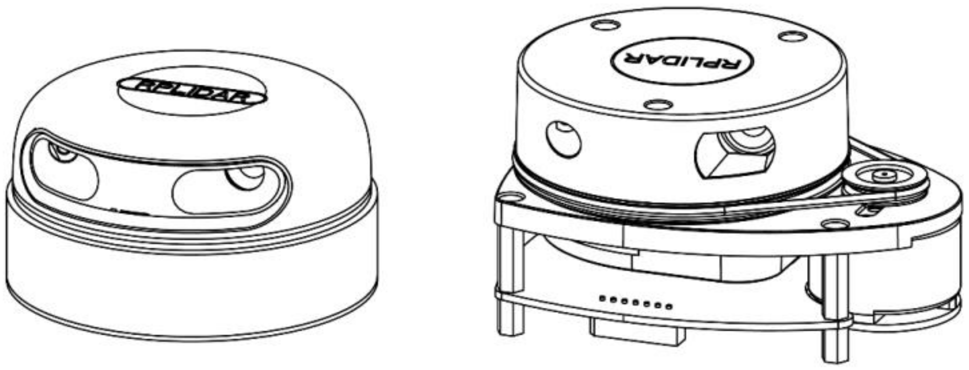
SLAMTEC

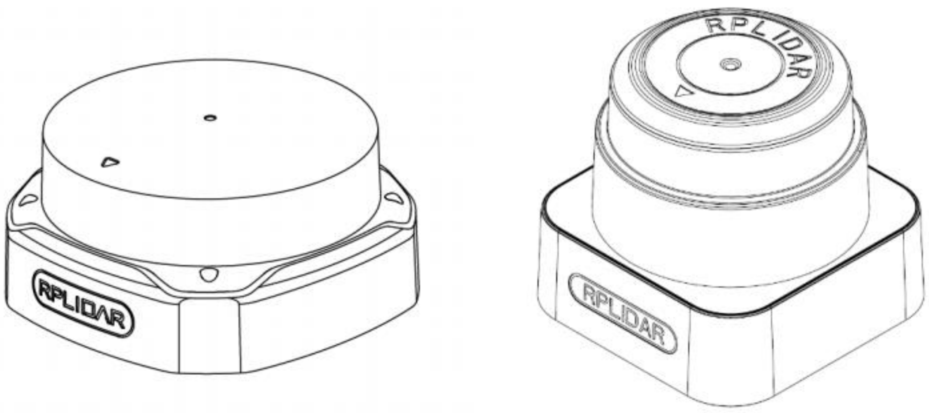
RPLIDAR

Low Cost 360 Degree Laser Range Scanner

Introduction to Standard SDK

2022-04-1 rev.2.0





www.slamtec.com

Shanghai Slamtec.Co.,Ltd

This document introduces the standard open source version of the RPLIDAR SDK V2.0. The SDK uses Microsoft Visual C++ 2010, Microsoft Visual C++ 2019 and Makefile to compile under the environment of Windows, MacOS (10.x) and Linux. RPLIDAR SDK V2.0 is designed in principle and compatible with v1.x version through macro definition.

SDK Organization

The RPLIDAR standard SDK organized as bellow:

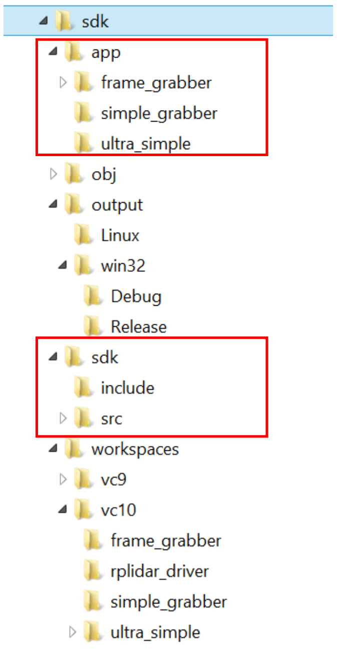

```
sdk
  app
    frame_grabber
    simple_grabber
    ultra_simple
  obj
  output
    Linux
    win32
      Debug
      Release
  sdk
    include
    src
  workspaces
    vc9
    vc10
      frame_grabber
      rplidar_driver
      simple_grabber
      ultra_simple
```

Figure 1-1 The RPLIDAR standard SDK organization

The **workspaces** directory contains VS project files for SDK and related demo applications.

The **sdk** directory contains the external header files(the **include** folder) of RPLIDAR driver application and the internal implementation code of SDK itself(the **src** folder).

The **app** directory contains the related sample code. And SLAMTEC provides the following demo applications in the app directory:

ultra_simple

An ultra-simple command line application demonstrates the simplest way to connect to an RPLIDAR device and continuously fetching the scan data and outputting the data to the console. Users can quickly integrate RPLIDAR to their existing system based on this demo application.

simple_grabber

A command line grab application. Each execution will grab two round of laser data and show as histogram.

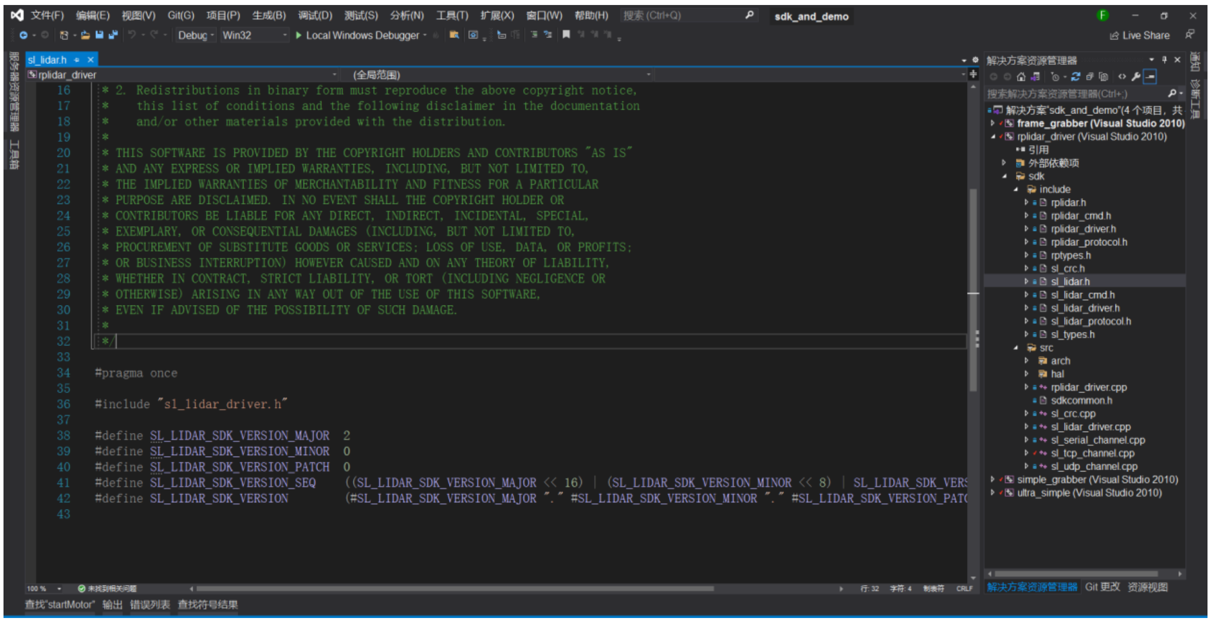
frame_grabber

A win32 GUI grab application. When pressing start scan button, it will start scan continuously and show the data in the UI.

For SDK after compilation, there will be two more subfolders in the SDK: **obj** and **output**. The **output** folder contains generated SDK static library (.lib or .a) and demo application executable files (exe or elf). obj folder contains intermediate files generated during compilation.

Build SDK and Demo Applications

If you're developing under Windows, please open VS solution file **sdk_and_demo.sln** under workspaces\vc10 or workspaces\vc14. It contains the SDK project and all demo application projects.


文件(F) 编辑(E) 视图(V) Git(G) 项目(P) 生成(B) 调试(D) 测试(S) 分析(N) 工具(T) 扩展(X) 窗口(W) 帮助(H)
搜索 (Ctrl+Q)
sdk_and_demo
Debug
Win32
Local Windows Debugger
Live Share
sl_lidar.h
rplidar_driver
(全局范围)
100 %
未找到相关问题
行: 32 字符: 4 制表符 CRLF
查找"startMotor" 输出 错误列表 查找符号结果
解决方案资源管理器 Git 更改 资源视图

```
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

#pragma once

#include "sl_lidar_driver.h"

#define SL_LIDAR_SDK_VERSION_MAJOR  2
#define SL_LIDAR_SDK_VERSION_MINOR  0
#define SL_LIDAR_SDK_VERSION_PATCH  0
#define SL_LIDAR_SDK_VERSION_SEQ    ((SL_LIDAR_SDK_VERSION_MAJOR << 16) | (SL_LIDAR_SDK_VERSION_MINOR << 8) | SL_LIDAR_SDK_VERS
#define SL_LIDAR_SDK_VERSION        (#SL_LIDAR_SDK_VERSION_MAJOR "." #SL_LIDAR_SDK_VERSION_MINOR "." #SL_LIDAR_SDK_VERSION_PATC
```

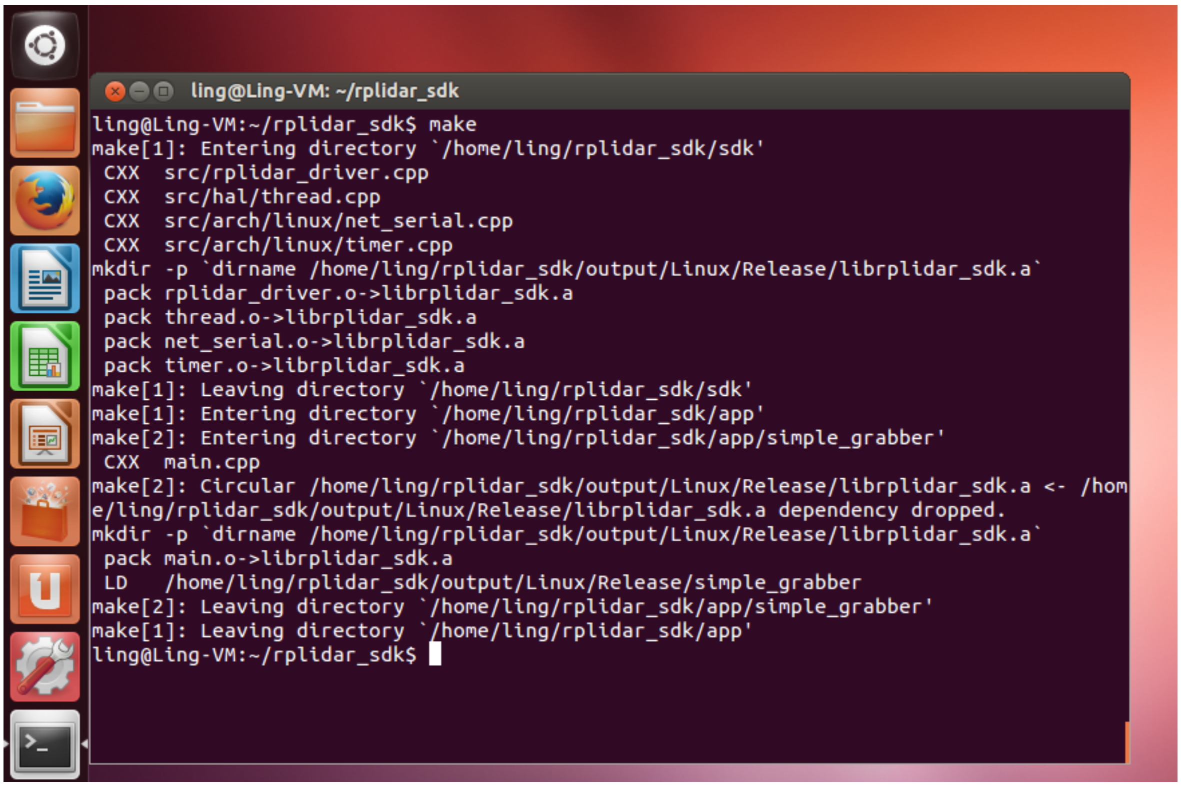
```
解决方案资源管理器
搜索解决方案资源管理器(Ctrl+;)
解决方案"sdk_and_demo"(4 个项目, 共
  frame_grabber (Visual Studio 2010)
  rplidar_driver (Visual Studio 2010)
    引用
    外部依赖项
    sdk
      include
        rplidar.h
        rplidar_cmd.h
        rplidar_driver.h
        rplidar_protocol.h
        rptypes.h
        sl_crc.h
        sl_lidar.h
        sl_lidar_cmd.h
        sl_lidar_driver.h
        sl_lidar_protocol.h
        sl_types.h
      src
        arch
        hal
        rplidar_driver.cpp
        sdkcommon.h
        sl_crc.cpp
        sl_lidar_driver.cpp
        sl_serial_channel.cpp
        sl_tcp_channel.cpp
        sl_udp_channel.cpp
  simple_grabber (Visual Studio 2010)
  ultra_simple (Visual Studio 2010)
```

Figure 1-2 The RPLIDAR Standard SDK Solution File

Now, you can develop SDK and all the demo applications in VS environment easily. There are two ways of compiling: Debug and Release. And you can choose according to your requirement. Depends on your build configure, the generated binary can be found under output\win32\Release or output\win32\Debug.

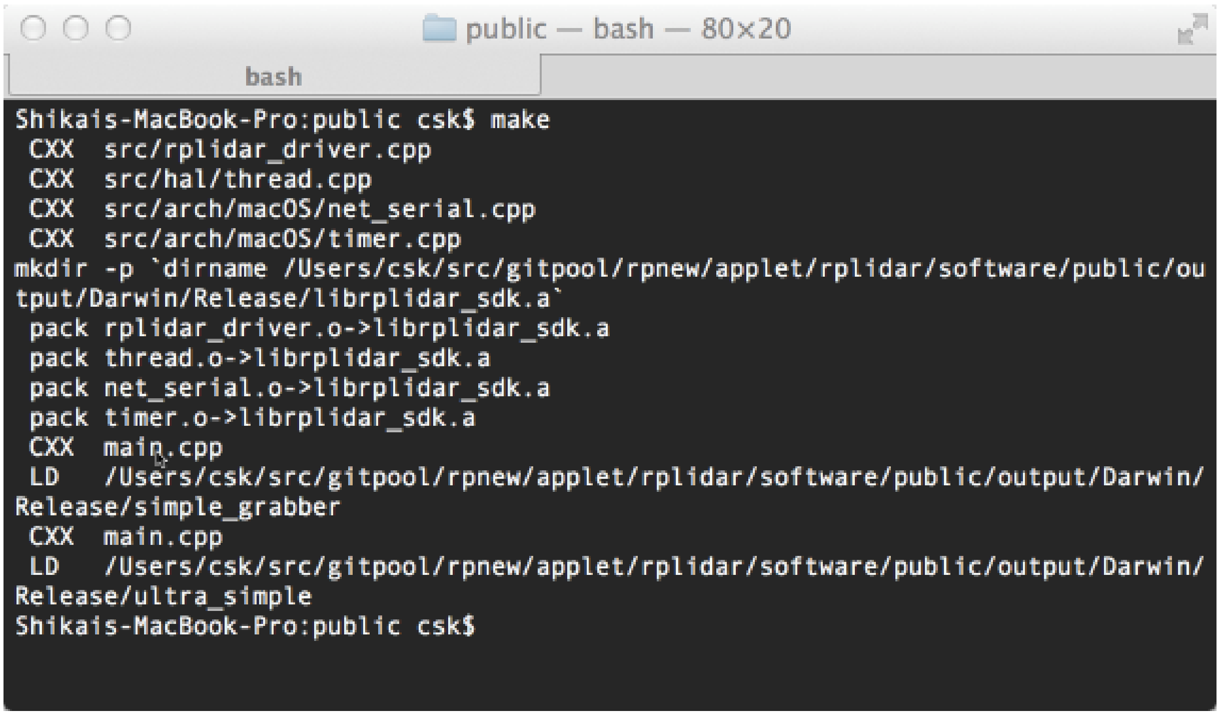
If you're developing under Linux or MacOS, just type "make" under the root of SDK directory to start compiling. It will do Release build by default, and you can also type "make DEBUG=1" to do Debug build. The generated binary can be found under the following folders:

- Linux
  - output\Linux\Release
  - output\Linux\Debug.
- MacOS
  - output\Darwin\Release
  - output\Darwin\Debug.



```
ling@Ling-VM:~/rplidar_sdk$ make
make[1]: Entering directory `/home/ling/rplidar_sdk/sdk'
 CXX  src/rplidar_driver.cpp
 CXX  src/hal/thread.cpp
 CXX  src/arch/linux/net_serial.cpp
 CXX  src/arch/linux/timer.cpp
mkdir -p `dirname /home/ling/rplidar_sdk/output/Linux/Release/librplidar_sdk.a`
 pack rplidar_driver.o->librplidar_sdk.a
 pack thread.o->librplidar_sdk.a
 pack net_serial.o->librplidar_sdk.a
 pack timer.o->librplidar_sdk.a
make[1]: Leaving directory `/home/ling/rplidar_sdk/sdk'
make[1]: Entering directory `/home/ling/rplidar_sdk/app'
make[2]: Entering directory `/home/ling/rplidar_sdk/app/simple_grabber'
 CXX  main.cpp
make[2]: Circular /home/ling/rplidar_sdk/output/Linux/Release/librplidar_sdk.a <- /hom
e/ling/rplidar_sdk/output/Linux/Release/librplidar_sdk.a dependency dropped.
mkdir -p `dirname /home/ling/rplidar_sdk/output/Linux/Release/librplidar_sdk.a`
 pack main.o->librplidar_sdk.a
 LD   /home/ling/rplidar_sdk/output/Linux/Release/simple_grabber
make[2]: Leaving directory `/home/ling/rplidar_sdk/app/simple_grabber'
make[1]: Leaving directory `/home/ling/rplidar_sdk/app'
ling@Ling-VM:~/rplidar_sdk$
```

Figure 1-3 Develop RPLIDAR Standard SDK in Linux



```
Shikais-MacBook-Pro:public csk$ make
 CXX  src/rplidar_driver.cpp
 CXX  src/hal/thread.cpp
 CXX  src/arch/macOS/net_serial.cpp
 CXX  src/arch/macOS/timer.cpp
mkdir -p `dirname /Users/csk/src/gitpool/rpnew/applet/rplidar/software/public/ou
tput/Darwin/Release/librplidar_sdk.a`
 pack rplidar_driver.o->librplidar_sdk.a
 pack thread.o->librplidar_sdk.a
 pack net_serial.o->librplidar_sdk.a
 pack timer.o->librplidar_sdk.a
 CXX  main.cpp
 LD   /Users/csk/src/gitpool/rpnew/applet/rplidar/software/public/output/Darwin/
Release/simple_grabber
 CXX  main.cpp
 LD   /Users/csk/src/gitpool/rpnew/applet/rplidar/software/public/output/Darwin/
Release/ultra_simple
Shikais-MacBook-Pro:public csk$
```

Figure 1-4 Develop RPLIDAR Standard SDK in MacOS

Cross Compile

The SDK build system allows you to generate binaries which run on another platform/system using the cross-compiling feature.

NOTE: this feature only works on where using Makefile.

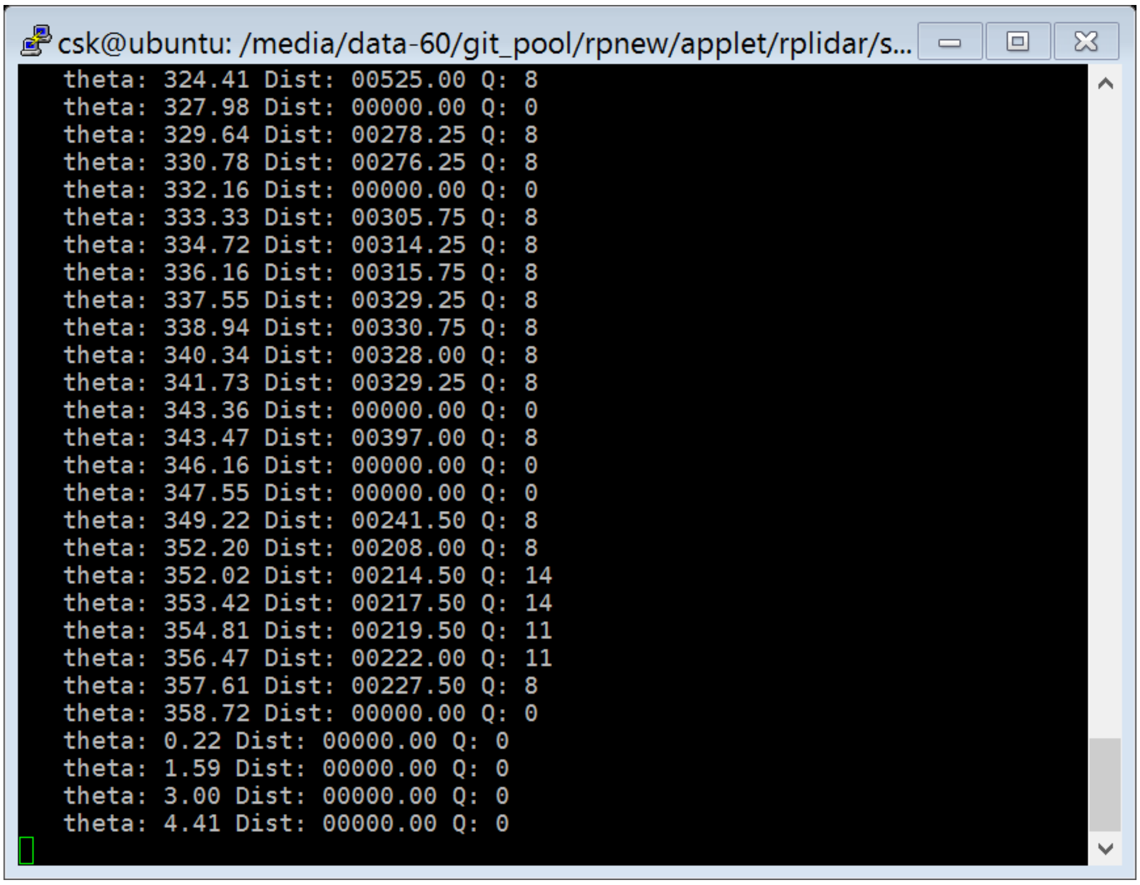
The cross compile process can be triggered by invoking the cross_compile.sh script under the SDK root folder. The common usage is:

CROSS_COMPILE_PREFIX=<COMPILE_PREFIX> ./cross_compile.sh

e.g. `CROSS_COMPILE_PREFIX=arm-linux-gnueabihf ./cross_compile.sh`

ultra_simple

The demo application simply connects to an RPLIDAR device and outputs the scan data to the console:



csk@ubuntu: /media/data-60/git_pool/rpnew/applet/rplidar/s...

```
theta: 324.41 Dist: 00525.00 Q: 8
theta: 327.98 Dist: 00000.00 Q: 0
theta: 329.64 Dist: 00278.25 Q: 8
theta: 330.78 Dist: 00276.25 Q: 8
theta: 332.16 Dist: 00000.00 Q: 0
theta: 333.33 Dist: 00305.75 Q: 8
theta: 334.72 Dist: 00314.25 Q: 8
theta: 336.16 Dist: 00315.75 Q: 8
theta: 337.55 Dist: 00329.25 Q: 8
theta: 338.94 Dist: 00330.75 Q: 8
theta: 340.34 Dist: 00328.00 Q: 8
theta: 341.73 Dist: 00329.25 Q: 8
theta: 343.36 Dist: 00000.00 Q: 0
theta: 343.47 Dist: 00397.00 Q: 8
theta: 346.16 Dist: 00000.00 Q: 0
theta: 347.55 Dist: 00000.00 Q: 0
theta: 349.22 Dist: 00241.50 Q: 8
theta: 352.20 Dist: 00208.00 Q: 8
theta: 352.02 Dist: 00214.50 Q: 14
theta: 353.42 Dist: 00217.50 Q: 14
theta: 354.81 Dist: 00219.50 Q: 11
theta: 356.47 Dist: 00222.00 Q: 11
theta: 357.61 Dist: 00227.50 Q: 8
theta: 358.72 Dist: 00000.00 Q: 0
theta: 0.22 Dist: 00000.00 Q: 0
theta: 1.59 Dist: 00000.00 Q: 0
theta: 3.00 Dist: 00000.00 Q: 0
theta: 4.41 Dist: 00000.00 Q: 0
```

Figure 2-1 ultra_simple Demo Application Data Output

Steps:

1) According to the lidar model and configuration, the following three connection modes are available:

a) Connect RPLIDAR to PC by using the provided USB cable and PIC. (The chip transforming the USB to serial port is embedded in the RPLIDAR development kit)

b) Connect RPLIDAR to PC by using the provided Ethernet module. (using TCP/IP protocol)

c) Connect RPLIDAR to PC by using the body network cable. (using UDP protocol)

2) According to the connection mode, the following command is used to start the sample program:

a) ultra_simple --channel --serial <com port> [baudrate]

<com port>: the serial port #

Windows:	\\.\com3
MacOS:	/dev/tty.SLAB_USBtoUART
Linux:	/dev/ttyUSB0

[baudrate]: the communication baudrate

A2: 115200

A3/S1: 256000

- S2: 1000000

b) ultra_simple --channel --tcp <ip> [port]

<ip>: the IP address of the radar or Ethernet module

[port]: Port number，20108

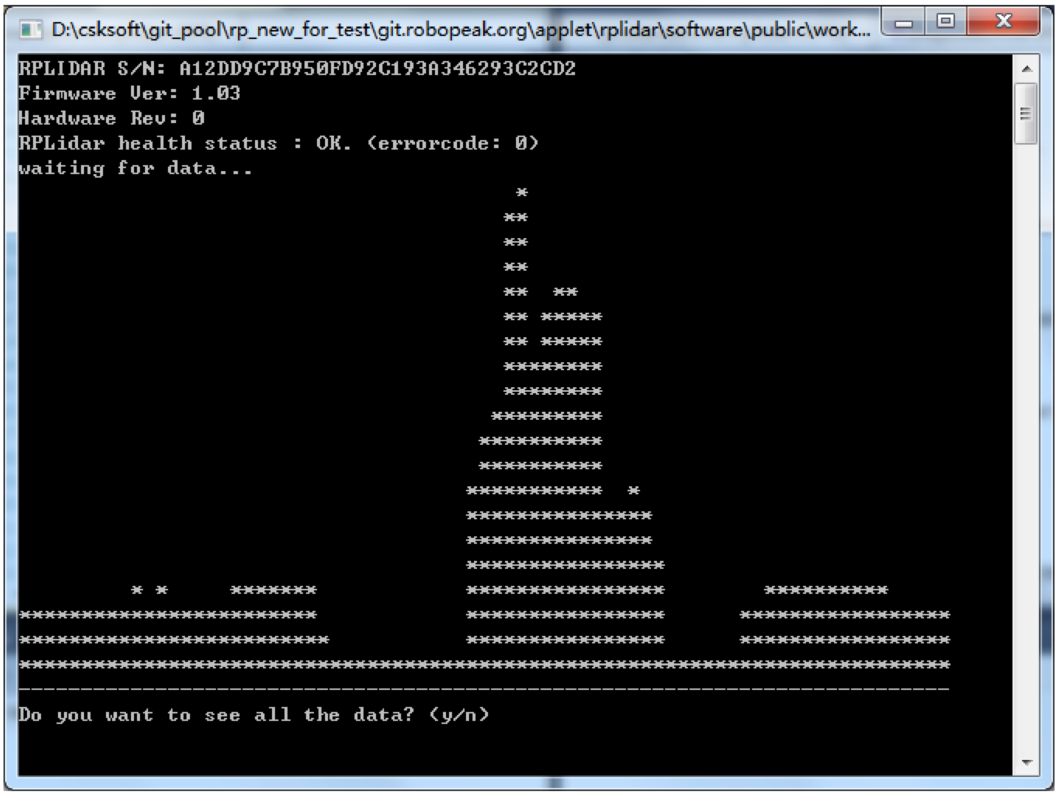
c) ultra_simple --channel --udp <ip> [port]

<ip>: the IP address of the radar

[port] : Port number，8089

simple_grabber

This application demonstrates the process of getting RPLIDAR's serial number, firmware version and healthy status after connecting the PC and RPLIDAR. Then the demo application grabs two round of scan data and shows the range data as histogram in the command line mode. User can print all scan data if needed.


D:\csksoft\git_pool\rp_new_for_test\git.robopeak.org\applet\rplidar\software\public\work...

```
RPLIDAR S/N: A12DD9C7B950FD92C193A346293C2CD2
Firmware Ver: 1.03
Hardware Rev: 0
RPLidar health status : OK. (errorcode: 0)
waiting for data...
                                        *
                                       **
                                       **
                                       **
                                       **  **
                                       ** *****
                                       ** *****
                                       ********
                                       ********
                                      *********
                                     **********
                                     **********
                                    ***********  *
                                    ***************
                                    ***************
                                    ****************
         * *     *******            ****************        **********
************************            ****************      *****************
*************************           ****************      *****************
***************************************************************************
---------------------------------------------------------------------------
Do you want to see all the data? (y/n)
```

Figure 2-2 simple_grabber Demo Application Data Output

Steps:

1) According to the lidar model and configuration, the following three connection modes are available:

a) Connect RPLIDAR to PC by using the provided USB cable and PIC. (The chip transforming the USB to serial port is embedded in the RPLIDAR development kit)

b) Connect RPLIDAR to PC by using the provided Ethernet module. (using TCP/IP protocol)

c) Connect RPLIDAR to PC by using the body network cable. (using UDP protocol)

2) According to the connection mode, the following command is used to start the sample program:

a) ultra_simple --channel --serial <com port> [baudrate]

<com port>: the serial port #

- Windows: \\.\com3
- MacOS: /dev/tty.SLAB_USBtoUART
- Linux: /dev/ttyUSB0

[baudrate]: the communication baudrate

- A2: 115200
- A3/S1: 256000
- S2: 1000000

b) ultra_simple --channel --tcp <ip> [port]

<ip>: the IP address of the radar or Ethernet module

[port]: Port number，20108

c) ultra_simple --channel --udp <ip> [port]

<ip>: the IP address of the radar

[port] : Port number，8089frame_grabber

This demo application can show real-time laser scan data in the GUI with 0-360 degree environment range data. Note, this demo application only has win32 version.

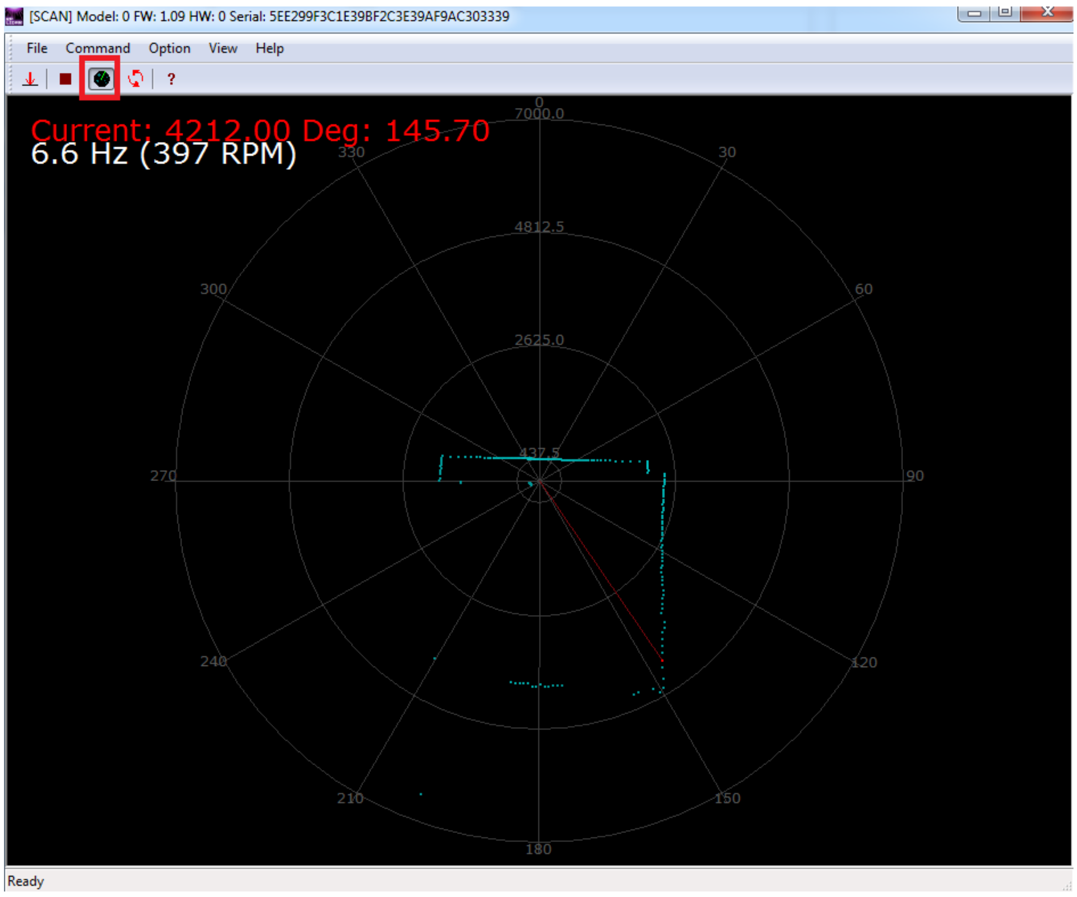

[SCAN] Model: 0 FW: 1.09 HW: 0 Serial: 5EE299F3C1E39BF2C3E39AF9AC303339
File
Command
Option
View
Help
Current: 4212.00 Deg: 145.70
6.6 Hz (397 RPM)
Ready

Figure 2-3 frame_grabber Demo Application Data Output

Steps:

1) According to the lidar model and configuration, the following three connection modes are available:

a) Connect RPLIDAR to PC by using the provided USB cable and PIC. (The chip transforming the USB to serial port is embedded in the RPLIDAR development kit)

b) Connect RPLIDAR to PC by using the provided Ethernet module. (using TCP/IP protocol)

c) Connect RPLIDAR to PC by using the body network cable. (using UDP protocol)

2) According to the connection mode, the following command is used to start the sample program:

a) Connect RPLIDAR to PC by using the provided USB cable and PIC. (The chip transforming the USB to serial port is embedded in the RPLIDAR development kit)

Select the correct serial port number in the dialog box.

Click the Start scan button (shown in the red box) to start the scan.

b) Always enter the correct IP address and port number in the dialog box

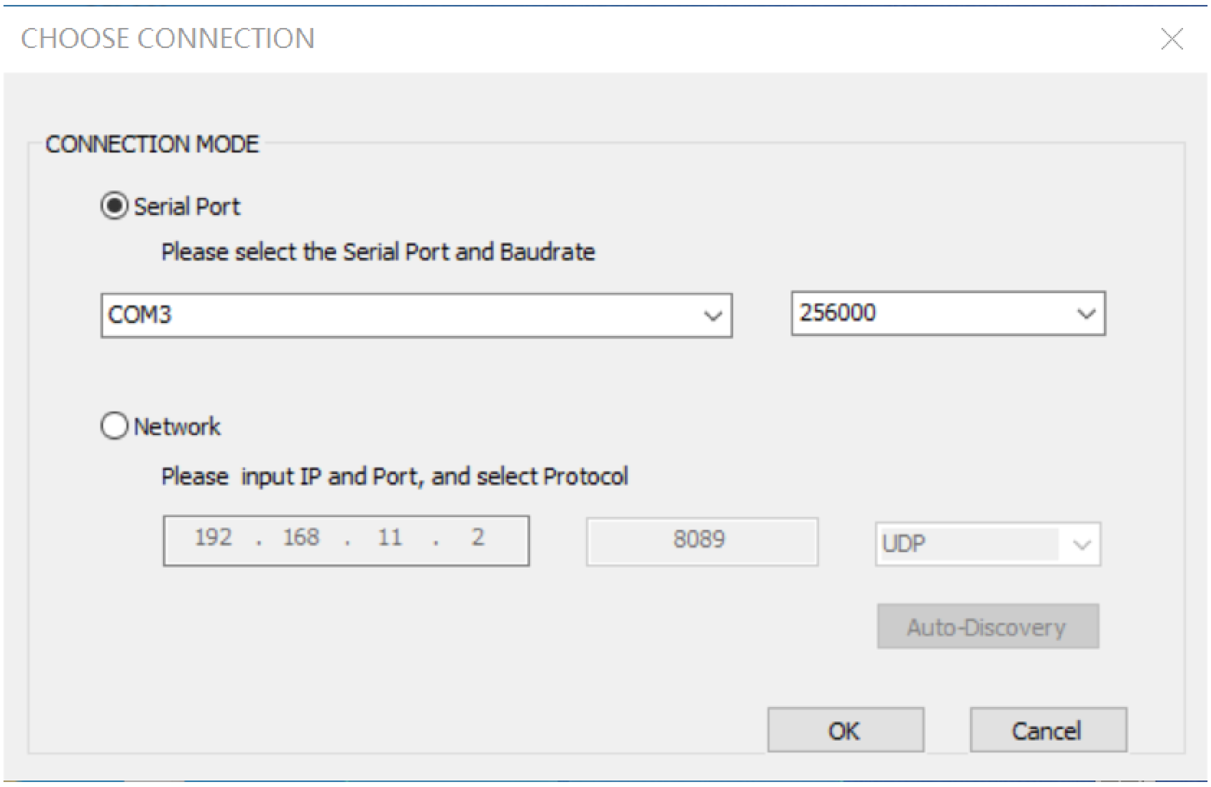
Select TCP and click the OK button.

Click the Start scan button (shown in the red box) to start the scan.

c) Always enter the correct IP address and port number in the dialog box.

Select UDP and click the OK button.

Click the Start scan button (shown in the red box) to start the scan.


CHOOSE CONNECTION

CONNECTION MODE

(●) Serial Port

Please select the Serial Port and Baudrate

COM3 | 256000

( ) Network

Please input IP and Port, and select Protocol

192 . 168 . 11 . 2 | 8089 | UDP

Figure 2-4 frame_grabber Demo Application Dialog

Assumption

We strongly recommend developers learn RPLIDAR's communication protocol and working mode before starting development by using the RPLIDAR SDK.

This document assumes developer has related knowledge about C++ development.

SDK usage

The RPLIDAR standard SDK provides static library for developers to integrated the SDK feature into their existing project. If the dynamic library is required, the developers can also set it by easily modifying the project configurations.

When developing via RPLIDAR SDK, developers only need to include SDK's external header files (under sdk\include) into their own source code and link the application with SDK's static library (rplidar_driver.lib or rplidar_driver.a).

For VS developers, they can also include the SDK's VC project into their own project and set the related project dependence. For Linux developers, please refer to the simple_grabber's Makefile for detailed settings.

PS: To be compatible with earlier SDK, static libraries are still named rplidar_drvier. Please check rplidar_driver.h, sl_lidar_driver.h for details.

Runtime consistency

For windows developers: the SDK static library c   uses VC10 MD C runtime library. If your project used different C runtime library may lead to compilation failure or unpredictable behavior. Then please change SDK settings accordingly.

SDK Headers

- sl_lidar.h

Usually, you only need to include this file to get all functions of RPLIDAR SDK.

- sl_lidar_driver.h

This header defines the SDK core drive interface: the RPLidarDriver class. Please refer to demo applications ultra_simple or simple_grabber to understand how to use it.

- sl_lidar_protocol.h

This header defines low-level data structures and constants for RPLIDAR protocol.

- sl rplidar_cmd.h

This header defines request/answer data structures and constants for RPLIDAR protocol.

- sl rptypes.h

This header defines platform-independent data structures and constants.

SDK Initialization and Termination

The communication between user program and an RPLIDAR device can be made by doing below actions (Details can be checked from sample program, for example, ultra simple):

1) Create a LIDAR driver instance.

```
ILidarDriver * drv = *createLidarDriver();
```

Create a corresponding communication channel based on different connections of Lidar.

a) Connect RPLIDAR to PC by using the provided USB cable and PIC.

```
   *  \param device Serial port device
   *  e.g. on Windows, it may be com3 or \\.\com10
   *  on Unix-Like OS, it may be /dev/ttyS1, /dev/ttyUSB2, etc
   *  \param baudrate Baudrate
   *  Please refer to the datasheet for the baudrate (maybe 115200 or
       256000)
   */
   std::string device = "com3";
   int          baudrate = 115200;
   Result<IChannel*> channel = createSerialPortChannel("/dev/ttyUSB0",
115200);
```

b) Connect RPLIDAR to PC by using the provided Ethernet module. (using TCP/IP protocol)

```
Result< IChannel *> channel = createTcpChannel("192.168.11.2", 20108);
```

c) Connect RPLIDAR to PC by using the body network cable. (using UDP protocol)

```
Result< IChannel *> channel = createTcpChannel("192.168.11.2",8089);
```

An ILidarDriver instance can communicate with only one RPLIDAR in the system at the same time. However a user program can create any several ILidarDriver instances to communicate with any several RPLIDAR devices.

Once user programs finish operations, all previously created lidar driver instances should be released explicitly using the following static function in order to free system memory.

```
delete drv;

delete channel;
```

Connecting to an RPLIDAR

After creating an IlidarDriver instance, the user program should invoke the `connect()` function firstly to open the serial port and connect with the RPLIDAR device. All RPLIDAR operations require invoking the `connect()` function first.

```
auto res = (*lidar)->connect(*channel);
```

The function returns RESULT_OK for success operation.

Once the user program finishes operation, it can call the `disconnect()` function to close the connection and release the serial port device.

Measurement Scan and Data Acquiring

The following functions are related to the measurement scan operation and help user programs to acquire the measurement data:

Function Name	Brief description
startScan()	Request the RPLIDAR core to start measurement scan operation and send out result data continuously SDK will use Express Scan mode automatically, when it is supported and the program invokes the startScan() by using the default parameters
startScanExpress()	Force the RPLIDAR core to start measurement scan operation in Express Scan mode. It the RPLIDAR firmware not supports Express Scan mode, the function will fail the execution.
stop()	Request the RPLIDAR core to stop the me as asurement scan operation.
grabScanDataHg()	Grab a complete 360-degrees' scan data sequence.
ascendScanData()	Rank the scan data from grabScanData() as the angle inscreases.

Figure 3-1 RPLIDAR Functions related to Measurement Scan Operation

The `startScan()` function will start a background worker thread to receive the measurement scan data sequence sent from RPLIDAR asynchronously. The received data sequence is stored in the driver's internal cache for the `grabScanDataHg()` function to fetch.

User programs can use the `grabScanDataHg()` function to retrieve the scan data sequence previously received and cached by the driver. This function always returns a latest and complete 360-degrees' measurement scan data sequence. After each `grabScanDataHg()` call, the internal data cache will be cleared to ensure the `grabScanDataHg()` won't get duplicated data.

In case a complete 360-degrees' scan sequence hasn't been available at the time when `grabScanDataHg()` is called, the function will wait until a complete scan data is received by the driver or the given timeout duration is expired. User programs can tune this timeout value to meet different application requirements.

Please refer to the comments in the header files and the implementation of SDK demo applications for details.

Retrieving Other Information of an RPLIDAR

The user program can retrieve other information of an RPLIDAR via the following functions. Please refer to the comments in the header files and the implementation of SDK demo applications for details.

Function Name	Brief description
getHealth()	Get the healthy status of an RPLIDAR
getDeviceInfo()	Retrieve the device information, e.g. serial number, firmware version etc, from and RPLIDAR
getFrequency()	Calculate an RPLIDAR's scanning speed from a complete scan sequence.

Figure 3-2 RPLIDAR Functions related to Retrieving Other Information Operation

Revision History

Date	Description
2013-3-5	Initial version
2014-1-25	Added Linux content and updated related information
2014-3-8	Added description for the ultra_simple demo application
2014-7-25	Added description about developing under MacOS environment Added description about the cross compiling
2016-4-12	Added description about the newly added firmware interface of 1.5.1
2016-5-3	Added description about the newly added interface of 1.5.2 Added new interface startMotor/stopMotor Updated connect interface and set stopMotor called by default
2017-5-15	Release 1.0 Version
2022-4-1	Release 2.0 Version

Image and Table Index