

Connecting Sensors to Pi

Kyle McCarty and Marc Los Huertos*

October 20, 2020

Contents

1	Introduction	1
1.1	What is particulate matter?	1
2	PMS5003 Particulate matter sensor	1
2.1	Sensor Purchase	1
2.2	Sensor Construction and Operation	2
2.3	Sensor IO	2
2.4	Wiring	2
2.5	Code and Testing	2
2.6	Python Code	4
3	MQ Sensors	7
3.1	Additional Sensors	7

1 Introduction

1.1 What is particulate matter?

PM stands for particulate matter (also called particle pollution). Particulate Matter is a mixture of solid particles and liquid droplets found in the air. Some particles, such as dust, dirt, soot, or smoke, are large or dark enough to be seen with the naked eye. Others are so small they can only be detected using an electron microscope.

Particle pollution can be classified by size:

PM10 inhalable particles, with diameters that are generally 10 micrometers and smaller; and

*Acknowledgments: Much of the project is based on summer research conducted by Anna Burns.

PM2.5 fine inhalable particles, with diameters that are generally 2.5 micrometers and smaller.

NOTE: The average human hair is about 70 micrometers in diameter — making it 30 times larger than the largest fine particle.

2 PMS5003 Particulate matter sensor

2.1 Sensor Purchase

The Digital Particle Concentration Laser Sensor Plantower PMS5003 PM2.5 PM10 is produced in China. With 25 purchases, they cost about \$23 each.

2.2 Sensor Construction and Operation

2.3 Sensor IO

This sensor comes with eight female-to-female connectors as well as an adapter for the Raspberry Pi, and a connector for the Raspberry Pi and the adapter itself.

2.4 Wiring

1. Begin by connecting the sensor to the adapter. Take the colorful wires with white blocks on either end and attach them to both the sensor and the small adapter board; they attach with the “shiny” side up, and you need to give them quite a push to get them to go in fully. They should be snug, and not at all loose.
2. Next, we’ll connect some breadboard jumpers (small wired without black rectangles). Connect the positive rail to A30 and the ground rail to A29.
3. Attach the PMS5003 adapter to the Breadboard – B23-B30, where the B30 aligns with the 5 volt power.
4. Connect 2 male-to-female connectors (the ones with the black rectangles on either end):
 - Connect the wire attached to the VCC terminal to PIN 2 on the Raspberry Pi – or connect to the 5V rail on the breadboard.
 - Connect the wire attached to the GND terminal to PIN 9 on the Raspberry Pi – or connect to the ground rail on the breadboard.
 - Connect the wire attached to the row corresponding to the RXD (recieve) terminal (d27) to TXD (transmit) PIN 8 on the Raspberry Pi.
 - Connect the wire attached to the TXD (transmit) terminal to RXD (recieve) PIN 10 on the Raspberry Pi.

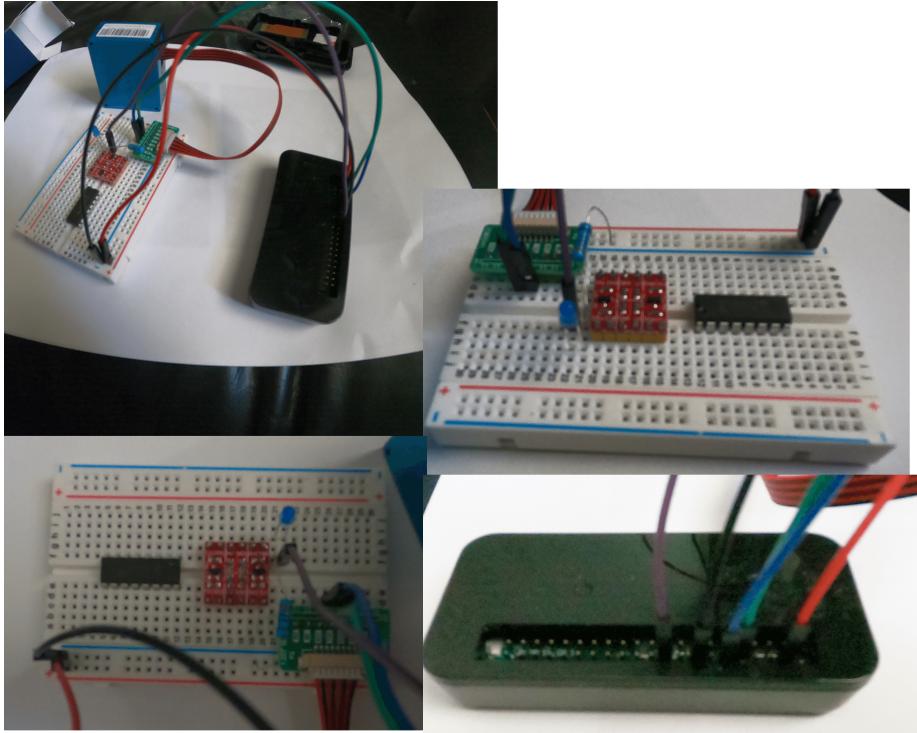


Figure 1: Wiring for PMS5003, LED, and Pi for deployment. I put the case top in the Pi to create a realively clean set up. The LED is programmed to flash every reading (1/minute).

- connect the resistor from the ground rail to the i21 on the breadboard.
 - connect the LED from the j21 to j22 (short wire in j21).
 - connect the LED (i22) to GPIO 24 (PIN 18).
5. Double check the wiring with the following images (Figure 1).

2.5 Code and Testing

1. Connect your Pi to power and PMS5003 fan should start.
2. VNC to the Pi.
3. Check to make sure that it is working. If not shut the Pi down (via the command ‘sudo shutdown’ and check the wiring.
4. Enter “sudo raspi-config”.
5. Select Option 5 (“Interfacing Options”).

6. Enable ARM I2C by selecting it in this menu and then pressing Yes.
7. Then to enable Serial, select Option 5 (“Interfacing Options”) again and choose Serial on the menu.
8. A window will pop up asking, “would you like a login shell to be accessible over serial?”, and the answer is NO.
9. Next, a window will pop up asking, “Would you like the serial port hardware to be enabled?”, this time say YES, and exit the window.
10. Click Finish, you may be prompted to reboot, and select YES.
11. We will also need to download a pre-made communication tool. Using the command window, type “

2.6 Python Code

We have made the code available on the EJnPi Respository, so you can download it directly from the web onto your Pi or follow the steps below:

1. Using the Pi’s webbrowser, navigate to the EJnPI Respository.
2. When you can get the files and “Blame” in Github to view the Air_quality_code_v2.py, then click to save on the home directory.

Then open Thorney and load the file.

```

1 # New Version — PMS and Air Quality Template
2 # Import packages (which sensor uses which package is noted)
3
4 # Datetime
5 import datetime
6 #import spidev # MiCS-2714
7 #import os # MiCS-2714
8 import serial # PMS5003
9 from collections import OrderedDict # PMS5003
10 #import board # BME280
11 #import busio # BME280
12 #import adafruit_bme280 # BME280
13 import csv # All
14 import time # All
15 import RPi.GPIO as GPIO # For LED
16
17 GPIO.setmode(GPIO.BCM) # For LED
18 GPIO.setwarnings(False) # For LED
19 GPIO.setup(24,GPIO.OUT,initial=GPIO.LOW) # For LED
20
21
22 # MiCS-2714 Code
23 #spi = spidev.SpiDev()
24 #spi.open(0,0)
25 #spi.max_speed_hz=1000000
26
27 #def ReadChannel(channel):

```

```

28 #  adc = spi.xfer2([1,(8+channel)<<4,0])
29 #  data = ((adc[1]&3) << 8) + adc[2]
30 #  return data
31
32 #def ConvertVolts(data,places):
33 #  volts = (data * 3.3) / float(1023)
34 #  volts = round(volts,places)
35 #  return volts
36
37 #def ConvertNO2(data,places):
38 #  NO2 = ((data * 10.05)/float(1023))+0.05
39 #  NO2 = round(NO2,places)
40 #  return NO2
41
42 #NO2_channel = 0
43
44 # MQ-131 Code
45
46 #def ConvertO3(data,places):
47 #  O3 = ((data * 1001)/float(1023))+1
48 #  O3 = round(O3,places)
49 #  return O3
50
51 #O3_channel = 1
52
53 # PMS5003 Code
54 class Sensor():
55     def __init__(self, tty = '/dev/ttyS0'):
56         self.tty = tty
57
58     def open(self):
59         self.serial = serial.Serial(self.tty, baudrate = 9600)
60
61     def close(self):
62         self.serial.close()
63
64     def read_bytes(self, data, idx, size = 2):
65         return int("".join(data[idx : idx + size]), 16)
66
67     def read(self):
68         data = self.serial.read(32)
69         data = ["{:02X}".format(d) for d in data]
70
71         if data[0] != '42' or data[1] != '4D':
72             return None
73
74         res = OrderedDict()
75         res['DateTime'] = datetime.datetime.now()
76         res['pm1_cf'] = self.read_bytes(data, 4)
77         res['pm25_cf'] = self.read_bytes(data, 6)
78         res['pm10_cf'] = self.read_bytes(data, 8)
79         res['pm1'] = self.read_bytes(data, 10)
80         res['pm25'] = self.read_bytes(data, 12)
81         res['pm10'] = self.read_bytes(data, 14)
82         return res
83
84 # BME280 Code

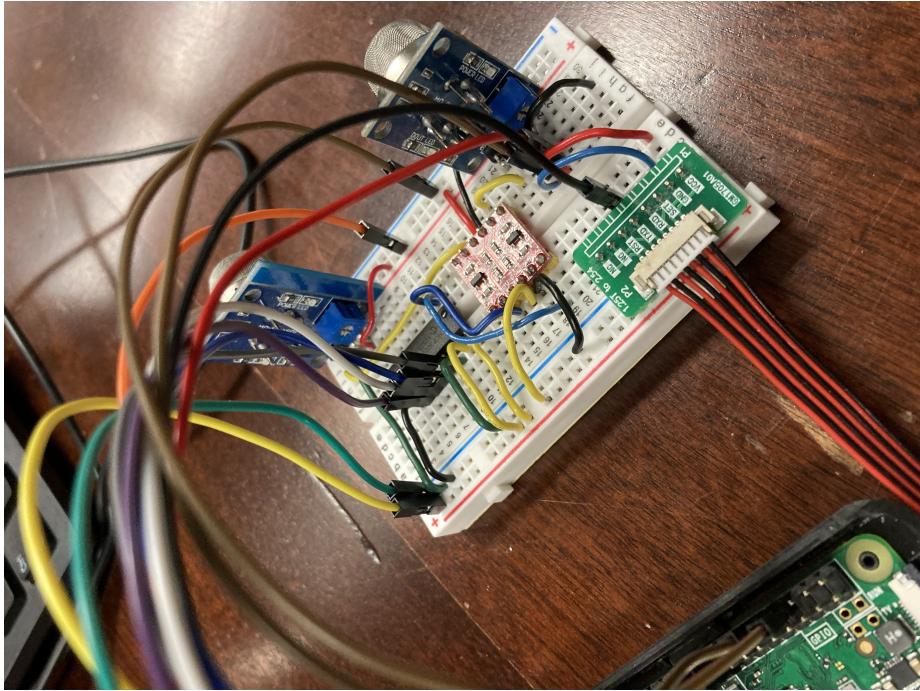
```

```

85 #i2c = busio.I2C(board.SCL, board.SDA)
86 #bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)
87 #bme280.sea_level_pressure = 1013.25
88
89 # CSV Code
90 def write_to_csv():
91     if __name__ == "__main__":
92         with open('/home/pi/airquality_data.csv', mode
93 = 'a') as csv_file:
94             csv_writer = csv.writer(csv_file)
95             csv_writer.writerow([data])
96         with open('/home/pi/airquality_data.csv', mode
97 = 'r') as csv_file:
98             csv_reader = csv.reader(csv_file)
99             for row in csv_reader:
100                 print(row)
101
102 # Command
103 if __name__ == "__main__":
104     while True:
105         GPIO.output(24,GPIO.HIGH)
106         # MiCS-2714 output
107         #NO2_level = ReadChannel(NO2_channel)
108         #NO2_volts = ConvertVolts(NO2_level,2)
109         #NO2      = ConvertNO2(NO2_level,2)
110         # MQ-131 output
111         #O3_level = ReadChannel(O3_channel)
112         #O3_volts = ConvertVolts(O3_level,2)
113         #O3      = ConvertO3(O3_level,2)
114         # PMS5003 output
115         sensor = Sensor()
116         sensor.open()
117         data = sensor.read()
118         sensor.close()
119         # Write to csv, sleep for 60 seconds
120         write_to_csv()
121         time.sleep(2)
122         GPIO.output(24,GPIO.LOW)
123         time.sleep(58)

```

3. Open the file using Thorny and hit run. You should see the results being displayed in the lower window.
4. It should release a reading of the different measurements specified at intervals of 60 seconds, and compile the data into a CSV file.



3 MQ Sensors

3.1 Additional Sensors

We have provided the electronics to evaluate other air quality parameters using MQ Sensors. You can read about how these sensors work and a list of types of sensors in this sensor guide.

