
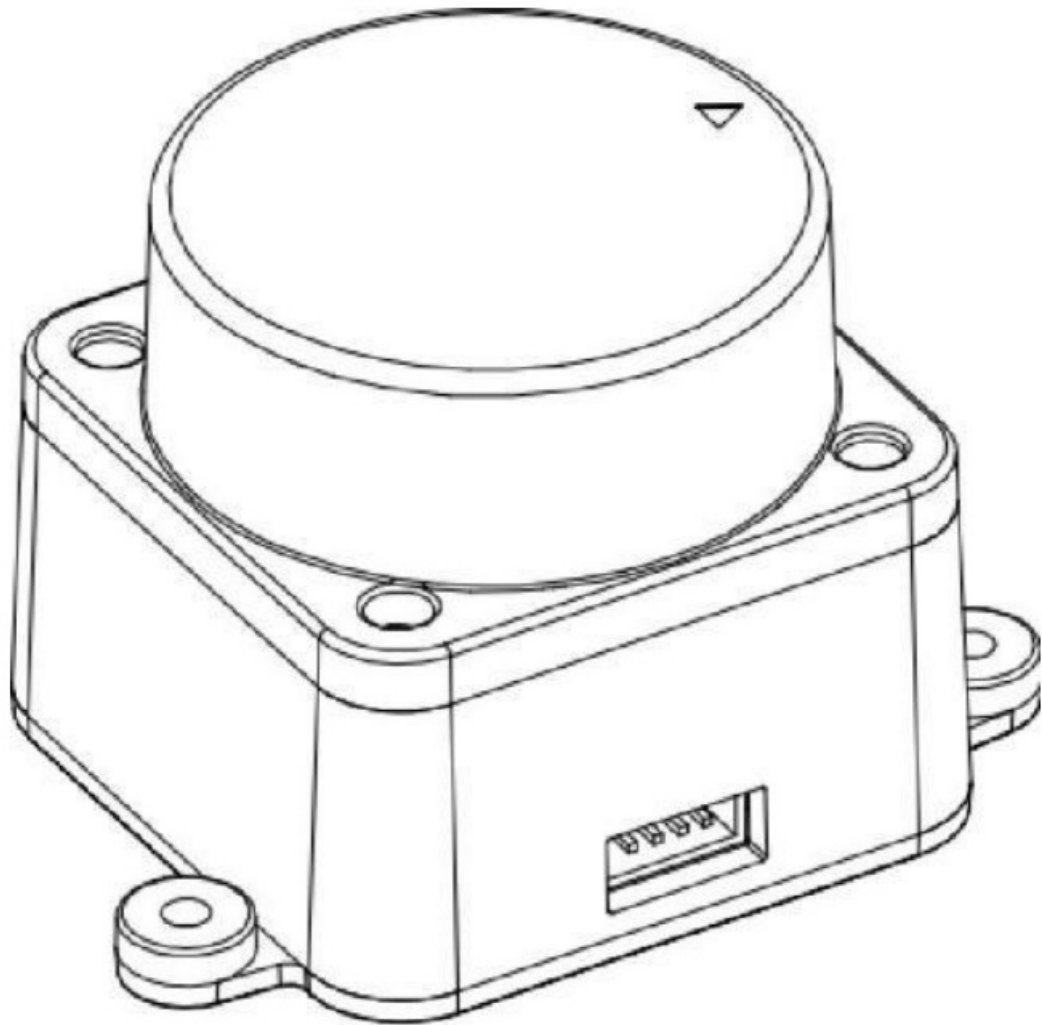




# DFRobot LiDAR LD19 Laser Sensor Kit Instruction Manual

[Home](#) » [DFROBOT](#) » DFRobot LiDAR LD19 Laser Sensor Kit Instruction Manual 

DFRobot LiDAR LD19 Laser Sensor Kit



## Contents

- [1 PRODUCT DESCRIPTION](#)
- [2 COMMUNICATION INTERFACE](#)
- [3 DATA PROTOCOL](#)
  - [3.1 Data packet format](#)
  - [3.2 Measurement data analysis](#)
- [4 Example](#)
- [5 COORDINATE SYSTEM](#)
- [6 DEVELOPMENT KIT INSTRUCTIONS](#)
  - [6.1 How to use the assessment tool](#)
  - [6.2 Product 3D model file](#)
  - [6.3 Operation based on ROS under Linux](#)
  - [6.4 Operation based on ROS2 under Linux](#)
  - [6.5 Instructions for using SDK under Linux](#)
  - [6.6 Instructions for using ROS based on Raspberry Pi SBC](#)
- [7 REVISION HISTORY](#)
- [8 Documents / Resources](#)
  - [8.1 References](#)
- [9 Related Posts](#)

## PRODUCT DESCRIPTION

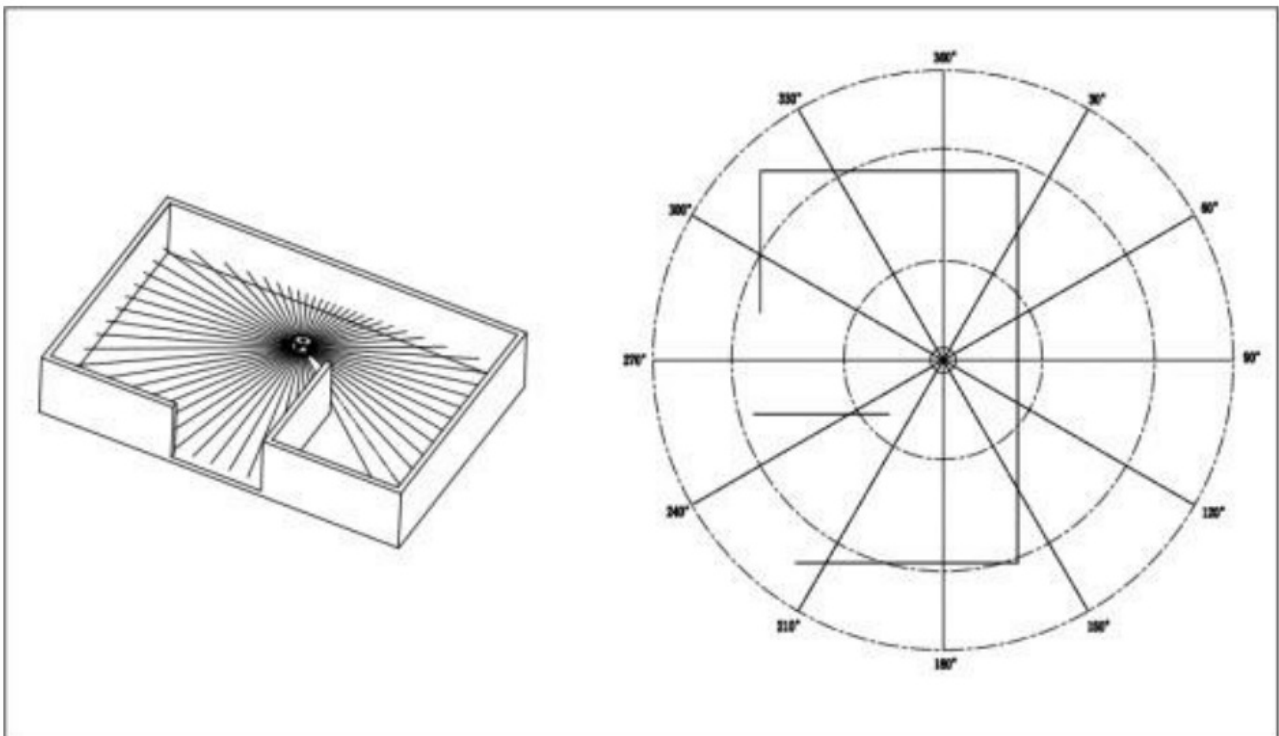
The LD19 is mainly composed of laser ranging core, wireless telex unit, wireless communication unit, angle

measurement unit, motor drive unit and mechanical casing.

The LD19 ranging core uses DTOF technology, which can measure 4,500 times per second. Each time the distance is measured, the LD19 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 LD19 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. LD19 supports internal speed control, the speed can be stabilized to  $10 \pm 0.1 \text{ Hz}$  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 LD19 reach the specified speed.

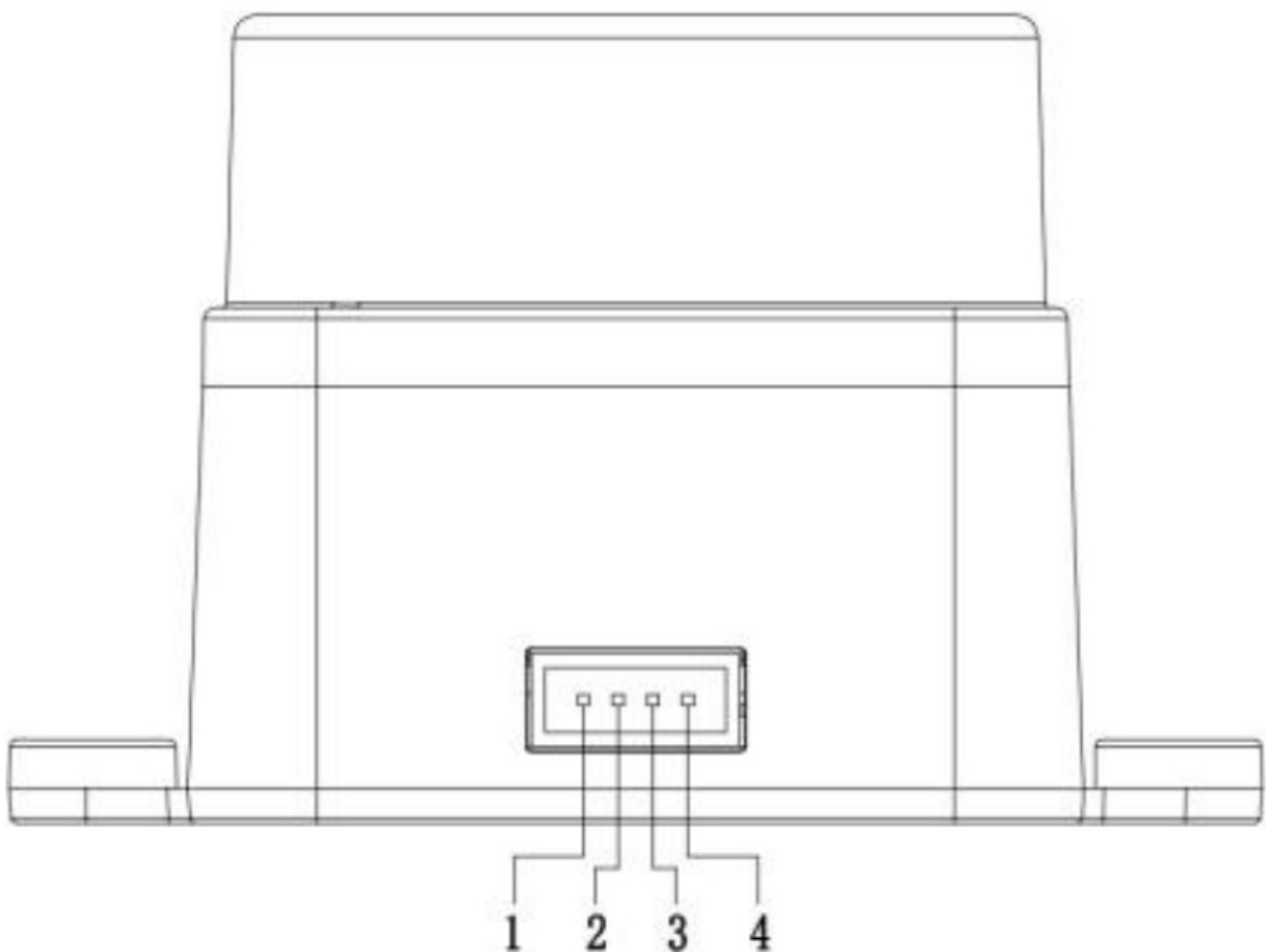
An illustration of the environmental scan formed by the LD19 point cloud data is shown below:



## COMMUNICATION INTERFACE

The LD19 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 LD19 has a motor driver with step less 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  $10 \pm 0.1$  Hz. 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 LD19 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
230400bit/s	8 Bits	1	1	1	none	1	none

## DATA PROTOCOL

### Data packet format

The LD19 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 PO/NT_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, 0xd1, 0x79, 0x34, 0xe3,
0xae, 0xf2, 0xbf, 0x68, 0x25, 0x8b, 0xc6, 0x11, 0x5c, 0xa9, 0xe4, 0x33,
0x1e, 0xd0, 0x9d, 0x4a, 0x01, 0x5b, 0x16, 0xc1, 0x8c, 0x22, 0x6f, 0xb8,
0xf5, 0x1f, 0x52, 0x85, 0xc8, 0x66, 0x2b, 0xfc, 0xb1, 0xed, 0xa0, 0x11,
0x3a, 0x94, 0xd9, 0x0e, 0x43, 0xb6, 0xfb, 0x2c, 0x61, 0xcf, 0x82, 0x55,
0x18, 0x44, 0x09, 0xde, 0x93, 0x3d, 0x10, 0xa1, 0xea, 0x3e, 0x73, 0xa4,
0xe9, 0x47, 0x0a, 0xdd, 0x90, 0xee, 0x81, 0x56, 0xb1, 0xb5, 0xf8, 0x2f,
0x62, 0x97, 0xda, 0x0d, 0x40, 0xee, 0xa3, 0x74, 0x39, 0x65, 0x28, 0xff,
0xb2, 0x1e, 0x51, 0x86, 0xeb, 0x21, 0x6e, 0xbb, 0xf6, 0x58, 0x15, 0xe2,
0x8f, 0xd3, 0x9e, 0x49, 0x04, 0xaa, 0xe1, 0x30, 0xd1, 0x88, 0xe5, 0x12,
0x5f, 0xf1, 0xbe, 0x6b, 0x26, 0x1a, 0x37, 0xe0, 0xad, 0x03, 0x4e, 0x99,
0xd4, 0x1e, 0x31, 0xe6, 0xab, 0x05, 0x48, 0x9f, 0xd2, 0x8e, 0xe3, 0x14,
0x59, 0xf1, 0xba, 0x6d, 0x20, 0xd5, 0x98, 0x4f, 0x02, 0xae, 0xe1, 0x36,
0xb1, 0x27, 0x6a, 0xbd, 0xf0, 0x5e, 0x13, 0xe4, 0x89, 0x63, 0x2e, 0xf9,
0xb4, 0x1a, 0x57, 0x80, 0xed, 0x91, 0xde, 0x0b, 0x46, 0xe8, 0xa5, 0x72,
0x3f, 0xca, 0x87, 0x50, 0xd1, 0xb3, 0xfe, 0x29, 0x64, 0x38, 0x75, 0xa2,
0xef, 0x41, 0x0e, 0xdb, 0x96, 0x42, 0x0f, 0xd8, 0x95, 0x3b, 0x76, 0xa1,
0xee, 0xb0, 0xfd, 0x2a, 0x67, 0xe9, 0x84, 0x53, 0x1e, 0xeb, 0xa6, 0x71,
0x3e, 0x92, 0xdf, 0x08, 0x45, 0x19, 0x54, 0x83, 0xee, 0x60, 0x2d, 0xfa,
0xb1, 0x5d, 0x10, 0xe1, 0x8a, 0x24, 0x69, 0xbe, 0xf3, 0xaf, 0xe2, 0x35,
0x 78, 0xd6, 0x9b, 0x4e, 0x01, 0xf4, 0xb9, 0x6e, 0x23, 0x8d, 0xe0, 0x1 7,
0x5a, 0x06, 0x4b, 0x9e, 0xd1, 0x1f, 0x32, 0xe5, 0xa8 };
uint8_t CaJCRC8(uint8_t *p, uint8_t Jen){
uint8_t ere= 0;
uint16_t i;
for (i = 0; i < Jen; i++){
ere= CrcTable[(ere ^ *p++) & 0xff]; }
return ere;
```

## 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	lByte



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. For a white object within 6m, the typical value of the signal strength value is around 200. 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:

$$\text{step} = (\text{end\_angle} - \text{start\_angle}) / (\text{Jen} - 1);$$
$$\text{angle} = \text{start\_angle} + \text{step} * i;$$

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

## Example

Suppose we receive a piece of data as shown below.

54 2C 68 08 AB 7E EO 00 E4 DC 00 E2 D9 00 ES DS 00 E3 D3 00 E4 DO 00 E9 CD 00 E4 CA 00 E2 C7 00 E9 CS 00 ES C2 00 ES CO 00 ES BE 82 3A IA 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	IAH	50H



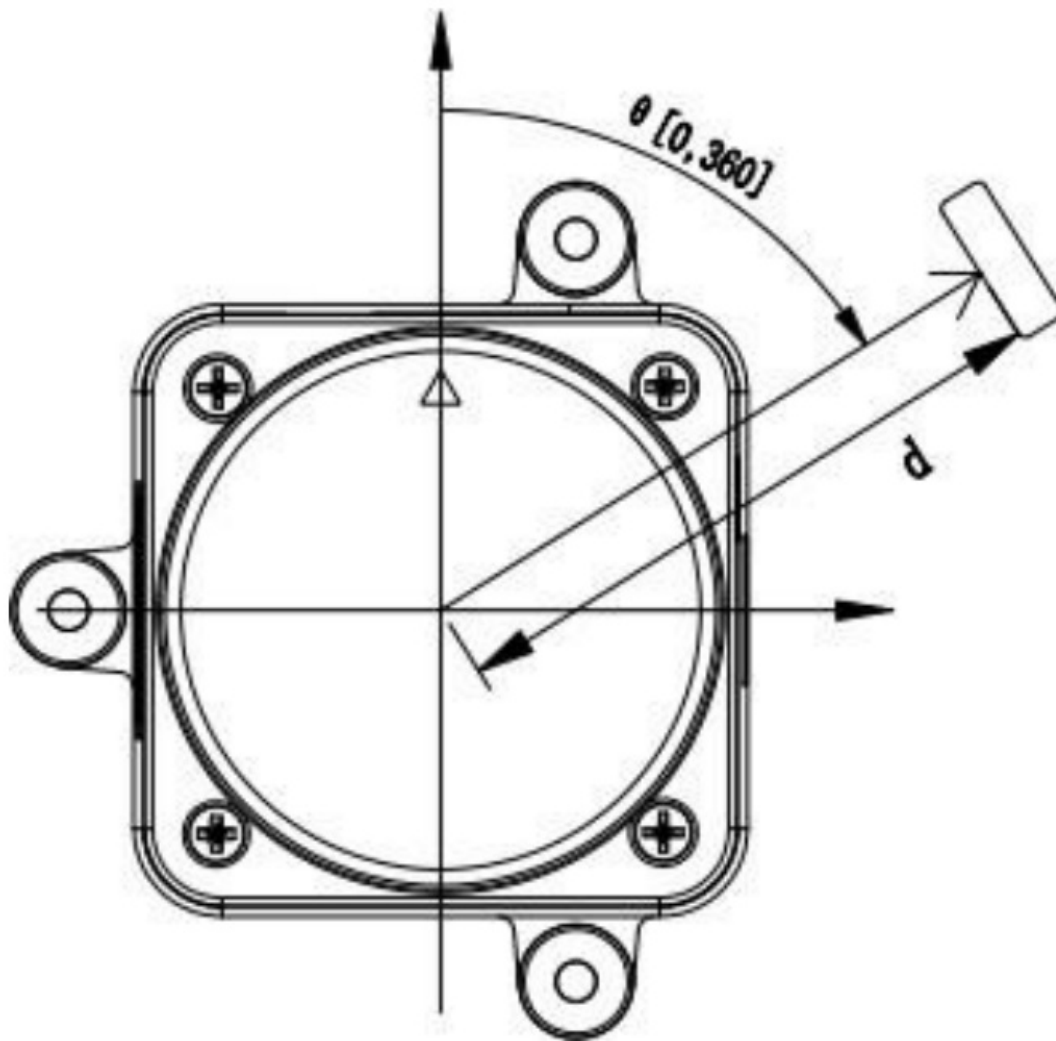
Measuring point 1			Measuring point 2			...	Measuring point 12		
distance		intensity	distance		intensity		distance		intensity
EOH	OOH	E4H	DCH	OOH	E2H	...	BOH	OOH	EAH



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	OOEOH = 224mm
Measuring point 1 intensity	E4H = 228
Measuring point 2 distance	OODCH = 200mm
Measuring point 2 intensity	OOE2H= 226
...	...
Measuring point 12 distance	OOBOH = 176mm
Measuring point 12 intensity	EAH=234

## COORDINATE SYSTEM

The LD19 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.



## DEVELOPMENT KIT INSTRUCTIONS

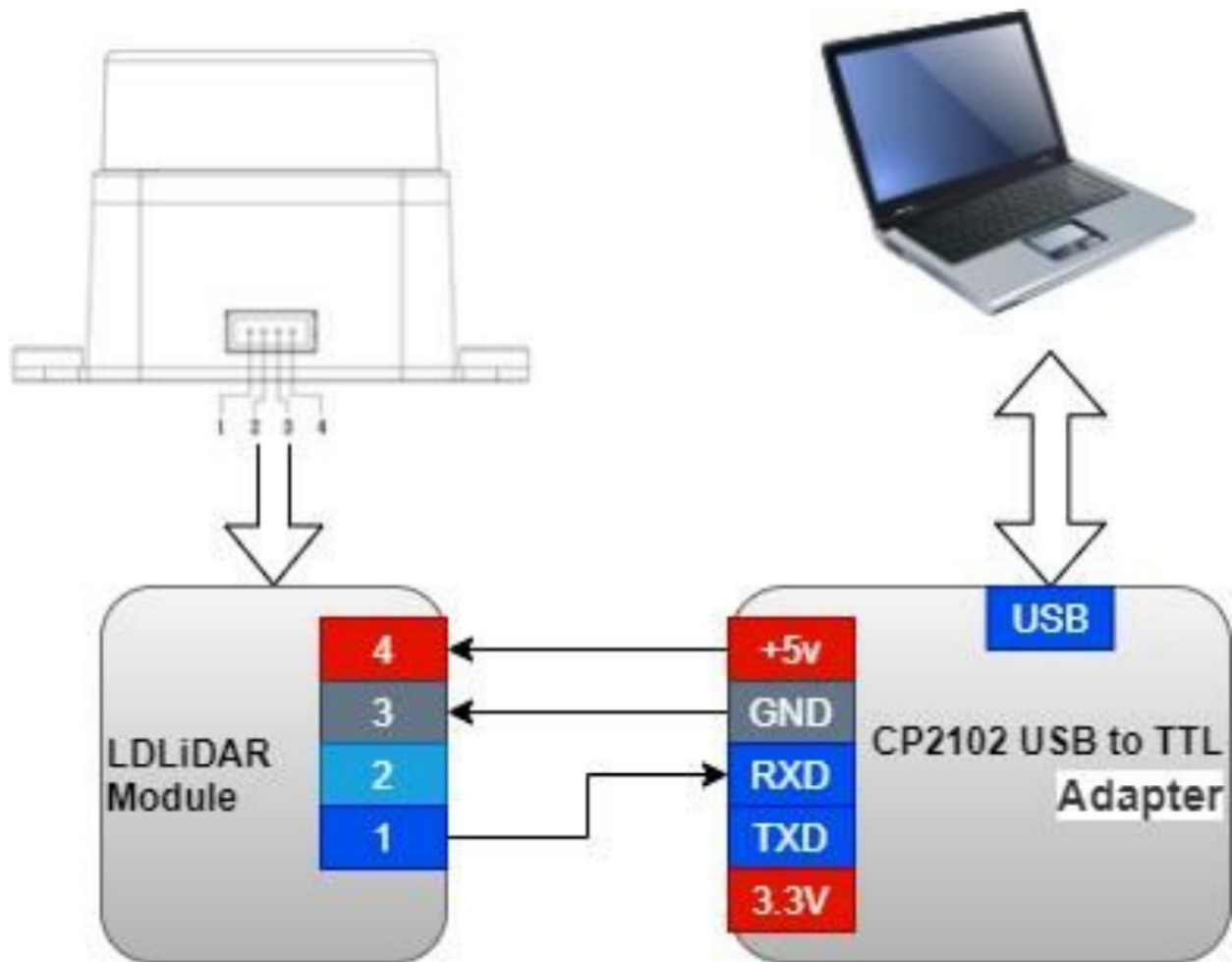
### How to use the assessment tool

### 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:



### 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, 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.

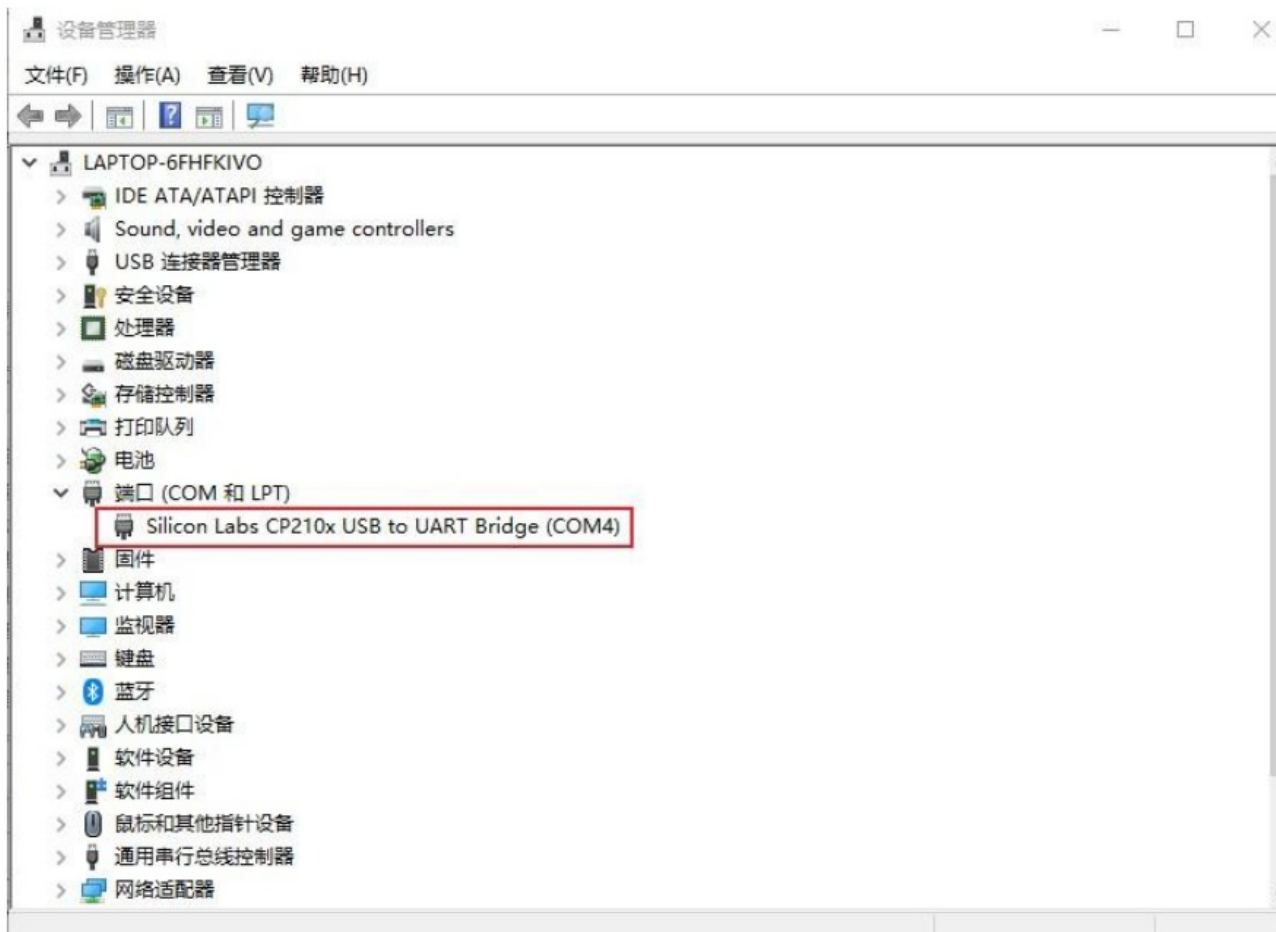
名称	修改日期	类型	大小
arm	2018/12/8 0:06	文件夹	
arm64	2018/12/8 0:06	文件夹	
x64	2018/12/8 0:06	文件夹	
x86	2018/12/8 0:06	文件夹	
CP210x_Universal_Windows_Driver_ReleaseNotes...	2018/12/7 23:53	TXT 文件	20 KB
CP210xVCPInstaller_x64.exe	2018/5/8 6:05	应用程序	1,026 KB
CP210xVCPInstaller_x86.exe	2018/5/8 6:05	应用程序	903 KB
dpinst.xml	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.

#### CP210x USB to UART Bridge Driver Installer

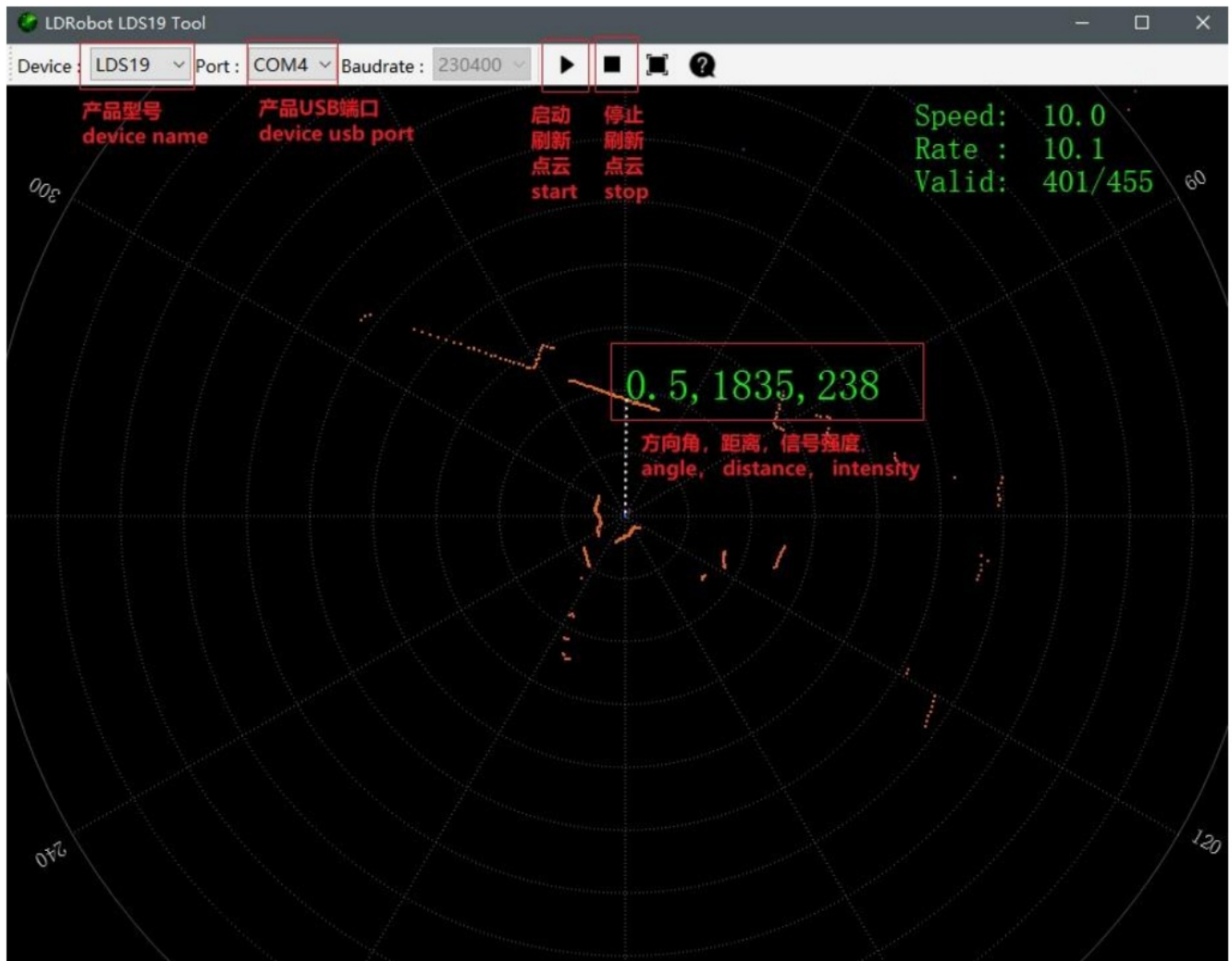


After the installation is complete, connect the USB adapter board in the development kit to the computer, right-click [My Computer], select [Properties], and in the opened [System] interface, select [Device Manager] in the left menu to enter Go to the device manager, expand [Ports], you can see the serial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



## 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.

#### Product 3D model file

Unzip the LiDAR\_LD19\_3D\_stp\_VI.0 file to obtain a 3D model file in STP format.

#### Operation based on ROS under Linux

#### 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).

## Get the source code of the ROS Package

The source code of the ROS function package of this product is hosted on the repository of Github. You can download the source code of the master or main branch by accessing the repository network link, or download it through the git tool. Users can also directly extract SDK LD19 > ldlidar\_stl\_ros.zi to the following path for use.

### 1. Repository website address

- <https://github.com/DFRobotdl/ldlidarstlros>

### 2. git tool download operation

```
# First open the terminal interface, you can use the shortcut key of ctrl+alt+t
# If the Ubuntu system you are using does not have the git tool installed, you can install it as
follows:
$ sudo apt-get install git
# Download the source code of the product ROS function package:
$ cd ~
$ mkdir -p ldlidar_ros_ws/src
$ cd ~/ldlidar_ros_ws/src
$ git clone https://github.com/DFRobotdl/ldlidar_stl_ros.git
#or
$ unzip ldlidar_stl_ros.zip
```

## Set device permissions

First, connect the lidar to our adapter module (CP2102 adapter), and connect the module to the computer. Then, open a terminal under the ubuntu system and enter `ls /dev/ttyUSB*` to check whether the serial device is connected. If a serial port device is detected, use the `sudo ch mod 777 /dev/ttyUSB*` command to give it the highest authority, that is, give the file owner, group, and other users read, write and execute permissions, as shown in the following figure.



```
linux@ubuntu: ~  
linux@ubuntu:~$ ls /dev/ttyUSB*  
/dev/ttyUSB0  
linux@ubuntu:~$ sudo chmod 777 /dev/ttyUSB*  
[sudo] password for linux:  
linux@ubuntu:~$
```

Finally, modify the `port_name` value in the `ld19.launch` file in the `~/ldlidar_ros_ws/src/ldlidar_stl_ros/launch/` directory. Take the lidar mounted in the system as `/dev/ttyUSB0` as an example, as shown below.

```
$ nano ~/Jdlidar_ros_ws/src/ldlidar_stl_ros/launch/ld19.launch
```

```
linux@ubuntu: ~  
GNU nano 2.5.3 File: ...lidar_stl_ros/launch/ld06.launch  
  
<launch>  
  <node name="LD06" pkg="ldlidar_stl_ros" type="ldlidar_stl_ros_node"$  
    <param name="product_name" value="LDLiDAR_LD06"/>  
    <param name="topic_name" value="LiDAR/LD06"/>  
    <param name="port_name" value = "/dev/ttyUSB0"/>  
    <param name="frame_id" value="lidar_frame"/>  
  </node>  
</launch>
```

Linux nano editor: Ctrl + O saves the edited file; Ctrl + X exits the editing interface.

## Build and environment settings

1. Use the catkin compilation system to compile and build the product function package:

```
$ cd ~/fdlidauos~ws  
.$ catkin_make
```

2. Function package environment variable settings:

After the compilation is completed, you need to add the relevant files generated by the compilation to the environment variables, so that the ROS environment can recognize them. The execution command is as follows. This command is to temporarily add environment variables to the terminal, which means that if you reopen a new terminal, you also need to re-execute it. The following command.

```
$ cd ~/tdlidar_ros_ws  
$ source devel/setup.bash
```

In order to never need to execute the above command to add environment variables after reopening the terminal, you can do the following.

```
$ echo source ~/.ldlidar_ros_ws/devel/setup.bash » ~/.bashrc
$ source ~/.bashrc
```

## Run node and Rviz display LiDAR point cloud

Start the lidar node and execute the following command.

```
$ roslaunch ldlidar_stl_ros ld19.launch
```

Start the lidar node and display the lidar point cloud data on Rviz, execute the following command.

```
# if ROS_DISTRO in 'kinetic' or 'melodic'
$ ros/aunch ldlidar_st/_ros viewer_ld19_kinetic_me/odic.launch
# if ROS_DISTRO in 'noetic'
$ ros/aunch ldlidar_st/_ros viewer_ld19_noetic.launch
```

## Operation based on ROS2 under Linux

### 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.

### Get the source code of ROS2 Package

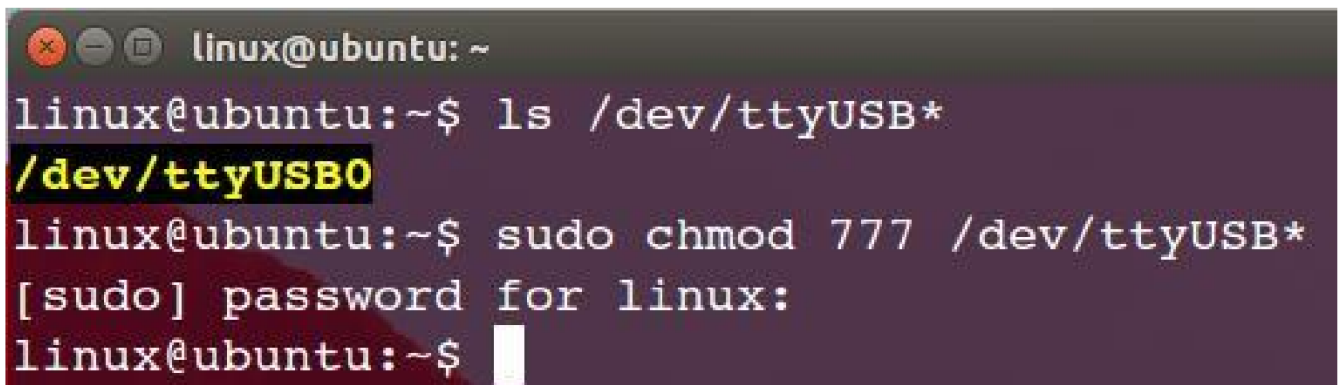
The source code of the ROS2 function package of this product is hosted on the repositories of Github. You can download the source code of the master or main branch by accessing the network link of the repository, or download it through the git tool. Users can also directly extract `SDK LD19 > ldlidar_stl_ros2.ziR` to the following path for use.

1. Repository website address
  - <https://github.com/DFRobotdl/ldlidarstlros2>
2. git tool download operation

```
# First open the terminal interface, you can use the shortcut key of ctrl+alt+t
# If the Ubuntu system you are using does not have the git tool installed, you can install it as
follows:
$ sudo apt-get install git
# Download the source code of the product ROS2 function package:
$ cd ~
$ mkdir -p ldlidar_ros2_ws/src
$ cd ~/ldlidar_ros2_ws/src
$ git clone https://github.com/DFRobotdl/ldlidar_stl_ros2.git
#or
$ unzip ldlidar_stl_ros2.zip
```

## Set device permissions

First, connect the lidar to our adapter module (CP2102 adapter), and connect the module to the computer. Then, open a terminal under the ubuntu system and enter `ls /dev/ttyUSB*` to check whether the serial device is connected. If a serial port device is detected, use the `sudo chmod 777 /dev/ttyUSB*` command to give it the highest authority, that is, give the file owner, group, and other users read, write and execute permissions, as shown in the following figure.

A terminal window titled 'linux@ubuntu: ~' with a dark background. The user enters the command 'ls /dev/ttyUSB\*' and the output is '/dev/ttyUSB0' in yellow. Then, the user enters 'sudo chmod 777 /dev/ttyUSB\*'. The terminal prompts '[sudo] password for linux:' and the user's password is entered (represented by a white box). The prompt returns to 'linux@ubuntu:~\$'.

```
linux@ubuntu: ~
linux@ubuntu:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
linux@ubuntu:~$ sudo chmod 777 /dev/ttyUSB*
[sudo] password for linux:
linux@ubuntu:~$
```

Finally, modify the `port_name` value in the `ld19.launch.py` file in the `~/ldlidar_ros2_ws/src/ldlidar_stl_ros2/launch/` directory. Take the lidar mounted in the system as `/dev/ttyUSB0` as an example, as shown below.

```
$ nano ~/ldlidar_ros2_ws/src/ldlidar_stl_ros2/launch/ld19.launch.py
```

```
linux@ubuntu: ~  
...gfs/hmd ubuntu/ldlidar_stl_ros2_ws/src/ldlidar_stl_ros2/launch/ld06.launch.py  
#!/usr/bin/env python3  
from launch import LaunchDescription  
from launch_ros.actions import Node  
  
def generate_launch_description():  
    return LaunchDescription([  
        Node(  
            package='ldlidar_stl_ros2',  
            executable='ldlidar_stl_ros2_node',  
            name='LD06',  
            output='screen',  
            parameters=[  
                {'product_name': 'LDLiDAR_LD06'},  
                {'topic_name': 'LiDAR/LD06'},  
                {'port_name': '/dev/ttyUSB0'},  
                {'frame_id': 'lidar_frame'}  
            ]  
        )  
    ]  
)
```

Linux nano editor: Ctrl + O saves the edited file; Ctrl + X exits the editing interface.

## Build and environment settings

1. Use the colcon compilation system to compile and build the product function package:

```
$ cd ~/fdlidaus2~ws  
.$ co/con build
```

2. Function package environment variable settings:

After the compilation is completed, you need to add the relevant files generated by the compilation to the environment variables, so that the ROS2 environment can be recognized. The execution command is as follows. This command is to temporarily add environment variables to the terminal, which means that if you reopen a new terminal, you also need to re-execute it. The following command.

```
$ cd ~/Jdlidar_ros2_ws  
$ source install/setup.bash
```

In order to never need to execute the above command to add environment variables after reopening the terminal, you can do the following.

```
$ echo source ~/Jdlidar_ros2_ws/install/setup.bash » ~/.bashrc
```

```
$ source ~/.bashrc
```

## Run node and Rviz2 display LiDAR point cloud

Start the lidar node and execute the following command.

```
$ ros2 launch ldlidar_stl_ros2 ld19.launch.py
```

Start the lidar node and display the lidar point cloud on Rviz2, execute the following command.

```
$ ros2 launch ldlidar_stl_ros2 viewer_ld19.launch.py
```

## Instructions for using SDK under Linux

### Get the source code of SDK

The source code of the Linux SOK of this product is hosted on the repositories of Github. You can download the source code of the master or main branch by accessing the network link of the repository, or download it through the gittool. Users can also directly extract `SOK L019 > ldlidar_stl_sdk.zip` to the following path for use.

#### 1. Repository website address

► <https://github.com/OFRobotdl/ldlidarstlsdk>

#### 2. git tool download operation

```
# First open the terminal interface, you can use the shortcut key of ctrl+alt+t
# If the Ubuntu system you are using does not have the git tool installed, you can install it as
follows:
$ sudo apt-get install git
# Download the source code:
$ cd ~
$ mkdir ldlidar_ws
$ cd ~/ldlidar_ws
$ git clone https://github.com/DFRobotdl/ldlidar_stl_sdk.git
#or
$ unzip ldlidar_stl_sdk.zip
```

### Set device permissions

First, connect the lidar to our adapter module (CP2102 adapter), and connect the module to the computer. Then, open a terminal under the ubuntu system and enter `ls /dev/ttyUSB*` to check whether the serial device is connected. If a serial port device is detected, use the `sudo chmod 777 /dev/ttyUSB*` command to give it the highest authority, that is, give the file owner, group, and other users read, write and execute permissions, as shown in the following figure.

```
linux@ubuntu: ~  
linux@ubuntu:~$ ls /dev/ttyUSB*  
/dev/ttyUSB0  
linux@ubuntu:~$ sudo chmod 777 /dev/ttyUSB*  
[sudo] password for linux:  
linux@ubuntu:~$
```

## Build

The source code is coded in C++11 standard C++ language and C99 standard C language. Use CMake, GNU-make, GCC and other tools to compile and build the source code. If you use Ubuntu system without the above tools installed, you can execute the following command to complete the installation.

```
$ sudo apt-get install build-essential cmake
```

If the tools indicated above already exist in the system, do the following.

```
$ cd ~/ldlidar_ws/ldlidar_stl_sdk  
# If the build folder does not exist in the ldlidar_stl/_sdk directory, it needs to be created  
$ mkdir build  
$ cd build  
$ cmake .. /  
$ make
```

## Run binary program

```
$ cd ~/ldlidar_ws/ldlidar_stl_sdk/build  
$ ./ldlidar_stl <serial_number>  
# eg: ./ldlidar_stl /dev/ttyUSB0
```

## Instructions for using ROS based on Raspberry Pi SBC

Please refer to the manual « LD19 Raspberry Pi Raspbian User manual\_ V2.9.pdf)) for details.  
In addition, we have provided a custom image for Raspberry Pi for this product, and its usage tutorial is as follows:

## Introduction to mirroring

### 1. Mirror composition:

- raspberrypi raspbian OS version: 2020-08-20-raspbian-buster-armhf
- ROS environment version: ROS melodic
- LiDAR LD19 ROS Package

### 2. Hardware support:

- raspberrypi 3B+ SBC , raspberrypi 4B SBC

- SD card with a capacity greater than or equal to 16GB

## Mirror usage

### 1. Download image file:

- Download link 1: <https://pan.baidu.com/s/lfvTfXBbWC9ESXNNUY5aJhw> 1Jt:&:7ky8a
- Download link 2:  
<https://drive.google.com/file/d/lyIMTFGRZ9cRcy3Njvf10cxDo4Wy3tfCB/view?usp=sharing>
- The image file name is [2022-03-24-raspios-buster-armhf-ldrobot-customization.img.xz](#)

### 2. Write the image file to SD card and run the system:

Write through the Win32DiskImager tool, insert it into the Raspberry Pi card slot after successful writing, and power on the system

#### 1. System login related information

- Username: `pi`
- Hostname: `raspberrypi`
- Pass ward `pi`

#### 2. Running the lidar node

```
#step1: Make sure the lidar device is connected to the raspberrypi SBC, and open a terminal via the
shortcut Ctrl+Alt+ T.
#step2: Retrieve the port device file corresponding to the radar device through ls -l /dev/l, give executable
permission, and then modify the lanuch file parameters. Take the port file corresponding to the lidar device
as /dev/ttyUSB0 as an example.
$ sudo chmod 777 /dev/ttyUSB0
# Note: It is recommended to update the Lldiar ROS driver package in the mirror for the first time
$ cd ~ && cd ~/ldlidar_ros_ws/src/
$ rm -rf ldlidar_stl_ros/
$ git clone https://github.com/DFRobotdl/ldlidar_stl_ros.git
```

Finally, modify the `port_name` value in the `ld19.launch` file in the `~/ldldiar_ros_ws/src/ldlidar_stl_ros/launch/` directory. Take the lidar mounted in the system as `/dev/ttyUSB0` as an example, as shown below.

```
$ nano ~/ldlidar_ros_ws/src/ldldiar_stl_ros/launch/ld19.launch
```



```
File Edit Tabs Help
GNU nano 3.2 ldlidar_ros_ws/src/ldlidar_stl_ros/launch/ld19.launch

<launch>
  <node name="LD19" pkg="ldlidar_stl_ros" type="ldlidar_stl_ros_node" output="screen">
    <param name="product_name" value="LDLiDAR_LD19"/>
    <param name="topic_name" value="LiDAR/LD19"/>
    <param name="port_name" value="/dev/ttyUSB0"/>
    <param name="frame_id" value="lidar_frame"/>
  </node>
</launch>

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Start the lidar node and execute the following command.

```
$ roslaunch ldlidar_stl_ros ld19.launch
```


Start the lidar node and display the lidar point cloud data on Rviz, execute the following command.

```
$ rosrun ldlidar_stl_ros viewer_ld19_kinetic_me/odc./launch
```


## REVISION HISTORY



version	revision date	modify the content
1.0	2020-09-01	Initial creation
1.1	2021-01-15	Remove the Transform() function
2.0	2022-02-27	Added the content of the development kit instructions
2.1	2022-03-06	Increase the graphic design of the document and revise the content format
2.2	2022-03-09	Modify the document cover title and part of the content
2.3	2022-03-15	Revise problematic statements in documentation
2.4	2022-04-02	<ol style="list-style-type: none"> <li>1. Modify the LOGO;</li> <li>2. Added 30 model file resource introduction;</li> <li>3. Add the introduction of Raspberry Pi custom image usage;</li> <li>4. Fix the error in the description of the document</li> </ol>
2.5	2022-06-25	<ol style="list-style-type: none"> <li>1. Added support for ROS2 Humble version;</li> <li>2. Modify Rviz, Rviz2 to display the relevant content of laser point cloud;</li> <li>3. Modify the relevant content of the Windows point cloud host computer</li> </ol>

<div> <div>LiDAR LD19</div> <div>Development Manual</div> <div>  </div> </div>	<div> <div> <a href="#">DFRobot LiDAR LD19 Laser Sensor Kit</a> [pdf] Instruction Manual         </div> <div>           LiDAR LD19 Laser Sensor Kit, LiDAR LD19, Laser Sensor Kit, Sensor Kit         </div> </div>
---	---

## References

-  [ROS/Installation - ROS Wiki](#)
-  [Installation — ROS 2 Documentation: Foxy documentation](#)
-  [GitHub - DFRobotdl/ldlidar\\_stl\\_ros](#)
-  [GitHub - DFRobotdl/ldlidar\\_stl\\_ros](#)
-  [GitHub - DFRobotdl/ldlidar\\_stl\\_ros2](#)
-  [GitHub - DFRobotdl/ldlidar\\_stl\\_ros2](#)
-  [GitHub - DFRobotdl/ldlidar\\_stl\\_sdk](#)
-  [GitHub - DFRobotdl/ldlidar\\_stl\\_sdk](#)
-  [\\_\\_\\_\\_\\_](#)
-  [CP210x USB to UART Bridge VCP Drivers - Silicon Labs](#)