WHEELTEC
INTELLIGENT TECHNOLOGY

轮 趣 科 技

STM32 读取 LD14 雷达

数据说明

推荐关注我们的公众号获取更新资料

版本说明：

版本	日期	内容说明
V1.0	2023/04/24	第一次发布

网址:www.wheeltec.net

目录

1. LD14 雷达

1.1 LD14 雷达简介

LD14 主要由激光测距核心，无线传电单元，无线通讯单元，角度测量单元、电机驱动单元和机械外壳组成。

LD14 测距核心采用三角测量法技术，可进行每秒 2300 次的测距。每次测距时，LD14 从一个固定的角度发射出红外激光，激光遇到目标物体后被反射到接收单元。通过激光、目标物体、接收单元形成的三角关系，从而解算出距离。

获取到距离数据后，LD14 会融合角度测量单元测量到的角度值组成点云数据，然后通过无线通讯将点云数据发送到外部接口。同时电机驱动单元会驱动电机，通过 PID 算法闭环控制到指定的转速。

角度分辨率为 1°，测距的最大量程为 8 米，最小量程为 15 厘米。

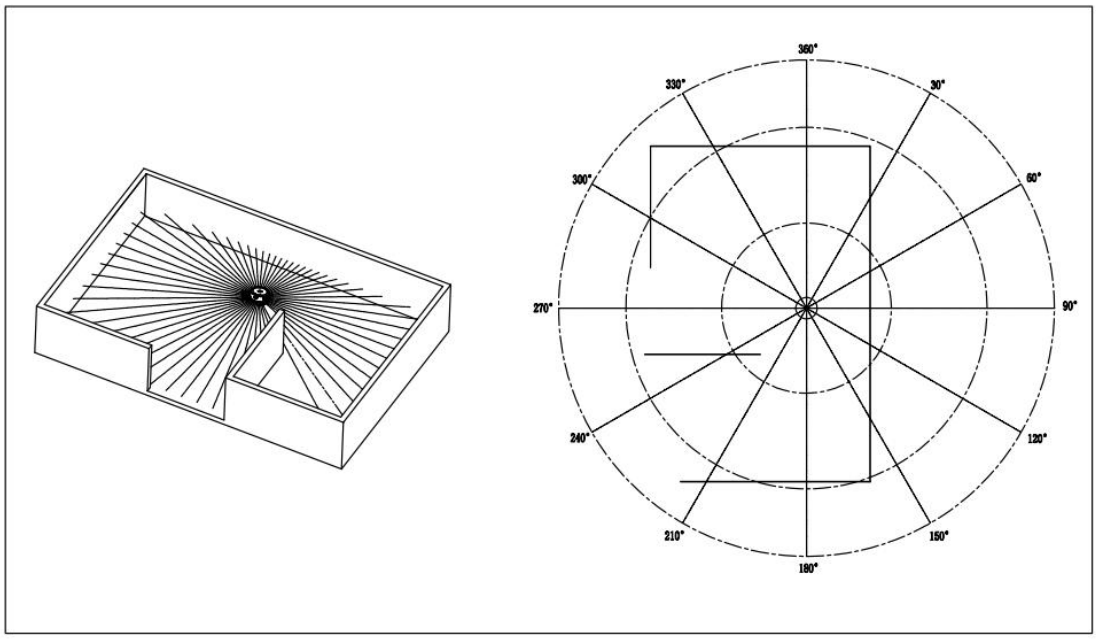

图 1-1 点云图示

1.2 接线定义

LD14 使用 1.25mm 4PIN 连接器与外部系统连接，实现供电和数据接收，具体接口定义和参数要求见下图/表：

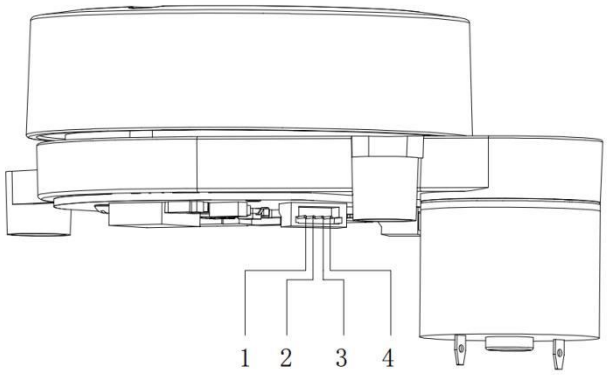

图 1-2 LD14 雷达的接口图示

表 1-2 LD14 雷达引脚表

序号	管脚	描述	最小值	典型值	最大值
1	PWM	雷达内部电机控制信号	0V	-	3.3V
2	GND	供电电压负极	0V	0	0
3	TX	雷达数据输出	0V	3.3V	3.5V
4	5V	供电电压正极	4.5V	5V	5.5V

LD14 支持内部控速和外部控速。在 PWM 引脚接地或悬空时，默认为内部调速，默认转速为 6Hz。外部控速需要在 PWM 引脚接入方波信号，可通过 PWM 信号占空比控制电机的启、停和转速。触发外部控速的条件：a：输入 PWM 频率 15-30K，推荐 24K；b：占空比在(45%, 55%)区间内(不包含 45%和 55%)，且最少 100ms 持续输入时间。触发外部控速后就一直处于外部控速状态，除非断电重启才会恢复内部控速；同时可以通过调节 PWM 占空比进行转速控制。由于每个产品电机的个体差异，占空比设置为典型值时实际转速可能会有差异，如要精确控制电机转速，需根据接收数据中的转速信息进行闭环控制。**注：不使用外部控速时，必须将 PWM 引脚接地或悬空。**

1.3 数据输出定义

雷达使用的是串口通信，波特率为 115200bps，点云输出协议如下：

表 1-3 数据输出协议

Byte_0	Byte_1	Byte_2	Byte_3	Byte_4
0x54	Length	Speed_L	Speed_H	Start_Angle_L
Byte_5	Byte_6	Byte_7	Byte_8	Byte_9
Start_Angle_H	Distance_1_L	Distance_1_H	PEAK_1	Distance_2_L
Byte_10	Byte_11	.....	.....	Byte_39
Distance_2_H	PEAK_2	.....	.....	Distance_12_L
Byte_40	Byte_41	Byte_42	Byte_43	Byte_44
Distance_12_H	PEAK_12	Stop_Angle_L	Stop_Angle_H	Time stamp_L
Byte_45	Byte_46			
Time stamp_H	CRC			

1.Byte_0：为帧头，固定值。

2.Byte_1：高三为表示帧类型，目前固定为 1，低五位表示一个包的测量点数，目前固定为 12，所以改字节固定为 0x2C。

3.Byte_2-Byte_3：雷达转速，单位为度每秒，低八位在前，例如 Speed_L=0x68，Speed_H=0x08，即 0x0868—2512 度/s。

4.Byte_4-Byte_5：一帧数据的起始角度，低位在前，是实际角度的 100 倍，例如：Start_Angle_L=0xAB，Start_Angle_H=0x7E，即 0x7EAB-32427-324.27 度。

5.Byte_6-Byte_41：点云数据：每个数据包括 2 字节距离和 1 字节强度信息。低位在前，距离单位 mm 例如：Distance_1_L=0x64，Distance_1_H=0X00，PEAK_1=0x64，表示距离 0x64=100mm，强度 100。

6.Byte_42-Byte_43：一帧数据的结束角度，低位在前，是实际角度的 100 倍，例如：Stop_Angle_L=0xBE，Stop_Angle_H=0x82，即 0x82BE-33470-334.70 度。

7.Byte_44-Byte_45：时间戳，单位为 ms，最大为 30000，到达 30000 会重新开始计数。

8.Byte46：从 Byte_0 到 Byte_45 数据和校验值。

CRC = byte0+byte1+…+byte56

1.4 输出数据案例分析

LD14 雷达的数据角度分辨率是 1 度，所以雷达转一圈那么会输出 360 个数据，但雷达一包数据包输出的点是 12 个点，按此可知，雷达一圈能输出差不多 30 个数据包，输出一包数据包的角度范围大概是 12 度。雷达接上串口后会自动以波特率 115200 发送数据包，不需要发送其他命令。数据包之间间隔一个点的角度，也就是角度分辨率 1 度。以下是使用串口助手接收到的数据，可以看出[54 2C 68 08 BE 82 E5 00 E8 D7 00 E4 D8 00 E1 D1 00 E4 D3 00 E8 D3 00 E4 CD 01 E7 C1 00 E4 C4 00 E0 C0 00 E2 C0 00 E1 C0 00 15 CE 86 3A 1A 50]其中的一个数据包。我们对其进行解析。

图 1-4 串口助手接收信息

1. 帧头和数据帧长度

帧头 1 个字节，固定值是 0X54。

数据帧长度也是固定的，值为 0X2C，单位为字节，即 47 字节。

2. 转速

分析捕获到的转速信息[68 08]，即 0X0868= 2152 度/s。

3. 起始角度和结束角度

角度信息是放大了 100 倍的信息，如捕获到的起始角度信息[BE 82]，即 =0X82BE = 33470 = 334.70 度；结束角度信息[CE 86]，即 0X86CE =34510 = 345.10 度。

4.点云数据

一包数据包会输出 12 个点的数据，每个点的数据包含距离和强度信息。比如捕获到的第一个点云数据[E5 0 E8]，可以知道距离是 0X00E5=229mm，强度为 0XE8=232。

5.校验和

校验和是前面所有数据相加的结果，只要 8 位数据。

例如数据为：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 这 46 个数据我们可以经过下面的代码来计算得出 50 这个结果。

```
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;
}
```

1.5 输出角度数据坐标定义

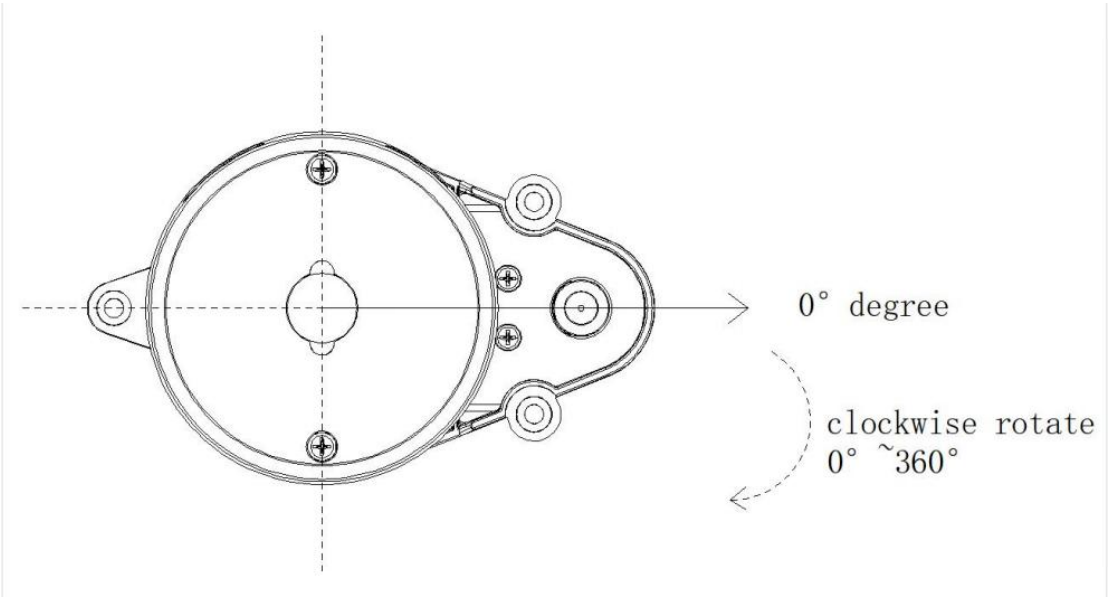

图 1-5 LD14 雷达坐标系定义

2. 小车避障程序说明

以下为避障例程进行说明。小车是 L150 阿克曼小车，使用串口 5 读取 LD14 雷达传回来的数据，因为 LD14 雷达的角度分辨率是 1 度，不重复的点大约有 360 个，因为没有对雷达的转速进行一个闭环控制，所以导致转速不是稳定不变的，也就导致了一圈的点数不一定是 360。为了简化，将一帧数据包传回来的 12 个点做平均处理，作为一个点储存。

2.1 接线说明

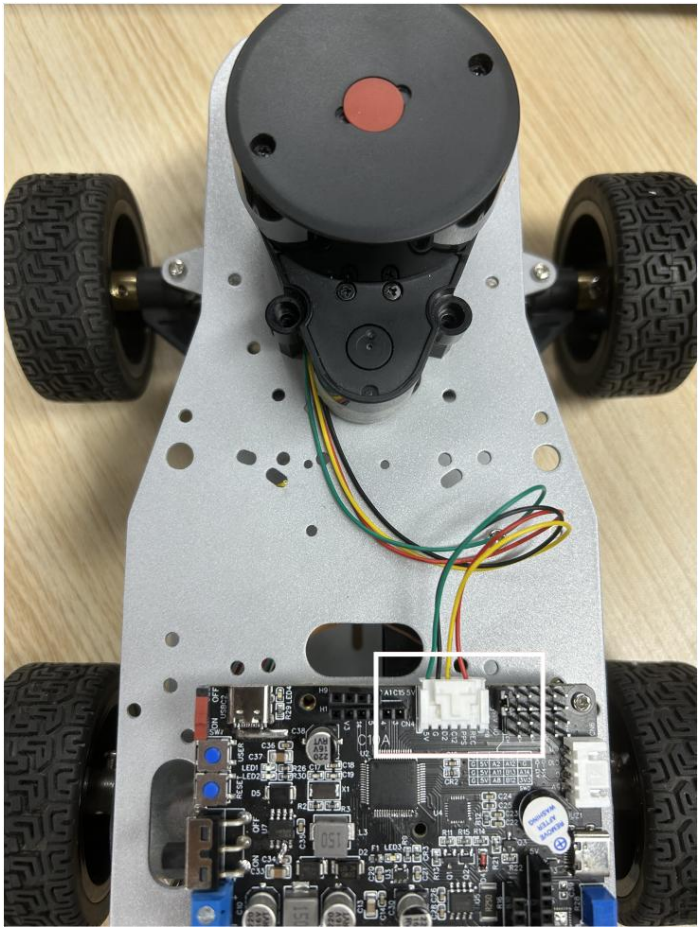

图 2-1 小车与雷达接线图示

表 2-1 小车与雷达接线表

	L150 小车	雷达
绿线	5V	5V
黑线	GND	GND
橙线	PD2	TX
红线	悬空	PWM

2.2 雷达安装朝向

LD14 雷达的安装位置：雷达的 180 度位置对应小车的前进方向

图 2-2 雷达的安装方向

2.3 程序的流程

程序框架简述：雷达在上电的时候会自动传回数据包，不需要进行额外的设置，此时只需要在串口中断中进行接收即可。雷达传回的数据在串口 5 中断进行接收，并且在串口 5 中断中做一定的数据处理（处理函数为 data_process()，在 Lidar.c 中）。我们可以看到雷达的 180 度对应的是小车前进的方向，所以我们在数据处理中将 180 度对应的方向修正为 0 度，这样就是 0 度对应着小车前进的方向。并且因为雷达的转速不是闭环控制，所以可能会导致每一圈的倒数第一帧数据很大概率是“前半段数据是当前这一圈的末尾，后半段数据是下一圈的开头”，数据处理中也将此情况解决，详情可以看例程代码，里面有详细的注释。

一帧数据有 12 个点的位置信息，包括角度和距离。每次传回一帧数据后，做平均简化，然后储存在一个 390 个变量的结构体数组中。此后，每次收到新数据，不断刷新这个数组，作为避障的判断。因为小车是前进着的避障，所以这里设置了以 0 度为中心，左右两边各 50 度的范围内的避障，即前方范围 100 度内的避障，这个时候只需要前方 100 度范围内的数据，其他数据舍弃不要。在

Lidar_Avoid(void)函数中，需要做避障的判断和运动处理。假设小车有一个前进的速度，小车避障的时候设置避障距离是 350mm，在避障算法中，遍历前方 100 度范围内的数据，找出所有距离是小于 350mm 的点，并且判断前方障碍物是位于左边还是位于右边，小车此时转动 z 轴方向向相反的方向转向，直至前方避障范围内没有需要避障的点。