LD14P LiDAR

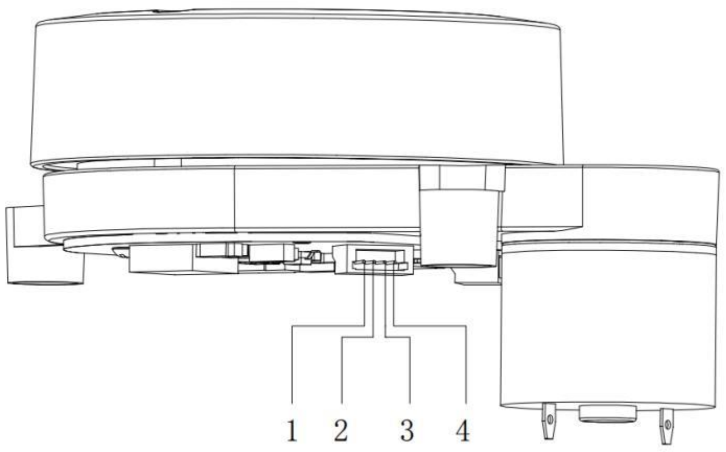
Development Manual V0.2

Contents

1. WORKING PRINCIPLE

1.1 COMMUNICATION INTERFACE

LD14P connects to external systems via the 1.25mm 4PIN connector for power supply and data reception. Please refer to the following diagram/table for specific interface definitions and parameter:



No.	Signal	Type	Remarks	Min.	Typ.	Max.
1	Rx	Input	LiDAR data input	0V	-	3.3V
2	GND	Power	Power negative	-	0V	-
3	Tx.	Output	LiDAR data output	0V	3.3V	3.5V
4	VCC	Power	Power positive	4.5V	5V	5.5V

1.2 INTRODUCTION TO FUNCTIONS

Adopting triangulation technology, the LD14P LiDAR measures distance at a rate of 4000 times per second. During each measurement, LD14P emits infrared laser from a fixed angle, which is then reflected by the target object and received by the receiving unit. By establishing a triangle relationship among the laser, the target object, and the receiving unit, the distance is calculated.

After obtaining distance data, LD14P merges the angle values measured by the angle measurement unit into point cloud data. This data is then wirelessly transmitted to an external interface.

LD14P supports both internal and external speed control. When the Rx pin of LD14P is pulled down to ground, upon power-up, LD14P enters the internal velocity control mode by default, with a scanning frequency of 6Hz. The external speed control supports both PWM control and serial port control.

1) PWM control:

Connect the Rx pin of LD14P to PWM signal, which can be used to control the start, stop and speed of the motor via the PWM signal duty ratio for activating external speed control.

a. Input PWM frequency of 500Hz-1.5KHz, 1KHz is recommended.

b. Duty ratio in the range of (45%, 55%) (excluding 45% and 55%), requires a minimum input duration of 100ms.

Once triggered, the LD14P remains in external speed control mode unless it is powered off and rebooted to internal speed control. In addition, speed control is available by adjusting the PWM duty ratio. Due to individual differences in each product motor, the actual speed may vary when the duty ratio is set to typical values. For precise control of the motor speed, closed-loop control is required based on the speed information in the received data.

Note: When external speed control is not used, the PWM pin must be grounded.

2) Serial port control:

The Rx pin of LD14P connects to the Tx pin of an external device's serial port. To enter external speed control, relevant control commands should be sent at the baud rate defined by the communication interface.

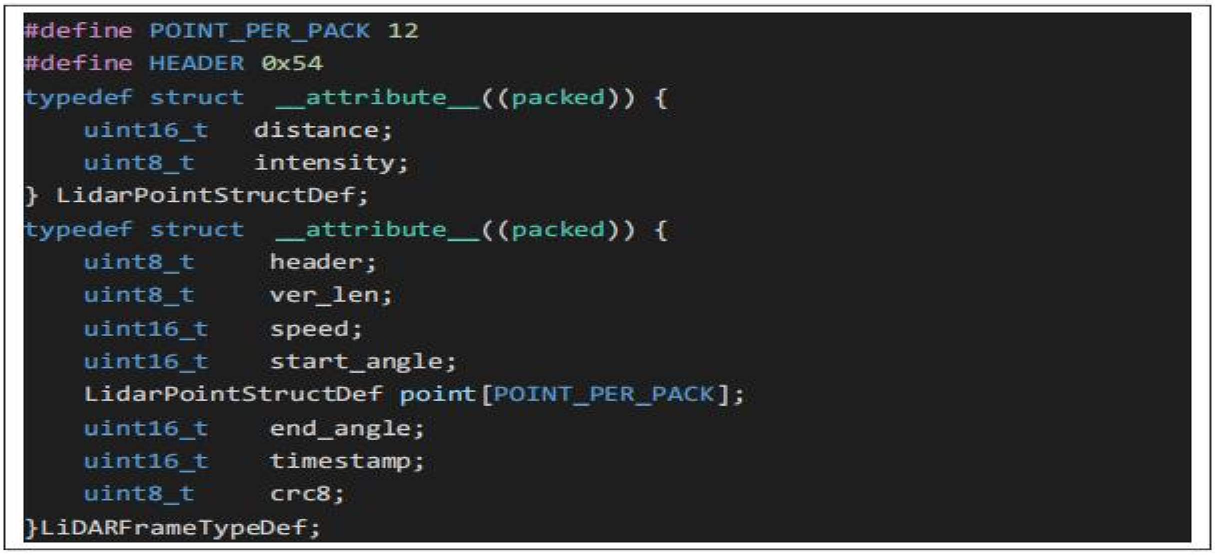
2. COMMUNICATION PROTOCOL

2.1 DATA PACKET FORMAT

Start Character	Frame	LiDAR Speed		Start Angle		Measurement Data	End Angle		Timestamp		Checksum
1 Byte	1 Byte	2 Bytes		2 Bytes		12x3 Bytes	2 Bytes		2 Bytes		1 Byte
0x54	0x2C	LSB	MSB	LSB	MSB	......	LSB	MSB	LSB	MSB	CRC8

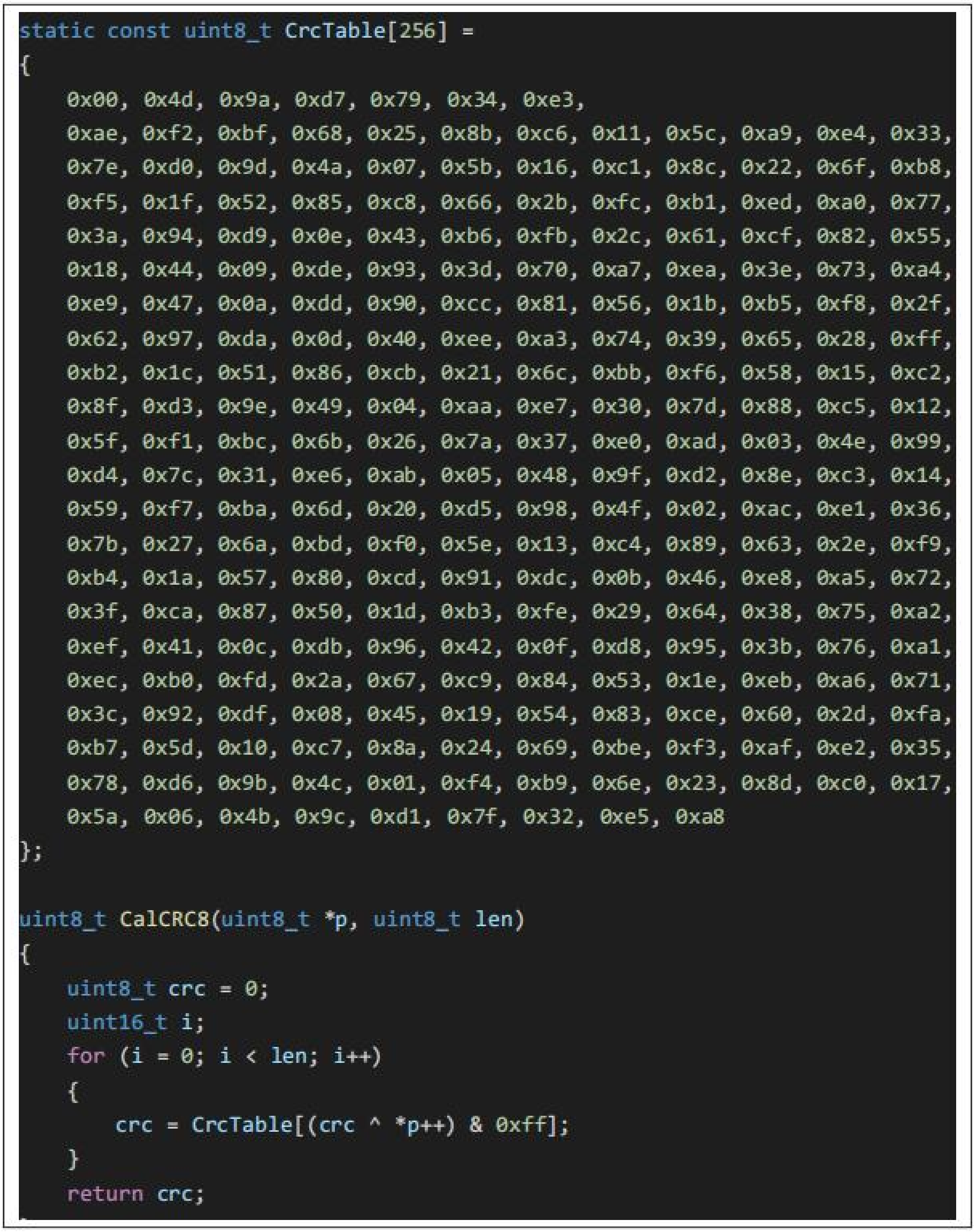
- **Start character:** 1 byte, fixed value of 0x54, indicating the start of the data packet.
- **Frame:** 1 byte, fixed value of 0x2C, the lower five bits indicate the number of measurement points in a packet, currently fixed at 12.
- **LiDAR speed:** 2 bytes, unit in degree per second.
- **Start angle:** 2 bytes, unit in 0.01 degrees.
- **Measurement data:** 3 bytes, consisting of LSB distance, MSB distance and intensity information.
- **End angle:** 2 bytes, unit in 0.01 degrees.
- **Timestamp:** 2 bytes, unit in milliseconds, up to 30000, reset upon reaching 30000.
- **CRC checksum:** CRC8 checksum for all preceding data.

Data structure reference as follows:



```
#define POINT_PER_PACK 12
#define HEADER 0x54
typedef struct  __attribute__((packed)) {
    uint16_t   distance;
    uint8_t    intensity;
} LidarPointStructDef;
typedef struct  __attribute__((packed)) {
    uint8_t     header;
    uint8_t     ver_len;
    uint16_t    speed;
    uint16_t    start_angle;
    LidarPointStructDef point[POINT_PER_PACK];
    uint16_t    end_angle;
    uint16_t    timestamp;
    uint8_t     crc8;
}LiDARFrameTypeDef;
```

CRC checksum calculation as follows:



```
static const uint8_t CrcTable[256] =
{
    0x00, 0x4d, 0x9a, 0xd7, 0x79, 0x34, 0xe3,
    0xae, 0xf2, 0xbf, 0x68, 0x25, 0x8b, 0xc6, 0x11, 0x5c, 0xa9, 0xe4, 0x33,
    0x7e, 0xd0, 0x9d, 0x4a, 0x07, 0x5b, 0x16, 0xc1, 0x8c, 0x22, 0x6f, 0xb8,
    0xf5, 0x1f, 0x52, 0x85, 0xc8, 0x66, 0x2b, 0xfc, 0xb1, 0xed, 0xa0, 0x77,
    0x3a, 0x94, 0xd9, 0x0e, 0x43, 0xb6, 0xfb, 0x2c, 0x61, 0xcf, 0x82, 0x55,
    0x18, 0x44, 0x09, 0xde, 0x93, 0x3d, 0x70, 0xa7, 0xea, 0x3e, 0x73, 0xa4,
    0xe9, 0x47, 0x0a, 0xdd, 0x90, 0xcc, 0x81, 0x56, 0x1b, 0xb5, 0xf8, 0x2f,
    0x62, 0x97, 0xda, 0x0d, 0x40, 0xee, 0xa3, 0x74, 0x39, 0x65, 0x28, 0xff,
    0xb2, 0x1c, 0x51, 0x86, 0xcb, 0x21, 0x6c, 0xbb, 0xf6, 0x58, 0x15, 0xc2,
    0x8f, 0xd3, 0x9e, 0x49, 0x04, 0xaa, 0xe7, 0x30, 0x7d, 0x88, 0xc5, 0x12,
    0x5f, 0xf1, 0xbc, 0x6b, 0x26, 0x7a, 0x37, 0xe0, 0xad, 0x03, 0x4e, 0x99,
    0xd4, 0x7c, 0x31, 0xe6, 0xab, 0x05, 0x48, 0x9f, 0xd2, 0x8e, 0xc3, 0x14,
    0x59, 0xf7, 0xba, 0x6d, 0x20, 0xd5, 0x98, 0x4f, 0x02, 0xac, 0xe1, 0x36,
    0x7b, 0x27, 0x6a, 0xbd, 0xf0, 0x5e, 0x13, 0xc4, 0x89, 0x63, 0x2e, 0xf9,
    0xb4, 0x1a, 0x57, 0x80, 0xcd, 0x91, 0xdc, 0x0b, 0x46, 0xe8, 0xa5, 0x72,
    0x3f, 0xca, 0x87, 0x50, 0x1d, 0xb3, 0xfe, 0x29, 0x64, 0x38, 0x75, 0xa2,
    0xef, 0x41, 0x0c, 0xdb, 0x96, 0x42, 0x0f, 0xd8, 0x95, 0x3b, 0x76, 0xa1,
    0xec, 0xb0, 0xfd, 0x2a, 0x67, 0xc9, 0x84, 0x53, 0x1e, 0xeb, 0xa6, 0x71,
    0x3c, 0x92, 0xdf, 0x08, 0x45, 0x19, 0x54, 0x83, 0xce, 0x60, 0x2d, 0xfa,
    0xb7, 0x5d, 0x10, 0xc7, 0x8a, 0x24, 0x69, 0xbe, 0xf3, 0xaf, 0xe2, 0x35,
    0x78, 0xd6, 0x9b, 0x4c, 0x01, 0xf4, 0xb9, 0x6e, 0x23, 0x8d, 0xc0, 0x17,
    0x5a, 0x06, 0x4b, 0x9c, 0xd1, 0x7f, 0x32, 0xe5, 0xa8
};

uint8_t CalCRC8(uint8_t *p, uint8_t len)
{
    uint8_t crc = 0;
    uint16_t i;
    for (i = 0; i < len; i++)
    {
        crc = CrcTable[(crc ^ *p++) & 0xff];
    }
    return crc;
```

2.2 PROTOCOL COMMAND

2.2.1 COMMAND DATA FORMAT REMARKS

Command data remarks:

Start Character	Command Code	Data Length	Command Data	Parity Bit
1 Byte	1 Byte	1 Byte	4 Bytes	1 Byte
0x54	Mode	0x04	DataBuffer	CRC8

- **Start character:** 1 byte, fixed value of 0x54, indicating the start of the data packet.
- **Command code:** 1 byte, command mode.
- **Data length:** data length, fixed at 0x04.
- **Command data:** command data.
- **CRC checksum:** CRC8 checksum for all preceding data.

Returned data remarks:

Start Character	Command Code	Data Length	Command Data	Parity Bit
1 Byte	1 Byte	1 Byte	4 Bytes	1 Byte
0x54	Mode	0x04	DataBuffer	CRC8

- **Start character:** 1 byte, fixed value of 0x54, indicating the start of the data packet.
- **Command code:** 1 byte, receive the command mode.
- **Data length:** the minimum data length is 4 bytes, depending on the specific command.
- **Command data:** response data, the length varies depending on the commands.
- **CRC checksum:** CRC8 checksum for all preceding data.

Function summary:

Command Code	Functions
0xA0	Start the LiDAR motor
0xA1	Stop the LiDAR motor
0xA2	Set the target speed of LiDAR, data lose when power off
0xA3	Query the LiDAR target speed

2.2.2 START (MODE=0XA0)

When Mode=0xA0, the command is to start the motor, and the DataBuffer in the command has no actual meaning and defaults to 0. The DataBuffer length in the LiDAR response is 4 bytes and has no actual meaning.

Reference command sent: 0x54 A0 04 00 00 00 00 5E

Reference command received: 0x54 A0 04 00 00 00 00 5E

2.2.3 STOP (MODE=0XA1)

When Mode=0xA1, the command is to stop the motor, and the DataBuffer in the command has no actual meaning and defaults to 0. The DataBuffer length in the LiDAR response is 4 bytes and has no actual meaning.

Reference command sent: 0x54 A1 04 00 00 00 00 4A

Reference command received: 0x54 A1 04 00 00 00 00 4A

2.2.4 SPEED CONTROL (MODE=0XA2)

When Mode=0xA2 in the command, the function is to set the speed, and the DataBuffer in the command represents the target speed. The DataBuffer length in the LiDAR response is 4 bytes, indicating the received target speed from the command.

Speed analysis:

speed = (DataBuffer[0]) | (DataBuffer[1] << 8);

Speed unit: degrees/second

Reference command sent: 0x54 A2 04 D8 09 00 00 16 (setting speed to 2520 degrees/second, i.e., 7Hz)

Reference response received: 0x54 A2 04 D8 09 00 00 16

Reference command sent: 0x54 A2 04 70 08 00 00 A1 (setting speed to 2160 degrees/second, i.e., 6Hz)

Reference response received: 0x54 A2 04 70 08 00 00 A1

2.2.5 QUERY THE CURRENT SPEED (MODE=0XA3)

When Mode=0xA3 in the command, the function is to query the current target speed, and the DataBuffer in the command has no actual meaning. The DataBuffer length in the LiDAR response is 4 bytes, indicating the current target speed of the LiDAR.

Speed analysis:

speed = (DataBuffer[0]) | (DataBuffer[1] << 8);

Speed unit: degrees/second

Reference command sent: 0x54 A3 04 00 00 00 00 62

Reference response received: 0x54 A3 04 D8 09 00 00 02 (setting the target speed to 2520 degrees/second, i.e., 7Hz)

2.3 HEALTH INFORMATION DATA PACKET

The LiDAR has an error reporting mechanism where it reports health information, including error codes, at intervals of every 1 second.

2.3.1 DATA FORMAT

Start Character	Information Remarks	Error Code	Error Mode	Parity Bit
1 Byte	1 Byte	2 Bytes	1 Byte	1 Byte
0x54	Information	ErrorMode	ErrorMode	CRC8

- **Start character:** 1 byte, fixed value of 0x54, indicating the start of the data packet.
- **Information remarks:** 1 byte, fixed at 0x1F.
- **Error code:** 2 bytes, as shown below:

Bit0~Bit15	Remarks
Bit0	Laser tube abnormal
Bit1	Laser tube abnormal
Bit2	Receiver board voltage abnormal
Bit3	Sensor abnormal
Bit4	Encoder abnormal
Bit5	Motor error or locked rotor
Bit6	LiDAR blockage
Bit7~Bit15	TBD

- **Error mode:** error mode, 0x00: normal, 0x01: warn, 0x02: error
- **CRC checksum:** CRC8 checksum for all preceding data.

2.3.2 ABNORMALITY REMARKS

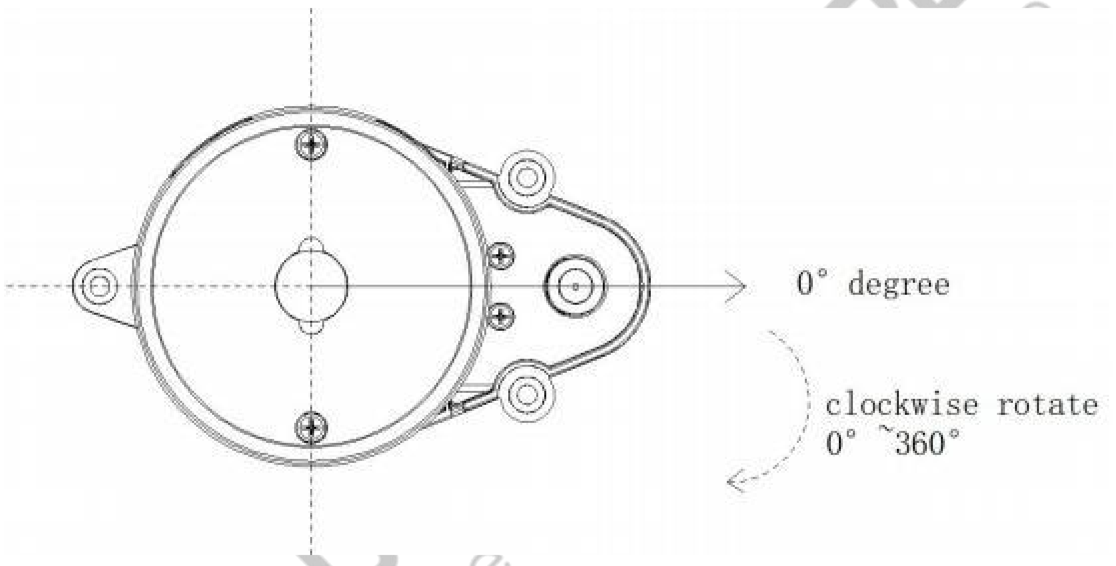
1) Laser tube abnormal: Check whether the laser tube is abnormal by the collected voltage when it is turned on, which occurs only once during power-on self-test. The possible causes of abnormality include laser tube damage or improper connection between the laser tube and the received board.
2) Receiver board voltage abnormal: Check whether the receiver board voltage is abnormal by the collected power voltage, which occurs only once during power-on self-test. The possible causes of this abnormality include coil issues or the abnormal power supply of the transmission board.
3) Sensor abnormal: The sensor is abnormal when the its timing is irregular. The possible causes of this abnormality may be the sensor damage or the improper sensor connection to the receiver board.
4) Encoder abnormal: Check whether the encoder is abnormal by counting the total number of teeth in one rotation. If the number of teeth in one rotation does not match the actual numbers, it’s abnormal. Possible reasons for this abnormality may be encoder or encoder disc damage.
5) Motor abnormal or locked rotor: Check whether the motor is blocked or not by judging the

accumulated number of points in one rotation. If the accumulated points exceed 4000 points in one rotation, it is abnormal. Possible causes include motor locked rotor, motor damage or encoder abnormal.

6) LiDAR blockage: if there is no effective data in one rotation, the LiDAR is blocked.

3. COORDINATE SYSTEM DEFINITION

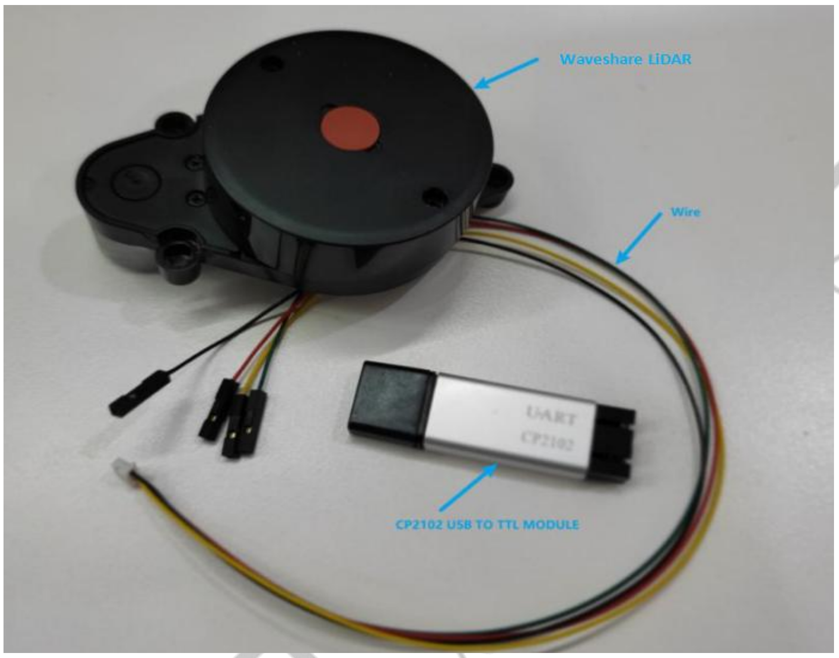
The LD14P commonly follows a left-hand rule coordinate system where the front of the sensor is defined as the X-axis of the coordinate system (i.e. the 0-angle position), the origin of the coordinate system is the center of rotation of the ranging unit, and the angle of rotation increases along the clockwise direction, as shown in the following diagram:



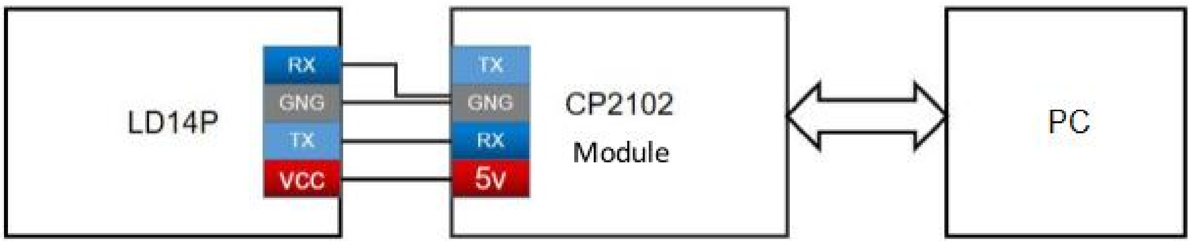
4. DEVELOPMENT KIT USAGE

4.1 HARDWARE CONNECTION AND REMARKS

1) LD14P LiDAR, wires, and CP2102 USB TO TTL Module



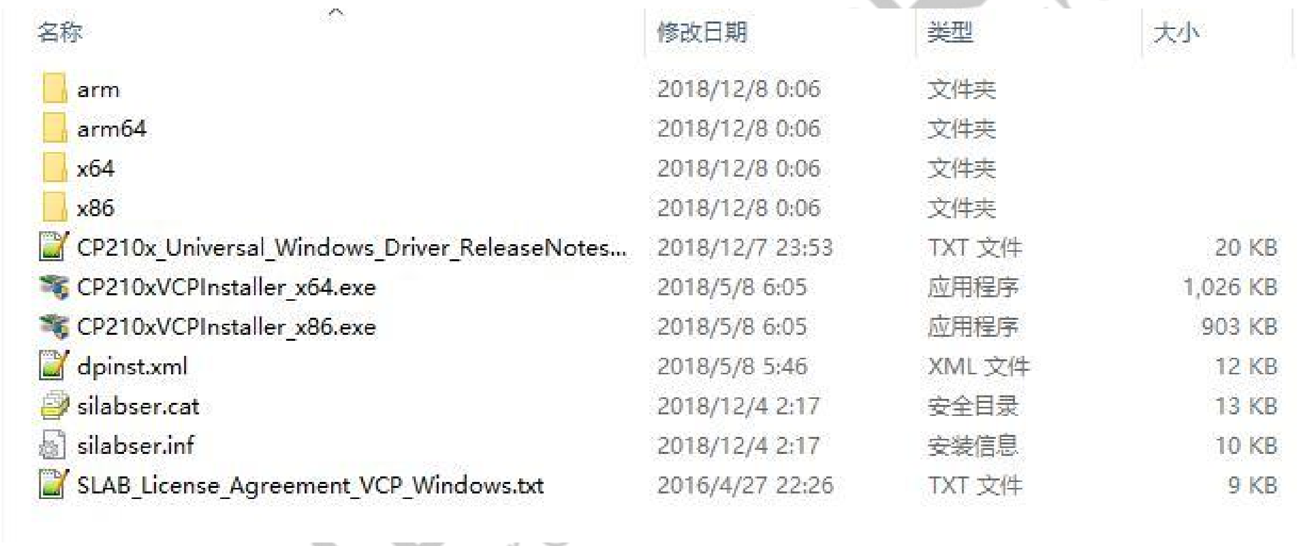
2) The connection is shown below:



4.2 DRIVER DOWNLOAD ON WINDOWS

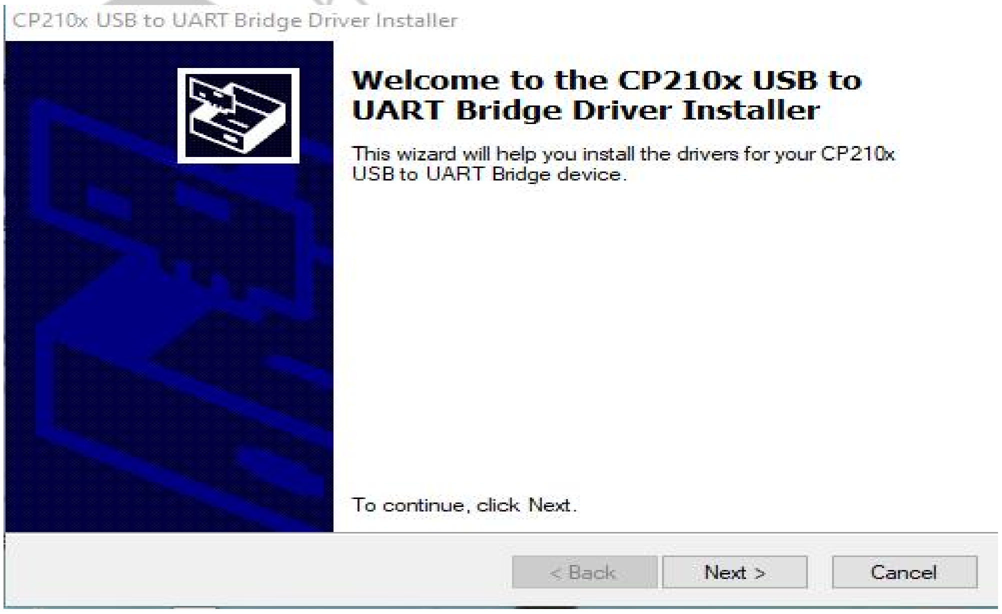
When evaluating our company's product on Windows, it's necessary to install the USB-to-serial driver for the USB adapter. This is because the USB adapter used in our development kit employs the CP2102 USB-to-serial signal chip, and its driver demo can be downloaded from Silicon Labs' official website.

After extracting the CP210x_Universal_Windows_Driver driver package, run the .exe file in the driver installation package directory, choosing between X86 (32-bit) or X64 (64-bit) based on the Windows system version.



名称	修改日期	类型	大小
arm	2018/12/8 0:06	文件夹	
arm64	2018/12/8 0:06	文件夹	
x64	2018/12/8 0:06	文件夹	
x86	2018/12/8 0:06	文件夹	
CP210x_Universal_Windows_Driver_ReleaseNotes...	2018/12/7 23:53	TXT 文件	20 KB
CP210xVCPInstaller_x64.exe	2018/5/8 6:05	应用程序	1,026 KB
CP210xVCPInstaller_x86.exe	2018/5/8 6:05	应用程序	903 KB
dpinst.xml	2018/5/8 5:46	XML 文件	12 KB
silabser.cat	2018/12/4 2:17	安全目录	13 KB
silabser.inf	2018/12/4 2:17	安装信息	10 KB
SLAB_License_Agreement_VCP_Windows.txt	2016/4/27 22:26	TXT 文件	9 KB

Double-click the .exe file and install it by prompts:



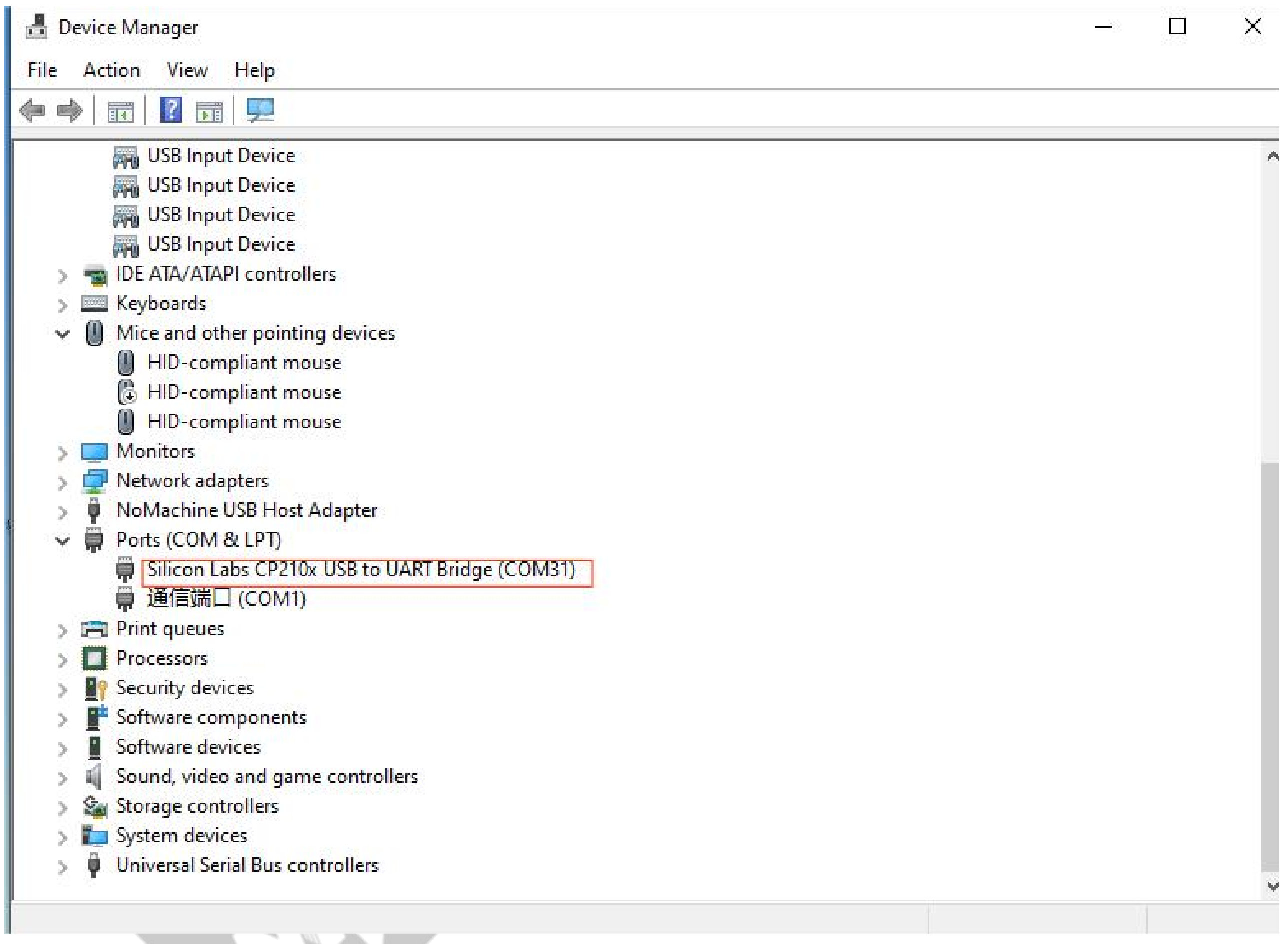
CP210x USB to UART Bridge Driver Installer

Welcome to the CP210x USB to UART Bridge Driver Installer

This wizard will help you install the drivers for your CP210x USB to UART Bridge device.

To continue, click Next.

After the installation is complete, connect the USB adapter from the development kit to your computer. Right-click on "My Computer," select "Properties," navigate to the "Device Manager" in the "System" interface that opens up, expand the "Ports" section, and there you'll find the assigned COM port number corresponding to the recognized CP2102 USB adapter. This indicates a successful driver installation. COM4 is shown in the image below:

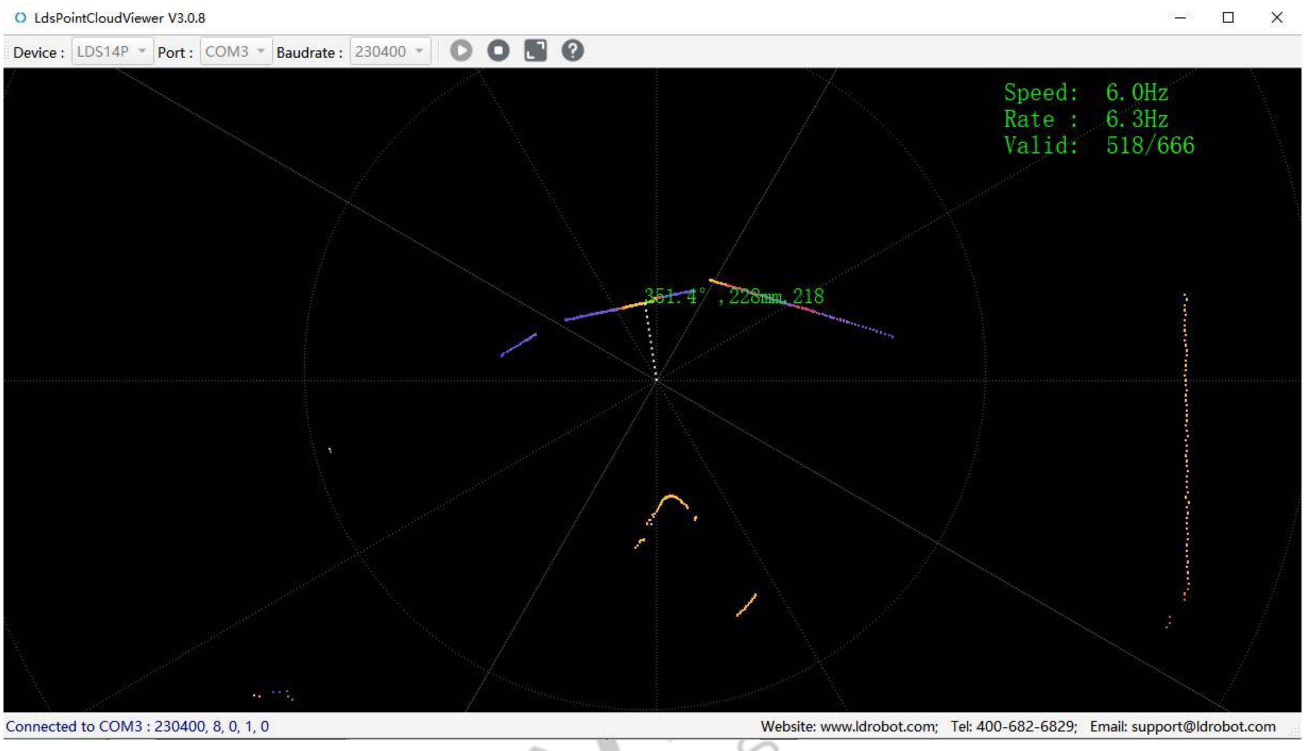

Device Manager
File Action View Help
USB Input Device
USB Input Device
USB Input Device
USB Input Device
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
HID-compliant mouse
HID-compliant mouse
HID-compliant mouse
Monitors
Network adapters
NoMachine USB Host Adapter
Ports (COM & LPT)
Silicon Labs CP210x USB to UART Bridge (COM31)
通信端口 (COM1)
Print queues
Processors
Security devices
Software components
Software devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers

4.3 HOST SOFTWARE USAGE

Our company provides the LdsPointCloudViewer software for real-time visualization of point clouds generated by our product. Developers can use this software to intuitively observe the scanning results of our product.

Before using this software, ensure that the USB driver for our product's USB adapter has been successfully installed. Connect our product to a USB port on a Windows PC, then double-click on

LdsPointCloudViewer.exe. Select the corresponding product model (e.g., LD14P) and port number. Click the 'Start Point Cloud Refresh' button as shown in the image below:


LdsPointCloudViewer V3.0.8
Device : LDS14P
Port : COM3
Baudrate : 230400
Speed: 6.0Hz
Rate : 6.3Hz
Valid: 518/666
351.4° , 228mm, 218
Connected to COM3 : 230400, 8, 0, 1, 0
Website: www.ldrobot.com; Tel: 400-682-6829; Email: support@ldrobot.com

In the above picture, "Speed" indicates the scanning frequency of the LiDAR in Hertz (Hz). "Rate" represents the data packet parsing speed of the LiDAR. "Valid" indicates the number of valid points measured by the LiDAR in one full rotation.

The binary package of the LdsPointCloudViewer software can be accessed for download from our company's open-source repository:

https://github.com/ldrobotSensorTeam/ld_desktop_tool/releases

4.4 SDK USAGE ON LINUX

4.4.1 GET SDK SOURCE CODE

Beta Version: The Linux SDK source code for this product requires contacting our company's FAE or other sales for access. The README document within the source code contains relevant usage instructions. If you wish to utilize a private repository on platforms like GitHub or Gitee for development integration, please feel free to inform our company's FAE or sales. We will strive to meet your development needs.

Official Version: The product's SDK is not currently available to the public.

4.4.2 SDK APPLICATION USAGE

To facilitate users to connect this product to the sensor, this product's SDK integrates fundamental functionalities such as data parsing, processing, and packaging. It's compatible with various commonly used development platforms (Linux, ROS, ROS2). Currently, the integrated functionalities within the SDK are as follows:

1. Data serial port reception buffer addressing
2. Data packet protocol parsing and processing
3. Data noise filtering
4. Data packaging and provision of corresponding interfaces to upper layers.

4.4.3 SDK DEBUG REMARKS

1) SDK platform

This demo is based on ROS platform on Ubuntu operating system, please install the correct Ubuntu and ROS environment following the example operation. If you intend to evaluate this product using SDK on other platforms, you can refer to the corresponding platform README document of the SDK source code.

2) Set up sensor permissions

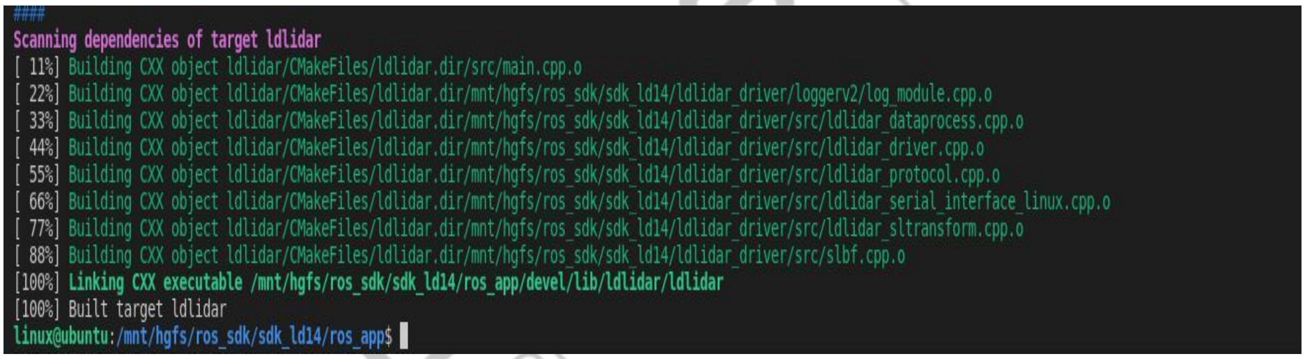
Firstly, connect the LiDAR to the computer via a serial-to-USB module. Then, in the Ubuntu system, open the terminal and inputs ls /dev/ttyUSB* to check if the serial port is connected. If the serial device is detected, use the command sudo chmod 777 /dev/ttyUSB* to grant it the highest permissions. This command grants read, write, and execute permissions to the file owner, group, and other users.

```
ls /dev/ttyUSB*
sudo chmod 777 /dev/ttyUSB0
```

3) Compile remarks

```
$ cd <sdk software package root directory>
$ catkin_make
```

Successful compilation will result in the following interface:



```
####
Scanning dependencies of target ldlidar
[ 11%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/src/main.cpp.o
[ 22%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/loggerv2/log_module.cpp.o
[ 33%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/ldlidar_dataprocess.cpp.o
[ 44%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/ldlidar_driver.cpp.o
[ 55%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/ldlidar_protocol.cpp.o
[ 66%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/ldlidar_serial_interface_linux.cpp.o
[ 77%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/ldlidar_sltransform.cpp.o
[ 88%] Building CXX object ldlidar/CMakeFiles/ldlidar.dir/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/slbf.cpp.o
[100%] Linking CXX executable /mnt/hgfs/ros_sdk/sdk_ld14/ros_app/devel/lib/ldlidar/ldlidar
[100%] Built target ldlidar
linux@ubuntu:/mnt/hgfs/ros_sdk/sdk_ld14/ros_app$
```

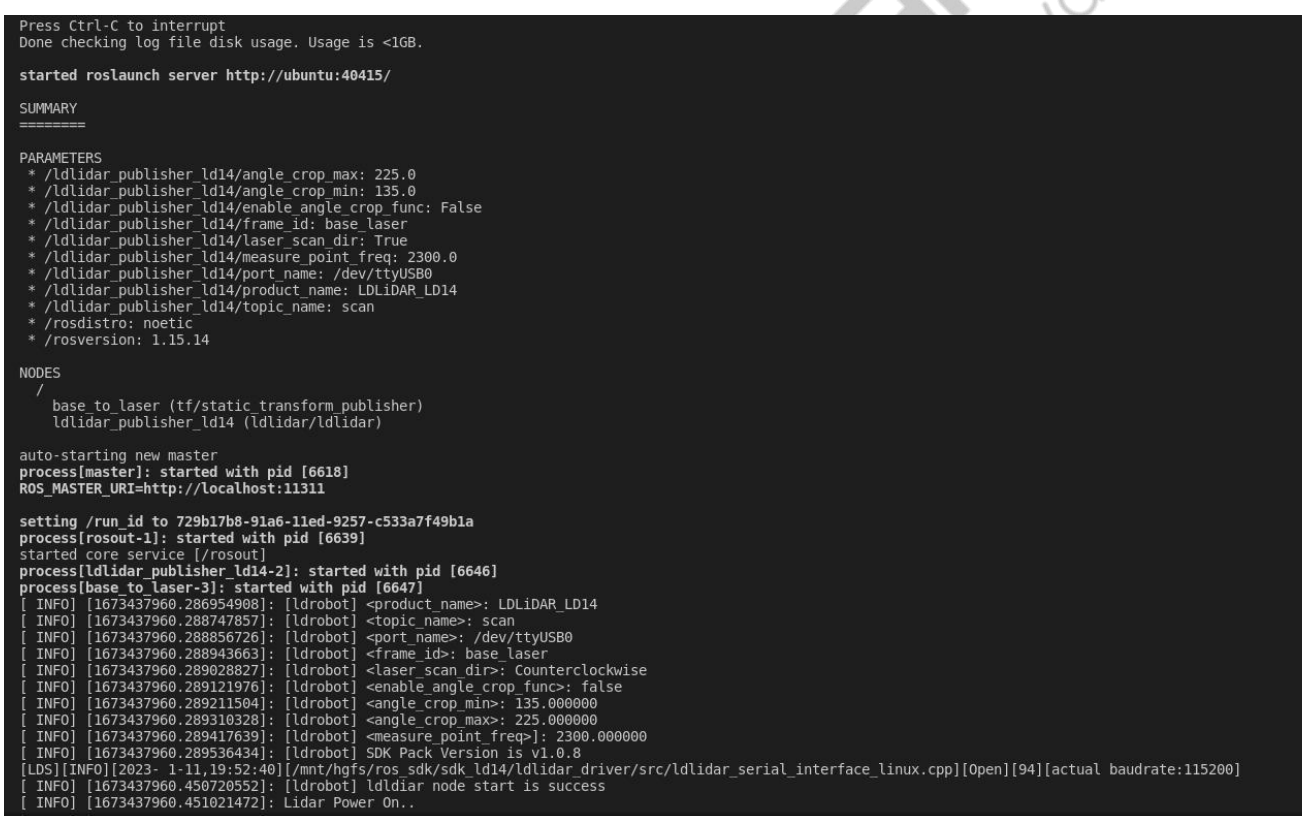
4) Run the corresponding node:

After the compilation is complete, it's necessary to add the compiled files to the environment variables using the command source devel/setup.bash. This command temporarily adds the environment variables to the terminal. This means that if you open a new terminal, you'll also need to navigate to the 'sdk_ld14' path and execute the command to add the environment variables.

Once the environment variables are added, the 'roslaunch' command will be able to locate the respective ROS package and launch files. Run the command roslaunch ldlidar ld14.launch.

```
$ cd <sdk software package root directory>
$ source devel/setup.bash
# Start the device node
$ roslaunch ldlidar ld14.launch
# or
# Start and display data on Rviz, suitable for ubuntu20.04 noetic
$ roslaunch ldlidar viewer_ld14_noetic.launch
# Start and display data on Rviz, suitable for ubuntu16.04 kineitic
# ubuntu18.04melodic$ source devel/setup.bash
$ roslaunch ldlidar viewer_ld14_kinetic_melodic.launch #
```

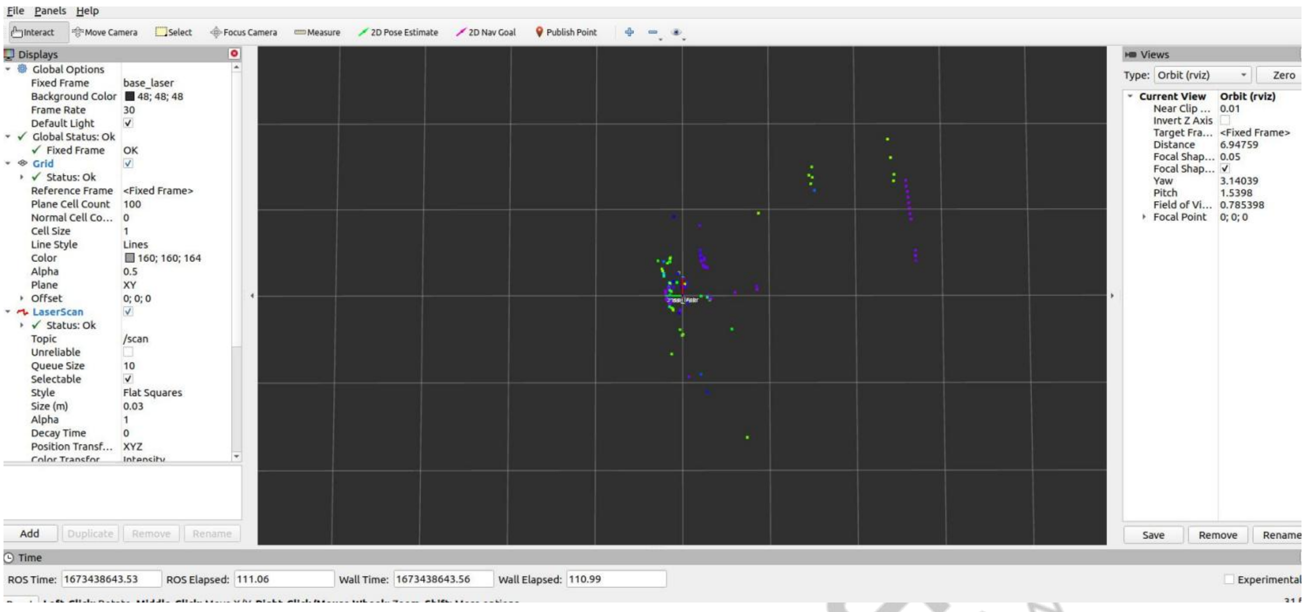
The operation interface is shown below:



```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:40415/

SUMMARY
========

PARAMETERS
 * /ldlidar_publisher_ld14/angle_crop_max: 225.0
 * /ldlidar_publisher_ld14/angle_crop_min: 135.0
 * /ldlidar_publisher_ld14/enable_angle_crop_func: False
 * /ldlidar_publisher_ld14/frame_id: base_laser
 * /ldlidar_publisher_ld14/laser_scan_dir: True
 * /ldlidar_publisher_ld14/measure_point_freq: 2300.0
 * /ldlidar_publisher_ld14/port_name: /dev/ttyUSB0
 * /ldlidar_publisher_ld14/product_name: LDLiDAR_LD14
 * /ldlidar_publisher_ld14/topic_name: scan
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES
  /
    base_to_laser (tf/static_transform_publisher)
    ldlidar_publisher_ld14 (ldlidar/ldlidar)

auto-starting new master
process[master]: started with pid [6618]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 729b17b8-91a6-11ed-9257-c533a7f49b1a
process[rosout-1]: started with pid [6639]
started core service [/rosout]
process[ldlidar_publisher_ld14-2]: started with pid [6646]
process[base_to_laser-3]: started with pid [6647]
[ INFO] [1673437960.286954908]: [ldrobot] <product_name>: LDLiDAR_LD14
[ INFO] [1673437960.288747857]: [ldrobot] <topic_name>: scan
[ INFO] [1673437960.288856726]: [ldrobot] <port_name>: /dev/ttyUSB0
[ INFO] [1673437960.288943663]: [ldrobot] <frame_id>: base_laser
[ INFO] [1673437960.289028827]: [ldrobot] <laser_scan_dir>: Counterclockwise
[ INFO] [1673437960.289121976]: [ldrobot] <enable_angle_crop_func>: false
[ INFO] [1673437960.289211504]: [ldrobot] <angle_crop_min>: 135.000000
[ INFO] [1673437960.289310328]: [ldrobot] <angle_crop_max>: 225.000000
[ INFO] [1673437960.289417639]: [ldrobot] <measure_point_freq>]: 2300.000000
[ INFO] [1673437960.289536434]: [ldrobot] SDK Pack Version is v1.0.8
[LDS][INFO][2023- 1-11,19:52:40][/mnt/hgfs/ros_sdk/sdk_ld14/ldlidar_driver/src/ldlidar_serial_interface_linux.cpp][Open][94][actual baudrate:115200]
[ INFO] [1673437960.450720552]: [ldrobot] ldldiar node start is success
[ INFO] [1673437960.451021472]: Lidar Power On..
```

5) Rviz display remarks:

Rviz is a commonly used 3D visualization tool in ROS, and supports displaying the LiDAR data. After successfully running roslaunch, open a new terminal and type rosrun rviz rviz. Then, click on file -> open -> Config, and then select sdk_ld14_ros/rviz/ldlidar.rviz file.



Displays

Global Options	
Fixed Frame	base_laser
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
✓ Status: Ok	
Reference Frame	<Fixed Frame>
Plane Cell Count	100
Normal Cell Co...	0
Cell Size	1
Line Style	Lines
Color	▨ 160; 160; 164
Alpha	0.5
Plane	XY
Offset	0; 0; 0
LaserScan	✓
✓ Status: Ok	
Topic	/scan
Unreliable	☐
Queue Size	10
Selectable	✓
Style	Flat Squares
Size (m)	0.03
Alpha	1
Decay Time	0
Position Transf...	XYZ
Color Transfor	Intensity

Views

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	☐
Target Fra...	<Fixed Frame>
Distance	6.94759
Focal Shap...	0.05
Focal Shap...	✓
Yaw	3.14039
Pitch	1.5398
Field of Vi...	0.785398
Focal Point	0; 0; 0

Time

ROS Time: 1673438643.53 ROS Elapsed: 111.06 Wall Time: 1673438643.56 Wall Elapsed: 110.99 ☐ Experimental

5. REVISION

Version	Revision Date	Contents
0.1	2022-10-19	Initial creation
0.2	2023-01-11	Reformatting, optimized relevant descriptions