# Table of Contents

## Overview
- Parts

## CircuitPython for Feather nRF52840
- Set up CircuitPython Quick Start!

## Assemble the Adalogger and Feather Doubler
- Header Pins
- Header Sockets
- Combine Boards

## Code the Pulse Oximeter Logger
- Text Editor
- Code.py
- How it Works

## Log and Graph the Data
- Data Logging
- Graphing the Data
Overview

You can measure your blood oxygen level and pulse rate using a fingertip pulse oximeter -- and now with this project you can log that data wirelessly onto an SD card! Bring the comma-separated log file into your graphing software or quantified self apps for total control of your data!

Using a Feather nRF52840 Express with the Adalogger FeatherWing for real-time clock and SD card read/write capabilities. Code it all in CircuitPython and you'll be automatically logging data each time you use the pulse oximeter.
Parts

**Adafruit Feather nRF52840 Express**
The Adafruit Feather nRF52840 Express is the new Feather family member with Bluetooth Low Energy and native USB support featuring the nRF52840! It's...
https://www.adafruit.com/product/4062

**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

**SD/MicroSD Memory Card - 16GB Class 10 - Adapter Included**
Add speedy mega-storage in a jiffy using this 16 GB Class 10 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...
https://www.adafruit.com/product/2693
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

Finger Pulse Oximeter with Bluetooth LE
Discontinued - you can grab the Finger Pulse Oximeter - BM1000 instead!
This Finger Pulse...
https://www.adafruit.com/product/4582

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

USB A/Micro Cable - 2m
This is your standard USB A-Plug to Micro-USB cable. It's 2 meters long so you'll have plenty of cord to work with for those longer extensions.
https://www.adafruit.com/product/2185
CircuitPython for Feather nRF52840

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.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

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

Click the link above to download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).
Plug your Feather nRF52840 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 next to the USB connector on your board, and you will see the NeoPixel RGB LED turn green (identified by the arrow in the image). If it turns red, check the USB cable, try another USB port, etc. Note: The little red LED next to the USB connector will pulse red. That's ok!

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 FTHR840BOOT.

Drag the adafruit_circuitpython_etc.uf2 file to FTHR840BOOT.
The LED will flash. Then, the FTHR840BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

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

Assemble the Adalogger and Feather Doubler

We'll use a FeatherWing Doubler to connect the Adalogger FeatherWing to the Feather nRF52840. This provides a nice, accessible layout.

The FeatherWing Doubler provides the same connections between the pins of the Adalogger and the Feather as if they were stacked in the more typical arrangement, so all data, power, and ground connections will work as normal.

You may, of course, choose a different arrangement, such as stacking the boards if you like.
Header Pins
Solder male headers to the underside of both the Feather nRF52840 and the Adalogger FeatherWing.

This guide has good tips () on soldering in the headers.

You will also need to install the battery (with the + side up) into the Adalogger’s battery holder -- this powers the real-time clock so it is accurate even when the Feather is powered down.

You can also insert the SD card at this point.
Header Sockets
Solder female socket headers to the top side of the Feather Doubler.
Combine Boards
Now, you can insert the Adalogger and the Feather into the Doubler.
Code the Pulse Oximeter Logger

Once your Feather nRF52840 is set up with CircuitPython, you'll also need to add some libraries. Follow this page for information on how to download and add libraries to your Feather.

From the library bundle you downloaded in that guide page, transfer the following libraries onto the Feather's /lib directory:

- adafruit_ble
- adafruit_ble_berrymed_pulse_oximeter
- adafruit_bus_device
- adafruit_register
- neopixel.mpy
- adafruit_pcf8523.mpy

Text Editor

Adafruit recommends using the Mu editor for using your CircuitPython code with the Feather boards. You can get more info in this guide.

Alternatively, you can use any text editor that saves files.

Code.py

Copy the code below and paste it into Mu. Then, save it to your Feather as code.py.

```python
# SPDX-FileCopyrightText: 2020 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

""
Read data from a BerryMed pulse oximeter, model BM1000C, BM1000E, etc.
Run this on Feather nRF52840
Log data to SD card on Autologger FeatherWing
""

# Protocol defined here:
# https://github.com/zh2x/BCI_Protocol
# Thanks as well to:
# https://github.com/ehborisov/BerryMed-Pulse-Oximeter-tool
# https://github.com/ScheindorfHyenetics/berrymedBluetoothOxymeter
```
# The sensor updates the readings at 100Hz.

```python
import time
import sdcardio
import board
import busio
import storage
import adafruit_pcf8523
import _bleio
import adafruit_ble
from adafruit_ble.advertising.standard import Advertisement
from adafruit_ble.services.standard.device_info import DeviceInfoService
from adafruit_ble_berrymed_pulse_oximeter import BerryMedPulseOximeterService

# Logging setup
SD_CS = board.D10
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
sd_card = sdcardio.SDCard(spi, SD_CS)
vfs = storage.VfsFat(sd_card)
storage.mount(vfs, "/sd_card")

log_interval = 2  # you can adjust this to log at a different rate

# RTC setup
I2C = busio.I2C(board.SCL, board.SDA)
rtc = adafruit_pcf8523.PCF8523(I2C)

days = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

set_time = False
if set_time:  # change to True if you want to write the time!
    #             year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2020, 4, 21, 18, 13, 0, 2, -1, -1))
    # you must set year, mon, date, hour, min, sec and weekday
    # yearday not supported, isdst can be set but we don't use it at this time
    print("Setting time to:", t)  # uncomment for debugging
    rtc.datetime = t
    print()

# PyLint can't find BLERadio for some reason so special case it here.
ble = adafruit_ble.BLERadio()  # pylint: disable=no-member

pulse_ox_connection = None

while True:
    t = rtc.datetime
    print("Scanning for Pulse Oximeter...")
    for adv in ble.start_scan(Advertisement, timeout=5):
        name = adv.complete_name
        if not name:
            continue
        # "BerryMed" devices may have trailing nulls on their name.
        if name.strip("\x00") == "BerryMed":
            pulse_ox_connection = ble.connect(adv)
            print("Connected")
            break

    # Stop scanning whether or not we are connected.
    ble.stop_scan()
    print("Stopped scan")

    try:
        if pulse_ox_connection and pulse_ox_connection.connected:
            print("Fetch connection")
            if DeviceInfoService in pulse_ox_connection:
                dis = pulse_ox_connection[DeviceInfoService]
                try:
                    # This is where you would write to the SD card.
                    # For example:
                    # dis.device_info_reply()
                    # dis.save("/sd_card/my_file.txt")
                    # print("Data saved")

                except:
                    pass
```

©Adafruit Industries
manufacturer = dis.manufacturer
except AttributeError:
    manufacturer = "(Manufacturer Not specified)"
try:
    model_number = dis.model_number
except AttributeError:
    model_number = "(Model number not specified)"
print("Device: ", manufacturer, model_number)
else:
    print("No device information")

pulse_ox_service = pulse_ox_connection[BerryMedPulseOximeterService]

while pulse_ox_connection.connected:
    values = pulse_ox_service.values
    if values is not None:
        # unpack the message to 'values' list
        valid, spo2, pulse_rate, pleth, finger = values
        if not valid:
            continue
        if (pulse_rate == 255):  # device sends 255 as pulse until it has a valid read
            continue
        print("SpO2: {}%  | " .format(spo2),
            "Pulse Rate: {} BPM  | ".format(pulse_rate),
            "Pleth: {}".format(pleth),
            )
    # print((pleth,))  # uncomment to see graph on Mu plotter

    try:  # logging to SD card
        with open("/sd_card/log.txt", "a") as sdc:
            t = rtc.datetime
            sdc.write({}
                "{} " .format(days[t.tm_wday],
            t.tm_mday,
            t.tm_mon,
            t.tm_year,
            t.tm_hour,
            t.tm_min,
            t.tm_sec
            )
        sdc.write({}
            "{}, {}, {:.2f}\n" .format(,
            spo2, pulse_rate, pleth
            )
        )
        time.sleep(log_interval)
    except OSError:
        pass
    except RuntimeError:
        pass
    except _bleio.ConnectionError:
        try:
            pulse_ox_connection.disconnect()
        except _bleio.ConnectionError:
            pass
    pulse_ox_connection = None
How it Works

The code does the following things:

- Imports necessary libraries, including adafruit_ble_berrymed_pulse_oximeter library
- Sets up SD card interface to use as a file system
- Sets up the real-time clock (RTC) on the I2C bus for setting and then keeping track of the day, date, and time
- Scan for and connect to a Bluetooth LE peripheral advertising the BerryMed pulse oximeter service
- Receive pulse oximeter service messages and unpack their values into SpO2%, pulse rate, and pleth (pulse strength graph)
- When a valid pulse oximeter message is received, log the values along with the day, date, and time from the RTC onto the SD card in a file called log.txt
- Repeat this process after the specified log_interval (a couple of seconds for example)
- If/when the connection is dropped, go back to BLE search mode

Now, let's try it out!

Log and Graph the Data

Data Logging

When you're ready to log the data, here are the steps (as shown in the video above):

- While the Feather is unpowered, insert the SD card into the Adallogger card slot
- Power on the Feather using USB or battery power
- Place your finger in the pulse oximeter reader and power it on
- Wear the pulse oximeter for as long as you'd like to log the data
- Remove the pulse oximeter from your finger and allow it to power down
- Power down the Feather
- Remove the SD card

The data will have been written to the SD card in a file called log.txt. Let's look at how to graph that data.
Graphing the Data

Using an SD card reader, and SD card adapter if necessary, plug your SD card into your computer. You'll have a file called log.txt appear on it. Copy that file to your computer.

The log file contents will look something like this:

- Tuesday 21/4/2020 18:15:34, 99, 74, 46.00
- Tuesday 21/4/2020 18:15:36, 99, 75, 67.00
- Tuesday 21/4/2020 18:15:38, 99, 75, 26.00
- Tuesday 21/4/2020 18:15:40, 99, 75, 91.00
- Tuesday 21/4/2020 18:15:43, 99, 75, 91.00
- Tuesday 21/4/2020 18:15:45, 97, 76, 71.00
- Tuesday 21/4/2020 18:15:47, 97, 76, 16.00

This has the day/date/time stamp followed by the SpO2%, pulse, and pleth, all separated by commas.

Note that some spreadsheet programs require you to change the extension to ".csv". Otherwise they don't know what to look for and may import incorrectly.

Next, open the file into a spreadsheet program of your choice. We've used Google Sheets. Open a new spreadsheet.

Click File and choose Import.
At the top of the Import file window, click Upload. Then click "Select a file from your computer" and choose your log.txt file.

No changes are needed in the Import File dialog box - the defaults work for our project.

Click Import data.

Now we have our data! It's almost ready to graph.

You can see here that the automatic delimiter detection worked well -- it placed the time-date stamp in one column and each piece of data in its own column, all based on the commas found in the original log.txt file.
You can now select the data you want to graph and click Insert > Chart in the menus.

You can adjust the chart drawing style, labels, colors, and more to your satisfaction.

Here's an intro guide on using charts in Sheets.

There's also some great info in this guide to tuning your data logging graphs.

While your data should ideally be pretty consistent, you can try running around and doing jumping jacks to get some bigger changes in the data you log, just be safe!