



Adafruit PMSA003I Air Quality Breakout

Created by Kattni Rembor



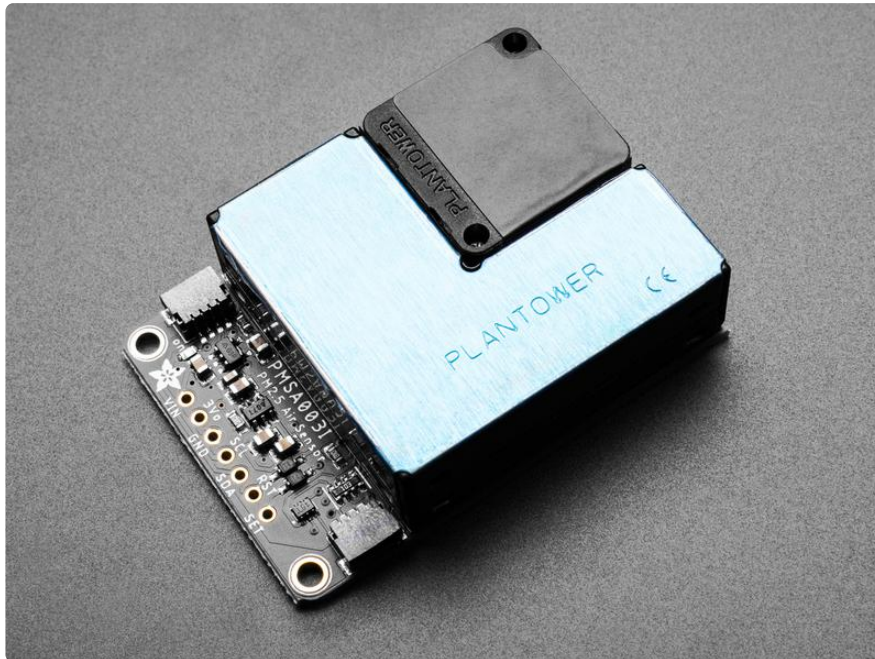
<https://learn.adafruit.com/pmsa003i>

Last updated on 2023-08-29 04:29:14 PM EDT

Table of Contents

Overview	3
Pinouts	6
<ul style="list-style-type: none">• Power Pins• I2C Logic Pins• Other Pins	
Arduino	7
<ul style="list-style-type: none">• I2C Wiring• Library Installation• Load Example• Example Code	
Arduino Docs	11
Python & CircuitPython	11
<ul style="list-style-type: none">• CircuitPython Microcontroller Wiring• Python Computer Wiring• CircuitPython Installation of PM25 Library• Python Installation of PM25 Library• CircuitPython & Python Usage	
Python Docs	16
WipperSnapper Setup	16
<ul style="list-style-type: none">• What is WipperSnapper• Wiring• Usage	
Downloads	23
<ul style="list-style-type: none">• Files• Schematic• Fab Print	

Overview



Breathe easy, knowing that you can track and sense the quality of the air around you with this Adafruit PMSA003I Air Quality Breakout. This sensor is great for monitoring air quality, in a compact plug-in format.

Best of all, unlike almost all other sensors we've seen that are UART interface, this one has an I2C interface, which makes it a great match for single board Linux computers like Raspberry Pi, or even plain Arduino UNO's that normally would use software serial.



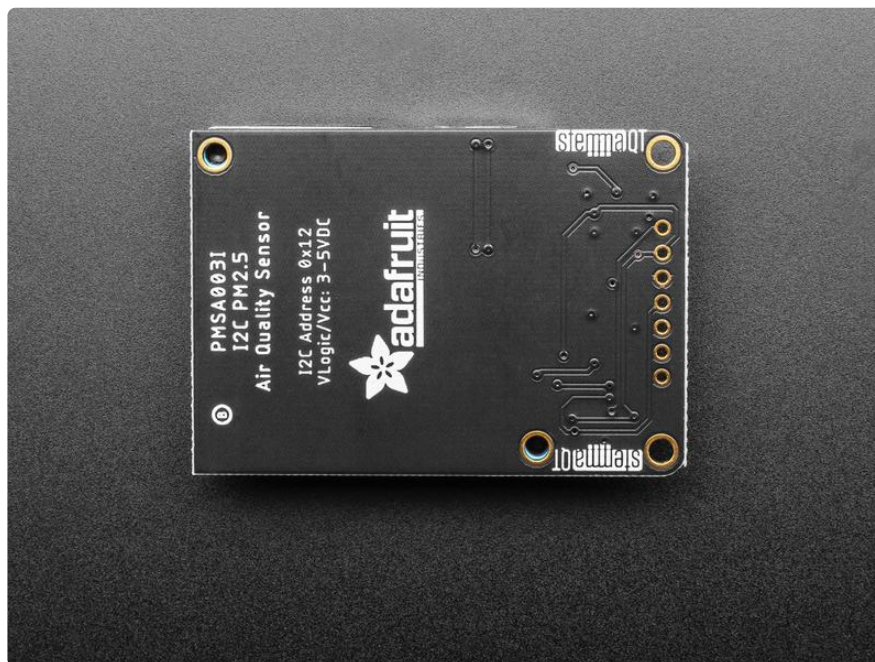
If you're an I2C fan (who isn't?), we've included two of our handy dandy [SparkFun Qwiic \(\)](#) compatible [STEMMA QT \(\)](#) connectors for the I2C bus so you don't even need to solder! Plug and play with other 'QT boards and sensors to add quick air quality sensing.

This sensor uses laser scattering to radiate suspending particles in the air, then collects scattering light to obtain the curve of scattering light change with time. The microprocessor calculates equivalent particle diameter and the number of particles with different diameters per unit volume.

The I2C data stream updates once per second, you'll get:

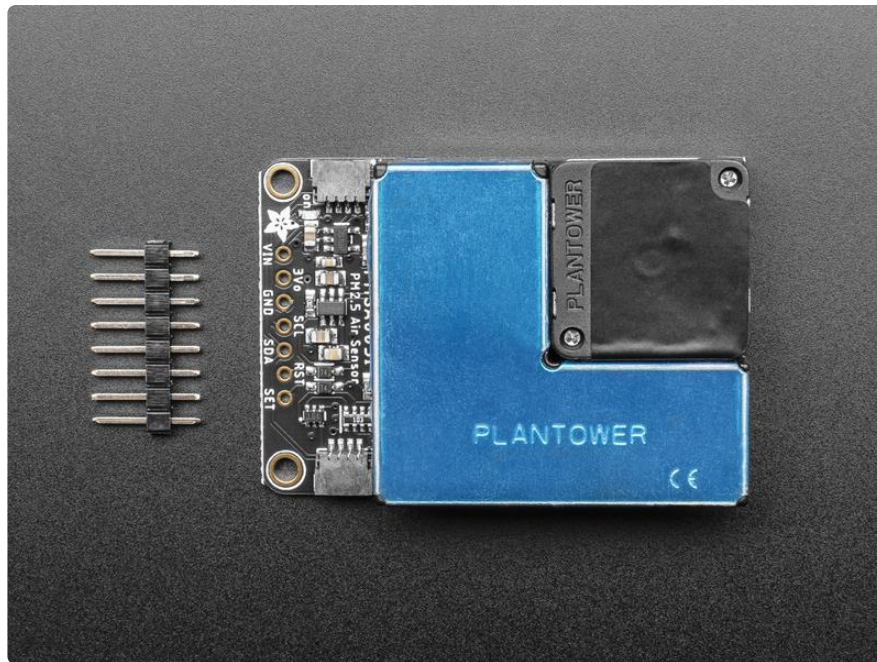
- PM1.0, PM2.5 and PM10.0 concentration in both standard & environmental units
- Particulate matter per 0.1L air, categorized into 0.3um, 0.5um, 1.0um, 2.5um, 5.0um and 10um size bins

As well as checksum, in binary format.



Each order comes with one fully assembled sensor module breakout, including some header if you'd like to solder it to a breadboard. The breakout board has a 5V mini boost circuit so you can power it from 3.3V and 5V and the motor fan inside the sensor will run just fine

Lastly, it wouldn't be an Adafruit breakout if it didn't come with [libraries for Arduino \(\)](#) and [CircuitPython & Python \(\)](#) that will read and checksum data, and print it out in human-readable format.



Pinouts



Power Pins

- Vin - this is the power pin. Since the sensor chip uses 3 VDC, we have included a voltage regulator on board 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 microcontroller 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

I2C Logic Pins

- SCL - I2C clock pin, connect to your microcontroller I2C clock line. On the breakouts, this pin is level shifted so you can use 3-5V logic. There's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontroller I2C data line.
- On the breakouts, this pin is level shifted so you can use 3-5V logic. There's a 10K pullup on this pin.
- [STEMMA QT \(\)](#) - These connectors allow you to make I2C connections to dev boards with STEMMA QT connectors or to other things with [various associated accessories \(\)](#).

Other Pins

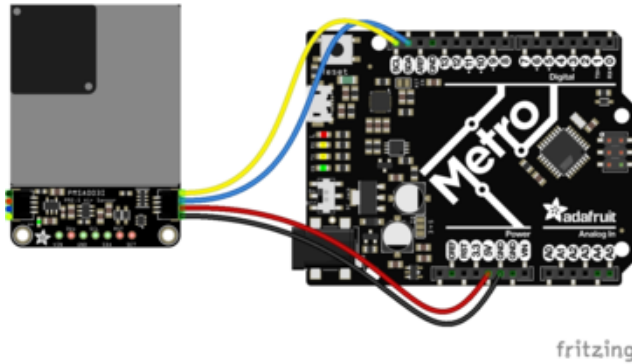
- RST - Module signal reset. Reset low.
- SET - Set pin. High when working status, low level is sleeping mode.

Arduino

Using the PMSA300I with Arduino is a simple matter of wiring up it to your Arduino-compatible microcontroller, installing the [Adafruit PM25AQI \(\)](#) library we've written, and running the provided example code.

I2C Wiring

Wiring the PMSA300I is made simple by using the I2C interface either via the STEMMA QT connector or a solderless breadboard.

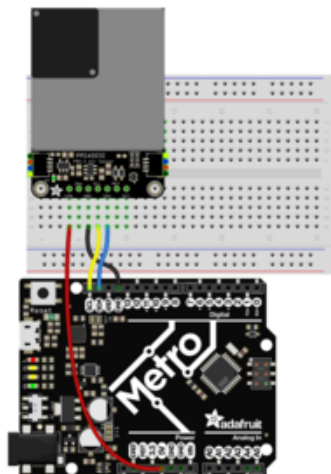


Connect PMSA300I VCC (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.

Connect PMSA300I GND (black wire) to Arduino GND

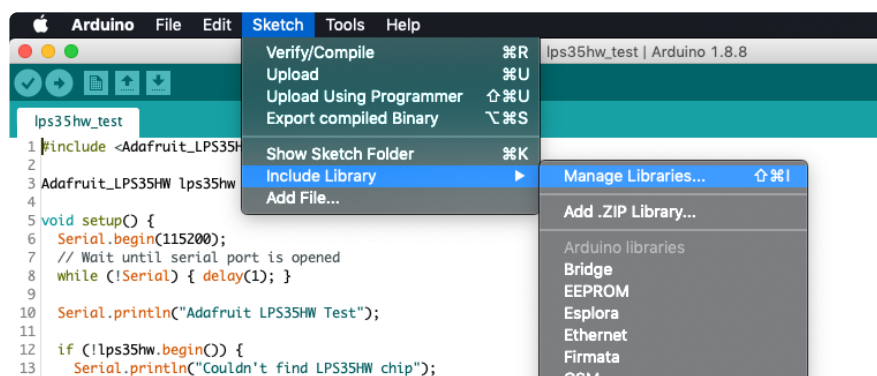
Connect PMSA300I SCL (yellow wire) to Arduino SCL

Connect PMSA300I SDA (blue wire) to Arduino SDA

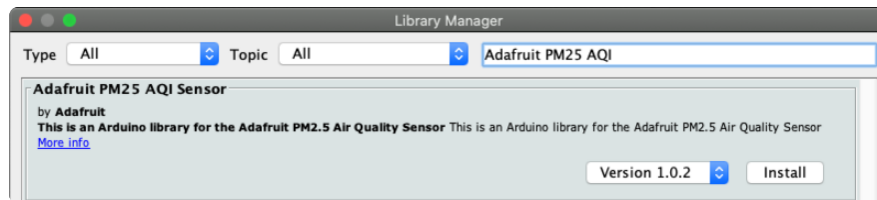


Library Installation

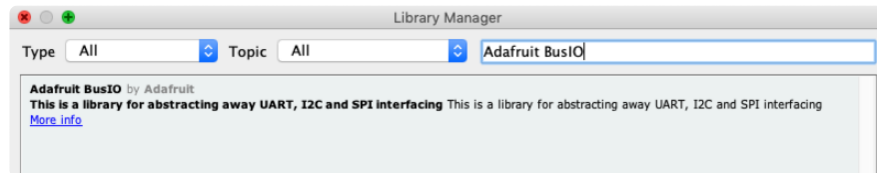
You can install the [Adafruit PM25AQI \(\)](#) library for Arduino using the Library Manager in the Arduino IDE.



Click the Manage Libraries ... menu item, search for Adafruit PM25 AQI, and select the Adafruit PM25 AQI library:



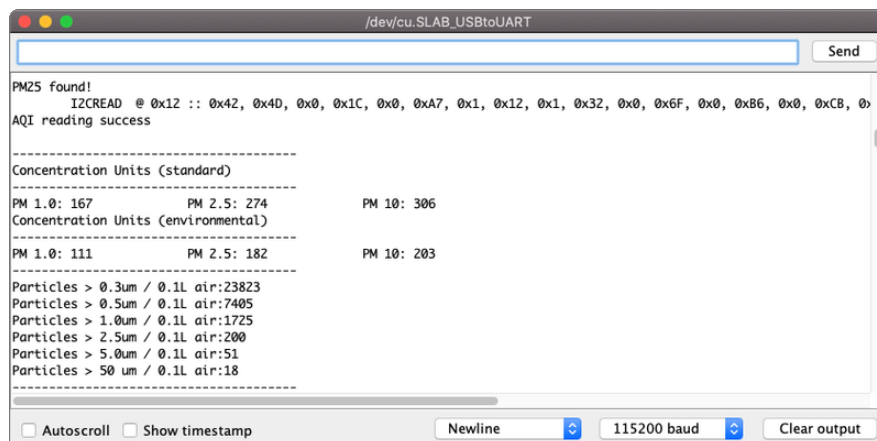
Follow the same process for the Adafruit BusIO library.



Load Example

Open up File -> Examples -> Adafruit PM25 AQI Sensor -> PM25_test

After opening the demo file, upload to your Arduino wired up to the sensor. Once you upload the code, you will see the air quality data being printed when you open the Serial Monitor (Tools->Serial Monitor) at 115200 baud, similar to this:



Example Code

```
/* Test sketch for Adafruit PM2.5 sensor with UART or I2C */

#include "Adafruit_PM25AQI.h"

// If your PM2.5 is UART only, for UNO and others (without hardware serial)
// we must use software serial...
// pin #2 is IN from sensor (TX pin on sensor), leave pin #3 disconnected
// comment these two lines if using hardware serial
// #include <SoftwareSerial.h>
// SoftwareSerial pmSerial(2, 3);

Adafruit_PM25AQI aqi = Adafruit_PM25AQI();

void setup() {
```

```

// Wait for serial monitor to open
Serial.begin(115200);
while (!Serial) delay(10);

Serial.println("Adafruit PMSA003I Air Quality Sensor");

// Wait one second for sensor to boot up!
delay(1000);

// If using serial, initialize it and set baudrate before starting!
// Uncomment one of the following
//Serial1.begin(9600);
//pmSerial.begin(9600);

// There are 3 options for connectivity!
if (!aqi.begin_I2C()) { // connect to the sensor over I2C
//if (! aqi.begin_UART(&Serial1)) { // connect to the sensor over hardware serial
//if (! aqi.begin_UART(&pmSerial)) { // connect to the sensor over software
serial
    Serial.println("Could not find PM 2.5 sensor!");
    while (1) delay(10);
}

    Serial.println("PM25 found!");
}

void loop() {
    PM25_AQI_Data data;

    if (! aqi.read(&data)) {
        Serial.println("Could not read from AQI");
        delay(500); // try again in a bit!
        return;
    }
    Serial.println("AQI reading success");

    Serial.println();
    Serial.println(F("-----"));
    Serial.println(F("Concentration Units (standard)"));
    Serial.println(F("-----"));
    Serial.print(F("PM 1.0: ")); Serial.print(data.pm10_standard);
    Serial.print(F("\t\tPM 2.5: ")); Serial.print(data.pm25_standard);
    Serial.print(F("\t\tPM 10: ")); Serial.println(data.pm100_standard);
    Serial.println(F("Concentration Units (environmental)"));
    Serial.println(F("-----"));
    Serial.print(F("PM 1.0: ")); Serial.print(data.pm10_env);
    Serial.print(F("\t\tPM 2.5: ")); Serial.print(data.pm25_env);
    Serial.print(F("\t\tPM 10: ")); Serial.println(data.pm100_env);
    Serial.println(F("-----"));
    Serial.print(F("Particles > 0.3um / 0.1L air:"));
    Serial.println(data.particles_03um);
    Serial.print(F("Particles > 0.5um / 0.1L air:"));
    Serial.println(data.particles_05um);
    Serial.print(F("Particles > 1.0um / 0.1L air:"));
    Serial.println(data.particles_10um);
    Serial.print(F("Particles > 2.5um / 0.1L air:"));
    Serial.println(data.particles_25um);
    Serial.print(F("Particles > 5.0um / 0.1L air:"));
    Serial.println(data.particles_50um);
    Serial.print(F("Particles > 10 um / 0.1L air:"));
    Serial.println(data.particles_100um);
    Serial.println(F("-----"));

    delay(1000);
}

```

Arduino Docs

[Arduino Docs \(\)](#)

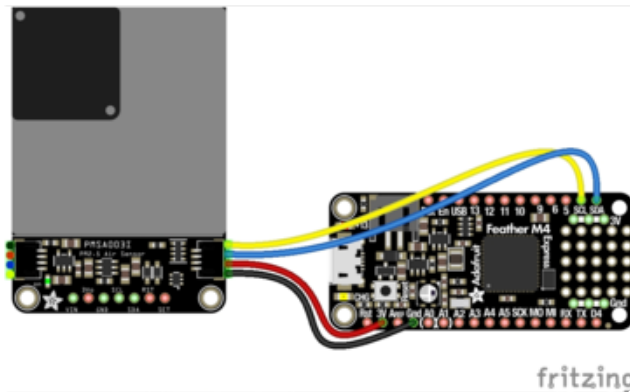
Python & CircuitPython

It's easy to use the PMSA300I and the [Adafruit CircuitPython PM25 \(\)](#) module. This library allows you to easily write Python code that reads particle concentrations, and particle diameter and the number of particles with different diameters per unit volume.

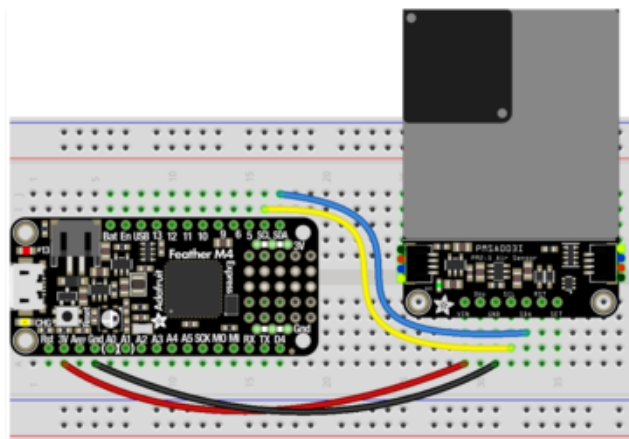
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library \(\)](#).

CircuitPython Microcontroller Wiring

Wire up a PMSA300I to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using STEMMA QT and a solderless breadboard.



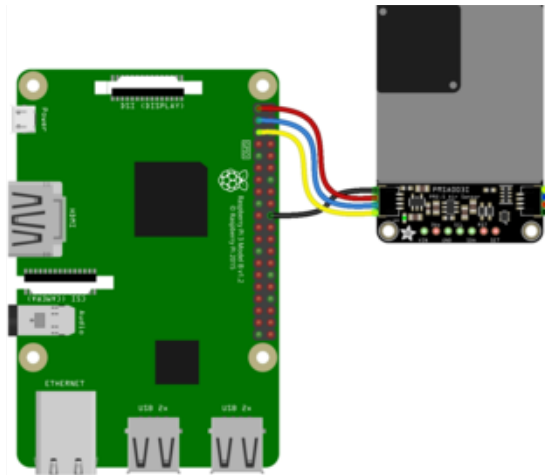
Board 3V to PMSA300I VIN (red wire)
 Board GND to PMSA300I GND (black wire)
 Board SCL to PMSA300I SCL (yellow wire)
 Board SDA to PMSA300I SDA (blue wire)



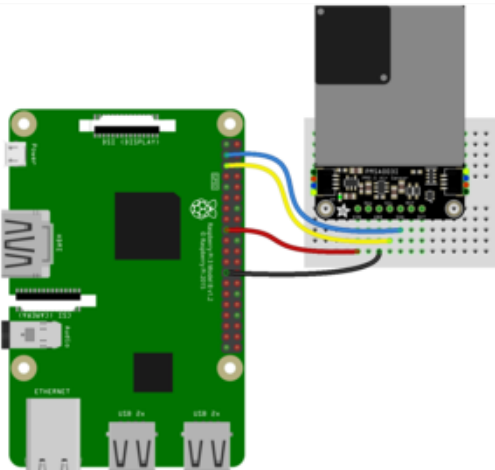
Python Computer Wiring

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 \(\)](#).

Here's the Raspberry Pi wired to the sensor with I2C using STEMMA QT and a solderless breadboard.



Pi 3V to PMSA300I VIN (red wire)
 Pi GND to PMSA300I GND (black wire)
 Pi SCL to PMSA300I SCL (yellow wire)
 Pi SDA to PMSA300I SDA (blue wire)



CircuitPython Installation of PM25 Library

You'll need to install the [Adafruit CircuitPython PM25 \(\)](#) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython \(\)](#) 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 \(\)](#).

Our CircuitPython starter guide has [a great page on how to install libraries from the bundle \(\)](#).

Load the the following libraries into the lib folder on your CIRCUITPY drive:

- adafruit_pm25
- adafruit_bus_device

Before continuing make sure your board's lib folder or root filesystem has the adafruit_pm25.mpy file and adafruit_bus_device folder copied over.

Next [connect to the board's serial console](#) () so you are ready to see the example output.

Python Installation of PM25 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](#) ()!

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

```
pip3 install adafruit-circuitpython-pm25
```

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 PMSA300I, we'll use a complete code example to read the particle data.

Save the following code to your CIRCUITPY drive as code.py:

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
Example sketch to connect to PM2.5 sensor with either I2C or UART.
"""

# pylint: disable=unused-import
import time
import board
import busio
from digitalio import DigitalInOut, Direction, Pull
from adafruit_pm25.i2c import PM25_I2C

reset_pin = None
# If you have a GPIO, its not a bad idea to connect it to the RESET pin
# reset_pin = DigitalInOut(board.G0)
# reset_pin.direction = Direction.OUTPUT
```

```

# reset_pin.value = False

# For use with a computer running Windows:
# import serial
# uart = serial.Serial("COM30", baudrate=9600, timeout=1)

# For use with microcontroller board:
# (Connect the sensor TX pin to the board/computer RX pin)
# uart = busio.UART(board.TX, board.RX, baudrate=9600)

# For use with Raspberry Pi/Linux:
# import serial
# uart = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=0.25)

# For use with USB-to-serial cable:
# import serial
# uart = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=0.25)

# Connect to a PM2.5 sensor over UART
# from adafruit_pm25.uart import PM25_UART
# pm25 = PM25_UART(uart, reset_pin)

# Create library object, use 'slow' 100KHz frequency!
i2c = busio.I2C(board.SCL, board.SDA, frequency=100000)
# Connect to a PM2.5 sensor over I2C
pm25 = PM25_I2C(i2c, reset_pin)

print("Found PM2.5 sensor, reading data...")

while True:
    time.sleep(1)

    try:
        aqdata = pm25.read()
        # print(aqdata)
    except RuntimeError:
        print("Unable to read from sensor, retrying...")
        continue

    print()
    print("Concentration Units (standard)")
    print("-----")
    print(
        "PM 1.0: %d\tPM2.5: %d\tPM10: %d"
        % (aqdata["pm10 standard"], aqdata["pm25 standard"], aqdata["pm100
standard"])
    )
    print("Concentration Units (environmental)")
    print("-----")
    print(
        "PM 1.0: %d\tPM2.5: %d\tPM10: %d"
        % (aqdata["pm10 env"], aqdata["pm25 env"], aqdata["pm100 env"])
    )
    print("-----")
    print("Particles > 0.3um / 0.1L air:", aqdata["particles 03um"])
    print("Particles > 0.5um / 0.1L air:", aqdata["particles 05um"])
    print("Particles > 1.0um / 0.1L air:", aqdata["particles 10um"])
    print("Particles > 2.5um / 0.1L air:", aqdata["particles 25um"])
    print("Particles > 5.0um / 0.1L air:", aqdata["particles 50um"])
    print("Particles > 10 um / 0.1L air:", aqdata["particles 100um"])
    print("-----")

```

If you haven't already, connect to the serial console to see the example output.

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Found PM2.5 sensor, reading data...

Concentration Units (standard)
-----
PM 1.0: 143    PM2.5: 242    PM10: 258
Concentration Units (environmental)
-----
PM 1.0: 95     PM2.5: 160    PM10: 171
-----
Particles > 0.3um / 0.1L air: 20952
Particles > 0.5um / 0.1L air: 6426
Particles > 1.0um / 0.1L air: 1483
Particles > 2.5um / 0.1L air: 177
Particles > 5.0um / 0.1L air: 30
Particles > 10 um / 0.1L air: 3
-----
```

That's all there is to reading air quality data from the PSMA300!

Python Docs

[Python Docs \(\)](#)

WipperSnapper Setup

To use the PMSA003I with WipperSnapper, you must be running WipperSnapper Beta 44 or newer

What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to [Adafruit IO \(\)](#), a web platform designed ([by Adafruit! \(\)](#)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

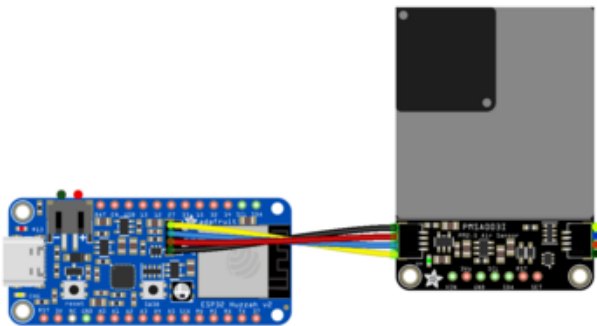
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

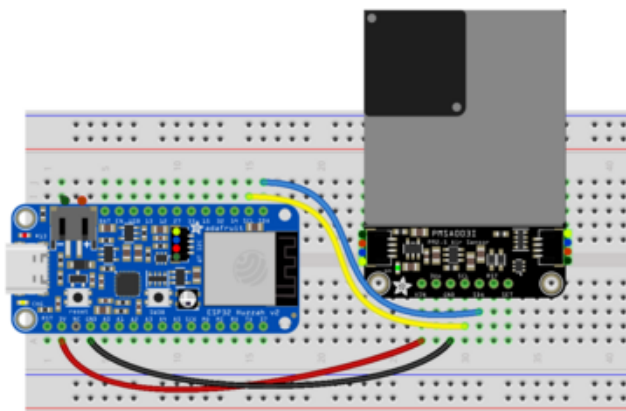
Quickstart: Adafruit IO WipperSnapper

Wiring

First, wire up a PMSA003I to your board exactly as follows. Here is an example of the PMSA003I wired to an [Adafruit ESP32 Feather V2 \(\)](#) using I2C [with a STEMMA QT cable \(no soldering required\) \(\)](#)



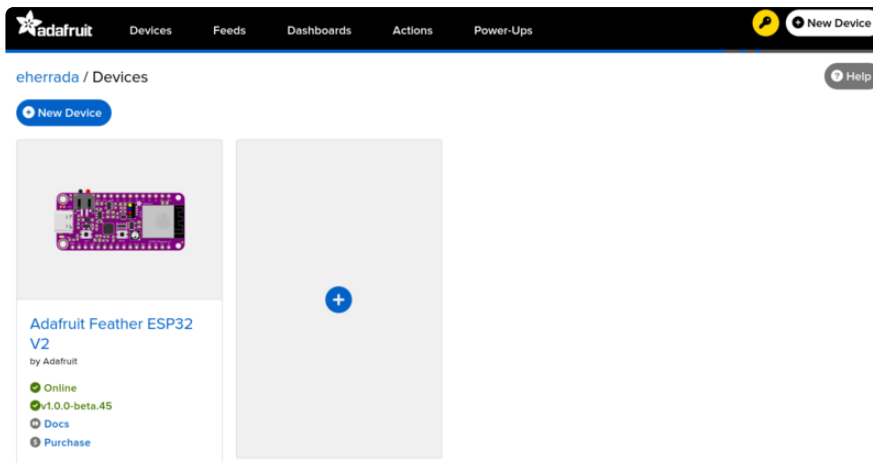
Board 3V to sensor VIN (red wire on STEMMA QT)
Board GND to sensor GND (black wire on STEMMA QT)
Board SCL to sensor SCL (yellow wire on STEMMA QT)
Board SDA to sensor SDA (blue wire on STEMMA QT)



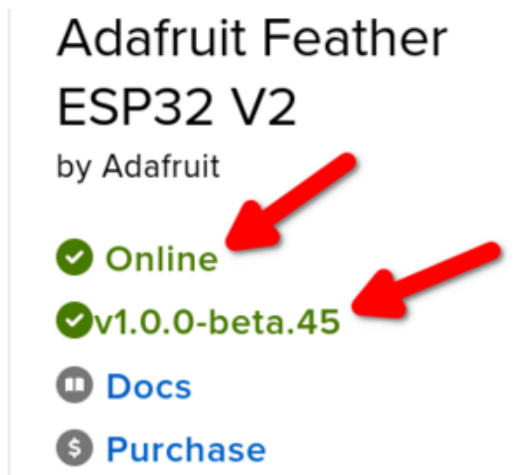
Usage

Connect your board to Adafruit IO Wippersnapper and [navigate to the WipperSnapper board list \(\)](#).

On this page, select the WipperSnapper board you're using to be brought to the board's interface page.



If you do not see your board listed here - you need [to connect your board to Adafruit IO \(\)](#) first.

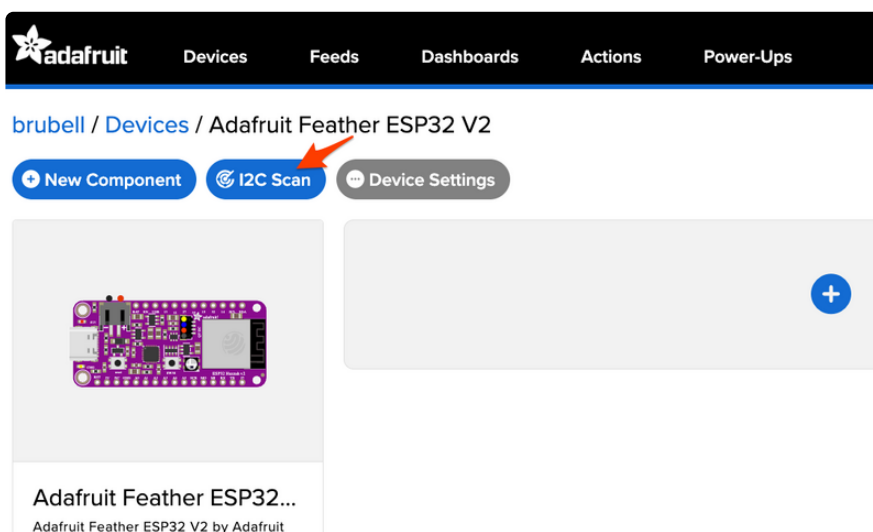


On the device page, quickly check that you're running the latest version of the WipperSnapper firmware.

The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide. If the firmware version is red with an "X" - [update to the latest WipperSnapper firmware \(\)](#) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the I2C Scan button.



You should see the PMSA003I's default I2C address of **0x12** pop-up in the I2C scan list.

I2C Scan Complete



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00								--	--	--	--	--	--	--	--	--
10	--	--	12	--	--	--	--	--	--	--	--	--	--	--	--	--
20	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Close

Scan Again

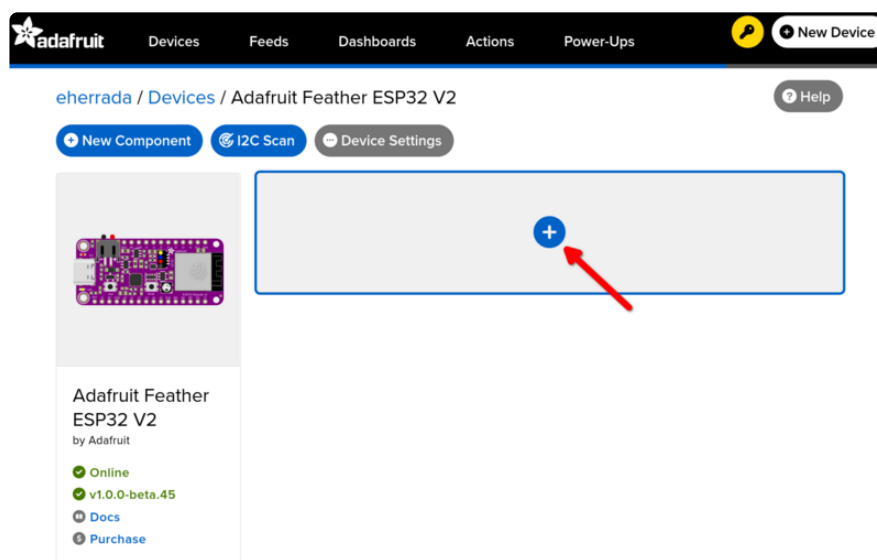
I don't see the sensor's I2C address listed!

First, double-check the connection and/or wiring between the sensor and the board.

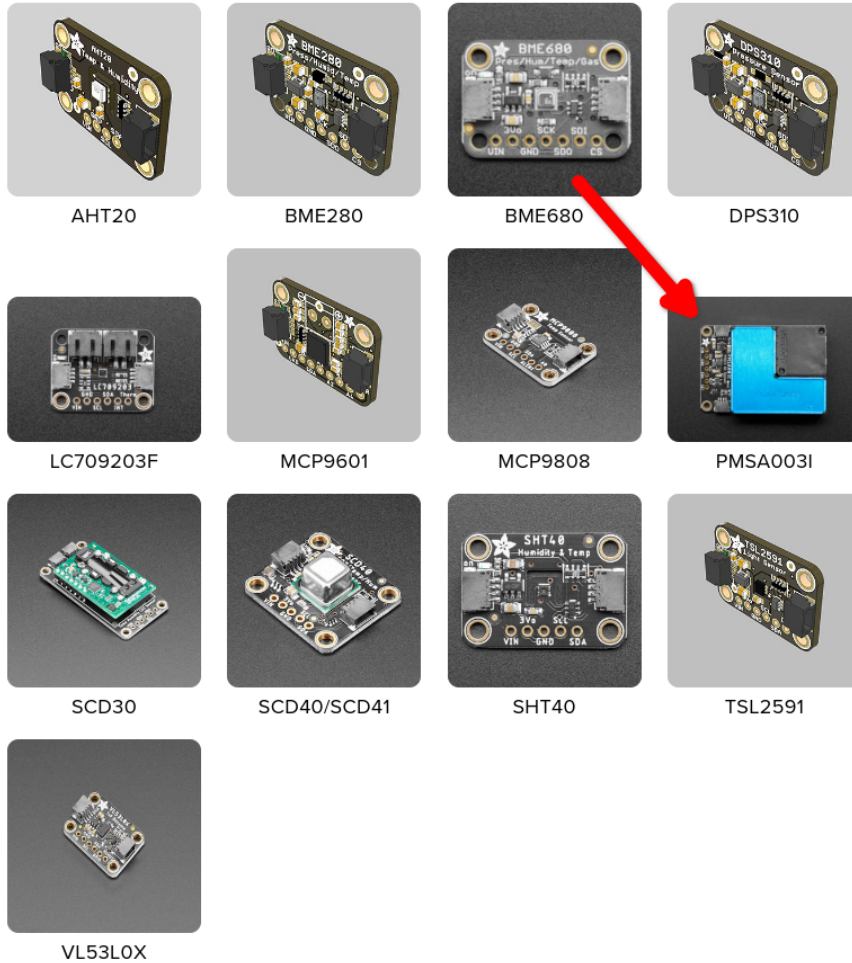
Then, reset the board and let it re-connect to Adafruit IO WhipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the New Component button or the + button to bring up the component picker.



I2C Components



On the component configuration page, the PMSA003I's sensor address should be listed along with the sensor's settings.

The Send Every option is specific to each sensor's measurements. This option will tell the Feather how often it should read from each of the PMSA003I's three sensors and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

On WipperSnapper, the PMSA003I reports particulate matter PM1.0, PM2.5, and PM10.0 concentrations in standard units only. If you need particulate matter reported in environmental units or particulate matter categorized into 0.3um, 0.5um, 1.0um, 2.5um, 5.0um, and 10um size bins, use the CircuitPython or Arduino libraries.

For this example, set the Send Every interval for each sensor to every 30 seconds.

Create PMSA003I Component



Select I2C Address:

0x12

☒ Enable PMSA003I: PM10 Standard?

Name:

PMSA003I: PM10 Standard

Send Every:

Every 30 seconds

☒ Enable PMSA003I: PM25 Standard?

Name:

PMSA003I: PM25 Standard

Send Every:

Every 15 minutes

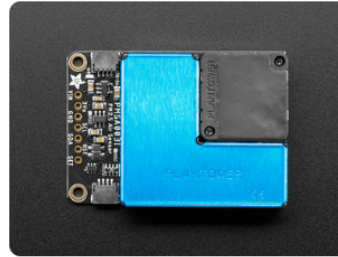
☒ Enable PMSA003I: PM100 Standard?

Name:

PMSA003I: PM100 Standard

Send Every:

Every 30 seconds




< Previous Step

Create Component

Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.

eherrada / Devices / Adafruit Feather ESP32 V2

New Component I2C Scan Device Settings Help



Adafruit Feather ESP32 V2
by Adafruit

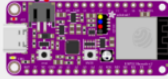
- Online
- v1.0.0-beta.45
- Docs
- Purchase

PMSA003I: PM100 Standard	<input type="text" value="pmsa003i:pm100-std"/>	20.00ppm
Create Action Add to Dashboard		
PMSA003I: PM10 Standard	<input type="text" value="pmsa003i:pm10-std"/>	9.00ppm
Create Action Add to Dashboard		
PMSA003I: PM25 Standard	<input type="text" value="pmsa003i:pm25-std"/>	12.00ppm
Create Action Add to Dashboard		

To view the data that has been logged from the sensor, click on the graph next to the sensor name.







eherrada / Devices / Adafruit Feather ESP32 V2

[New Component](#) [I2C Scan](#) [Device Settings](#) [Help](#)



Adafruit Feather
ESP32 V2
by Adafruit

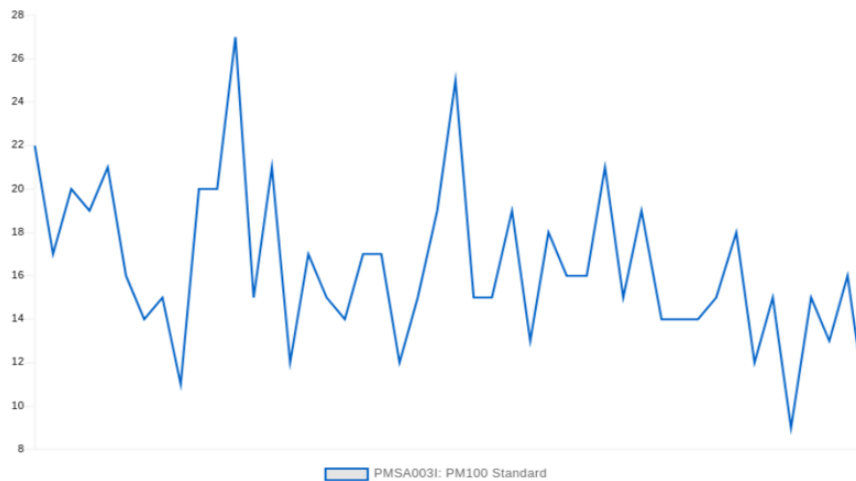
- Online
- v1.0.0-beta.45
- [Docs](#)
- [Purchase](#)

PMSA003I: PM100 Standard	pmsa003i:pm100-std	 
Create Action Add to Dashboard		16.00ppm
PMSA003I: PM10 Standard	pmsa003i:pm10-std	 
Create Action Add to Dashboard		9.00ppm
PMSA003I: PM25 Standard	pmsa003i:pm25-std	 
Create Action Add to Dashboard		12.00ppm

Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, [check out this page \(\)](#).

The PMSA003I sensor reads data and optionally categorizes it into PM1.0, PM2.5, and PM10.0 bins. Each bin has its own feed on Adafruit IO. In this picture, we're looking at the PM100 readings, but if you click on the graph icon for the different sensors you'll see their feed history.

eherrada / Feeds / PMSA003I: PM100 Standard



[+ Add Data](#) [Download All Data](#) [Filter](#)

[< Prev](#) [First](#) page of 1 [Next >](#)

Created at	Value	Location
2022/09/02 4:34:0...	10	×
2022/09/02 4:33:3...	16	×
2022/09/02 4:33:0...	13	×
2022/09/02 4:32:3...	15	×
2022/09/02 4:31:5...	9	×
2022/09/02 4:31:2...	15	×
2022/09/02 4:30:5...	12	×
2022/09/02 4:30:2...	18	×

For IO Free accounts, feed data is stored for a maximum of 30 days and there's a maximum of 10 feeds. In this guide, you created three feeds (one for each of the PMSA003I's sensors). If you'd like to store data for more than 30 days, increase the number of feeds (components) you can use with WipperSnapper, or increase your data rate to send more sensor measurements to Adafruit IO - [upgrade your account to Adafruit IO Plus \(\)](#).

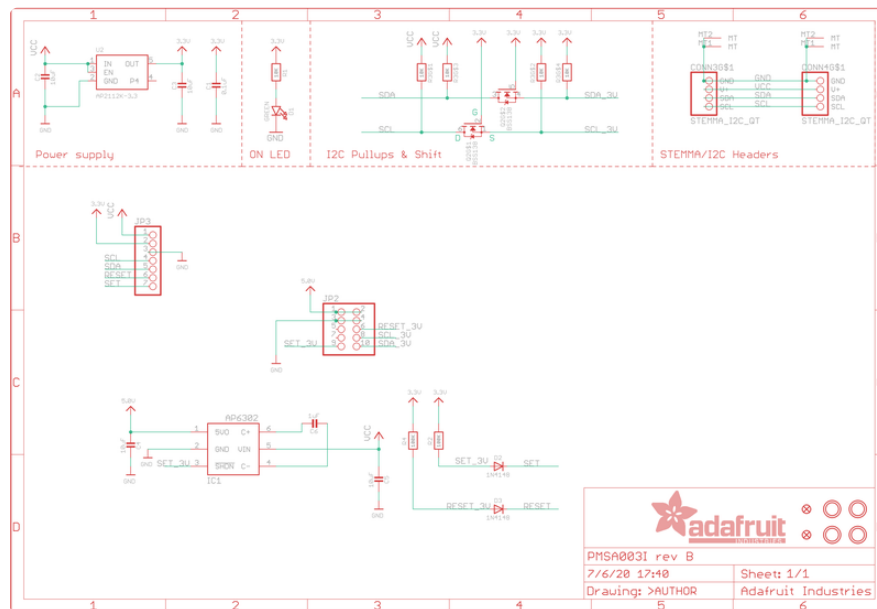
Downloads

Files

- [PMSA003I datasheet \(\)](#)
- [EagleCAD files on GitHub \(\)](#)
- [3D Models on GitHub \(\)](#)

- [Fritzing object in Adafruit Fritzing Library \(\)](#)

Schematic



Fab Print

