Adafruit BMP280 Library Documentation

Release 1.0

ladyada

Jun 07, 2021

Contents

CircuitPython driver from BMP280 Temperature and Barometic Pressure sensor

CHAPTER 1

Installation and 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.

1.1 Installing from PyPI

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

```
pip3 install adafruit-circuitpython-bmp280
```

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

```
sudo pip3 install adafruit-circuitpython-bmp280
```

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-bmp280
```

CHAPTER 2

Usage Example

```
import time
import board
# import digitalio # For use with SPI
import adafruit_bmp280

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()   # uses board.SCL and board.SDA
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR Create sensor object, communicating over the board's default SPI bus
# spi = board.SPI()
# bmp_cs = digitalio.DigitalInOut(board.D10)
# bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

# change this to match the location's pressure (hPa) at sea level
bmp280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bmp280.temperature)
    print("Pressure: %0.1f hPa" % bmp280.pressure)
    print("Altitude = %0.2f meters" % bmp280.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

Documentation

For information on building library documentation, please check out this guide.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/bmp280_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Simpletest Example that shows how to get temperature,
   pressure, and altitude readings from a BMP280"""
import time
import board

# import digitalio # For use with SPI
import adafruit_bmp280

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()  # uses board.SCL and board.SDA
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR Create sensor object, communicating over the board's default SPI bus
# spi = board.SPI()
# bmp_cs = digitalio.DigitalInOut(board.D10)
# bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

# change this to match the location's pressure (hPa) at sea level
bmp280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bmp280.temperature)
    print("Pressure: %0.1f hPa" % bmp280.pressure)
    print("Altitude = %0.2f meters" % bmp280.altitude)
```

(continues on next page)

(continued from previous page)

```
    time.sleep(2)
```

5.2 Normal Mode

Example showing how the BMP280 library can be used to set the various parameters supported by the sensor.

Listing 2: examples/bmp280_normal_mode.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
Example showing how the BMP280 library can be used to set the various
parameters supported by the sensor.
Refer to the BMP280 datasheet to understand what these parameters do
"""
import time
import board
import adafruit_bmp280

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()  # uses board.SCL and board.SDA
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR Create sensor object, communicating over the board's default SPI bus
# spi = busio.SPI()
# bmp_cs = digitalio.DigitalInOut(board.D10)
# bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

# change this to match the location's pressure (hPa) at sea level
bmp280.sea_level_pressure = 1013.25
bmp280.mode = adafruit_bmp280.MODE_NORMAL
bmp280.standby_period = adafruit_bmp280.STANDBY_TC_500
bmp280.iir_filter = adafruit_bmp280.IIR_FILTER_X16
bmp280.overscan_pressure = adafruit_bmp280.OVERSCAN_X16
bmp280.overscan_temperature = adafruit_bmp280.OVERSCAN_X2
# The sensor will need a moment to gather inital readings
time.sleep(1)

while True:
    print("\nTemperature: %0.1f C" % bmp280.temperature)
    print("Pressure: %0.1f hPa" % bmp280.pressure)
    print("Altitude = %0.2f meters" % bmp280.altitude)
    time.sleep(2)
```

5.3 adafruit_bmp280

CircuitPython driver from BMP280 Temperature and Barometric Pressure sensor

- Author(s): ladyada

5.3.1 Implementation Notes

Hardware:

- Adafruit from BMP280 Temperature and Barometric Pressure sensor

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_bmp280.**Adafruit_BMP280**

Base BMP280 object. Use *Adafruit_BMP280_I2C* or *Adafruit_BMP280_SPI* instead of this. This checks the BMP280 was found, reads the coefficients and enables the sensor for continuous reads

---

Note: The operational range of the BMP280 is 300-1100 hPa. Pressure measurements outside this range may not be as accurate.

---

altitude

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

iir_filter

Controls the time constant of the IIR filter Allowed values are set in the IIR_FILTER enum class

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 set in the MODE enum class

overscan_pressure

Pressure Oversampling Allowed values are set in the OVERSCAN enum class

overscan_temperature

Temperature Oversampling Allowed values are set in the OVERSCAN enum class

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 set the STANDBY enum class

temperature

The compensated temperature in degrees Celsius.

class adafruit_bmp280.**Adafruit_BMP280_I2C**(*i2c*, *address=119*)

Driver for I2C connected BMP280.

Parameters

- **i2c** (*I2C*) – The I2C bus the BMP280 is connected to.

- **address** (*int*) – I2C device address. Defaults to `0x77`. but another address can be passed in as an argument

Quickstart: Importing and using the BMP280

Here is an example of using the `BMP280_I2C` class. First you will need to import the libraries to use the sensor

```
import board
import adafruit_bmp280
```

Once this is done you can define your `board.I2C` object and define your sensor object

```
i2c = board.I2C()   # uses board.SCL and board.SDA
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)
```

You need to setup the pressure at sea level

```
bmp280.sea_level_pressure = 1013.25
```

Now you have access to the `temperature`, `pressure` and `altitude` attributes

```
temperature = bmp280.temperature
pressure = bmp280.pressure
altitude = bmp280.altitude
```

class adafruit_bmp280.**Adafruit_BMP280_SPI**(*spi*, *cs*, *baudrate=100000*)

Driver for SPI connected BMP280.

Parameters

- **spi** (*SPI*) – SPI device
- **cs** (*DigitalInOut*) – Chip Select
- **baudrate** (*int*) – Clock rate, default is 100000. Can be changed with `baudrate()`

Quickstart: Importing and using the BMP280

Here is an example of using the `BMP280_SPI` class. First you will need to import the libraries to use the sensor

```
import board
from digitalio import DigitalInOut, Direction
import adafruit_bmp280
```

Once this is done you can define your `board.SPI` object and define your sensor object

```
cs = digitalio.DigitalInOut(board.D10)
spi = board.SPI()
bme280 = adafruit_bmp280.Adafruit_bmp280_SPI(spi, cs)
```

You need to setup the pressure at sea level

```
bmp280.sea_level_pressure = 1013.25
```

Now you have access to the `temperature`, `pressure` and `altitude` attributes

```
temperature = bmp280.temperature
pressure = bmp280.pressure
altitude = bmp280.altitude
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index