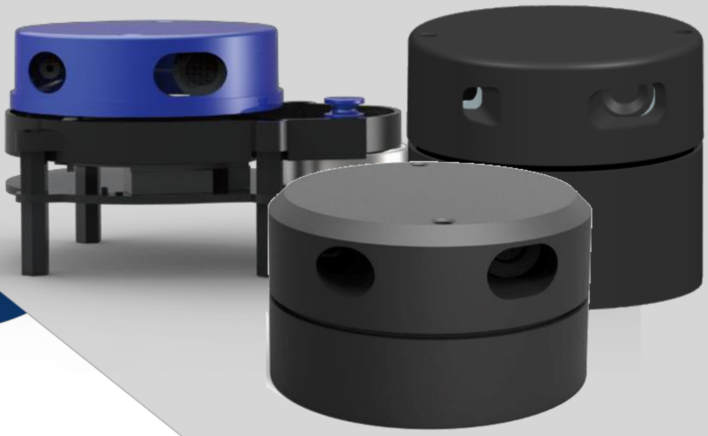
YDLIDAR

YDLIDAR SDK

Manual



文档编码：01.13.000020

Contents

YDLIDAR SDK Common interface functions

The common interface of YDLIDAR driver class YDlidarDriver under Linux is as follows:

Chart 1 YDLIDAR SDK LINUX API

Item	Content
Create an instance	static void initDriver()
Get an instance	static YDlidarDriver* singleton()
Delete instances	static void done()
Open the serial port	result_t connect(const char * port_path, uint32_t baudrate);
Close the serial port	void disconnect();
Get status information	result_t getHealth(device_health & health, uint32_t timeout = DEFAULT_TIMEOUT);
Get device information	result_t getDeviceInfo(device_info & info, uint32_t timeout = DEFAULT_TIMEOUT);
Start Scan	result_t startScan(bool force = false, uint32_t timeout = DEFAULT_TIMEOUT) ;
Stop Scan	result_t stop();
Grab Scan Data	result_t grabScanData(node_info * nodebuffer, size_t & count, uint32_t timeout = DEFAULT_TIMEOUT);
Get SDK version	const std::string getSDKVersion();
Reset	result_t reset(uint32_t timeout = DEFAULT_TIMEOUT);
Get Frequency	result_t getFrequency(uint32_t model, size_t count, float & frequency);
Get Sampling rate	result_t getSamplingRate(sampling_rate & rate, uint32_t timeout = DEFAULT_TIMEOUT);
Set sampling ratte	result_t setSamplingRate(sampling_rate & rate, uint32_t timeout = DEFAULT_TIMEOUT);

*Note: **Result_t is a macro definition of int.***

YDLIDAR SDK Description

Create an instance

```
static void initDriver()
```

The initDriver() static method creates a driver instance with no return value.

Get driver instance

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

Singleton() gets the driver instance, and the return value is the pointer to the driver instance.

Delete instance

```
  static void done()
```

Done () destroys the driver instance and does not return a value.

Open the serial port

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

Connect () is the serial port name and baud rate (X4 defaults to 128000, F4 defaults to 153600, G4 defaults to 230400). If the return value is not -1, the serial port is successfully opened.

Close the serial port

```
  void disconnect();
```

Disconnect() closes the serial port and does not return a value.

Get device status information

```
  result t  getHealth(device health  &  health,  uint32 t  timeout  =
DEFAULT TIMEOUT);
```

Device_health is the device state structure. getHealth() can be passed as an instance of the state structure. The return values are 0, -1, and - 2. When the return value is 0, it indicates that the data is correct. When it is -1, it indicates that the data acquisition fails. . When -2 indicates that the data acquisition timed out.

Get device information

```
  result t  getDeviceInfo(device info  &  info,  uint32 t  timeout  =
DEFAULT TIMEOUT);
```

Device_info is the device information structure body. getDeviceInfo () is passed as the device information structure instance. The return values are 0, -1, and -2. When the return value is 0, it indicates that the data is correct. When it is -1, it indicates that the data acquisition fails. When it is -2, it indicates that the data acquisition time out.

Start scanning

```
  result t  startScan(bool  force  =  false,  uint32 t  timeout  =
DEFAULT TIMEOUT) ;
```

startScan() does not pass arguments, and the return values are 0, -1, and -2. When the return value is 0, it indicates that the lidar started the scan successfully. When -1 indicates that the scan command failed to send, -2 indicates that the send scan command times out.

Stop Scanning

```
  result_t stop();
```

Stop () does not pass arguments, the return value is 0, -1, and -2. When the return value is 0, the radar stops scanning successfully. When -1 indicates that the sending scan command failed, -2 indicates that the sending scan command times out.

Grab Lidar Scan Data

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

Node_info is the lidar data structure. The grabScanData() parameter is the number of lidar data structure instances and a circle of lidar data. The return values are 0, -1, and -2. When the return value is 0, it means that the data acquisition is successful. When -1 indicates that the data acquisition failed, -2 indicates that the data acquisition timed out.

Get SDK Version

```
  const std::string getSDKVersion();
```

The return value is the SDK version number.

Reset

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

When the return value is 0, the device reset is successful.

Get Frequency

```
  result_t getFrequency(uint32_t model, size_t count, float & frequency);
```

The parameter of getFrequency(): obtained value frequency is the scanning frequency of the corresponding model radar. When the return value is 0, the data is successfully retrieved.

Get Sampling Rate

```
  result_t   getSamplingRate(sampling_rate   &   rate,   uint32_t   timeout   =
DEFAULT_TIMEOUT);
```

When the return value is 0, the data is successfully retrieved.

Set Sampling Rate

```
  result_t   setSamplingRate(sampling_rate   &   rate,   uint32_t   timeout   =
DEFAULT_TIMEOUT);
```