Adafruit BME280 Library Documentation

Release 1.0

ladyada

Mar 13, 2019

Contents

1 Installation and Dependencies **3**
1.1 Installing from PyPI 3

2 Usage Example **5**

3 Contributing **7**

4 Building Locally **9**
4.1 Sphinx Documentation 9

5 Table of Contents **11**
5.1 Simple test 11
5.2 `adafruit_bme280` - Adafruit BME280 - Temperature, Humidity & Barometic Pressure Sensor . . 12

6 Indices and tables **13**

Python Module Index **15**

I2C and SPI driver for the Bosch BME280 Temperature, Humidity, and Barometric Pressure sensor

CHAPTER 1

Installation and Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device

Please ensure that the driver and all dependencies are available on the CircuitPython filesystem. This can be most easily achieved by downloading and installing the latest Adafruit library and driver bundle on your device.

1.1 Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-bme280
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-bme280
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-bme280
```

CHAPTER 2

Usage Example

```
import board
import digitalio
import busio
import time
import adafruit_bme280

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# OR create library object using our Bus SPI port
#spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
#bme_cs = digitalio.DigitalInOut(board.D10)
#bme280 = adafruit_bme280.Adafruit_BME280_SPI(spi, bme_cs)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bme280.temperature)
    print("Humidity: %0.1f %%" % bme280.humidity)
    print("Pressure: %0.1f hPa" % bme280.pressure)
    print("Altitude = %0.2f meters" % bme280.altitude)
    time.sleep(2)
```

CHAPTER 3

Contributing

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

CHAPTER 4

Building Locally

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

```
python3 -m venv .env
source .env/bin/activate
pip3 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-veml6070 --
↪library_location .
```

4.1 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
pip3 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/bme280_simpletest.py

```
import time

import board
import busio
import adafruit_bme280

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# OR create library object using our Bus SPI port
#spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
#bme_cs = digitalio.DigitalInOut(board.D10)
#bme280 = adafruit_bme280.Adafruit_BME280_SPI(spi, bme_cs)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bme280.temperature)
    print("Humidity: %0.1f %%" % bme280.humidity)
    print("Pressure: %0.1f hPa" % bme280.pressure)
    print("Altitude = %0.2f meters" % bme280.altitude)
    time.sleep(2)
```

5.2 adafruit_bme280 - Adafruit BME280 - Temperature, Humidity & Barometic Pressure Sensor

CircuitPython driver from BME280 Temperature, Humidity and Barometic Pressure sensor

- Author(s): ladyada

class adafruit_bme280.**Adafruit_BME280**

Driver from BME280 Temperature, Humidity and Barometic Pressure sensor

altitude

The altitude based on current `pressure` versus the sea level pressure (`sea_level_pressure`) - which you must enter ahead of time)

humidity

The relative humidity in RH % returns None if humidity measurement is disabled

iir_filter

Controls the time constant of the IIR filter Allowed values are the constants IIR_FILTER_*

measurement_time_max

Maximum time in milliseconds required to complete a measurement in normal mode

measurement_time_typical

Typical time in milliseconds required to complete a measurement in normal mode

mode

Operation mode Allowed values are the constants MODE_*

overscan_humidity

Humidity Oversampling Allowed values are the constants OVERSCAN_*

overscan_pressure

Pressure Oversampling Allowed values are the constants OVERSCAN_*

overscan_temperature

Temperature Oversampling Allowed values are the constants OVERSCAN_*

pressure

The compensated pressure in hectoPascals. returns None if pressure measurement is disabled

sea_level_pressure = None

Pressure in hectoPascals at sea level. Used to calibrate `altitude`.

standby_period

Control the inactive period when in Normal mode Allowed standby periods are the constants STANDBY_TC_*

temperature

The compensated temperature in degrees celsius.

class adafruit_bme280.**Adafruit_BME280_I2C**(*i2c*, *address=<sphinx.ext.autodoc.importer._MockObject object>*)

Driver for BME280 connected over I2C

class adafruit_bme280.**Adafruit_BME280_SPI**(*spi*, *cs*, *baudrate=100000*)

Driver for BME280 connected over SPI

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

adafruit_bme280, 11

Index

A

Adafruit_BME280 (class in adafruit_bme280), 12
adafruit_bme280 (module), 11
Adafruit_BME280_I2C (class in adafruit_bme280), 12
Adafruit_BME280_SPI (class in adafruit_bme280), 12
altitude (adafruit_bme280.Adafruit_BME280 attribute), 12

H

humidity (adafruit_bme280.Adafruit_BME280 attribute), 12

I

iir_filter (adafruit_bme280.Adafruit_BME280 attribute), 12

M

measurement_time_max (adafruit_bme280.Adafruit_BME280 attribute), 12
measurement_time_typical (adafruit_bme280.Adafruit_BME280 attribute), 12
mode (adafruit_bme280.Adafruit_BME280 attribute), 12

O

overscan_humidity (adafruit_bme280.Adafruit_BME280 attribute), 12
overscan_pressure (adafruit_bme280.Adafruit_BME280 attribute), 12
overscan_temperature (adafruit_bme280.Adafruit_BME280 attribute), 12

P

pressure (adafruit_bme280.Adafruit_BME280 attribute), 12

S

sea_level_pressure (adafruit_bme280.Adafruit_BME280 attribute), 12
standby_period (adafruit_bme280.Adafruit_BME280 attribute), 12

T

temperature (adafruit_bme280.Adafruit_BME280 attribute), 12