LiDAR STL-27L
Development Manual

Content:

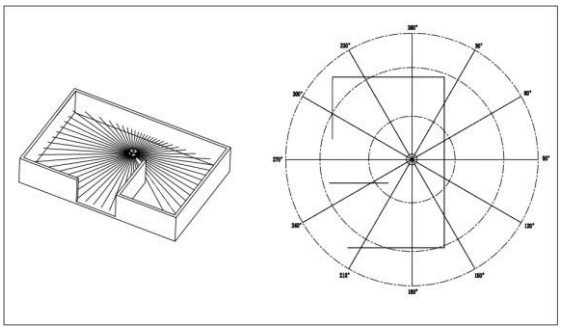
1. PRODUCT DESCRIPTION

The STL-27L is mainly composed of laser ranging core, wireless telex unit, wireless communication unit, angle measurement unit, motor drive unit and mechanical casing.

The STL-27L ranging core uses DTOF technology, which can measure 21,600 times per second. Each time the distance is measured, the STL-27L emits an infrared laser forward, and the laser is reflected to the single-photon receiving unit after encountering the target object. From this, we obtained the time when the laser was emitted and the time when the single-photon receiving unit received the laser. The time difference between the two is the time of flight of light. The time of flight can be combined with the speed of light to calculate the distance.

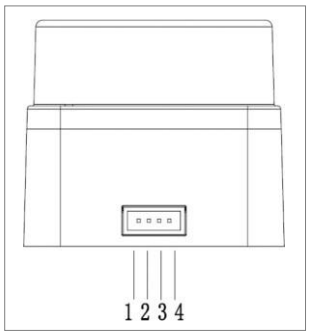
After obtaining the distance data, the STL-27L will combine the angle values measured by the angle measurement unit to form point cloud data, and then send the point cloud data to the external interface through wireless communication. STL-27L supports internal speed control, the speed can be stabilized to 10±0.1Hz within 3 seconds after power-on. At the same time, PWM external input interface is provided to support external speed control. After the external control unit obtains the speed, it is controlled by PID algorithm closed-loop, and the PWM signal is input to make the STL-27L reach the specified speed.

An illustration of the environmental scan formed by the STL-27L point cloud data is shown below:



2. COMMUNICATION INTERFACE

The STL-27L uses ZH1.5T-4P 1.5mm connector to connect with external system to realize power supply and data reception. The specific interface definition and parameter requirements are shown in the following figure/table:



port number	signal name	type	description	minimum	typical	maximum
1	Tx	output	LiDAR data output	0V	3.3V	3.5V
2	PWM	input	motor control	0V	-	3.3V
3	GND	power supply	negative	-	0V	-
4	P5V	power supply	positive	4.5V	5V	5.5V

The STL-27L has a motor driver with stepless speed regulation, which supports internal speed control and external speed control. When the PWM pin is grounded, the default is internal speed regulation, and the default speed is 10±0.1Hz. For external speed control, a square wave signal needs to be connected to the PWM pin, and the start, stop and speed of the motor can be controlled through the duty cycle of the PWM signal. Conditions for triggering external speed control: a. Input PWM frequency 20-50K, recommended 30K; b. Duty cycle is within (45%, 55%) interval (excluding 45% and 55%), and at least 100ms continuous input time. After the external speed control is triggered, it is always in the external speed control state, and the internal speed control will be restored unless the power is turned off and restarted; at the same time, the speed control can be performed by adjusting the PWM duty cycle. Due to the individual differences of each product motor, the actual speed may be different when the duty cycle is set to a typical value. To accurately control the motor speed, it is necessary to perform closed-loop control according to the speed information in the received data. **Note: When not using external speed control, the PWM pin must be grounded.**

The data communication of STL-27L adopts standard universal asynchronous serial port (UART) one-way transmission, and its transmission parameters are shown in the following table:

baud rate	data length	stop bit	parity bit	flow control
921600bit/s	8 Bits	1	none	none

3. DATA PROTOCOL

3.1. Data packet format

The STL-27L adopts one-way communication. After stable operation, it starts to send measurement data packets without sending any commands. The measurement packet format is shown in the figure below.

Header	VerLen	Speed		Start angle		Data	End angle		Timestamp		CRC check
54H	1 Byte	LSB	MSB	LSB	MSB	……	LSB	MSB	LSB	MSB	1 Byte

- ***Header***：The length is 1 Byte, and the value is fixed at 0x54, indicating the beginning of the data packet;
- ***VerLen***：The length is 1 Byte, the upper three bits indicate the packet type, which is currently fixed at 1, and the lower five bits indicate the number of measurement points in a packet, which is currently fixed at 12, so the byte value is fixed at 0x2C;
- ***Speed***：The length is 2 Byte, the unit is degrees per second, indicating the speed of the lidar;
- ***Start angle***: The length is 2 Bytes, and the unit is 0.01 degrees, indicating the starting angle of the data packet point;
- ***Data***：Indicates measurement data, a measurement data length is 3 bytes, please refer to the next section for detailed analysis;
- ***End angle***：The length is 2 Bytes, and the unit is 0.01 degrees, indicating the end angle of the data packet point;
- ***Timestamp***：The length is 2 Bytes, the unit is milliseconds, and the maximum is 30000. When it reaches 30000, it will be counted again, indicating the timestamp value of the data packet;

- ***CRC check***：The length is 1 Byte, obtained from the verification of all the previous data except itself. For the CRC verification method, see the following content for details;

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:

```
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,
```

```
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;
}
```

3.2. Measurement data analysis

Each measurement data point consists of a 2-byte distance value and a 1-byte confidence value, as shown in the figure below.

Header	VerLen	Speed		Start angle		Data	End angle		Timestamp		CRC check
54H	2CH	LSB	MSB	LSB	MSB	……	LSB	MSB	LSB	MSB	1Byte

Measuring point 1			Measuring point 2			…	Measuring point n		
distance		intensity	distance		intensity		distance		intensity
LSB	MSB	1 Byte	LSB	MSB	1 Byte	…	LSB	MSB	1 Byte

The unit of distance value is mm. The signal intensity value reflects the light reflection intensity. The higher the intensity, the larger the signal intensity value; the lower the intensity, the smaller the signal intensity value.

The angle value of each point is obtained by linear interpolation of the starting angle and the ending angle. The angle calculation method is as follows:

step = (end_angle – start_angle)/(len – 1);

*angle = start_angle + step*i;*

where len is the number of measurement points in a data packet, and the value range of i is [0, len).

3.3. Example

Suppose we receive a piece of data as shown below.

54 2C 68 08 AB 7E E0 00 E4 DC 00 E2 D9 00 E5 D5 00 E3 D3 00 E4 D0 00 E9 CD 00 E4 CA 00 E2 C7 00 E9 C5 00 E5 C2 00 E5 C0 00 E5 BE 82 3A 1A 50

We analyze it as follows:

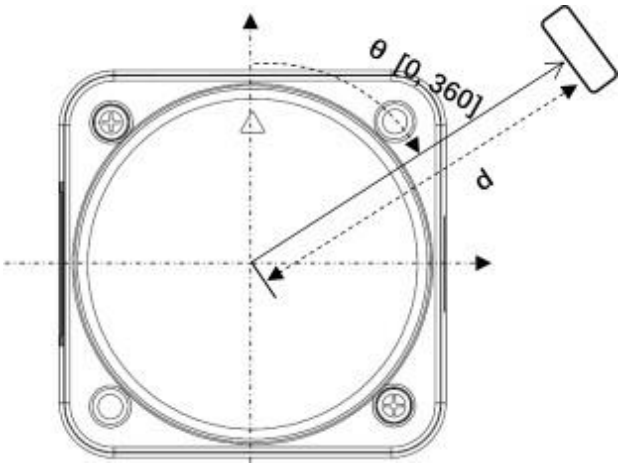
Header	VerLen	Speed		Start angle		Data	End angle		Timestamp		CRC check
54H	2CH	68H	08H	ABH	7EH	……	BEH	82H	3AH	1AH	50H

Measuring point 1			Measuring point 2			…	Measuring point 12		
distance		intensity	distance		intensity		distance		intensity
E0H	00H	E4H	DCH	00H	E2H	…	B0H	00H	EAH

Field information	Parsing process
Speed	0868H = 2152 degrees per second;
Start angle	7EABH = 32427, or 324.27 degrees;
End angle	82BEH = 33470, or 334.7 degrees;
Measuring point 1 distance	00E0H = 224mm
Measuring point 1 intensity	E4H = 228
Measuring point 2 distance	00DCH = 200mm
Measuring point 2 intensity	00E2H = 226
…	…
Measuring point 12 distance	00B0H = 176mm
Measuring point 12 intensity	EAH = 234

4. COORDINATE SYSTEM

The STL-27L uses a left-handed coordinate system, the rotation center is the coordinate origin, the front of the sensor is defined as the zero-degree direction, and the rotation angle increases clockwise, as shown in the figure below.



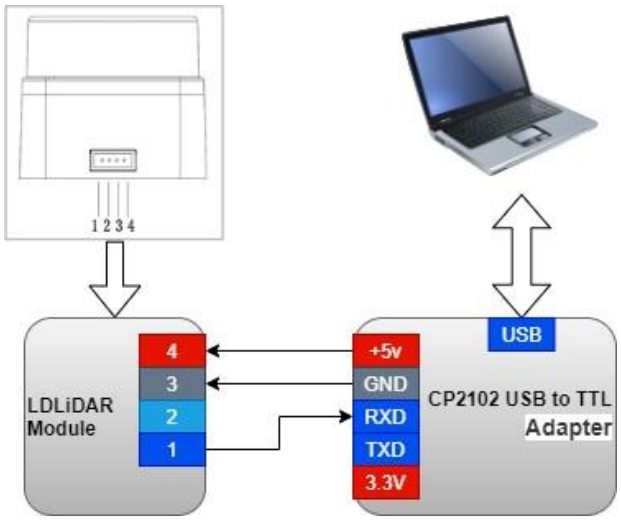
5. DEVELOPMENT KIT INSTRUCTIONS

5.1. How to use the assessment tool

5.1.1. Hardware cable connection and description

1) LiDAR, wire, USB adapter board, as shown in the following figure:

2) Connection diagram, as shown in the figure below:



5.1.2. Driver installation under Windows

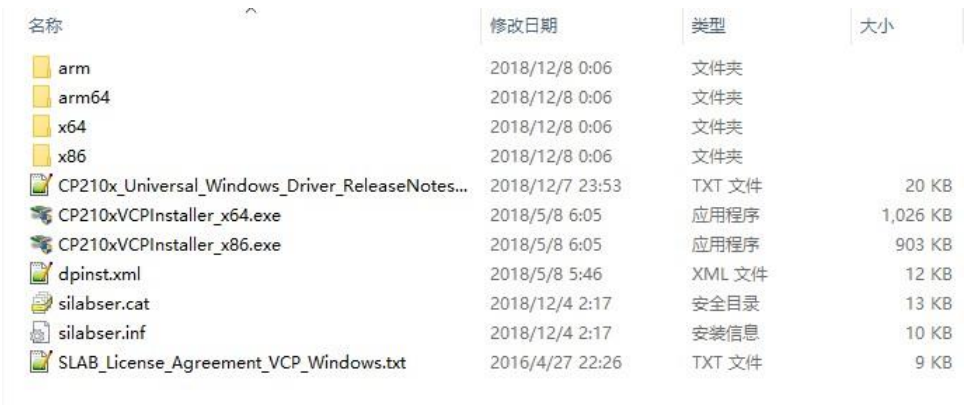
When evaluating the company's products under Windows, it is necessary to install the serial port driver of the USB adapter board. The reason is that the USB adapter board in the development kit provided by the company adopts the CP2102 USB to serial port adapter chip, and its driver can be obtained from Silicon Download from Labs' official website:

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

Or download it from the company's repository:

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

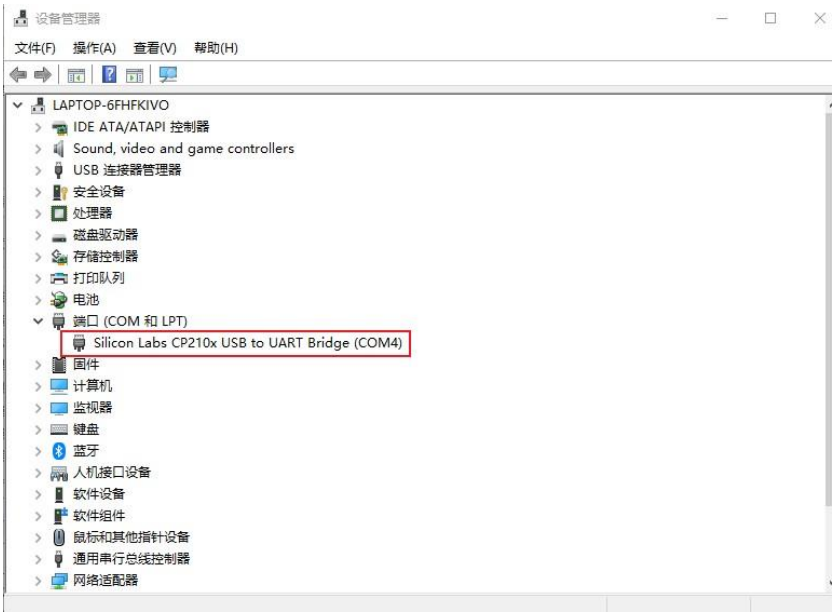
After decompressing the CP210x_Universal_Windows_Driver driver package, execute the exe file in the driver installation package directory, and select X86 (32-bit) or X64 (64-bit) according to the Windows system version.



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

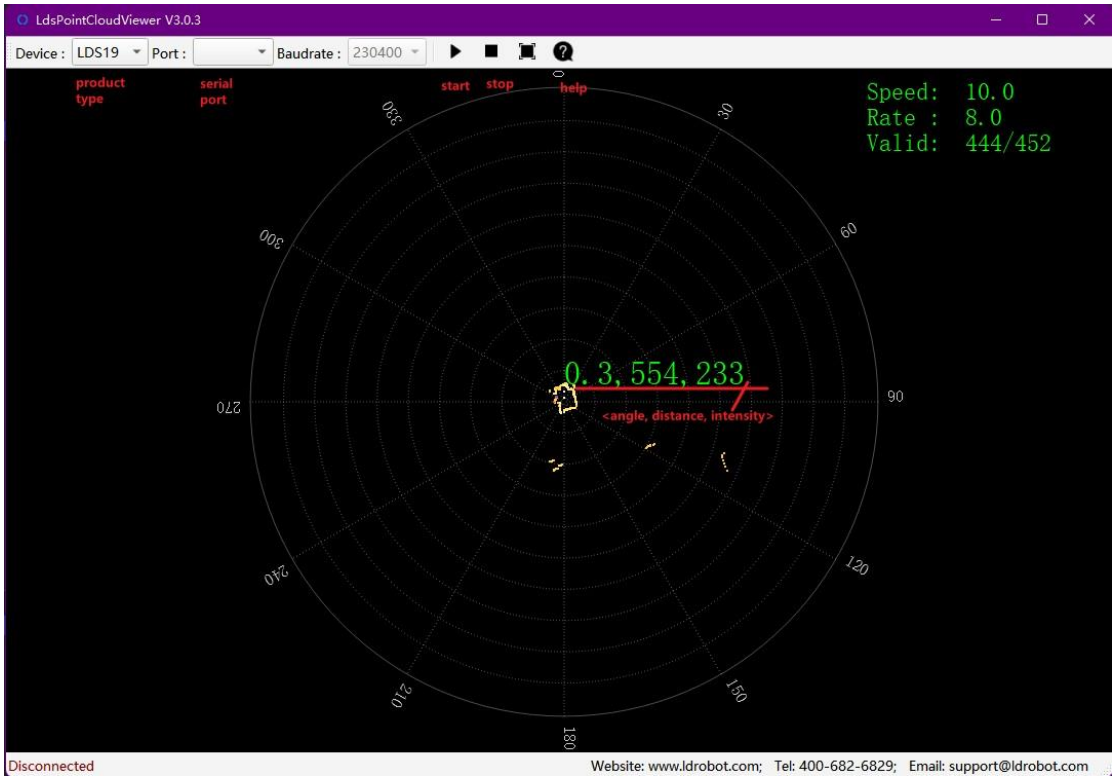
Double-click the exe file and follow the prompts to install it.

After the installation is complete, connect the USB adapter board in the development kit to the computer, right-click [My Computer], select [Properties], and in the opened [System] interface, select [Device Manager] in the left menu to enter Go to the device manager, expand [Ports], you can see the serial port number corresponding to the recognized CP2102 USB adapter, that is, the driver is installed successfully, and the figure below is COM4.



5.1.3. Using LdsPointCloudViewer software under Windows

The point cloud visualization software LdsPointCloudViewer can display the scanned data of this product in real time, and developers can use this software to visually observe the scanning renderings of this product. Before using this software, it is necessary to distinguish that the driver of the USB adapter board of this product has been installed successfully, and the product is interconnected with the USB port of the Windows system PC, then double-click the LdsPointCloudViewer.exe, and select the corresponding product model and port number, click the Start point cloud refresh button, as shown in the following figure.



In the above figure, 'Speed' represents the lidar scanning frequency, unit: Hz; 'Rate' represents the lidar data packet resolution rate; 'Valid' represents the valid point for the lidar to measure a circle.

The LdsPointCloudViewer software binary package can be downloaded from the company's repository:

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

5.2. Operation based on ROS under Linux

5.2.1. ROS environment introduction and installation

ROS (Robot Operating System) is an open source meta-operating system for robots and middleware built on Linux. It provides the services expected of an operating system, including hardware abstraction, low-level device control, implementation of commonly used functions, message passing between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run code across computers. For the installation steps of each version of ROS, please refer to the official ROS website:

http://wiki.ros.org/ROS/Installation

The ROS function package of this product supports the following versions and environments:

- ROS Kinetic(Ubuntu16.04);
- ROS Melodic(Ubuntu18.04);
- ROS Noetic(Ubuntu20.04).

5.2.2. Get the source code of the ROS Package

The source code of the ROS function package of this product needs to be obtained by contacting the company's FAE and other marketing personnel. The README document in the source code has relevant instructions for use. If you want to use private libraries for related development and docking through hosting platforms such as Github or Gitee, please also give feedback to the company's FAE and other marketers, and we will try our best to meet your development needs.

5.3. Operation based on ROS2 under Linux

5.3.1. ROS2 environment introduction and installation

ROS (Robot Operating System) is an open source meta-operating system for robots and middleware built on Linux. It provides the services expected of an operating system, including hardware abstraction, low-level device control, implementation of commonly used functions, message passing between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run code across computers. The robotics and ROS community has changed a lot since ROS was launched in 2007. The goal of the ROS2 project is to adapt to these changes, leveraging the strengths of ROS1 and improving on the weaknesses. For the installation steps of ROS2, please refer to the official website of ROS2:https://docs.ros.org/en/foxy/Installation.html

The ROS2 function package of this product supports the use of the ROS2 foxy version and above.

5.3.2. Get the source code of ROS2 Package

The source code of the ROS2 function package of this product needs to be obtained by contacting the company's FAE and other marketing personnel. The README document in the source code has relevant instructions for use. If you want to use private libraries for related development and docking through hosting platforms such as Github or Gitee, please also give feedback to the company's FAE and other marketers, and we will try our best to meet your development needs.

5.4. Instructions for using SDK under Linux

5.4.1. Get the source code of SDK

The Linux SDK source code of this product needs to be obtained by contacting the company's FAE and other marketing personnel. The README document in the source code has relevant instructions for use. If you want to use private libraries for related development and docking through hosting platforms such as Github or Gitee, please also give feedback to the company's FAE and other marketers, and we will try our best to meet your development needs.

6. REVISION HISTORY

version	revision date	modify the content
0.1	2022-07-15	Initial creation
0.2	2022-08-08	Lidar model changed from STL-19P to STL-27L