Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

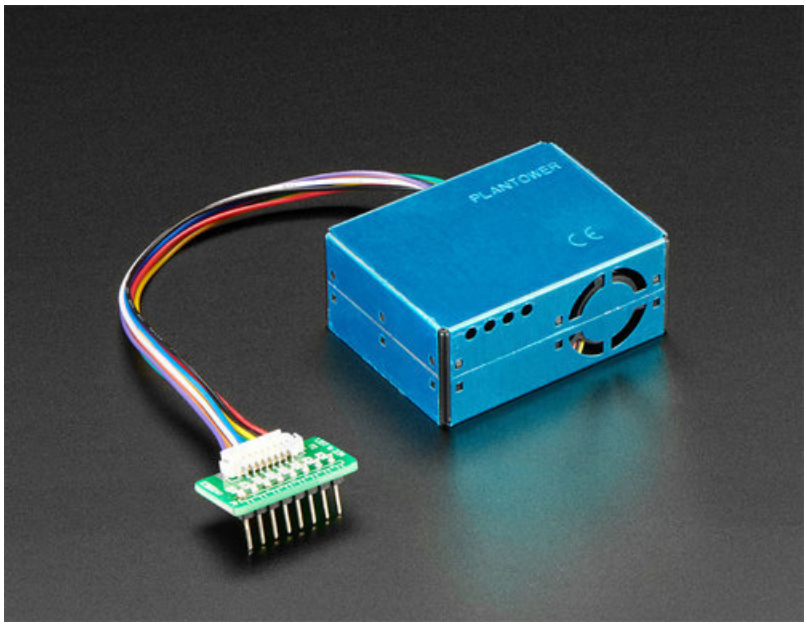
Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

adafruit learning system

PM2.5 Air Quality Sensor

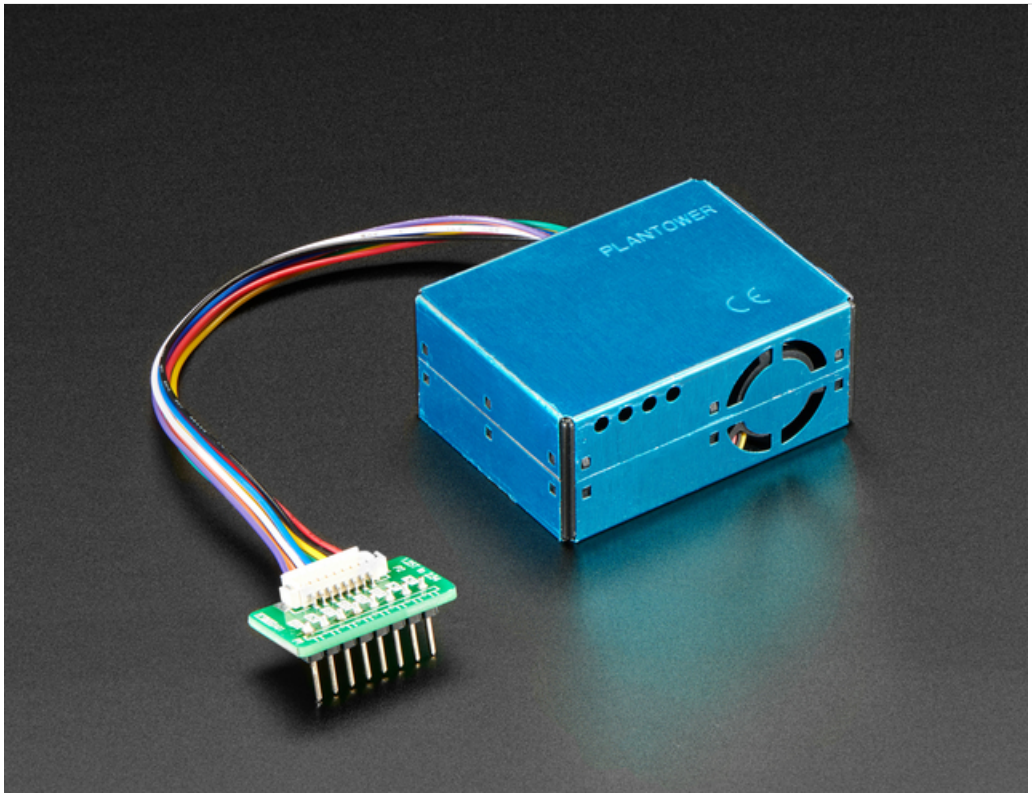
Created by lady ada



Last updated on 2018-04-25 06:37:28 PM UTC

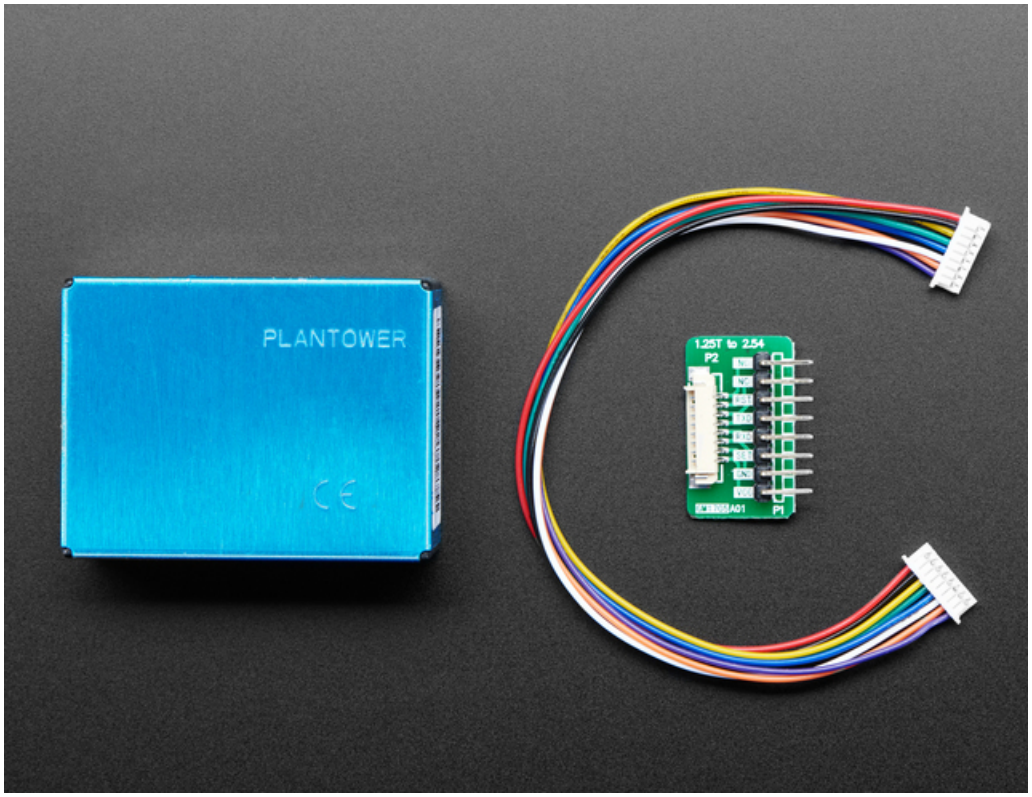
Guide Contents

Overview



Breathe easy, knowing that you can track and sense the quality of the air around you with th**e PM2.5 Air Quality Sensor with Breadboard Adapter** particulate sensor. Mad Max & Furiosa definitely should have hooked up one of these in their truck while scavenging the dusty desert wilderness of post-apocalyptic Australia. And for those of us not living in an Outback dystopia, this sensor + adapter kit is great for monitoring air quality, and super easy to use!

WITNESS real-time, reliable measurement of PM2.5 dust concentrations! (PM2.5 refers to particles that are 2.5 microns or smaller in diameter.) 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 diameter per unit volume.



You'll need to hook this up to a microcontroller with UART input (or you could theoretically wire it up to a USB-Serial converter and parse the data on a computer) - we have code for both Arduino and CircuitPython. 9600 baud data streams out once per second, you'll get:

- PM1.0, PM2.5 and PM10.0 concentration in both standard & enviromental 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 (its fairly easy to parse the binary format, but it doesn't come out as pure readable ascii text)

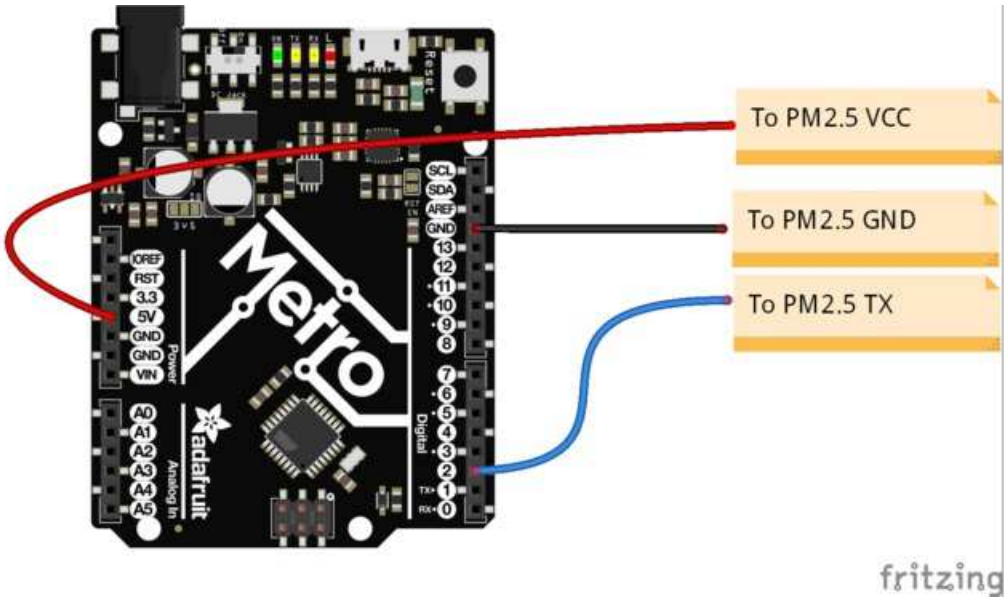
We give you the sensor box as well as the cable and a 0.1" / 2.54mm breakout board so you can wire it easily. You only need power plus one data pin (for the UART TX). Power is 5V, logic is 3.3V

Arduino Code

This code will get you started with any Arduino compatible (e.g. Arduino UNO, Adafruit Metro, ESP8266, Teensy, etc. As long as you have either a hardware serial or software serial port that can run at 9600 baud.

Wiring

Wiring is simple! Power the sensor with **+5V** and **GND** and then connect the data out pin (3.3V logic) to the serial input pin you'll use. Whether or not you are using hardware or software UART/serial may affect the pin, so adjust that as necessary. This wiring works for ATMega328P-based boards for sure, with Digital #2 as the data pin:



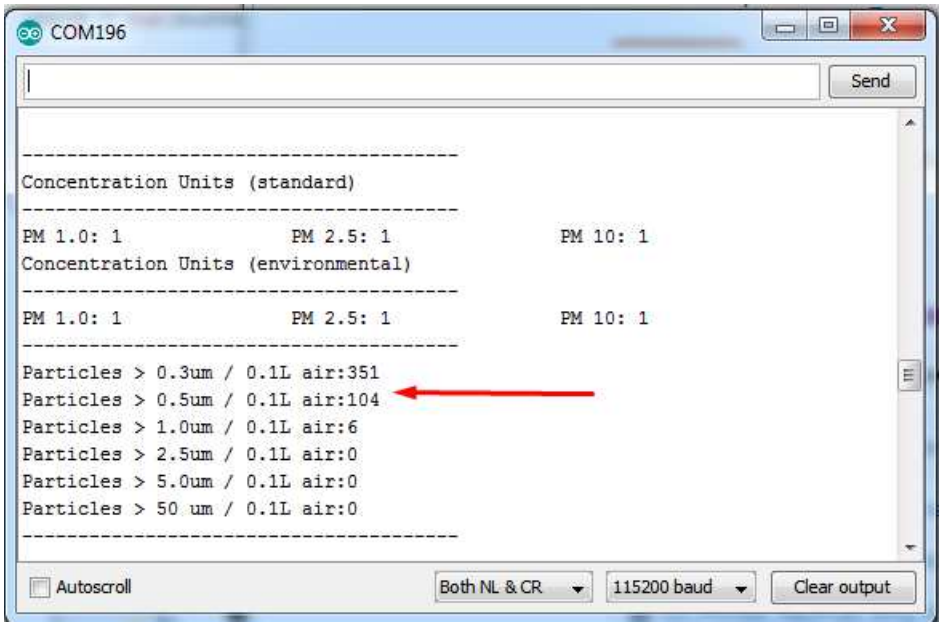
Upload this code to your board, and open up the serial console at **115200** baud

```
// On Leonardo/Micro or others with hardware serial, use those!
// uncomment this line:
// #define pmsSerial Serial1

// 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 pmsSerial(2, 3);

void setup() {
  // our debugging output
  Serial.begin(115200);

  // sensor baud rate is 9600
  pmsSerial.begin(9600);
}

struct pms5003data {
  uint16_t framelen;
  uint16_t pm10_standard, pm25_standard, pm100_standard;
  uint16_t pm10_env, pm25_env, pm100_env;
  uint16_t particles_03um, particles_05um, particles_10um, particles_25um, particles_50um, particles_100u
  uint16_t unused;
```

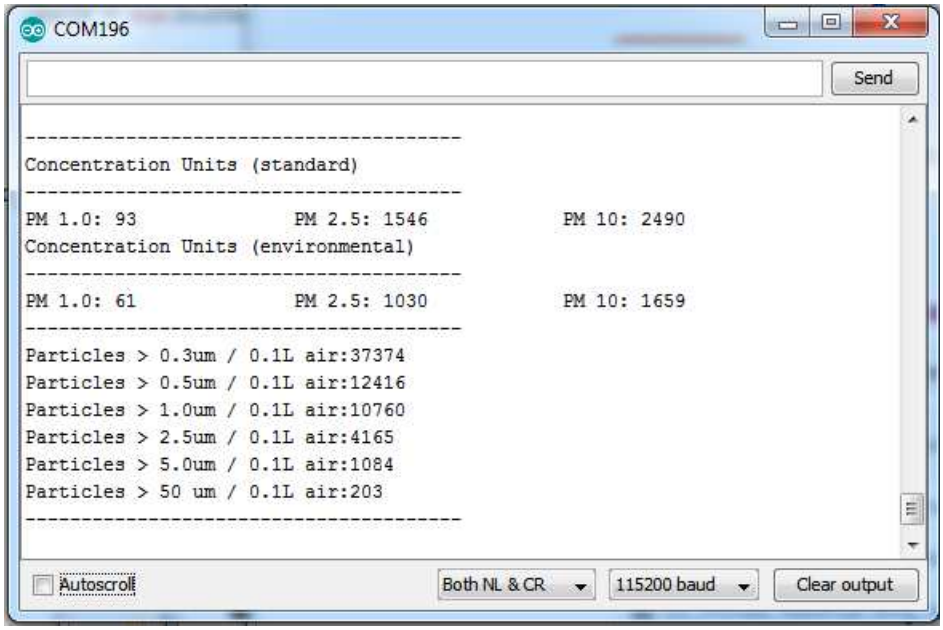
```
  uint16_t checksum;
};

struct pms5003data data;

void loop() {
  if (readPMSdata(&pmsSerial)) {
    // reading data was successful!
    Serial.println();
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (standard)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_standard);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_standard);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_standard);
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (environmental)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_env);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_env);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_env);
    Serial.println("---------------------------------------");
    Serial.print("Particles > 0.3um / 0.1L air:"); Serial.println(data.particles_03um);
    Serial.print("Particles > 0.5um / 0.1L air:"); Serial.println(data.particles_05um);
    Serial.print("Particles > 1.0um / 0.1L air:"); Serial.println(data.particles_10um);
    Serial.print("Particles > 2.5um / 0.1L air:"); Serial.println(data.particles_25um);
    Serial.print("Particles > 5.0um / 0.1L air:"); Serial.println(data.particles_50um);
    Serial.print("Particles > 50 um / 0.1L air:"); Serial.println(data.particles_100um);
    Serial.println("---------------------------------------");
  }
}

boolean readPMSdata(Stream *s) {
  if (! s->available()) {
    return false;
  }

  // Read a byte at a time until we get to the special '0x42' start-byte
  if (s->peek() != 0x42) {
    s->read();
    return false;
  }

  // Now read all 32 bytes
  if (s->available() < 32) {
    return false;
  }

  uint8_t buffer[32];
  uint16_t sum = 0;
  s->readBytes(buffer, 32);

  // get checksum ready
  for (uint8_t i=0; i<30; i++) {
    sum += buffer[i];
  }

  /* debugging
  for (uint8_t i=2; i<32; i++) {
    Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
  }
  Serial.println();
```

```
  */

  // The data comes in endian'd, this solves it so it works on all platforms
  uint16_t buffer_u16[15];
  for (uint8_t i=0; i<15; i++) {
    buffer_u16[i] = buffer[2 + i*2 + 1];
    buffer_u16[i] += (buffer[2 + i*2] << 8);
  }

  // put it into a nice struct :)
  memcpy((void *)&data, (void *)buffer_u16, 30);

  if (sum != data.checksum) {
    Serial.println("Checksum failure");
    return false;
  }
  // success!
  return true;
}
```

You'll see data printed out once a second, with all the measurements. For a clean-air indoor room you'll see something like this:



If you hold up a smoking soldering iron or something else that creates a lot of dust, you'll see much higher numbers!

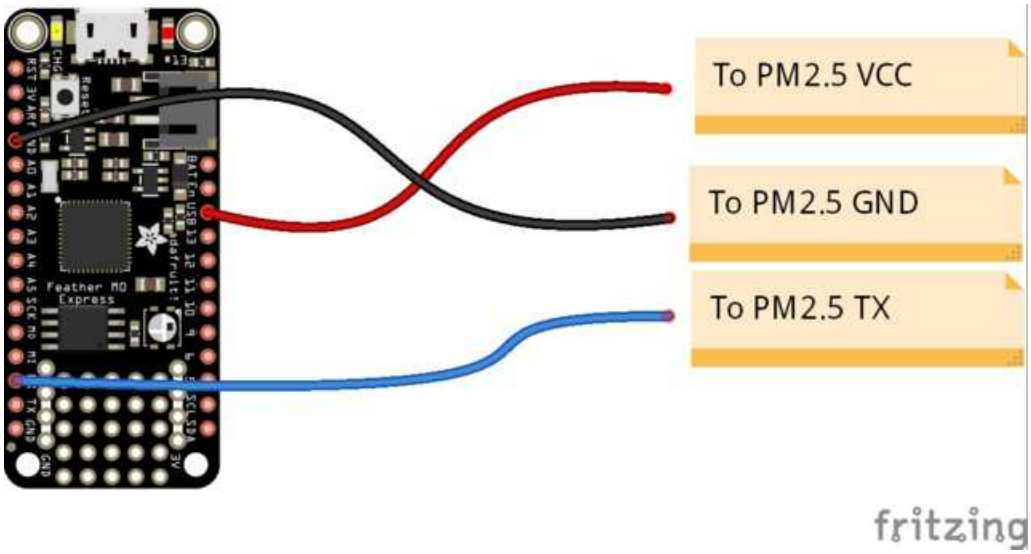


Note that the numbers are very precise looking but we don't believe that they're going to be perfectly accurate, calibration may be necessary!

CircuitPython Code

This code's pretty simple, you can hook up the power and ground pins to 5V and GND, respectively

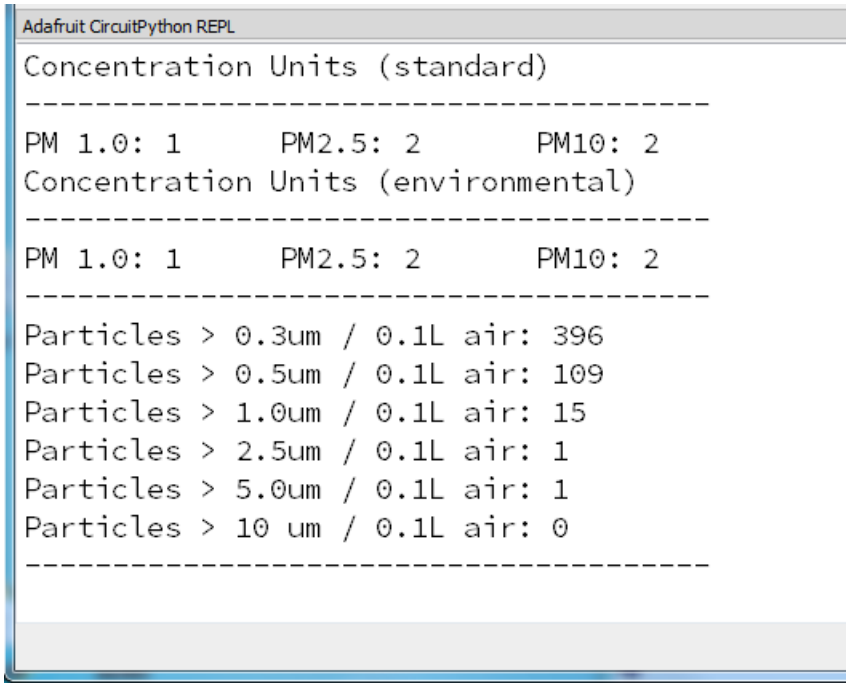
Then wire up the **TX** pin from the sensor to **RX** pin on your CircuitPython board. Note, **RX** does not connect to **RX**!




```
from digitalio import DigitalInOut, Direction
import board
import busio
import time
import ustruct as struct

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

# Connect the Sensor's TX pin to the board's RX pin
uart = busio.UART(board.TX, board.RX, baudrate=9600)

buffer = []

while True:
    data = uart.read(32)  # read up to 32 bytes
    data = list(data)
    #print("read: ", data)          # this is a bytearray type

    buffer += data

    while buffer and buffer[0] != 0x42:
        buffer.pop(0)

    if len(buffer) < 32:
        continue

    if buffer[1] != 0x4d:
        buffer.pop(0)
        continue

    frame_len = struct.unpack(">H", bytes(buffer[2:4]))[0]
    if frame_len != 28:
        continue
```

```
        continue

    frame = struct.unpack(">HHHHHHHHHHHHHH", bytes(buffer[4:]))

    pm10_standard, pm25_standard, pm100_standard, pm10_env, pm25_env, pm100_env, particles_03um, particle

    check = sum(buffer[0:30])

    if check != checksum:
        continue
    print("Concentration Units (standard)")
    print("---------------------------------------")
    print("PM 1.0: %d\tPM2.5: %d\tPM10: %d" % (pm10_standard, pm25_standard, pm100_standard))
    print("Concentration Units (environmental)")
    print("---------------------------------------")
    print("PM 1.0: %d\tPM2.5: %d\tPM10: %d" % (pm10_env, pm25_env, pm100_env))
    print("---------------------------------------")
    print("Particles > 0.3um / 0.1L air:", particles_03um)
    print("Particles > 0.5um / 0.1L air:", particles_05um)
    print("Particles > 1.0um / 0.1L air:", particles_10um)
    print("Particles > 2.5um / 0.1L air:", particles_25um)
    print("Particles > 5.0um / 0.1L air:", particles_50um)
    print("Particles > 10 um / 0.1L air:", particles_100um)
    print("---------------------------------------")

    buffer = buffer[32:]
    #print("Buffer ", buffer)
```

Then open up the REPL to see the data printed out nicely for you!



Downloads

Files:

- PMS5003 Datasheet / Manual