Connecting Sensors to Pi

Kyle McCarty and Marc Los Huertos*

October 20, 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

*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, 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 Air_quality_code_v2.py, then click to save on the home directory.

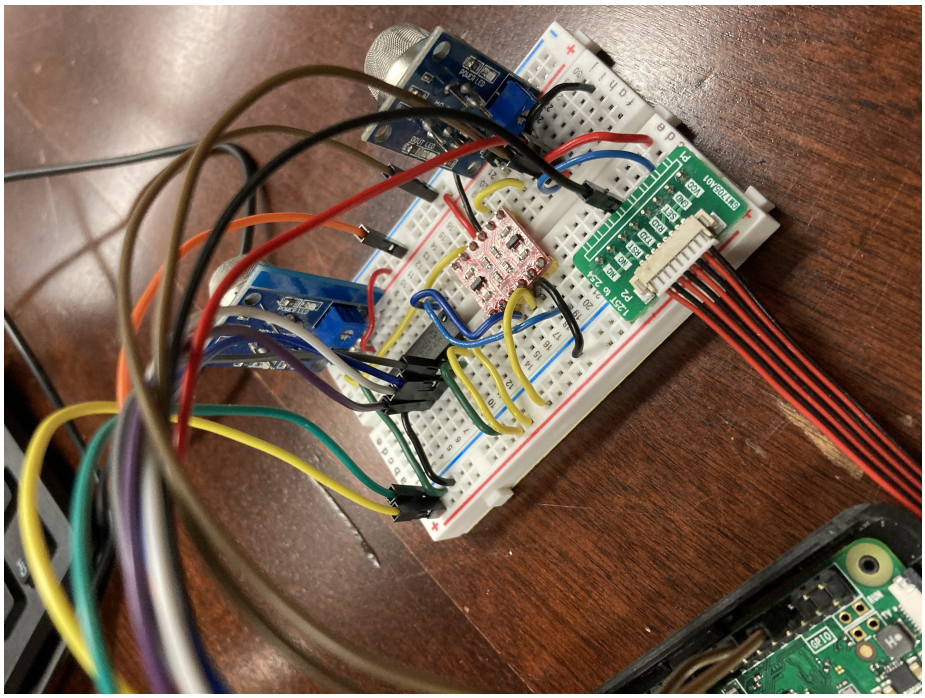
   Then open Thorny and load the file.

```
# New Version -- PMS and Air Quality Template
# Import packages (which sensor uses which package is noted)

# Datetime
import datetime
#import spidev # MiCS-2714
#import os # MiCS-2714
import serial # PMS5003
from collections import OrderedDict # PMS5003
#import board # BME280
#import busio # BME280
#import adafruit_bme280 # BME280
import csv # All
```

```
import time # All
import RPi.GPIO as GPIO # For LED

GPIO.setmode(GPIO.BCM) # For LED
GPIO.setwarnings(False) # For LED
GPIO.setup(24,GPIO.OUT, initial=GPIO.LOW) # For LED


# MiCS-2714 Code
#spi = spidev.SpiDev()
#spi.open(0,0)
#spi.max_speed_hz=1000000

#def ReadChannel(channel):
#  adc = spi.xfer2([1,(8+channel)<<4,0])
#  data = ((adc[1]&3) << 8) + adc[2]
#  return data

#def ConvertVolts(data,places):
#  volts = (data * 3.3) / float(1023)
#  volts = round(volts,places)
#  return volts

#def ConvertNO2(data,places):
#  NO2 = ((data * 10.05)/float(1023))+0.05
#  NO2 = round(NO2,places)
#  return NO2

#NO2_channel = 0

# MQ-131 Code

#def ConvertO3(data,places):
#  O3 = ((data * 1001)/float(1023))+1
#  O3 = round(O3,places)
#  return O3

#O3_channel = 1

# PMS5003 Code
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':
      return None

    res = OrderedDict()
    res['DateTime'] = datetime.datetime.now()
    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)
    return res

# BME280 Code
#i2c = busio.I2C(board.SCL, board.SDA)
#bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)
#bme280.sea_level_pressure = 1013.25

# CSV Code
def  write_to_csv():
        if __name__ == "__main__":
                with open('/home/pi/airquality_data.csv', mode
    ='a') as csv_file:
                        csv_writer = csv.writer(csv_file)
                        csv_writer.writerow([data])
                with open('/home/pi/airquality_data.csv', mode
    ='r') as csv_file:
                        csv_reader = csv.reader(csv_file)
                        for row in csv_reader:
                                print(row)


# Command
if __name__ == "__main__":
  while True:
    GPIO.output(24,GPIO.HIGH)
    # MiCS-2714 output
    #NO2_level = ReadChannel(NO2_channel)
    #NO2_volts = ConvertVolts(NO2_level,2)
    #NO2       = ConvertNO2(NO2_level,2)
    # MQ-131 output
    #O3_level = ReadChannel(O3_channel)
    #O3_volts = ConvertVolts(O3_level,2)
    #O3       = ConvertO3(O3_level,2)
    # PMS5003 output
    sensor = Sensor()
    sensor.open()
    data = sensor.read()
    sensor.close()
    # Write to csv, sleep for 60 seconds
    write_to_csv()
    time.sleep(2)
    GPIO.output(24,GPIO.LOW)
    time.sleep(58)
```

3. Open the file using Thorny and hit run. You should see the results being



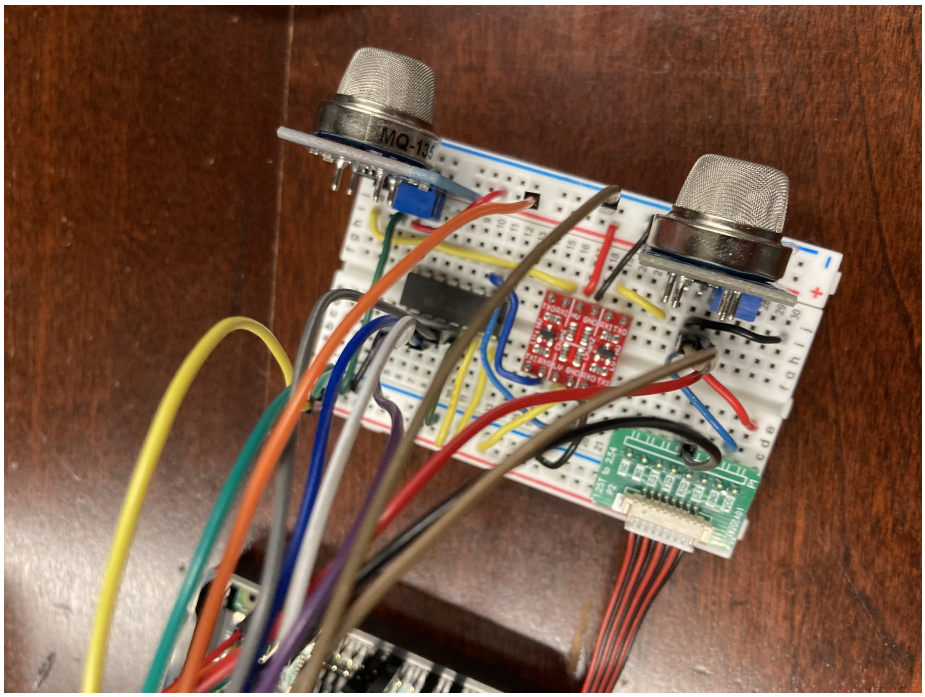
displayed in the lower window.

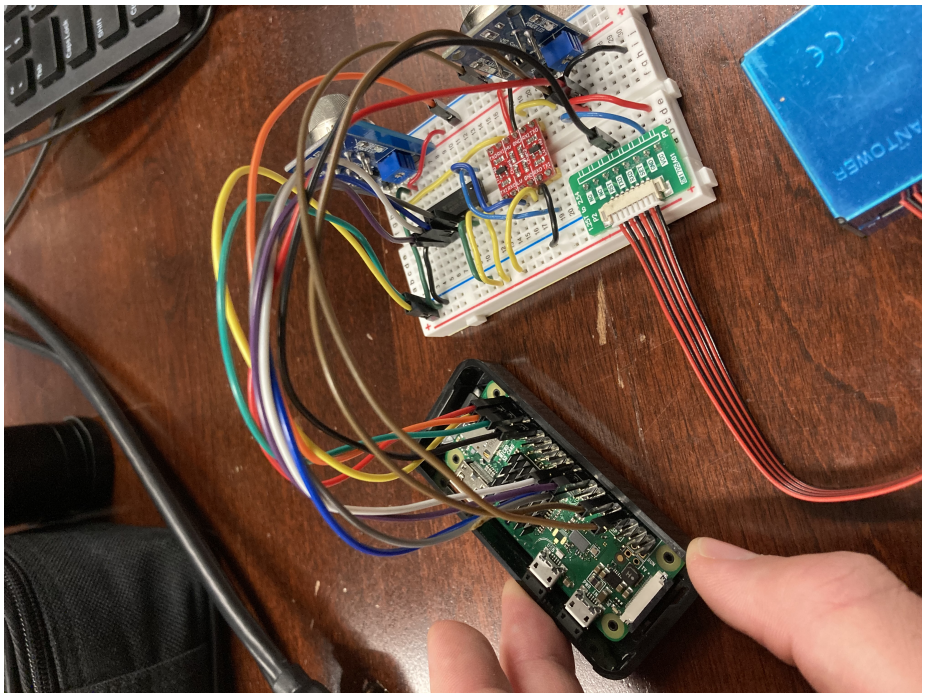
4. It should release a reading of the different measurements specified at intervals of 60 seconds, and compile the data into a CSV file.

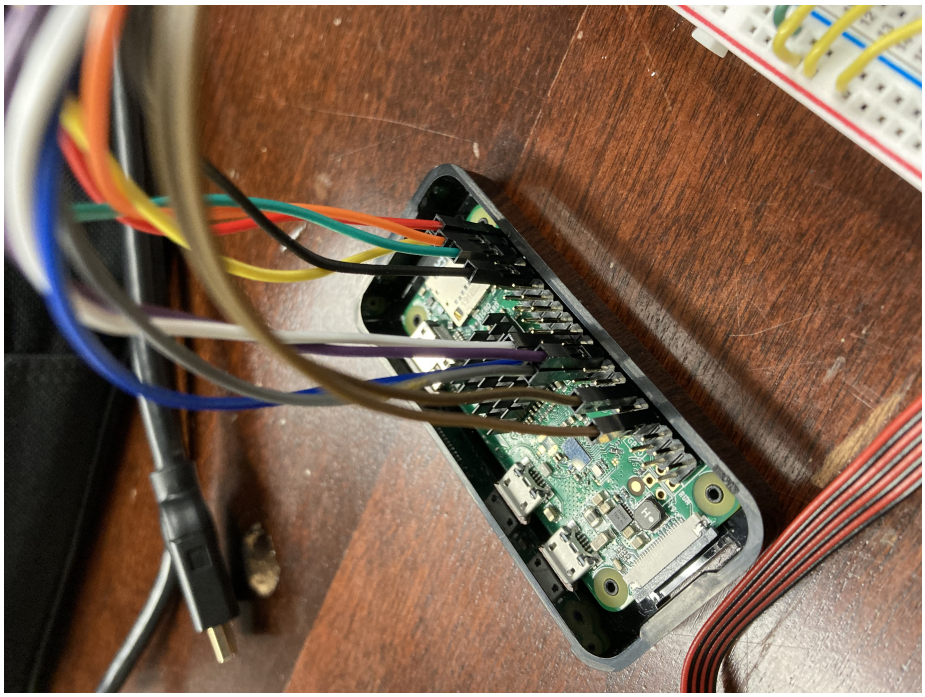
3 MQ Sensors

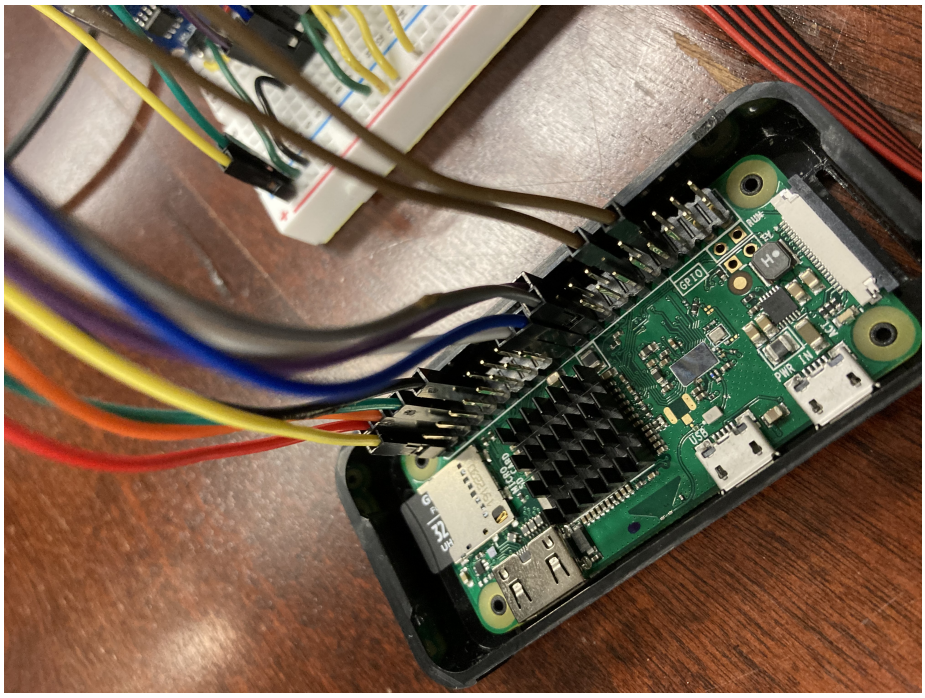
3.1 Additional Sensors

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.








GPIO
PWR IN
USB