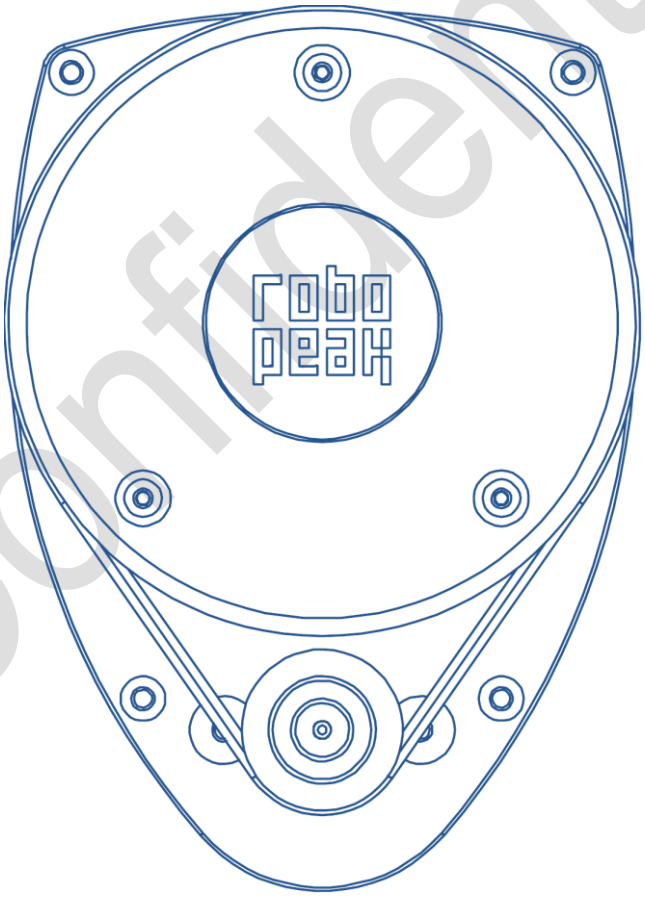
RPLIDAR
低成本 360 度二维激光扫描测距系统

标准版 SDK 使用简介

2014-3



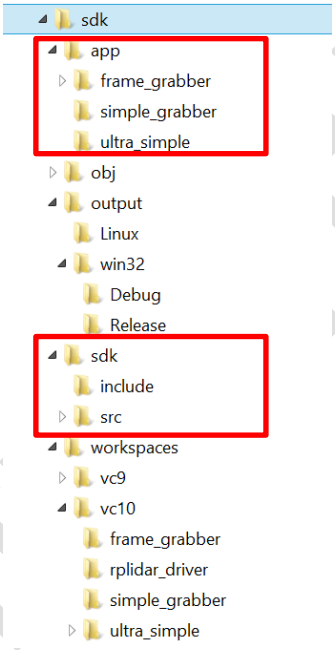
目录:

1. 简介

本文档针对标准开源版本的 RPLIDAR SDK。目前该 SDK 可以在 Windows 和 Linux 环境下使用。采用 Microsoft Visual C++ 2008，2010 和 Makefile 编译。

SDK 文件组织

SDK 的文件结构如下图所示：



workspaces 目录包含了 SDK 和相关示例程序的 VS 工程项目文件。

sdk 目录包含了 RPLIDAR 驱动程序的外部头文件（include 目录）以及 SDK 自身的内部实现代码（src 目录）。

app 目录包含了相关的示例程序代码。RoboPeak 提供了如下几个示例程序：

- ultra_simple

一个极简的命令行的演示程序，实现了连接 RPLIDAR，并不断的输出扫描测距数据。用户可以参考该程序快速的将 RPLIDAR SDK 集成到现有系统当中。

- simple_grabber

一个基于命令行的采集程序，每次执行会采集两圈的雷达数据，并以柱状图的方式呈现。

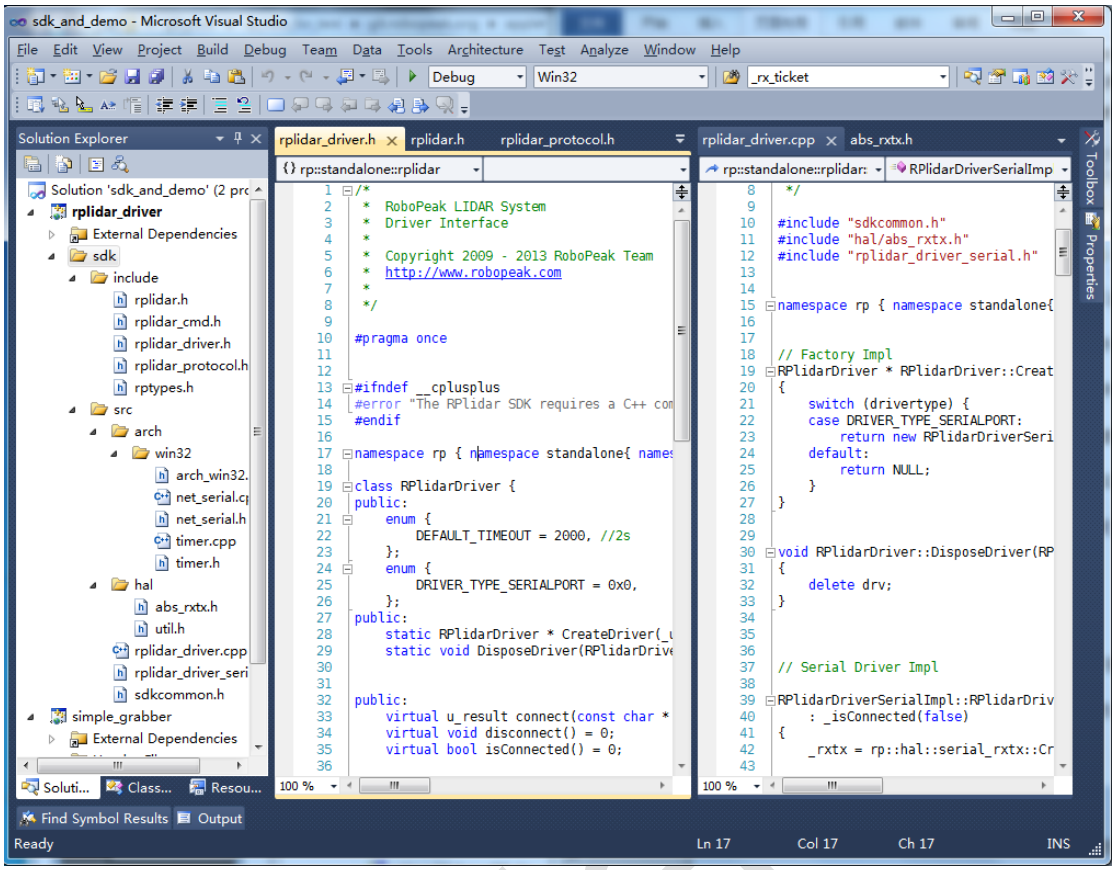
- frame_grabber

一个基于 win32 的 GUI 采集程序，当点击开始采集按钮后，它会把雷达的采集数据实时呈现在界面上。

对于经过编译的 SDK，上述目录结构还会新增 2 个子目录：obj 和 output。其中 output 目录存放了编译产生的 SDK 静态库(.lib 或者.a)以及示例程序的可执行文件(exe 或者 elf 格式)。obj 目录存放了编译过程中的中间文件。

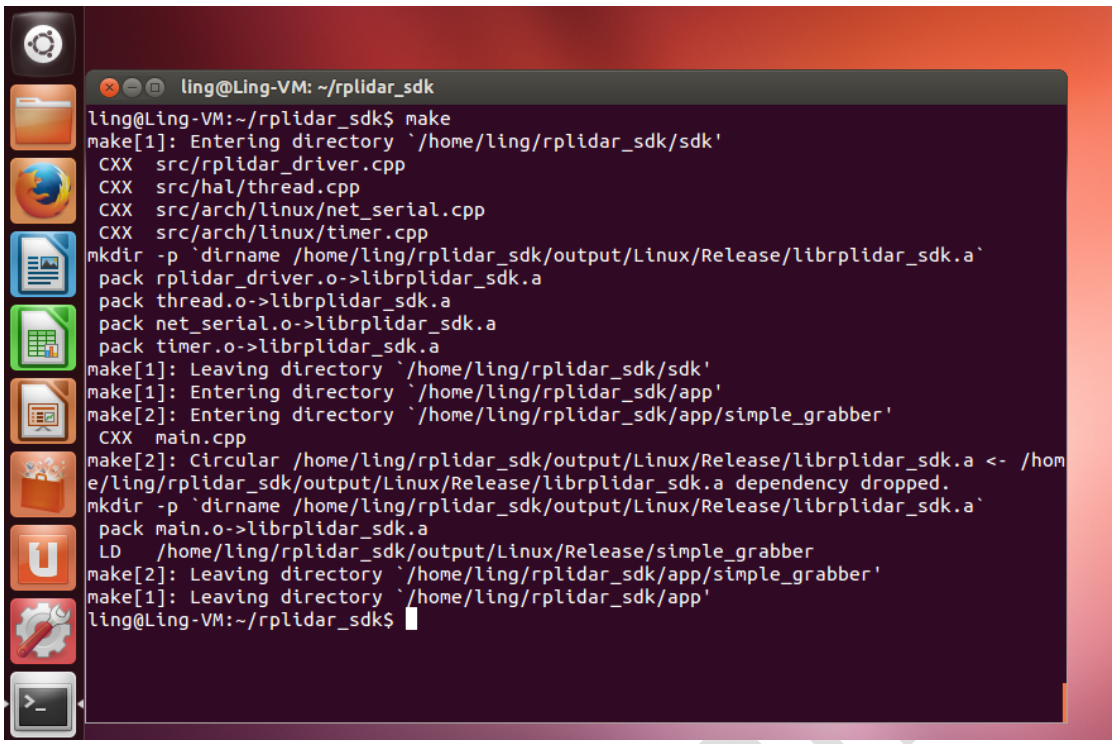
SDK 和实例程序的编译

如果您使用 Windows 进行开发，请打开位于 workspaces\vc10 或者 workspaces\vc9 下的 VS 解决方案文件：sdk_and_demo.sln。其中包含了 SDK 项目工程以及所有的示例程序项目。



您可以直接在 VS 环境中使用编译命令对 SDK 本身以及所有示例程序进行编译。按照开发需要，可以选择 Debug 或者 Release 编译方式。编译结果可以在 output\win32\Debug 或者 output\win32\Release 中找到。

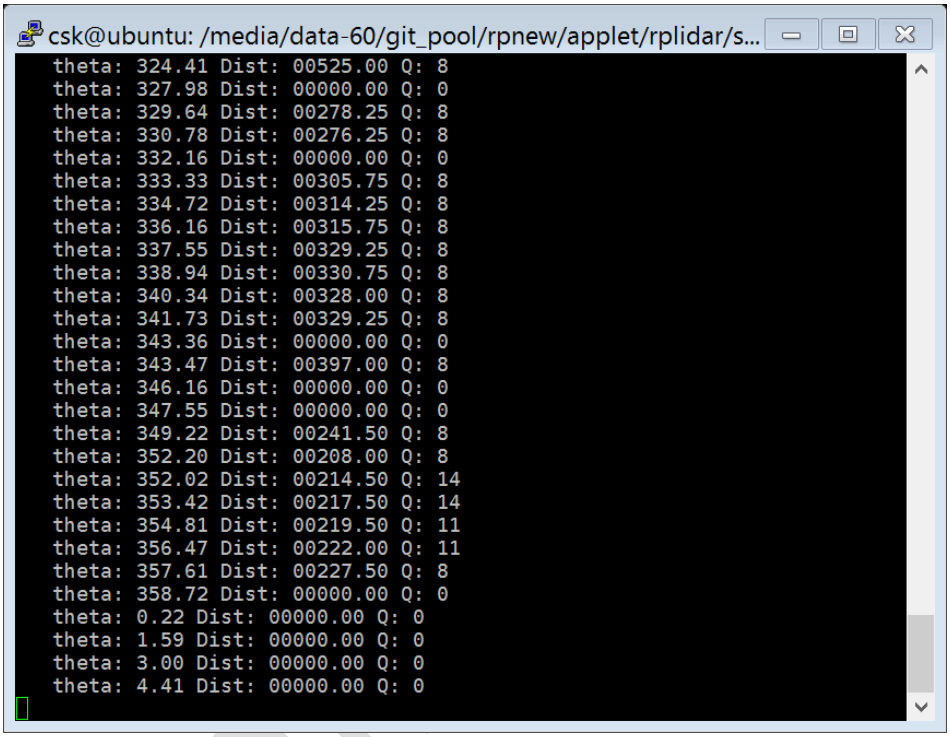
如果您使用 Linux 进行开发，请在 SDK 的根目录运行 make 命令进行编译。默认为 Release 编译方式，您也可以使用 make DEBUG=1 来选择 Debug 编译方式。编译结果可以在 output\Linux\Release 或者 output\Linux\Debug 中找到。



2. 示例程序介绍

ultra_simple

该示例程序演示 PC 通过串口与 RPLIDAR 进行连接，并不断的将 RPLIDAR 扫描数据输出的最简单过程。



使用方式：

1）使用包装里提供的 USB 线连接 RPLIDAR 至 PC 机 （开发板集成了 USB 转串口芯片）

2) 使用如下命令启动本示例程序：

- Windows:

```
ultra_simple <com 号>
```

注意：如果当前的串口编号大于 9，如 com11，则使用如下命令启动程序：

```
ultra_simple \\.\com11
```

如果不指定 COM 设备号，则程序会尝试打开 COM3。

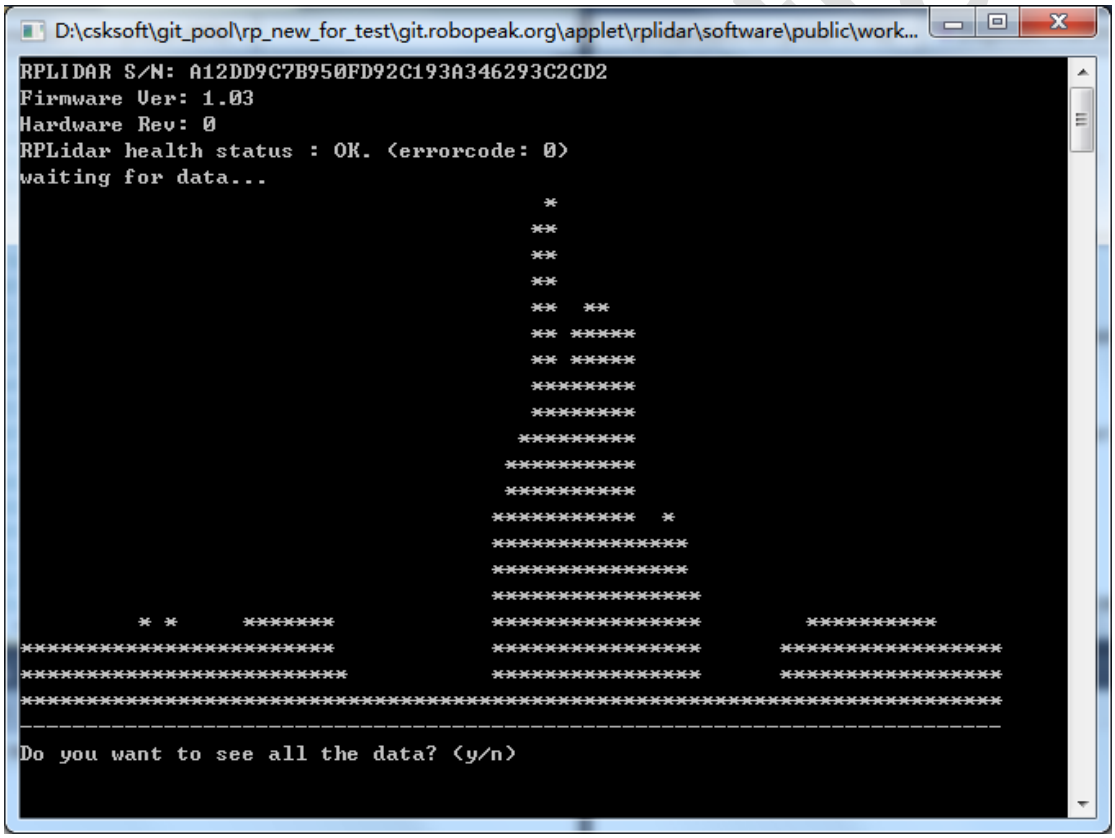
- Linux

`ultra_simple <tty 设备>`

如：`ultra_simple /dev/ttyUSB0`。如果不指定 tty 设备号，则程序默认使用/dev/ttyUSB0 设备。

simple_grabber

该示例程序演示 PC 通过串口与 RPLIDAR 进行连接，并获取 RPLIDAR 序列号、固件版本以及自身健康状况等信息。随后示例程序将采集 2 周的 360 度扫描数据，并采用柱状图的形式在命令行模式下将 0-360 度环境下的测距信息显示出来。用户可以根据需要获取完整的扫描数据。



使用方式：

1）使用包装里提供的 USB 线连接 RPLIDAR 至 PC 机 （开发板集成了 USB 转串口芯片）

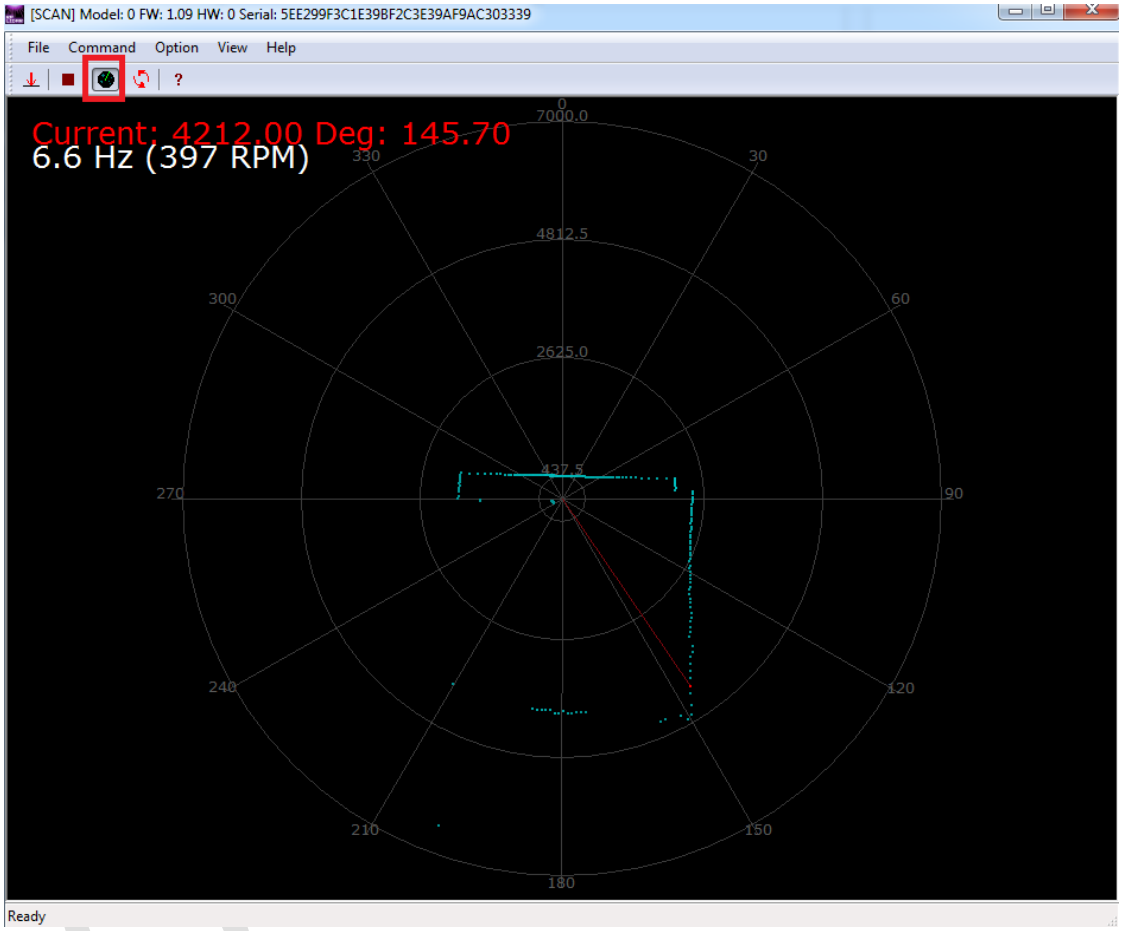
2）使用如下命令启动本示例程序：simple_grabber <com 号>

注意：如果当前的串口编号大于 9，如 com11，则使用如下命令启动程序：

simple_grabber \\.\com11

frame_grabber

该示例程序演示 PC 通过串口与 RPLIDAR 进行连接，实时采集雷达扫描数据，并在 GUI 界面上将 0-360 度环境下的测距信息以平面图的方式显示出来。注意，此示例程序只有 Win32 版本。



使用方式：

1）使用包装里提供的 USB 线连接 RPLIDAR 至 PC 机 （开发板集成了 USB 转串口芯片）

2）从串口对话框中选择正确的串口号

3）点击开始扫描按钮（图中红色框所示）启动扫描

3. SDK 使用和开发指南

注意事项

建议开发人员在使用 RPLIDAR SDK 前，对 RPLIDAR 的通讯协议和工作模式有所了解。可以参考 RPLIDAR 的通讯接口协议与应用文档获取相关细节。

SDK 使用 C++方式开发，这里假设开发人员具有相关知识。

SDK 构成

RPLIDAR 标准版 SDK 采用静态库方式组织，以便开发人员将 SDK 功能整合进自身项目当中。同时也可以通过简单修改工程设置，使用动态库等方式。

开发需要使用 RPLIDAR SDK 的项目时，只需要引用 SDK 的外部头文件(位于 sdk\include 文件夹)。并且在程序的链接阶段，引用 SDK 的静态库(rplidar_driver.lib 或者 rplidar_driver.a)。

另外也可以直接在开发项目当中引入 SDK 的 VC 工程（针对采用 VS 环境开发），并设置对应的项目依赖即可。对于 Linux 项目开发者，您可以参照 simple_grabber 的 Makerfile 进行设置。

运行库一致性

对于 Windows 开发者，采用 SDK 自身项目工程编译得到的 SDK 静态库将采用 VC10 MD 模式的 C 运行库。如果正在开发的项目采用了不同的 C 运行库版本/链接方式，则可能导致程序编译失败、运行时行为怪异问题。此时请修改 SDK 的项目设置，或者使用对应版本的 VS 开发环境重新进行编译。

头文件介绍

- rplidar.h

一般情况下开发的项目中仅需要引入该头文件即可使用 RPLIDAR SDK 的所有功能。

- rplidar_driver.h

定义了SDK核心驱动接口: RPlidarDriver的类声明。请参考ultra_simple或者simple_grabber示例代码了解如何使用该接口。

- rplidar_protocol.h

定义了 RPLIDAR 通讯协议文档中描述的底层相关数据结构和常量定义。

- rplidar_cmd.h

定义了 RPLIDAR 通讯协议文档中描述的各类请求/应答相关的数据结构和常量定义。

- rptypes.h

平台无关的结构和常量定义

SDK 初始化与退出

在用户程序与一个 RPLIDAR 设备进行通讯操作前，首先需要通过 SDK 创建一个对应的 RPlidarDriver 实例。该操作可以通过如下静态函数接口实现:

```
RPlidarDriver *RPlidarDriver::CreateDriver (_u32 drivertype)
```

一个 RPlidarDriver 实例同时只能与系统中的一台 RPLIDAR 进行通讯。但用户程序可以创建任意多个 RPlidarDriver 实例，用于实现对任意多个 RPLIDAR 设备通讯。

在用户程序完成对 RPLIDAR 设备的操作后，需要显式地调用如下静态接口函数析构 RPlidarDriver 实例，从而释放内存：

```
RPlidarDriver::DisposeDriver(RPlidarDriver * drv)
```

连接 RPLIDAR

在创建 RPlidarDriver 实例后，用户程序需要调用 connect()函数进行串口打开并连接到 RPLIDAR 设备。对于 RPLIDAR 的任何操作均要求用户程序事先调用过 connect()函数后进行。

```
u_result RPlidarDriver::connect(const char * port_path, _u32 baudrate, _u32 flag = 0)
```

如果连接完成，该函数将返回 RESULT_OK。

在完成了 RPLIDAR 设备通讯后，用户程序可以调用 disconnect()函数断开 RPLIDAR 设备的连接。

测距扫描与扫描数据获取

对于 RPLIDAR 测距扫描的操作和数据获取涉及到了如下函数：

函数名	简介
startScan()	请求 RPLIDAR 核心开始进行测距扫描，开始输出数据
stop()	请求 RPLIDAR 核心停止测距扫描
grabScanData()	抓取一圈扫描测距数据序列

用户程序首先需要调用 startScan()函数让 RPLIDAR 核心开始测距扫描。在 RPLIDAR 的旋转速度趋于稳定后，将开始持续向外输出扫描测距数据。

startScan()函数将启动一个后台工作线程，异步的接受来自 RPLIDAR 的扫描测距数据序列，并保存在内部的缓冲当中。供 grabScanData()函数获取。

用户程序需要通过 grabScanData()函数抓取被 RPLIDAR 驱动事先接受并缓存的测距数据序列。该函数将始终返回一个最新的完整的 360 度的扫描测距序列。在一次 grabScanData()调用后，保存扫描数据序列的内部缓存将会清空，以确保每次的 grabScanData()调用将始终获得不重复的数据。

如果在 grabScanData()调用时，一圈完整的 360 度的扫描测距序列尚未接受完毕，则该函数将进行等待，直到获得了完整的扫描数据或者超过了等待时间。用户可以指定每次函数的最大等待时间以适应不同应用的需求。

注意：startScan()与 stop()函数并不会控制 RPLIDAR 的实际转动或者停止。外部系统需要使用 PWM 驱动电路来控制扫描电机实现该功能。

请参考头文件的相关注释以及 SDK 配套的演示程序的实现了解上述函数的具体使用方法。

获取 RPLIDAR 设备的其他信息

用户程序也可以通过如下函数获取 RPLIDAR 设备的其他信息。具体的使用请参考文件的相关注释以及 SDK 配套的演示程序的实现。

函数名	简介
getHealth ()	获取 RPLIDAR 设备的健康状态
getDeviceInfo ()	获取 RPLIDAR 设备序列号、固件版本等信息
getFrequency ()	从实现抓取的一圈扫描数据序列计算 RPLIDAR 的转速

4. 修订历史

日期	内容
2013-3-5	初稿
2014-1-25	增加 Linux 支持，更新相关内容
2014-3-8	1. 增加对 ultra_simple 演示程序的描述 2. 增加对 SDK 主要函数的使用介绍