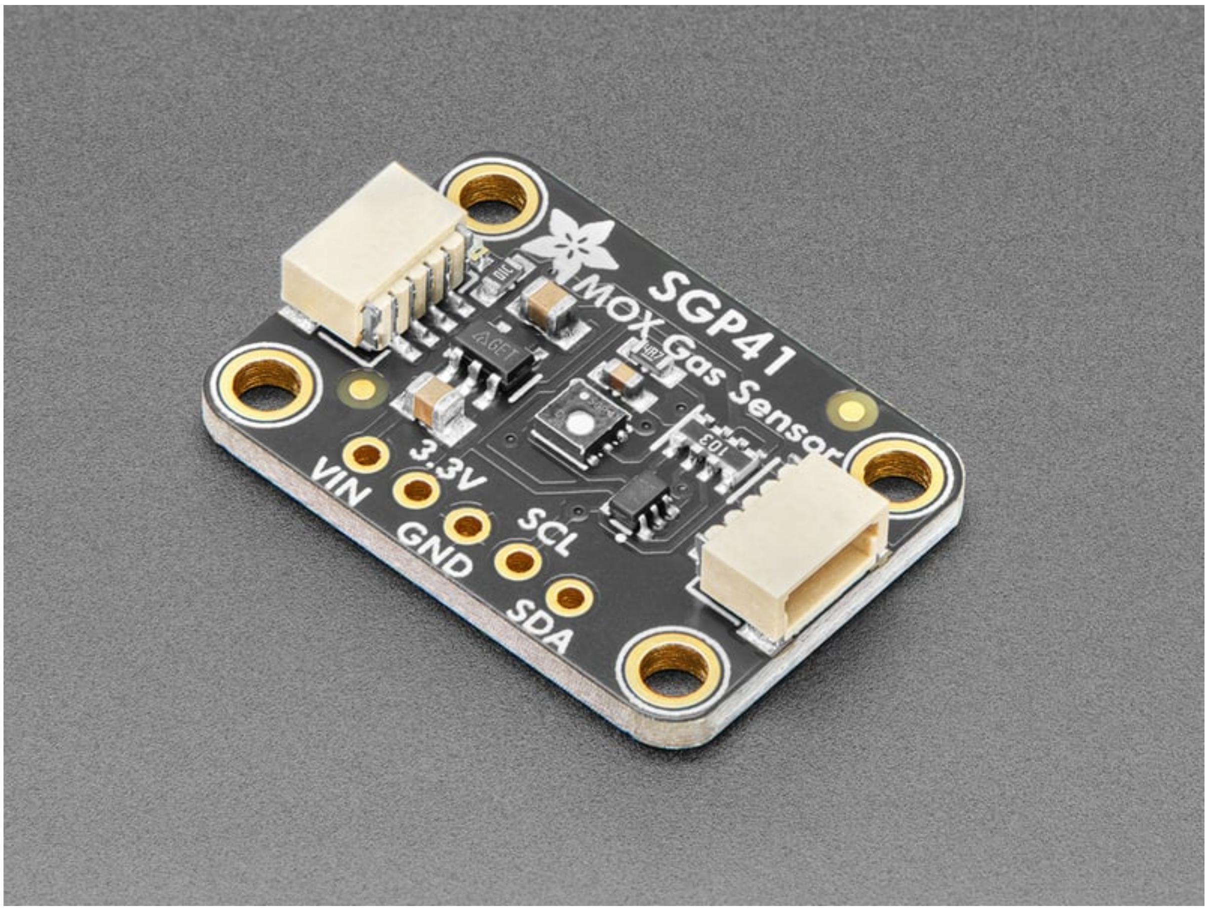
Adafruit SGP41 Multi-Pixel Gas Sensor Breakout

Created by Liz Clark



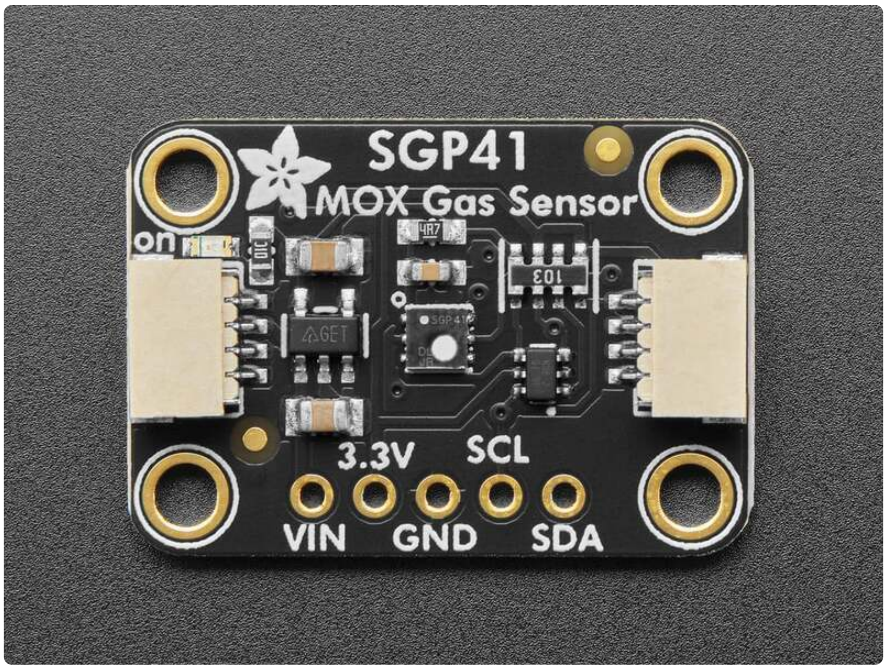
https://learn.adafruit.com/adafruit-sgp41-multi-pixel-gas-sensor-breakout

Last updated on 2026-02-17 08:19:33 PM UTC

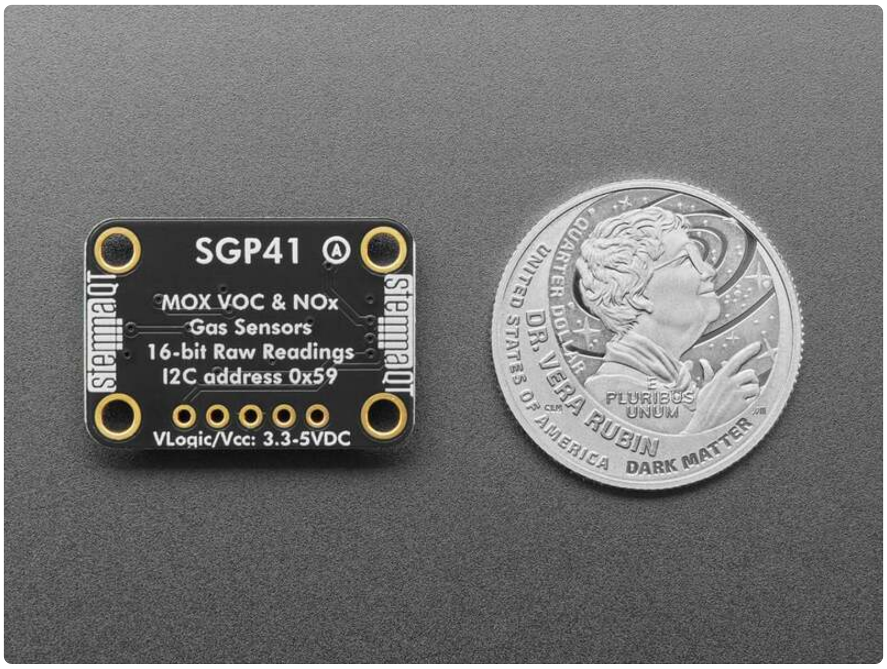
Table of Contents

Overview

sniff *sniff* ... do you smell that? No need to stick your nose into a carton of milk anymore, you can build a digital nose with the **Adafruit SGP41 Multi-Pixel Gas Sensor**, a fully integrated MOX gas sensor. This is a very fine air quality sensor from the sensor experts at Sensirion, with I2C interfacing so you don't have to manage the heater and analog reading of a MOX sensor. It combines multiple metal-oxide sensing and heating elements on one chip to provide more detailed air quality signals - and, unlike the SGP40, the SGP41 adds a dedicated NOx sensing pixel so you can measure both volatile organic compounds (https://adafru.it/1azV) (VOC) and nitrogen oxides!

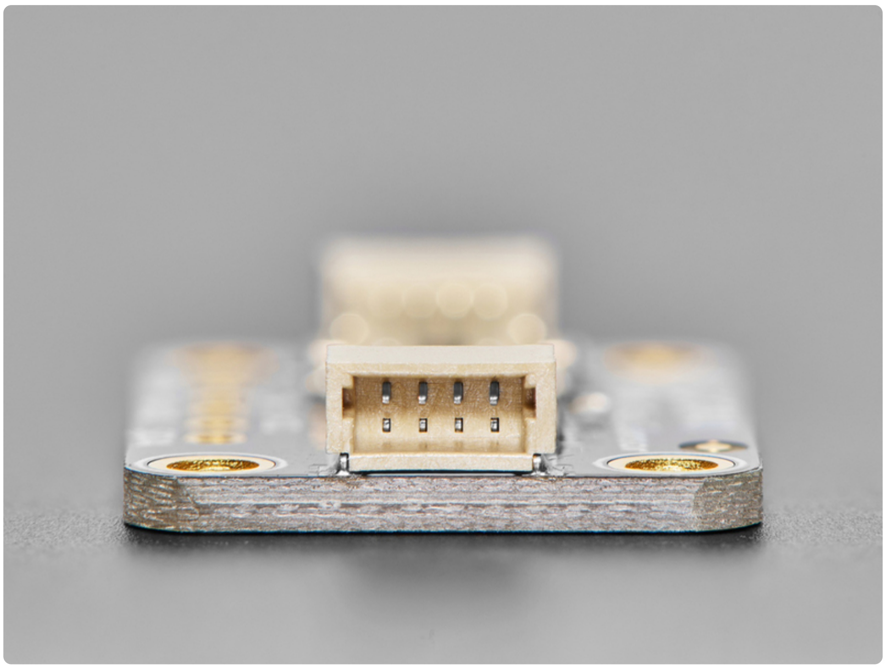


The SGP41 has two MOX hot-plate sensor elements (one for VOC, one for NOx), as well as a small microcontroller that controls power to the plates, reads the analog voltages, and provides an I2C interface to read from. Unlike the CCS811, this sensor does not require I2C clock stretching. We currently have an Arduino library for reading the raw VOC and NOx signals (https://adafru.it/1azW) and a Python/CircuitPython library (https://adafru.it/1azX) that can be used with Linux computers like the Raspberry Pi or our CircuitPython boards.

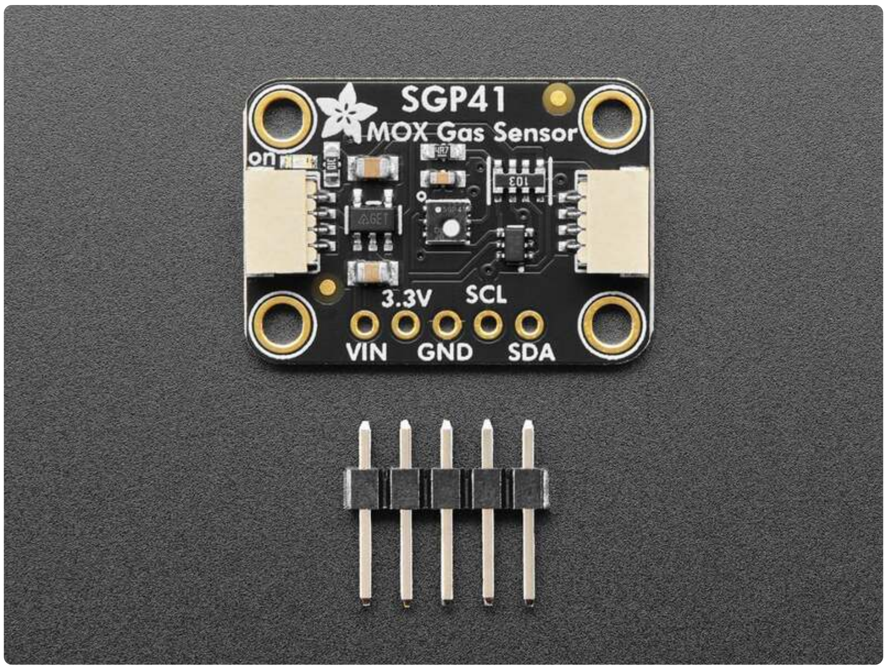


This is a gas sensor that can detect a wide range of Volatile Organic Compounds (VOCs) as well as oxidizing gases such as NOx, and is intended for indoor air quality monitoring. **The SGP41 is the next generation after the SGP40, adding a second NOx sensing pixel. It does not give TVOC/eCO2 values out like the SGP30.** Instead, the two raw signals from the sensor are processed using Sensirion's Gas Index Algorithm to give VOC and NOx Index values on a scale from 1 to 500 (https://adafru.it/1azY) - where 100 indicates a typical indoor environment.

Please note, this sensor, like all VOC/gas sensors, has variability, and to get precise measurements you will want to calibrate it against known sources! That said, for general environmental sensors, it will give you a good idea of trends and comparison.



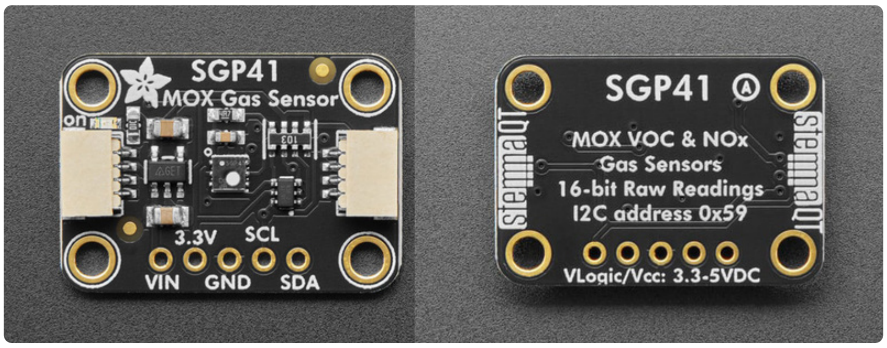
Another nice element to this sensor is the ability to set humidity and temperature compensation for better accuracy. An external humidity/temperature sensor is required and then the RH% and temperature are written over I2C to the sensor, so it can better calibrate the MOX sensor readings and reduce humidity/temperature-based variations.



Nice sensor right? So we made it easy for you to get right into your next project. The surface-mount sensor is soldered onto a custom made PCB in the **STEMMA QT** form factor (https://adafru.it/LBQ), making them easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the SGP41 or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB). QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE)

We've, of course, broken out all the pins to standard headers and added a 3.3V voltage regulator and level shifting to allow you to use it with either 3.3V or 5V systems, such as the Arduino Uno, or Feather M4.

Pinouts



The default I2C address is **0x59**.

Power Pins

- **VIN** - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V. For a 3.3V microcontroller, use 3.3V.
- **3.3V** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like.
- **GND** - common ground for power and logic.

I2C Logic Pins

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- STEMMA QT **(https://adafru.it/Ft4) -** These connectors allow you to connect to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6).

Power LED

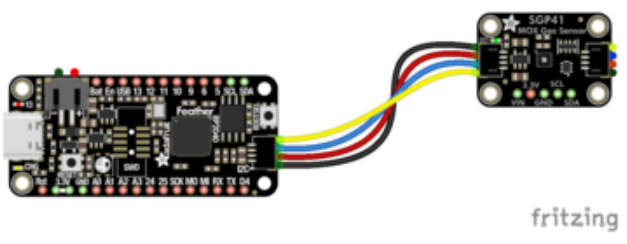
- **Power LED -** In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled **on**. It is a green LED.

CircuitPython and Python

It's easy to use the **SGP41** with Python or CircuitPython, and the Adafruit_CircuitPython_SGP41 (https://adafru.it/1azX) module. This module allows you to easily write Python code that allows you to read the **SGP41** gas sensor. You can use this gas sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

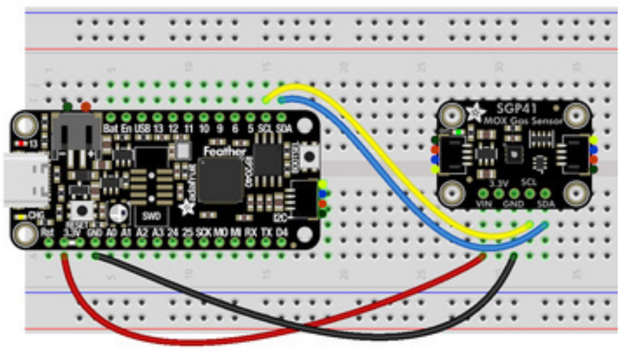
CircuitPython Microcontroller Wiring

First, wire up an SGP41 to your board exactly as shown below. Here's an example of wiring a Feather RP2040 to the SGP41 with I2C using one of the handy STEMMA QT (https://adafru.it/Ft4) connectors:



Board STEMMA 3V to breakout **STEMMA VIN (red wire)**
Board STEMMA GND to breakout **STEMMA GND (black wire)**
Board STEMMA SCL to breakout **STEMMA SCL (yellow wire)**
Board STEMMA SDA to breakout **STEMMA SDA (blue wire)**

You can also use standard **0.100" pitch** headers to wire it up on a breadboard:

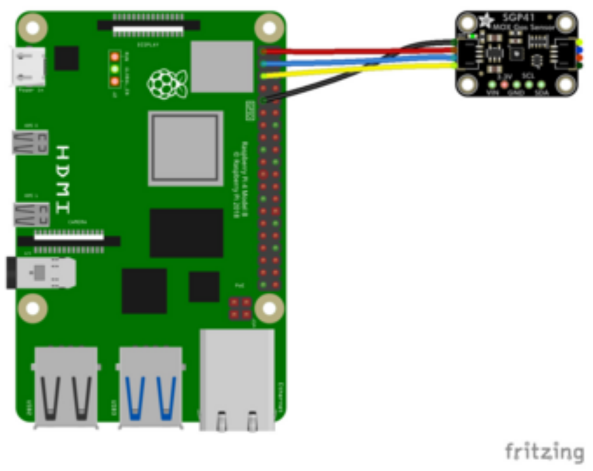


Board 3V to breakout **VIN (red wire)**
Board GND to breakout **GND (black wire)**
Board SCL to breakout **SCL (yellow wire)**
Board SDA to breakout **SDA (blue wire)**

Python Computer Wiring

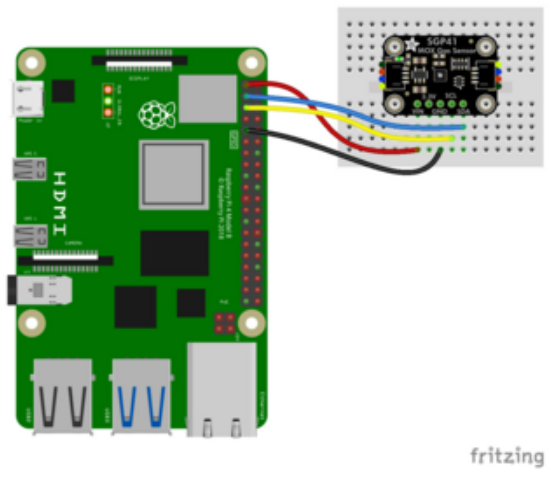
Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the gas sensor using I2C and a **STEMMA QT** (https://adafru.it/Ft4) connector:



Pi 3V to **breakout STEMMA VIN (red wire)**
Pi GND to **breakout STEMMA GND (black wire)**
Pi SCL to **breakout STEMMA SCL (yellow wire)**
Pi SDA to **breakout STEMMA SDA (blue wire)**

Finally, here is an example of how to wire up a Raspberry Pi to the gas sensor using a solderless breadboard:



Pi 3V to **breakout VIN (red wire)**
Pi GND to **breakout GND (black wire)**
Pi SCL to **breakout SCL (yellow wire)**
Pi SDA to **breakout SDA (blue wire)**

Python Installation of SGP41 Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `pip3 install adafruit-circuitpython-sgp41`

If your default Python is version 3, you may need to run `pip` instead. Make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

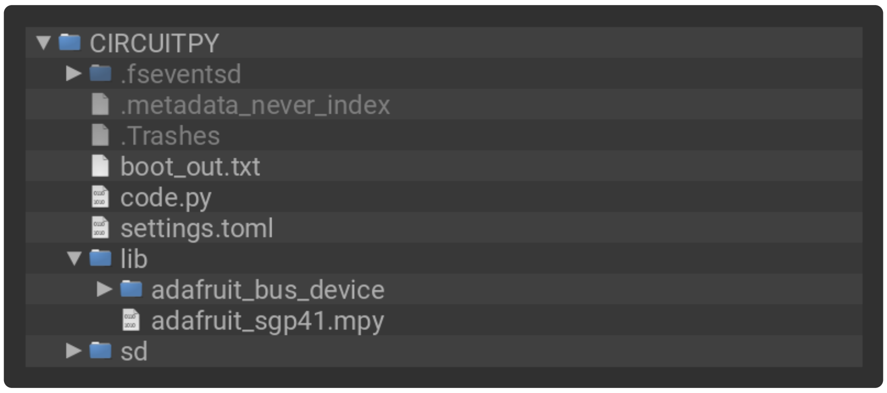
CircuitPython Usage

To use with CircuitPython, you need to first install the SGP41 library and its dependencies into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_bus_device/**
- **adafruit_sgp41.mpy**



```
▼ CIRCUITPY
  ▶ .fseventsd
    .metadata_never_index
    .Trashes
    boot_out.txt
    code.py
    settings.toml
  ▼ lib
    ▶ adafruit_bus_device
      adafruit_sgp41.mpy
  ▶ sd
```

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`python3 code.py`

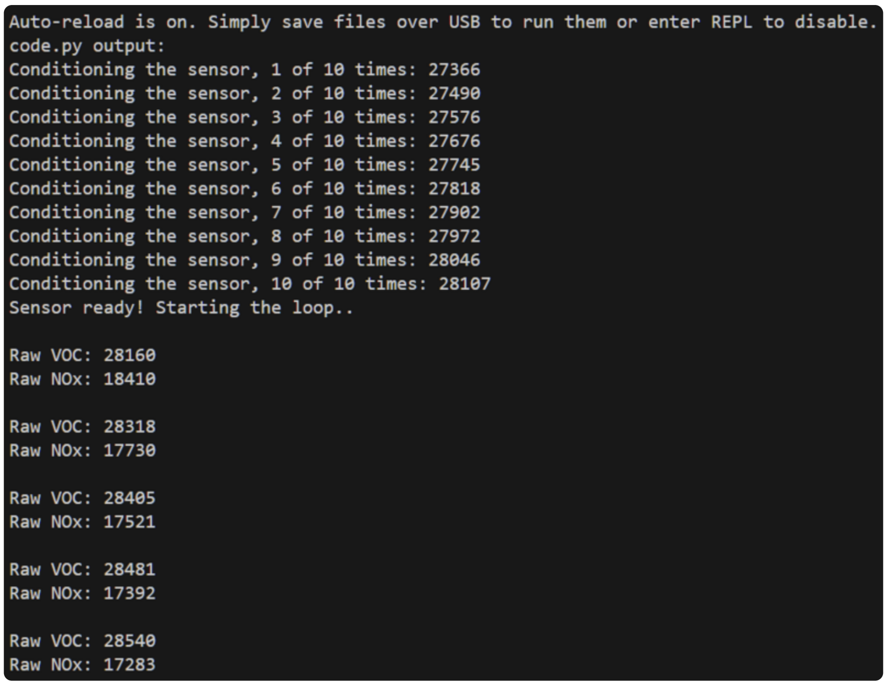
Example Code

If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, [connect to the serial console](https://adafru.it/Bec) (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

```python
# SPDX-FileCopyrightText: 2026 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""Simple test for the SGP41 sensor"""

import time

import board

from adafruit_sgp41 import Adafruit_SGP41

i2c = board.I2C()
sensor = Adafruit_SGP41(i2c)

# set ambient temperature and relative humidity
# for more accurate readings from the sensor
# can be used in conjunction with an external
# temp and humidity sensor
# sensor.temperature = 22.2 # Celsius
# sensor.relative_humidity = 30.9

for i in range(10):
    condition = sensor.conditioning()
    print(f"Conditioning the sensor, {(i + 1)} of 10 times: {condition}")
    time.sleep(1)

print("Sensor ready! Starting the loop..")
print()

while True:
    print(f"Raw VOC: {sensor.raw_voc}")
    print(f"Raw NOx: {sensor.raw_nox}")
    print()
    time.sleep(1)
```

In the example, the gas sensor is instantiated on I2C and conditioned 10 times over 10 seconds. Then, in the loop, raw VOC and NOx readings are printed to the REPL every second.



```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Conditioning the sensor, 1 of 10 times: 27366
Conditioning the sensor, 2 of 10 times: 27490
Conditioning the sensor, 3 of 10 times: 27576
Conditioning the sensor, 4 of 10 times: 27676
Conditioning the sensor, 5 of 10 times: 27745
Conditioning the sensor, 6 of 10 times: 27818
Conditioning the sensor, 7 of 10 times: 27902
Conditioning the sensor, 8 of 10 times: 27972
Conditioning the sensor, 9 of 10 times: 28046
Conditioning the sensor, 10 of 10 times: 28107
Sensor ready! Starting the loop..

Raw VOC: 28160
Raw NOx: 18410

Raw VOC: 28318
Raw NOx: 17730

Raw VOC: 28405
Raw NOx: 17521

Raw VOC: 28481
Raw NOx: 17392

Raw VOC: 28540
Raw NOx: 17283
```

Python Docs

Python Docs (https://adafru.it/1azJ)

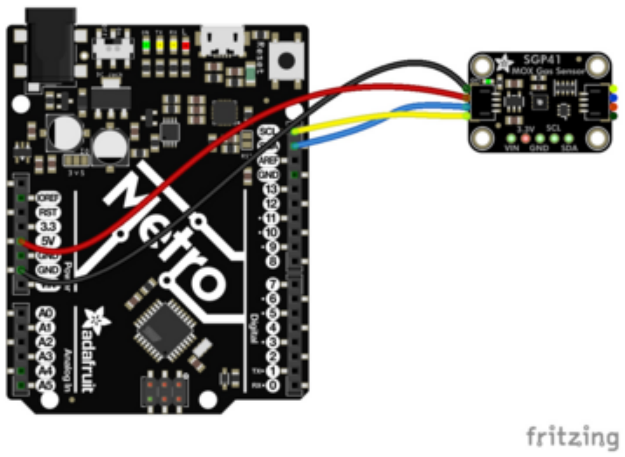
Arduino

Using the SGP41 gas sensor with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit_SGP41 (https://adafru.it/1azW) library and running the provided example code.

Wiring

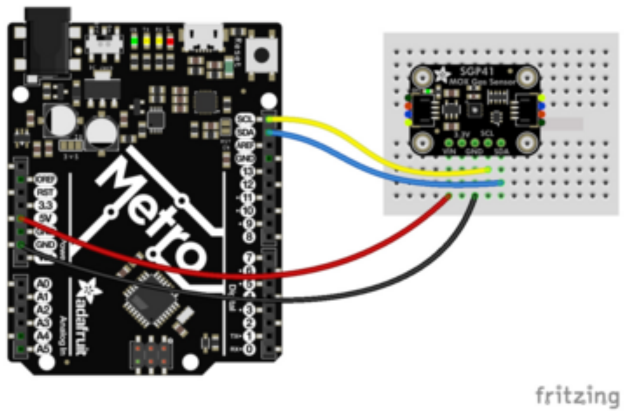
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the SGP41 VIN.

Here is an Adafruit Metro wired up to the SGP41 using the STEMMA QT connector:



Board 5V to **breakout STEMMA VIN (red wire)**
Board GND to **breakout STEMMA GND (black wire)**
Board SCL to **breakout STEMMA SCL (yellow wire)**
Board SDA to **breakout STEMMA SDA (blue wire)**

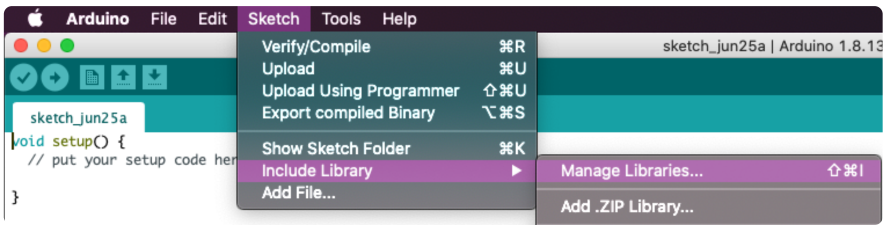
Here is an Adafruit Metro wired up using a solderless breadboard:



Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**

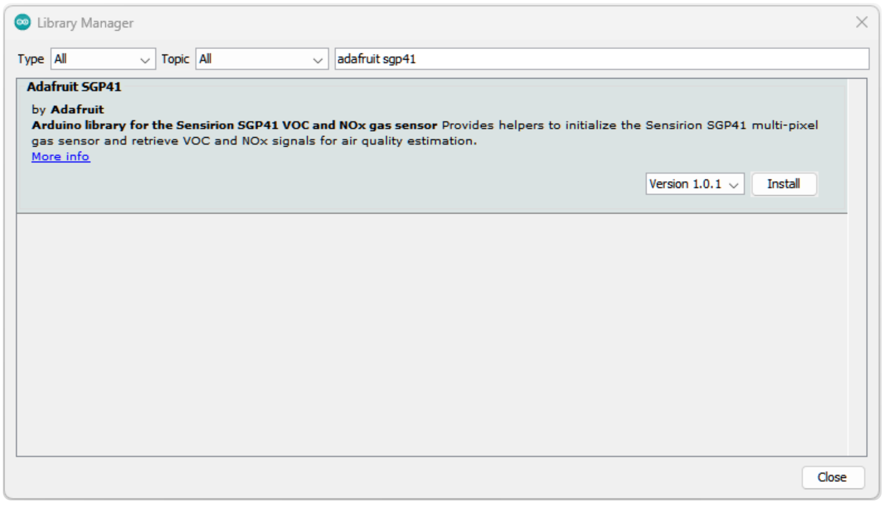
Library Installation

You can install the **Adafruit SGP41** library for Arduino using the Library Manager in the Arduino IDE.



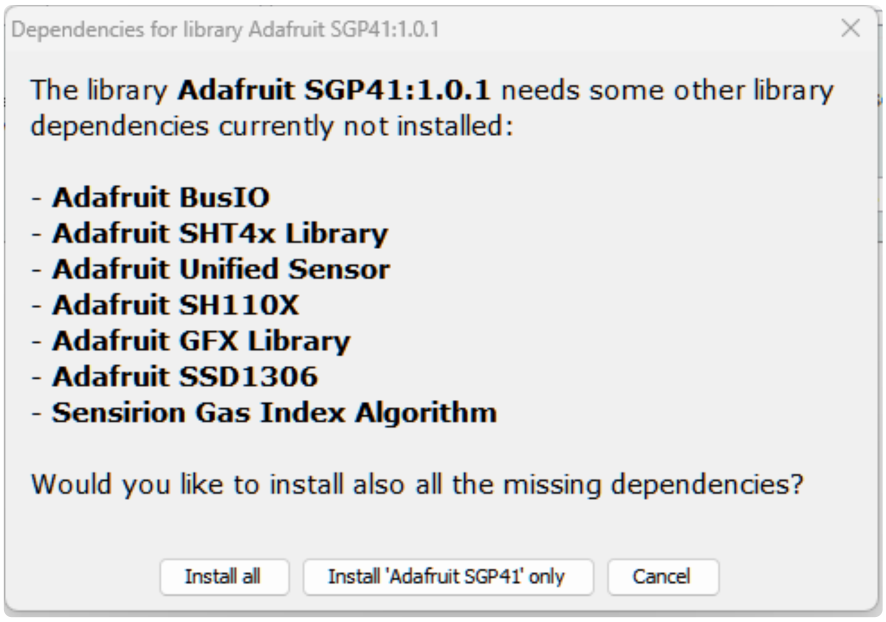
```
void setup() {
  // put your setup code her
}
```

Click the **Manage Libraries ...** menu item, search for **Adafruit SGP41**, and select the **Adafruit SGP41** library:



Adafruit SGP41
by **Adafruit**
Arduino library for the Sensirion SGP41 VOC and NOx gas sensor Provides helpers to initialize the Sensirion SGP41 multi-pixel gas sensor and retrieve VOC and NOx signals for air quality estimation.
More info

If asked about dependencies, click "**Install all**".


Dependencies for library Adafruit SGP41:1.0.1

The library **Adafruit SGP41:1.0.1** needs some other library dependencies currently not installed:

- **Adafruit BusIO**
- **Adafruit SHT4x Library**
- **Adafruit Unified Sensor**
- **Adafruit SH110X**
- **Adafruit GFX Library**
- **Adafruit SSD1306**
- **Sensirion Gas Index Algorithm**

Would you like to install also all the missing dependencies?

Install all
Install 'Adafruit SGP41' only
Cancel

If the "Dependencies" window does not come up, then you already have the dependencies installed.

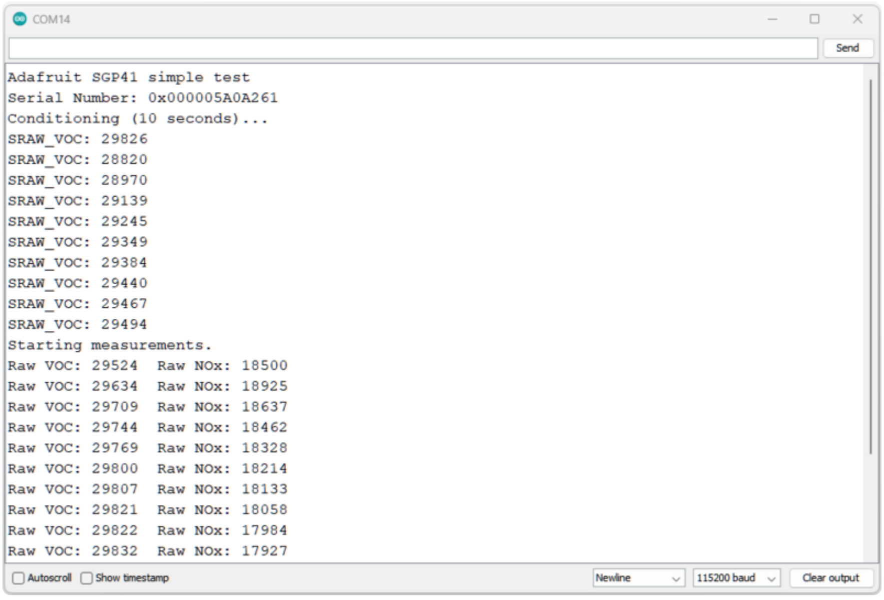
e dependencies are already installed, you must make sure you update
n through the Arduino Library Manager before loading the example!

Example Code

```
/**
 * @file simpletest.ino
 *
 * Simple test for the Adafruit SGP41 sensor.
 *
 * Written by Limor 'ladyada' Fried with assistance from Claude Code for
 * Adafruit Industries. MIT license, all text above must be included in any
 * redistribution.
 */

#include <Adafruit_SGP41.h>

Adafruit_SGP41 sgp41;

void setup(void) {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }

  Serial.println(F("Adafruit SGP41 simple test"));

  if (!sgp41.begin()) {
    Serial.println(F("Could not find SGP41! Halting."));
    while (1) {
      delay(10);
    }
  }

  uint16_t serial_number[3];
  if (sgp41.getSerialNumber(serial_number)) {
    Serial.print(F("Serial Number: 0x"));
    for (uint8_t i = 0; i < 3; i++) {
      if (serial_number[i] < 0x1000) {
        Serial.print(F("0"));
      }
      if (serial_number[i] < 0x100) {
        Serial.print(F("0"));
      }
      if (serial_number[i] < 0x10) {
        Serial.print(F("0"));
      }
      Serial.print(serial_number[i], HEX);
    }
    Serial.println();
  } else {
    Serial.println(F("Failed to read serial number! Halting."));
    while (1) {
      delay(10);
    }
  }

  Serial.println(F("Conditioning (10 seconds)..."));
  for (uint8_t i = 0; i < 10; i++) {
    uint16_t sraw_voc = 0;
    if (sgp41.executeConditioning(&sraw_voc)) {
      Serial.print(F("SRAW_VOC: "));
      Serial.println(sraw_voc);
    } else {
      Serial.println(F("Conditioning failed!"));
    }
    delay(1000);
  }

  Serial.println(F("Starting measurements."));
}
```

```
void loop(void) {
  uint16_t sraw_voc = 0;
  uint16_t sraw_nox = 0;

  if (sgp41.measureRawSignals(&sraw_voc, &sraw_nox)) {
    Serial.print(F("Raw VOC: "));
    Serial.print(sraw_voc);
    Serial.print(F("\tRaw NOx: "));
    Serial.println(sraw_nox);
  } else {
    Serial.println(F("Measurement failed!"));
  }

  delay(1000);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools** -> **Serial Monitor**) at 115200 baud. You should see that the sketch has found your connected SGP41 sensor. Then, after conditioning the sensor, raw readings for VOC and NOx are printed to the Serial Monitor every second.



```
Adafruit SGP41 simple test
Serial Number: 0x000005A0A261
Conditioning (10 seconds)...
SRAW_VOC: 29826
SRAW_VOC: 28820
SRAW_VOC: 28970
SRAW_VOC: 29139
SRAW_VOC: 29245
SRAW_VOC: 29349
SRAW_VOC: 29384
SRAW_VOC: 29440
SRAW_VOC: 29467
SRAW_VOC: 29494
Starting measurements.
Raw VOC: 29524  Raw NOx: 18500
Raw VOC: 29634  Raw NOx: 18925
Raw VOC: 29709  Raw NOx: 18637
Raw VOC: 29744  Raw NOx: 18462
Raw VOC: 29769  Raw NOx: 18328
Raw VOC: 29800  Raw NOx: 18214
Raw VOC: 29807  Raw NOx: 18133
Raw VOC: 29821  Raw NOx: 18058
Raw VOC: 29822  Raw NOx: 17984
Raw VOC: 29832  Raw NOx: 17927
```

Arduino Docs

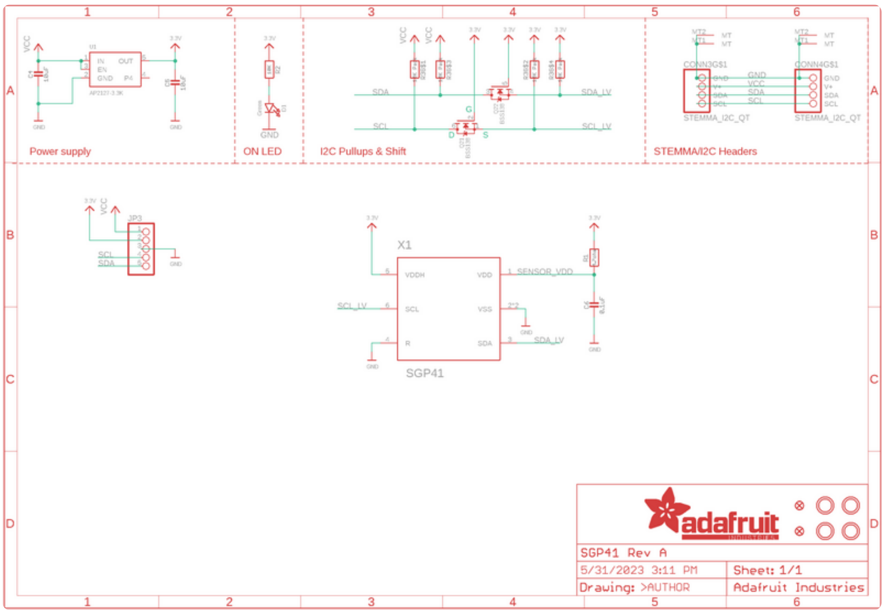
[Arduino Docs](https://adafru.it/1azK) (https://adafru.it/1azK)

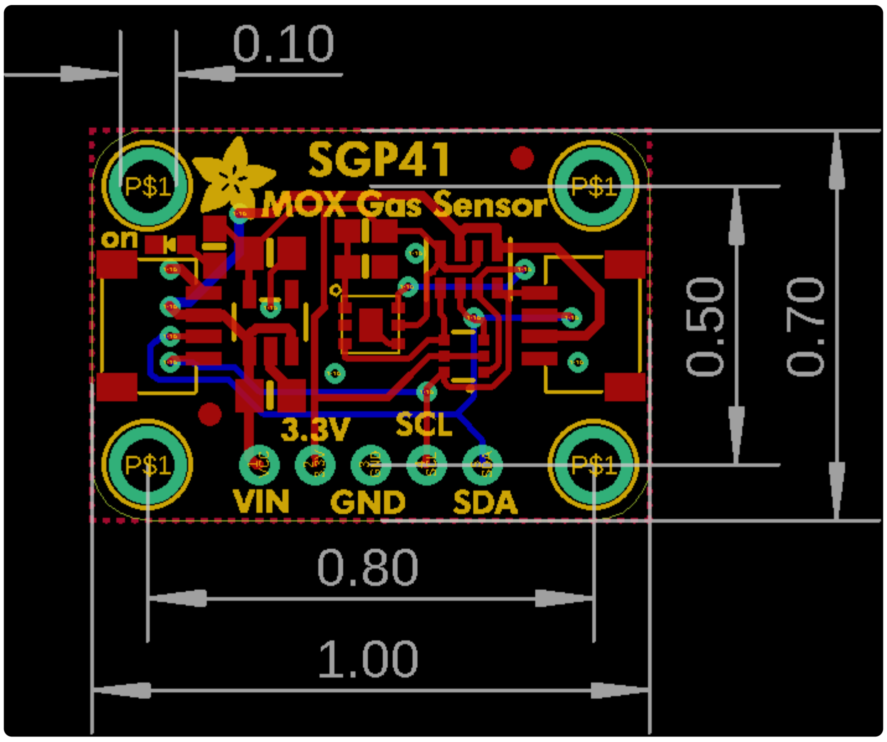
Downloads

Files

- SGP41 Datasheet (https://adafru.it/1azZ)
- EagleCAD PCB files on GitHub (https://adafru.it/1aA0)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1aA1)

Schematic and Fab Print




0.10
SGP41
MOX Gas Sensor
on
P$1
3.3V
SCL
VIN
GND
SDA
0.50
0.70
0.80
1.00