Air Quality Monitor and Case

Created by Ruiz Brothers

https://learn.adafruit.com/qa-case

Last updated on 2023-08-29 04:54:06 PM EDT
# Table of Contents

## Overview
- Build an AQI monitor
- CO2 + AQI
- Adafruit IO + WipperSnapper
- Parts

## Circuit Diagram

## Install CircuitPython
- CircuitPython Quickstart

## Coding the AQI Case
- Upload the Code and Libraries to the ESP32-S3 Reverse TFT Feather
- How the CircuitPython Code Works
- Sensor Setup
- Functions
- Display Setup
- Sensor Text
- Arrays and Ticks
- The Loop

## 3D Printing
- Parts List
- Slice with settings for PLA material
- Supports

## Assemble
- Secure PMSA
- NeoPixel JST
- Stemma cable
- Secure Feather
- Secure NeoPixel stick
- Attach Sensor Bracket
- Secure Sensor
- Place in Case
- Attach Screen frame
- Press fit Feet
- Wrist Band
- Complete
Overview

Build an AQI monitor
You can build a compact air quality monitor with 3D Printing and CircuitPython.

Powered by the Adafruit Feather ESP32-S2 Reverse TFT and a PMSA003I air sensor, this build can monitor air quality in real time! You can even read CO2, humidity and temperatures with a SCD4x sensor.

The 3D printed case houses all the components to make a portable environmental monitor that you can even wear!

CO2 + AQI
The easy to read interface displays AQI levels along side three colored squares to visually show air levels.

NeoPixel LEDs mirror green, yellow and red AQI levels.

Display CO2, humidity and even temperatures in Celsius or Fahrenheit!
Adafruit IO + WipperSnapper
This project is also compatible with WipperSnapper, Adafruit’s no-code solution to IoT Projects.

With WipperSnapper, you can easily set up sensors and start logging data.

Adafruit’s WipperSnapper has support for the PMSA, so you can build projects without having to write any code.

Quick Start: WipperSnapper

Parts
Adafruit ESP32-S2 Reverse TFT Feather
Like Missy Elliott, we like to "put our [Feather] down, flip it and reverse it" and that's exactly what...
https://www.adafruit.com/product/5345

Adafruit PMSA003I Air Quality Breakout
Breathe easy, knowing that you can track and sense the quality of the air around you with this Adafruit PMSA003I Air Quality Breakout. This sensor is great for...
https://www.adafruit.com/product/4632

NeoPixel Stick - 8 x 5050 RGB LED with Integrated Drivers
Make your own little LED strip arrangement with this stick of NeoPixel LEDs. We crammed 8 of the tiny 5050 (5mm x 5mm) smart RGB LEDs onto a PCB with mounting holes and a chainable...
https://www.adafruit.com/product/1426

Adafruit SCD-41 - True CO2 Temperature and Humidity Sensor
Take a deep breath in...now slowly breathe out. Mmm isn't it wonderful? All that air around us, which we bring into our lungs, extracts oxygen from and then breathes out carbon...
https://www.adafruit.com/product/5190
Pink and Purple Woven USB A to USB C Cable - 1 meter long
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also made for USB C for our modernized breakout boards, Feathers, and...
https://www.adafruit.com/product/5153

USB Battery Pack for Raspberry Pi - 10000mAh - 2 x 5V outputs
A large-sized rechargeable battery pack for your Raspberry Pi (or Arduino, or...
https://www.adafruit.com/product/1566

STEMMA QT / Qwiic JST SH 4-Pin Cable - 50mm Long
This 4-wire cable is 50mm / 1.9" long and fitted with JST SH female 4-pin connectors on both ends. Compared with the chunkier JST PH these are 1mm pitch instead of 2mm, but...
https://www.adafruit.com/product/4399

4 x M2.5x6mm Screws
M2.5x6mm Screws for Feather and PMSA

2 x M2x8mm Screws
M2x8mm Screws for Feather

2 x M2x6mm Screws
M2x6mm Screws for LED stick
Circuit Diagram

The wiring diagram below provides a visual reference for connecting the components. It is not true to scale, it is just meant to be used as reference. This diagram was created using the Fritzing software package.

Take a moment to review the components in the circuit diagram. This illustration is meant for referencing wired connections - the length of wire, position and size of components are not exact.
NeoPixel

- 5v (Red) to 3v3
- G (Black) to GND
- DIN (White) to D13

Install CircuitPython

CircuitPython () is a derivative of MicroPython () designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org
Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

Plug your board into your computer, using a known-good data-sync cable, directly, or via an adapter if needed.

Double-click the reset button (highlighted in red above), and you will see the RGB status LED(s) turn green (highlighted in green above). If you see red, try another port, or if you're using an adapter or hub, try without the hub, or different adapter or hub.

For this board, tap reset and wait for the LED to turn purple, and as soon as it turns purple, tap reset again. The second tap needs to happen while the LED is still purple.

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
You will see a new disk drive appear called FTHRS3BOOT.

Drag the adafruit_circuitpython_etc.uf2 file to FTHRS3BOOT.

The BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it!
Once you've finished setting up your ESP32-S3 Reverse TFT Feather with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
# SPDX-FileCopyrightText: 2023 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import neopixel
import displayio
from adafruit_ticks import ticks_ms, ticks_add, ticks_diff
from adafruit_pm25.i2c import PM25_I2C
import adafruit_scd4x
from adafruit_display_text import bitmap_label
from adafruit_bitmap_font import bitmap_font
from adafruit_display_shapes.roundrect import RoundRect

pixel_pin = board.D13
num_pixels = 8

pixels = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=0.1, auto_write=False)
red = (80, 0, 0)
yellow = (75, 80, 0)
green = (0, 80, 0)
i2c = board.STEMMA_I2C()
reset_pin = None

pm25 = PM25_I2C(i2c, reset_pin)

scd4x = adafruit_scd4x.SCD4X(i2c)
scd4x.start_periodic_measurement()

time.sleep(5)

try:
aqdata = pm25.read()
    pm2 = int(aqdata["pm25 standard"])
except RuntimeError:
    pm2 = 0

c02 = scd4x.CO2
temp = scd4x.temperature
humidity = scd4x.relative_humidity

def rate_pm25(pm25_data):
```
if pm25_data <= 12:
    pm25_outline = 94
    pm25_color = green
elif pm25_data <= 35:
    pm25_color = yellow
    pm25_outline = 140
else:
    pm25_color = red
    pm25_outline = 185
return pm25_color, pm25_outline

def c_to_f(temp_data):
    temperature_celsius = temp_data
    temperature_fahrenheit = temperature_celsius * 9 / 5 + 32
    return temperature_fahrenheit

# display setup
display = board.DISPLAY

bitmap = displayio.OnDiskBitmap("/airBG.bmp")
tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
group = displayio.Group()
group.append(tile_grid)

small_font_file = "/OCRA_small.pcf"
small_font = bitmap_font.load_font(small_font_file)
big_font_file = "/OCRA_big.pcf"
big_font = bitmap_font.load_font(big_font_file)

pm2_text = bitmap_label.Label(big_font, text="%d" % pm2, x=37, y=40, color=0xFFFFFF)
group.append(pm2_text)

c02_text = bitmap_label.Label(small_font, text="%d" % co2, x=50, y=107,
color=0xFFFFFF)
temp_text = bitmap_label.Label(small_font, text="%d" % temp, x=130, y=107,
color=0xFFFFFF)
humid_text = bitmap_label.Label(small_font, text="%d" % humidity, x=205, y=107,
color=0xFFFFFF)
group.append(c02_text)
group.append(temp_text)
group.append(humid_text)

pm2_outline = RoundRect(94, 19, 46, 46, 10, fill=None, outline=0xFFFFFF, stroke=3)
group.append(pm2_outline)
display.show(group)

sensor_texts = [pm2_text, co2_text, temp_text, humid_text]
sensor_data = [pm2, co2, temp, humidity]
sensor_clock = ticks_ms()
sensor_check = 5000
first_run = True

while True:
    if first_run or ticks_diff(ticks_ms(), sensor_clock) > sensor_check:
        co2 = scd4x.CO2
        temp = c_to_f(scd4x.temperature)
        humidity = scd4x.relative_humidity
        try:
            aqdata = pm25.read()
            pm2 = int(aqdata["pm25 standard"])
        except RuntimeError:
            print("Unable to read from PM2.5 sensor, no new data..")
            continue
        pm2_color, pm2_outline.x = rate_pm25(pm2)
sensor_data = [pm2, co2, temp, humidity]
pixels.fill(pm2_color)
pixels.show()
for s in range(4):
    sensor_texts[s].text = "%d" % sensor_data[s]
    print("updated %d data" % sensor_data[s])
    time.sleep(0.2)
    sensor_clock = ticks_add(sensor_clock, sensor_check)
if first_run:
    sensor_clock = ticks_ms()
    first_run = False

Upload the Code and Libraries to the ESP32-S3 Reverse TFT Feather

After downloading the Project Bundle, plug your ESP32-S3 Reverse TFT Feather into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the ESP32-S3 Reverse TFT Feather's CIRCUITPY drive:

- lib folder
- OCRA_big.pcf
- OCRA_small.pcf
- airBG.bmp
- code.py

Your ESP32-S3 Reverse TFT Feather CIRCUITPY drive should look like this after copying the lib folder, font files, bitmap file and the code.py file:
How the CircuitPython Code Works

The code begins by setting up the NeoPixel stick. Three colors are defined: red, yellow and green. These colors will be used to represent the AQI reading.

```python
pixel_pin = board.D13
num_pixels = 8
pixels = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=0.1, auto_write=False)
red = (80, 0, 0)
yellow = (75, 80, 0)
green = (0, 80, 0)
```

Sensor Setup

Next, the SCD40 and PMSA003 sensors are instantiated over I2C and begin reading data.

```python
i2c = board.STEMMA_I2C()
reset_pin = None
pm25 = PM25_I2C(i2c, reset_pin)
aqdata = pm25.read()
scd4x = adafruit_scd4x.SCD4X(i2c)
scd4x.start_periodic_measurement()
time.sleep(5)
co2 = scd4x.CO2
temp = scd4x.temperature
humidity = scd4x.relative_humidity
pm2 = int(aqdata["pm25_standard"])```

Functions

Two functions are used to translate the sensor data. `rate_pm25()` passes the AQI reading and determines what color the NeoPixels should show to represent it. It also determines which square on the display should be outlined with a white rectangle.

`c_to_f()` converts the temperature reading from Celsius to Fahrenheit.

```python
def rate_pm25(pm25_data):
    if pm25_data <= 12:
        pm25_outline = 94
        pm25_color = green
    elif pm25_data <= 35:
        pm25_color = yellow
        pm25_outline = 140```
else:
    pm25_color = red
    pm25_outline = 185
    return pm25_color, pm25_outline

def c_to_f(temp_data):
    temperature_celsius = temp_data
    temperature_fahrenheit = temperature_celsius * 9 / 5 + 32
    return temperature_fahrenheit

Display Setup

Then comes the display setup. The background bitmap is loaded using the `OnDiskBitmap` function. Two font files are loaded as bitmap fonts.

```python
# display setup
display = board.DISPLAY
bitmap = displayio.OnDiskBitmap("/airBG.bmp")
tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
group = displayio.Group()
group.append(tile_grid)
small_font_file = "/OCRA_small.pcf"
small_font = bitmap_font.load_font(small_font_file)
big_font_file = "/OCRA_big.pcf"
big_font = bitmap_font.load_font(big_font_file)
```

Sensor Text

Each sensor reading has a text object to display the data on the screen. The AQI reading utilizes the `big_font` and the rest use the `small_font`.

A `RoundRect` is used to outline one of the green, yellow or red squares on the bitmap to further illustrate the AQI reading. The `x` coordinate is changed for the `RoundRect` with the `rate_pm25()` function.

```python
pm2_text = bitmap_label.Label(big_font, text="%d" % pm2, x=37, y=40, color=0xFFFFFF)
group.append(pm2_text)
co2_text = bitmap_label.Label(small_font, text="%d" % co2, x=50, y=107, color=0xFFFFFF)
temp_text = bitmap_label.Label(small_font, text="%d" % temp, x=130, y=107, color=0xFFFFFF)
humid_text = bitmap_label.Label(small_font, text="%d" % humidity, x=205, y=107, color=0xFFFFFF)
group.append(co2_text)
group.append(temp_text)
group.append(humid_text)

pm2_outline = RoundRect(94, 19, 46, 46, 10, fill=None, outline=0xFFFFFF, stroke=3)
group.append(pm2_outline)
display.show(group)
```
Arrays and Ticks

There are two arrays: `sensor_texts` and `sensor_data`. These arrays will be used in the loop to update the on screen texts with the sensor readings.

`ticks_ms()` is used to keep time. Every 5 seconds, the sensors will be read in the loop and the screen will be updated.

```python
sensor_texts = [pm2_text, co2_text, temp_text, humid_text]
sensor_data = [pm2, co2, temp, humidity]
sensor_clock = ticks_ms()
sensor_check = 5000
first_run = True
```

The Loop

In the loop, the SCD40 and PMSA003 sensors are read. The `rate_pm25()` function is used to determine which color the NeoPixels should be.

The sensor readings are packed into the `sensor_data` array. A `for` statement is used to iterate through the array and update the corresponding text objects.

Finally, the `sensor_clock` is reset to begin counting up to 5 seconds again.

```python
if first_run or ticks_diff(ticks_ms(), sensor_clock) &gt; sensor_check:
    co2 = scd4x.CO2
temp = c_to_f(scd4x.temperature)
humidity = scd4x.relative_humidity
aqdata = pm25.read()
pm2 = int(aqdata["pm25 standard"]) pm2_color, pm2_outline.x = rate_pm25(pm2)
sensor_data = [pm2, co2, temp, humidity]
pixels.fill(pm2_color)
pixels.show()
for s in range(4):
    sensor_texts[s].text = "%d" % sensor_data[s]
    print("updated %d data" % sensor_data[s])
    time.sleep(0.2)
sensor_clock = ticks_add(sensor_clock, sensor_check)
```
3D Printing

Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

Edit Design

Download STLs

Slice with settings for PLA material

The parts were sliced using CURA using the slice settings below.

PLA filament 220c extruder
0.2 layer height
10% gyroid infill
60mm/s print speed
60c heated bed

Supports

Support Overhang Angle: 50
Support Destiny: 6%
Enable Support Interface
Enable Support Roof
Support Z Distance: .21
Assemble

**Secure PMSA**

Use two M2.5x6mm screws to attach the PMSA to the frame.

**NeoPixel JST**

Solder a socket 3 Pin JST cable to easily connect the NeoPixel stick.

- 5v (Red) to 3v3
- G (Black) to GND
- DIN (White) to D13

**Stemma cable**

Attach a short STEMMA cable to the Feather.

**Secure Feather**

Use M2.5x6mm screws on the plated screws holes. Use M2x8mm long screws on the non plated screws holes.
Secure NeoPixel stick
Solder a short 3 Pin JST jack to the NeoPixel stick.

5v (Red) to 3v3
G (Black) to GND
DIN (White) to D13
Use M2x6mm screws to secure the Stick to the frame.

Attach Sensor Bracket
Secure the sensor bracket to the PMSA with two M2.5x8mm screws.
Secure Sensor
Use M3x5mm screws to secure the SCD4x sensor to the bracket. Connect both sensor with an additional short STEMMA cables.

Place in Case
Insert the assembly into the case. The back cover aligns to the SCD4x sensor and press fits to the case.
Attach Screen frame
Press fit the screen frame over the display cutout.

Press fit Feet
The short feet press fits to the back of the case. The taller feet press fit to the front of the case.

Wrist Band
The wrist band is attached by sewing a .6" wide Elastic band to the band frame.

Complete