Livox SDK - Installation and ROS Driver

Livox SDK is the software development kit designed for all Livox products. It is developed based on C/C++ following Livox Communication Protocol, and provides easy-to-use C style API. With Livox SDK, users can quickly connect to Livox products and receive point cloud data.

#lidar #livox #ros

Last update: 2021-08-04 17:31:07

Table of Content

1. Prerequisites

- Ubuntu 14.04/ 16.04/ 18.04, support x64 x86 and ARM (Nvidia TX2)
- Windows 7/ 10, Visual Studio 2015 Update3/ 2017/ 2019
- C++11 compiler

Livox SDK needs to be built in the host machine, therefore, some tool-chain and build tools have to be installed.

```
sudo apt-get update && \
sudo apt-get install -y build-essential && \
sudo apt-get install -y curl && \
sudo apt-get install -y git && \
sudo apt-get install -y cmake
```

2. Dependencies

2.1. Point Cloud Library

The Point Cloud Library (PCL) is a large scale, open project[1] for point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation.

```
sudo apt install -y libpcl-dev
```

2.2. Eigen

Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

```
sudo apt install -y libeigen3-dev
```

2.3. OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision library and has bindings for C++, Python, and Java. It is used for a very wide range of applications, including medical image analysis, stitching street view images, surveillance video, detecting and recognizing faces, tracking moving objects, extracting 3D models, and much more. OpenCV can take advantage of multi-core processing and features GPU acceleration for real-time operation.

```
sudo apt install -y python-opencv python3-opencv
```

2.4. Re-link libraries

On Desktop:

```
sudo ln -s /usr/bin/vtk6 /usr/bin/vtk
sudo ln -s /usr/lib/python2.7/dist-
packages/vtk/libvtkRenderingPythonTkWidgets.x86_64-linux-gnu.so /usr/lib/x86_64-
linux-gnu/libvtkRenderingPythonTkWidgets.so
```

On Raspberry Pi:

```
sudo ln -s /usr/bin/vtk6 /usr/bin/vtk
sudo ln -s /usr/lib/python2.7/dist-
packages/vtk/libvtkRenderingPythonTkWidgets.arm-linux-gnueabihf.so /usr/lib/arm-
linux-gnueabihf/libvtkRenderingPythonTkWidgets.so
```
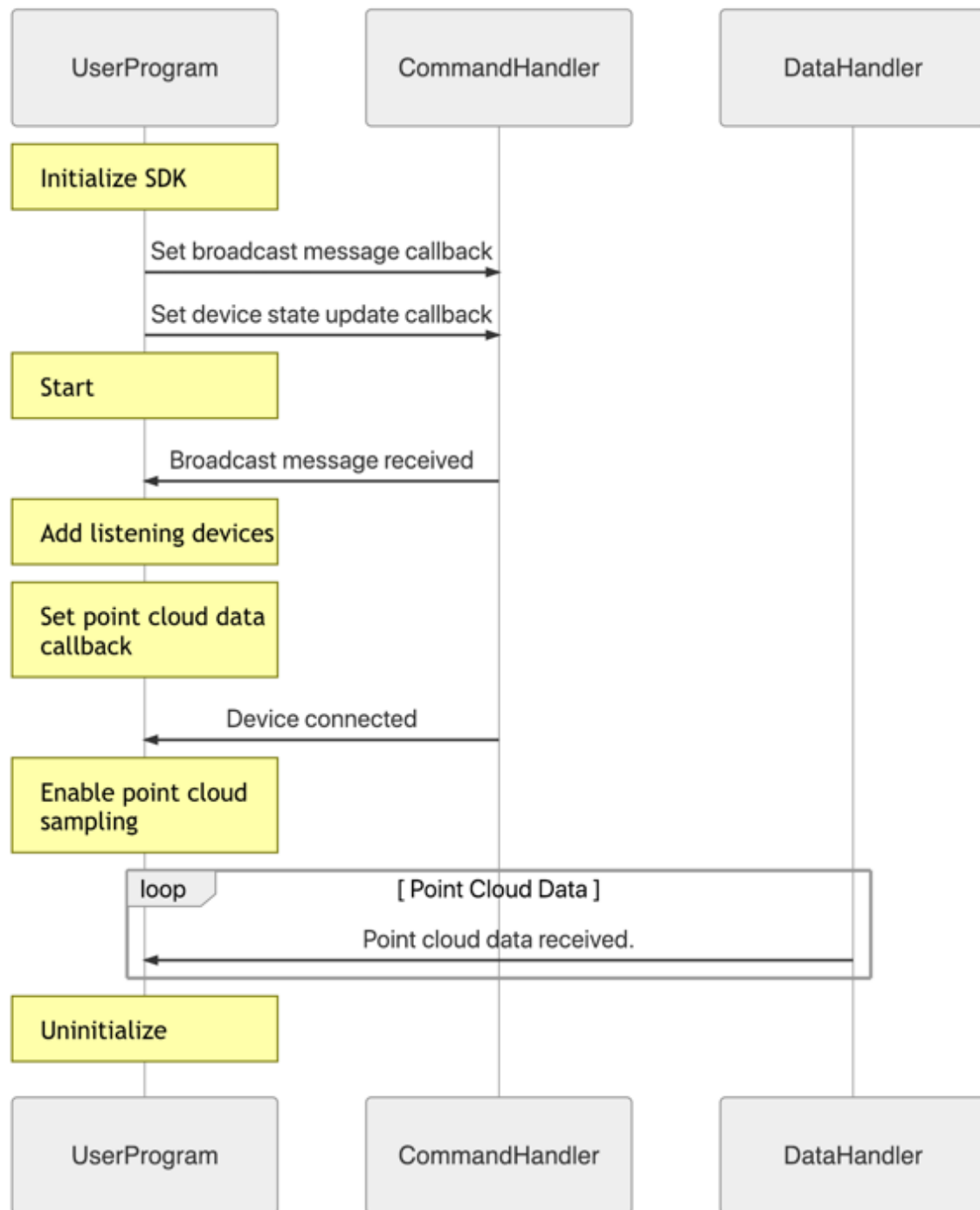
3. Install Livox SDK

> ⓘ The official guide is at https://github.com/Livox-SDK/Livox-SDK.

Livox SDK is the software development kit designed for all Livox products. It is developed based on C/C++ following Livox SDK Communication Protocol, and provides easy-to-use C style API. With Livox SDK, users can quickly connect to Livox products and receive point cloud data.

Livox SDK provides the implementation of control commands and point cloud data transmission. Please refer to the Livox SDK Communication Protocol for detailed information.

In general, the SDK will connect to the Lidar sensor, through UDP packages in a network, the sequence flow is shown as below:

Livox sequence data flow

Installation

```
git clone https://github.com/Livox-SDK/Livox-SDK.git && \
cd Livox-SDK && \
cd build && \
cmake .. && \
```

```
make && \
sudo make install
```

> ✏️ **ARM-Linux Cross Compile**
>
> The following commands will install C and C++ cross compiler tool-chains for 32bit and 64bit ARM board. You need to install the correct tool-chain for your ARM board. For 64bit SoC ARM board, only install 64bit tool-chain, and for 32bit SoC ARM board, only install 32bit tool-chain.
>
> Install **ARM 32 bits** cross compile tool-chain:
>
> ```
> sudo apt-get install gcc-arm-linux-gnueabi g++-arm-linux-gnueabi
> ```
>
> Install **ARM 64 bits** cross compile tool-chain:
>
> ```
> sudo apt-get install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
> ```
>
> Cross Compile Livox-SDK for ARM 32 bits tool-chain, In the Livox SDK directory, run the following commands to cross compile the project:
>
> ```
> cd Livox-SDK && \
> cd build && \
> cmake .. -DCMAKE_SYSTEM_NAME=Linux && \
> -DCMAKE_C_COMPILER=arm-linux-gnueabi-gcc && \
> -DCMAKE_CXX_COMPILER=arm-linux-gnueabi-g++ && \
> make  && \
> sudo make install
> ```
>
> For ARM 64 bits tool-chain, in the Livox SDK directory, run the following commands to cross compile the project:
>
> ```
> cd Livox-SDK && \
> cd build && \
> cmake .. -DCMAKE_SYSTEM_NAME=Linux && \ -DCMAKE_C_COMPILER=aarch64-linux-gnu-gcc
> && \ -DCMAKE_CXX_COMPILER=aarch64-linux-gnu-g++ && \
> make && \
> sudo make install
> ```
>
> Note that `gcc`/`g++` cross compiler need to support C ++11 standard.

4. Install Livox ROS Driver

ROS driver can be run under Ubuntu 14.04/ 16.04/ 18.04 operating system with ROS environment (indigo, kinetic, melodic) installed.

4.1. Install ROS

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. Be sure to install the full version of ROS (ros-<distro>-desktop-full).

Refer to Install ROS guide to complete the installation.

Main steps include:

1. Setup source list to get ROS packages:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Add keys:

```
sudo apt install -y curl && \
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo
apt-key add -
```

3. Then pull the package list:

```
sudo apt update
```

4. Finally, install a desktop-full package as recommended to start learning:

```
sudo apt install -y ros-melodic-desktop-full
```

4.2. Install Livox ROS driver

Get `livox_ros_driver` from GitHub

```
git clone https://github.com/Livox-SDK/livox_ros_driver.git ws_livox/src
```

⚠ Be sure to use the above command to clone the code to the local, otherwise it will compile error due to the file path problem.

Then build it:

```
cd ws_livox && \
catkin_make
```

ⓘ

> If running `catkin_make` gives error of command not found, it's probably that the ROS `setup.bash` is not executed and included in `~/.bashrc`. See above section to source it.

Read more in https://github.com/Livox-SDK/livox_ros_driver.

This driver supports both 2 modes: HUB `*_hub_*` or LIDAR Direct `*_lidar_*`.

4.2.1. ROS Node

This driver will create a new node named `livox_lidar_publisher`, which publishes 2 new types of messages:

Livox Point Cloud message

```
# Livox publish pointcloud msg format.

Header header              # ROS standard message header
uint64 timebase            # The time of first point
uint32 point_num           # Total number of pointclouds
uint8  lidar_id            # Lidar device id number
uint8[3]  rsvd             # Reserved use
CustomPoint[] points       # Pointcloud data
```

Livox Point

```
# Livox costom pointcloud format.

uint32 offset_time         # offset time relative to the base time
float32 x                  # X axis, unit:m
float32 y                  # Y axis, unit:m
float32 z                  # Z axis, unit:m
uint8 reflectivity         # reflectivity, 0~255
uint8 tag                  # livox tag
uint8 line                 # laser number in lidar
```

4.2.2. Configurations

The configuration file is in `ws_livox/src/livox_ros_driver/config`.

LiDAR configuration parameter

Parameter	Type	Description	Default
`broadcast_code`	String	LiDAR broadcast code	x
`enable_connect`	Boolean		false

Parameter	Type	Description	Default
return_mode	Int	Return mode: 0 – First single return mode 1 – Strongest single return mode 2 – Dual return mode	0
coordinate	Int	Coordinate: 0 – Cartesian 1 – Spherical	0
imu_rate	Int	Push frequency of IMU sensor data: 0 – stop push 1 – 200 Hz Currently only Horizon supports this, MID serials do not support it	0
extrinsic_parameter_source	Int	Whether to enable extrinsic parameter automatic compensation of LiDAR external reference 0 – Disabled 1 – Enabled	0

Timestamp synchronization

Parameter	Type	Description	Default
enable_timesync	Boolean		false
device_name	String	Name of the serial device which outputs GPRMC/GNRMC messages every second	/dev/ttyUSB0
comm_device_type	Int	Type of device sending timestamp information 0 – Serial port or USB virtual serial port device other – not support	0
baudrate_index	Int	Baud rate of serial device: 0 – 2400 1 – 4800 2 – 9600 3 – 19200 4 – 38400 5 – 57600 6 – 115200 7 – 230400 8 – 460800 9 – 500000 10 – 576000 11 – 921600	2 (9600)

Parameter	Type	Description	Default
`parity_index`	Int	parity type 0 – 8 bits data without parity 1 – 7 bits data 1bit even parity 2 – 7 bits data 1bit odd parity 3 – 7 bits data 1bit 0, without parity	0

livox_lidar_config.json

```json
{
    "lidar_config": [
        {
            "broadcast_code": "1PQDH5B00100041",
            "enable_connect": false,
            "return_mode": 0,
            "coordinate": 0,
            "imu_rate": 0,
            "extrinsic_parameter_source": 0,
            "enable_high_sensitivity": false
        }
    ],
    "timesync_config": {
        "enable_timesync": false,
        "device_name": "/dev/ttyUSB0",
        "comm_device_type": 0,
        "baudrate_index": 2,
        "parity_index": 0
    }
}
```

4.2.3. Timestamp

Prepare a GPS device to ensure that the GPS can output UTC time information in GPRMC/GNRMC format through the serial port or USB virtual serial port, and support PPS signal output.

1. Connect the GPS serial port to the host running `livox_ros_driver`, set the corresponding device name in the config file
2. Connect the GPS PPS signal line to LiDAR
3. Be sure to set the output frequency of GPRMC/GNRMC time information of GPS to 1Hz

4.2.4. Launches

Different launch files have different configuration parameter values and are used in different scenarios:

Launch file name	Description
`livox_lidar.launch`	Connect to Livox LiDAR device Publish `pointcloud2` format data

Launch file name	Description
`livox_lidar_msg.launch`	Connect to Livox LiDAR device Publish `livox customized pointcloud` data
`livox_lidar_rviz.launch`	Connect to Livox LiDAR device Publish `pointcloud2` format data Autoload rviz

Launch params

Parameter	Description	Default
`publish_freq`	Set the frequency of point cloud publish	10.0
`multi_topic`	0 – All LiDAR devices use the same topic to publish pointcloud data 1 – Each LiDAR device has its own topic to publish point cloud data	0
`xfer_format`	0 – Livox pointcloud2(PointXYZRTL) pointcloud format 1 – Livox customized pointcloud format 2 – Standard pointcloud2 (pcl :: PointXYZI) pointcloud format in the PCL library	0

4.2.5. Missing features

Even the build is completed, there are still some warning during the compilation due to missing definitions.

```
-- Could NOT find ensenso (missing: ENSENSO_LIBRARY ENSENSO_INCLUDE_DIR)
** WARNING ** io features related to ensenso will be disabled
-- Could NOT find DAVIDSDK (missing: DAVIDSDK_LIBRARY DAVIDSDK_INCLUDE_DIR)
** WARNING ** io features related to davidSDK will be disabled
-- Could NOT find DSSDK (missing: _DSSDK_LIBRARIES)
** WARNING ** io features related to dssdk will be disabled
** WARNING ** io features related to pcap will be disabled
** WARNING ** io features related to png will be disabled
-- Found libusb-1.0: /usr/include
** WARNING ** io features related to libusb-1.0 will be disabled
-- Checking for module 'flann'
--    Found flann, version 1.9.1
-- Found Flann: /usr/lib/x86_64-linux-gnu/libflann_cpp_s.a
-- Could NOT find ensenso (missing: ENSENSO_LIBRARY ENSENSO_INCLUDE_DIR)
** WARNING ** visualization features related to ensenso will be disabled
-- Could NOT find DAVIDSDK (missing: DAVIDSDK_LIBRARY DAVIDSDK_INCLUDE_DIR)
** WARNING ** visualization features related to davidSDK will be disabled
-- Could NOT find DSSDK (missing: _DSSDK_LIBRARIES)
** WARNING ** visualization features related to dssdk will be disabled
-- Could NOT find RSSDK (missing: _RSSDK_LIBRARIES)
** WARNING ** visualization features related to rssdk will be disabled
```

At this time, they do not harm, then these warning may be solved later.