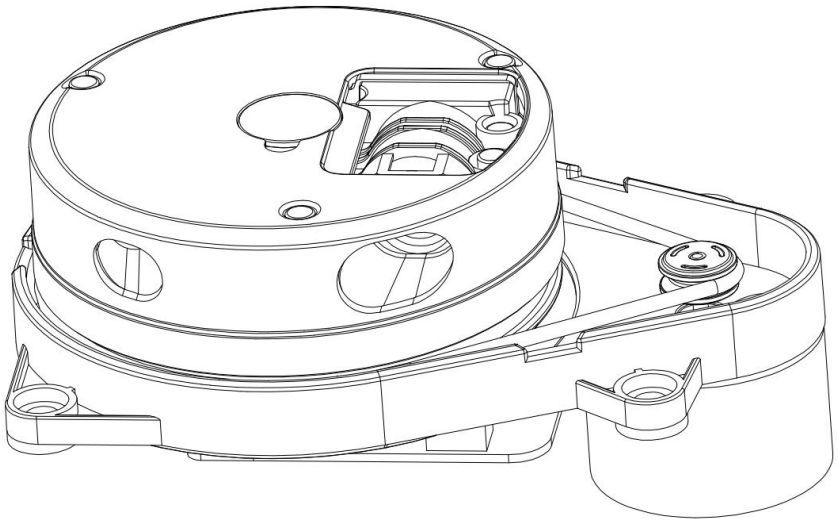
Delta-2A Radar SDK User Guide

[Model: Delta-2A 5k/s, 8m]

version **Version: V1.0**



Shenzhen Shanchuan Robotics Co., Ltd.

Order record

1. SDK Introduction.....3

2. Preparation before SDK compilation and debugging.....4

2.1 Connect correctly according to the radar manual..... 4

2.2 Installing USB to Serial Driver..... 4

3. Windows Platform SDK.....4

3.1 Development Environment..... 4

3.2 Windows SDK Solution..... 4

3.3 Solution Path..... 5

3.4 Solution Engineering Drawings..... 5

3.5 Obtaining the initial information of a complete circle of points (without calculating the angle) function..... 5

3.6 Create point cloud project frame_grabber.....7

3.7 Radar driver simple application project ultra_simple.....8

4. Linux Platform SDK.....9

4.1 Development Environment..... 9

4.2 Directory Structure..... 9

4.3 Radar class interface functions provided by SDK.....10

4.4 Example Application.....10

5. Ros Platform SDK.....11

5.1 Development Environment.....11

5.2 Directory Structure.....11

5.3 Function Interface Reference.....12

5.4 Starting publisher and subscriber routines.....12

5.5 Start rviz to create a point cloud map.....13

6. Muc Platform DEMO..... 13

6.1 Development Environment: Stm32f103cb..... 13

6.2 Engineering.....13

1. SDK Introduction

Delta-2A SDK provides basic development codes for Windows, ROS, Linux, and MCU platforms;

The basic content includes receiving radar data, analyzing radar data, and organizing the analyzed radar data into a complete scan.

The information of a circle is stored in an array for use in the client application. The client can directly

Using the provided SDK source code, you can also write your own SDK receiving and parsing (refer to the SDK source code and radar communication

(communication protocol).

The radar rotates and measures once, scanning and obtaining information about evenly distributed points around it (angle and distance of the points).

The SDK receives the parsed data and obtains the information of each circle point. A circle of 360° is evenly divided into 16

The scanning information is reported in frames (see the command word list in the radar communication protocol document), so each of the 16 frames is obtained.

The frame start angles are 0° (zero point - see the specification for position), 22.5°, 45°, 67.5°, 90°...

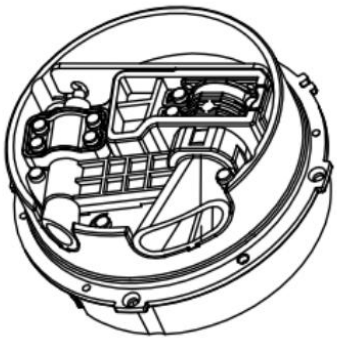
270°, 292.5°, 315°, 337.5°, 360°. 16 frames of data add up to a complete circle.

Points = 16*points per frame; the total number of points per frame can be obtained by calculating the distance number of the scan information frame (distance

The information of each frame data point (angle and distance): The distance of the Nth point in a frame is the number of scan points.

Scan the N distance value in the information frame, the angle corresponding to the distance of the Nth point in that frame = the starting angle of this frame +

(N-1)*22.5/(total number of points per frame), so that all the point information (angle and distance) of one frame are available.



2. Preparation before SDK compilation and debugging

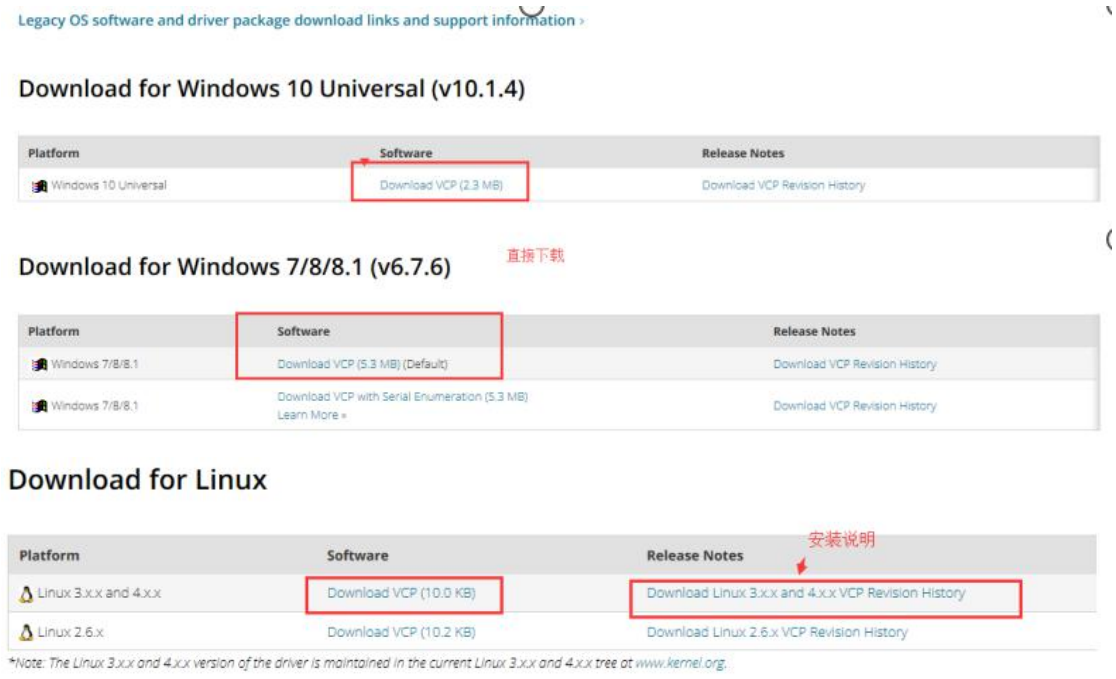
2.1 Connect correctly according to the radar manual

2.2 Install USB to serial port driver

The radar board uses cp2102, you can download the driver from the official website:

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers ; Download the corresponding driver file according to your platform. For Linux platform,

Install according to the documentation on the official website



3. Windows Platform SDK

3.1 Development Environment

Development language: C++

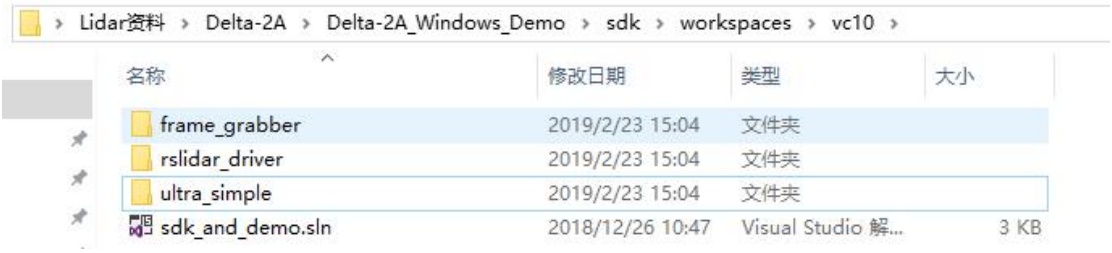
Compilation environment: vs2017

3.2 Windows SDK Solution

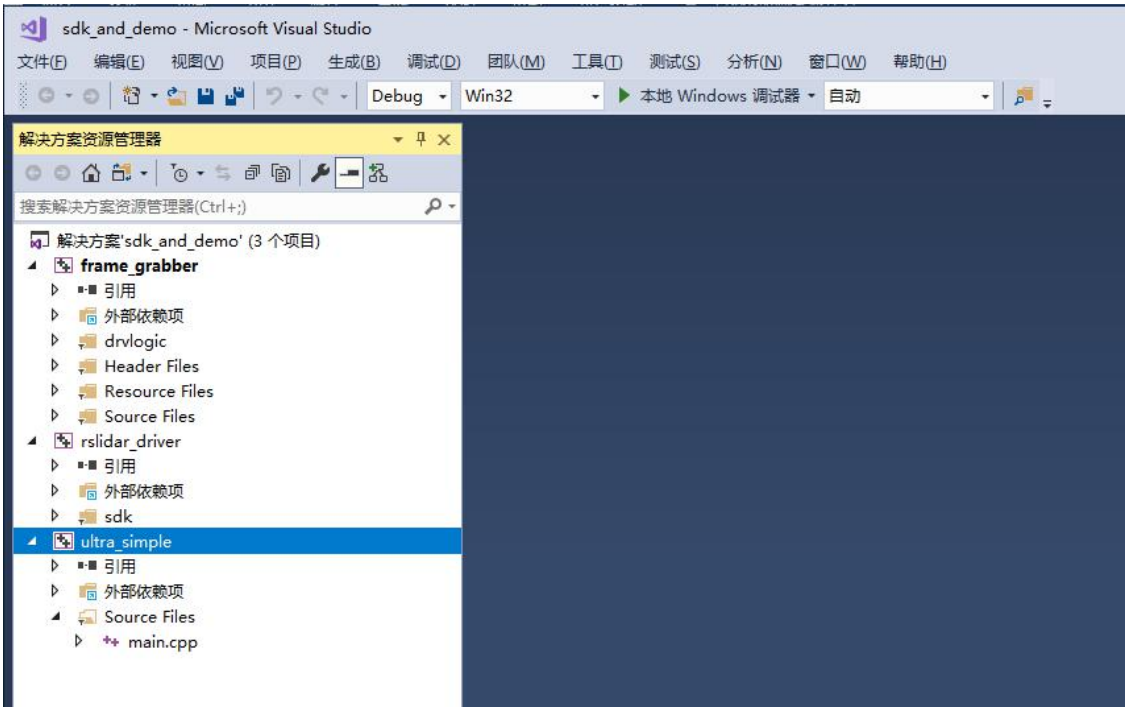
Windows SDK is a solution SDK_and_demo, including: radar driver project rslidar_driver, point cloud project frame_grabber, radar driver simple application project

ultra_simple 3 items.

3.3 Solution Path



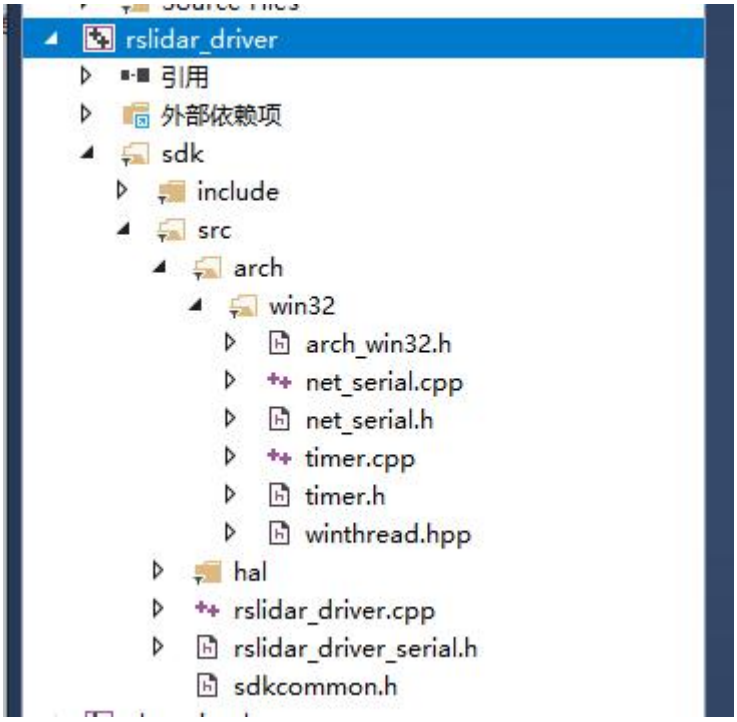
3.4 Solution Engineering Diagram



3.5 Radar driver project rslidar_driver

Provides functions such as opening the serial port, receiving the serial port, parsing the serial port, setting the speed, resetting the radar, starting scanning, and stopping

Interface functions such as stop scanning, read scanning results, etc.



3.5.1 Open the configuration serial port function:

u_result RSlidarDriverSerialImpl::connect(const char*port_path,

_u32baudrate,_u32flag)

parameter:port_path configures the serial port number baudrate: baud rate

Return Value:RESULT_ALREADY_DONE The serial port is already opened

RESULT_INVALID_DATA Open failed

3.5.2 Close serial port function

void RSlidarDriverSerialImpl::disconnect():

3.5.3 Reset radar function

u_result RSlidarDriverSerialImpl::resetlidar(_u32 timeout)

Parameter: timeout timeout for receiving radar response

3.5.4 Get the initial information of a complete circle of scan points (without calculating the angle) function

u_result RSlidarDriverSerialImpl::grabScanData(LIDAR_MEASURE_INFO_T

* nodebuffer, size_t & count, _u32 timeout)

Parameter: nodebuffer stores the initial information of scanning a circle of points count The total number of points in a scan

Timeout Scanning timeout

Return value: RESULT_OK means a complete scan is completed successfully

RESULT_OPERATION_FAIL indicates that the scan failed

RESULT_OPERATION_TIMEOUT indicates that the scan timed out

3.5.5 Convert the initial information of a circle of points into complete point information (distance, angle)

u_result

RSlidarDriverSerialImpl::ascendScanData(LIDAR_MEASURE_INFO_T *

nodebuffer, size_t count)

Parameters: nodebuffer stores the complete information of a circle of points scanned count the total number of points scanned in a circle

Timeout Scanning timeout

RESULT_OK Open successfully

3.6 Build point cloud project frame_grabber

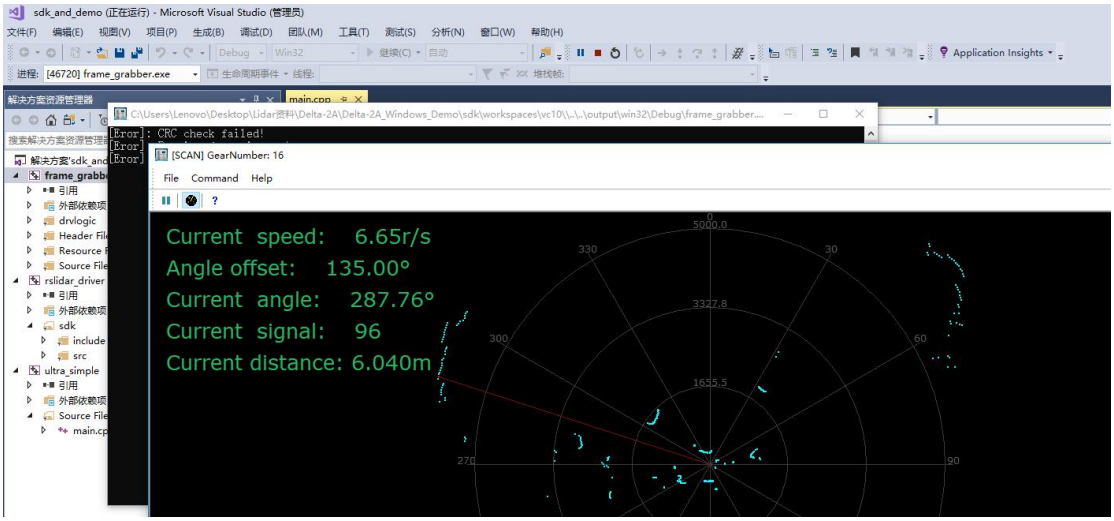
By creating a point cloud map, you can get the current speed, angle and distance of each point

3.6.1 Development environment: C++

3.6.2 Interfaces created by MFC

3.6.3 This project instance relies on the interface in the rslidar_driver project

3.6.4 Compile and debug results



3.7 Radar driver simple application project ultra_simple

You can refer to this project and apply it in actual projects

3.7.1 This project example relies on the interfaces in the rslidar_driver project to help customers call these interfaces.

Use it in your own projects.

3.7.2 Print information: After scanning a circle, print the total number of points in the scan.

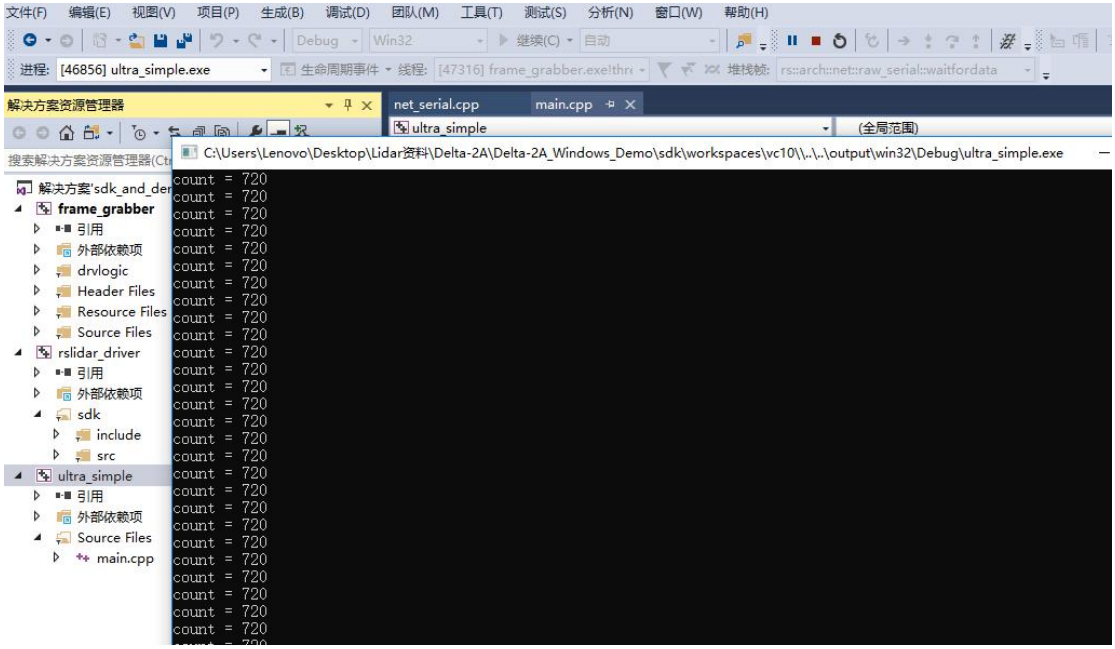
```
op_result = drv->grabScanData(nodes, count);

if (IS_OK(op_result))
{
    drv->ascendScanData(nodes, count);
    //CLogout("++++++++++++++++++++++++++++++++++++:%d \r\n", count);

    printf("count = %d\n", count);
}
else
{
    printf("grabScanData error: %x\n", op_result);
}

/*n++;
```

3.7.3 Compile and debug

3.7.3.1 Modify the main.c to the correct COM port

```
    if (!opt_com_path) {
#ifdef _WIN32
        // use default com port
        opt_com_path = "\\\\.\\com5";
#else
        opt_com_path = "/dev/ttyUSB0";
#endif
    }
```

3.7.3.2 Debugging results



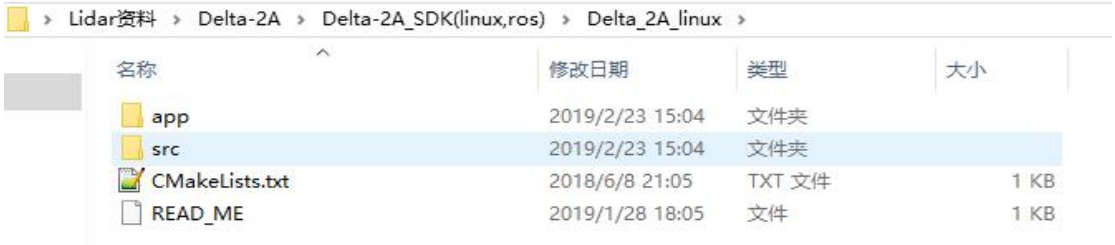
4. Linux Platform SDK

4.1 Development Environment

4.1.1 System: Ubuntu 16.0 、 c++

4.1.2 Compilation tool: Cmake

4.2 Directory Structure



4.2.1 src: stores serial port class, time class, radar interface class (parse, start scan, stop, etc.)

4.2.2 app: Use the radar class interface to implement a call instance

4.2.3 READ_ME: Compilation process

4.2.4 CMakeList.txt: cmake configuration

4.3 Radar class interface functions provided by SDK:

4.3.1 Initialize serial port function

bool C3iroboticsLidar::initilize(CDeviceConnection *device_connect)

Parameter: device_connect serial port class pointer can set the serial port baud rate

4.3.2 Receive and parse serial port data function

TLidarGrabResult C3iroboticsLidar::getScanData()

Return value: LIDAR_GRAB_ING: Scanning

LIDAR_GRAB_SUCCESS: Scanned a circle OK

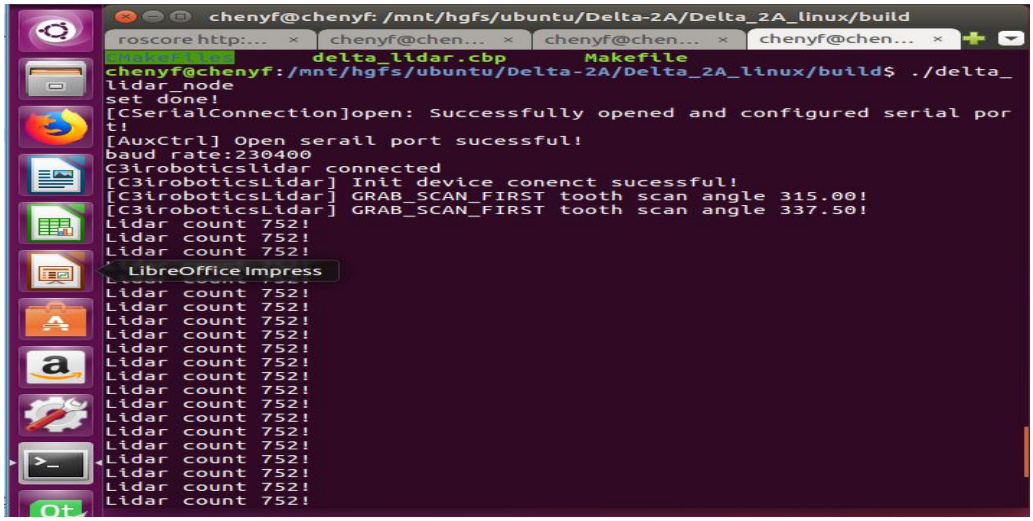
LIDAR_GRAB_ERRO: Scanning error

4.4 Example application: In app/node.c, use the radar interface to implement simple radar operation and print radar

The total number of points per scan can be modified as needed.

4.4.1 Compile: Compile according to the READ_ME prompt

4.4.2 Operation results: Print the total number of points in one scan

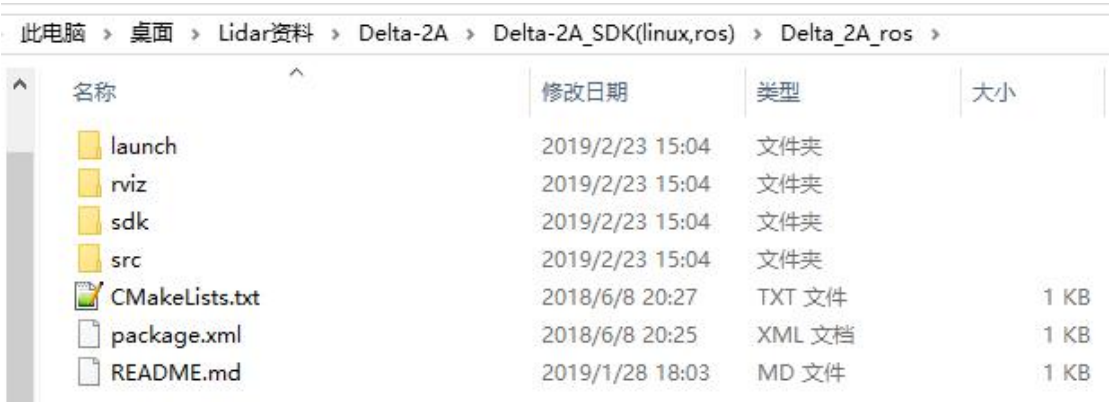


5. Ros Platform SDK

5.1 Development Environment

Ubuntu16.04LTS + kinetic

5.2 Directory Structure



5.2.1 src: node.cpp publisher node code; client.cpp subscriber node code.

5.2.2 sdk: stores serial port class, time class, radar interface class (parse, start scanning, stop, etc.)

mouth).

5.2.3 launch: delta_lidar.launch Launch publisher node configuration

view_delta_lidar.launch launches rviz node configuration.

5.2.4 rviz: Create point cloud configuration.

5.2.5 README.md: Compile process to start the node process.

5.3 Function interface reference: Linux function interface analysis

5.4 Start the publisher and subscriber routines: Through the lidar data structure in the ros system (as shown below)

It can realize simple application of radar data. You can refer to it and apply it to your own solution.

ros中激光雷达的消息类型（sensor_msgs/LaserScan Message）说明

2018年07月11日 10:46:07 ultimate1212 阅读数：1474

最近在做一些视觉和激光数据融合的项目，但是对激光数据的结构不是太了解，因此查了很多相关的内容，记录以下。

下图是在http://wiki.ros.org中截取的图片：

File: sensor_msgs/LaserScan.msg

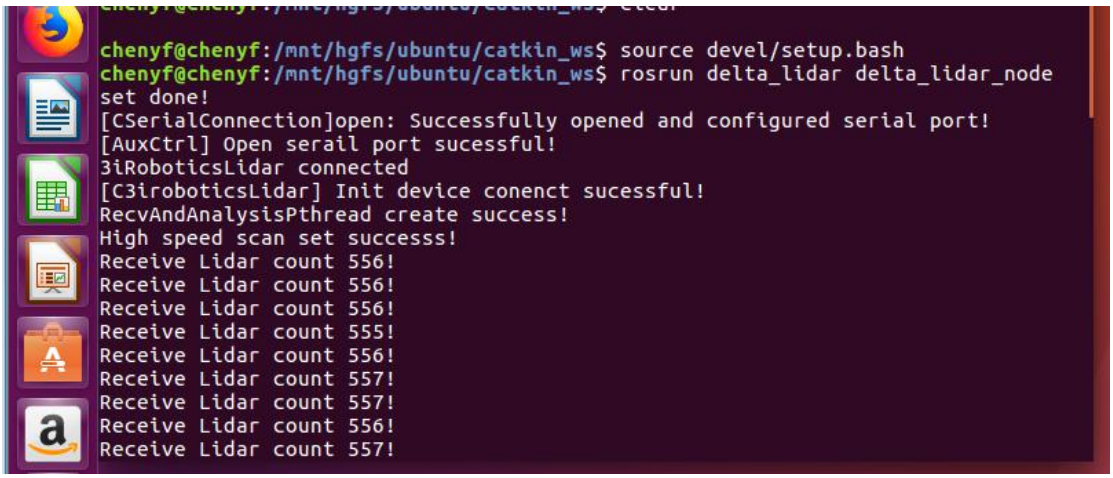
Raw Message Definition

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
```

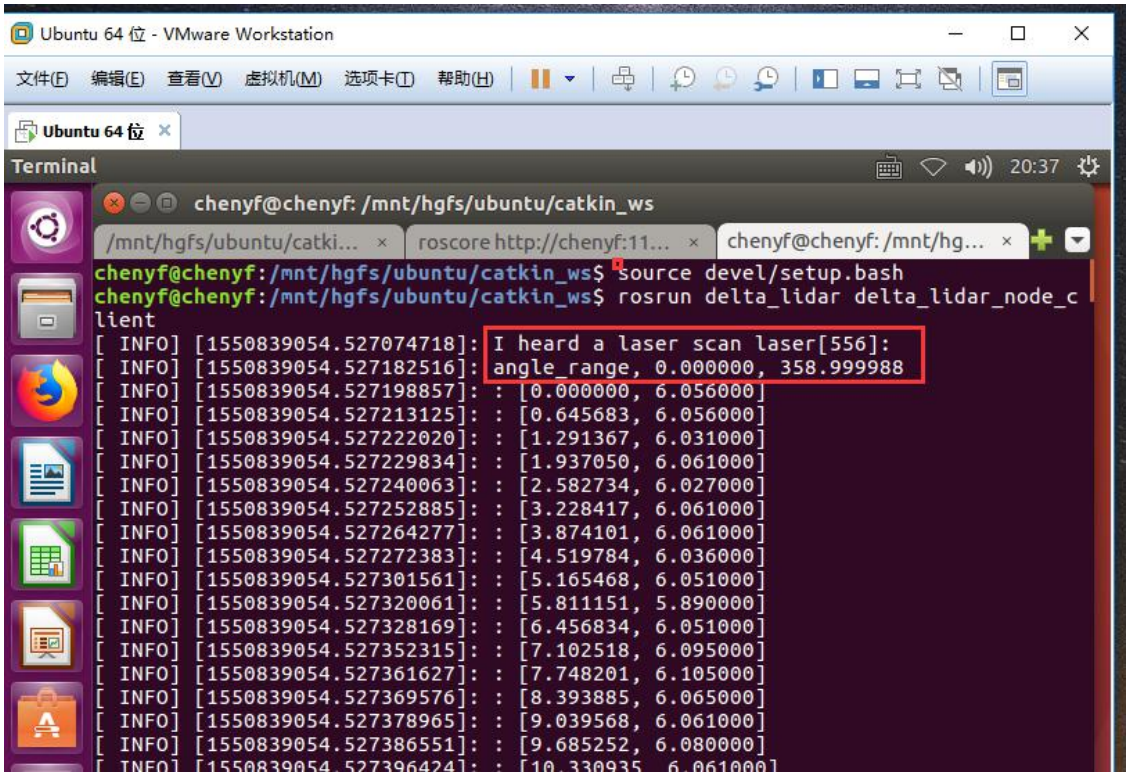
5.4.1 Compilation process reference: README.md:

5.4.2 Start the publisher: A terminal starts, and there will be debugging print information in the publishing node. Scan

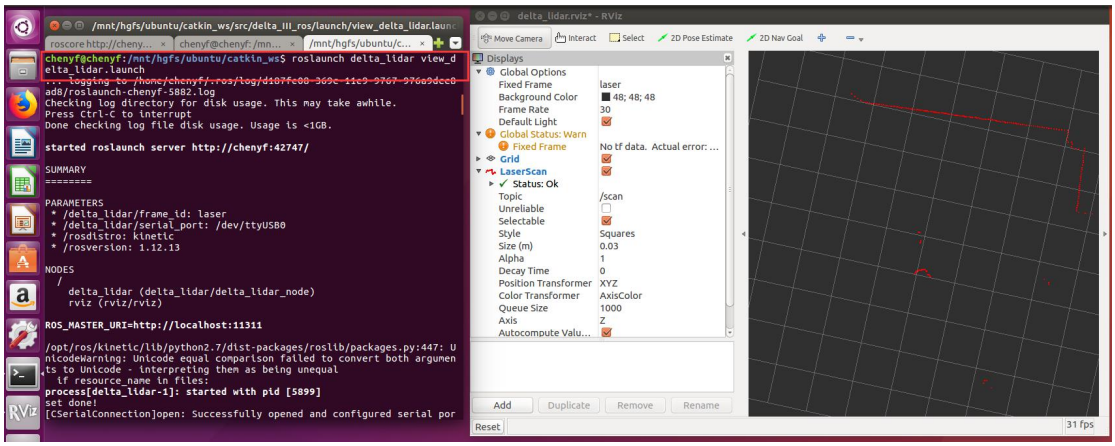
To the number of points in a circle.



5.4.3 Start the subscriber: Start in another terminal. When the subscriber receives a message from the publisher



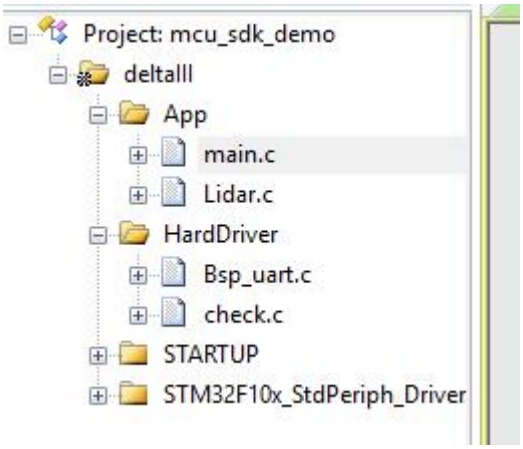
5.5 Start rviz to create a point cloud map



6. Muc Platform DEMO

6.1 Development Environment: Stm32f103cb: Customers can port it to other MCU applications according to their needs.

6.2 Engineering:



Bsp_uart.c: serial port driver, serial port interrupt receiving

Lidar.c : Lidar parsing interface

Main.c: Start the radar, set the speed, use the radar to receive and analyze the scan data, and print the scan data.

The total number of points in the circle.