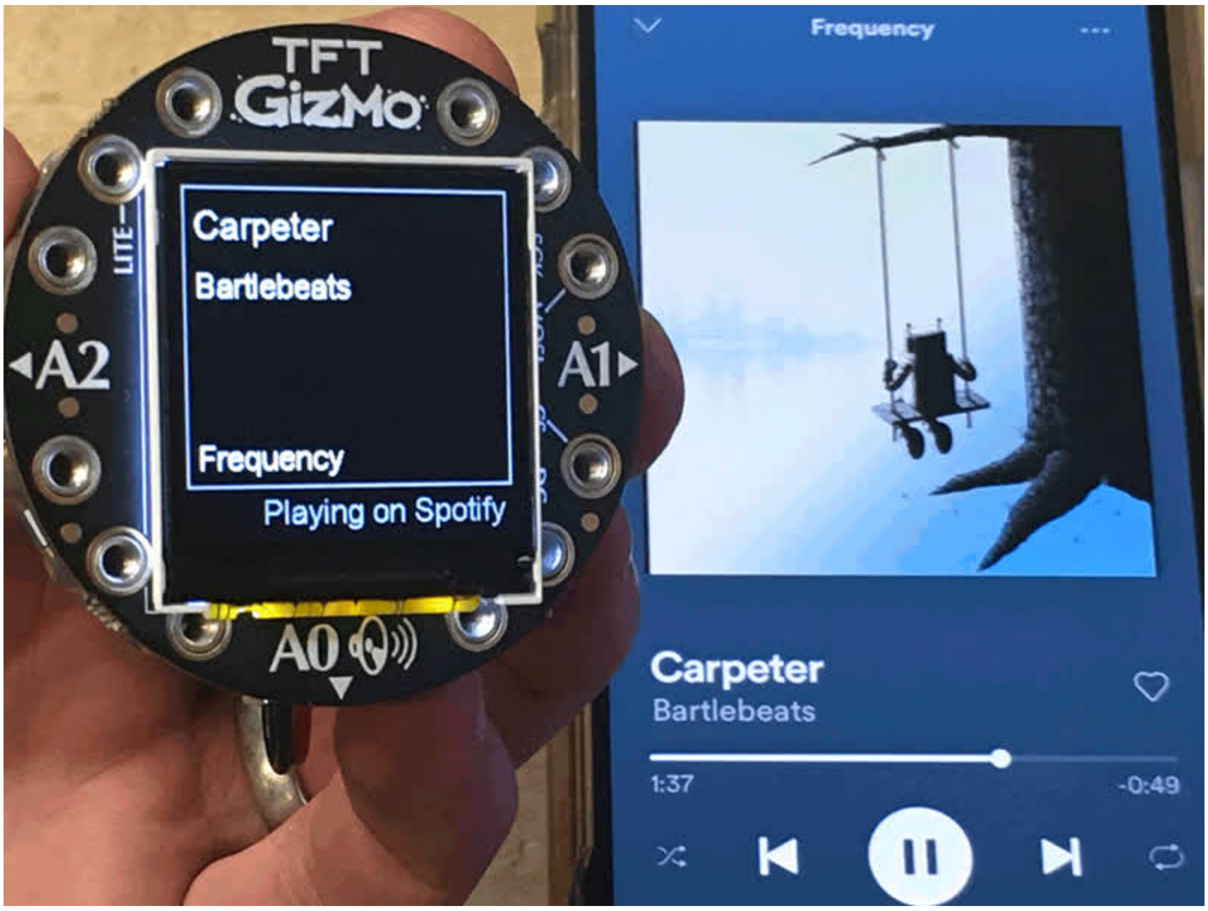
Now Playing: Bluetooth Apple Media Service Display

Created by John Park



https://learn.adafruit.com/now-playing-bluetooth-apple-media-service-display

Last updated on 2024-06-03 03:01:38 PM EDT

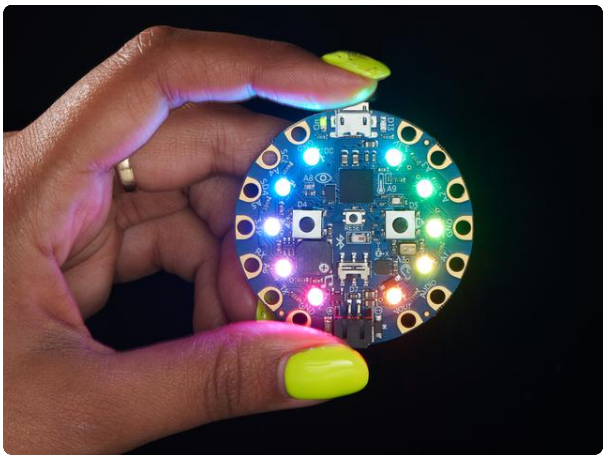
Table of Contents

Overview

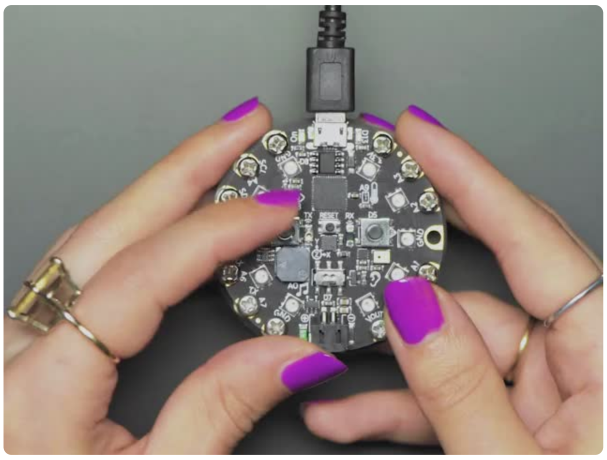
Now Playing! This wireless Bluetooth LE device displays whatever song/artist/album is playing on your iOS device, such as iPhone or iPad. Bonus feature -- use the Circuit Playground Bluefruit's buttons and slide switch to play and pause a track, or skip ahead or backwards to a different track.

Using CircuitPython code and the Apple Media Service library, we can easily connect the CPB to an iOS device, and then automatically retrieve track info. Buttons are assigned to play/pause and track skip, but you can customize them to use many other media control commands!

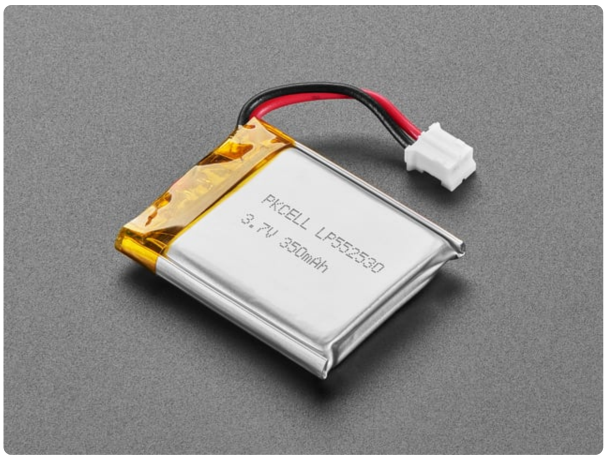
Parts



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

Or, get all the parts you need and more in the Project Pack:

Adafruit Circuit Playground Bluefruit Express Starter Kit
If you missed out on ADABOX 014, its not too late for you to pick up the parts necessary to build many of the projects! This kit pack doesn't come with tissue paper or the nifty...
https://www.adafruit.com/product/4504

Apple Media Service

The Apple Media Service (https://adafru.it/IAO)is a Bluetooth Low Energy (BLE) service that allows BLE devices, such as Apple Watches and remote controls, to interface with an iOS device's media apps, such as Music, Spotify, and Podcast apps.

Here's Apple's definition of the service:

> The **Apple Media Service** (AMS) is used with **Bluetooth** accessories that connect to iOS devices through **Bluetooth** low-energy links. It gives them a simple and convenient way to control **media** apps and access information about the **media** states of the connected iOS devices.

AMS lets devices control the media player with **remote commands** for many common controls, such as:

- play/pause
- next track

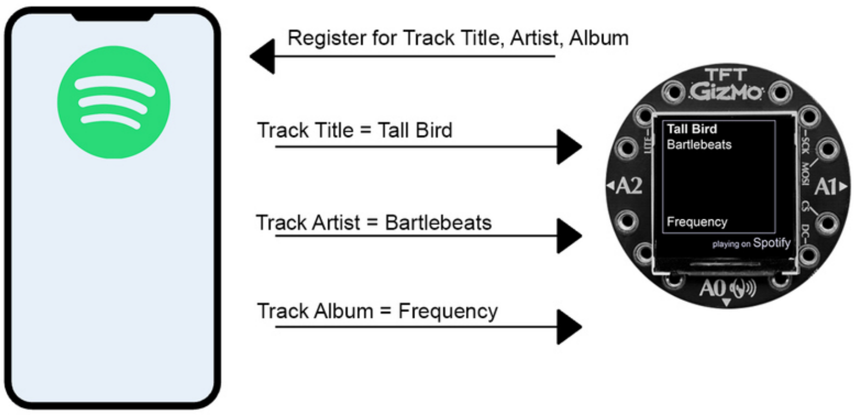
- previous track
- volume up/down
- repeat mode

AMS also can send information from the media player to the remote device when requested, including:

- track title
- track artist
- track album
- repeat mode
- track duration
- media player name (e.g., Spotify, Music, Podcast)

In this example, the remote device requests attributes values for **Track Title**, **Track Artist**, and **Track Album**.

The media player receives these requests and responds with **Tall Bird**, **Bartlebeats**, and **Frequency.**



For a full list of Remote Commands and Entity Attributes, check the [Apple Media Service Reference here](https://adafru.it/IAP) (https://adafru.it/IAP).

In order to make it simple to interface with your iOS device from an Adafruit Bluefruit nRF52840 board, we've created a BLE Apple Media library for CircuitPython. This library takes care of the nitty gritty details, and lets you get on with the fun part of creating a device to show what song is playing and to control the player remotely!

In code, you'll import the Apple Media Service library. Once the devices are connected, the service is instantiated with `ams = connection[AppleMediaService]` and you can then use simple commands in CircuitPython to control the media player, such as:

`ams.toggle_play_pause()`

`ams.next_track()`

`ams.volume_up()`

`ams.like_track()`

And, you can request media attribute info such as:

`ams.title`

`ams.album`

`ams.artist`

`ams.playing`

`ams.paused`

Full Command and Attribute Lists

You may decide you want to do something different with your device, or display different information. Below you'll find the commands that we can send and info we can request using the Adafruit_CircuitPython_BLE_Apple_Media library.

NOTE: Not all media apps support every command or info request.

Commands

- **`play`** -- Plays the current track. Does nothing if already playing
- **`pause`** -- Pauses the current track. Does nothing if already paused

- `toggle_play_pause` -- Plays the current track if it is paused. Otherwise it pauses the track
- `next_track` -- Stops playing the current track and plays the next one
- `previous_track` -- Stops playing the current track and plays the previous track
- `volume_up` -- Increases the playback volume
- `volume_down` -- Decreases the playback volume
- `advance_repeat_mode` -- Advances the repeat mode. Modes are: Off, One and All
- `advance_shuffle_mode` -- Advances the shuffle mode. Modes are: Off, One and All
- `skip_forward` -- Skips forwards in the current track
- `skip_backward` -- Skips backwards in the current track
- `like_track` -- Likes the current track
- `dislike_track` -- Dislikes the current track
- `bookmark_track` -- Bookmarks the current track

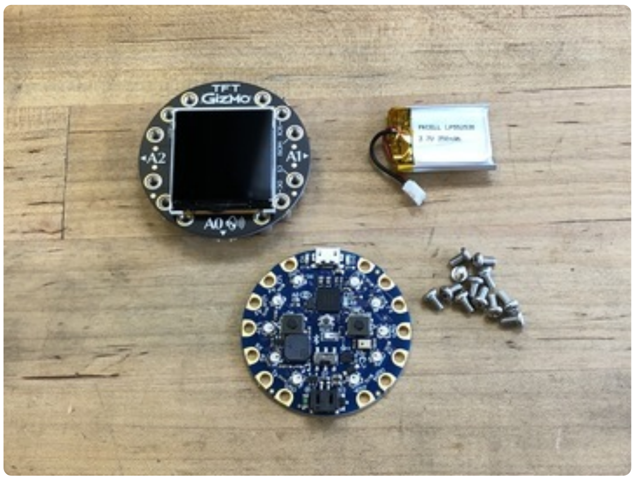
Info

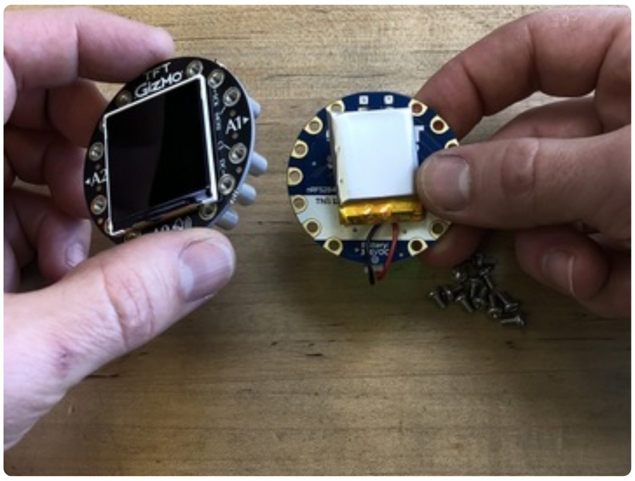
- `player_name` -- Name of the media player app
- `paused` -- True when playback is paused. False otherwise
- `playing` -- True when playback is playing
- `rewinding` -- True when playback is rewinding
- `fast_forwarding` -- True when playback is fast-forwarding
- `playback_rate` -- Playback rate as a decimal of normal speed
- `elapsed_time` -- Time elapsed in the current track. Not updated as the track plays. Use `(amount of time since read elapsed time) * playback_rate` to estimate the current `elapsed_time`
- `volume` -- Current volume
- `queue_index` -- Current track's index in the queue
- `queue_length` -- Count of tracks in the queue
- `shuffle_mode` -- Current shuffle mode as an integer. `0` = Off`, 1` = One`, 2` = All
- `repeat_mode` -- Current repeat mode as an integer. `0` = Off`, 1` = One`, 2` = All
- `artist --` Current track's artist name
- `album --` Current track's album name
- `title --` Current track's title (a.k.a., song name)
- `duration --` Current track's duration as a string

Assemble the Board

The TFT Gizmo will attach to either the CPB or CPX with enough room for the 350mAh battery to sandwich in between.

Prep the TFT Gizmo following these instructions (https://adafru.it/Hb9) by removing the twelve little Kapton tape dots on the end of each standoff.



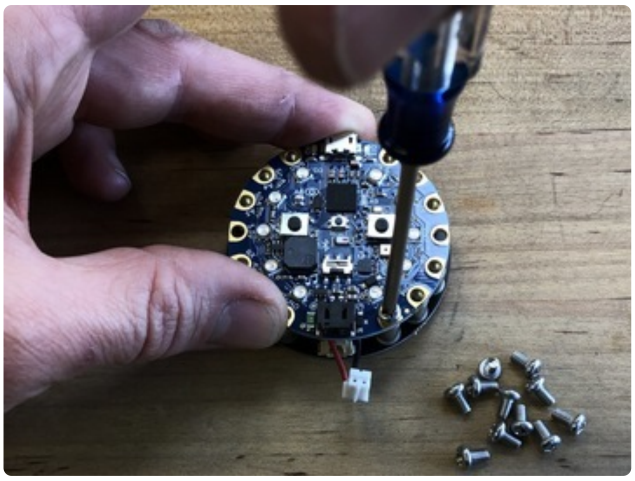


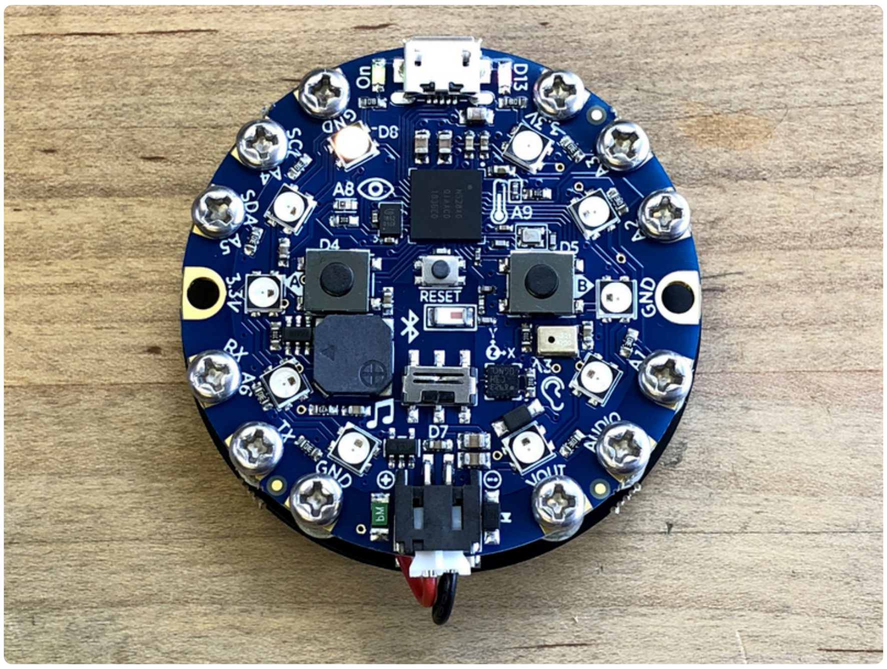
Board Connection

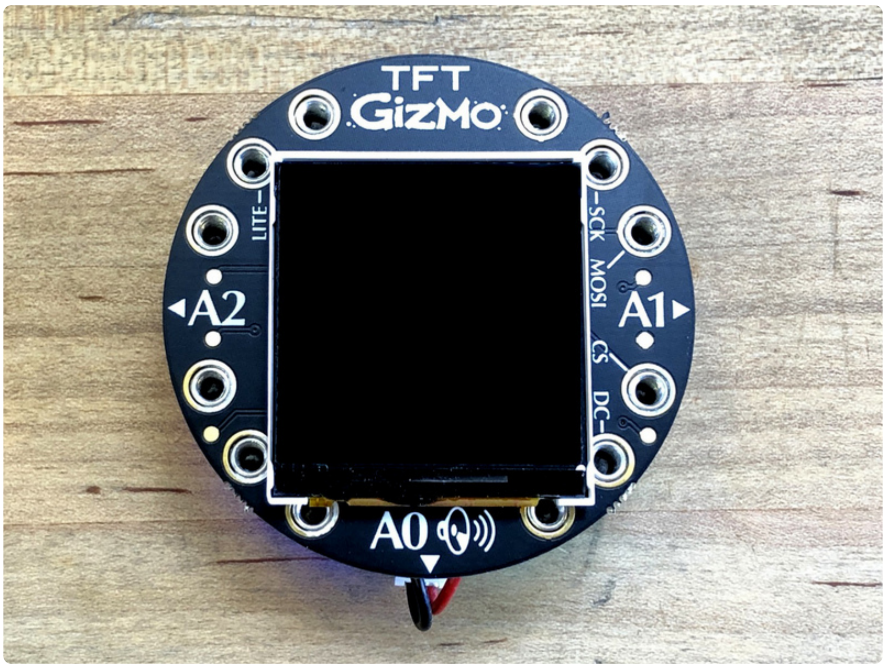
Making sure to orient the Circuit Playground board with the USB jack pointed "north" and with the TFT Gizmo's 12 o'clock marking also pointed "north", place the battery between the boards as shown. You can use a small bit of blue tack or thin double sided tape to keep the battery from wiggling around.

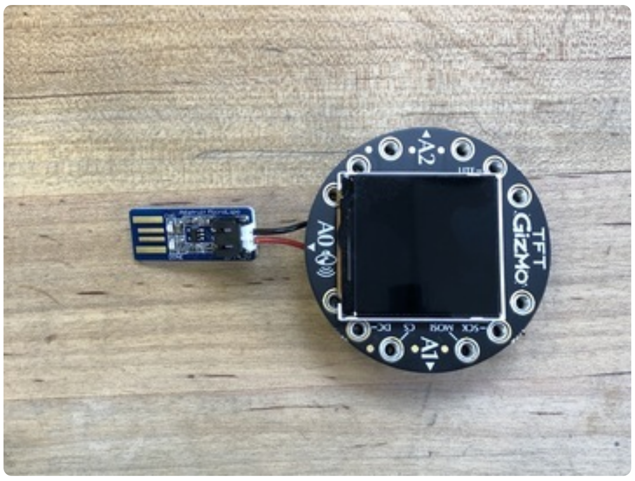
Also make sure the battery is oriented so the JST cable can plug into the Circuit Playground battery connector.

Screw the 12 M3 screws into the standoffs to connect the boards both electrically and mechanically.






TFT
GizMo
LITE
SCK
MOSI
CS
DC
A2
A1
A0



Charging

To charge the LiPoly battery, you'll need to unplug it from the Circuit Playground and into a LiPo charger.

Next, we'll put the code on the board -- for this project we'll use the **CPB_Eye_Terminator.UF2** file or **CPX_Eye_Terminator.UF2** file seen on the next page.

CircuitPython on Circuit Playground Bluefruit

Install or Update CircuitPython

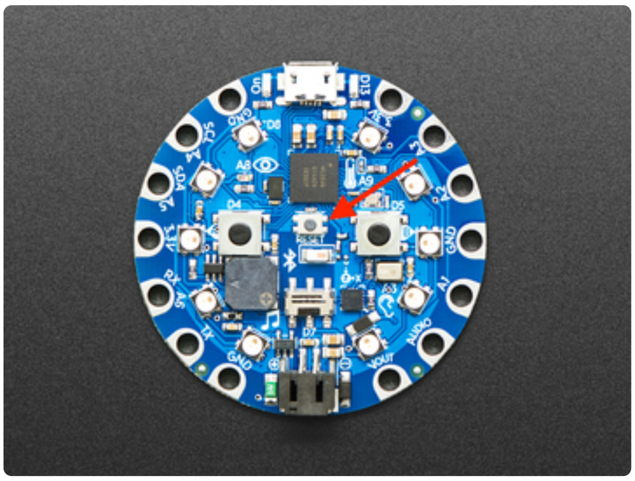
Follow this quick step-by-step to install or update CircuitPython on your Circuit Playground Bluefruit.

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)



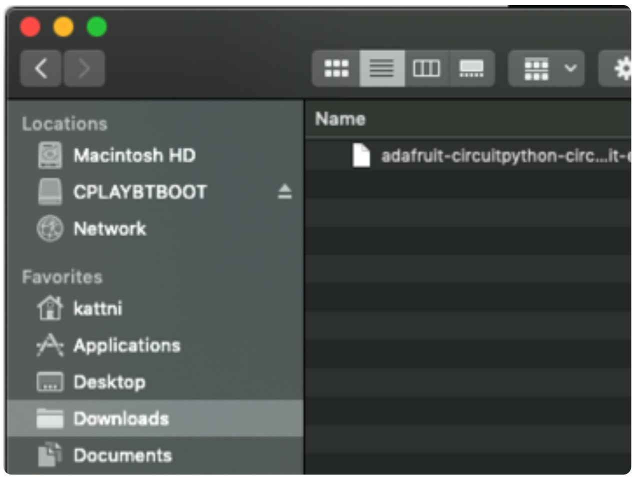
Plug your Circuit Playground Bluefruit into your computer using a known-good data-capable USB cable.

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.

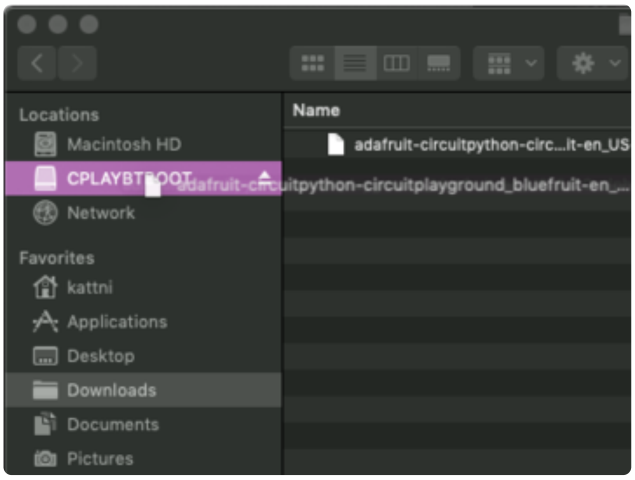
Double-click the small **Reset** button in the middle of the CPB (indicated by the red arrow in the image). The ten NeoPixel LEDs will all turn red, and then will all turn green. If they turn all red and stay red, check the USB cable, try another USB port, etc. The little red LED next to the USB connector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

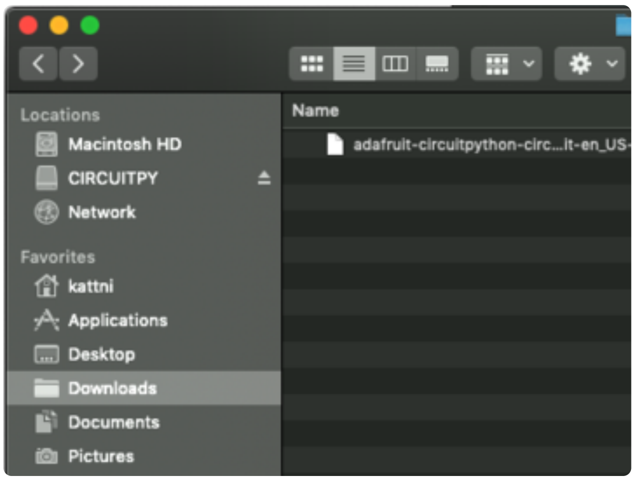
(If double-clicking doesn't do it, try a single-click!)



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.

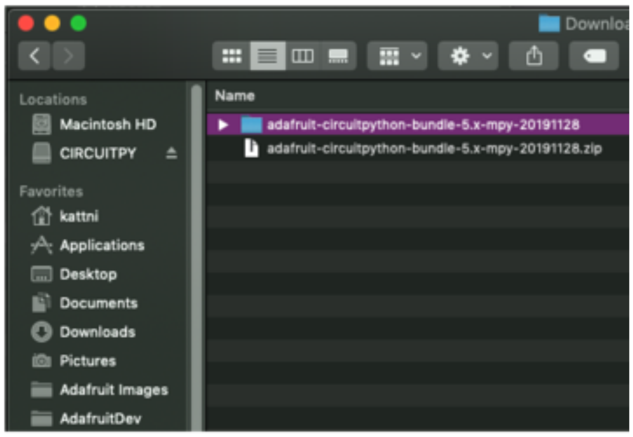
Installing CircuitPython Libraries on Circuit Playground Bluefruit

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.

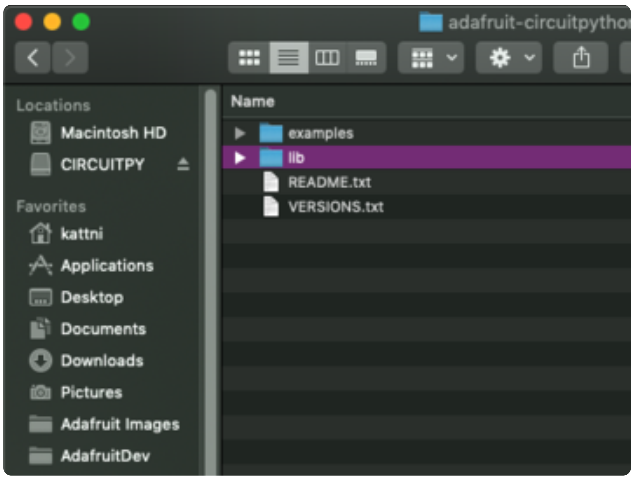
Download the latest library bundle from circuitpython.org

https://adafru.it/ENC



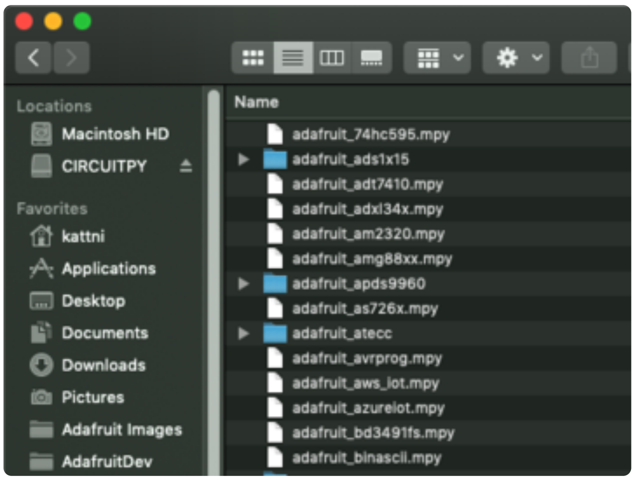
adafruit-circuitpython-bundle-5.x-mpy-20191128
adafruit-circuitpython-bundle-5.x-mpy-20191128.zip

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



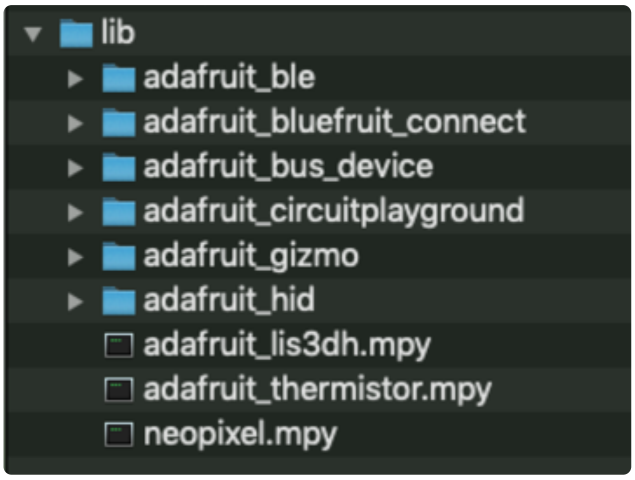
examples
lib
README.txt
VERSIONS.txt

Open the **lib** folder found within.



adafruit_74hc595.mpy
adafruit_ads1x15
adafruit_adt7410.mpy
adafruit_adxl34x.mpy
adafruit_am2320.mpy
adafruit_amg88xx.mpy
adafruit_apds9960
adafruit_as726x.mpy
adafruit_atecc
adafruit_avrprog.mpy
adafruit_aws_iot.mpy
adafruit_azureiot.mpy
adafruit_bd3491fs.mpy
adafruit_binascii.mpy

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**.



lib
adafruit_ble
adafruit_bluefruit_connect
adafruit_bus_device
adafruit_circuitplayground
adafruit_gizmo
adafruit_hid
adafruit_lis3dh.mpy
adafruit_thermistor.mpy
neopixel.mpy

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 the Apple Media Service Display

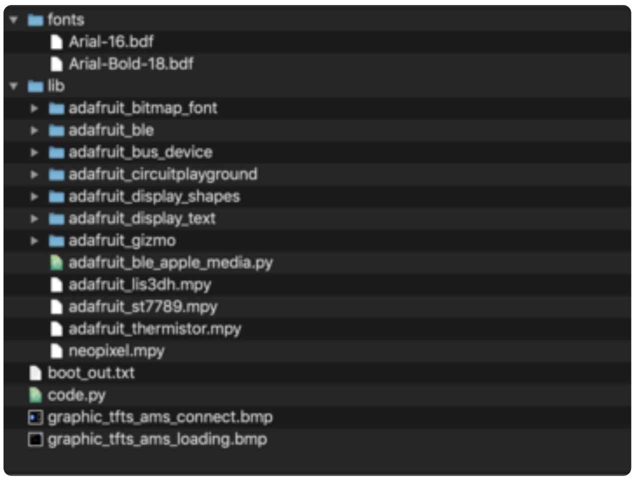
We've created this fairly simple program that allows your CPB to connect to an iOS device, Pair and Bond (so they'll auto-reconnect later), and send data between the two devices to display track info and playback commands.

Libraries

First, make sure you have these libraries that you copied over to the board following this guide page (https://adafru.it/I3c)

- **adafruit_ble**
- **adafruit_bus_device**
- **adafruit_circuitplayground**
- **adafruit_lis3dh.mpy**
- **adafruit_st7789.mpy**
- **adafruit_thermistor.mpy**
- **neopixel.mpy**

Then, we'll also add some libraries for dealing with the Apple Media Service, the TFT display, and the Circuit Playground buttons and switch.



```
fonts
    Arial-16.bdf
    Arial-Bold-18.bdf
lib
    adafruit_bitmap_font
    adafruit_ble
    adafruit_bus_device
    adafruit_circuitplayground
    adafruit_display_shapes
    adafruit_display_text
    adafruit_gizmo
    adafruit_ble_apple_media.py
    adafruit_lis3dh.mpy
    adafruit_st7789.mpy
    adafruit_thermistor.mpy
    neopixel.mpy
boot_out.txt
code.py
graphic_tfts_ams_connect.bmp
graphic_tfts_ams_loading.bmp
```

From the library bundle you downloaded in that guide page, transfer the following library onto the CPB boards' **/lib** directory:

adafruit_bitmap_font
adafruit_display_shapes
adafruit_display_text
adafruit_gizmo
adafruit_ble_apple_media.mpy

Your CBP should look like the screenshot above.

You'll also need to get the fonts and .bmp images for the project. 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.**

Text Editor

Adafruit recommends using the Mu editor for using your CircuitPython code with the Circuit Playground Bluefruit boards. You can get more info in this guide (https://adafru.it/ANO).

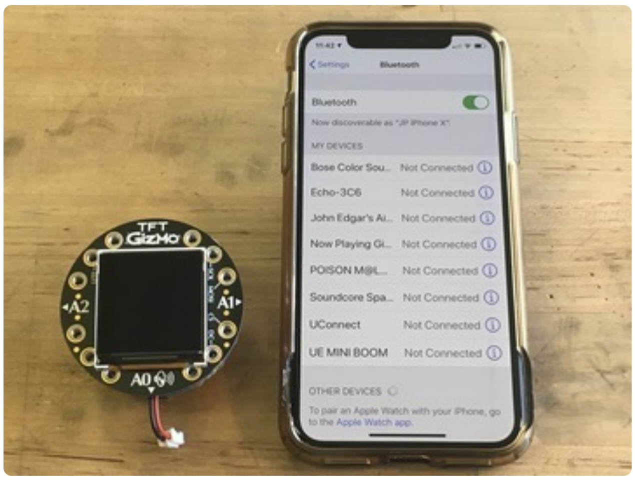
Alternatively, you can use any text editor that saves files.

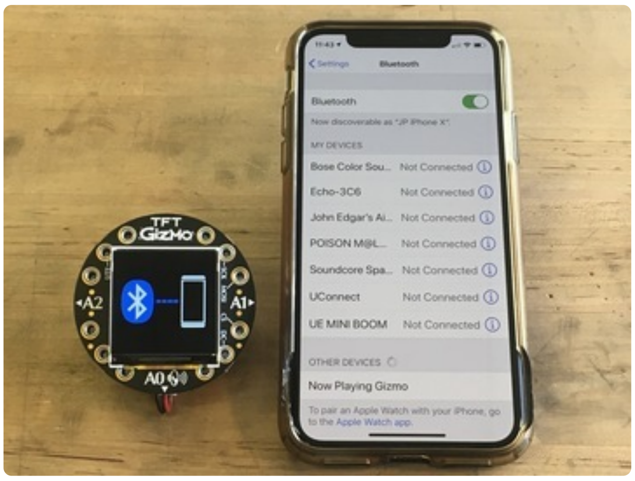
```
# SPDX-FileCopyrightText: 2020 Melissa LeBlanc-Williams for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
This example solicits that apple devices that provide notifications connect to it,
initiates
pairing, prints existing notifications and then prints any new ones as they arrive.
"""

import time
import displayio
import terminalio
from adafruit_gizmo import tft_gizmo
from adafruit_display_text.label import Label
from adafruit_display_shapes.rect import Rect
from adafruit_bitmap_font import bitmap_font
import adafruit_ble
from adafruit_ble.advertising.standard import SolicitServicesAdvertisement
from adafruit_ble_apple_media import AppleMediaService
from adafruit_ble_apple_media import UnsupportedCommand
from adafruit_circuitplayground import cp

BACKGROUND_COLOR = 0x49523b  # Gray
TEXT_COLOR = 0xFF0000  # Red
BORDER_COLOR = 0xAAAAAA  # Light Gray
STATUS_COLOR = BORDER_COLOR

# PyLint can't find BLERadio for some reason so special case it here.
radio = adafruit_ble.BLERadio() # pylint: disable=no-member
radio.name = "Now Playing Gizmo"
a = SolicitServicesAdvertisement()
a.solicited_services.append(AppleMediaService)
radio.start_advertising(a)

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)

def make_background(width, height, color):
    color_bitmap = displayio.Bitmap(width, height, 1)
    color_palette = displayio.Palette(1)
    color_palette[0] = color

    return displayio.TileGrid(color_bitmap,
                              pixel_shader=color_palette,
                              x=0, y=0)
```

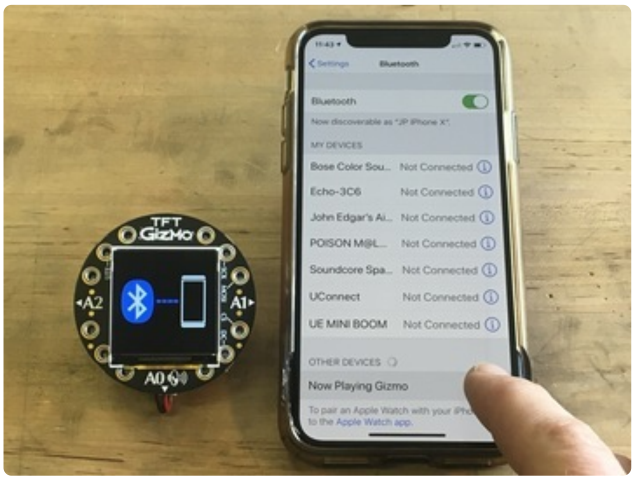
```
def load_font(fontname, text):
    font = bitmap_font.load_font(fontname)
    font.load_glyphs(text.encode('utf-8'))
    return font

def make_label(text, x, y, color, font=terminalio.FONT):
    if isinstance(font, str):
        font = load_font(font,
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz.,?()")
    text_area = Label(font, text=text, color=color)
    text_area.x = x
    text_area.y = y
    return text_area

def set_label(label, value, max_length):
    text = "{}".format(value)
    if len(text) > max_length:
        text = text[:max_length-3] + "..."
    label.text = text

def set_status(label, action_text, player):
    label.text = "{} on {}".format(action_text, player)
    _, _, label_width, _ = label.bounding_box
    label.x = display.width - 10 - label_width

display = tft_gizmo.TFT_Gizmo()
group = displayio.Group()
display.root_group = group

while True:
    if not radio.connected:
        group.append(wrap_in_tilegrid("/graphic_tfts_ams_connect.bmp"))

        while not radio.connected:
            pass

        group.pop()
    print("connected")

    known_notifications = set()

    # Draw the text fields
    print("Loading Font Glyphs...")
    group.append(wrap_in_tilegrid("/graphic_tfts_ams_loading.bmp"))
    title_label = make_label("None", 12, 30, TEXT_COLOR, font="/fonts/Arial-
Bold-18.bdf")
    artist_label = make_label("None", 12, 70, TEXT_COLOR, font="/fonts/
Arial-16.bdf")
    album_label = make_label("None", 12, 184, TEXT_COLOR, font="/fonts/
Arial-16.bdf")
    status_label = make_label("None", 80, 220, STATUS_COLOR, font="/fonts/
Arial-16.bdf")
    group.pop()
    group.append(make_background(240, 240, BACKGROUND_COLOR))
    border = Rect(4, 4, 232, 200, outline=BORDER_COLOR, stroke=2)
    group.append(title_label)
    group.append(artist_label)
    group.append(album_label)
    group.append(status_label)
    group.append(border)

    while radio.connected:
        for connection in radio.connections:
            try:
                if not connection.paired:
                    connection.pair()
                    print("paired")

                ams = connection[AppleMediaService]
            except (RuntimeError, UnsupportedCommand, AttributeError):
                # Skip Bad Packets, unknown commands, etc.
                continue
            set_label(title_label, ams.title, 18)
            set_label(album_label, ams.album, 21)
            set_label(artist_label, ams.artist, 21)
            action = "?"
            if ams.playing:
                action = "Playing"
            elif ams.paused:
                action = "Paused"
            set_status(status_label, action, ams.player_name)
        if cp.button_a:
            ams.toggle_play_pause()
            time.sleep(0.1)

        if cp.button_b:
            if cp.switch:
                ams.previous_track()
            else:
                ams.next_track()
            time.sleep(0.1)

    print("disconnected")
    # Remove all layers
    while len(group):
        group.pop()
```

Use

Next, let's look at how to connect your devices and use them together.


TFT Gizmo
A2
A1
A0
Settings
Bluetooth
Bluetooth
MY DEVICES
Bose Color Sou... Not Connected
Echo-3C6 Not Connected
John Edgar's Ai... Not Connected
Now Playing Gi... Not Connected
POISON M@L... Not Connected
Soundcore Spa... Not Connected
UConnect Not Connected
UE MINI BOOM Not Connected
OTHER DEVICES
To pair an Apple Watch with your iPhone, go to the Apple Watch app.


TFT Gizmo
A2
A1
A0
Settings
Bluetooth
Bluetooth
MY DEVICES
Bose Color Sou... Not Connected
Echo-3C6 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
Now Playing Gizmo
To pair an Apple Watch with your iPhone, go to the Apple Watch app.


TFT Gizmo
A2
A1
A0
Settings
Bluetooth
Bluetooth
MY DEVICES
Bose Color Sou... Not Connected
Echo-3C6 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
Now Playing Gizmo

Connection

Plug in the power on the Now Playing Gizmo, and open the Bluetooth setting on your iOS device.

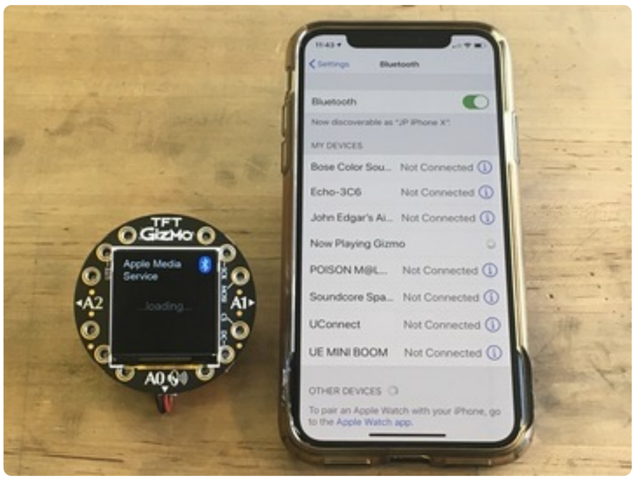
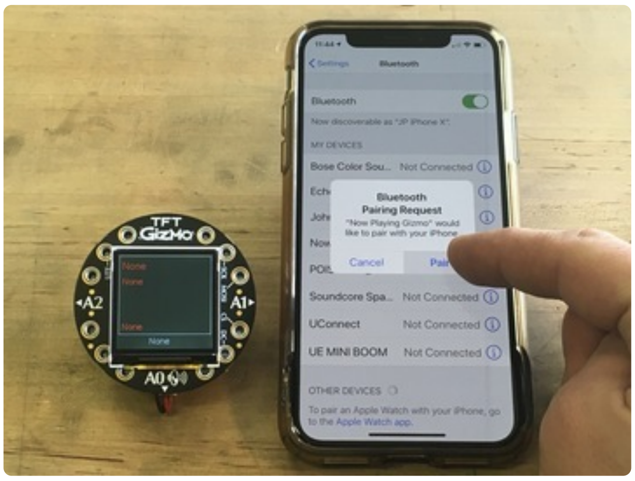
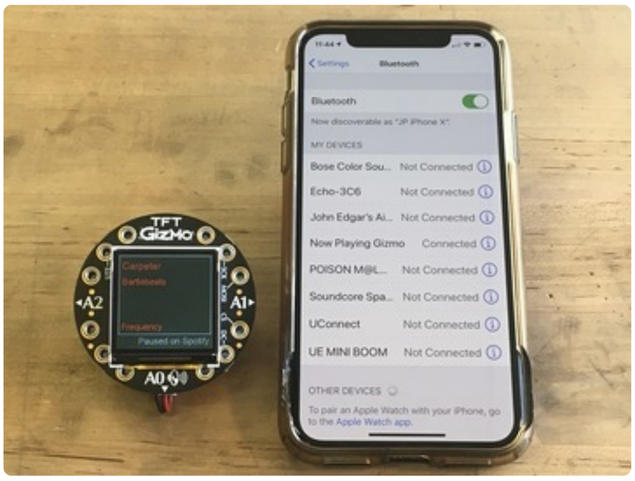
Once the Gizmo starts up, you'll see a BLE connection icon on the display. With Bluetooth turned on on the iOS device, you will see the **Now Playing Gizmo** pop up in the Other Devices list. Go ahead and click it.

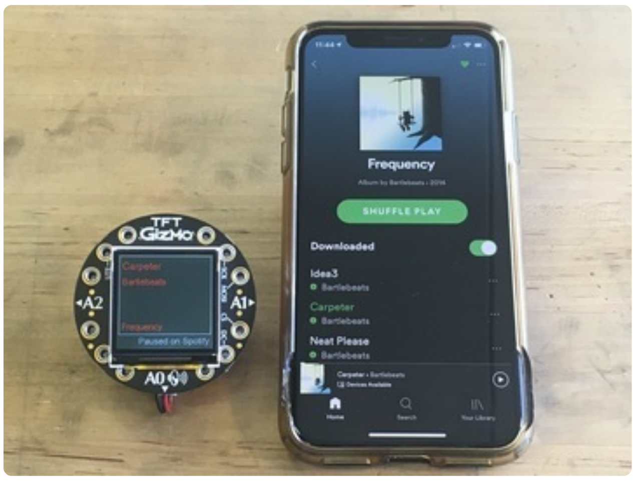
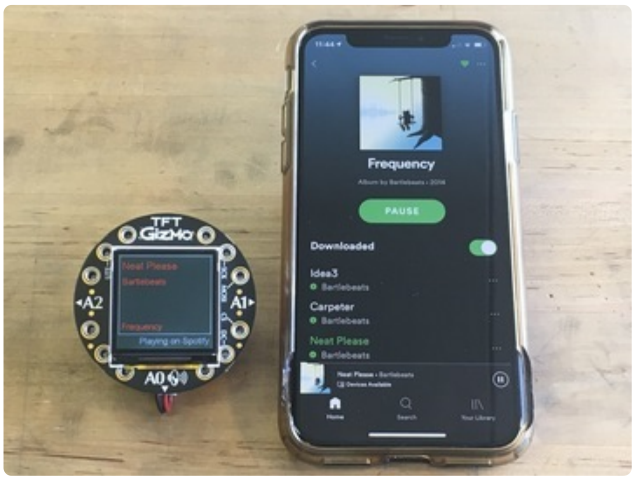
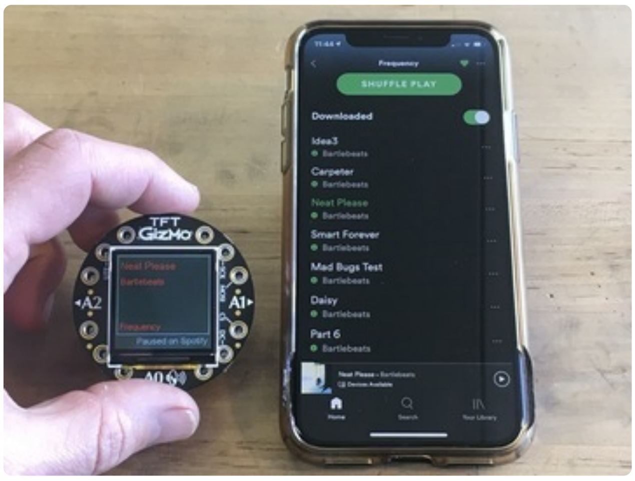
Note: the device may have a name like **CIRCUITPY939d** the first time you connect.

The Gizmo will switch to the loading screen while it prepares the font glyphs for display and does some other setup.

You'll then see a **Bluetooth Pairing Request** dialog box pop up on the iOS device. Go ahead and press **Pair**.

The **Now Playing Gizmo** will show up in your **My Devices** list as **Connected**.


Apple Media Service
...loading...

Bluetooth Pairing Request
"Now Playing Gizmo" would like to pair with your iPhone.
Cancel
Pair

Carpenter
Battlebeats
Frequency
Paused on Spotify

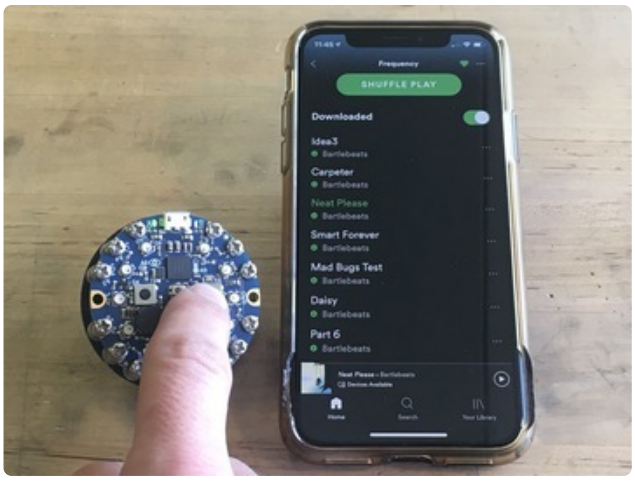
Get Track Info

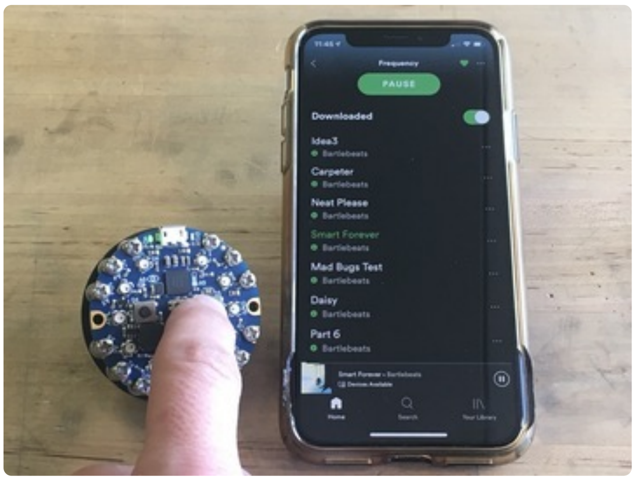
Go ahead and launch a media player app, such as Spotify, shown here.

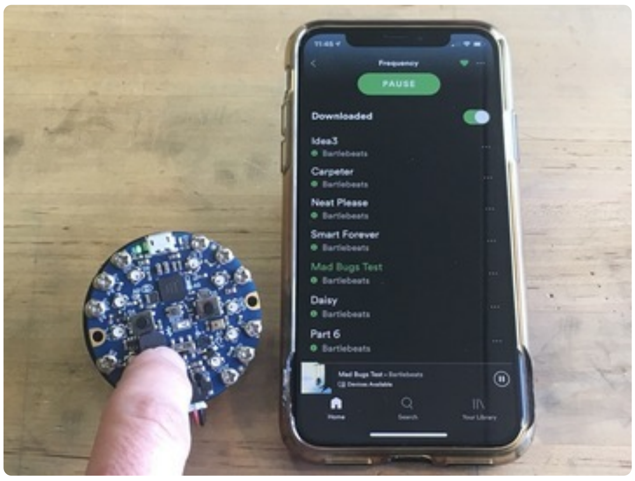
You'll see that the Now Playing Gizmo displays the track title, artist, and album names in the track info box.

There's also a line at the bottom of the display that tells you if the player is paused or playing, and the name of the player.

Change the song on your iOS device, and it will update on the Now Playing Gizmo!





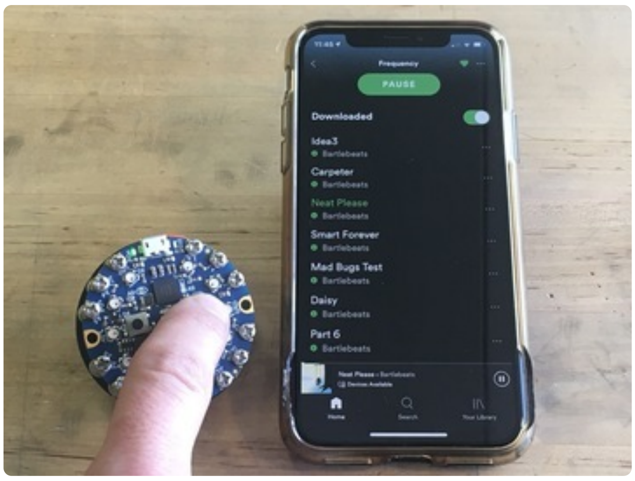


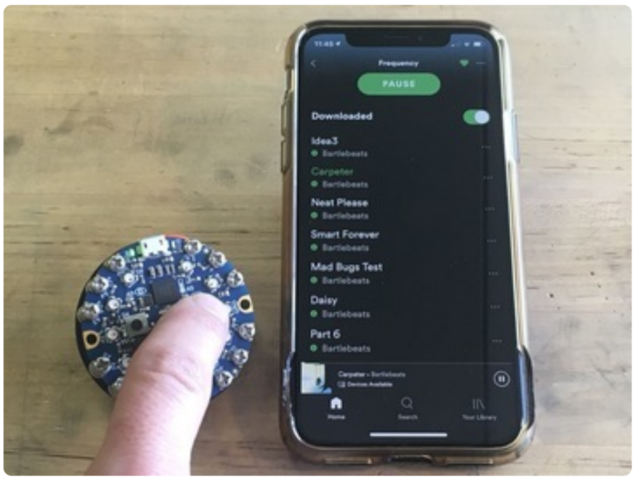
Send Media Control Commands

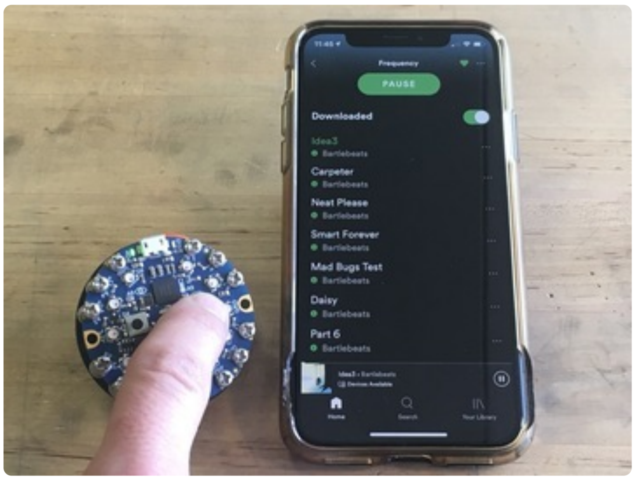
You can also send the player commands from the Now Playing Gizmo.

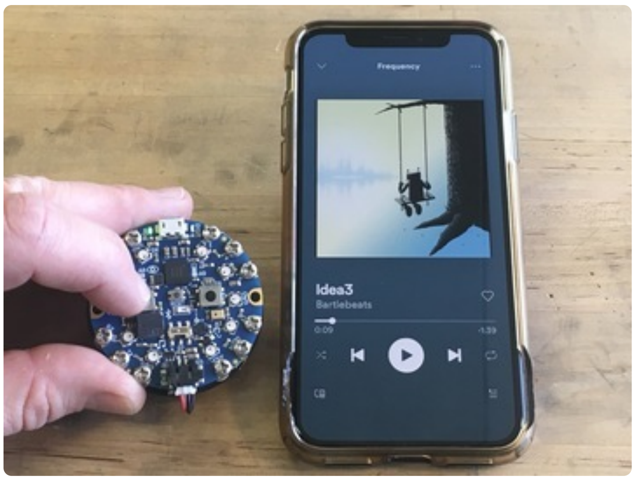
Press the B button (on the right) of the Circuit Playground Bluefruit to skip to the next track.

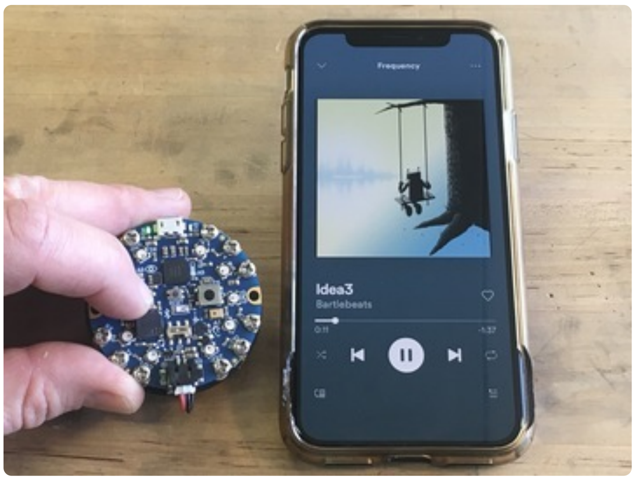
Or, flip the slide switch to the left, and now the B button will skip backwards a track.


Frequency
PAUSE
Downloaded
Idea3
Carpeter
Neat Please
Smart Forever
Mad Bugs Test
Daisy
Part 6


Frequency
PAUSE
Downloaded
Idea3
Carpeter
Neat Please
Smart Forever
Mad Bugs Test
Daisy
Part 6


Frequency
PAUSE
Downloaded
Idea3
Carpeter
Neat Please
Smart Forever
Mad Bugs Test
Daisy
Part 6

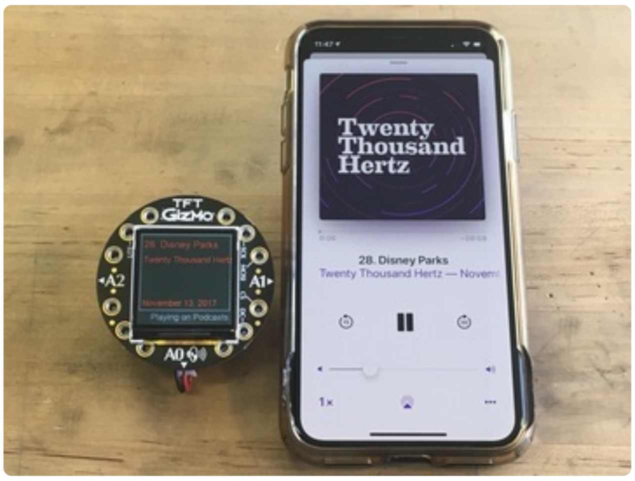
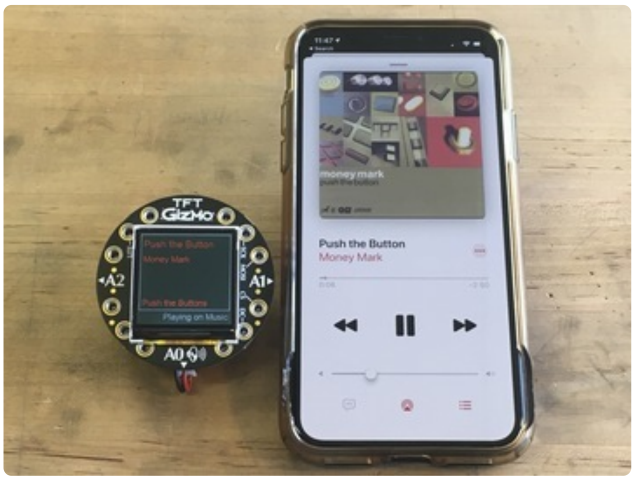




Play/Pause

At any time, you can use the A button to toggle the pause/play of the player.

When you do, not only will the player react, but it will also send the `ams.playing` and `ams.paused` values so the text on the Now Playing Gizmo will update accordingly.

Switch Apps

When you start playing media from a different app, such as Apple's default iOS player Music (formerly iTunes) or Podcast, the Now Playing Gizmo updates automatically!

Have fun with your Now Playing Gizmo, and try out some of the other info and commands that are possible in CircuitPython with the library!