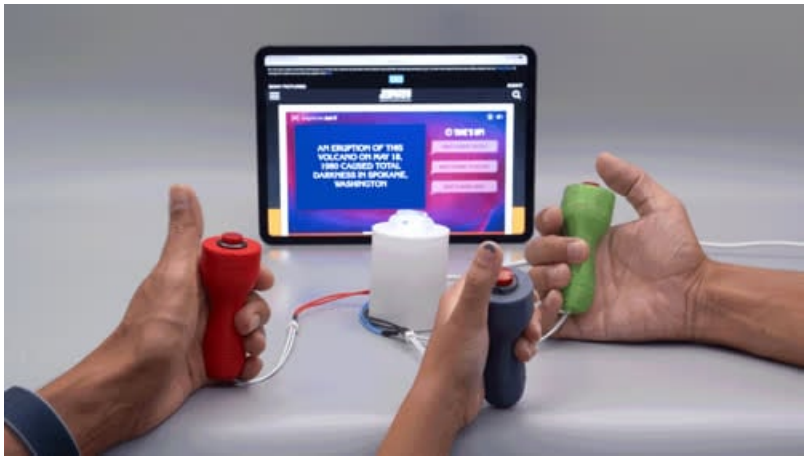
adafruit learning system

DIY Quiz Show Controller System

Created by Dylan Herrada



Last updated on 2021-06-24 10:59:02 AM EDT

Guide Contents

Overview

Build a Jeopardy-like Game Show Buzzer with a Circuit Playground Bluefruit!

The first player to press their buzzer will illuminate the Host controller with the color of the buzzer handle!

You can even enable Bluetooth button presses to a computer or mobile device!

The code is written in CircuitPython, so it's easy to change the sound effects, colors and button presses!

Parts

Components are housed in the Host Case with access to the USB port.

3D Printed

The parts are 3D Printed with an easy to assemble snap fit parts and print with minimal supports.

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...

$24.95

In Stock

Add to Cart

Bolt-On Kit for Circuit Playground, micro:bit, Flora or Gemma

You have a Circuit Playground Express, and want to connect some wires to it for adding LEDs or sensors or speakers? You can use our...

$1.50

In Stock

Add to Cart

Your browser does not support the video tag.

Adafruit STEMMA Speaker - Plug and Play Audio Amplifier

Hey, have you heard the good news? With Adafruit STEMMA boards you can easily and safely plug sensors and devices together, like this Adafruit STEMMA Speaker - Plug and Play...

$5.95

In Stock

Add to Cart

Your browser does not support the video tag.

Arcade Button with LED - 30mm Translucent Clear

A button is a button, and a switch is a switch, but these translucent arcade buttons are in a class of their own. Particularly because they have LEDs built right...

$2.50

In Stock

Add to Cart

Arcade Button Quick-Connect Wire Pairs - 0.11" (10 pack)

Quick connector wire sets will make wiring up our arcade-style or metal buttons quicky-quick. Each wire comes as a 'pair' with two 0.11" quick-connects pre-crimped onto...

$4.95

In Stock

Add to Cart

16mm Panel Mount Momentary Pushbutton - Red

OK, this item is pretty simple - it's a panel mount pushbutton. It's not that exciting, no LEDs, no bells & whistles. But we really like it anyways – look at that...

$0.95

In Stock

Add to Cart

Premium Silicone Covered Extension Jumper Wires - 200mm x 40

These premium extension jumper wires are handy for making wire harnesses or jumpering between headers on PCBs. They're 200mm (~7.8") long and come loose as a pack of...

$11.95

In Stock

Add to Cart

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.

- buzz-button-top
- buzz-button-mid
- buzz-button-btm
- CPX-Case-lid
- CPX-Case
- CPX-Case-btm

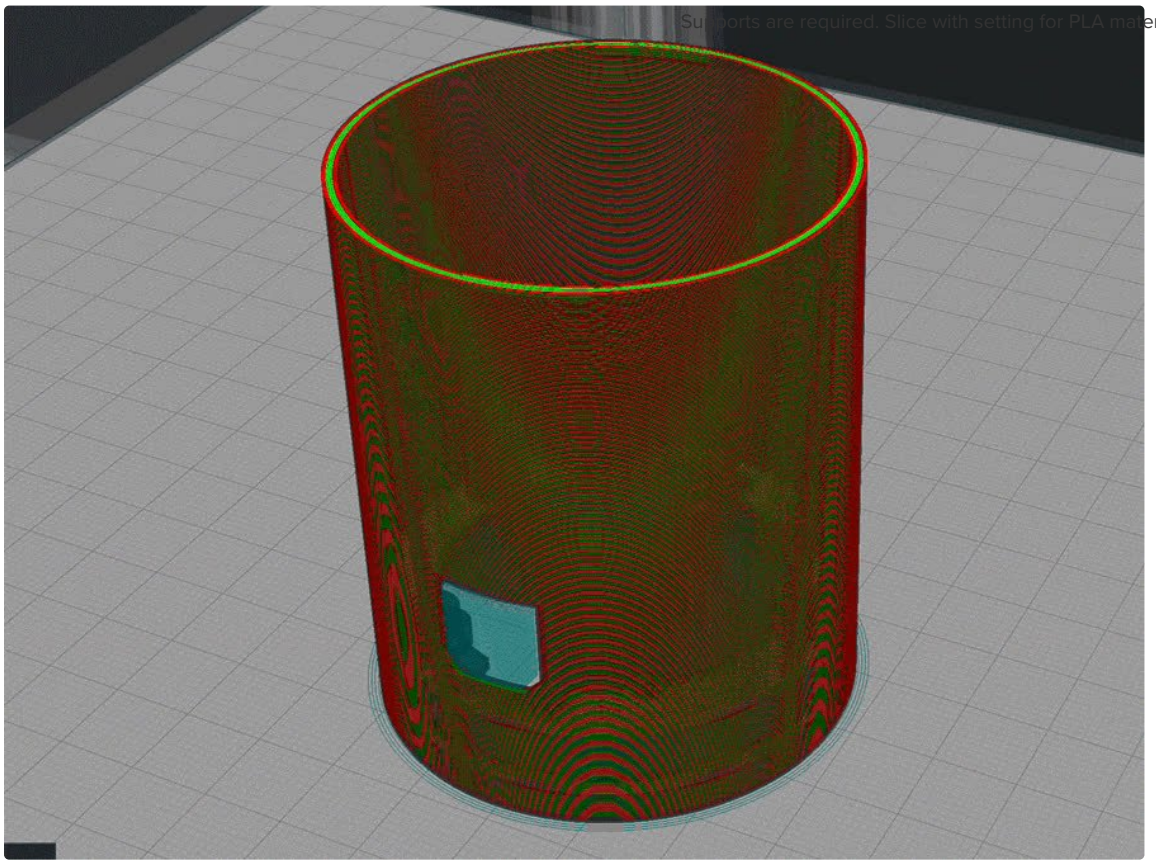
https://adafru.it/Tfh

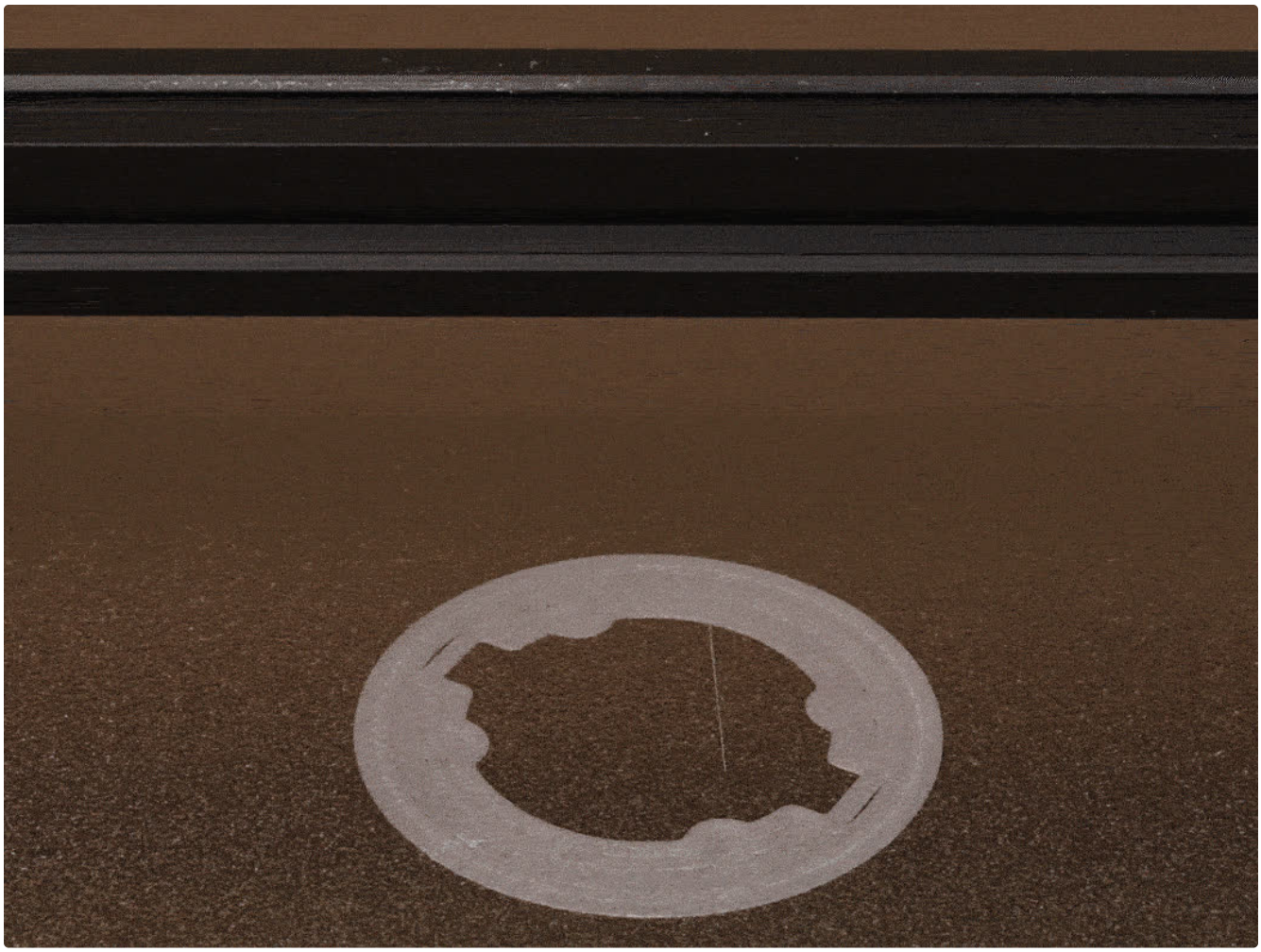
https://adafru.it/Tfh

https://adafru.it/TtB

https://adafru.it/TtB

Slicing Parts





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

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

Supports

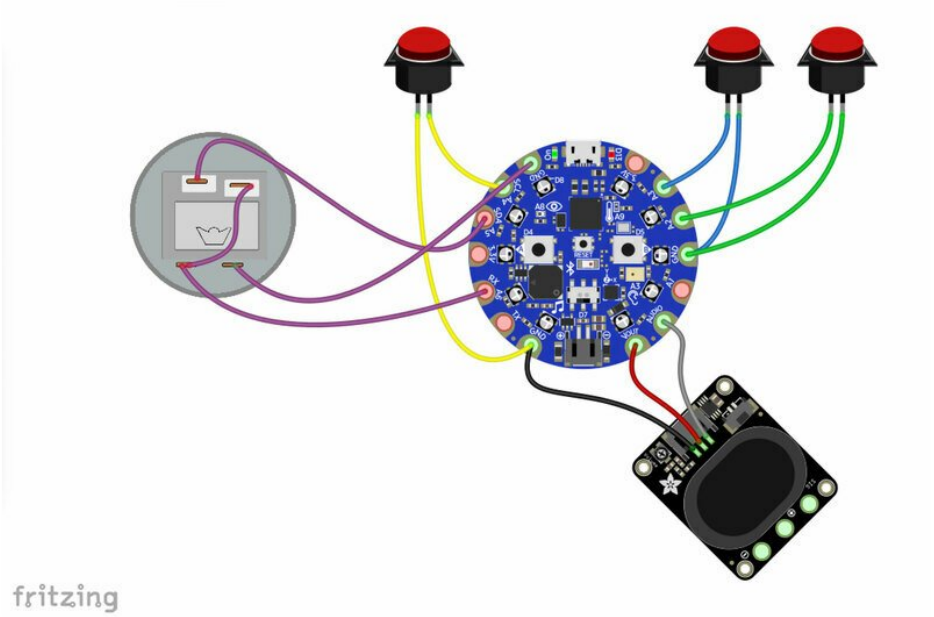
- Support Extrusion Width: .2
- Support Density: 4%
- Support Z Height: .21
- Interface: Off
- Support Roof: On
- Support Pattern: Zig Zag
- Support Roof Pattern: Zig Zag

Build Plate Adhesion

- Type: Brim
- Line Count: 6
- Brim on inside + outside

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).



Wired Connections

The Circuit Playground Bluefruit is powered by USB battery via the USB port. The STEMMA Speaker, Host and player buttons are connected to the pads on the Circuit Playground

Player Buzzers

- **Blue** Player connects to **A3** and one **Ground** on the **Circuit Playground**
- **Green** Player connects to **A2** and one **Ground** on the **Circuit Playground**
- **Yellow** Player connects to **A4** and one **Ground** on the **Circuit Playground**

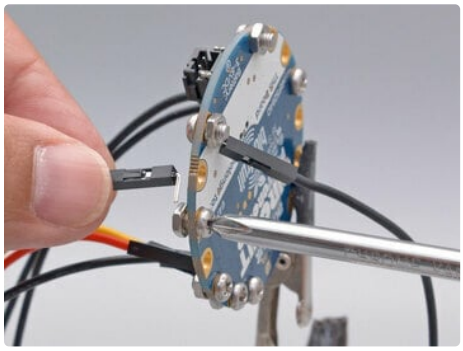
Host Button

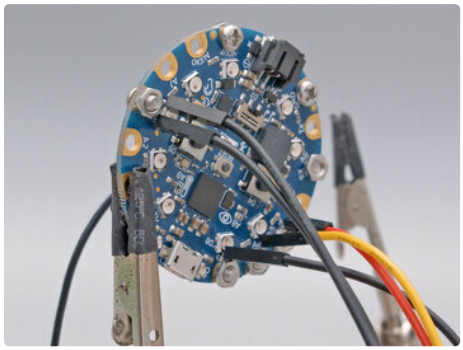
- Host connects to **A6** and **A5** and a **Ground**on the **Circuit Playground**

STEMMA Speaker

- **Ground** (black) connects to **Ground** on the **Circuit Playground**
- **Power** (red) connects to **VOUT** on the **Circuit Playground**
- **Signal** (white) connects to **AUDIO** on the **Circuit Playground**

Assemble





Bolt-on Screws

You can make this project plug and play by using the bolt-on screw kit on the Circuit Playground Bluefruit.

They screw on with hex nuts to hold wires against the pad.

This will allow jumper wires to easily attach into each connection making this project completely modular!

Mount CPB inside Case

Use M3x6mm screws to mount the Circuit Playground Bluefruit inside the case

Prep Speaker Wires

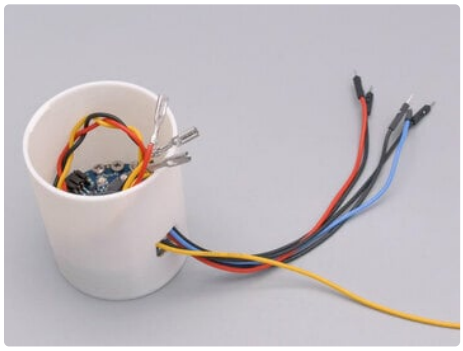
Shorten the speaker wires to fit inside the bottom side of the case.

Mount Speaker

Use M2.5x6mm long screws to mount the Stemma Speaker

Speaker Lid

The speaker lid press fits to the bottom of the case

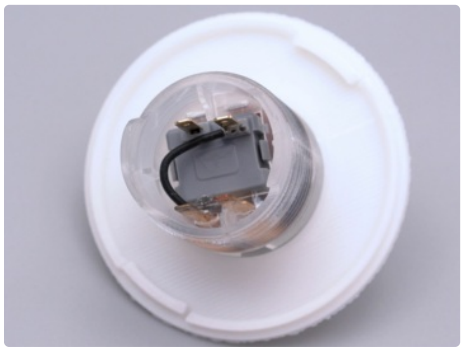


Buzzer Wires

The buzzer wires pass through the USB port opening on the case.

Carefully coil the wires to fit inside the case.

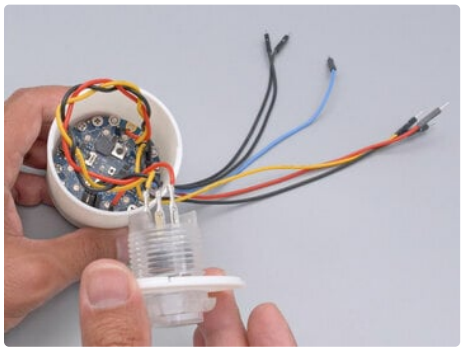
Align the Host button lid to the snap fit edges on the case to press fit them together.



Host Button

Solder the ground connections together to minimize wiring. Shorten the Arcade Button Quick-Connect Wires to fit inside the case.

You can modify the Quick-Connect ends with male jumpers if you are using the bolt-on screws on the Circuit Playground.



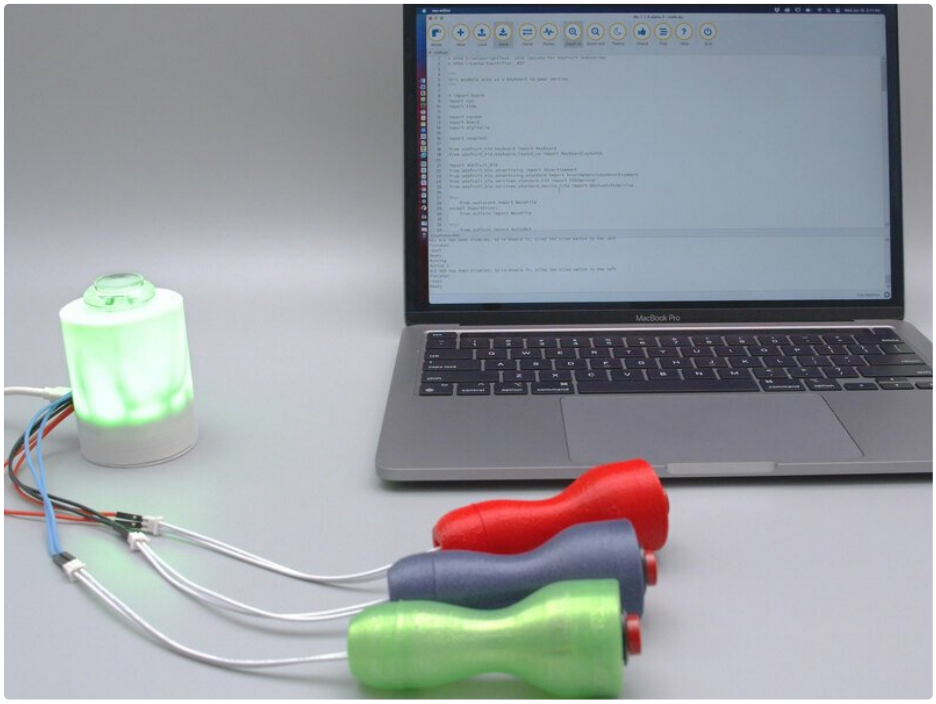
Player Buzzers

The Red 16mm Panel Mount Momentary Pushbutton twist on to the top handle part. Quick-Connects attach to each leg and then pass through the handle center and then the bottom handle part.

The three buzzer parts screw fit together. Connect jumpers to extend the length of the buzzer.

Code the Controller

Installing the Project Code



Download a zip of the project by clicking **Download Project Bundle** below.

After unzipping the file, copy **code.py** and **jeopardy.wav** to the **CIRCUITPY** drive which appears in your operating system File Explorer or Finder when the Circuit Playground is connected to your computer via a known good USB cable.

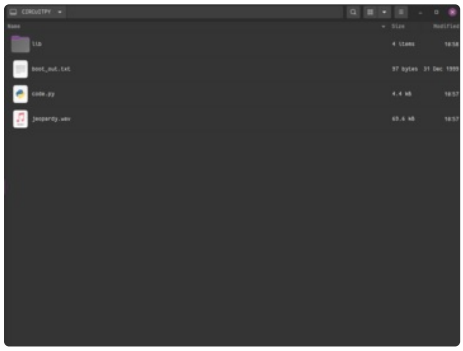
```
# SPDX-FileCopyrightText: 2021 Dylan Herrada for Adafruit Industries
# SPDX-License-Identifier: MIT

# General imports
import time
import random
import board
import digitalio
import neopixel

# HID imports
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS

# BLE imports
import adafruit_ble
from adafruit_ble.advertising import Advertisement
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.standard.hid import HIDService
from adafruit_ble.services.standard.device_info import DeviceInfoService

try:
    from audiocore import WaveFile
except ImportError:
    from audioio import WaveFile

try:
    from audioio import AudioOut
except ImportError:
```

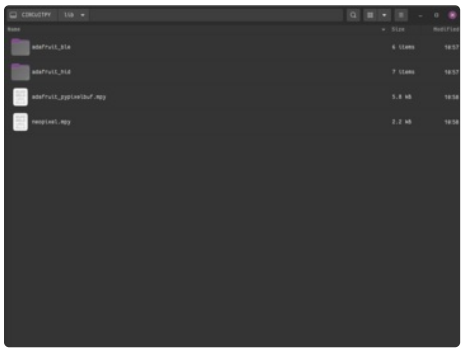
```
except ImportError:
    try:
        from audiopwmio import PWMAudioOut as AudioOut
    except ImportError:
        pass  # not always supported by every board!

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


# Make the input buttons
btn1 = digitalio.DigitalInOut(board.D10)  # Marked A3
btn1.direction = digitalio.Direction.INPUT
btn1.pull = digitalio.Pull.UP

btn2 = digitalio.DigitalInOut(board.D9)  # Marked A2
btn2.direction = digitalio.Direction.INPUT
btn2.pull = digitalio.Pull.UP

btn3 = digitalio.DigitalInOut(board.D3)  # Marked SCL A4
btn3.direction = digitalio.Direction.INPUT
btn3.pull = digitalio.Pull.UP

central = digitalio.DigitalInOut(board.D0)  # Marked RX A6
central.direction = digitalio.Direction.INPUT
central.pull = digitalio.Pull.UP

led = digitalio.DigitalInOut(board.D2)  # Marked SDA A5
led.switch_to_output()
led.value = False

buttons = [btn1, btn2, btn3]
upper = len(buttons) - 1

ble_enabled = digitalio.DigitalInOut(board.SLIDE_SWITCH)
ble_enabled.direction = digitalio.Direction.INPUT
ble_enabled.pull = digitalio.Pull.UP

pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=0.5)
# Use default HID descriptor
hid = HIDService()
device_info = DeviceInfoService(
    software_revision=adafruit_ble.__version__, manufacturer="Adafruit Industries"
)
advertisement = ProvideServicesAdvertisement(hid)
advertisement.appearance = 961
scan_response = Advertisement()

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

if ble_enabled.value:
    print("advertising")
    ble.start_advertising(advertisement, scan_response)

k = Keyboard(hid.devices)
kl = KeyboardLayoutUS(k)

wave_file = open("jeopardy.wav", "rb")
wave = WaveFile(wave_file)
audio = AudioOut(board.SPEAKER)

while True:
    if ble_enabled.value:
        while not ble.connected:
            pass
        if ble.connected:
            print("Connected")
```

```
            led.value = True
            time.sleep(0.1)
            led.value = False
            time.sleep(0.1)
            led.value = True
            time.sleep(0.1)
            led.value = False

    while ble.connected or not ble_enabled.value:
        if not central.value:
            led.value = True
            print("Running")
            while True:
                i = random.randint(0, upper)
                if not buttons[i].value:
                    break

            audio.play(wave)
            if i == 0:
                print("Button 1")
                pixels.fill((0, 0, 255))
                if ble_enabled.value:
                    kl.write("Button 1")
            elif i == 1:
                print("Button 2")
                pixels.fill((0, 255, 0))
                if ble_enabled.value:
                    kl.write("Button 2")
            elif i == 2:
                print("Button 3")
                pixels.fill((255, 255, 255))
                if ble_enabled.value:
                    kl.write("Button 3")

            if not ble_enabled.value:
                print(
                    "BLE HID has been disabled, slide the slide switch to the left to re-enable"
                )

            print("Finished")
            led.value = False

            while central.value:
                pass

            print("reset")
            pixels.fill((0, 0, 0))
            led.value = True
            time.sleep(0.5)
            led.value = False
            print("Ready")
        if ble_enabled.value:
            if not ble.connected:
                break
    else:
        continue
    break
```

There are also a few libraries you'll need to copy over to the Circuit Playground in the bundle **lib** directory - copy the whole **lib** directory to **CIRCUITPY**:

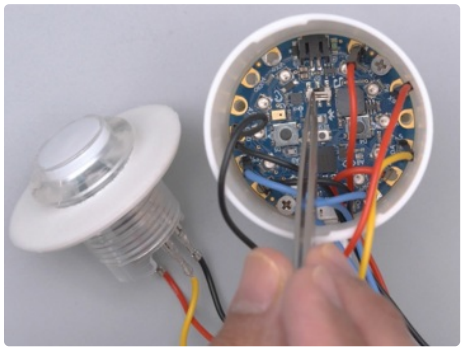
- **neopixel**
- **adafruit_pypixelbuf**
- **adafruit_hid**
- **adafruit_ble**





After you've done all that, this is what your **CIRCUITPY** drive should look like.

Usage



First, you should decide if you want to use BLE or not. If you do, make sure the slide switch is to the left. To disable BLE, slide it to the right.

If BLE is enabled, the central button will flash a few times when it is connected to your device. If it is disabled, then the code should just be ready to run about a second or two after power is supplied.

When you're ready for people to buzz in, press the central button. It should light up to show that it is ready. At this point, the players can buzz in. The Circuit Playground will turn the color of the player who buzzed in and if BLE is enabled, it will also type something to say which buzzer was pressed.

Finally, to reset this, press the central button again and it will flash to say that it is ready to start again.

Code Run-Through

First, the code makes all the required imports.

```
# General imports
import time
import random
import board
import digitalio
import neopixel

# HID imports
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS

# BLE imports
import adafruit_ble
from adafruit_ble.advertising import Advertisement
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.standard.hid import HIDService
from adafruit_ble.services.standard.device_info import DeviceInfoService

try:
    from audiocore import WaveFile
except ImportError:
    from audioio import WaveFile

try:
    from audioio import AudioOut
except ImportError:
    try:
        from audiopwmio import PWMAudioOut as AudioOut
    except ImportError:
        pass  # not always supported by every board!
```

Then, it sets up all the speaker that will be used to play a sound when a contestant buzzes in.

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

Next, all the buttons are defined and a list is created to make determining which button is pressed a bit more elegant.

```
# Make the input buttons
btn1 = digitalio.DigitalInOut(board.D10)  # Marked A3
btn1.direction = digitalio.Direction.INPUT
btn1.pull = digitalio.Pull.UP

btn2 = digitalio.DigitalInOut(board.D9)  # Marked A2
btn2.direction = digitalio.Direction.INPUT
btn2.pull = digitalio.Pull.UP

btn3 = digitalio.DigitalInOut(board.D3)  # Marked SCL A4
btn3.direction = digitalio.Direction.INPUT
btn3.pull = digitalio.Pull.UP

central = digitalio.DigitalInOut(board.D0)  # Marked RX A6
central.direction = digitalio.Direction.INPUT
central.pull = digitalio.Pull.UP

led = digitalio.DigitalInOut(board.D2)  # Marked SDA A5
led.switch_to_output()
led.value = False

buttons = [btn1, btn2, btn3]
upper = len(buttons) - 1
```

Bluetooth is now set up, and the NeoPixels are enabled.

```
ble_enabled = digitalio.DigitalInOut(board.SLIDE_SWITCH)
ble_enabled.direction = digitalio.Direction.INPUT
ble_enabled.pull = digitalio.Pull.UP

pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=0.5)
# Use default HID descriptor
hid = HIDService()
device_info = DeviceInfoService(
    software_revision=adafruit_ble.__version__, manufacturer="Adafruit Industries"
)
advertisement = ProvideServicesAdvertisement(hid)
advertisement.appearance = 961
scan_response = Advertisement()

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

if ble_enabled.value:
    print("advertising")
    ble.start_advertising(advertisement, scan_response)

k = Keyboard(hid.devices)
kl = KeyboardLayoutUS(k)
```

After that, the audio file is loaded.

```
wave_file = open("jeopardy.wav", "rb")
wave = WaveFile(wave_file)
audio = AudioOut(board.SPEAKER)
```

Now, the code enters the main loop. The first section of it connects to Bluetooth, if Bluetooth is enabled with the slide switch. When it connects, it flashes the LED on the central button to notify the user that it is connected.

```
while True:
    if ble_enabled.value:
        while not ble.connected:
            pass
        if ble.connected:
            print("Connected")
            led.value = True
            time.sleep(0.1)
            led.value = False
            time.sleep(0.1)
            led.value = True
            time.sleep(0.1)
            led.value = False
```

If it is connected, or if Bluetooth is disabled, the code now enters the section of the loop where it can poll the buttons to see which is pressed. It waits for a press from the central button, and if there is one, it turns that LED on and waits for the contestants to buzz in and only moves on to the next section when one has.

```
while ble.connected or not ble_enabled.value:
    if not central.value:
        led.value = True
        print("Running")
        while True:
            i = random.randint(0, upper)
            if not buttons[i].value:
                break
```

Then, the code will play the sound and set the NeoPixels to the color of the button that was pressed first. If BLE is enabled, it will send a string to the keyboard with the name of the button that got pressed.

After this, it waits for the central button to be pressed.

```
audio.play(wave)
if i == 0:
 print("Button 1")
 pixels.fill((0, 0, 255))
 if ble_enabled.value:
  kl.write("Button 1")
elif i == 1:
 print("Button 2")
 pixels.fill((0, 255, 0))
 if ble_enabled.value:
  kl.write("Button 2")
elif i == 2:
 print("Button 3")
 pixels.fill((255, 255, 255))
 if ble_enabled.value:
  kl.write("Button 3")

if not ble_enabled.value:
 print(
  "BLE HID has been disabled, slide the slide switch to the left to re-enable"
 )

print("Finished")
led.value = False

while central.value:
 pass
```

Now that the central button has been pressed, the code resets and goes over the inner loop again.

```
print("reset")
pixels.fill((0, 0, 0))
led.value = True
time.sleep(0.5)
led.value = False
print("Ready")
```

This section checks to see if BLE has been disconnected and breaks out of the loop if it has.

```
    if ble_enabled.value:
  if not ble.connected:
   break
else:
 continue
break
```