Connecting Sensors to Pi

Kyle McCarty and Marc Los Huertos*

October 18, 2020

Contents

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

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

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

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, connect the adapter to the Raspberry Pi. Separate four of the female-to-female connectors (the ones with the black rectangles on either end) and attach one of the ends to the TXD, RXD, GND and VCO terminals on the adapter (these correspond to the 4th, 5th, 7th, and 8th terminals from the top).

    - Connect the wire attached to the VCC terminal to PIN 2 on the Raspberry Pi – or connect to the 5V rail on the breadboad.
    - 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 RXD terminal to PIN 8 on the Raspberry Pi.
    - Connect the wire attached to the TXD terminal to PIN 10 on the Raspberry Pi.

3. Double check the wiring with the following images

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 the files and "Blame" in Github to view the pm25_simpletest.py, then click to save on the home directory.

   Then open Thorny and load the file.

```
"""
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
import adafruit_pm25

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
pm25 = adafruit_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 = adafruit_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("-------------------------------------------")
```

Open the code in XYZ and add the following comment lines:

import serial These are the pre-coded Github packages that this code uses; serial allows python to run on different back-end connections. from collections import OrderedDict Imports the ordered dictionary function import csv Package that allows you to compile to a csv file import time Allows you to define time intervals

class Sensor(): Defining what you are gathering data from self.tty = tty

def open(self): Opening the connection between the sensor and the Pi self.serial = serial.Serial(self.tty, baudrate = 9600) Baudrate is the signal rate between the sensor and Pi

def close(self): Closing the connection between the sensor and the Pi self.serial.close()

return int("".join(data[idx : idx + size]), 16)

def read(self): data = self.serial.read(32) data = [":02X".format(d) for d in data]

if data[0] != '42' or data[1] != '4D': return None

res = OrderedDict() This code returns an ordered dictionary of the following elements:

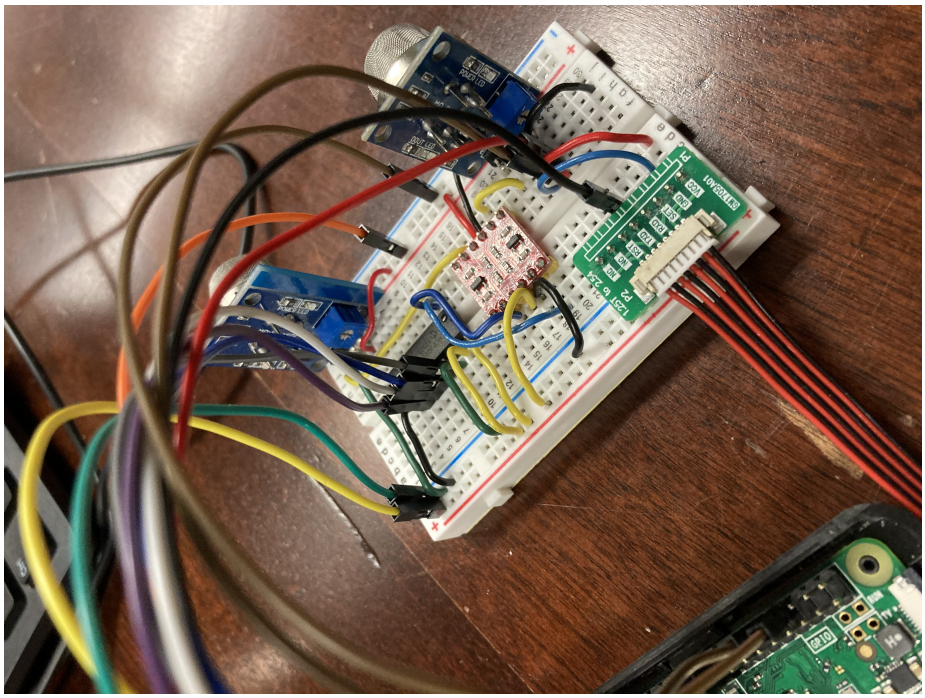
return res

print(row)

while True: Creates a loop so that the code will read in intervals as defined below. "' Test code "' sensor = Sensor() Defining the sensor as above sensor.open() Opening the sensor data = sensor.read() Reading data from the sensor sensor.close() Closing the sensor print(data) Printing the data to the terminal time.sleep(10) Establishing that the data will be read in ten second intervals (can be changed).

3. Re-enter the nano file you created above, and right click anywhere within the file to bring up a menu; select the "paste" option. This should place the code above into the file without altering the spacing or indentations.

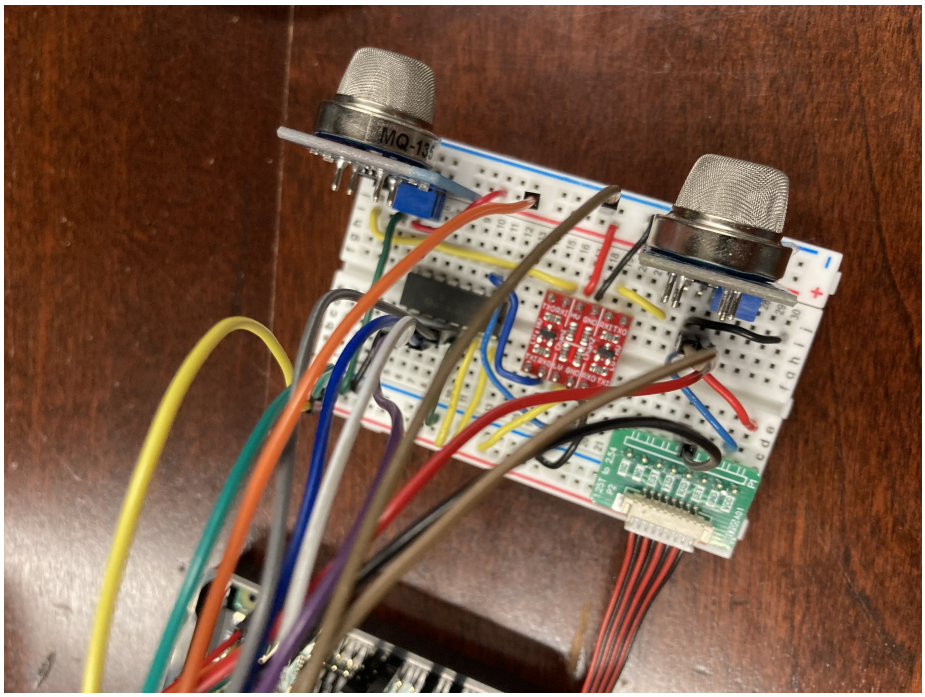
4. Save the text document with ctrl o (it will ask it what you want to name it, you can leave it as pms5003.py), and then exit it with ctrl x.

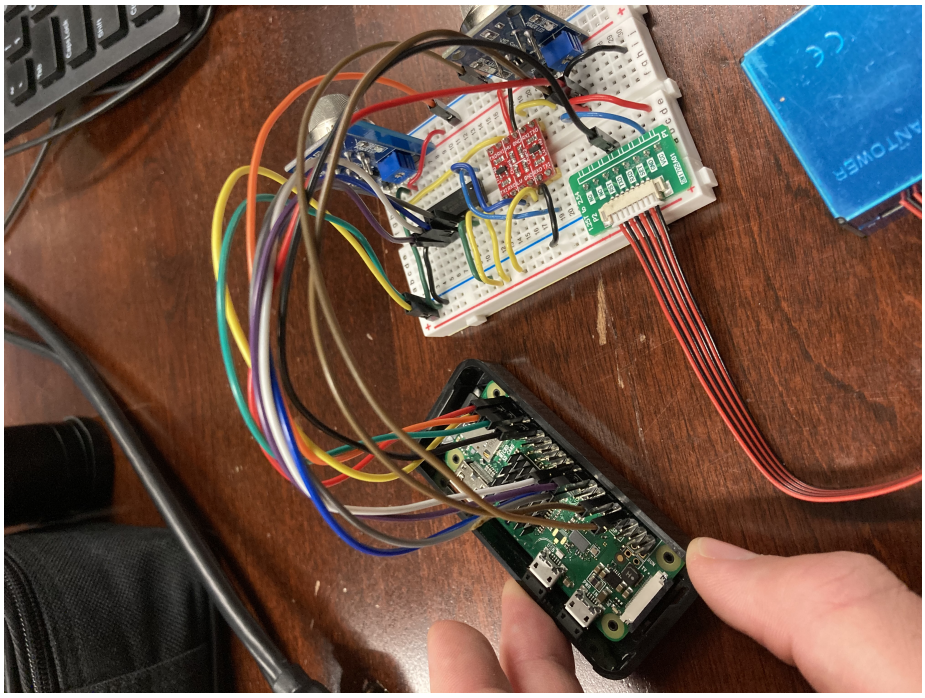


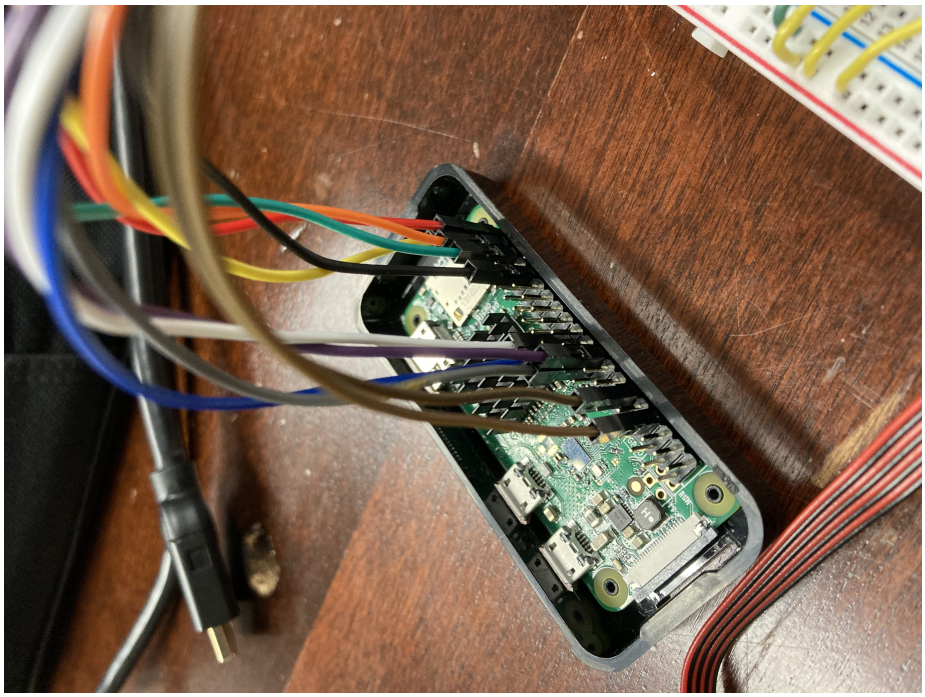
5. Now within the main shell, type "python3 pms5003.py". When I submitted the text, I got one of two responses – some error, usually about some indentation problem that took way too long to sort out OR it seem to quietly be in standby mode. I think it was sending the program to my Pi, perhaps compiling and sending the commands to the sensor.

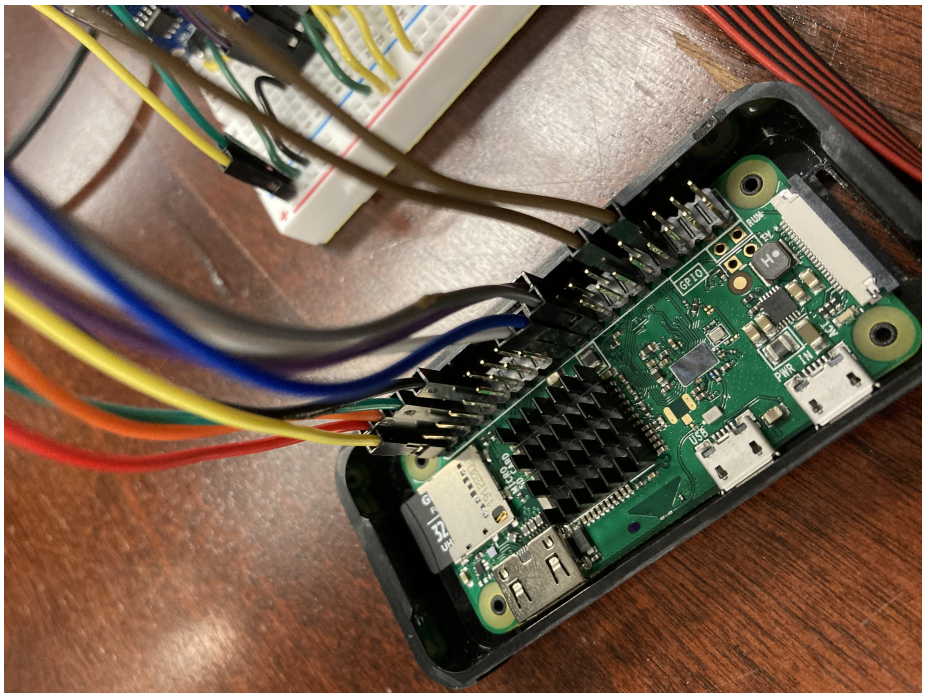
It should release a reading of the different measurements specified at intervals of ten seconds, and compile the data into a CSV file.

3 MQ Sensors








GPIO
PWR IN
USB