YDLIDAR SDK

Introduction

YDLIDAR(https://www.ydlidar.com/) series is a set of high-performance and low-cost LIDAR sensors, which is the perfect sensor of 2D SLAM, 3D reconstruction, multi-touch, and safety applications.

If you are using ROS (Robot Operating System), please use our open-source ROS Driver .

Release Notes

Title	Version	Data
SDK	1.4.0	2019-03-25

- [new feature] User can set the maximum number of abnormal checks.

Dataset

Model	Baudrate	Sampling Frequency	Range(m)	Scanning Frequency(HZ)	Working temperature(°C)	Laser power max(mW)	voltage(V)	Current(mA)
S4	115200	4000	0.1-8	6-12	0-40	~5	4.8-5.2	330-380
S4Pro	153600	4000	0.1-8	6-12	0-40	~5	4.8-5.2	330-380

How to build YDLIDAR SDK samples

```
$ git clone https://github.com/ydlidar/sdk
$ cd sdk
$ git checkout S4
$ cd ..
$ mkdir build
$ cd build
$ cmake ../sdk
$ make          ###linux
$ vs open Project.sln   ###windows
```

How to run YDLIDAR SDK samples

```
$ cd samples
```

linux:

```
$ ./ydlidar_test
$Please enter the lidar serial port:/dev/ttyUSB0
```

windows:

```
$ ydlidar_test.exe
$Please enter the lidar serial port:/dev/ttyUSB0
```

You should see YDLIDAR's scan result in the console:

```
[YDLIDAR]:SDK Version: 1.4.0
[YDLIDAR]:Lidar running correctly ! The health status: good
[YDLIDAR] Connection established in [/dev/ttyUSB0][115200]:
Firmware version: 1.2
Hardware version: 3
Model: S4
Serial: 2018101800011111
[YDLIDAR INFO] Now YDLIDAR is scanning ......
Scan received: 625 ranges
Scan received: 626 ranges
```

code:

```
    void ParseScan(node_info* data, const size_t& size) {

        double current_frequence, current_distance, current_angle, current_intensity;

        uint64_t current_time_stamp;

        for (size_t i = 0; i < size; i++ ) {

            if( data[i].scan_frequence != 0) {

                current_frequence =  data[i].scan_frequence;//or current_frequence = data[0].scan_frequence

            }

            current_time_stamp = data[i].stamp;

            current_angle = ((data[i].angle_q6_checkbit>>LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f);//LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT equal
s 8

            current_distance =  data[i].distance_q2/4.f;

            current_intensity = (float)(data[i].sync_quality);

        }

        if (current_frequence != 0 ) {

            printf("current lidar scan frequency: %f\n", current_frequence);

        } else {

            printf("Current lidar does not support return scan frequency\n");

        }
    }
```

Data structure

data structure:

```
//! A struct for returning configuration from the YDLIDAR
struct LaserConfig {

    //! Start angle for the laser scan [rad].  0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.
    float min_angle;

    //! Stop angle for the laser scan [rad].   0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.
    float max_angle;

    //! Scan resolution [rad].
    float ang_increment;

    //! Scan resoltuion [ns]
    float time_increment;

    //! Time between scans
    float scan_time;

    //! Minimum range [m]
    float min_range;

    //! Maximum range [m]
    float max_range;

    //! Range Resolution [m]
    float range_res;

  };


  struct LaserScan {

    //! Array of ranges
    std::vector<float> ranges;

    //! Array of intensities
    std::vector<float> intensities;

    //! Self reported time stamp in nanoseconds
    uint64_t self_time_stamp;

    //! System time when first range was measured in nanoseconds
```

```
    uint64_t system_time_stamp;

    //! Configuration of scan
    LaserConfig config;

  };
```

example angle parsing:

```
LaserScan scan;

for(size_t i =0; i < scan.ranges.size(); i++) {

  // current angle
  double angle = scan.config.min_angle + i*scan.config.ang_increment;// radian format

  //current distance
  double distance = scan.ranges[i];//meters

  //current intensity
  int intensity = scan.intensities[i];

}
```

Coordinate System

The relationship between the angle value and the data structure in the above figure:

```
double current_angle =  scan.config.min_angle + index*scan.config.ang_increment;// radian format
doube Angle = current_angle*180/M_PI;//Angle fomat
```

Upgrade Log

2019-03-25 version:1.4.0

1.fix Large motor resistance at startup issues.

2.fix ascendScanData timestamp issues.

3.check lidar abnormality when turn on lidar.

4.only support S4 lidar

5.Remove other lidar model interfaces functions.

6.fix turnOn function.

2018-12-07 version:1.3.9

1.Remove other lidar model interfaces functions.

2.Remove heartbeat

2018-11-24 version:1.3.8

1.Reduce abnormal situation recovery time.

2.fix timestamp from zero.

2018-10-26 version:1.3.7

1.add input angle calibration file.

2.remove network.

2018-10-15 version:1.3.6

1.add network support.

2018-05-23 version:1.3.4

1.add automatic reconnection if there is an exception

2.add serial file lock.

2018-05-14 version:1.3.3

1.add the heart function constraint.

2.add packet type with scan frequency support.

2018-04-16 version:1.3.2

1.add multithreading support.

2018-04-16 version:1.3.1

1.Compensate for each laser point timestamp.

Contact EAI

If you have any extra questions, please feel free to contact us