Three Button Foot Switch
Created by Ruiz Brothers

https://learn.adafruit.com/three-button-foot-switch

Last updated on 2023-09-20 03:34:05 PM EDT
# Table of Contents

**Overview**  
• Prerequisite Guides  
• Parts  

**Circuit Diagram**  

**CircuitPython**  
• CircuitPython Quickstart  

**Code**  
• Download Project Bundle  
• Upload Code, Libraries and Assets  
• USB HID Keyboard and Mouse  
• Modify Controls  

**3D Printing**  
• Parts List  
• Filaments  

**Assemble**  
• Assemble switch brackets  
• Mount switch brackets  
• Solder board  
• Mount board  
• Attach lid  
• Complete!
Overview

In this project we’re making a USB foot switch with CircuitPython!

We were inspired to make a new foot switch which could be really nice for young folks and even animals.

For tasks that require both hands, we think using feet can help complete tasks much quicker!

We designed and 3d printed a three button foot pedal that snap fits together.

This project can be powered by either an Adafruit QT Py RP2040 or the KB2040.

CircuitPython makes USB HID project really easy to put together.

You can customize the key codes and set up different keyboard shortcuts or media controls like play/pause and volume.
You can 3D print the parts without any support material using your favorite filament.

The bottom features mounting holes to use M3 hardware to secure the switches and brackets.

The case is compatible with the QTPy or KB2040 board. We used the QTPy in this build.

Boards press fit into a mount that can then attach to the case.

Top covers can then be installed to the built-in hinges on the base. There's nubbin's on the sides that are press fitted into the dimples on the hinges.

The micro-switch features a bump actuator so it works nicely without the need of an additional spring!
Prerequisite Guides

If you’re new to electronics and soldering, I suggest walking through the following guides to get the basics. The Adafruit Excellent guide to soldering will walk you through the process of learning how to use a soldering iron to make solid electrical connections.

- Lemon KeyPad
- Single Button Foot Switch
- Adafruit Guide to Excellent Soldering
- Welcome to CircuitPython

![Adafruit QT Py RP2040](https://www.adafruit.com/product/4900)

Parts

Adafruit QT Py RP2040
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with one of our new favorite chips, the RP2040. It’s been made famous in the new [https://www.adafruit.com/product/4900](https://www.adafruit.com/product/4900)
Micro Switch - Premium Zippy 3-Terminal
Micro-switches are often found in arcade buttons and joysticks but they're also really handy in any kind of mechatronics project or when you need a basic sensor. They are always...
https://www.adafruit.com/product/817

Arcade Button and Switch Quick-Connect Wires - 0.187" (10-pack)
Quick connector wire sets make wiring up our arcade-style or metal buttons quicky-quick. Each wire comes as a 'pair' with two 0.187" quick-connects pre-crimped. The wires...
https://www.adafruit.com/product/3835

Circuit Diagram
This provides a visual reference for wiring of the components. They aren't true to scale, just meant to be used as reference. Generic micro switches follow a standard pin out with visible markings on the body of the switch. Here's each label markings:

- (C1) – "common ground".
- (NO2) – "normally open".
- (NC3) – "normally closed".
(C 1) Common from micro switch to Ground on QTPy

(NO2) Normally open from micro switch to #0 on on QTPy

---

CircuitPython

CircuitPython is a derivative of MicroPython designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CircuitPython drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

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

Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.
Plug your board into your computer, using a known-good data-sync cable, directly, or via an adapter if needed.

Click the reset button once (highlighted in red above), and then click it again when you see the RGB status LED(s) (highlighted in green above) turn purple (approximately half a second later). Sometimes it helps to think of it as a "slow double-click" of the reset button.

On some very old versions of the UF2 bootloader, the status LED turns red instead of purple.

For this board, tap reset and wait for the LED to turn purple, and as soon as it turns purple, tap reset again. The second tap needs to happen while the LED is still purple.

Once successful, you will see the RGB status LED(s) turn green (highlighted in green above). If you see red, try another port, or if you’re using an adapter or hub, try without the hub, or different adapter or hub.

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

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
If after several tries, and verifying your USB cable is data-ready, you still cannot get to the bootloader, it is possible that the bootloader is missing or damaged. Check out the Factory Reset page for details on resolving this issue.

You will see a new disk drive appear called QTPYS2BOOT.

Drag the adafruit_circuitpython_etc.uf2 file to QTPYS2BOOT.

The BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it!

Code
Download Project Bundle

Once you've installed the latest version of CircuitPython onto your board, you'll need to grab the code, libraries and any assets included with the project.

Click the download project bundle button below to get the code, libraries and assets all in one!

```python
# SPDX-FileCopyrightText: 2022 Ruiz Bros for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import digitalio
import board
import usb_hid
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode

# The button pins we'll use, each will have an internal pullup
buttonpins = [board.A1, board.A2, board.A3]

# The keycode sent for each button, will be paired with a control key
buttonkeys = [
    ConsumerControlCode.PLAY_PAUSE,  # Center Foot Pad
    ConsumerControlCode.VOLUME_DECREMENT,  # Left Foot Pad
    ConsumerControlCode.VOLUME_INCREMENT,  # Right Foot Pad
]

# the keyboard object!
cc = ConsumerControl(usb_hid.devices)
# our array of button objects
buttons = []

# make all pin objects, make them inputs w/pullups
for pin in buttonpins:
    button = digitalio.DigitalInOut(pin)
    button.direction = digitalio.Direction.INPUT
    button.pull = digitalio.Pull.UP
    buttons.append(button)

print("Waiting for button presses")

while True:
    # check each button
    for button in buttons:
        if not button.value:  # pressed?
            i = buttons.index(button)

            print("Button #%d Pressed" % i)

            while not button.value:
                pass  # wait for it to be released!

            # type the keycode!
            k = buttonkeys[i]  # get the corresp. keycode
            cc.send(k)

    time.sleep(0.01)
```
Upload Code, Libraries and Assets

Unzip the project bundle and upload the files to the CIRCUITPY drive.

Your CIRCUITPY drive should look like this after you've uploaded the code, libraries and assets.

![CIRCUITPY Drive Content](image)

USB HID Keyboard and Mouse

Check out the guides linked below for more information on using the USB-HID library for CircuitPython.

- **HID Keyboard Mouse Guide**
- **ReadTheDocs – HID Library**

Modify Controls

The three switches are assigned the consumer controls for play/pause volume up, and volume down. Adjust the following code to change the keys control or the order of the arrangement. The `buttonpins` and `buttonkeys` are ordered chronologically.

```python
# The button pins we'll use, each will have an internal pullup
buttonpins = [board.A1, board.A2, board.A3]

# The keycode sent for each button, will be paired with a control key
buttonkeys = [
    ConsumerControlCode.PLAY_PAUSE,
    ConsumerControlCode.VOLUME_INCREMENT,
    ConsumerControlCode.VOLUME_DECREMENT
]
```
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.

Edit Design Files
Download STLs

Slice with settings for PLA material.
The parts were sliced using CURA using the slice settings below.
PLA filament 220c extruder
0.2 layer height
10% gyroid infill
60mm/s print speed
60c heated bed

Filaments
The filamented used in this project.

- Evryone PLA Sparkly Glitter ()
  - Purple
  - Black
  - Red
Assemble switch brackets

M3 x 16mm long screws are used to attach the micro switches to the printed brackets.
Mount switch brackets

Brackets are mounted to the base with M3 x 5mm long screws.

Connect the arcade quick connect wires before mounting to the base.
Solder board

Wires are soldered to the pins on the micro-controller.

The board is inserted into the PCB mount at an angle with the corners fitting under these clips.

The mount is then secured to the main base using hardware screws with all of the connections made.
Mount board

The board mount attaches to the base with two M3x5mm long screws.

Panel Mount 1/8" / 3.5mm TRS Audio Jack Connector fit next to the USB port on the case.
Attach lid

The top covers can then be installed to the built-in hinges on the base.

There's a nubbin on the sides that are press fitted into the dimples on the hinges.

The micro-switch features a bump actuator so it works nicely without the need of an additional spring.

Complete!

Additional rubber feet make it grip to any hard surface and they just stick to the bottom of the case.

This project uses code from the Lemon Keypad: https://learn.adafruit.com/qtpy-lemon-mechanical-keypad-macropad/code

We hope this inspires you to try out CircuitPython for your next USB controller project.