Lidar Sensor STL-19P (FHL-LD19)

Development Manual V1.00

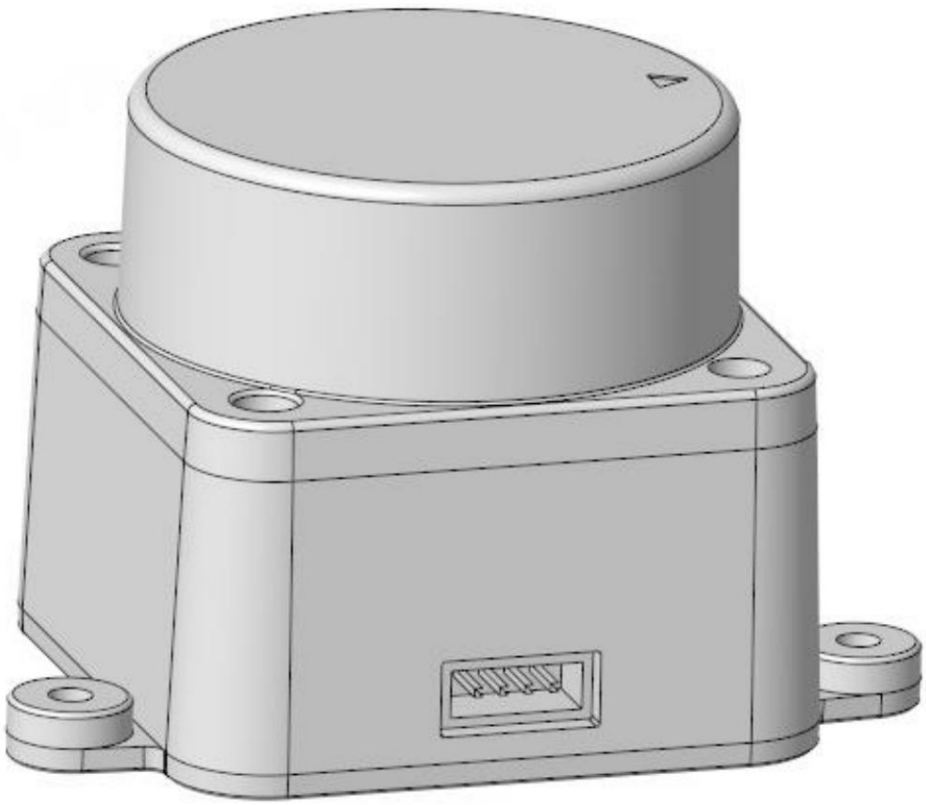


Table of contents

1. Working mechanism

1.1. Communication interface

STL-19P uses ZH1.5T-4P 1.5mm connector to connect to external systems to achieve power supply, data reception and external control.

See the following figure/table for specific interface definitions and parameter requirements:

Serial number	Signal name	Type	Description	Minimum value	Typical value	Maximum value
1	TX	output	radar data output 0V		3.3V	3.6V
2	PWM	input	motor control signal 0V		-	3.6
3	GND		negative pole of power supply	-	0V	-
4	P5V		power supply positive pole	4.5V	5V	5.5V

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

The data communication of STL-19P uses standard asynchronous serial port (UART) for one-way transmission. Its transmission parameters are as shown in the following table:

baud rate	Data length	Stop bit	Parity bit	flow control
230400bit/s	8 Bits	1	none	none

1.2.Workflow

The STL-19P distance measurement core uses DTOF technology and can perform distance measurement 5000 times per second.

The infrared laser is emitted from the front, and the laser is reflected to the single photon receiving unit after encountering the target object. From this, we obtained the inspiration

The time difference between the emission time of light and the time when the single photon receiving unit receives the laser is the flight time of light. The flight time

Combined with the speed of light, the distance can be calculated. After obtaining the distance data, the STL-19P will fuse the angle measured by the angle measurement unit

The values are composed into point cloud data, which is then sent to an external interface via internal wireless communication.

After the radar is started, the communication system is first initialized and adjusted. This process lasts about 500ms; then the radar is sent out.

Basic information is obtained, ranging is started and ranging data is sent out, angle measurement is initialized simultaneously, and the initialization is completed after about 200ms.

initialization, at this time the radar continues to output valid point cloud data.

The radar design has three modes, internal closed-loop speed control mode, external speed control mode and standby mode. Through radar PWM

Pins can be switched and controlled freely;

Internal closed-loop speed control mode: After the radar is powered on, it defaults to the internal closed-loop speed control mode and scans at a frequency of 10Hz by default.

If the radar works in other modes, the user can input the frequency 3KHz~5KHz (recommended 4KHz) and duty cycle through the PWM pin

45%-55% (recommended 50%), the signal duration is greater than 100ms, switch to the internal speed control mode.

External speed control mode: When the PWM pin input frequency is 0.5KHz~1.5KHz (1KHz is recommended), the duty cycle is 45%-55%

(50% recommended), after the signal duration is greater than 100ms, the radar will switch to the external speed control mode. At this time, the radar continues to

Capture external speed control signals (frequency 0.5KHz~1.5KHz, duty cycle 15%~84%), and capture the signal duty cycle information

It is directly output to the motor control terminal, and the external system performs closed-loop control by receiving the speed information fed back in the radar protocol.

Standby mode: When the PWM pin input frequency is 6KHz~8KHz (7KHz recommended), duty cycle 45%-55% (recommended)

50%) and the duration is greater than 100ms, the radar will switch to standby mode. At this time, the radar will turn off the motor and transmit power.

Wait for power to be restored.

When the radar is working, it will monitor the external power supply. When an abnormal power supply is detected (such as the voltage is too low or too high), the radar will

Start protection, shut down the motor, transmit power, and restart after the power supply returns to normal.

The radar working mode switching flow chart is as follows:

工作模式切换条件：

① ⑤

a、输入PWM频率：0.5~1.5kHz(推荐1kHz)
b、占空比：45%~55%(推荐50%)
c、最少100ms输入时间

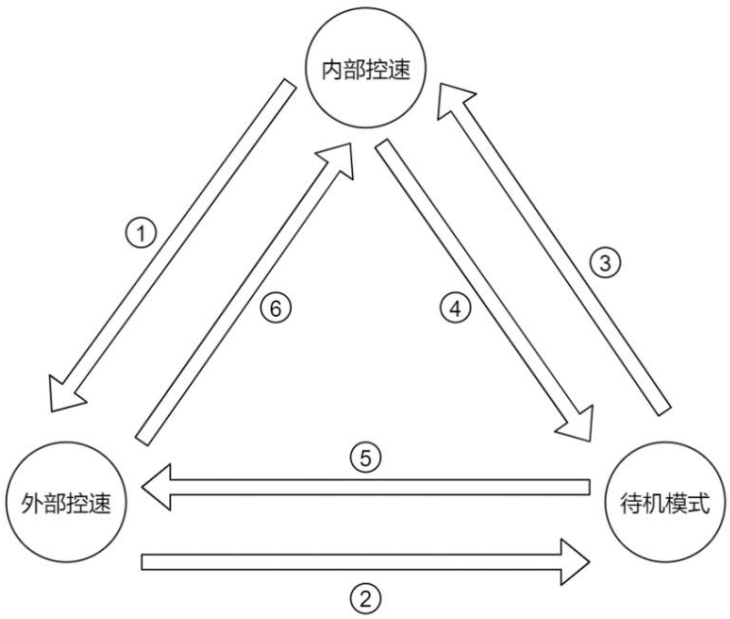
③ ⑥

a、输入PWM频率：3~5kHz(推荐4kHz)
b、占空比：45%~55%(推荐50%)
c、最少100ms输入时间

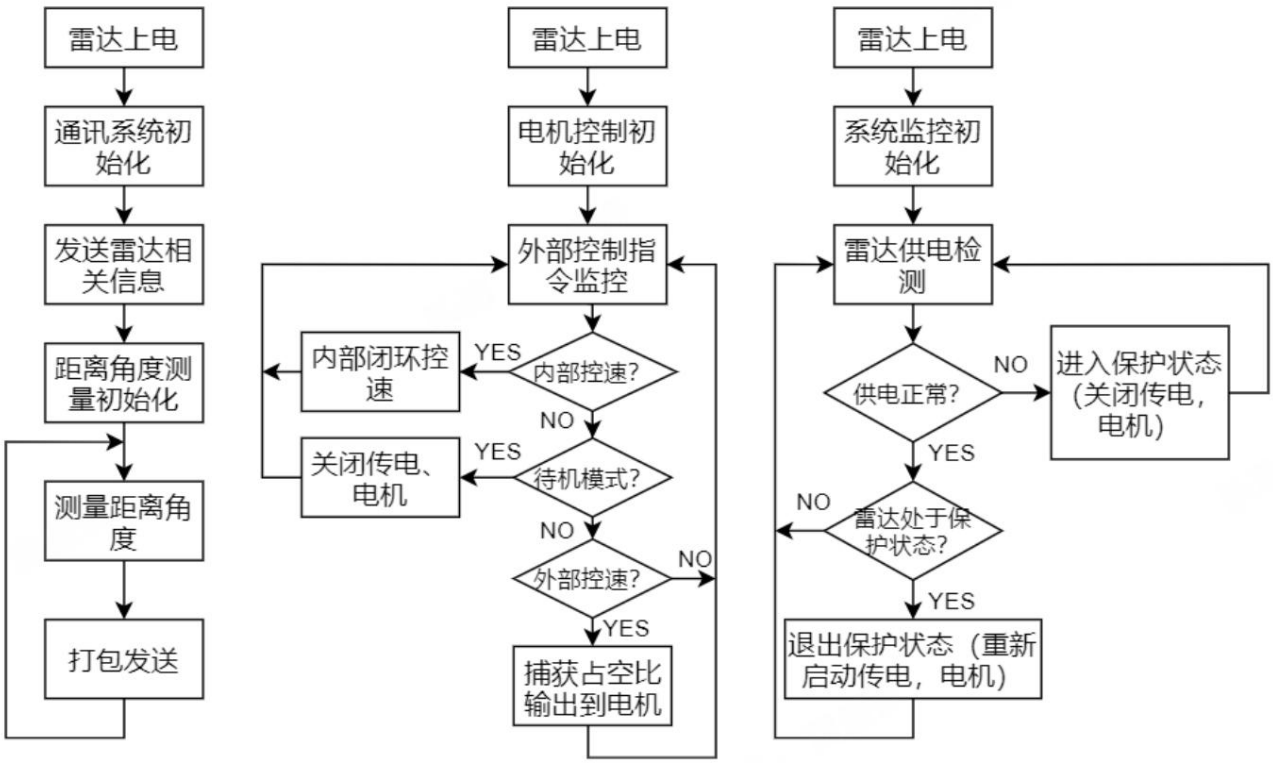
② ④

a、输入PWM频率：6~8kHz(推荐7kHz)
b、占空比：45%~55%(推荐50%)
c、最少100ms输入时间

注意：
外部控速模式下，输入PWM频率在0.5~1.5kHz区间内，通过调节占空比来进行转速控制，占空比需在15%~84%区间内



The radar workflow is shown in the figure below:



am2@youyeetoo.com

2. Communication Protocol

2.1.**Data** packet format

Start symbol	**frame information**	**Radar speed**		**starting angle**		**measurement data**				**ending angle**		**timestamp**		**check digit**
1Byte	1 Byte	2Bytes		2Bytes		12x3Bytes				2Bytes		2Bytes		1Byte
0x54	0x2C	LSB	MSB	LSB	MSB	…				LSB	MSB	LSB	MSB	CRC8

ÿStart character: length 1 Byte, value fixed at 0x54, indicating the beginning of the data packet;

ÿFrame **information: length** 1 Byte, value fixed at 0x2C, low five bits represent the number of measurement points of a packet, currently fixed at

12,;

ÿRadar **speed: length** 2 Byte, unit is degrees per second;

ÿStart **angle:** length 2 Byte, unit is 0.01 degree;

ÿMeasurement data: The length of a measurement data is 3 Byte, followed by distance LSB, distance MSB, and intensity information;

ÿEnd **angle: length** 2 Byte, unit is 0.01 degree;

ÿTimestamp : length 2 Byte, unit is ms, maximum is 30000, will be counted again when reaching 30000;

ÿ **CRC** check: CRC8 check of all previous data;

The data structure reference is 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;
```

The CRC check calculation method is as follows:

am2@youyeetoo.com

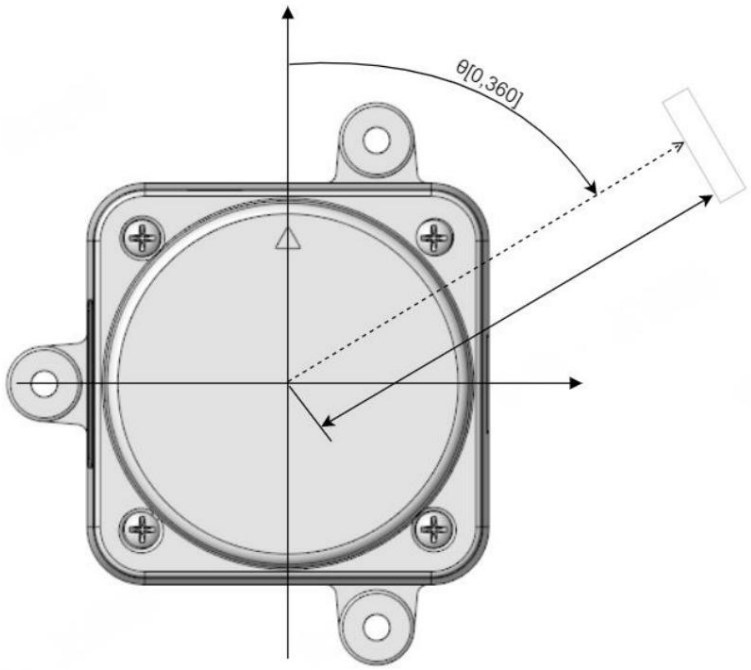
```
static const uint8_t CrcTable[256] = {

        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, 0x 90, 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, 0x7 a, 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, 0x 0b, 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;
}
```

am2@youyeetoo.com

3. Coordinate system

STL-19P uses a left-handed coordinate system, the rotation center is the origin of the coordinates, and the front of the sensor is defined as the zero-degree direction.

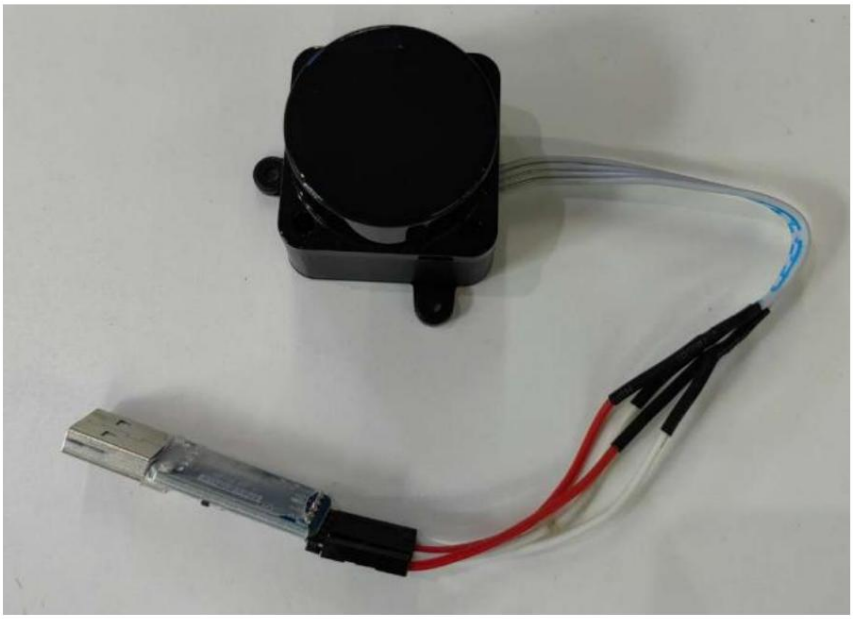
The angle increases in the clockwise direction, as shown in the figure below.



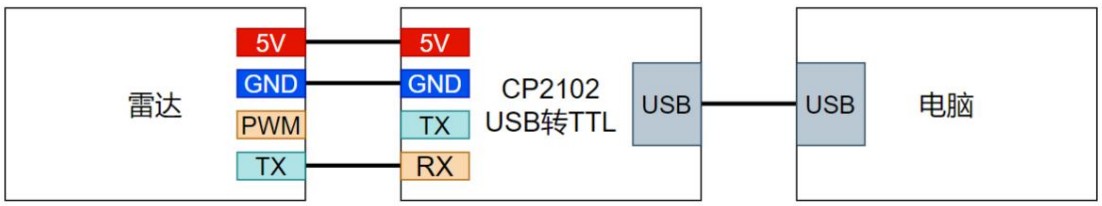
4. Development kit usage

4.1.Hardware cable connection and instructions

1) Products, cables, and USB adapter boards, as shown in the figure below:



2) Radar connection diagram, as shown in the figure below:



4.2. Driver installation under Windows

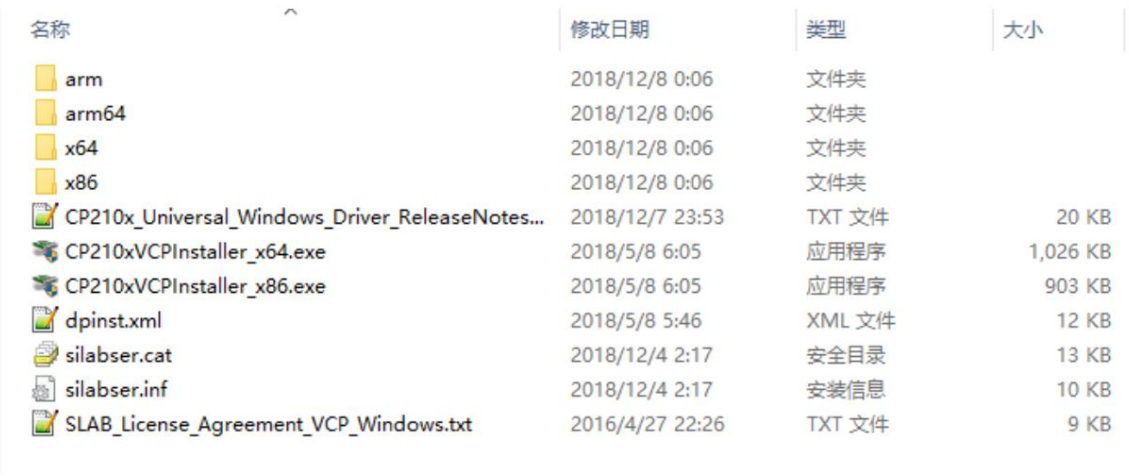
When evaluating our company's products under Windows, you need to install the adapter board driver. The adapter used by STL-19P

The connection board needs to support 230400 baud rate. The adapter board in the development kit provided by our company uses CP2102 USB to serial port signal

Chip, its driver can be downloaded from the official website of Silicon Labs:

https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

Unzip the CP210x_Universal_Windows_Driver driver package.



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

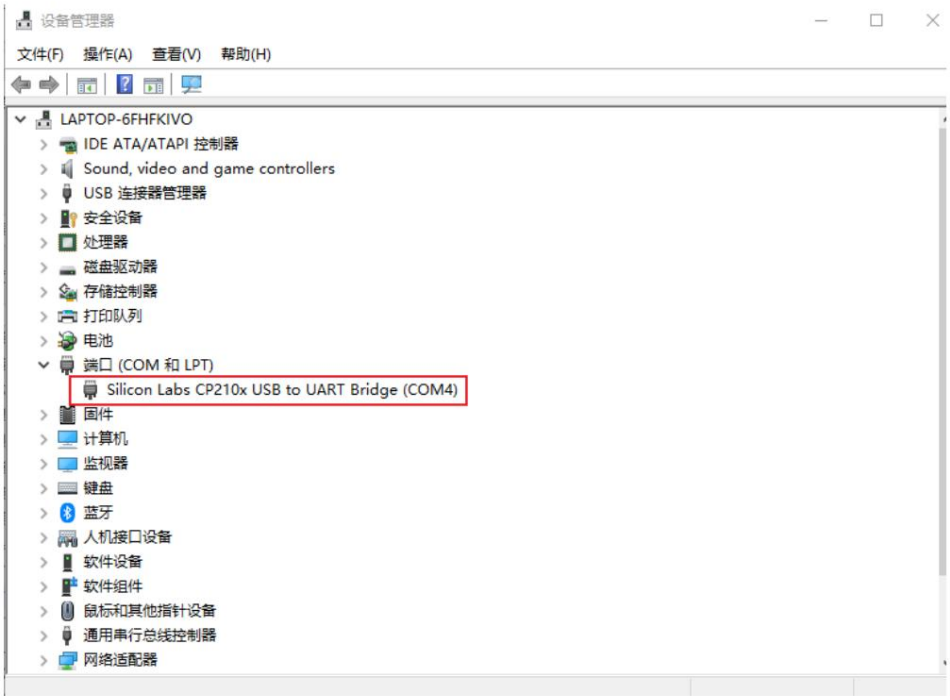
Depending on the Windows system version, double-click to execute CP210xVCPInstaller_x64.exe in the driver installation package directory.

Or CP210xVCPInstaller_x86.exe, follow the prompts to install.

After the installation is completed, connect the adapter board in the development kit to the computer. You can right-click [My Computer] and select [Properties].

Properties], in the opened [System] interface, select [Device Manager] in the left menu to enter the device manager, expand

[Port], you can see the serial port number corresponding to the recognized CP2102 adapter board, that is, the driver is installed successfully.



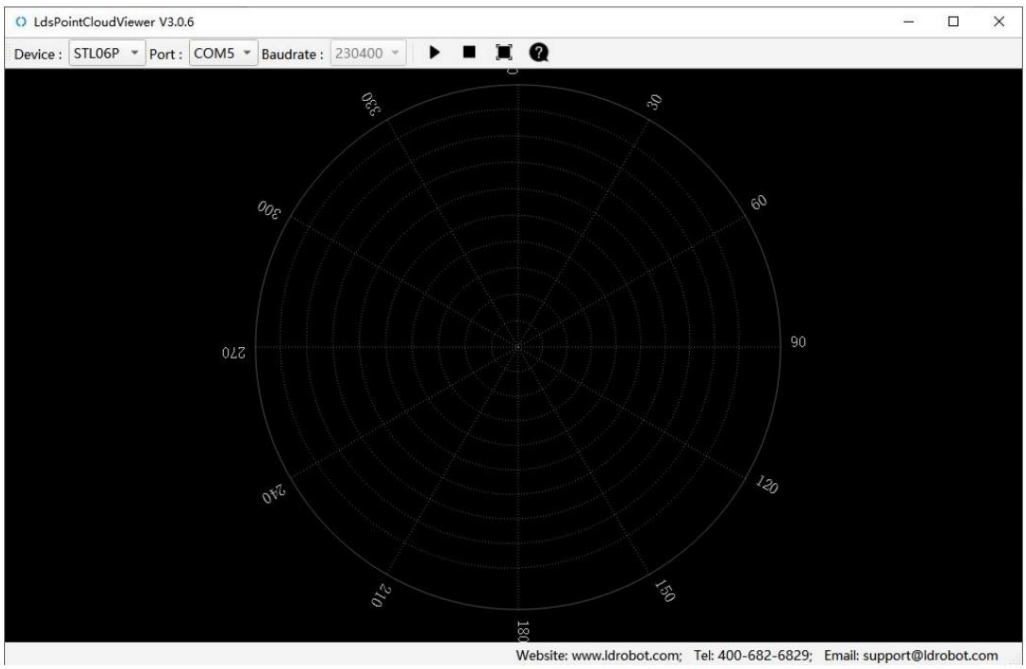
4.3. Use of host computer software

Our company provides point cloud visualization software LdsPointCloudViewer for real-time scanning of this product. Developers can use this

The software allows you to visually observe the scanning renderings of this product. Before using this software, make sure that the driver for the USB adapter board of this product is

The program has been installed successfully, and the product is connected to the USB port of the Windows PC. , Then double click

LdsPointCloudViewer.exe opens the software, and the software interface is as shown in the figure below.

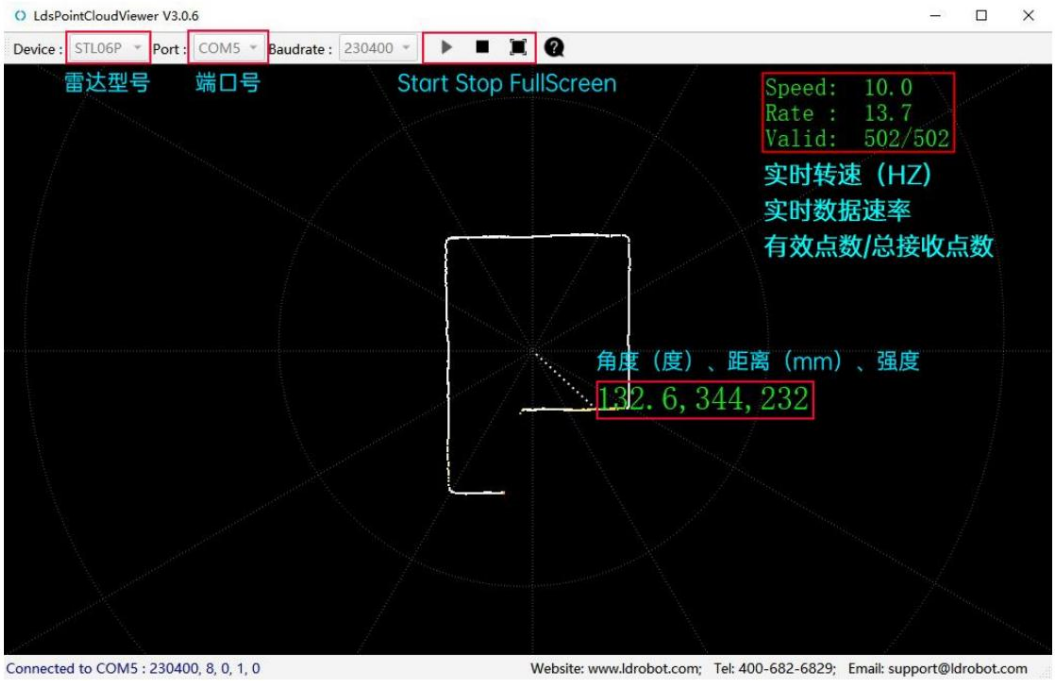


Select the corresponding product model (for example: STL06P, STL-19P model is shared with STL-06P) and the corresponding port number, click

The Start button starts the point cloud display. The endpoints of the indicator lines display corresponding angle point cloud information. The information is displayed in order of angle, distance,

Intensity information; click the FullScreen button to display in full screen, press the ESC key to exit full screen; click the Stop button to disconnect the upper screen

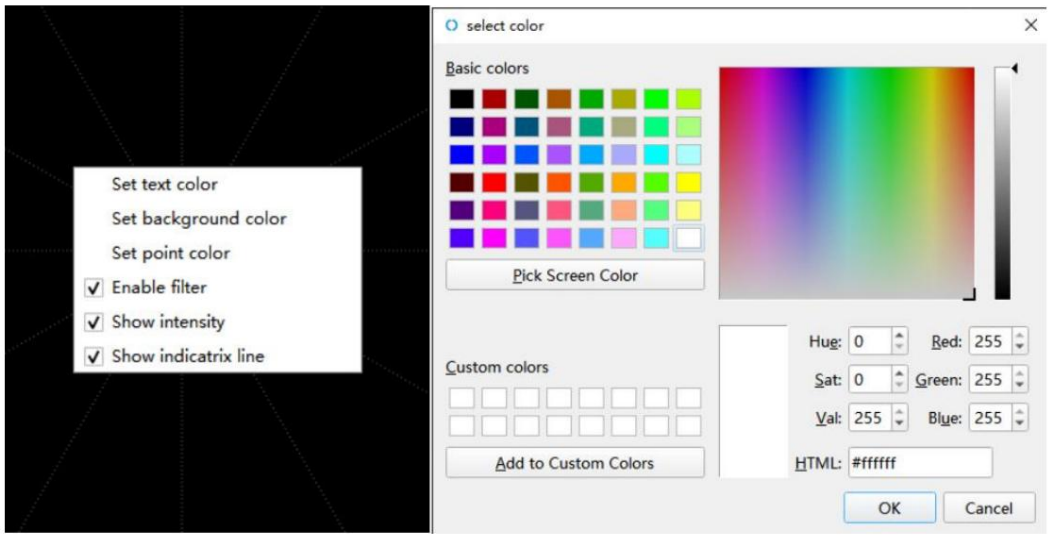
The connection between the machine and the port. As shown below.



Click the left button of the mouse on the cloud image interface to view point cloud information from different angles. Press and hold the right button of the mouse to drag the electronic display.

Position, scroll the mouse wheel to zoom in and out of the point cloud. Right-click to configure text, background, point cloud color and

Choose whether to turn on data filtering, intensity display, indicator line display and other functions.



4.4. Instructions for using **SDK** under Linux

4.4.1. Obtain **SDK** source code

For the Linux SDK source code of this product, you need to contact the company's FAE and other marketing personnel to obtain it. The README text in the source code

The file has relevant instructions for use. If you want to use private libraries for related development docking through hosting platforms such as Github or Gitee, the same

Please provide feedback to our company's FAE and other marketing personnel, and we will work hard to meet your development needs.

4.4.2. SDK Application Instructions

In order to facilitate users to quickly access the sensor, this product integrates data analysis, processing, packaging and other basics in the SDK

function, and is compatible with a variety of commonly used development platforms (Linux, ROS, ROS2); the functions currently integrated in the SDK are as follows:

1. Data serial port reception buffer processing;

2. Data packet protocol analysis and processing;

3. Data noise filtering;

5. Pack the data and provide corresponding interfaces to the upper layer.

4.4.3. SDK running and debugging instructions

1) SDK operating platform description

This example is based on the ROS platform under the Ubuntu operating system. Please make sure you have the correct Ubuntu installed before following the example.

and ROS environment, if you need to use the SDK to evaluate this product on other platforms; please refer to the corresponding platform in the SDK source code

README document.

2) Set sensor device permissions

First, connect the radar to the computer through the serial port to USB module. Then open the terminal under the Ubuntu system and enter ls

/dev/ttyUSB* Check whether the serial device is connected. If a serial device is detected, use sudo chmod 777

The /dev/ttyUSB* command gives it the highest permissions, that is, the file owner, group, and other users have read, write, and execute permissions.

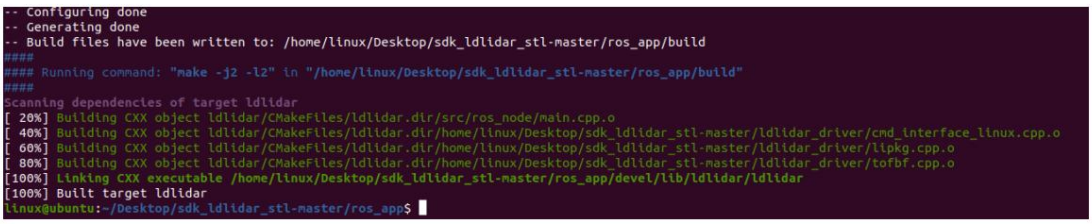
```
ls /dev/ttyUSB* sudo chmod

777 /dev/ttyUSB0
```

3) Compilation instructions

Enter the corresponding platform directory in the SDK source code and execute catkin_make to compile:

```
$ cd <sdk software package root directory>/<corresponding platform directory>

$catkin_make
```

If the compilation is successful, the following interface will appear:



4) Run the corresponding node instructions

After compilation is completed, you need to add the compiled file to the environment variable. The command is source devel/setup.bash. The command is

Temporarily adding environment variables to the terminal means that if you open a new terminal, you also need to enter the current compilation path.

Execute the add environment variable command.

After adding the environment variable, the roslaunch command can find the corresponding ros package and launch file and start the corresponding device.

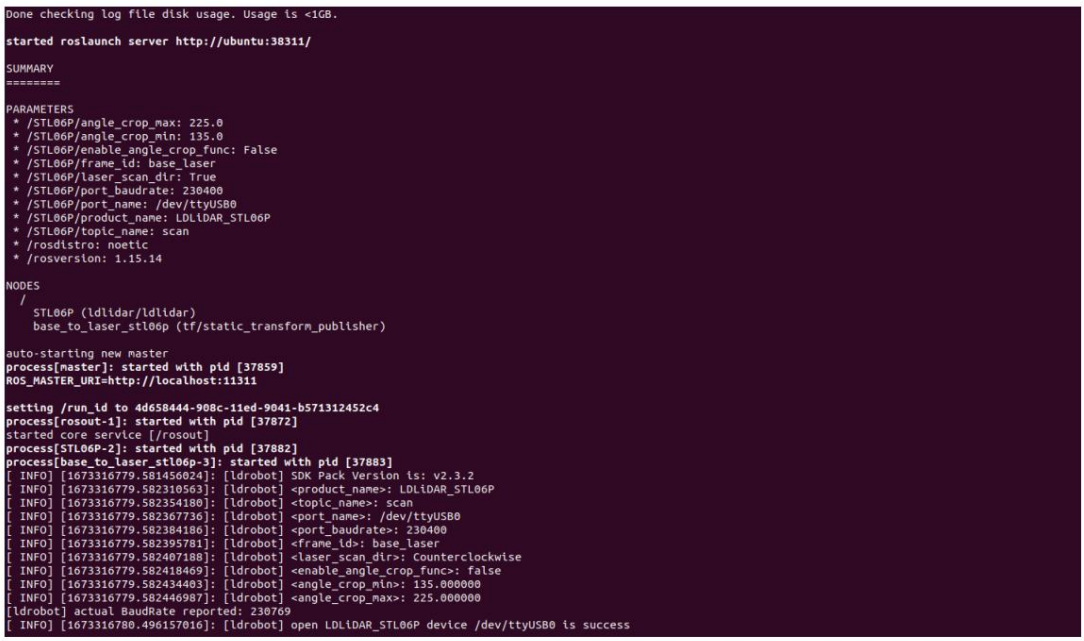
The node running command is roslaunch ldlidar stl06p.launch; if you need to view the point cloud visualization data, you can also directly execute the launch command.

To activate and display the command under Rviz, the corresponding platform command is as follows (for example, the command under ROS Noetic platform is roslaunch

ldlidar viewer_stl06p_noetic.launch).

```
$ cd <sdk software package root directory>/<corresponding platform directory>

$ source devel/setup.bash

# Start device node

$ roslaunch ldlidar stl06p.launch
# or


# Start and display data in Rviz, suitable for ubuntu20.04 noetic

$ roslaunch ldlidar viewer_stl06p_noetic.launch

# Start and display data in Rviz, suitable for ubuntu16.04 kineitc,

# ubuntu18.04melodic$ source devel/setup.bash $ roslaunch ldlidar
viewer_stl06p_kinetic_melodic..launch #
```

The running interface is as shown below:

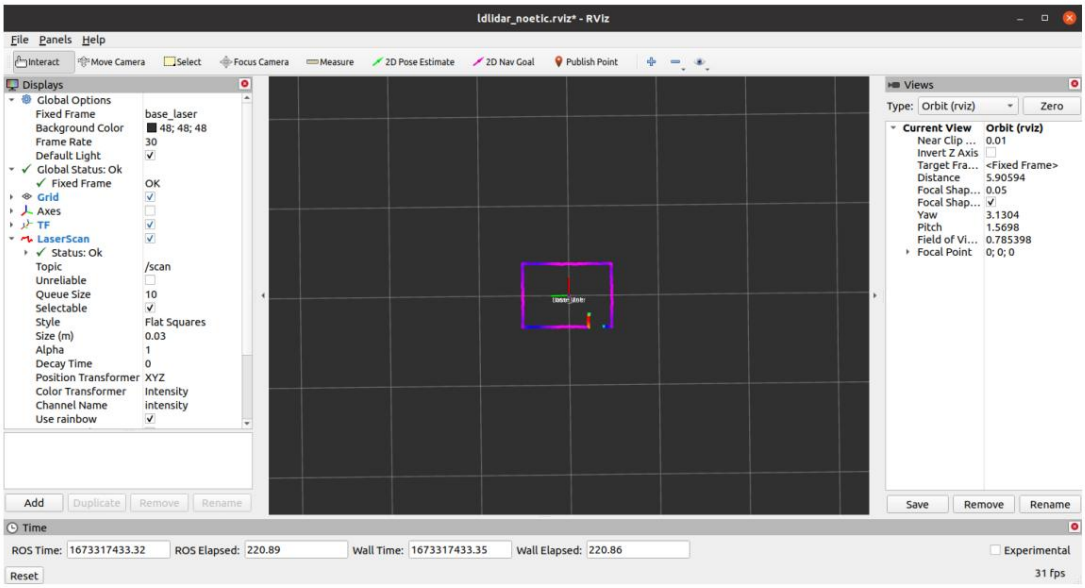


5) rviz display instructions

rviz is a commonly used 3D visualization tool under ROS, in which radar data can be displayed. Run in roslaunch

After success, open a new terminal, enter rosrun rviz rviz, click file->open->Config, and then select <sdk software

Package root directory>/ros_app/src/ldlidar/rvizrviz/<corresponding platform.rviz file>.



am2@youyeetoo.com

5. Revision records

Version	Revision date	modify the content
1.0	2023-10-30	initial creation

am2@youyeetoo.com