# AdafruitSGP40 Library Documentation

***Release 1.0***

**Bryan Siepert**

**Jun 06, 2021**

# CONTENTS

CircuitPython library for the Adafruit SGP40 Air Quality Sensor / VOC Index Sensor Breakouts

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-sgp40
```

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

```
sudo pip3 install adafruit-circuitpython-sgp40
```

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-sgp40
```

CHAPTER

# THREE

# USAGE EXAMPLE

```
import time
import board
import adafruit_sgp40

i2c = board.I2C()  # uses board.SCL and board.SDA
sgp = adafruit_sgp40(i2c)

while True:
    print("Measurement: ", sgp.raw)
    print("")
    sleep(1)
```

For humidity compensated raw gas readings, we'll need a secondary sensor such as the bme280

```
import time
import board
import adafruit_sgp40
import adafruit_bme280

i2c = board.I2C()  # uses board.SCL and board.SDA
sgp = adafruit_sgp40.SGP40(i2c)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

while True:
    temperature = bme280.temperature
    humidity = bme280.relative_humidity
    compensated_raw_gas = sgp.measure_raw(temperature = temperature, relative_humidity =␣
↪humidity)
    print(compensated_raw_gas)
    print("")
    time.sleep(1)
```

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/sgp40_simpletest.py

```
# SPDX-FileCopyrightText: 2020 by Bryan Siepert for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time
import board
import adafruit_sgp40

# If you have a temperature sensor, like the bme280, import that here as well
# import adafruit_bme280

i2c = board.I2C()  # uses board.SCL and board.SDA
sgp = adafruit_sgp40.SGP40(i2c)
# And if you have a temp/humidity sensor, define the sensor here as well
# bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

while True:
    print("Raw Gas: ", sgp.raw)
    # Lets quickly grab the humidity and temperature
    # temperature = bme280.temperature
    # humidity = bme280.relative_humidity
    # compensated_raw_gas = sgp.measure_raw(temperature = temperature, relative_humidity
↪= humidity)
    print("")
    time.sleep(1)
```

## 6.2 adafruit_sgp40

CircuitPython library for the Adafruit SGP40 Air Quality Sensor / VOC Index Sensor Breakouts

- **Author(s): Bryan Siepert** Keith Murray

### 6.2.1 Implementation Notes

**Hardware:**

- Adafruit SGP40 Air Quality Sensor Breakout - VOC Index <https://www.adafruit.com/product/4829>
- In order to use the *measure_raw* function, a temperature and humidity sensor which updates at at least 1Hz is needed (BME280, BME688, SHT31-D, SHT40, etc. For more, see: https://www.adafruit.com/category/66)

**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_sgp40.**SGP40**(*i2c*, *address=89*)

Class to use the SGP40 Air Quality Sensor Breakout

**Parameters** **address** (*int*) – The I2C address of the device. Defaults to `0x59`

**Quickstart: Importing and using the SGP40 temperature sensor**

Here is one way of importing the *SGP40* class so you can use it with the name `sgp`. First you will need to import the libraries to use the sensor

```
import board
import adafruit_sgp40
# If you have a temperature sensor, like the bme280, import that here as␣
↪well
# import adafruit_bme280
```

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
sgp = adafruit_sgp40.SGP40(i2c)
# And if you have a temp/humidity sensor, define the sensor here as well
# bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)
```

Now you have access to the raw gas value using the *raw* attribute. And with a temperature and humidity value, you can access the class function *measure_raw()* for a humidity compensated raw reading

```
raw_gas_value = sgp.raw
# Lets quickly grab the humidity and temperature
# temperature = bme280.temperature
# humidity = bme280.relative_humidity
# compensated_raw_gas = sgp.measure_raw(temperature=temperature,
# relative_humidity=humidity)
# temperature = temperature, relative_humidity = humidity)
```

---

**Note:** The operational range of temperatures for the SGP40 is -10 to 50 degrees Celsius and the operational range of relative humidity for the SGP40 is 0 to 90 % (assuming that humidity is non-condensing).

Humidity compensation is further optimized for a subset of the temperature and relative humidity readings. See Figure 3 of the Sensirion datasheet for the SGP40. At 25 degrees Celsius, the optimal range for relative humidity is 8% to 90%. At 50% relative humidity, the optimal range for temperature is -7 to 42 degrees Celsius.

Prolonged exposures outside of these ranges may reduce sensor performance, and the sensor must not be exposed towards condensing conditions at any time.

For more information see: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_Datasheet_SGP40.pdf and https://learn.adafruit.com/adafruit-sgp40

---

`initialize()`
Reset the sensor to it's initial unconfigured state and configure it with sensible defaults so it can be used

`measure_raw`(*temperature=25*, *relative_humidity=50*)
A humidity and temperature compensated raw gas value which helps address fluctuations in readings due to changing humidity.

**Parameters**

- **temperature** (*float*) – The temperature in degrees Celsius, defaults to `25`
- **relative_humidity** (*float*) – The relative humidity in percentage, defaults to `50`

The raw gas value adjusted for the current temperature (c) and humidity (%)

`property raw`
The raw gas value

CHAPTER

SEVEN

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## a

# INDEX