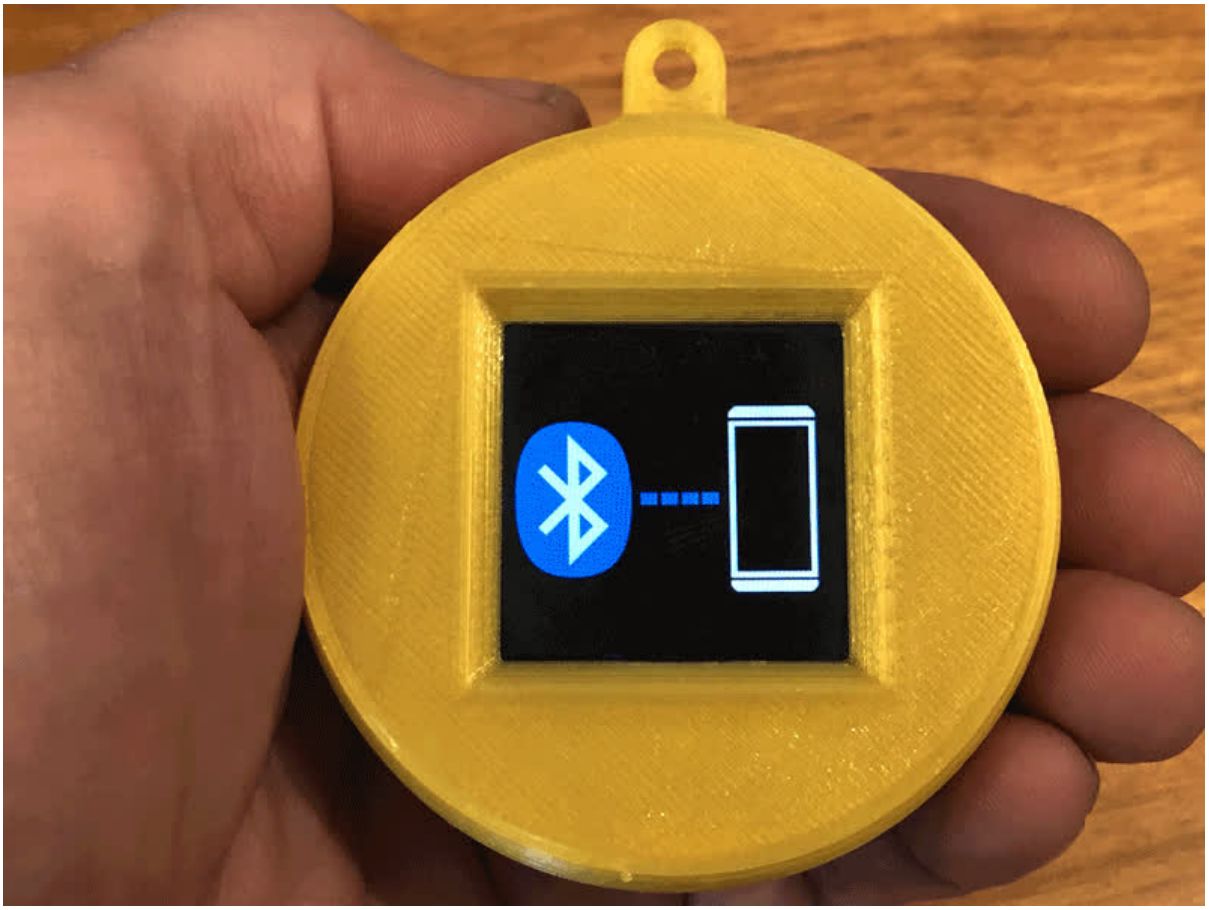
# Bluefruit TFT Gizmo ANCS Notifier for iOS

Created by John Park

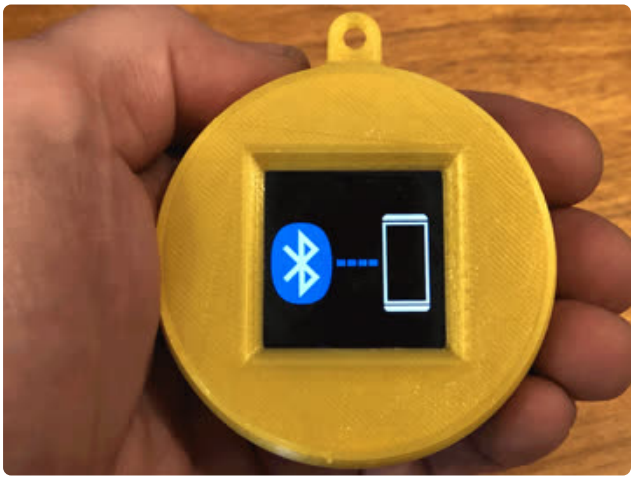


https://learn.adafruit.com/ancs-gizmo

Last updated on 2024-04-05 01:09:06 PM EDT

# Table of Contents

# Overview



Circuit Playground Bluefruit displays your notification icons so you know when there's fresh activity!

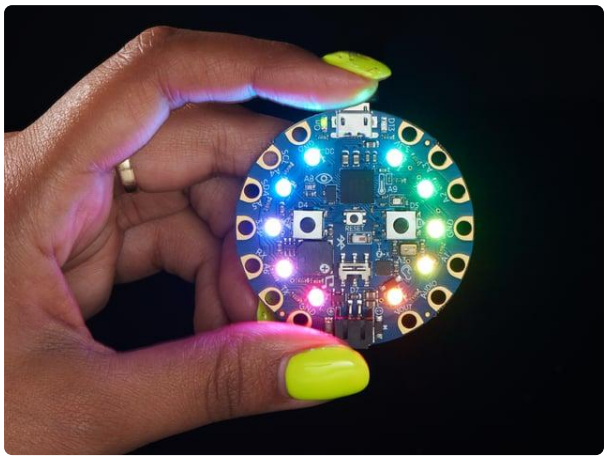
Using the Apple Notification Center Service (**ANCS**), this project displays app icons on your TFT Gizmo display whenever there's an important alert from one of your apps, such as Slack, Basecamp, Discord, SMS messages, calendar events and more!

You can even use the A and B buttons on the CPB to scroll between current notification icons.

Once you dismiss a notification on your phone, the associated icon will disappear from the Gizmo screen.

This tutorial only works with Apple iOS devices. Android does not support ANCS and there is no ETA on when a guide might be developed in the future for Android notification

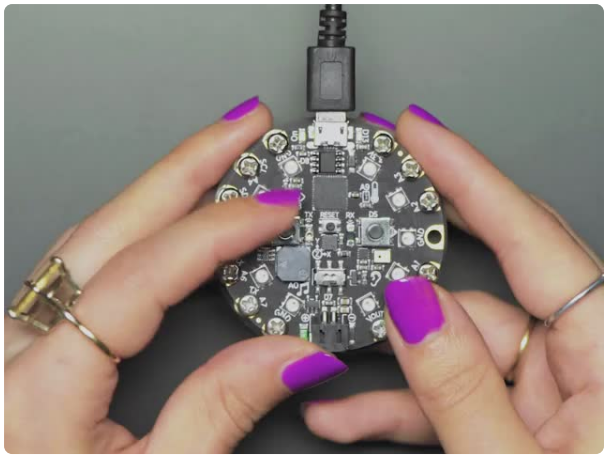
## Parts



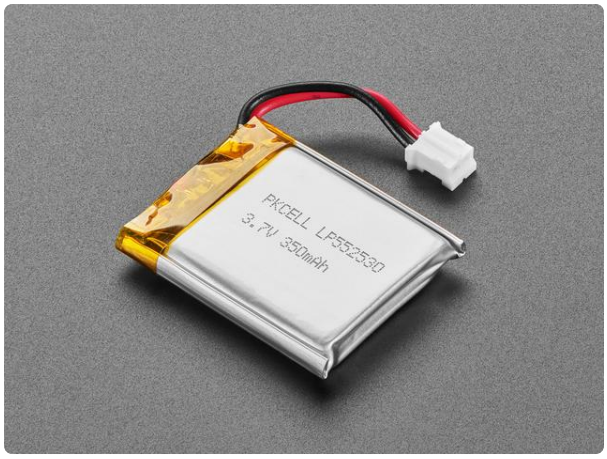
Circuit Playground Bluefruit - Bluetooth Low Energy

Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...

https://www.adafruit.com/product/4333



[Circuit Playground TFT Gizmo - Bolt-on Display + Audio Amplifier]
Extend and expand your Circuit Playground projects with a bolt on TFT Gizmo that lets you add a lovely color display in a sturdy and reliable fashion. This PCB looks just like a round...
https://www.adafruit.com/product/4367




[Lithium Ion Polymer Battery with Short Cable - 3.7V 350mAh]
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/4237

[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

---

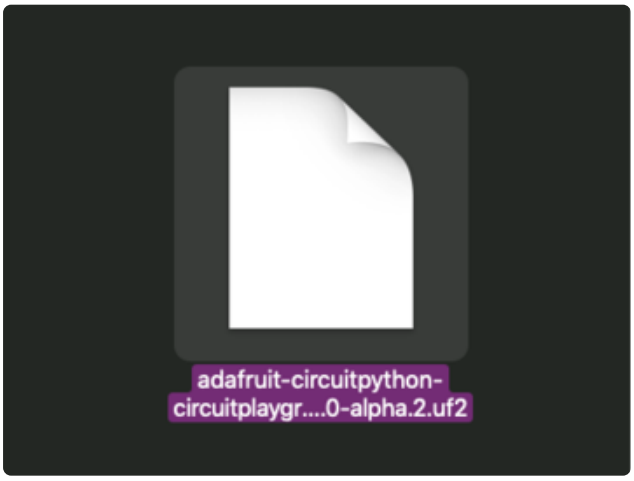
# CircuitPython on Circuit Playground Bluefruit

## Install or Update CircuitPython

Follow this quick step-by-step to install or update CircuitPython on your Circuit Playground Bluefruit.

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

https://adafru.it/FNK

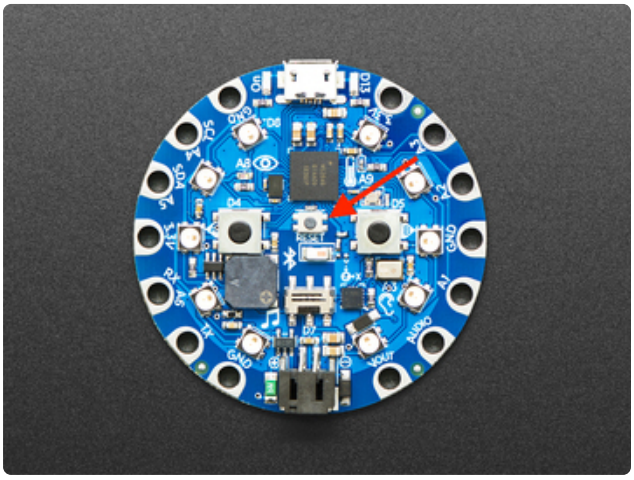



**Click the link above and download the latest UF2 file**

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

Plug your Circuit Playground Bluefruit into your computer using a known-good data-capable USB cable.

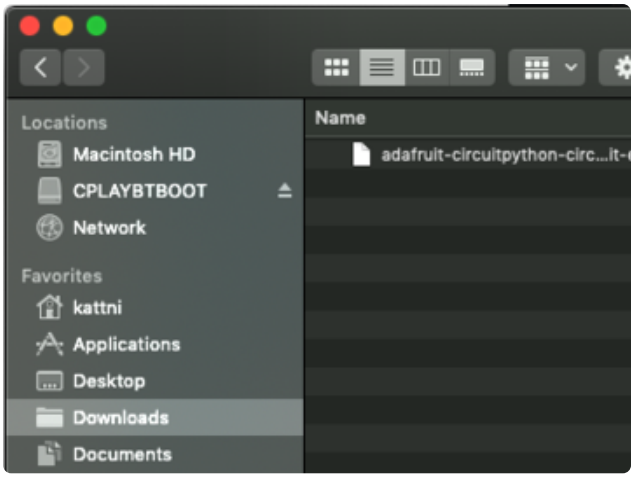
**A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.**



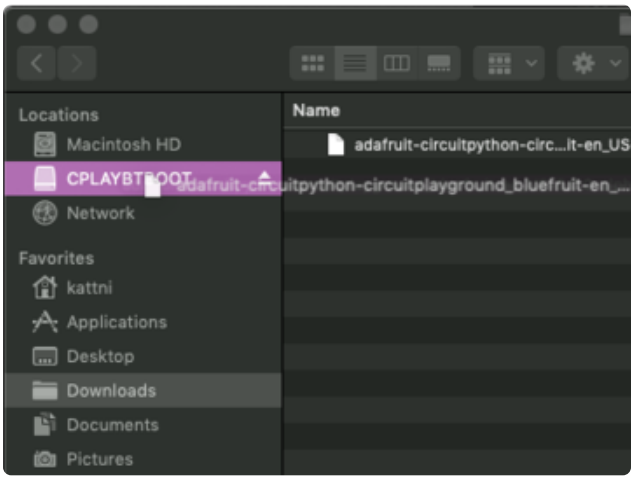
Double-click the small **Reset** button in the middle of the CPB (indicated by the red arrow in the image). The ten NeoPixel LEDs will all turn red, and then will all turn green. If they turn all red and stay red, check the USB cable, try another USB port, etc. The little red LED next to the USB connector will pulse red - this is ok!

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

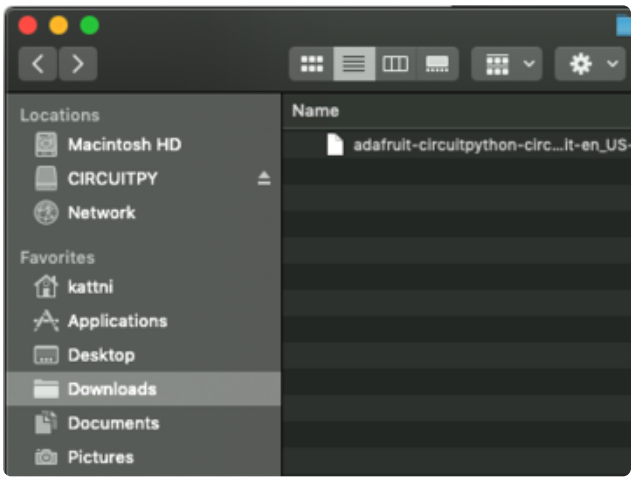
(If double-clicking doesn't do it, try a single-click!)




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




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




The LEDs will turn red. Then, the **CPLAYBTBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

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

# Circuit Playground Bluefruit CircuitPython Libraries

The Circuit Playground Bluefruit is packed full of features like Bluetooth and NeoPixel LEDs. Now that you have CircuitPython installed on your Circuit Playground Bluefruit, 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.

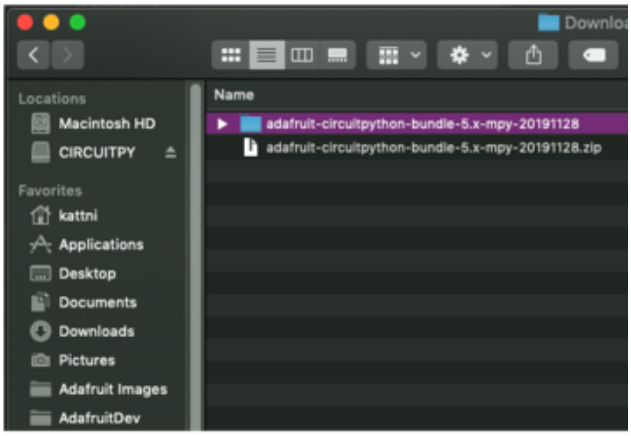
# Installing CircuitPython Libraries on Circuit Playground Bluefruit

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

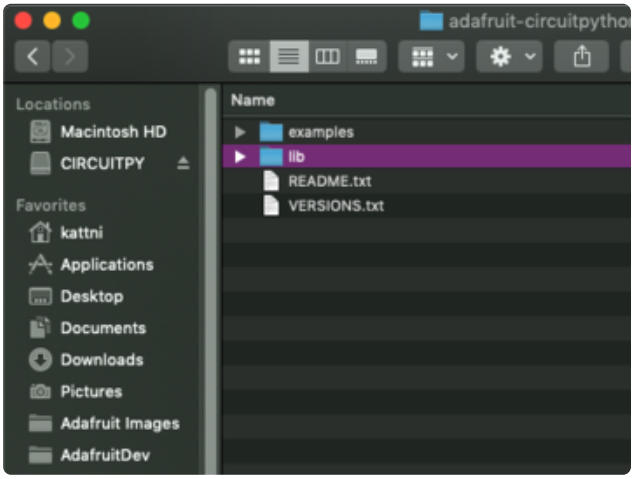
Then, download the CircuitPython library bundle that matches your version of CircuitPython from CircuitPython.org.

**Download the latest library bundle from circuitpython.org**

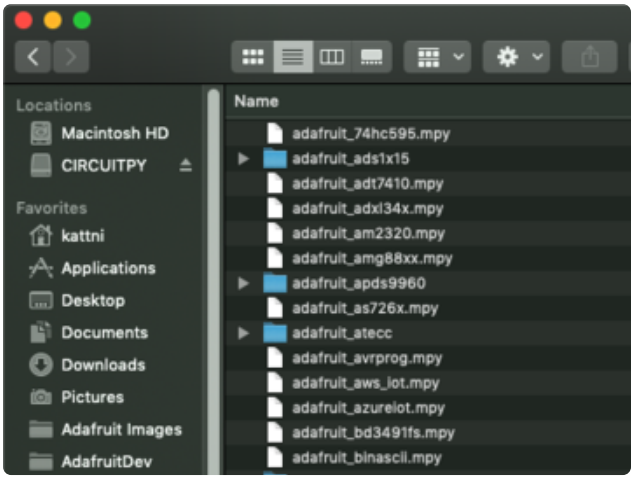
https://adafru.it/ENC



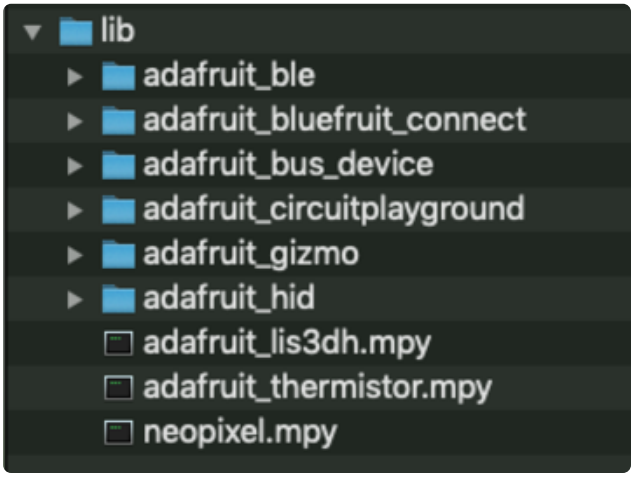

The bundle download as a .zip file. Extract the file. Open the resulting folder.




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_ble**
**adafruit_bluefruit_connect**
**adafruit_bus_device**
**adafruit_circuitplayground**
**adafruit_gizmo**
**adafruit_hid**
**adafruit_lis3dh.mpy**
**adafruit_thermistor.mpy**
**neopixel.mpy**
Your lib folder should look like the image on the left.

Now you're all set to use CircuitPython with the features of the Circuit Playground Bluefruit!

---

# Code with CircuitPython

The Apple Notification Center Service (ANCS) allows iOS devices to act as a provider of notification alerts to Bluetooth Low Energy (BLE) accessories, such as the Apple Watch, or in our case, the Circuit Playground Bluefruit!

We've created a simple program that allows the CPB to pair with your iOS device over Bluetooth and then it will receive ANCS notifications to display on the TFT Gizmo.


In general, we recommend installing the libraries mentioned on the previous page for your Circuit Playground Bluefruit projects, however you can get away with a subset of them for this one. You can see them listed in the image here.

You'll also need to add the **adafruit_ble_apple_notification_center.mpy** file from the library bundle as seen in the image here.

Once you save the **code.py**, graphics, and .wav file to your **CIRCUITPY** drive as directed below, this image is what your drive should look like.

Click the "Download: **Project Zip**" link in the code block below to get all the files from the project's GitHub repo.

Then, uncompress the zip file and open the **code.py** file in Mu, then save it to your CPB's **CIRCUITPY** drive as **code.py.**

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
This demo shows the latest icons from a connected Apple device on a TFT Gizmo
screen.

The A and B buttons on the CircuitPlayground Bluefruit can be used to scroll
through all active
notifications. The screen's backlight will turn off after a certain number of
seconds to save power.
New notifications or pressing the buttons should turn it back on.
"""

import time
import board
import digitalio
import displayio
import adafruit_ble
from adafruit_ble.advertising.standard import SolicitServicesAdvertisement
from adafruit_ble_apple_notification_center import AppleNotificationCenterService
from adafruit_gizmo import tft_gizmo
from audiocore import WaveFile
from audiopwmio import PWMAudioOut as AudioOut

# Enable the speaker
speaker_enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
speaker_enable.direction = digitalio.Direction.OUTPUT
speaker_enable.value = True
```

```
audio = AudioOut(board.SPEAKER)

# This is a whitelist of apps to show notifications from.
APP_ICONS = {
    "com.tinyspeck.chatlyio": "/ancs_slack.bmp",
    "com.basecamp.bc3-ios": "/ancs_basecamp.bmp",
    "com.apple.MobileSMS": "/ancs_sms.bmp",
    "com.hammerandchisel.discord": "/ancs_discord.bmp",
    "com.apple.mobilecal": "/ancs_ical.bmp",
    "com.apple.mobilephone": "/ancs_phone.bmp"
}

BLOCKLIST = []
DELAY_AFTER_PRESS = 15
DEBOUNCE = 0.1
DIM_TIMEOUT = 20   # Amount of timeout to turn off backlight
DIM_LEVEL = 0.05

a = digitalio.DigitalInOut(board.BUTTON_A)
a.switch_to_input(pull=digitalio.Pull.DOWN)
b = digitalio.DigitalInOut(board.BUTTON_B)
b.switch_to_input(pull=digitalio.Pull.DOWN)

file = open("/triode_rise.wav", "rb")
wave = WaveFile(file)

def play_sound():
    audio.play(wave)
    time.sleep(1)

def find_connection():
    for connection in radio.connections:
        if AppleNotificationCenterService not in connection:
            continue
        if not connection.paired:
            connection.pair()
        return connection, connection[AppleNotificationCenterService]
    return None, None

class Dimmer:
    def __init__(self):
        self._update_time = time.monotonic()
        self._level = DIM_LEVEL
        self._timeout = DIM_TIMEOUT

    def update(self):
        self._update_time = time.monotonic()

    def check_timeout(self):
        if a.value or b.value:
            self._update_time = time.monotonic()
        if time.monotonic() - self._update_time > self._timeout:
            if display.brightness > self._level:
                display.brightness = self._level
        else:
            if display.brightness == self._level:
                display.brightness = 1.0

dimmer = Dimmer()

# Start advertising before messing with the display so that we can connect
immediately.
radio = adafruit_ble.BLERadio()
advertisement = SolicitServicesAdvertisement()
advertisement.complete_name = "CIRCUITPY"
advertisement.solicited_services.append(AppleNotificationCenterService)

def wrap_in_tilegrid(filename:str):
    # CircuitPython 6 & 7 compatible
```

```
    odb = displayio.OnDiskBitmap(open(filename, "rb"))
    return displayio.TileGrid(
        odb, pixel_shader=getattr(odb, 'pixel_shader', displayio.ColorConverter())
    )

    # # CircuitPython 7+ compatible
    # odb = displayio.OnDiskBitmap(filename)
    # return displayio.TileGrid(odb, pixel_shader=odb.pixel_shader)

display = tft_gizmo.TFT_Gizmo()
group = displayio.Group()
group.append(wrap_in_tilegrid("/ancs_connect.bmp"))
display.root_group = group

current_notification = None
current_notifications = {}
all_ids = []
last_press = time.monotonic()
active_connection, notification_service = find_connection()
cleared = False

while True:
    if not active_connection:
        radio.start_advertising(advertisement)

    while not active_connection:
        active_connection, notification_service = find_connection()
        dimmer.check_timeout()

    # Connected
    dimmer.update()
    play_sound()

    no_notifications = "/ancs_none.bmp"
    group.append(wrap_in_tilegrid(no_notifications))
    while active_connection.connected:
        all_ids.clear()
        current_notifications = notification_service.active_notifications
        for notif_id in current_notifications:
            notification = current_notifications[notif_id]
            if notification.app_id not in APP_ICONS or notification.app_id in
BLOCKLIST:
                continue
            all_ids.append(notif_id)

        # pylint: disable=protected-access
        all_ids.sort(key=lambda x: current_notifications[x]._raw_date)
        # pylint: enable=protected-access

        if current_notification and current_notification.removed:
            # Stop showing the latest and show that there are no new notifications.
            current_notification = None

        if not current_notification and not all_ids and not cleared:
            cleared = True
            dimmer.update()
            group[1] = wrap_in_tilegrid(no_notifications)
        elif all_ids:
            cleared = False
            now = time.monotonic()
            if current_notification and current_notification.id in all_ids and \
                now - last_press < DELAY_AFTER_PRESS:
                index = all_ids.index(current_notification.id)
            else:
                index = len(all_ids) - 1
            if now - last_press >= DEBOUNCE:
                if b.value and index > 0:
                    last_press = now
                    index += -1
```

```
                if a.value and index < len(all_ids) - 1:
                    last_press = now
                    index += 1
            notif_id = all_ids[index]
            if not current_notification or current_notification.id != notif_id:
                dimmer.update()
                current_notification = current_notifications[notif_id]
                # pylint: disable=protected-access
                print(current_notification._raw_date, current_notification)
                # pylint: enable=protected-access
                group[1] = wrap_in_tilegrid(APP_ICONS[current_notification.app_id])

        dimmer.check_timeout()

    # Bluetooth Disconnected
    group.pop()
    dimmer.update()
    active_connection = None
    notification_service = None
```

## Code Explaination

Apple devices centralize the management of notifications into the "Notification Center", which is accessed on the lock screen or after swiping down from the top of the screen. Since this info is centralized, Apple provides access to the current notifications through a Bluetooth Low Energy Service on the device. Bluetooth Services are a collection of data referred to as Characteristics. In CircuitPython libraries we provide definitions for common services like the Apple Notification Center Service (or ANCS for short) so that they are easier to use.

The ANCS library is designed for two main uses. First, it allows one to list all currently active notifications. This is done by reading the `active_notifications` attribute of the service. Second, it allows you to wait for new notifications to come in and react to them. This is done by looping over `wait_for_new_notifications()`.

Both of these uses provide a Notification object for each active notification. Reading the attributes of these objects will load the data from the peer device as needed. This can make printing slow but it conserves data which in turn saves battery. So, only read the attributes you need to know. Printing the object is useful for debugging but it loads many attributes and therefore, takes time.

The most useful attribute for this project is the `app_id`. The `app_id` identifies which app generated the notification. It is not the app name, but may be derived from it or the name of the folks who created the app. In the example code.py we've already added a few `app_id`s. For example, Twitter is `com.atebits.Tweetie2`, Slack is `com.tinyspeck.chatlyio` and GMail is `com.google.Gmail`. To add support for an app you use, connect to the serial terminal while running the example the Apple Notification Center library's simple test, which prints out all of the current notification's app_ids and title.

# Notification Icons

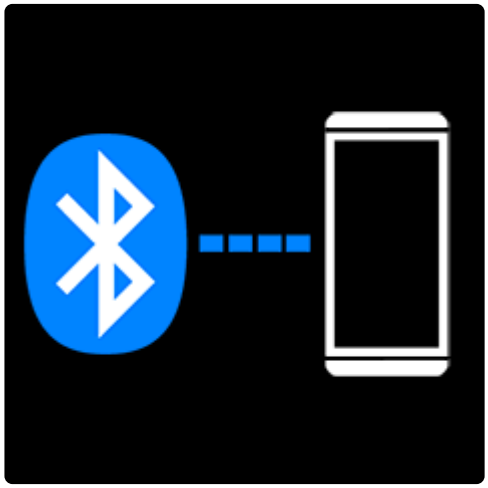
## Graphics

In keeping with the nature of a simple alert system, we decided to use graphic icons rather than text for these notifications. If you see a logo pop up, you can then go look at the actual notification on your iOS device. This is a great place to read the details of your message or alert, and this action will also clear the notification from the CPB!

By using the "Download: **Project Zip**" in the code window on the previous pages, you also got all the graphics files from the project GitHub repo. Copy these .bmp files to the root directory of your CIRCUITPY drive.

### Connect to Bluetooth

This icon lets you know you need to pair your iOS device (from the Bluetooth settings page) with the CPB.



### No Notifications

This bell icon lets you know that the CPB is connected over to an iOS device and awaiting new notifications.

## Notification Icons

Here are some icons we made for popular notifications that are also included in the code.

## Discord

Slack

SMS

Phone

## iCal


## Basecamp

Here's the old style Basecamp logo, in case you want to go that way...

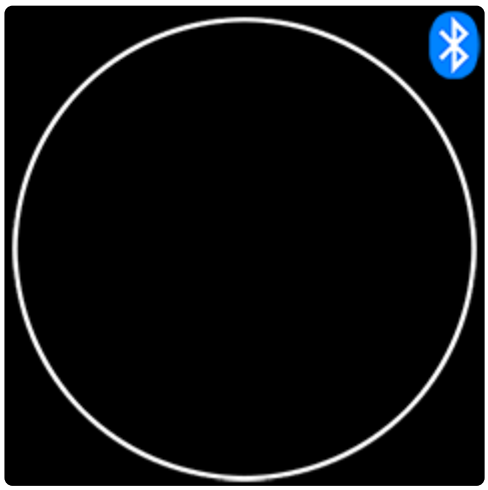
CircuitPython is so fast and easy, pleasing clients with logo changes in minutes is possible (https://adafru.it/HBa), so here's the new logo, too! You can just change the icon name in code and use the one you want!

## Customization

If you'd like to make new icons for other notifications, you'll need to create 16-bit .bmp files at 240x240 pixels.

You can use image software such as Photoshop (https://adafru.it/HD2), Affinity Photo (https://adafru.it/HD3), GIMP (https://adafru.it/GCa), or others to mask and layer an application icon on top of the black background and inside the white circle. Below you'll find a blank version of our design to use as a starting point.

Or, you can use any image you like, so long as you save or convert it first to the 240x240 .bmp file format first. You can use an online image converter (https://adafru.it/CWD) for this.



## Audio

We will trigger the playback of a .wav file when the Bluetooth connection is made. Get the file **triode_rise.wav** from the Project Zip noted above and move it to the root level of your **CIRCUITPY** drive.

---

# ANCS Simple Demo

## REPL Print Only Version

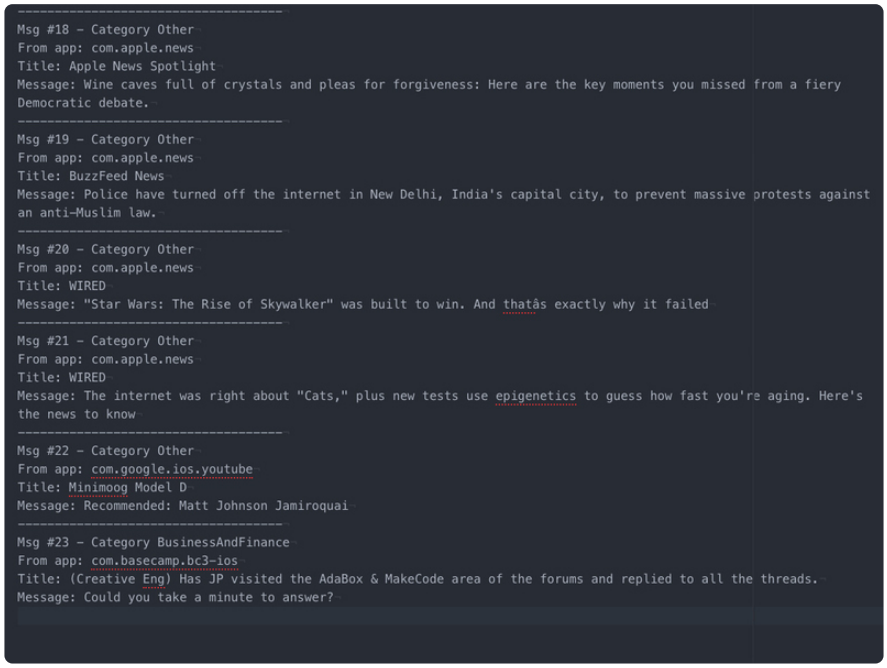
If you want to add more icon pairings, you'll need to know what sort of data ANCS is sending from the phone so you can match your icon to the app type.

This version of the code will give you just the emitted data! It's very informative to see what info comes along with your notifications. Just download the `print_code.py` file and paste it into your Mu window, then save it to your Circuit Playground Bluefruit as `code.py`.

Pair your iOS device with the CIRCUITPY BLE device and you'll see lots of great notification info!

```
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""This demo shows the latest notification from a connected Apple device on the
REPL and Gizmo"""
import time
import adafruit_ble
from adafruit_ble.advertising.standard import SolicitServicesAdvertisement
from adafruit_ble_apple_notification_center import AppleNotificationCenterService
from adafruit_gizmo import tft_gizmo

# This is a whitelist of apps to show notifications from.
#APPS = ["com.tinyspeck.chatlyio", "com.atebits.Tweetie2"]
APPS = []

DELAY_AFTER_PRESS = 15
DEBOUNCE = 0.1

def find_connection():
    for connection in radio.connections:
        if AppleNotificationCenterService not in connection:
            continue
        if not connection.paired:
            connection.pair()
        return connection, connection[AppleNotificationCenterService]
    return None, None

# Start advertising before messing with the display so that we can connect
immediately.
radio = adafruit_ble.BLERadio()
```

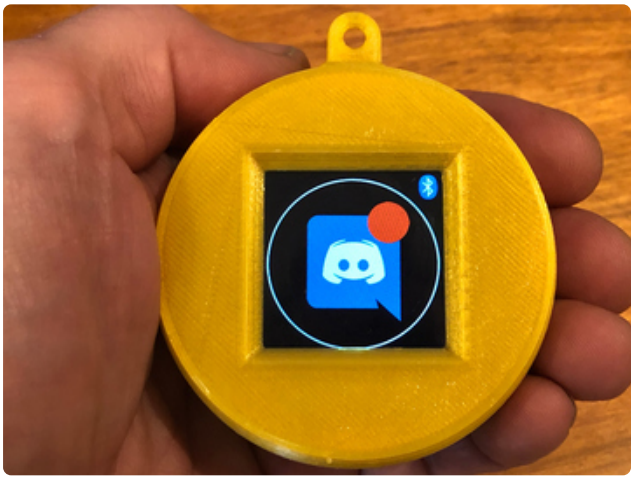
```
advertisement = SolicitServicesAdvertisement()
advertisement.complete_name = "CIRCUITPY"
advertisement.solicited_services.append(AppleNotificationCenterService)

display = tft_gizmo.TFT_Gizmo()

current_notification = None
new_ids = []
displayed_ids = []
active_connection, notification_service = find_connection()
while True:
    if not active_connection:
        radio.start_advertising(advertisement)

    while not active_connection:
        print("waiting for connection...")
        active_connection, notification_service = find_connection()
        time.sleep(0.1)

    while active_connection.connected:
        current_notifications = notification_service.active_notifications
        for notification_id in current_notifications:
            if notification_id in displayed_ids:
                continue # already seen!
            notification = current_notifications[notification_id]
            print('-'*36)
            category = str(notification).split(" ", 1)[0]
            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)
            displayed_ids.append(id)
    active_connection = None
    notification_service = None
```



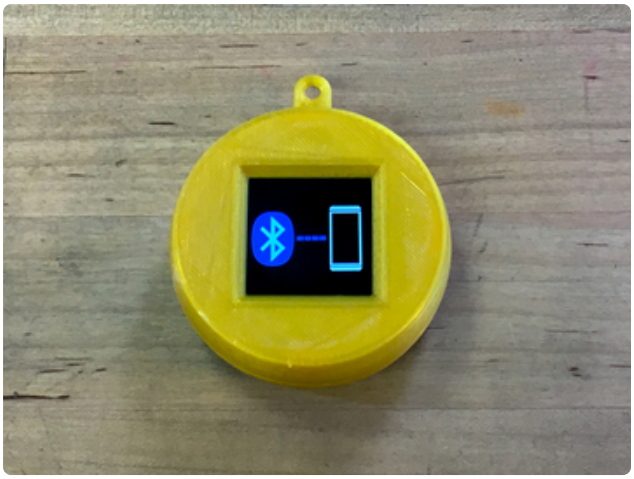
# Assembly

## Assemble the Notifier

Now, you can use your iOS Notification Gizmo! I assembled the board with a battery between the TFT Gizmo and the CPB as shown here (https://adafru.it/HBc).



I decided to print one of the the 3D cases designed by the Ruiz Bros. in this guide (https://adafru.it/HBd) so I can use it as a sort of pocket watch style alert device!





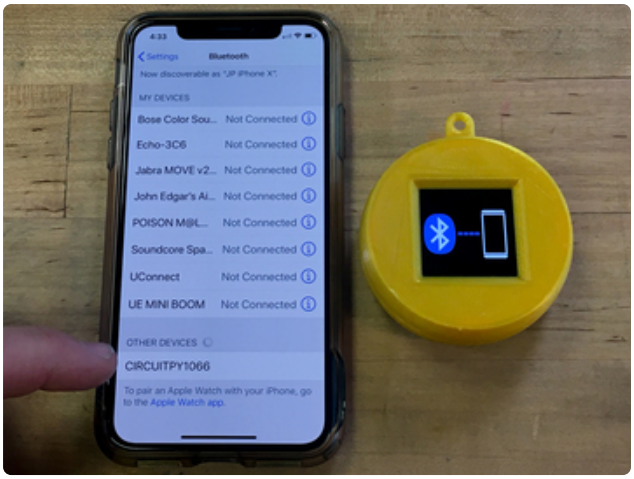
## Use the Notifier

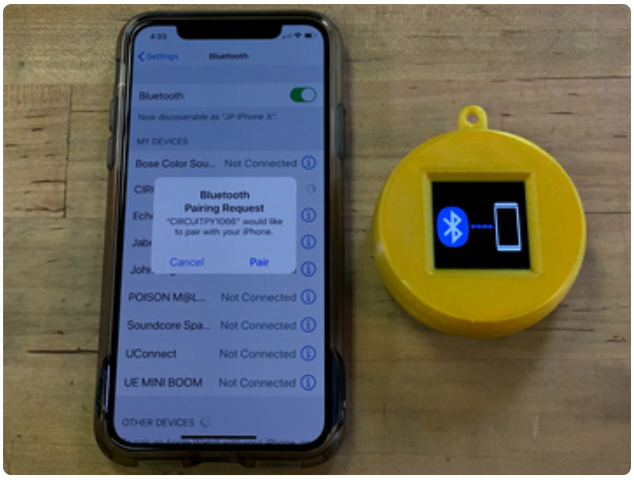
To use the notifier, the first step is to connect to Bluefruit.

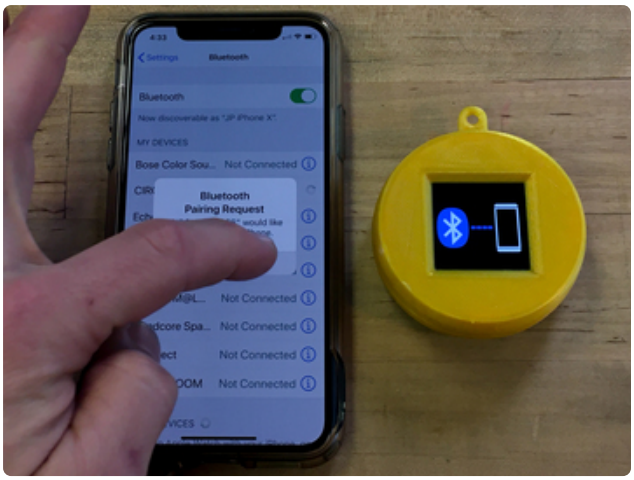
First, make sure the Notifier is powered on. You'll see the connect to Bluetooth device screen.

On your iOS device, make sure Bluetooth is turned on and go to Settings > Bluetooth.

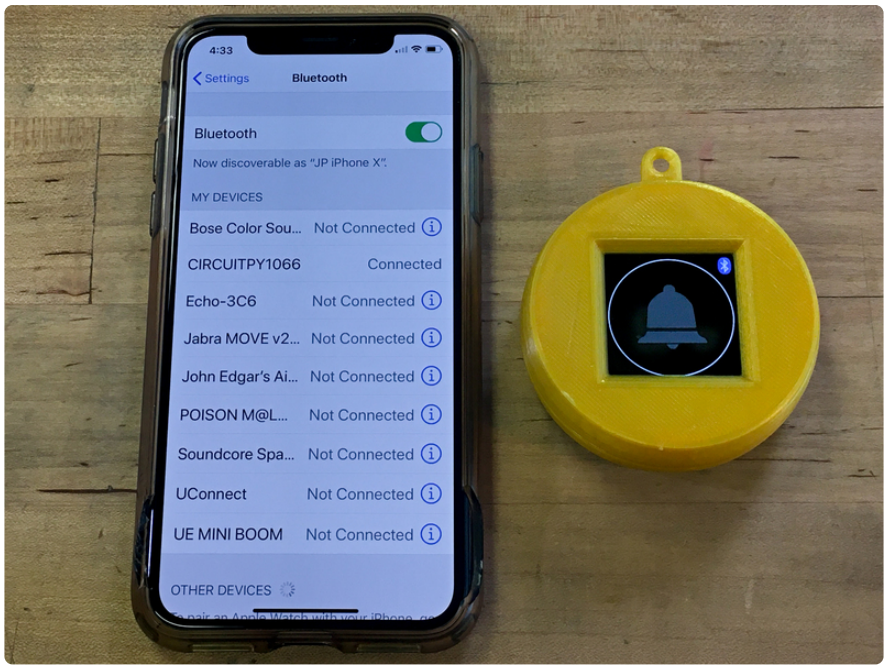
Pick the Circuit Playground Bluefruit device that shows up, and then press the "**Pair**" button.




Once connected, you'll see the default notifications bell screen. This indicates that there is a Bluetooth connection to your iOS device and no new notifications are in your notification center.


4:33
Settings
Bluetooth
Bluetooth
Now discoverable as "JP iPhone X".
MY DEVICES
Bose Color Sou... Not Connected
CIRCUITPY1066 Connected
Echo-3C6 Not Connected
Jabra MOVE v2... Not Connected
John Edgar's Ai... Not Connected
POISON M@L... Not Connected
Soundcore Spa... Not Connected
UConnect Not Connected
UE MINI BOOM Not Connected
OTHER DEVICES




## Notifications

Hey look! An SMS notification came in! Now I now to look at my phone and read the message.Oh, it's an Arrested Development GIF from my friend Tod featuring Tobias Fünke, awesome!

Now, I can clear the message from the iOS Notification screen and this will also clear it from the Gizmo screen.

## Multiple Notifications

By pressing the A and B buttons on the CPB we can see there are notifications from a couple of different applications, in this case Basecamp and Discord.