



# Solid-State LiDAR ML-X User Guide

Version History	Last updated	Description
Release v1.4	2023-05-16	

### **Table of Contents**

1. Hard	dware Configuration ······	03
1	1.1. Mechanical Interface ······	04
1	1.2. Electrical Interface ······	07
2. SO	S Studio ······	21
2	2.1. Installation ······	22
2	2.2. TCP/IP Setting ·····	23
2	2.3. Connection ·····	25
2	2.4. Data Load	28
2	2.5. Device Setting ·····	29
2	2.6. Point Cloud Setting ·····	34
2	2.7. Image Setting ·····	35
2	2.8. Grid Setting ·····	36
2	2.9. X-Y / Y-Z / Z-X Axis ······	37
2	2.10. Play / Record Mode ······	38
3. ML-	-X Lidar api ·····	39
3	3.1. ML-X API C++ Code Build from Source ······	40
3	3.2. ML-X ROS Example Code Build ······	45
3	3.3. Example Code for Ubuntu/ROS ······	48
Appen	ndix ·····	54
А	A.1. ML-X API – Classes ······	55
A	A.2. UDP Protocol Packet Structure ······	62
А	A.3. TCP Protocol Packet Structure ······	66

## **Chapter 1**

Hardware Configuration

### 1.1. Mechanical Interface



#### 1.1.1 Included Components

ML-X is provided with the following items:

- ML-X 80°, 80° Dedicated (Power/Ethernet/Time Sync) Cable
- ML-X 120°, 120° Dedicated (Power/Ethernet/Time Sync) Cable
- Sensor AC/DC Power Adapter/Power Cable(80°, 120° Common)



(a) ML-X 120°



(c) 120° Dedicated Cable



(e) AC/DC Power Adapter



(b) ML-X 80°



(d) 80° Dedicated Cable



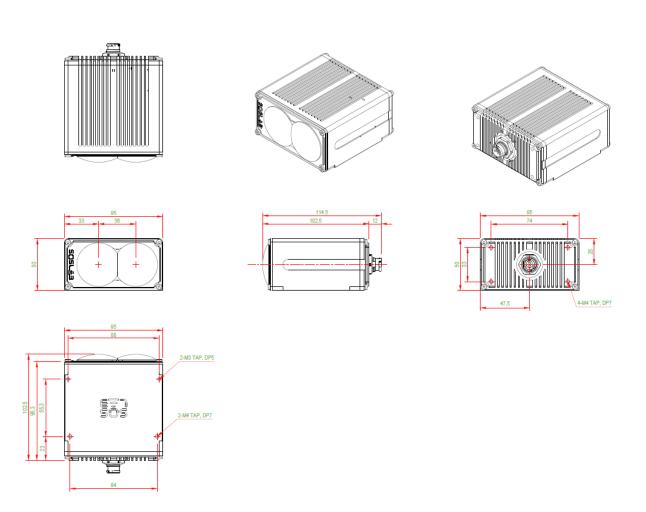
(f) AC/DC Power Cable

### 1.1. Mechanical Interface



#### 1.1.2 Exterior Mechanical Dimensions

#### ML-X 120° Dimensions



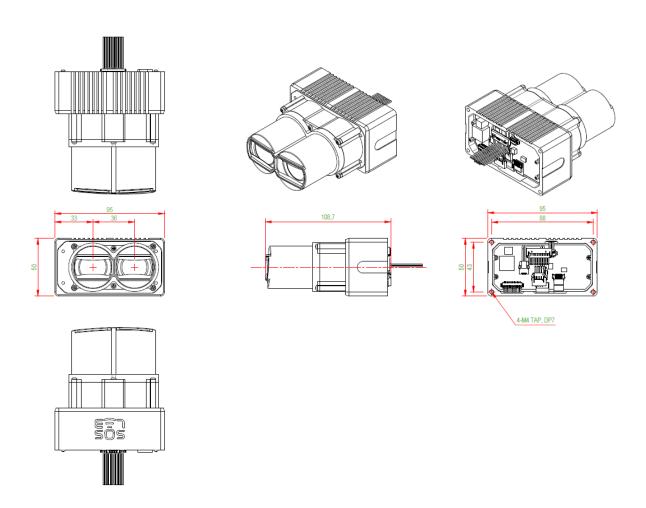
<The sensor has 4×M4 mounting holes>

### 1.1. Mechanical Interface



#### 1.1.2 Exterior Mechanical Dimensions

#### ML-X 80° Dimensions



<The sensor has 4×M4 mounting holes>



#### 1,2,1 Cable Connection

The sequence for connecting the cables is as follows:

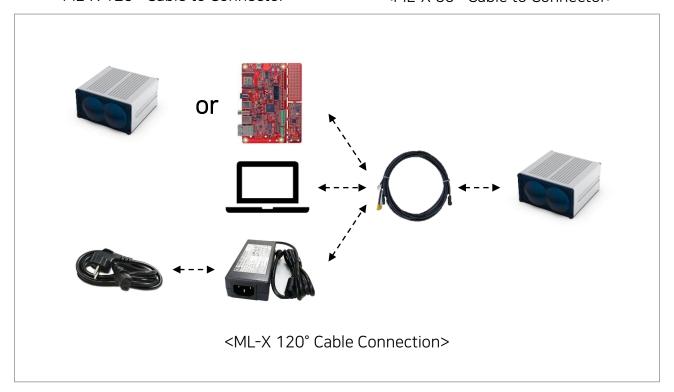
- 1. Connect the provided sensor to the integrated cable.
- 2. Connect the Ethernet cable to the PC.
- 3. If using Timing Sync., connect the provided sensor to the external device.
- 4. Connect the power cable to the power adapter.





<ML-X 120 ° Cable to Connector>

<ML-X 80 ° Cable to Connector>





#### 1.2.2 IP/Port Information

The default IP/Port information for the provided sensor is as follows:

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

There are two methods for connecting power to the sensor: using the provided power adapter or using an external power cable.

#### 1.2.3.1 Power Adapter Information

The specification for the provided power adapter is as follows:

• Rated Input: 100-240V 50/60Hz 1.7A

Rated Output: +12.0V 5.0A 60.0W

#### 1.2.3.2 External Power Cable

External power cable supplies voltage in the range of 12V~24V and consists of 4 wires.

Power Cable Wire	DC 5.5 전원 잭	
White: EX_GND	Negative pole (-)	
White+DOT: EX_GND	Negative pole (-)	
Orange: EX_VIN	Positive pole (+)	
Orange+DOT: EN_VIN		

The 4 wires can be connected to the DC 5.5 power jack as follows to be used for power.

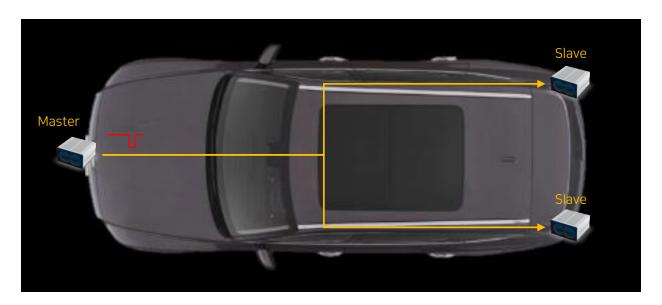




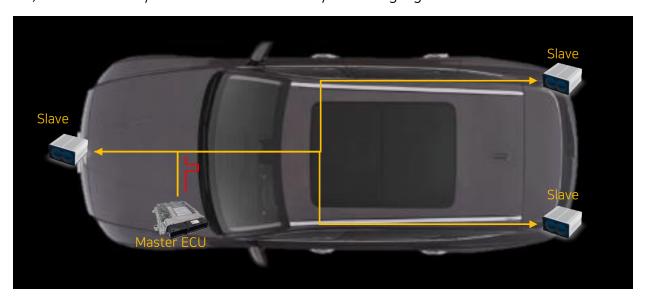
#### 1.2.4 Timing Synchronization

ML-X provides frame synchronization functionality using input/output signals, enabling the configuration of various types of applications as shown in the diagram below.

1) ML-X units can be configured as Master/Slave to implement frame synchronization



2) ML-X can be synchronized as a Slave by receiving signals from an external device





#### 1.2.4.1 External SYNC IN/OUT Cable

ML-X External Cable provides SYNC IN/OUT ports, which can be used to configure external circuits.



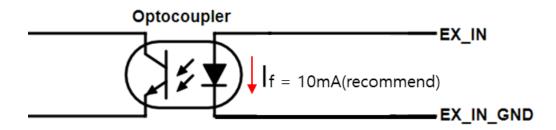
- SIG\_IN (Orange, EX\_IN)
- IN\_GND (Orange+DOT, EX\_IN\_GND),
- SIG\_OUT (White, EX\_OUT)
- OUT\_GND (White+DOT, EX\_OUT\_GND)



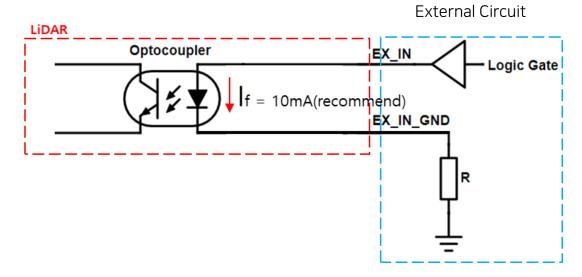
#### 1,2,4,2 External SYNC IN

The internal and external circuit configuration of ML-X's sync input are as follows.

1) The internal circuit configuration of ML-X's sync input is as follows.



2) Please refer to the following diagram for the external circuit configuration.



3) Selection of resistance values based on EX\_IN Voltage

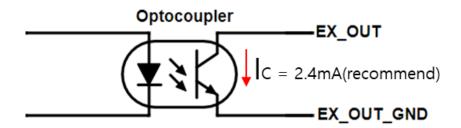
EX_IN Voltage	External Resistor(R)	Forward Current (I <sub>F</sub> )	Recommended I <sub>F</sub>
3.3 V (Min)	200	10.25mA	$7.5 \text{ mA} < I_F < 15 \text{ mA}$
5 V	360	10.41 mA	I <sub>E</sub> (mA)
12 V	1000	10.75 mA	$= \frac{V - 1.25}{R} * 1000$
24 V (Max)	2200	10.34 mA	K



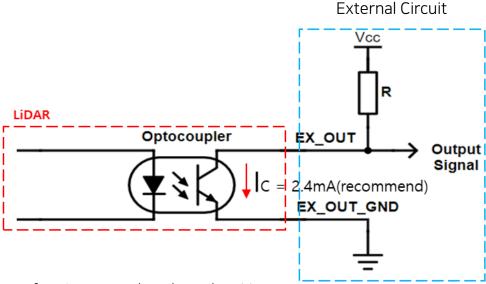
#### 1,2,4,3 External SYNC OUT

The internal and external circuit configuration of ML-X's sync output are as follows.

1) The internal circuit configuration of ML-X's sync output is as follows.



2) Please refer to the following diagram for the external circuit configuration.



3) Selection of resistance values based on  $V_{CC}$ 

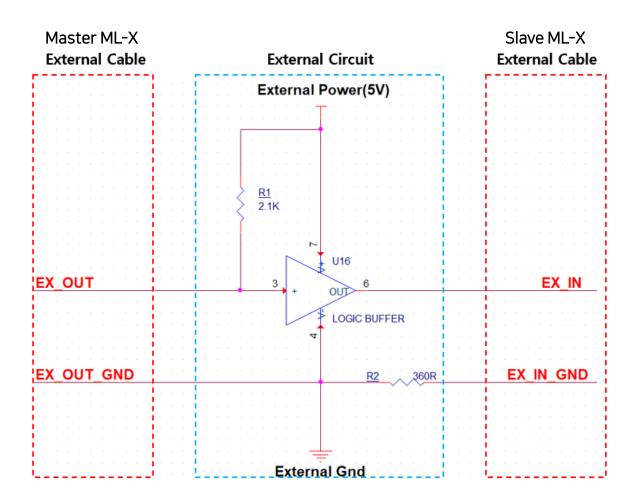
External Voltage (V <sub>CC</sub> )	External Resistor (R)	Collector Current (I <sub>C</sub> )	Recommended I <sub>C</sub>
3.3 V (Min)	1375	2.4 mA	
5 V	2100	2.4 mA	$1 \text{ mA} < I_{\text{C}} < 10 \text{ mA}$
12 V	5000	2.4 mA	$I_{c}(mA) = \frac{V_{CC}}{R} * 1000$
24 V (Max)	10000	2.4 mA	



#### 1.2.4.4 Sync IN/OUT Combine

Here is an example of connecting a single master ML-X to a single slave ML-X.

- 1) External Circuit Configuration
  - External Power: 5V





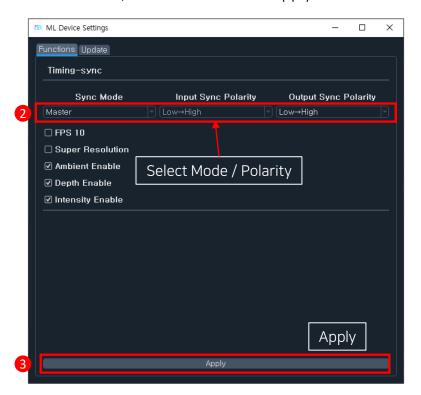
#### 1.2.4.5 SYNC FUNCTION

Users can utilize SOS Studio to easily configure the Sync Mode of ML-X as either Master or Slave, granting them full control over the synchronization functionality.

- 1) Procedure for setting the sync mode
  - After launching SOS Studio, click on "Device Setting."



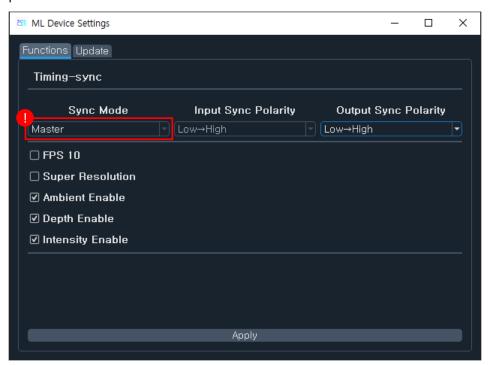
• In the "Functions" tab, user can select and apply the desired mode.



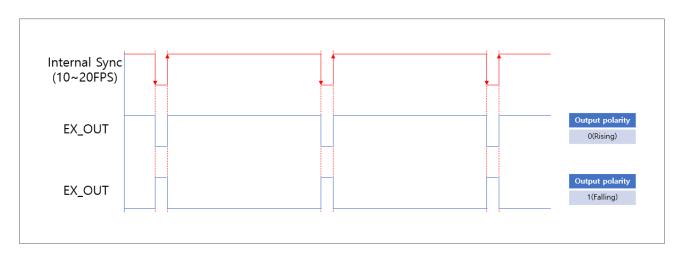


#### 2) MASTER

• When user select "Master" for the sync mode and click on "Apply," ML-X will operate in the master mode.



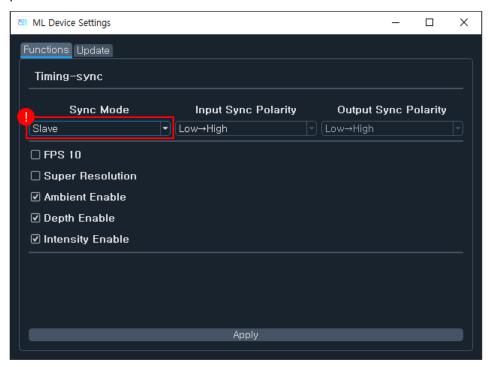
• ML-X outputs synchronization signals for every frame according to the configured frame rate. The polarity of the sync out signal can be changed, allowing for either a High-to-Low or Low-to-High transition.



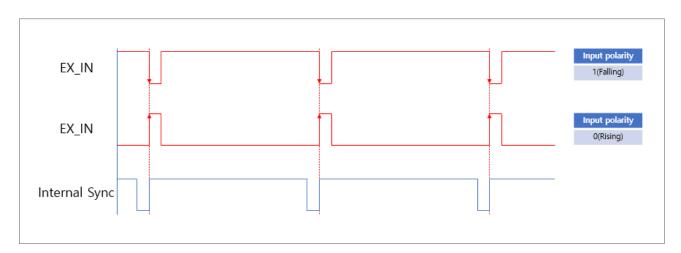


#### 3) SLAVE

• When user select "Slave" for the sync mode and click on "Apply," ML-X will operate in the slave mode.



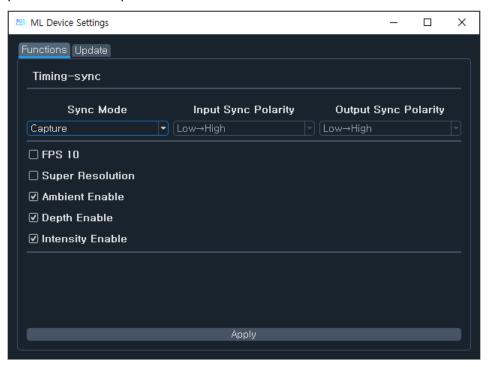
• ML-X operates on a frame-by-frame basis, synchronized with the timing of the incoming synchronization signal. The sync input polarity can be changed, supporting both High-to-Low and Low-to-High transitions.



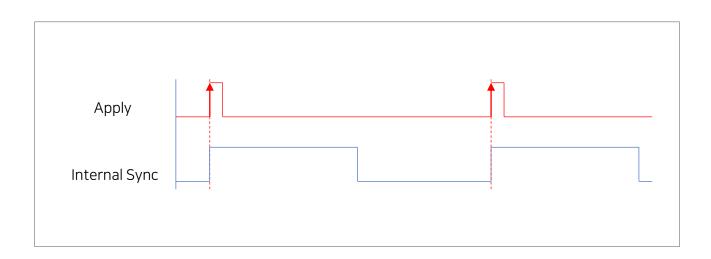


#### 4) CAPUTE

• When user select "Capture" for the sync mode and click on "Apply," ML-X will operate in the capture mode.



• Each time user click on "Apply," a single internal sync is generated, triggering a scene update.

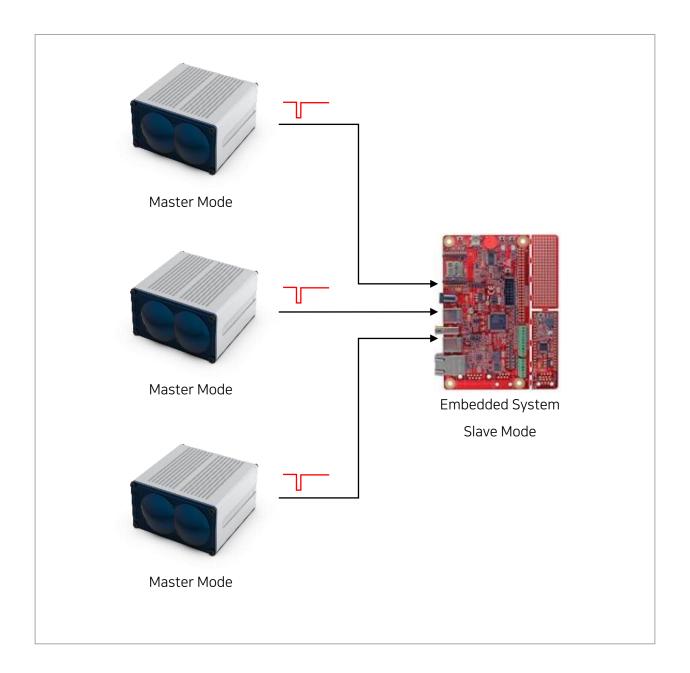




#### 1,2,4,5 APPLICATION

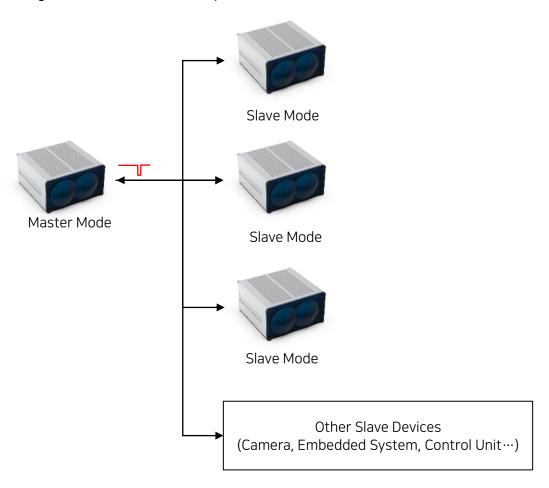
Using input/output synchronization signals, it is possible to configure a synchronized setup with multiple ML-X units or other devices.

1) Multiple Master ML-X  $\rightarrow$  Single Slave Device

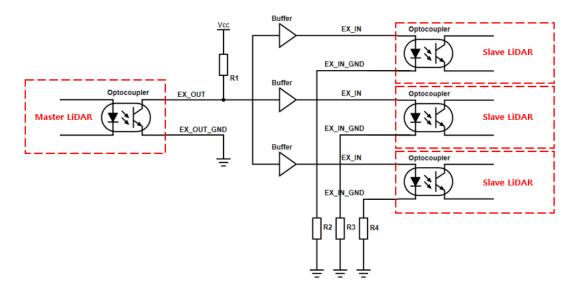




2) Single Master ML-X → Multiple Slave Device

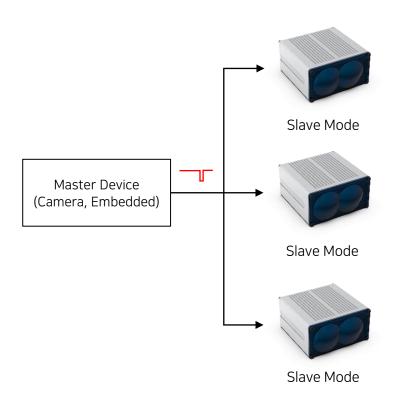


• External Circuit Configuration

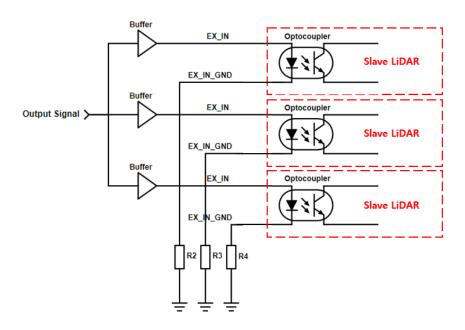




3) Single Master Device → Multiple Slave Device



• External Circuit Configuration



**Chapter 2** 

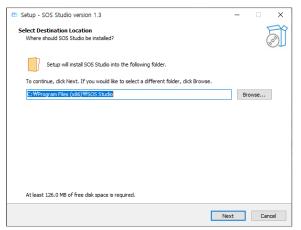
**SOS Studio** 

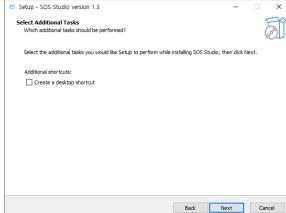
### 2.1 Installation



#### 2.1 Installation

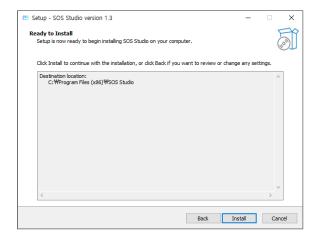
Please execute the installation file "SOS Studio\_setup.exe" to begin the installation process. The installation steps are as follows.





(a) Select Destination Location

(b) Desktop shortcut



Completing the SOS Studio Setup Wizard

Setup has finished installing SOS Studio on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

Launch SOS Studio

(c) Install SOS Studio

(d) SOS Studio Complete

Once the installation is complete, SOS Studio will be launched automatically.

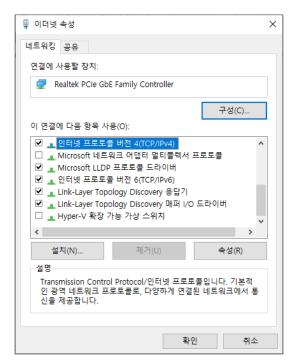
### 2.2 TCP/IP Setting

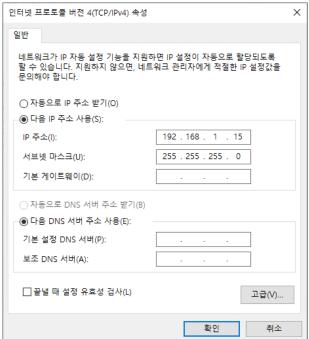


#### 2.2 TCP/IP Setting

To establish the connection between ML-X LiDAR and the PC, TCP/IP settings need to be configured.

- Connect the ML-X power cable and use a network cable to connect the PC and LiDAR.
- 2) Go to Control Panel > Network and Internet > Network and Sharing Center.
- 3) Click on the active Ethernet connection and open the Properties window.
- 4) Select Internet Protocol Version 4 (TCP/IPv4) and click on the Properties.
- 5) In the Properties window, click on the "Use the following IP address" option.
- 6) Set the IP address (192.168.1.15) and Subnet mask (255.255.255.0) as shown in the figure below.





Setting of the Ethernet and Internet Protocol Version 4 (TCP/IPv4)

### 2.2 TCP/IP Setting



Once the cable connection and IP configuration are completed, user can verify if the PC and ML-X LiDAR are successfully connected by following the Ping test procedure.

- 1) Launch the Command Prompt.
- 2) In the command prompt, enter the command 'ping 192.168.1.10'.
- 3) If the PC and ML-X are properly connected, user should see below messages.

```
™ 관리자: 영형 프롤프트

Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

C:\minnows\system32>\ping 192.168.1.10

Ping 192.168.1.101 32\thol= dlolet 사용:
192.168.1.102 응답: \thol=32 시간<\ms TTL=255
192.168.1.102| 응답: \thol=32 시간<\ms TTL=255
192.168.1.103| 응답: \thol=32 \thol=32
```

Example of the Command Prompt window and successful ping test result

### 2.3 Connection



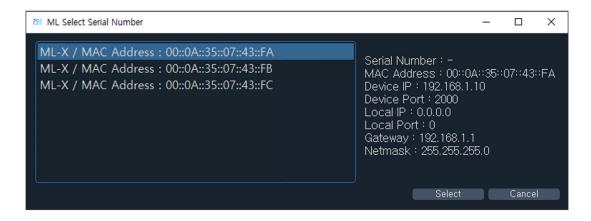
#### 2.3 Connection

This is the SOS Studio execution screen.



SOS Studio Execution Screen

To establish a connection between PC and ML-X, click on the first button on the left in SOS Studio labeled (2) "Connection." This will bring up a list of available ML-X devices that user can connect to.

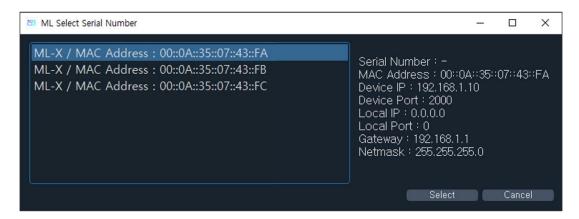


ML-X list

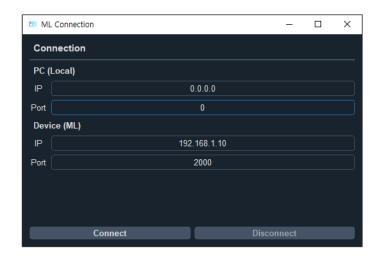
### 2.3 Connection



After selecting the desired ML-X from the ML-X List, click the "Select" button. This will automatically update the IP and Port of the selected ML-X in the Connection popup window. Click the "Connect" button in the Connection popup window to establish a connection between PC and the ML-X device.



ML-X List



Connection Window

### 2.3 Connection

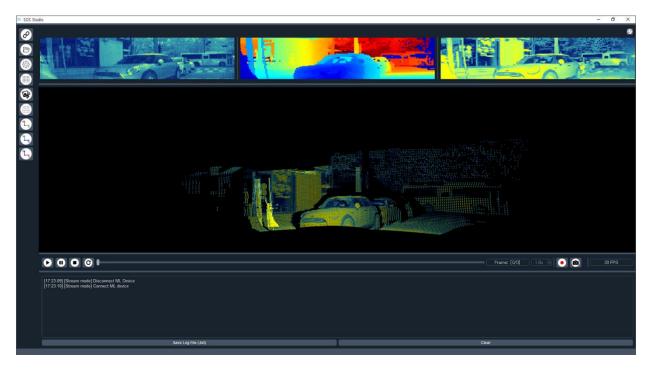


When initially connecting the PC and ML-X, user may encounter a Windows Security Alert window as follows. Check the two checkboxes as shown in the image below, then click the "Allow access (A)" button.



Windows Security Alert window

Once the connection between the PC and ML-X is established, the Image Viewer and the Point Cloud Viewer in the center become active.



SOS Studio Execution Screen

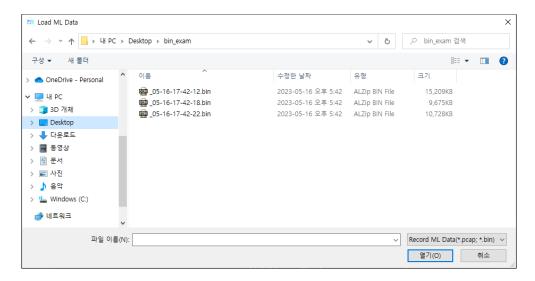
### 2.4 Data Load



#### 2.4 Data Load

Data load is used to load stored ML-X data. The functionality for storing data and playing data is described in the **2.10 Play / Record Mode**.

To load ML-X data, click on the second button on the left in SOS Studio labeled "Data Load." This will bring up a window where user can select a file, as shown below. Then, choose the stored ML-X data (.bin) file and click the "Open" button.



Data Load Window

Once the data loading is complete, user can play the ML-X data using the functionalities provided by the play bar.

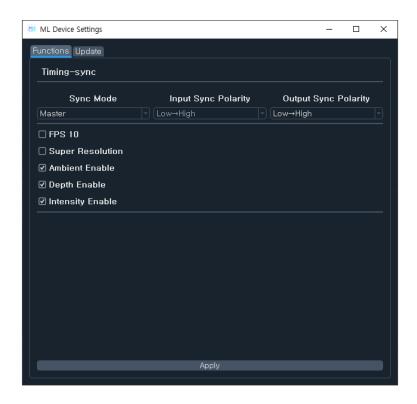
### 2.5 Device Setting



#### 2.5 Device Setting

Device setting provides functionalities for the ML-X device, including Timing Sync, Super Resolution on/off, and IP change.

Timing sync and function on/off are provided under the "Functions" tab, while the IP change feature and firmware update functionality are provided under the "Update" tab.



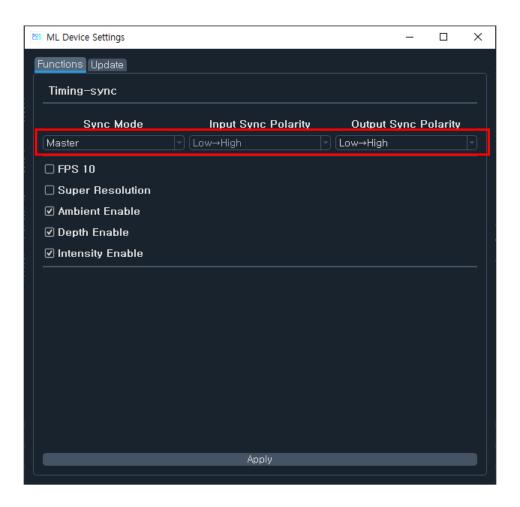
**Device Setting Window** 

### 2.5.1 Device Setting – Timing Sync



#### 2.5.1 Timing Sync

The Timing Sync for the ML-X Device is available under the "Functions" tab. Timing Sync supports three modes: Master mode, Slave mode, and Capture mode. In Master mode, user can configure the Output Sync Polarity, while in Slave mode, user can configure the Input Sync Polarity. After selecting the desired mode and polarity, click the "Apply" button to apply the settings. For more detailed information about the Timing Sync, please refer to pages 9 to 20 of the User Guide.



Device Setting Window - Timing sync

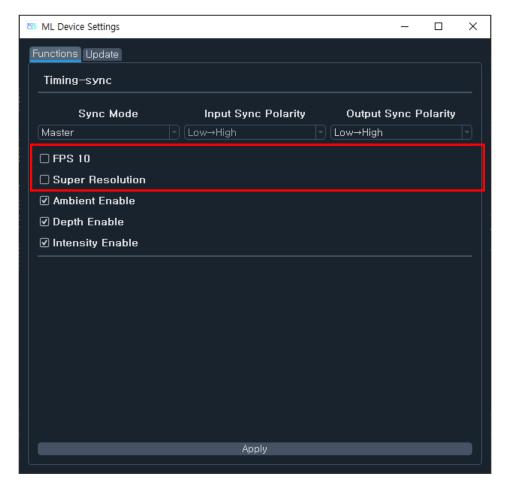
### 2.5.2 Device Setting – Functions on/off



#### 2.5.2 Functions on/off

The FPS 10 and Super Resolution for the ML-X Device is available under the "Functions" tab. Each feature has a checkbox button, and when user select the checkbox and click the "Apply" button, the feature will be turned on. Conversely, if user do not select the checkbox and click the "Apply" button, the feature will be turned off. For more detailed information about the Functions, please refer to page 60 of the User Guide.

Functions	Default Setting
FPS 10	Off
Super Resolution	Off



Device Setting Window - Functions On/Off

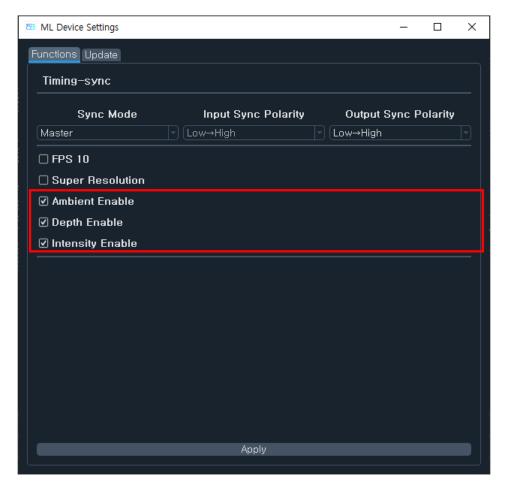
### 2.5.2 Device Setting – Functions on/off



#### 2.5.2 Functions on/off

The on/off control for the Ambient, Depth, and Intensity enable of the ML-X Device is available under the "Functions" tab. Each feature has a checkbox button, and when user select the checkbox and click the "Apply" button, the corresponding data will be transmitted. Conversely, if user do not select the checkbox and click the "Apply" button, the corresponding data will not be transmitted. For more detailed information about the Functions, please refer to page 61 of the User Guide.

Functions	Default Setting
Ambient Enable	On
Depth Enable	On
Intensity Enable	On



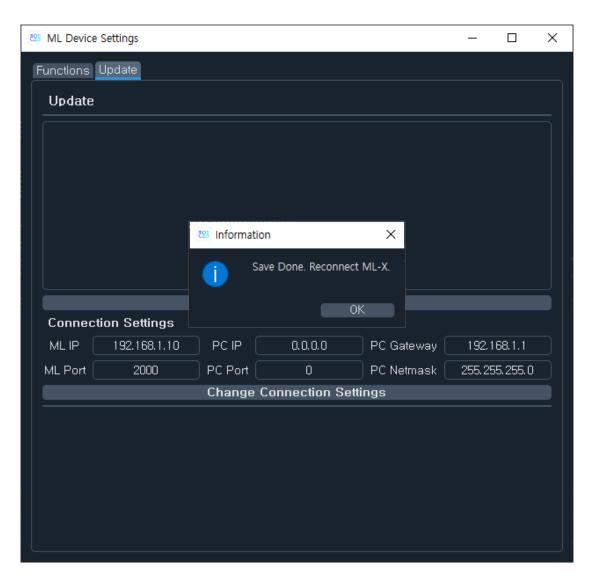
### 2.5.3 Device Setting - IP Changer



#### 2.5.3 IP Changer

The IP change feature for the ML-X Device is available under the "Update" tab. User can perform the IP change for the ML-X Device using the following steps:

- ① Enter the new IP for the ML-X Device that the user want to change.
- ② Click the "Change Connection Settings" button.
- ③ After the change, reconnect the power cable for the ML-X Device.



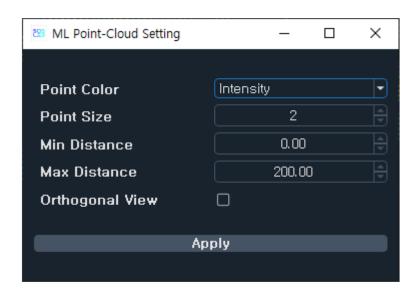
IP Changer Result

### 2.6 Point Cloud Setting



#### 2.6 Point Cloud Setting

In the Point Cloud Setting, user can configure the Point Cloud Viewer located in the center of SOS Studio. When user click on the fourth button on the left in SOS Studio labeled "Point Cloud Setting," a popup window will appear as shown below.



Point Cloud Setting Window

The following is an explanation of the available options for Point Cloud Setting.

List	Description
Point Color	Point Color (White, Red, Green, Blue, Ambient, Depth, Intensity)
Point Size	Point Size (0~5)
Min Distance	Minimum distance for visible points.
Max Distance	Maximum distance for visible points.
Orthogonal View	Orthogonal View Enable

### 2.7 Image Setting



#### 2.7 Image Setting

In the Image Setting, user can configure the Image Viewer located at the top of SOS Studio. When user click on the fifth button on the left in SOS Studio labeled "Image Setting," a popup window will appear as shown below.

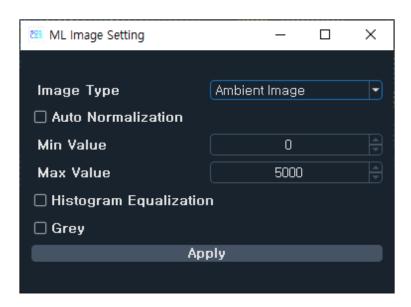


Image Setting Window

The following is an explanation of the available options for Image Setting.

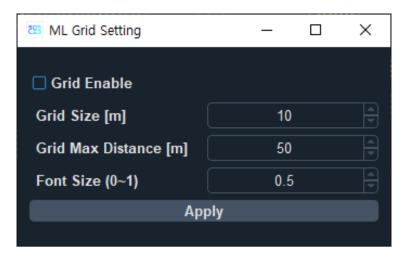
List	Description
Image Type	Select Image (Ambient / Depth / Intensity Image)
Auto Normalization	Normalize the image based on its minimum/maximum values
Min Value	Minimum value for normalization
Max Value	Maximum value for normalization
Histogram Equalization	Histogram equalization enable
Grey	Convert to grey image

### 2.8 Grid Setting



#### 2.8 Grid Setting

In the Grid Setting, user can configure the grid for the Point Cloud Viewer located at the center of SOS Studio. When user click on the sixth button on the left in SOS Studio labeled "Grid Setting," a popup window will appear as shown below.



**Grid Setting Window** 

The following is an explanation of the available options for Grid Setting.

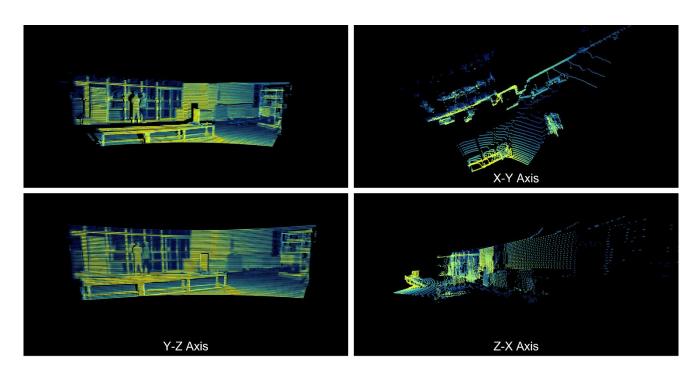
List	Description
Grid Enable	Enable grid visualization
Grid Size	Grid spacing size
Grid Max Distance	Grid maximum distance
Font Size	Grid distance display text size

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



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

X-Y / Y-Z / Z-X Axis provides a functionality to adjust the perspective of the Point Cloud Viewer located at the center of SOS Studio according to the specified axis. When user select each of these axis, the Point Cloud Viewer will be adjusted to the corresponding axis from a viewpoint.



Changing View-Point by Selecting X-Y / Y-Z / Z-X Axis Button

# 2.10 Play / Record Mode



## 2.10 Play / Record Mode

The Play Bar located below the Point Cloud Viewer provides functionalities for playing ML-X data loaded in the **2.4 Data Load** and recording the data.



SOS Studio Play Bar

The following is an explanation of the available options for Play Bar.

List	Description			
Play	Playback of data			
Pause	Pause the playback of data			
Stop	Stop the playback of data			
Repeat	Repeat the playback of data			
Scroll Bar	Display of the surrently playing frame within the everyll frame			
Frame	Display of the currently playing frame within the overall frames			
Speed	Speed of playback			
Record	Recording continuously streaming data (.bin)			
Capture	Capturing single frame streaming or playback data (.png, .pcd)			
FPS	Frame frequency of streaming data			

# **Chapter 3**

**ML-X LIDAR API** 



The ML-X SDK is an open-source code and can download it through the Github link. (Github link: <a href="https://github.com/SOSLAB-SS/MLX\_LiDAR\_SDK">https://github.com/SOSLAB-SS/MLX\_LiDAR\_SDK</a>)

The environment required for build the ML-X API is as follows.

- 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

User can generate a Visual Studio Solution file by entering the following command.

\$ git clone https://github.com/SOSLAB-SS/MLX\_LiDAR\_SDK.git

\$ cd MLX\_LiDAR\_SDK/MLX\_API/

\$ cmake CMakeLists.txt

After entering the command, a "libsoslab.sln" will be generated in the folder. Run this solution and change the build type to Release. Build the ALL\_BUILD project. Once the build is complete, the following files will be generated in the output/Release folder: "libsoslab\_core.lib", "libsoslab\_core.dll", "libsoslab\_ml.lib", "libsoslab\_ml.dll", and "test ml.exe".

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

User can build the code by entering the following command.

\$ 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

After the build is completed, the following files will be generated in the output/Release folder: "libsoslab\_core.so", "libsoslab\_ml.so", and "test\_ml".



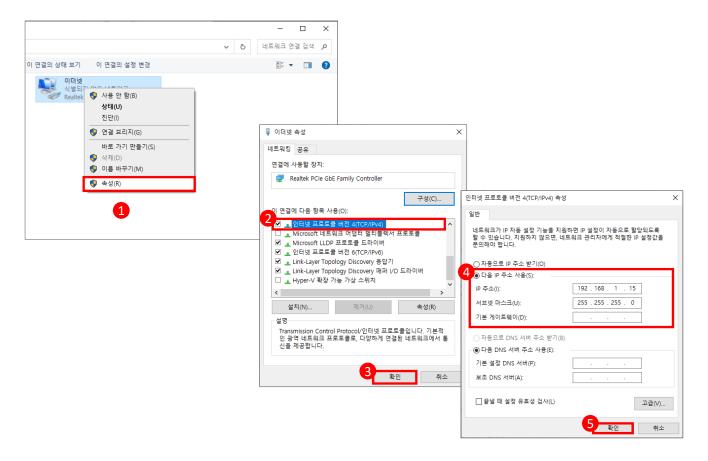
#### 3.1.2. Running the Example Code

After building the source code, user can run the "test\_ml" program to save ML-X's point cloud data as a ".csv" file. Before running the program, user need to configure the network settings of the host connected to ML-X.

### 3.1.2.1. Network Setting on Windows

Open Control Panel and go to Network and Internet. Click on Network Connections. Right-click on the Ethernet connection associated with ML-X and click on the Properties button. Click on Internet Protocol Version 4 (TCP/IPv4), then click on the Properties button. Choose "Use the following IP address" and configure the following settings:

IP Address: 192.168.1.15Subnet Mask: 255.255.255.0



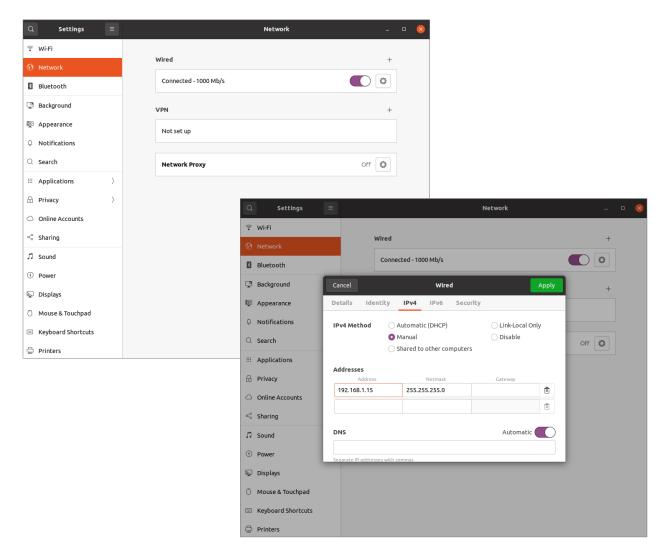
IPv4 Setting @Windows



#### 3.1.2.2. Network Setting on Linux

Open the Settings window and navigate to the Network section. Modify the IPv4 settings as follows:

IPv4 Method - ManualAddress: 192.168.1.15Netmask: 255.255.255.0

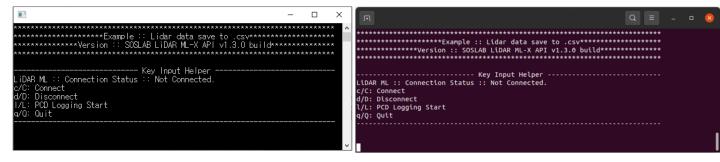


IPv4 Setting @Linux



#### 3.1.2.3. Running the Example Code

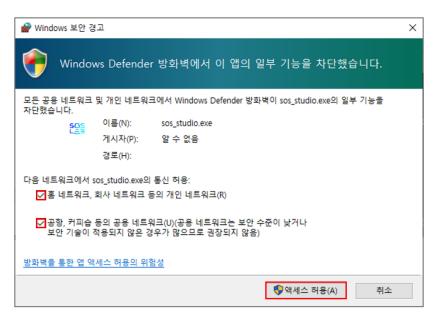
Open the terminal (Command Prompt in Windows) and execute the "test\_ml" file. When user run the file, user can see the following message displayed.



Windows Linux

- LiDAR ML :: Connection Status :: Not Connected. → ML-X Status
- c/C + enter → Connect ML-X
- d/D + enter → Disconnect ML-X
- I/L + enter → Save the PCD data(.csv)
- q/Q + enter → Quit the Program

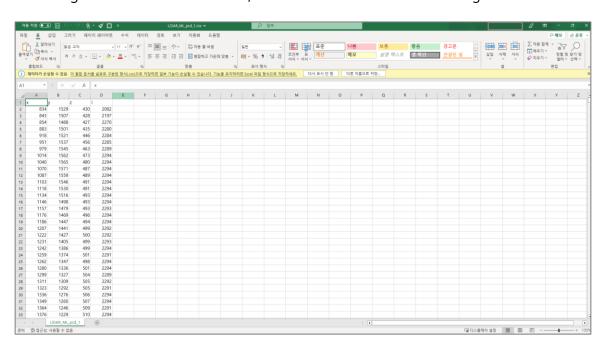
When initially connecting the PC and ML-X, user may encounter a Windows Security Alert window as follows. Check the two checkboxes as shown in the image below, then click the "Allow access (A)" button.



Windows Security Alert window



When saving PCD data as a '.csv' file, it will be stored in the following format:



The first column contains the x values, the second column contains the y values, the third column contains the z values, and the fourth column contains the Intensity data.

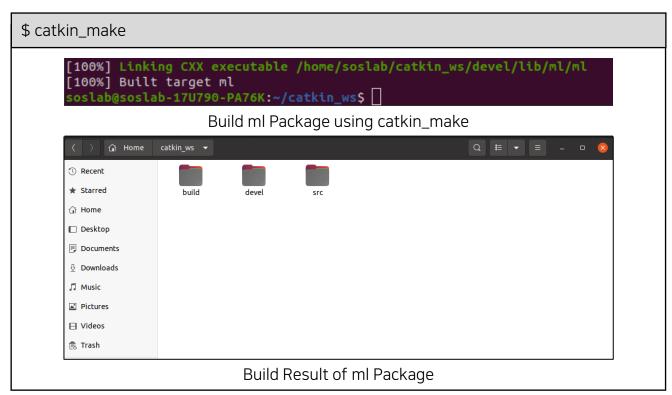
# 3.2. ML-X ROS Example Code Build



The ML-X ROS Example Code is compatible with Ubuntu 18.04/20.04 and ROS Melodic / Noetic. It provides code that allows user to run ML-X in a ROS environment.

#### 3.2. ML-X ROS Example Code Build

1) Enter the following command in the catkin\_ws folder to create the 'ml' package.



2) To add the ml package to ROS environment after building it, enter the following command.

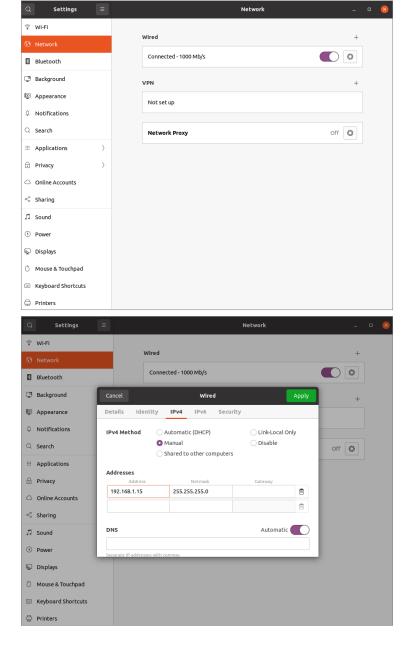


# 3.2. ML-X ROS Example Code Build



3) To establish the connection between the ML-X device and PC, follow these steps to modify the IPv4 settings in the Network section of the Settings window:

IPv4 Method - ManualAddress: 192.168.1.15Netmask: 255.255.255.0



IPv4 Setting @Linux

# 3.2. ML-X ROS Example Code Build



- 4) After configuring the network settings, execute the following command to run the ml.launch file.
- 5) If the connection with the ML-X device is successful, user can see the message "Lidar ML:: Streaming started!", as shown in the provided image.

\$ cd ~/catkin\_ws/src/ml/launch \$ roslaunch ml.launch

```
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
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/
SHMMARV
_____
 * /ml/ip_address_device: 192.168.1.10
 * /ml/tp_address_pc: 0.0.0.0
* /ml/ip_port_device: 2000
 * /ml/ip_port_pc: 0
* /rosdistro: 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 0adid044-1957-11ed-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
> lp_address_pc: 0.0.0.0
 iDAR ML :: Streaming started!
```

Connecting and Visualizing ML-X through ROS in Ubuntu



#### 3.3. Example Code for Ubuntu/ROS

The example code includes ML-X connection and data visualization in Rviz, and the code explanation is as follows.

### 1) Acquiring raw data

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

### Code of Acquiring Raw Data

Here is the code for acquiring raw data from the ML-X Device. The raw data from the ML-X Device is obtained as scene\_t, which consists of four 1-dimensional data arrays (ambient, intensity, depth, point cloud). Detailed information about the scene\_t structure is provided in the Appendix.

Line	Description
2	Declare a variable named 'scene' of type SOSLAB::LidarMI::scene_t to store the raw data
3	Obtain the raw data and store it in the 'scene' variable using the 'get_scene' function of the SOSLAB::LidarMl class, which takes a variable of type scene_t as input
4-7	Obtain the ambient, intensity, depth, and point cloud data from the scene_t structure named 'scene'. The variable names and types for each data are provided in the Appendix.
8-10	Obtain the height and width values for the Depth and Intensity images from the scene_t structure named 'scene'. The width2 variable represents the width value for the Ambient image



### 2) Rviz - Publish Ambient, Depth, Intensity Image

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

Code for publish Ambient, Depth, and Intensity Images in Rviz

Here is the code for converting and publishing the Ambient, Depth, and Intensity data obtained from the ML-X Device as images for visualization in Rviz. The code provided above is for the conversion and creation of Ambient image publishing.

Line	Description
1-4	To create a Publisher Node for sending messages, I create objects such as ImageTransport and ImagePtr from ROS. The Publisher provides ambient data on the topic 'ambient'
5-6	To visualize the Ambient data from the scene_t structure as an image using OpenCV, here is the process of converting it to the cv::Mat format. Initialize a cv::Mat object with the input values of Height (scene.rows), Width (scene.cols), Ambient data type (CV_32SC1), and the Ambient data values.
7	To visualize the image, the values are converted from the range of the maximum value (2000) and the minimum value (0) to the 8-bit (uchar) format ranging from 0 to 255.
8	To store the OpenCV's cv::Mat object as a message in the Publisher Node, here is the process of converting it to ImagePtr.
9	The publish function of the Publisher object with the topic specified as 'ambient_color' is called, using the ImagePtr variable as the input argument.

The publishing process for each data is the same as the Ambient publishing process, and the image conversion types are as follows:

Ambient: CV\_32SC1Depth: CV\_32SC1Intensity: CV\_16UC1



#### 3) Rviz - Publish Point Cloud

```
    typedef pcl::PointCLoud<pcl::PointXYZRGB> PointCloud_T

static const char* DEFAULT_FRAME_ID = "map"
ros::NodeHandle nh("~");
4. ros::Publisher pub_lidar = nh.advertise<PointCloud_T>("pointcloud",10);
PointCloud_T::Ptr msg_pointcloud(new PointCloud_T);
6. msq_pointcloud->header.frame_id = DEFAULT_FRAME_ID
7. msq_pointcloud->width = width;
msq_pointcloud->height = height;
msq_pointcloud->points.resize(scene.pointcloud.size())
10.for (int i = 0; i < scene.pointcloud.size(); i++) {
     msq_pointcloud \rightarrow points[i].x = pointcloud[i].x/1000.0;
11.
     msq_pointcloud->points[i].y = pointcloud[i].y/1000.0;
12.
     msq_pointcloud->points[i].z = pointcloud[i].z/1000.0;
13.
14.}
15.pcl_conversions::toPCL(ros::Time::now(), msg_pointcloud->header.stamp);
16.pub_lidar.publish(msg_pointcloud);
```

### Code for publish Point Cloud in Rviz

Here is the code for publishing the point cloud data obtained from the ML-X Device for visualizing the point cloud in Rviz in a ROS environment.

Line	Description				
3-4	To create a Publisher Node for sending messages, I create a ros::Publisher object from ROS. The Publisher provides point cloud on the topic 'pointcloud'.				
5-9	Define a message variable and initialize its data size and name.				
10-14	copy the point cloud data from the scene_t structure to the msg_pointcloud variable and change the distance unit from millimeters (mm) to meters (m).				
15-16	After saving the timestamp of the Point Cloud, the publish function of the Publisher object with the topic specified as 'pointcloud' is called, using the PointCloud_T::Ptr variable as the input argument to complete the publish				

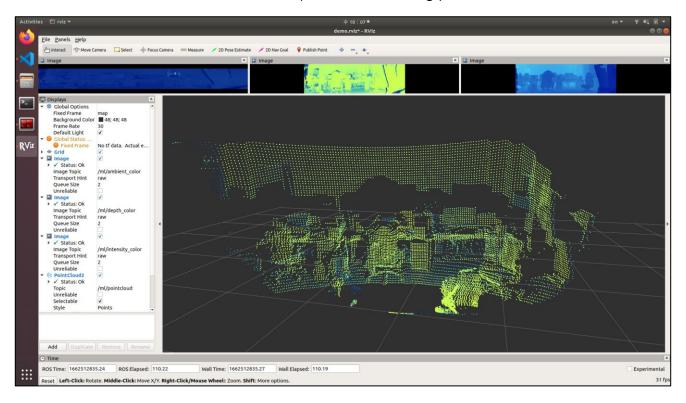


#### 4) Rviz - Visualization

To visualize it in Rviz, execute the example code and connect to the ML-X Device by entering the following command.

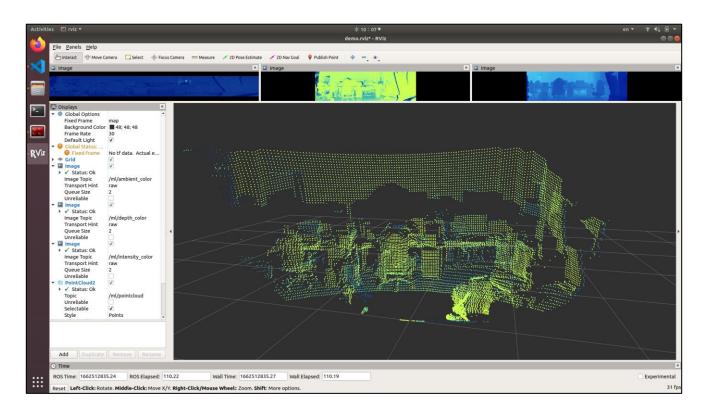
\$ cd ~/catkin\_ws \$ catkin\_make \$ source ~/catkin\_ws/devel/setup.sh \$ cd ~/catkin\_ws/src/ml/launch \$ roslaunch ml ml.launch

If user click the Add button in the bottom left corner, as shown in the image below, user will see a window similar to the image below. By clicking the 'By topic' menu at the top of that window, user can see all the topics that are being published





To visualize the Ambient, Depth, and Intensity images, select 'Image' in the Topic (/ambient, /depth, /intensity) menu and click the OK button. This will allow user to visualize the images for each topic. For Point Cloud data, select 'PointCloud2' in the Topic (/pointcloud) menu and click the OK button to visualize it.



Rviz base ML-X Data (Ambient/Depth/Intensity Images, Point Cloud) Visualization



#### 5) Rviz - Multi-LiDAR Visualization

To enable multi-LiDAR visualization in Rviz, user need to configure the IP address of each ML-X device differently. In the multi-LiDAR example, the IP settings for ML-X are as follows:

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

After changing the IP, execute the example code by entering the following command.

#### [Terminal #1]

\$ cd ~/catkin\_ws

\$ catkin make

\$ source ~/catkin\_ws/devel/setup.sh

\$ cd ~/catkin\_ws/src/ml/launch

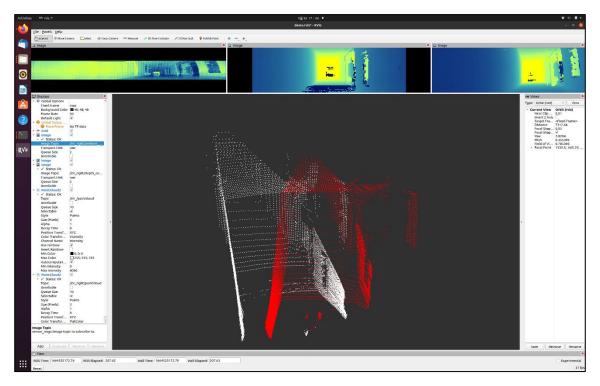
\$ roslaunch ml ml.launch

#### [Terminal #2]

\$ cd ~/catkin ws/src/ml/launch

\$ roslaunch ml ml\_second.launch

If user click the Add button in the bottom left corner, as shown in the image below, can see a window like the image below. By clicking the 'By topic' menu at the top of that window, can see the Multi-LiDAR data being transmitted through the 'ml' and 'ml\_second' topics.

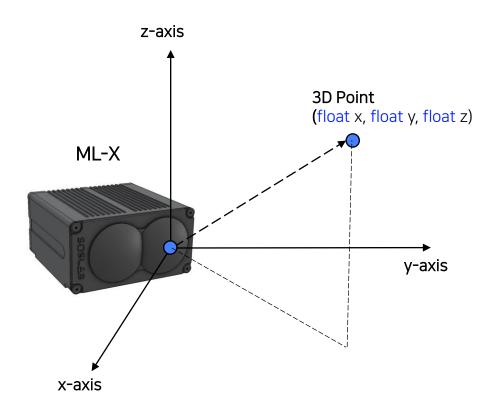


# **Appendix**



# Classes

Variable	SOSLAB::point_t		
Header	#include "soslab_typedef.h"		
Description	Structure corresponding to 3D point in coordinate		
Member Variables	Description of Member Variables		
float x	The x-coordinate of a point (unit: mm)		
float y	The y-coordinate of a point (unit: mm)		
float z	The z-coordinate of a point (unit: mm)		



Cartesian Coordinate System of 3D Point Cloud



# Classes

Variable	SOSLAB::ip_setting_t		
Header	#include "soslab_typedef.h"		
Description	Structure to store IP address and port		
Member Variables	Description of Member Variables		
std::string ip_address	IP Address		
int port_number	Port Number		



# Classes

Variable	SOSLAB::LidarMI::scene_t			
Header	#include "ml/libsoslab_ml.h"			
Description	Structure of Raw Data			
Member Variables	Description of Member Variables			
std::vector <uint32_t> timestamp</uint32_t>	Time stamp of Raw Data [size : rows]			
uint8_t frame_id	Frame Index [Maximum : 255]			
uint16_t rows	Height of Raw Data			
uint16_t cols	Width of Raw Data			
std::vector <uint32_t> ambient_image</uint32_t>	Ambient Data			
std::vector <std::vector<uint32_ t&gt;&gt; depth_image</std::vector<uint32_ 	Depth Data [1 <sup>st</sup> vector size : # echo, 2 <sup>nd</sup> vector size : rows x cols]			
std::vector <std::vector<uint16_ t&gt;&gt; intensity_image</std::vector<uint16_ 	Intensity Data [1 <sup>st</sup> vector size : # echo, 2 <sup>nd</sup> vector size : rows x cols]			
std::vector <std::vector<point_t>&gt; pointcloud</std::vector<point_t>	Point cloud Data [1 <sup>st</sup> vector size : # echo, 2 <sup>nd</sup> vector size : rows x cols]			



# Classes

Variable	SOSLAB::LidarMl			
Header	#include "ml/libsoslab_ml.h"			
Description	ML-X Device Connection & Signal/Data Processing			

#### **Functions**

bool connect(const ip\_settings\_t ml, const ip\_settings\_t local);

- Description: Connect between the local PC and the ML-X Device
- Input: IP address and port number for the local PC and ML-X Device
- Output: Connection status as a boolean variable (returning true if connected)

#### bool disconnect();

- Description: Disconnect between the local PC and the ML-X Device
- Output: Disconnection status as a boolean variable (returning true if disconnected)

#### bool run();

- Description: Run the ML-X Device
- Output: Run status as a boolean variable (returning true if it is running)

### bool stop();

- Description : Stop the ML-X Device
- Output: Stop status as a boolean variable (returning true if it is stopping)

### bool get\_scene(scene\_t& scene);

- Description: Acquire raw data using the input variable 'scene'
- Input: scene\_t& scene
- Output: Acquire the status as a boolean variable (returning true if is acquired)



## Classes

Variable	SOSLAB::LidarMl		
Header	#include "ml/libsoslab_ml.h"		
Description	ML-X Play Mode		

#### **Functions**

### void record\_start(std::string filepath)

- Description: Start the recording ML-X data
- Input: Save location of the recording file (.bin)

### void record\_stop()

• Description: Stop the recording ML-X data

### void play\_start(const std::string filepath)

- Description: Play the recording ML-X data
- Input: Load location of the recording file (.bin)

### void play\_stop()

• Description : Stop the playing ML-X data

### bool get\_scene(scene\_t& scene, uint64\_t index);

- Description: Retrieve the data of a specific frame from the recording file using the input variables 'scene' and frame number
- Input (1): Variable to store the raw data (scene\_t& scene)
- Input (2): Frame (uint64\_t index)
- Output: scene\_t variable that stores the raw data from the ML-X Device

### void get\_file\_size(uint64\_t& size)

- Description: Load the recording file and return the total number of frames
- Output: Total Frame



# Classes

Variable	SOSLAB::LidarMl		
Header	#include "ml/libsoslab_ml.h"		
Description	ML-X Functions		

### **Functions**

### bool fps10(bool en)

- Description: Control the On/Off state of the ML-X FPS 10Hz
- Input: 10 FPS On (true) / Off (false)

### bool depth\_completion(bool en)

- Description: Control the On/Off state of Super Resolution
- Input: Super Resolution On (true) / Off (false)



# Classes

Variable	SOSLAB::LidarMl		
Header	#include "ml/libsoslab_ml.h"		
Description	ML-X Functions		

### **Functions**

### bool ambient\_enable(bool en)

- Description: Control the On/Off state of the transmission of Ambient data
- Input: Ambient Enable (true) / Disable (false)

### bool depth\_enable(bool en)

- Description: Control the On/Off state of the transmission of Depth data
- Input: Depth Enable (true) / Disable (false)

### bool intensity\_enable(bool en)

- Description: Control the On/Off state of the transmission of Intensity data
- Input: Intensity Enable (true) / Disable (false)



# A.3.1 Normal Packet Structure (192 pixels)

The basic packet structure of UDP communication is as follows.

Byte							
7	6	5	4	3	2	1	0
	header						
			times	stamp			
			sta	tus			
		-			row_num	frame_id	type
ambient[1]				ambient[0]			
	ambier	nt[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])							



UDP Protocol Basic Packet Structure (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	
2336~233 9 2340~234 1 2342~234 3 2344~235	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	
5392~539 5 5396~539 7 5398~539 9 5400~540 7	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)

The packet structure for Depth completion in UDP communication is as follows.

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])							



UDP Protocol Depth Completion Packet Structure (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
11536~1153 9 11540~1154 1 11542~1154 3 11544~1155	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 communication is structured in JSON format. It is composed of commands sent from the PC and responses received from the ML-X. The structure in JSON format is as shown in the table below

TCP Protocol Packet Structure		
JSON Format	Description	
{ "command": "run" }	Device Run	
{ "command": "stop" }	Device Stop	



# Solid-State LiDAR ML-X

User Guide

Thank you

