PMS5003 Particulate Sensor

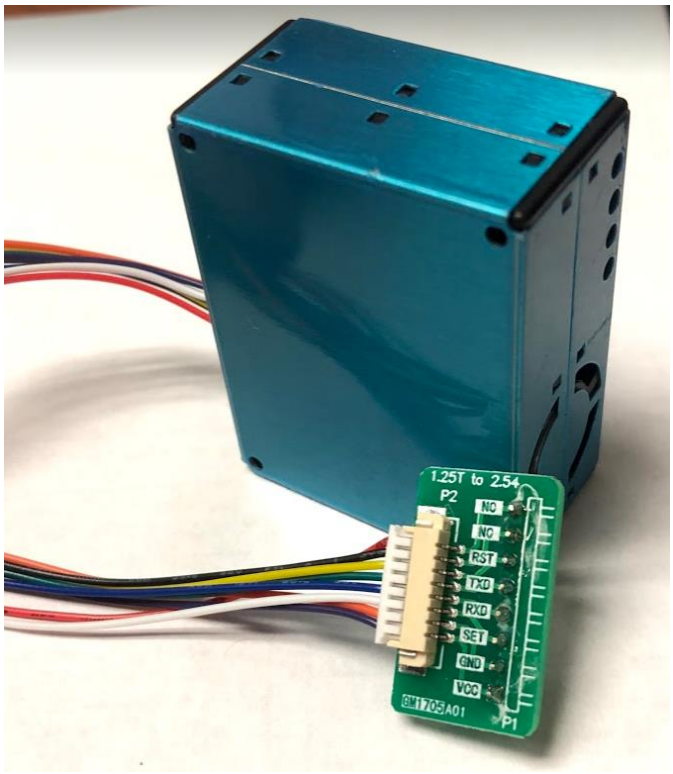

Figure 1: PMS5003 Sensor

The PMS5003 particulate sensor is a laser scattering sensor that gives digital outputs of the concentration of suspended particles in the air. It can detect a particle diameter down to 0.3 µm. Other characteristics that make the PMS5003 desirable include the real-time responses, high anti-interference performance, and optional direction of air inlet and outlet flow.[1] This walkthrough will describe how the sensor functions and how to set up the sensor for use on a Raspberry Pi.

How does the PMS5003 work?

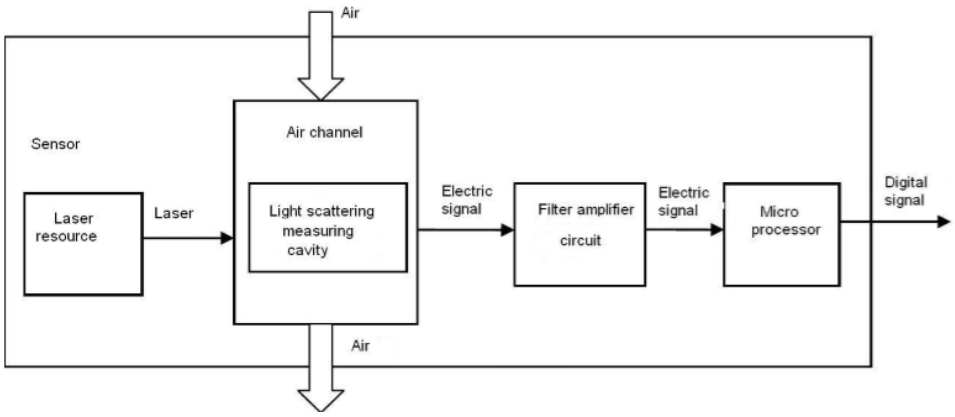

Figure 2: Flow of laser scattering in PMS5003

This sensor analyzes duct particles with the use of laser scattering. Laser scattering radiates light off the particles in the air and obtains the scattering pattern of the emitted light.[1] The emitted light can travel in multiple directions (forwards, backwards, up, down, left, and right) depending on the size of the particle irradiated. These "patterns" are referred to as the light intensity distribution pattern.[2] Following figure 2 above, the sensor emits a laser into the cavity in which a fan blows surrounding air threw. The scattered light from the particles is converted into an electrical signal. This signal is then sent to an amplifier to enhance the signal for processing using the MIE theory in the microprocessor. The digital output is then given in units of diameter / unit volume.[1]

MIE theory is the solution of the Maxwell equations for the scattering of electromagnetic waves on spherical particles. A few notes about the MIE theory: "particles are opaque discs, light is scattered at narrow angles, all particles have the same efficiency when scattering light, and the refractive index difference between the particle and surrounding medium is infinite."[3]

The PMS5003 digital output has two optional modes. The default mode is the active mode. This mode sends the serial data automatically. If there are no changes in the particle concentration the active mode reads every 2.3 seconds. Once a substantial change in concentration has been detected a fast mode is activated and the sensor will read every 200~800 milliseconds.[1]

Connected the PMS5003

Figure 3: Connecting board for the PMS5003

The PMS5003 unit comes with a board that allows for easy attachment to the raspberry pi breadboard. Very little wiring is needed to connect the PMS5003 to the pi unit. It communicates through the serial port utilizing the TXD and RXD ports on the cobbler. Once the unit is attached to the supplied board and the board is connected to the breadboard, connect the unit to the raspberry pi as follows:

Connect **Pi 5.0V** to **PMS VOU**.
Connect **Pi GND** to **PMS GND**.
Connect **Pi TXD** to **PMS5003 RXD**.
Connect **Pi RXD** to **PMS5003 TXD**.

After the everything is connected, the serial port needs to be enabled. This is done from the command line on the raspberry pi as follows:

Enter sudo raspi-config into command line

Go to the interfacing option and select serial
Disable the serial login shell and enable the serial port

Now the sensor is ready for analysis.

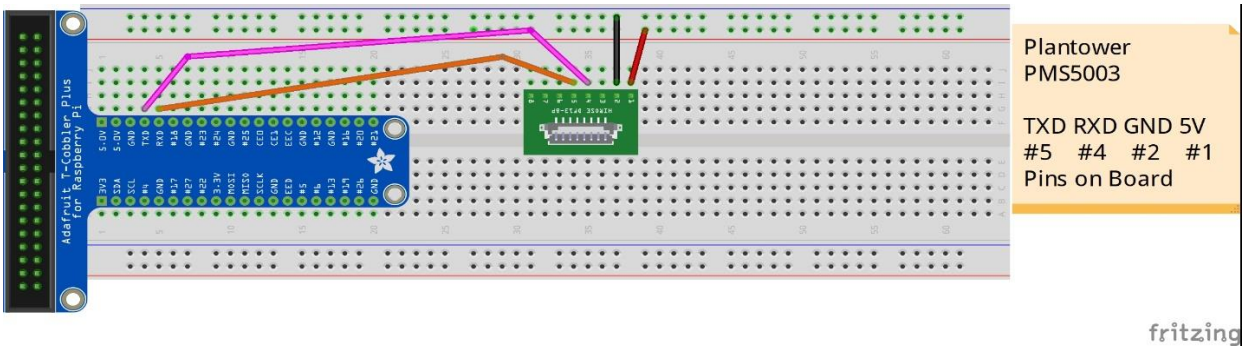

Figure 4: Connecting PMS5003

Coding the PMS5003

Coding for this sensor is based off the active mode that starts once the sensor is powered. If using a Raspberry Pi 3, note that the serial port is now /dev/ttyS0.[4]

```
import serial
import datetime
import time

#serial port definition for Raspi 3
port = serial.Serial('/dev/ttyS0', baudrate=9600, timeout=2.0)
```

These are the libraries needed to run the PMS5003 and to calculate the concentrations of the particles' sizes. (These libraries should already be installed on Raspberry Pi when purchased). In order to obtain the digital output values, the port location is set up in the code. As mentioned above, the serial port is /dev/ttyS0 for a raspberry pi 3. From the data sheet, the baudrate is set to 9600 bps.

```
def read_pm_line(_port):
    rv = b''
    while True:
        ch1 = _port.read()
        if ch1 == b'\x42':
            ch2 = _port.read()
            if ch2 == b'\x4d':
                rv += ch1 + ch2
                rv += _port.read(28)
                return rv
```

This part of the codes makes sure the start characters 1 and 2 are reading correctly in the serial port. Character 1 (ch1) is fixed at 0x42 and character 2 (ch2) is fixed at 0x4d. These start

characters ensure that the data is being read properly from the serial port, they correlate with the hexadecimal values the pi is reading as data.

```
while True:
    try:
        rcv = read_pm_line(port)
        res = {'timestamp': datetime.datetime.now(),
               'apm10': rcv[4] * 256 + rcv[5],
               'apm25': rcv[6] * 256 + rcv[7],
               'apm100': rcv[8] * 256 + rcv[9],
               'pm10': rcv[10] * 256 + rcv[11],
               'pm25': rcv[12] * 256 + rcv[13],
               'pm100': rcv[14] * 256 + rcv[15],
               'gt03um': rcv[16] * 256 + rcv[17],
               'gt05um': rcv[18] * 256 + rcv[19],
               'gt10um': rcv[20] * 256 + rcv[21],
               'gt25um': rcv[22] * 256 + rcv[23],
               'gt50um': rcv[24] * 256 + rcv[25],
               'gt100um': rcv[26] * 256 + rcv[27]
              }
```

This part of the code is taking each piece of the serial stream (16 byte stream) and converting it from the hexadecimal values to the output values. The apm10, 25, and 100 are the factory calibration settings. PM10, 25, and 100 are the generic atmospheric conditions for calibration. The gt03, 05, 10, 25, 50, 100 are the number of particles with certain diameter readings. (all units are unit ug/m^3). The rcv values are the high and low bit values obtained from the serial port. The [#] are the corresponding placement in the data stream. The first high 8 bit reading is multiplied by 256 to shift the bit value over by 8 places to the left in order to add it to the lower 8 bit value reading. This is also the same as << 8.

```
        print('===============\n'
              'PM1.0(CF=1): {}\n'
              'PM2.5(CF=1): {}\n'
              'PM10 (CF=1): {}\n'
              'PM1.0 (STD): {}\n'
              'PM2.5 (STD): {}\n'
              'PM10  (STD): {}\n'
              '>0.3um     : {}\n'
              '>0.5um     : {}\n'
              '>1.0um     : {}\n'
              '>2.5um     : {}\n'
              '>5.0um     : {}\n'
              '>10um      : {}'.format(res['apm10'], res['apm25'], res['apm100'],
                                       res['pm10'], res['apm25'], res['pm100'],
                                       res['gt03um'], res['gt05um'], res['gt10um'],
                                       res['gt25um'], res['gt50um'], res['gt100um']))
        time.sleep(1)
    except KeyboardInterrupt:
        break
```

The rest of the code simply prints the values obtained above with the correct units, units g/m^3 in 0.1 L air. The sensor will also loop every second.

References

(1) plantower-pms5003-manual_v2-3.pdf, https://cdn-shop.adafruit.com/product-files/3686/plantower-pms5003-manual_v2-3.pdf, (accessed Feb 15, 2009).

(2) Particle Size Distribution Measurement by the Laser Diffraction/Scattering Method Part1 : SHIMADZU (Shimadzu Corporation) https://www.shimadzu.com/an/powder/support/middle/m01.html (accessed Feb 15, 2019).

(3) Mie-Theory-the-first-hundred-years-MRK1304-02.pdf, https://www.atascientific.com.au/wp-content/uploads/2017/02/Mie-Theory-the-first-hundred-years-MRK1304-02.pdf, (accessed Feb 15, 2019).

(4) 262588213843476. Raspberry Pi and PMS5003 https://gist.github.com/lackofdream/bc8649d647240ef14311 (accessed Feb 15, 2019).