Disconnected CO2 Data Logger
Created by Liz Clark

https://learn.adafruit.com/disconnected-co2-data-logger

Last updated on 2021-11-15 08:29:51 PM EST
# Table of Contents

**Overview**  
- Parts 4

**Circuit Diagram**  
- Wiring Connections 7

**3D Printing**  
- 8

**Install CircuitPython**  
- CircuitPython Quickstart 9
- Safe Mode 10
- Flash Resetting UF2 12

**Coding the Disconnected CO2 Data Logger**  
- Upload the Code and Libraries to the Feather RP2040 14
- How the CircuitPython Code Works 15

**Wiring**  
- 17

**Assembly**  
- 19

**Usage**  
- 23
Overview

Have you ever wondered what the air quality is around you while you're out and about? With this project, you'll be able to monitor CO2 levels and log them, like an IoT device, but without having to worry about connecting to a network. This means you can collect data from places like public transit, concert venues or even on your bike ride.

The SCD-40 STEMMA sensor monitors the true CO2 level of the air. It also has options for temperature and humidity sensing.

A FeatherWing Doubler connects the Feather RP2040 and Adalogger FeatherWing together. The Adalogger FeatherWing takes care of storing the logged data from the SCD-40 on a removeable SD card and keeps time with an onboard real time clock (RTC).
To keep things portable, this project is powered by a LiPoly battery that plugs directly into the Feather RP2040. The circuit includes an on/off switch so that you can choose when to log data and prolong battery life.

**Parts**

**Adafruit Feather RP2040**
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought “this chip is going to be awesome when we give it the Feather...”
https://www.adafruit.com/product/4884

**Adalogger FeatherWing - RTC + SD Add-on For All Feather Boards**
A Feather board without ambition is a Feather board without FeatherWings! This is the Adalogger FeatherWing: it adds both a battery-backed Real Time Clock and micro SD...
https://www.adafruit.com/product/2922
FeatherWing Doubler - Prototyping Add-on For All Feather Boards
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our https://www.adafruit.com/product/2890

Adafruit SCD-40 - 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/5187

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

CR1220 12mm Diameter - 3V Lithium Coin Cell Battery
These are the highest quality & capacity batteries, the same as shipped with the iCufflinks, iNecklace, Datalogging and GPS Shields, GPS HAT, etc. One battery per order...
https://www.adafruit.com/product/380
4GB Blank SD/MicroSD Memory Card
Add mega-storage in a jiffy using this 4 GB micro-SD card. It comes with a SD adapter so you can use it with any of our shields or adapters! Preformatted to FAT so it works out of...
https://www.adafruit.com/product/102

Breadboard-friendly SPDT Slide Switch
These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They're easy to switch...
https://www.adafruit.com/product/805
Plug the Feather RP2040 and Adalogger FeatherWing into the FeatherWing Doubler.
- EN to slide switch pin 1
- GND to slide switch pin 2
- SCD-40 STEMMA QT port to the Feather RP2040’s STEMMA QT port with a STEMMA QT JST cable
- LiPoly battery to the Feather RP2040’s LiPoly battery connector
3D Printing

The enclosure is a 3D printed, snap fit case that prints with no supports.

It consists of three parts:

- co2_topLid
- co2_botLid
- co2_mainBox

The files can be downloaded directly here or through Thingiverse.

The top and bottom lids are removable to allow for easy access to the SD card and electronics.

The case has mounting holes for the FeatherWing Doubler and the SCD-40 STEMMA sensor.

There are cutouts for the on/off switch and the Feather RP2040's USB-C port so that you can charge the LiPoly battery or update the code.

Install CircuitPython

CircuitPython (https://adafruit.it/tB7) is a derivative of MicroPython (https://adafruit.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.
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
https://adafruit.io/R1D

Click the link above to download the latest CircuitPython UF2 file.
Save it wherever is convenient for you.

To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.
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 RPI-RP2.

Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.

The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

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

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.
Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

**Entering Safe Mode in CircuitPython 6.x**

To enter safe mode when using CircuitPython 6.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 700ms. On some boards, the onboard status LED (highlighted in green above) will turn solid yellow during this time. If you press reset during that 700ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

**Entering Safe Mode in CircuitPython 7.x**

To enter safe mode when using CircuitPython 7.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will
blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

Once you've entered safe mode successfully in CircuitPython 6.x, the LED will pulse yellow.

If you successfully enter safe mode on CircuitPython 7.x, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.
CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.
Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

https://adafruit.it/RLE

Coding the Disconnected CO2 Data Logger

Once you've finished setting up your Feather RP2040 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: 2021 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import digitalio
import adafruit_scd4x
import adafruit_sdcard
import busio
import storage
import adafruit_pcf8523

# setup for I2C
i2c = board.I2C()
# setup for SCD40
scd4x = adafruit_scd4x.SCD4X(i2c)
# setup for RTC
rtc = adafruit_pcf8523.PCF8523(i2c)
# start measuring co2 with SCD40
scd4x.start_periodic_measurement()
# list of days to print to the text file on boot
days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

# SPI SD CS pin
SD_CS = board.D10
# SPI setup for SD card
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
cs = digitalio.DigitalInOut(SD_CS)
sdcard = adafruit_sdcard.SDCard(spi, cs)
vfs = storage.VfsFat(sdcard)
try:
    storage.mount(vfs, "/sd")
    print("sd card mounted")
except ValueError:
    print("no SD card")

# to update the RTC, change set_clock to True
# otherwise RTC will remain set
# it should only be needed after the initial set
# if you've removed the coin cell battery
set_clock = False

if set_clock:
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2021, 10, 31, 00, 00, 00, 0, -1, -1))
    print("Setting time to:", t)
    rtc.datetime = t
    print()

# variable to hold RTC datetime
t = rtc.datetime

time.sleep(1)

# initial write to the SD card on startup
try:
    with open("/sd/co2.txt", "a") as f:
        # writes the date
        f.write('The date is {} {}{}\'.format(days[t.tm_wday], t.tm_mday,
```

©Adafruit Industries
t.tm_mon, t.tm_year))
    # writes the start time
    f.write('Start time: {}:{}:{}'.format(t.tm_hour, t.tm_min, t.tm_sec))
    # headers for data, comma-delimited
    f.write('CO2,Time
    # debug statement for REPL
    print("initial write to SD card complete, starting to log")
except ValueError:
    print("initial write to SD card failed - check card")

while True:
    try:
        # variable for RTC datetime
        t = rtc.datetime
        # append SD card text file
        with open("/sd/co2.txt", "a") as f:
            # read co2 data from SCD40
            co2 = scd4x.CO2
            # write co2 data followed by the time, comma-delimited
            f.write('{}={},{}:{}'.format(co2, t.tm_hour, t.tm_min, t.tm_sec))
            print("data written to sd card")
        # repeat every 30 seconds
        time.sleep(30)
    except ValueError:
        print("data error - cannot write to SD card")
        time.sleep(10)

Upload the Code and Libraries to the Feather RP2040

After downloading the Project Bundle, plug your Feather RP2040 into the computer's USB port. 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 Feather RP2040's CIRCUITPY drive.

- lib folder
- code.py

Your Feather RP2040 CIRCUITPY drive should look like this after copying the lib folder and the code.py file.
How the CircuitPython Code Works

The SCD-40 and RTC are controlled over I2C. The SCD-40 begins measuring CO2 levels with `start_periodic_measurement()`.

```python
# setup for I2C
i2c = board.I2C()
# setup for SCD40
scd4x = adafruit_scd4x.SCD4X(i2c)
# setup for RTC
rtc = adafruit_pcf8523.PCF8523(i2c)
# start measuring co2 with SCD40
scd4x.start_periodic_measurement()
```

The SD card is accessed using SPI. `board.D10` is used for the `SD_CS` pin on the RP2040 boards.

```python
# SPI SD_CS pin
SD_CS = board.D10

# SPI setup for SD card
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
cs = digitalio.DigitalInOut(SD_CS)
sdcard = adafruit_sdcard.SDCard(spi, cs)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")
```
Set the Time

The RTC is set using the `set_clock` variable. If `set_clock` is `True`, then the date and time is set to reflect what is entered in the `time.struct_time()` function. If `set_clock` is `False`, then the RTC is not reset.

```python
# to update the RTC, change set_clock to True
# otherwise RTC will remain set
# it should only be needed after the initial set
# if you've removed the coin cell battery
set_clock = False
if set_clock:
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2021, 10, 31, 00, 00, 00, 0, -1, -1))
    print("Setting time to:", t)
    rtc.datetime = t
    print()
```

Write Data

Before the loop, some data is written to the SD card's text file to assist in separating the data from different logging sessions. First, the date is written, followed by the time. Then, header labels are added, which will be useful when importing the data into a spreadsheet.

```python
# initial write to the SD card on startup
with open("/sd/co2.txt", "a") as f:
    # writes the date
    f.write('The date is {} {}\{}\{}\n'.format(days[t.tm_wday], t.tm_mday, t.tm_mon, t.tm_year))
    # writes the start time
    f.write('Start time: {}:{}:{}\n'.format(t.tm_hour, t.tm_min, t.tm_sec))
    # headers for data, comma-delimited
    f.write('CO2,Time\n')
    # debug statement for REPL
    print("initial write to SD card complete, starting to log")
```

In the loop, the CO2 level from the SCD-40, along with the current time, is written to the SD card's text file every 30 seconds.
while True:
    # variable for RTC datetime
    t = rtc.datetime
    # append SD card text file
    with open("/sd/co2.txt", "a") as f:
        # read co2 data from SCD40
        co2 = scd4x.CO2
        # write co2 data followed by the time, comma-delimited
        f.write('{},{}:{}:{}
'.format(co2, t.tm_hour, t.tm_min, t.tm_sec))
    # repeat every 30 seconds
    time.sleep(30)

Wiring

Solder socket headers to the FeatherWing Doubler and plug headers to the Feather RP2040 and the Adaloggier FeatherWing.

Tin two pieces of silicon wire. Solder these to two pins on the slide switch.
Add heat shrink to cover the switch's solder connections.

Solder one wire to the EN pin on the FeatherWing Doubler. Solder the second wire to GND on the FeatherWing Doubler. This creates the on/off switch connection when you plug in the Feather RP2040.
Assembly

Attach M2.5 standoffs to the SCD-40 STEMMA board.
Mount the SCD-40 to the top lid of the case using four M2.5 screws.
Mount the FeatherWing Doubler to the bottom lid of the case using M2.5 standoffs and screws.

Plug the LiPoly battery into the Feather RP2040. Then, plug the Feather RP2040 into the FeatherWing Doubler, securing the LiPoly battery underneath the Feather RP2040.
Plug the Adalogger FeatherWing into the FeatherWing Doubler.

Snap the bottom lid of the case into the main box. Secure the on/off switch into the cutout on the side. Make sure that the Feather RP2040's USB port is facing the case's USB-C cutout.
Connect the SCD-40 and Feather RP2040 with a STEMMA QT cable.

Snap the top lid onto the case to enclose the CO2 Data Logger.

---

**Usage**

After powering up, the Disconnected CO2 Data Logger will begin logging CO2 data with a timestamp to the Adalogger FeatherWing’s SD card every 30 seconds.

When you've finished logging, you can import the text file from the SD card, which is comma delimited, to your preferred spreadsheet editor to create charts to visualize your data.
This line graph was created with Google Sheets and shows the CO2 data in a public space over the course of a day. You can tell when a larger crowd gathered in the afternoon.

By collecting data in this way, you can really do whatever you want with it. It isn't tied to a service and the text file format makes it so that it travels easily between applications.