Neocontroller Color Grading Input Box

Created by John Park

https://learn.adafruit.com/neocontroller-color-grading-input-box

Last updated on 2022-02-21 11:18:37 AM EST
Table of Contents

Overview 3

• Parts 4

Install CircuitPython 7

• CircuitPython Quickstart 7
• Safe Mode 9
• Flash Resetting UF2 11

Code the Neocontroller 11

• Text Editor 11
• Download the Project Bundle 11
• Customize Settings 13

Build and Use the Neocontroller 14

• Set Addresses 14
• NeoKey Prep 15
• Connect the Boards 16
• Cut the Enclosure 17
• Connect the Base 17
• NeoKey Mounting 19
• Mount the Feather 20
• Top Plate 20
• Caps 21
• Feet 21
• Use the Neocontroller 22
Overview

Build this sliders-and-keyswitches input device to speed up your photo color grading workflow!

Adobe Lightroom Classic can be controlled via MIDI messages, giving you quick access to essential processing controls. It's a snap to adjust fundamental photo developing parameters such as exposure, color temperature, contrast, and saturation with the handy sliders. Use the keys to swap photos, and pull up the crop and heal tools.

Or, use it for any MIDI-capable application or host device, such as synthesizers, mixers, and more.

This project can be expanded to use more buttons and sliders by adding additional STEMMA QT boards with unique addresses, then making adjustments to the code.
Parts

**Black Adafruit Feather RP2040**
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather..."
https://www.adafruit.com/product/4884

**4ea. NeoSlider**

**Adafruit NeoSlider I2C QT Slide Potentiometer with 4 NeoPixels**
Our family of I2C-friendly user interface elements grows by one with this new product that makes it plug-n-play-easy to add a 75mm long slide potentiometer to any microcontroller or...
https://www.adafruit.com/product/5295
NeoKey 1x4 QT I2C - Four Mechanical Key Switches with NeoPixels
The only thing better than a nice mechanical key is, perhaps, FOUR mechanical keys that also can glow any color of the rainbow - and that's what the Adafruit...
https://www.adafruit.com/product/4980

4ea. Keycaps

Black Windowed Lamp R4 Keycap for MX Compatible Switches
These stark and elegant Black Windowed Lamp R4 Keycaps are made of opaque black plastic with a small cutout in the south-center edge that wraps around the top and...
https://www.adafruit.com/product/5112

4 x STEMMA QT / Qwiic JST SH cable
50mm
https://www.adafruit.com/product/4399

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long
This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...
https://www.adafruit.com/product/4210
Black Nylon Machine Screw and Stand-off Set – M2.5 Thread
Totaling 380 pieces, this M2.5 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel your maker...
https://www.adafruit.com/product/3299

Black Nylon Machine Screw and Stand-off Set – M3 Thread
Totaling 420 pieces, this M3 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel...
https://www.adafruit.com/product/4685

Black LED Diffusion Acrylic Panel 12" x 12"
- 0.1" / 2.6mm thick
A nice whoppin' slab of some lovely black acrylic to add some extra diffusion to your LED Matrix project. This material is 2.6mm (0.1") thick and is made of special cast...
https://www.adafruit.com/product/4594

Little Rubber Bumper Feet - Pack of 4
Keep your electronics from going barefoot, give them little rubber feet! These small sticky bumpers are our favorite accessory for any electronic kit or device. They are sticky, but...
https://www.adafruit.com/product/550
Install CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) 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 CIRCUITPY 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

https://adafru.it/R1D

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

Save it wherever is convenient for you.
To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

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.

You will see a new disk drive appear called RPI-RP2.

Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.
The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.

Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

Entering Safe Mode in CircuitPython 6.x

This section explains entering safe mode on CircuitPython 6.x.
To enter safe mode when using CircuitPython 6.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 700ms. On some boards, the onboard status LED (highlighted in green above) will turn solid yellow during this time. If you press reset during that 700ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

**Entering Safe Mode in CircuitPython 7.x**

To enter safe mode when using CircuitPython 7.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

**In Safe Mode**

Once you've entered safe mode successfully in CircuitPython 6.x, the LED will pulse yellow.

If you successfully enter safe mode on CircuitPython 7.x, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.
Auto-reload is off.
Running in safe mode! Not running saved code.

CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.

Press any key to enter the REPL. Use CTRL-D to reload.

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

**Flash Resetting UF2**

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

[Download flash erasing "nuke" UF2](https://adafru.it/RLE)

---

**Code the Neocontroller**

**Text Editor**

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in [this guide](https://adafru.it/ANO).

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 and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.
Drag the contents of the uncompressed bundle directory onto your Feather board’s CI
RCUITPY drive, replacing any existing files or directories with the same names, and
adding any new ones that are necessary.

```python
# SPDX-FileCopyrightText: 2021 john park for Adafruit Industries
# SPDX-License-Identifier: MIT
# NeoController: NeoSlider(x4) + NeoKey MIDI input device

import board
import busio
import adafruit_simplemath import map_range
import usb_midi
import adafruit_midi
from adafruit_midi.control_change import ControlChange
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff
from adafruit_seesaw.seesaw import Seesaw
from adafruit_seesaw.analoginput import AnalogInput
from adafruit_seesaw.neokey import NeoKey1x4

i2c = busio.I2C(board.SCL, board.SDA) # for QT Py RP2040, use SCL1/SDA1

# --- NeoSlider object setup
addresses = (0x30, 0x31, 0x32, 0x33) # cut jumpers to proper addresses (none, A0, A1, A0+A1)
neosliders = [] # list to hold neoslider objects
for address in addresses: # create each neoslider w proper address
    temp_neosliders = Seesaw(i2c, address)
    neosliders.append(temp_neosliders) # append to the list

# --- NeoSlider analog read setup
analogin_pin = 18 # slider is on this analog pin
analog_pins = (0, 1)
analog_inputs = []
for n in range(len(neosliders)):
    temp_analog_in = AnalogInput(neosliders[n], analogin_pin)
    analog_inputs.append(temp_analog_in)

# --- Slider NeoPixels Setup
PIX_PIN = 14
PIX_NUM = 4
pixels = []
for p in range(len(neosliders)):
    temp_pix = neopixel.NeoPixel(neosliders[p], PIX_PIN, PIX_NUM)
    pixels.append(temp_pix)
    pixels[p].brightness = 1.0
    pixels[p].fill((20, 20, 0))
    pixels[p].show()

# --- NeoKey 1x4 Setup --- #
neokey = NeoKey1x4(i2c, addr=0x38)
amber = 0x300800
blue = 0x002040
red = 0x900000
neokey.pixels.fill(amber)

keys = [] # neokey (adjust if multiple sets), key number, keypress color, MIDI note
(neokey, 0, blue, 40),
(neokey, 1, red, 42),
(neokey, 2, blue, 43),
(neokey, 3, red, 46),

# states for key presses
key_states = [False, False, False, False]
```

©Adafruit Industries
# --- MIDI setup
midi = adafruit_midi.MIDI(
midi_in=usb_midi.ports[0],
in_channel=0,
midi_out=usb_midi.ports[1],
out_channel=0
)
cc_numbers = [48, 49, 50, 51]  # pick the CC number for each slider to send over
last_sliders = [0] * len(neosliders)  # list to hold state variables

while True:
    # check NeoKeys
    for k in range(len(keys)):
        neokey, key_number, color, midi_note = keys[k]
        if neokey[key_number] and not key_states[key_number]:
            midi.send(NoteOn(midi_note, 127))
            key_states[key_number] = True
            neokey.pixels[key_number] = color
        if not neokey[key_number] and key_states[key_number]:
            midi.send(NoteOff(midi_note, 0))
            neokey.pixels[key_number] = amber
            key_states[key_number] = False

    # check sliders
    for i in range(len(neosliders)):
        slider = analog_inputs[i].value
        if abs(slider - last_sliders[i]) > 1:
            cc_val = int(map_range(slider, 0, 1023, 0, 127))
            midi.send(ControlChange(cc_numbers[i], cc_val))
            color_val = int(map_range(slider, 0, 1023, 5, 255))
            pixels[i].fill((color_val, color_val, 0))
            last_sliders[i] = slider

Customize Settings

The code allows you to customize the MIDI note values and CC numbers. To change the notes the keyswitches send, change the fourth value of the items in this list:

```python
keys = [
    # neokey (adjust if multiple sets), key number, keypress color, MIDI note
    (neokey, 0, blue, 40),
    (neokey, 1, red, 42),
    (neokey, 2, blue, 43),
    (neokey, 3, red, 46),
]
```

In order to send the **0-127** values over different CC numbers when you move the NeoSliders, adjust this list:

```python
cc_numbers = [48, 49, 50, 51]  # pick the CC number for each slider to send over
```
Build and Use the Neocontroller

Set Addresses
To use four NeoSliders on one I2C bus, each board needs a unique address. You will cut the traces on three of the boards to set them.

Set them to:

• 0x30 (default, don’t cut any traces)
• 0x31 -- cut trace A0
• 0x32 -- cut trace A1
• 0x33 -- cut traces A0 and A1

More info is available here (https://adafruit.it/XvD).

The NeoKey 1x4 has a default I2C address of 0x30 so to prevent a conflict, you can solder the A3 jumper closed, changing it to address 0x38 as shown here (https://adafruit.it/YfD).

NeoKey_1x4_plate.stl
https://adafruit.it/XvE
NeoKey Prep

Insert the keyswitches through the keyplate (3D printing file linked above), minding the orientation so they’ll fit into the hot swap sockets. Position them so the "window" in the switch is above the reverse-mount NeoPixels.

Then, carefully align the legs with the NeoKey board and insert them all at once, squeezing with even pressure.
Connect the Boards
Connect the Feather, NeoKey, and NeoSliders as shown using the STEMMA QT cables.

neocontroller.svg
https://adafruit.it/XvF
Cut the Enclosure
Use the linked .svg file to cut the case parts from acrylic. You can do this on a laser cutter or CNC machine, by hand with a saw and drill (print the .svg as a template), or send the .svg file to a service bureau for cutting.

Connect the Base
Use the M2.5 screws and nuts to secure the NeoSliders to the case bottom as shown.
NeoKey Mounting
Mount the NeoKey onto the screws as shown, by threading the nuts on first, and then sliding the board and keyplate holes over the screws. This places the board at a good height and it will stay put in use, no need for longer screws or top nuts.
Mount the Feather
Use short screws and nuts to mount the feather board as shown, with the USB port facing the back of the board, over the notched cutout.

Top Plate
Mount the M3 standoffs from the bottom with short M3 screws.
Place the top plate over the controls, then secure with four more screws.
Caps

Place the four fader caps and keycaps onto their respective controls.

Feet

Stick four adhesive bump-on feet under the enclosure to prevent sliding.

Plug in USB and you're ready to go!
Use the Neocontroller

Plug in the Neocontroller to your computer over USB-A to USB-C data/power cable. For typical MIDI use, simply select the Feather RP2040 MIDI device. To use with Lightroom Classic, first download and install the free MIDI2LR plug-in as shown here (https://adafru.it/Xwa).

There are plenty of good setup and configuration videos out there, such as this. (https://adafru.it/Xwb)
Here it is in action: