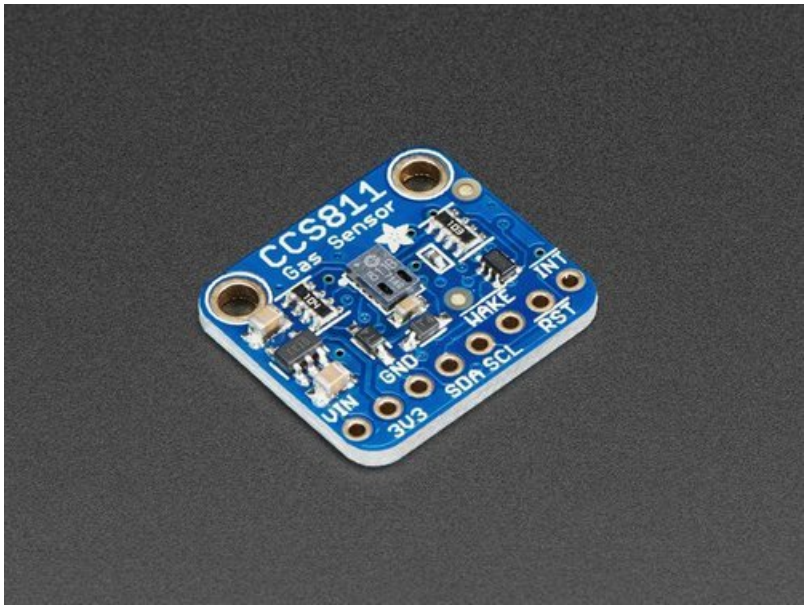
adafruit learning system

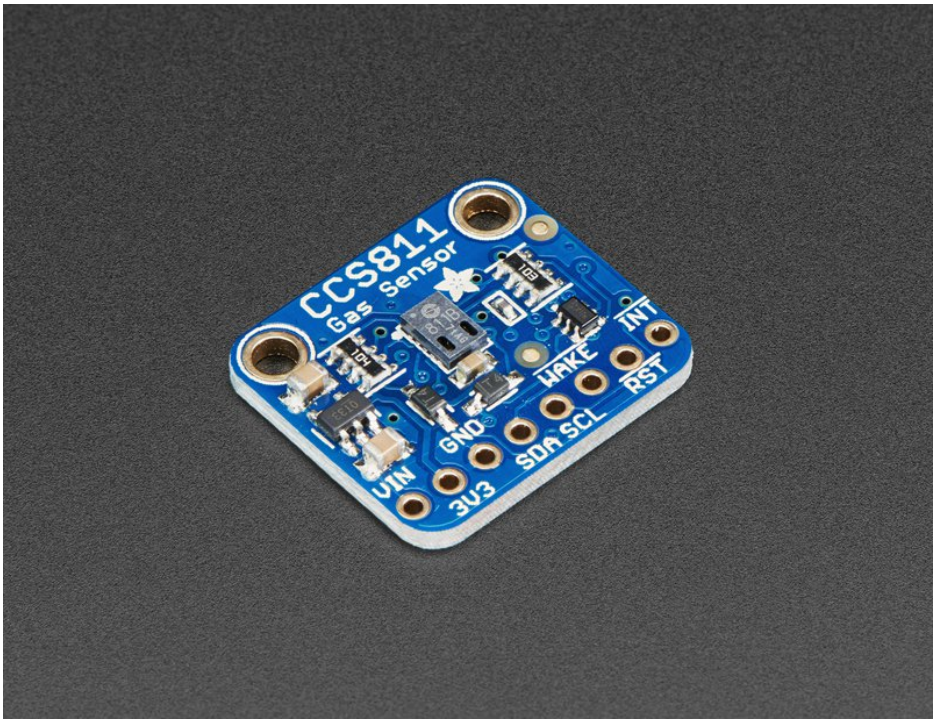
# Adafruit CCS811 Air Quality Sensor

Created by Dean Miller




Last updated on 2019-07-17 06:48:41 PM UTC

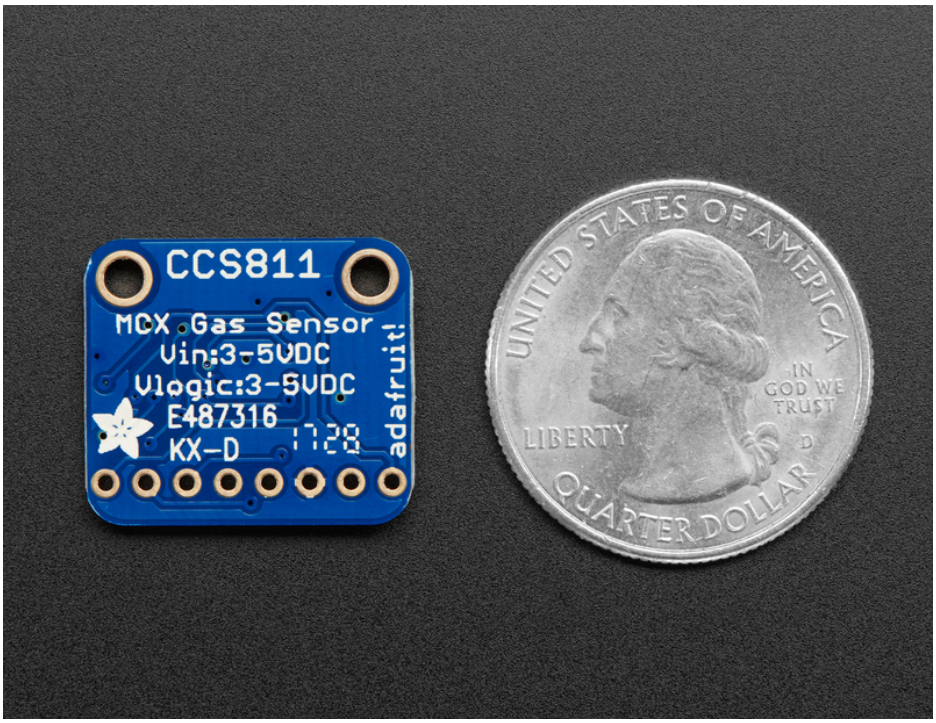
# Overview




Breathe easy - we finally have an I2C VOC/eCO2 sensor in the Adafruit shop! Add air quality monitoring to your project and with an **Adafruit CCS811 Air Quality Sensor Breakout**. This sensor from AMS is a gas sensor that can detect a wide range of Volatile Organic Compounds (VOCs) and is intended for indoor air quality monitoring. When connected to your microcontroller (running our library code) it will return a Total Volatile Organic Compound (TVOC) reading and an equivalent carbon dioxide reading (eCO2) over I2C.

This sensor is not well supported on Raspberry Pi. This is because it uses I2C clock stretching which the Pi cannot do without drastically slowing down the I2C speed. CircuitPython and Arduino are supported.

This chip USED to support a thermistor for temperature sensing, but it no longer does. Please use an external temperature sensor!

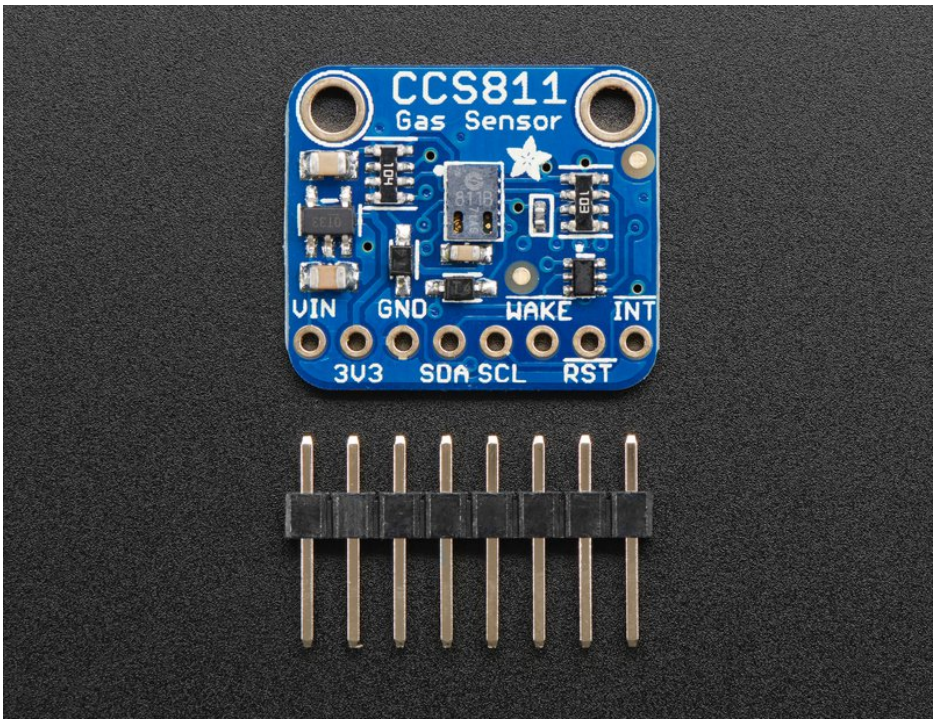



The CCS811 has a 'standard' hot-plate MOX sensor, as well as a small microcontroller that controls power to the plate, reads the analog voltage, and provides an I2C interface to read from.

This part will measure **eCO2** (equivalent calculated carbon-dioxide) concentration within a range of 400 to 8192 parts per million (ppm), and **TVOC** (Total Volatile Organic Compound) concentration within a range of 0 to 1187 parts per billion (ppb). According to the fact sheet it can detect Alcohols, Aldehydes, Ketones, Organic Acids, Amines, Aliphatic and Aromatic Hydrocarbons.

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 comparisons.

AMS recommends that you run this sensor for 48 hours when you first receive it to "burn it in", and then 20 minutes in the desired mode every time the sensor is in use. This is because the sensitivity levels of the sensor will change during early use.

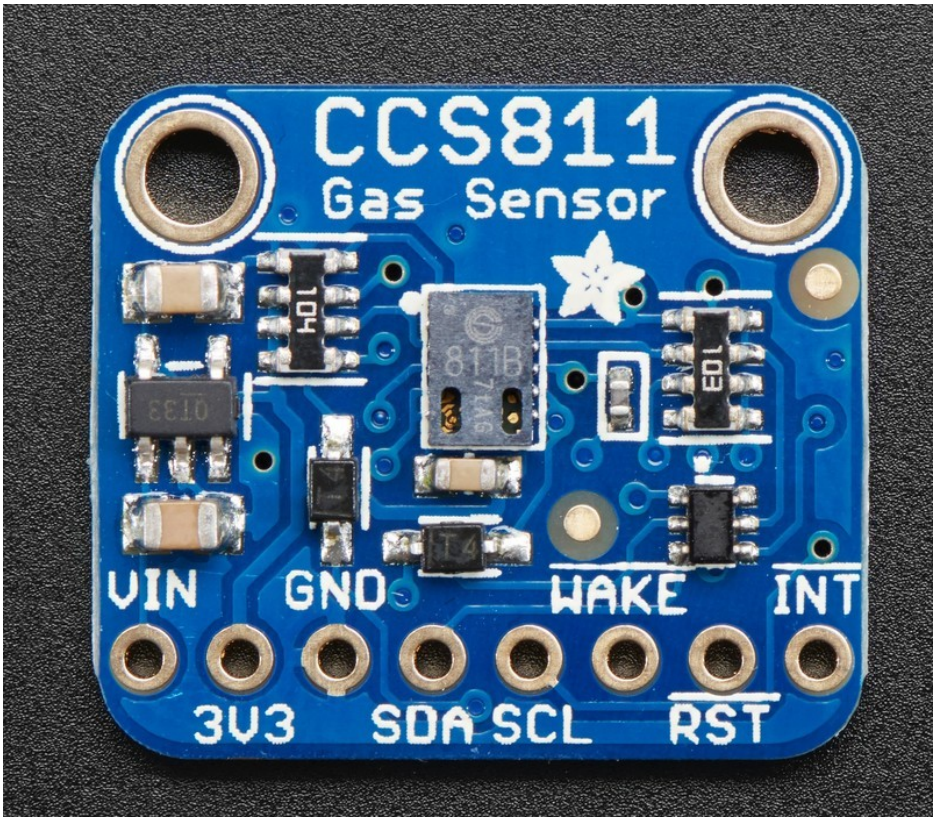



The CCS811 has a configurable interrupt pin that can fire when a conversion is ready and/or when a reading crosses a user-settable threshold. The CCS811 supports multiple drive modes to take a measurement every 1 second, every 10 seconds, every 60 seconds, or every 250 milliseconds.

For your convenience we've pick-and-placed the sensor on a PCB with a 3.3V regulator and some level shifting so it can be easily used with your favorite 3.3V or 5V microcontroller.

We've also prepared software libraries to get you up and running in Arduino or CircuitPython with just a few lines of code!

## Pinouts




This sensor has 2 mounting holes and one header breakout strip.

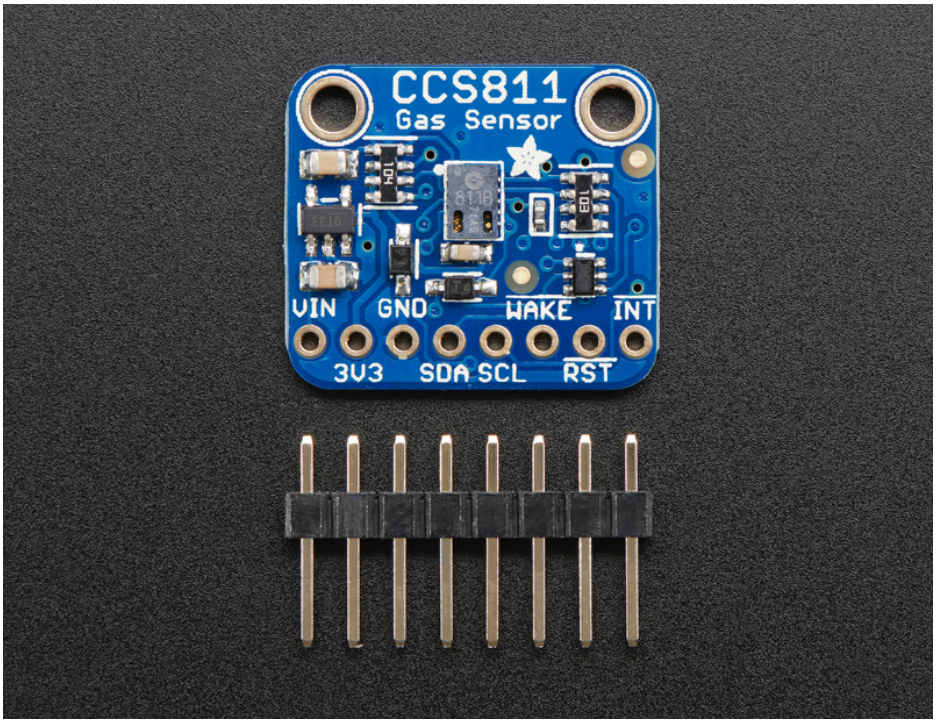
### Power Pins:

- **Vin** - this is the power pin. Since the sensor uses 3.3V, we have included an onboard voltage regulator that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **3Vo** - 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

### Logic pins:

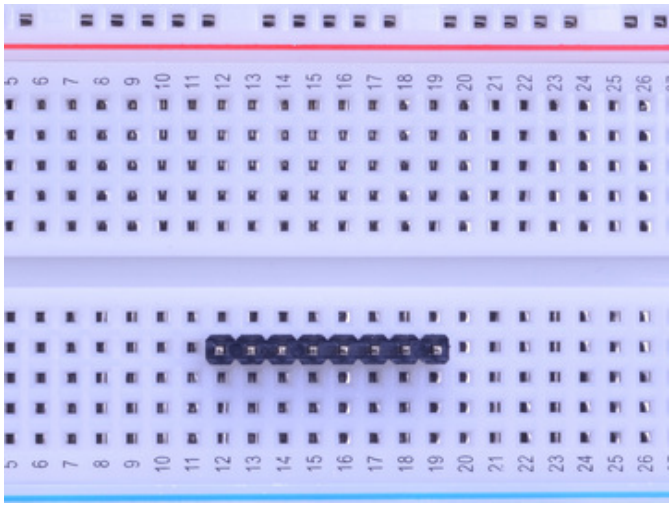
- **SCL** - this is the I2C clock pin, connect to your microcontrollers I2C clock line. There is a 10K pullup on this pin and it is level shifted so you can use 3 - 5VDC.
- **SDA** - this is the I2C data pin, connect to your microcontrollers I2C data line. There is a 10K pullup on this pin and it is level shifted so you can use 3 - 5VDC.
- **INT** - this is the interrupt-output pin. It is 3V logic and you can use it to detect when a new reading is ready or when a reading gets too high or too low.
- **WAKE** - this is the wakeup pin for the sensor. It needs to be pulled to ground in order to communicate with the sensor. This pin is level shifted so you can use 3-5VDC logic.
- **RST** - this is the reset pin. When it is pulled to ground the sensor resets itself. This pin is level shifted so you can use 3-5VDC logic.

## Assembly



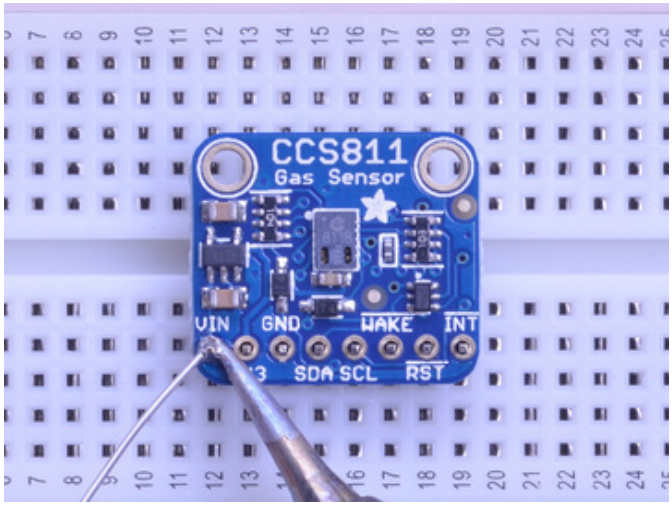

Before use, you'll need to attach headers so you can plug in this sensor into a breadboard. Soldering is essential!

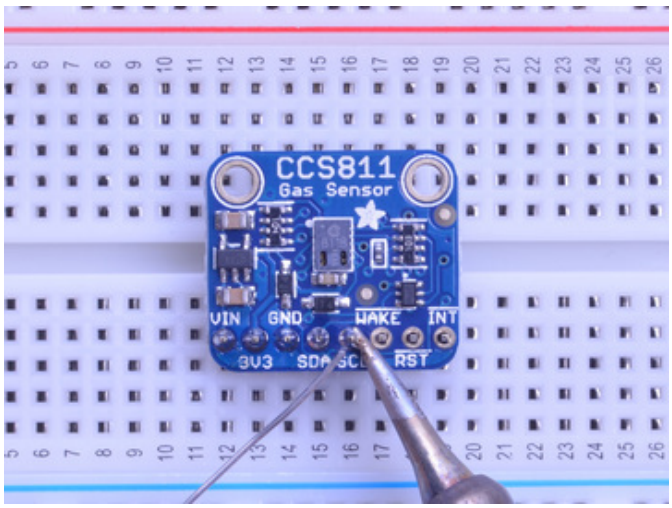
Note we show images of the BME280 sensor, which looks similar - the process is the same as it would be for the CCS811

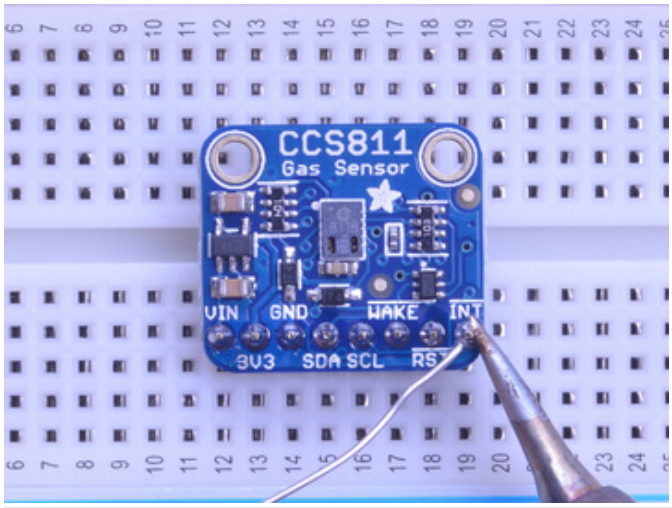


### Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



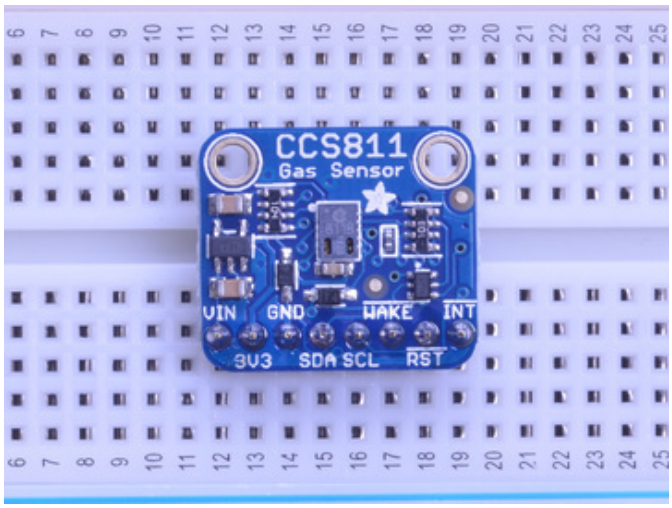




## Add the breakout board and Solder:

Place the breakout board over the pins so that the short pins poke through the breakout pads

Be sure to solder all pins for reliable electrical contact.

*(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk).*




You're done!

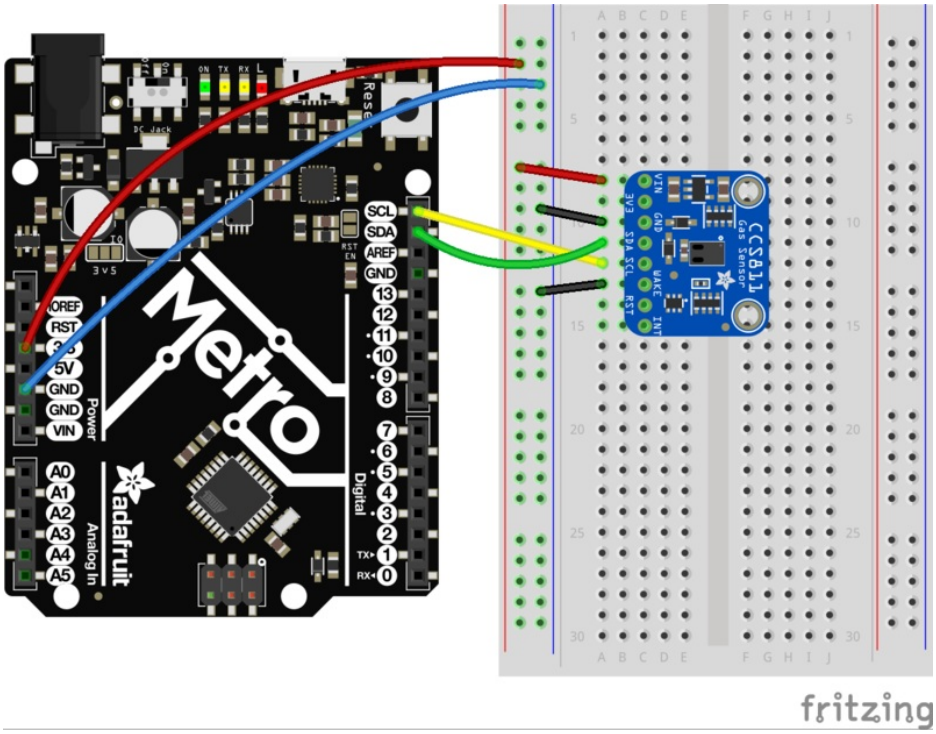
# Arduino Wiring & Test

You can easily wire this breakout to any microcontroller, we'll be using an Adafruit Metro (Arduino compatible) with the Arduino IDE. But, you can use any other kind of microcontroller as well as long as it has I2C clock and I2C data lines. Note this chip uses clock stretching so make sure your microcontroller supports that in hardware!

## I2C Wiring

- Connect **Vin** to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino.
  On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino.
  On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**
- Connect the **WAKE** pin to ground.

This sensor uses I2C address **0x5A**.




Don't forget to tie WAKE to Ground - this is required!

## Download Adafruit_CCS811 library

To begin reading sensor data, you will need to download Adafruit_CCS811 from our github repository. You can do that

by visiting the github repo and manually downloading or, easier, just click this button to download the zip

https://adafru.it/ycs

https://adafru.it/ycs

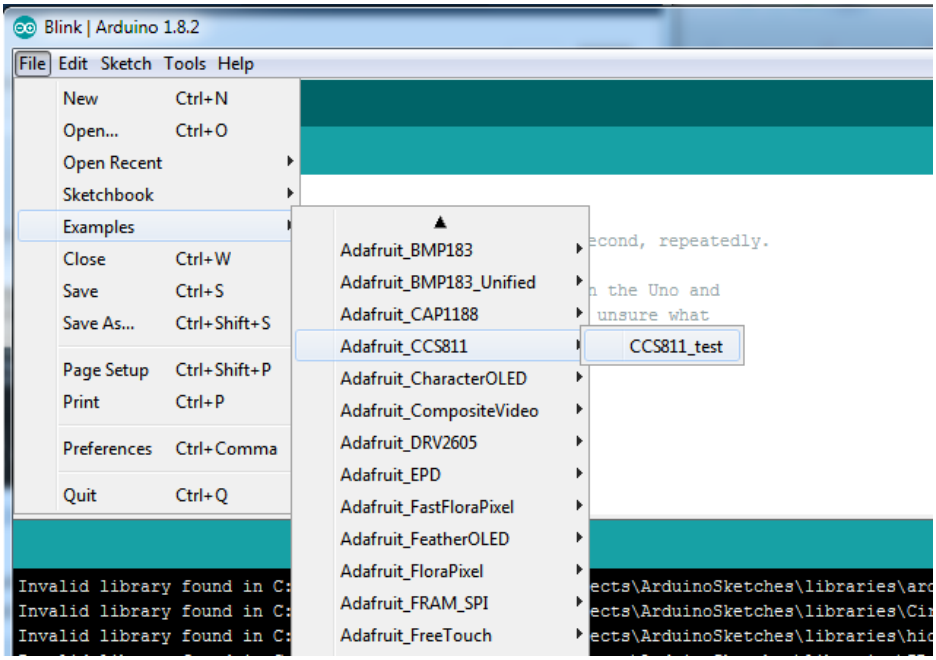
Rename the uncompressed folder **Adafruit_CCS811** and check that the **Adafruit_CCS811** folder contains **Adafruit_CCS811 .cpp** and **Adafruit_CCS811 .h**

Place the **Adafruit_CCS811** library folder your **arduinosketchfolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

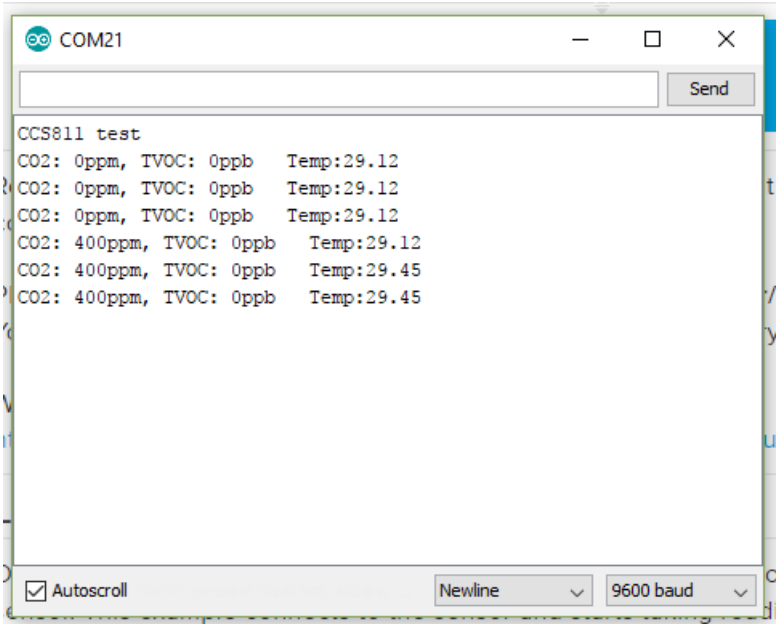
We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

## Load Test Example

Open up **File->Examples->Adafruit_CCS811->CCS811_test** and upload to your Arduino wired up to the sensor. This example connects to the sensor and starts taking readings.




Once uploaded to your Arduino, open up the serial console at 9600 baud speed to see the readings. Your sensor will take 3 zero readings while it does some internal calibration and correction things and then start outputting real data. In clean air and a typical indoor space your serial monitor will look something like this:




AMS recommends that you run this sensor for 48 hours when you first receive it to "burn it in", and then 20 minutes in the desired mode every time the sensor is in use. This is because the sensitivity levels of the sensor will change during early use.

## Library Reference

To create the object, use

```
Adafruit_CCS811 ccs;
```

initialize the sensor with:

```
if(!ccs.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    while(1);
  }
```

To poll the sensor for available data you can use:

```
bool dataAvailable = ccs.available(); //returns true if data is available to be read
```

Data can be read using:

```
bool error = ccs.readData(); //returns True if an error occurs during the read
```

and then the readings can be accessed with:

```
int eCO2 = ccs.geteCO2(); //returns eCO2 reading
int TVOC = ccs.getTVOC(); //return TVOC reading
```

Approximate ambient temperature can be read using:

```
float temp = ccs.calculateTemperature();
```

# Arduino Library Docs

Arduino Library Docs (https://adafru.it/Au8)

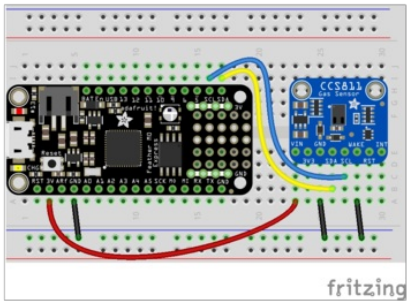
# Python & CircuitPython

It's easy to use the CCS811 sensor with Python or CircuitPython, and the Adafruit CircuitPython CCS811 (https://adafru.it/yaX) module. This module allows you to easily write Python code that reads the eCO2, TVOC and temperature from the sensor.

You can use this 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 a CCS811 to your board exactly as shown on the previous pages for Arduino. Here's an example of wiring a Feather M0 to the sensor with I2C:

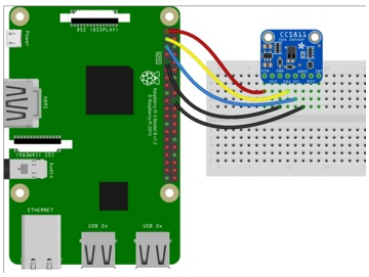



- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**
- **Board GND** to **sensor WAKE**

## Python Computer Wiring

Since there's *dozens* of Linux computers/boards you can use we will show 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 with I2C:



- **Pi 3V3** to **sensor VIN**
- **Pi GND** to **sensor GND**
- **Pi SCL** to **sensor SCL**
- **Pi SDA** to **sensor SDA**
- **Pi GND** to **sensor WAKE**

## CircuitPython Installation of CCS811 Library

You'll need to install the Adafruit CircuitPython CCS811 (https://adafru.it/yaX) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/uap). Our CircuitPython starter guide has a great

page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_ccs811.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_ccs811.mpy,** and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

## Python Installation of CCS811 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:

- `sudo pip3 install adafruit-circuitpython-ccs811`

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

## CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the eCO2, TVOC, and temperature from the board's Python REPL.

If you're using an I2C connection run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import time
import board
import busio
import adafruit_ccs811
i2c_bus = busio.I2C(board.SCL, board.SDA)
ccs811 = adafruit_ccs811.CCS811(i2c_bus)
```

Then you'll want to calibrate the thermistor:

Now you're ready to read values from the sensor using any of these properties:

- **eco2** - Equivalent Carbon Dioxide in parts per million. Clipped to 400 to 8192ppm.
- **tvoc** - Total Volatile Organic Compound in parts per billion.
- **temperature** - Temperature based on optional thermistor in Celsius.

For example to print eCO2, TVOC, and temperature:

```
print("CO2: %1.0f PPM" % ccs811.eco2)
print("TVOC: %1.0f PPM" % ccs811.tvoc)
print("Temp: %0.1f C" % ccs811.temperature)
```

```
>>> print("CO2: %1.0f PPM" % ccs811.eco2)
CO2: 446 PPM
>>> print("TVOC: %1.0f PPM" % ccs811.tvoc)
TVOC: 7 PPM
>>> print("Temp: %0.1f C" % ccs811.temperature)
Temp: 25.4 C
```

That's all there is to using CCS811 with Python and CircuitPython!

## Full Example Code

```
import time
import board
import busio
import adafruit_ccs811

i2c = busio.I2C(board.SCL, board.SDA)
ccs811 = adafruit_ccs811.CCS811(i2c)

# Wait for the sensor to be ready
while not ccs811.data_ready:
    pass

while True:
    print("CO2: {} PPM, TVOC: {} PPM"
          .format(ccs811.eco2, ccs811.tvoc)
    time.sleep(0.5)
```

# Python Docs

Python Docs (https://adafru.it/C46)

# Raspberry Pi Wiring & Test

The Raspberry Pi also has an I2C interface that can be used to communicate with this sensor.

## Install Python Software

Once your Pi is all set up, and you have internet access set up, lets install the software we will need. First make sure your Pi package manager is up to date

```
sudo apt-get update
```

Next, we will install the Raspberry Pi library and Adafruit_GPIO which is our hardware interfacing layer

```
sudo apt-get install -y build-essential python-pip python-dev python-smbus git
git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
cd Adafruit_Python_GPIO
sudo python setup.py install
```

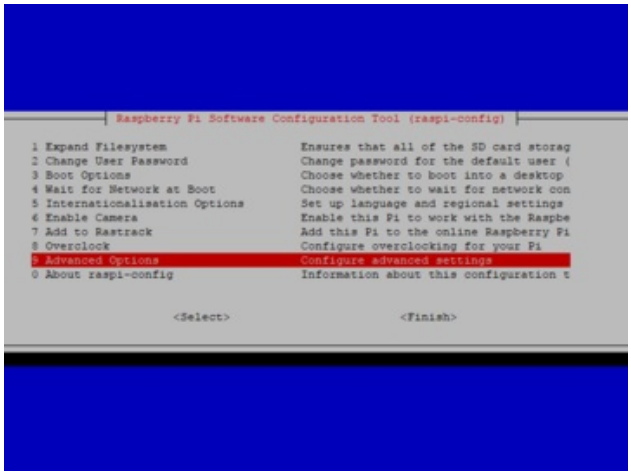
Next install the adafruit CCS811 python library.

```
sudo pip install Adafruit_CCS811
```

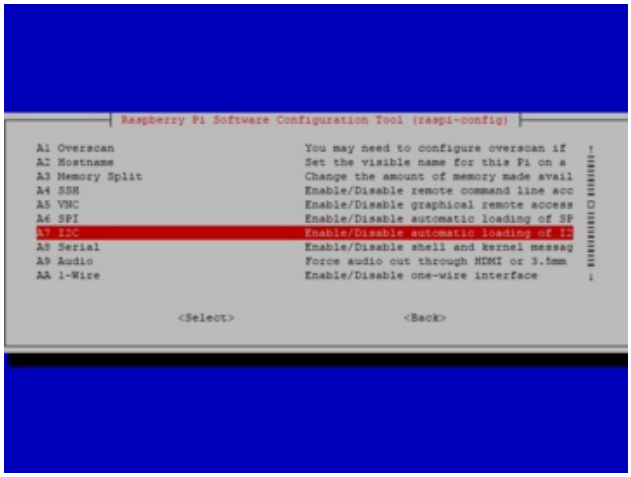
## Enable I2C

We need to enable the I2C bus so we can communicate with the sensor.

```
sudo raspi-config
```




select Advanced options, enable I2C, and then finish.






Once I2C is enabled, we need to slow the speed way down due to constraints of this particular sensor.

```
sudo nano /boot/config.txt
```

add this line to the file

```
dtparam=i2c_baudrate=10000
```

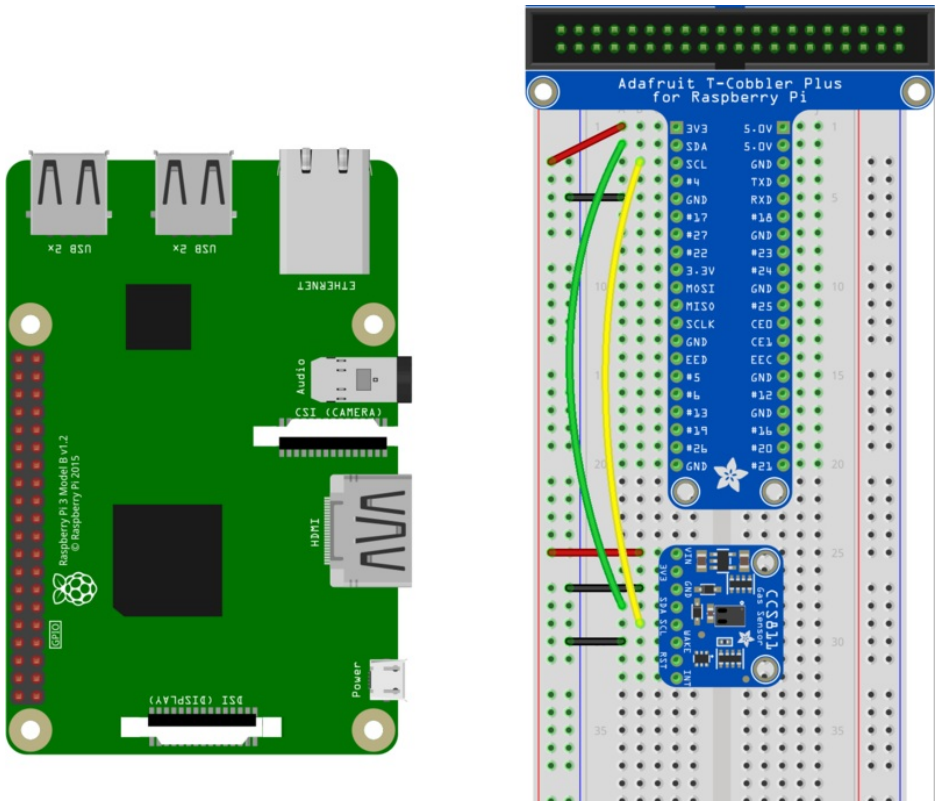
press **Ctrl+X**, then **Y**, then **enter** to save and exit. Then run **sudo shutdown -h now** to turn off the Pi and prepare for wiring.

## Wiring Up Sensor

With the Pi powered off, we can wire up the sensor to the Pi Cobbler like this:

- Connect **Vin** to the 3V or 5V power supply (either is fine)
- Connect **GND** to the ground pin on the Cobbler
- Connect **SDA** to **SDA** on the Cobbler
- Connect **SCL** to **SCL** on the Cobbler
- Connect **Wake** to the ground pin on the Cobbler

You can also use direct wires, we happen to have a Cobbler ready. remember you can plug the cobbler into the bottom of the PiTFT to get access to all the pins!




Now you should be able to verify that the sensor is wired up correctly by asking the Pi to detect what addresses it can see on the I2C bus:

```
sudo i2cdetect -y 1
```

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- 5a -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

It should show up under it's default address (**0x5A**). If you don't see 5A, check your wiring, did you install I2C support, etc?

## Run example code

At long last, we are finally ready to run our example code

```
cd ~/
git clone https://github.com/adafruit/Adafruit_CCS811_python.git
cd Adafruit_CCS811_python/examples
sudo python CCS811_example.py
```

If everything is set up correctly, you should see it print out a few 0 readings, and then every few seconds it will print out another reading

```
pi@raspberrypi:~/Adafruit_CCS811_python/examples $ sudo python CCS811_example.py
CO2:  409 ppm, TVOC:  1  temp:  25.0
CO2:  0 ppm, TVOC:  0  temp:  25.0
CO2:  0 ppm, TVOC:  0  temp:  25.0
CO2:  400 ppm, TVOC:  0  temp:  25.0
CO2:  400 ppm, TVOC:  0  temp:  25.0
CO2:  400 ppm, TVOC:  0  temp:  25.0
CO2:  407 ppm, TVOC:  1  temp:  25.0
CO2:  407 ppm, TVOC:  1  temp:  25.0
```

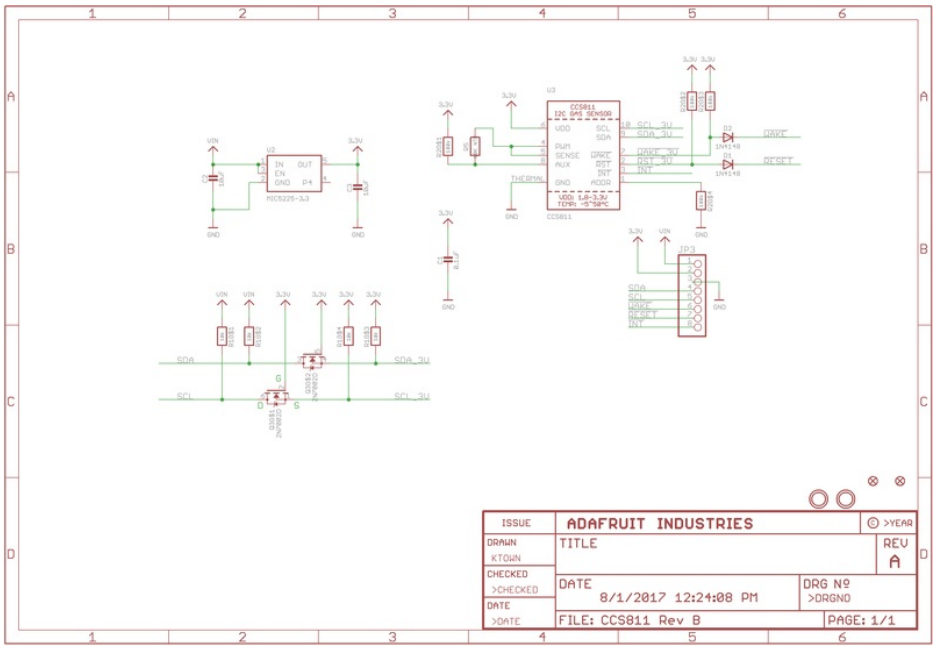
# Downloads

## Documents

- CCS811 Datasheet (https://adafru.it/yaV)
- CCS811 Fact sheet (https://adafru.it/ycv)
- CCS811 Mechanical Considerations App Note (https://adafru.it/ycw)
- CCS811 NTC Thermistor App Note (https://adafru.it/ycx)
- CCS811 Baseline Clear/Restore App Note (https://adafru.it/ycy)
- Adafruit CCS811 Arduino Driver (https://adafru.it/yaW)
- CCS811 CircuitPython Driver (https://adafru.it/yaX)
- Fritzing object in the Adafruit Fritzing library (https://adafru.it/aP3)
- CCS811 breakout PCB files (EAGLE format) (https://adafru.it/yaY)

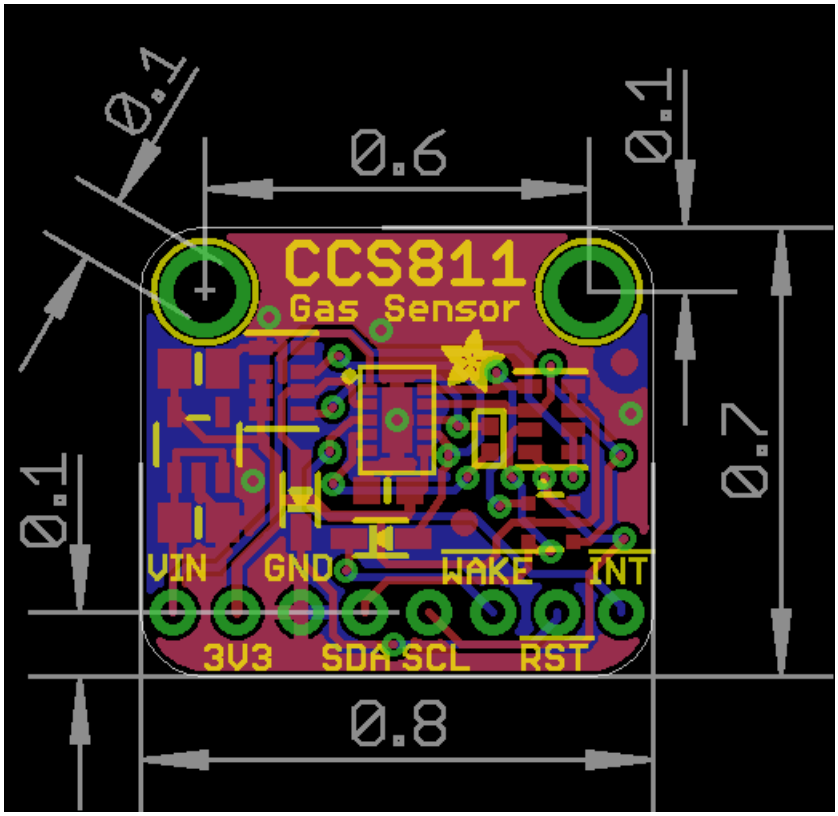
## Schematic

click to enlarge




## Dimensions

in inches. Click to enlarge


0.1
0.6
0.1
CCS811
Gas Sensor
0.7
0.1
VIN
GND
WAKE
INT
3V3
SDA
SCL
RST
0.8