LineLidar class

Generated by Doxygen 1.9.1

Chapter 1

Python3 LineLidar class

1.1 Description

Low-level class to communicate with a LineLidar device in Python

1.2 Notes

Copyright 2023 (c) Noptel Oy, Oulu Finland

Tested on:

- Linux
- Windows

Authors: PCo

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

Module Documentation

6.1 Base classes

Classes

- class ipaddress
- class macaddress

6.1.1 Detailed Description

6.2 Enums

Classes

- class LLcmd
- class LLsrv
- class LLchr
- class LLsta

6.2.1 Detailed Description

6.3 Main classes

Classes

- class LLresponse
- class LineLidar

6.3.1 Detailed Description

Type definitions.

Main classes

6.4 Default parameters

6.5 Routines

Chapter 7

Namespace Documentation

7.1 linelidarclass Namespace Reference

Namespaces

- default
- linelidar

Classes

- class ipaddress
- class macaddress
- class _LLintEnum
- class _LLcmd
- class _LLsrv
- class _LLsta
- class _LLchr
- class LLcmd
- class LLsrv
- class LLchr
- class LLsta
- class LLresponse

Variables

- list __all__ = ["default", "linelidar"]

7.2 linelidarclass.default Namespace Reference

Variables

- int _ll_default_port = 9907
- float _ll_default_udp_timeout = 1.5
- int _ll_default_ssh_timeout = 5
- int _ll_default_reboot_timeout = 5
- int _ll_default_retries = 1
- int _ll_default_sent_cmds_log_depth = 16
- string _ll_default_ssh_python_path = "python"

7.3 linelidarclass.linelidar Namespace Reference

Classes

- class LineLidar

Functions

- List[str] discover (Optional[str] network=None, Optional[int] port=_ll_default_port, Optional[str] sshcmd=None, Optional[str] sshpypath=None, int retries=None, bool debug=False, Optional[float] timeout=None)

7.3.1 Function Documentation

7.3.1.1 discover()

```
List[str] linelidarclass.linelidar.discover (
             Optional[str]  network = None,
             Optional[int]  port = _ll_default_port,
             Optional[str]  sshcmd = None,
             Optional[str]  sshpypath = None,
             int  retries = None,
             bool  debug = False,
             Optional[float]  timeout = None )
```

Discover LineLidar devices on a network.

Parameters

network	Network / netmask to discover devices on, in xx.xx.xx.xx/mm format, or None to discover on all interfaces
port	Port of devices to discover
sshcmd	If specified, ssh command to log into a shell account to use the host as a relay to talk to the LineLidar. Works with a Linux or Windows host with sshd and Python 2 or 3 installed.
sshpypath	Path of the Python executable on the SSH relay host
retries	How many more times the discovery packet should be sent for redundancy
debug	Enable or disable debugging messages
timeout	Communication timeout in seconds

Returns

Broadcast a read request for the NETWORK characteristic, wait for replies and return the list of addresses of the devices that replied.

Definition at line 2089 of file linelidar.py.

Chapter 8

Class Documentation

8.1 _LLchr Class Reference

Public Member Functions

- None __init__ (_LLchr self, str name, Tuple[_LLsrv, int] value)
- bool __eq__ (_LLchr self, object other)
- None __setattr__ (_LLchr self, str attr, val)
- int __hash__ (_LLchr self)

Private Attributes

- **_hash**

8.1.1 Detailed Description

```
Pseudo-enum element to store a (_LLsrv, int) tuple describing a
characteristic

@ingroup BaseClasses
```

Definition at line 229 of file __init__.py.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 __init__()

```
None __init__ (
            _LLchr self,
            str name,
            Tuple[_LLsrv, int] value )
```

```
Initialize the enum element
```

Definition at line 244 of file __init__.py.

8.1.3 Member Function Documentation

8.1.3.1 __eq__()

```
bool __eq__ (
            _LLchr self,
            object other )
```

```
___eq__ method to compare two enum elements
```

Definition at line 257 of file __init__.py.

8.1.3.2 __hash__()

```
int __hash__ (
            _LLchr self )
```

```
__hash__ method
```

Definition at line 278 of file __init__.py.

8.1.3.3 __setattr__()

```
None __setattr__ (
            _LLchr self,
            str attr,
             val )
```

```
Disabled setter, as enums are immutable
```

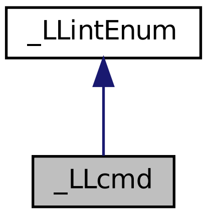
Definition at line 267 of file __init__.py.

The documentation for this class was generated from the following file:

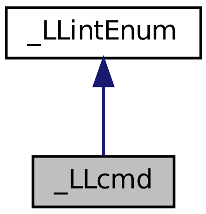
- linelidarclass/__init__.py

8.2 _LLcmd Class Reference

Inheritance diagram for _LLcmd:



Collaboration diagram for _LLcmd:



Additional Inherited Members

8.2.1 Detailed Description

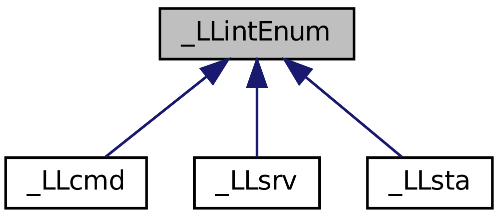
Definition at line 208 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.3 _LLintEnum Class Reference

Inheritance diagram for _LLintEnum:



Public Member Functions

- None __init__ (_LLintEnum self, str name, int value)
- bool __eq__ (_LLintEnum self, object other)
- None __setattr__ (_LLintEnum self, str attr, val)
- int __hash__ (_LLintEnum self)

Public Attributes

- **value**

8.3.1 Detailed Description

```
Pseudo-enum element to store an integer value

@ingroup BaseClasses
```

Definition at line 153 of file __init__.py.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 __init__()

```
None __init__ (
            _LLintEnum self,
            str name,
            int value )
```

```
Initialize the enum element
```

Definition at line 166 of file __init__.py.

8.3.3 Member Function Documentation

8.3.3.1 __eq__()

```
bool __eq__ (
             _LLintEnum self,
             object other )
```

```
___eq__ method to compare two enum elements
```

Definition at line 178 of file __init__.py.

8.3.3.2 __hash__()

```
int __hash__ (
             _LLintEnum self )
```

```
__hash__ method
```

Definition at line 199 of file __init__.py.

8.3.3.3 __setattr__()

```
None __setattr__ (
             _LLintEnum self,
             str attr,
              val )
```

```
Disabled setter, as enums are immutable
```

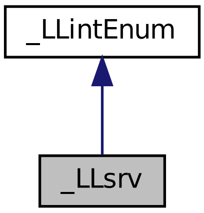
Definition at line 188 of file __init__.py.

The documentation for this class was generated from the following file:

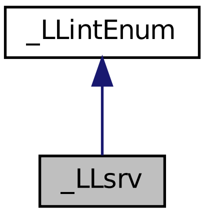
- linelidarclass/__init__.py

8.4 _LLsrv Class Reference

Inheritance diagram for _LLsrv:



Collaboration diagram for _LLsrv:



Additional Inherited Members

8.4.1 Detailed Description

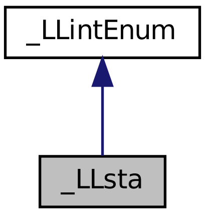
Definition at line 215 of file __init__.py.

The documentation for this class was generated from the following file:

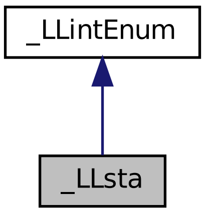
- linelidarclass/__init__.py

8.5 _LLsta Class Reference

Inheritance diagram for _LLsta:



Collaboration diagram for _LLsta:



Additional Inherited Members

8.5.1 Detailed Description

Definition at line 222 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.6 ipaddress Class Reference

Public Member Functions

- None __init__ (ipaddress self, Union[str, bytes] addr)
- bool __eq__ (ipaddress self, object other)
- str __repr__ (ipaddress self)

Public Attributes

- ip

8.6.1 Detailed Description

Definition at line 42 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.7 LineLidar Class Reference

Public Member Functions

- None __init__ (LineLidar self, Optional[str] addr=None, Optional[int] port=None, Optional[str] sshcmd=None, Optional[str] sshpypath=None, Optional[int] sent_cmds_log_depth=None, int retries=None, bool set_clean_state=True, bool autostop_ranging=True, bool debug=False, Optional[float] timeout=None)
- LineLidar __enter__ (LineLidar self)
- None __exit__ (LineLidar self, Optional[Type[BaseException]] exc_type, Optional[BaseException] exc_value, Optional[TracebackType] exc_traceback)
- None __del__ (LineLidar self)
- Union[socket.socket, Tuple[subprocess.Popen, Optional[multiprocessing.queues.Queue]]] open (LineLidar self, str addr, Optional[int] port=None, Optional[str] sshcmd=None, Optional[str] sshpypath=None, Optional[int] sent_cmds_log_depth=None, int retries=None, bool set_clean_state=True, bool autostop↩ _ranging=True, Optional[float] timeout=None)
- None close (LineLidar self)
- LLresponse read_chr (LineLidar self, _LLchr char, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- LLresponse write_chr (LineLidar self, _LLchr char, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None, ∗∗_CHR_ARGS_TYPE kwargs)
- LLresponse set_notification (LineLidar self, _LLchr char, bool enabled, int retries=None, bool exc_on_↩ nok=True, Optional[float] timeout=None)
- LLresponse enable_notification (LineLidar self, _LLchr char, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- LLresponse disable_notification (LineLidar self, _LLchr char, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- LLresponse disable_all_notifications (LineLidar self, bool incl_restricted=False, int retries=None, bool exc_↩ on_nok=True, Optional[float] timeout=None)
- LLresponse report_zero_results (LineLidar self, bool on, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- LLresponse set_sampling_rate (LineLidar self, int frequency, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- LLresponse stop_sampling (LineLidar self, int retries=None, bool exc_on_nok=True, Optional[float] time-out=None)
- LLresponse save_srv (LineLidar self, _LLsrv srv, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- LLresponse restore_srv (LineLidar self, _LLsrv srv, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)
- None flush_notifications (LineLidar self)
- LLresponse get_notification (LineLidar self, Union[List[_LLchr], Tuple[_LLchr], None] chrmask=None, Optional[float] timeout=None)
- None wait_device_quiet (LineLidar self, Optional[float] timeout=None)
- None set_clean_state (LineLidar self, bool incl_restricted=False, int retries=None, Optional[float] time-out=None)
- None reset (LineLidar self, bool reconnect=True, Optional[float] reconnect_timeout=None, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None)

Public Attributes

- retries
- notifications

Static Public Attributes

- int

Private Member Functions

- str __highlighted_bytes (LineLidar self, bytes b, Optional[int] start=None, Optional[int] end=None)
- None _send_data (LineLidar self, Union[bytes, List[bytes]] data)
- Tuple[bytes, Optional[str]] __encode_cmd (LineLidar self, int msgid, _LLcmd cmd, Union[_LLsrv, _LLchr] chr_or_srv, ∗bool args, ∗∗_CHR_ARGS_TYPE kwargs)
- int _send_cmd (LineLidar self, _LLcmd cmd, Union[_LLsrv, _LLchr] chr_or_srv, ∗bool args, ∗∗_CHR_↩ ARGS_TYPE kwargs)
- None __windows_reader_thread (LineLidar self, IO fd, multiprocessing.queues.Queue queue)
- str __recv_ssh_line (LineLidar self, Optional[float] timeout=None)
- LLresponse __decode_msg (LineLidar self, bytes msg)
- LLresponse _recv_msg (LineLidar self, Optional[float] timeout=None)
- LLresponse _get_cmd_response (LineLidar self, bool exc_on_cmd=True, bool exc_on_msgid=True, bool exc_on_chr=True, bool exc_on_nok=True, bool ignore_sent_cmds_log=False, Optional[float] timeout=None)
- LLresponse __retry_cmd (LineLidar self, _LLcmd cmd, Union[_LLsrv, _LLchr] chr_or_srv, int retries=None, bool exc_on_nok=True, Optional[float] timeout=None, ∗bool args, ∗∗_CHR_ARGS_TYPE kwargs)

Private Attributes

- _debug
- __default_timeout
- __timeout
- __autostop_ranging
- __is_ranging
- __ssh_recvbuf
- **_netmask**
- __addrport
- _sent_cmds_log_depth
- __sent_cmds_log
- __msgid

Static Private Attributes

- __socket = None
- __sshproc = None
- __sshproc_win_stdout = None
- __win_reader_thread = None

8.7.1 Detailed Description

Definition at line 183 of file linelidar.py.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 __init__()

```
None __init__ (
            LineLidar self,
            Optional[str]  addr = None,
            Optional[int]  port = None,
            Optional[str]  sshcmd = None,
            Optional[str]  sshpypath = None,
            Optional[int]  sent_cmds_log_depth = None,
            int  retries = None,
            bool  set_clean_state = True,
            bool  autostop_ranging = True,
            bool  debug = False,
            Optional[float]  timeout = None )
```

SSH receive buffer.

Default communication timeout in seconds

Current communication timeout in seconds

Whether to automatically stop ranging upon closing the device

Whether the device is currently ranging

last sent message ID

Sent commands log depth

Sent commands log

How many times a failed command should be retried for fault-tolerance

Notifications stack

Debugging messages toggle

Enum-by-value mappings

__init__ method

Parameters

addr	IP address of the device. If unspecified, the device is not automatically opened.
port	Port of the device (if the device is automatically opened)
sshcmd	If specified, ssh command to log into a shell account to use the host as a relay to talk to the LineLidar(if the device is automatically opened). Works with a Linux or Windows host with sshd and Python 2 or 3 installed.
sshpypath	Path of the Python executable on the SSH relay host (if the device is automatically opened)
sent_cmds_log_depth	Number of sent commands tracked, to ignore out-of-order responses that have already generated a timeout (if the device is automatically opened). Set to 1 to disable filtering out out-of-order UDP packets.
retries	How many times a failed command should be retried (if the device is automatically opened)
set_clean_state	Set device in a known, stopped state after opening (if the device is automatically

Definition at line 266 of file linelidar.py.

8.7.3 Member Function Documentation

8.7.3.1 __decode_msg()

```
LLresponse __decode_msg (
             LineLidar self,
             bytes msg )  [private]
```

Decode a message from the LineLidar.

Parameters

msg	Packet to decode

Returns

Received response

Definition at line 1152 of file linelidar.py.

8.7.3.2 __encode_cmd()

```
Tuple[bytes, Optional[str]] __encode_cmd (
             LineLidar self,
             int msgid,
             _LLcmd cmd,
             Union[_LLsrv, _LLchr] chr_or_srv,
             *bool args,
             **_CHR_ARGS_TYPE kwargs )  [private]
```

Encode a LineLidar command.

Parameters

msgid	Message ID
cmd	Command to encode
chr_or_srv	Characteristic or service
args	Positional arguments (here, only True or False allowed - see below)
kwargs	Keyworded arguments (see below)

If the command is SAVE_SERVICE or RESTORE_SERVICE, pass a service (_LLsrv) in chr_or_srv. Otherwise pass a characteristic (_LLchr)

If the command is SET_NOTIFICATION, pass a single argument True or False to enable or disable notification - e.g. _encode_cmd(LLcmd.SET_NOTIFICATION, LLchr.TEMPERATURE, True)

If the command is WRITE, pass the relevant parameters as keyworded arguments - e.g. _encode_↩ cmd(LLcmd.WRITE, LLchr.MIN_DISTANCE, distance = 3.5)

Returns

(Encoded command, optional debug message)

Definition at line 687 of file linelidar.py.

8.7.3.3 __highlighted_bytes()

```
str __highlighted_bytes (
            LineLidar self,
            bytes b,
            Optional[int]  start = None,
            Optional[int]  end = None )  [private]
```

Generate a printable byte sequence in hex with a slice of it highlighted.

Parameters

b	Byte sequence
start	Start of the byte sequence slice to highlight
end	End of the byte sequence slice to highlight

Returns

Printable hex byte sequence

Definition at line 354 of file linelidar.py.

8.7.3.4 __recv_ssh_line()

```
str __recv_ssh_line (
            LineLidar self,
            Optional[float]  timeout = None )  [private]
```

Get a line of text from the SSH client.

Parameters

timeout	Communication timeout in seconds

Returns

Received text line

Definition at line 1061 of file linelidar.py.

8.7.3.5 __retry_cmd()

```
LLresponse __retry_cmd (
             LineLidar self,
             _LLcmd cmd,
             Union[_LLsrv, _LLchr] chr_or_srv,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None,
             *bool args,
             **_CHR_ARGS_TYPE kwargs )  [private]
```

Try to send a command to the device and get a response from it.

If the response times out, retry sending the command and getting a new response until the number of allowed retries runs out.

Parameters

cmd	Command to send
chr_or_srv	Characteristic or service
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK
timeout	Communication timeout in seconds
args	Positional arguments for _send_cmd
kwargs	Keyworded arguments for _send_cmd

Returns

Response to the command

Definition at line 1591 of file linelidar.py.

8.7.3.6 __windows_reader_thread()

```
None __windows_reader_thread (
             LineLidar self,
             IO fd,
             multiprocessing.queues.Queue queue )  [private]
```

Windows-only blocking file reader thread.

This thread reads data from a blocking file and transfer the data to a queue, which in turn can be read with a timeout.

This workaround is required because Windows doesn't provide select() on file descriptors.

Definition at line 1014 of file linelidar.py.

8.7.3.7 _get_cmd_response()

```
LLresponse _get_cmd_response (
            LineLidar self,
            bool  exc_on_cmd = True,
            bool  exc_on_msgid = True,
            bool  exc_on_chr = True,
            bool  exc_on_nok = True,
            bool  ignore_sent_cmds_log = False,
            Optional[float]  timeout = None )  [private]
```

Get a response to the latest command (i.e.

not a notification).

Parameters

exc_on_cmd	Raise an exception if the response's command doesn't match that of the last sent command
exc_on_msgid	Raise an exception if the response's message ID doesn't match that of the last sent command
exc_on_chr	Raise an exception if the response's characteristic doesn't match that of the last sent command in responses to READ commands
exc_on_nok	Raise an exception on response not OK (see below)
ignore_sent_cmds_log	If asserted, incoming messages will not be checked against older entries in the sent commands log to catch out-of-order responses, and the last sent command will not be removed from the log
timeout	Communication timeout in seconds

if ignore_sent_cmds_log isn't asserted, if a response to a command is received and its message ID, command and service or characteristic match one of the older entries in the sent commands log, the response is considered an out-of-order message, the corresponding entry is removed from the sent commands log and the response is silently discarded.

After discarding out-of-order messages, if either the message's command, message ID or characteristic (in the case of response to a READ command) don't match the last sent command's and either exc_on_cmd, exc_on_msgid or exc_on_chr are asserted (default), an exception is raised.

if ignore_sent_cmds_log isn't asserted and the response matches the last sent command, it is removed from the log.

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

If a NOTIFICATION is received, it is pushed into the notifications stack and the function keeps on waiting for a response to a command.

Returns

Received response

Definition at line 1425 of file linelidar.py.

8.7.3.8 _recv_msg()

```
LLresponse _recv_msg (
            LineLidar self,
            Optional[float]  timeout = None )  [private]
```

Get a message from the LineLidar.

Parameters

timeout	Communication timeout in seconds

Returns

Received response

Definition at line 1312 of file linelidar.py.

8.7.3.9 _send_cmd()

```
int _send_cmd (
            LineLidar self,
            _LLcmd cmd,
            Union[_LLsrv, _LLchr] chr_or_srv,
            *bool args,
            **_CHR_ARGS_TYPE kwargs )  [private]
```

Send a command to the device.

Parameters

cmd	Command to send
chr_or_srv	Characteristic or service
args	Positional arguments (here, only True or False allowed - see below)
kwargs	Keyworded arguments (see below)

If the command is SAVE_SERVICE or RESTORE_SERVICE, pass a service (_LLsrv) in chr_or_srv. Otherwise pass a characteristic (_LLchr)

If the command is SET_NOTIFICATION, pass a single argument True or False to enable or disable notification - e.g. _send_cmd(LLcmd.SET_NOTIFICATION, LLchr.TEMPERATURE, True)

If the command is WRITE, pass the relevant parameters as keyworded arguments - e.g. _send_cmd(LLcmd.WRITE, LLchr.MIN_DISTANCE, distance = 3.5)

Returns

Sent message ID

Definition at line 931 of file linelidar.py.

8.7.3.10 _send_data()

```
None _send_data (
            LineLidar self,
            Union[bytes, List[bytes]] data )  [private]
```

Send data to the LineLidar.

Parameters

data	Data to send in the form of a byte array or a list of byte arrays

If the data is sent to the LineLidar directly, the data in the form of a byte array is sent.

If the data is sent to the LineLidar through a SSH relay, if the data is in the form of a byte array, it is first encapsulated in a list. Then the list of byte array(s) is pickled, base-64 encoded and sent to the SSH relay stub.

Definition at line 646 of file linelidar.py.

8.7.3.11 disable_all_notifications()

```
LLresponse disable_all_notifications (
             LineLidar self,
             bool  incl_restricted = False,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None )
```

Disable notification on all characteristics that support it.

Parameters

incl_restricted	Also disable notification on characteristics where notifications are restricted
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

Definition at line 1769 of file linelidar.py.

8.7.3.12 disable_notification()

```
LLresponse disable_notification (
             LineLidar self,
             _LLchr char,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None )
```

Disable notification on a characteristic.

Parameters

char	Characteristic to disable notification on
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

SET_NOTIFICATION command response

Definition at line 1743 of file linelidar.py.

8.7.3.13 enable_notification()

```
LLresponse enable_notification (
            LineLidar self,
            _LLchr char,
            int  retries = None,
            bool  exc_on_nok = True,
            Optional[float]  timeout = None )
```

Enable notification on a characteristic.

Parameters

char	Characteristic to enable notification on
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

SET_NOTIFICATION command response

Definition at line 1718 of file linelidar.py.

8.7.3.14 get_notification()

```
LLresponse get_notification (
            LineLidar self,
            Union[List[_LLchr], Tuple[_LLchr], None]  chrmask = None,
            Optional[float]  timeout = None )
```

Get a notification, either from the notifications stack, or receive it from the device if the stack is empty.

Parameters

chrmask	List of expected notification characteristics (see below)
timeout	Communication timeout in seconds

If a message other than a notification is received from the device (i.e. a response to a command), an exception is raised.

If a list of characteristics is specified in chrmask and the notification's characteristic isn't in that list, an exception is raised.

Returns

Received notification.

Definition at line 1933 of file linelidar.py.

8.7.3.15 open()

```
Union[socket.socket,Tuple[subprocess.Popen, Optional[ multiprocessing.queues.Queue]]] open (
             LineLidar self,
             str addr,
             Optional[int]  port = None,
             Optional[str]  sshcmd = None,
             Optional[str]  sshpypath = None,
             Optional[int]  sent_cmds_log_depth = None,
             int  retries = None,
             bool  set_clean_state = True,
             bool  autostop_ranging = True,
             Optional[float]  timeout = None )
```

Open communication with a LineLidar.

Parameters

addr	IP address of the device
port	Port of the device
sshcmd	If specified, ssh command to log into a shell account to use the host as a relay to talk to the LineLidar. Works with a Linux or Windows host with sshd and Python 2 or 3 installed.
sshpypath	Path of the Python executable on the SSH relay host
sent_cmds_log_depth	Number of sent commands tracked, to ignore out-of-order responses that have already generated a timeout. Set to 1 to disable filtering out out-of-order UDP packets.
retries	How many times a failed command should be retried
set_clean_state	Set device in a known, stopped state after opening
autostop_ranging	Enable or disable automatically stopping ranging upon closing the device
timeout	Communication timeout in seconds

Returns

Open UDP socket if the communication with the device is direct, or the tuple (open SSH client process, Windows-only stdout queue) if the communication with the device goes through an SSH relay host. On Linux machines, the Windows-only stdout queue is always None. On Windows machine, it is a multiprocessing.↩ Queue() object from which the SSH process' stdout can be read in non-blocking mode instead of the process' stdout handle, which is always blocking.

Definition at line 379 of file linelidar.py.

8.7.3.16 read_chr()

```
LLresponse read_chr (
            LineLidar self,
            _LLchr char,
            int  retries = None,
            bool  exc_on_nok = True,
            Optional[float]  timeout = None )
```

Read a characteristic.

Parameters

char	Characteristic to read
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

READ command response.

Definition at line 1640 of file linelidar.py.

8.7.3.17 report_zero_results()

```
LLresponse report_zero_results (
            LineLidar self,
            bool on,
            int  retries = None,
            bool  exc_on_nok = True,
            Optional[float]  timeout = None )
```

Enable or disable zero results reporting.

Parameters

on	Whether to report zero results
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

WRITE command esponse

Definition at line 1806 of file linelidar.py.

8.7.3.18 reset()

```
None reset (
            LineLidar self,
            bool  reconnect = True,
            Optional[float]  reconnect_timeout = None,
            int  retries = None,
            bool  exc_on_nok = True,
            Optional[float]  timeout = None )
```

Soft-reset the device.

Parameters

reconnect	Attempt to reconnect the device after reset
reconnect_timeout	Timeout in seconds when attempting to reconnect
retries	How many times a failed command should be retried. Only applies if reconnect is False.
exc_on_nok	Raise an exception on response not OK (see below). Only applies if reconnect is False.
timeout	Communication timeout in seconds. Only applies if reconnect is False.

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Definition at line 2027 of file linelidar.py.

8.7.3.19 restore_srv()

```
LLresponse restore_srv (
            LineLidar self,
            _LLsrv srv,
            int  retries = None,
            bool  exc_on_nok = True,
            Optional[float]  timeout = None )
```

Restore a service from non-volatile memory.

Parameters

srv	Service to restore from non-volatile memory
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

RESTORE_SERVICE command response

Definition at line 1900 of file linelidar.py.

8.7.3.20 save_srv()

```
LLresponse save_srv (
             LineLidar self,
             _LLsrv srv,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None )
```

Save a service to non-volatile memory.

Parameters

srv	Service to save to non-volatile memory
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

SAVE_SERVICE command response

Definition at line 1876 of file linelidar.py.

8.7.3.21 set_clean_state()

```
None set_clean_state (
             LineLidar self,
             bool  incl_restricted = False,
             int  retries = None,
             Optional[float]  timeout = None )
```

Stop any active ranging, disable all notifications and flush the notifications stack, so that the device is left in a known, stopped state.

Parameters

incl_restricted	Also disable notification on characteristics where notifications are restricted
retries	How many times a failed command should be retried
timeout	Communication timeout in seconds

Definition at line 2000 of file linelidar.py.

8.7.3.22 set_notification()

```
LLresponse set_notification (
             LineLidar self,
             _LLchr char,
             bool enabled,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None )
```

Enable or disable notification on a characteristic.

Parameters

char	Characteristic to enable or disable notification on
enabled	Whether notification is enabled
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

SET_NOTIFICATION command response

Definition at line 1692 of file linelidar.py.

8.7.3.23 set_sampling_rate()

```
LLresponse set_sampling_rate (
             LineLidar self,
             int frequency,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None )
```

Set the sampling rate.

Parameters

frequency	Sampling frequency
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

WRITE command esponse

Definition at line 1830 of file linelidar.py.

8.7.3.24 stop_sampling()

```
LLresponse stop_sampling (
            LineLidar self,
            int  retries = None,
            bool  exc_on_nok = True,
            Optional[float]  timeout = None )
```

Stop sampling.

Parameters

retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

WRITE command esponse

Definition at line 1854 of file linelidar.py.

8.7.3.25 wait_device_quiet()

```
None wait_device_quiet (
            LineLidar self,
            Optional[float]  timeout = None )
```

Wait until the device becomes silent.

Parameters

timeout	Communication timeout in seconds

Definition at line 1980 of file linelidar.py.

8.7.3.26 write_chr()

```
LLresponse write_chr (
             LineLidar self,
             _LLchr char,
             int  retries = None,
             bool  exc_on_nok = True,
             Optional[float]  timeout = None,
             **_CHR_ARGS_TYPE kwargs )
```

Write a characteristic.

Parameters

char	Characteristic to write
retries	How many times a failed command should be retried
exc_on_nok	Raise an exception on response not OK (see below)
timeout	Communication timeout in seconds
kwargs	Keyworded arguments (see below)

Pass the relevant parameters as keyworded arguments - e.g. write_chr(LLchr.MIN_DISTANCE, distance = 3.5)

If exc_on_nok is asserted (default), an exception is raised when the response's status is not OK.

Returns

WRITE command esponse

Definition at line 1663 of file linelidar.py.

8.7.4 Member Data Documentation

8.7.4.1 int

```
int  [static]
```

Address and port of the device or network.

Netmask to filter incoming UDP replies by IP - Default: /32 (single host)

Definition at line 194 of file linelidar.py.

The documentation for this class was generated from the following file:

- linelidarclass/linelidar.py

8.8 LLchr Class Reference

Static Public Attributes

- SERIAL_NUMBER
- FW_VERSION
- NETWORK
- TIME
- TEMPERATURE
- MAC
- DEFAULT_NETWORK
- MIN_DISTANCE
- MAX_DISTANCE
- AMPLITUDE_THRESHOLD
- MIN_ANGLE
- MAX_ANGLE
- TRIGGER_SOURCE
- NB_PEAKS
- REPORT_ZERO_RESULTS
- TRIGGER_TYPE
- TRIGGER_DIVISION
- CALIBRATED_ANGLES
- RANGE
- MEASURING_RATE
- RESET_DEVICE

8.8.1 Detailed Description

8.8.2 Notes

Underscore in a name indicates a private characteristic

Definition at line 356 of file __init__.py.

8.8.3 Member Data Documentation

8.8.3.1 AMPLITUDE_THRESHOLD

```
AMPLITUDE_THRESHOLD  [static]
```

Initial value:

```
=  _LLchr("AMPLITUDE_THRESHOLD",
               (LLsrv.DEVICE_CONFIG, 0x0005))
```

Amplitude threshold.

Definition at line 416 of file __init__.py.

8.8.3.2 CALIBRATED_ANGLES

```
CALIBRATED_ANGLES  [static]
```

Initial value:
```
=  _LLchr("CALIBRATED_ANGLES",
                (LLsrv.HW_CONFIG, 0x0008))
```

Calibrated angles table.

Definition at line 458 of file __init__.py.

8.8.3.3 DEFAULT_NETWORK

```
DEFAULT_NETWORK  [static]
```

Initial value:
```
=  _LLchr("DEFAULT_NETWORK",
                (LLsrv.DEVICE_INFO, 0x0013))
```

Default network.

Definition at line 399 of file __init__.py.

8.8.3.4 FW_VERSION

```
FW_VERSION  [static]
```

Initial value:
```
=  _LLchr("FW_VERSION",
                (LLsrv.DEVICE_INFO, 0x0003))
```

Firmware version.

Definition at line 374 of file __init__.py.

8.8.3.5 MAC

```
MAC  [static]
```

Initial value:
```
=  _LLchr("MAC",
                (LLsrv.DEVICE_INFO, 0x000b))
```

MAC address.

Definition at line 394 of file __init__.py.

8.8.3.6 MAX_ANGLE

```
MAX_ANGLE  [static]
```

Initial value:
```
=  _LLchr("MAX_ANGLE",
                (LLsrv.DEVICE_CONFIG, 0x000b))
```

Maximum angle.

Definition at line 426 of file __init__.py.

8.8.3.7 MAX_DISTANCE

```
MAX_DISTANCE  [static]
```

Initial value:
```
=  _LLchr("MAX_DISTANCE",
                (LLsrv.DEVICE_CONFIG, 0x0003))
```

Maximum ranging distance.

Definition at line 411 of file __init__.py.

8.8.3.8 MEASURING_RATE

```
MEASURING_RATE  [static]
```

Initial value:
```
=  _LLchr("MEASURING_RATE",
                (LLsrv.RESULTS, 0x0008))
```

Measuring rate.

Definition at line 470 of file __init__.py.

8.8.3.9 MIN_ANGLE

```
MIN_ANGLE  [static]
```

Initial value:
```
=  _LLchr("MIN_ANGLE",
                (LLsrv.DEVICE_CONFIG, 0x000a))
```

Minimum angle.

Definition at line 421 of file __init__.py.

8.8.3.10 MIN_DISTANCE

```
MIN_DISTANCE  [static]
```

Initial value:
```
=  _LLchr("MIN_DISTANCE",
                (LLsrv.DEVICE_CONFIG, 0x0002))
```

Minimum ranging distance.

Definition at line 406 of file __init__.py.

8.8.3.11 NB_PEAKS

```
NB_PEAKS  [static]
```

Initial value:
```
=  _LLchr("NB_PEAKS",
                (LLsrv.DEVICE_CONFIG, 0x000d))
```

Number of peaks.

Definition at line 436 of file __init__.py.

8.8.3.12 NETWORK

```
NETWORK  [static]
```

Initial value:
```
=  _LLchr("NETWORK",
                (LLsrv.DEVICE_INFO, 0x0004))
```

NETWORK: network settings.

Definition at line 379 of file __init__.py.

8.8.3.13 RANGE

```
RANGE  [static]
```

Initial value:
```
=  _LLchr("RANGE",
                (LLsrv.RESULTS, 0x0003))
```

Rangefinding results.

Definition at line 465 of file __init__.py.

8.8.3.14 REPORT_ZERO_RESULTS

```
REPORT_ZERO_RESULTS  [static]
```

Initial value:
```
=  _LLchr("REPORT_ZERO_RESULTS",
                (LLsrv.DEVICE_CONFIG, 0x000e))
```

Report zero results toggle.

Definition at line 441 of file __init__.py.

8.8.3.15 RESET_DEVICE

```
RESET_DEVICE  [static]
```

Initial value:
```
=  _LLchr("RESET_DEVICE",
                (LLsrv.DEBUG, 0x0007))
```

Reset device.

Definition at line 477 of file __init__.py.

8.8.3.16 SERIAL_NUMBER

```
SERIAL_NUMBER  [static]
```

Initial value:
```
=  _LLchr("SERIAL_NUMBER",
                (LLsrv.DEVICE_INFO, 0x0002))
```

Serial number.

Definition at line 369 of file __init__.py.

8.8.3.17 TEMPERATURE

```
TEMPERATURE  [static]
```

Initial value:
```
=  _LLchr("TEMPERATURE",
                (LLsrv.DEVICE_INFO, 0x0007))
```

Temperature of the device.

Definition at line 389 of file __init__.py.

8.8.3.18 TIME

```
TIME  [static]
```

Initial value:

```
=  _LLchr("TIME",
                (LLsrv.DEVICE_INFO, 0x0005))
```

TIME: current time.

Definition at line 384 of file __init__.py.

8.8.3.19 TRIGGER_DIVISION

```
TRIGGER_DIVISION  [static]
```

Initial value:

```
=  _LLchr("TRIGGER_DIVISION",
                (LLsrv.DEVICE_CONFIG, 0x0010))
```

Trigger division.

Definition at line 451 of file __init__.py.

8.8.3.20 TRIGGER_SOURCE

```
TRIGGER_SOURCE  [static]
```

Initial value:

```
=  _LLchr("TRIGGER_SOURCE",
                (LLsrv.DEVICE_CONFIG, 0x000c))
```

Trigger source.

Definition at line 431 of file __init__.py.

8.8.3.21 TRIGGER_TYPE

```
TRIGGER_TYPE  [static]
```

Initial value:

```
=  _LLchr("TRIGGER_TYPE",
                (LLsrv.DEVICE_CONFIG, 0x000f))
```

Trigger type.

Definition at line 446 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.9 LLcmd Class Reference

Static Public Attributes

- ERROR = _LLcmd("ERROR", 0x0000)
- READ = _LLcmd("READ", 0x0001)
- WRITE = _LLcmd("WRITE", 0x0002)
- NOTIFICATION = _LLcmd("NOTIFICATION", 0x0005)
- SET_NOTIFICATION = _LLcmd("SET_NOTIFICATION", 0x0006)
- SAVE_SERVICE = _LLcmd("SAVE_SERVICE", 0x0007)
- RESTORE_SERVICE = _LLcmd("RESTORE_SERVICE", 0x0008)

8.9.1 Detailed Description

Definition at line 289 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.10 LLresponse Class Reference

Public Member Functions

- None __init__ (LLresponse self, Optional[_LLcmd] cmd=None, Optional[int] msgid=None, Optional[_LLsta] status=None, Optional[_LLchr] char=None, Optional[datetime] recv_local_timestamp=None)
- str __repr__ (LLresponse self)

Public Attributes

- recv_local_timestamp
- cmd
- msgid
- status
- char
- value
- major
- minor
- bugfix
- addr
- defaultgw
- netmask
- port
- time
- temperature
- mac
- distance
- angle
- source

- type
- div
- frequency
- threshold
- angles
- peaks
- timestamp
- measurementid
- nbresults
- reset

8.10.1 Detailed Description

Definition at line 517 of file __init__.py.

8.10.2 Member Function Documentation

8.10.2.1 __repr__()

```
str __repr__ (
             LLresponse self )
```

Generate a printable description of the response.

Returns

printable response

Definition at line 691 of file __init__.py.

8.10.3 Member Data Documentation

8.10.3.1 reset

```
reset
```

Attribute present in notifications or read command responses for characteristic RANGE.

Attribute present in read command responses for characteristic RESET

Definition at line 687 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.11 LLsrv Class Reference

Static Public Attributes

- DEVICE_INFO = _LLsrv("DEVICE_INFO", 0x0001)
- DEVICE_CONFIG = _LLsrv("DEVICE_CONFIG", 0x0002)
- HW_CONFIG = _LLsrv("HW_CONFIG", 0x0003)
- RESULTS = _LLsrv("RESULTS", 0x0004)
- DEBUG = _LLsrv("DEBUG", 0x0005)

8.11.1 Detailed Description

8.11.2 Notes

Underscore in a name indicates a private service

Definition at line 325 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.12 LLsta Class Reference

Static Public Attributes

- OK = _LLsta("OK", 0x0000)
- CHAR_DECODING_FAILED = _LLsta("CHAR_DECODING_FAILED", 0x0001)
- CHAR_OUT_OF_RANGE = _LLsta("CHAR_OUT_OF_RANGE", 0x0002)
- PROCEDURE_IN_PROGRESS = _LLsta("PROCEDURE_IN_PROGRESS", 0x0003)
- PROCEDURE_FAILED = _LLsta("PROCEDURE_FAILED", 0x0004)
- NOT_ALLOWED = _LLsta("NOT_ALLOWED", 0x0006)

8.12.1 Detailed Description

Definition at line 482 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

8.13 macaddress Class Reference

Public Member Functions

- None __init__ (macaddress self, Union[str, bytes] addr)
- bool __eq__ (macaddress self, object other)
- str __repr__ (macaddress self)

Public Attributes

- mac

8.13.1 Detailed Description

Definition at line 96 of file __init__.py.

The documentation for this class was generated from the following file:

- linelidarclass/__init__.py

Index