN_GND),
- SIG_OUT (하얀색, EX_OUT)
- OUT_GND (하얀색+DOT, EX_OUT_GND)

1.2.4.2 External SYNC IN

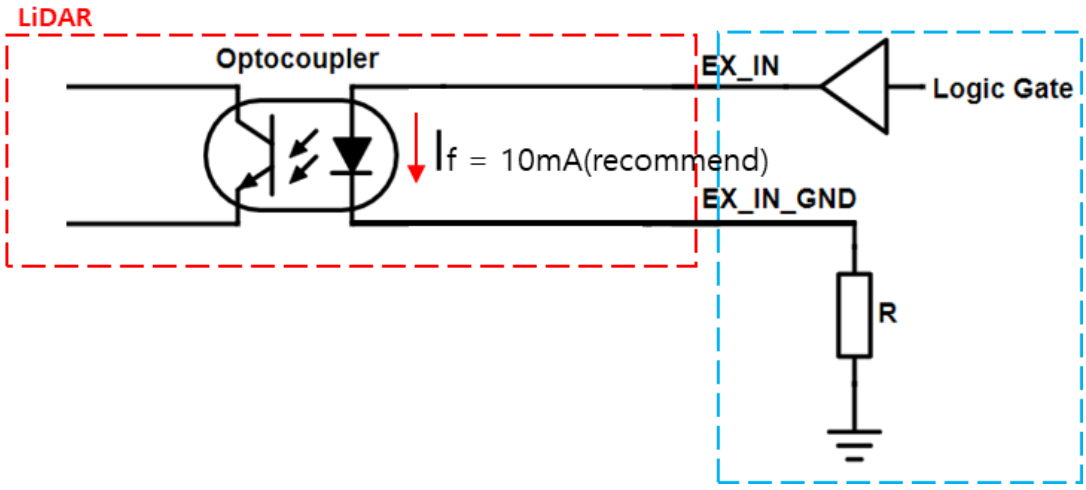
ML-X의 Sync Input 내부 회로 구성과, 외부 회로 구성입니다.

1) ML-X의 Sync Input 내부 회로구성은 아래와 같습니다.



2) 외부 회로 구성은 아래 그림을 참조하시기 바랍니다.

External Circuit 구성



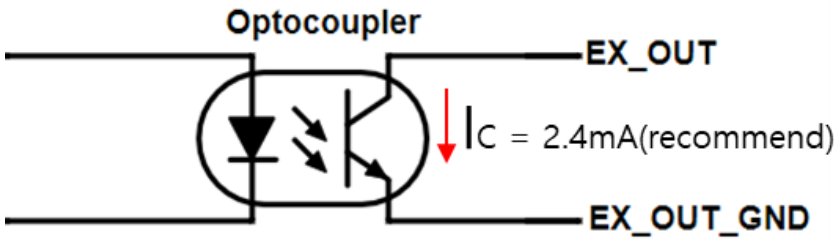
3) EX_IN Voltage에 따른 저항 값 선정

EX_IN Voltage	External Resistor(R)	Forward Current (I_F)	Recommended I_F
3.3 V (Min)	200	10.25mA	$7.5\ mA < I_F < 15\ mA$ $I_F(mA) = \frac{V - 1.25}{R} * 1000$
5 V	360	10.41 mA	
12 V	1000	10.75 mA	
24 V (Max)	2200	10.34 mA	

1.2.4.3 External SYNC OUT

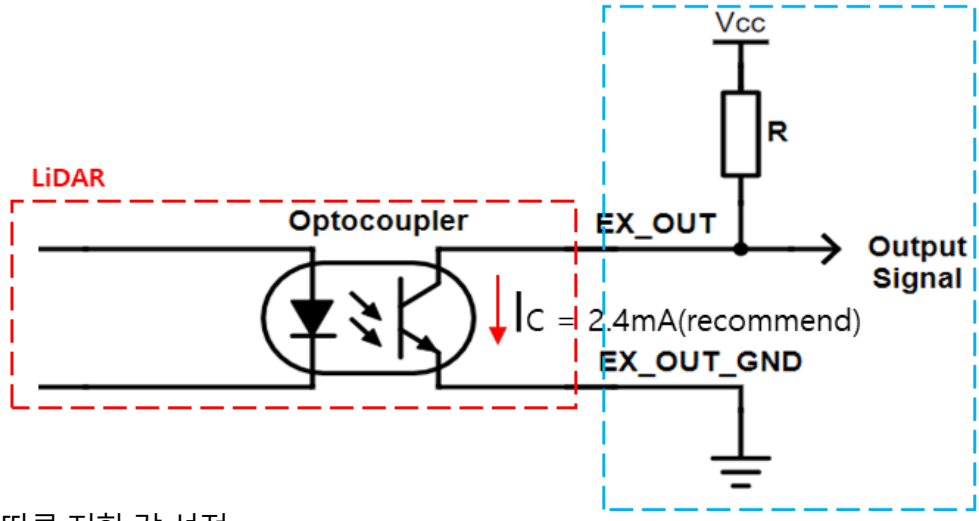
ML-X의 Sync Output 내부 회로 구성과, 외부 회로 구성입니다.

1) ML-X의 Sync Output 내부 회로구성은 아래와 같습니다.



2) 외부 회로 구성은 아래 그림을 참조하시기 바랍니다.

External Circuit 구성



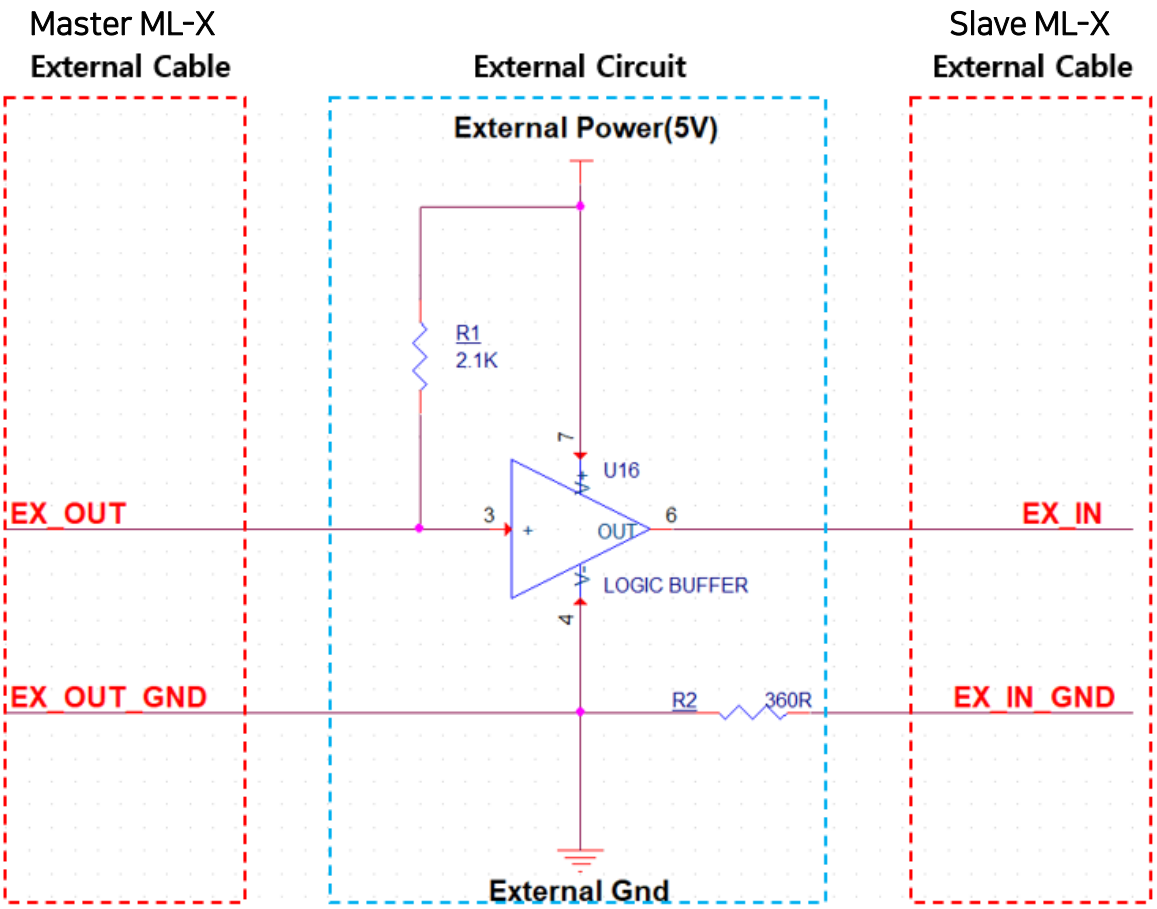
3) V_{CC} 에 따른 저항 값 선정

External Voltage (V_{CC})	External Resistor (R)	Collector Current (I_C)	Recommended I_C
3.3 V (Min)	1375	2.4 mA	$1\ \text{mA} < I_C < 10\ \text{mA}$ $I_C(\text{mA}) = \frac{V_{CC}}{R} * 1000$
5 V	2100	2.4 mA	
12 V	5000	2.4 mA	
24 V (Max)	10000	2.4 mA	

1.2.4.4 Sync IN/OUT Combine

Single Master ML-X ↔ Single Slave ML-X 연결 예제입니다.

1) External Circuit 구성
   - 외부 전원: 5V

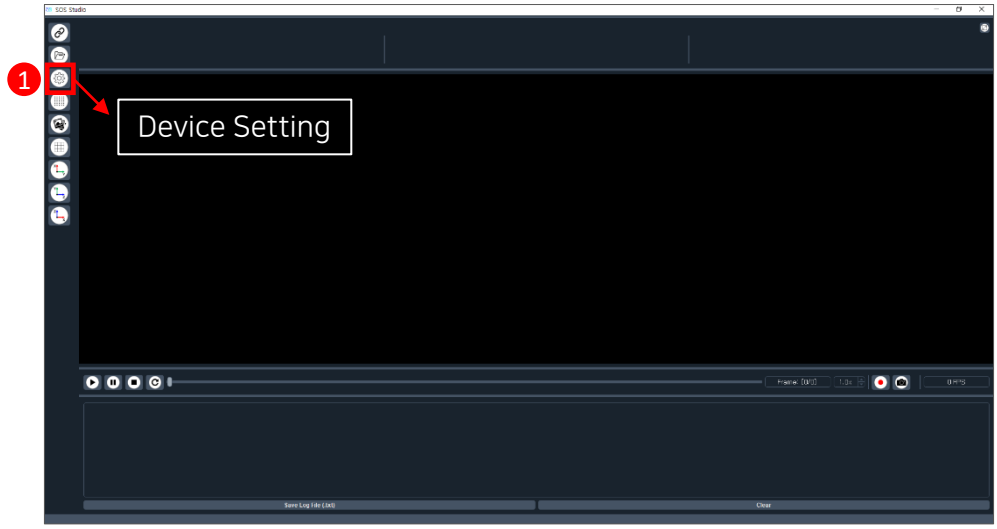


1.2.4.5 SYNC FUNCTION

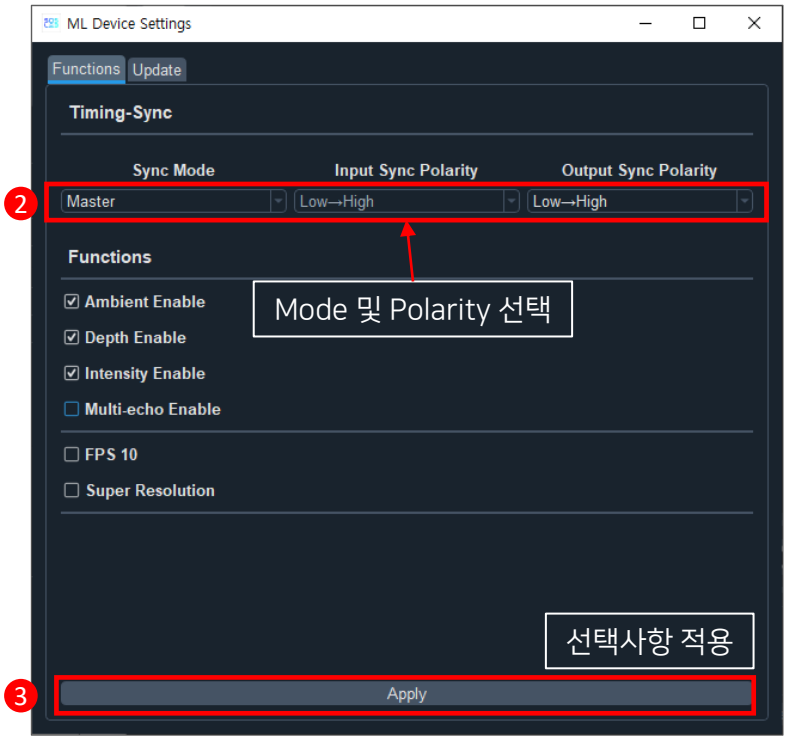
SOS Studio를 사용하여 ML-X의 Sync Mode를 Master 또는 Slave로 설정 할 수 있습니다.

1) Sync Mode 설정 방법

- SOS Studio를 실행 후 Device Setting 클릭

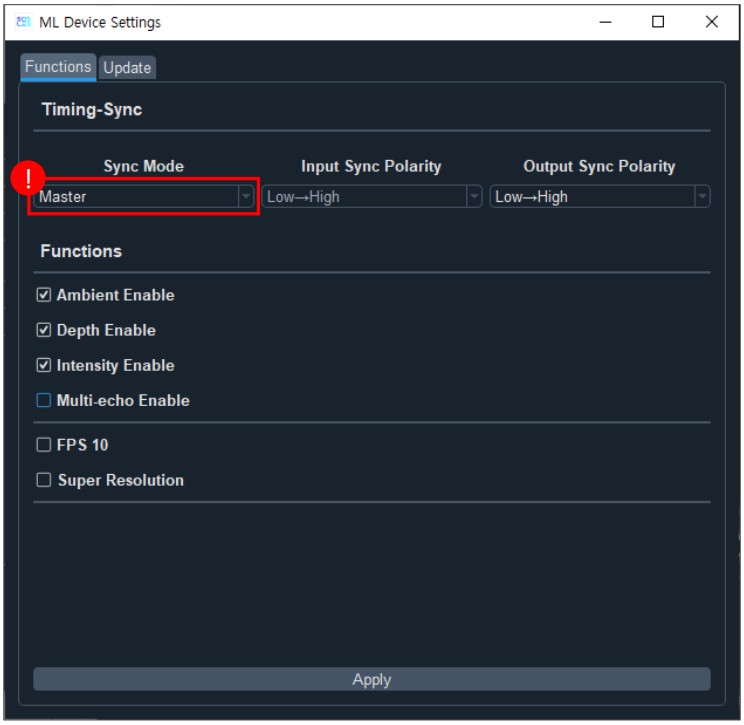


- Functions 탭에서 모드 선택 및 적용가능

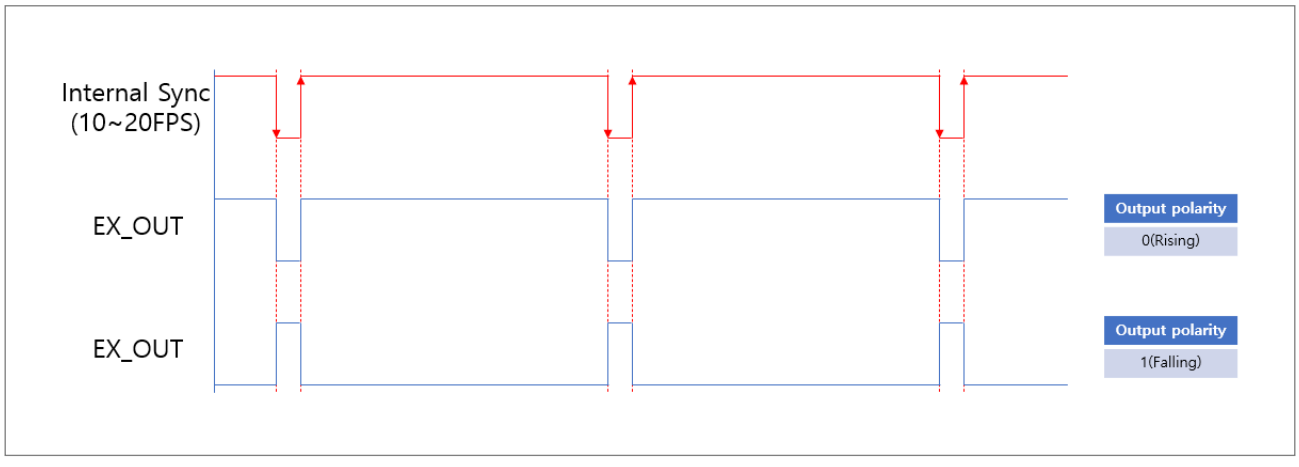


2) MASTER

- Sync Mode를 Master 선택 후 Apply 클릭하면 Master Mode로 작동합니다.

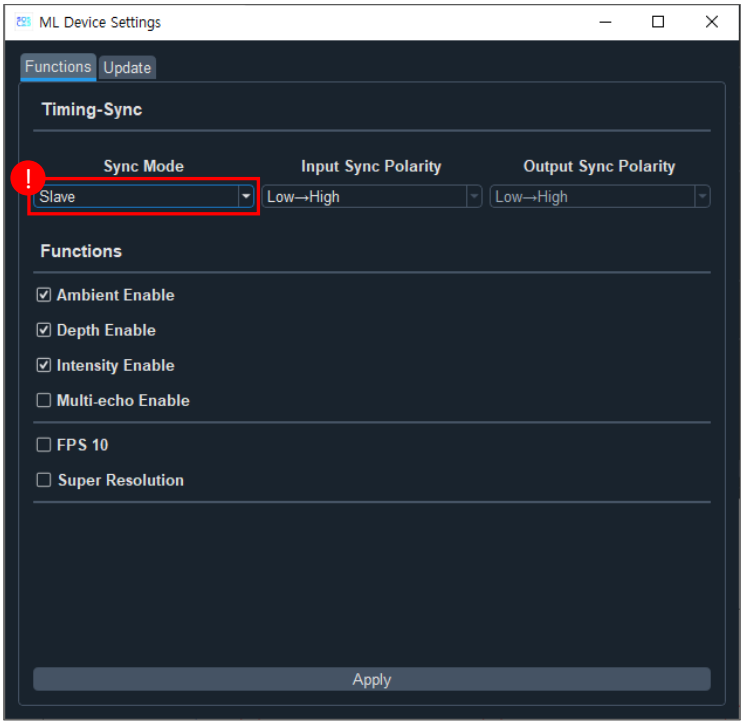


- 설정된 Frame Rate에 맞춰 매 프레임마다 동기화 신호를 출력합니다. Sync Out의 Polarity(High → Low 또는 Low → High) 변경이 가능합니다

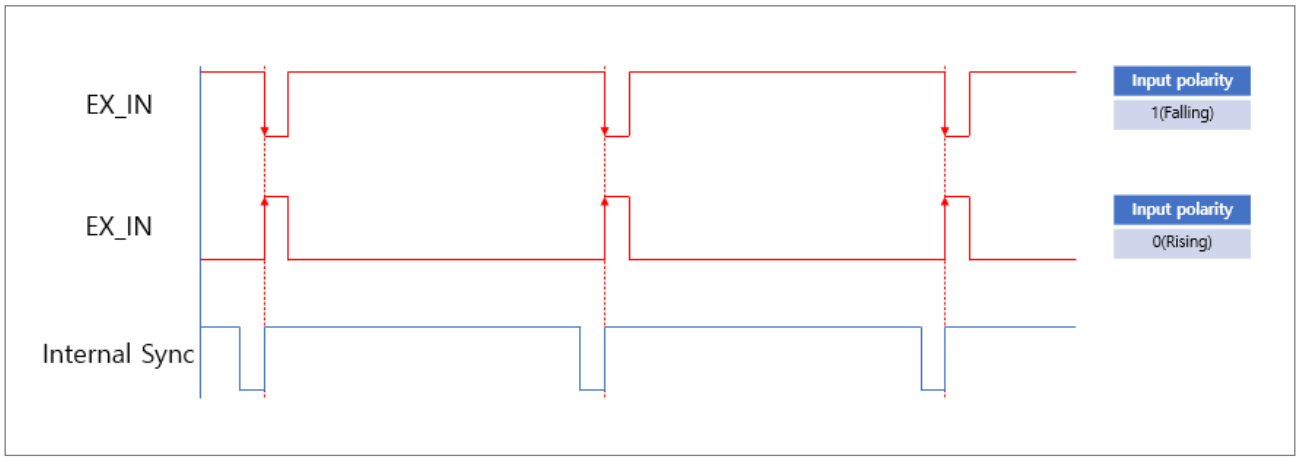


3) SLAVE

- Sync Mode를 Slave 선택 후 Apply 클릭하면 Slave Mode로 작동합니다.

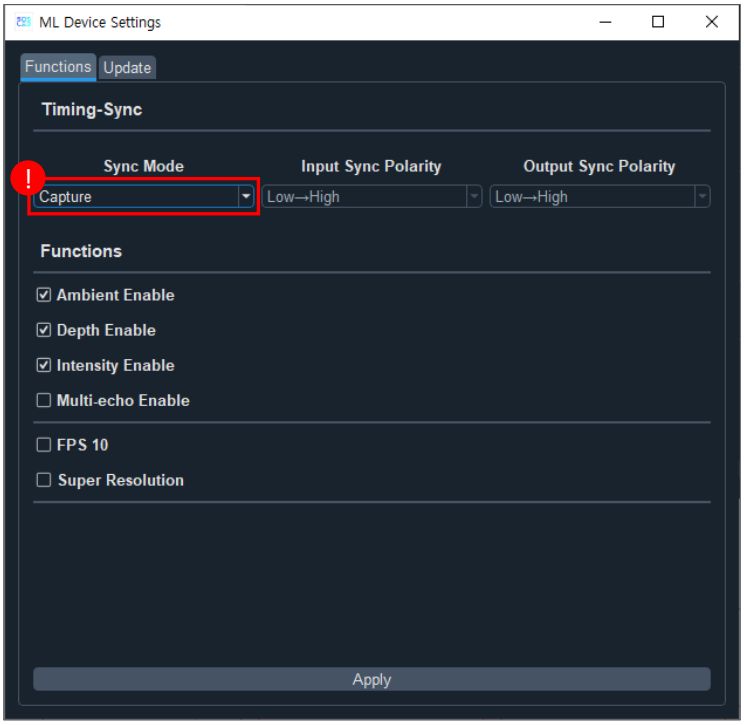


- 입력 받는 동기화 신호의 타이밍에 맞춰 프레임 단위로 동작합니다. Sync Input Polarity(High → Low 또는 Low → High) 변경이 가능합니다

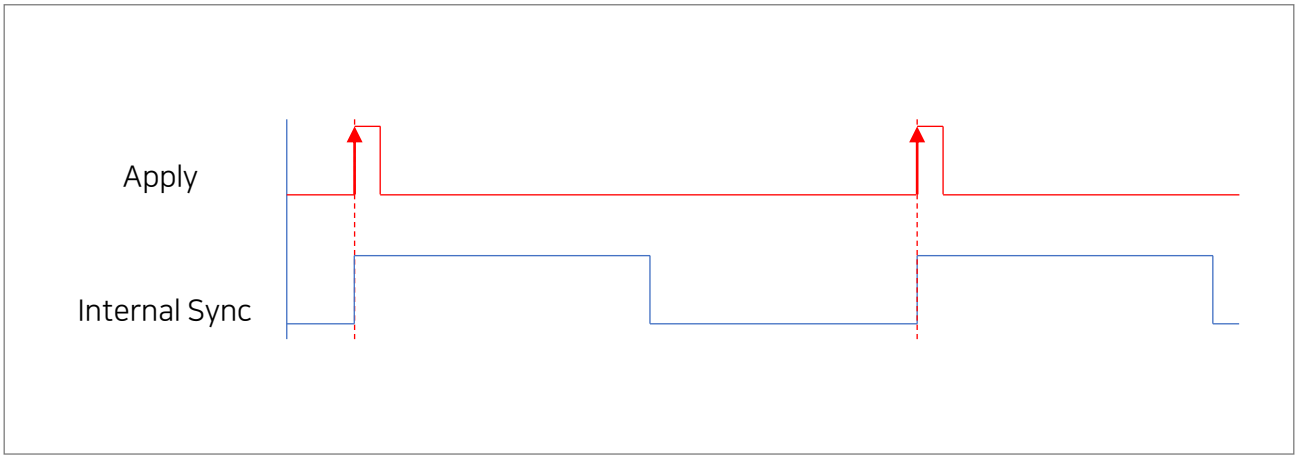


4) CAPUTE (비동기)

- Sync Mode를 Capture 선택 후 Apply 클릭하면 Capture Mode로 작동합니다



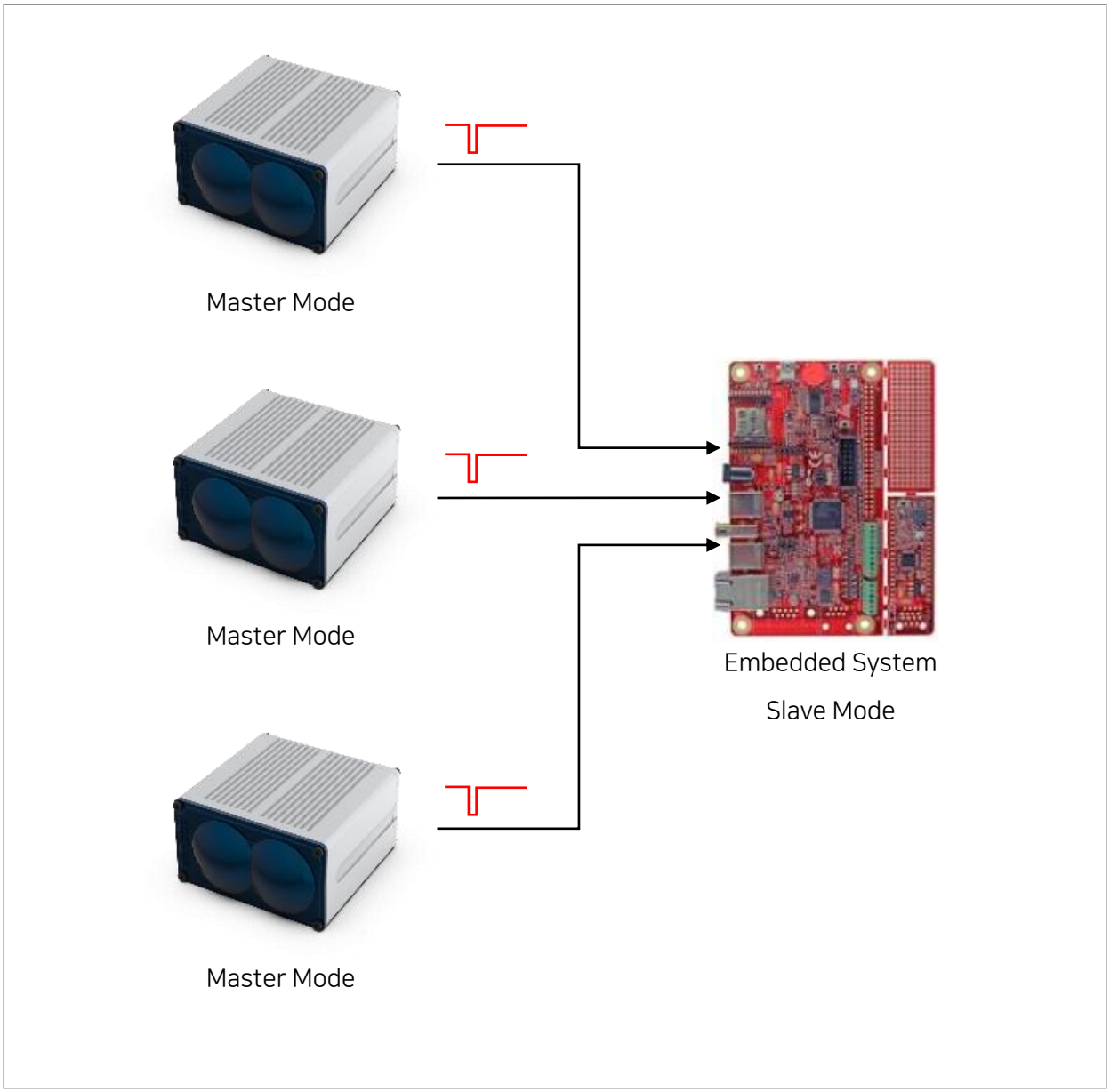
- Apply 누를 때 마다 Internal Sync를 1회만 발생시켜 Scene을 Update 합니다.



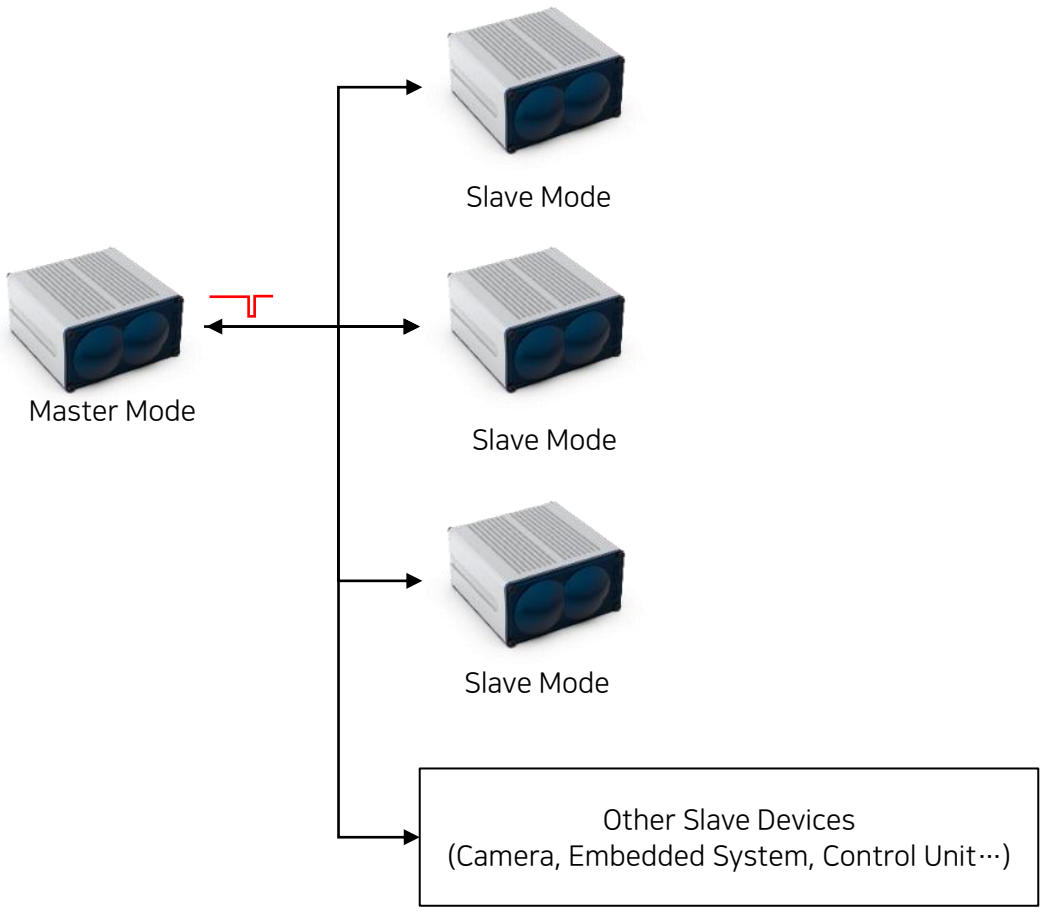
1.2.4.5 APPLICATION

입/출력 동기화 신호를 사용하여 다수의 ML-X 또는 기타 장치들과 동기화된 구성을 할 수 있습니다.

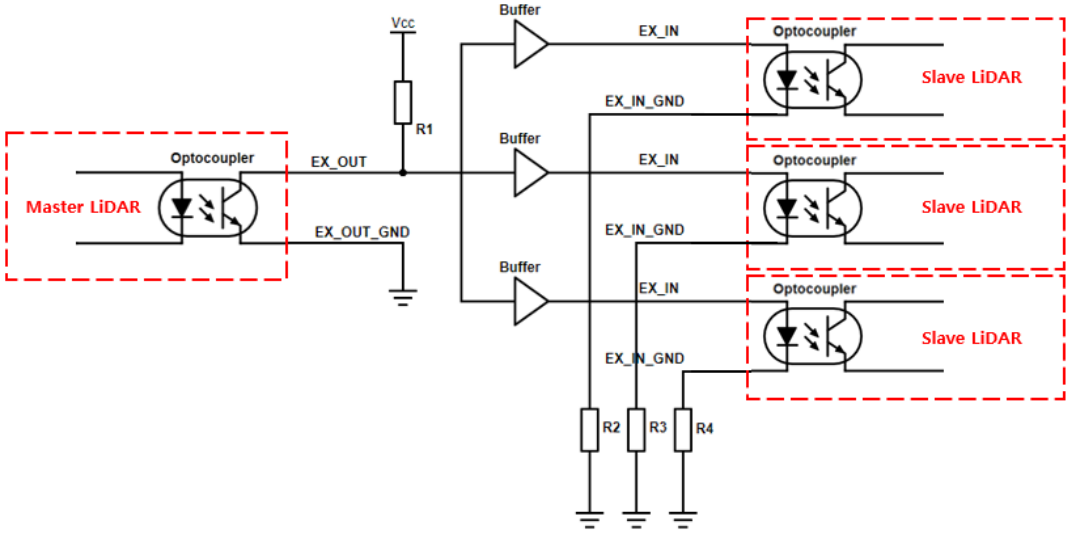
1) Multiple Master ML-X → Single Slave Device



2) Single Master ML-X → Multiple Slave Device

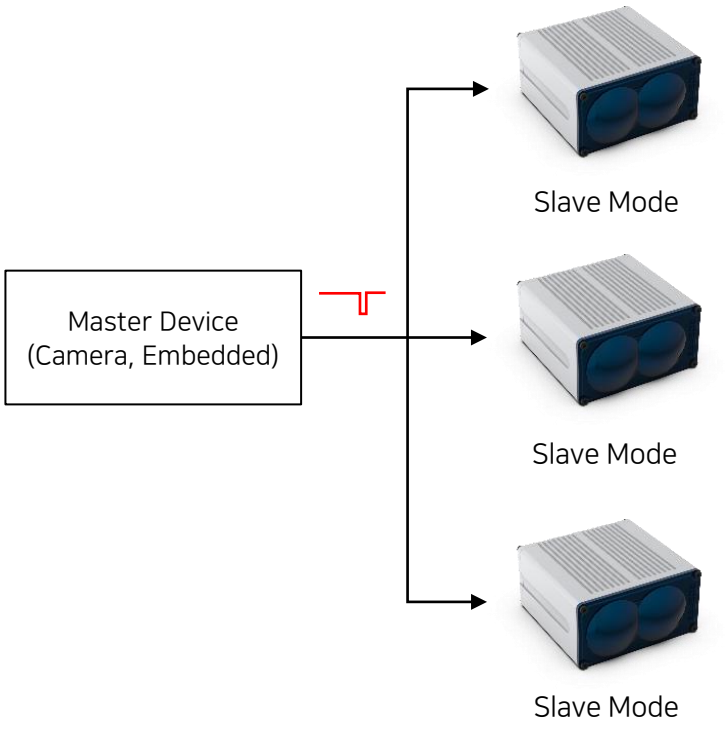


- External Circuit 구성

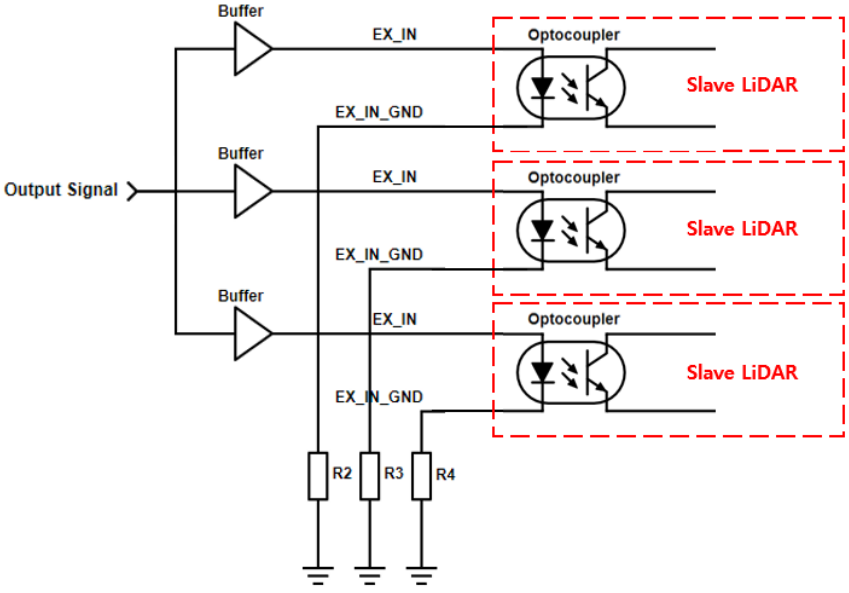


1.2. Electrical Interface

3) Single Master Device → Multiple Slave Device



- External Circuit 구성

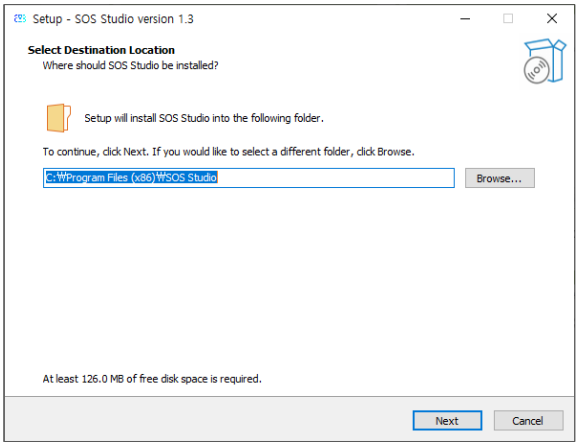


Chapter 2

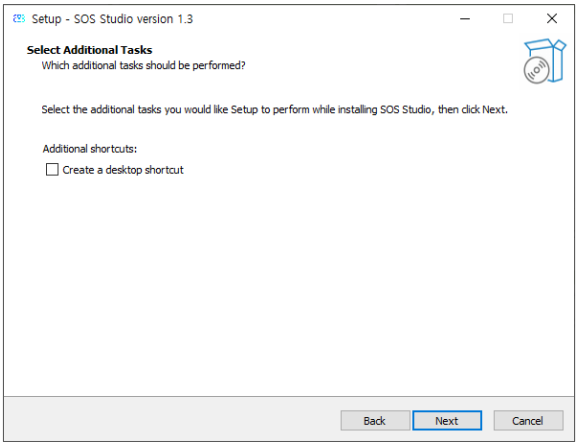
SOS Studio

2.1 Installation

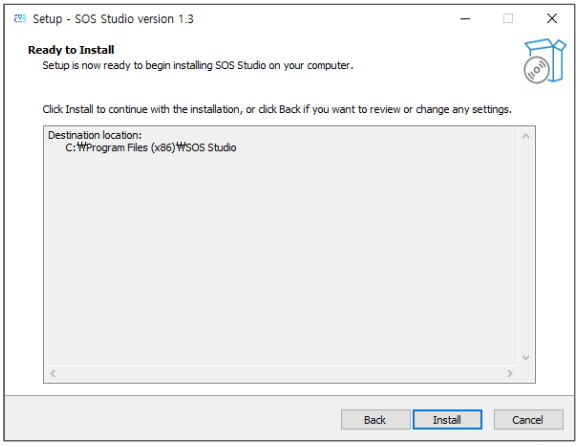
SOS Studio 설치를 진행합니다. “SOS Studio_setup.exe” 설치파일을 실행하여 설치를 진행합니다. 설치 순서는 다음과 같습니다.



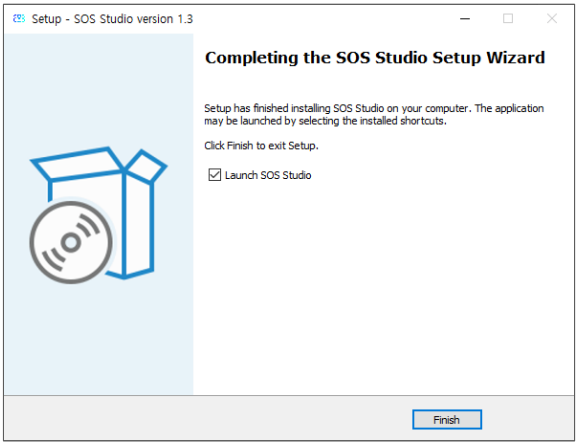
(a) 설치 파일 경로 설정



(b) 바탕화면 바로가기 설정



(c) SOS Studio 설치 시작



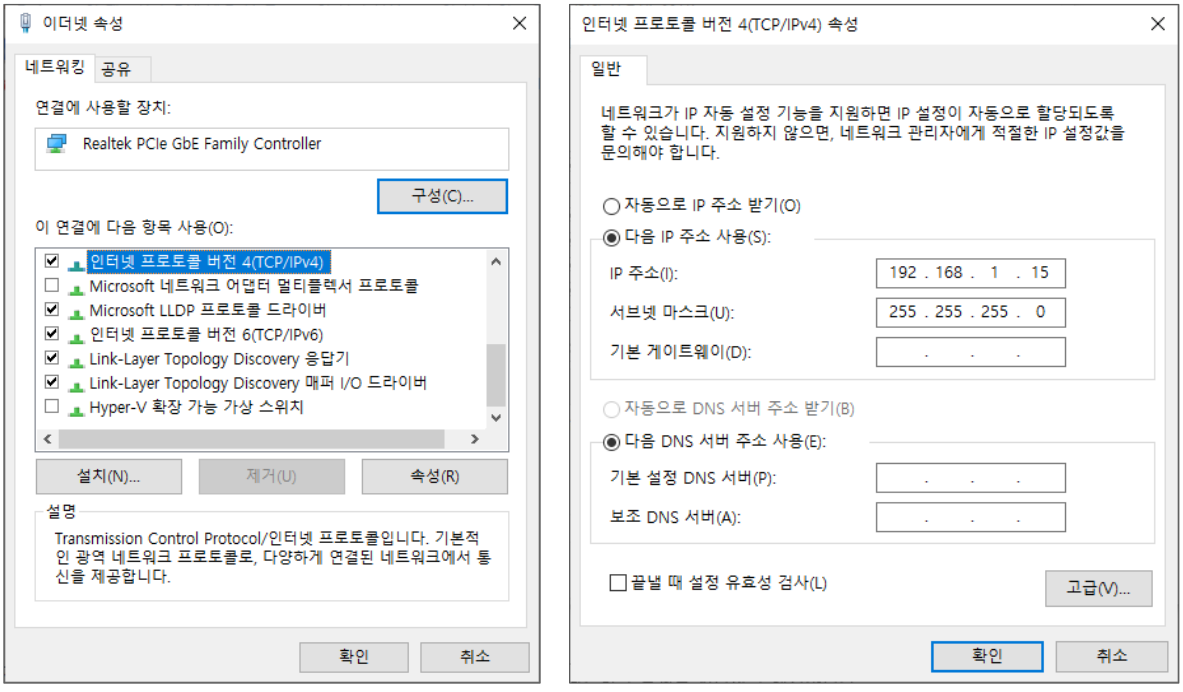
(d) SOS Studio 설치 완료

설치가 완료되면 SOS Studio가 실행됩니다.

2.2 TCP/IP Setting

ML-X LiDAR와 PC의 연결을 위해 TCP/IP 설정을 진행 합니다.

1) ML-X 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.

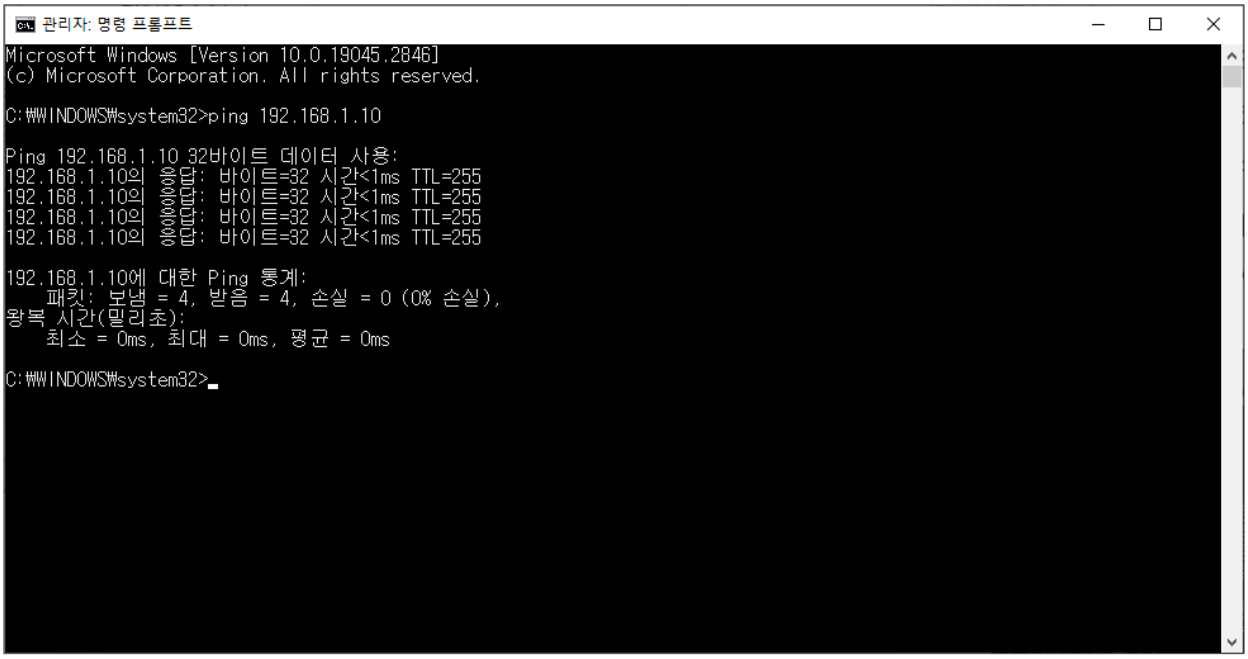


이더넷 속성 창 / 인터넷 프로토콜 버전4(TCP/IPv4) 속성 창 설정

2.2 TCP/IP Setting

케이블 연결과 IP 설정이 완료되면 아래의 Ping 테스트 과정을 통해 PC와 ML-X LiDAR와 PC가 정상적으로 연결되었는지 확인이 가능합니다.

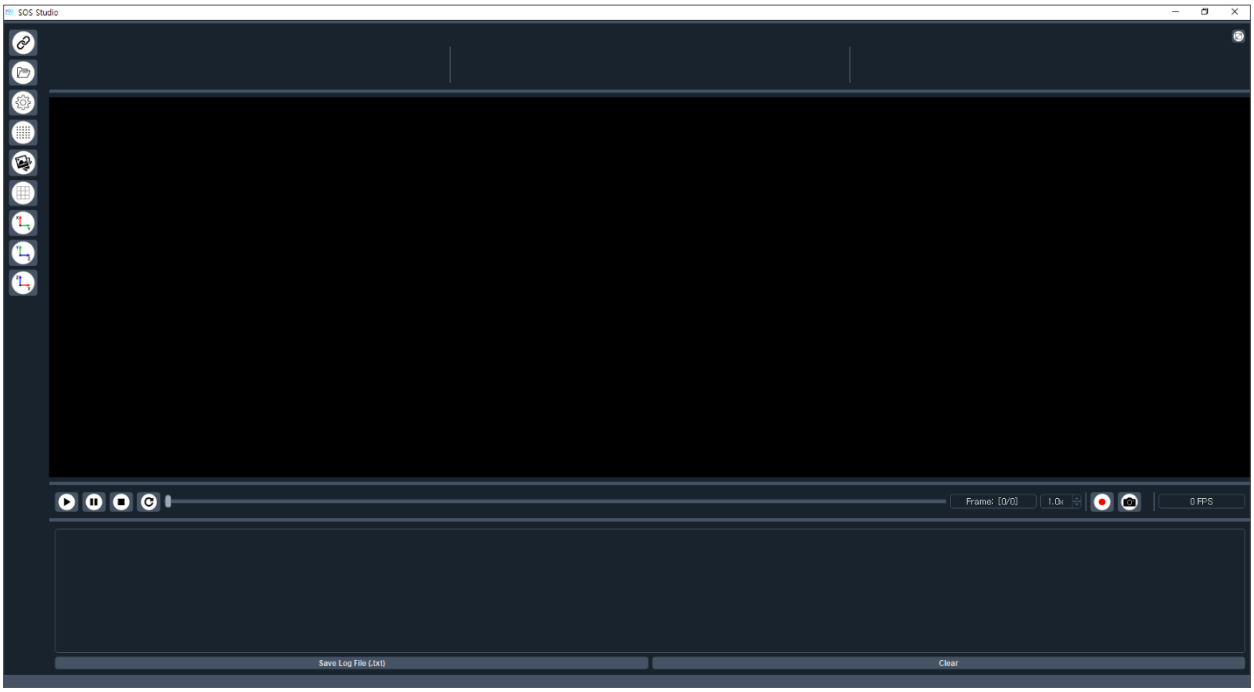
1) 윈도우 검색창에 'cmd' 명령어를 입력하여 명령 프롬프트를 실행합니다.
2) 명령어에 'ping 192.168.1.10'를 입력합니다.
3) ML-X LiDAR와 PC가 정상적으로 연결되었다면 아래와 같은 메시지가 출력됩니다.



명령 프롬프트 창 및 ping 테스트 성공 결과 예시

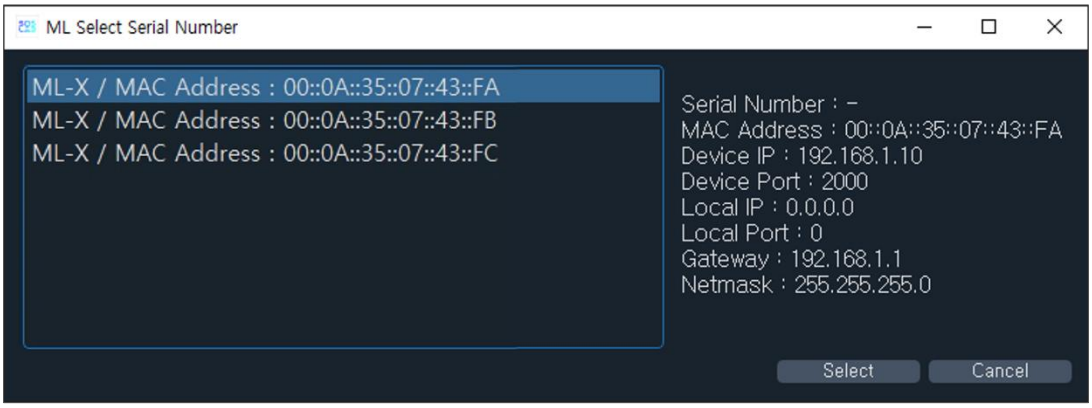
2.3 Connection

아래는 SOS Studio 실행화면 입니다.



SOS Studio 실행화면

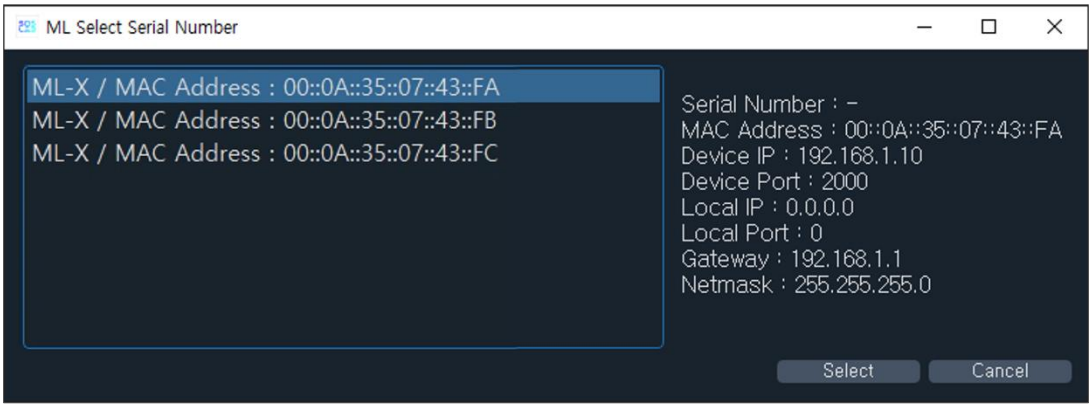
PC와 ML-X의 연결을 위해 SOS Studio 왼쪽 1번째 버튼  “Connection”을 클릭하면 연결 가능한 ML-X List가 나옵니다.



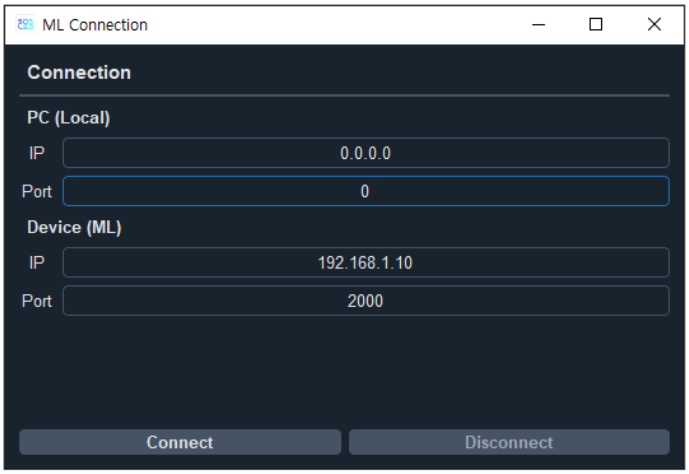
ML-X list

2.3 Connection

ML-X List에서 연결하고자 하는 ML-X를 선택한 후 “Select” 버튼을 누르면 선택된 ML-X의 IP와 Port가 Connection 팝업 창에 자동으로 업데이트 됩니다. Connection 팝업 창의 “Connect” 버튼을 클릭하면 PC와 ML-X Device를 연결할 수 있습니다.



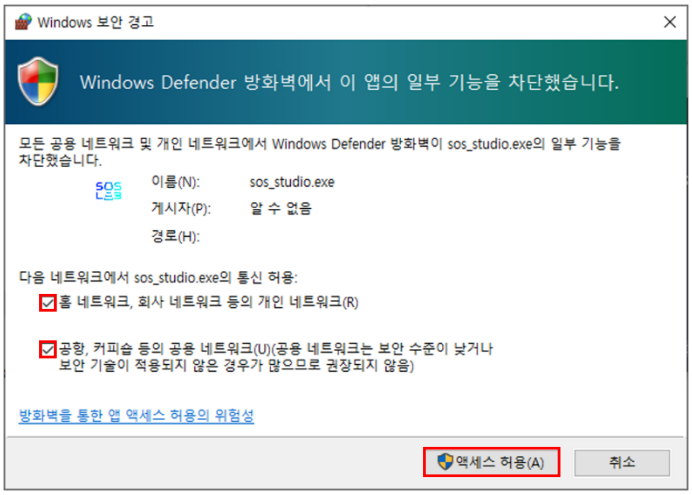
ML-X List



Connection 팝업 창

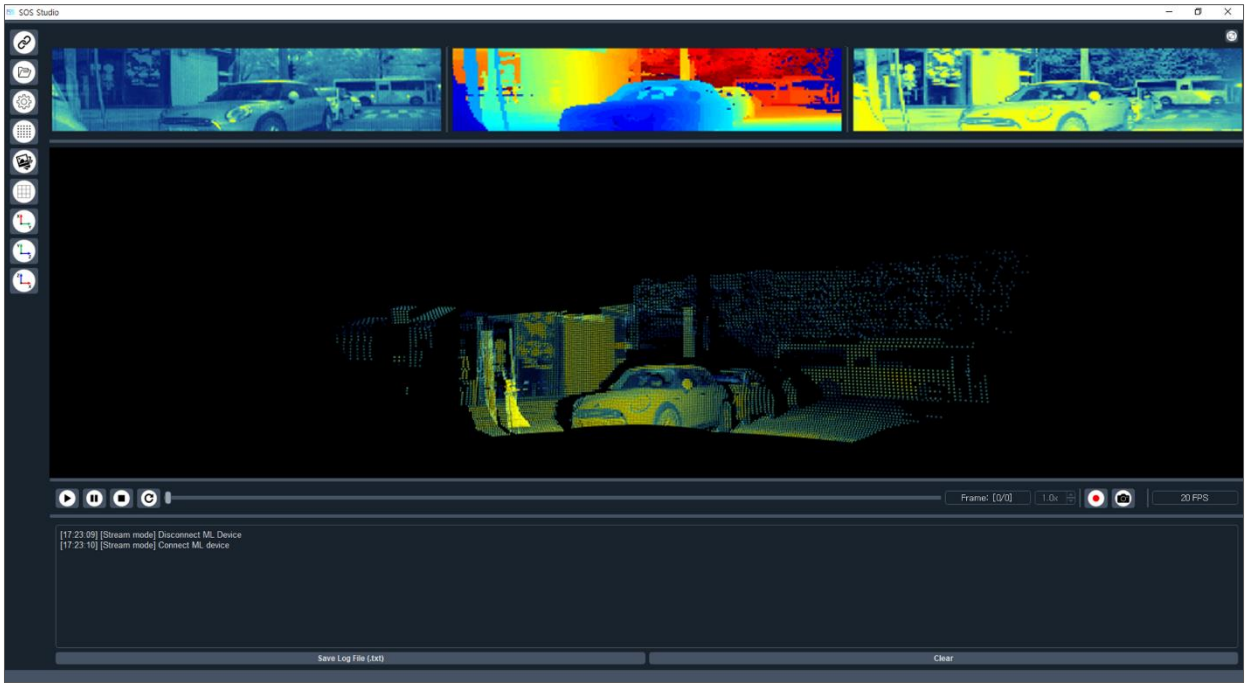
2.3 Connection

PC와 ML-X 처음 연결할 경우, 아래와 같이 Windows 보안 경고 창이 나타납니다. 다음 네트워크에서 SOS Studio ML-X.exe의 통신 허용 아래의 2개의 체크 박스를 아래의 그림과 같이 체크 한 뒤, "액세스 허용(A)" 버튼을 클릭합니다.



Window 보안 경고 화면

PC와 ML-X의 연결이 완료되면, 아래의 그림과 같이 상단의 Image Viewer와 중앙의 Point Cloud Viewer가 활성화 됩니다.

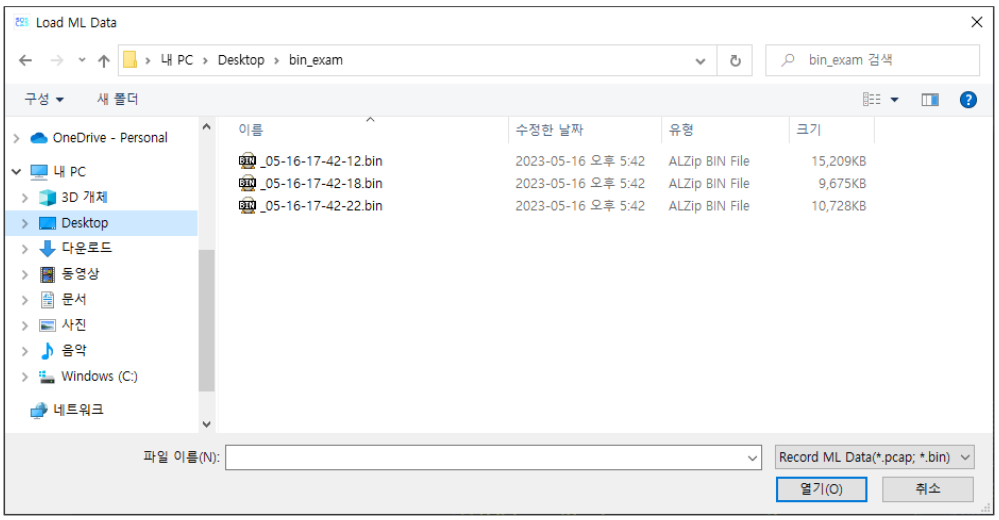


SOS Studio 실행화면

2.4 Data Load

Data Load 기능은 저장된 ML-X 데이터를 불러올 때 사용하는 기능입니다. 데이터 저장 및 불러온 데이터를 재생하는 기능은 **2.10 Play / Record Mode**에 기술되어 있습니다.

ML-X 데이터를 불러오기 위해 SOS Studio 왼쪽 2번째 버튼 “Data Load”를 클릭하면 아래와 같이 파일을 선택할 수 있는 창이 나타납니다. 이 후, 저장된 ML-X 데이터 (.bin) 파일을 선택한 뒤, “열기(O)” 버튼을 클릭하면 ML-X 데이터를 재생할 준비가 완료됩니다.



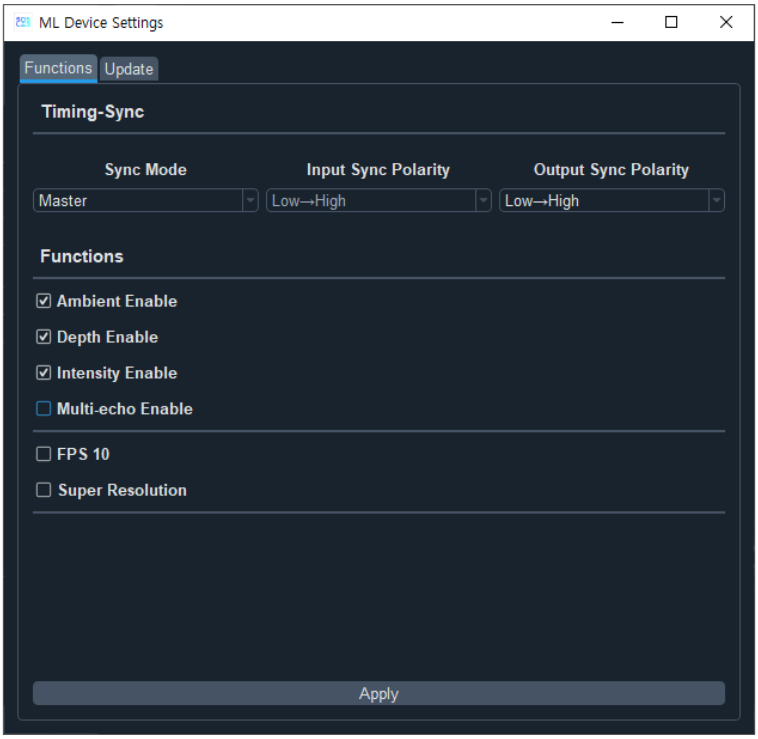
데이터 불러오기 팝업 창

데이터 불러오기가 완료되면 아래의 Play Bar 기능들을 통해 ML-X 데이터를 재생할 수 있습니다. Play Bar 기능들은 **2.10 Play / Record Mode**에 자세히 기술되어 있습니다.

2.5 Device Setting

Device Setting에서는 ML-X Device의 Timing Sync 기능, Data Enable, FPS 10, Super Resolution on/off 기능과 IP 변경 기능을 제공합니다.

Timing Sync 기능과 Function on/off 기능 등은 “Functions” 탭에서 제공하며, IP 변경 기능과 기능은 “Update” 탭에서 제공합니다.

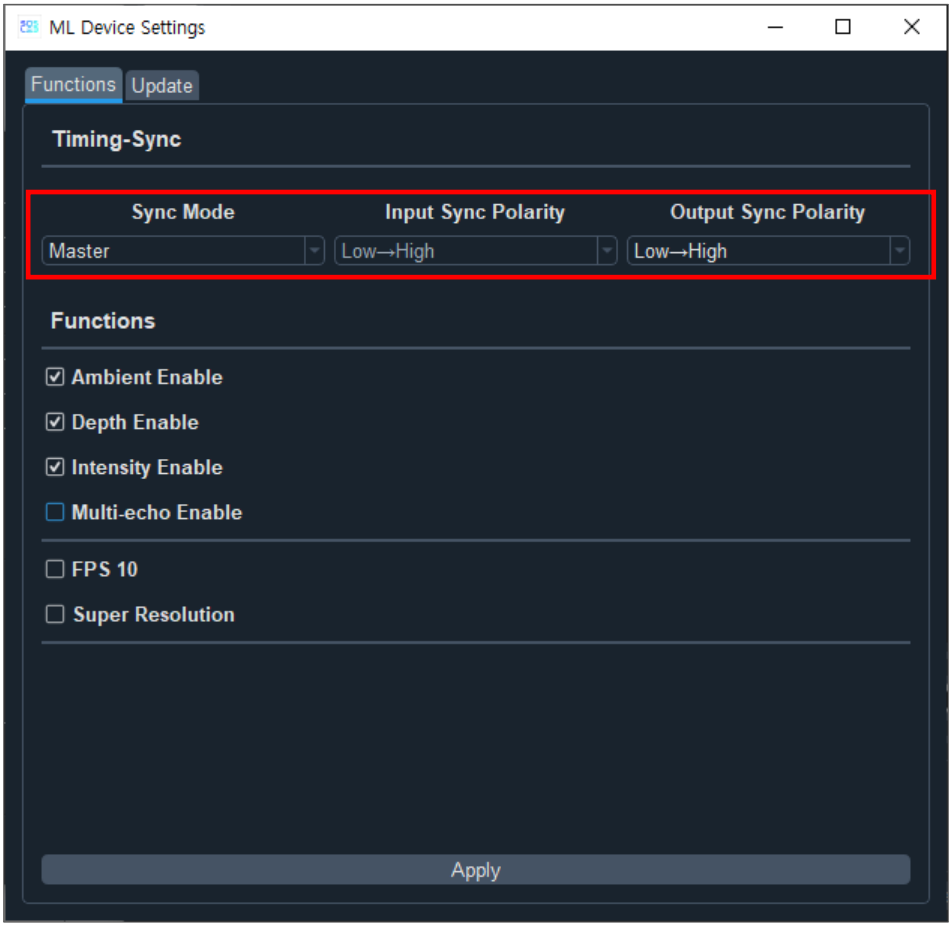


Device Setting 팝업 창

2.5.1 Timing Sync

ML-X Device의 Timing Sync 기능은 “Device Setting” – “Functions” 탭에서 제공합니다. Timing Sync에서는 Master 모드, Slave 모드, Capture 모드를 지원합니다. Master 모드에서는 Output Sync Polarity를 설정할 수 있으며, Slave 모드에서는 Input Sync Polarity를 설정할 수 있습니다. 원하는 모드와 Polarity를 설정한 후에 “Apply” 버튼을 누르면 해당 모드로 설정됩니다.

Timing Sync에 대한 자세한 설명은 User Guide p.9 ~ 20을 참조하시기 바랍니다.



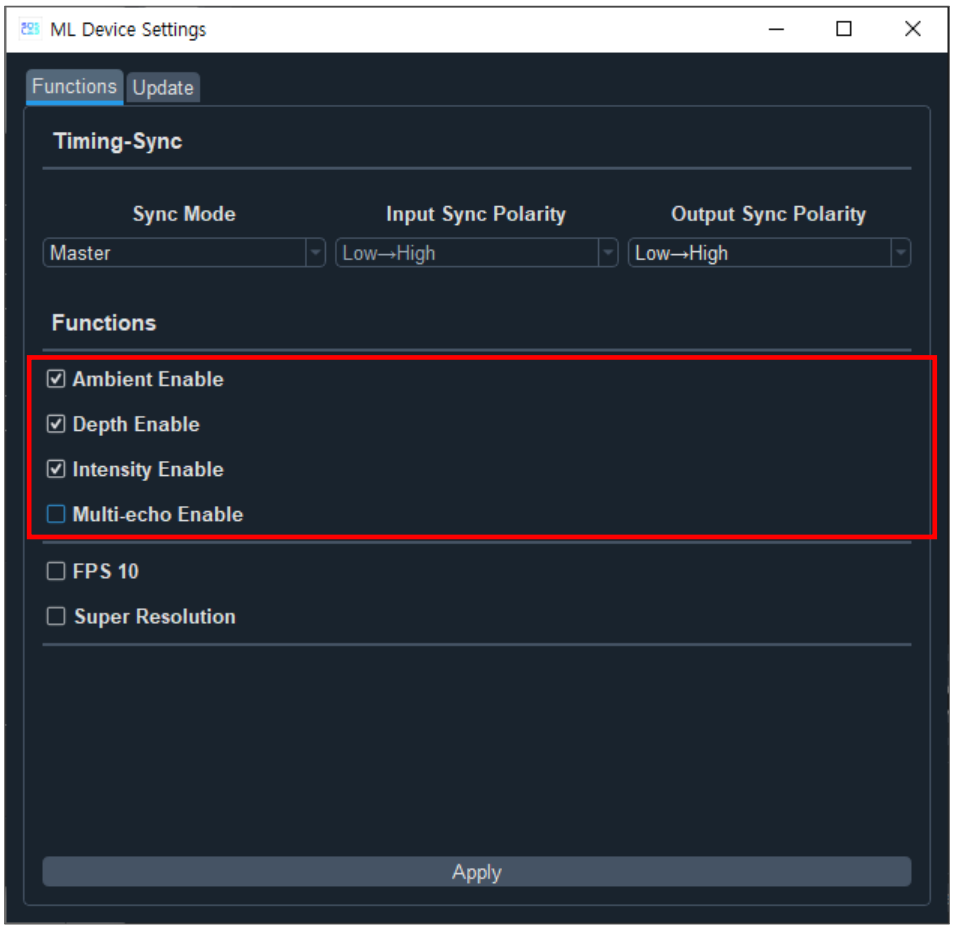
Device Setting 팝업 창 – Timing sync 기능

2.5.2 Functions on/off

ML-X Device의 Ambient / Depth / Intensity / Multi-echo Enable 기능들의 on/off는 “Device Setting” – “Functions” 탭에서 제공합니다.

각 기능들의 체크 박스 버튼을 선택한 상태에서 “Apply” 버튼을 누르면 해당 데이터를 전송합니다. 반대로, 체크 박스 버튼을 선택하지 않은 상태에서 “Apply” 버튼을 누르면 해당 데이터가 전송되지 않습니다. Functions에 대한 자세한 설명은 User Guide p.61을 참조하시기 바랍니다.

Functions	Default Setting
Ambient Enable	On
Depth Enable	On
Intensity Enable	On
Multi-echo Enable	Off



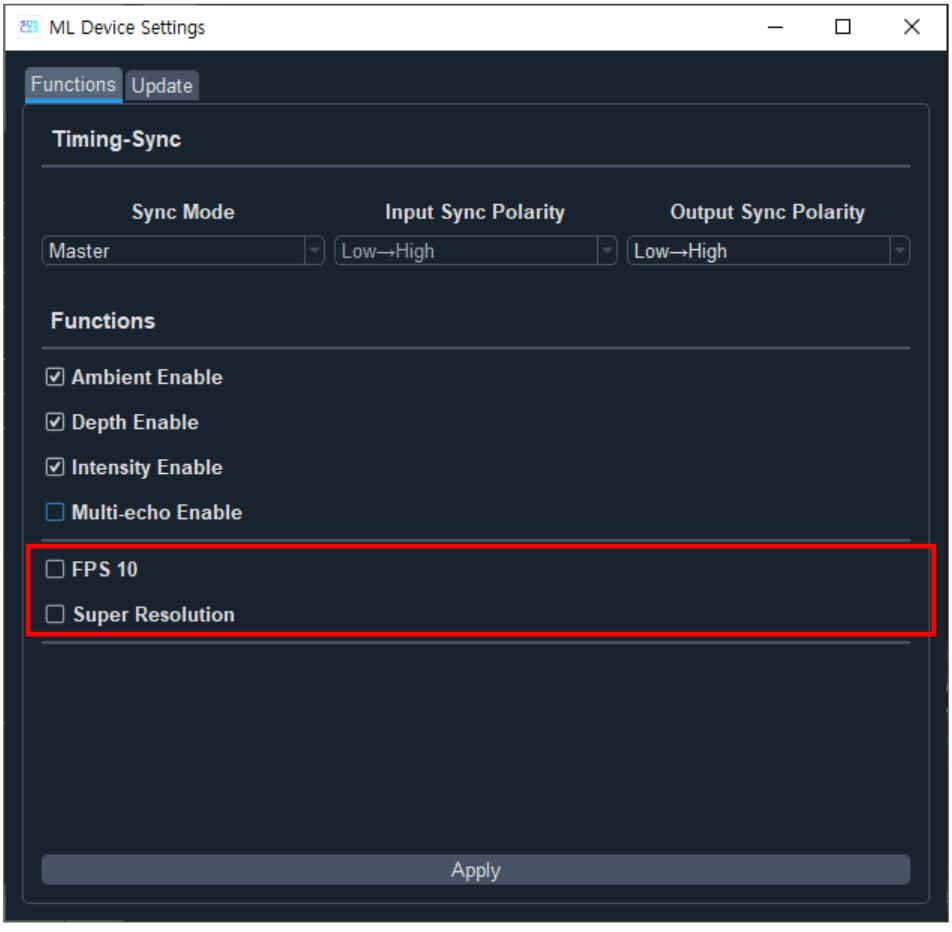
Device Setting 팝업 창 – Functions On/Off 기능

2.5.2 Functions on/off

ML-X Device의 FPS 10 제어 / Super Resolution 기능들의 on/off는 “Device Setting” – “Functions” 탭에서 제공합니다.

각 기능들의 체크 박스 버튼을 선택한 상태에서 “Apply” 버튼을 누르면 기능이 on 됩니다. 반대로, 체크 박스 버튼을 선택하지 않은 상태에서 “Apply” 버튼을 누르면 기능이 off됩니다. Functions에 대한 자세한 설명은 User Guide p.60을 참조하시기 바랍니다.

Functions	Default Setting
FPS 10	Off
Super Resolution	Off



Device Setting 팝업 창 – Functions On/Off 기능

2.5.3 IP Changer

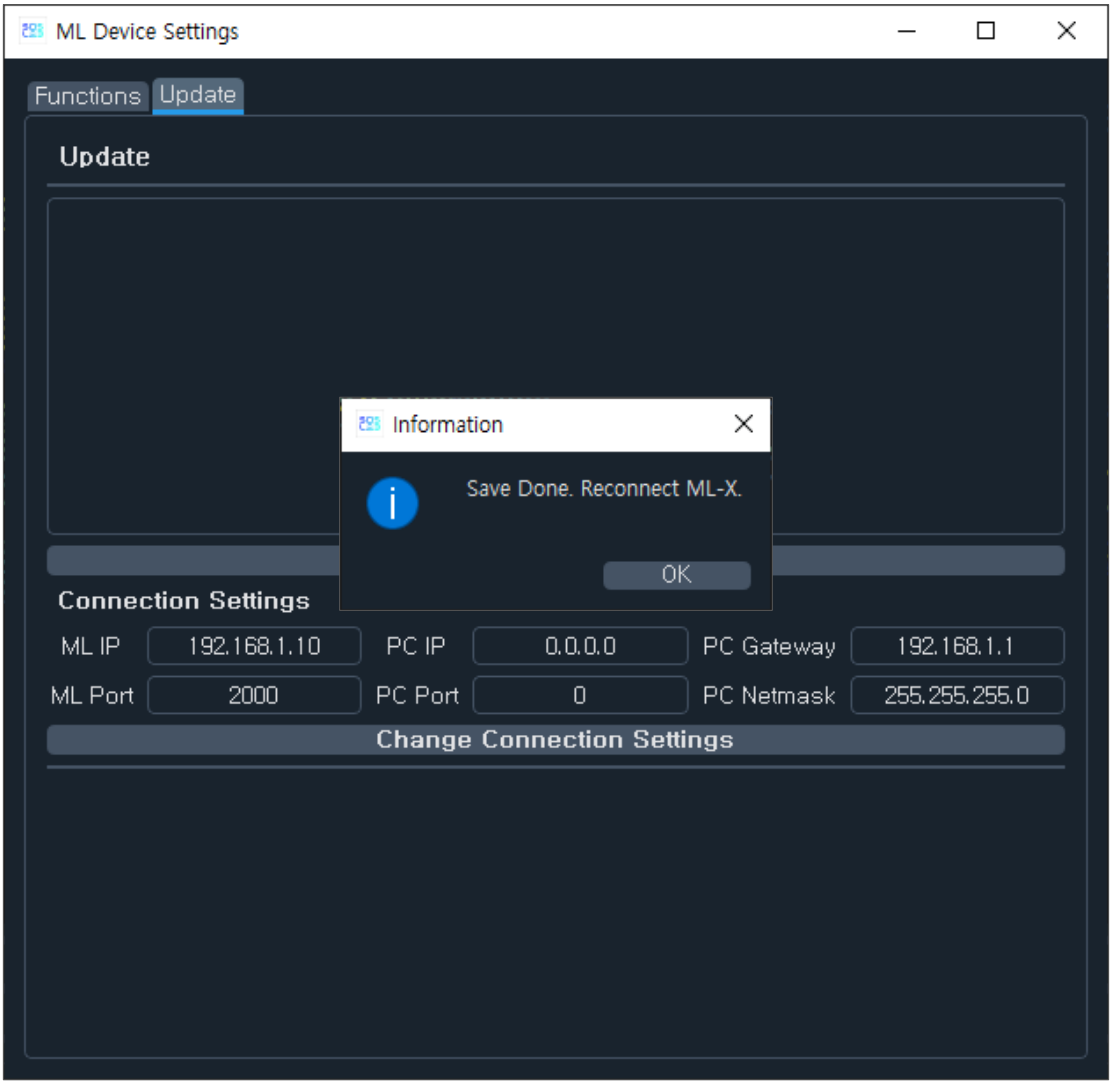
ML-X Device의 IP 변경 기능은 “Device Setting” – “Update” 탭에서 제공합니다.

ML-X Device의 IP 변경은 아래의 과정들을 통해 수행할 수 있습니다.

① 변경 될 ML-X Device의 IP 입력

② “Change Connection Settings” 버튼 클릭

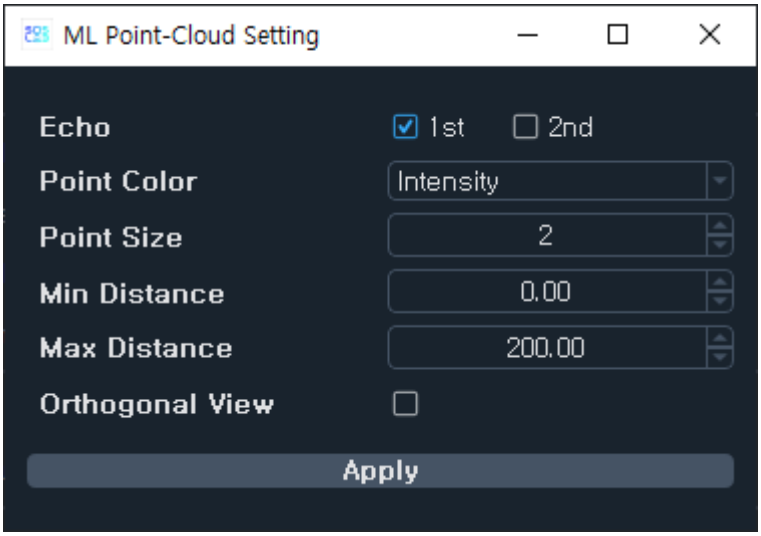
③ 변경 후, 전원 케이블 재연결



IP 변경 결과 화면

2.6 Point Cloud Setting

Point Cloud Setting에서는 SOS Studio 중앙에 위치한 Point Cloud Viewer를 설정할 수 있습니다. SOS Studio 왼쪽 4번째 버튼 “Point Cloud Setting”을 클릭하면 아래와 같은 팝업 창이 나타납니다.



Point Cloud Setting 팝업 창

설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

항목	설명
Echo	가시화 되는 Echo 선택 (Multi-echo 활성화 시 생성)
Point Color	Point 색상 (White, Red, Green, Blue, Ambient, Depth, Intensity)
Point Size	Point 크기
Min Distance	가시화 되는 Point 최소 거리
Max Distance	가시화 되는 Point 최대 거리
Orthogonal View	Orthogonal View 활성화

2.7 Image Setting

Image Setting에서는 SOS Studio 상단에 위치한 Image Viewer를 설정할 수 있습니다.

SOS Studio 왼쪽 5번째 버튼 “Image Setting”을 클릭하면 아래와 같은 팝업 창이 나타납니다.

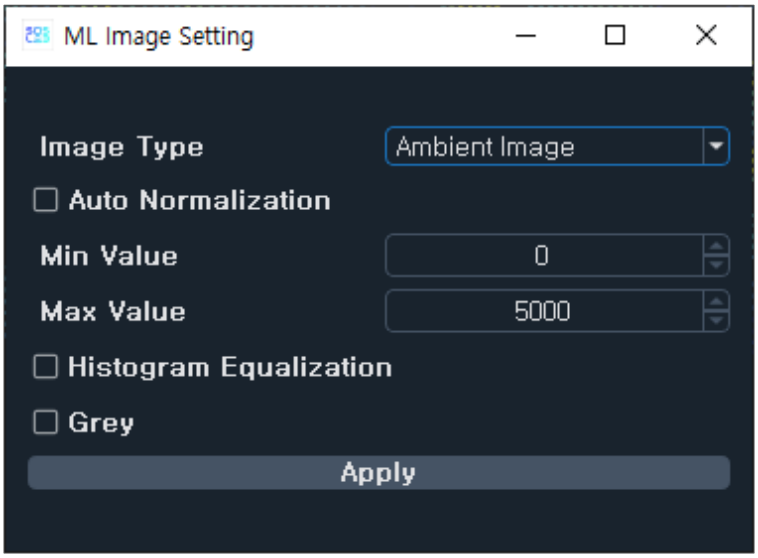


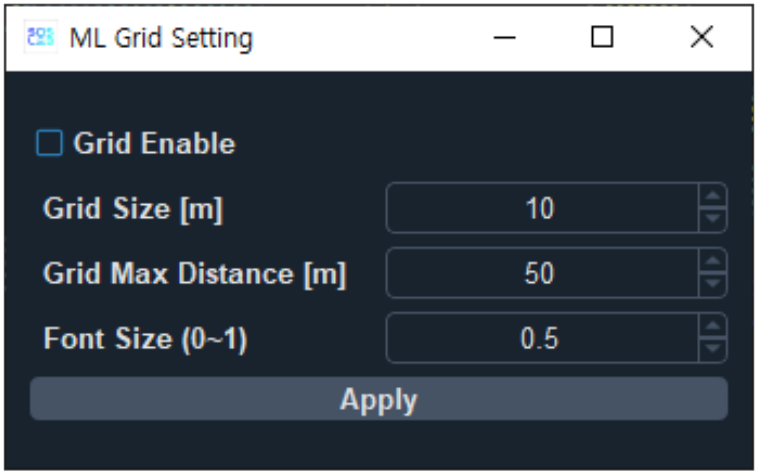
Image Setting 팝업 창

설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

항목	설명
Image Type	설정 이미지 선택 (Ambient / Depth / Intensity Image)
Auto Normalization	자동 정규화(Normalization): 이미지의 최소/최대 값으로 정규화
Min Value	정규화 최소 값
Max Value	정규화 최대 값
Histogram Equalization	Histogram Equalization 활성화
Grey	흑백 이미지로 변경

2.8 Grid Setting

Grid Setting에서는 SOS Studio SOS Studio 중앙에 위치한 Point Cloud Viewer의 Grid를 설정할 수 있습니다. SOS Studio 왼쪽 6번째 버튼 “Grid Setting”을 클릭하면 아래와 같은 팝업 창이 나타납니다.



Grid Setting 팝업 창

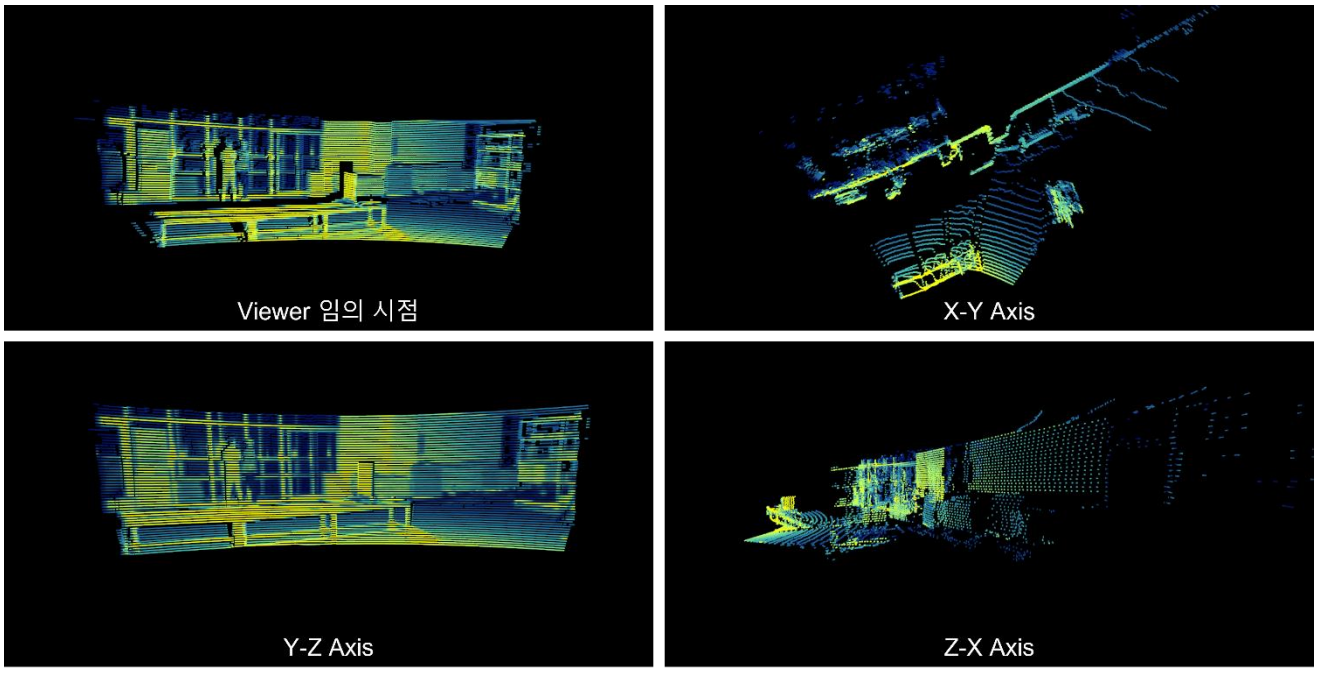
설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

항목	설명
Grid Enable	Grid 가시화 활성화
Grid Size	Grid 간격
Grid Max Distance	Grid 최대 거리
Font Size	Grid 거리[m] 단위 표시 글자 크기

2.9 X-Y / Y-Z / Z-X Axis

 X-Y /  Y-Z /  Z-X Axis 은 SOS Studio 중앙에 위치한 Point Cloud Viewer의 시점을 해당 Axis에 맞게 변경하는 기능을 제공합니다.

해당 기능들을 각각 선택하면 아래와 같이 Point Cloud Viewer의 임의의 시점에서 해당 Axis에 맞게 변경됩니다.



X-Y / Y-Z / Z-X Axis 버튼에 따른 Point Cloud 시점 변경

2.10 Play / Record Mode

Point Cloud Viewer 아래에 위치한 Play Bar는 **2.4 Data Load**에서 불러온 ML-X 데이터를 재생하는 기능 및 연결 후, 가시화 되는 데이터를 녹화하는 기능을 제공합니다.

SOS Studio Play Bar

Play Bar에 각 기능들에 명칭 및 기능은 아래와 같습니다.

기능	설명
Play	데이터 재생
Pause	데이터 재생 일시정지
Stop	데이터 재생 정지
Repeat	데이터 반복 재생
Scroll Bar	전체 프레임 중 현재 재생 중인 프레임 표시 (Bar 형태로 가시화)
Frame	전체 프레임 중 현재 재생 중인 프레임 표시 (현재/전체로 표시)
Speed	데이터 재생 속도
Record	스트리밍 되는 연속된 데이터(.bin) 저장
Capture	스트리밍 또는 재생되는 단일 데이터 저장 (Images, Point Cloud)
FPS	스트리밍 되는 데이터 전송속도

Chapter 3

ML-X LiDAR API

3.1. ML-X API C++ Code Build from Source

ML-X SDK는 Open Source Code로써, Github 링크를 통해 다운로드 받을 수 있습니다.
(Github link: https://github.com/SOSLAB-SS/MLX_LiDAR_SDK)

ML-X API 빌드에 필요한 환경은 아래와 같습니다.

- Window 10 / 11
- Ubuntu 18.04 / 20.04
- C++ 11 (GCC 5 / Visual C++ 13) or later
- CMake 3.10 or later

3.1.1. ML-X API C++ Building on Windows

아래 Command를 입력하여 Visual Studio Solution 파일을 생성할 수 있습니다.

```
$ git clone https://github.com/SOSLAB-SS/MLX_LiDAR_SDK.git
$ cd MLX_LiDAR_SDK/MLX_API/
$ cmake CMakeLists.txt
```

해당 Command 입력을 하면 폴더에 “libsoslab.sln”이 생성됩니다. 해당 Solution을 실행하여 Build Type을 Release로 변경 후에 ALL_BUILD 프로젝트를 빌드합니다.
Build 완료 후에 output/Release 폴더에 “libsoslab_core.lib”, “libsoslab_core.dll”, “libsoslab_ml.lib”, “libsoslab_ml.dll”, “test_ml.exe” 파일이 생성됩니다.

3.1.2. ML-X API C++ Building on Linux

아래 Command를 입력하여 Code를 Build 할 수 있습니다.

```
$ git clone https://github.com/SOSLAB-SS/MLX_LiDAR_SDK.git
$ cd MLX_LiDAR_SDK/MLX_API/
$ cmake CMakeLists.txt -DCMAKE_BUILD_TYPE=Release
$ make
```

Build 완료 후에 output₩Release 폴더에 “libsoslab_core.so”, “libsoslab_ml.so”, “test_ml” 파일이 생성됩니다.

3.1.2. Running the Example Code

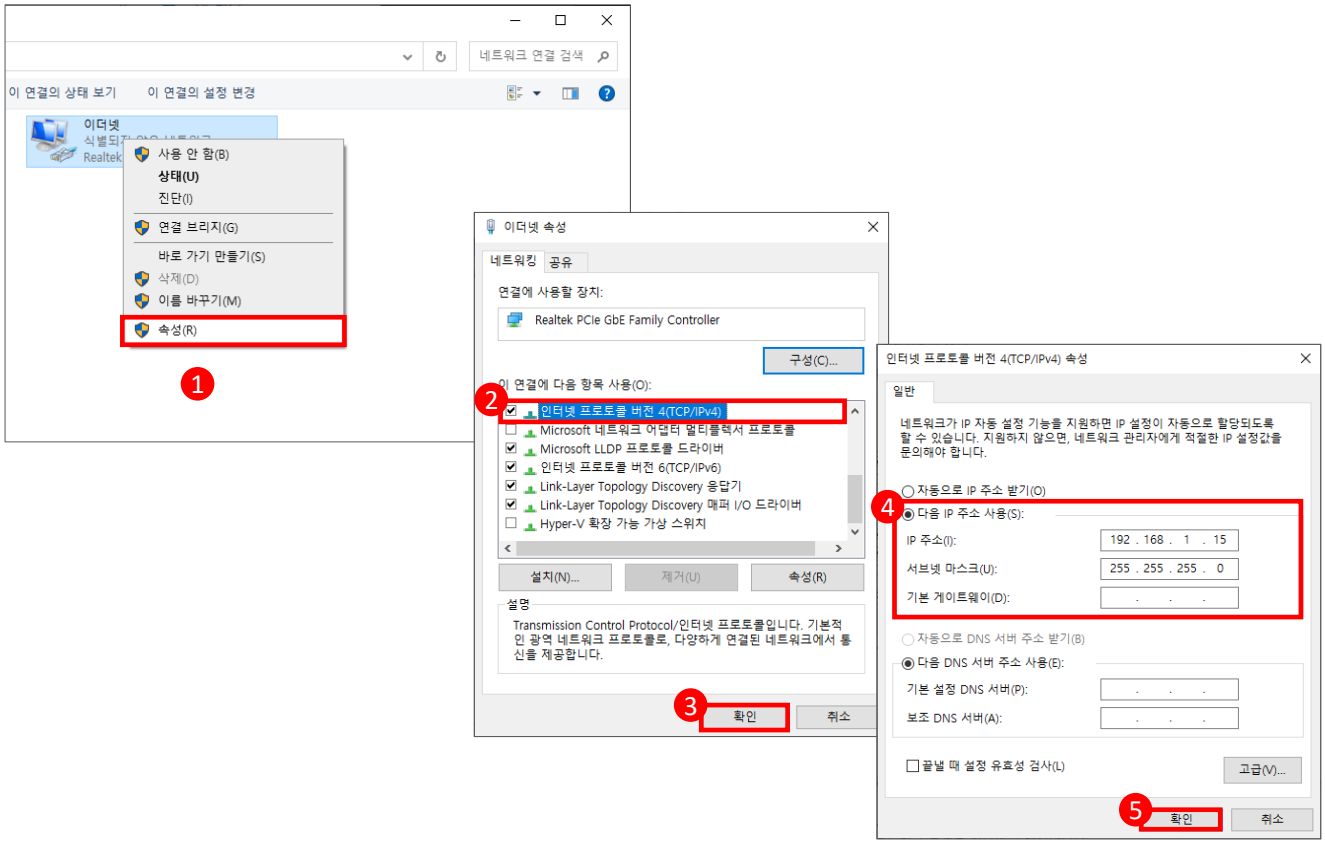
Source Code Build 후에 생성된 “test_ml” 프로그램을 실행하여 ML-X의 Point Cloud 데이터를 “.csv” 파일로 저장할 수 있습니다. 해당 프로그램을 실행하기 전, ML-X와 연결한 Host의 Network Setting을 해야 합니다.

3.1.2.1. Network Setting on Windows

제어판 – 네트워크 및 인터넷 – 네트워크 연결에서 ML-X와 연결된 이더넷을 우클릭 후 속성 버튼을 클릭합니다. 인터넷 프로토콜 버전 4(TCP/IPv4)을 클릭 후 속성 버튼을 클릭합니다. 다음 IP 주소 사용을 선택 후 다음과 같은 설정합니다.

- IP 주소 : 192.168.1.15
- 서브넷 마스크 : 255.255.255.0

확인 버튼을 클릭합니다.

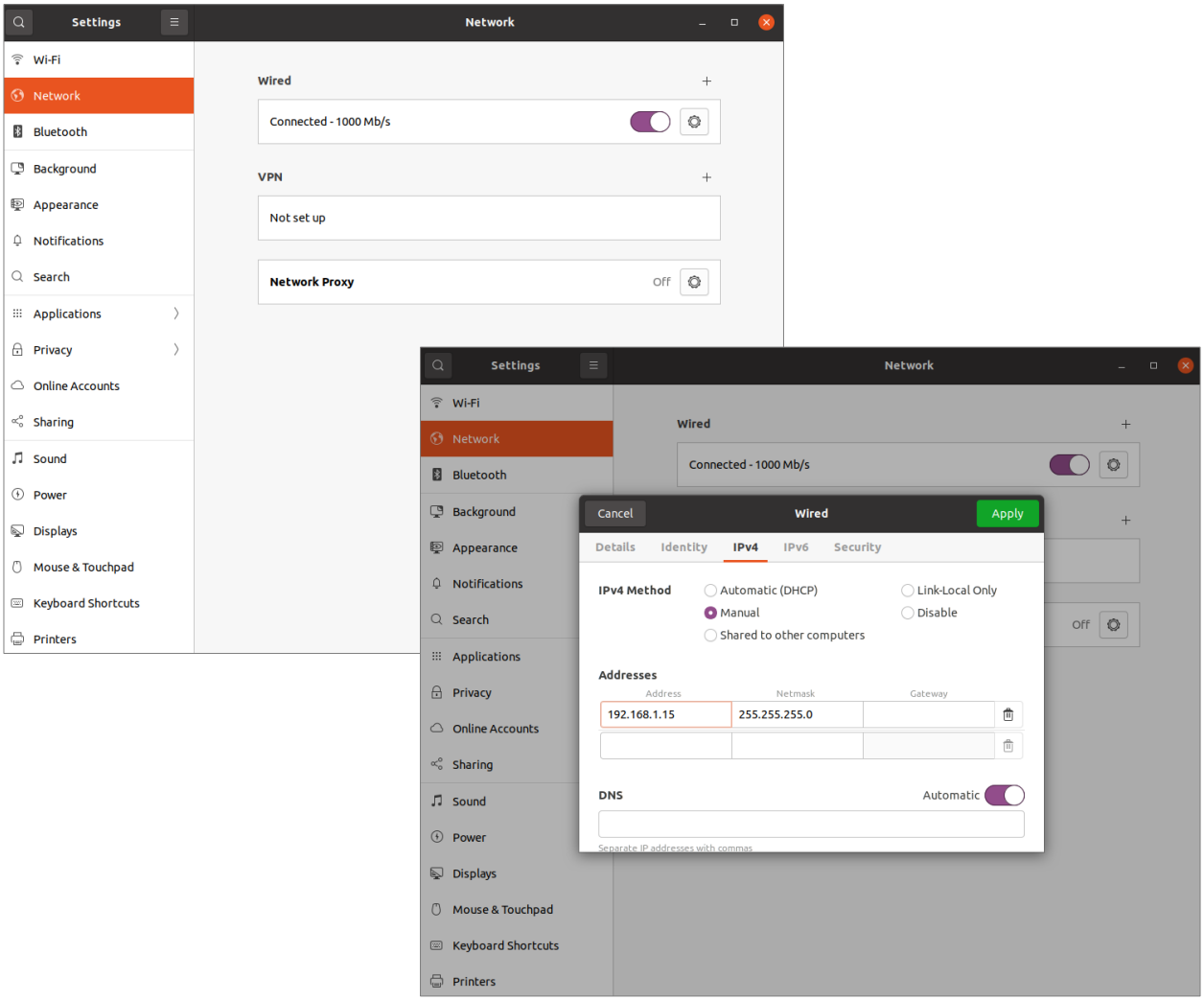


IPv4 설정 변경 @Windows

3.1.2.2. Network Setting on Linux

Setting창을 열어 Network 항목에서 아래와 같이 IPv4 설정을 변경합니다.

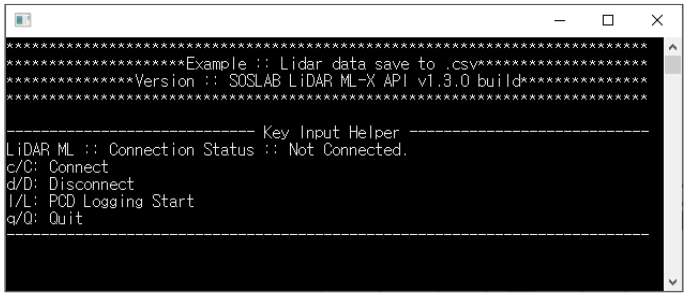
- IPv4 Method – Manual
- Address : 192.168.1.15
- Netmask : 255.255.255.0



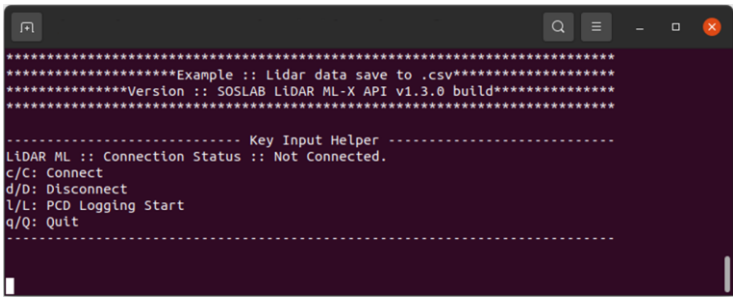
IPv4 설정 변경 @Linux

3.1.2.3. Running the Example Code

터미널(윈도우의 경우 cmd)을 열어 “test_ml”을 실행합니다. 해당 파일을 실행하면 아래와 같은 문구가 나타납니다.



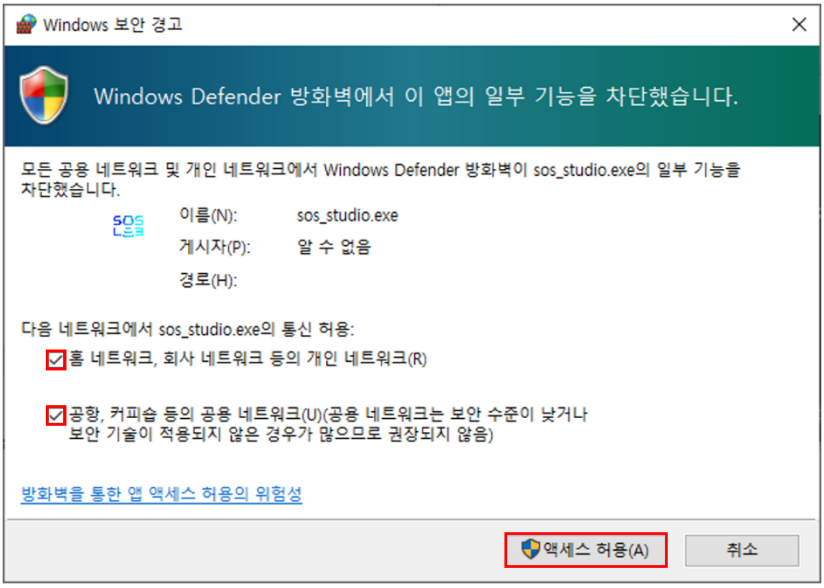
Windows



Linux

- LiDAR ML :: Connection Status :: Not Connected. → ML-X의 연결 상태를 나타냅니다.
- c/C + enter 입력 → ML-X를 연결합니다.
- d/D + enter 입력 → ML-X의 연결을 해제합니다.
- l/L + enter 입력 → ML-X의 PCD data를 .csv 파일로 저장합니다.
- q/Q + enter 입력 → 프로그램을 종료합니다.

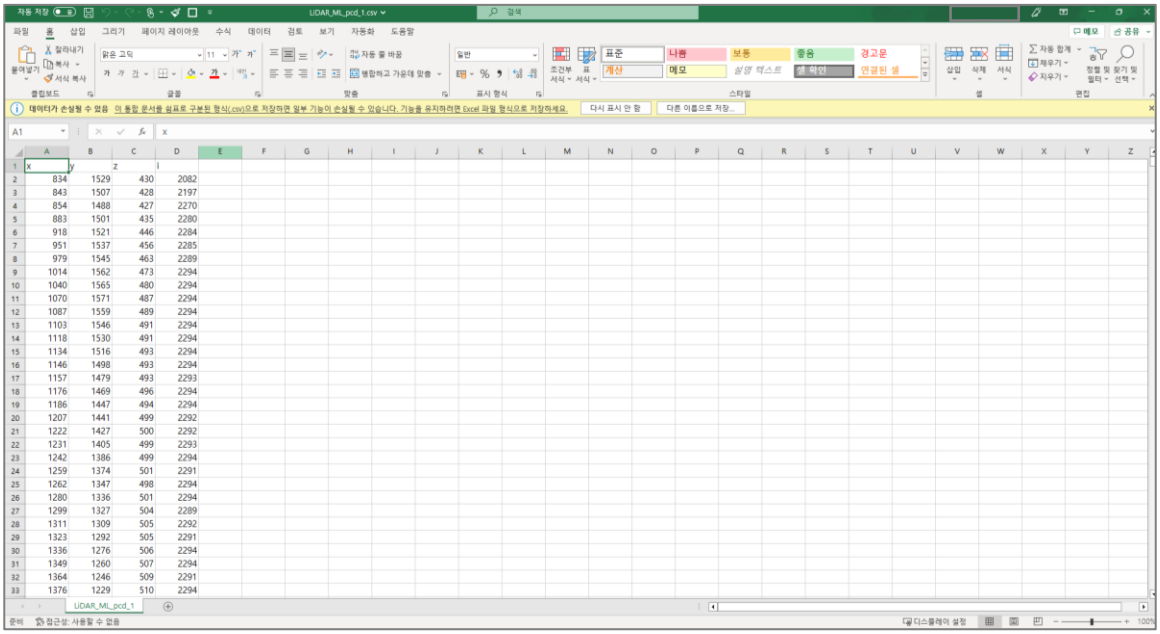
Windows의 경우. 처음 ML-X를 연결할 때 보안 관련된 경고창이 나타납니다. 해당 경고창에서 “개인 네트워크”와 “공용 네트워크”를 모두 선택 후 “액세스 허용” 버튼을 클릭합니다.



Window 보안 경고 화면

3.1. ML-X API C++ Code Build from Source

ML-X의 PCD Data를 '.csv'파일로 저장하게 되면 아래와 같은 형식으로 저장됩니다.



x	y	z	I
834	1529	430	2082
843	1507	428	2197
854	1488	427	2270
883	1501	435	2280
918	1521	446	2284
951	1537	456	2285
979	1545	463	2289
1014	1562	473	2294
1040	1565	480	2294
1070	1571	487	2294
1087	1559	489	2294
1103	1546	491	2294
1118	1530	491	2294
1134	1516	493	2294
1146	1498	493	2294
1157	1479	493	2293
1176	1469	496	2294
1186	1447	494	2294
1207	1441	499	2292
1222	1427	500	2292
1231	1405	499	2293
1242	1386	499	2294
1259	1374	501	2291
1262	1347	498	2294
1280	1336	501	2294
1299	1327	504	2289
1311	1309	505	2292
1323	1292	505	2291
1336	1276	506	2294
1349	1260	507	2294
1364	1246	509	2291
1376	1229	510	2294

첫 번째 열에는 x, 두 번째 열에는 y, 세 번째 열에는 z, 네 번째 열에는 Intensity 데이터가 저장됩니다.

3.2. ML-X ROS Example Code Build

ML-X ROS Example Code는 Ubuntu 18.04/20.04 및 ROS는 Melodic/Noetic 버전에서 사용 가능하며, ML-X를 ROS 환경에서 구동할 수 있는 코드를 제공합니다.

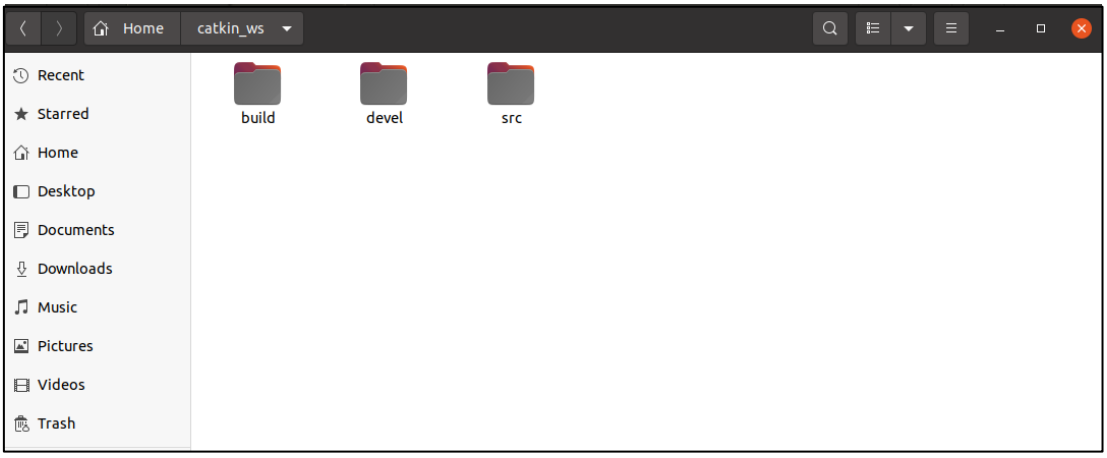
3.2. ML-X ROS Example Code Build

1) catkin_ws 폴더 위치에서 아래 Command를 입력하여 ml package를 생성합니다.

```
$ catkin_make
```

```
[100%] Linking CXX executable /home/soslab/catkin_ws/devel/lib/ml/ml
[100%] Built target ml
soslab@soslab-17U790-PA76K:~/catkin_ws$
```

catkin_make를 활용한 ml Package 생성



ml Package 빌드 결과

2) Build를 통해 생성된 ml package를 ROS 환경에 추가하기 위하여 아래 Command를 입력하여 ROS 환경에 ml package 추가합니다.

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

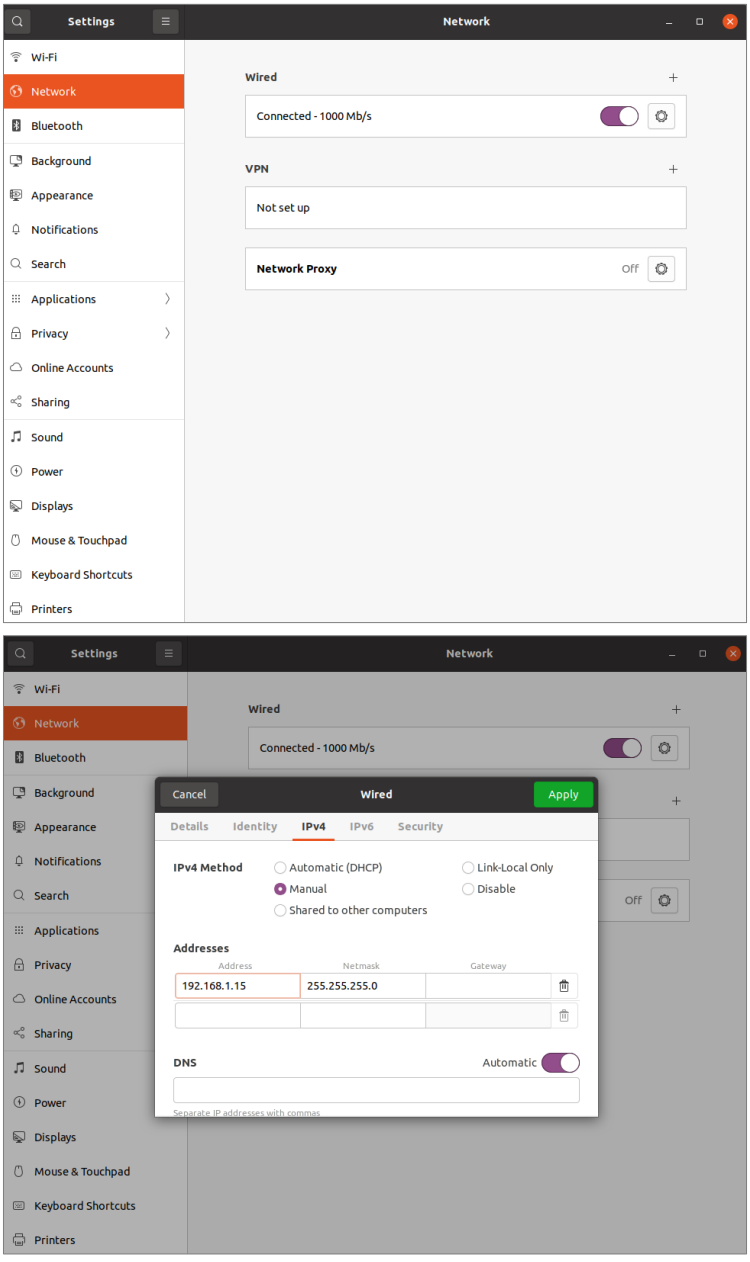
```
soslab@soslab-17U790-PA76K: ~/catkin_ws
soslab@soslab-17U790-PA76K:~/catkin_ws$ source ~/catkin_ws/devel/setup.sh
soslab@soslab-17U790-PA76K:~/catkin_ws$ rospack find ml
/home/soslab/catkin_ws/src/ml
soslab@soslab-17U790-PA76K:~/catkin_ws$
```

ml package를
ROS 환경에 추가

3.2. ML-X ROS Example Code Build

3) ML-X Device와 PC의 연결을 위해 Setting창의 Network 항목에서 아래와 같이 IPv4 설정을 변경합니다.

- IPv4 Method – Manual
- Address : 192.168.1.15
- Netmask : 255.255.255.0

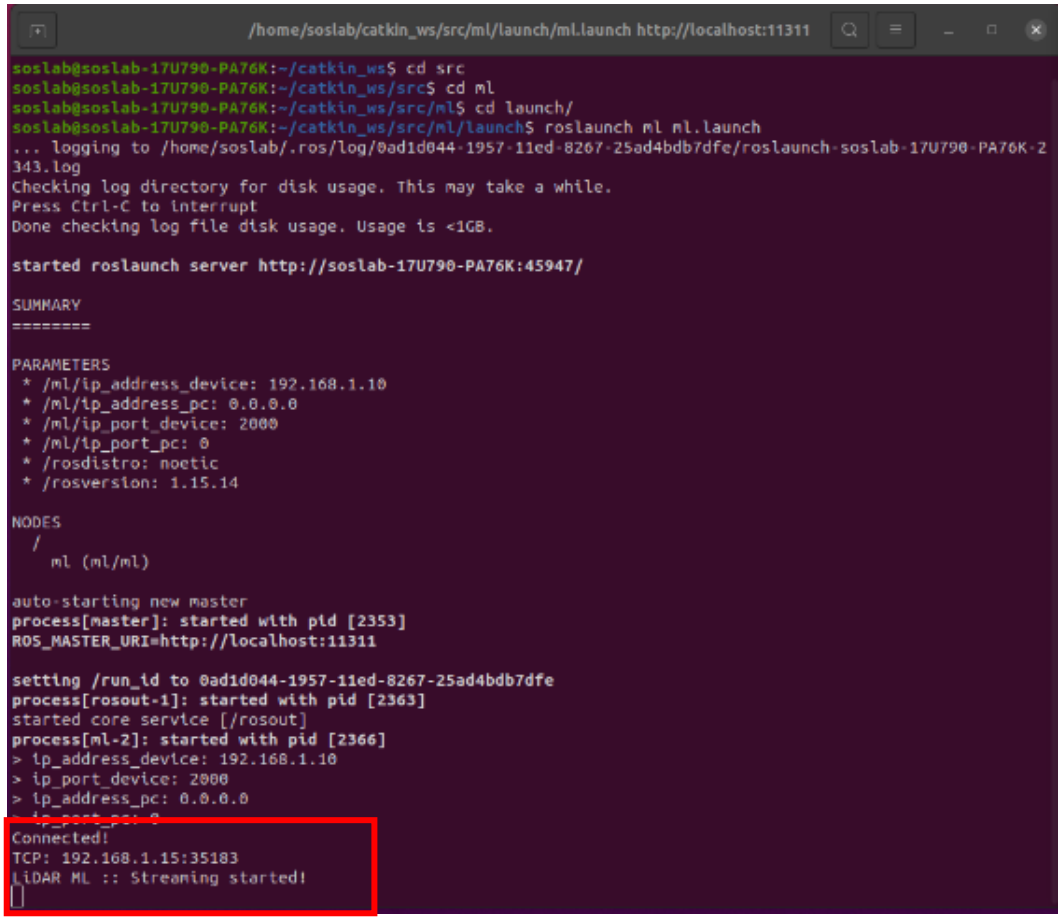


IPv4 설정 변경

3.2. ML-X ROS Example Code Build

4) Network 설정 후, 아래 command를 입력하여 ml.launch 파일을 실행하여 PC와 ML-X Device를 연결 합니다.
5) 정상적으로 ML-X Device와 연결되면 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-X Device 연결 및 가시화

3.3. Example Code for Ubuntu/ROS

예제 코드에는 ML-X 연결 및 Rviz에서의 데이터 가시화 기능이 구현되어 있으며, 코드 설명은 아래와 같습니다.

1) Raw Data 획득

```
1.  while (ros::ok()) {
2.    SOSLAB::LidarML::scene_t scene;
3.      if (lidar_ml->get_scene(scene)) {
4.       std::vector<uint32_t> ambient = scene.ambient_image;
5.       std::vector<uint16_t> intensity = scene.intensity_image[0];
6.       std::vector<uint32_t> depth = scene.depth_image[0];
7.       std::vector<SOSLAB::point_t> pointcloud = scene.pointcloud[0];
8.       std::size_t height = scene.rows;
9.       std::size_t width = scene.cols;
10.           std::size_t width2 = (scene.cols ==192)?scene.cols*3:scene.cols;
11.  }
12.}
```

Raw Data 획득 코드

ML-X Device으로부터 Raw Data를 획득하는 방법에 대한 코드입니다. ML-X Device의 Raw Data는 scene_t로 획득되며, 총 4개의 1차원 데이터 배열(ambient, intensity, depth, point cloud)로 구성되어 있습니다. 자세한 scene_t 구조체에 대한 설명은 Appendix에 기재되어 있습니다.

Line	Description
2	Raw Data를 저장할 SOSLAB::LidarMl::scene_t 변수를 scene 변수명으로 선언합니다.
3	scene_t 변수를 입력으로 받는 SOSLAB::LidarMl Class의 get_scene 함수를 통해 scene 변수에 Raw Data를 획득합니다.
4-7	Raw Data를 저장한 scene_t scene 구조체로부터 ambient, intensity, depth, pointcloud 데이터를 획득합니다. 각 데이터의 구조체 변수명 및 변수형은 Appendix에 기재되어 있습니다.
8-10	scene_t scene 구조체로부터 Depth, Intensity 이미지의 height, width 값을 획득합니다. width2 변수는 Ambient 이미지의 width 값 입니다.

3.3. Example Code for Ubuntu/ROS

2) Rviz - Publish Ambient, Depth, Intensity Image

```
1. ros::NodeHandle nh("~");
2. image_transport::ImageTransport it(nh);
3. image_transport::Publisher pub_ambient = it.advertise("ambient_color", 1);
4. sensor_msg::ImagePtr msg_ambient;
5. cv::Mat ambient_image
6. ambient_image(scene.rows, scene.cols, CV_32SC1, ambient.data());
7. ambient_image.convertTo(ambient_image, CV_8UC1, (255.0 / 2000),0);
8. msg_ambient = cv_bridge::CvImage(std_msg::Header(), "rgb8",
   ambient_image).toImageMsg();
9. pub_ambient.publish(msg_ambient);
```

Rviz에서의 Ambient, Depth, Intensity 이미지 Publish 코드

ROS 환경에서의 Rviz를 통한 이미지 가시화를 위해 ML-X Device로부터 획득한 Ambient, Depth, Intensity 데이터를 이미지로 변환하고 Publish를 수행하는 코드입니다. 위 코드는 Ambient 이미지 변환 및 Publish 생성 코드입니다.

Line	Description
1-4	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 ImageTransport, ImagePtr 등의 객체를 생성합니다. Publisher는 Topic "ambient" 로 ambient 데이터를 제공합니다.
5-6	scene_t 구조체의 Ambient 데이터를 이미지로 가시화하기 위해 OpenCV의 cv::Mat 형식으로 변환하는 과정입니다. cv::Mat 초기화 입력 값으로 Raw data의 Height(scene.rows), Width(scene.cols), Ambient 데이터 Type(CV_32SC1), Ambient Data 값을 입력합니다.
7	이미지 가시화를 위하여 최대 값(2000)과 최소 값(0)을 0~255 값의 8bit(uchar) 형태로 변환합니다.
8	OpenCV의 cv::Mat 객체를 Publisher Node의 Message 형태로 저장하기 위해 ImagePtr로 변환하는 과정입니다.
9	Topic이 "ambient_color"로 지정된 Publisher 객체의 publish 함수에 ImagePtr 변수를 입력 인수로 사용하여 Publish를 완료합니다.

각 데이터의 Publish 과정은 위의 Ambient Publish 과정과 동일하고, 이미지 변환 타입은 아래와 같습니다.

- Ambient : CV_32SC1
- Depth : CV_32SC1
- Intensity : CV_16UC1

3.3. Example Code for Ubuntu/ROS

3) Rviz - Publish Point Cloud

```
1.  typedef pcl::PointCLoud<pcl::PointXYZRGB> PointCloud_T
2.  static const char* DEFAULT_FRAME_ID = "map"

3.  ros::NodeHandle nh("~");
4.  ros::Publisher pub_lidar = nh.advertise<PointCloud_T>("pointcloud",10);

5.  PointCloud_T::Ptr msg_pointcloud(new PointCloud_T);
6.  msg_pointcloud->header.frame_id = DEFAULT_FRAME_ID
7.  msg_pointcloud->width = width;
8.  msg_pointcloud->height = height;
9.  msg_pointcloud->points.resize(scene.pointcloud.size())
10.for (int i = 0; i < scene.pointcloud.size(); i++) {
11.     msg_pointcloud->points[i].x = pointcloud[i].x/1000.0;
12.     msg_pointcloud->points[i].y = pointcloud[i].y/1000.0;
13.     msg_pointcloud->points[i].z = pointcloud[i].z/1000.0;
14.}
15.pcl_conversions::toPCL(ros::Time::now(), msg_pointcloud->header.stamp);
16.pub_lidar.publish(msg_pointcloud);
```

Rviz에서의 Point Cloud Publish 코드

ROS 환경에서의 Rviz를 통한 Point Cloud 가시화를 위해 ML-X Device로부터 획득한 pointcloud 데이터를 Publish하는 코드입니다.

Line	Description
3-4	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 ros::Publisher 객체를 생성합니다. Publisher는 Topic "pointcloud"로 Point Cloud 데이터를 제공합니다.
5-9	Message 변수를 정의하여 데이터 크기, 이름을 초기화합니다.
10-14	scene_t의 pointcloud 데이터를 msg_pointcloud 변수로 복사하고, 거리 단위를 mm에서 m로 변경합니다.
15-16	Point Cloud가 Scanning된 timestamp를 저장한 뒤, Topic이 "pointcloud"로 지정된 Publisher객체의 publish 함수에 PointCloud_T::Ptr 변수를 입력 인수로 사용하여 Publish를 완료합니다.

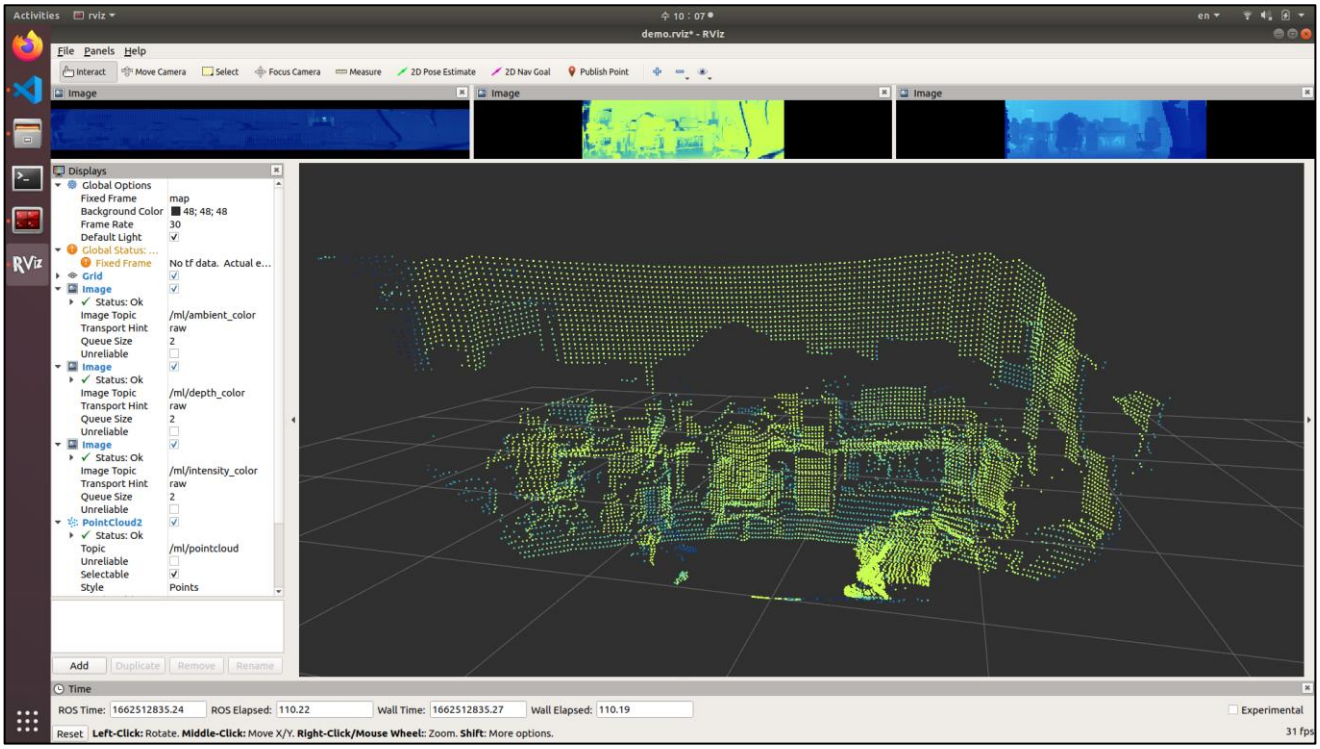
3.3. Example Code for Ubuntu/ROS

4) Rviz - Visualization

Rviz를 위한 가시화를 위해 아래 Command를 입력하여 ML-X Device 연결 및 예제 코드를 실행합니다.

```
$ cd ~/catkin_ws
$ catkin_make
$ source ~/catkin_ws/devel/setup.sh
$ cd ~/catkin_ws/src/ml/launch
$ roslaunch ml ml.launch
```

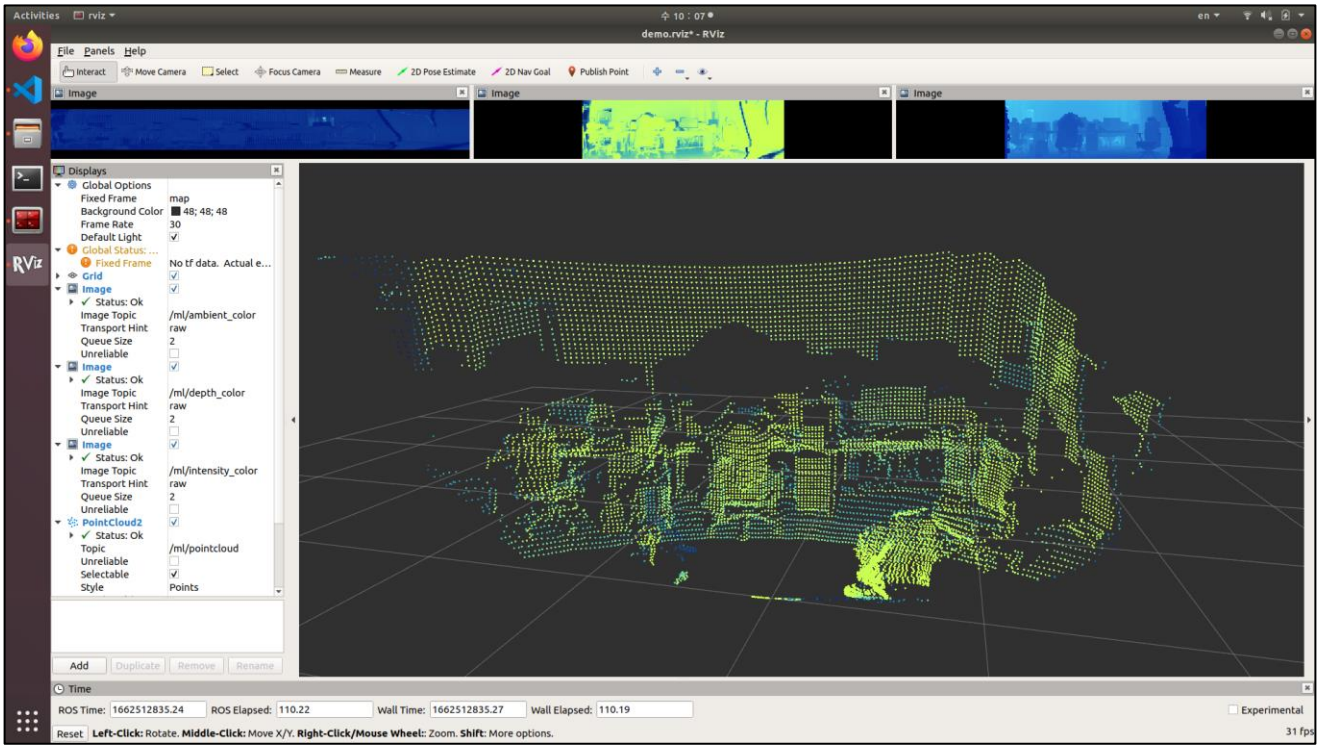
아래 그림과 같이 왼쪽 하단의 Add 버튼을 클릭하면 아래 그림과 같은 창을 확인할 수 있습니다. 해당 창의 상단 “By topic” 메뉴를 클릭하면 Publish 되고 있는 전체 Topic을 확인할 수 있습니다.



Rviz 프로그램 실행 화면

3.3. Example Code for Ubuntu/ROS

Ambient, Depth, Intensity 이미지를 가시화하기 위해서 Topic (/ambient, /depth, /intensity) 메뉴의 “Image”를 선택한 뒤 OK 버튼을 클릭하면 각 토픽에 대한 이미지를 가시화할 수 있으며, Point Cloud 데이터는 Topic (/pointcloud)의 “PointCloud2”를 선택한 뒤 OK 버튼을 클릭하면 가시화할 수 있습니다.



Rviz 기반 ML-X 데이터(Ambient/Depth/Intensity 이미지, Point Cloud) 가시화

5) Rviz – Multi-LiDAR Visualization

Rviz를 위한 다중 라이다 가시화를 위해 ML-X의 IP를 다르게 설정합니다. Multi-LiDAR 예제에서 ML-X의 IP 세팅은 아래와 같습니다.

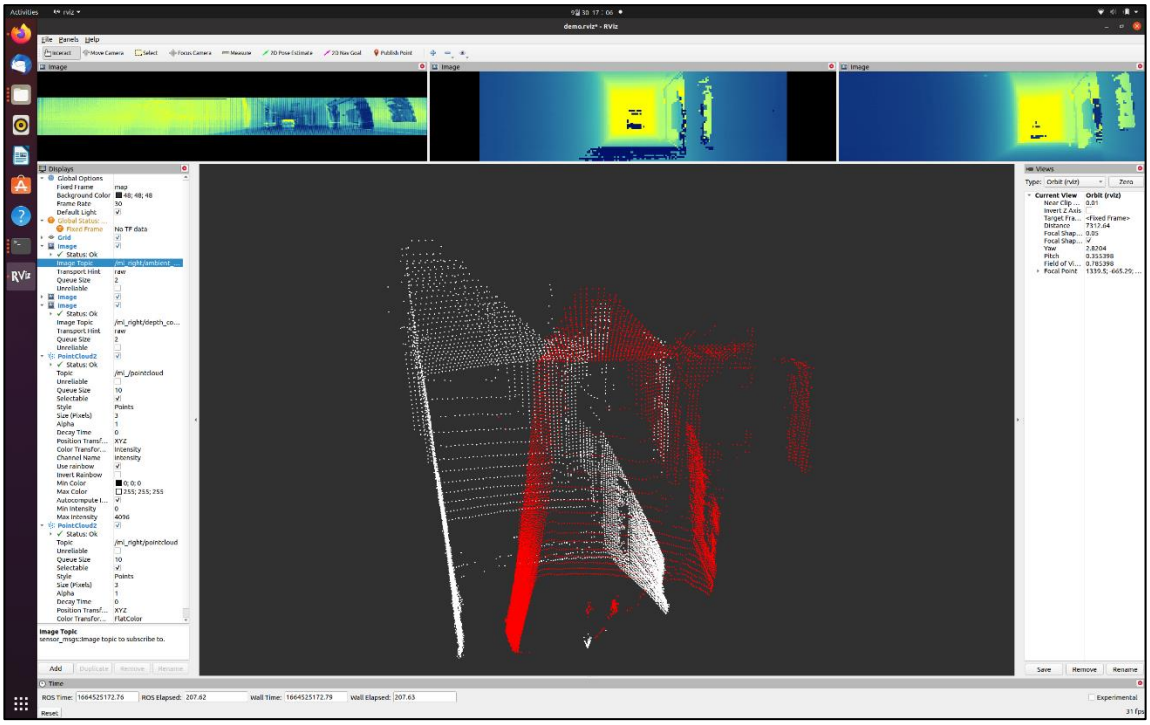
- ML-X IP #1 : 192.168.1.10
- ML-X IP #2 : 192.168.1.11

IP 변경 후, 아래 Command를 입력하여 ML-X Devices 연결 및 예제 코드를 실행합니다.

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

[Terminal #2]
$ cd ~/catkin_ws/src/ml/launch
$ roslaunch ml ml_second.launch
```

아래 그림과 같이 왼쪽 하단의 Add 버튼을 클릭하면 아래 그림과 같은 창을 확인할 수 있습니다. 해당 창의 상단 “By topic” 메뉴를 클릭하면 “ml”과 “ml_second” topic으로 Multi-LiDAR 데이터들이 전송됩니다.



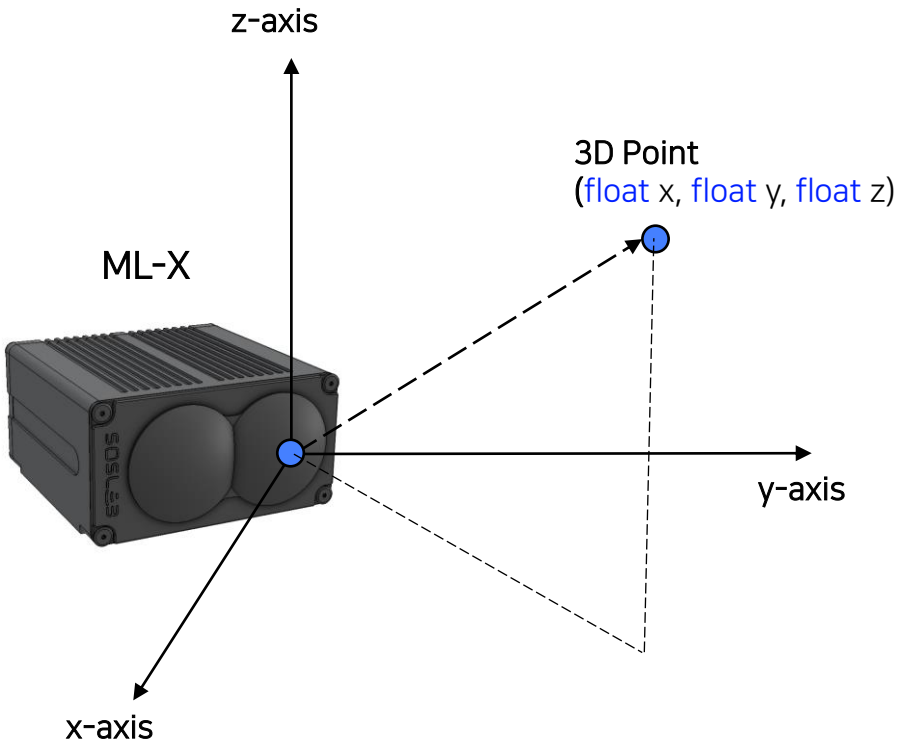
Multi-LiDAR 가시화 화면

Appendix

A.1. ML-X API - Classes

Classes

변수형	SOSLAB::point_t
Header	#include "soslab_typedef.h"
설명	3D Point에 대응하는 Coordinate System 구조체
Member Variables	Member Variables 설명
float x	Point의 x 좌표 정보 (단위 : mm)
float y	Point의 y 좌표 정보 (단위 : mm)
float z	Point의 z 좌표 정보 (단위 : mm)



3D Point Cloud의 Cartesian Coordinate System

Classes

변수형	SOSLAB::ip_setting_t
Header	#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.1. ML-X API - Classes

Classes

변수형	SOSLAB::LidarMl::scene_t
Header	#include "ml/libsoslab_ml.h"
설명	Raw Data에 대한 구조체
Member Variables	Member Variables 설명
std::vector<uint32_t> timestamp	Raw Data 획득 시간 [size : rows]
uint8_t frame_id	Frame Index [최대 : 255]
uint16_t rows	Raw Data의 Height 값
uint16_t cols	Raw Data의 Width 값
std::vector<uint32_t> ambient_image	Ambient 데이터 배열 (Ambient : 측정된 모든 신호 강도)
std::vector<std::vector<uint32_t>> depth_image	Depth 데이터 배열 [1st vector size : # echo, 2nd vector size : rows x cols] (Depth : 측정된 거리 데이터)
std::vector<std::vector<uint16_t>> intensity_image	Intensity 데이터 배열 [1st vector size : # echo, 2nd vector size : rows x cols] (Intensity : 측정된 LiDAR 신호 강도)
std::vector<std::vector<point_t>> pointcloud	Point cloud 데이터 배열 [1st vector size : # echo, 2nd vector size : rows x cols] (Point cloud : 3차원 point의 집합 데이터)

A.1. ML-X API - Classes

Classes

변수형	SOSLAB::LidarMl
Header	#include "ml/libsoslab_ml.h"
설명	ML-X Device Connection 및 Signal/Data Processing

Functions

`bool connect(const ip_settings_t ml, const ip_settings_t local);`

- 설명 : Local(PC)과 ML-X Device를 연결하는 함수
- Input : Local(PC) 및 ML-X Device의 IP 주소, Port 번호
- Output : 연결 상태에 대해 bool 변수형으로 반환 (연결되면 true 반환)

`bool disconnect();`

- 설명 : Local(PC)과 ML-X Device를 연결을 해제하는 함수
- Output : 연결 해제 상태에 대해 bool 변수형으로 반환(연결 해제되면 true 반환)

`bool run();`

- 설명 : ML-X Device를 구동하는 함수
- Output : 구동 유무에 대해 bool 변수형으로 반환(시작되면 true 반환)

`bool stop();`

- 설명 : ML-X Device를 구동을 중단하는 함수
- Output : 구동 중단 유무에 대해 bool 변수형으로 반환(중단되면 true 반환)

`bool get_scene(scene_t& scene);`

- 설명 : 입력 변수 scene으로 Raw Data를 획득하는 함수
- Input : scene_t 변수형 scene
- Output : Raw Data 획득 유무에 대해 bool 변수형으로 반환 (획득하면 true 반환)

A.1. ML-X API - Classes

Classes

변수형	SOSLAB::LidarMl
Header	#include "ml/libsoslab_ml.h"
설명	ML-X Play Mode
Functions	
void record_start(std::string filepath)	
• 설명 : ML-X 데이터 녹화 기능을 시작하는 함수 • Input : 녹화 파일(.bin)이 저장될 위치	
void record_stop()	
• 설명 : ML-X 데이터 녹화 기능을 정지하는 함수	
void play_start(const std::string filepath)	
• 설명 : ML-X 데이터 녹화 파일을 재생하는 함수 • Input : 녹화 파일(.bin)이 저장된 위치	
void play_stop()	
• 설명 : ML-X 데이터 재생 기능을 정지하는 함수	
bool get_scene(scene_t& scene, uint64_t index);	
• 설명 : 녹화 파일을 불러온 뒤, 입력 변수 scene과 frame 번호를 통해 해당 frame의 Data를 획득하는 함수 • Input (1): Raw Data를 저장할 변수 (scene_t& scene) • Input (2): Frame (uint64_t index) • Output : ML-X Device Raw Data를 저장한 scene_t 변수	
void get_file_size(uint64_t& size)	
• 설명 : 녹화 파일을 불러온 뒤, 총 Frame 수를 반환하는 함수 • Output : 총 Frame 수	

A.1. ML-X API - Classes

Classes

변수형	SOSLAB::LidarMl
Header	#include “ml/libsoslab_ml.h"
설명	ML-X Functions
Functions	
bool fps10(bool en)	
• 설명 : ML-X FPS 10Hz 기능의 On/Off를 제어하는 함수 • Input : 기능 사용(true) / 미사용(false)	
bool depth_completion(bool en)	
• 설명 : Super Resolution 모듈을 적용하는 기능의 On/Off를 제어하는 함수 • Input : 기능 사용(true) / 미사용(false)	

A.1. ML-X API - Classes

Classes

변수형	SOSLAB::LidarMl
Header	#include “ml/libsoslab_ml.h"
설명	ML-X Functions
Functions	
bool ambient_enable(bool en)	
• 설명 : Ambient 데이터 전송 On/Off를 제어하는 함수 • Input : Ambient 데이터 사용(true) / 미 사용(false)	
bool depth_enable(bool en)	
• 설명 : Depth 데이터 전송 On/Off를 제어하는 함수 • Input : Depth 데이터 사용(true) / 미 사용(false)	
bool intensity_enable(bool en)	
• 설명 : Intensity 데이터 전송 On/Off를 제어하는 함수 • Input : Intensity 데이터 사용(true) / 미 사용(false)	
bool multi_echo_enable(bool en)	
• 설명 : Multi-echo 데이터 전송 On/Off를 제어하는 함수 • Input : Multi-echo 데이터 사용(true) / 미 사용(false)	

A.3.1 Normal Packet Structure (192 pixels)

UDP 통신의 기본 패킷 구조는 아래와 같습니다.

Byte

7	6	5	4	3	2	1	0
header							
timestamp							
status							
-					row_num	frame_id	type
ambient[1]				ambient[0]			
...				...			
ambient[575]				ambient[574]			
-		intensity[0][echo0]		range[0][echo0]			
point_cloud[0][echo0] (x[20:0], y[41:21], z[62:42])							
...							
-		intensity[191][echo0]		range[191][echo0]			
point_cloud[191][echo0] (x[20:0], y[41:21], z[62:42])							

A.2. UDP Protocol Packet Structure

UDP Protocol 기본 패킷 구조 (192 pixels)				
Byte	Name	Sub Name	Type	Description
0~7	header	-	char	Header: "LIDARPKT"
8~15	timestamp	-	uint64_t	Timestamp. Unit: 1[ns] Little Endian (Byte 8: Lowest Byte)
16~23	status	-	uint64_t	Sensing Data
24	type	-	uint8_t	Normal Packet: 0x01
25	frame_id	-	uint8_t	Frame Count Increase in order with frame
26	row_number	-	uint8_t	Row: 0~55
27~31	rsvd	-	uint8_t	Reserved
32~2335	ambient_data	-	uint32_t	576 pixels
2336~2339 2340~2341 2342~2343 2344~2351	lidar_data [0][echo 0]	range intensity rsvd point_cloud(x, y,z)	uint32_t uint16_t uint16_t int64_t	Distance: 0~262143 [mm] Intensity Reserved Point cloud (x, y, z). Unit [mm] [20:0] x, [41:21] y, [62:42] z
...	...	...	...	...
5392~5395 5396~5397 5398~5399 5400~5407	lidar_data [191][echo 0]	range intensity rsvd point_cloud(x, y,z)	uint32_t uint16_t uint16_t int64_t	Distance: 0~262143 [mm] Intensity Reserved Point cloud (x, y, z). Unit [mm] [20:0] x, [41:21] y, [62:42] z

A.3.2 Depth Completion Packet Structure (576 pixels)

UDP 통신의 Depth completion 패킷 구조는 아래와 같습니다.

Byte

7	6	5	4	3	2	1	0
header							
timestamp							
status							
-					row_num	frame_id	type
ambient[1]				ambient[0]			
...				...			
ambient[575]				ambient[574]			
-		intensity[0][echo0]		range[0][echo0]			
point_cloud[0][echo0] (x[20:0], y[41:21], z[62:42])							
...							
-		intensity[575][echo0]		range[575][echo0]			
point_cloud[575][echo0] (x[20:0], y[41:21], z[62:42])							

A.2. UDP Protocol Packet Structure

UDP Protocol Depth Completion 패킷 구조 (576 pixels)				
Byte	Name	Sub Name	Type	Description
0~7	header	-	char	Header: "LIDARPKT"
8~15	timestamp	-	uint64_t	Timestamp. Unit: 10[ns] Little Endian (Byte 8: Lowest Byte)
16~23	status	-	uint64_t	Sensing Data
24	type	-	uint8_t	Depth Completion Packet: 0x11
25	frame_id	-	uint8_t	Frame Count Increase in order with frame
26	row_number	-	uint8_t	Row: 0~55
27~31	rsvd	-	uint8_t	Reserved
32~2335	ambient_data	-	uint32_t	576 pixels
2336~2339 2340~2341 2342~2343 2344~2351	lidar_data [0][echo 0]	range intensity rsvd point_cloud(x, y,z)	uint32_t uint16_t uint16_t int64_t	Distance: 0~262143 [mm] Intensity Reserved Point cloud (x, y, z). Unit [mm] [20:0] x, [41:21] y, [62:42] z
...	...	...	...	...
11536~11539 11540~11541 11542~11543 11544~11551	lidar_data [575][echo 0]	range intensity rsvd point_cloud(x, y,z)	uint32_t uint16_t uint16_t int64_t	Distance: 0~262143 [mm] Intensity Reserved Point cloud (x, y, z). Unit [mm] [20:0] x, [41:21] y, [62:42] z

A.4.1 TCP Protocol Packet Structure

TCP 통신은 JSON Format으로 구성됩니다. PC에서 설정된 명령어를 전송하면 ML-X에서 응답하는 형태로 구성되어 있습니다. JSON 형태의 구조는 아래 Table과 같습니다.

TCP Protocol 패킷 구조	
JSON Format	Description
{ "command": "run" }	Device Run
{ "command": "stop" }	Device Stop

Solid-State LiDAR ML-X

User Guide

감사합니다.