BLE Vibration Bracelet

Created by Ruiz Brothers

https://learn.adafruit.com/ble-vibration-bracelet

Last updated on 2024-06-21 10:21:37 AM EDT

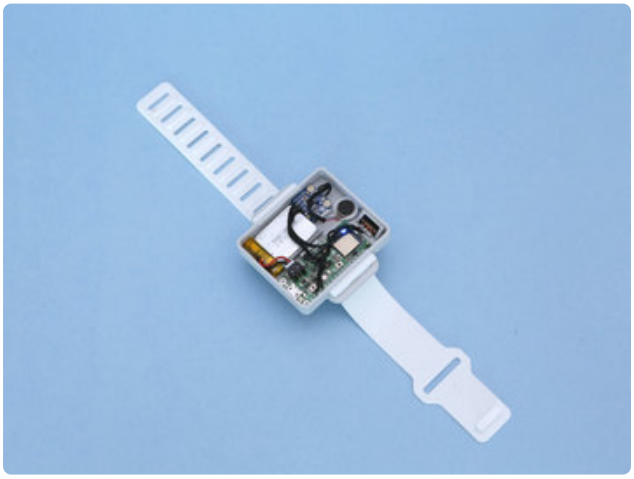
Table of Contents

Usage 36

Overview

Mindfulness & BLE Notifications

Build yourself a bracelet that buzzes when you've received notifications from an iOS device using Adafruit Feather Sense and DRV2605L breakout. Get subtle haptic feedback and a NeoPixel LED to indicate the app. It's also great for reminding yourself to get up and walk away from your desk or as a way to have a new awareness of how the perception of passing time varies based on what you're doing.



Wearable Case

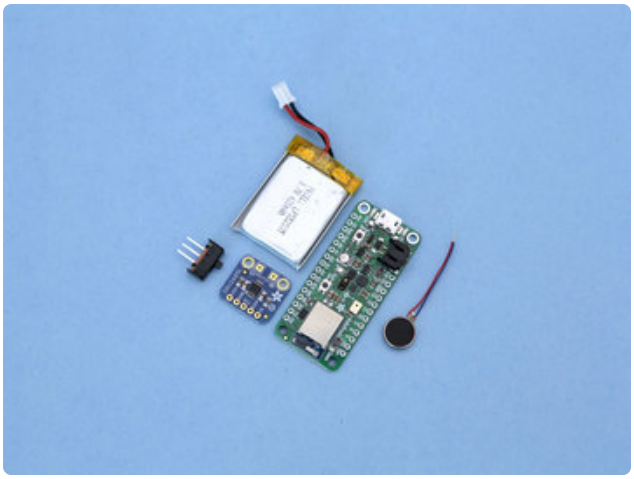
Brands are 3D printed in NinjaFlex TPU filament making a flexible wearable device. Strap features several slots for fitting different sizes wrists. A snap fit cover lets you easy remove and get access to components and on-board sensors.

Prerequisite Guides

The following learn guides will provide additional information and demo code.

- DRV2605L Learn Guide (https://adafru.it/jDA)
- Adafruit Feather Sense Learn Guide (https://adafru.it/L6d)
- Adafruit Feather nRF52840 Express Guide (https://adafru.it/DLI)
- Getting Started with CircuitPython and BLE (https://adafru.it/FxH)

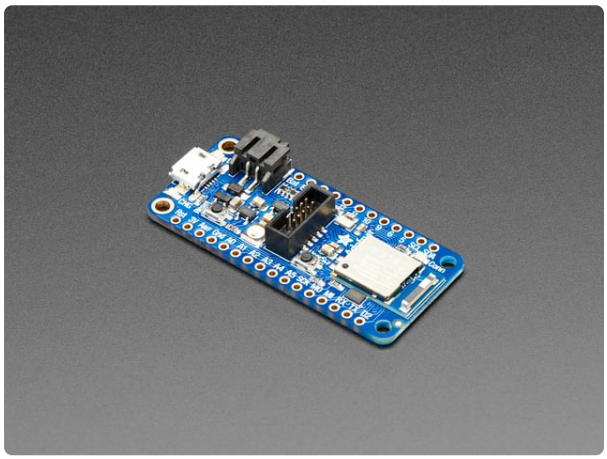
Parts



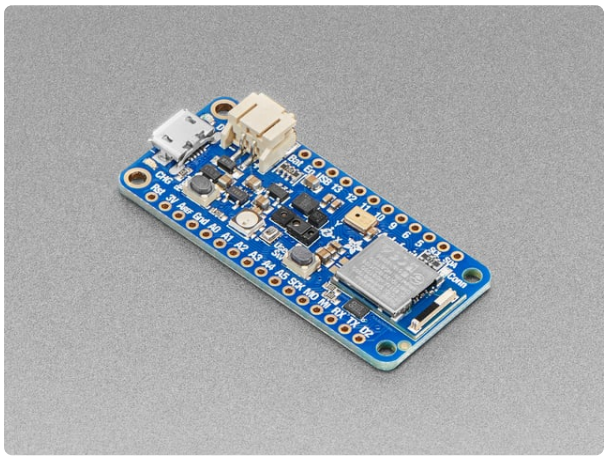
Adafruit Feather Sense nRF52840 (http://adafru.it/4516)
Adafruit Feather nRF52840 Express (http://adafru.it/4062)
DRV2605L Haptic Motor Controller (http://adafru.it/2305)
Vibration Mini Motor Disc (http://adafru.it/1201)
3.7v 420mAh Battery (http://adafru.it/4236)
Slide Switch (http://adafru.it/1578)
10-wire Silicone Ribbon Wire (http://adafru.it/3890)

Adafruit Feather nRF52840 Express

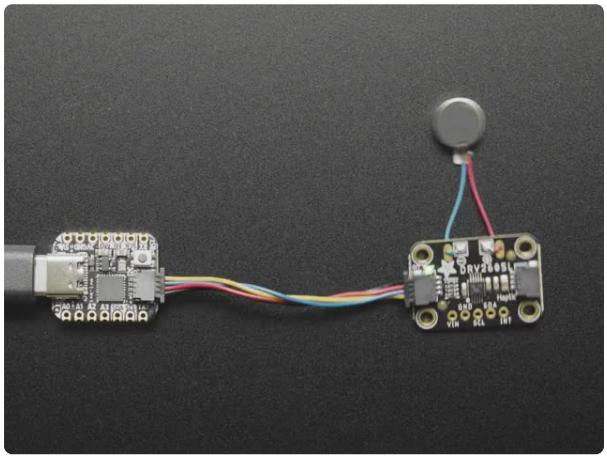
Don't have the Adafruit Feather nRF52840 Sense? No worries! You can also use the Adafruit Feather nRF52840 Express (http://adafru.it/4062).



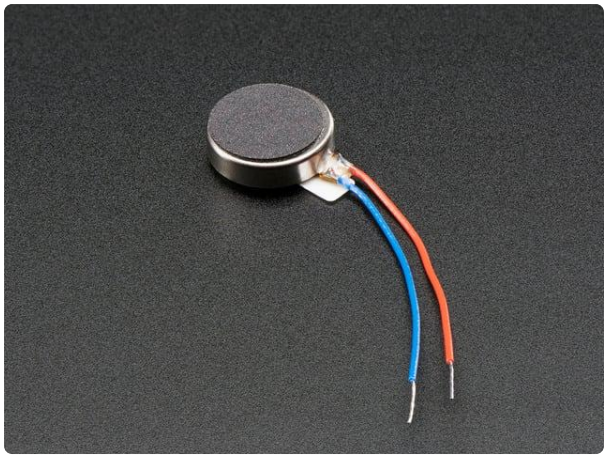
Adafruit Feather nRF52840 Express
The Adafruit Feather nRF52840 Express is the new Feather family member with Bluetooth Low Energy and native USB support featuring the nRF52840! It's...
https://www.adafruit.com/product/4062



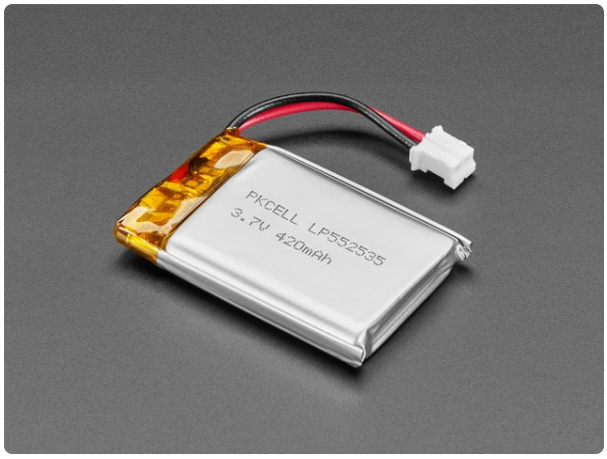
Adafruit Feather nRF52840 Sense
The Adafruit Feather Bluefruit Sense takes our popular Feather nRF52840 Express and adds a smorgasbord of sensors...
https://www.adafruit.com/product/4516



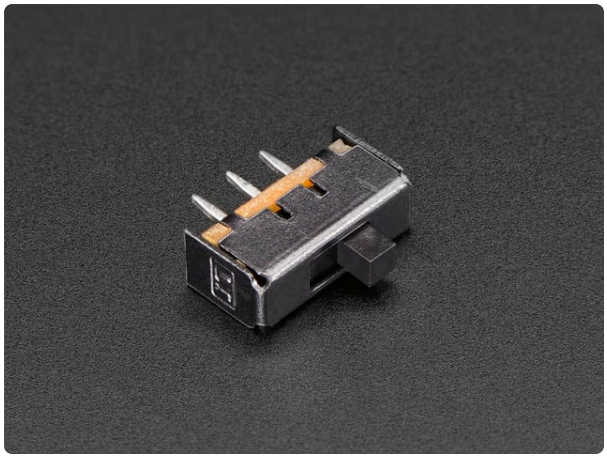
Adafruit DRV2605L Haptic Motor Controller - STEMMA QT / Qwiic
The DRV2605 from TI is a fancy little motor driver. Rather than controlling a stepper motor or DC motor, its designed specifically for controlling haptic motors -...
https://www.adafruit.com/product/2305



[Vibrating Mini Motor Disc]
BZZZZZZZZZZ Feel that? That's your little buzzing motor, and for any haptic feedback project you'll want to pick up a few of them. These vibe motors are tiny discs,...
https://www.adafruit.com/product/1201



[Lithium Ion Polymer Battery with Short Cable - 3.7V 420mAh]
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/4236



[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

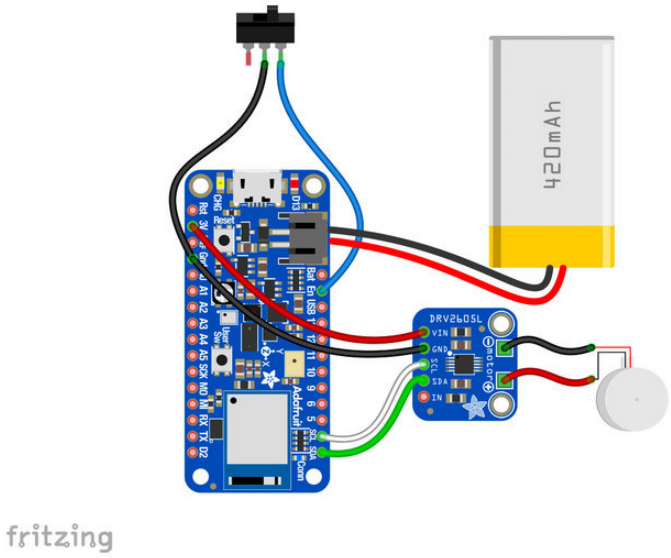
[Silicone Cover Stranded-Core Ribbon Cable - 10 Wire 1 Meter Long]
For those who are fans of our silicone-covered wires, but are always looking to up their wiring game. We now have Silicone Cover Ribbon cables! These may look...
https://www.adafruit.com/product/3890

Circuit Diagram

The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package Fritzing (https://adafru.it/oEP).

Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts (https://adafru.it/AYZ).

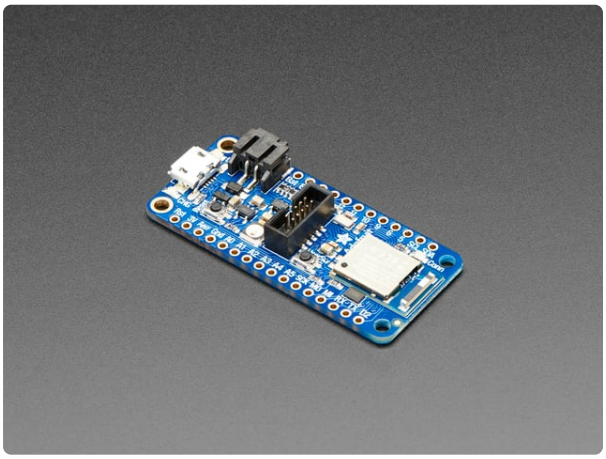


Wired Connections

- **VCC** from DRV2605L to **VCC** on Feather
- **GND** from DRV2605L to **GND** on Feather
- **SCL** from DRV2605L to **SCL** on Feather
- **SDA** from DRV2605L to **SDA** on Feather
- Switch to **GND**
- Switch to **EN**
- Motor to **GND** on DRV2605L
- Motor to **VCCC** on DRV2605L

Adafruit Feather nRF52840 Express

The Adafruit Feather nRF52840 Express can also be used. Use the exact same wiring.

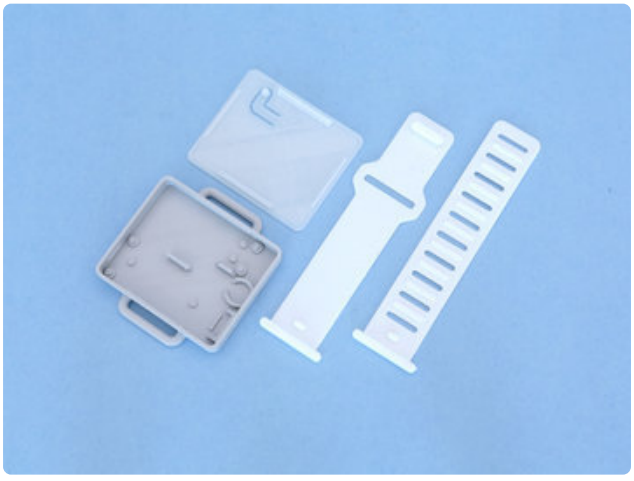


Adafruit Feather nRF52840 Express
The Adafruit Feather nRF52840 Express is the new Feather family member with Bluetooth Low Energy and native USB support featuring the nRF52840! It's...
https://www.adafruit.com/product/4062

Powering

The Adafruit board can be powered via USB or JST using a 3.7v lipo battery. In this project, a 420mAh lipo battery is used. The lipo battery is rechargeable via the USB port on the board. The switch is wired to the **enable** and **ground** pins on the board.

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.

vib-case.stl
vib-cover.stl
vib-band.stl
vib-strap.stl

Download CAD from Fusion 360

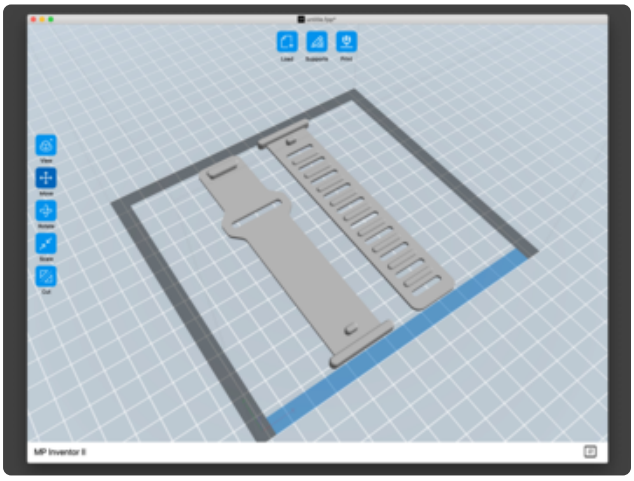
https://adafru.it/1a3b

Download CAD from Thingiverse

https://adafru.it/Lek

Download Vibration Watch STEP and F3Z

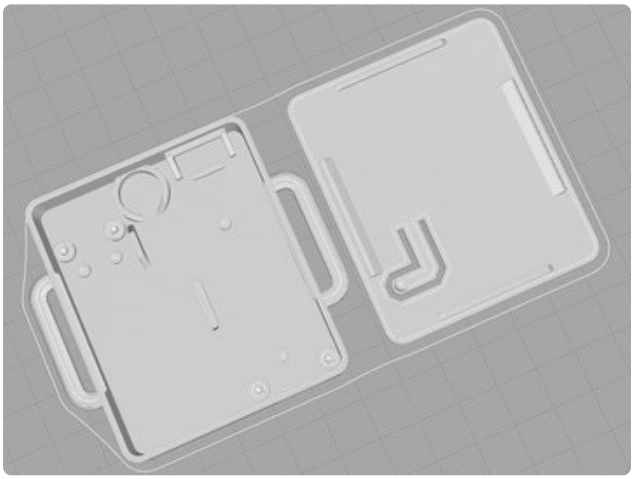
https://adafru.it/1a3c



Ninjaflex TPU

The strap and band are printed in Ninjaflex TPU filament. Use the following setting for slicing software. TPU material profiles are recommenced.

printing speed: 30mm/s
nozzle temperature: 240c
retraction: off
infill: 10%
perimeters: 2

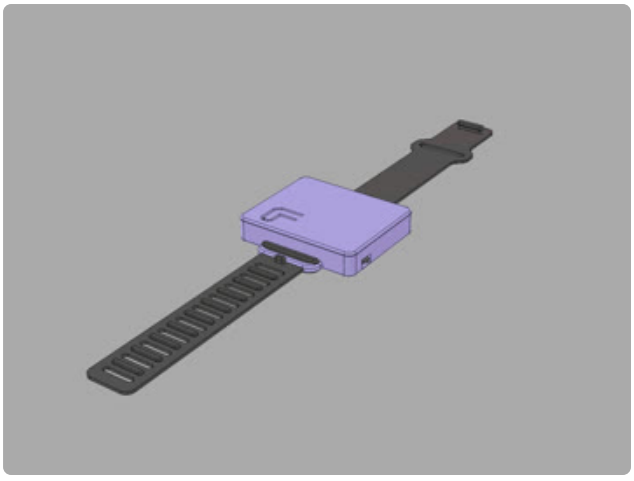


Slicing Case and Cover

No supports are required. Slice with setting for PLA material. Use white or translucent filament for the cover to diffuser the NeoPixel LED.

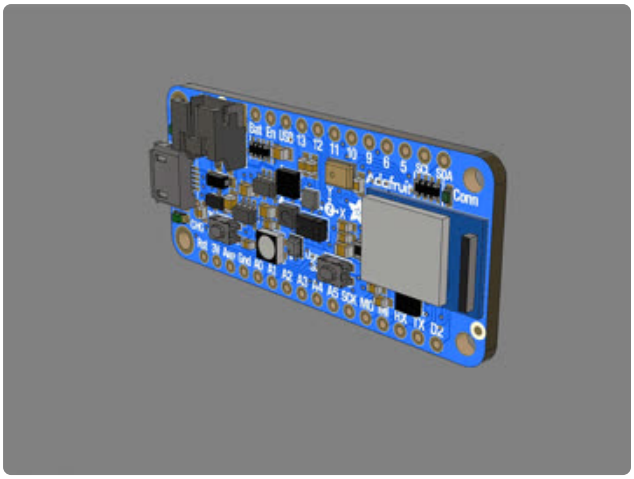
The parts were sliced using CURA using the slice settings below.

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



CAD Assembly

The Feather is installed at an angle and secured down by tabs with nubs. The vibe motor is press fitted into holder and secured with sticky adhesive backing. The DVR2605L is press fitted into the standoffs. The slide switch is press fitted into the holder. The tabs on the band and strap are press fitted through the loops in the case. The cover is press fitted on top of the case.



Design Source Files

The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo (https://adafru.it/AW8).

CircuitPython on Feather Sense

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.

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

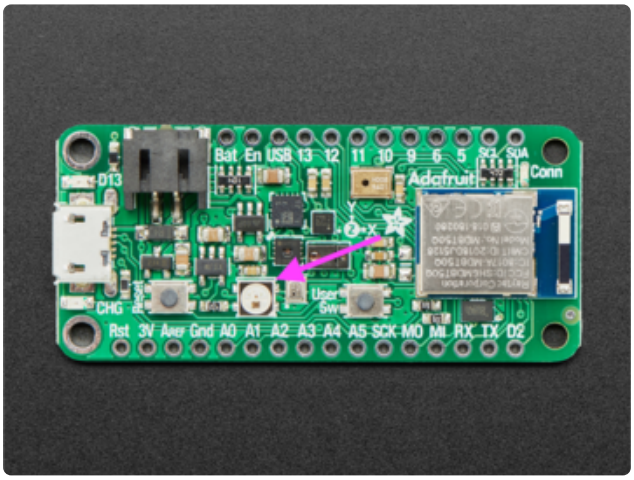
https://adafru.it/JqE

Click the link above to download the latest UF2 file.

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

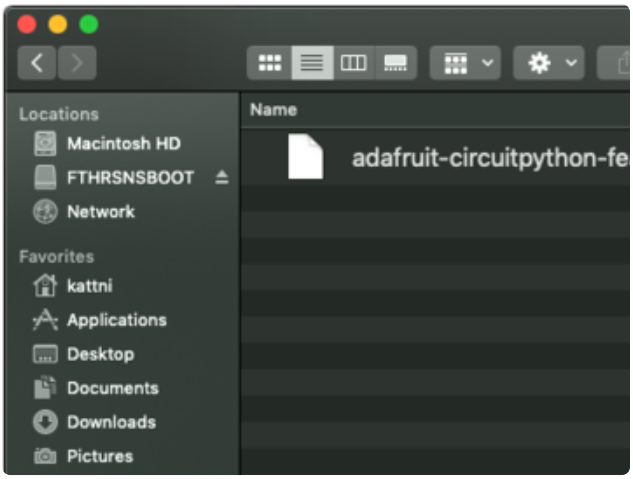
Plug your Feather Sense into your computer using a known-good USB cable.

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.

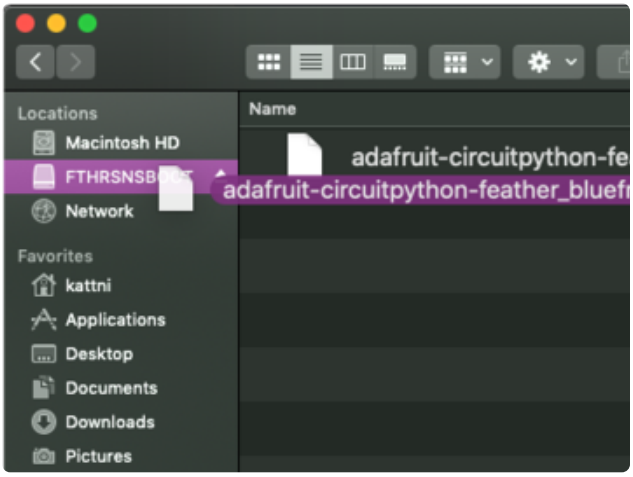


Double-click the **Reset** button next to the USB connector on your board, and you will see the NeoPixel RGB LED turn green (identified by the arrow in the image). If it turns red, check the USB cable, try another USB port, etc. **Note:** The little red LED next to the USB connector will pulse red. That's ok!

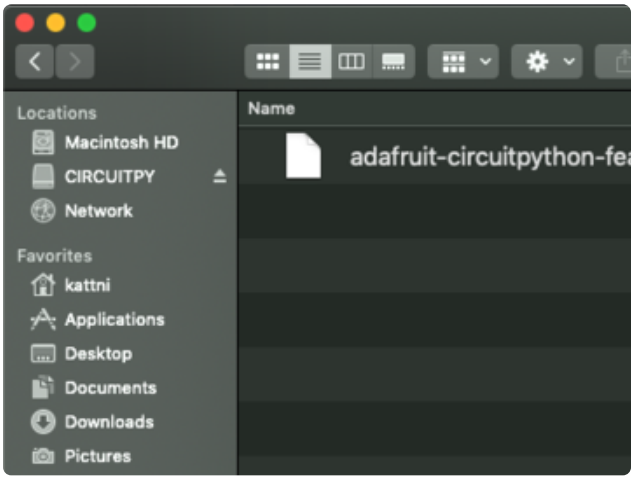
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



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



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



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

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

Note: Some early release Sense boards had the drive named FTHR840BOOT. You can still copy .UF2s to the board, just copy to the board name appearing when the board is plugged in.

Feather Sense CircuitPython Libraries

The Feather Sense is packed full of sensors. Now that you have CircuitPython installed on your Feather Sense, you'll need to install a base set of CircuitPython libraries to use the features of the board with CircuitPython.

Follow these steps to get the necessary libraries installed.

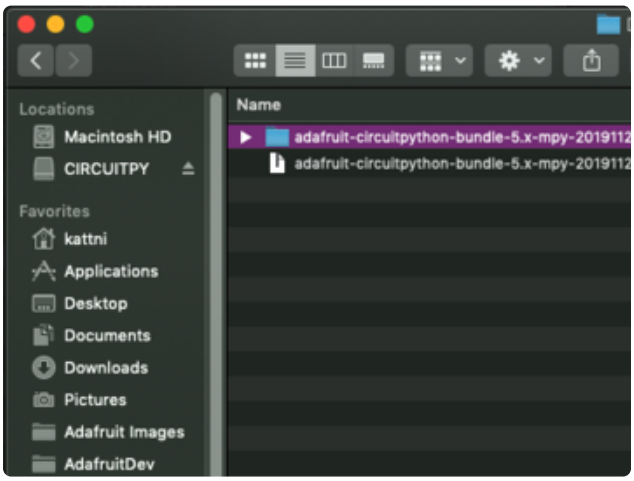
Installing CircuitPython Libraries on your Feather Sense

If you do not already have a **lib** folder on your **CIRCUITPY** drive, create one now.

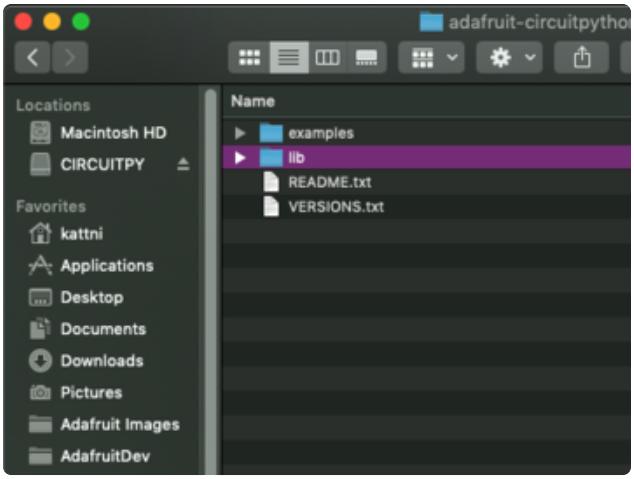
Then, download the CircuitPython library bundle that matches your version of CircuitPython from CircuitPython.org.

Download the latest library bundle from circuitpython.org

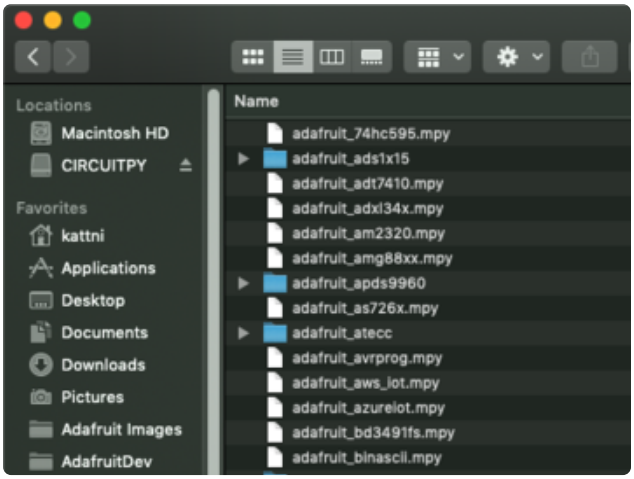
https://adafru.it/ENC



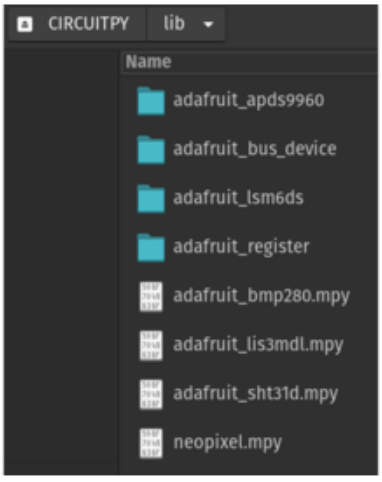
The bundle downloads as a .zip file. Extract the file. Open the resulting folder.



Open the **lib** folder found within.



Once inside, you'll find a lengthy list of folders and **.mpy** files. To install a CircuitPython library, you drag the file or folder from the bundle lib folder to the **lib** folder on your **CIRCUITPY** drive.



Copy the following folders and files from the bundle **lib** folder to the **lib** folder on your **CIRCUITPY** drive:

adafruit_apds9960
adafruit_bmp280.mpy
adafruit_bus_device
adafruit_lis3mdl.mpy
adafruit_lsm6ds
adafruit_register
adafruit_sht31d.mpy
neopixel.mpy

Your **lib** folder should look like the image on the left. These libraries will let you run the demos in the Feather Sense guide.

Coding the BLE Vibration Bracelet

Adding Libraries

Once you've finished setting up your Feather Sense with CircuitPython, you can add these additional libraries to the **lib** folder:

- **adafruit_ble**
- **adafruit_bluefruit_connect**
- **adafruit_led_animation**
- **adafruit_ble_apple_notification_center.mpy**
- **adafruit_ble_broadcastnet.mpy**
- **adafruit_drv2605.mpy**

Then, you can click on the Download: Project Zip link above the code to download the code file.

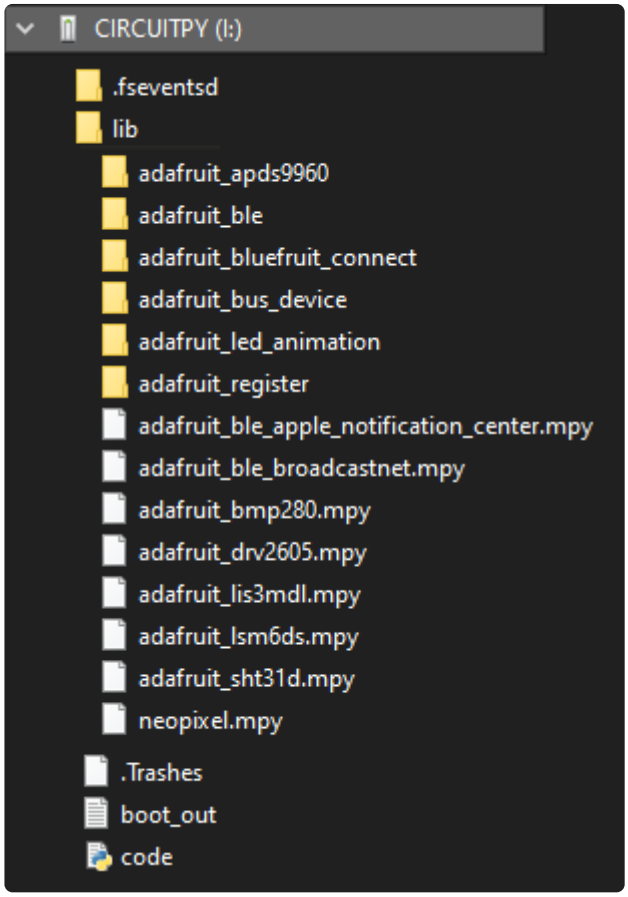
```
# SPDX-FileCopyrightText: 2020 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import busio
import neopixel
import adafruit_drv2605
import adafruit_led_animation.color as color
import adafruit_ble
from adafruit_ble.advertising.standard import SolicitServicesAdvertisement
from adafruit_ble.services.standard import CurrentTimeService
from adafruit_ble_apple_notification_center import AppleNotificationCenterService
from digitalio import DigitalInOut, Direction

#  setup for onboard NeoPixel
pixel_pin = board.NEOPIXEL
num_pixels = 1

pixel = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=0.3, auto_write=False)

#  setup for haptic motor driver
i2c = busio.I2C(board.SCL, board.SDA)
drv = adafruit_drv2605.DRV2605(i2c)

#  onboard blue LED
blue_led = DigitalInOut(board.BLUE_LED)
blue_led.direction = Direction.OUTPUT

#  setup for BLE
ble = adafruit_ble.BLERadio()
if ble.connected:
    for c in ble.connections:
        c.disconnect()

advertisement = SolicitServicesAdvertisement()
```

```
#  adds ANCS and current time services for BLE to advertise
advertisement.solicited_services.append(AppleNotificationCenterService)
advertisement.solicited_services.append(CurrentTimeService)

#  state machines
current_notification = None #  tracks the current notification from ANCS
current_notifications = {} #  array to hold all current notifications from ANCS
cleared = False #  state to track if notifications have been cleared from ANCS
notification_service = None #  holds the array of active notifications from ANCS
all_ids = [] #  array to hold all of the ids from ANCS
hour = 0 #  used to track when it is on the hour for the mindfulness reminder
mindful = False #  state used to track if it is time for mindfulness
vibration = 16 #  vibration effect being used for the haptic motor

APP_COLORS = {
    "com.basecamp.bc3-ios": color.YELLOW, #  Basecamp
    "com.apple.MobileSMS": color.GREEN, #  Texts
    "com.hammerandchisel.discord": color.PURPLE, #  Discord
    "com.apple.mobilecal": color.CYAN, #  Calendar
    "com.apple.mobilephone": color.GREEN, #  Phone
    "com.google.ios.youtube": color.ORANGE, #  YouTube
    "com.burbn.instagram": color.MAGENTA, #  Instagram
    "com.apple.mobilemail": color.CYAN #  Apple Email
}

#  function for blinking NeoPixel
#  blinks: # of blinks
#  speed: how fast/slow blinks
#  color1: first color
#  color2: second color
def blink_pixel(blinks, speed, color1, color2):
    for _ in range(0, blinks):
        pixel.fill(color1)
        pixel.show()
        time.sleep(speed)
        pixel.fill(color2)
        pixel.show()
        time.sleep(speed)

#  function for haptic motor vibration
#  num_zzz: # of times vibrates
#  effect: type of vibration
#  delay: time between vibrations
def vibe(num_zzz, effect, delay):
    drv.sequence[0] = adafruit_drv2605.Effect(effect)
    for _ in range(0, num_zzz):
        drv.play()  # play the effect
        time.sleep(delay)  # for 0.5 seconds
        drv.stop()

#  start BLE
ble.start_advertising(advertisement)

while True:

    blue_led.value = False
    print("Waiting for connection")

    #  NeoPixel is red when not connected to BLE
    while not ble.connected:
        blue_led.value = False
        pixel.fill(color.RED)
        pixel.show()
    print("Connected")

    while ble.connected:
        blue_led.value = True #  blue LED is on when connected
        all_ids.clear()
        for connection in ble.connections:
```

```
    if not connection.paired:
        #  pairs to phone
        connection.pair()
        print("paired")
    #  allows connection to CurrentTimeService
    cts = connection[CurrentTimeService]
    notification_service = connection[AppleNotificationCenterService]
#  grabs notifications from ANCS
current_notifications = notification_service.active_notifications

for notif_id in current_notifications:
    #  adds notifications into array
    notification = current_notifications[notif_id]
    all_ids.append(notif_id)

#  all_ids.sort(key=lambda x: current_notifications[x]._raw_date)

if current_notification and current_notification.removed:
    # Stop showing the latest and show that there are no new notifications.
    current_notification = None
    pixel.fill(color.BLACK)
    pixel.show()

if not current_notification and not all_ids and not cleared:
    #  updates cleared state for notification
    cleared = True
    #  turns off NeoPixel when notifications are clear
    pixel.fill(color.BLACK)
    pixel.show()

elif all_ids:
    cleared = False
    if current_notification and current_notification.id in all_ids:
        index = all_ids.index(current_notification.id)
    else:
        index = len(all_ids) - 1
        notif_id = all_ids[index]
    #  if there is a notification:
    if not current_notification or current_notification.id != notif_id:
        current_notification = current_notifications[notif_id]
        #  if the notification is from an app that is not
        #  defined in APP_COLORS then the NeoPixel will be white
        if current_notification.app_id not in APP_COLORS:
            notif_color = color.WHITE
        #  if the notification is from an app defined in
        #  APP_COLORS then the assigned color will show
        else:
            notif_color = APP_COLORS[current_notification.app_id]
        #  parses notification info into a string
        category = str(notification).split(" ", 1)[0]
        #  haptic motor vibrates
        vibe(2, vibration, 0.5)
        #  all info for notification is printed to REPL
        print('-'*36)
        print("Msg #%d - Category %s" % (notification.id, category))
        print("From app:", notification.app_id)
        if notification.title:
            print("Title:", notification.title)
        if notification.subtitle:
            print("Subtitle:", notification.subtitle)
        if notification.message:
            print("Message:", notification.message)
        #  NeoPixel blinks and then stays on until cleared
        blink_pixel(2, 0.5, notif_color, color.BLACK)
        pixel.fill(notif_color)
        pixel.show()
#  if it's on the hour:
if cts.current_time[4] == hour and not mindful:
    print(cts.current_time[4])
```

```
            print("mindful time")
            #  haptic motor vibrates
            vibe(5, vibration, 1)
            #  NeoPixel blinks and then stays on
            blink_pixel(5, 1, color.BLUE, color.BLACK)
            mindful = True
            pixel.fill(color.BLUE)
            pixel.show()
            print("hour = ", hour)
        #  if it's no longer on the hour:
        if cts.current_time[4] == (hour + 1) and mindful:
            #  NeoPixel turns off
            mindful = False
            pixel.fill(color.BLACK)
            pixel.show()
            print("mindful time over")

    #  if BLE becomes disconnected then blue LED turns off
    #  and BLE begins advertising again to reconnect
    print("Disconnected")
    blue_led.value = False
    print()
    ble.start_advertising(advertisement)
    notification_service = None
```

Your Feather Sense **CIRCUITPY** drive should look like this after you load the libraries and **code.py** file:



CircuitPython Code Walkthrough

Inspiration

The code for this project is based on two previous projects on the Adafruit Learn System: John Park's Bluefruit TFT Gizmo ANCS Notifier for iOS (https://adafru.it/IIB) and Becky Stern's Buzzing Mindfulness Bracelet (https://adafru.it/jDC). Be sure to check out both of those Learn Guides for additional project inspiration.

Setup

CircuitPython Libraries

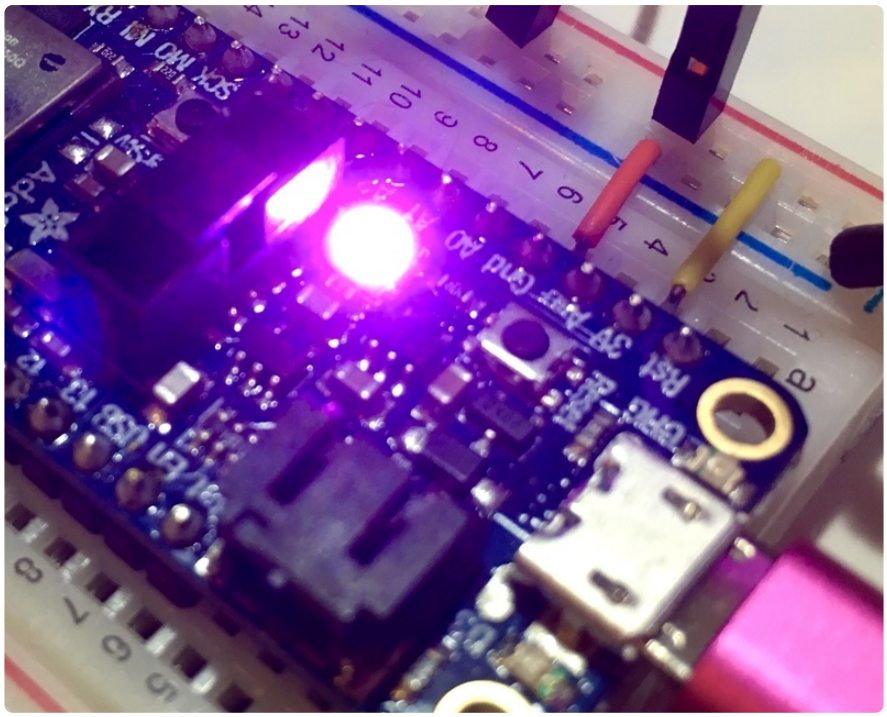
The CircuitPython code begins by importing the libraries.

```
import time
import board
import busio
import neopixel
import adafruit_drv2605
import adafruit_led_animation.color as color
import adafruit_ble
from adafruit_ble.advertising.standard import SolicitServicesAdvertisement
from adafruit_ble.services.standard import CurrentTimeService
from adafruit_ble_apple_notification_center import AppleNotificationCenterService
from digitalio import DigitalInOut, Direction
```

LEDs and Motors

After the libraries are imported, the onboard NeoPixel is setup as `pixel_pin`. This NeoPixel will serve as the visual notifier on the vibration bracelet.

```
pixel_pin = board.NEOPIXEL
num_pixels = 1

pixel = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=0.3, auto_write=False)
```



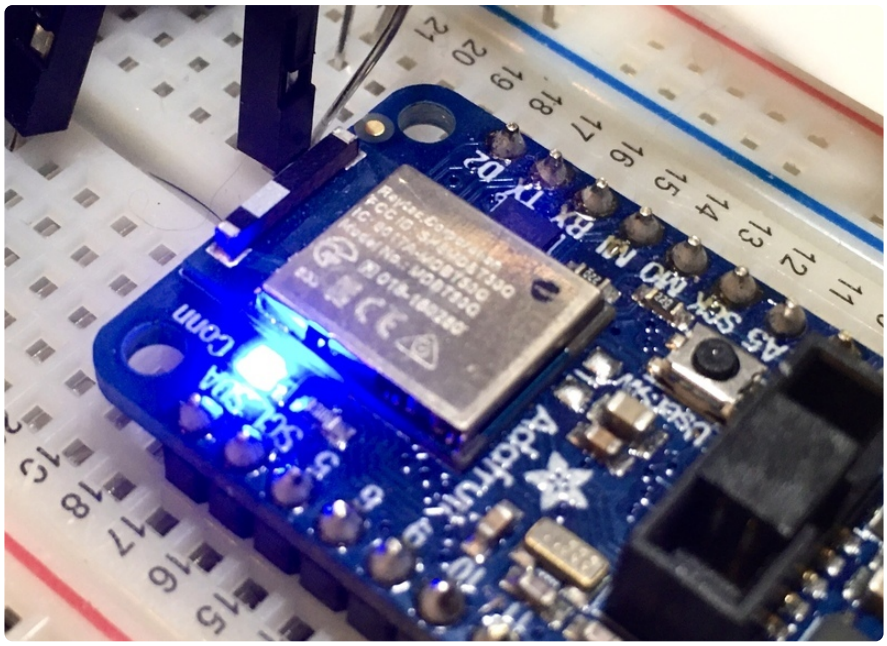
Up next, the DRV2605L driver breakout is setup to communicate over I2C. This driver is what powers the haptic motor to make the vibration bracelet vibrate.

```
i2c = busio.I2C(board.SCL, board.SDA)
drv = adafruit_drv2605.DRV2605(i2c)
```

For more information on the DRV2605L, check out this Learn Guide

https://adafru.it/jDA

In addition to the onboard NeoPixel, the onboard blue LED on the Feather Sense is also used for this project. This LED is located towards the back of the Feather, next to the nRF52840 chip. Later in the code, it will indicate whether or not your iOS device is connected to BLE.



```
blue_led = DigitalInOut(board.BLUE_LED)
blue_led.direction = Direction.OUTPUT
```

Setting Up BLE

Next in the setup is BLE, allowing for BLE connectivity and functionality to be accessed via the `adafruit_ble` library.

```
ble = adafruit_ble.BLERadio()
if ble.connected:
    for c in ble.connections:
        c.disconnect()
```

There are a lot of ways to use BLE, but in this project the Apple Notification Center Service (ANCS) and Current Time Service are used. These two services are called out to be connected to when a BLE connection is established.

```
advertisement = SolicitServicesAdvertisement()

advertisement.solicited_services.append(AppleNotificationCenterService)
advertisement.solicited_services.append(CurrentTimeService)
```

State Machines

Following the setup are state machines that will be used in the loop. Their purpose is commented below in the code.

```
#  state machines
current_notification = None #  tracks the current notification from ANCS
current_notifications = {} #  array to hold all current notifications from ANCS
```

```
cleared = False #  state to track if notifications have been cleared from ANCS
notification_service = None #  holds the array of active notifications from ANCS
all_ids = [] #  array to hold all of the ids from ANCS
hour = 0 #  used to track when it is on the hour for the mindfulness reminder
mindful = False #  state used to track if it is time for mindfulness
vibration = 16 #  vibration effect being used for the haptic motor
```

Which app is notifying me?

The vibration bracelet allows you to be away from your phone yet still be aware of any incoming notifications. The `APP_COLORS` array lists the application ID's of a few commonly used apps. Each app has its own unique application ID that is stored in ANCS.

In addition to the notification ID's, NeoPixel colors are defined in the code and linked to each ID. These colors are from the **adafruit_led_animations** CircuitPython library. You can call on predefined colors with, for example, `color.RED` and have the NeoPixel light-up red without any additional setup.

This array will be used in the loop to have the onboard NeoPixel light-up a specific color depending on the app, letting you know from a distance if you need to check your phone or if it can wait for later.

For more information on the adafruit_led_animation library colors, check out the library's page on GitHub.

https://adafru.it/Ld-

```
APP_COLORS = {
    "com.basecamp.bc3-ios": color.YELLOW, #  Basecamp
    "com.apple.MobileSMS": color.GREEN, #  Texts
    "com.hammerandchisel.discord": color.PURPLE, #  Discord
    "com.apple.mobilecal": color.CYAN, #  Calendar
    "com.apple.mobilephone": color.GREEN, #  Phone
    "com.google.ios.youtube": color.ORANGE, #  YouTube
    "com.burbn.instagram": color.MAGENTA, #  Instagram
    "com.apple.mobilemail": color.CYAN #  Apple Email
}
```

You can also edit this array to have more or less apps depending on your preferences. The same is true for the NeoPixel colors.

Putting the "Fun" in Function

There are two functions that will be called in the loop to control the Feather's NeoPixel and the haptic motor.

`blink_pixel()` allows you to blink the NeoPixel with parameters for how many time you want it to blink (`blinks`), how fast you want it to blink (`speed`) and then the two colors that the NeoPixel will blink between (`color1` and `color2`).

If you were to discretely code multiple blinks for the NeoPixel line by line it could get really long. This lets you blink it as many times as you want with one line of code in the loop.

```
def blink_pixel(blinks, speed, color1, color2):
    for _ in range(0, blinks):
        pixel.fill(color1)
        pixel.show()
        time.sleep(speed)
        pixel.fill(color2)
        pixel.show()
        time.sleep(speed)
```

`vibe()` does something similar to `blink_pixel()` but for the haptic motor. It allows you to vibrate the motor multiple times with parameters for the number of times you want the motor to vibrate (`num_zzz`), the motor effect type (`effect`) and the length of the delay between each vibration (`delay`).

```
def vibe(num_zzz, effect, delay):
    drv.sequence[0] = adafruit_drv2605.Effect(effect)
    for _ in range(0, num_zzz):
        drv.play()
        time.sleep(delay)
        drv.stop()
```

Finally, the setup is wrapped up with BLE starting to advertise its signal for your iOS device to connect to.

```
ble.start_advertising(advertisement)
```

The Loop

Waiting for BLE

The loop begins with some visual notifications to let you know that BLE is not connected. If BLE is not connected, the onboard NeoPixel will be red and the onboard

blue LED will be off. "`Waiting for connection`" will also print out to the REPL. Once BLE is connected, "`Connected`" will print to the REPL.

```
while True:

    blue_led.value = False
    print("Waiting for connection")
    while not ble.connected:
        blue_led.value = False
        pixel.fill(color.RED)
        pixel.show()
    print("Connected")
```

BLE Setup: Apple Notification Center Service and CurrentTimeService

A few other things occur once a BLE connection is established. The onboard blue LED will turn on and the NeoPixel will turn off.

The Feather nRF52840 will also pair with your iOS device in order to have access to the Apple Notification Center Service (ANCS).

`notification_service` holds the connection to ANCS. These notifications are stored in an array. Additionally, `current_notifications` holds the current notifications that are active in ANCS.

```
while ble.connected:
        blue_led.value = True
        all_ids.clear()
        for connection in ble.connections:
            if not connection.paired:
                connection.pair()
                print("paired")
            cts = connection[CurrentTimeService]
            notification_service = connection[AppleNotificationCenterService]
        current_notifications = notification_service.active_notifications
```

Tracking Notifications

In this `for` statement, `notification` is setup to hold the notification ID's in an array for the current notifications in ANCS. The notification ID has all of the information about the notification: the time stamp, the app name, the information in the notification, etc. The notification ID's are also added to the `all_ids` array.

```
for notif_id in current_notifications:
            notification = current_notifications[notif_id]
            all_ids.append(notif_id)
```

Dismissing Notifications

Next, two `if` statements take care of handling what happens when a notification has been cleared and is no longer active.

In the first `if` statement, `current_notification` is reset to `None` and the NeoPixel, which functions as a visual cue that you have a notification, is turned off.

From the **adafruit_led_animations** library, `color.BLACK` means that the NeoPixel is off with RGB values of `(0, 0, 0)`.

```
if current_notification and current_notification.removed:
            # Stop showing the latest and show that there are no new notifications.
            current_notification = None
            pixel.fill(color.BLACK)
            pixel.show()
```

The second `if` statement, checks that `current_notification` is inactive and that `all_ids` is empty. If that is correct, then `cleared` is updated to `True` to demonstrate that all of the notifications in ANCS are currently cleared.

```
if not current_notification and not all_ids and not cleared:
            cleared = True
            pixel.fill(color.BLACK)
            pixel.show()
```

Indexing Notifications

The next portion revolves around what happens if a notification is active in ANCS.

First, `cleared` is updated to `False` and then the notification ID's are indexed. This allows for new notifications to trigger the Feather's NeoPixel and the haptic motor even if older notifications have not been cleared. The newest notification is then stored in `notif_id`.

```
elif all_ids:
  cleared = False
  if current_notification and current_notification.id in all_ids:
    index = all_ids.index(current_notification.id)
  else:
    index = len(all_ids) - 1
    notif_id = all_ids[index]
```

Notification Alerts with Haptic Motors and NeoPixels

The following `if` statement is where the action is for a notification triggering the Feather nRF52840.

If the state of `current_notification.id` does not match the notification index that was just stored in `notif_id`, then the code knows that a new notification is present.

The `app_id` is checked against the array of application ID's in the `APP_COLORS` array. If the current notification is not from one of those apps, then the NeoPixel will be white. However, if the notification is in that array, then the NeoPixel will light-up as the color defined in the array. The color for the NeoPixel is stored as `notif_color`.

`current_notification` is updated to hold this new notification and `category` holds all of the metadata for the notification as a string. Storing it in this way allows it to be printed to the REPL and also accessed in the code.

```
if not current_notification or current_notification.id != notif_id:
                current_notification = current_notifications[notif_id]
                #  if the notification is from an app that is not
                #  defined in APP_COLORS then the NeoPixel will be white
                if current_notification.app_id not in APP_COLORS:
                    notif_color = color.WHITE
                #  if the notification is from an app defined in
                #  APP_COLORS then the assigned color will show
                else:
                    notif_color = APP_COLORS[current_notification.app_id]
                #  parses notification info into a string
                category = str(notification).split(" ", 1)[0]
```

The haptic motor vibrates to alert you to the new notification. This is done with the `vibe()` function.

Following the haptic motor, the notification's metadata is printed out to the REPL. The information included is the ID number, category of notification, app name, title, subtitle and the message in the notification. The notifications are separated by 36 dashes.

```
vibe(2, vibration, 0.5)
print('-'*36)
print("Msg #%d - Category %s" % (notification.id, category))
print("From app:", notification.app_id)
if notification.title:
      print("Title:", notification.title)
if notification.subtitle:
      print("Subtitle:", notification.subtitle)
if notification.message:
      print("Message:", notification.message)
```

After the information has been printed to the REPL, the onboard NeoPixel will blink the previously defined `notif_color` with the `blink_pixel()` function and then stay on until the notifications are cleared, just in case you aren't around to notice the haptic motor or blinking.

```
#  NeoPixel blinks and then stays on until cleared
blink_pixel(2, 0.5, notif_color, color.BLACK)
pixel.fill(notif_color)
pixel.show()
```

That wraps up the ANCS notification portion of the loop. Beyond notifications though, you may want to keep track of time while you're deep in emails, taking a walk or Fusion360. Luckily, there's BLE functionality to help with that too.

Mindfulness

Using the `CurrentTimeService` BLE library, you can access your connected device's clock to sync up with the Feather NRF52840.

In this instance, you'll use `CurrentTimeService` to check when a new hour begins (9:00, 10:00, 11:00, etc). If it's on the hour, the haptic motor will vibrate and the NeoPixel will blink to alert you. This can help remind you to get up and stretch or to be reminded of the time without having to look at a clock or your devices.

The information from `CurrentTimeService` is stored in `cts` after being setup earlier in the loop. The information is stored as an array, with all of the time data separated in their own entries. The minutes for the hour is indexed as `4` in the array. You can check a predetermined value, in this case `hour`, to see if it matches the current time being held by `cts`.

In the case of the code, if `cts.current_time[4]` matches with `hour`, which is set to `0`, then the current time is printed to the REPL along with the string "`mindful time`". The haptic motor vibrates with the `vibe()` function and the onboard NeoPixel blinks using the `blink_pixel ()` function, this time in blue.

```
if cts.current_time[4] == hour and not mindful:
    print(cts.current_time[4])
    print("mindful time")
    vibe(5, vibration, 1)
    blink_pixel(5, 1, color.BLUE, color.BLACK)
```

The `mindful` state is set to `True` and the onboard NeoPixel will remain blue for the duration of the minute.

```
mindful = True
pixel.fill(color.BLUE)
pixel.show()
print("hour = ", hour)
```

Once one minute after the hour occurs, the `mindful` state is reset to `False` and the onboard NeoPixel is turned off. The string "`mindful time over`" is also printed to the REPL. This process will occur every hour alongside the ANCS notifications.

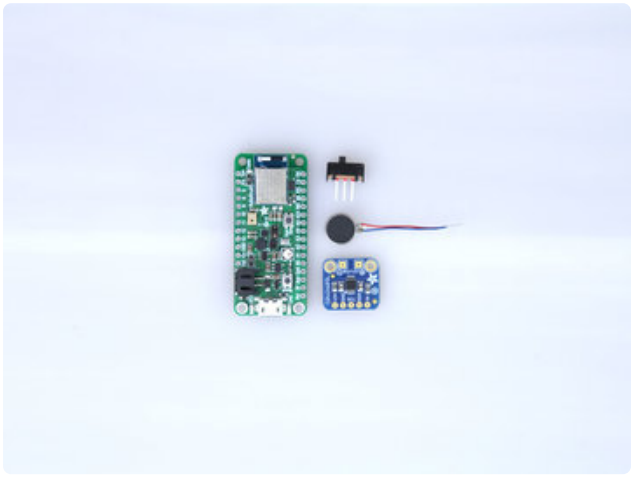
```
if cts.current_time[4] == (hour + 1) and mindful:
            #  NeoPixel turns off
            mindful = False
            pixel.fill(color.BLACK)
            pixel.show()
            print("mindful time over")
```

Dropped Signals

The code closes with a safety net for lost BLE connections. If the Feather becomes disconnected from your iOS device, "`Disconnected`" will be printed to the REPL and the onboard blue LED will turn off. The nRF52840 will begin advertising the BLE connection again and `notifcation_service` will be reset to `None`. The loop will go back to the beginning to reconnect.

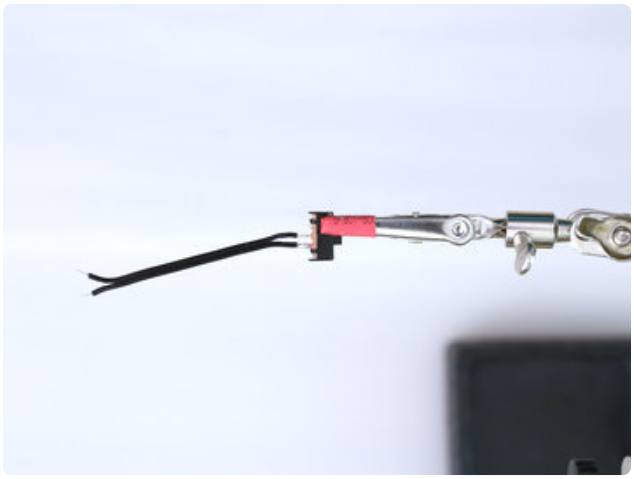
```
print("Disconnected")
blue_led.value = False
print()
ble.start_advertising(advertisement)
notification_service = None
```

Wiring



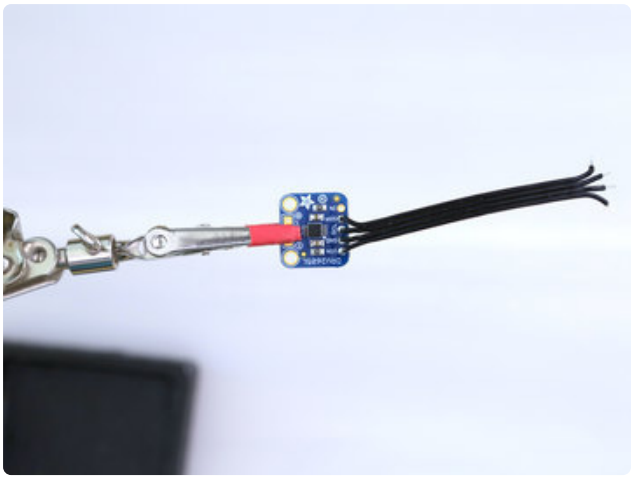
Wiring Parts

Get the components ready for wiring. The 10-wire silicone cover ribbon wire is recommended for soldered connections.



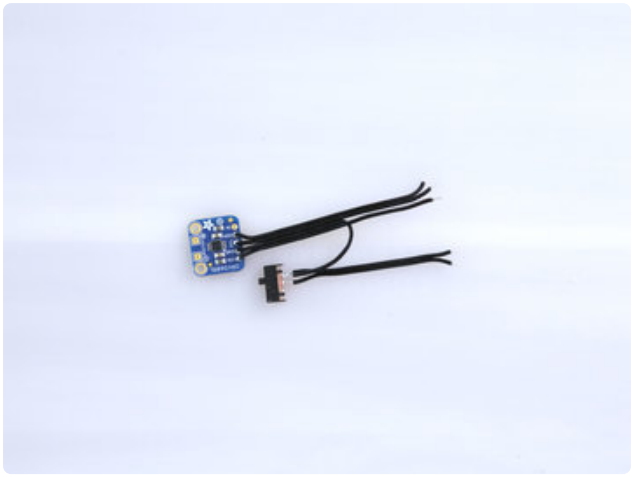
Switch Wiring

Use a piece of 2-wire ribbon wire that is 56mm in length. Solder to the middle and either far left or right pin.



Wiring DRV2605L

Use a piece of 4-wire ribbon cable 70mm in length. Solder wires to the SDA, SCL, VIN and GND pins.

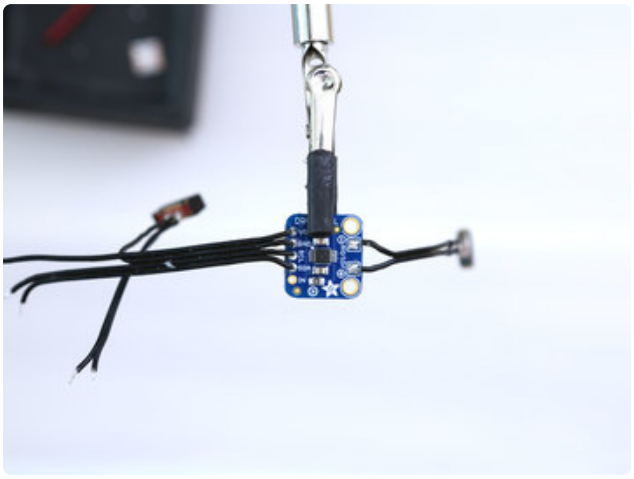


Wiring Common Ground

Solder the ground wire from the DRV2605L to the middle pin on the switch. These components share common ground.

Motor Wiring

The wires from the vibration motor are soldered to a piece of 2-wire ribbon cable 40mm in length.



Wire Motor to DRV2605L

The two wires from the vibration motor are soldered to the voltage and ground motor pins on the DRV2605L breakout.

Wiring to Feather

The wires from the switch and DRV2605L are soldered to the pins on the Feather.

Circuit Wired

Solder the SDA, SCL and VIN wires from the DRV2605L to the SDA, SLC and 3V pins Feather. Solder the wires from the switch to the GND and EN pins on the Feather.

Assembly

Install Circuit to Case

Insert the Feather into the case at an angle with the edge facing the tab. Press the Feather down into the standoffs to secure in place. Insert the switch into the built in holder at an angle. Press down to secure switch. Peel protective backing from vibration motor and stick into built in holder. Press DRV2605L breakout into built in standoffs.

Install Battery

Plug in JST cable form battery into JST port on the Feather. Place battery down next to Feather.

Test Circuit

Use slide switch to power the Feather on and off. The blue LED indicator is used for BLE connections. The NeoPixel will glow red until it has been paired with an iOS device.

Install Cover & Bands

Place the cover on top of the case with the button tab oriented the reset button. Get the strap and band ready to install onto the case.

Installing Bands

Firmly push the tabs from the band through the bottom of the case on either loop. Repeat for strap.

Installed Straps

The strap and band can be installed on either side of the case. Pull out tab from loops to remove bands.

Wear It!

Place case over the wrist and hold the strap over the band. Place fingers underneath band and press nub into one of the slots on strap. Use fingers to press nub through the slot. Tuck excess strap into slotted hole in the band.

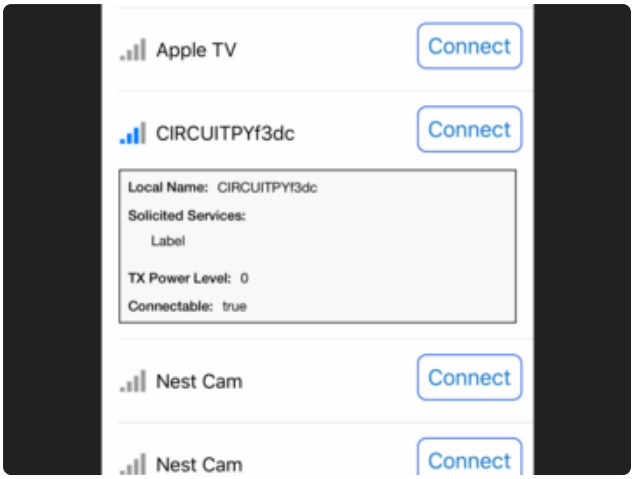
Adjust Band

Use different slot in strap to change fitting on wrist. Change the orientation of the case by swapping the band and strap on the loops.

Usage

Connect iOS devices to CircuitPython Bluetooth Devices

Use the iOS settings app and select Bluetooth from the list. Search for **CIRCUITPY** under My Devices. Tap to connect and proceed to pair and allow the device to display notifications.

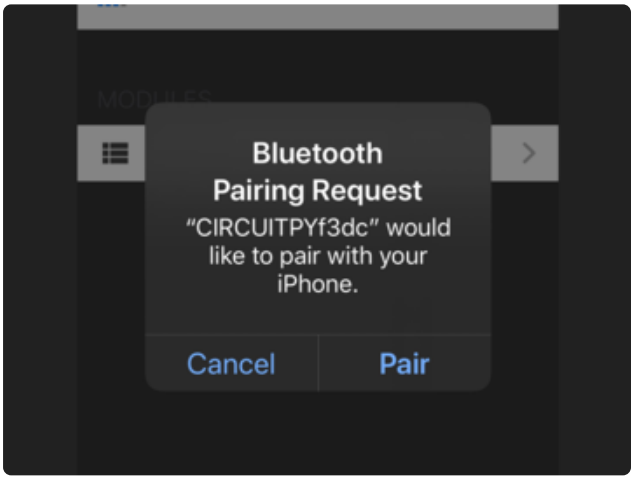


Connect Device

You can optionally use the Adafruit Bluefruit Connect LE app to connect and pair with the **CIRCUITPY** device. Search and tap connect on the **CIRCUITPYxxxx** device.

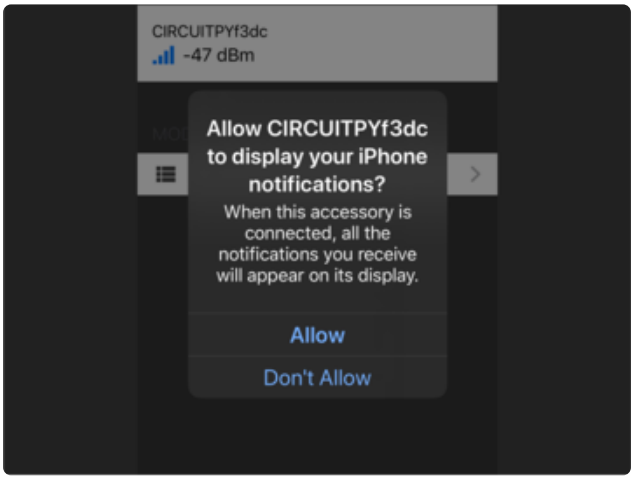
Download Adafruit Bluefruit LE Connect App for iOS

https://adafru.it/ddu



Bluetooth Pairing

The iOS device will prompt a dialog box with option to pair or cancel. Tap on the Pair button.



Allow Notifications

The next prompt will ask if you'd like to allow the **CIRCUITPY** device to display notifications from the iOS device.

Connection Status!

The on-board NeoPixel will light up and stay RED until an iOS device is paired. Incoming notifications will trigger a buzz and blink the NeoPixel twice with a purple color. The NeoPixel will remain lit until the notification has been cleared on the iOS device. The **CIRCUITPY** device will automatically connect to the iOS device automatically.