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

User Guide

Release v1.2.2

2022-10-12

Table of Contents

Chapter 1

Hardware Configuration

1.1. Mechanical Interface

1.1.1 Included Components

Following items are provided for ML-X

- Consolidated Capble for ML-X 80°, 80° (Power/Ethernet/Time Sync)
- Consolidated Capble for ML-X 120°, (Power/Ethernet/Time Sync)
- Sensor AC/DC Power adapter/Power cable(80°, 120° in Common)

(a) ML-X 120°

(b) ML-X 80°

(c) 120° Consolidated Cable

(d) 80° Consolidated Cable

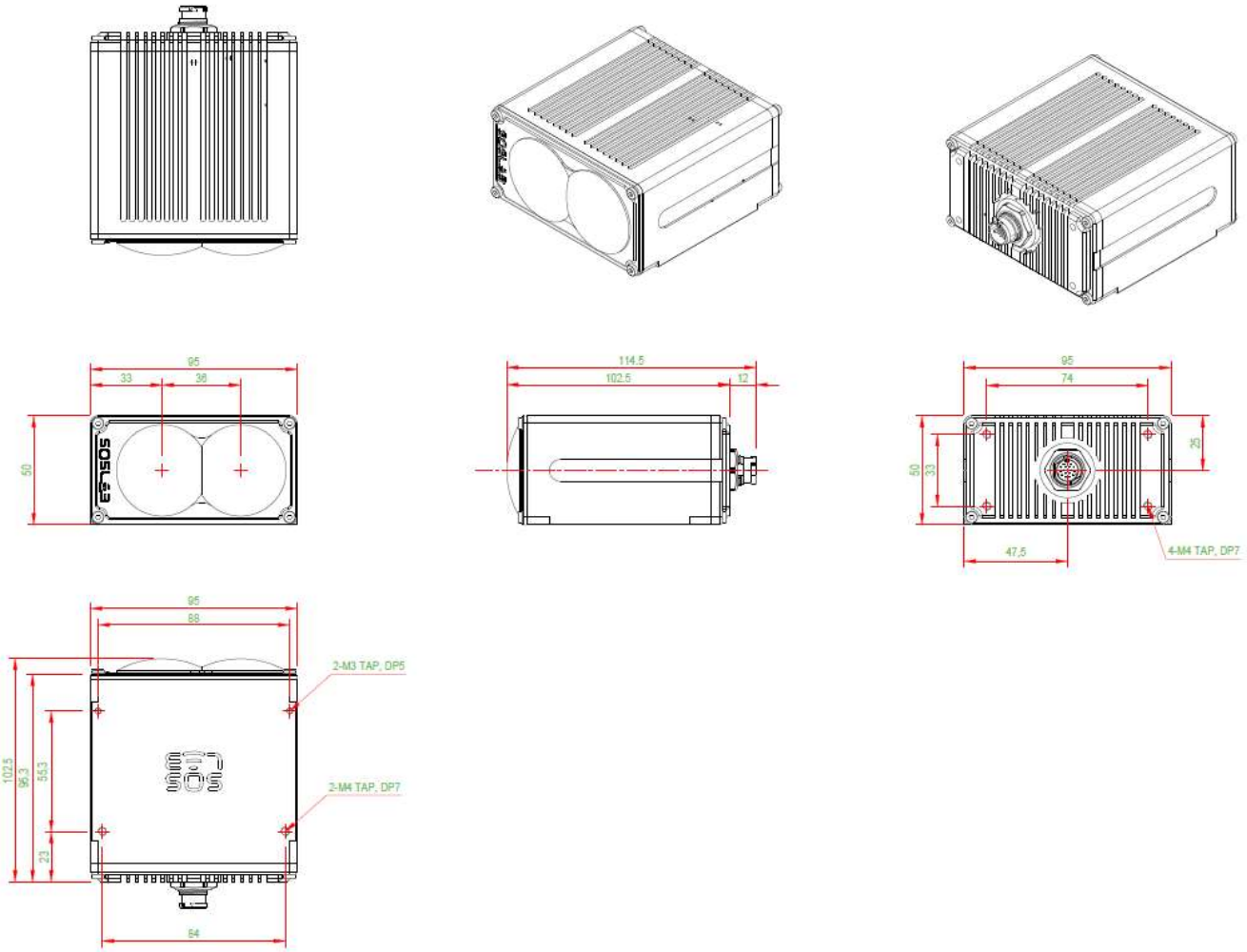
(e) AC/DC Power Adapter

(f) AC/DC Power Cable

1.1. Mechanical Interface

1.1.2 Exterior Mechanical Dimensions

ML-X 120° Dimensions

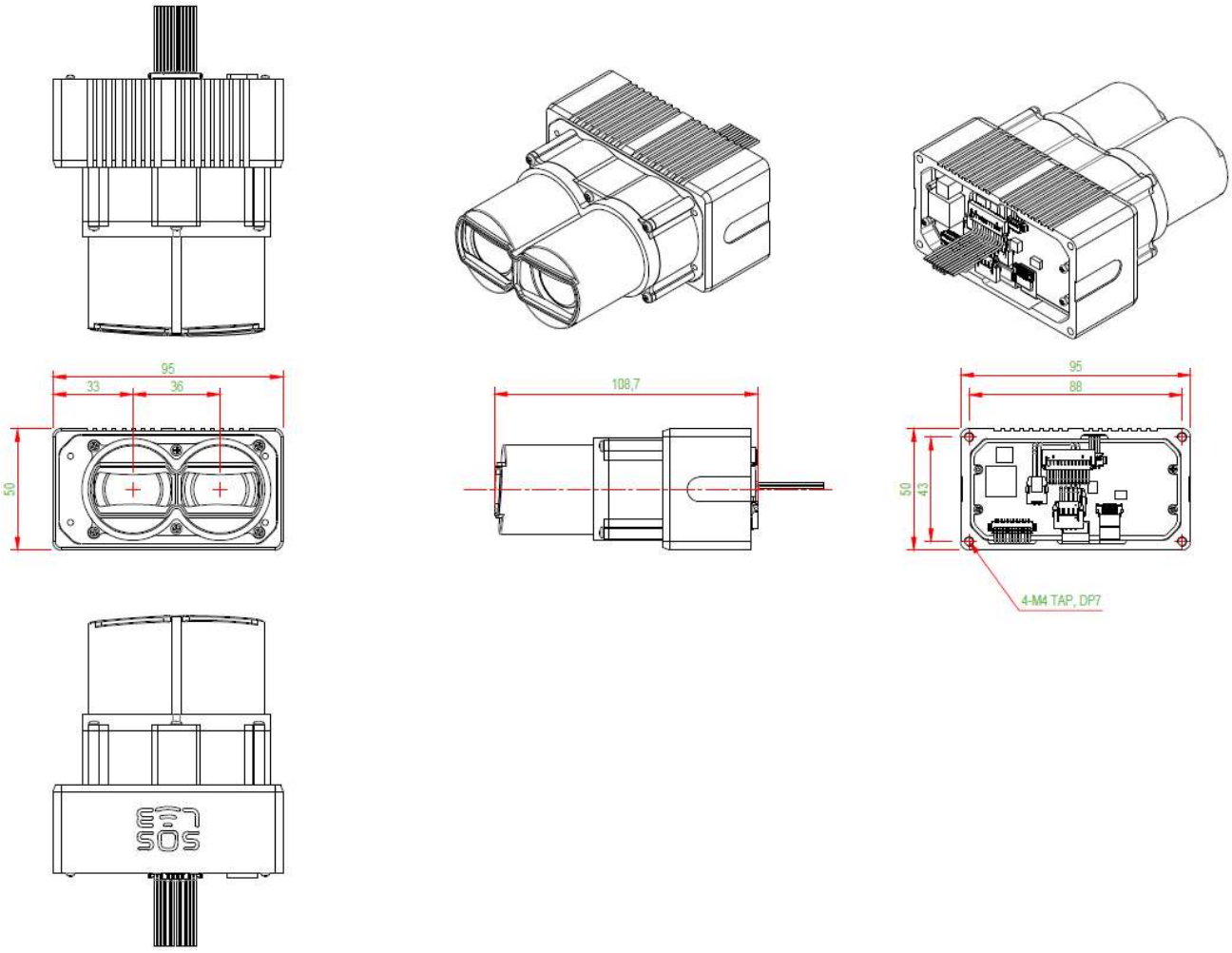


<The sensor has 4×M4 mounting holes>

1.1. Mechanical Interface

1.1.2 Exterior Mechanical Dimensions

ML-X 80° Dimensions



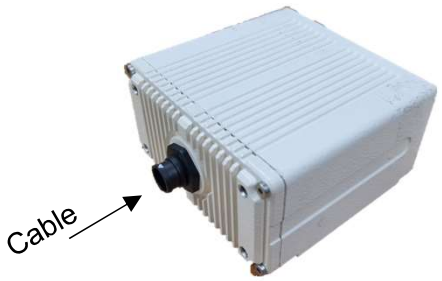
<The sensor has 4×M4 mounting holes>

1.2. Electrical Interface

1.2.1 Cable Connection

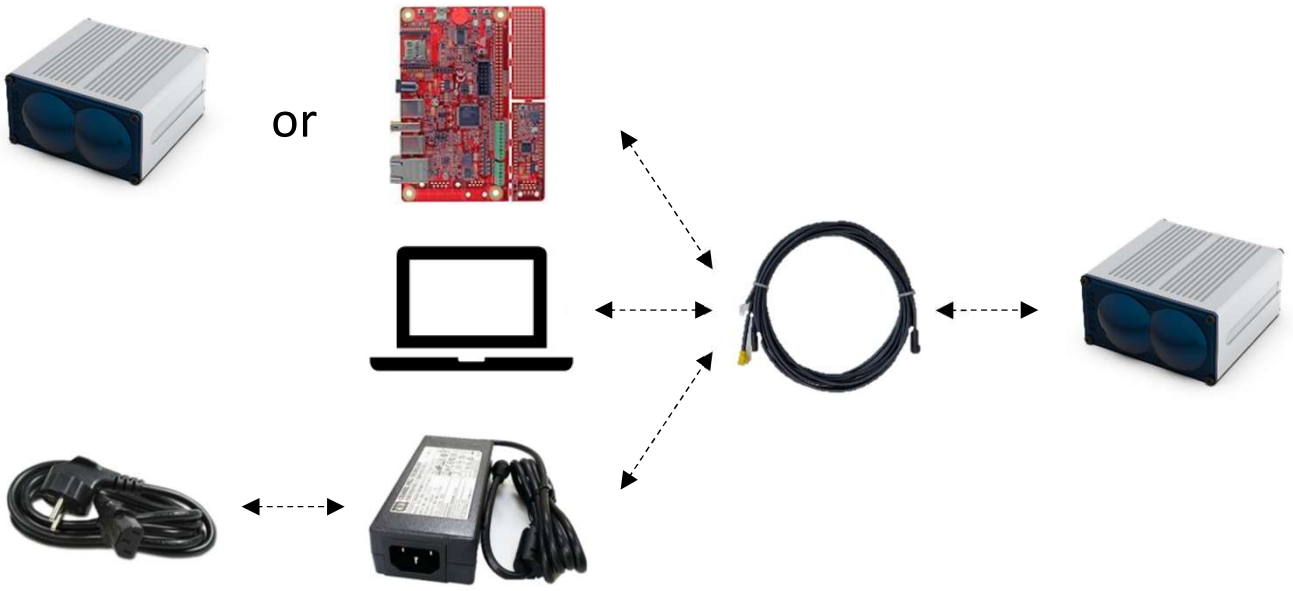
Following is the procedure to connect a cable

1. Connect a consolidated cable to provided sensor
2. Connect the ethernet cable to PC.
3. Connect a external device to provided sensor
   when using Timing Synchronization
4. Connect Power adapter to Power cable



<ML-X 120 ° Cable to Connector>

<ML-X 80 ° Cable to Connector>



<ML-X 120° Cable Connection>

1.2.2 IP/Port Information

IP/Port information for provided sensor is as follows. You can change IP/Port information through device setup.

- 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

Information for Power adaptor provided is as follows. SOSLAB does not guarantee the product reliability in issues caused by invalid rated power use that does not meet below input and outpout standard.

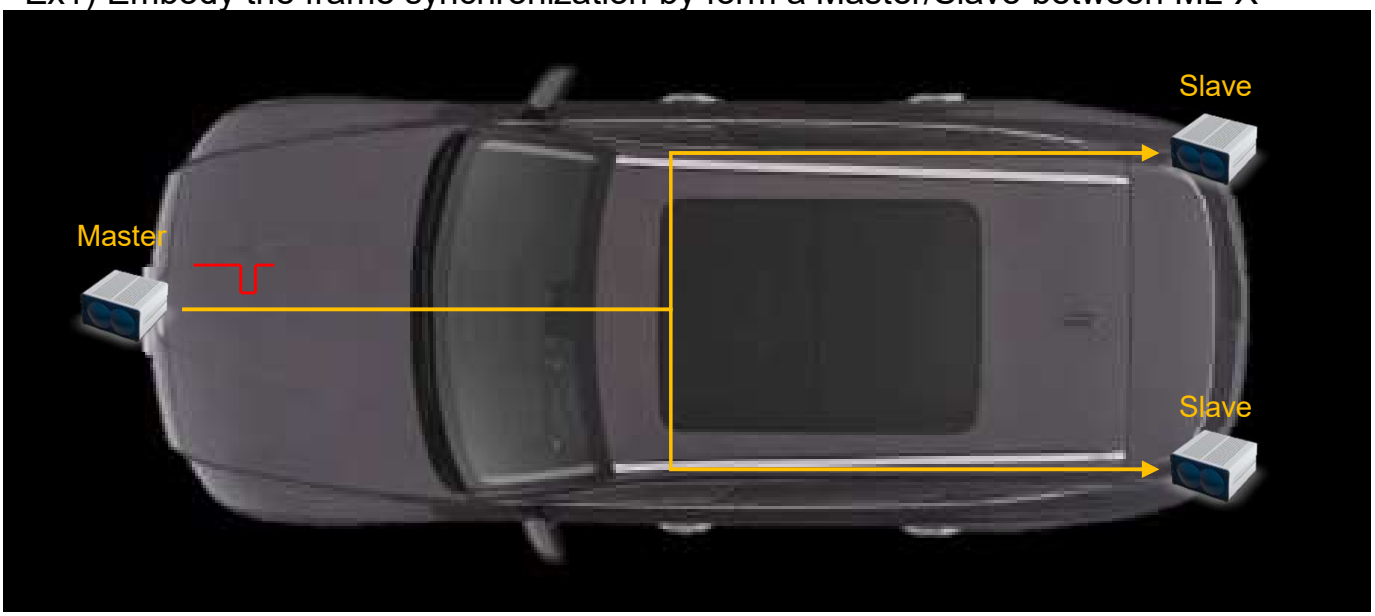
- Rated input : 100-240V 50/60Hz 1.7A
- Rated output : +12.0V 5.0A 60.0W

1.2.4 Timing Synchronization

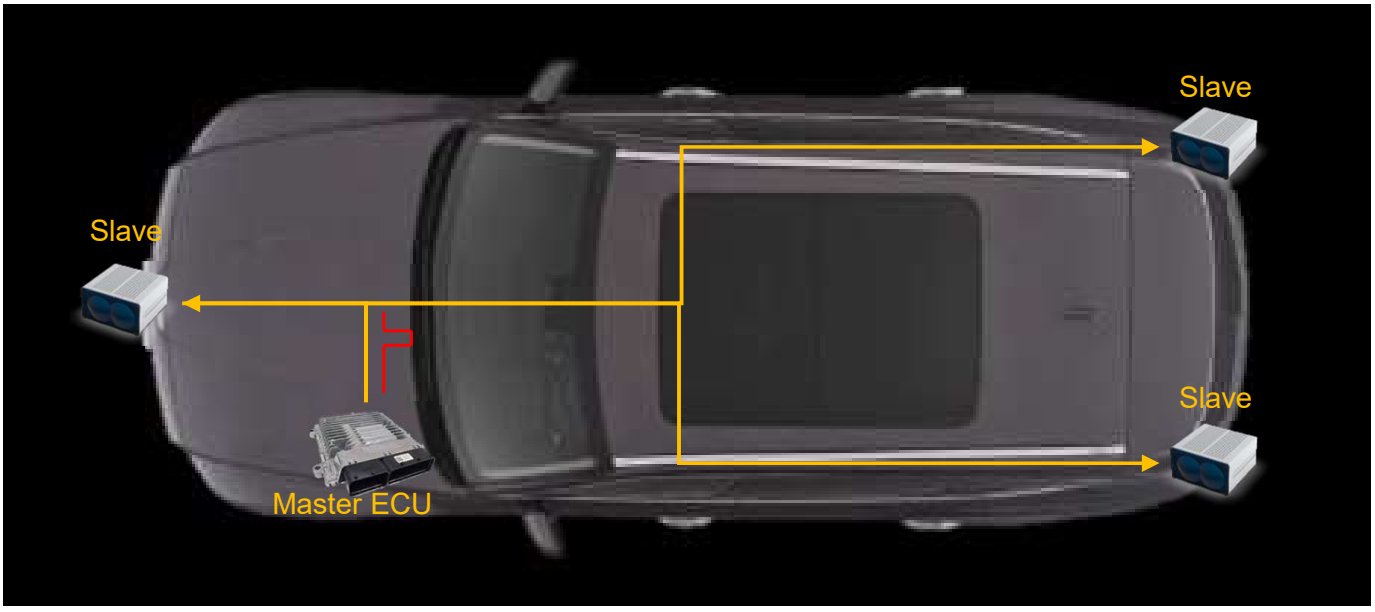
ML-X is able to form a variety of applications and provide user with frame synchronization with using in/ou-tput signal

.

Ex1) Embody the frame synchronization by form a Master/Slave between ML-X



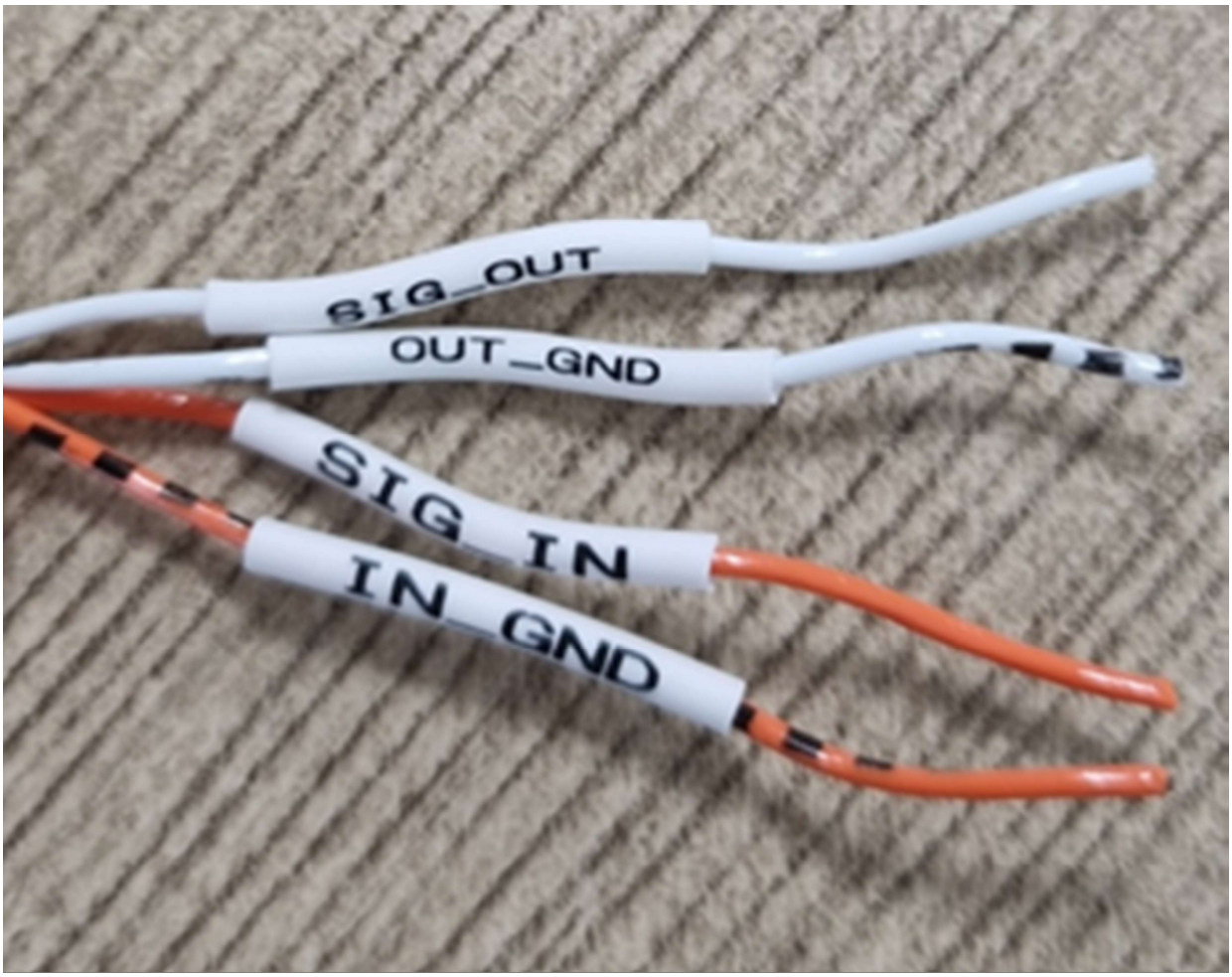
Ex2) ML-X can be synchronized as Slave by receiving the external device signal lsuch as ECU



1.2.4.1 External SYNC IN/OUT Cable

ML-X External Cable provides SYNC IN/OUT port and forms a external circuit by using its port.

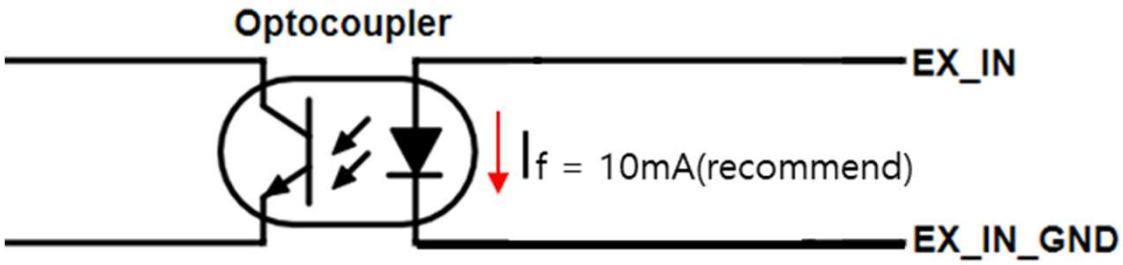
- SIG_IN(Orange, EX_IN)
- IN_GND(Orange+DOT, EX_IN_GND),
- SIG_OUT(White, EX_OUT)
- OUT_GND(White+DOT, EX_OUT_GND)



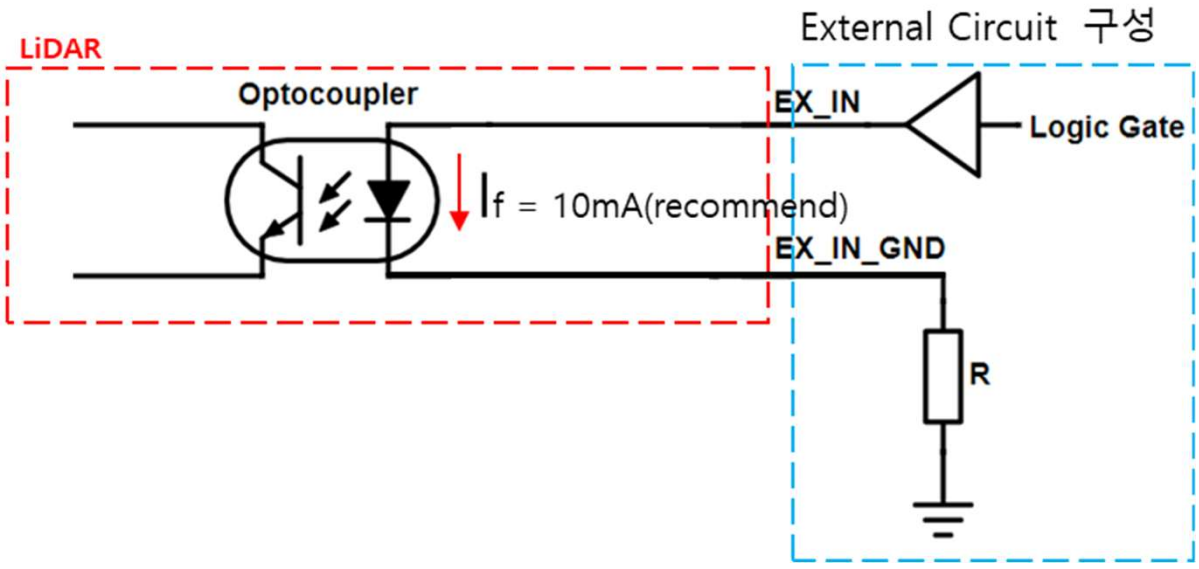
1.2.4.2 External SYNC IN

Provides information on a structure of ML-X sync input inner circuit and outer circuit.

1) Inner Circuit structure of Sync input for ML-X is as follows.



2) Recommands below structure of outer curcuit as below.



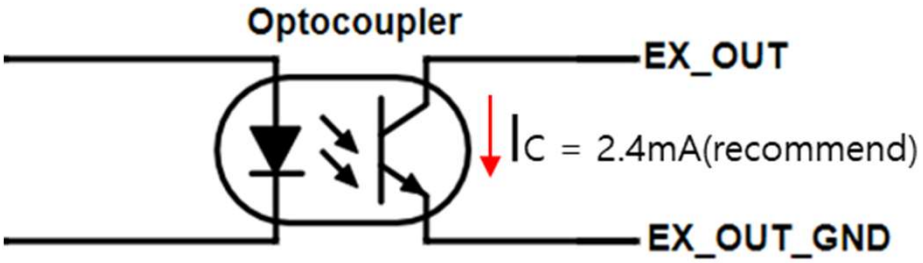
1) EX_Set up resistance value according to IN Voltage

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

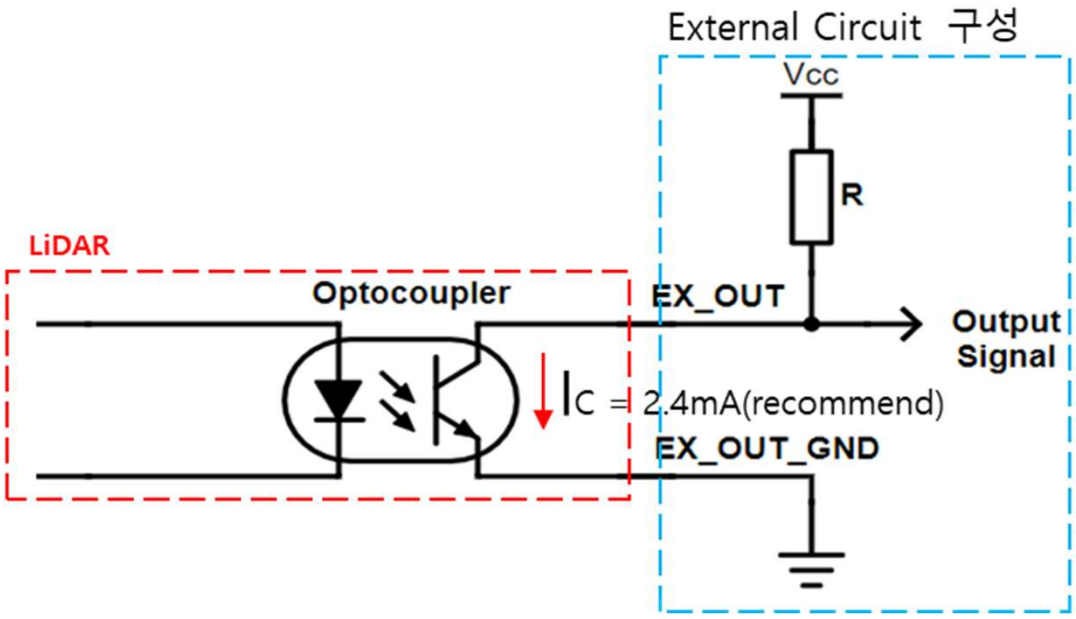
1.2.4.3 External SYNC OUT

Provides information to form a Inner curcuit and Out curcuit structure for ML-X

1) Sync Output Inner curcuit structure for ML-X is as follows.



2) Recommands below structure of outer curcuit as below



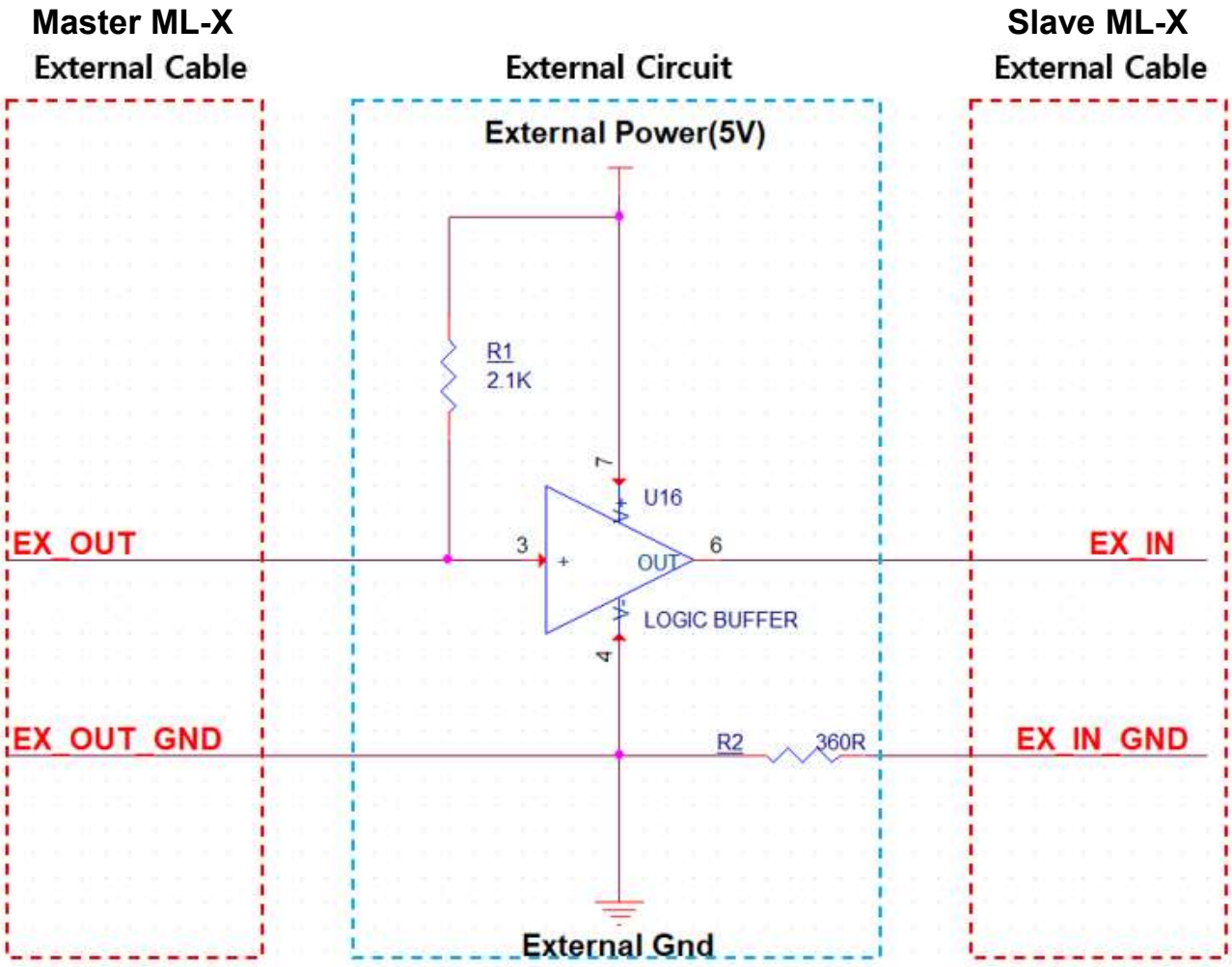
3) Set up resistance value according to VCC

External Voltage (V_{CC})	External Resistor (R)	Collector Current (I_C)	Recommended I_C
3.3 V (Min)	1375	2.4 mA	$1\ mA < I_C < 10\ mA$
5 V	2100	2.4 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

Connection Example of Single Master ML-X <-> Single Slave ML-X

1) Forms External Circuit
    - Outer Power: 5V

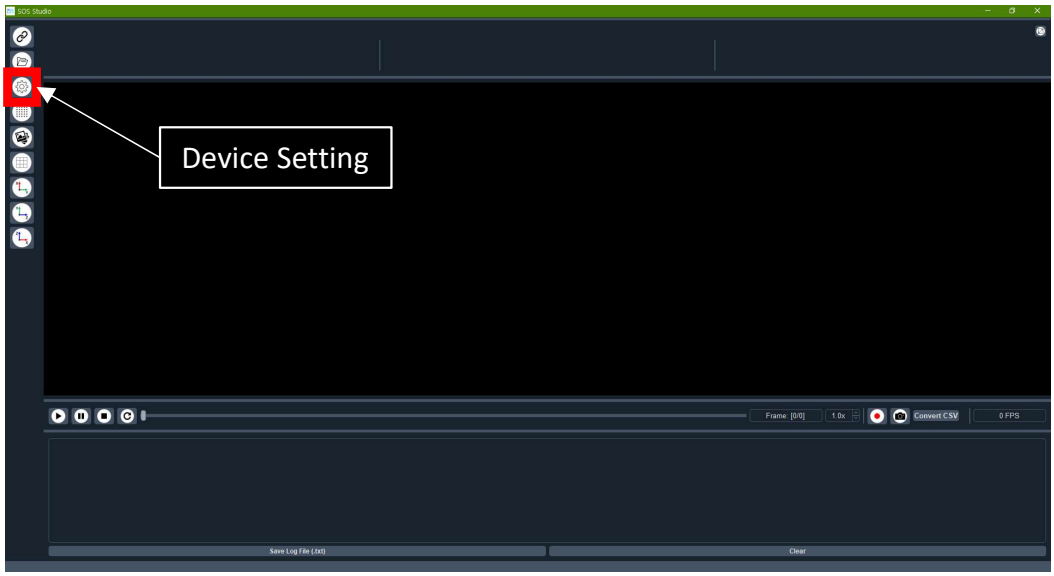


1.2.4.5 SYNC FUNCTION

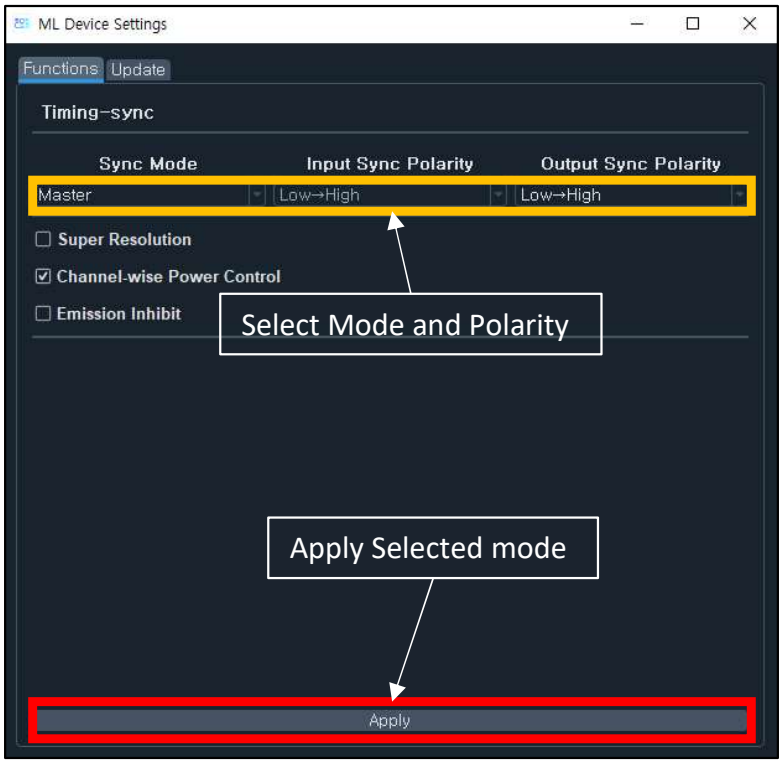
Can select SYNC MODE to Master or Slave for ML-X with SOS Studio

1) Sync Mode installment

- Device Setting Click after running SOS Studio



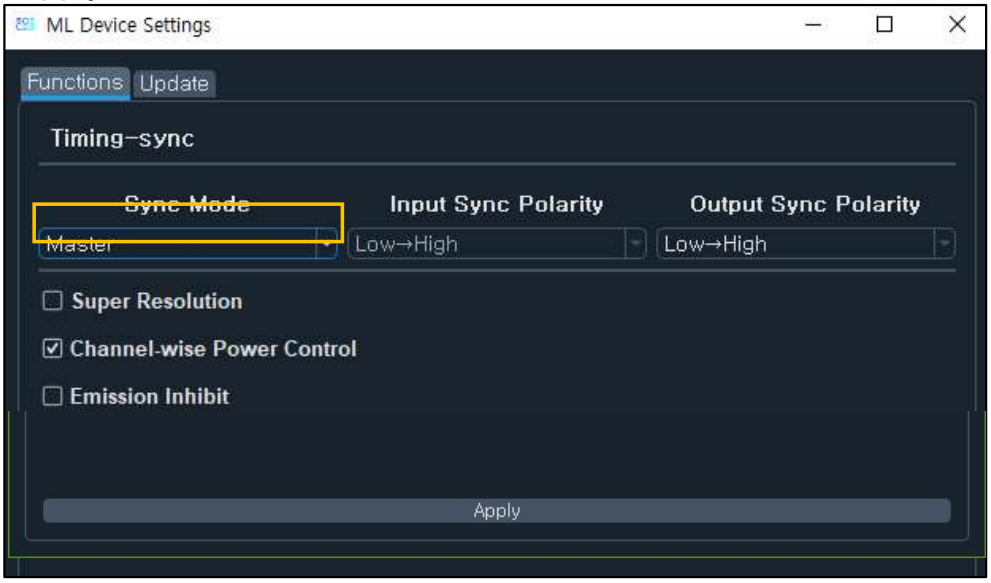
- Can select mode and apply on Function Tap



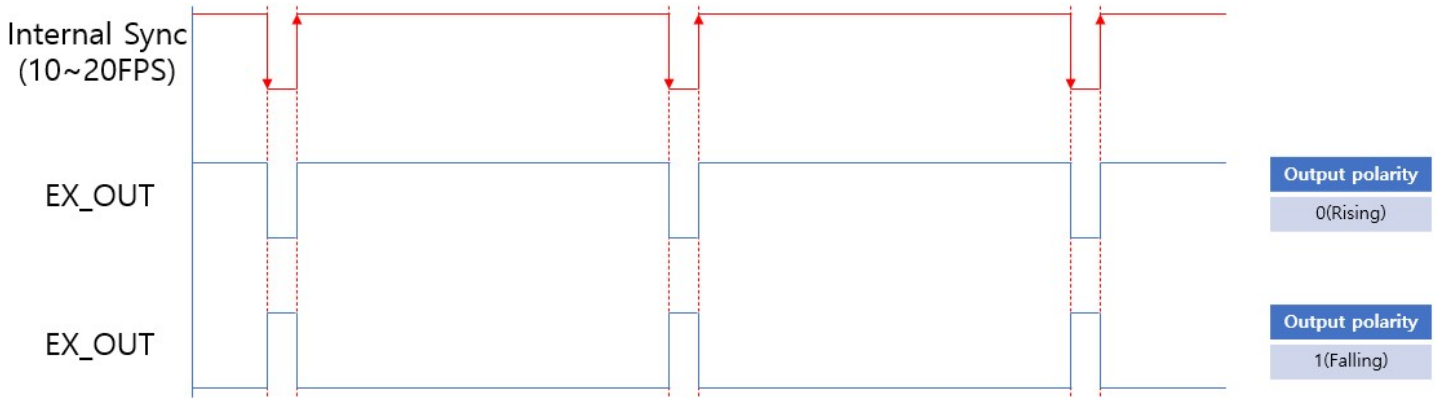
1.2. Electrical Interface

2) MASTER

- Master Mode can be activative by selecting Master on Sync Mode and Click Apply



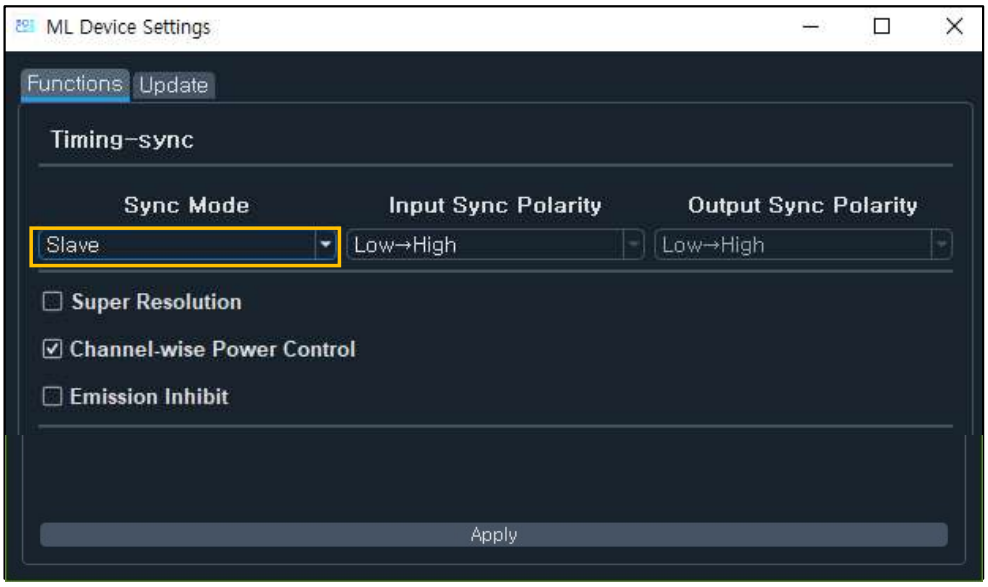
- Makes output for Synchronized signal every frame in line with Selected Frame Rate. Polarity of Sync out Can be changeable (High → Low or Low → High)



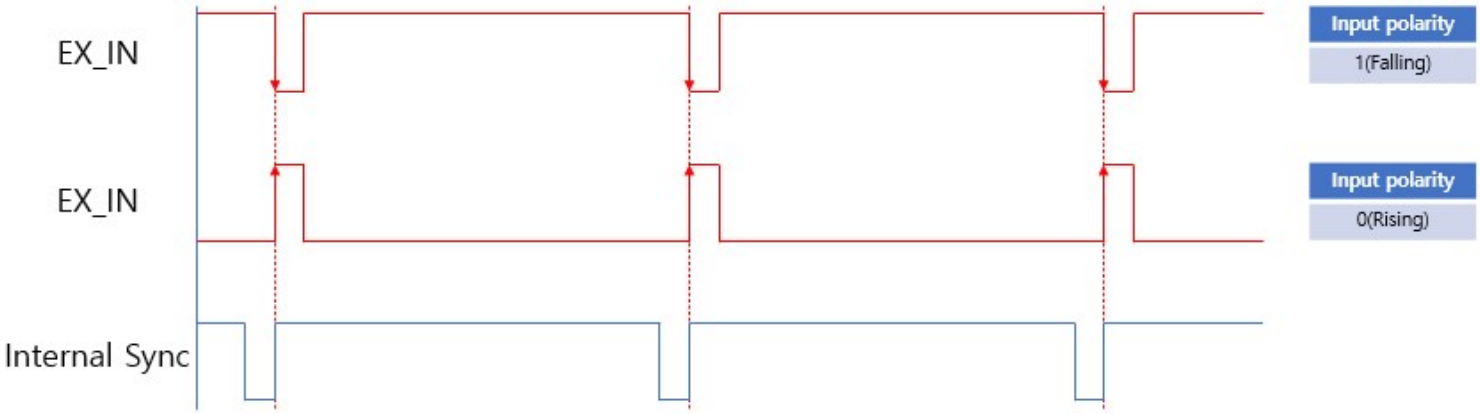
1.2. Electrical Interface

3) SLAVE

- Slave mode can be activative by selecting Slave on Sync Mode and Apply

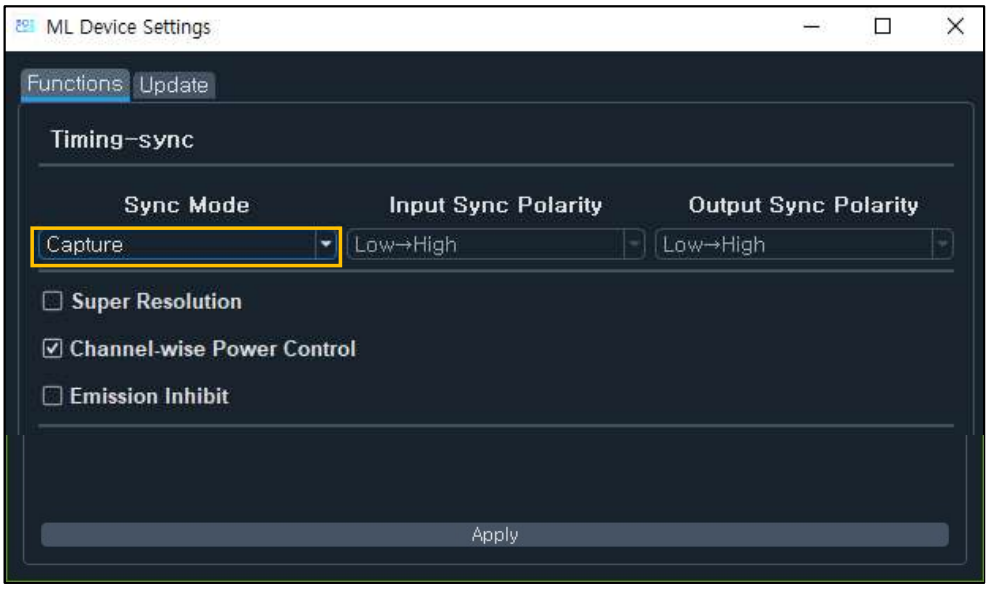


- Activates frame basis In line with Synchronized signal timing received.
  Sync input Polarity(High → Low or Low → High) changeable

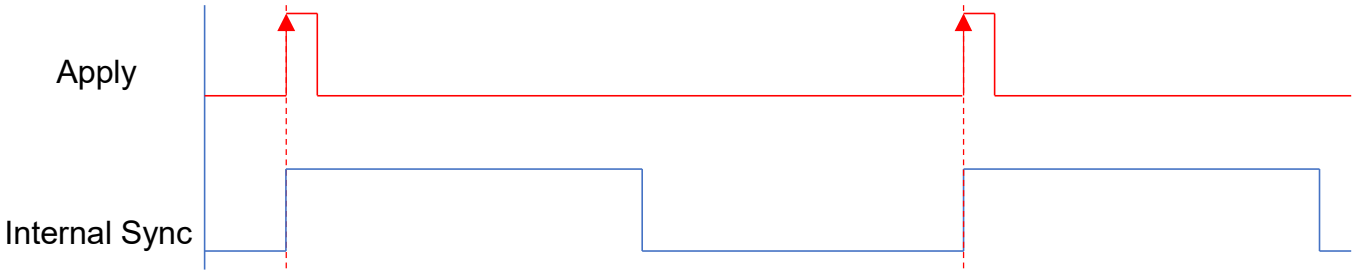


1.2. Electrical Interface

- Capture mode is activative by selecting Capture on Sync Mode and Click Apply



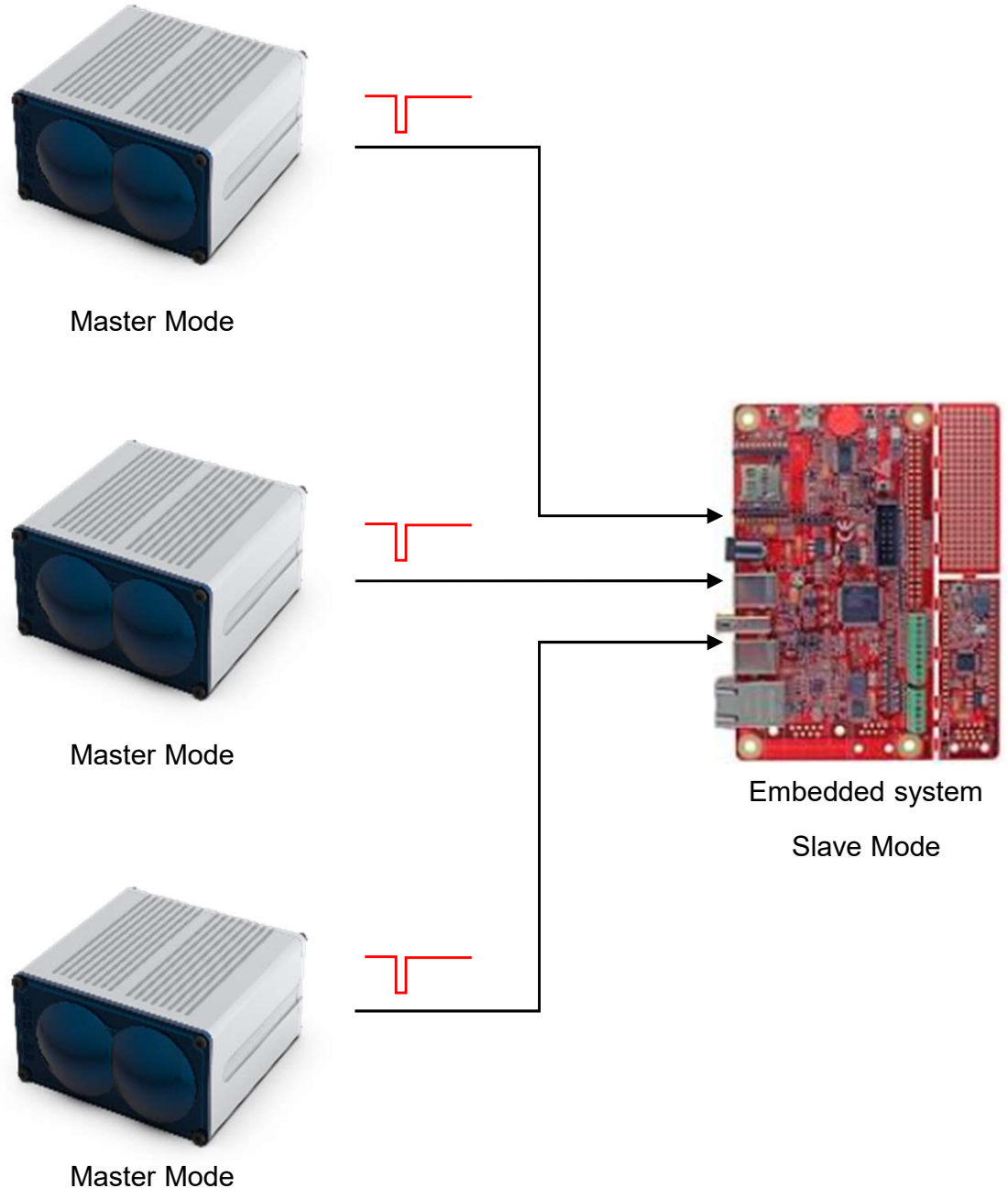
- Update the scene by only 1time internal Sync when clicking apply every times.



1.2.4.5 APPLICATION

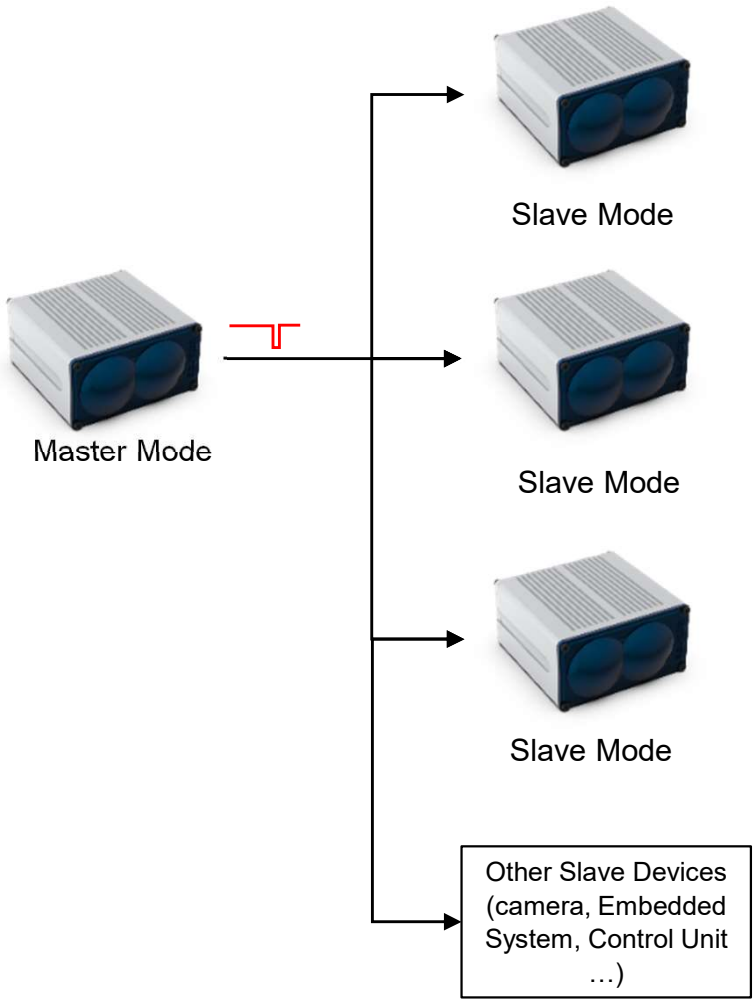
Can make a synchronized composition with other external devices or multiple ML-X with using In/out Synchronized signal.

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

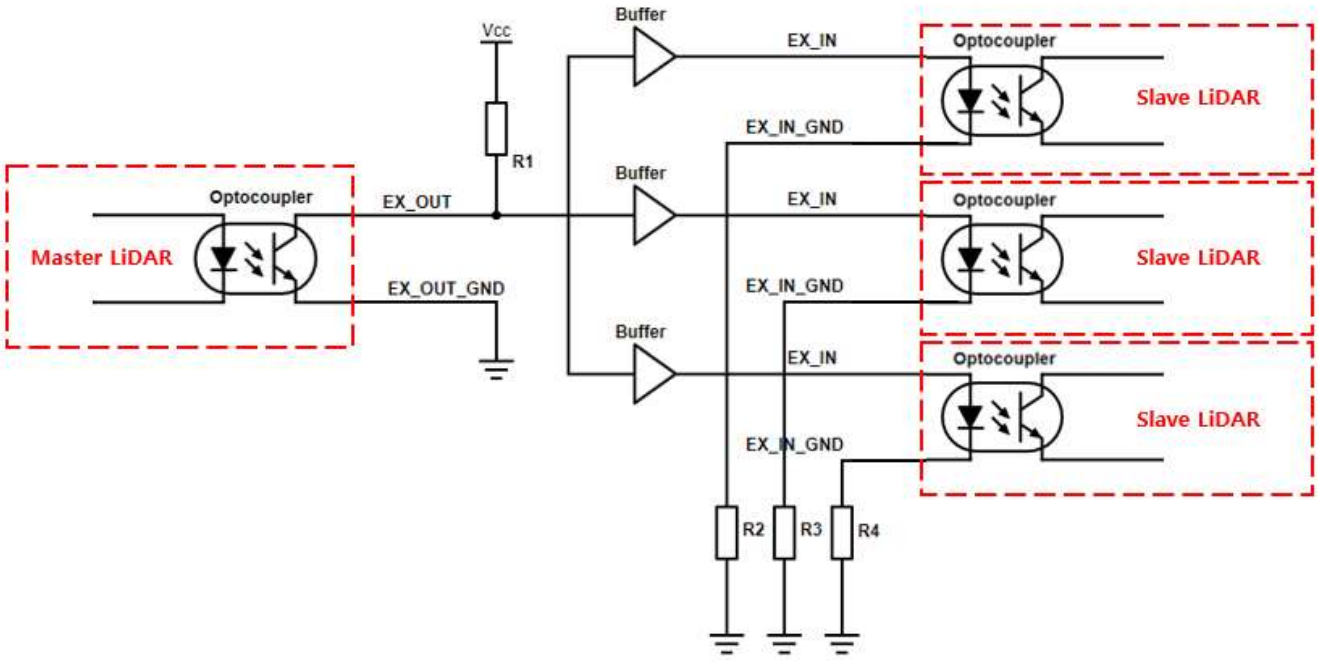


1.2. Electrical Interface

2) Single Master ML-X → Multiple Slave Device

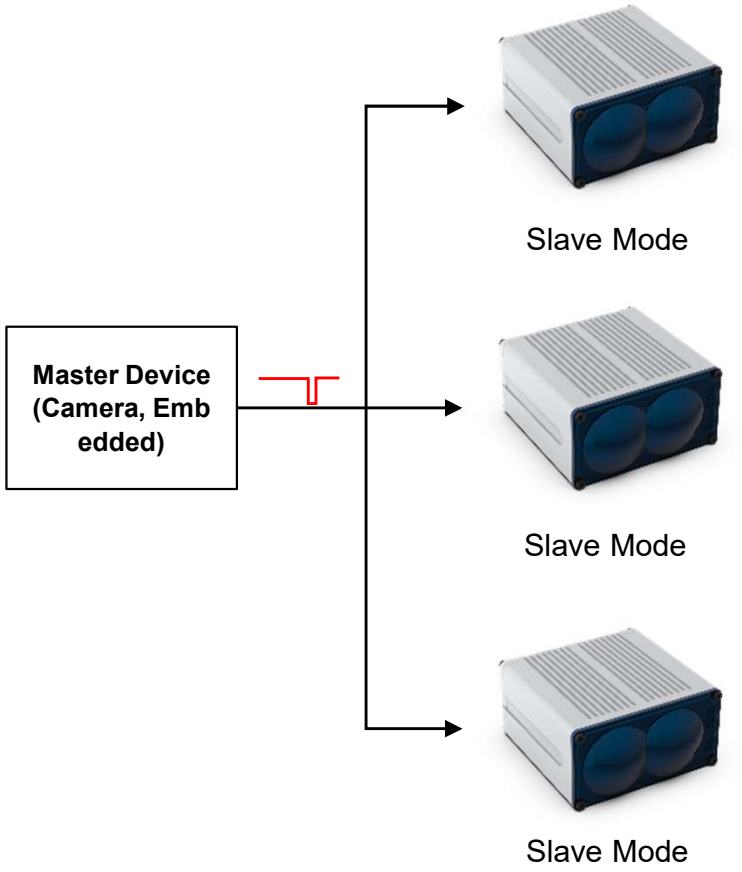


- External Circuit structure

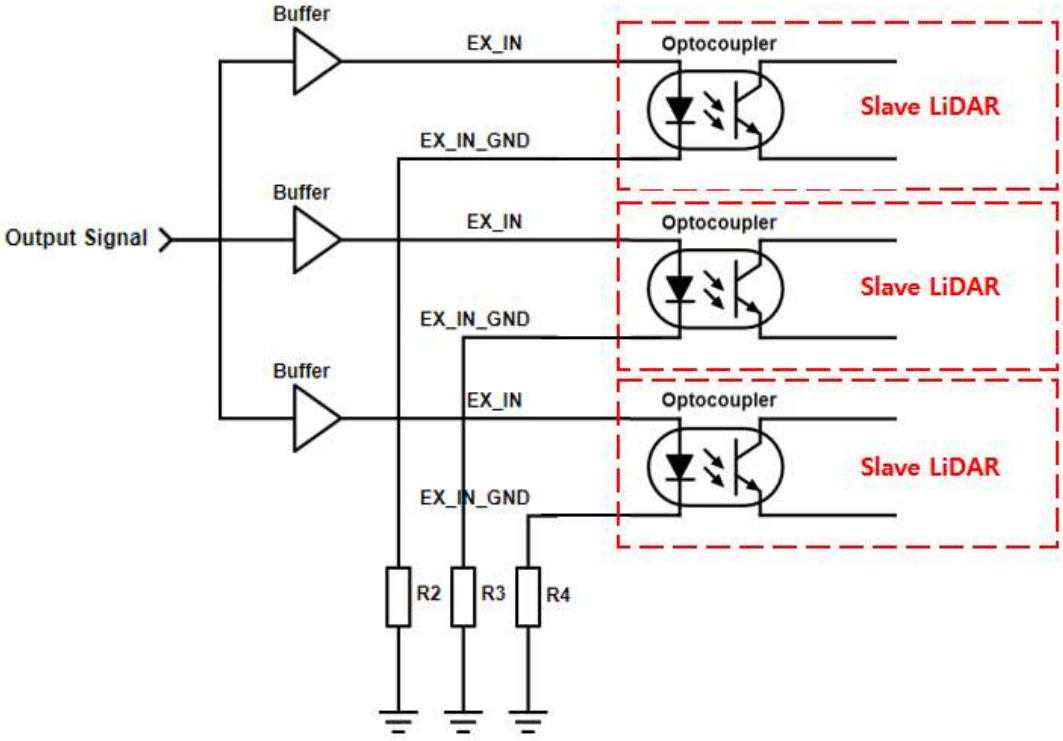


1.2. Electrical Interface

3) Single Master device → Multiple slave Device

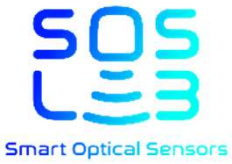


- External Circuit structure



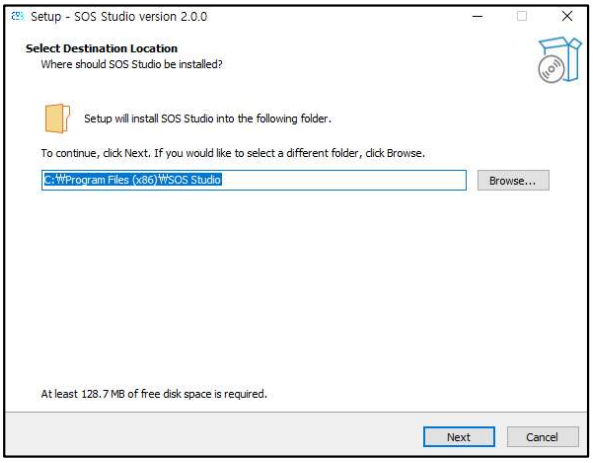
Chapter 2

SOS Studio

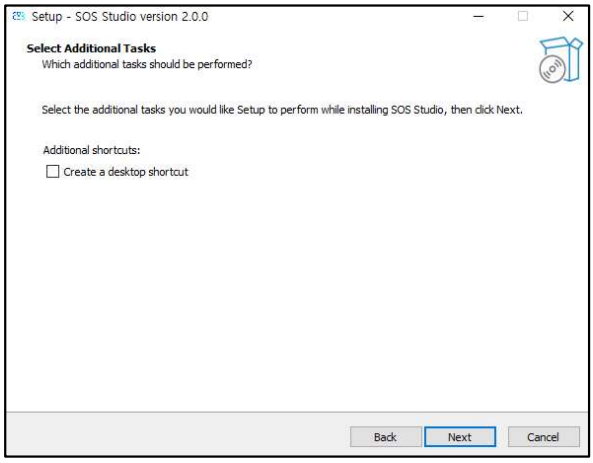


2.1 Installation

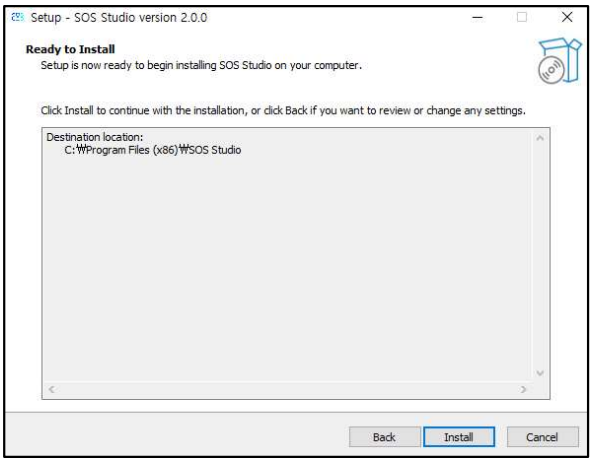
Install SOS studio. Run “SOS Studio_setup.exe” installation file to complete the set-up.



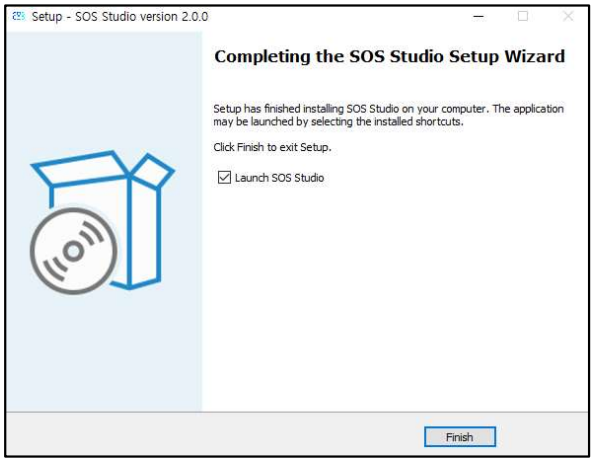
(a) Select the place for installation



(b) Check desktop shortcuts folder



(c) SOS Studio Installation



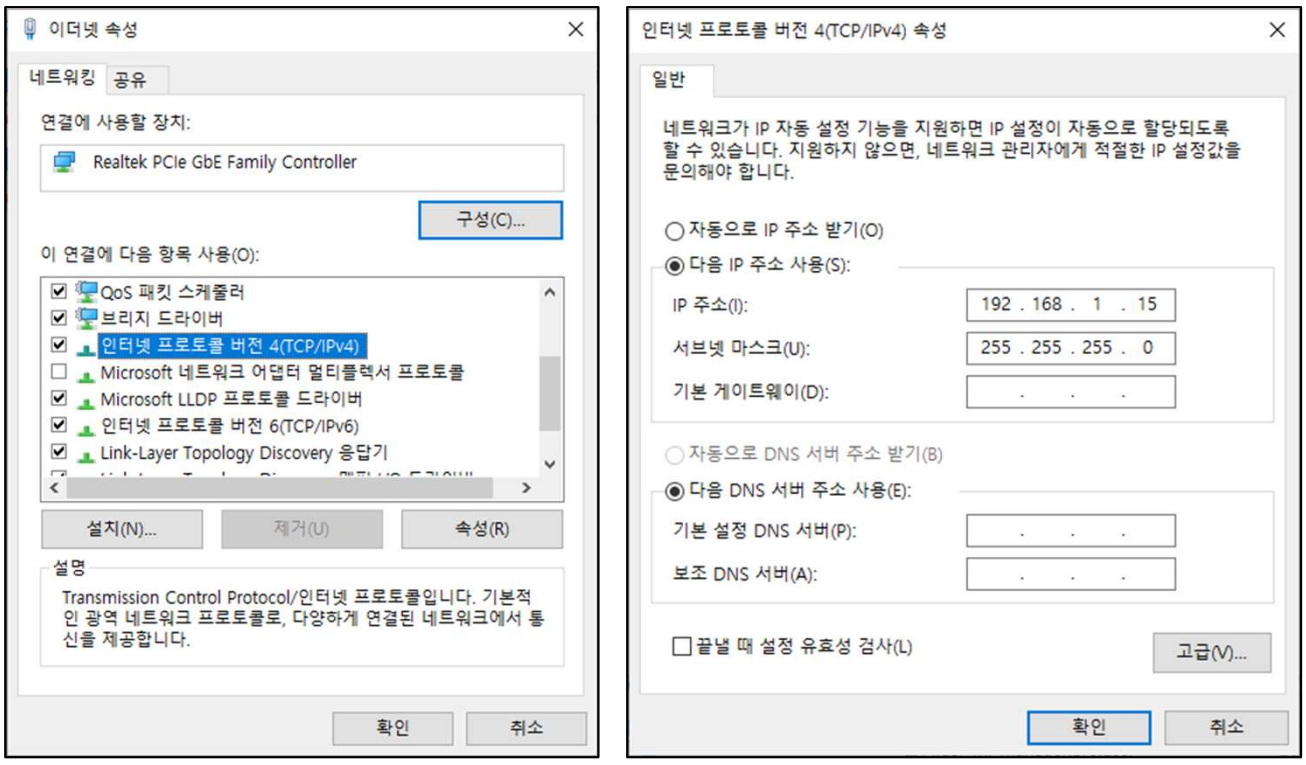
(d) SOS Studio Installation complete

SOS Studio can be run after the Installation.

2.2 TCP/IP Setting

Proceed with TCP/IP set up for connection between ML-X LiDAR and PC

1) Connect LiDAR and PC with Network cable after the power connection with ML-X
2) Run Control panel > Network and Internet > Network and share center
3) Open window of Properties after activated Ethernet.
4) Select Internet Protocol version4 (TCP/IPv4) and click Properties on it.
5) Click “Following IP Address use” option on Properties window.
6) Select IP Address (192.168.1.15) and subnet Mask (255.255.255.0) as belows and click confirm.

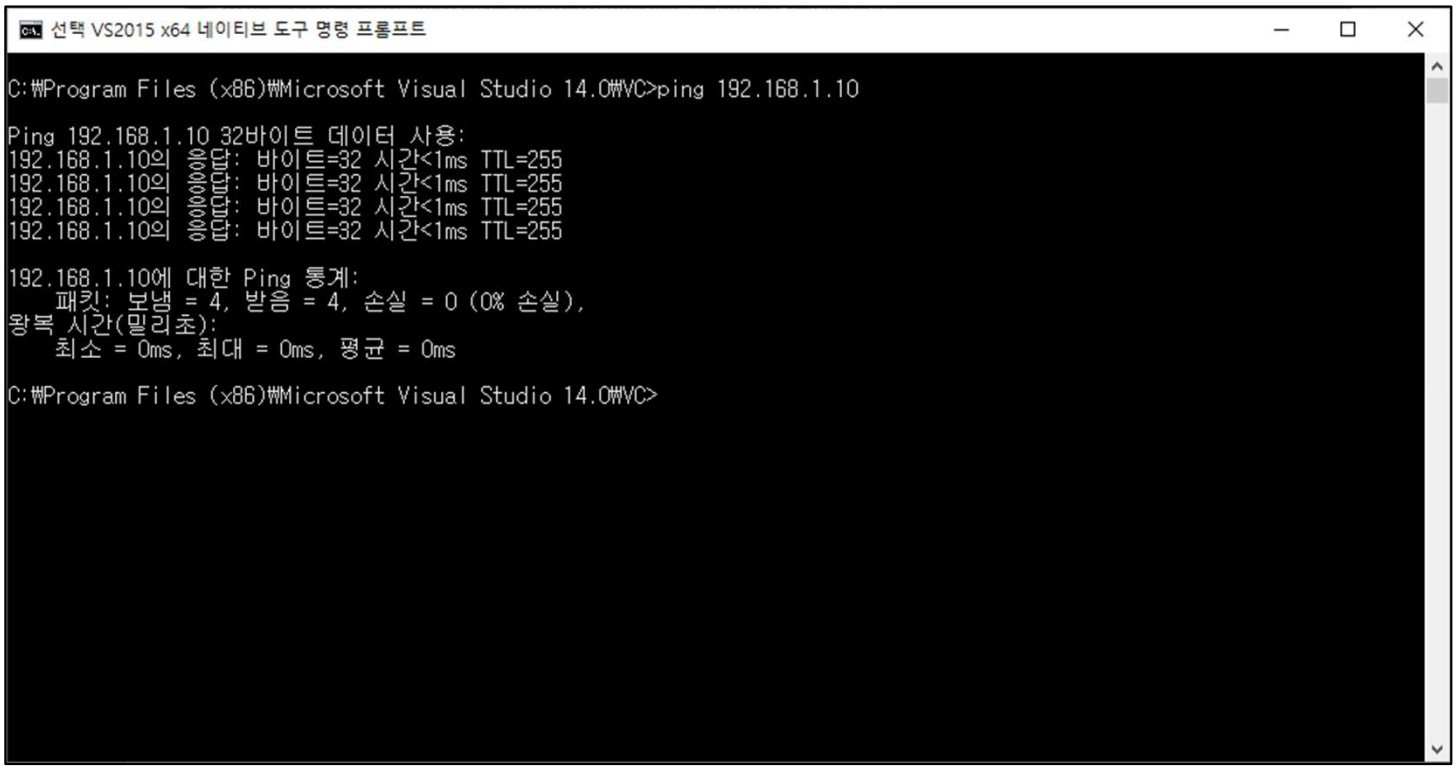


Ethernet Properties window / Internet Protocol Version4 (TCP/IPv4) Properties window Set-up

2.2 TCP/IP Setting

When completing IP and Cable connection set-up, You can make sure if the PC and ML-X are well connected through below Ping test.

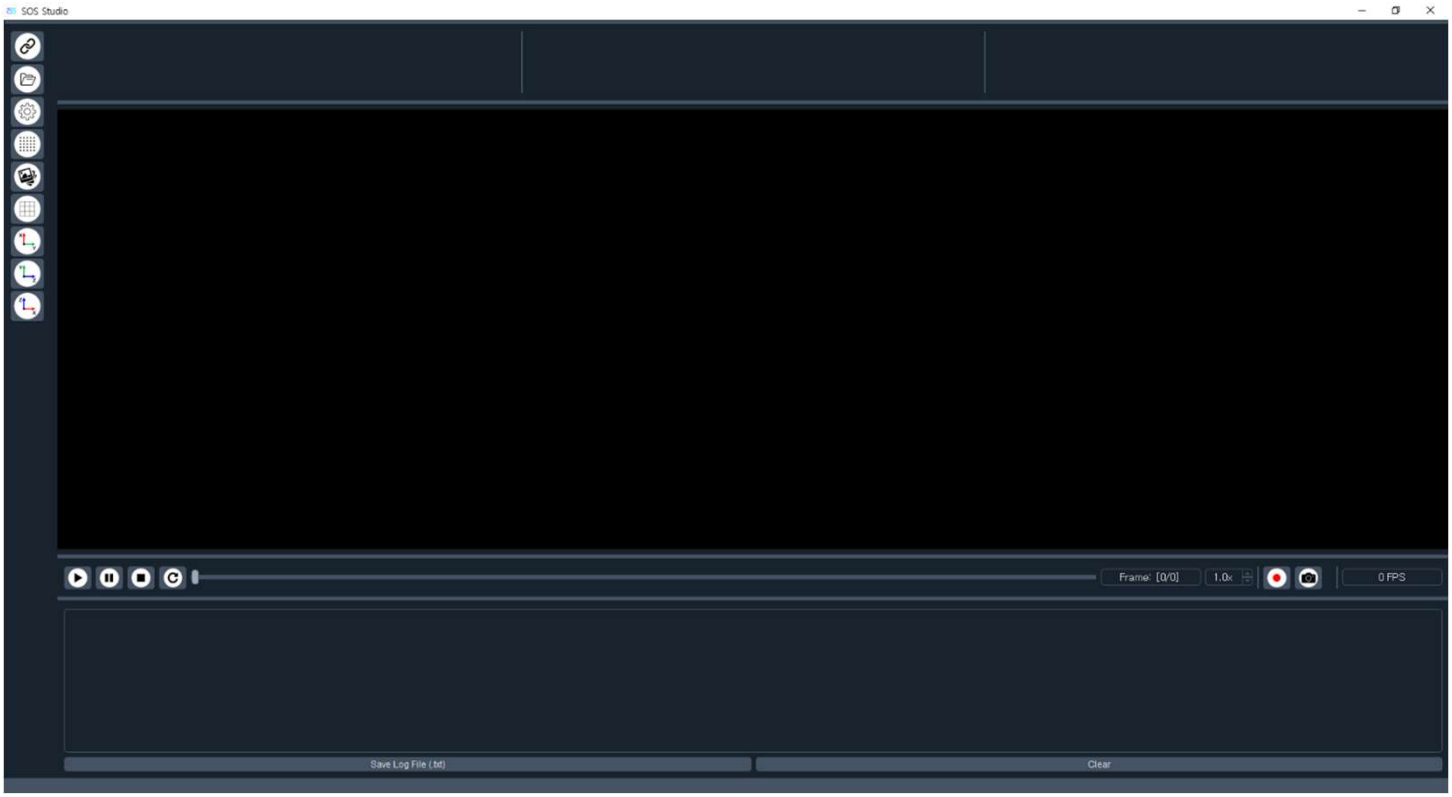
1) Input ‘cmd’ on window searching bar to activate Order prompt
2) Input ‘**ping 192.168.1.10 -t**’ for order
3) Below Message pops-up when ML-X LiDAR and PC are well connected.



.

2.3 Connection

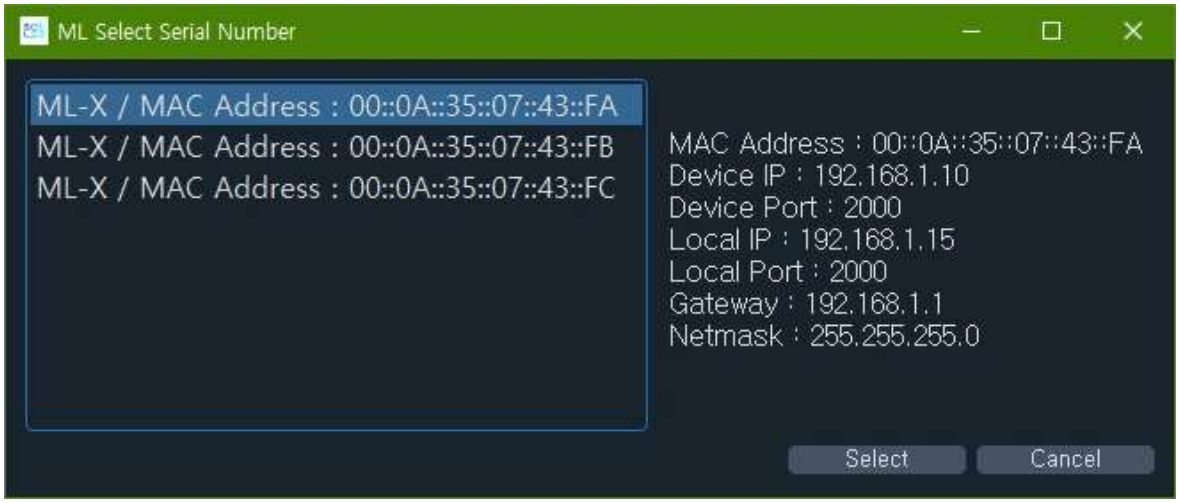
Below shows SOS Studio activated screen.



SOS Studio Activated screen

Click on top of menu of left side “Connection”

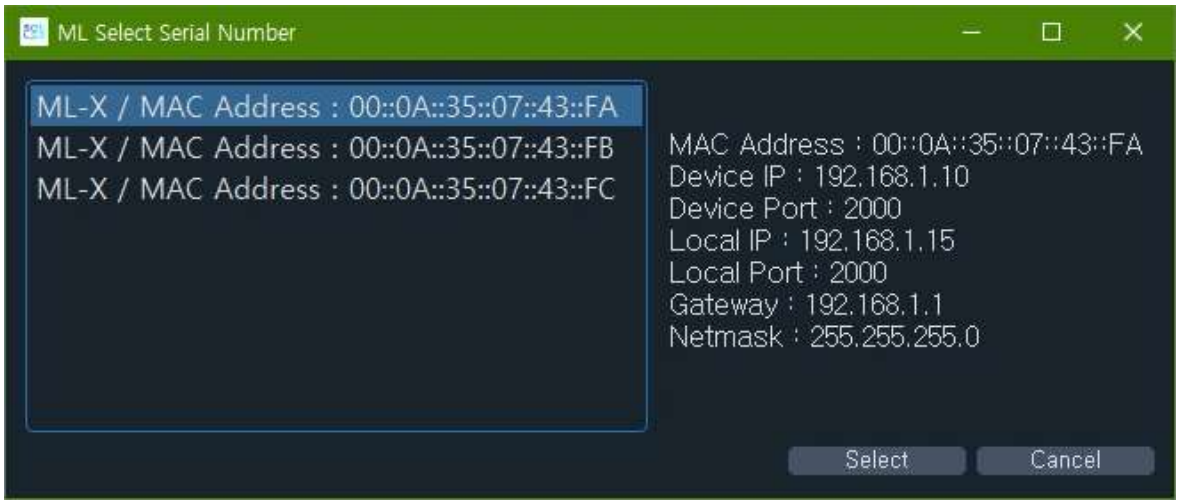
to pop up Available ML-X connection list in order to Connect PC and ML-X



ML-X list

2.3 Connection

Select ML-X to connect on ML-X list and click “Select”. Then Selected ML-X IP and Port are automatically updated on connection pop-up window. By clicking “Connect” on Connection Pop-up window, PC and ML-X device are connected.

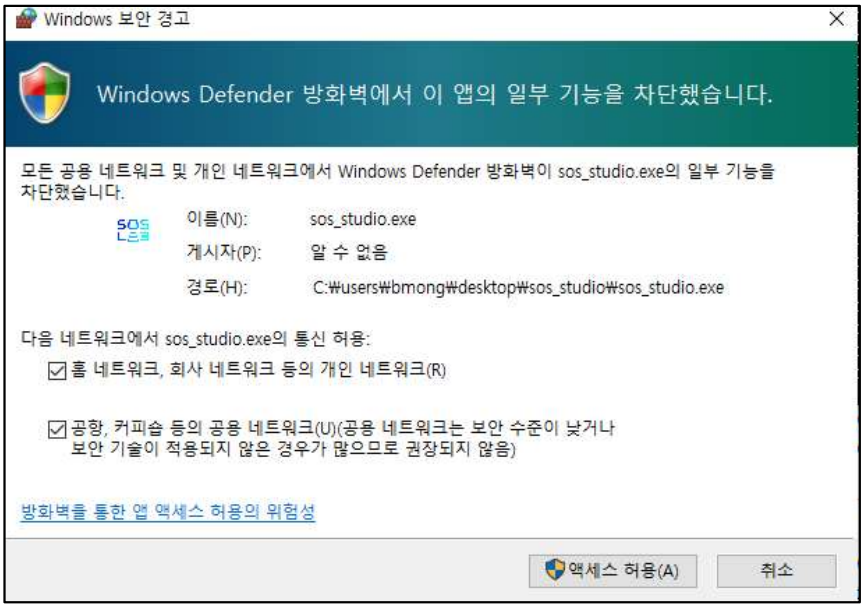


ML-X list

Connection Pop up window

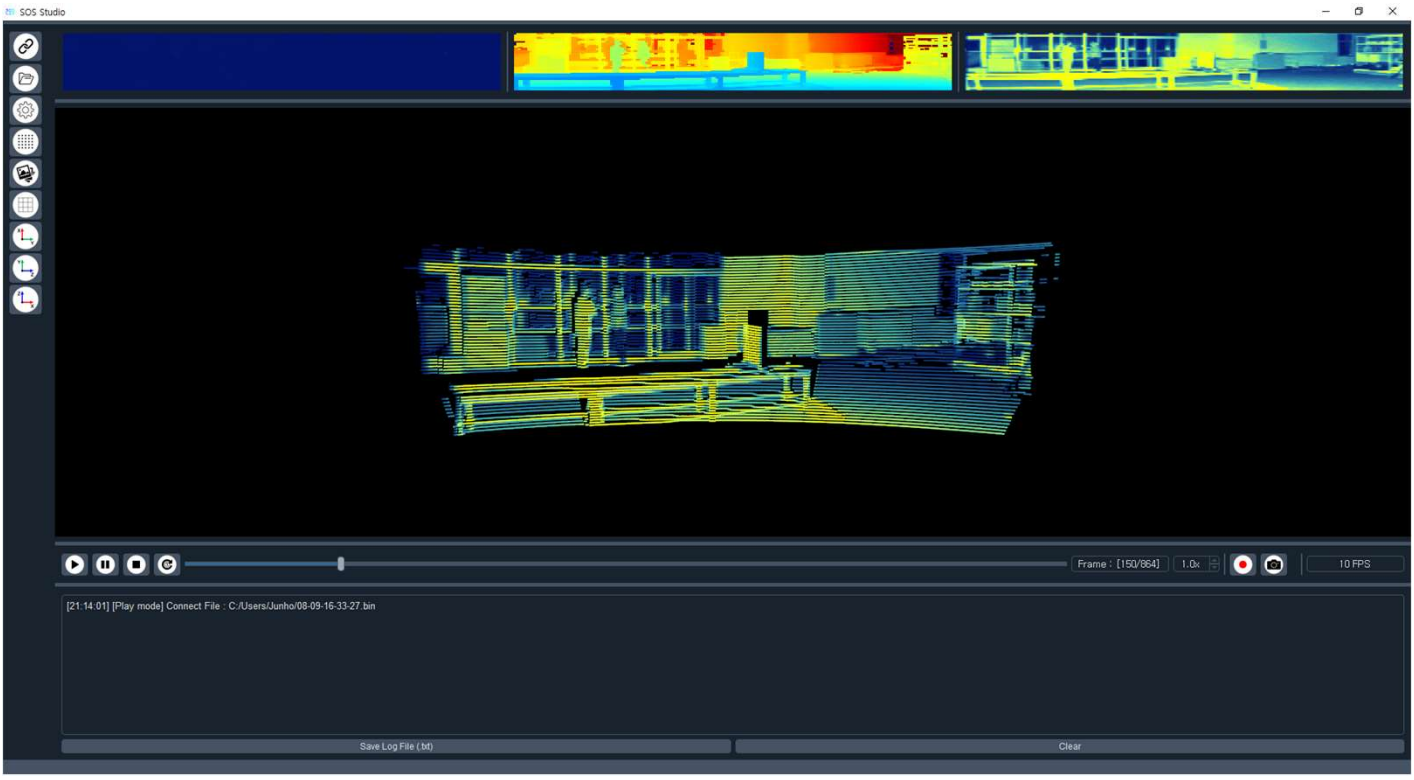
2.3 Connection

When to connect PC and ML-X first time, window security warning pops up as below. Click Access Allow(A) followed by checking 2check box as below .



Window Security warning

When Connection complete between PC and ML-X, Image viewer and central point cloud viewer will be activated as below.

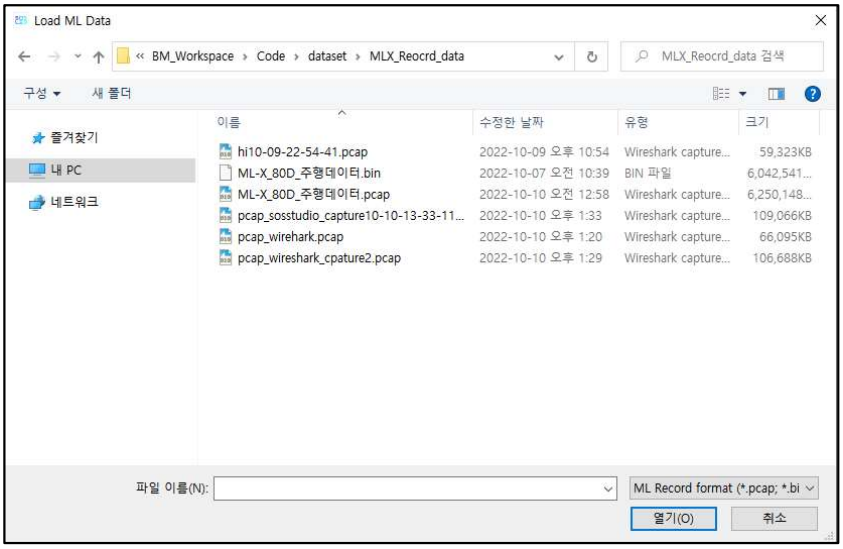


SOS Studio Activated screen

2.4 Data Load

Data Load tool is for recalling ML-X data recorded. Fuction to data storage and replay the data is described in 2.10 Play / Record Mode.

File selective window pops up as below if you click “data Load” on the 2nd botton on the left side of Menu in SOS Studio to recall ML-X data. then, Select stored ML-X data file (.pcap, .bin) and Click “open” button to replay ML-X data

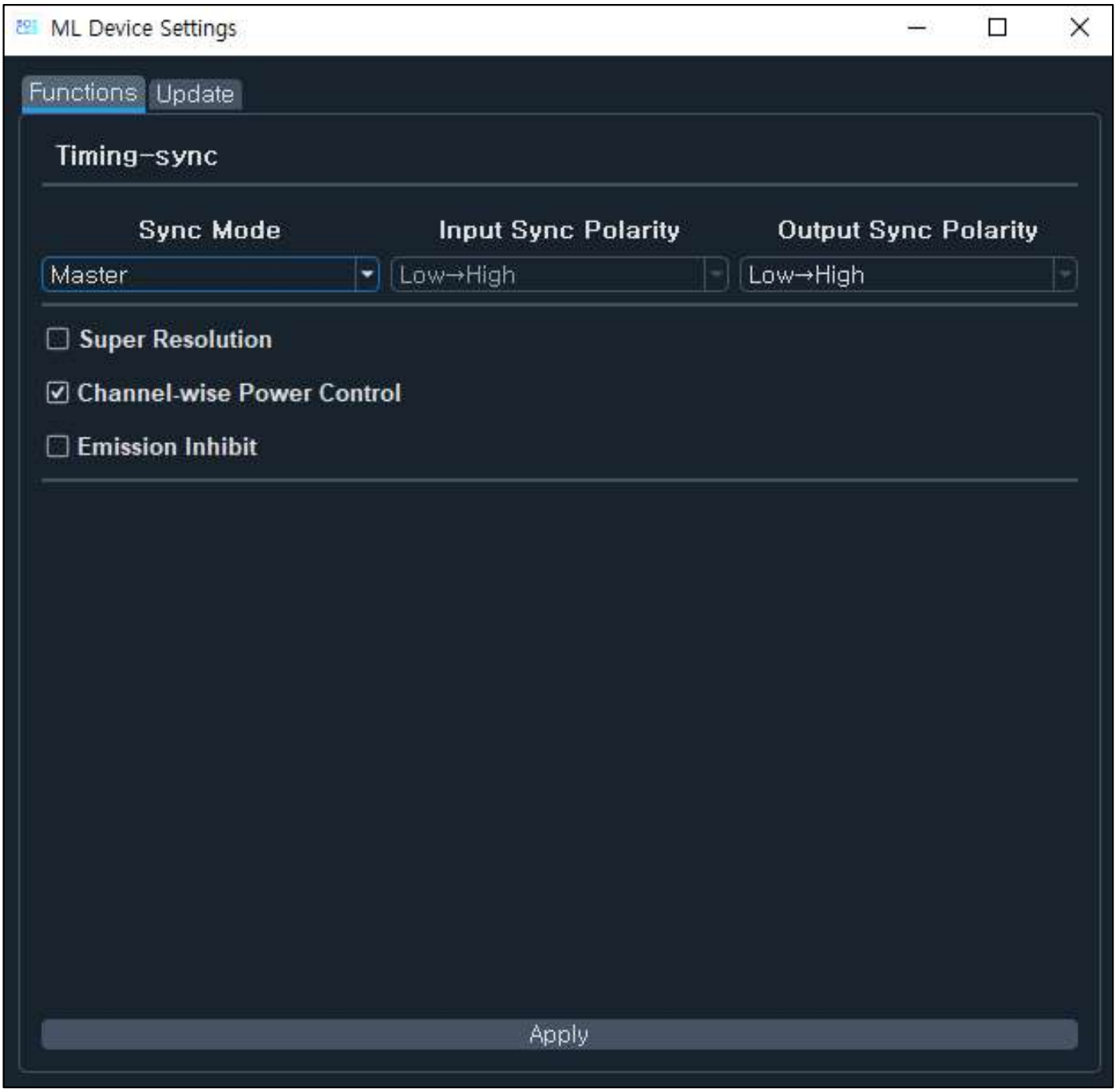


데이터 불러오기 팝업 창

Through above play bar tool, you can replay ML-X data after recalling the data. Play bar tool is described on **2.8 Play / Record Mode**

2.5 Device Setting

Device setting provides Timing Sync tool, , Super Resolution on/off tool, IP chanfinf rool, FW update tool for ML-X Device. Timing Sync and Super Resolution on/off tool are provided at “Functions” tap and IP changing and FW update tool are provided at “Update” tap.

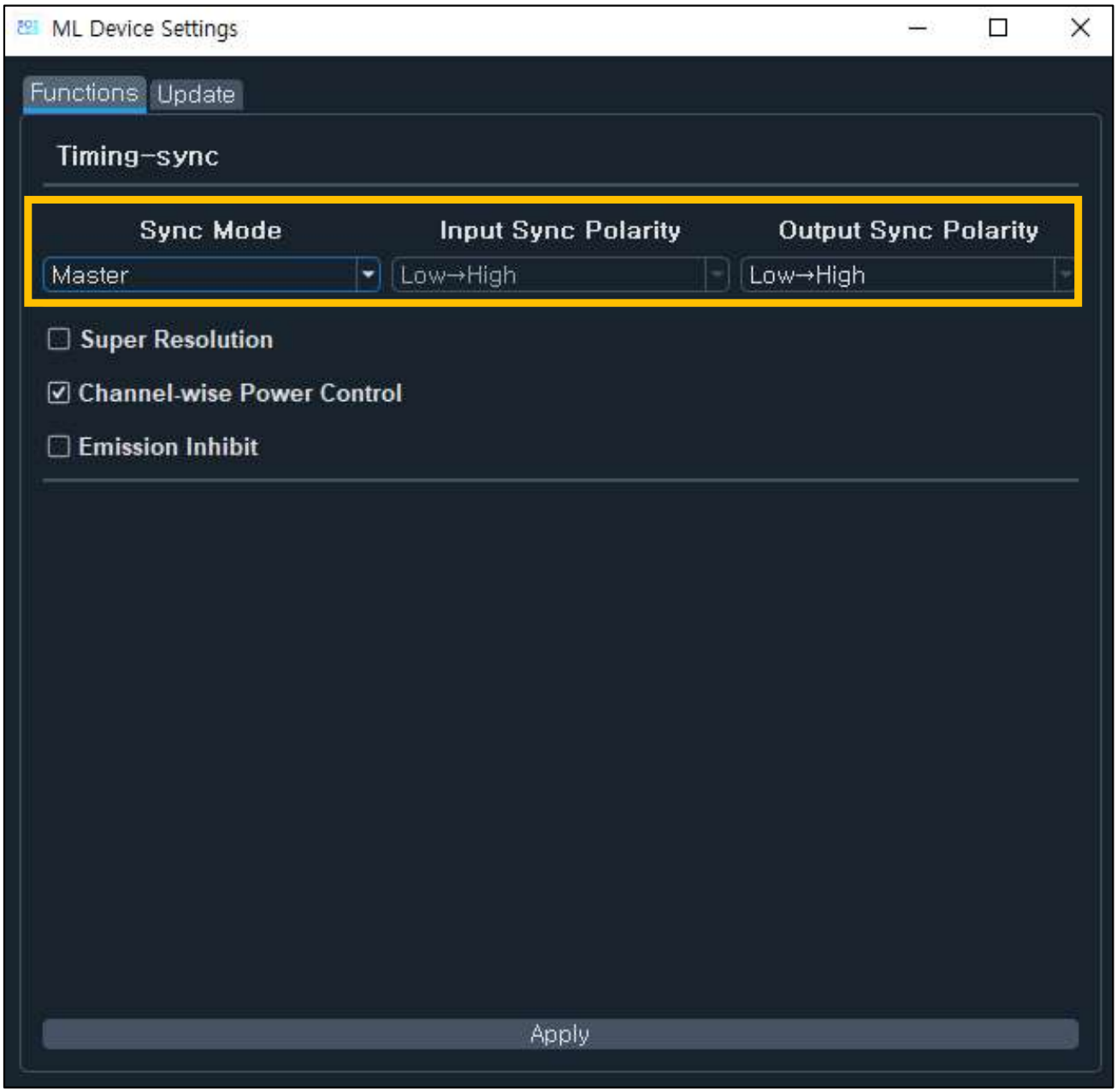


Device Setting Window

2.5.1 Timing Sync

Timing sync tool for ML_X Device is provided at “Device Setting” – “Functions” tap. Timing Sync supports “Master” mode, “Slave” mode, “Capture” mode. “Output Sync Polarity” can be set up at “Master” mode and “Input Sync Polarity” can be set at , “Slave“ mode. Please seletct the mode you want and Polarity and click “Apply” button to set up the certain mode.

Also refer to ML-X Hardware Configuration document for more detail.



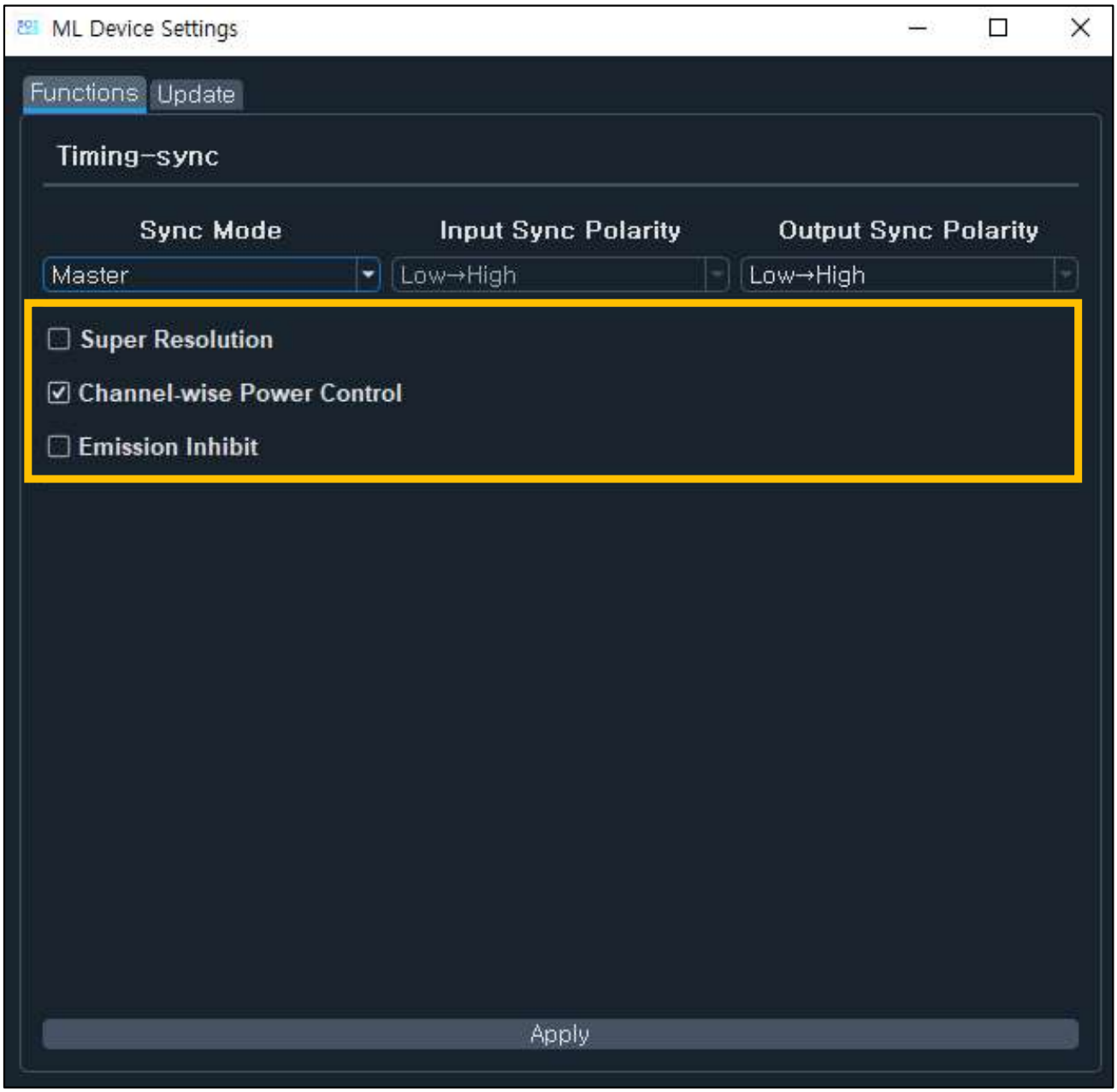
Device Setting window – Timing sync tool

2.5.2 Functions on/off

On/off for Super Resolution / Channel-wise Power Control / Emission Inhibit tool for ML-X Device are provided at “Device Setting” – “Functions” . Click all check box for each tool and click “Apply” botton to turn it “on”. On the other hand, leave all check box not checked and clik “Apply“ button to turn it “off”.

Initial screen is as below.

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



Device Setting window – Functions On/Off tool

2.5.3 IP Changer

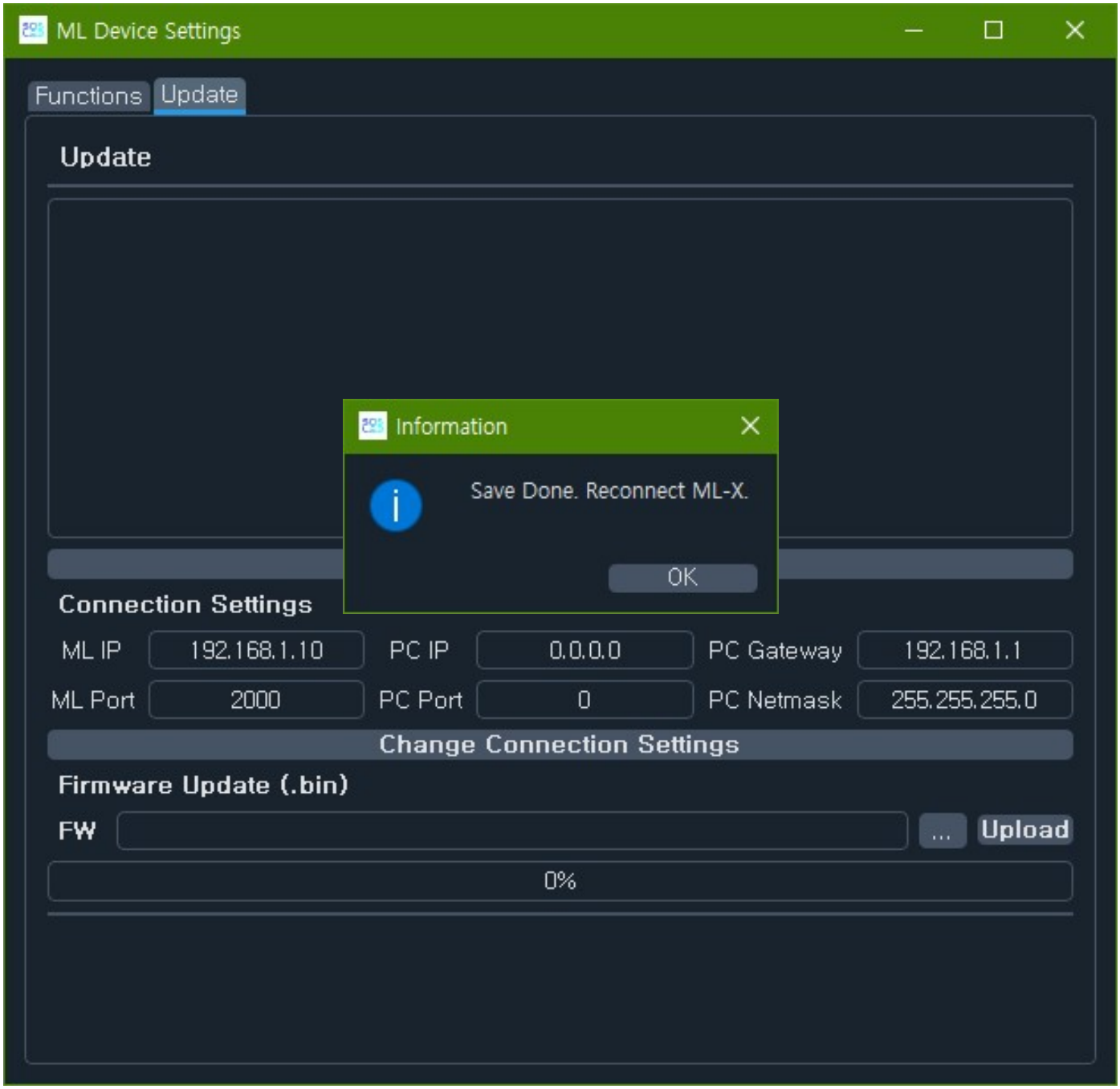
ML-X Device IP Changing tool is provided at “Device Setting” – “Update” tap.

IP changing procedure for ML-X Device is below in order.

① Input IP to change for ML-X Device

② Click “Change Connection Settings” button

③ Power cable reconnect after IP Change



IP change result screen

2.5.4 FW Update

FW Update tool for ML-X Device is provided at “Device Setting” – “Update” tap.

FW update for ML-X Device procedure is below in order.

① “…” button click

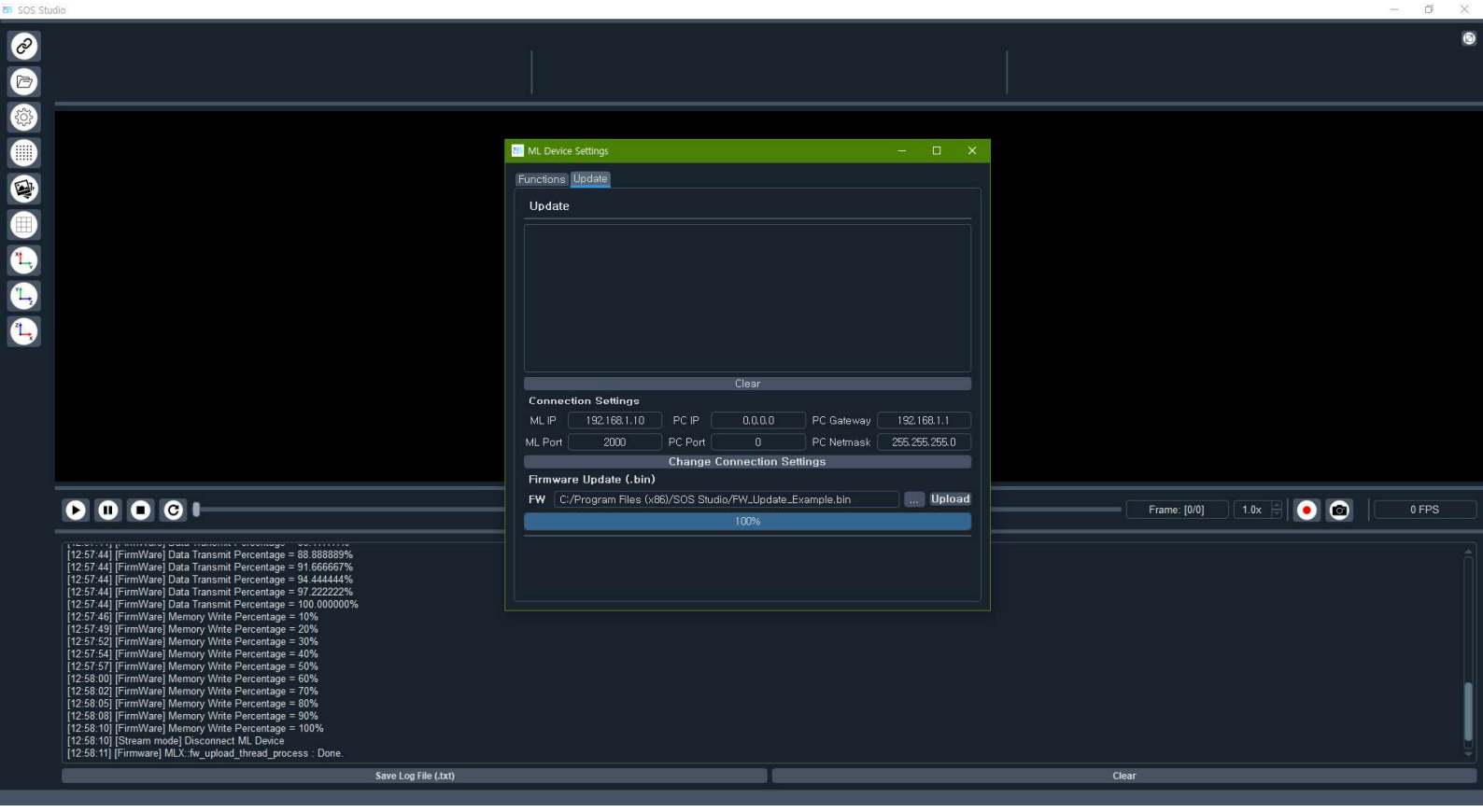
② Select *.bin file for FW update

③ “Upload“ button click.

④ FW Update Complete after Progress bar gets 100%.

⑤ Reconnect the power cable after FW update completion

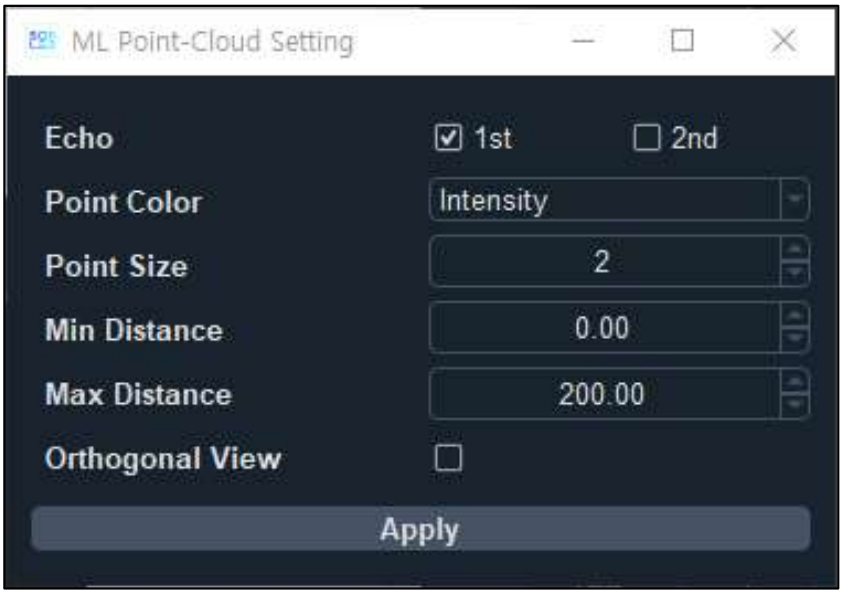
.



FW Update completion screen

2.6 Point Cloud Setting

Point Cloud Viewer located at center of SOS studio can be set up at Point Cloud Setting. Click 4th botton of the left side “Point Cloud Setting” at SOS studio to pop up the window as below.



Point Cloud Setting window

Following is the description of each categories to set up

Section	Description
Echo	Multi Echo Point Cloud visualization set up (First peak, Second peak)
Point Color	Point colar (White, Red, Green, Blue, Ambient, Depth, Intensity)
Point Size	Point size
Min Distance	Visualized Point minimum distance
Max Distance	Visualized Point maximum distance
Orthogonal View	Orthogonal View activation

2.7 Image Setting

Image Viewer located at top of SOS studio can be set up at Image setting.

Click “Image Setting” located at 5th button of left side at SOS studio to pop up below window.

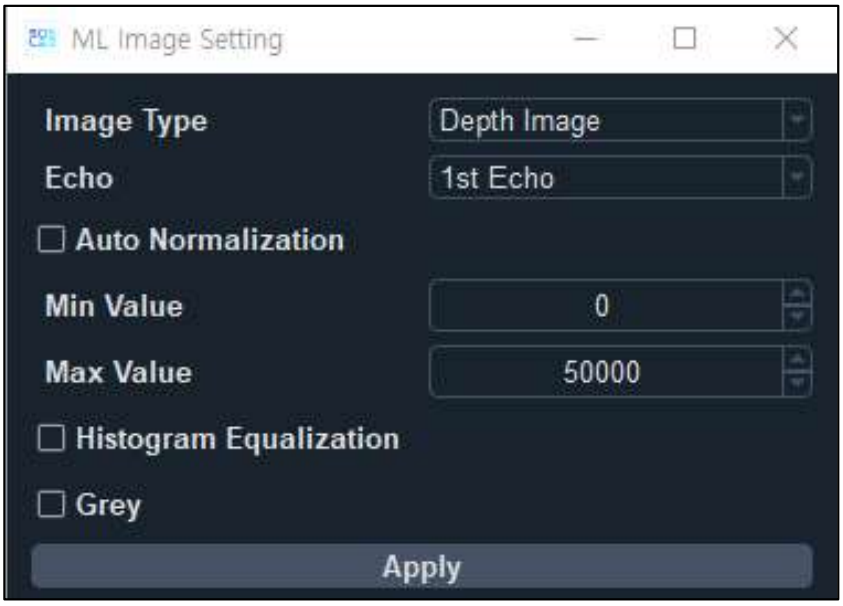


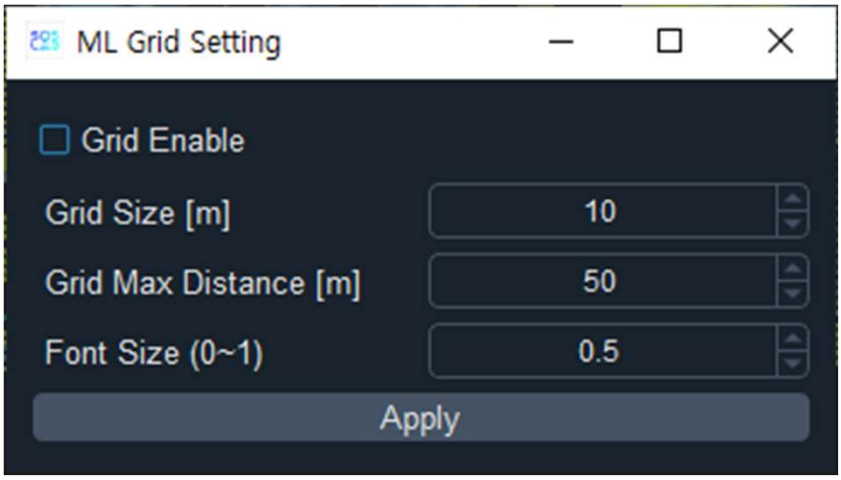
Image Setting window

Following is the description of each categories to set up

Section	Description
Image Type	Select an Image set up (Ambient / Depth / Intensity Image)
Echo	Set up Multi Echo Image visualization(First peak, Second peak)
Auto Normalization	Automatic Normalization : Nomalization to min/max value of Image
Min Value	Nomalization min value
Max Value	Nomalization max value
Histogram Equalization	Histogram Equalization activation
Grey	Change to black/white Image

2.8 Grid Setting

Grid of Point Cloud Viewer located at center of SOS Studio SOS Studio can be set up at Grid Setting. Click “Grid Setting” located at 6th button of left side of SOS studio to pop up window below.



Grid Setting window

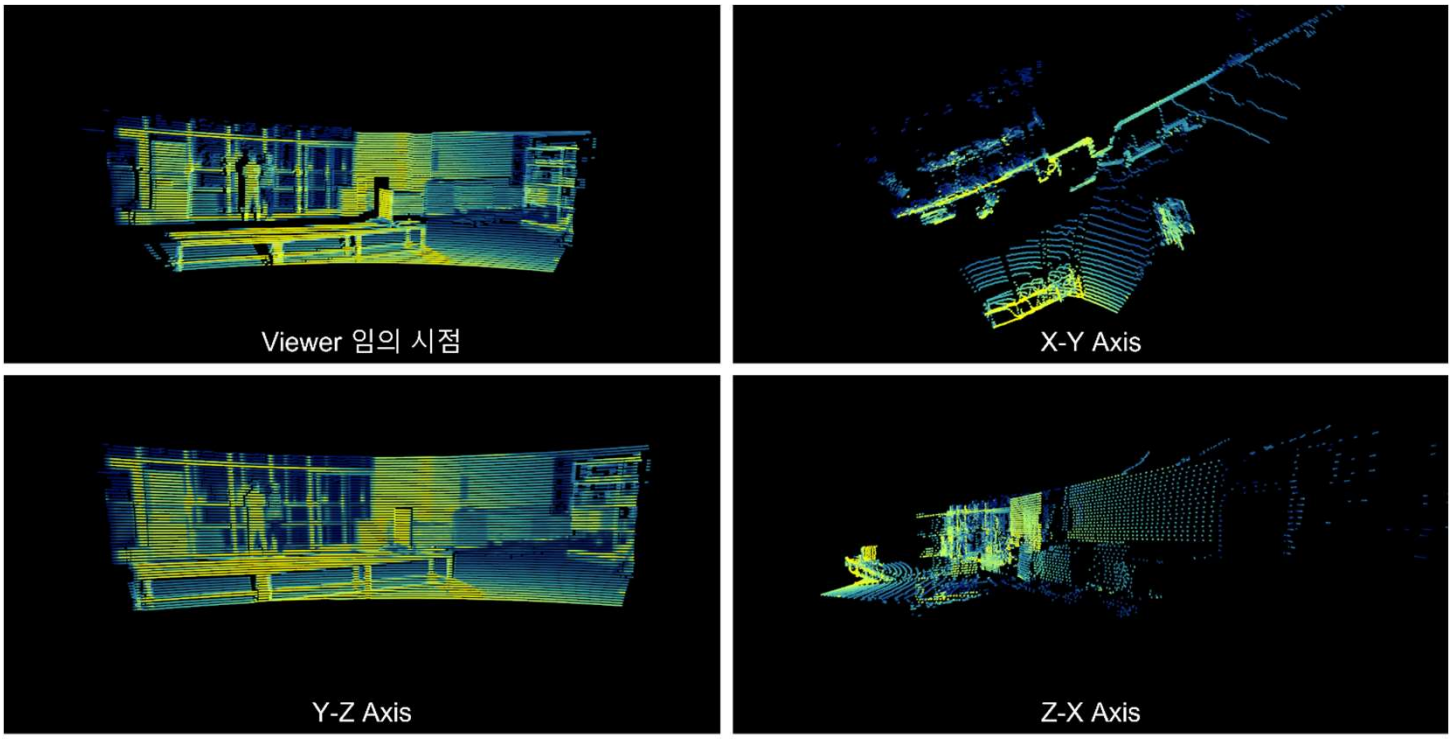
Following is the description of each categories to set up

Section	Description
Grid Enable	Grid Visualization activation
Grid Size	Grid terms
Grid Max Distance	Grid max distance
Font Size	Grid distance[m] unit letter size

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

X-Y / Y-Z / Z-X Axis Supports view angle of Point Cloud Viewer to adequately change in line with certain Axis located at Center of SOS studio.

Each tool supports adequate change of view for certain Axis at certain view angle on Point Cloud Viewer as below.



Point Cloud View angle change up to X-Y / Y-Z / Z-X Axis button

2.10 Play / Record Mode

Play Bar located under Point Cloud Viewer provides tool to record the visualized data after the connection and the replay mode for recalled data for ML-X from **2.2 Data Load.**

.

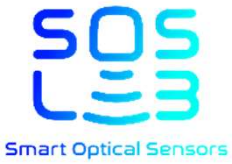
SOS Studio Play Bar

Each tool and description at play bar is as below.

tool	Description
Play	Data replay
Pause	Data replay and stop
Stop	Stop of replay
Repeat	Data replay repeat
Scroll Bar	Show real-time frame out of total frame(Visualized by Bar)
Frame	Show real-time frame out of total frame(show Real-time/Total)
Speed	Data replay speed
Record	Storage of Streaming data or continued data (.pcap, .bin)
Capture	Storage of Streaming or Single data replayed (Images, Point Cloud)
FPS	Data transmit speed for streaming

Chapter 3

ML-X LiDAR API



3.1. ML-X Example Code Build for Ubuntu

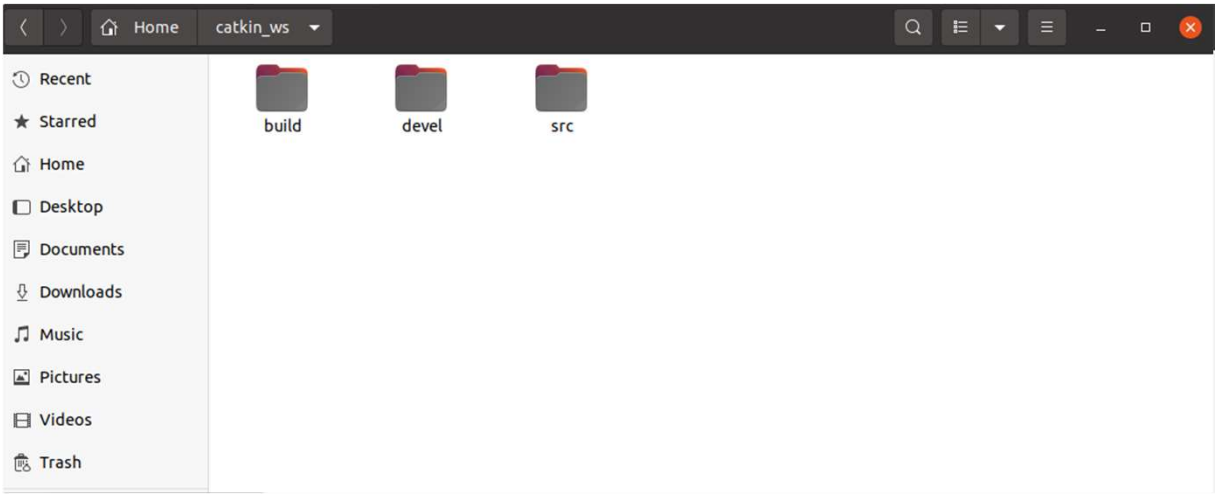
1) ML-X SDK can be utilized at Ubuntu 18.04 / 20.04 and ROS can be utilized at Melodic / Noetic version.
Example code with API and ML-X API are both provided.
ML-X API consists of libsoslab_core, libsoslab_ml. Data Visualizaion and ML-X connection test will be performed via test_ml-x sample code.

3.1. ML-X API Build for Ubuntu/ROS

ML-X API build will be performed through below procedure after ROS set-up complete. ml package is created by inputting below Command at catkin_ws folder.

```
$ catkin_make
```

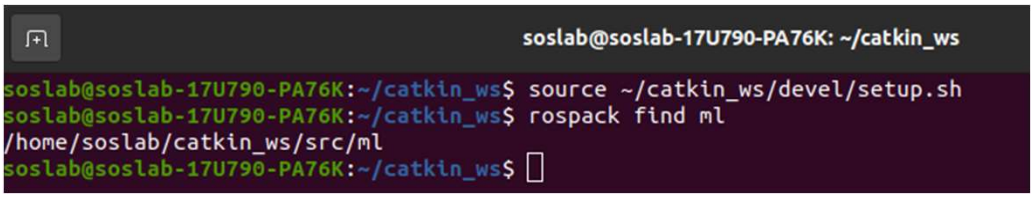
ml Package creation by using catkin_make



2) In order to add ml package created by Build to ROS environment, pls input command below at ROS environment to add ml package.

ml Package build result

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

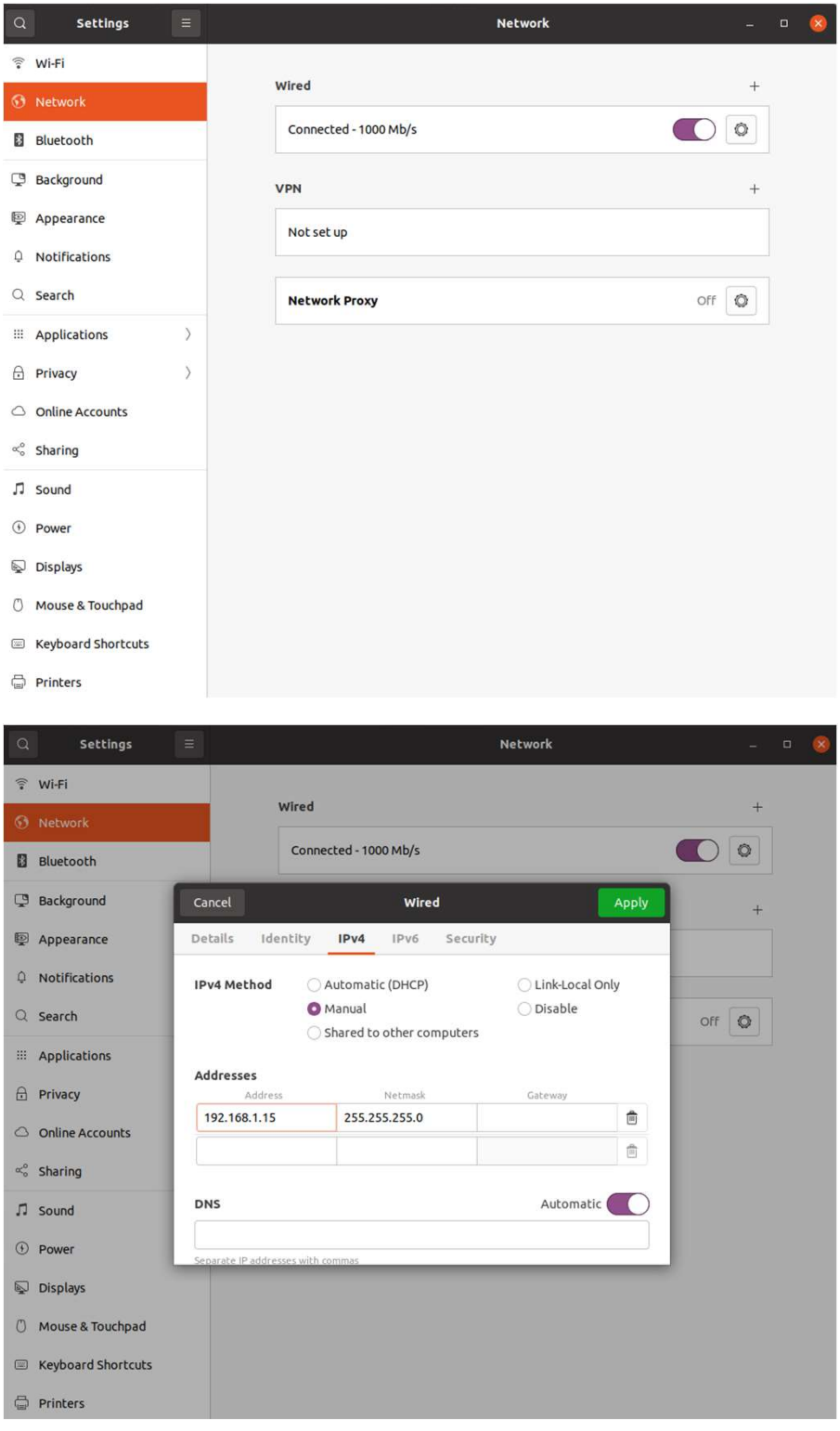


ml package to be added at ROS environment

3.1. ML-X Example Code Build for Ubuntu

Change IPv4 setting at setting Network window as below in order to connect between ML-X Device and PC.

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

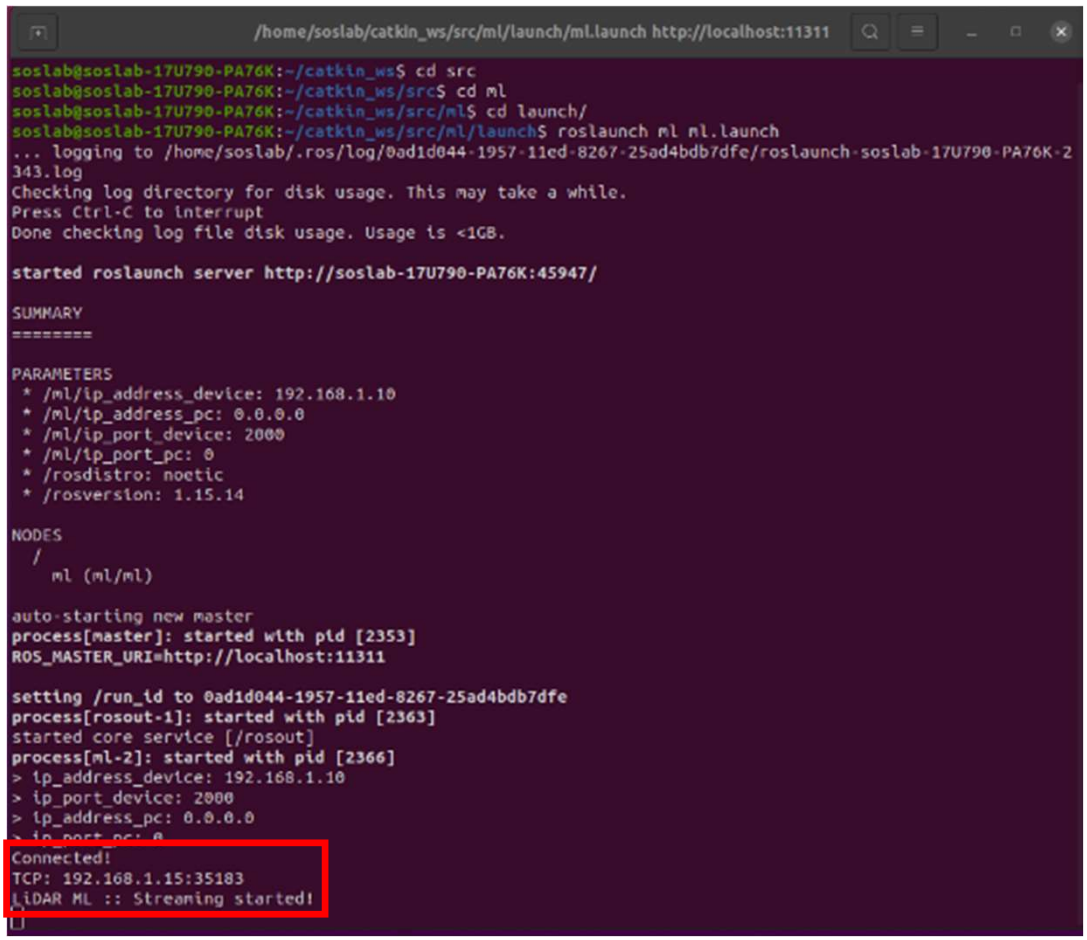


IPv4 setting change

3.1. ML-X Example Code Build for Ubuntu

4) Connect PC and ML-X device by inputting command below to run ml.launch file after Network setting. You can make sure of Lidar ML :: Streaming started out-put at terminal window as below picture if ML-X Device is properly connected.

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



Connect ML-X Device via ROS at Ubuntu environment

3.2. Example Code for Ubuntu/ROS

Description about code is as below. Example code test_ml-x consists of ML-X connection and data visualization tool at rviz

1) Raw Data aquisition

```
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 acquired code

Here is the code to acquire Raw data from ML-Device. Raw data of ML_X Device is acquired as scene_t. It consists of 1D data arry (ambient, intensity, depth, pointcloud). For more detail on scene_t structure, Please refer to Appendix

Line	Description
2	We named scene variable for SOSLAB::LidarMl::scene_t that stores Raw Data.
3	It obtains Raw data based on scene variable through get_scene fuction of SOSLAB::LidarMl Class receiving scene_t fuction as input.
4-7	ambient, intensity, depth, pointcloud data are acquired from scene_t scene structure that stores Raw data.Please find Appendix for variable of datal
8-10	It acquires height, width of Depth, Intensity image from scene_t scene structure

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_msg::ImagePtr msg_ambient;
5. cv::Mat ambient_image
6. ambient_image(scene.rows, scene.cols, CV_32SC1, ambient.data());
7. ambient_image.convertTo(ambient_image, CV_8UC1, (255.0 / 2000),0);
8. msg_ambient = cv_bridge::CvImage(std_msg::Header(), “rgb8”,
   ambient_image).toImageMsg();
9. pub_ambient.publish(msg_ambient);
```

Ambient, Depth, Intensity image Publish code at Rviz

This is the code for changing Ambient, Depth, Intensity data acquired from ML-X Device to Image and publish in order to the image visualization through Rviz at ROS environment. Above code is Ambient image transfer and publish creation code.

Line	Description
1-4	Create such as image transport, ImagePtr at ROS in order to create Publisher Node sending Message. Publisher provides ambient data as Topic “ambient”
5-6	This is the procedure to transfer to cv::Mat type at OpenCV in order to visualize the scene_t structural Ambient data to Image. Input Height(scene.rows), Width(scene.cols), Ambient data Type(CV_32SC1), Ambient Data of Raw data as initial input value.
7	Transfer from max value (2000) and Min value (0) to 8bit(uchar) of 0~255value for Image visualization.
8	Procedure to transfer to ImagePtr in order to store OpenCV의 cv::Mat in shape of Publisher Node’s Message.
9	Complete publish by using ImagePtr variable for input factorat publish fuction of Publisher appointed as “ambient_color” for topic.

Each data’s publish procedure is same as above Ambient Publish, Image transfer type is as below.

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

3.2. Example Code for Ubuntu/ROS

3) Rviz - Publish Point Cloud

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

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

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

Point cloud publish code on Rviz

Code to publish pointcloud data acquired from ML-X in order to visualize point cloud on Rviz at ROS environment

Line	Description
3-4	Create image transport, sensor_msg for ROS in order to create publisher Node sending Message Publisher provides Point Cloud via Topic "pointcloud"
5-9	Initialize Name, data size by justifying a Message Variable
10-14	Copy as msg_pointcloud variable from pointcloud data for scene_t And change to m from mm for unit of distance.
15-16	Complete publish by using for input factor from PointCloud_T::Ptr variable on publisher object's publish fuction designated by "pointcloud" for Topic after saving point cloud scanned timestamp

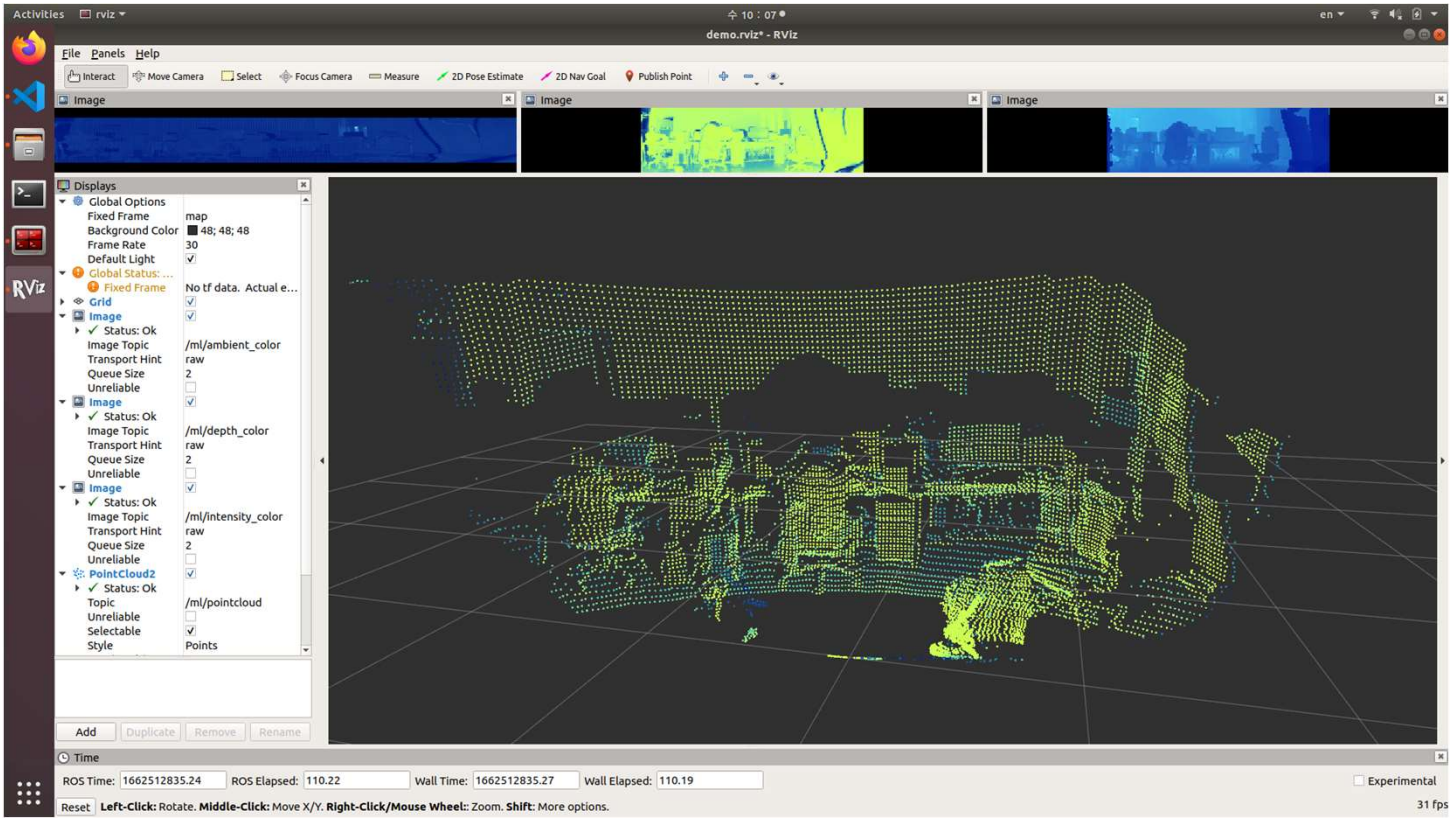
3.2. Example Code for Ubuntu/ROS

4) Rviz - Visualization

Run example code and ML-X device connection by inputting below command to visualize for Rviz

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

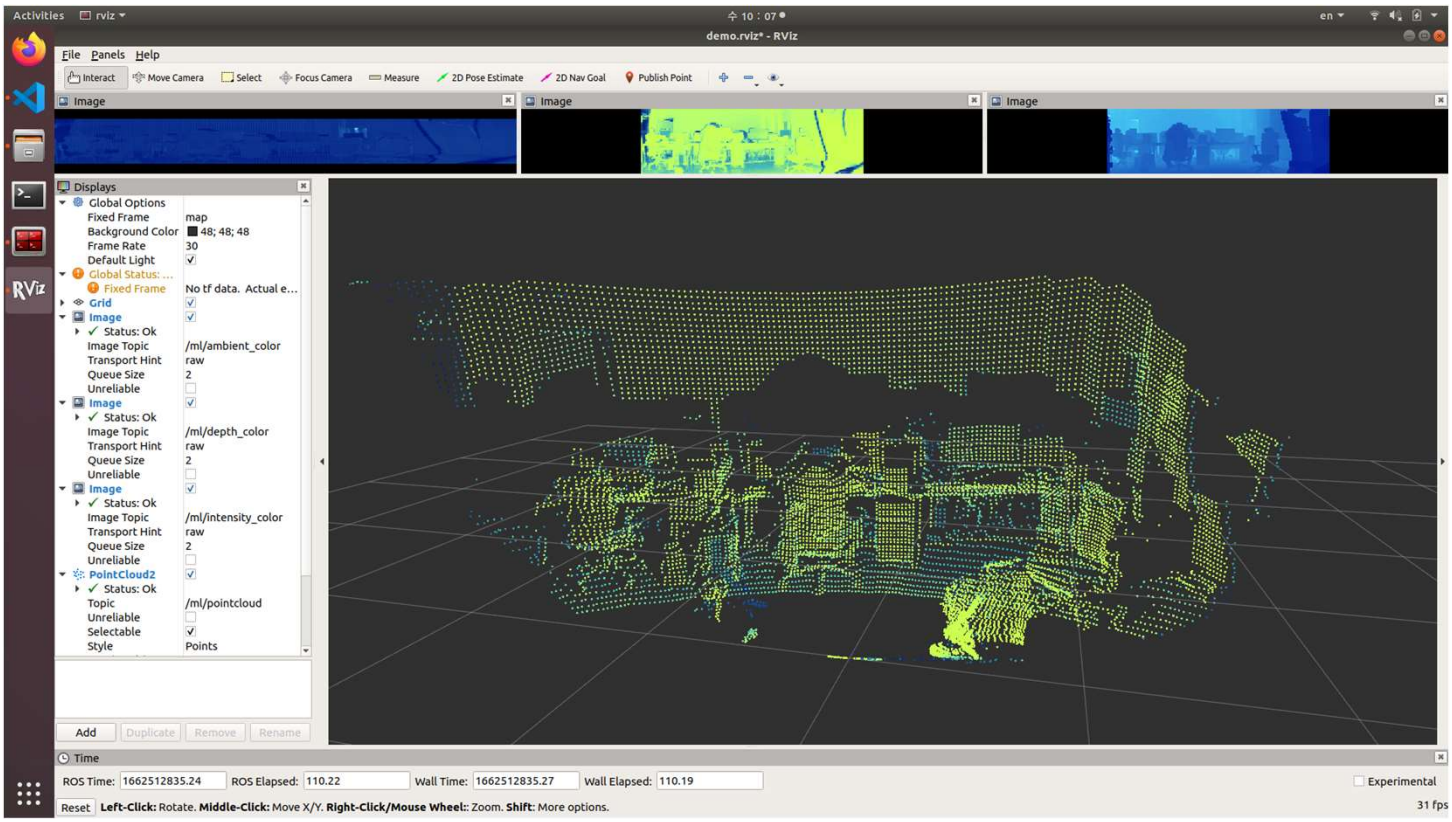
You can make sure of window as below picture by clicking add button at the botton of left side.
All topic being pub.lished can be made sure by clicking “By topic” menu on above window



Rviz Program running window

3.2. Example Code for Ubuntu/ROS

All topic image can be visualized by clicking ok botton after selecting “Image” on theTopic (/ambient, /depth, /intensity) menu to visualize the Ambient, Depth, Intensity image and point cloud data can be visualized by clicking ok botton after selecting “PointCloud2” on Topic (/pointcloud)



Rviz based ML-X data visualization (Ambient/Depth/Intensity Image, Point Cloud)

3.2. Example Code for Ubuntu/ROS

5) Rviz – Multi-LiDAR Visualization

ML-X IP setting is in order as below at Multi-LiDAR example. Different IP set-up for ML-X required to visualize multiple LiDAR for Rviz

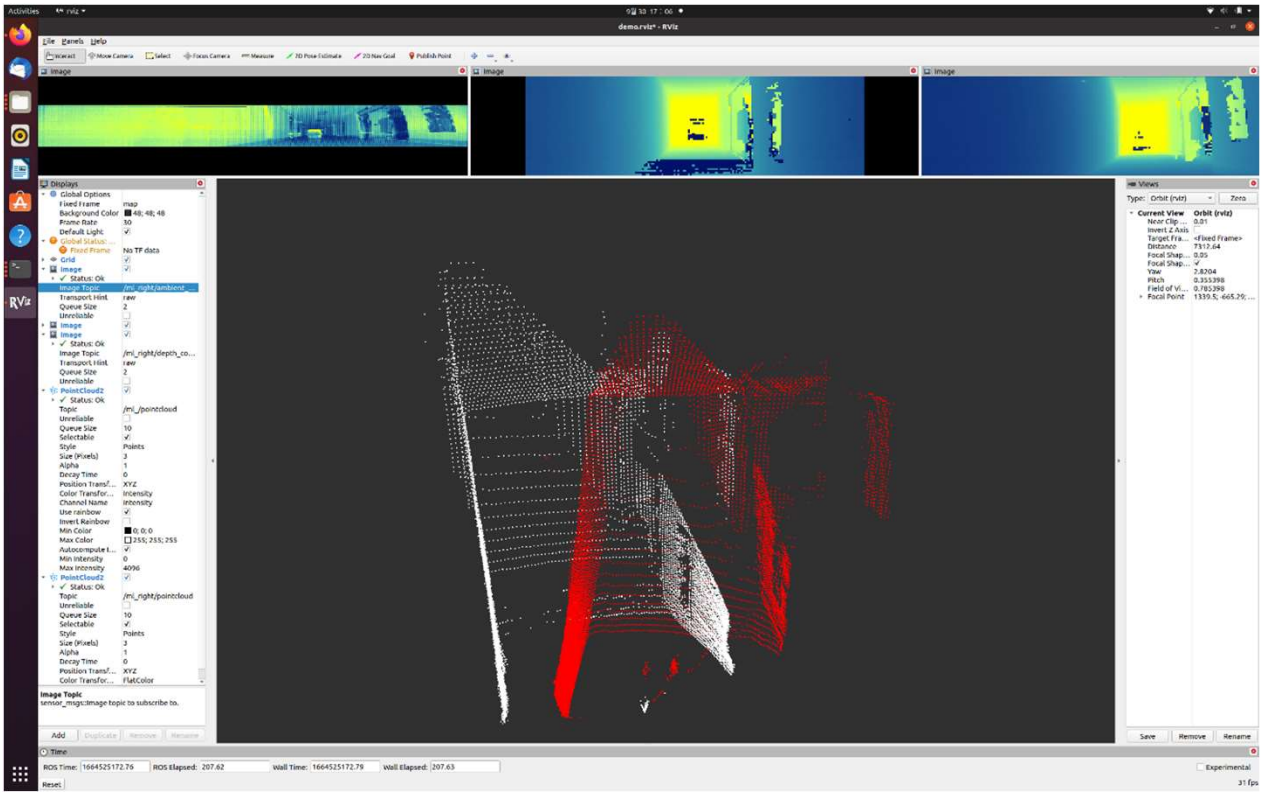
ML-X IP #1 : 192.168.1.10
ML-X IP #2 : 192.168.1.11

After IP change, input below command to run example code and ML-X devices

```
[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
```

You can make sure of window as below picture to click add button on left side below. Multi-LiDAR data can be forwarded as “ml” and “ml_second” topic by clicking “By topic” menu at certain window above.



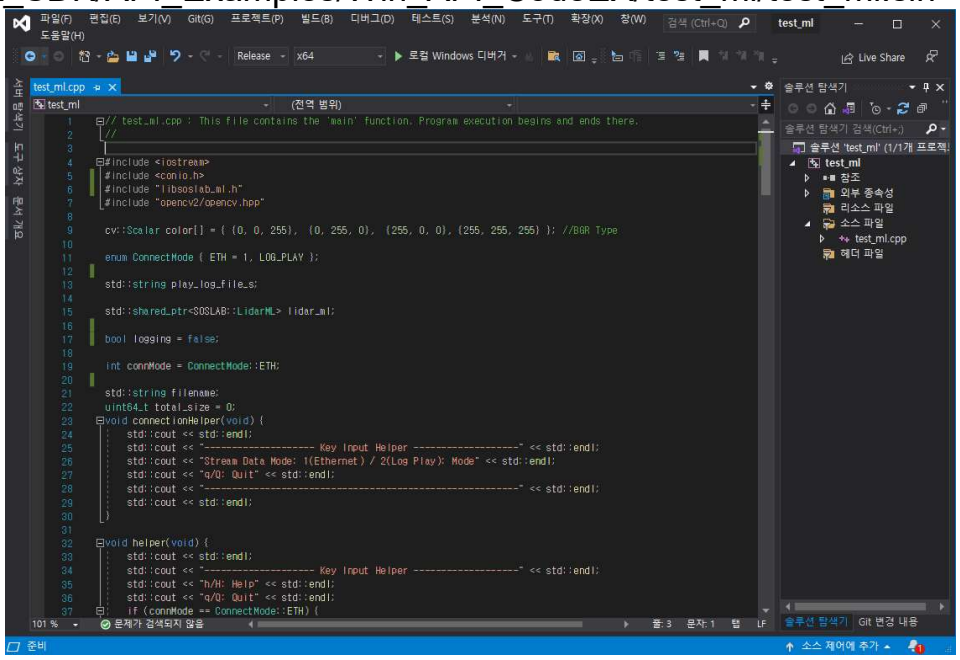
Multi-LiDAR visualization window

3.3. ML-X Example Code Build for Window

ML_X Example code at window environment is example of data visualization via ML-X API and OpenCV library.
Build Process is as follows.

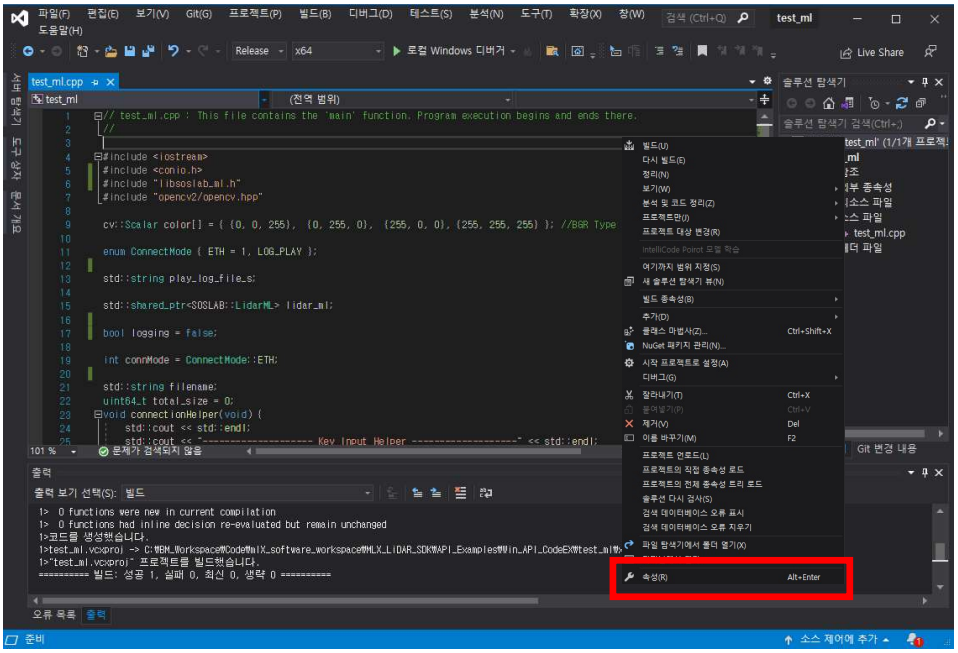
1) Run Visual Studio 2019 by clicking

MLX_LiDAR_SDK/API_Examples/Win_API_CodeEX/test_ml/test_ml.sln



test_ml.sln running screen

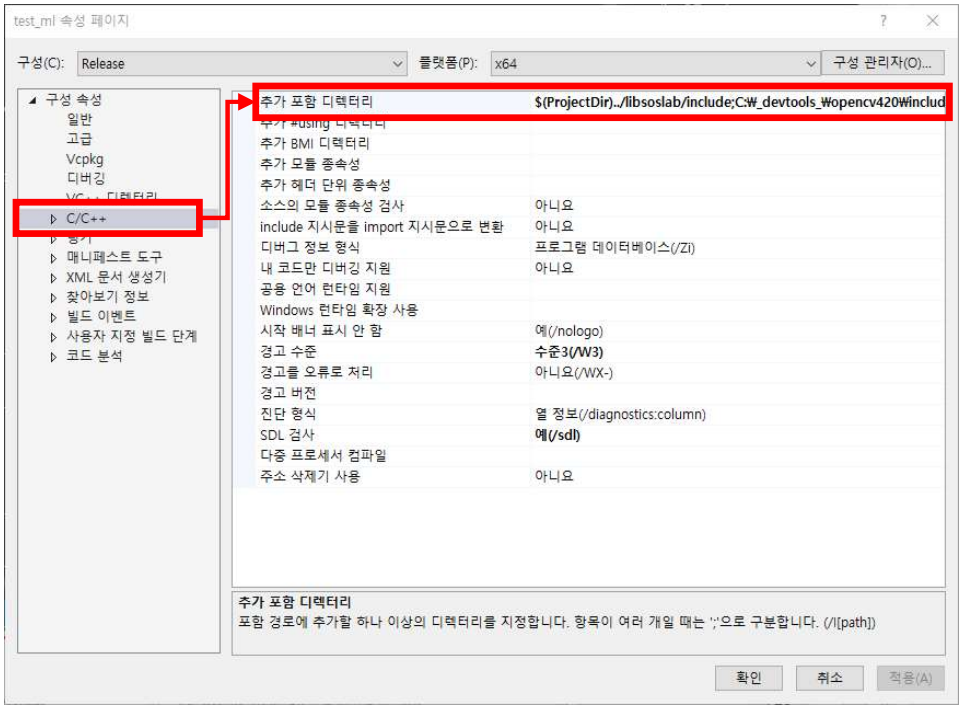
2) Input key (Alt + Enter) or click projector properties as below picture



Where projector properties is

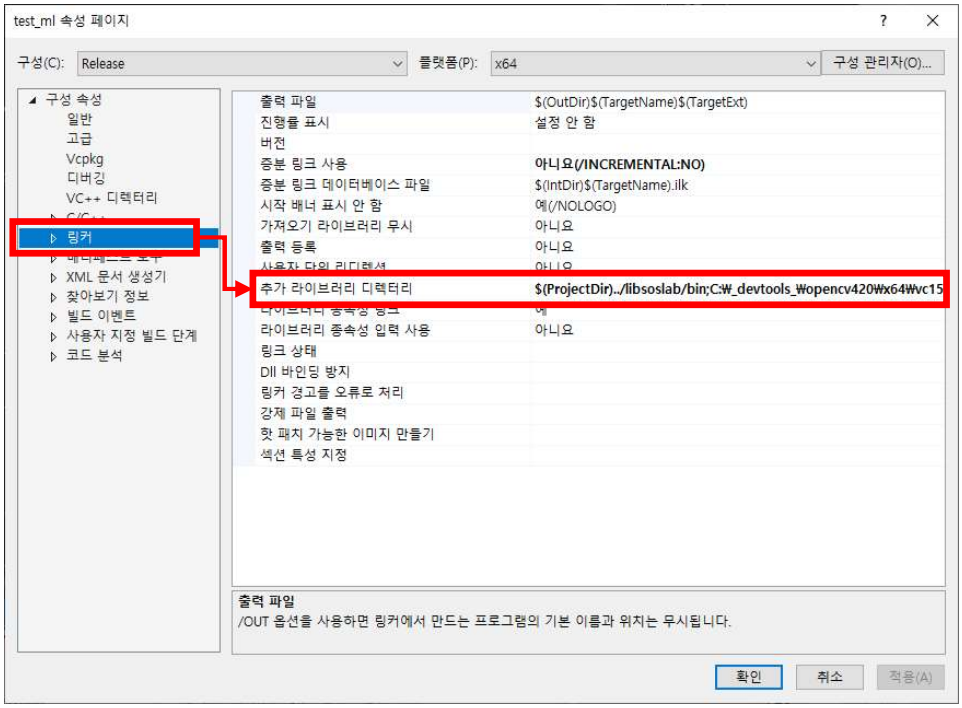
3.3. ML-X Example Code Build for Window

3)Add OpenCV library include location at [add included directory] on the list of [C/C++] as below picture



Add included directory at properties window for running test_ml

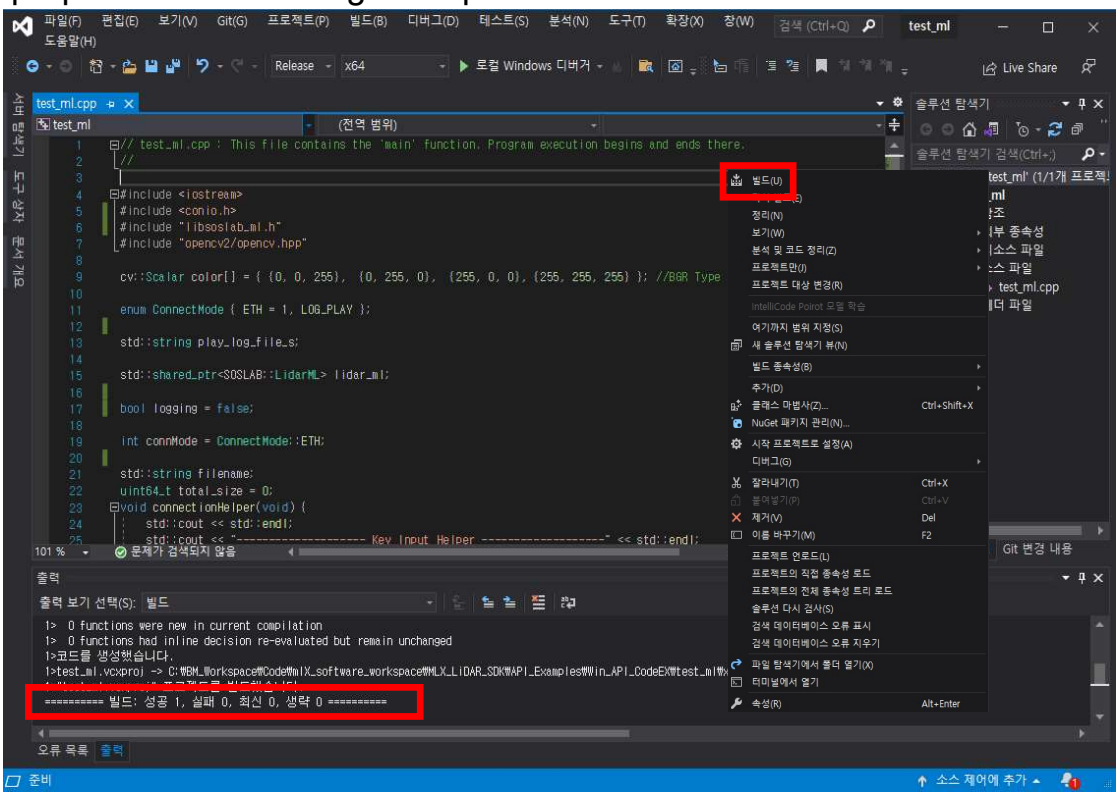
4)Add OpenCV library include location at [add library directory] on the list of [Linker] as below picture



Add library directory at properties window for running test_ml

3.3. ML-X Example Code Build for Window

5) test_ml.exe running file will be created at MLX_LiDAR_SDK / API_Examples / Win_API_CodeEX / Example folder by building
test_ml example code as long as **“Success 1, failure 0”** is output at out-put window after changing properties and building example code.

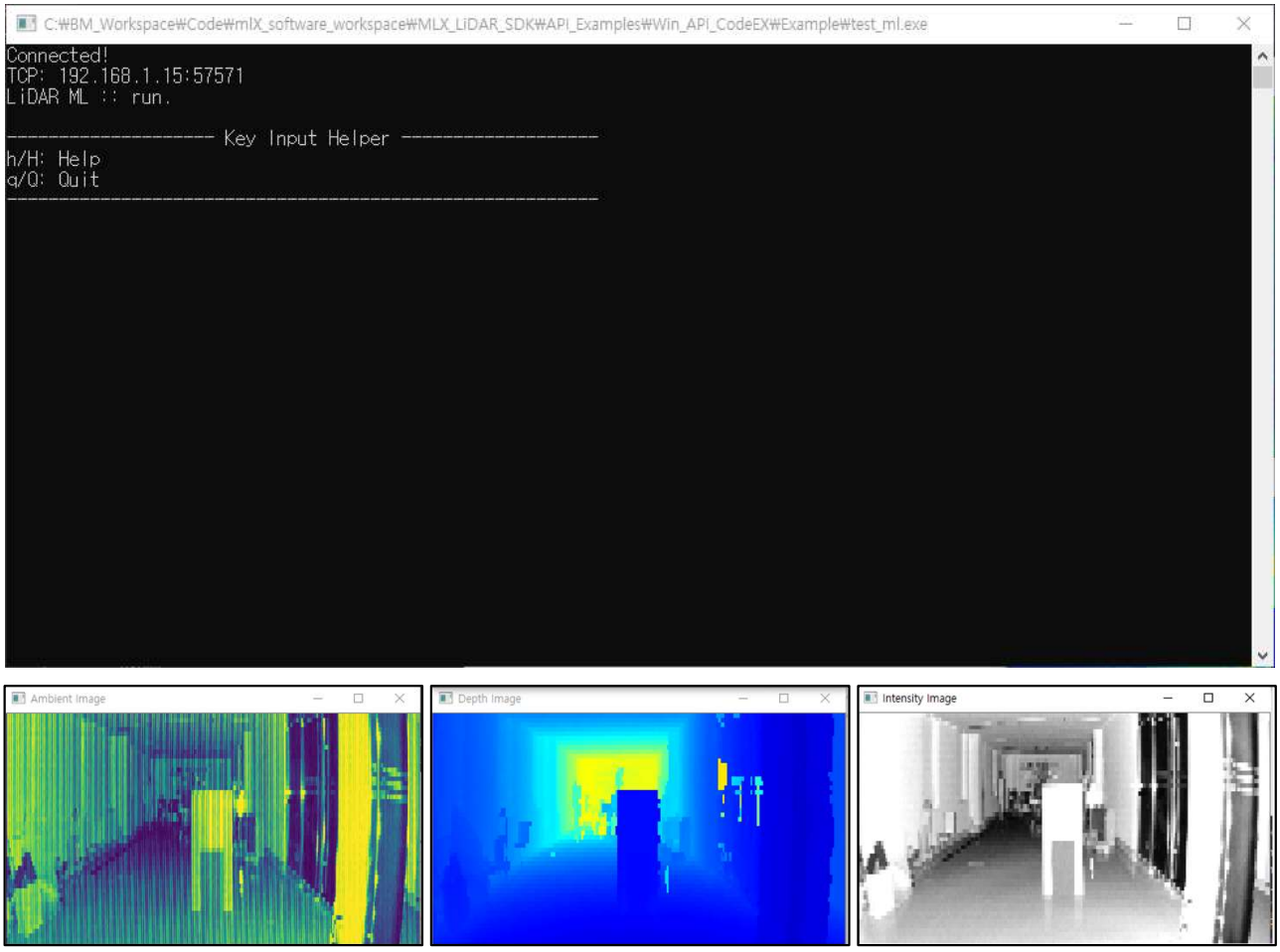


test_ml build result

3.3. ML-X Example Code Build for Window

6)Ambient / Depth / Intensity image can be visualized as below picture after running test_ml.exe

.



test_ml running result : terminal (above picture),
Ambient / Depth / Intensity Image visualization (below picture)

3.4. Example Code for Window

ML-X connection and Data visualization on Rviz tool is set-up at example code test_ml..
Explanation on code is as follows.

1) Raw Data acquire

```
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 acquired code

Here is the code how to acquire Raw data from ML-X device. Raw data for ML-X device is acquired as 는 scene_t,
It consists of Total 4 kind of 1D data array (ambient, intensity, depth, pointcloud). For more details on scene_t structure, Please see appendix.

Line	Description
2	Justifying to scene variable name from SOSLAB::LidarMl::scene_t variable where saves Raw data.
3	Raw data can be acquired at scene variables via SOSLAB::LidarMl Class's get_scene functions receiving scene_t variables as input.
4-7	ambient, intensity, depth, pointcloud data are acquired from scene_t scene structure where Raw Data is saved. Each data's structual variables name and shape is described on Appendix.
8-9	Image's height, width value is acquired from scene_t scene structure.

3.4. Example Code for Window

2) Ambient, Depth, Intensity Image Transfer

```
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);
```

Transfer from Raw data to Ambient Image

Code that performs visualization and tranfer Ambient, Depth, Intensity data acquired from ML-X device to Image in order to Image visualization.
Above code is Ambient Image transfer and visualization code.

Line	Description
1	This describes how to transfer to OpenCV의 cv::Mat format in oder to visualize the image from scene_t structural Ambient data. Input Raw data's Height(scene.rows), Width(scene.cols), Ambient data Type(CV_32SC1), Ambient Data value as cv::Mat initialization input value.
3	Transfer from Max value(2000) and Min value(0) to 0~255 value's 8bit (uchar) for Image visualization.
4	Create the color of cv::Mat object via OpenCV's applyColorMap functions. VIRIDIS colormap are used at this example.
5	Ambient image to be visualized by cv::imshow functions.

Transfer process of each data is same as above ambient procedure and Image transfer type is as below.

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

Appendix

Smart Optical Sensors

A.1. ML-X API - Typedefs

Typedefs

Name	SOSLAB::INT_T
Justify	typedef int32_t SOSLAB::INT_T
Header	#include "soslab_typedef.h"
description	integer variables

Name	SOSLAB::UINT_T
Justify	typedef uint32_t SOSLAB::UINT_T
Header	#include "soslab_typedef.h"
description	unsigned int Variables type

Name	SOSLAB::FLOAT_T
Justify	typedef double SOSLAB::FLOAT_T
Header	#include"soslab_typedef.h"
description	double Variables type

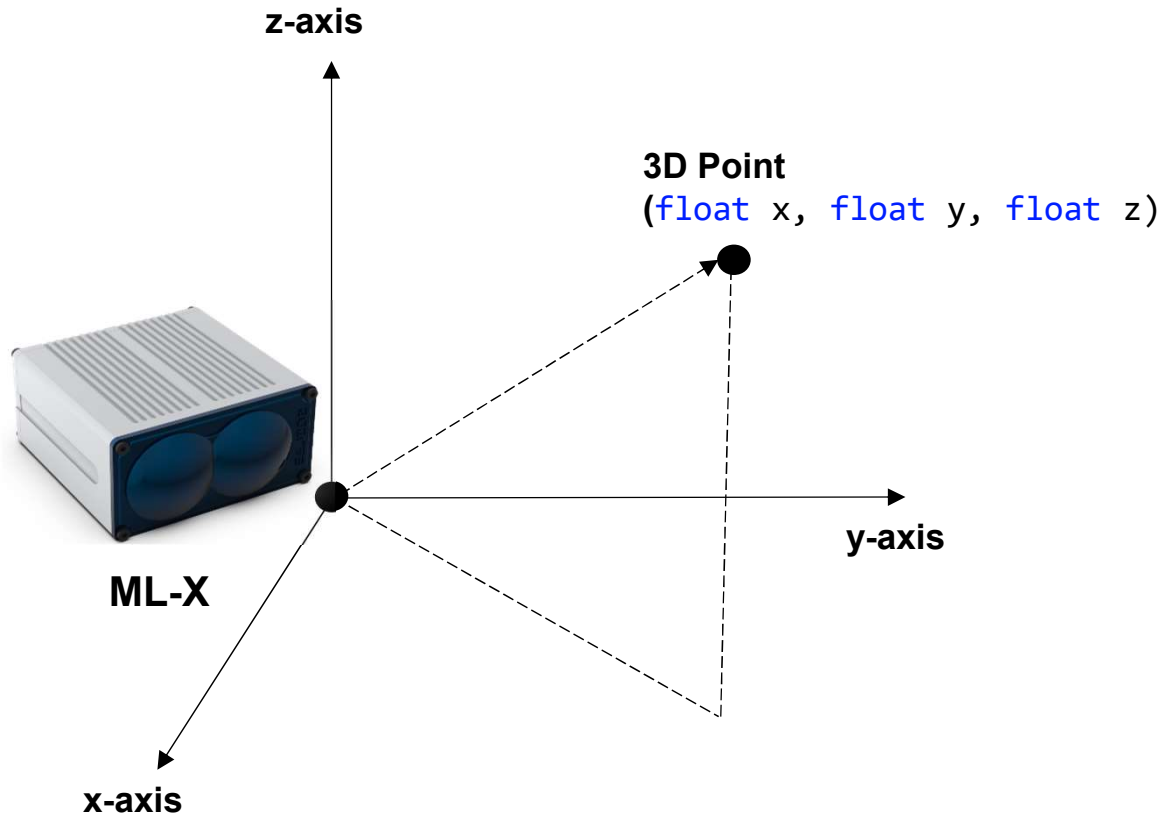
Name	SOSLAB::hex_array_t
Justify	typedef std::vector<uint8_t> SOSLAB::hex_array_t
Header	#include "soslab_typedef.h"
description	16진수 배열 변수형

Name	SOSLAB::pointcloud_t
Justify	typedef std::vector<point_t> SOSLAB::pointcloud_t
Header	#include "soslab_typedef.h"
description	Point Cloud Variable type

A.2. ML-X API - Classes

Classes

Variable type	SOSLAB::point_t
Justification	typedef struct _POINT_T point_t
Header	#include "soslab_typedef.h"
Description	oordinate System structural object covering 3D point
Member Variables	**Member Variables explanation**
float x	Point의 x coordinates information (unit : mm)
float y	Point의 y coordinates information (unit : mm)
float z	Point의 z coordinates information (unit : mm)



3D Point Cartesian Coordinate System

A.2. ML-X API - Classes

Classes

Variables type	SOSLAB::ip_setting_t
Justification	typedef struct IP_ADRESS_T ip_setting_t
Header	#include "soslab_typedef.h"
Description	Structural object to save IP and Port address
Member Variables	**Member Variables description**
std::string ip_address	IP address
int port_number	Port number

A.2. ML-X API - Classes

Classes

Variables type	SOSLAB::LidarMl::scene_t
Justification	typedef struct _ML_SCENE_T point_t
Header	#include “ml/libsoslab_ml.h"
Description	Structure of Raw Data
Member Variables	**Member Variables description**
std::vector<uint32_t> timestamp	Raw Data acquired time [size : rows]
uint8_t frame_id	Frame Index [Max : 255]
uint16_t rows	Raw Data Height Value
uint16_t cols	Raw Data Width Value
std::vector<uint32_t> ambient_image	Ambient Data array (Ambient : all signal strength evaluated)
std::vector<std::vector<uint32_t>> depth_image	Multi-Echo Depth Data array [1st vector size : # echo, 2nd vector size : rows x cols] (Depth : Range data measured)
std::vector<std::vector<uint16_t>> intensity_image	Multi-Echo Multi-Echo Intensity Data array [1st vector size : # echo, 2nd vector size : rows x cols] (Intensity : LiDAR signal strength measured)
std::vector<std::vector<pointcloud_t>> pointcloud	Multi-Echo Point cloud data array [1st vector size : # echo, 2nd vector size : rows x cols] (Point cloud : 3D point의 array data)

A.2. ML-X API - Classes

Classes

Variables type	SOSLAB::LidarMl
Justification	class LidarMl
Header	#include “ml/libsoslab_ml.h"
Description	ML-X Device Connection 및 Signal/Data Processing

Functions

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

- Description : functions of Local(PC) and ML-X Device connection
- Input : Local(PC) and ML-X Device의 IP and Port address
- Output : Transfer to bool variables type of connection status(Transfer to True when connected)

```
bool disconnect();
```

- Description : function to disconnect Local(PC) and ML-X Device
- Output :Transfer to bool variables type of dis-connection status(Transfer to True when connected)

```
bool run();
```

- Description : Function to run ML-X Device
- Output :Transfer to bool variables run status(Transfer to True when connected)

```
bool stop();
```

- Description : Function to stop run ML-X Device
- Output : Transfer to bool variables stop operation status(Transfer to True when connected)

```
bool get_scene(scene_t& scene);
```

- Description : Function to acquire Raw data with Input variables scene
- Input : scene_t variables type
- Output : scene_t variables to save ML-X Device Raw Data

A.2. ML-X API - Classes

Classes

Variables type	SOSLAB::LidarMl
Justification	class LidarMl
Header	#include “ml/libsoslab_ml.h"
Description	ML-X Play Mode
Functions	
void record_start(std::string filepath)	
• Description : Function to start recording ML-X data • Input : Save location for recorded file(.pcap, .bin)	
void record_stop()	
• Description : Function to stop recording ML-X data	
void play_start(const std::string filepath)	
• Description : Function to replay recorded ML_X data. • Input : location to save recorded file(.pcap, .bin)	
void play_stop()	
• Description : Fuction to stop replay ML-X data	
bool get_scene(scene_t& scene, uint64_t index);	
• Description : Function to acquire data of certain frame via input variable scene and frame number after recall recorded file. • Input (1): Function to save Raw Data (scene_t& scene) • Input (2): Frame (uint64_t index) • Output : Scene_t Variables to save ML-X Device Raw data	
void get_file_size(uint64_t& size)	
• Description : Function to transfer total Frame # after recall the recorded file • Output : Total Frame #	

A.3. UDP Protocol Packet Structure

A.3.1 Normal Packet Structure (192 pixels)

UDP communication standard packet structure is as below.

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 standard 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
2336~2339 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.3.2 Depth Completion Packet Structure (576 pixels)

UDP communication's Depth completion packet structure is as below.

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 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
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. TCP Protocol Packet Structure

A.4.1 TCP Protocol Packet Structure

TCP communication consists of JSON Format. It consists of answering mechanism from ML when to transmit an order set by PC. JSON structure is as table below.

TCP Protocol packet structure	
JSON Format	**Description**
`{ "command": "run" }`	Device Run
`{ "command": "stop" }`	Device Stop