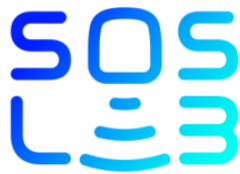




*Solid-State LiDAR **ML-X***



Software User Guide



Release v1.0

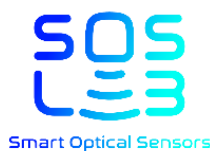
2022-08-15

Table of Contents

1. ML-X API (Ubuntu/ROS)	03
1.1. ML-X API Build for Ubuntu/ROS	04
1.2. Example Code for Ubuntu/ROS	07
2. ML-X Software: SOS Studio	12
2.1. Connection	13
2.2. Data Load	17
2.3. Device IP Setting	18
2.4. Point Cloud Setting	20
2.5. Image Setting	21
2.6. Grid	22
2.7. X-Y / Y-Z / Z-X Axis	23
2.8. Play / Record Mode	24
Appendix	25
A.1. ML-X API – Typedefs	26
A.2. ML-X API – Classes	27

Chapter 1

ML-X LiDAR API (Ubuntu/ROS)



1.1. ML-X API Build for Ubuntu/ROS

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

ML-X API는 libsoslab_core, libsoslab_ml 2가지로 구성되며, test_ml-x 예제 코드를 통해 ML-X 연결 및 데이터 가시화를 테스트할 수 있습니다.

1.1. ML-X API Build for Ubuntu/ROS

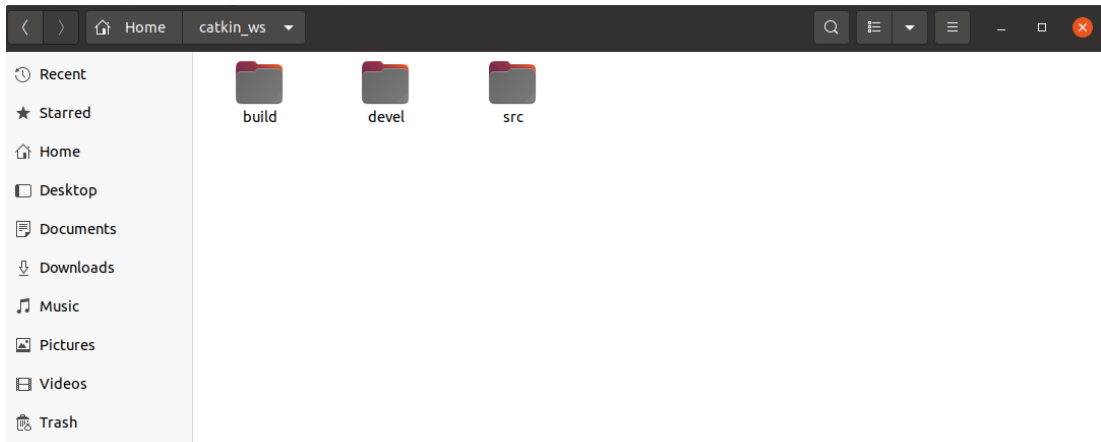
ML-X API 빌드는 ROS 설치가 완료 된 후, 아래 과정들을 통해 진행할 수 있습니다.

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

```
$ catkin_make
```

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

catkin_make를 활용한 ml Package 생성



ml Package 빌드 결과

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

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

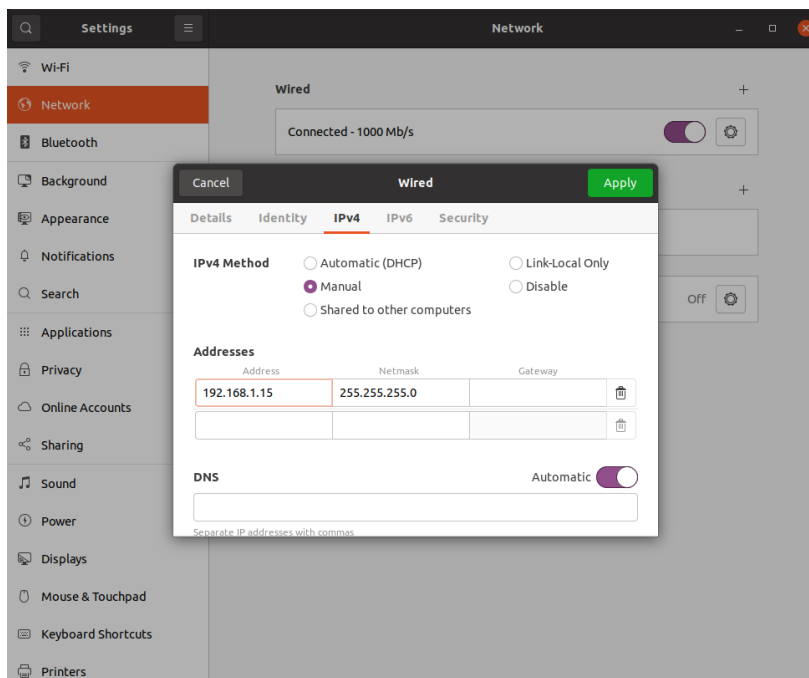
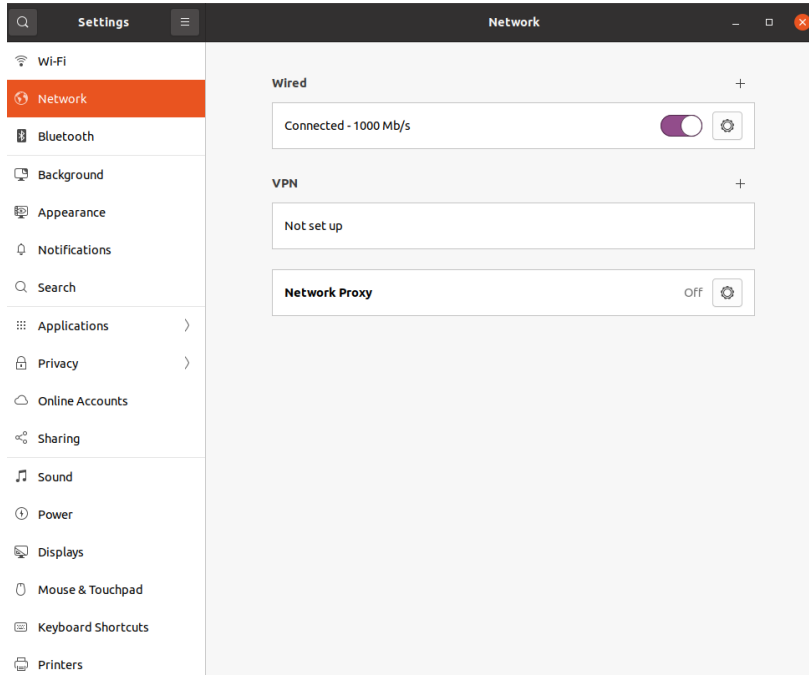
```
soslab@soslab-17U790-PA76K: ~/catkin_ws
soslab@soslab-17U790-PA76K:~/catkin_ws$ source ~/catkin_ws/devel/setup.sh
soslab@soslab-17U790-PA76K:~/catkin_ws$ rospack find ml
/home/soslab/catkin_ws/src/ml
soslab@soslab-17U790-PA76K:~/catkin_ws$
```

ml package를 ROS 환경에 추가

1.1. ML-X API Build for Ubuntu/ROS

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

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

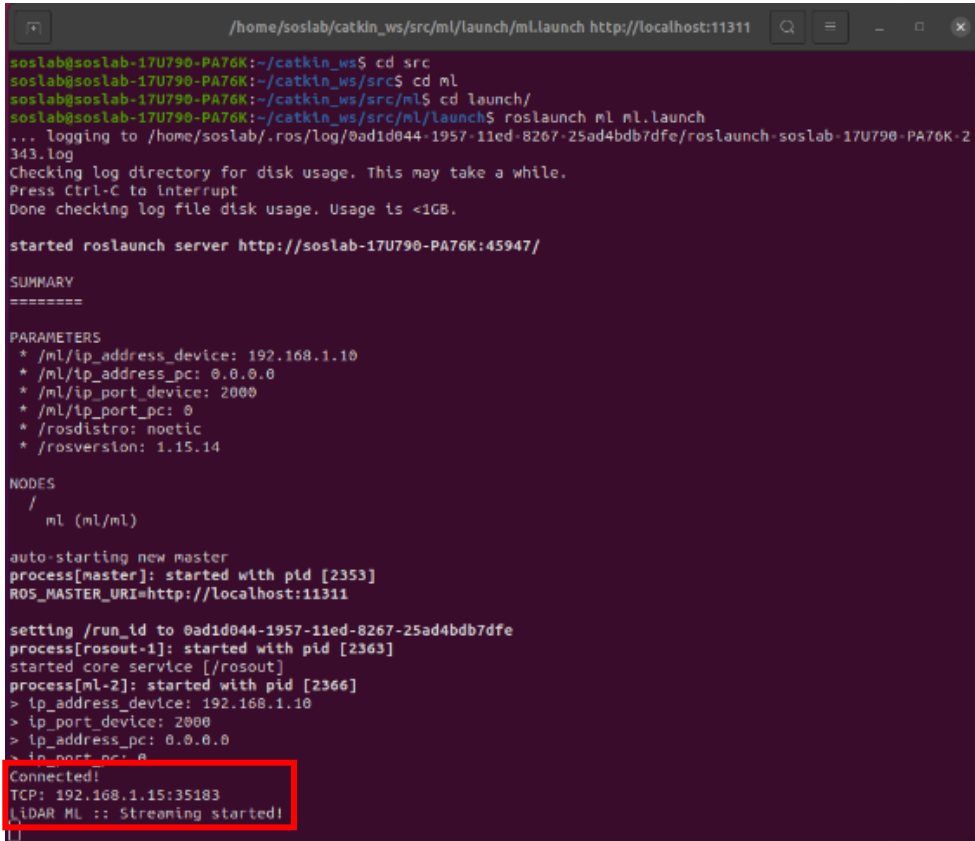


IPv4 설정 변경

1.1. ML-X API Build for Ubuntu/ROS

- 4) Network 설정 후, 아래 command를 입력하여 ml.launch 파일을 실행하여 PC와 ML-X device를 연결 합니다. 정상적으로 ML-X Device와 연결되면 아래 그림과 같이 Terminal 창에 Lidar ML :: Streaming started! 출력을 확인할 수 있습니다.

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



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

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

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

NODES
/
  ml (ml/ml)

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

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

Ubuntu 환경에서 ROS를 통해 ML-X Device 연결

1.2. Example Code for Ubuntu/ROS

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

1) Raw Data 획득

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

Raw Data 획득 코드

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

Line	Description
2	Raw Data를 저장할 SOSLAB::LidarML::scene_t 변수를 scene 변수명으로 선언합니다.
3	scene_t 변수를 입력으로 받는 SOSLAB::LidarML Class의 get_scene 함수를 통해 scene 변수에 Raw Data를 획득합니다.
4-7	Raw Data를 저장한 scene_t scene 구조체로부터 ambient, intensity, depth, pointcloud 데이터를 획득합니다. 각 데이터의 구조체 변수명 및 변수형은 Appendix에 기재되어 있습니다.
8-9	scene_t scene 구조체로부터 이미지의 height, width 값을 획득합니다.

1.2. Example Code for Ubuntu/ROS

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

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

Rviz에서의 Ambient, Depth, Intensity 이미지 Publish 코드

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

Line	Description
1-4	Message를 보내는 Publisher Node를 생성하기 위하여 ROS의 image transport , ImagePtr 등의 객체를 생성합니다. Publisher는 Topic "ambient" 로 ambient 데이터를 제공합니다.
5-6	scene_t 구조체의 Ambient 데이터를 이미지로 가시화하기 위해 OpenCV의 cv::Mat 형식으로 변환하는 과정입니다. cv::Mat 초기화 입력 값으로 Raw data의 Height(scene.rows), Width(scene.cols), Ambient 데이터 Type(CV_16UC1), Ambient Data 값을 입력합니다.
7	이미지 가시화를 위하여 최대 값(2000)과 최소 값(0)을 0~255 값의 8bit(uchar) 형태로 변환합니다.
8	OpenCV의 cv::Mat 객체를 Publisher Node의 Message 형태로 저장하기 위해 ImagePtr 로 변환하는 과정입니다.
9	Topic이 "ambient_color"로 지정된 Publisher 객체의 publish 함수에 ImagePtr 변수를 입력 인수로 사용하여 Publish를 완료합니다.

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

- Ambient : CV_32SC1
- Depth : CV_32FC1
- Intensity : CV_16UC1

1.2. 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 = scene.cols;
8. msg_pointcloud->height = scene.rows;
9. msg_pointcloud->points.resize(scene.pointcloud.size())
10. for (int i = 0; i < scene.pointcloud.size(); i++) {
11.     msg_pointcloud->points[i].x = scene.pointcloud[i].xyz.x/1000.0;
12.     msg_pointcloud->points[i].y = scene.pointcloud[i].xyz.y/1000.0;
13.     msg_pointcloud->points[i].z = scene.pointcloud[i].xyz.z/1000.0;
14. }
15. pcl_conversions::toPCL(ros::Time::now(), msg_pointcloud-
    >header.stamp);
16. pub_lidar.publish(msg_pointcloud);
```

Rviz에서의 Point Cloud Publish 코드

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

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

1.2. Example Code for Ubuntu/ROS

4) Rviz - Visualization

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

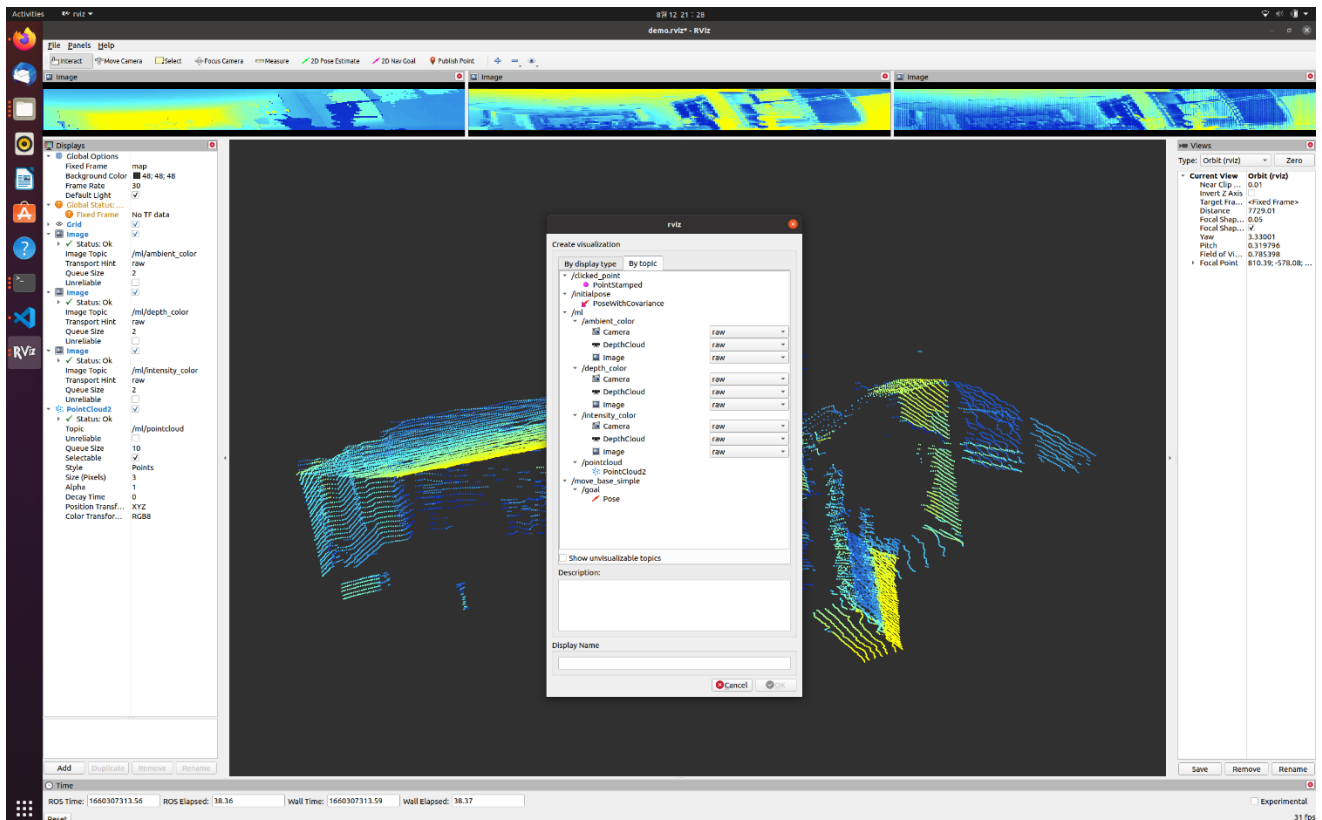
[Terminal #1]

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

[Terminal #2]

```
$ rviz
```

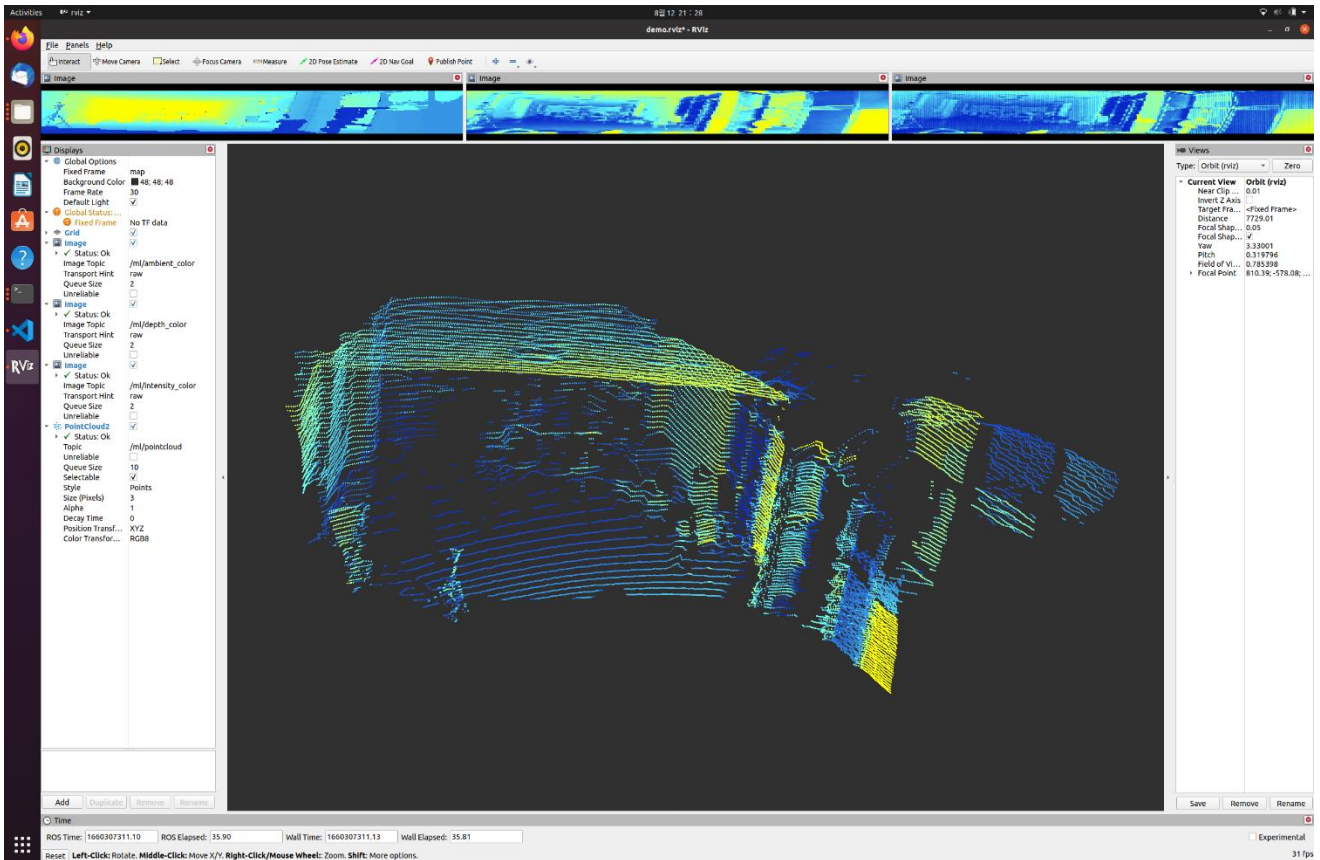
아래 그림과 같이 왼쪽 하단의 Add 버튼을 클릭하면 아래 그림과 같은 창을 확인할 수 있습니다. 해당 창의 상단 “By topic” 메뉴를 클릭하면 publish 되고 있는 전체 topic을 확인할 수 있습니다.



Rviz 프로그램 실행 화면

1.2. Example Code for Ubuntu/ROS

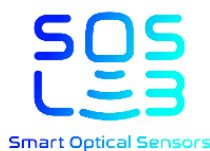
Ambient, Depth, Intensity 이미지를 가시화하기 위해서 Topic (/ambient, /depth, /intensity) 메뉴의 “Image”를 선택한 뒤 OK 버튼을 클릭하면 각 토픽에 대한 이미지를 가시화할 수 있으며, Point Cloud 데이터는 Topic (/pointcloud)의 “PointCloud2”를 선택한 뒤 OK 버튼을 클릭하면 가시화할 수 있습니다.



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

Chapter 2

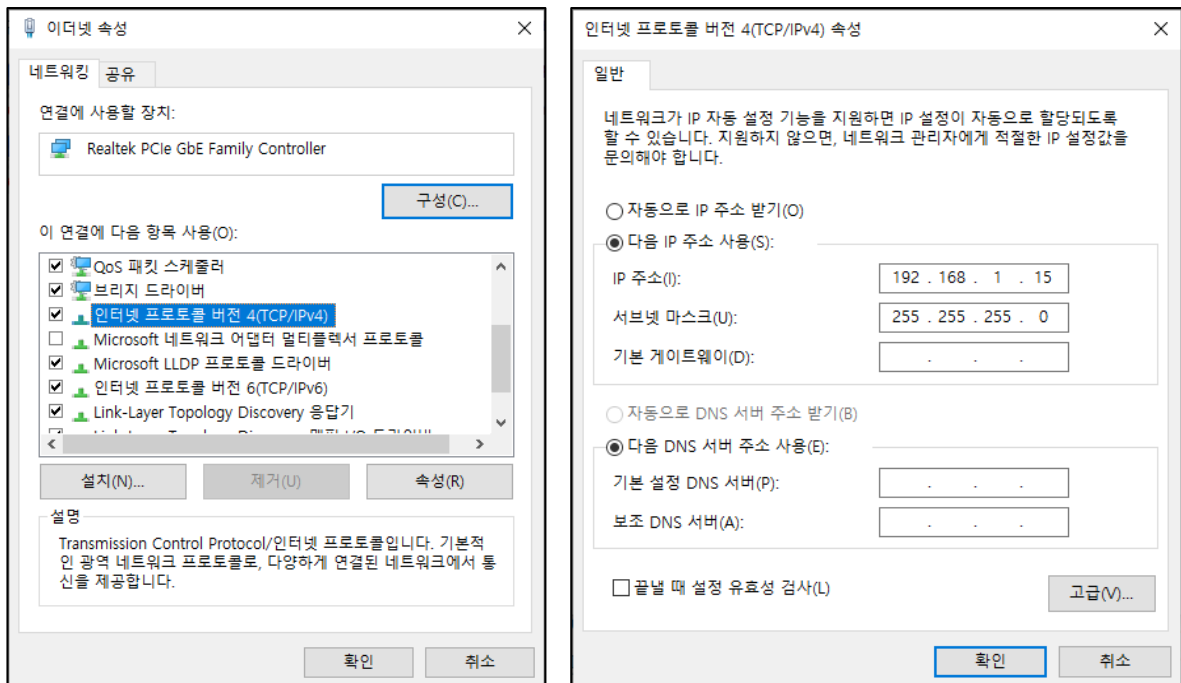
ML-X Software: SOS Studio



2.1 Connection

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

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

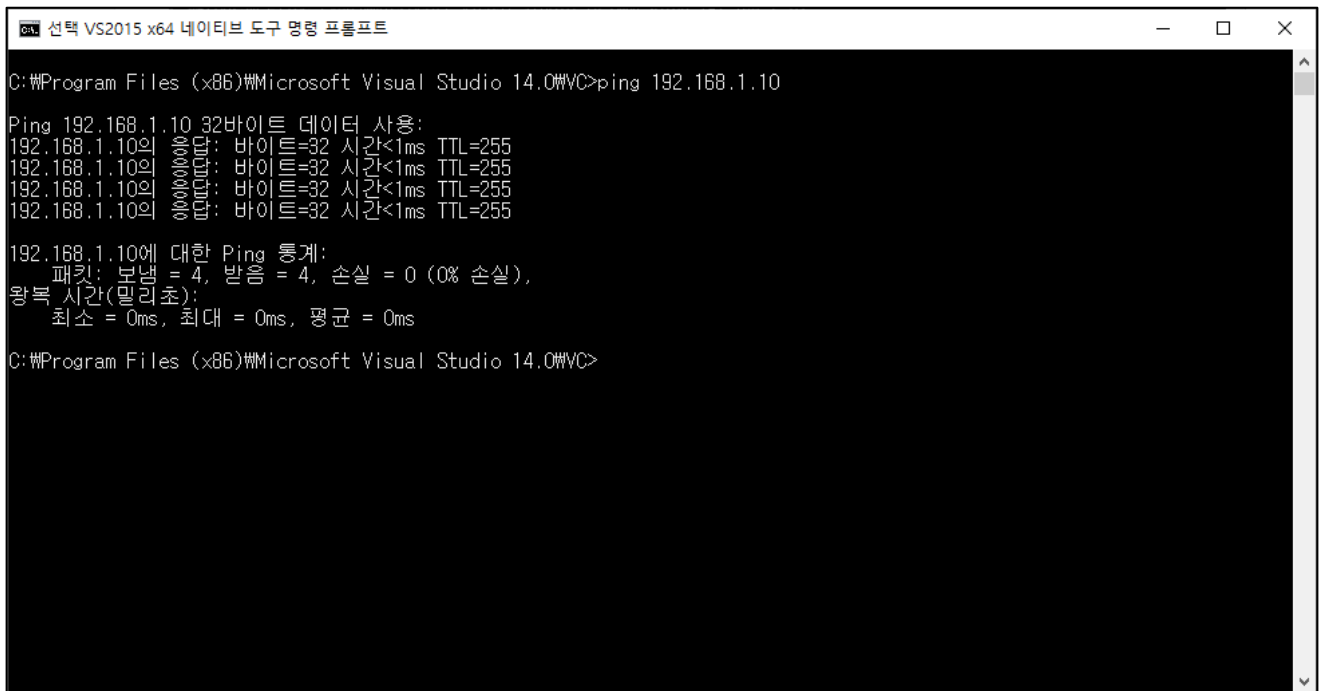


이더넷 속성 창 / 인터넷 프로토콜 버전4(TCP/IPv4) 속성 창 설정

2.1 SOS Studio – Connection

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

- 1) 윈도우 검색창에 'cmd' 명령어를 입력하여 명령 프롬프트를 실행합니다.
- 2) 명령어에 '**ping 192.168.1.10 -t**'를 입력합니다.
- 3) ML-X LiDAR와 PC가 정상적으로 연결되었다면 아래와 같은 메시지가 출력됩니다.



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

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

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

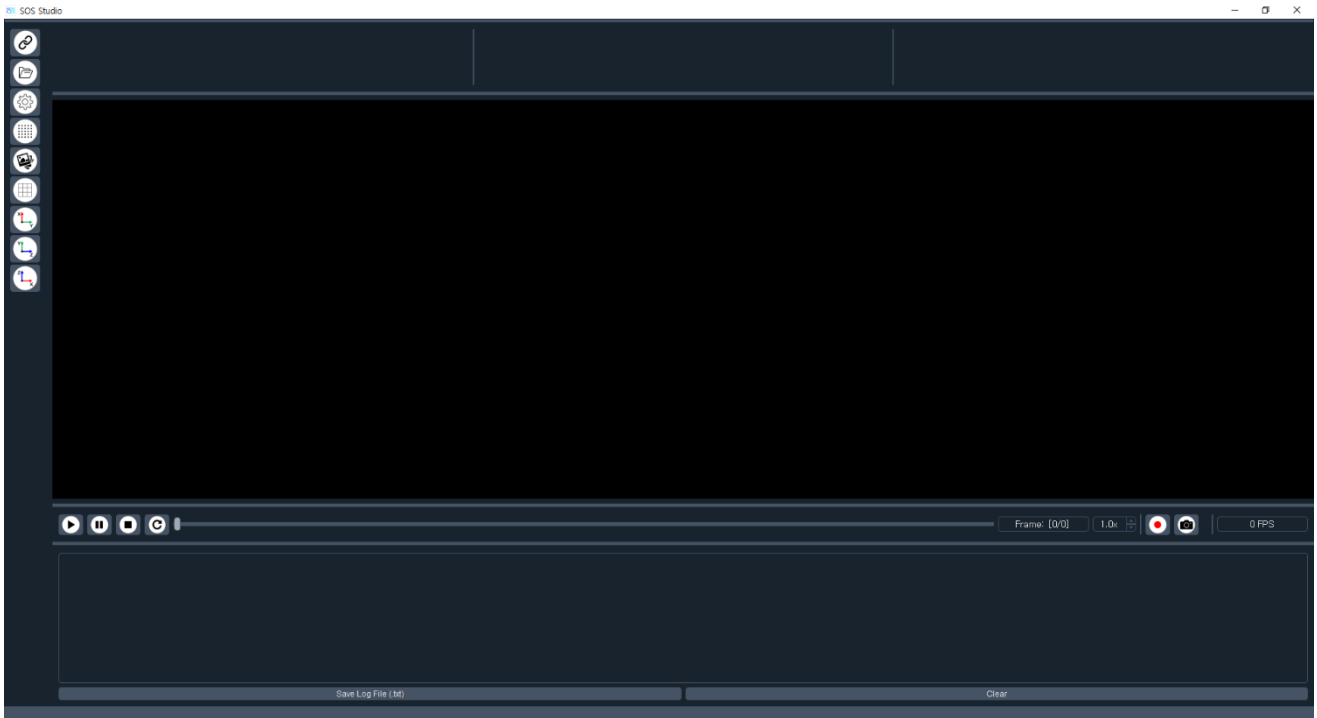
C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>
```

명령 프롬프트 창 및 ping 테스트 성공 결과 예시

위 과정이 성공적으로 완료되면,  SOS Studio ML-X.exe 프로그램을 실행합니다.

2.1 SOS Studio – Connection

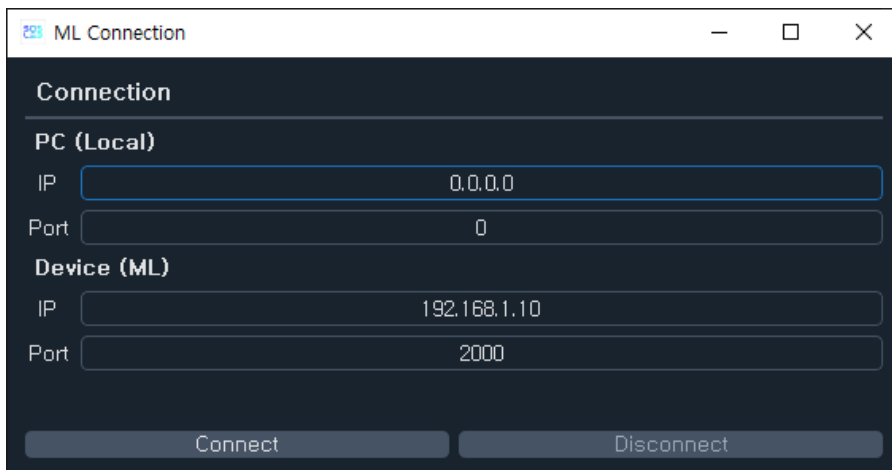
아래는 SOS Studio 실행화면 입니다.



SOS Studio 실행화면

PC와 ML-X의 연결을 위해 SOS Studio 왼쪽 1번째 버튼 “Connection”을 클릭하면 아래와 같은 팝업 창이 나타납니다. 이 후, 아래와 같이 정보를 입력한 뒤, “Connect” 버튼을 클릭하면 PC와 ML-X Device를 연결할 수 있습니다.

(※ ML Device에 대한 IP가 변경되었을 경우, 변경된 IP로 입력합니다.)



Connection 팝업 창

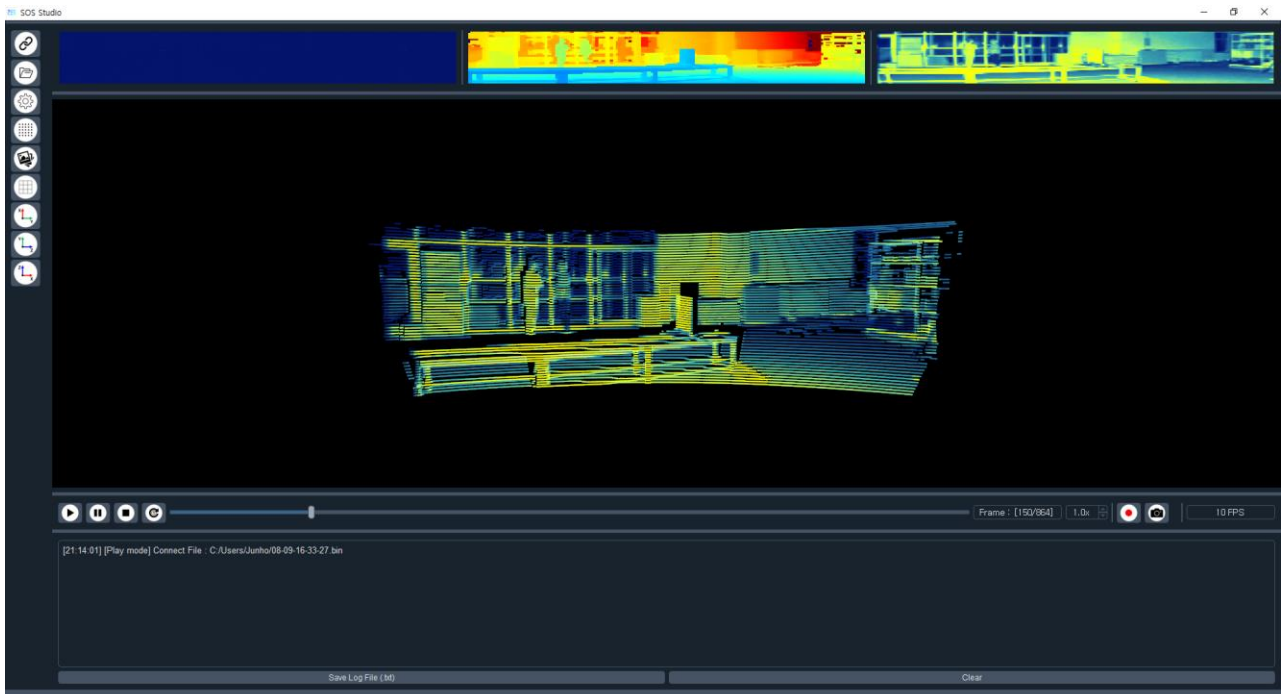
2.1 SOS Studio – Connection

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



Window 보안 경고 화면

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

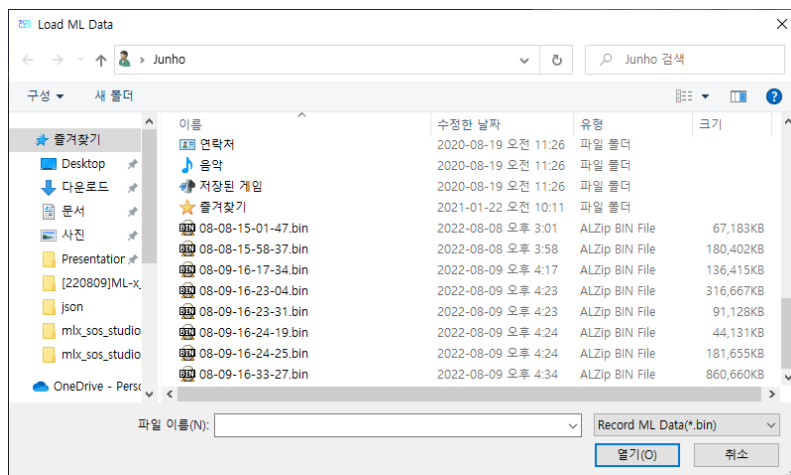


SOS Studio 실행화면

2.2 Data Load

Data Load 기능은 저장된 ML-X 데이터를 불러올 때 사용하는 기능입니다. 데이터 저장 및 불러온 데이터를 재생하는 기능은 2.8 Play / Record Mode에 기술되어 있습니다.

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

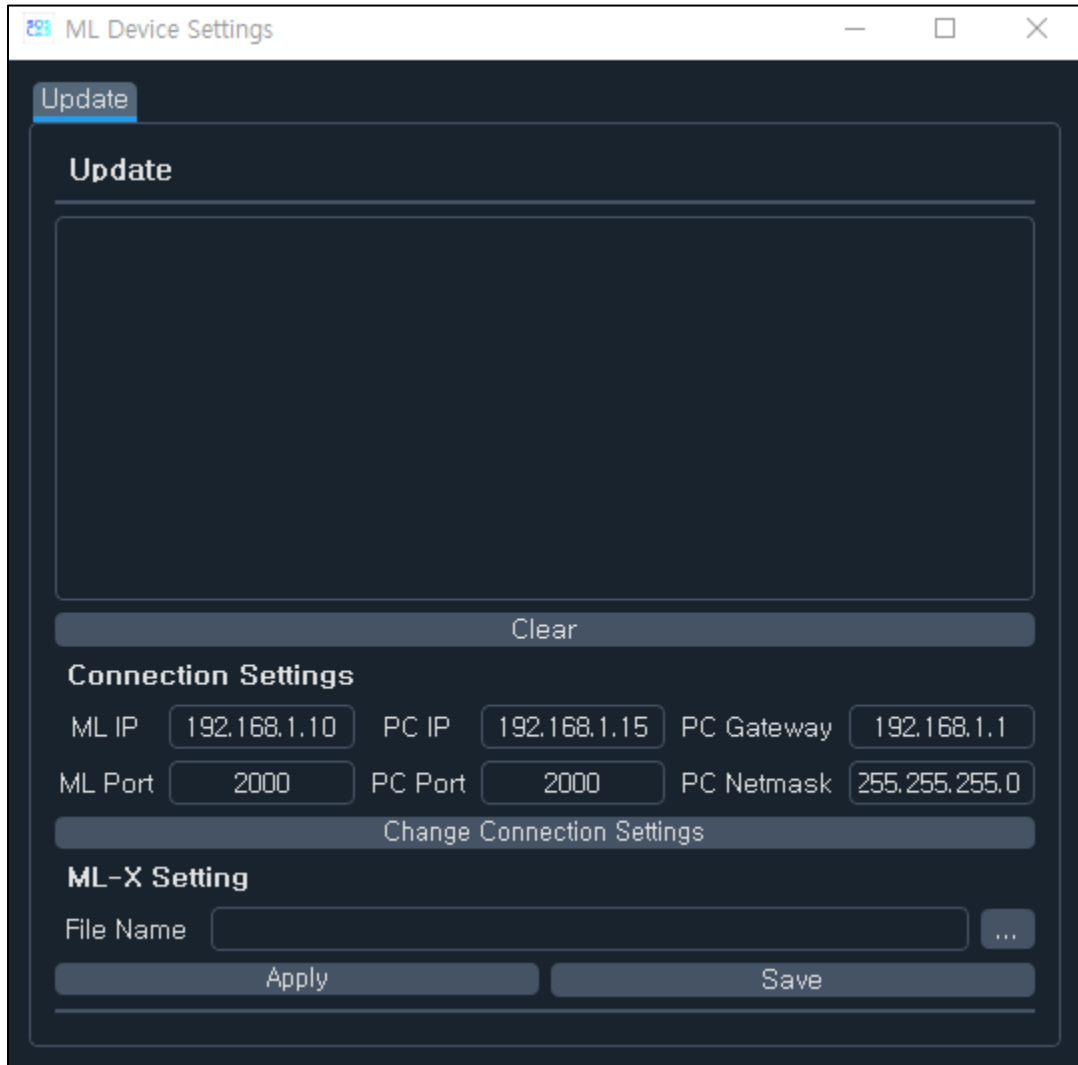


데이터 불러오기 팝업 창

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

2.3 Device IP Setting

Device Setting에서는 ML-X Device의 IP 변경 기능을 제공합니다. SOS Studio 왼쪽 3번째 버튼 “Device Setting”을 클릭하면 아래와 같은 팝업 창이 나타납니다.



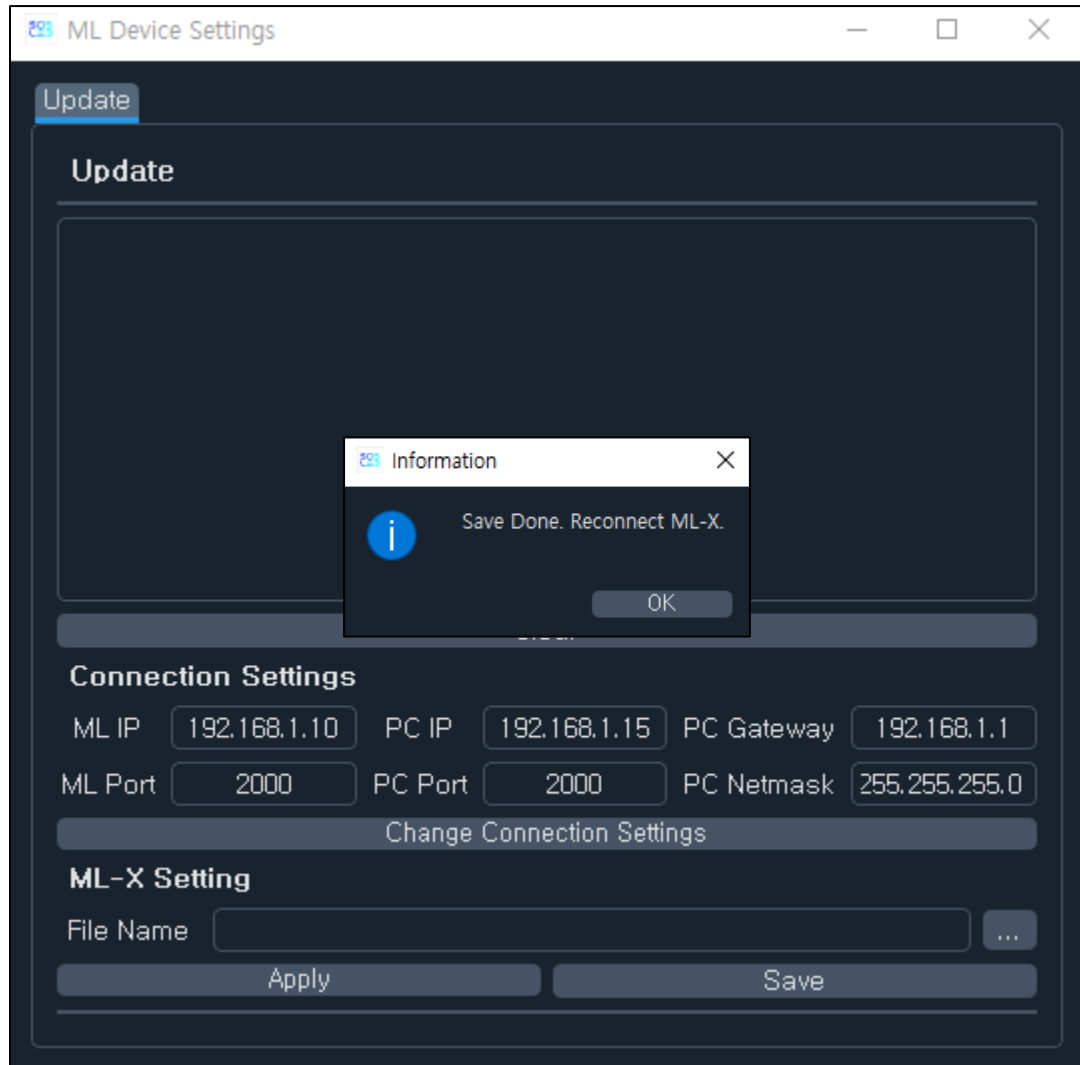
The image shows a software window titled "ML Device Settings". It has a dark theme. At the top left is an "Update" button. Below it is a large empty rectangular box. Underneath this box is a "Clear" button. The next section is titled "Connection Settings" and contains six input fields: "ML IP" (192.168.1.10), "PC IP" (192.168.1.15), "PC Gateway" (192.168.1.1), "ML Port" (2000), "PC Port" (2000), and "PC Netmask" (255.255.255.0). Below these fields is a "Change Connection Settings" button. The final section is titled "ML-X Setting" and contains a "File Name" input field with a browse button (three dots) to its right. At the bottom are two buttons: "Apply" and "Save".

Device Setting 팝업 창

2.3 Device IP Setting

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

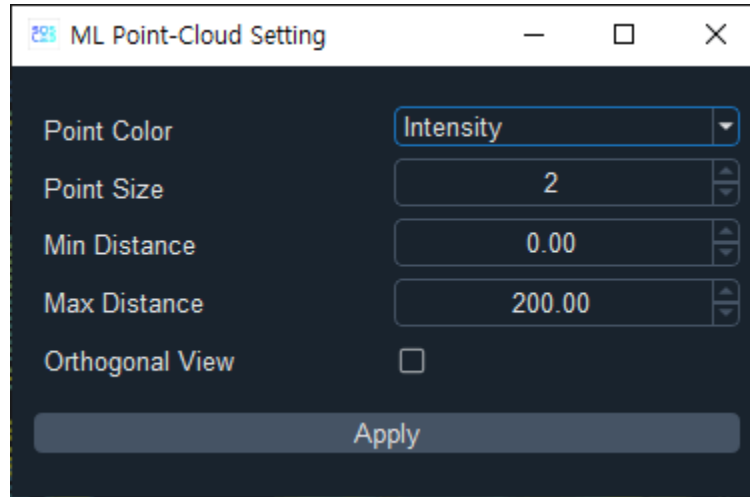
- ① 변경 될 ML-X Device의 IP 입력
- ② “Change Connection Settings” 버튼 클릭
- ③ 변경 후, 전원 케이블 재연결



IP 변경 결과 화면

2.4 Point Cloud Setting

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



Point Cloud Setting 팝업 창

설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

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

2.5 Image Setting

Image Setting에서는 SOS Studio 상단에 위치한 Image Viewer를 설정할 수 있습니다. SOS Studio 왼쪽 5번째 버튼 “Image Setting”을 클릭하면 아래와 같은 팝업 창이 나타납니다.

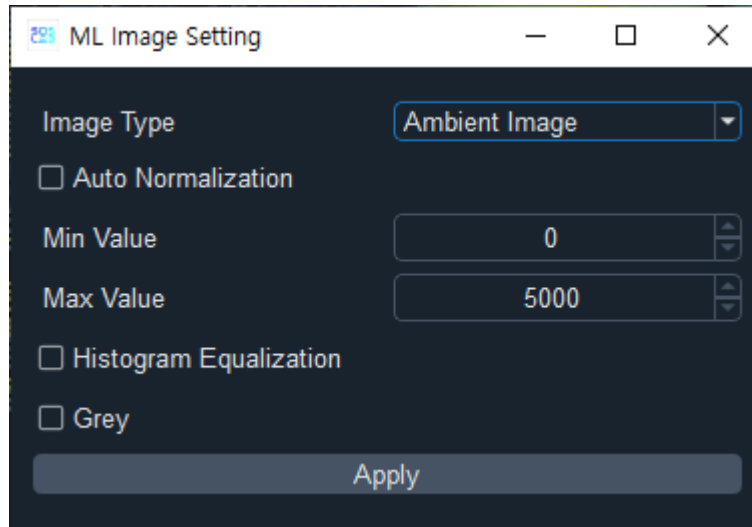


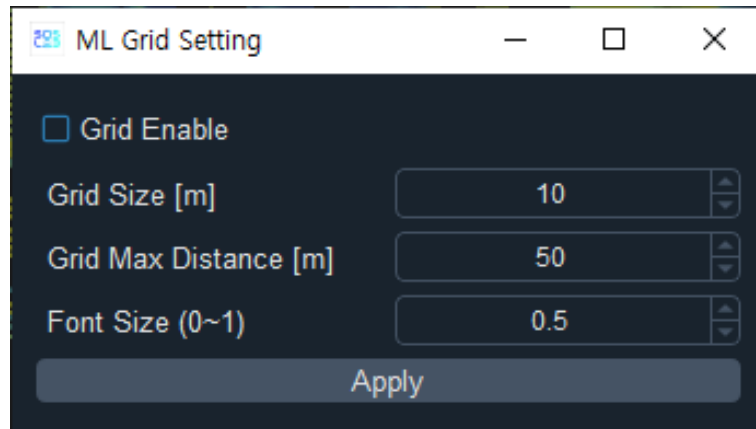
Image Setting 팝업 창

설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

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

2.6 Grid Setting

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



Grid Setting 팝업 창

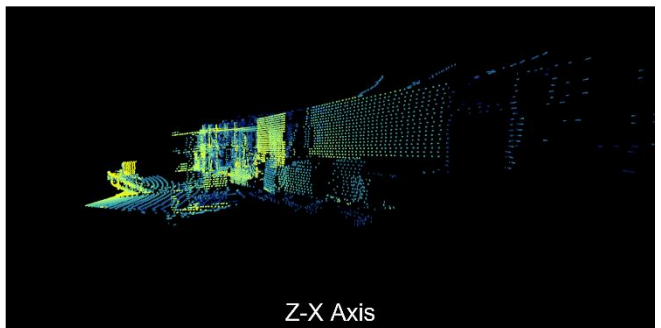
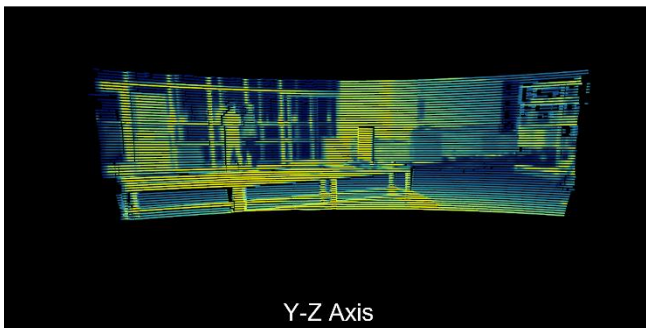
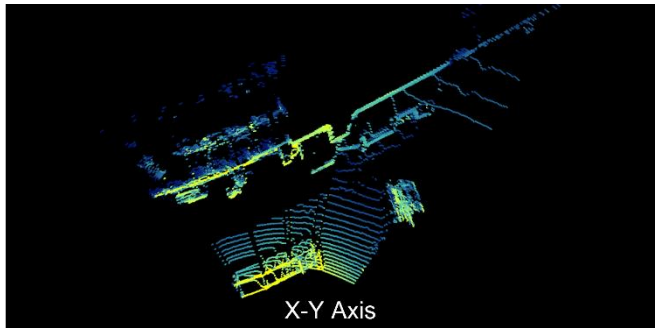
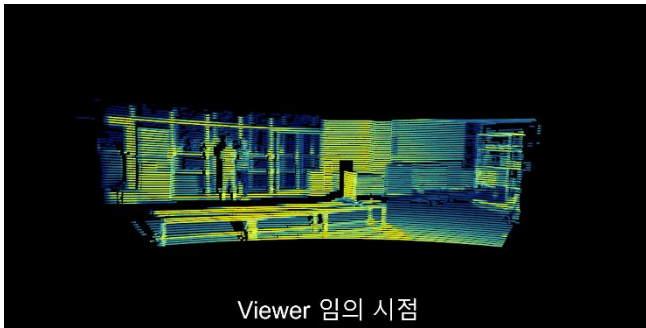
설정할 수 있는 항목들에 대한 설명은 아래와 같습니다.

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

2.7 X-Y / Y-Z / Z-X Axis

SOS Studio 왼쪽 7~9번째 버튼 X-Y / Y-Z / Z-X Axis 은 SOS Studio SOS Studio 중앙에 위치한 Point Cloud Viewer의 시점을 해당 Axis에 맞게 변경하는 기능을 제공합니다.

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



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

2.8 Play / Record Mode

Point Cloud Viewer 아래에 위치한 Play Bar는 **2.2 Data Load**에서 불러온 ML-X 데이터를 재생하는 기능 및 연결 후, 가시화 되는 데이터를 녹화하는 기능을 제공한다.



SOS Studio Play Bar

Play Bar에 각 기능들에 명칭 및 기능은 아래와 같습니다.

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

Appendix

Typedefs

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

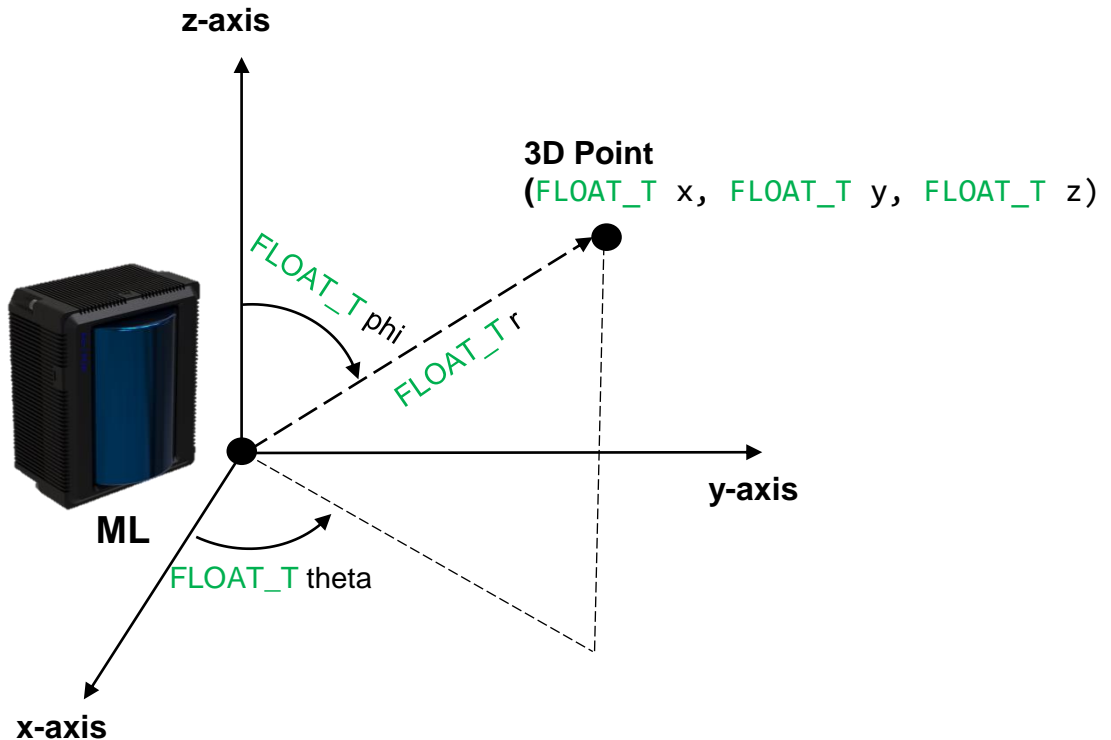
이름	SOSLAB:: <code>UINT_T</code>
선언	<code>typedef uint32_t SOSLAB::UINT_T</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	unsigned int 변수형

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

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

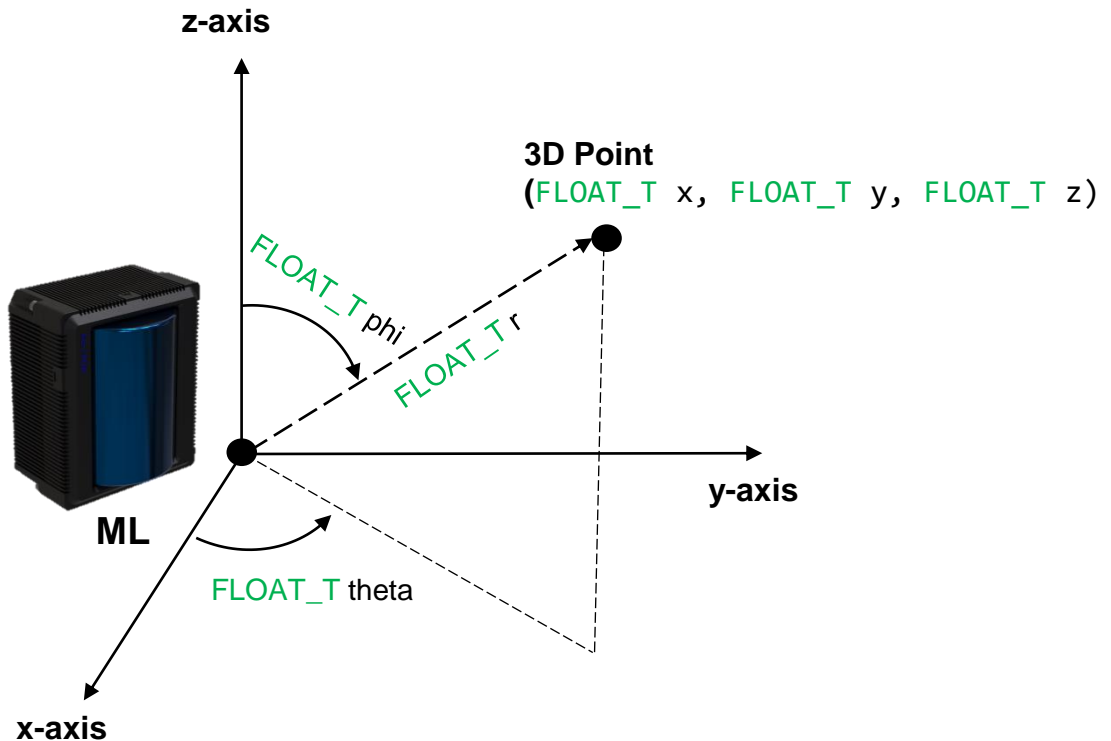
이름	SOSLAB:: <code>pointcloud_t</code>
선언	<code>typedef std::vector<point_t> SOSLAB::pointcloud_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	Point Cloud 변수형

Classes	
변수형	SOSLAB::spherical_point_t
선언	<code>typedef struct _SPHERICAL_POINT_T spherical_point_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	3D Point에 대응하는 Spherical Coordinate System 구조체
Member Variables	Member Variables 설명
<code>FLOAT_T r</code>	Spherical Coordinate System의 Range 값(Depth)
<code>FLOAT_T theta</code>	Spherical Coordinate System의 theta 값
<code>FLOAT_T phi</code>	Spherical Coordinate System의 phi 값



3D Point의 Spherical / Cartesian Coordinate System

Classes	
변수형	SOSLAB:: <code>cartesian_point_t</code>
선언	<code>typedef struct _CARTESIAN_POINT_T cartesian_point_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	3D Point에 대응하는 Cartesian Coordinate System 구조체
Member Variables	Member Variables 설명
<code>FLOAT_T x</code>	Cartesian Coordinate System의 x 값
<code>FLOAT_T y</code>	Cartesian Coordinate System의 y 값
<code>FLOAT_T z</code>	Cartesian Coordinate System의 z 값



3D Point의 Spherical / Cartesian Coordinate System

Classes

변수형	SOSLAB:: <code>point_t</code>
선언	<code>typedef struct _POINT_T point_t</code>
Header	<code>#include "soslab_typedef.h"</code>
설명	3D Point에 대응하는 Coordinate System 구조체
Member Variables	Member Variables 설명
<code>spherical_point_t</code> rtp	Point의 Spherical Coordinate System 정보
<code>cartesian_point_t</code> xyz	Point의 Cartesian Coordinate System 정보
<code>FLOAT_T</code> pw	Point에 대응하는 Intensity 값

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

Classes

변수형	SOSLAB::LidarMl::scene_t
선언	typedef struct _ML1_SCENE_T point_t
Header	#include "ml/libsoslab_ml.h"
설명	Raw Data에 대한 구조체
Member Variables	Member Variables 설명
std::size_t rows	Raw Data의 Height 값
std::size_t cols	Raw Data의 Width 값
std::vector<uint32_t> ambient_image	Ambient 데이터 배열 [size : cols x rows] (Ambient : 측정된 모든 신호 강도)
std::vector<float> depth_image	Depth 데이터 배열 (Depth : 측정된 거리 데이터)
std::vector<uint16_t> intensity_image	Intensity 데이터 배열 (Intensity : 측정된 LiDAR 신호 강도)
pointcloud_t pointcloud	Point cloud 데이터 배열 (Point cloud : 3차원 point의 집합 데이터)

Classes

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

Classes

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