AdafruitLIDARLite Library Documentation

Release 1.0

ladyada

Sep 04, 2019

Contents

A CircuitPython & Python library for Garmin LIDAR Lite sensors over I2C

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Usage Example

```
import time
import board
import busio
import adafruit_lidarlite

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)

# Default configuration, with only i2c wires
sensor = adafruit_lidarlite.LIDARLite(i2c)

while True:
    try:
        # We print tuples so you can plot with Mu Plotter
        print((sensor.distance,))
    except RuntimeError as e:
        # If we get a reading error, just print it and keep truckin'
        print(e)
    time.sleep(0.01) # you can remove this for ultra-fast measurements!
```

CHAPTER 3

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 4

Building locally

4.1 Zip release files

To build this library locally you'll need to install the circuitpython-build-tools package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-lidarlite --
↪library_location .
```

4.2 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the index.html in your browser to view them. It will also (due to -W) error out on any warning like Travis will. This is a good way to locally verify it will pass.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/lidarlite_simpletest.py

```
import time
import board
import busio
import adafruit_lidarlite


# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)

# Default configuration, with only i2c wires
sensor = adafruit_lidarlite.LIDARLite(i2c)

# Optionally, we can pass in a hardware reset pin, or custom config
#import digitalio
#reset = digitalio.DigitalInOut(board.D5)
#sensor = adafruit_lidarlite.LIDARLite(i2c, reset_pin=reset,
#    configuration=adafruit_lidarlite.CONFIG_MAXRANGE)

# If you want to reset, you can do so, note that it can take 10-20 seconds
# for the data to 'normalize' after a reset (and this isnt documented at all)
# sensor.reset()

while True:
    try:
        # We print tuples so you can plot with Mu Plotter
        print((sensor.distance,))
    except RuntimeError as e:
```

(continues on next page)

(continued from previous page)

```
28          # If we get a reading error, just print it and keep truckin'
29          print(e)
30      time.sleep(0.01) # you can remove this for ultra-fast measurements!
```

5.2 adafruit_lidarlite

A CircuitPython & Python library for Garmin LIDAR Lite sensors over I2C

- Author(s): ladyada

5.2.1 Implementation Notes

Hardware:

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class adafruit_lidarlite.**LIDARLite**(*i2c_bus*, ***, *reset_pin=None*, *configuration=0*, *address=98*)

A driver for the Garmin LIDAR Lite laser distance sensor. :param i2c_bus: The `busio.I2C` object to use. This is the only required parameter. :param int address: (optional) The I2C address of the device to set after initialization.

configure(*config*)

Set the LIDAR desired style of measurement. There are a few common configurations Garmin suggests: CONFIG_DEFAULT, CONFIG_SHORTFAST, CONFIG_DEFAULTFAST, CONFIG_MAXRANGE, CONFIG_HIGHSENSITIVE, and CONFIG_LOWSENSITIVE.

distance

The measured distance in cm. Will take a bias reading every 100 calls

read_distance(*bias=False*)

Perform a distance reading with or without 'bias'. It's recommended to take a bias measurement every 100 non-bias readings (they're slower)

reset()

Hardware reset (if pin passed into init) or software reset. Will take 100 readings in order to 'flush' measurement unit, otherwise data is off.

status

The status byte, check datasheet for bitmask

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index