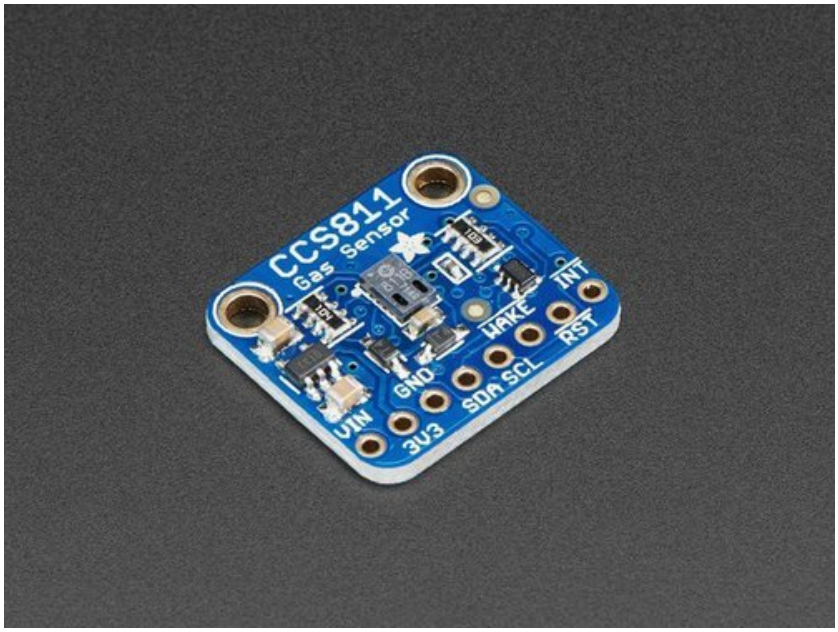
adafruit learning system

Adafruit CCS811 Air Quality Sensor

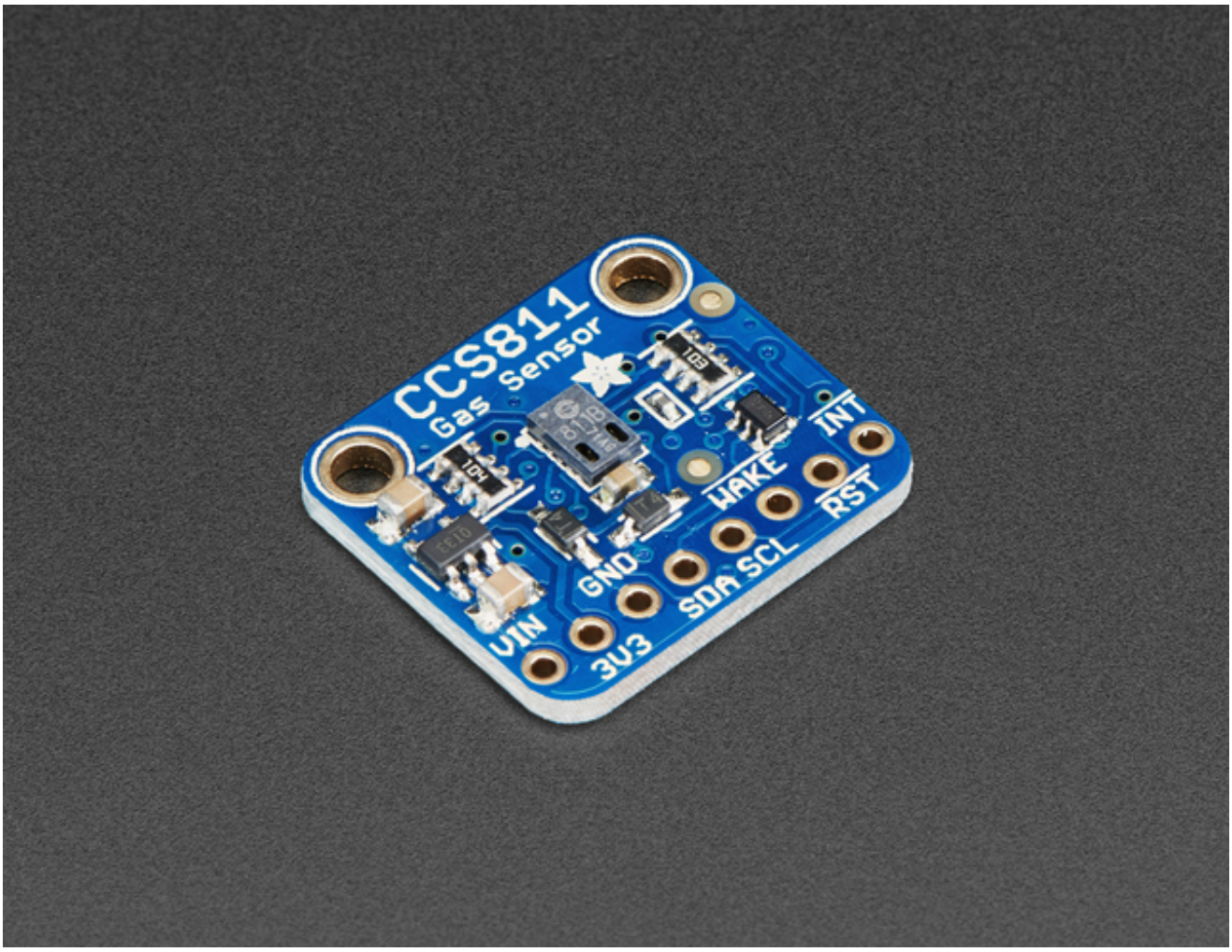
Created by Dean Miller



Last updated on 2017-08-21 07:15:50 PM UTC

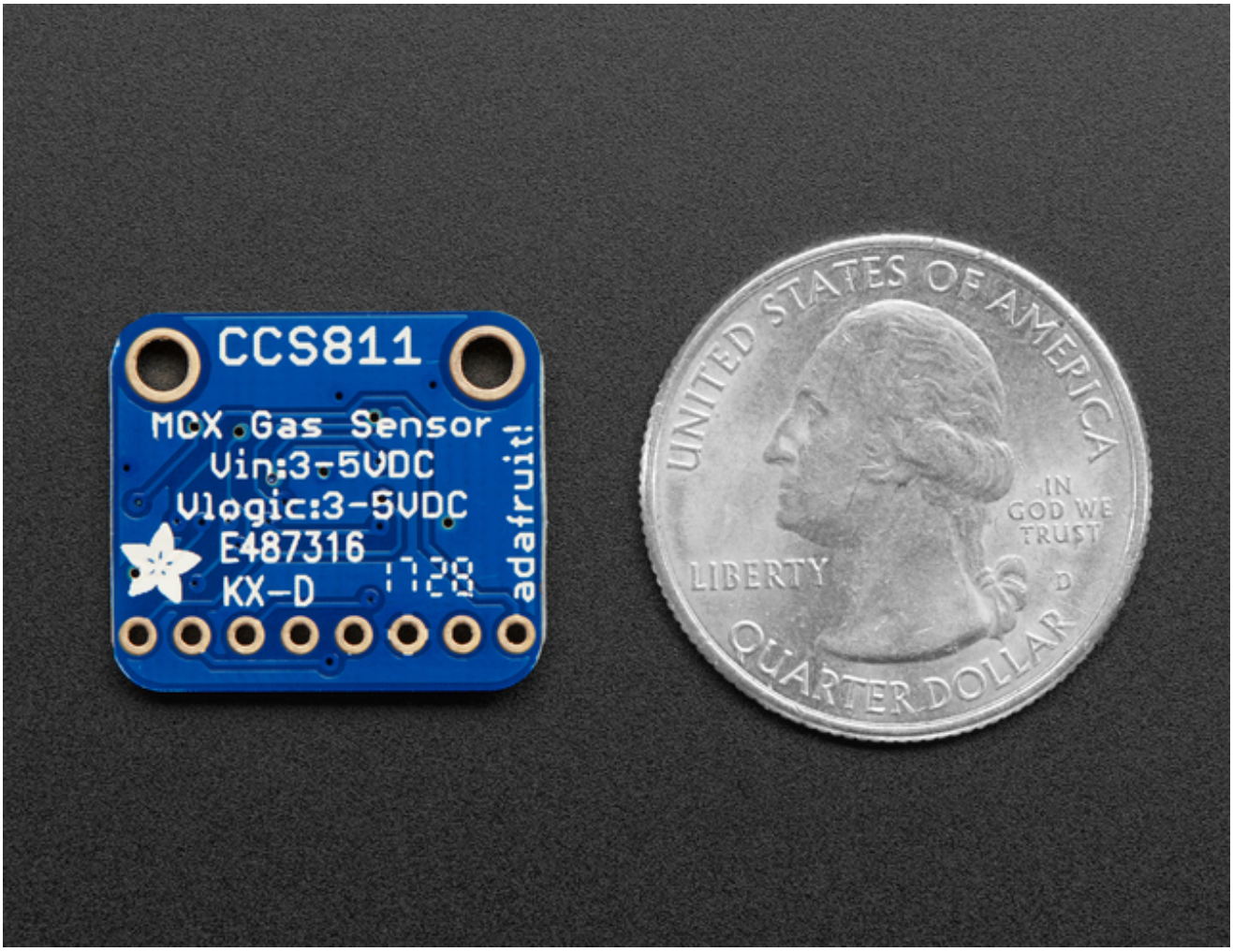
Guide Contents

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. There is also an on-board thermistor that can be used to calculate the approximate local ambient temperature.

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.

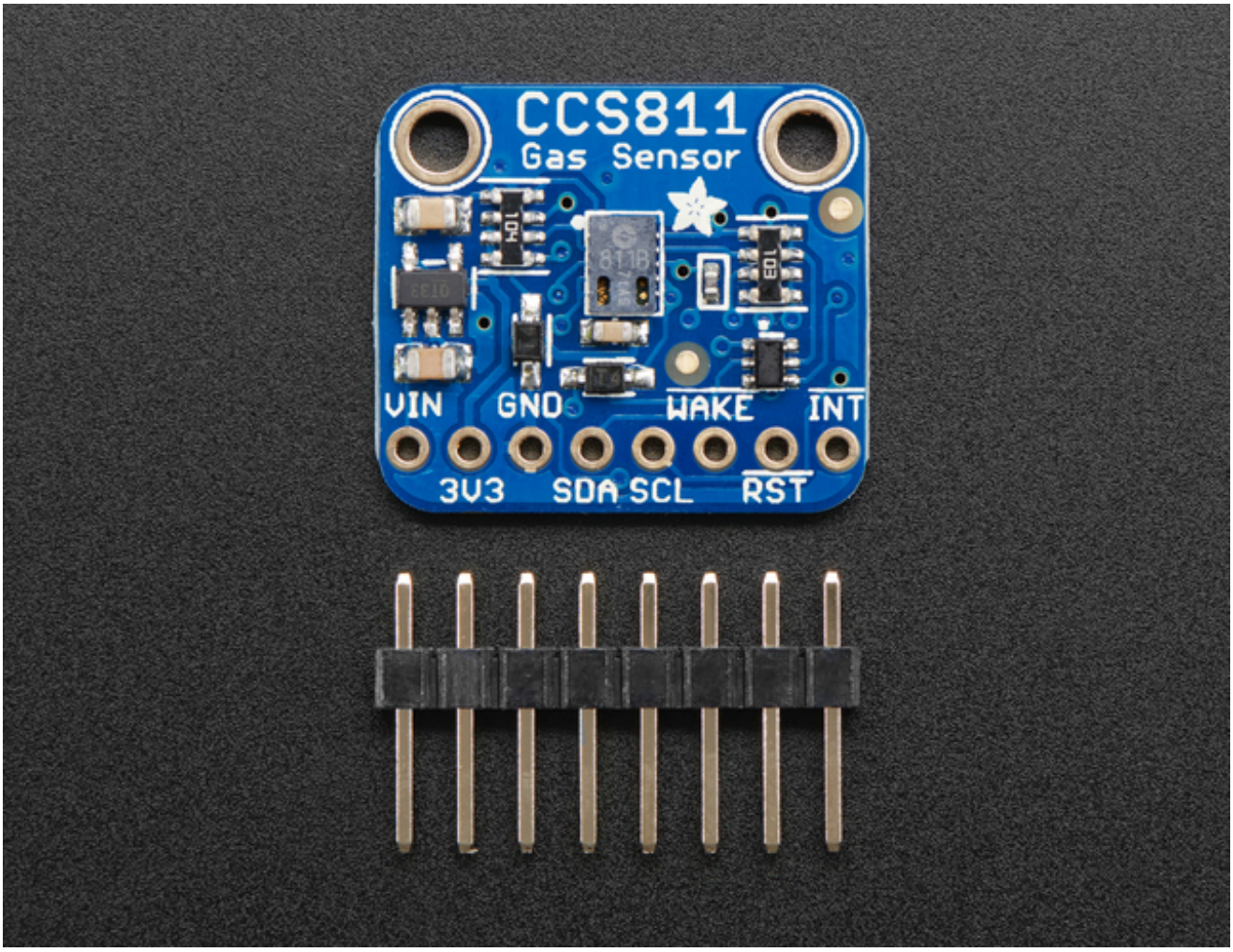


The CCS811 has a 'standard' hot-plate MOX sensor, as well as a small microcontroller that controls power to the plate, reads the analog voltage, and provides an I2C interface to read from.

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. We include a 10K NTC thermistor with matching balancing resistor which can be read by the CCS811 to calculate temperature

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.

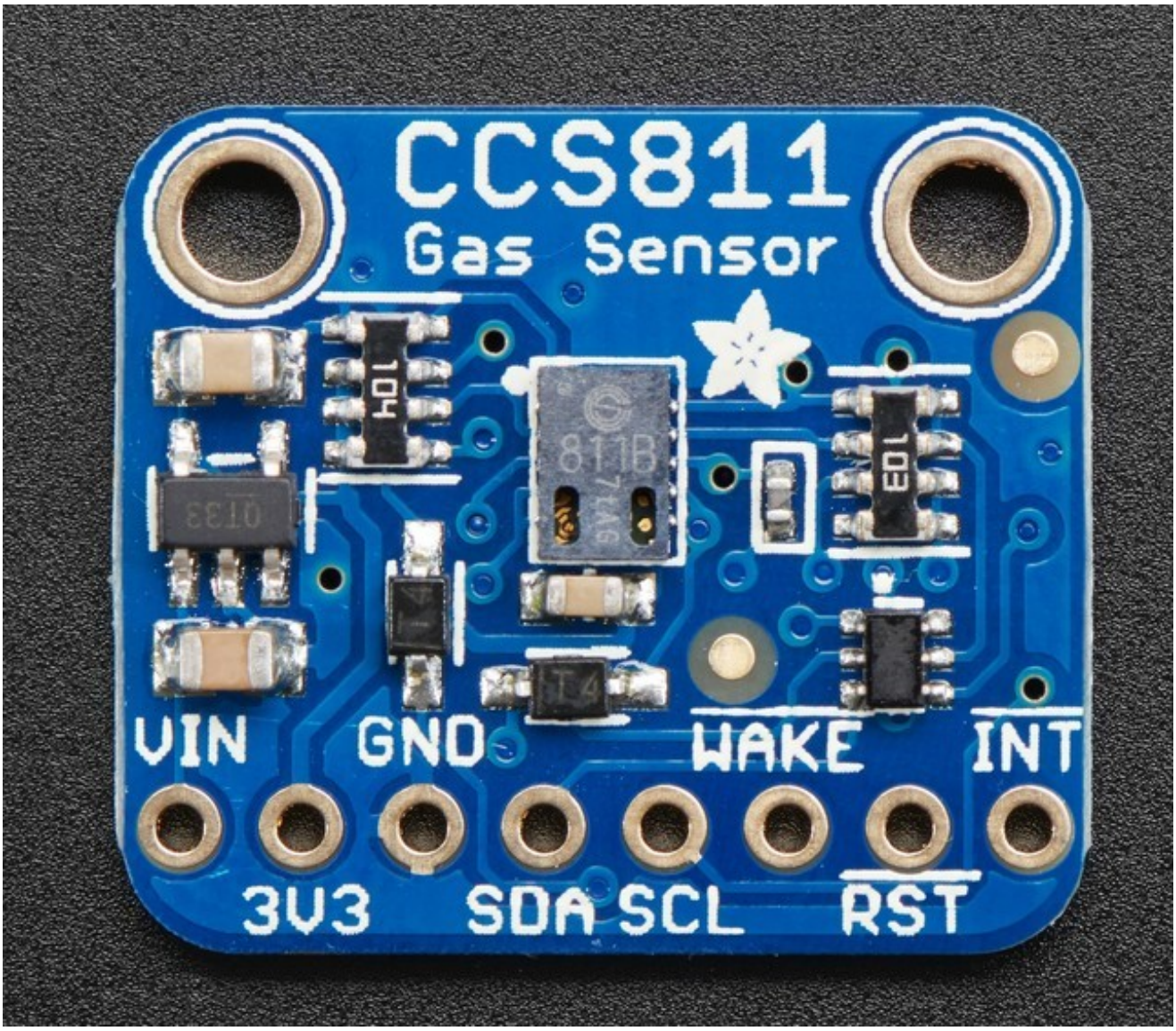


The CCS811 has a configurable interrupt pin that can fire when a conversion is ready and/or when a reading crosses a user-settable threshold. The CCS811 supports multiple drive modes to take a measurement every 1 second, every 10 seconds, every 60 seconds, or every 250 milliseconds.

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.

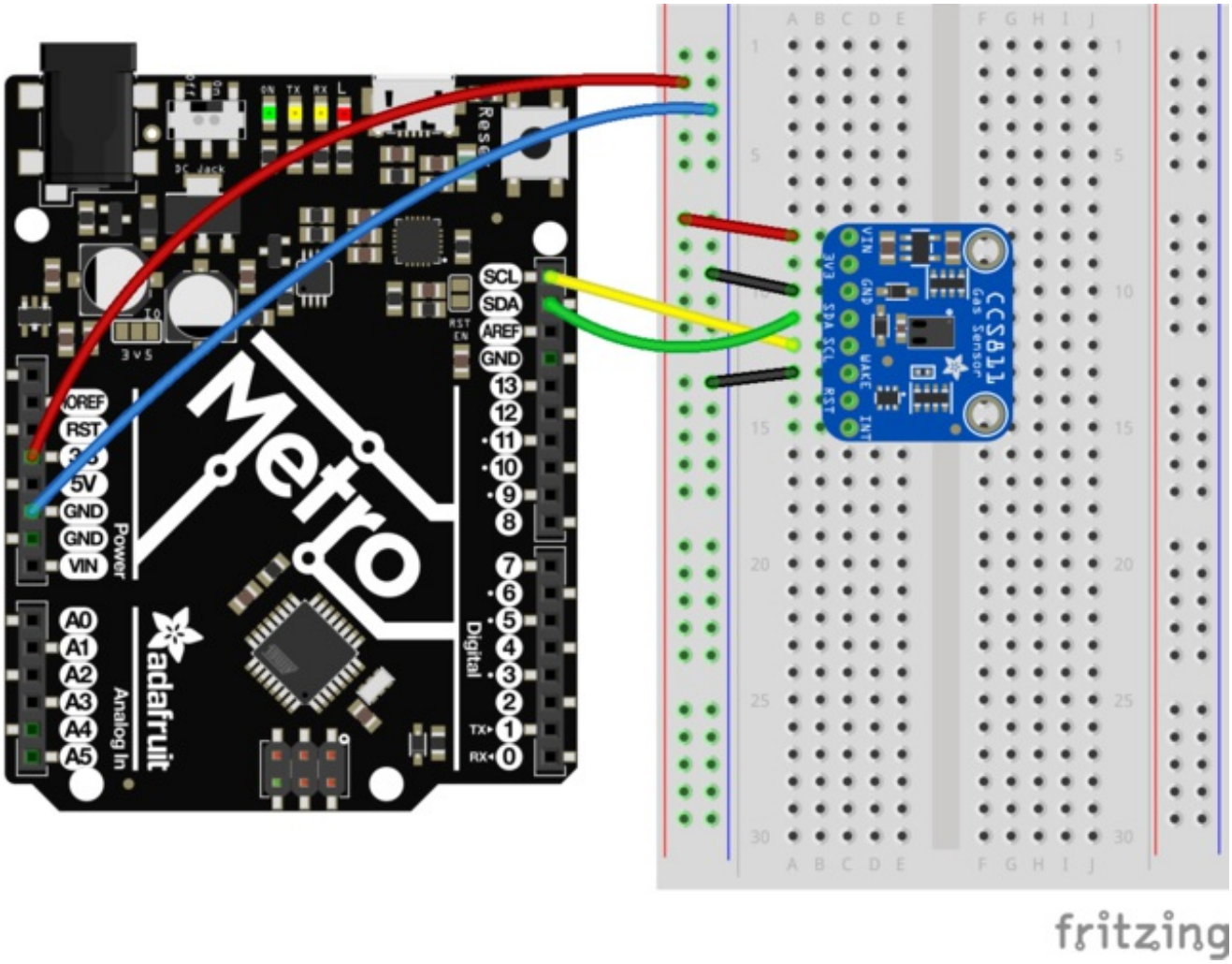
Arduino Wiring & Test

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

[Download Adafruit CCS811 Library]
http://adafru.it/ycs

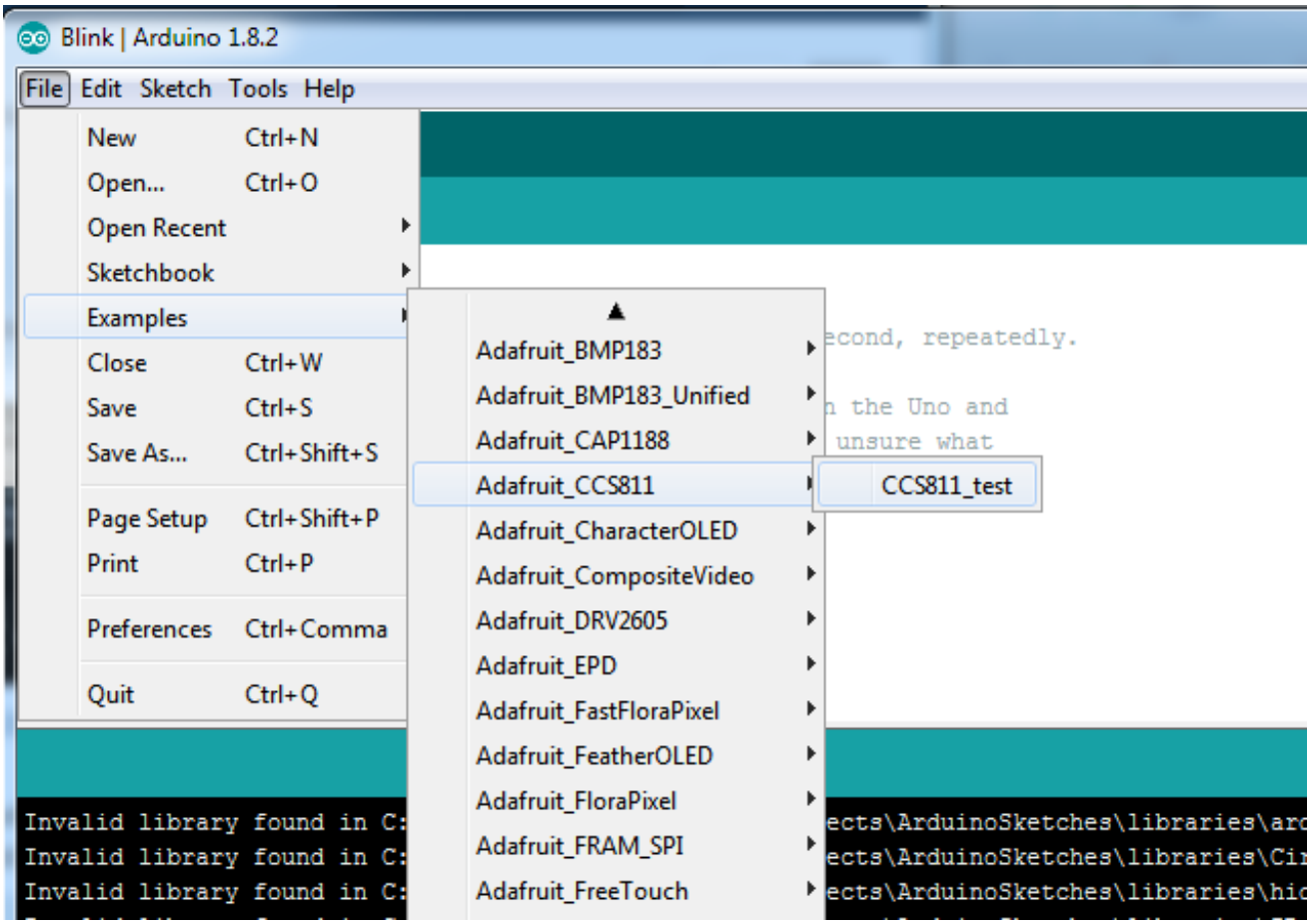
Rename the uncompressed folder **Adafruit_CCS811** and check that the **Adafruit_CCS811** folder contains **Adafruit_CCS811 .cpp** and **Adafruit_CCS811 .h**

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.

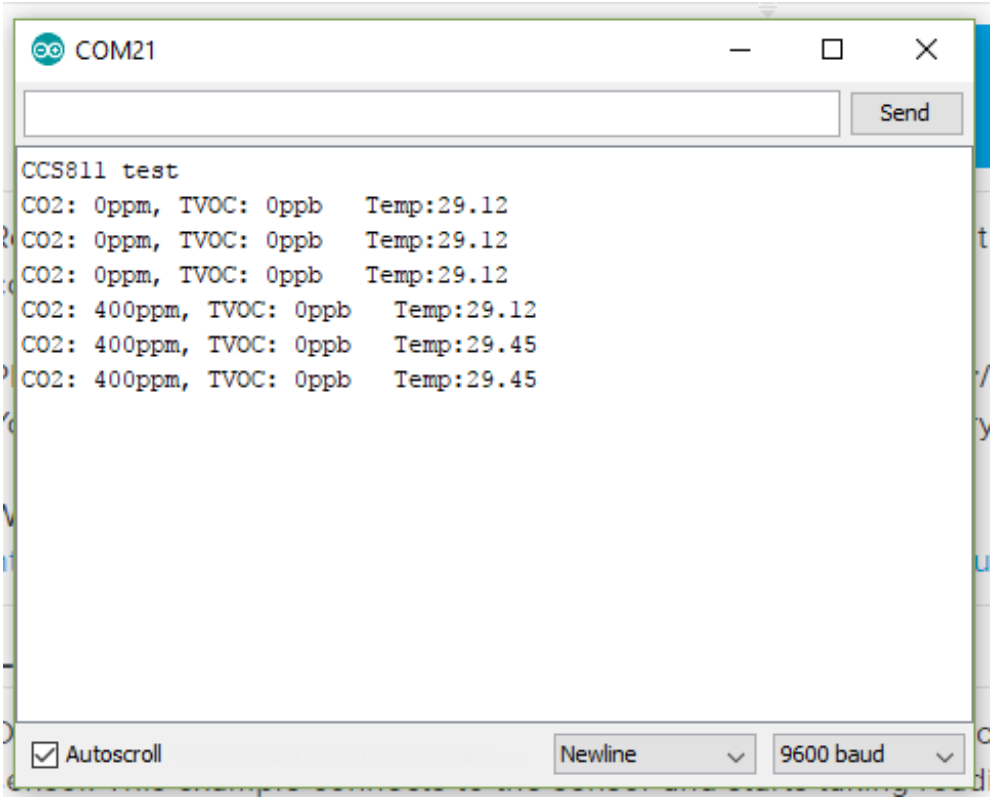
We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (http://adafru.it/aYM)

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();
```

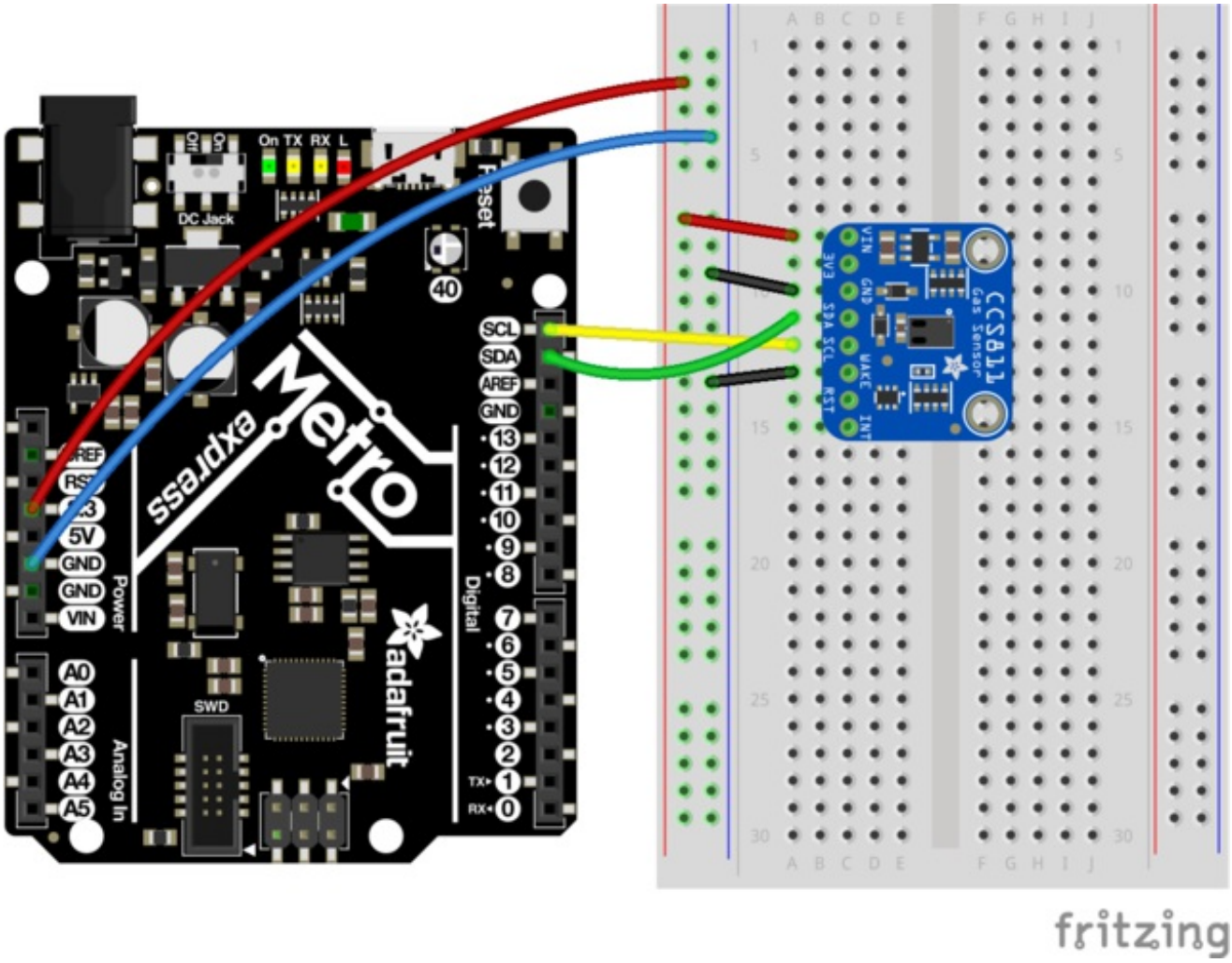
CircuitPython Wiring & Test

You can easily wire this breakout to a microcontroller running CircuitPython. We will be using a Metro M0 Express.

I2C Wiring

- Connect **Vin** to the power supply, 3-5V is fine.
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Feather or Metro M0. On a Gemma M0 this would be **Pad #2**/ **A1**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Feather or Metro M0. On an Gemma M0 this would be **Pad #0**/**A2**
- Connect the **WAKE** pin to ground.

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

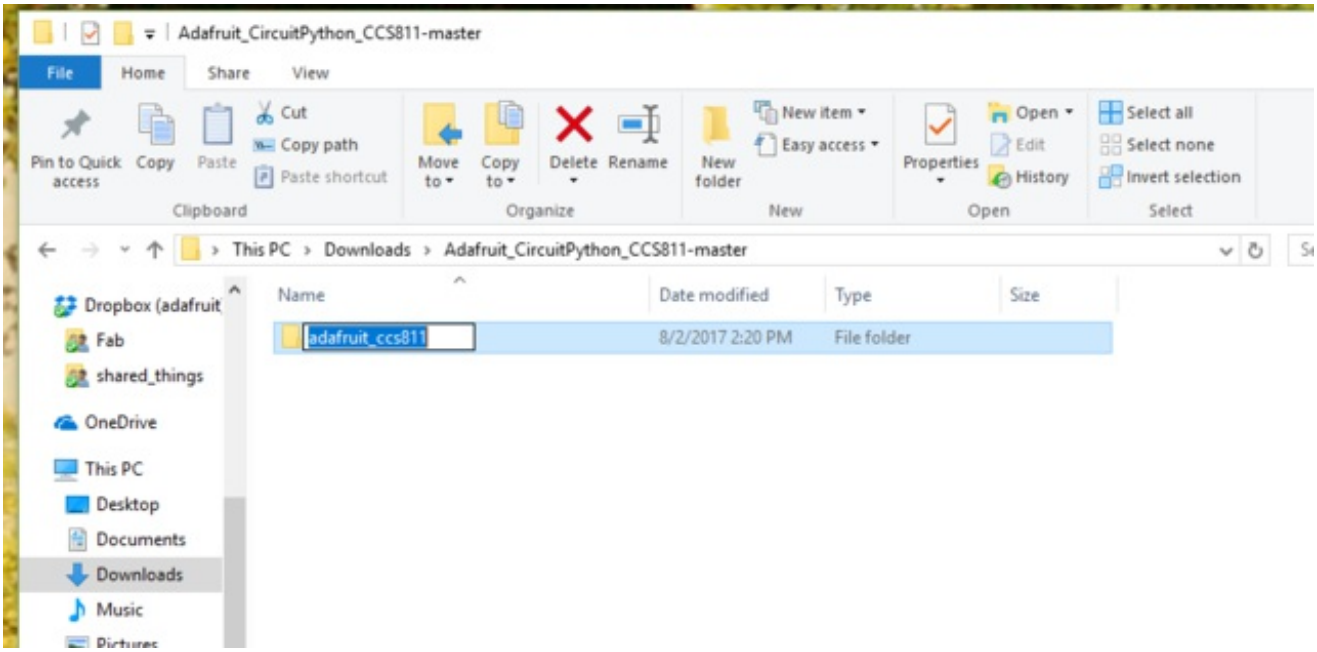


Download Adafruit_CircuitPython_CCS811 library

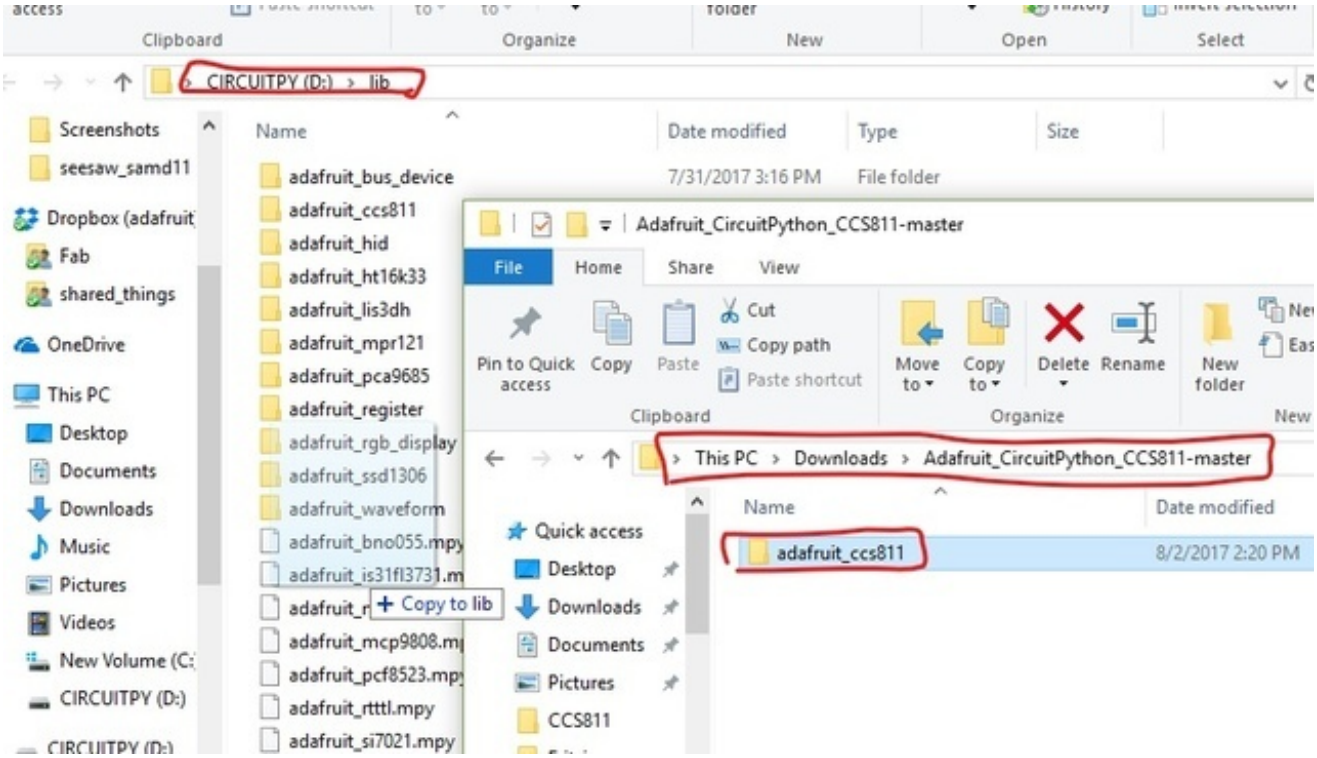
To begin reading sensor data, you will need to download Adafruit_CircuitPython_CCS811 from our github repository. You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

[Adafruit CircuitPython CCS811 Library]
http://adafru.it/yct

Extract the zipped folder and rename the **folder it contains** called Adafruit_CircuitPython_CCS811-master to **adafruit_ccs811**



drag the **adafruit_ccs811** folder to the **lib** folder that appears on the **CIRCUITPY** drive



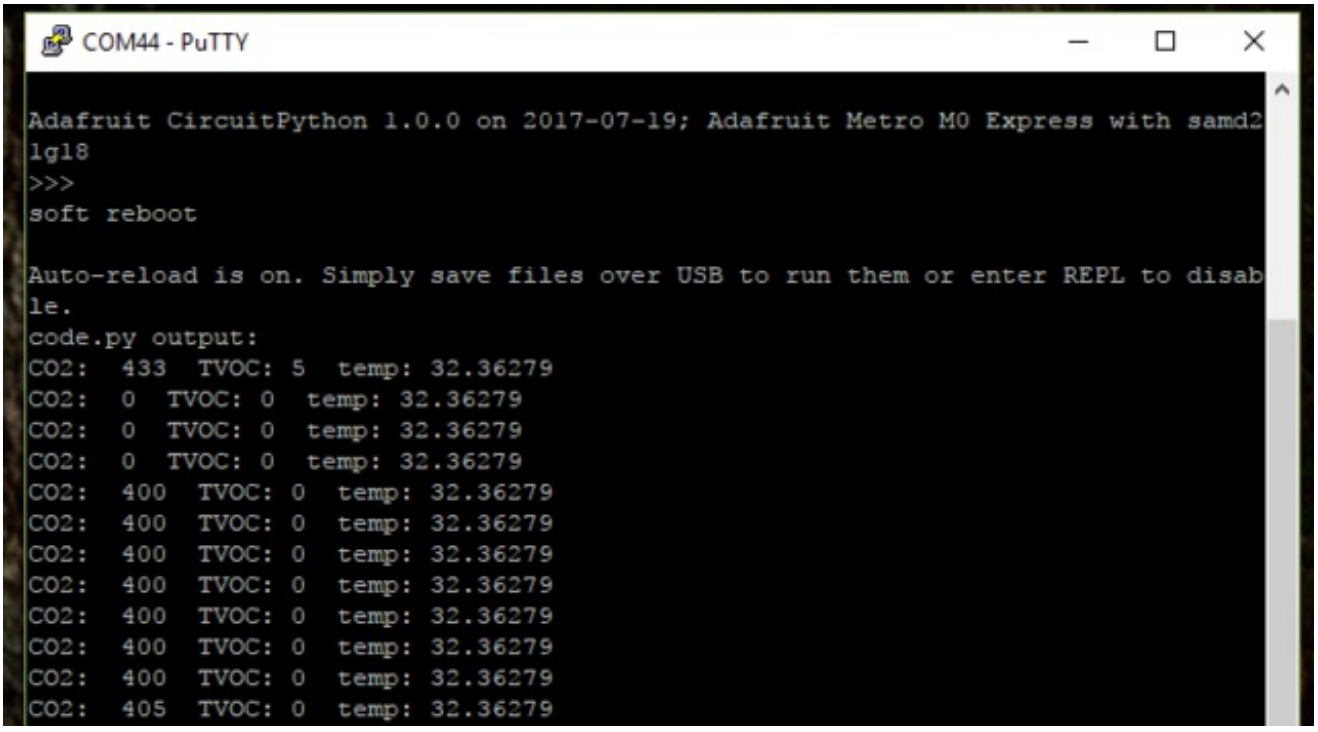
make sure the new folder you just created at **CIRCUITPY/lib/adafruit_ccs811** contains the **Adafruit_CCS811.py** file.

Open the **code.py** file on the **CIRCUITPY** drive and copy and paste the following code:

```
from board import *
import time
```

```
import busio

from Adafruit_CCS811 import Adafruit_CCS811

myI2C = busio.I2C(SCL, SDA)

ccs =  Adafruit_CCS811.Adafruit_CCS811(myI2C)

#wait for the sensor to be ready and calibrate the thermistor
while not ccs.data_ready:
 pass
temp = ccs.calculateTemperature()
ccs.tempOffset = temp - 25.0


while(1):
 if ccs.data_ready:
  temp = ccs.calculateTemperature()
  if not ccs.readData():
   print("CO2: ", ccs.eCO2, " TVOC:", ccs.TVOC, " temp:", temp)
  else:
   print("ERROR!")
   while(1):
    pass
 time.sleep(.5)
```

Connect to your CircuitPython device using your serial port terminal software (see here (http://adafru.it/ycu) if you are unsure how to do this) and if everything is correct you should see output that looks like this



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
```

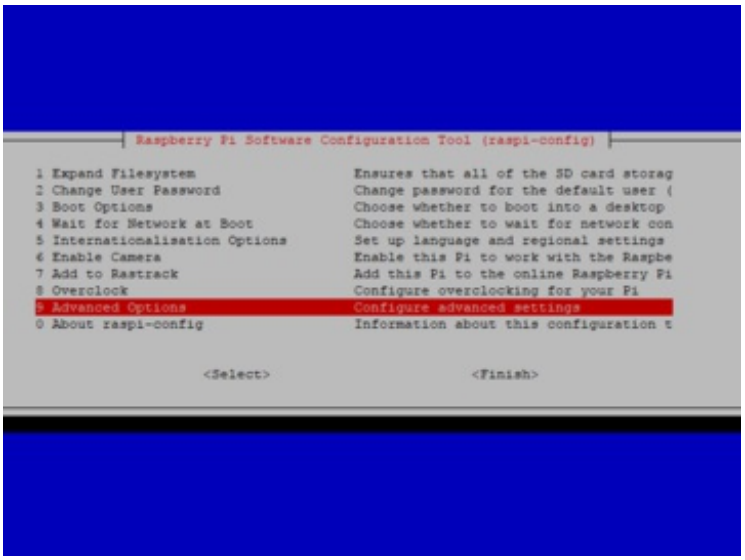
Next install the adafruit CCS811 python library.

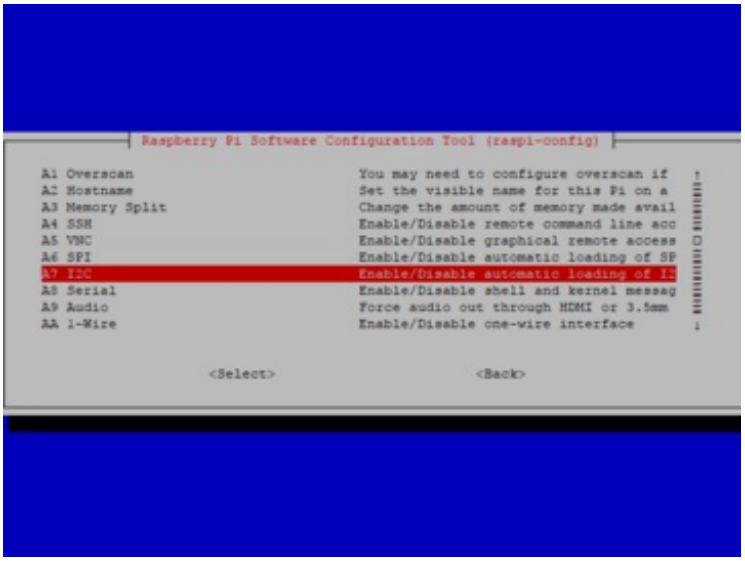
```
sudo pip install Adafruit_CCS811
```

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

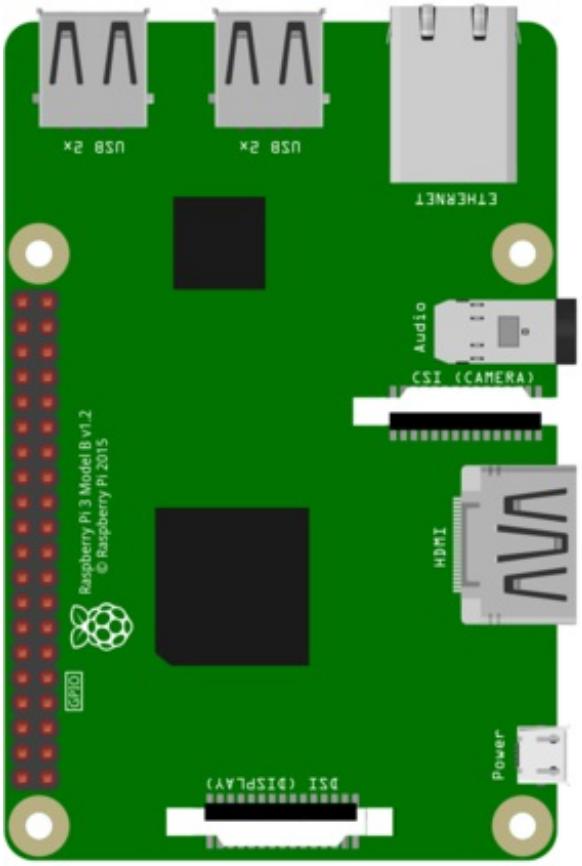
press **Ctrl+X**, then **Y**, then **enter** to save and exit. Then run **sudo shutdown -h now** to turn off the Pi and prepare for wiring.

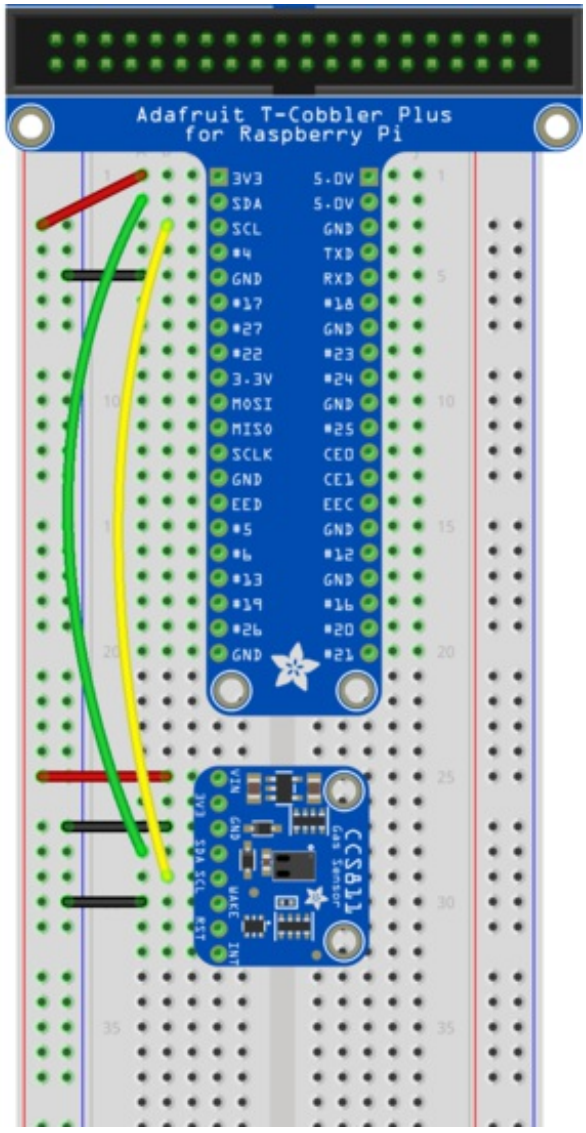
Wiring Up Sensor

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

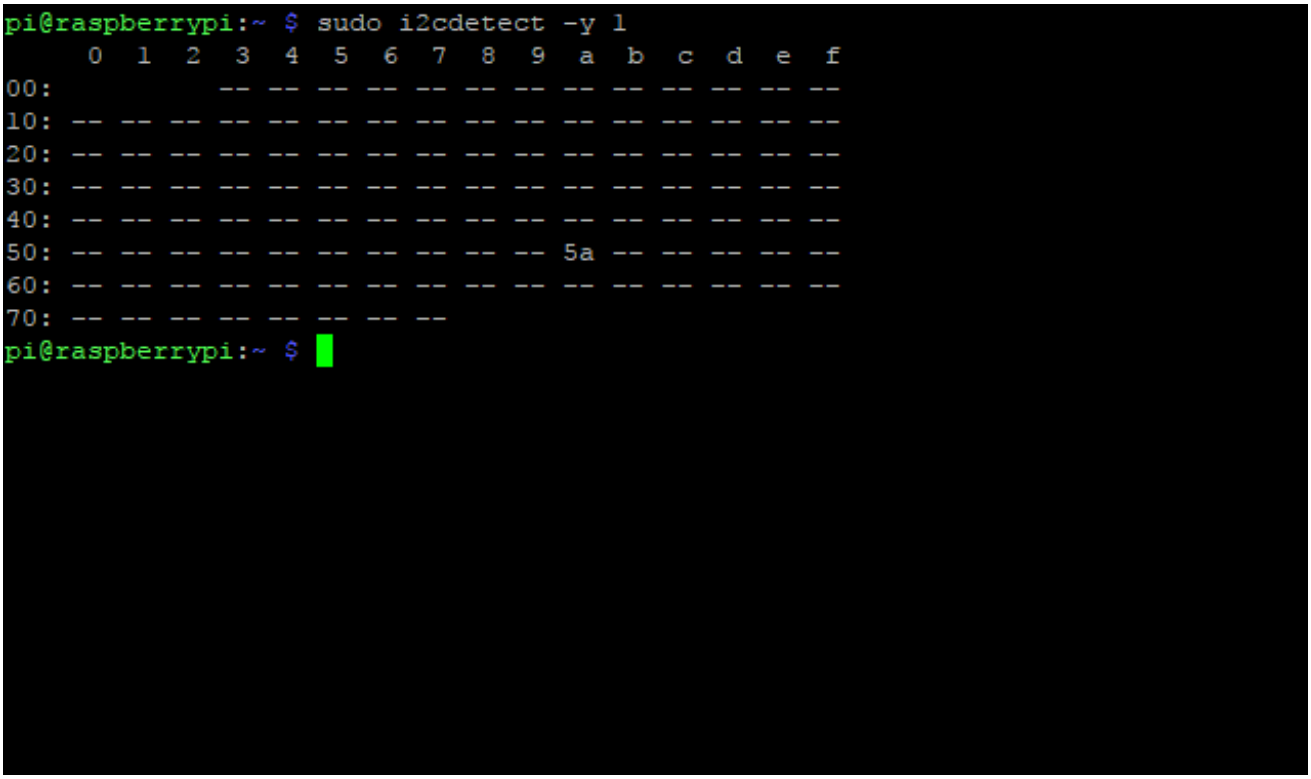
You can also use direct wires, we happen to have a Cobbler ready. remember you can plug the cobbler into the bottom of the PiTFT to get access to all the pins!





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
```



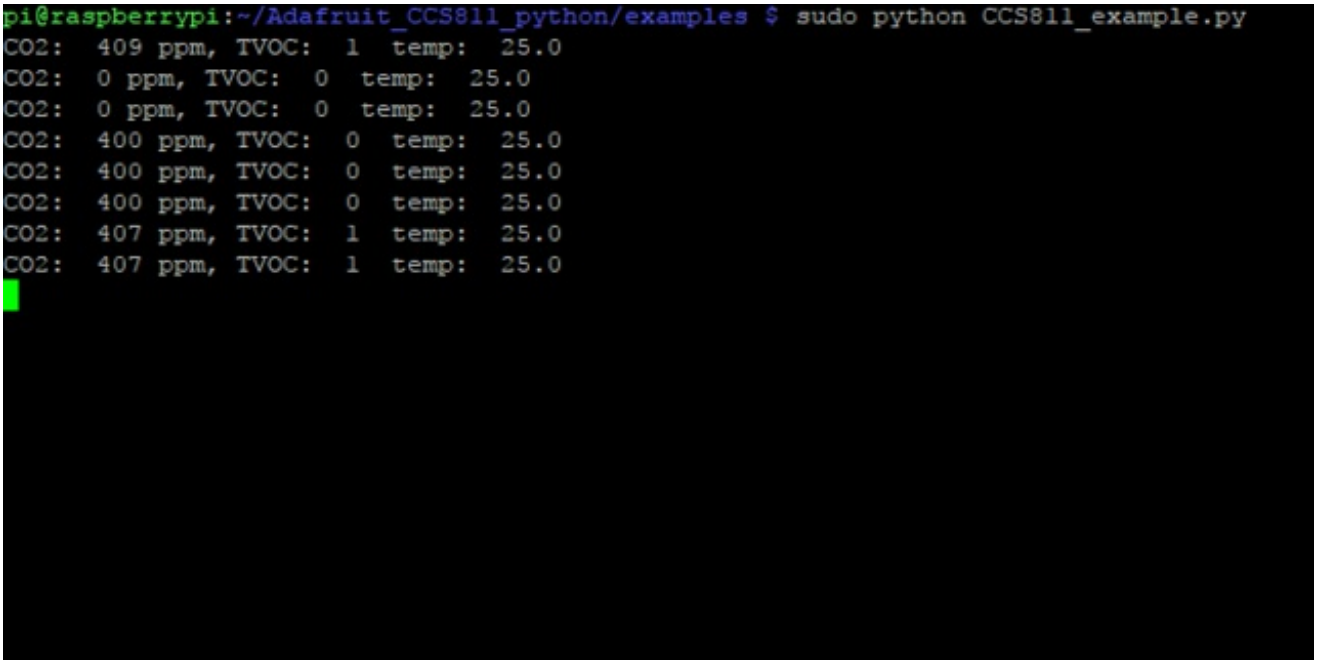
It should show up under it's default address (**0x5A**). If you don't see 5A, check your wiring, did you install I2C support, etc?

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

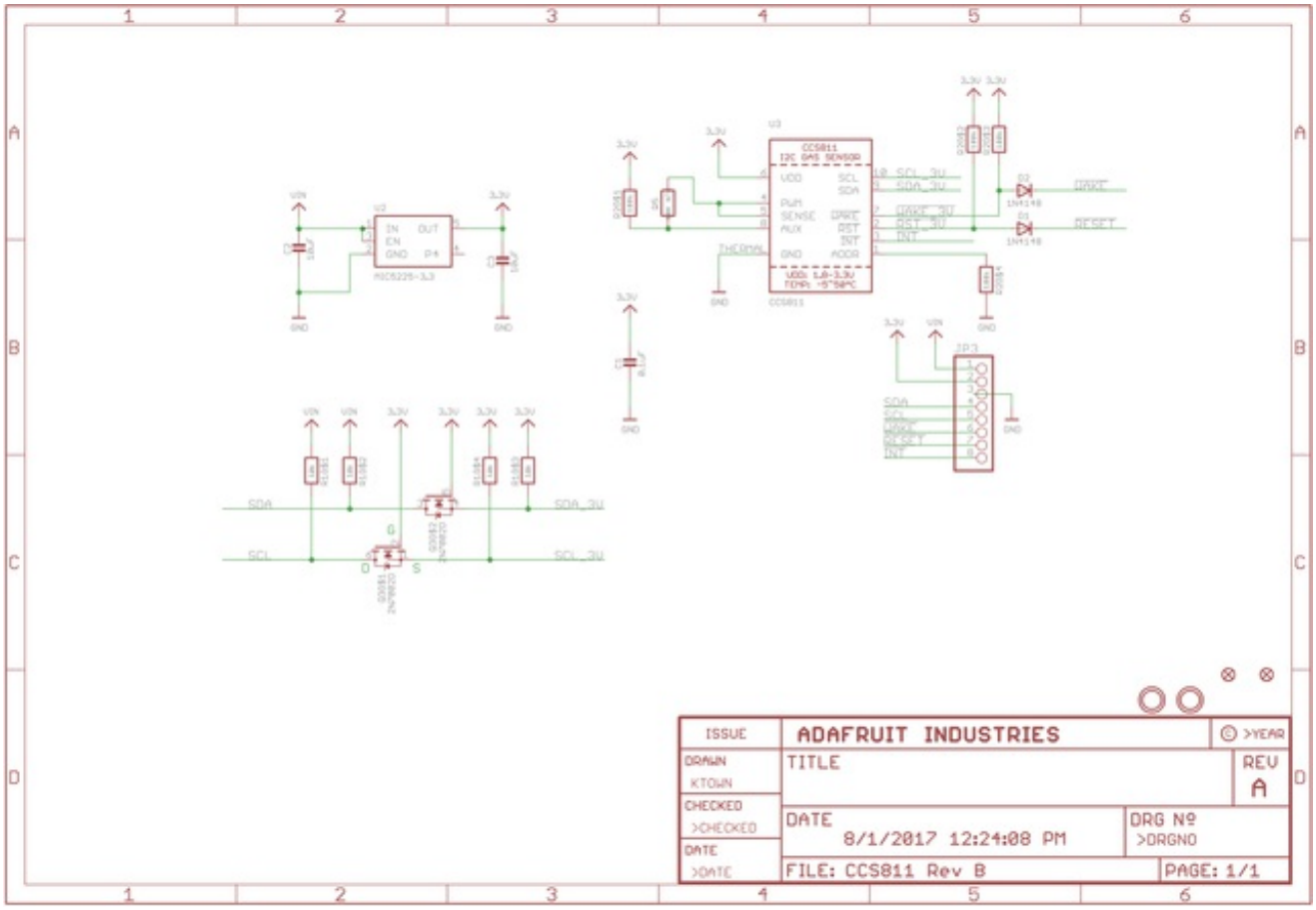
Documents

- CCS811 Datasheet (http://adafru.it/yaV)
- CCS811 Fact sheet (http://adafru.it/ycv)
- CCS811 Mechanical Considerations App Note (http://adafru.it/ycw)
- CCS811 NTC Thermistor App Note (http://adafru.it/ycx)
- CCS811 Baseline Clear/Restore App Note (http://adafru.it/ycy)

- Adafruit CCS811 Arduino Driver (http://adafru.it/yaW)
- CCS811 CircuitPython Driver (http://adafru.it/yaX)
- Fritzing object in the Adafruit Fritzing library (http://adafru.it/aP3)
- CCS811 breakout PCB files (EAGLE format) (http://adafru.it/yaY)

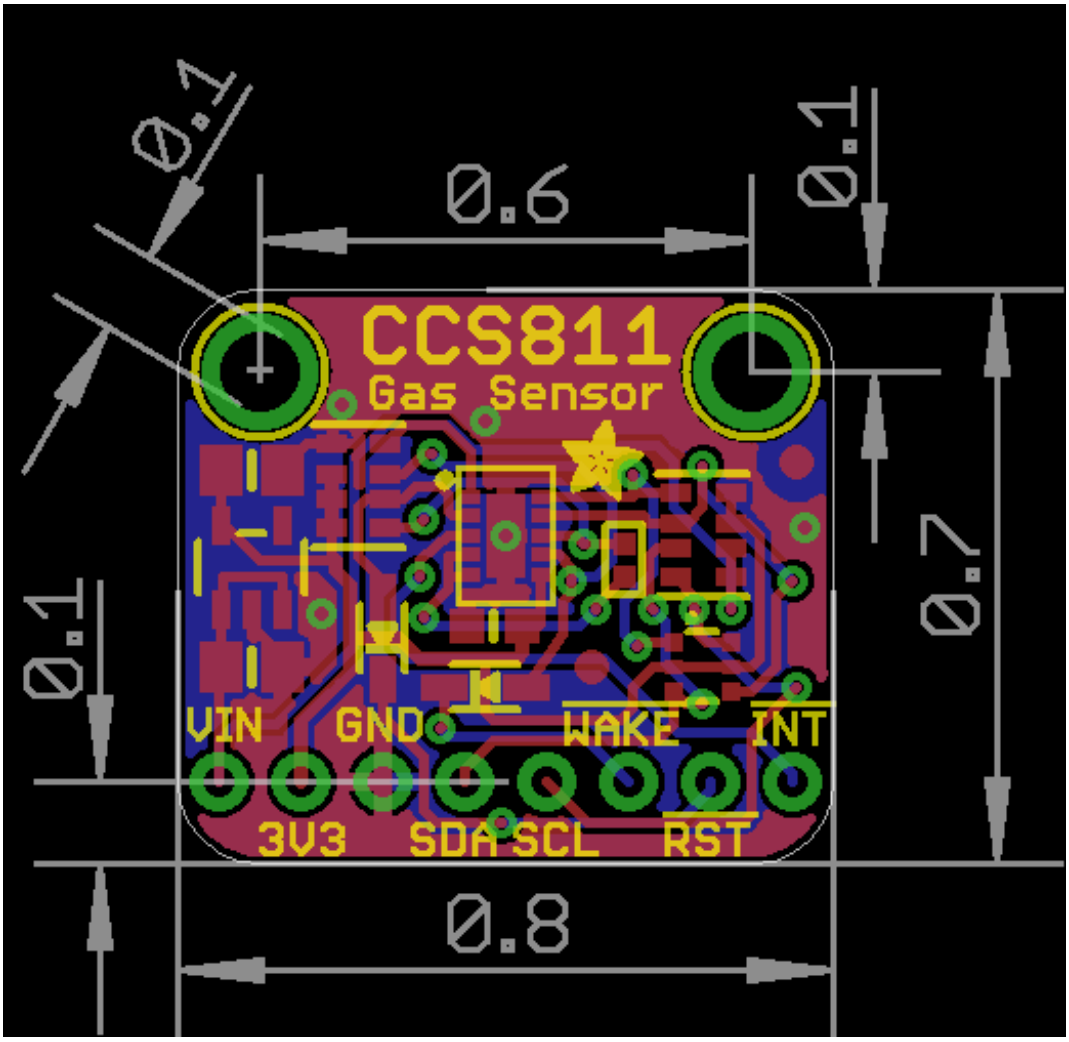
Schematic

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