Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>• Parts</td>
<td></td>
</tr>
<tr>
<td>Build the BLE TV Zapper</td>
<td>6</td>
</tr>
<tr>
<td>• Case</td>
<td></td>
</tr>
<tr>
<td>• Cable</td>
<td></td>
</tr>
<tr>
<td>• Connections</td>
<td></td>
</tr>
<tr>
<td>• Battery</td>
<td></td>
</tr>
<tr>
<td>• Stack It</td>
<td></td>
</tr>
<tr>
<td>CircuitPython on Circuit Playground Bluefruit</td>
<td>11</td>
</tr>
<tr>
<td>• Install or Update CircuitPython</td>
<td></td>
</tr>
<tr>
<td>Code the BLE TV Zapper</td>
<td>14</td>
</tr>
<tr>
<td>• Text Editor</td>
<td></td>
</tr>
<tr>
<td>• Download the Project Bundle</td>
<td></td>
</tr>
<tr>
<td>• Use the TV Zapper</td>
<td></td>
</tr>
<tr>
<td>• Libraries</td>
<td></td>
</tr>
<tr>
<td>• IR Pin</td>
<td></td>
</tr>
<tr>
<td>• IR Code Send Function</td>
<td></td>
</tr>
<tr>
<td>• Main Loop</td>
<td></td>
</tr>
</tbody>
</table>
Overview

Turn off (or on) any television set via Bluetooth and IR light! This project combines the ever-popular IR blaster concept with an additional layer of remoteness -- Bluetooth Low Energy! Plant the device near your target and then safely trigger it with your phone from another location.

The Circuit Playground Bluefruit combined with the STEMMA High Power Infrared LED Emitter, running CircuitPython makes it simple to build this super sneaky device.

This project is a riff on the excellent Circuit Playground Express TV Zapper project, check out the guide here for the history of the TV-B-Gone as well as tons of details on the process of creating the IR remote code file for use with CircuitPython!
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

**Adafruit High Power Infrared LED Emitter - STEMMA JST PH 2mm**
*pew* *pew*! This board is like a little ray gun for infrared light, with two high powered LED outputs. When controlled with the onboard N-Channel FET driver,...
https://www.adafruit.com/product/5639

**JST PH 2mm 3-pin Plug to Color Coded Alligator Clips Cable**
This cable will let you turn a JST PH 3-pin cable port into 3 individual wires with grippy mini alligator clips. We're carrying these to match up with any of our boards or...
https://www.adafruit.com/product/4030
1 x Clear Adhesive Squares - 6 pack
Clear Adhesive Squares - 6 pack
https://www.adafruit.com/product/4813

1 x USB cable
USB A to Micro-B - 3 foot long
https://www.adafruit.com/product/592

Adafruit Circuit Playground Express or Bluefruit Enclosure
We've got nice cases for many of our beloved boards, but the Circuit Playground Express and
https://www.adafruit.com/product/3915

3 x AAA Battery Holder with On/Off Switch and 2-Pin JST
This battery holder connects 3 AAA batteries together in series for powering all kinds of projects. We spec'd these out because the box is slim, and 3 AAA's add up to about...
https://www.adafruit.com/product/727

Alkaline AAA batteries - 3 pack
Battery power for your portable project! These batteries are good quality at a good price, and work fantastic with any of the kits or projects in the shop that use AAA's. This is a...
https://www.adafruit.com/product/3520
Build the BLE TV Zapper

Assembly of the Bluefruit TV Zapper is plug-and-play, thanks to the 3-pin STEMMA port and alligator clip cable.

Case

The first step here is optional -- if you want to snap the Circuit Playground Bluefruit into the case, go ahead and do so now, before connecting the alligator clips.
Cable
Next, plug the JST 3-pin 2mm cable into the IR Emitter board.
Connections
Connect the alligator clips to the following pins on the Circuit Playground Bluefruit:

- black to GND
- red to 3.3V
- white to A2
Battery
If using AAA x3 battery pack, plug the battery pack's 2-pin JST connector into the power port on the Circuit Playground Bluefruit.

If using a LiPo battery, plug its 2-pin JST connector into the power port on the Circuit Playground Bluefruit.
Stack It

You can use the BLE TV Zapper as is -- which may be easier to hide in a drawer with just the IR LEDs peeking out -- or make the package a bit more compact with some adhesive squares.

Attach the battery to the case first, then attach the IR breakout as shown here.
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

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! :)

©Adafruit Industries
Code the BLE TV Zapper

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in [this guide](https://docs.adafruit.com/circuitpython/mu).

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

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries, .txt files, and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Plug the Circuit Playground into your computer via a known good data+power USB cable. Your board should show up in your File Explorer/Finder (depending on your Operating System) as a thumb drive named CIRCUITPY.

Drag the contents of the uncompressed bundle directory onto your board's CIRCUITPY drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

```bash
# SPDX-FileCopyrightText: 2018 Limor Fried for Adafruit Industries
# SPDX-License-Identifier: MIT
```
# Circuit Playground Bluefruit version 2022 John Park

import array
import time
import board
import pulseio
from digitalio import DigitalInOut, Direction, Pull
from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService
from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.button_packet import ButtonPacket

# pylint: disable=eval-used
# Switch to select 'stealth-mode'
switch = DigitalInOut(board.SLIDE_SWITCH)
switch.direction = Direction.INPUT
switch.pull = Pull.UP
# Button to see output debug
led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT
# which pin for IR LED/blaster
ir_pin = board.A2  # JST IR Blaster board
# Speaker as haptic feedback
spkr_en = DigitalInOut(board.SPEAKER_ENABLE)
spkr_en.direction = Direction.OUTPUT
spkr_en.value = True
spkr = DigitalInOut(board.SPEAKER)
spkr.direction = Direction.OUTPUT

# BLE setup
ble = BLERadio()
uart_service = UARTService()
advertisement = ProvideServicesAdvertisement(uart_service)
def ir_code_send(code):
f = open(code, "r")
for line in f:
    code = eval(line)
    print(code)
    if switch.value:
        led.value = True
    else:
        spkr.value = True
    # If this is a repeating code, extract details
    try:
        repeat = code["repeat"]
        delay = code["repeat_delay"]
    except KeyError:  # by default, repeat once only!
        repeat = 1
        delay = 0
    # The table holds the on/off pairs
    table = code["table"]
    pulses = []  # store the pulses here
    # Read through each indexed element
    for i in code["index"]:
        pulses += table[i]  # and add to the list of pulses
    pulses.pop()  # remove one final 'low' pulse

    with pulseio.PulseOut(ir_pin, frequency=code["freq"], duty_cycle=2**15)
as pulse:
    for i in range(repeat):
        pulse.send(array.array("H", pulses))
        time.sleep(delay)

led.value = False
spkr.value = False
time.sleep(code["delay"])

f.close()

while True:
    ble.name = "TVRemote"
    ble.start_advertising(advertisement)

    while not ble.connected:
        # Wait for a connection.
        if button_a.value or button_b.value:
            print("All codes")
            time.sleep(0.1)  # wait a moment
            ir_code_send("/full_codes.txt")

    while ble.connected:
        if button_a.value or button_b.value:
            print("all")
            time.sleep(0.1)  # wait a moment
            ir_code_send("/full_codes.txt")
        if uart_service.in_waiting:
            # Packet is arriving.
            packet = Packet.from_stream(uart_service)
            if isinstance(packet, ButtonPacket) and packet.pressed:
                if packet.button == ButtonPacket.UP:
                    print("Select codes")
                    time.sleep(0.1)  # wait a moment
                    ir_code_send("/codes.txt")
                elif packet.button == ButtonPacket.DOWN:
                    print("All codes")
                    time.sleep(0.1)  # wait a moment
                    ir_code_send("/full_codes.txt")
                elif packet.button == ButtonPacket.BUTTON_1:
                    print("Sony power")
                    time.sleep(0.1)  # wait a moment
                    ir_code_send("/sony_pwr.txt")
                elif packet.button == ButtonPacket.BUTTON_2:
                    print("Toshiba power")
                    time.sleep(0.1)  # wait a moment
                    ir_code_send("/toshiba_pwr.txt")

Use the TV Zapper

To use the Zapper, you can simply press either button on the Circuit Playground
Bluefruit and point the emitter at the target TVs. Once the proper code is sent, ZAP!
The TV will power off! (Or on, if it was already off).
To use it in BLE mode, connected from the Bluefruit mobile app, and use the controller buttons. See this guide () for step-by-step instructions on using the Bluefruit Connect app.

- Up sends the codes in codes.txt
- Down sends the codes defined in the file full_codes.txt
- Left sends the Sony power code.
- Right sends a Toshiba power code.

See the alert box below on how to make your own custom code assignments.

You can figure out which codes to use for particular TV sets by simply running the full_codes.txt function and counting how many tries it took to get to the TV you want. Then copied that line of code into its own text file, for example sony_pwr.txt in the project bundle.

This section of the main TV Zapper guide () shows how most of the code works in this project. The main differences here are the use of the IR breakout board, and the Bluetooth functionality.

Libraries

Additional libraries used are related to adafruit_ble and adafruit_bluefruit_connect:

```python
from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService
from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.button_packet import ButtonPacket
```

IR Pin

The IR emitter breakout is connected on pin A2:

```python
ir_pin = board.A2  # JST IR Blaster board
```

IR Code Send Function

The `ir_code_send()` function is used to open up the requested text file and then emit the relevant code pulses. It will be called by the button presses and Bluefruit buttons to run different code text files.
def ir_code_send(code):
    f = open(code, "r")
    for line in f:
        code = eval(line)
        print(code)
        if switch.value:
            led.value = True
        else:
            spkr.value = True
        # If this is a repeating code, extract details
        try:
            repeat = code["repeat"]
            delay = code["repeat_delay"]
        except KeyError:  # by default, repeat once only!
            repeat = 1
            delay = 0
        # The table holds the on/off pairs
        table = code["table"]
        pulses = []  # store the pulses here
        # Read through each indexed element
        for i in code["index"]:
            pulses += table[i]  # and add to the list of pulses
        pulses.pop()  # remove one final 'low' pulse

        with pulseio.PulseOut(
            ir_pin, frequency=code["freq"], duty_cycle=2**15
        ) as pulse:
            for i in range(repeat):
                pulse.send(array.array("H", pulses))
                time.sleep(delay)

        led.value = False
        spkr.value = False
        time.sleep(code["delay"])
    f.close()

Main Loop

Then main loop advertises the BLE and then waits for a connection. While it is waiting, you can still use the buttons to send all codes.

while True:
    ble.name = 'TVRemote'
    ble.start_advertising(advertisement)
    while not ble.connected:
        # Wait for a connection.
        if button_a.value or button_b.value:
            print("All codes")
            time.sleep(0.1)  # wait a moment
            ir_code_send("/full_codes.txt")
            pass

Once connected, the physical buttons on the Circuit Playground Bluefruit still work, but now the button packets from the Bluefruit app are also used to call different IR code send text files.
while ble.connected:
    if button_a.value or button_b.value:
        print("all")
        time.sleep(0.1)  # wait a moment
        ir_code_send("/full_codes.txt")
    if uart_service.in_waiting:
        # Packet is arriving.
        packet = Packet.from_stream(uart_service)
        if isinstance(packet, ButtonPacket) and packet.pressed:
            if packet.button == ButtonPacket.UP:
                print("Select codes")
                time.sleep(0.1)  # wait a moment
                ir_code_send("/codes.txt")
            if packet.button == ButtonPacket.DOWN:
                print("All codes")
                time.sleep(0.1)  # wait a moment
                ir_code_send("/full_codes.txt")
            elif packet.button == ButtonPacket.BUTTON_1:
                print("Sony power")
                time.sleep(0.1)  # wait a moment
                ir_code_send("/sony_pwr.txt")
            elif packet.button == ButtonPacket.BUTTON_2:
                print("Toshiba power")
                time.sleep(0.1)  # wait a moment
                ir_code_send("/toshiba_pwr.txt")