PM Sensors

Anna Madison Burns

6/10/2020

Our sensor

We are using a Digital Particle Concentration Laser Sensor PMS5003 PM2.5 PM10+Cable for Arduino from Amazon, which costs 38.75 USD. This sensor comes with eight female-to-female connectors as well as an adapter for Arduino or Raspberry Pi, and a connector for the Raspberry Pi and the adapter itself.

Helpful resource

The guide below was compiled from an instructional Youtube video published on 11 January 2019 by Chengyi Wu. While the video is not in English, it is easy to follow along both with the physical construction and code for the PMS5003: https://www.youtube.com/watch?v=B0bzyEnjZY4

Physically connecting the sensor to Raspberry Pi

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; 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).

2a. Connect the wire attached to the VOC terminal to node 2 on the Raspberry Pi.

2b. Connect the wire attached to the GND terminal to node 14 on the Raspberry Pi.

2c. Connect the wire attached to the RXD terminal to node 8 on the Raspberry Pi.

2d. Connect the wire attached to the TXD terminal to node 10 on the Raspberry Pi.

3. Connect your Pi to power, and your sensor is good to go!

Coding the PM10 sensor into Raspberry Pi

I did this entire step within the Raspberry Pi terminal interface.

Preliminary steps

1. Enter “sudo raspi-config”.
2. Select option 5 (“Interfacing Options”).
3. Enable ARM I2C by selecting it in this menu and then pressing Yes.
4. Then to enable Serial, choose Serial on the menu.
5. A window will pop up asking, “would you like a login shell to be accessible over serial?”, and the answer is NO.
6. 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.
7. Click Finish, you may be prompted to reboot, and select YES.

Code

1. Open up a text document in nano by typing “nano pms5003.py” into the bar in the terminal (pms5003.py will be our document name).
2. Within the text document, enter the following code:

```
import serial
from collections import OrderedDict
class Sensor():
  def __init__(self, tty = '/dev/ttyS0'):
    self.tty = tty

  def open(self):
    self.serial = serial.Serial(self.tty, baudrate = 9600)

  def close(self):
    self.serial.close()

  def read_bytes(self, data, idx, size = 2):
    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' # Maybe lowercase d? No, the  code won't run with a lowercase
      return None

    res = OrderedDict()
    res['pm1_cf'] = self.read_bytes(data, 4)
    res['pm25_cf'] = self.read_bytes(data, 6)
    res['pm10_cf'] = self.read_bytes(data, 8)
    res['pm1'] = self.read_bytes(data, 10)
    res['pm25'] = self.read_bytes(data, 12)
    res['pm10'] = self.read_bytes(data, 14)
```

```
    res['temperature'] = self.read_bytes(data, 24) / 10
    res['humidity'] = self.read_bytes(data, 26) / 10
    return res # Indentation issues? Make sure the return res line is inside of the def read(self) line

if __name__ == '__main__':
  '''
  Test code
  '''
  sensor = Sensor()
  sensor.open()
  data = sensor.read()
  sensor.close()
  print(data)
```

A few notes on this Python code:

- Pay very close attention to spacing and indentations! This is very, very important in Python. The code will not run if the indentations are at all off.

3. 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.
4. Now within the main shell, type "python3 pms5003.py". It should release a reading of the different measurements specified.

Still to come

- Storing the data somewhere that we can see variability (perhaps Adafruit IO? Info (for a different PM sensor): https://www.raspberrypi.org/blog/monitor-air-quality-with-a-raspberry-pi/)
- Getting the Pi to automatically read data when it is on, rather than with a code (might also be information in the above resource)
- Checking for accuracy (why divide the temperature/humidity by ten?)
- Spacing on code is weird... how to fix? Maybe within Latex?
- Getting it to read onto a Box CSE file ?