
RPLidar Documentation

Release 0.9.1

Artyom Pavlov

May 13, 2017

Contents

Python Module Index

5

Simple and lightweight module for working with RPLidar rangefinder scanners.

Usage example:

```
>>> from rplidar import RPLidar
>>> lidar = RPLidar('/dev/ttyUSB0')
>>>
>>> info = lidar.get_info()
>>> print(info)
>>>
>>> health = lidar.get_health()
>>> print(health)
>>>
>>> for i, scan in enumerate(lidar.iter_scans()):
...     print('%d: Got %d measurments' % (i, len(scan)))
...     if i > 10:
...         break
...
>>> lidar.stop()
>>> lidar.stop_motor()
>>> lidar.disconnect()
```

For additional information please refer to the RPLidar class documentation.

class rplidar.**RPLidar** (*port, baudrate=115200, timeout=1, logger=None*)
Class for communicating with RPLidar rangefinder scanners

Methods

__init__ (*port, baudrate=115200, timeout=1, logger=None*)
Initialize RPLidar object for communicating with the sensor.

Parameters **port** : str

Serial port name to which sensor is connected

baudrate : int, optional

Baudrate for serial connection (the default is 115200)

timeout : float, optional

Serial port connection timeout in seconds (the default is 1)

logger : logging.Logger instance, optional

Logger instance, if none is provided new instance is created

motor = False

Is motor running?

port = ''

Serial port name, e.g. /dev/ttyUSB0

baudrate = 115200

Baudrate for serial port

timeout = 1

Serial port timeout

connect ()

Connects to the serial port with the name *self.port*. If it was connected to another serial port disconnects from it first.

disconnect ()

Disconnects from the serial port

start_motor ()

Starts sensor motor

stop_motor ()

Stops sensor motor

get_info ()

Get device information

Returns dict

Dictionary with the sensor information

get_health ()

Get device health state. When the core system detects some potential risk that may cause hardware failure in the future, the returned status value will be 'Warning'. But sensor can still work as normal. When sensor is in the Protection Stop state, the returned status value will be 'Error'. In case of warning or error statuses non-zero error code will be returned.

Returns status : str

'Good', 'Warning' or 'Error' statuses

error_code : int

The related error code that caused a warning/error.

clear_input ()

Clears input buffer by reading all available data

stop ()

Stops scanning process, disables laser diode and the measurement system, moves sensor to the idle state.

reset ()

Resets sensor core, reverting it to a similar state as it has just been powered up.

iter_measurements (max_buf_meas=500)

Iterate over measurements. Note that consumer must be fast enough, otherwise data will be accumulated inside buffer and consumer will get data with increasing lag.

Parameters max_buf_meas : int

Maximum number of measurements to be stored inside the buffer. Once number exceeds this limit buffer will be emptied out.

Yields new_scan : bool

True if measurement belongs to a new scan

quality : int

Reflected laser pulse strength

angle : float

The measurement heading angle in degree unit [0, 360)

distance : float

Measured object distance related to the sensor's rotation center. In millimeter unit. Set to 0 when measurement is invalid.

iter_scans (*max_buf_meas=500, min_len=5*)

Iterate over scans. Note that consumer must be fast enough, otherwise data will be accumulated inside buffer and consumer will get data with increasing lag.

Parameters **max_buf_meas** : int

Maximum number of measurements to be stored inside the buffer. Once number exceeds this limit buffer will be emptied out.

min_len : int

Minimum number of measurements in the scan for it to be yielded.

Yields scan : list

List of the measurements. Each measurement is tuple with following format: (quality, angle, distance). For values description please refer to *iter_measurements* method's documentation.

exception `rplidar.RPLidarException`

Bases: Exception

Basic exception class for RPLidar

r

rplidar, ??

Symbols

`__init__()` (rplidar.RPLidar method), 1

B

`baudrate` (rplidar.RPLidar attribute), 1

C

`clear_input()` (rplidar.RPLidar method), 2

`connect()` (rplidar.RPLidar method), 1

D

`disconnect()` (rplidar.RPLidar method), 2

G

`get_health()` (rplidar.RPLidar method), 2

`get_info()` (rplidar.RPLidar method), 2

I

`iter_measurments()` (rplidar.RPLidar method), 2

`iter_scans()` (rplidar.RPLidar method), 3

M

`motor` (rplidar.RPLidar attribute), 1

P

`port` (rplidar.RPLidar attribute), 1

R

`reset()` (rplidar.RPLidar method), 2

RPLidar (class in rplidar), 1

rplidar (module), 1

RPLidarException, 3

S

`start_motor()` (rplidar.RPLidar method), 2

`stop()` (rplidar.RPLidar method), 2

`stop_motor()` (rplidar.RPLidar method), 2

T

`timeout` (rplidar.RPLidar attribute), 1