RGB Matrix Portal Room CO2 Monitor

Created by Carter Nelson

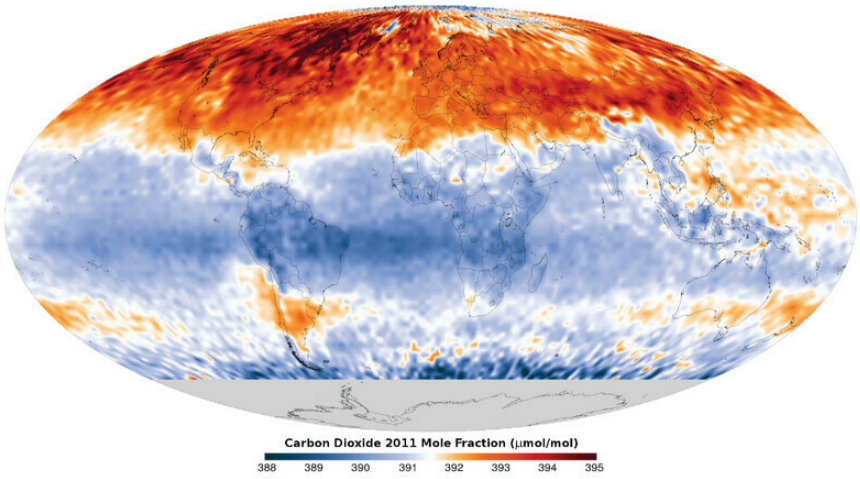
https://learn.adafruit.com/matrix-portal-room-co2-monitor

Last updated on 2024-06-03 03:19:49 PM EDT

Table of Contents

Overview

Carbon dioxide, aka CO_2, is a gas that is an essential part of the Earth's atmosphere (https://adafru.it/CQj) and life in general. However, while essential, it can have negative effects when concentrations exceed certain level. It can impact the entire planet via global warming (https://adafru.it/QfG). But it can also have very local effects, for example on indoor air quality (https://adafru.it/QfH). Elevated levels of CO_2 can lead to reduced cognitive ability and other health related concerns. Therefore, monitoring CO_2 levels of inside air can be useful as a part of gauging general air quality.



Numerous projects have already been done to monitor indoor CO_2. This project posted to Hackster (https://adafru.it/QfI) uses an ESP8266 and a CCS811 to send values to Adafruit IO (https://adafru.it/fsU). This tweet (https://adafru.it/QfJ) (and also

blog post (https://adafru.it/Qek) with more info) shares a project done by an 11 year old and is based on the Feather ecosystem using the SCD-30 (http://adafru.it/4867) CO_2 sensor from Sensirion. This is an excellent **true** CO_2 concentration sensor which we also use in this project.

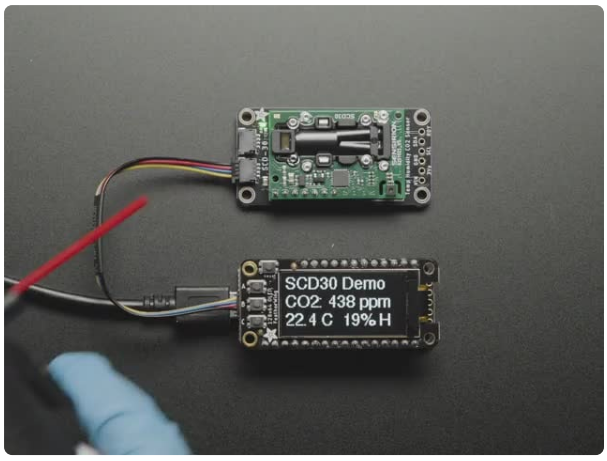
In this project we use the **SCD-30** along with a **Matrix Portal** to drive a 64x32 RGB **LED matrix**. This provides a nice way to display the current conditions and have it be readable to an entire room. And as a bonus, with this hardware arrangement, **there is no soldering required!**

This project does NOT require any soldering and the matrix is really big so its visible in a classroom, workshop, or on the wall of a home!

Hardware

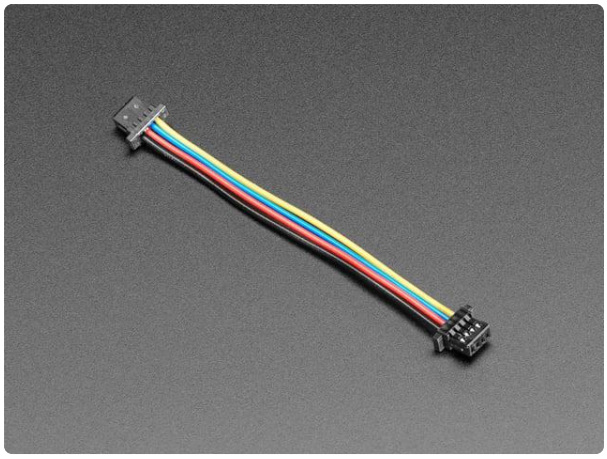
Here's a summary of the hardware needed for this project. Some items are optional. Also note that if you purchased an **Adabox 016** (https://adafru.it/ODd), you will have most of these items already.

The key item is of course the **SCD-30 CO2 sensor** itself:



Adafruit SCD-30 - NDIR 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/4867

To provide a solderless way to connect the SCD-30 to the Matrix Portal, you can use a **STEMMA QT cable**. They come in various lengths:

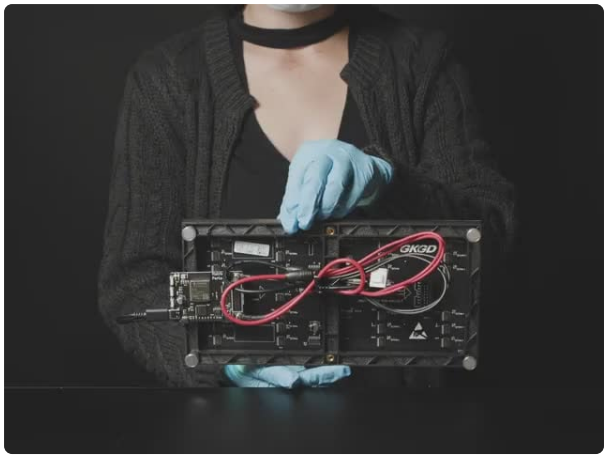


[STEMMA QT / Qwiic JST SH 4-Pin Cable - 50mm Long]
This 4-wire cable is 50mm / 1.9" long and fitted with JST SH female 4-pin connectors on both ends. Compared with the chunkier JST PH these are 1mm pitch instead of 2mm, but...
https://www.adafruit.com/product/4399

[STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long]
This 4-wire cable is a little over 100mm / 4" 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...
https://www.adafruit.com/product/4210

[STEMMA QT / Qwiic JST SH 4-Pin Cable - 200mm Long]
This 4-wire cable is a little over 200mm / 7.8" 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...
https://www.adafruit.com/product/4401

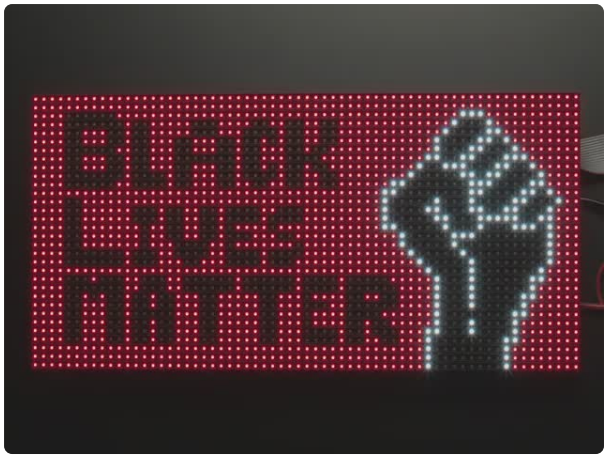
You'll also need a **64x32 LED matrix** and a **Matrix Portal** to drive the matrix:



Adafruit Matrix Portal - CircuitPython Powered Internet Display

Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...

https://www.adafruit.com/product/4745



64x32 RGB LED Matrix - 4mm pitch

Bring a little bit of Times Square into your home with this sweet 64 x 32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them...

https://www.adafruit.com/product/2278

OPTIONAL. You can add one of these acrylic diffusers to provide a more matted look to the display:

Black LED Diffusion Acrylic Panel - 10.2" x 5.1"

nice whoppin' rectangular slab of some lovely black acrylic to add some extra diffusion to your LED Matrix project. This material is 2.6mm (0.1") thick and is made of...

https://www.adafruit.com/product/4749

CO2 Air Levels

The SCD-30 sensor reports the CO_2 levels in units of parts per million (https://adafru.it/QfL) (ppm). This is a bit of a wacky unit of measure, but is pretty much what it says. If you had a million "parts" of air, then how many "parts" of CO_2 does it contain.

Or as OSHA (https://adafru.it/QfM) defines it:

Parts of vapor or gas per million parts of contaminated air by volume at 25 °C and 760 torr.

From the SCD-30 datasheet (https://adafru.it/QfN) we can see that the sensor range is 400 to 10000 ppm:

But how do these ppm levels translate into air quality? Below is a PDF that covers lots of issues related to indoor air quality:

OSHA Indoor Air Quality in Commercial and Institutional Buildings

https://adafru.it/QfO

Buried in **Appendix A** is a section that discusses carbon dioxide levels. The key one they mention is the **5000 ppm Permissible Exposure Limit** (PEL). Think of that as the upper limit - your workplace should ideally be below that level.

Values below the 5000 ppm PEL limit are commonly broken down into these subjective ranges, which we have also adopted for use in this project.

- < **1000** = **Good** air. Your body will be happy!
- **1000 - 2000** = **Poor** air. See if there is any way to improve.
- **2000 - 5000** = **Warning** levels. Good idea to investigate why.
- > **5000** = **Dang**, you are above the OSHA Permissible Exposure Limit for 8 hour exposure. Something should be done.

So by "Dang" do you mean "Dangerous"? Sort of, but not like immediately dangerous. That requires levels greater than around 15,000 ppm (see PDF linked above). So it's not like you need to run for your life if the ppm jumps above 5000. But you don't want to spend extended periods of time in that environment.

Matrix Display of Air Quality

The general idea for the Matrix Portal based air quality display is pretty simple. It has a readout of the current CO_2 levels in ppm so that value it always viewable. Then, for each of the four ranges above, there is an associated "smiley face" icon and word. This combination of icon/word was chosen as way to make the display more universally readable. The smileys don't rely on any specific language and additionally use color to help reinforce the condition. The green=good and red=bad association is fairly universal. But not everyone can distinguish the red/green colors. So the word provides an additional textual (but in English only) indication in a single color.

Hopefully with the combination of the two, smiley + word, the message is universally understood.

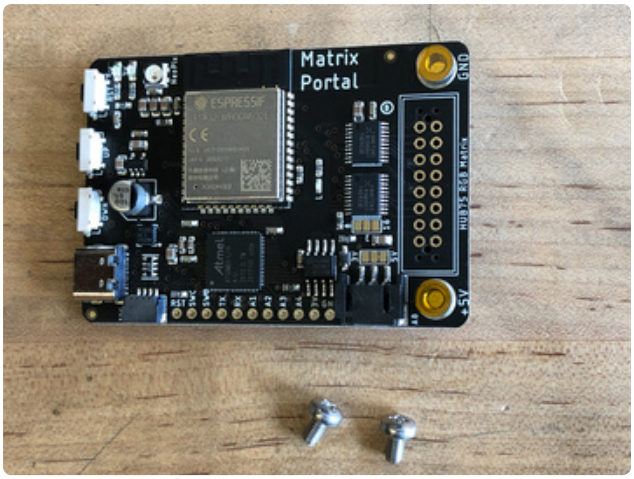
CO2 ppm < 1000

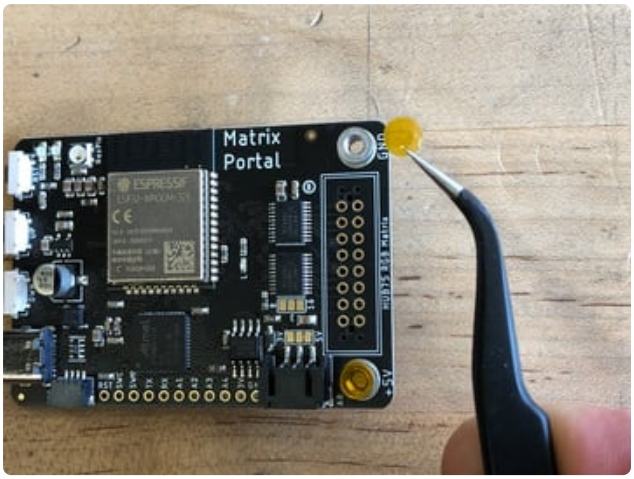
1000 <= CO2 ppm < 2000

2000 <= CO2 ppm < 5000

CO2 ppm >= 5000

Prep the MatrixPortal

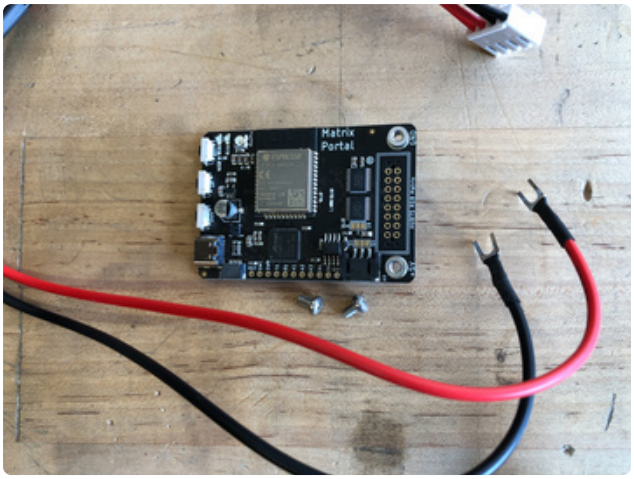


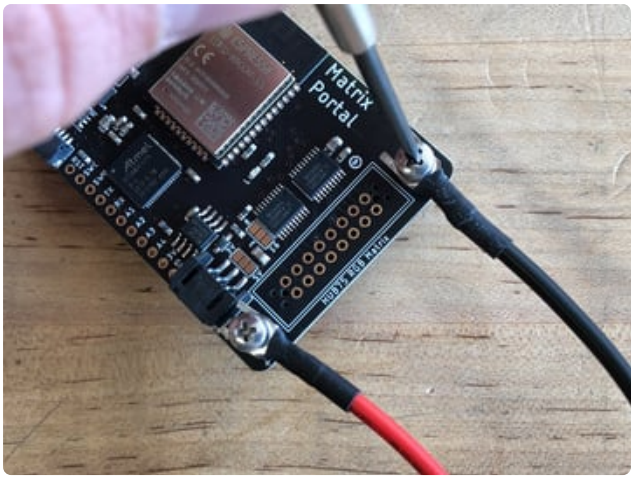


Power Prep

The MatrixPortal supplies power to the matrix display panel via two standoffs. These come with protective tape applied (part of our manufacturing process) which MUST BE REMOVED!

Use some tweezers or a fingernail to remove the two amber circles.

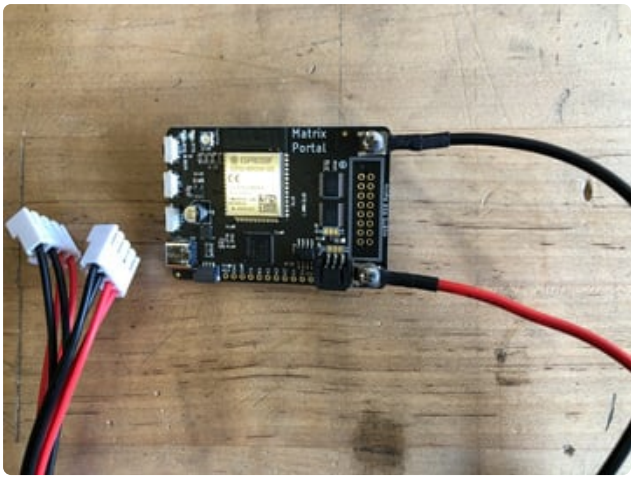


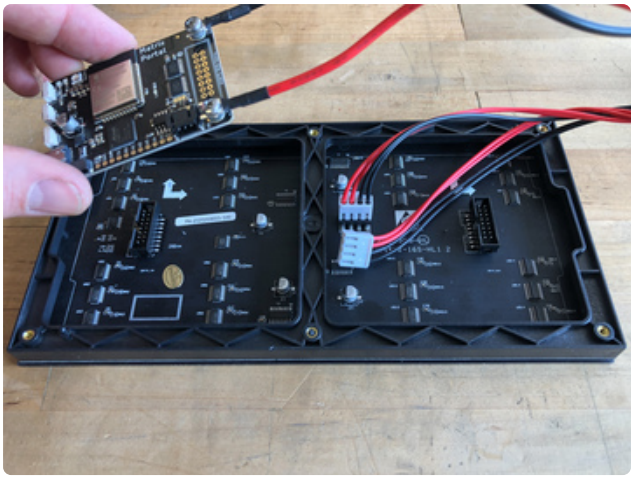


Power Terminals

Next, screw in the spade connectors to the corresponding standoff.

red wire goes to **+5V**
black wire goes to **GND**



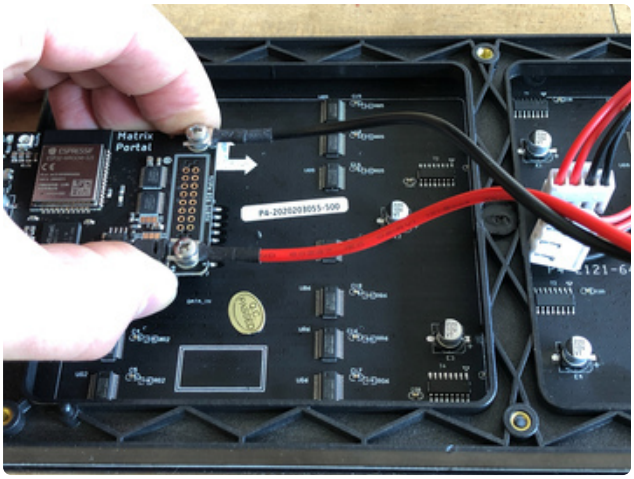


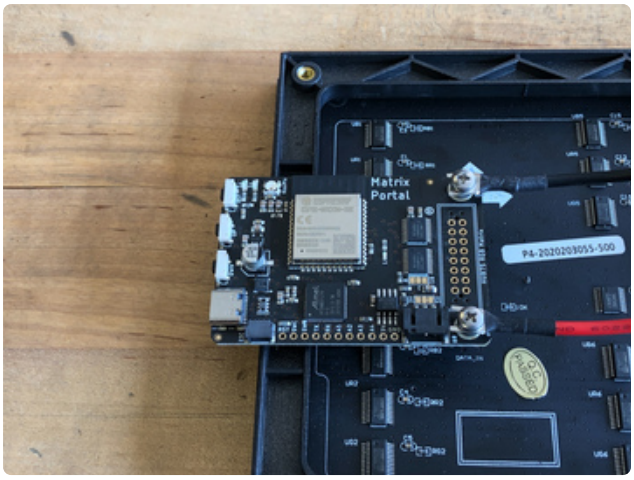
Panel Power

Plug either one of the four-conductor power plugs into the power connector pins on the panel. The plug can only go in one way, and that way is marked on the board's silkscreen.

Dual Matrix Setup

If you're planning to use a 64x64 matrix, follow these instructions on soldering the Address E Line jumper (https://adafru.it/OdJ).

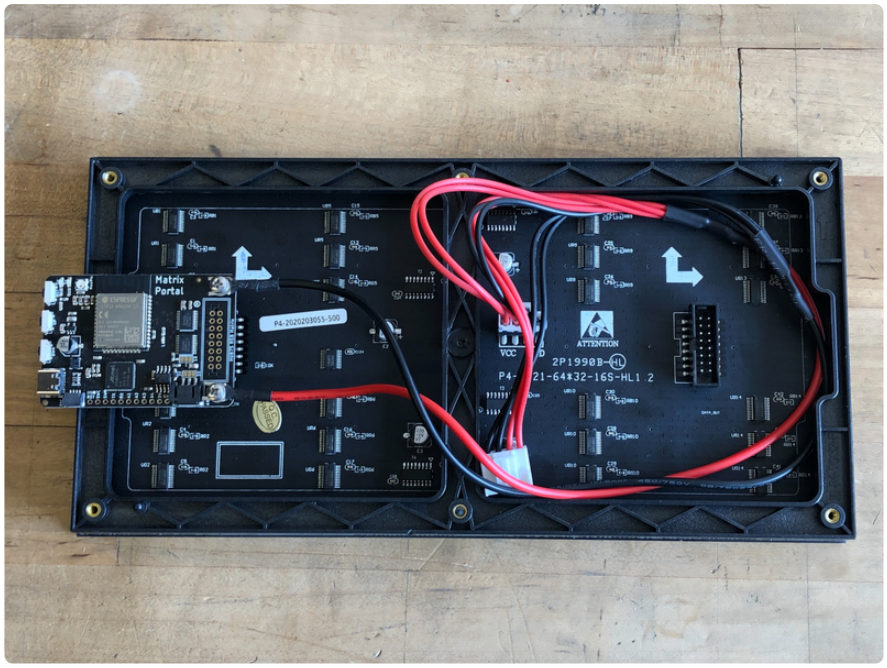




Board Connection

Now, plug the board into the left side shrouded 8x2 connector as shown. The orientation matters, so take a moment to confirm that the **white indicator arrow on the matrix panel is oriented pointing up and right** as seen here and the MatrixPortal overhangs the edge of the panel when connected. This allows you to use the edge buttons from the front side.

Check nothing is impeding the board from plugging in firmly. If there's a plastic nub on the matrix that's keeping the Portal from sitting flat, cut it off with diagonal cutters



For info on adding LED diffusion acrylic, see the page LED Matrix Diffuser.

Install CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) 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.

Set up CircuitPython Quick Start!

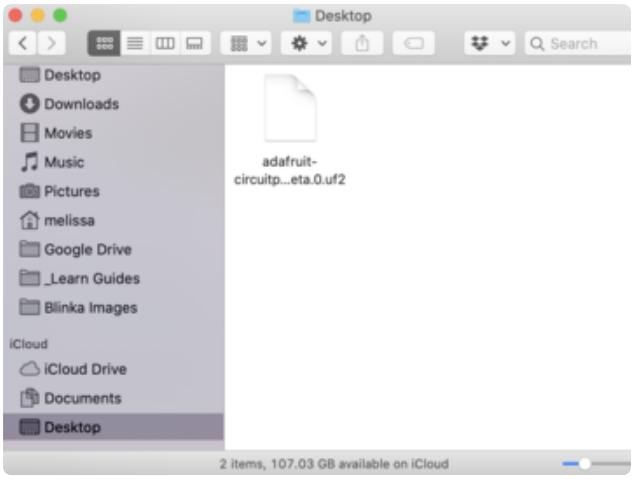
Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/Nte

Further Information

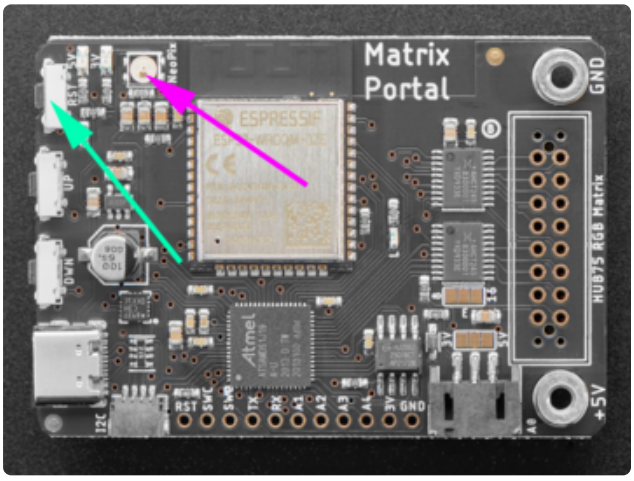
For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).



Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).

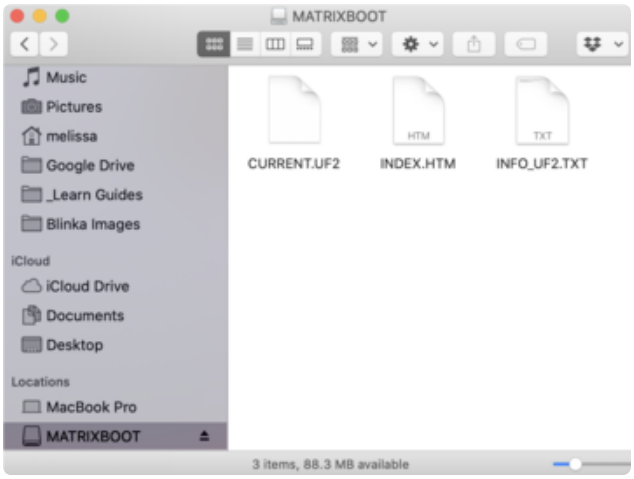
Plug your MatrixPortal M4 into your computer using a known-good USB cable.



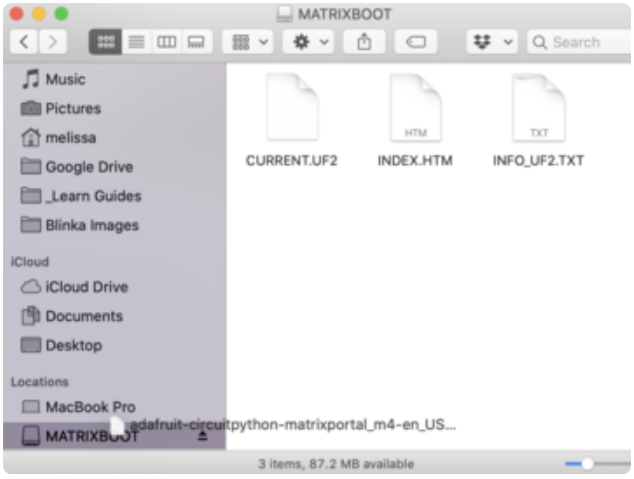
A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the **Reset** button (indicated by the green arrow) on your board, and you will see the NeoPixel RGB LED (indicated by the magenta arrow) turn green. If it turns red, check the USB cable, try another USB port, etc.

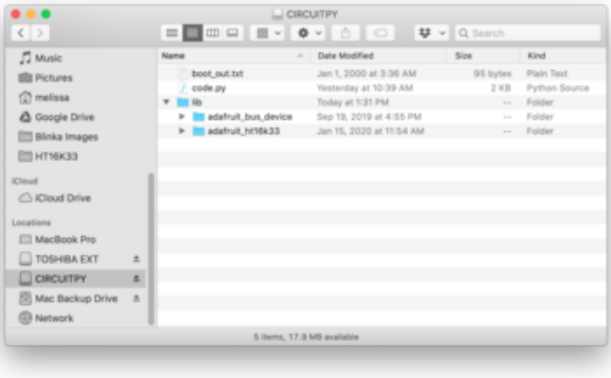
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



You will see a new disk drive appear called **MATRIXBOOT**.



Drag the **adafruit_circuitpython_etc.uf2** file to **MATRIXBOOT.**



The LED will flash. Then, the **MATRIXBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Code

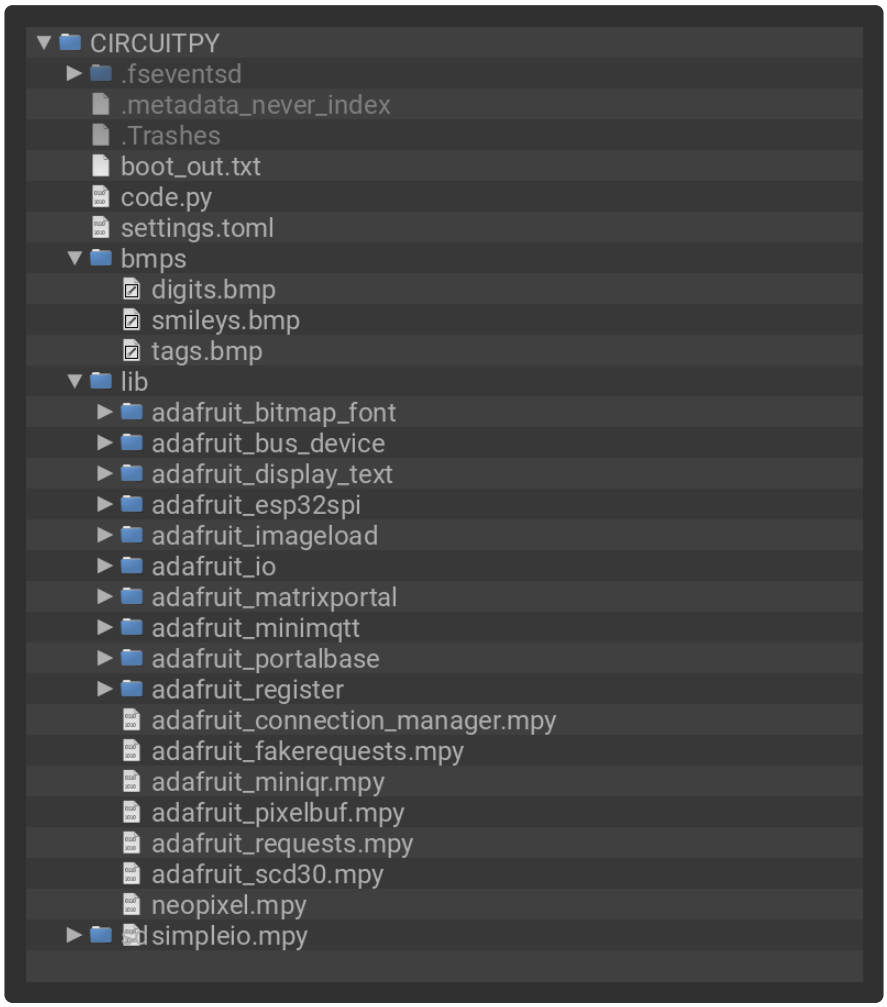
Here's how to load the code and assets as well as some additional libraries you'll need.

Additional Libraries

To use with CircuitPython, you need to first install a few libraries, into the lib folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, open the directory **Matrix_Portal_CO2_Monitor/** and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should now look similar to the following image:



Code

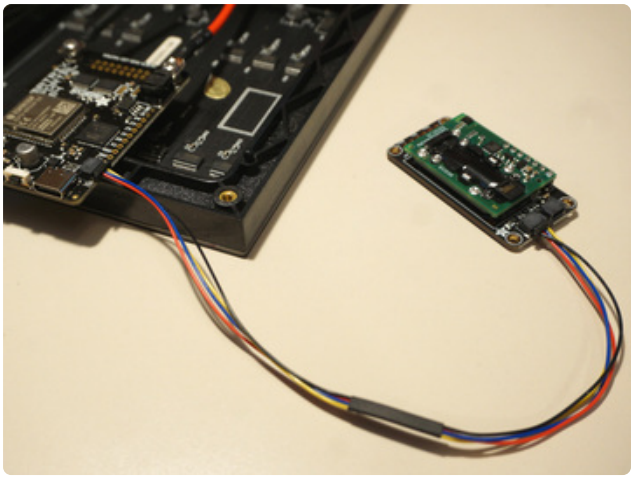
Here's the code for the Matrix Portal CO_2 Monitor.

```
# SPDX-FileCopyrightText: 2021 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import displayio
import adafruit_imageload
from adafruit_matrixportal.matrix import Matrix
import adafruit_scd30

# --| User Config |----
CO2_CUTOFFS = (1000, 2000, 5000)
UPDATE_RATE = 1
# ---------------------

# the sensor
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
scd30 = adafruit_scd30.SCD30(i2c)

# optional if known (pick one)
# scd30.ambient_pressure = 1013.25
# scd30.altitude = 0

# the display
matrix = Matrix(width=64, height=32, bit_depth=6)
display = matrix.display
display.rotation = 90  # matrixportal up
# display.rotation = 270 # matrixportal down

# current condition smiley face
smileys_bmp, smileys_pal = adafruit_imageload.load("/bmps/smileys.bmp")
smiley = displayio.TileGrid(
    smileys_bmp,
    pixel_shader=smileys_pal,
    x=0,
    y=0,
    width=1,
    height=1,
    tile_width=32,
    tile_height=32,
)

# current condition label
tags_bmp, tags_pal = adafruit_imageload.load("/bmps/tags.bmp")
label = displayio.TileGrid(
    tags_bmp,
    pixel_shader=tags_pal,
    x=0,
    y=32,
    width=1,
    height=1,
    tile_width=32,
    tile_height=16,
)

# current CO2 value
digits_bmp, digits_pal = adafruit_imageload.load("/bmps/digits.bmp")
co2_value = displayio.TileGrid(
    digits_bmp,
    pixel_shader=digits_pal,
    x=0,
    y=51,
    width=4,
    height=1,
    tile_width=8,
```

```
    tile_height=10,
)

# put em all together
splash = displayio.Group()
splash.append(smiley)
splash.append(label)
splash.append(co2_value)

# and show em
display.root_group = splash


def update_display(value):

    value = abs(round(value))

    # smiley and label
    if value < CO2_CUTOFFS[0]:
        smiley[0] = label[0] = 0
    elif value < CO2_CUTOFFS[1]:
        smiley[0] = label[0] = 1
    elif value < CO2_CUTOFFS[2]:
        smiley[0] = label[0] = 2
    else:
        smiley[0] = label[0] = 3

    # CO2 value
    # clear it
    for i in range(4):
        co2_value[i] = 10
    # update it
    i = 3
    while value:
        co2_value[i] = value % 10
        value = int(value / 10)
        i -= 1


while True:
    # protect against NaNs and Nones
    try:
        update_display(scd30.CO2)
    except:
        pass
    time.sleep(UPDATE_RATE)
```

CO2 Sensor Setup

Connecting the SCD-30 sensor to the Matrix Portal is easy thanks to the STEMMA QT connector system (https://adafru.it/Ft4).



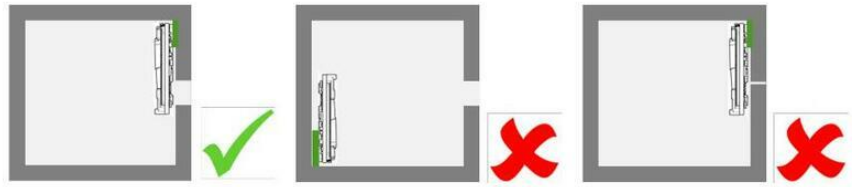
Pretty simple. Use the **STEMMA QT cable** to connect the **SCD-30** to the **Matrix Portal**.

Where to actually place the SCD-30 is sort of up to you and your specific setup. However, you'll want to put a little thought into it to make sure the sensor is properly exposed to the ambient air conditions. Sensirion actually has a document with some guidelines you can read below:

SCD30 Sensor Placement Guidelines

https://adafru.it/PFf

For example, make sure the sensor isn't exposed to direct sunlight. They also have a nice diagram to illustrate suggested placement within an enclosure:



There's also information related to self heating and use in a moving air duct. So give that document a read to help determine where and how best to locate the SCD-30.

CircuitPython Usage

If you want more in depth coverage of using the SCD-30 sensor in CircuitPython, checkout the main guide for the sensor below:

SCD-30 Python & CircuitPython

https://adafru.it/QfP

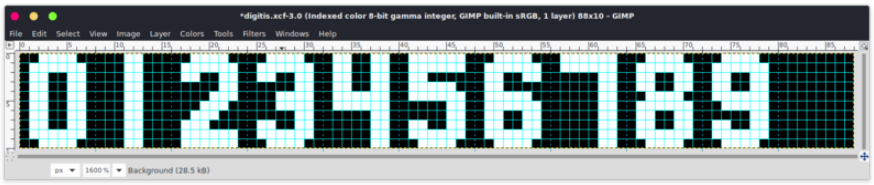
What Font?

If you've done other CircuitPython projects involving text and displays, you've probably used a custom font file. That's a great way to get a nice custom look to project read outs. But you may have noticed - **this project has no font file**.

So how did we get those nice bold text readouts? Let's talk about that a bit. It's a trick that could be useful for other projects. We'll mainly focus on the way the CO_2 reading was done.

Sprite Sheet Instead of Font File

At the heart of the CircuitPython displayio (https://adafru.it/EGh) library is the concept of TileGrids (https://adafru.it/GC1). This lets you carve up a source bitmap into multiple sections (tiles) and then layout one or more of them (grid). You can do this with any bitmap, but here we use one that contains the numbers 0-9. It was created by hand in GIMP (https://adafru.it/GCa):



The max value the SCD30 can report is 10000 ppm. Ignoring that upper limit, every other reading is at most 4 digits wide. The LED matrix we use is 32 pixels wide. Mathy McMathy says "32 / 4 = 8" so we make each digit be 8 pixels wide. We chose 10 pixels for the height as a "meh, that looks about right" value.

Loading the source **Bitmap** into our code is just one line of code:

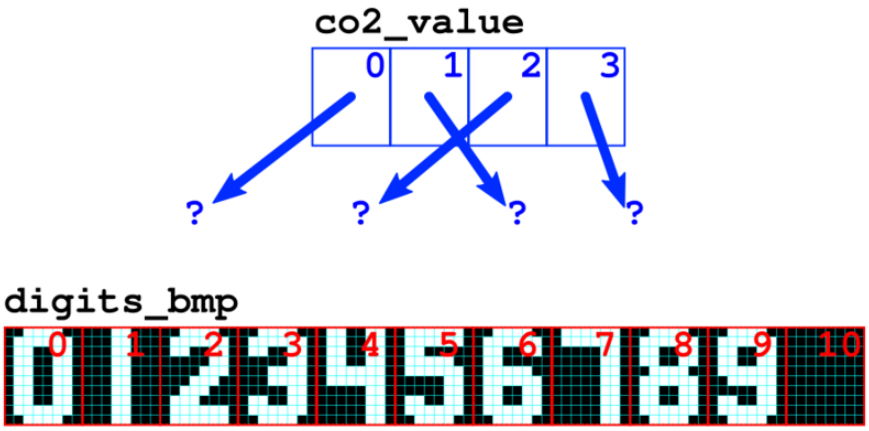
```
digits_bmp, digits_pal = adafruit_imageload.load("/bmps/digits.bmp")
```

Setting up our **TileGrid** that will use this source **Bitmap** is also a single line of code, but there are numerous parameters that get used:

```
co2_value = displayio.TileGrid(
    digits_bmp,
    pixel_shader=digits_pal,
    x=0,
    y=51,
    width=4,
    height=1,
    tile_width=8,
```

```
    tile_height=10,
)
```

We tell it to use `digits_bmp`, and its associate color palette `digits_pal`, as our source Bitmap. We also go ahead and set our location on the matrix with `x` and `y`. Now for the more important ones to understand. With `width` and `height` we are setting the size of the resulting **TileGrid** in terms of number of tiles. The similarly named `tile_width` and `tile_height` are what actually set the individual tile size - and how the source bitmap is carved up. The result looks something like this:



The `co2_value` **TileGrid** is 4 x 1 tiles. Each of these tiles can "point" to any of the possible source tiles, 0 - 10, from the source **Bitmap** `digits_bmp`. The syntax for doing that would look like:

```
co2_value[1] = 4
```

to set the second digit (index 1) to 4 as an example.

By creating our source bitmap with the digits arranged as they are, the indices correspond to the actual digits, 3=3, 7=7, etc. That lets us use the digits from the actual CO_2 reading to set each tile. That's what this bit of code does:

```
# CO2 value
# clear it
for i in range(4):
    co2_value[i] = 10
# update it
i = 3
while value:
    co2_value[i] = value % 10
    value = int(value / 10)
    i -= 1
```

So it looks like we are printing text using a blocky font, but we are not. It's all done with a bitmap file. Pretty neat trick, huh?

Why Do That?

So why do this TileGrid / Bitmap file approach vs. just using a font file and formatted prints? Mainly for precise control of the resulting output. Like pixel-by-pixel control. The LED matrix is physically large but is pretty low resolution - only 64 x 32. And our CO_2 value readout uses only 8x10 pixel digits. Getting a font to render exactly like you want at that low of a resolution can sometimes be tricky. And since we only needed digits, not a full alphabet, we just felt we could get there quicker with this approach.

TileGrids can be a bit tricky to wrap your head around at first. Hopefully this example not only helps illustrate how they work, but also maybe a neat way you can use them in your projects. Be sure to checkout the Displayio Learn Guide (https://adafru.it/EGh) for more details.

Changing The Text Labels

The text readouts are done with the four English words GOOD, POOR, WARN, and DANG. This was also done using a TileGrid / Bitmap combo. Here's the source bitmap:

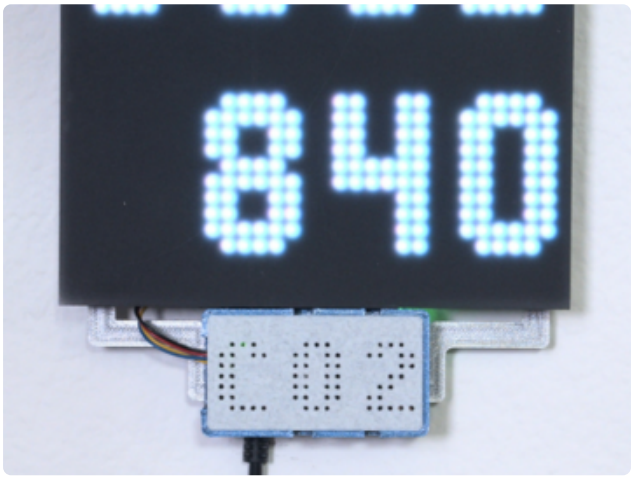
It's simpler than the CO2 value read out. It's just a single tile which points to one of the 32x16 pixel words. If you wanted to change the text, you could do so by editing the source bitmap. Since it's a bitmap, you can make it be whatever you want. No need to find a font that supports a specific language.

Changing The Smileys

The smiley face icons also come from a source bitmap:

So they could also be changed if you wanted by simply editing each of the sub 32x32 pixel icons.

3D Printed Sensor Case



We designed a 3D printed bracket for the display and sensor so the parts are nicely secured.

You can use a screw or nail to hang it on a wall with our 3D printed bracket.

The bracket is secured to the heat set inserts that are built into the frame of the display.

The sensor is press fitted into the case with openings on the side for the cables.

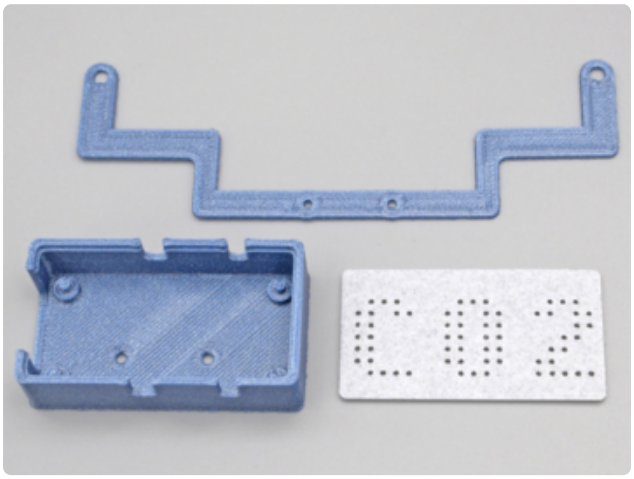
The cover snap fits over the case and features holes to allow air to reach the sensor.

The bracket and enclosure parts are 3d printed without any support material using PLA filament.

The Wall hanger is attached the Matrix with M3 screws.

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.



Medium Sized Matrix (Adabox 16 (https://adafru.it/ODd))

CO2-Lid-txt
CO2-Case
CO2-Bracket (https://www.adafruit.com/product/2278 (http://adafru.it/2278))
CO2-Wall-Hang
Large Matrix PID: 2276 (https://www.adafruit.com/product/2276 (http://adafru.it/2276))

CO2Bracket-LG
CO2-Wall-Hang-LG.stl
Wall-Hang-LG

Edit Matrix CO2 Parts

https://adafru.it/QBV

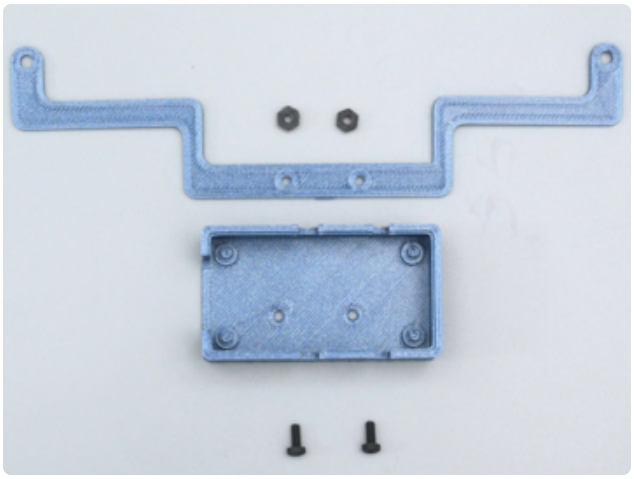
Download STLs

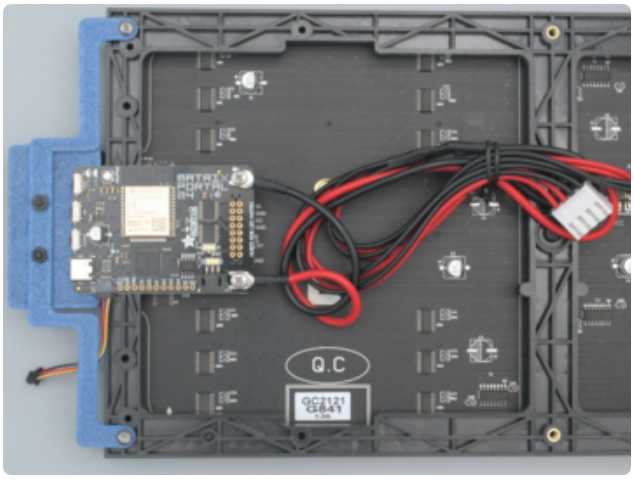
https://adafru.it/QCG

Slicing Parts

Slice with settings for PLA material. The parts were sliced using CURA using the slice settings below.

PLA filament 215c extruder
0.2 layer height
10% gyroid infill
60mm/s print speed
60c heated bed

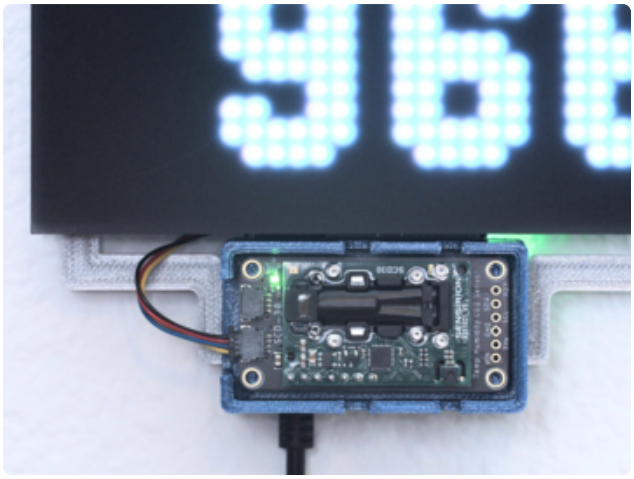


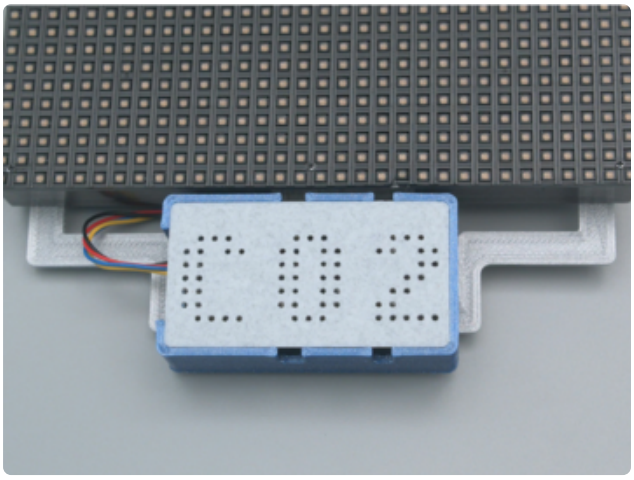


Case Assembly

The bracket is secured to the enclosure using M2.5 hardware screws and hex nuts.

The bracket is secured to the heat set inserts that are built into the frame of the display.





The sensor is press fitted into the case with openings on the side for the cables.

The cover snap fits over the case and features holes to allow air to reach the sensor.