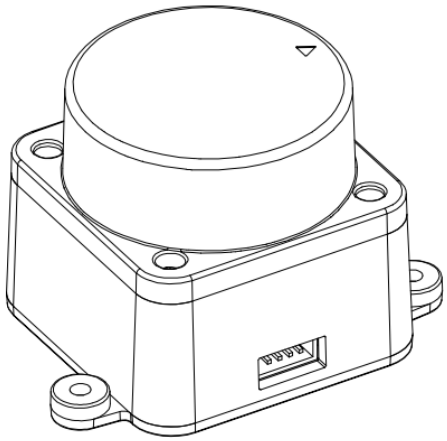
LiDAR Sensor LD19

Development Manual



Content:

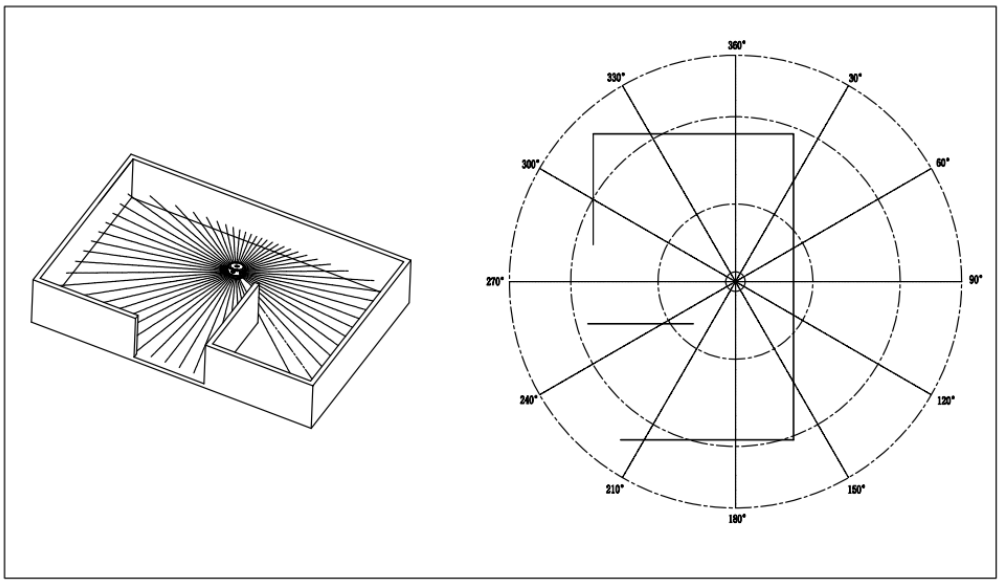
1. PRODUCT DESCRIPTION

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.

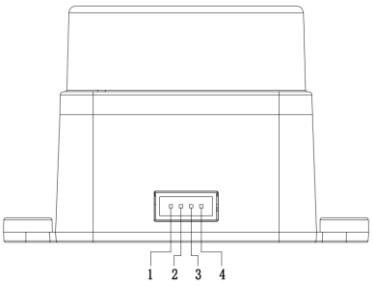
After obtaining the distance data, the LD19 will combine the angle values measured by the angle measurement unit to form point cloud data, and then send the point cloud data to the external interface through wireless communication. LD19 supports internal speed control, the speed can be stabilized to 10±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 LD19 reach the specified speed.

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


360°
330°
30°
300°
60°
270°
90°
240°
120°
210°
150°
180°

2. 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 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 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	none	none

3. DATA PROTOCOL

3.1. 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 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,
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;
}
```