



*Solid-State LiDAR **ML-X***



# User Guide



Release v1.2.2

2022-10-12

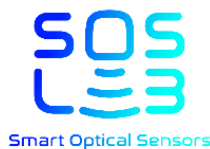
# Table of Contents

---

<b>1. Hardware Configuration</b>	<b>03</b>
1.1. Mechanical Interface	04
1.2. Electrical Interface	07
<b>2. SOS Studio</b>	<b>21</b>
2.1. Installation	22
2.2. TCP/IP Setting	23
2.3. Connection	25
2.4. Data Load	28
2.5. Device Setting	29
2.6. Point Cloud Setting	34
2.7. Image Setting	35
2.8. Grid Setting	36
2.9. X-Y / Y-Z / Z-X Axis	37
2.10. Play / Record Mode	38
<b>3. ML-X LiDAR API</b>	<b>39</b>
3.1. ML-X Example Code Build for Ubuntu	40
3.2. Example Code for Ubuntu/ROS	43
3.3. ML-X Example Code Build for Windows	49
3.4. Example Code for Windows	53
<b>Appendix</b>	<b>55</b>
A.1. ML-X API – Typedefs	56
A.2. ML-X API – Classes	57
A.3. UDP Protocol Packet Structure	62
A.4. TCP Protocol Packet Structure	66 <sub>2</sub>

# 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

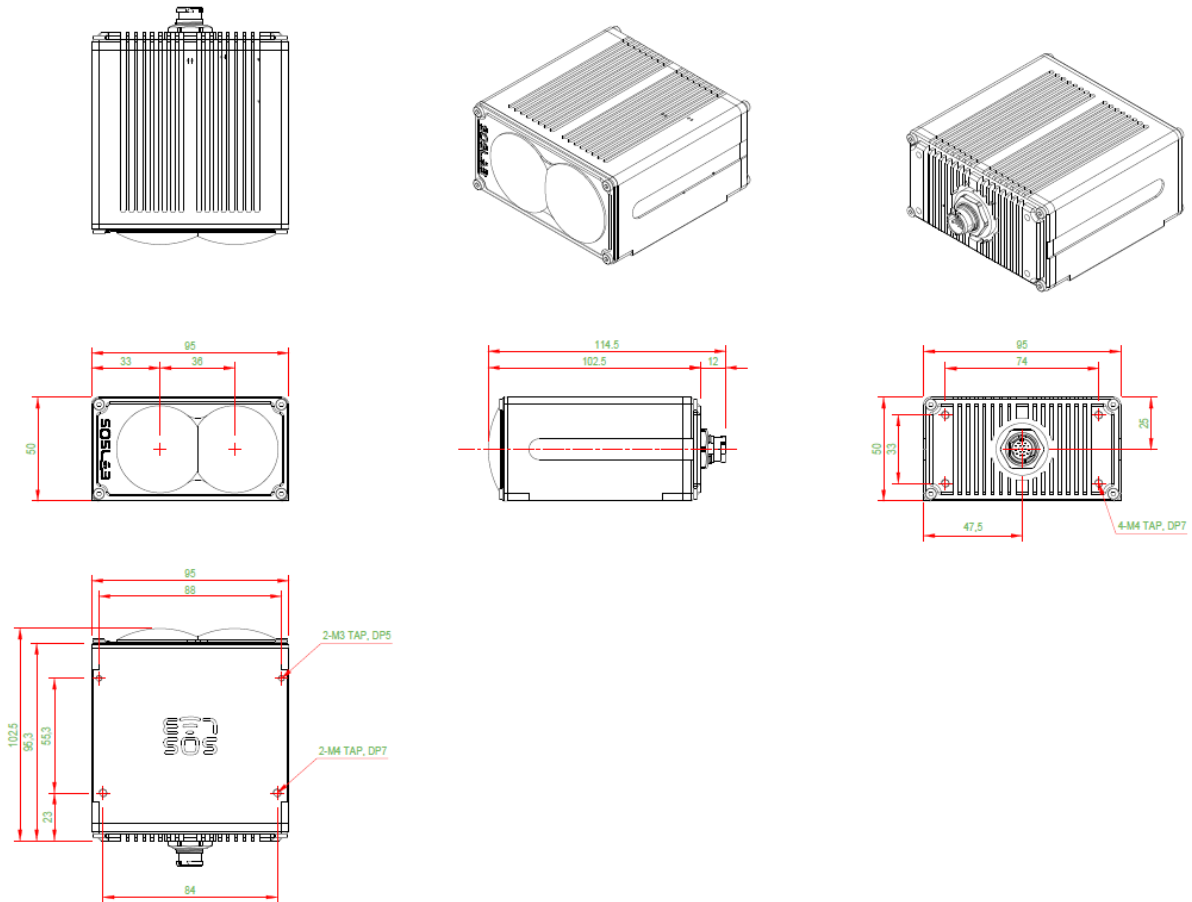


(f) AC/DC Power Cable

# 1.1. Mechanical Interface

## 1.1.2 Exterior Mechanical Dimensions

### ML-X 120° Dimensions

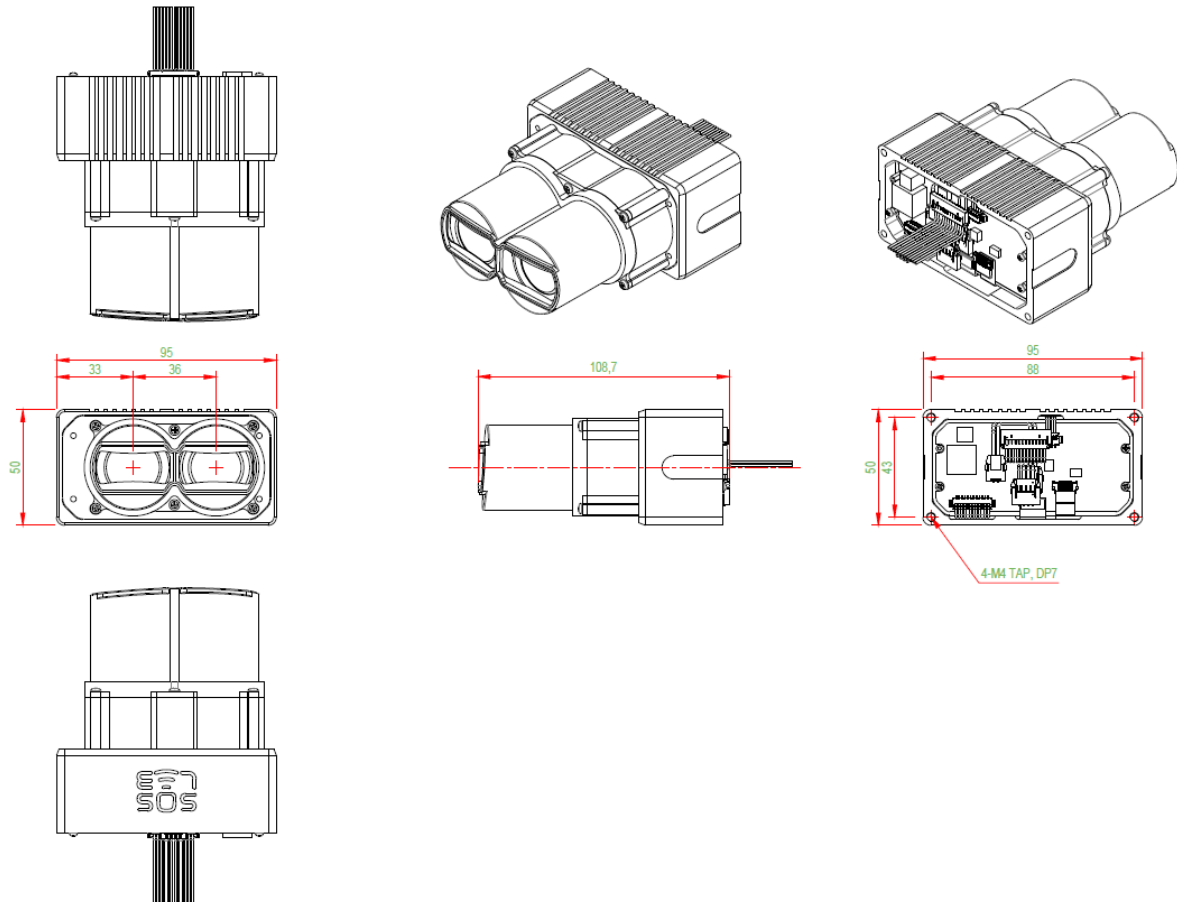


<The sensor has 4×M4 mounting holes>

# 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.

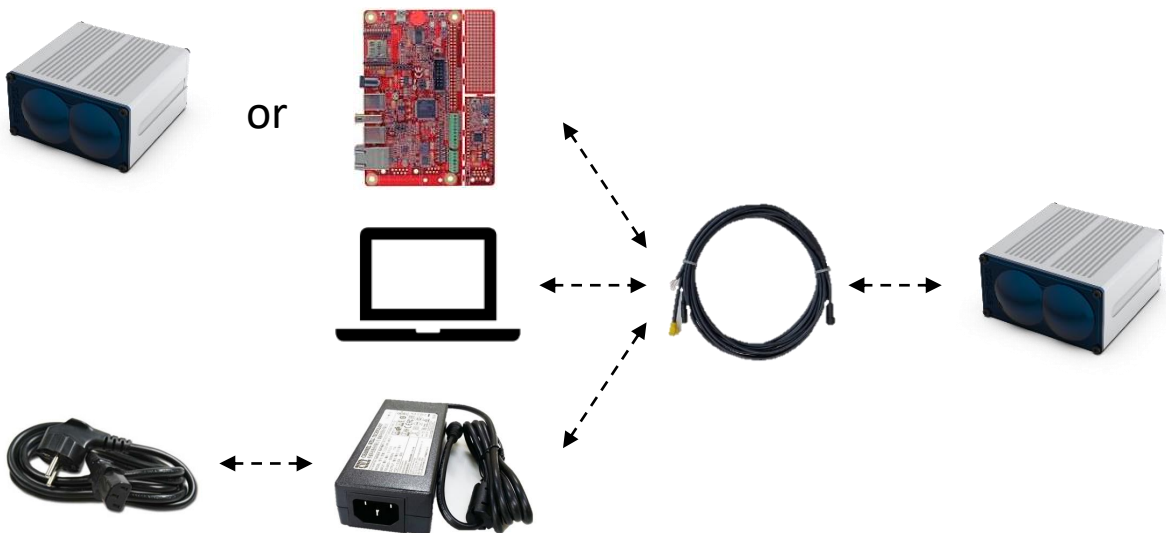
1. 제공된 센서와 통합 케이블을 연결합니다.
2. 이더넷 케이블과 PC를 연결합니다.
3. Timing Synchronization을 사용할 경우, 제공된 센서와 외부 device를 연결합니다.
4. 파워케이블과 파워 어댑터를 연결합니다.



<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.  
Device Setup을 이용해 IP/Port 정보를 변경할 수 있습니다.

- 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 Adapter Information

제공된 파워 어댑터의 정보는 다음과 같습니다. 아래 규격을 만족하지 않는 전원의 사용으로 인해 발생한 문제에 대해서는 제품의 신뢰성을 보증하지 않습니다.

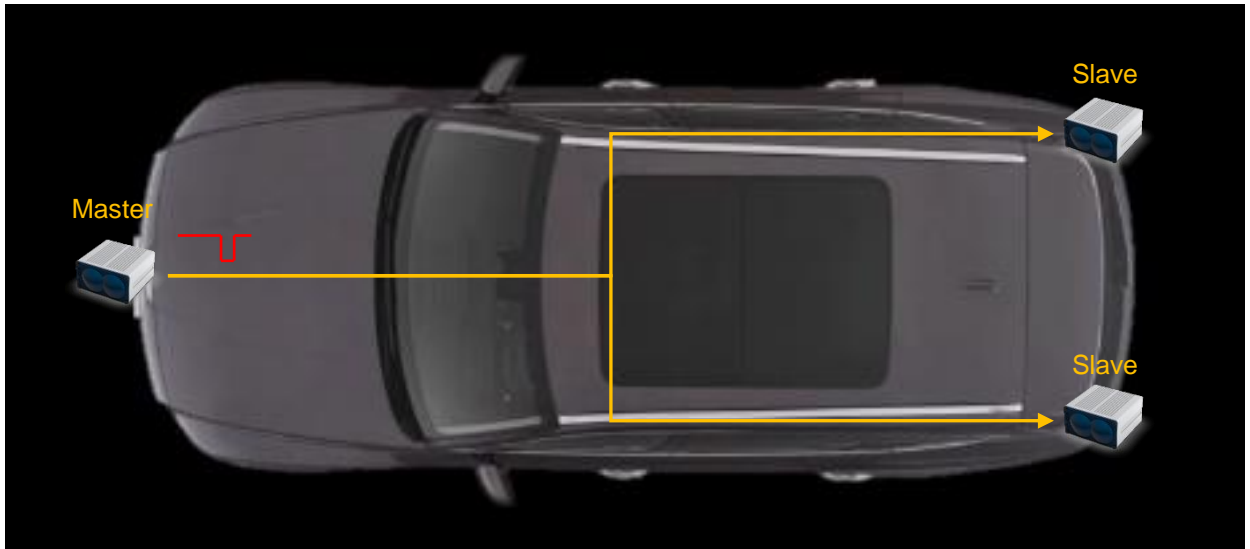
- 정격입력 : 100-240V 50/60Hz 1.7A
- 정격출력 : +12.0V 5.0A 60.0W



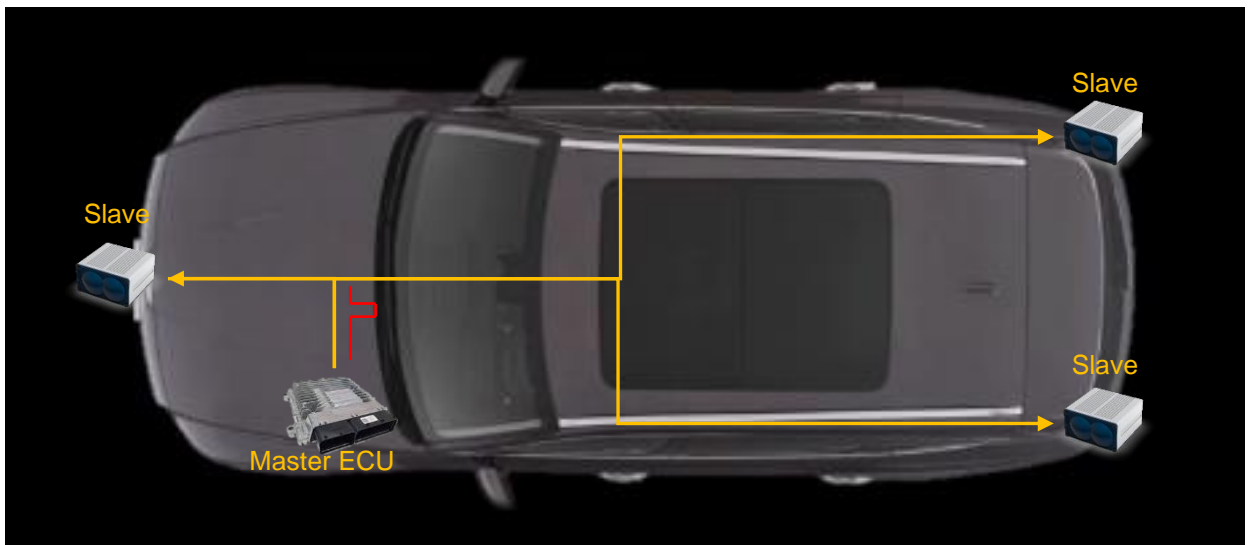
### 1.2.4 Timing Synchronization

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

Ex1) ML\_X간 Master/Slave로 구성하여 프레임 동기화를 구현할 수 있습니다.



Ex2) ECU 같은 외부 DEVICE의 신호를 받아서 ML\_X가 Slave로 동기화될 수 있습니다.



### 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)

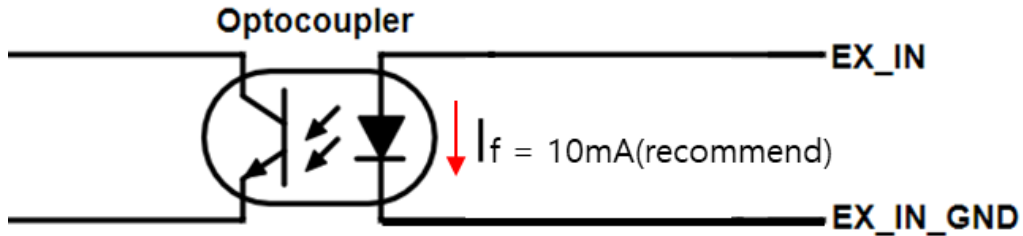


## 1.2. Electrical Interface

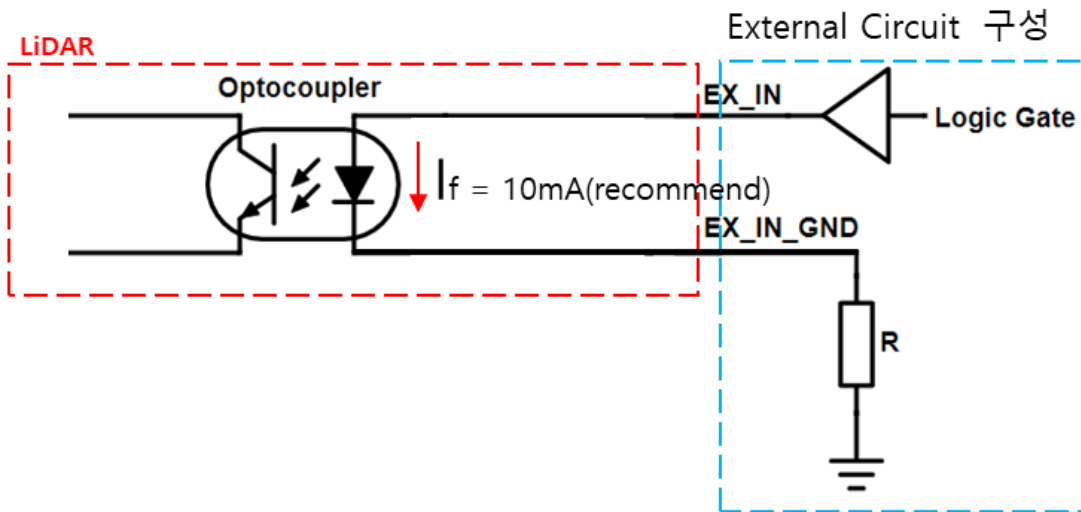
### 1.2.4.2 External SYNC IN

ML X의 Sync input 내부 회로 구성과, 외부 회로 구성을 위한 정보를 제공합니다.

1) ML X의 Sync input 내부 회로 구성은 아래와 같습니다.



2) 외부 회로 구성은 아래형태를 Recommend 합니다.



3) EX\_IN Voltage에 따른 저항 값 선정

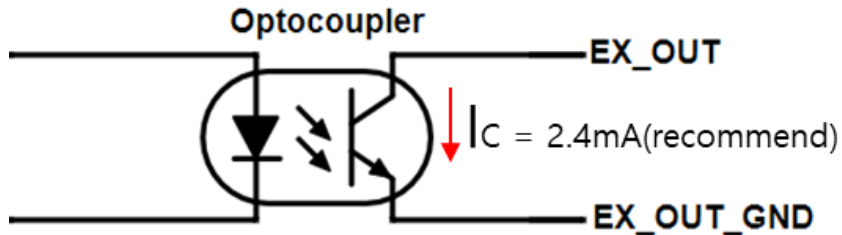
EX_IN Voltage	External Resistor(R)	Forward Current ( $I_F$ )	Recommended $I_F$
3.3 V (Min)	200	10.25mA	$7.5\text{ mA} < I_F < 15\text{ mA}$ $I_F(\text{mA}) = \frac{V - 1.25}{R} * 1000$
5 V	360	10.41 mA	
12 V	1000	10.75 mA	
24 V (Max)	2200	10.34 mA	

## 1.2. Electrical Interface

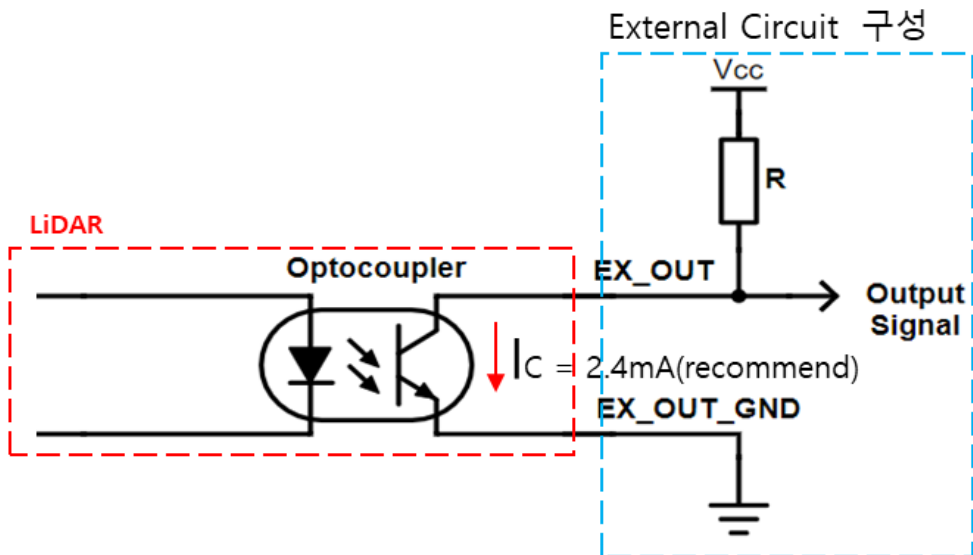
### 1.2.4.3 External SYNC OUT

ML X의 Sync input 내부 회로 구성과, 외부 회로 구성을 위한 정보를 제공합니다.

1) ML X의 Sync Output 내부 회로구성은 아래와 같습니다.



2) 외부 회로 구성은 아래형태를 Recommend 합니다.



3) Vcc 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	

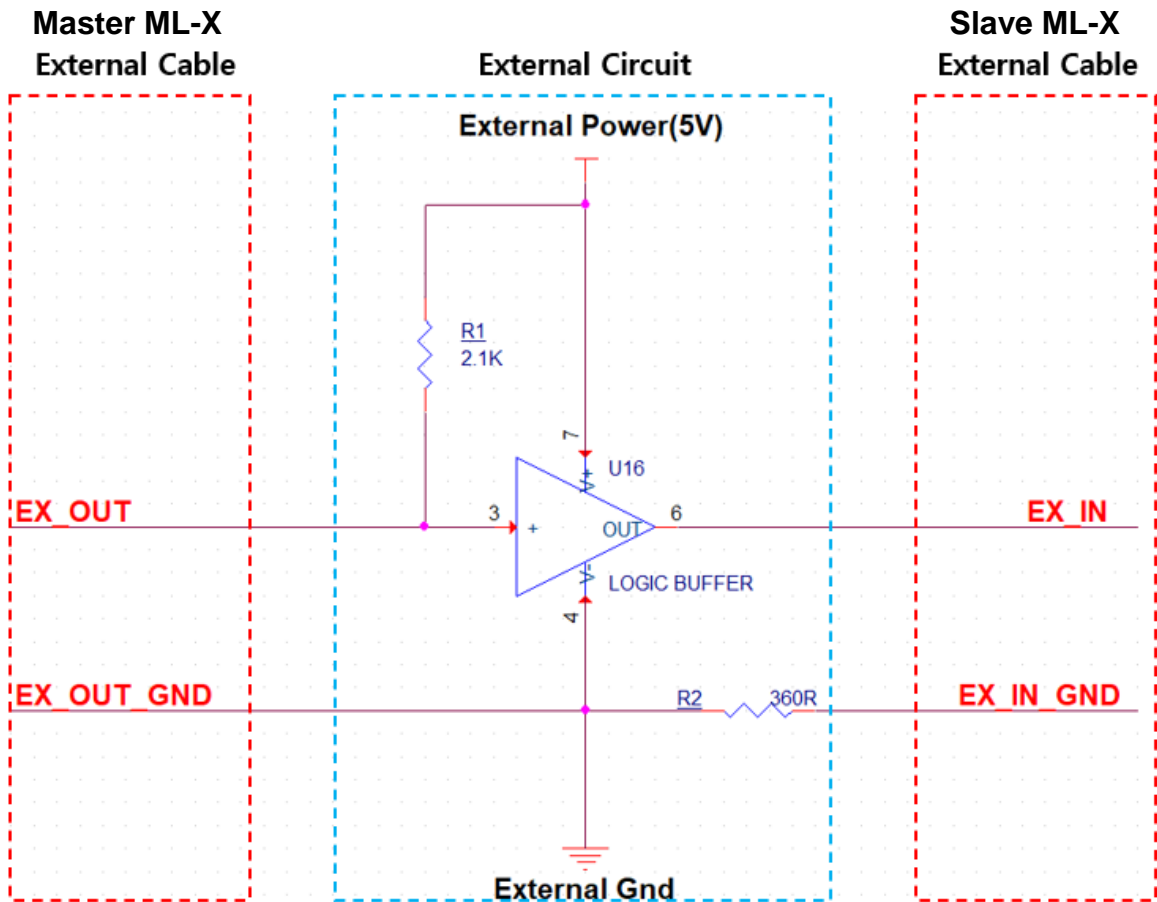
## 1.2. Electrical Interface

### 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

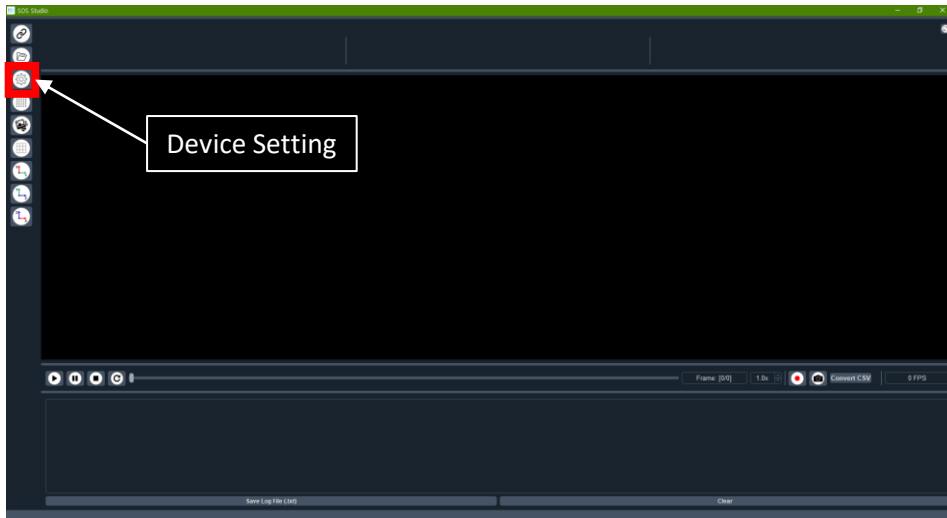


### 1.2.4.5 SYNC FUNCTION

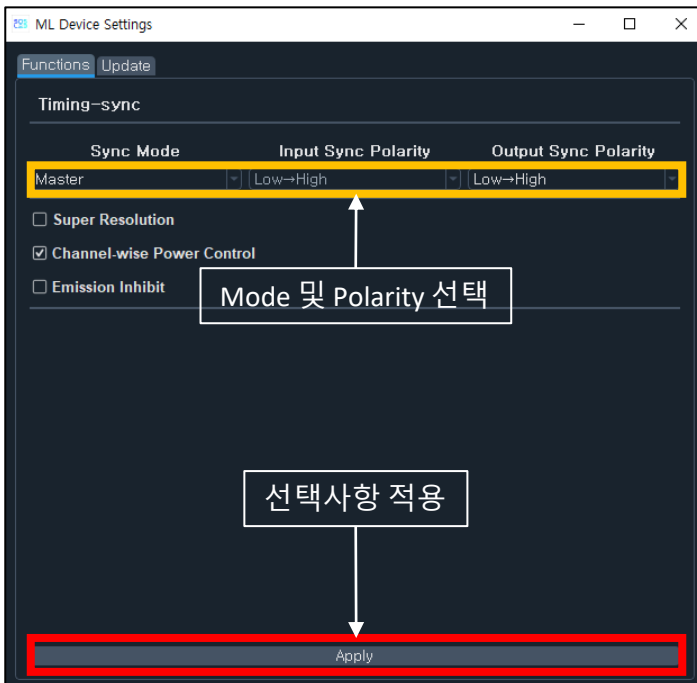
SOS Studio를 사용하여 ML X의 SYNC MODE를 Master 또는 Slave로 설정 할 수 있습니다.

#### 1) Sync Mode 설정 방법

- SOS Studio를 실행 후 Device Setting 클릭



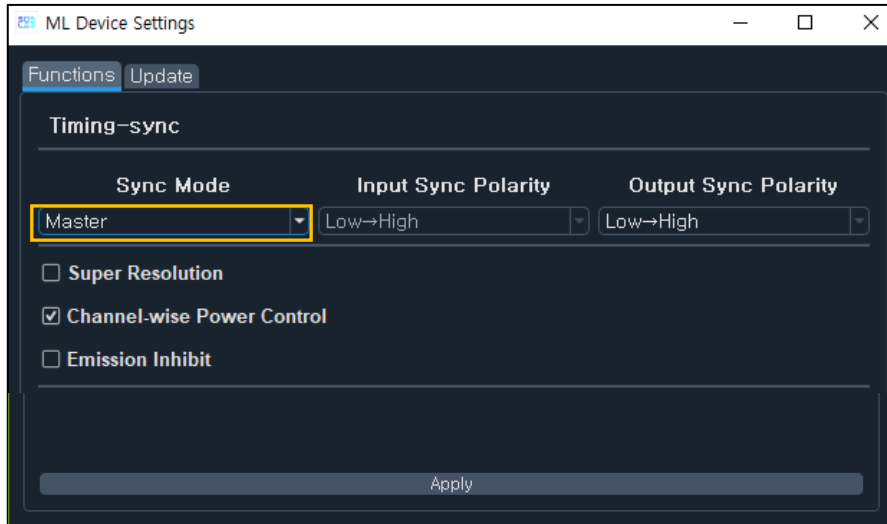
- Functions 탭에서 모드 선택 및 적용가능.



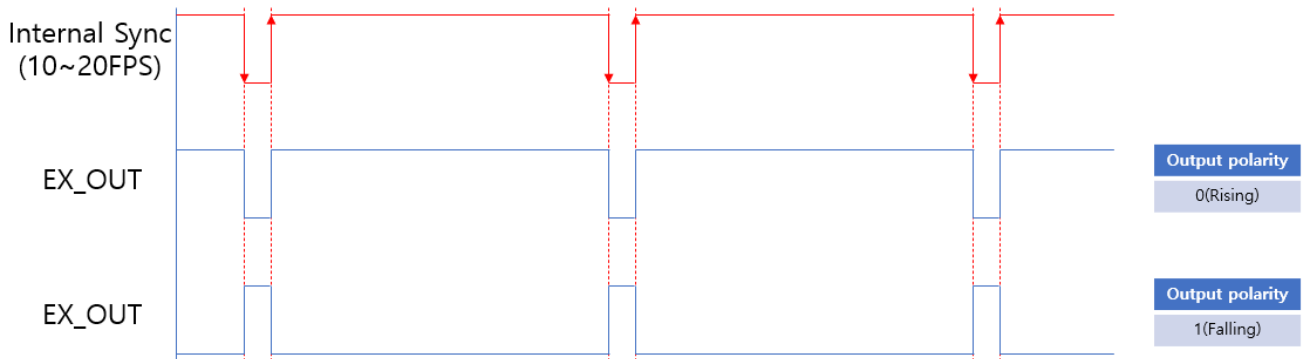
## 1.2. Electrical Interface

### 2) MASTER

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



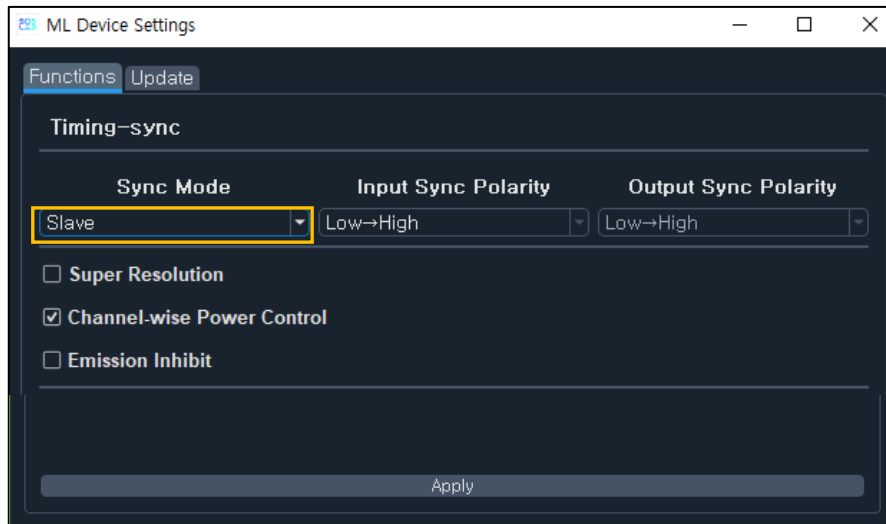
- 설정된 Frame Rate에 맞춰 매 프레임마다 동기화 신호를 출력합니다. Sync Out의 Polarity(High → Low 또는 Low → High) 변경이 가능합니다



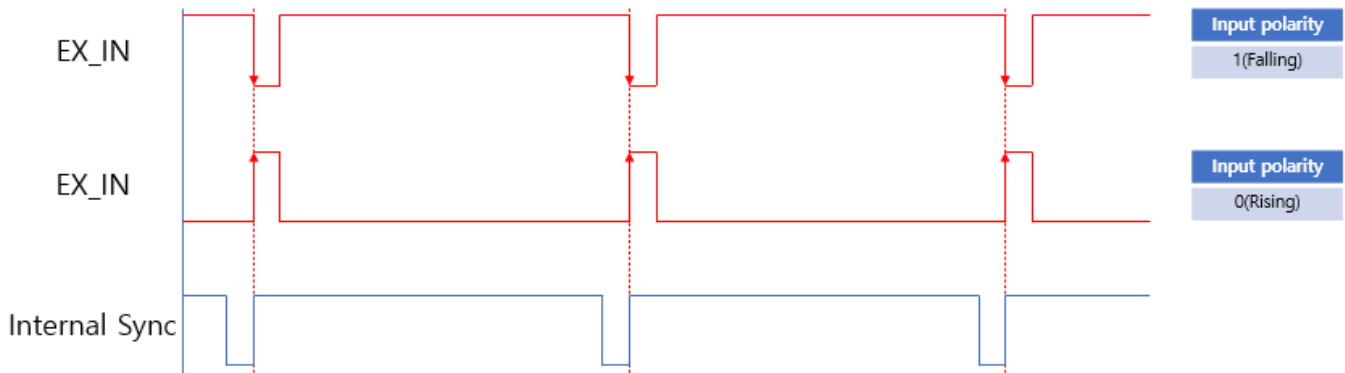
## 1.2. Electrical Interface

### 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

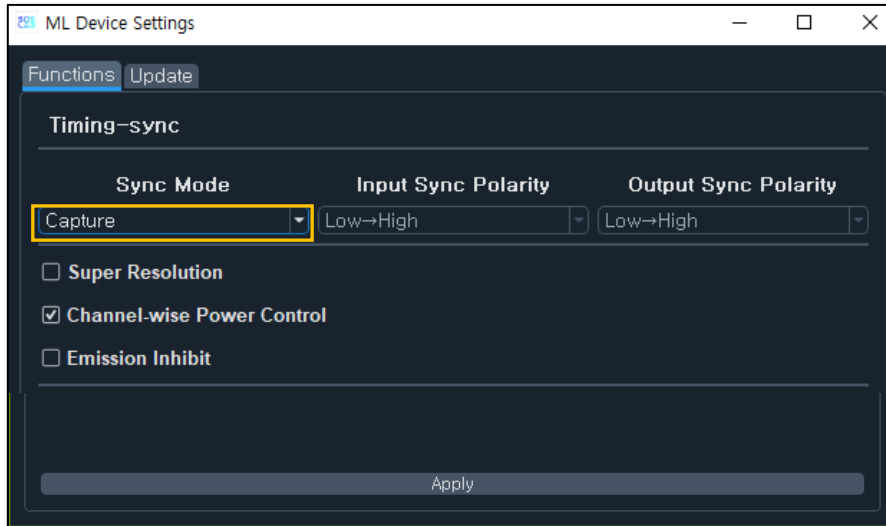




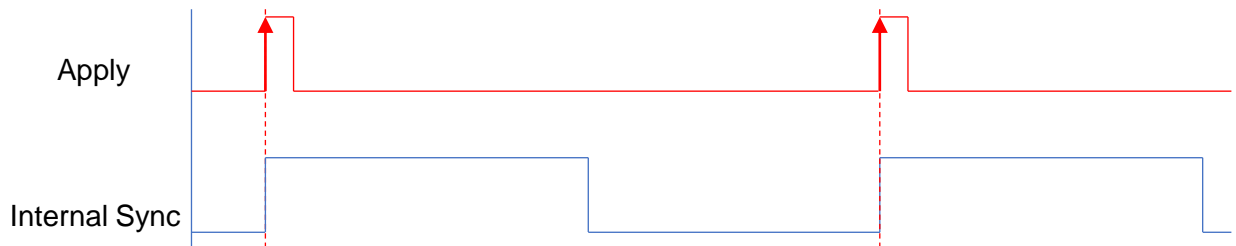
## 1.2. Electrical Interface

### 4) CAPUTE (비동기)

- Sync Mode를 Capture 선택 후 Apply 클릭하면 Capture mode로 작동합니다



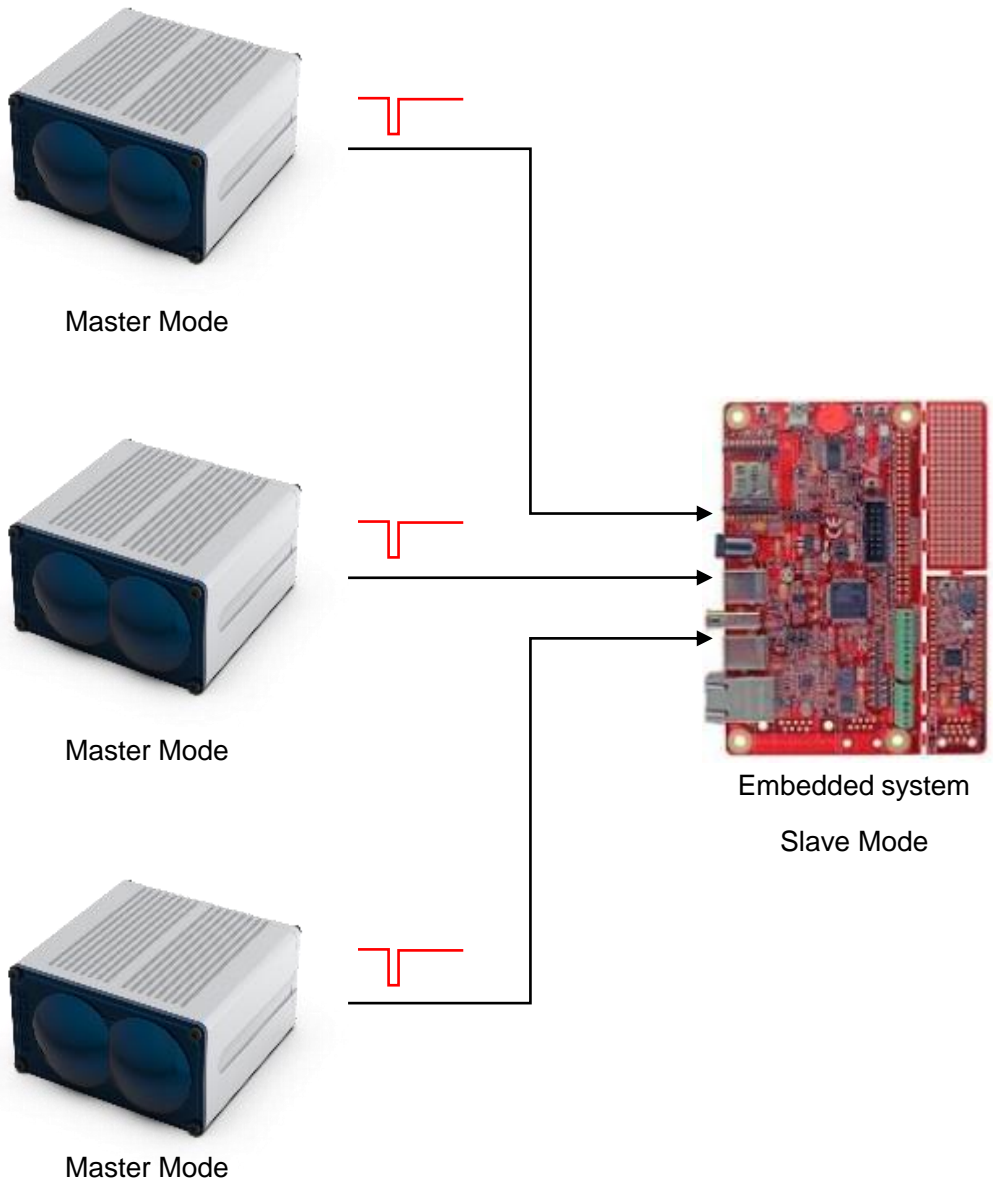
- Apply 누를 때 마다 Internal Sync를 1회만 발생시켜 scene을 update 합니다.



### 1.2.4.5 APPLICATION

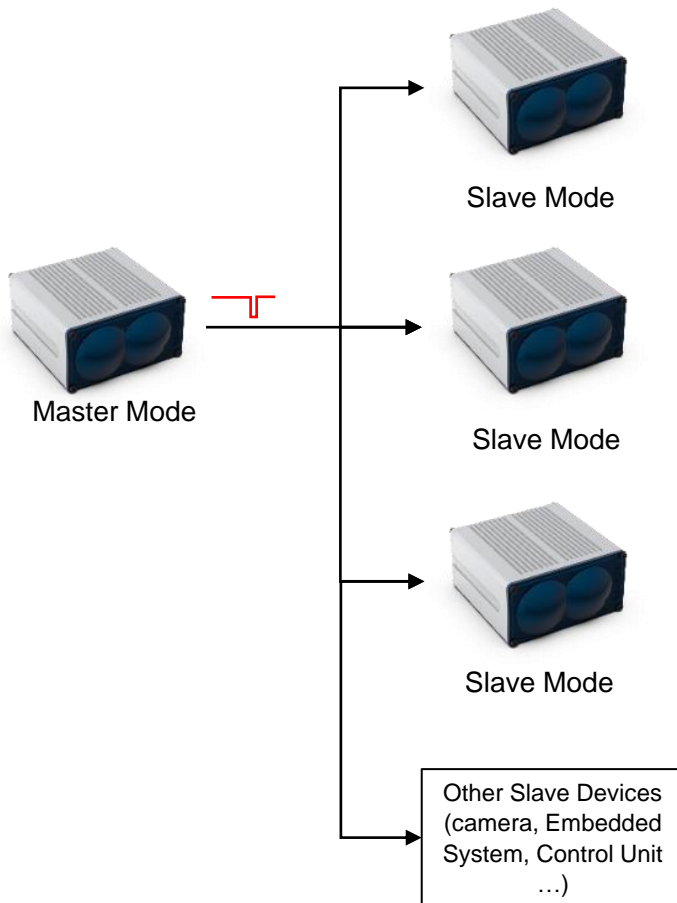
입/출력 동기화 신호를 사용하여 다수의 ML X 또는 기타 장치들과 동기화된 구성을 할 수 있습니다.

#### 1) Multiple master ML-X -> Single slave device

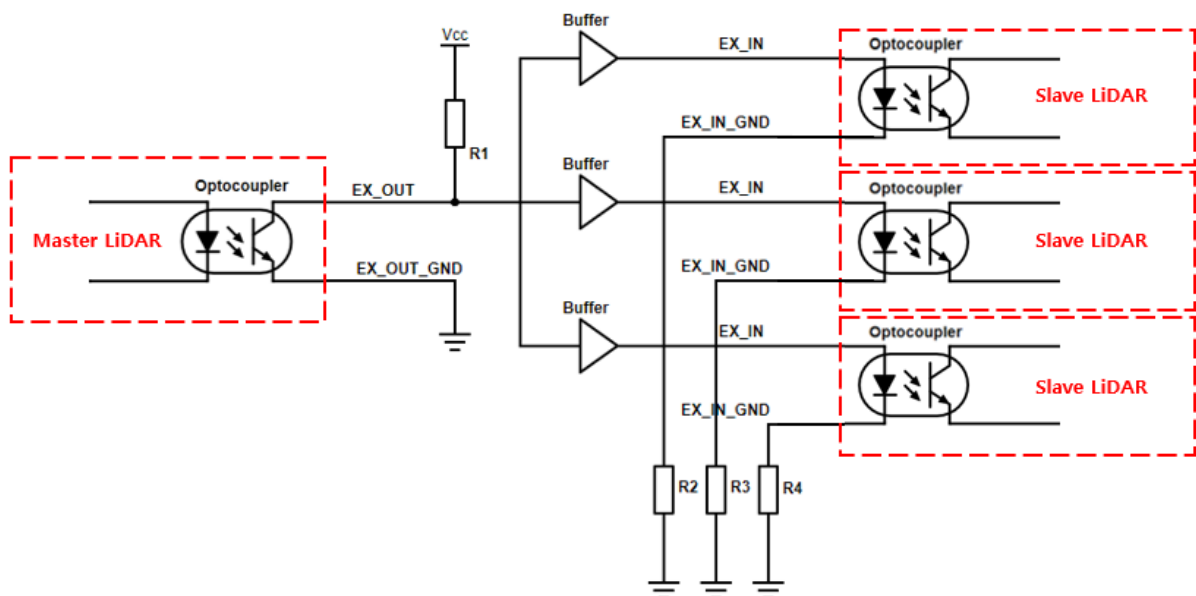


## 1.2. Electrical Interface

### 2) Single Master ML-X → Multiple Slave Device

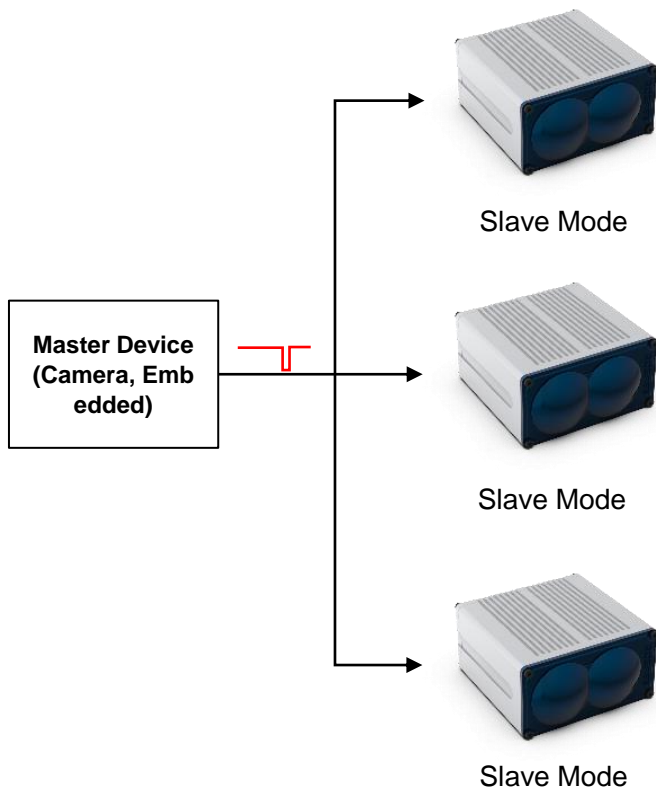


- External Circuit 구성

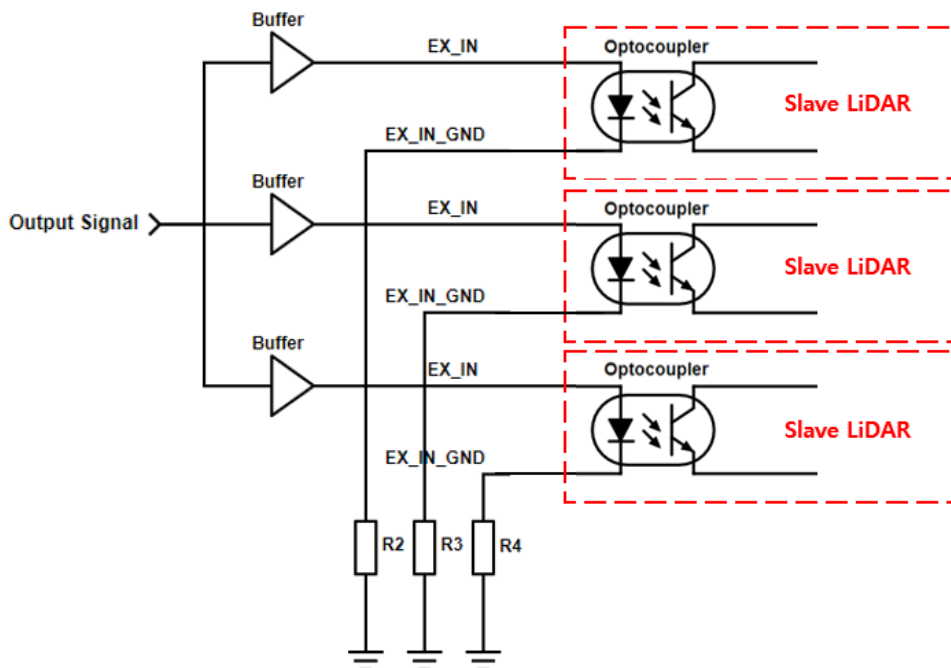


## 1.2. Electrical Interface

### 3) Single Master device → Multiple slave Device

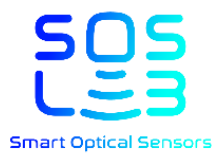


- External Circuit 구성



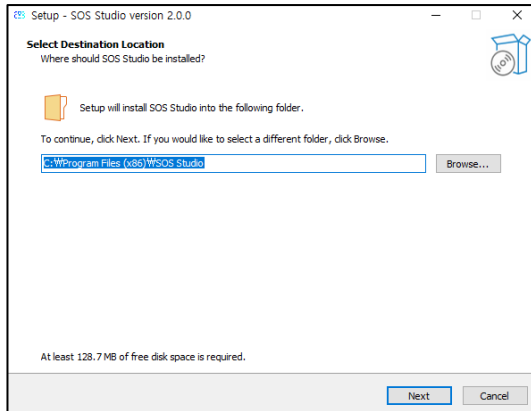
# Chapter 2

## SOS Studio

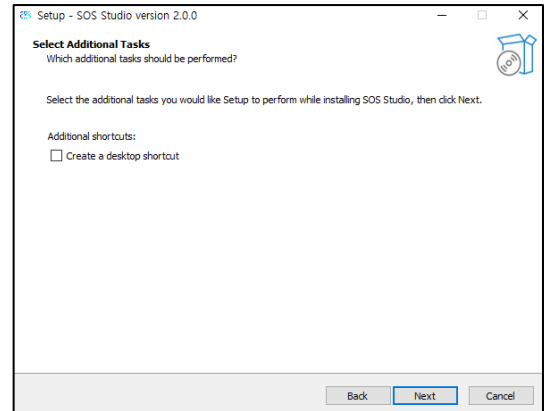


## 2.1 Installation

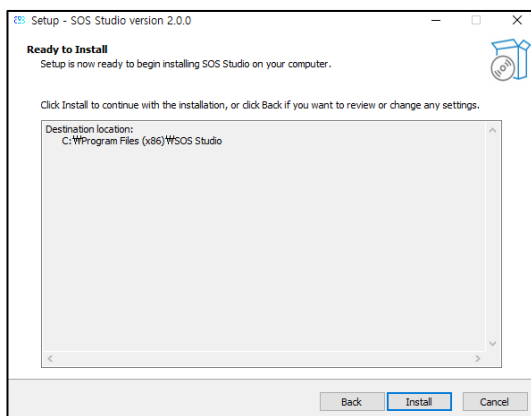
SOS Studio 설치를 진행합니다. “SOS Studio\_setup.exe” 설치파일을 실행하여 설치를 진행합니다. 설치 순서는 다음과 같습니다.



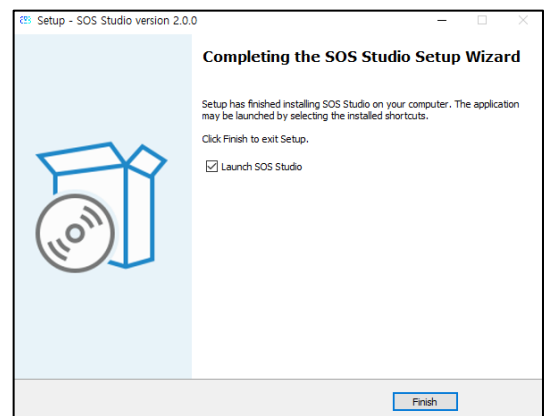
(a) 설치 파일 경로 설정



(b) 바탕화면 바로가기 설정



(c) SOS Studio 설치 시작



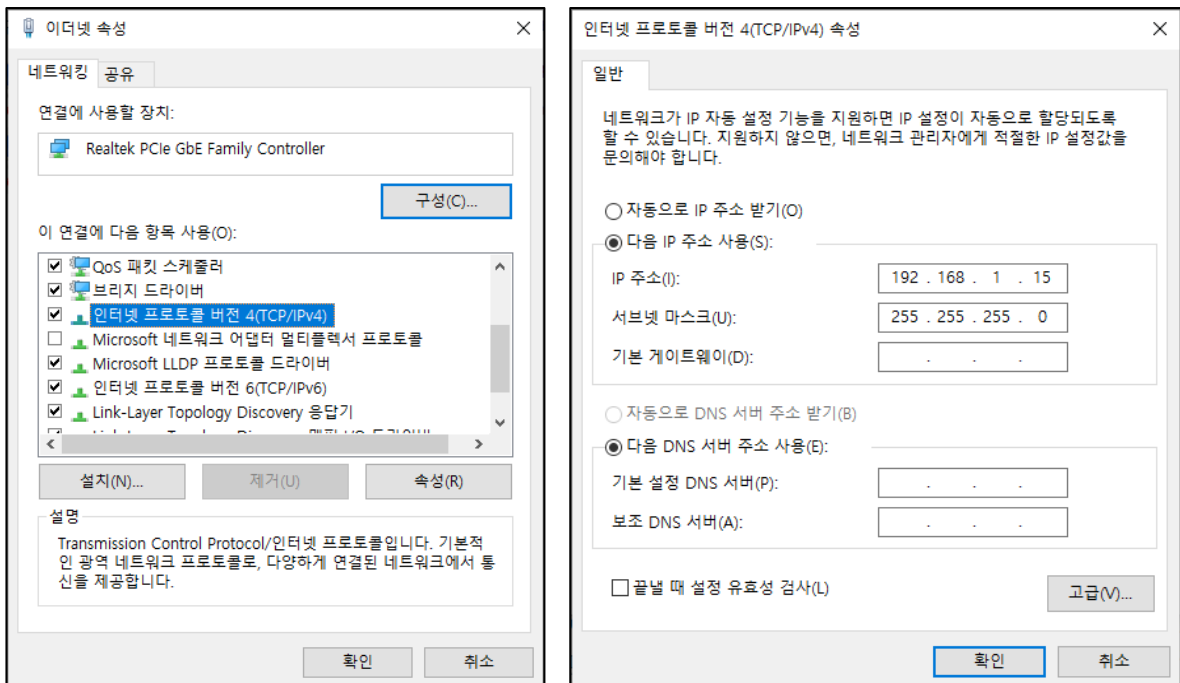
(d) SOS Studio 설치 완료

설치가 완료되면 SOS Studio가 실행됩니다.

### 2.2 TCP/IP Setting

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

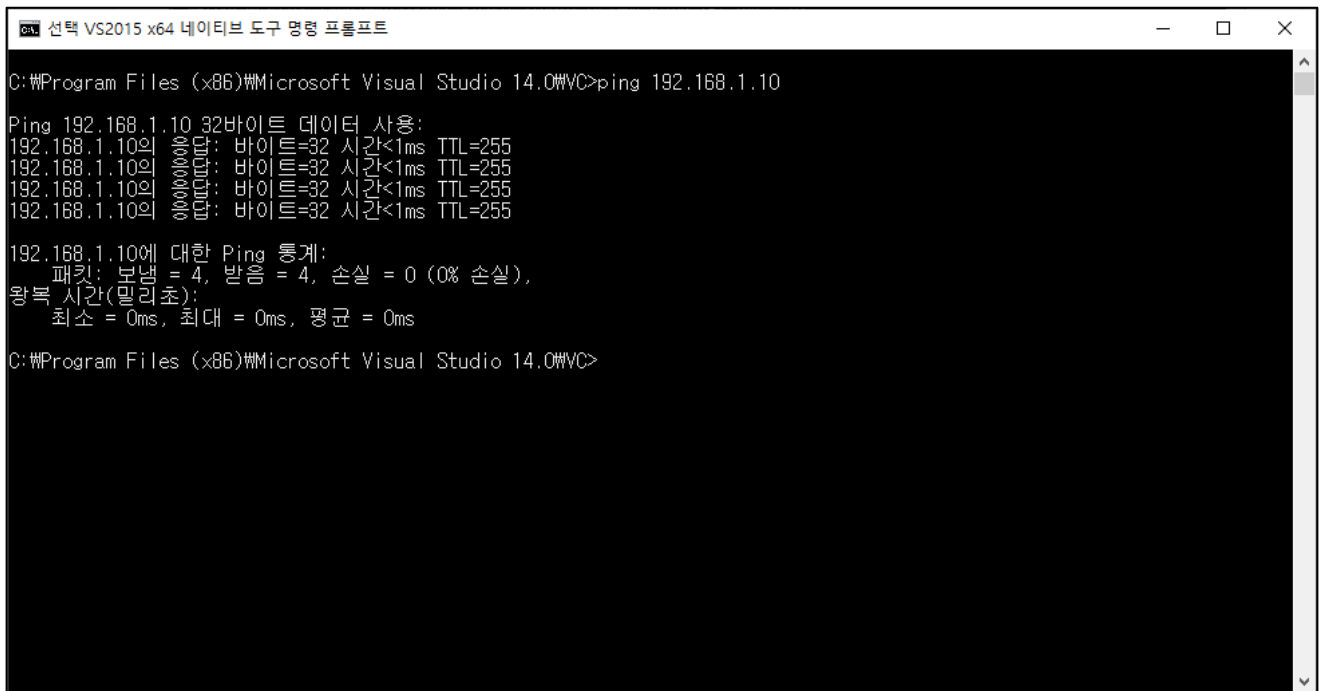
- 1) ML-X 전원 케이블을 연결하고, 네트워크 케이블로 PC와 LiDAR를 연결합니다.
- 2) 제어판 > 네트워크 및 인터넷 > 네트워크 및 공유 센터를 실행합니다.
- 3) 활성화 된 이더넷을 클릭 후 속성 창을 엽니다.
- 4) 인터넷 프로토콜 버전 4(TCP/IPv4) 선택 후 속성 버튼을 클릭합니다.
- 5) 속성 창에서 “다음 IP 주소 사용” 옵션을 클릭합니다.
- 6) IP 주소(192.168.1.15)와 서브넷 마스크(255.255.255.0)를 아래 그림과 같이 설정 후 확인 버튼을 클릭합니다.



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

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 -t**'를 입력합니다.
- 3) ML-X LiDAR와 PC가 정상적으로 연결되었다면 아래와 같은 메시지가 출력됩니다.



```
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
192.168.1.10의 응답: 바이트=32 시간<1ms TTL=255
192.168.1.10의 응답: 바이트=32 시간<1ms TTL=255
192.168.1.10의 응답: 바이트=32 시간<1ms TTL=255

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

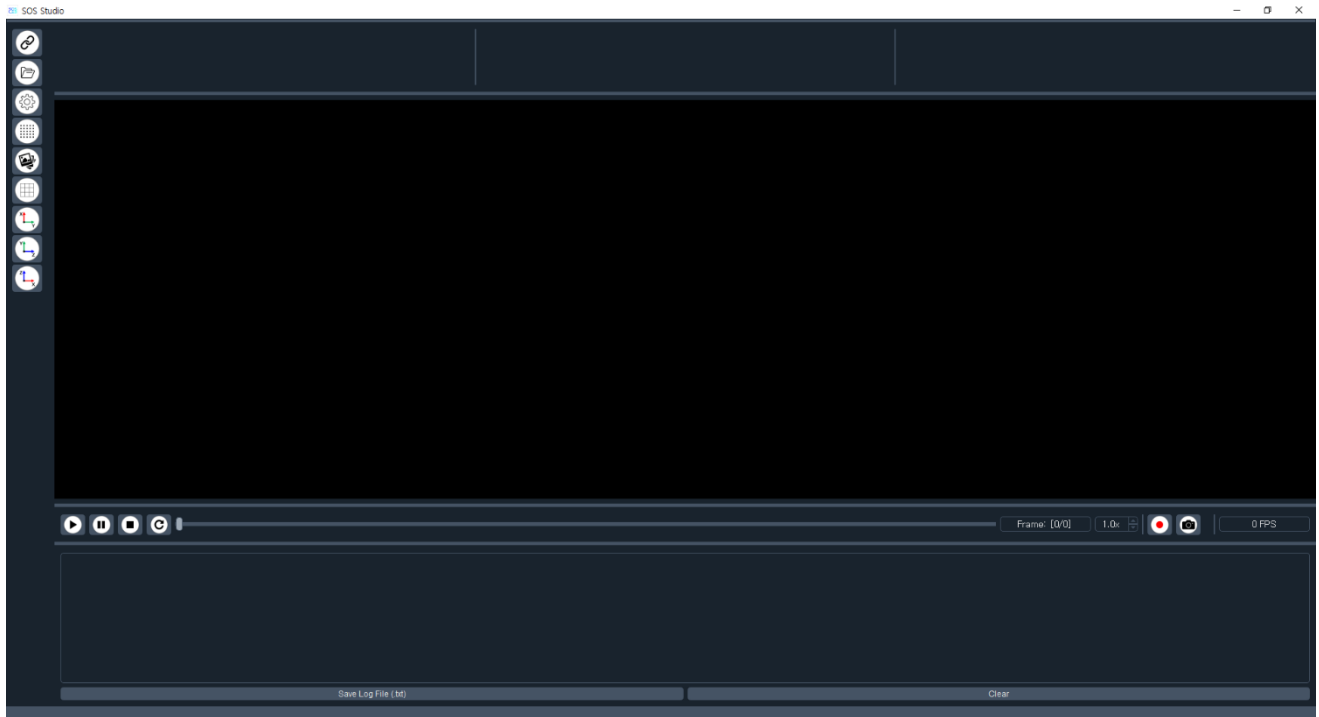
C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>
```

명령 프롬프트 창 및 ping 테스트 성공 결과 예시




### 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.



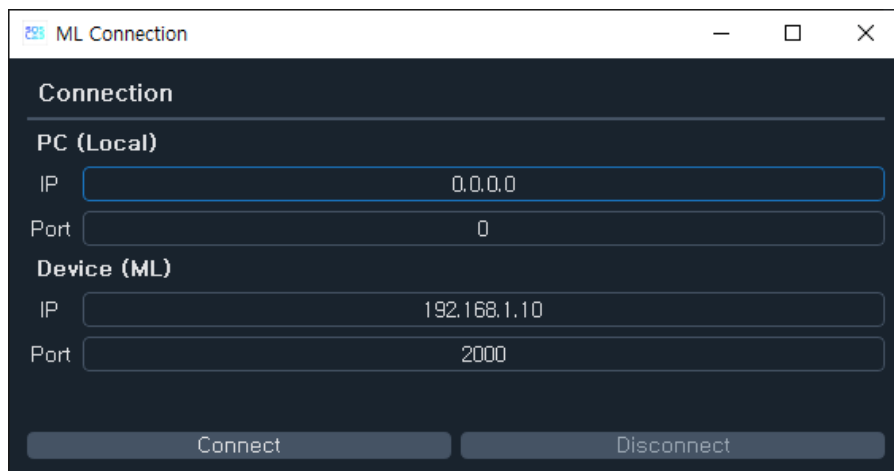
ML-X list

## 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.



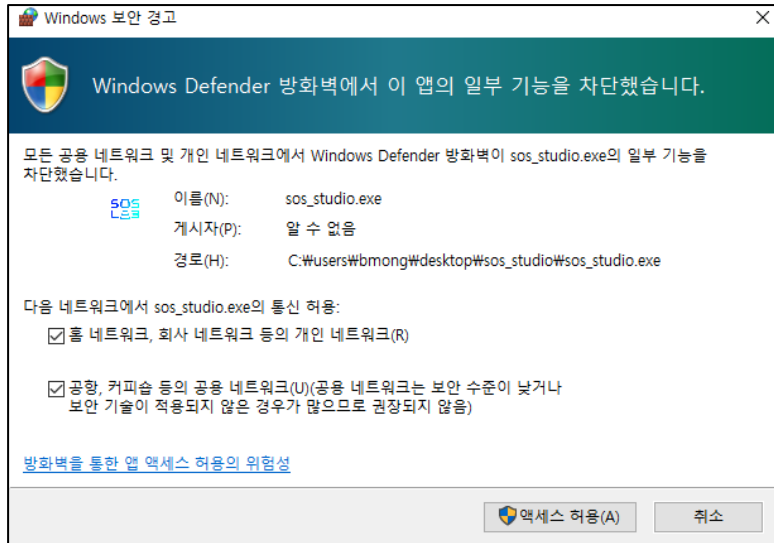
ML-X list



Connection 팝업 창

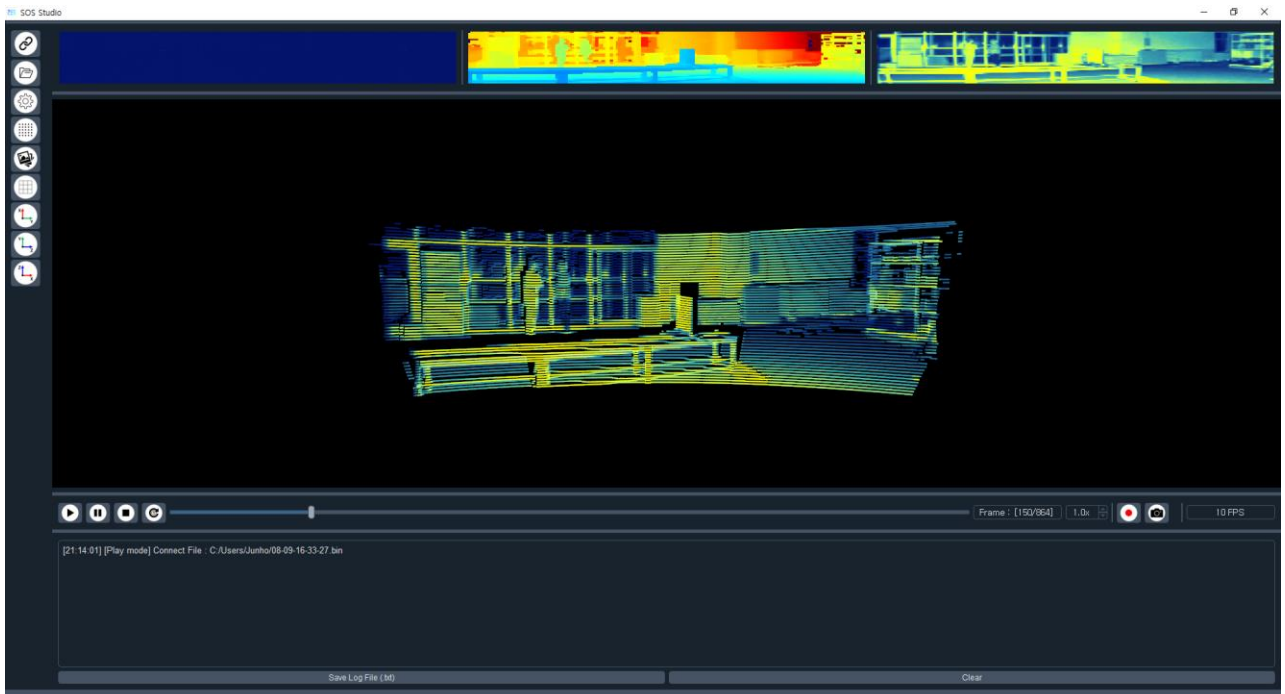
## 2.3 Connection

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.

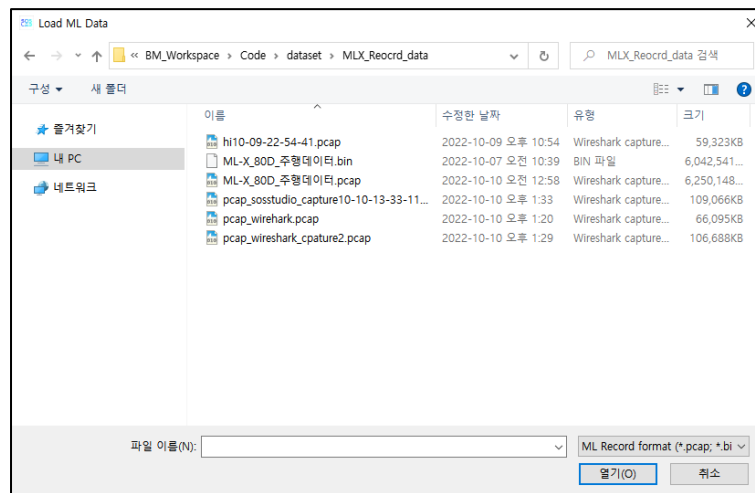


SOS Studio 실행화면

### 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 Mode에 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 (.pcap, .bin) 파일을 선택한 뒤, “열기(O)” 버튼을 클릭하면 ML-X 데이터를 재생할 준비가 완료됩니다.



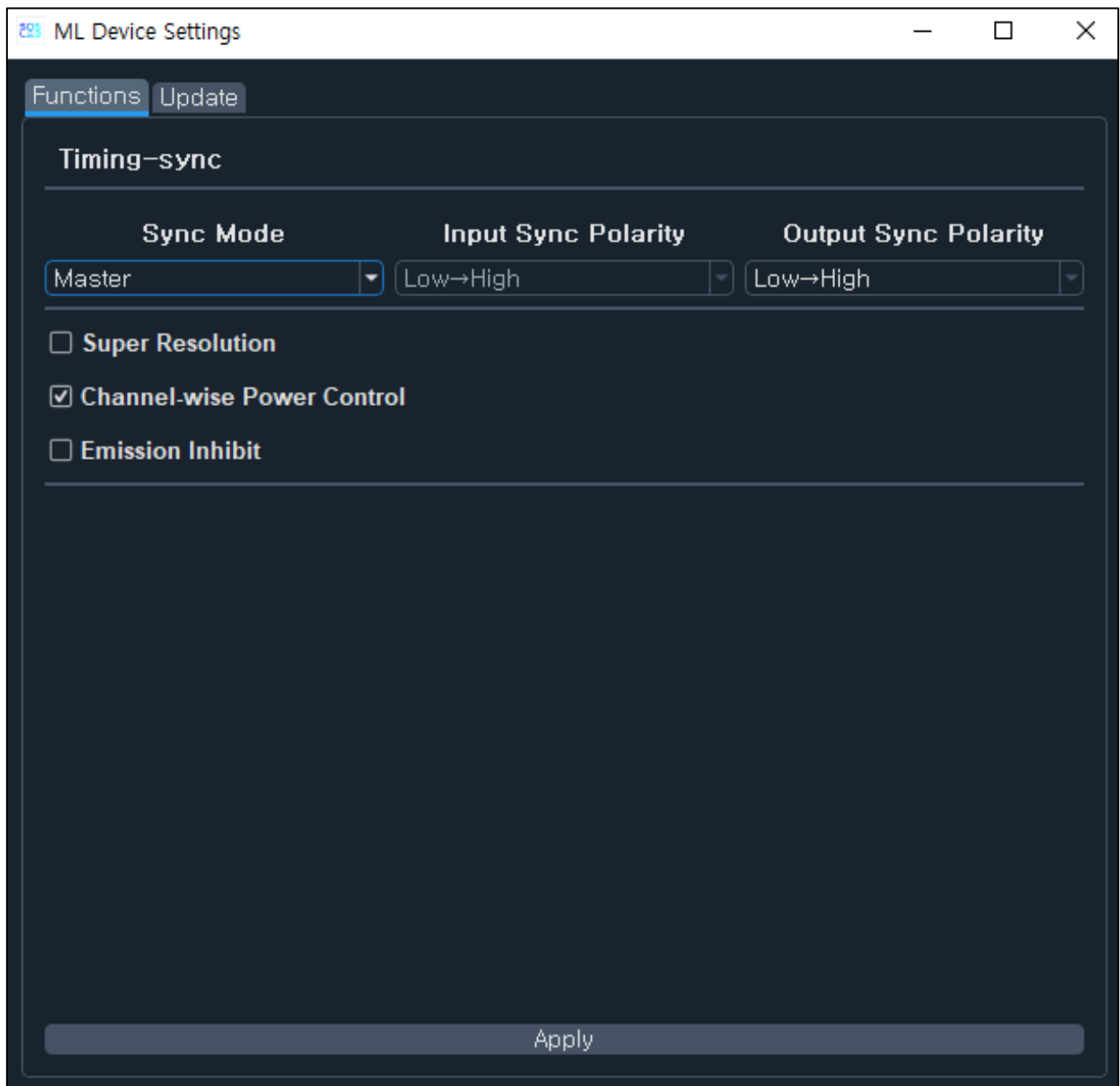
데이터 불러오기 팝업 창

데이터 불러오기가 완료되면 아래의 Play Bar 기능들을 통해 ML-X 데이터를 재생할 수 있습니다. Play Bar 기능들은 **2.8 Play / Record Mode**에 자세히 기술되어 있습니다.

### 2.5 Device Setting



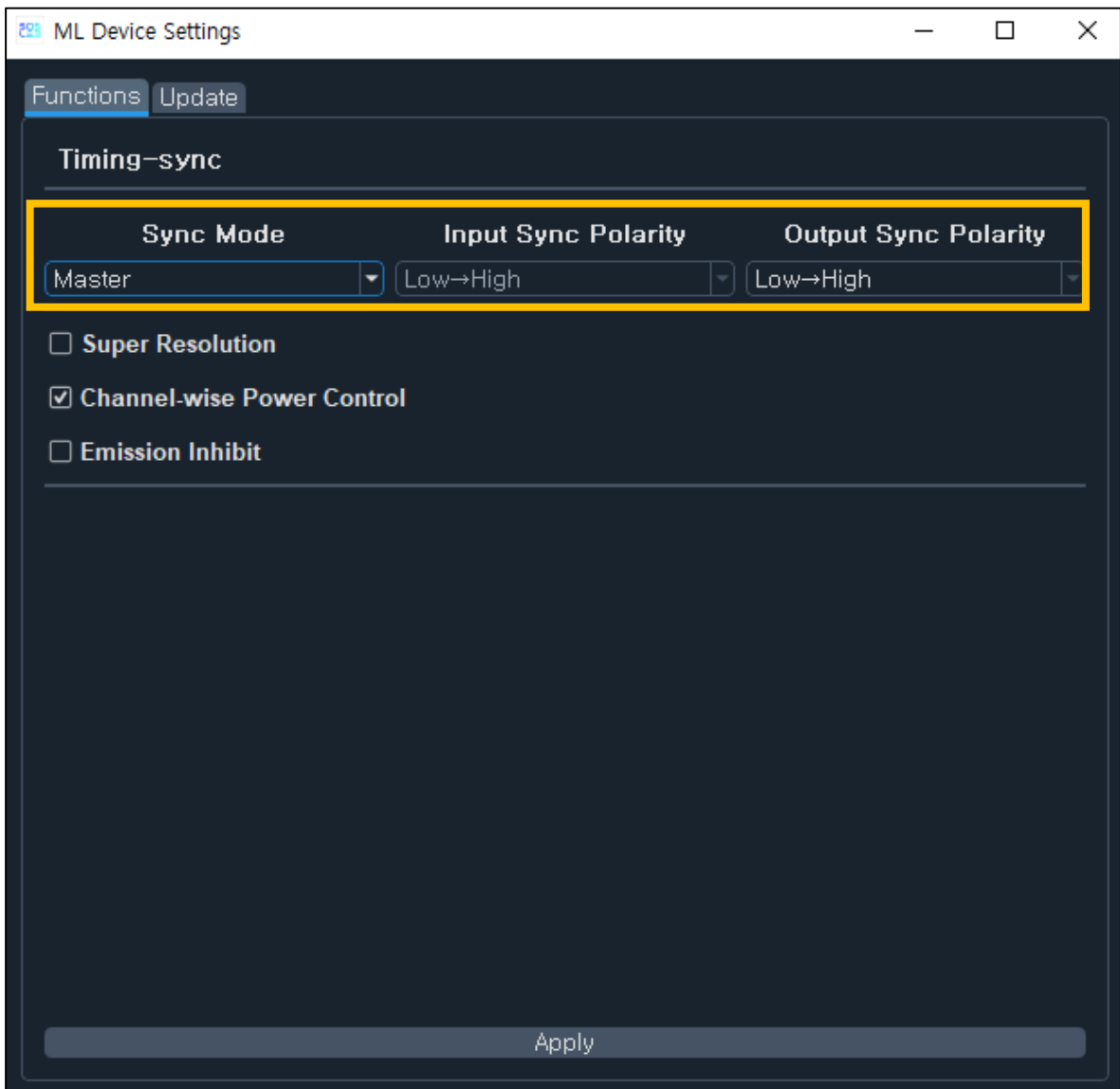
Device Setting에서는 ML-X Device의 Timing Sync 기능, Super Resolution on/off 기능과 IP 변경 기능, FW 업데이트 기능을 제공합니다. Timing Sync 기능과 Super Resolution on/off 기능등은 “Functions” 탭에서 제공하며, IP 변경 기능과 FW 업데이트 기능은 “Update” 탭에서 제공합니다.



The screenshot shows a window titled "ML Device Settings" with two tabs: "Functions" (selected) and "Update". Under the "Functions" tab, there is a section titled "Timing-sync". This section contains three dropdown menus: "Sync Mode" (set to "Master"), "Input Sync Polarity" (set to "Low→High"), and "Output Sync Polarity" (set to "Low→High"). Below these are three checkboxes: "Super Resolution" (unchecked), "Channel-wise Power Control" (checked), and "Emission Inhibit" (unchecked). At the bottom of the window is an "Apply" button.

### 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 Polarity”를 설정할 수 있으며, “Slave” 모드에서는 “Input Sync Polarity”를 설정할 수 있습니다. 원하는 모드와 Polarity를 설정한 후에 “Apply” 버튼을 누르면 해당 모드로 설정됩니다. Timing Sync에 대한 자세한 설명은 [ML-X Hardware Configuration](#) 문서를 참조하시기 바랍니다.

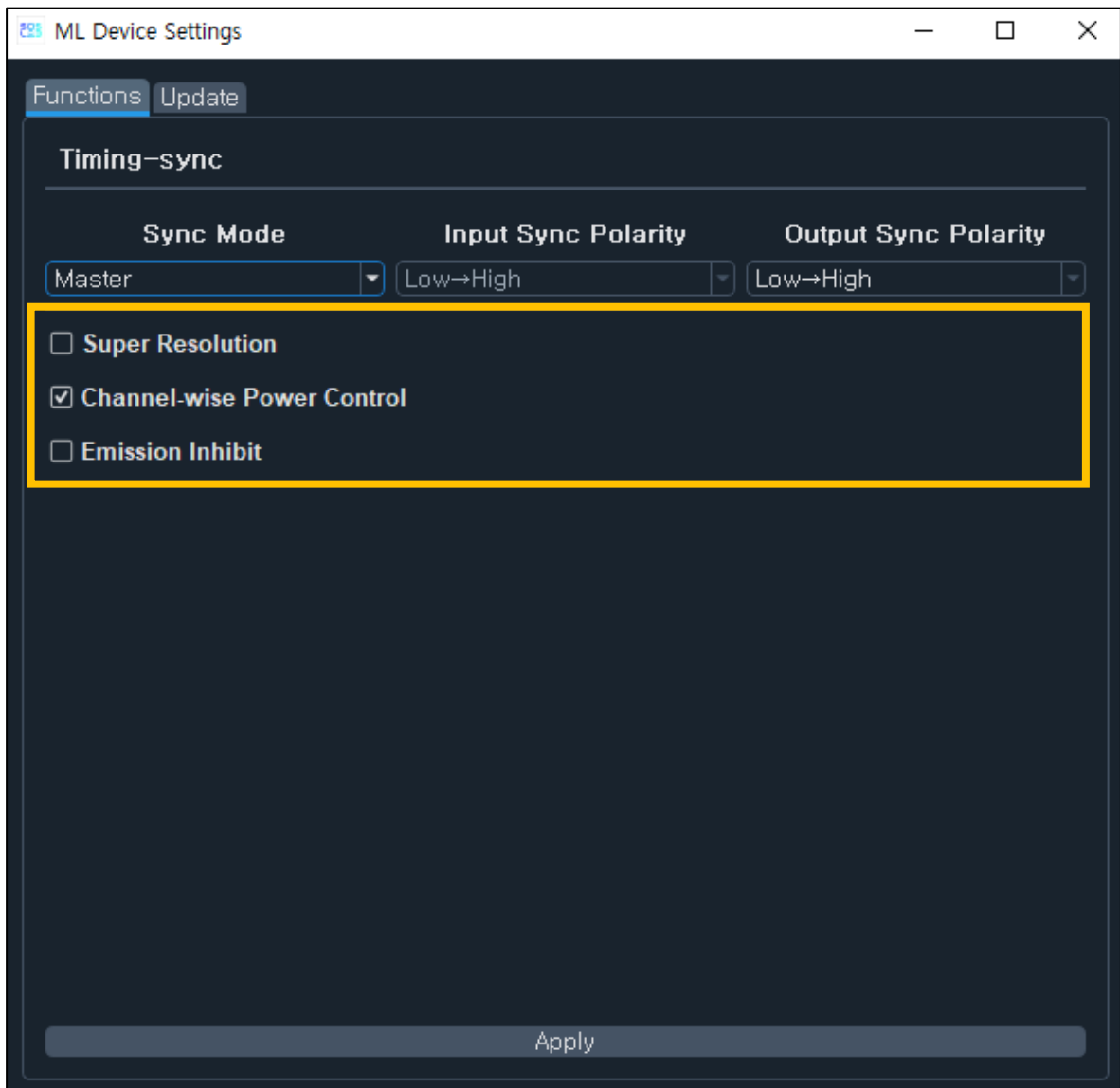


### 2.5.2 Functions on/off

ML-X Device의 Super Resolution / Channel-wise Power Control / Emission Inhibit

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. 초기 상태는 아래와 같습니다.

- Super Resolution : Off
- Channel-wise Power Control : On
- Emission Inhibit : Off

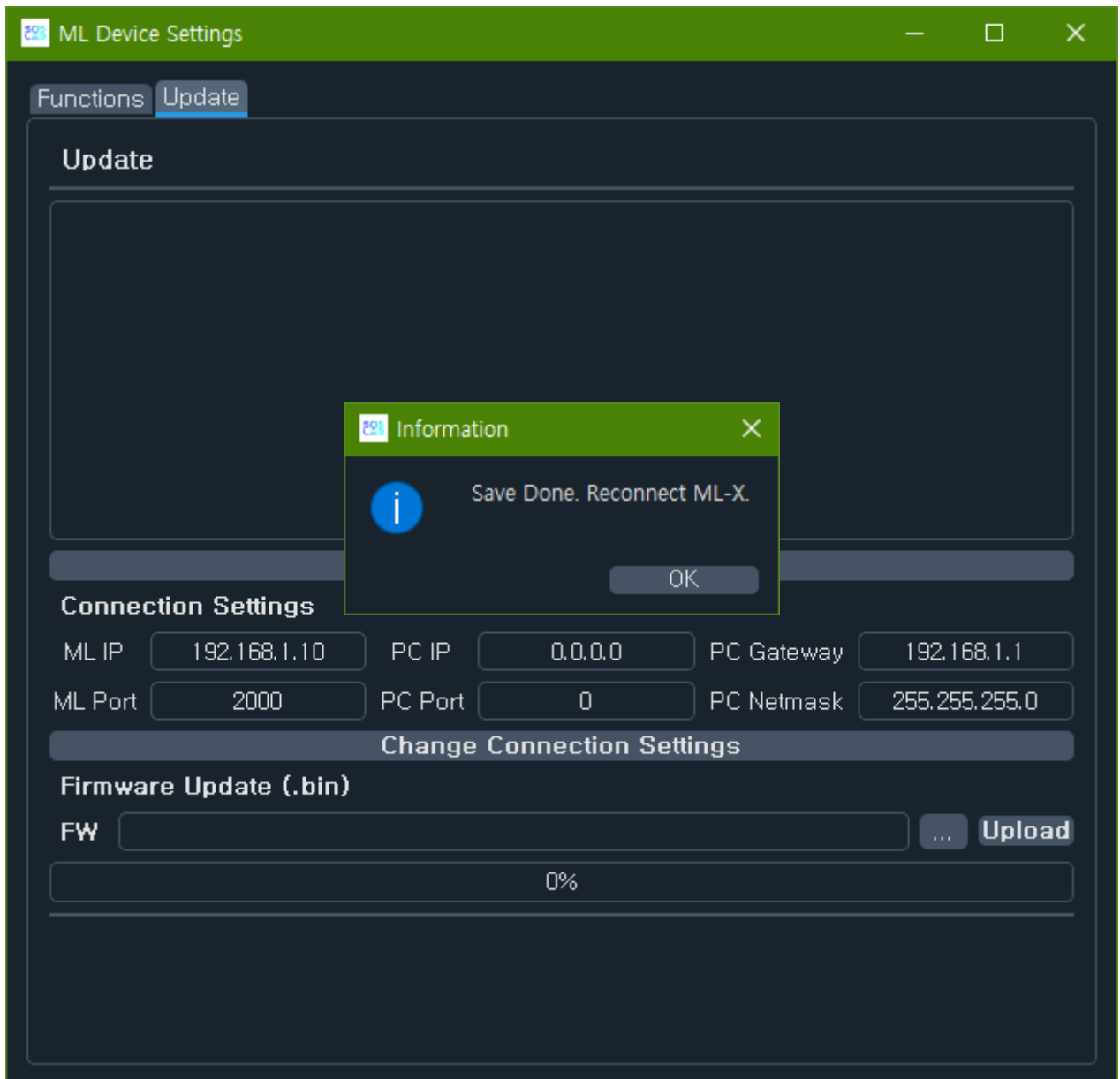


### 2.5.3 IP Changer

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

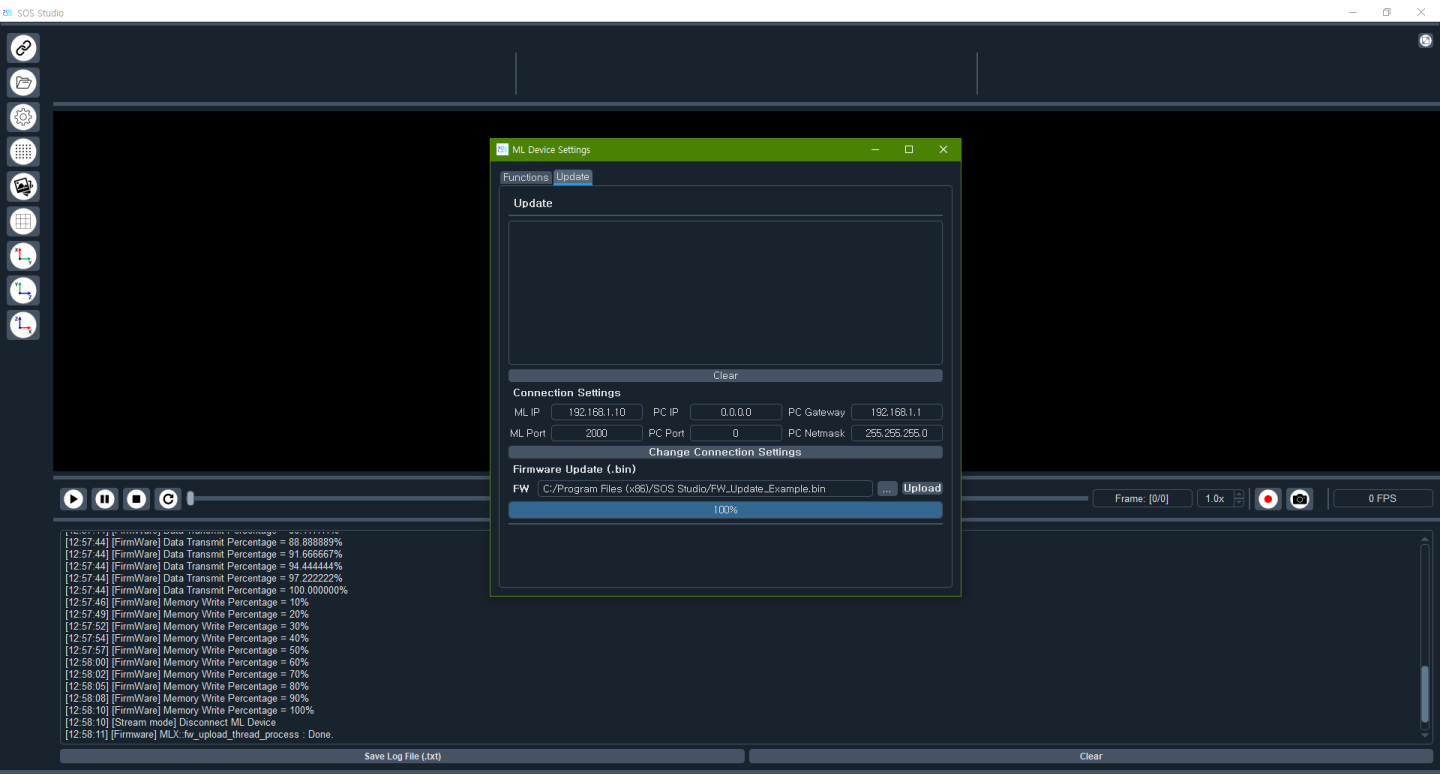


### 2.5.4 FW Update

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


ML-X Device의 FW Update는 아래의 과정들을 통해 수행할 수 있습니다.

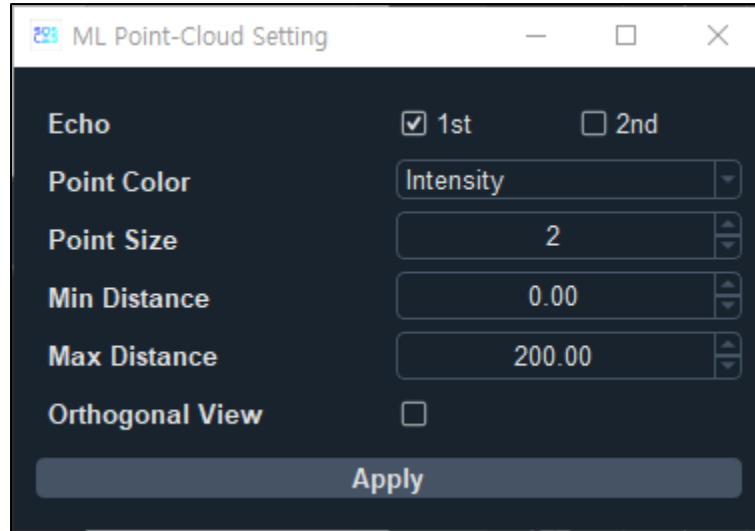
- ① “...” 버튼을 클릭합니다.
- ② FW Update할 \*.bin 파일을 선택합니다.
- ③ “Upload” 버튼을 클릭합니다.
- ④ Progress bar가 100%로 표시되면 FW Update 완료.
- ⑤ FW Update 완료 후 전원 케이블을 재연결 합니다.



FW Update 완료 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
<b>Echo</b>	Multi Echo Point Cloud 가시화 설정(First peak, Second peak)
<b>Point Color</b>	Point 색상 (White, Red, Green, Blue, Ambient, Depth, Intensity)
<b>Point Size</b>	Point 크기
<b>Min Distance</b>	가시화 되는 Point 최소 거리
<b>Max Distance</b>	가시화 되는 Point 최대 거리
<b>Orthogonal View</b>	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.

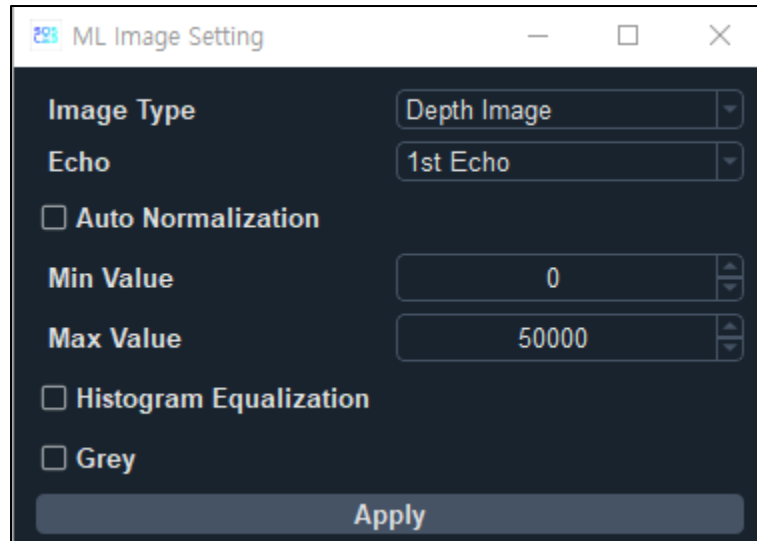



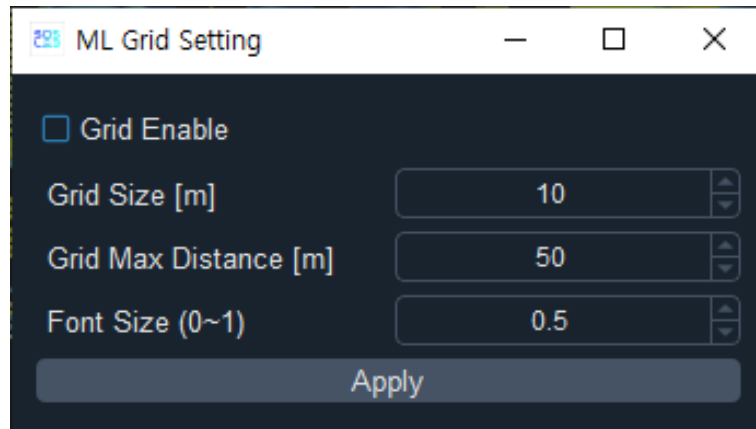
Image Setting 팝업 창

설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

항목	설명
<b>Image Type</b>	설정 이미지 선택 (Ambient / Depth / Intensity Image)
<b>Echo</b>	Multi Echo Image 가시화 설정(First peak, Second peak)
<b>Auto Normalization</b>	자동 정규화(Normalization): 이미지의 최소/최대 값으로 정규화
<b>Min Value</b>	정규화 최소 값
<b>Max Value</b>	정규화 최대 값
<b>Histogram Equalization</b>	Histogram Equalization 활성화
<b>Grey</b>	흑백 이미지로 변경

### 2.8 Grid Setting

 Grid Setting에서는 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

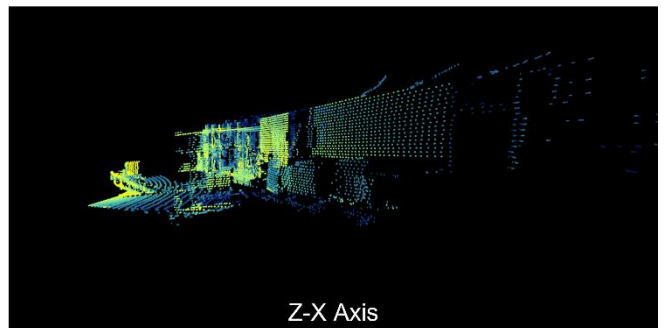
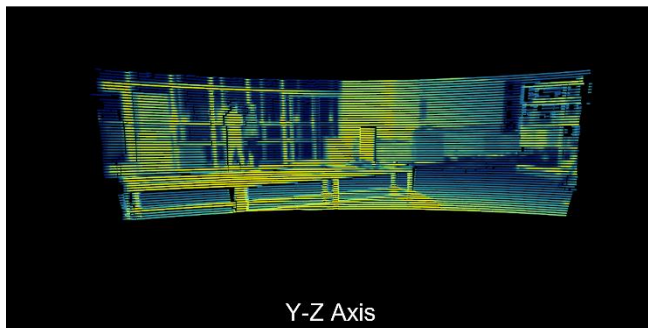
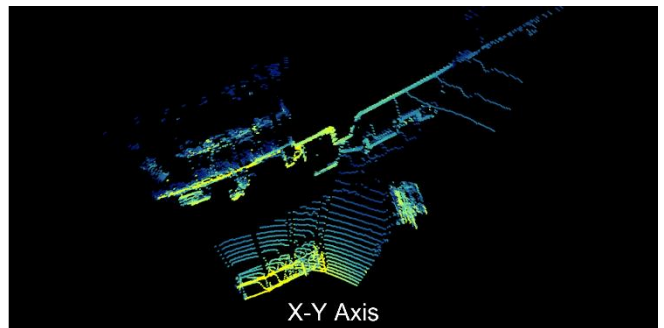
설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

항목	설명
<b>Grid Enable</b>	Grid 가시화 활성화
<b>Grid Size</b>	Grid 간격
<b>Grid Max Distance</b>	Grid 최대 거리
<b>Font Size</b>	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 **2.2 Data Load**에서 불러온 ML-X 데이터를 재생하는 기능 및 연결 후, 가시화 되는 데이터를 녹화하는 기능을 제공한다.



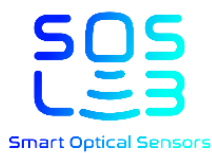
SOS Studio Play Bar

Play Bar에 각 기능들에 명칭 및 기능은 아래와 같습니다.

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

# Chapter 3

## ML-X LiDAR API



## 3.1. ML-X Example Code Build for Ubuntu

ML-X SDK는 Ubuntu 18.04 / 20.04 및 ROS는 Melodic / Noetic 버전에서 사용 가능하며, ML-X API와 API를 활용한 예제 코드를 함께 제공합니다.

ML-X API는 libsoslab\_core, libsoslab\_ml 2가지로 구성되며, test\_ml-x 예제 코드를 통해 ML-X 연결 및 데이터 가시화를 테스트할 수 있습니다.

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

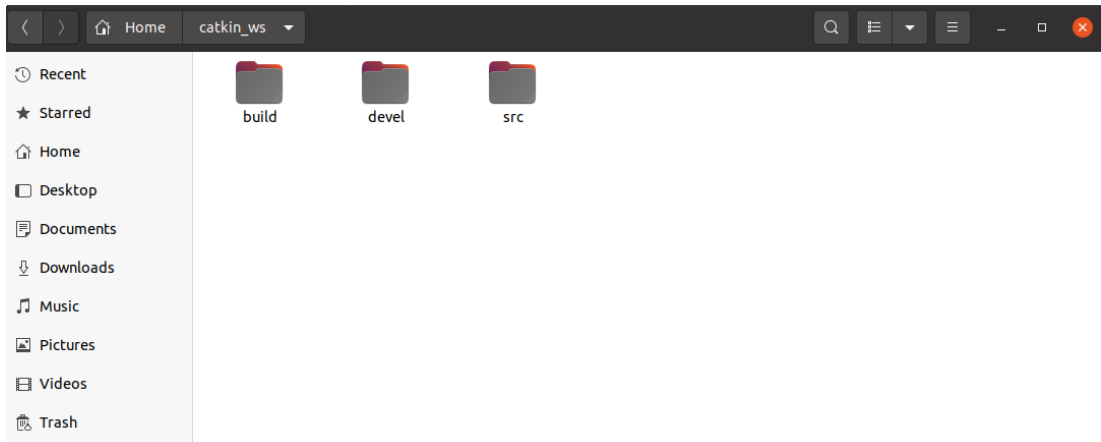
ML-X API 빌드는 ROS 설치가 완료 된 후, 아래 과정들을 통해 진행할 수 있습니다.

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$
```

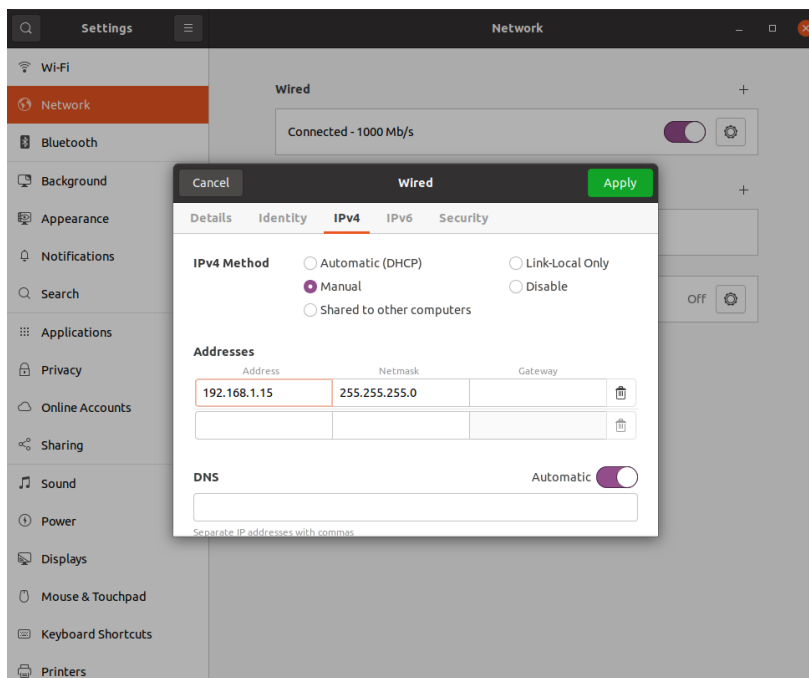
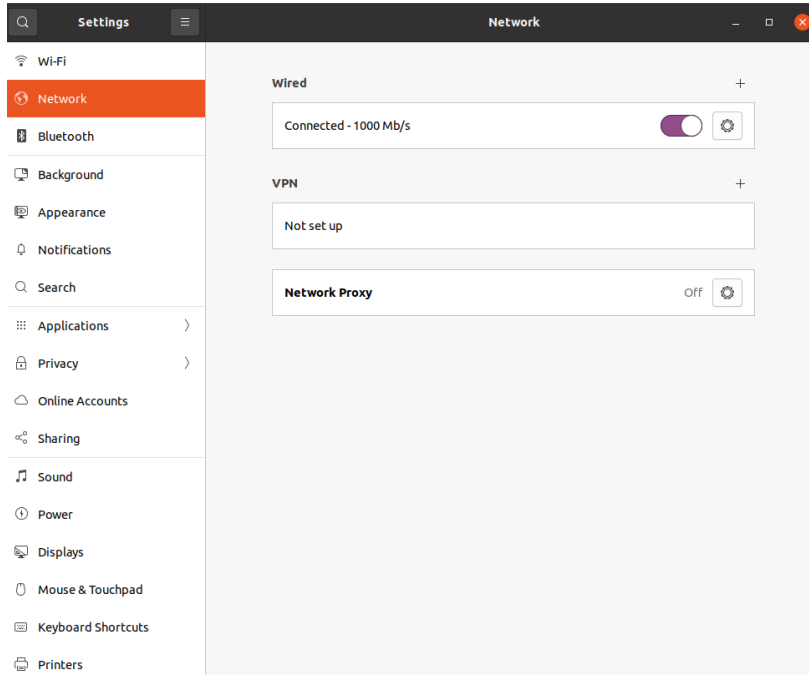
ml package를 ROS 환경에 추가



## 3.1. ML-X Example Code Build for Ubuntu

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

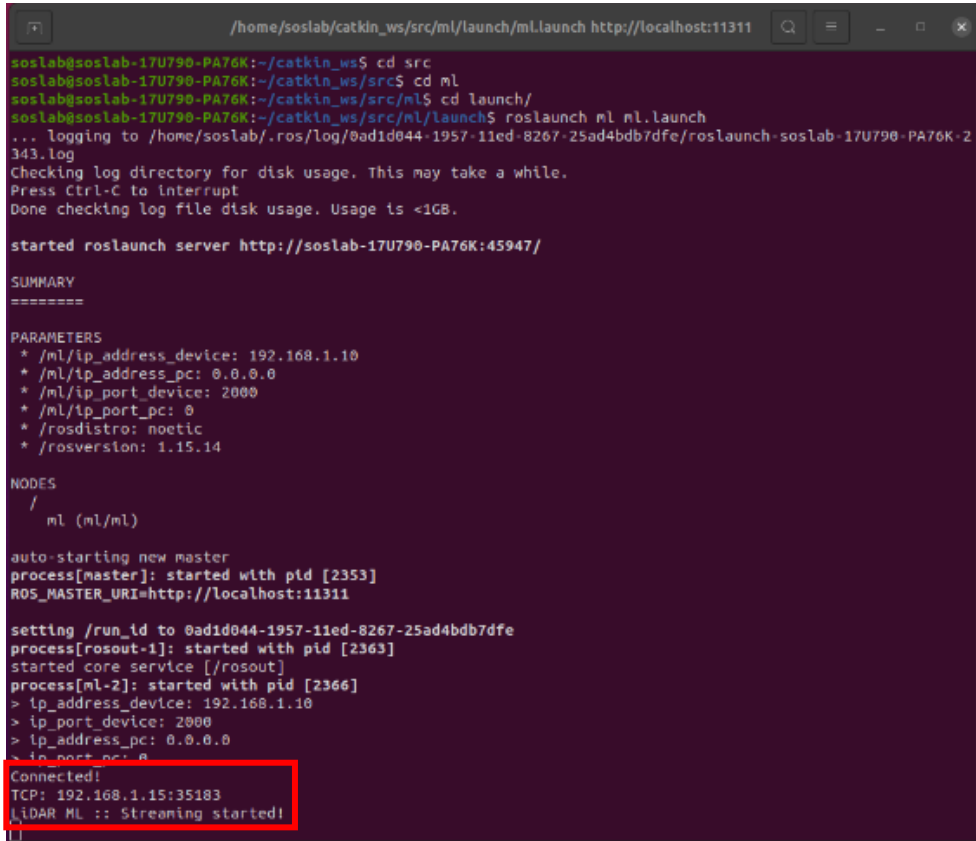


IPv4 설정 변경

## 3.1. ML-X Example Code Build for Ubuntu

- 4) Network 설정 후, 아래 command를 입력하여 ml.launch 파일을 실행하여 PC와 ML-X Device를 연결 합니다. 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
```



```
/home/soslab/catkin_ws/src/ml/launch/ml.launch http://localhost:11311
soslab@soslab-17U790-PA76K:~/catkin_ws$ cd src
soslab@soslab-17U790-PA76K:~/catkin_ws/src$ cd ml
soslab@soslab-17U790-PA76K:~/catkin_ws/src/ml$ cd launch/
soslab@soslab-17U790-PA76K:~/catkin_ws/src/ml/launch$ roslaunch ml ml.launch
... logging to /home/soslab/.ros/log/0ad1d044-1957-11ed-8267-25ad4bdb7dfe/roslaunch-soslab-17U790-PA76K-2
343.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://soslab-17U790-PA76K:45947/

SUMMARY
=====
PARAMETERS
* /ml/ip_address_device: 192.168.1.10
* /ml/ip_address_pc: 0.0.0.0
* /ml/ip_port_device: 2000
* /ml/ip_port_pc: 0
* /rostdistro: noetic
* /rosversion: 1.15.14

NODES
/
  ml (ml/ml)

auto-starting new master
process[master]: started with pid [2353]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 0ad1d044-1957-11ed-8267-25ad4bdb7dfe
process[rosout-1]: started with pid [2363]
started core service [/rosout]
process[ml-2]: started with pid [2366]
> ip_address_device: 192.168.1.10
> ip_port_device: 2000
> ip_address_pc: 0.0.0.0
> ip_port_pc: 0
Connected!
TCP: 192.168.1.15:35183
LIDAR ML :: Streaming started!
```

Ubuntu 환경에서 ROS를 통해 ML-X Device 연결

### 3.2. Example Code for Ubuntu/ROS

예제 코드 test\_ml-x에는 ML-X 연결 및 rviz에서의 데이터 가시화 기능이 구현되어 있으며, 코드에 대한 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_t`로 획득되며, 총 4개의 1차원 데이터 배열(ambient, intensity, depth, pointcloud)로 구성되어 있습니다. 자세한 `scene_t` 구조체에 대한 설명은 Appendix에 기재되어 있습니다.

Line	Description
2	Raw Data를 저장할 <code>SOSLAB::LidarML::scene_t</code> 변수를 <code>scene</code> 변수명으로 선언합니다.
3	<code>scene_t</code> 변수를 입력으로 받는 <code>SOSLAB::LidarML</code> Class의 <code>get_scene</code> 함수를 통해 <code>scene</code> 변수에 Raw Data를 획득합니다.
4-7	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 변수명 및 변수형은 Appendix에 기재되어 있습니다.
8-10	<code>scene_t</code> <code>scene</code> 구조체로부터 <code>Depth</code> , <code>Intensity</code> 이미지의 <code>height</code> , <code>width</code> 값을 획득합니다. <code>width2</code> 변수는 <code>Ambient</code> 이미지의 <code>width</code> 값입니다.

## 3.2. 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_msgs::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_msgs::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 <b>image transport</b> , <b>ImagePtr</b> 등의 객체를 생성합니다. Publisher는 Topic "ambient" 로 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(scene.rows), Width(scene.cols), Ambient 데이터 Type( <b>CV_32SC1</b> ), 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 <b>cv::Mat</b> 객체를 Publisher Node의 Message 형태로 저장하기 위해 <b>ImagePtr</b> 로 변환하는 과정입니다.
9	Topic이 "ambient_color"로 지정된 <b>Publisher</b> 객체의 publish 함수에 <b>ImagePtr</b> 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) 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 <b>image transport</b> , <b>sensor_msg</b> 객체를 생성합니다. Publisher는 Topic “pointcloud”로 Point Cloud 데이터를 제공합니다.
5-9	Message 변수를 정의하여 데이터 크기, 이름을 초기화합니다.
10-14	<b>scene_t</b> 의 pointcloud 데이터를 msg_pointcloud 변수로 복사하고, 거리 단위를 mm에서 m로 변경합니다.
15-16	Point Cloud가 Scanning된 timestamp를 저장한 뒤, Topic이 “pointcloud”로 지정된 <b>Publisher</b> 객체의 publish 함수에 PointCloud_T::Ptr 변수를 입력 인수로 사용하여 Publish를 완료합니다.

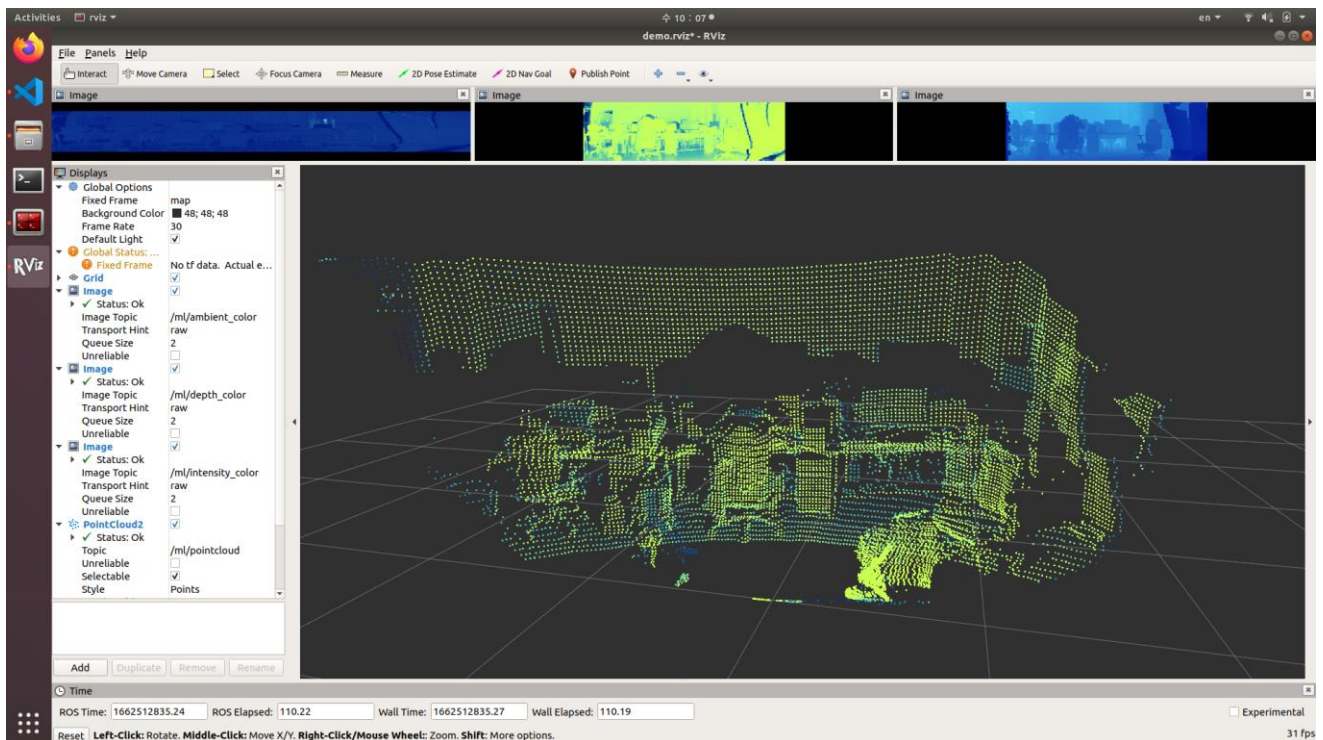
## 3.2. Example Code for Ubuntu/ROS

### 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
```

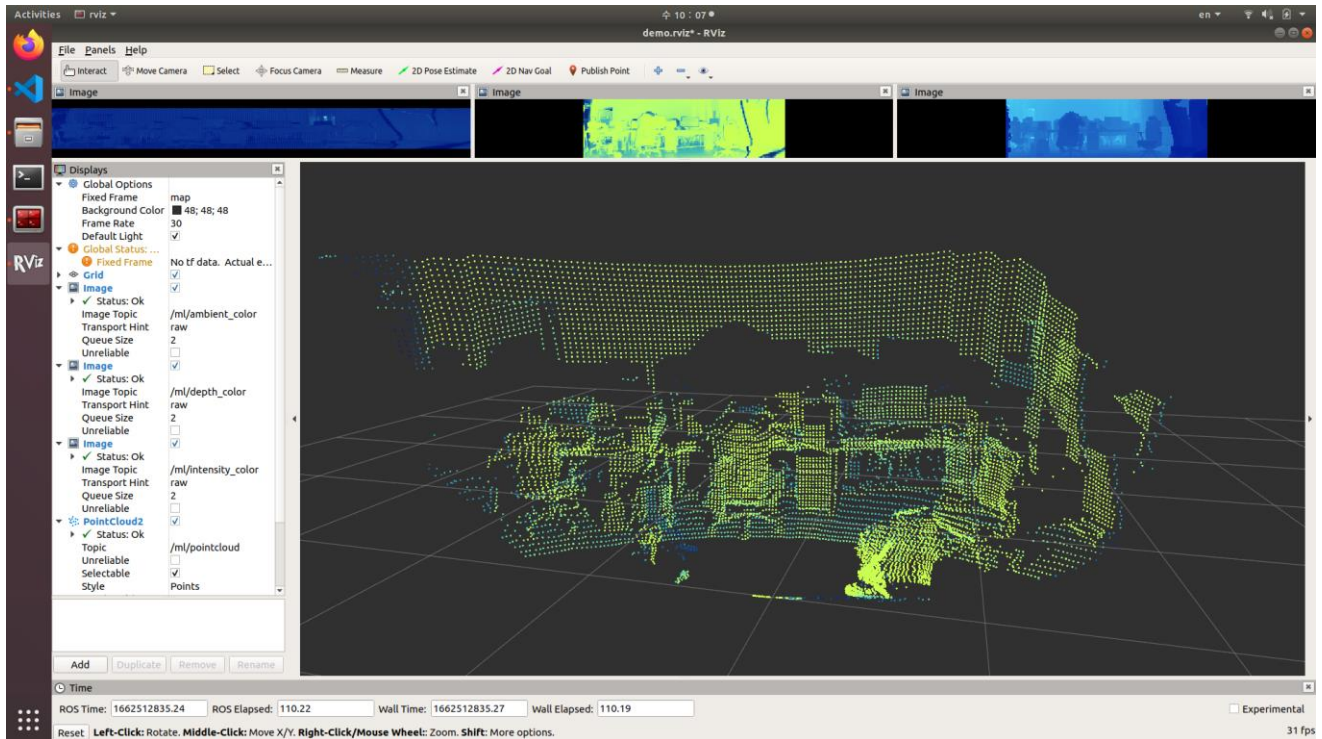
아래 그림과 같이 왼쪽 하단의 Add 버튼을 클릭하면 아래 그림과 같은 창을 확인할 수 있습니다. 해당 창의 상단 “By topic” 메뉴를 클릭하면 publish 되고 있는 전체 topic을 확인할 수 있습니다.



Rviz 프로그램 실행 화면

## 3.2. 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



## 3.2. Example Code for Ubuntu/ROS

### 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.

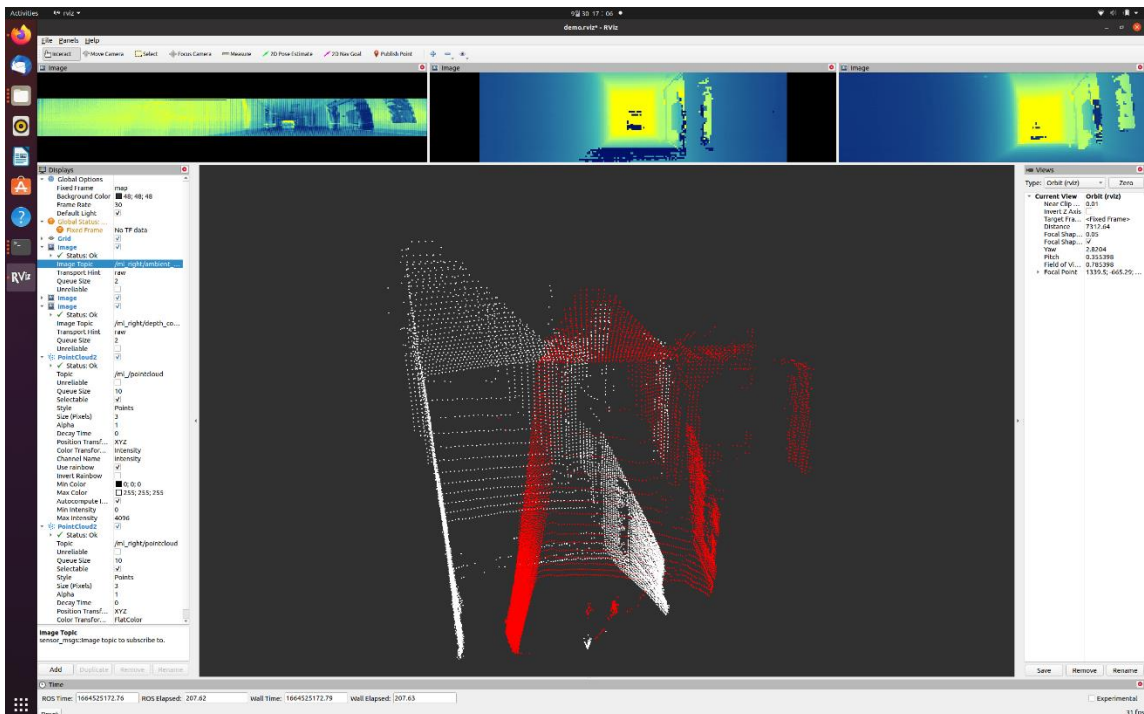
#### [Terminal #1]

```
$ 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

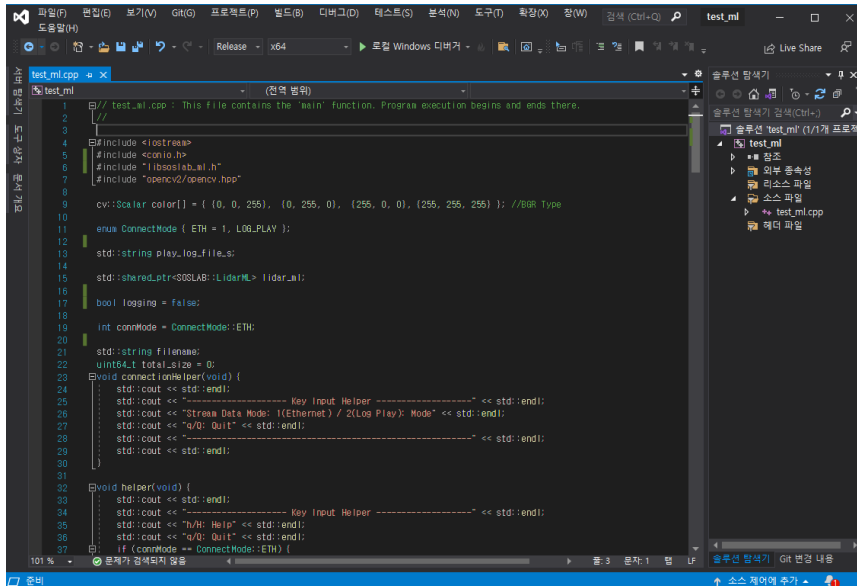


## 3.3. ML-X Example Code Build for Window

### 3.3. ML-X Example Code Build for Window

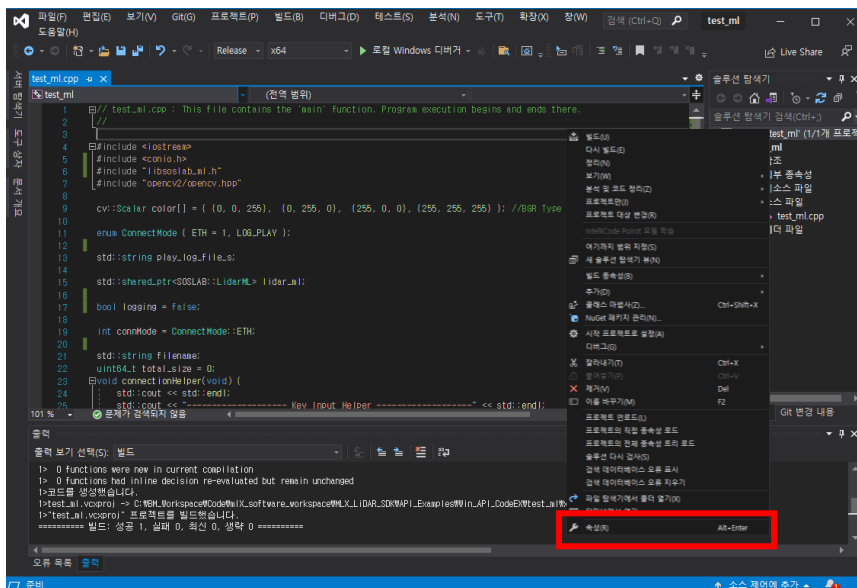
Window 환경의 ML-X Example 코드는 ML-X API와 OpenCV 라이브러리 통해 데이터를 가시화하는 예제입니다. 빌드 과정은 다음과 같습니다.

1) MLX\_LiDAR\_SDK/API\_Examples/Win\_API\_CodeEX/test\_ml/test\_ml.sln 을 클릭하여 Visual Studio 2019를 실행합니다.



test\_ml.sln 실행 화면

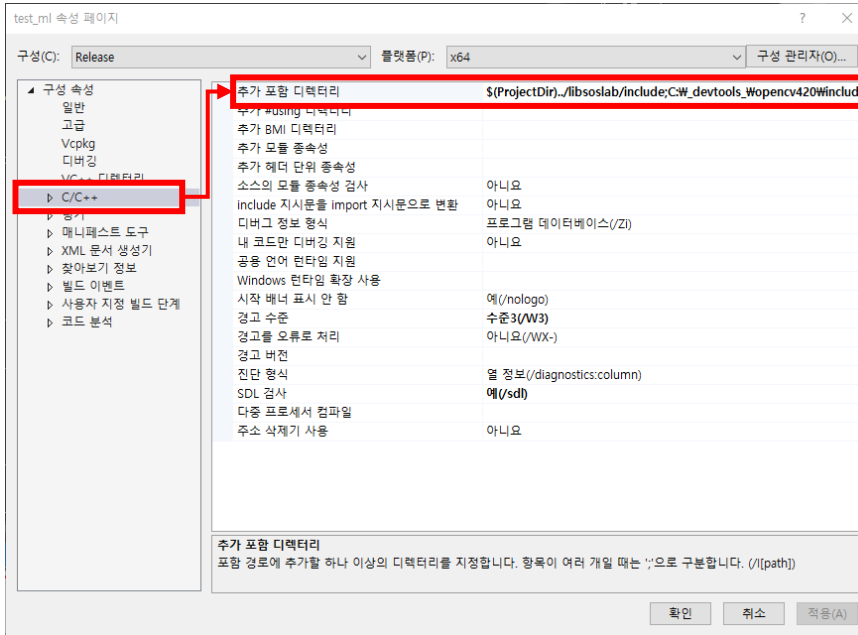
2) 아래 그림과 같이 프로젝트 속성을 클릭 또는 단축키(Alt + Enter)를 입력합니다.



프로젝터 속성 위치

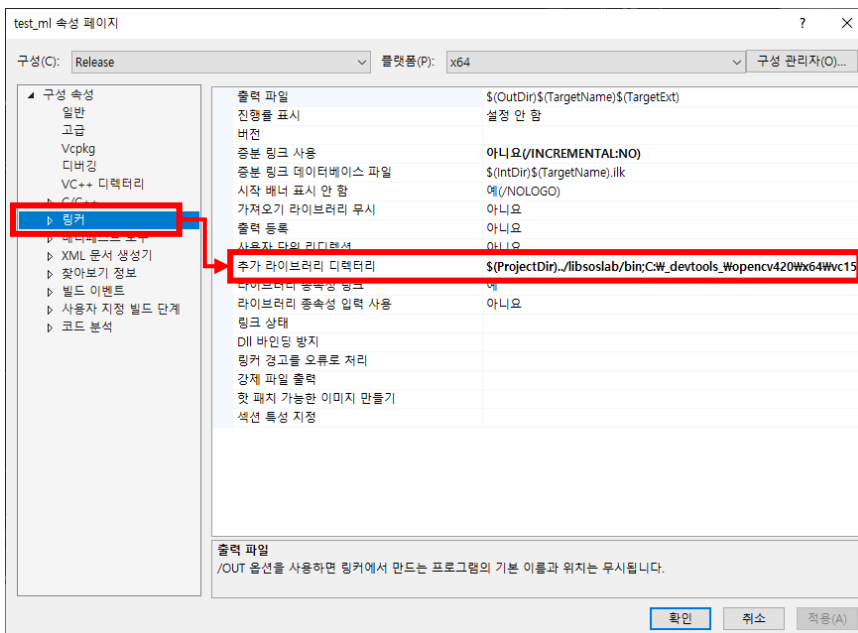
### 3.3. ML-X Example Code Build for Window

3) 아래 그림과 같이 왼쪽의 [C/C++] 항목에서 [추가 포함 디렉터리]에 OpenCV 라이브러리 Include 위치를 추가합니다.



test\_ml 실행을 위한 속성 창의 추가 포함 디렉터리

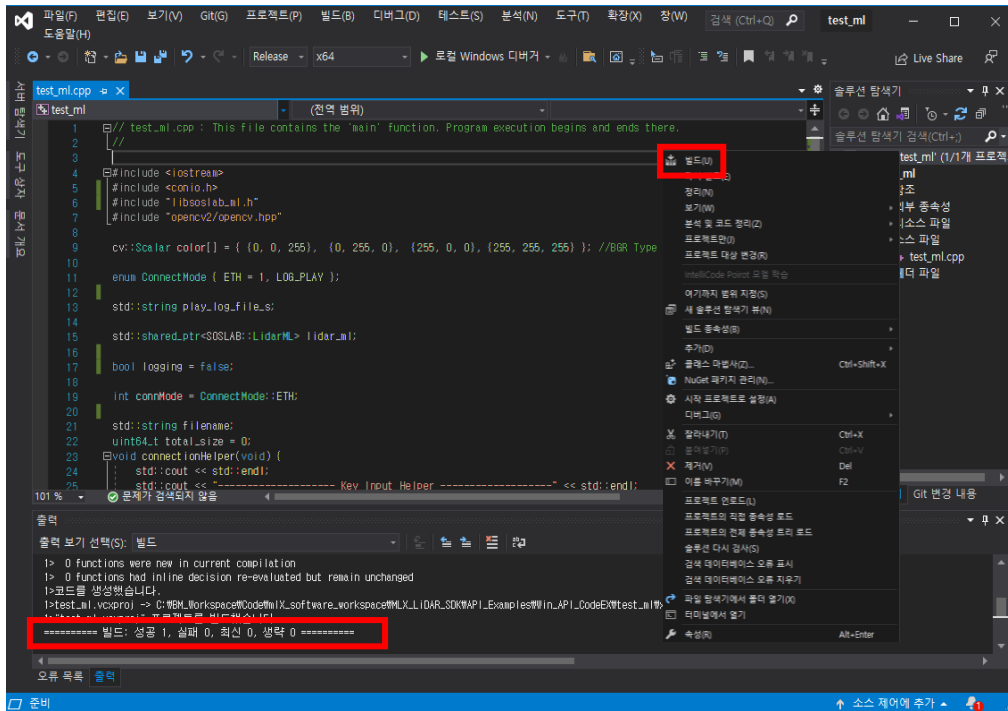
4) 아래 그림과 같이 왼쪽의 [링크] 항목에서 [추가 라이브러리 디렉터리]에 OpenCV 라이브러리 위치를 추가합니다



test\_ml 실행을 위한 속성 창의 추가 라이브러리 디렉터리

### 3.3. ML-X Example Code Build for Window

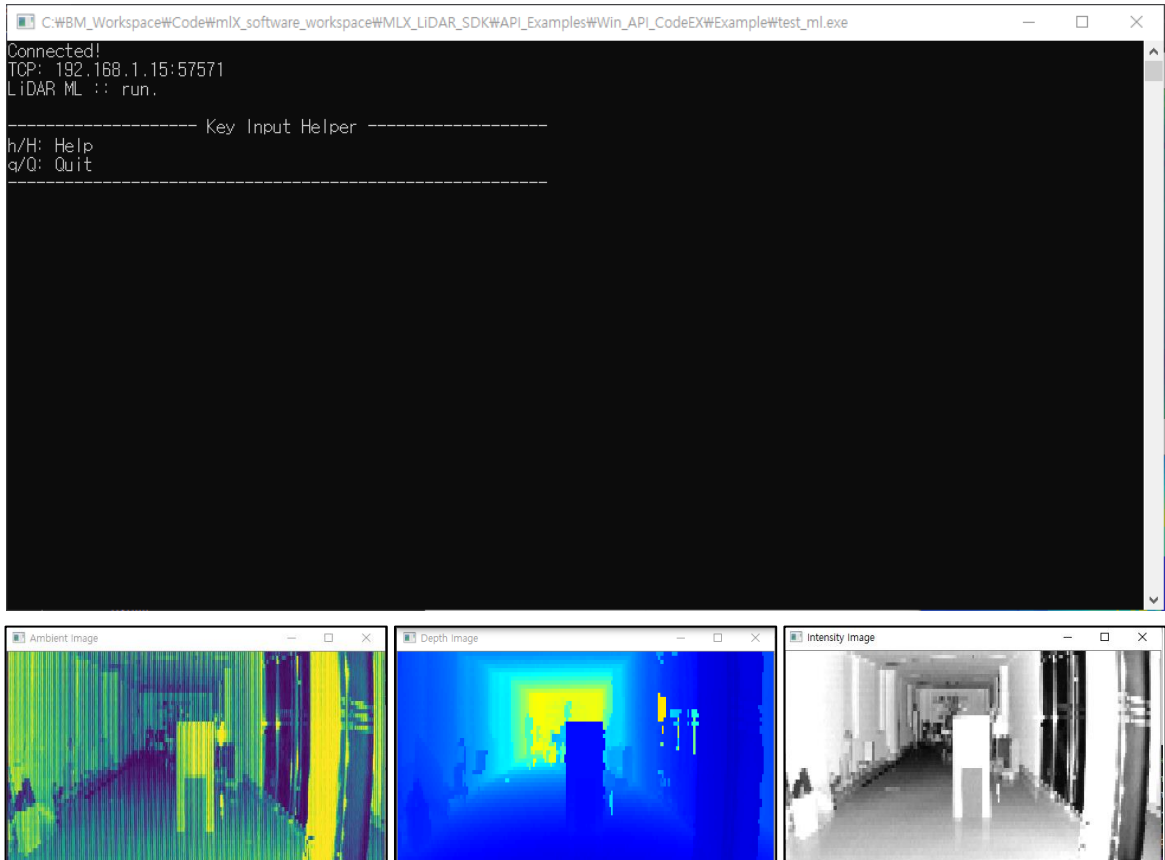
5) 속성 변경 후, 예제 코드를 빌드하여 출력 창에 “성공 1, 실패 0”이 출력되면 test\_ml 예제 코드가 빌드 되어 MLX\_LiDAR\_SDK / API\_Examples / Win\_API\_CodeEX / Example 폴더에 test\_ml.exe 실행 파일이 생성됩니다.



test\_ml 빌드 결과

### 3.3. ML-X Example Code Build for Window

6) test\_ml.exe을 실행하면 아래 그림과 같이 Ambient / Depth / Intensity 이미지가 아래 그림과 같이 가시화됩니다.



test\_ml 실행 결과: 터미널 (상), Ambient / Depth / Intensity 이미지 가시화 (하)

### 3.4. Example Code for Window

예제 코드 test\_ml에는 ML-X 연결 및 rviz에서의 데이터 가시화 기능이 구현되어 있으며, 코드에 대한 설명은 아래와 같습니다.

#### 1) Raw Data 획득

```
1. while (keyinput_checker(cv::waitKey(1))) {
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.    }
11.}
```

Raw Data 획득 코드

ML-X Device로부터 Raw Data를 획득하는 방법에 대한 코드입니다. ML-X Device의 Raw Data는 `scene_t`로 획득되며, 총 4개의 1차원 데이터 배열(ambient, intensity, depth, pointcloud)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.
8-9	scene_t scene 구조체로부터 이미지의 height, width 값을 획득합니다.

## 3.4. Example Code for Window

### 2) Ambient, Depth, Intensity Image 변환

```
1. cv::Mat ambient_image_raw(scene.rows, scene.cols, CV_32SC1,
    ambient.data());
2. cv::Mat ambient_rgb;
3. ambient_image_raw.convertTo(ambient_rgb, CV_8UC1, (255.0 / 2000),0);
4. cv::applyColorMap(ambient_rgb, ambient_rgb, cv::COLORMAP_VIRIDIS);
5. cv::imshow(ambient_viz_name, ambient_rgb);
```

Raw data에서 Ambient 이미지로 변환

이미지 가시화를 위해 ML-X Device로부터 획득한 Ambient, Depth, Intensity 데이터를 이미지로 변환하고 가시화를 수행하는 코드입니다. 위 코드는 Ambient 이미지 변환 및 가시화 코드입니다.

Line	Description
1	<code>scene_t</code> 구조체의 Ambient 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> ), Ambient 데이터 Type( <code>CV_32SC1</code> ), Ambient Data 값을 입력합니다.
3	이미지 가시화를 위하여 최대 값(2000)과 최소 값(0)을 0~255 값의 8bit(uchar) 형태로 변환합니다.
4	OpenCV의 <code>applyColorMap</code> 함수를 통해 <code>cv::Mat</code> 객체의 색상을 생성합니다. 본 예제에서는 VIRIDIS colormap을 사용합니다.
5	<code>cv::imshow</code> 함수를 통해 Ambient 이미지를 가시화합니다.

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

- Ambient : CV\_32SC1
- Depth : CV\_32SC1
- Intensity : CV\_16UC1

# Appendix

## Typedefs

이름	SOSLAB:: <code>INT_T</code>
선언	<code>typedef int32_t SOSLAB::INT_T</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	integer 변수

이름	SOSLAB:: <code>UINT_T</code>
선언	<code>typedef uint32_t SOSLAB::UINT_T</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	unsigned int 변수형

이름	SOSLAB:: <code>FLOAT_T</code>
선언	<code>typedef double SOSLAB::FLOAT_T</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	double 변수형

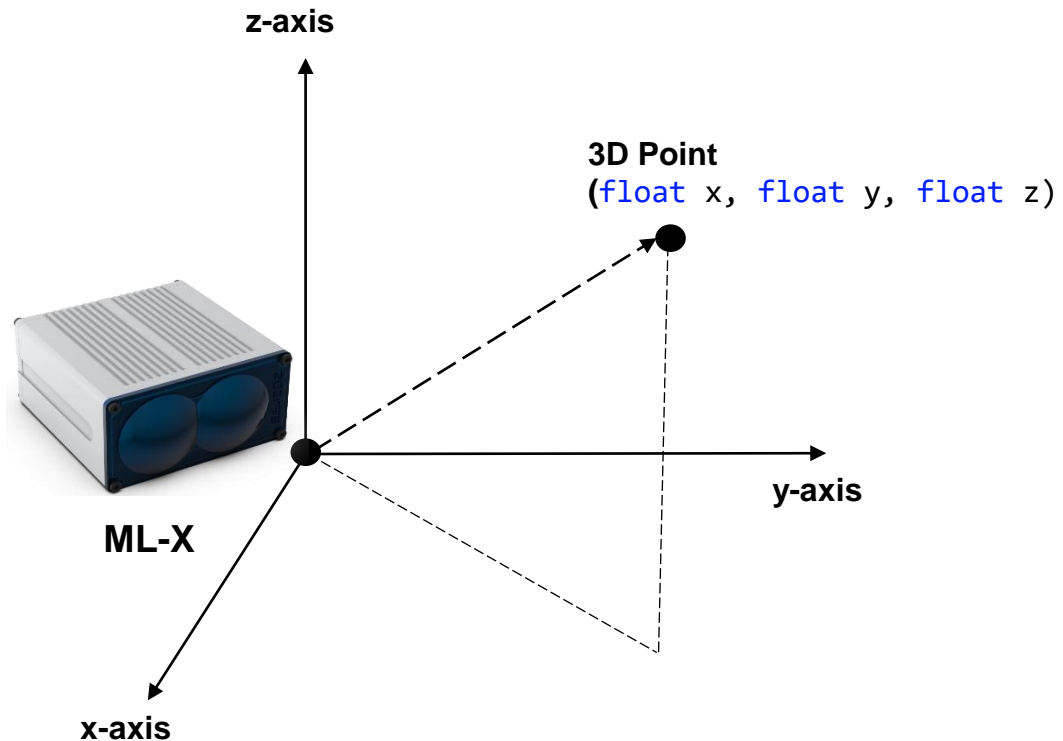
이름	SOSLAB:: <code>hex_array_t</code>
선언	<code>typedef std::vector&lt;uint8_t&gt; SOSLAB::hex_array_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	16진수 배열 변수형

이름	SOSLAB:: <code>pointcloud_t</code>
선언	<code>typedef std::vector&lt;point_t&gt; SOSLAB::pointcloud_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	Point Cloud 변수형



### Classes

변수형	SOSLAB::point_t
선언	<code>typedef struct _POINT_T point_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	3D Point에 대응하는 Coordinate System 구조체
Member Variables	<b>Member Variables 설명</b>
<code>float x</code>	Point의 x 좌표 정보 (단위 : mm)
<code>float y</code>	Point의 y 좌표 정보 (단위 : mm)
<code>float z</code>	Point의 z 좌표 정보 (단위 : mm)



3D Point의 Cartesian Coordinate System

### Classes

변수형	SOSLAB:: <code>ip_setting_t</code>
선언	<code>typedef struct IP_ADRESS_T ip_setting_t</code>
Header	<code>#include "soslab_ttypedef.h"</code>
설명	IP 주소와 Port 정보를 저장하는 구조체
Member Variables	Member Variables 설명
<code>std::string ip_address</code>	IP 주소
<code>int port_number</code>	Port 번호

### Classes

변수형	SOSLAB::LidarMl::scene_t
선언	typedef struct _ML_SCENE_T point_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	Multi-Echo Depth 데이터 배열 [1 <sup>st</sup> vector size : # echo, 2 <sup>nd</sup> vector size : rows x cols] (Depth : 측정된 거리 데이터)
std::vector<std::vector<uint16_t>> intensity_image	Multi-Echo Multi-Echo Intensity 데이터 배열 [1 <sup>st</sup> vector size : # echo, 2 <sup>nd</sup> vector size : rows x cols] (Intensity : 측정된 LiDAR 신호 강도)
std::vector<std::vector<pointcloud_t>> pointcloud	Multi-Echo Point cloud 데이터 배열 [1 <sup>st</sup> vector size : # echo, 2 <sup>nd</sup> vector size : rows x cols] (Point cloud : 3차원 point의 집합 데이터)

### Classes

변수형	SOSLAB::LidarMl
선언	<code>class LidarMl</code>
Header	<code>#include "ml/libsoslab_ml.h"</code>
설명	ML-X Device Connection 및 Signal/Data Processing
<b>Functions</b>	
<code>bool connect(const ip_settings_t ml, const ip_settings_t local);</code>	
<ul style="list-style-type: none"> <li>• 설명 : Local(PC)과 ML-X Device를 연결하는 함수</li> <li>• Input : Local(PC) 및 ML-X Device의 IP 주소, Port 번호</li> <li>• Output : 연결 상태에 대해 bool 변수형으로 반환 (연결되면 true 반환)</li> </ul>	
<code>bool disconnect();</code>	
<ul style="list-style-type: none"> <li>• 설명 : Local(PC)과 ML-X Device를 연결을 해제하는 함수</li> <li>• Output : 연결 해제 상태에 대해 bool 변수형으로 반환(연결 해제되면 true 반환)</li> </ul>	
<code>bool run();</code>	
<ul style="list-style-type: none"> <li>• 설명 : ML-X Device를 구동하는 함수</li> <li>• Output : 구동 유무에 대해 bool 변수형으로 반환(시작되면 true 반환)</li> </ul>	
<code>bool stop();</code>	
<ul style="list-style-type: none"> <li>• 설명 : ML-X Device를 구동을 중단하는 함수</li> <li>• Output : 구동 중단 유무에 대해 bool 변수형으로 반환(중단되면 true 반환)</li> </ul>	
<code>bool get_scene(scene_t&amp; scene);</code>	
<ul style="list-style-type: none"> <li>• 설명 : 입력 변수 scene으로 Raw Data를 획득하는 함수</li> <li>• Input : scene_t 변수형</li> <li>• Output : ML-X Device Raw Data를 저장한 scene_t 변수</li> </ul>	

### Classes

변수형	SOSLAB::LidarMl
선언	<code>class LidarMl</code>
Header	<code>#include "ml/libsoslab_ml.h"</code>
설명	ML-X Play Mode
<b>Functions</b>	
<code>void record_start(std::string filepath)</code>	
<ul style="list-style-type: none"> <li>• 설명 : ML-X 데이터 녹화 기능을 시작하는 함수</li> <li>• Input : 녹화 파일(.pcap, .bin)이 저장될 위치</li> </ul>	
<code>void record_stop()</code>	
<ul style="list-style-type: none"> <li>• 설명 : ML-X 데이터 녹화 기능을 중지하는 함수</li> </ul>	
<code>void play_start(const std::string filepath)</code>	
<ul style="list-style-type: none"> <li>• 설명 : ML-X 데이터 녹화 파일을 재생하는 함수</li> <li>• Input : 녹화 파일(.pcap, .bin)이 저장된 위치</li> </ul>	
<code>void play_stop()</code>	
<ul style="list-style-type: none"> <li>• 설명 : ML-X 데이터 재생 기능을 중지하는 함수</li> </ul>	
<code>bool get_scene(scene_t&amp; scene, uint64_t index);</code>	
<ul style="list-style-type: none"> <li>• 설명 : 녹화 파일을 불러온 뒤, 입력 변수 scene과 frame 번호를 통해 해당 frame의 Data를 획득하는 함수</li> <li>• Input (1): Raw Data를 저장할 변수 (scene_t&amp; scene)</li> <li>• Input (2): Frame (uint64_t index)</li> <li>• Output : ML-X Device Raw Data를 저장한 scene_t 변수</li> </ul>	
<code>void get_file_size(uint64_t&amp; size)</code>	
<ul style="list-style-type: none"> <li>• 설명 : 녹화 파일을 불러온 뒤, 총 Frame 수를 반환하는 함수</li> <li>• Output : 총 Frame 수</li> </ul>	

## A.3. UDP Protocol Packet Structure

### 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[0][echo1]		range[0][echo1]			
point_cloud[0][echo1] (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])							
-		intensity[191][echo1]		range[191][echo1]			
point_cloud[191][echo1] (x[20:0], y[41:21], z[62:42])							

## A.3. 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
2352~2355 2356~2357 2358~2359 2360~2367	lidar_data [0][echo 1]	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
...	...	...	...	...
8448~8451 8452~8453 8454~8455 8456~8463	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
8464~8467 8468~8469 8470~8471 8472~8479	lidar_data [191][echo 1]	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. UDP Protocol Packet Structure

### 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[0][echo1]		range[0][echo1]			
point_cloud[0][echo1] (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])							
-		intensity[575][echo1]		range[575][echo1]			
point_cloud[575][echo1] (x[20:0], y[41:21], z[62:42])							



## A.3. UDP Protocol Packet Structure

UDP Protocol Depth Completion 패키지 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
2352~2355 2356~2357 2358~2359 2360~2367	lidar_data [0][echo 1]	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
...	...	...	...	...
20736~20739 20740~20741 20742~20743 20744~20751	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
20752~20755 20756~20757 20758~20759 20760~20767	lidar_data [575][echo 1]	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 ML에서 응답하는 형태로 구성되어 있습니다. JSON형태의 구조는 아래 Table과 같습니다.

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