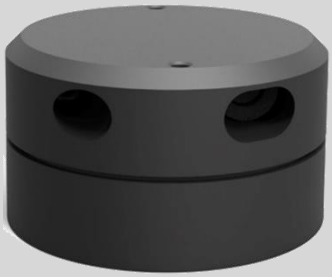
YDLIDAR SDK

使用手册



文档编码：01.13.000020

目录

YDLIDAR SDK 常用接口函数

YDLIDAR 在Linux 下的驱动类YDlidarDriver 的常用接口如下：

表 1 YDLIDAR SDK API

项目	平台	接口函数
创建实例	通用	static void initDriver()
获取实例	通用	static YDlidarDriver* singleton()
销毁实例	通用	static void done()
打开串口	通用	result_t connect(const char * port_path, uint32_t baudrate);
关闭串口	通用	void disconnect();
开始扫描	通用	result_t startScan(bool force = false, uint32_t timeout = DEFAULT_TIMEOUT) ;
停止扫描	通用	result_t stop();
获取状态信息	通用	result_t getHealth(device_health & health, uint32_t timeout = DEFAULT_TIMEOUT)
获取设备信息	通用	result_t getDeviceInfo(device_info & info, uint32_t timeout = DEFAULT_TIMEOUT);
获取一圈点云数据	通用	result_t grabScanData(node_info * nodebuffer, size_t & count, uint32_t timeout = DEFAULT_TIMEOUT) ;
点云数据角度规则化	通用	result_t ascendScanData(node_info * nodebuffer, size_t count);
设备重启	通用	result_t reset(uint32_t timeout = DEFAULT_TIMEOUT);
获取 SDK 版本信息	通用	static std::string getSDKVersion();
获取采样率	G4/F4Pro	result_t getSamplingRate(sampling_rate & rate, uint32_t timeout = DEFAULT_TIMEOUT);
设置采样率	G4/F4Pro	result_t setSamplingRate(sampling_rate & rate, uint32_t timeout = DEFAULT_TIMEOUT);
获取扫描频率	G4/F4/F4Pro	result_t getScanFrequency(scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT);
扫描频率增加 1Hz	G4/F4/F4Pro	result_t setScanFrequencyAdd(scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT);
扫描频率减小 1Hz	G4/F4/F4Pro	result_t setScanFrequencyDis(scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT);
扫描频率增加 0.1Hz	G4/F4/F4Pro	result_t setScanFrequencyAddMic(scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT);
扫描频率减小 0.1Hz	G4/F4/F4Pro	result_t setScanFrequencyDisMic(scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT);
获取雷达扫描方向	G4/F4/F4Pro	result_t setRotationPositive(scan_rotation &

		rotation, uint32_t timeout = DEFAULT_TIMEOUT);
设置雷达扫描方向	G4/F4/F4Pro	result_t setRotationInversion(scan_rotation & rotation, uint32_t timeout = DEFAULT_TIMEOUT);
低功耗模式使能	G4/F4/F4Pro	result_t enableLowerPower(function_state & state, uint32_t timeout = DEFAULT_TIMEOUT);
低功耗模式关闭	G4/F4/F4Pro	result_t disableLowerPower(function_state & state, uint32_t timeout = DEFAULT_TIMEOUT);
恒频模式使能	G4/F4/F4Pro	result_t enableConstFreq(function_state & state, uint32_t timeout = DEFAULT_TIMEOUT);
恒频模式关闭	G4/F4/F4Pro	result_t disableConstFreq(function_state & state, uint32_t timeout = DEFAULT_TIMEOUT);
查询掉电模式状态	G4/F4Pro	const bool getHeartBeat();
设置掉电模式状态	G4/F4Pro	void setHeartBeat(const bool enable);
发送在线信号	G4/F4Pro	result_t sendHeartBeat();
启动电机	X4/S4	result_t startMotor();
停止电机	X4/S4	result_t stopMotor();
设置信号强度标志	S4	void setIntensities(const bool isintensities);

注：result_t 为 int 的宏定义。

YDLIDAR SDK 通用接口描述

创建实例

```
static void initDriver()
```

initDriver ()静态方法创建驱动实例，没有返回值。

获取驱动实例

```
static YDlidarDriver* singleton()
```

singleton ()获取驱动实例，返回值便是驱动实例指针。

销毁实例

```
static void done()
```

打开串口

```
result_t connect(const char * port_path, uint32_t baudrate)
```

connect ()传参为串口名称与波特率，其中波特率应对应相应的雷达型号。

关闭串口

```
void disconnect()
```

disconnect()关闭串口，没有返回值。

获取状态信息

```
result_t getHealth(device_health & health, uint32_t timeout = DEFAULT_TIMEOUT)
```

device_health 为设备状态结构体，getHealth()传参为状态结构体实例便可，返回值有0、-1 与-2，当返回值为 0 时说明获取数据正确，为-1 时说明获取数据失败，为-2 时说明获取数据超时。

获取设备信息

```
result_t getDeviceInfo(device_info & info, uint32_t timeout = DEFAULT_TIMEOUT)
```

device_info 为设备信息结构体，getDeviceInfo ()传参为设备信息结构体实例便可，返回值有 0、-1 与-2，当返回值为0 时说明获取数据正确，为-1 时说明获取数据失败，为-2 时说明获取数据超时。

开始扫描

```
result_t startScan(bool force = false, uint32_t timeout = DEFAULT_TIMEOUT)
```

startScan ()不用传参，返回值有0、-1 与-2，当返回值为0 时说明雷达开始扫描成功，为-1 时说明发送扫描命令失败，为-2 时说明发送扫描命令超时。

停止扫描

```
result_t stop()
```

stop ()不用传参，返回值有0、-1 与-2，当返回值为0 时说明雷达停止扫描成功，为-1 时说明发送扫描命令失败，为-2 时说明发送扫描命令超时。

抓取一圈雷达数据

```
result_t grabScanData(node_info * nodebuffer, size_t & count, uint32_t timeout = DEFAULT_TIMEOUT)
```

node_info 为雷达数据结构体，grabScanData ()传参为雷达数据结构体实例与一圈雷达数据个数，返回值有0、-1 与-2，当返回值为0 时说明获取数据成功，为-1 时说明获取数据失败，为-2 时说明获取数据超时。

点云数据角度规则化

```
result_t ascendScanData(node_info * nodebuffer, size_t count)
```

node_info 为雷达数据结构体，count 为返回的一圈点云数量，ascendScanData()将雷达输出的角度大于 360 度和小于 0 度的数据调整到 0-360 范围内，返回值有 0、-1 与-2，当返回值为 0 时说明获取数据成功，为-1 时说明获取数据失败，为-2 时说明获取数据超时。

设备重启

```
result_t reset(uint32_t timeout = DEFAULT_TIMEOUT)
```

reset()不用传参，返回值为0 时，设备重置成功

获取 SDK 版本信息

```
static std::string getSDKVersion()
```

getSDKVersion()不用传参，返回值为SDK 版本号。

其他接口函数说明

除了通用接口外，不同雷达还有与其相针对的接口函数。具体信息需要结合雷达的开发手册来了解，本文将不作过多阐述。

修订

日期	版本	修订内容
2017-12-05	1.0	初撰
2018-04-03	1.1	优化排版，新增专用接口函数描述