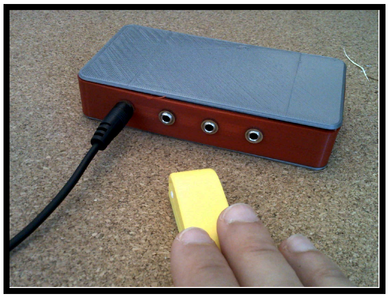


No Board BLE Switch Interface

A switch interface is a device users with disabilities utilize to connect their custom switches to in order to navigate computers and mobile devices. A switch activation sends a key command to the connected device. This build provides Bluetooth connectivity to allow for mobile device (iOS) connection given limitations in the direct USB connection. The project does not use a PCB board and can be put together using commercially available parts and 3D printed case.

** This build is geared towards the nontechnical/ novice user and uses the Circuitpython language and libraries. At the time of publishing (Aug 14, 2021), Circuitpython 6.x is the stable release version.

Materials List

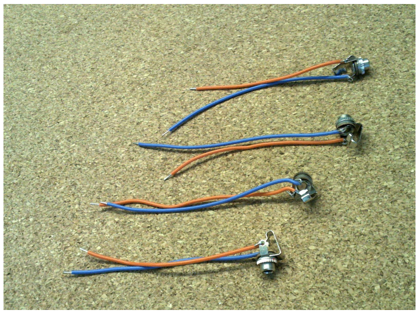
Qty	Item
1	Adafruit Feather nRF52840 Express
1	Lipo Battery
4	M2 4mm screws
4	3.5 mm Mono Jack
1	Micro B USB Cable (data)
1	3D Printed Case
1	Latching 12mm push button

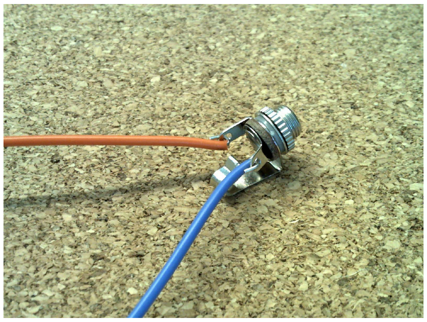
Equipment List

Soldering Iron and solder
Wire - solid thread if possible- 2 colors
Wirecutter and stripper
Needlenose pliers
Optional: heat gun and heatsink

Never soldered before? Check out this beginner-friendly how-to video from Adafruit.

Step #1: Connect Each Ground and Input Wire to Jack





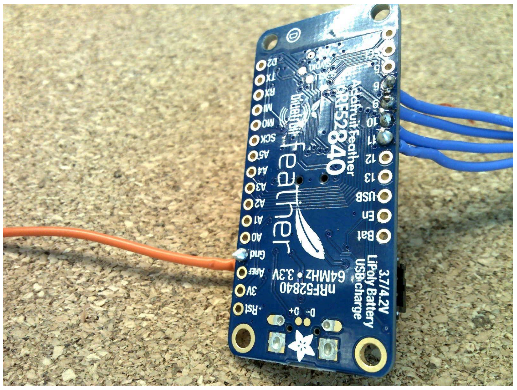
Cut (8) 70mm wires using two different colors if possible.

Hook one end and insert one of each color into the available holes on the mono jack.

Solder in place.
Repeat step for each mono jack.

Zero Day Camp (EIN: 81-4645962) is a Brooklyn based computer programming and robotics non-profit working to improve youth critical media literacy through direct services as well as development of instructional materials and tools. Learn more at https://zeroday.camp

Step #2: Solder the Ground and Mono jack wires to the board

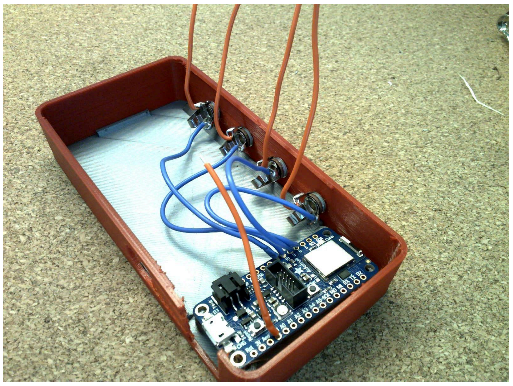


Cut a 40mm long wire and **solder** it to the Ground pin on the feather board.

Solder one of the wires from the mono jack into the board, **repeating** each mono jack's step.

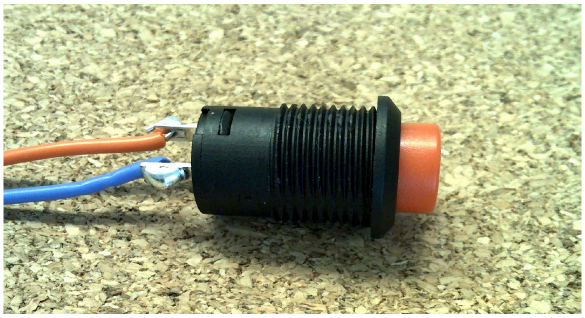
** Be sure to use the same color wire for the input pins and solder to pins `6 , 9, 10,` and `11`

Step #3: Insert the mono jacks into the case



Insert the mono jacks into the case and fasten them into place using the provided rings.

Step #4: Solder wires to the on/ off button

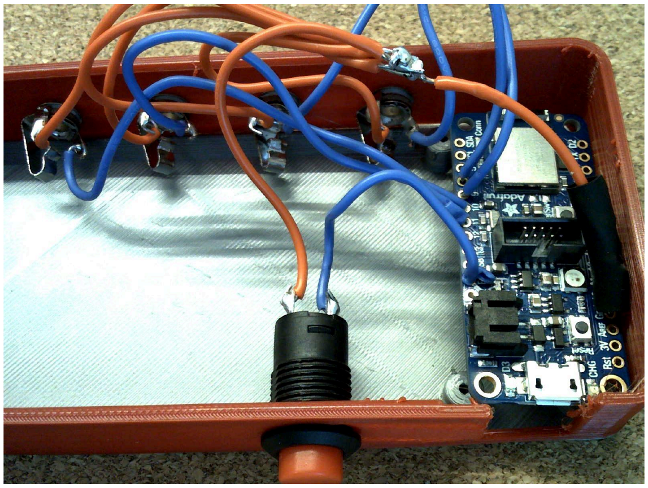


Cut 2 40mm wires using different colors if available.

Hook the ends using needlenose pliers and **insert** them into the available hooks on the power button.

Solder the wires into place.

Step #5: Insert the Power Button and Solder the Ground Wires



Insert the button into the available hole in the case and fasten it into place using the provided ring.

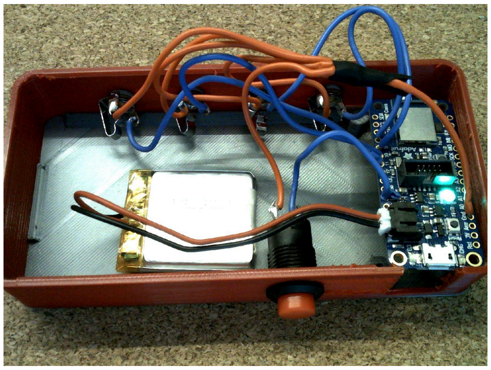
Solder one wire from the button to the board's `EN` pin.

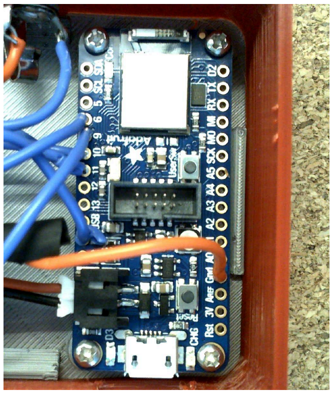
Solder the ground wires from the mono jacks and button to the board's `Ground` pin. There should be 5 ground wires.

** Note: if you plan on using a heatsink, insert the tubing prior to soldering.

Cover the joint with electrical tape if not using a heatsink.

Step #6: Insert the LiPo Battery and Fasten





Insert the LiPo battery into the holder.

Fasten the board into the available posts using the 2mm screws.

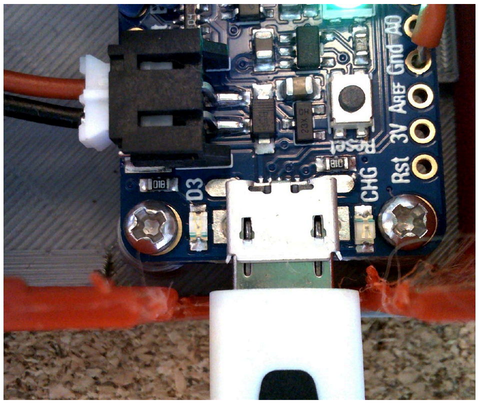
Step #7: Connect the Board and Flash Circuitpython

Download the needed project files onto your computer and unzip the folder if needed.

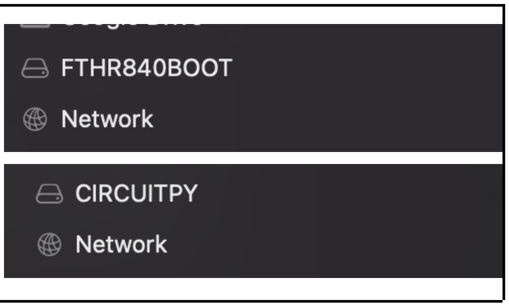
Connect the board to the computer using the USB cable (ensure the cable is data enabled and not charge only).

Double click the reset button on the board should now appear in Bootloader mode `'FTHR840BOOT'`.

Click and drag the `'adafruit-circuitpython-feather_nrf52840_express-en_US-6.3.0.uf2'` file into the drive.



Once flashing is complete, the drive will be labeled `'CIRCUITPY'`

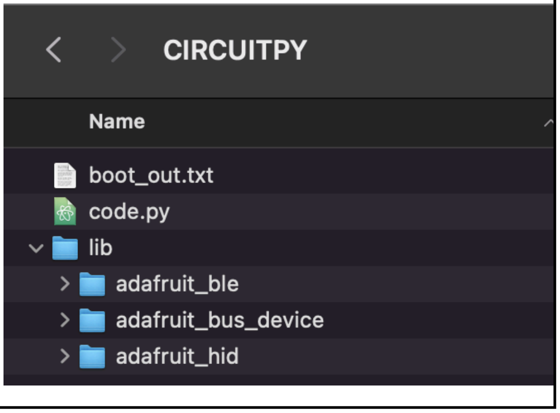


Step #8: Click and Drag the Code Files into the Drive

From the downloaded folder in step 7:

Click and drag the `'code.py'` file and `'lib'` folder into the `'CIRCUITPY'` folder.

You may override the existing files within the drive. Your drive should now have the following folders and file structure (see image)

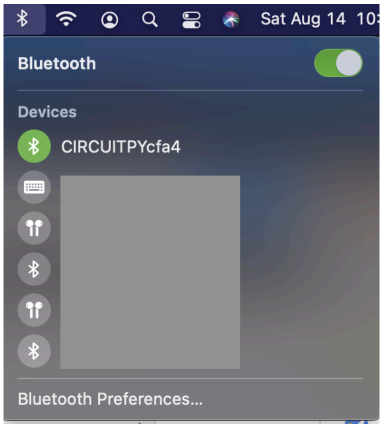


Step #9: Connect to the board via the device's Bluetooth menu and test

When powered on, the board should now be discoverable by your device's Bluetooth settings. The name of the device will begin with 'CIRCUITPY'.

Once the switch interface and device are connected, **insert** a desired switch to the mono jack port and test. By default, the following keystrokes are being sent:

O	O	O	O
Left arrow	Right arrow	Down arrow	Up arrow



Step #10: Customize the keycodes being sent

To customize the key codes being sent through the mono jack input, **you will need to open and modify** the `'code.py'` file. Your new code will automatically run after the file is saved.

If you already have a code editor of your choice, you may utilize it. However, if you are a beginner, **downloading and installing** the Mu Editor is a good first choice. The Adafruit website provides most of its Circuitpython tutorials using this editor.

** Note: You may be able to use a standard plain text editor such as Notepad or Text Edit however results are not guaranteed.

Code Walkthrough

Imports the needed core libraries.	`import time`
	`import board`
	`from digitalio import DigitalInOut, Direction, Pull`

Imports the Bluetooth and keyboard/ HID (Human Interface Device) libraries.	`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`
	`from adafruit_hid.keyboard import Keyboard`
	`from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS`
	`from adafruit_hid.keycode import Keycode`

Creates 4 buttons and lets them know which pin on the board they are connected to.	`button_1 = DigitalInOut(board.D11)`
	`button_2 = DigitalInOut(board.D10)`
	`button_3 = DigitalInOut(board.D9)`
	`button_4 = DigitalInOut(board.D6)`

Indicates that these pins are an input (button press).	`button_1.direction = Direction.INPUT`
	`button_2.direction = Direction.INPUT`
	`button_3.direction = Direction.INPUT`
	`button_4.direction = Direction.INPUT`

This indicates that the buttons are starting as an Up pull and are wired to the Ground pin. *(Down pull will be connected through the 3V pin.)*	`button_1.pull = Pull.UP`
	`button_2.pull = Pull.UP`
	`button_3.pull = Pull.UP`
	`button_4.pull = Pull.UP`

Begins a HID instance	`hid = HIDService()`

Starts advertising the board through a Bluetooth connection.	`device_info = DeviceInfoService(software_revision=adafruit_ble.__version__, manufacturer="Adafruit Industries")`
	`advertisement = ProvideServicesAdvertisement(hid)`
	`advertisement.appearance = 961`
	`scan_response = Advertisement()`
	`scan_response.complete_name = "CircuitPython HID"`
	
	`ble = adafruit_ble.BLERadio()`
	`if not ble.connected:`
	`    print("advertising")`

```
    ble.start_advertising(advertisement, scan_response)
else:
    print("already connected")
    print(ble.connections)
```

Creates a keyboard instance and indicates that it is a US layout.

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

This is the main loop in the program and will run from the top to the bottom repeating itself forever.

The program scans to see if any of the buttons has a change in pull (due to switch activation) and then sends the corresponding key press.

There is a 1 second delay inserted after the key press to prevent errors. This argument may also be modified to meet the user's needs as an example:

`time.sleep(.5)`

will increase the speed of key presses.

To modify the key press being sent, modify the argument for the desired button on the appropriate line. For example:

`k.send(Keycode.ENTER)`

Here is a list of available key codes

```
while True:
    while not ble.connected:
        pass
    print("Start typing:")

    while ble.connected:
        if not button_1.value:
            print("up")  # for debug in REPL
            k.send(Keycode.UP_ARROW)
            time.sleep(1)

        if not button_2.value:
            print("down")  # for debug in REPL
            k.send(Keycode.DOWN_ARROW)
            time.sleep(1)

        if not button_3.value:
            print("right")
            k.send(Keycode.RIGHT_ARROW)
            time.sleep(1)

        if not button_4.value:
            print("left")
            k.send(Keycode.LEFT_ARROW)
            time.sleep(1)

    ble.start_advertising(advertisement)
```

🚨 *At the time of this writing, if you would like to use the switch for MacOS and iOS accessibility switch access, you may not assign any letters or numbers. The OS will not recognize those keycodes as switches. You may however use function keys (tab, space, enter, etc.) and arrow keys.*

Next Steps:

- Integrate the mouse navigation library to use the inputs to navigate a mouse cursor. Adafruit's QTPY for Mouse Emulation Tutorial.
- Integrate the Game Pad navigation library to create a gaming controller.
- Integrate the Media Control library to play/pause media, control volume or navigate tracks.
- Connect additional pins and mono jacks.
- Access the onboard neopixle to provide visual feedback for button presses.

- Connect piezo or speaker to provide auditory feedback for button presses.