AdafruitSCD30 Library Documentation

Release 1.0

Bryan Siepert

Jun 15, 2021

CONTENTS

Helper library for the SCD30 CO2 sensor

CHAPTER

ONE

DEPENDENCIES

This driver depends on:

- Adafruit CircuitPython
- Bus Device
- Register

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

CHAPTER

TWO

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-scd30
```

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

```
sudo pip3 install adafruit-circuitpython-scd30
```

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-scd30
```

CHAPTER

THREE

USAGE EXAMPLE

```
import time
import board
import adafruit_scd30

i2c = board.I2C()   # uses board.SCL and board.SDA
scd = adafruit_scd30.SCD30(i2c)

while True:
    # since the measurement interval is long (2+ seconds) we check for new data before␣
↪reading
    # the values, to ensure current readings.
    if scd.data_available:
        print("Data Available!")
        print("CO2:", scd.CO2, "PPM")
        print("Temperature:", scd.temperature, "degrees C")
        print("Humidity:", scd.relative_humidity, "%%rH")
        print("")
        print("Waiting for new data...")
        print("")

    time.sleep(0.5)
```

CHAPTER

FOUR

CONTRIBUTING

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

CHAPTER

FIVE

DOCUMENTATION

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

CHAPTER

SIX

TABLE OF CONTENTS

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/scd30_simpletest.py

```
# SPDX-FileCopyrightText: 2020 by Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time
import board
import busio
import adafruit_scd30

# SCD-30 has tempremental I2C with clock stretching, datasheet recommends
# starting at 50KHz
i2c = busio.I2C(board.SCL, board.SDA, frequency=50000)
scd = adafruit_scd30.SCD30(i2c)

while True:
    # since the measurement interval is long (2+ seconds) we check for new data before reading
    # the values, to ensure current readings.
    if scd.data_available:
        print("Data Available!")
        print("CO2: %d PPM" % scd.CO2)
        print("Temperature: %0.2f degrees C" % scd.temperature)
        print("Humidity: %0.2f %% rH" % scd.relative_humidity)
        print("")
        print("Waiting for new data...")
        print("")

    time.sleep(0.5)
```

6.2 Tuning Knobs

Experiment with different tuning parameters and settings

Listing 2: examples/scd30_tuning_knobs.py

```
# SPDX-FileCopyrightText: 2020 by Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time
import board
import busio
import adafruit_scd30

# SCD-30 has tempremental I2C with clock stretching, datasheet recommends
# starting at 50KHz
i2c = busio.I2C(board.SCL, board.SDA, frequency=50000)
scd = adafruit_scd30.SCD30(i2c)
# scd.temperature_offset = 10
print("Temperature offset:", scd.temperature_offset)

# scd.measurement_interval = 4
print("Measurement interval:", scd.measurement_interval)

# scd.self_calibration_enabled = True
print("Self-calibration enabled:", scd.self_calibration_enabled)

# scd.ambient_pressure = 1100
print("Ambient Pressure:", scd.ambient_pressure)

# scd.altitude = 100
print("Altitude:", scd.altitude, "meters above sea level")

# scd.forced_recalibration_reference = 409
print("Forced recalibration reference:", scd.forced_recalibration_reference)
print("")

while True:
    data = scd.data_available
    if data:
        print("Data Available!")
        print("CO2:", scd.CO2, "PPM")
        print("Temperature:", scd.temperature, "degrees C")
        print("Humidity::", scd.relative_humidity, "%%rH")
        print("")
        print("Waiting for new data...")
        print("")

    time.sleep(0.5)
```

6.3 MCP2221 and SCD30 Example

MCP2221 is known to not like the SCD30. Here is how to avoid this!

Listing 3: examples/scd30_tuning_knobs.py

```
# SPDX-FileCopyrightText: 2020 by Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time
import board
import busio
import adafruit_scd30

# SCD-30 has tempremental I2C with clock stretching, datasheet recommends
# starting at 50KHz
i2c = busio.I2C(board.SCL, board.SDA, frequency=50000)
scd = adafruit_scd30.SCD30(i2c)
# scd.temperature_offset = 10
print("Temperature offset:", scd.temperature_offset)

# scd.measurement_interval = 4
print("Measurement interval:", scd.measurement_interval)

# scd.self_calibration_enabled = True
print("Self-calibration enabled:", scd.self_calibration_enabled)

# scd.ambient_pressure = 1100
print("Ambient Pressure:", scd.ambient_pressure)

# scd.altitude = 100
print("Altitude:", scd.altitude, "meters above sea level")

# scd.forced_recalibration_reference = 409
print("Forced recalibration reference:", scd.forced_recalibration_reference)
print("")

while True:
    data = scd.data_available
    if data:
        print("Data Available!")
        print("CO2:", scd.CO2, "PPM")
        print("Temperature:", scd.temperature, "degrees C")
        print("Humidity::", scd.relative_humidity, "%%rH")
        print("")
        print("Waiting for new data...")
        print("")

    time.sleep(0.5)
```

6.4 adafruit_scd30

Helper library for the SCD30 CO2 sensor

- Author(s): Bryan Siepert

6.4.1 Implementation Notes

Hardware:

- Adafruit SCD30 Breakout

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
- Adafruit's Register library: https://github.com/adafruit/Adafruit_CircuitPython_Register

class adafruit_scd30.**SCD30**(*i2c_bus*, *ambient_pressure=0*, *address=97*)

CircuitPython helper class for using the SCD30 CO2 sensor

Parameters

- **i2c_bus** (*I2C*) – The I2C bus the SCD30 is connected to.
- **ambient_pressure** (*int*) – Ambient pressure compensation. Defaults to `0`
- **address** (*int*) – The I2C device address for the sensor. Default is `0x61`

Quickstart: Importing and using the SCD30

Here is an example of using the *SCD30* class. First you will need to import the libraries to use the sensor

```
import board
import adafruit_scd30
```

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
scd = adafruit_scd30.SCD30(i2c)
```

Now you have access to the CO2, temperature and humidity using the *CO2*, *temperature* and *relative_humidity* attributes

```
temperature = scd.temperature
relative_humidity = scd.relative_humidity
co2_ppm_level = scd.CO2
```

property CO2

Returns the CO2 concentration in PPM (parts per million)

Note: Between measurements, the most recent reading will be cached and returned.

property altitude

Specifies the altitude at the measurement location in meters above sea level. Setting this value adjusts the CO2 measurement calculations to account for the air pressure's effect on readings.

Note: This value will be saved and will not be reset on boot or by calling *reset*.

property ambient_pressure
Specifies the ambient air pressure at the measurement location in mBar. Setting this value adjusts the CO2 measurement calculations to account for the air pressure's effect on readings. Values must be in mBar, from 700 to 1400 mBar

property data_available
Check the sensor to see if new data is available

property forced_recalibration_reference
Specifies the concentration of a reference source of CO2 placed in close proximity to the sensor. The value must be from 400 to 2000 ppm.

Note: Specifying a forced recalibration reference will override any calibration values set by Automatic Self Calibration

property measurement_interval
Sets the interval between readings in seconds. The interval value must be from 2-1800

Note: This value will be saved and will not be reset on boot or by calling *reset*.

property relative_humidity
Returns the current relative humidity in %rH.

Note: Between measurements, the most recent reading will be cached and returned.

reset()
Perform a soft reset on the sensor, restoring default values

property self_calibration_enabled
Enables or disables automatic self calibration (ASC). To work correctly, the sensor must be on and active for 7 days after enabling ASC, and exposed to fresh air for at least 1 hour per day. Consult the manufacturer's documentation for more information.

Note: Enabling self calibration will override any values set by specifying a *forced_recalibration_reference*

Note: This value will be saved and will not be reset on boot or by calling *reset*.

property temperature
Returns the current temperature in degrees Celsius

Note: Between measurements, the most recent reading will be cached and returned.

property temperature_offset
Specifies the offset to be added to the reported measurements to account for a bias in the measured signal.

Value is in degrees Celsius with a resolution of 0.01 degrees and a maximum value of 655.35 C

Note: This value will be saved and will not be reset on boot or by calling *reset*.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX