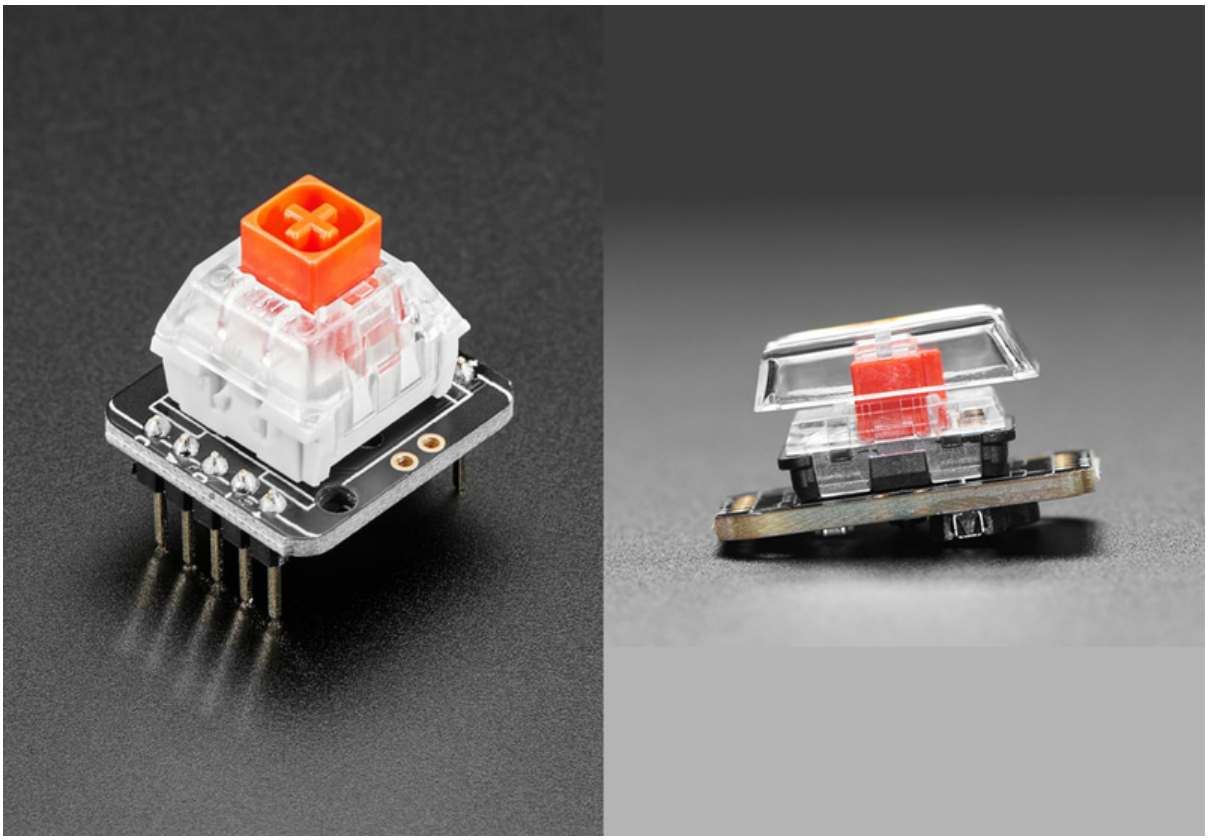
NeoKey Socket Breakout with NeoPixel for MX and CHOC Key Switches

Created by Kattni Rembor



https://learn.adafruit.com/neokey-breakout

Last updated on 2024-03-08 04:14:23 PM EST

Table of Contents

Overview 3

Pinouts 7

- Key Switch Pins
- NeoPixel Power Pins
- NeoPixel Data Pins

CircuitPython & Python 8

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of NeoPixel Library
- CircuitPython Usage
- Python Usage
- Key Press NeoPixel Random Color Demo
- Customisations
- Set Up
- Loop
- Key Press NeoPixel Rainbow Demo
- Customisations
- Set Up
- Loop
- HID Keyboard Demo
- Customisations
- Set Up
- Loop

CircuitPython keypad Docs 21

CircuitPython NeoPixel Docs 21

Arduino 21

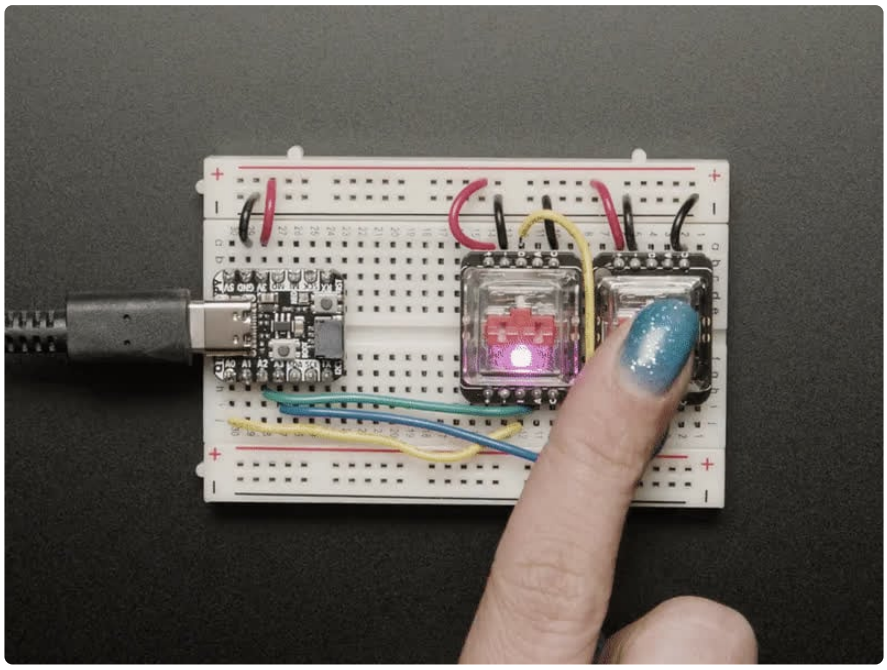
- Wiring
- Library Installation
- Example Code

Arduino NeoPixel Docs 23

Downloads 24

- Files
- MX Breakout Schematic and Fab Print
- CHOC Breakout Schematic and Fab Print

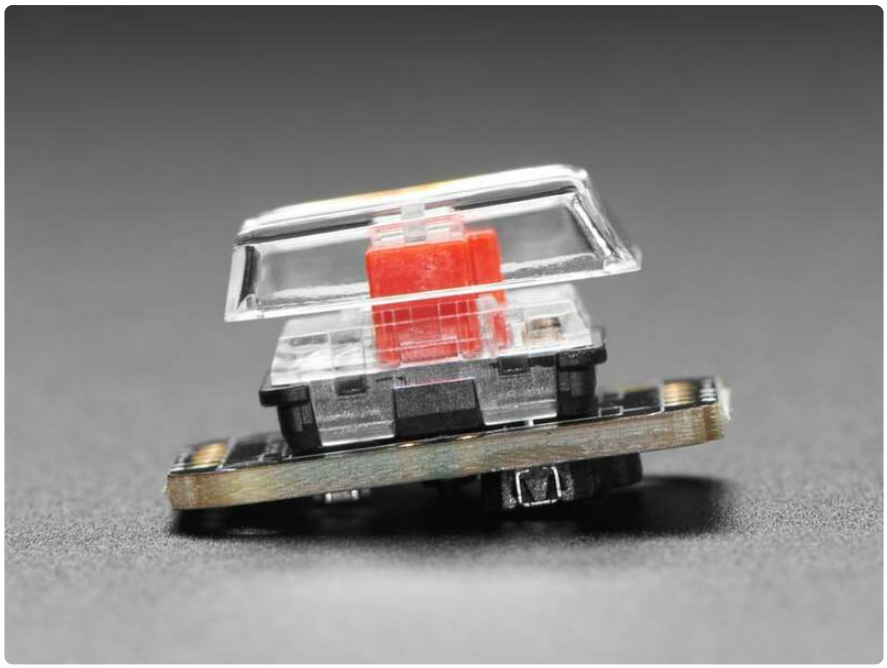
Overview



The only thing better than a nice mechanical key, is one that also can glow any color of the rainbow - and that's what the **Adafruit NeoKey Breakouts** will let you do! They make it super simple to use a mechanical keyswitch with a breadboard or perf board.

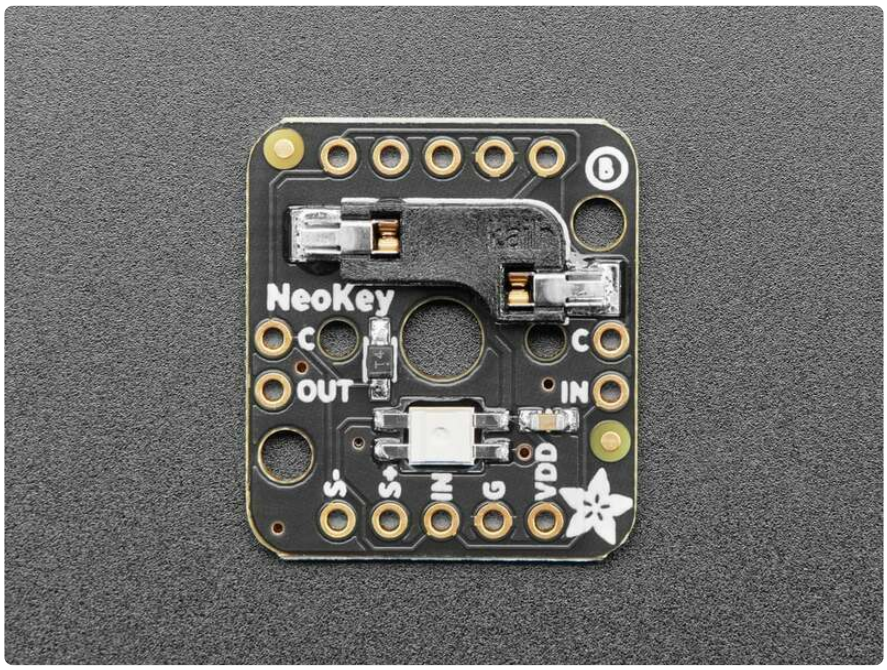
The **0.75" x 0.85**" breakout can fit one **Cherry MX or compatible switch**.

The **0.75" x 0.95"** breakout can fit one **Kailh CHOC or compatible switch**.



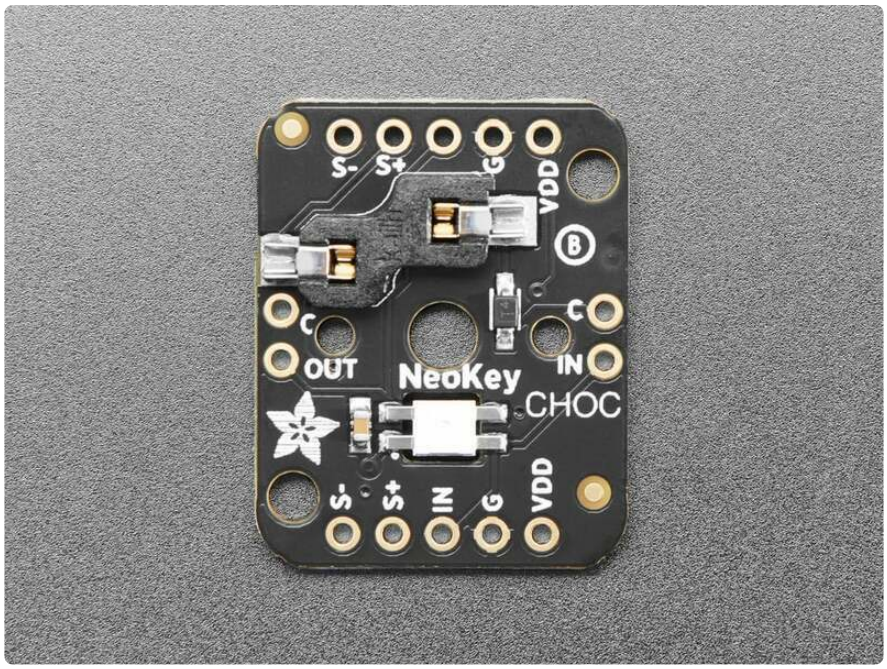
The MX and CHOC breakout PCBs are not the same size! The CHOC breakout is 0.1" (one breadboard row) taller.

This breakout has a **Kailh MX-compatible socket**, which means you can plug in any **MX-compatible switch** instead of soldering it in.



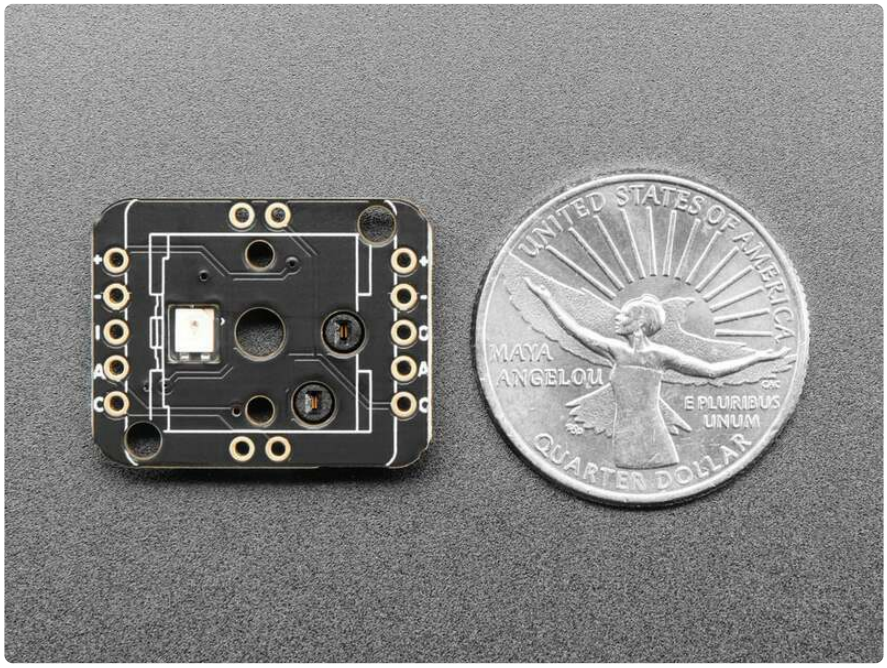
CHOC keyswitches are not compatible with MX sockets!

This breakout has a **Kailh CHOC socket**, which means you can plug in any **CHOC-compatible switch** instead of soldering it in.

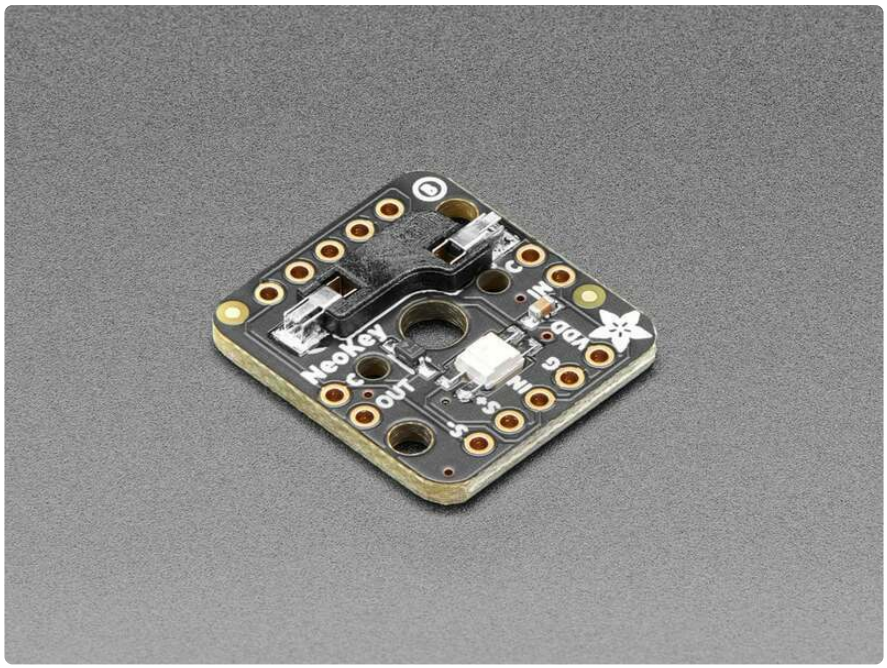


MX keyswitches are not compatible with CHOC sockets!

You may need a little glue to keep the switch in place; hot glue or a dot of epoxy worked fine for us. We also place a 1N4148 signal diode in series with the switch, so you can create key-grid matrices without worrying about ghosted keys.

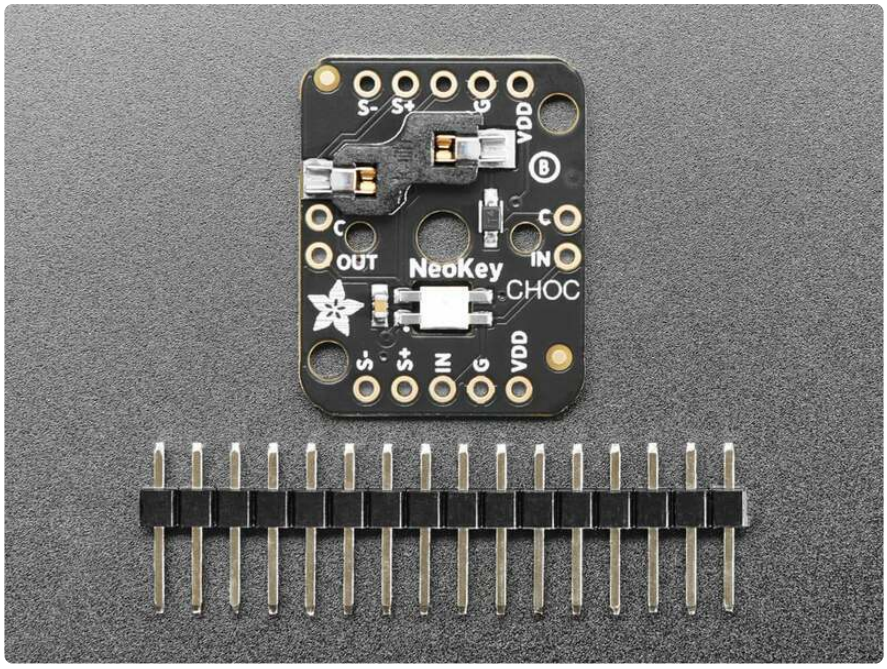


Each breakout also has a single reverse-mount NeoPixel pointing up through the spot where many switches would have an LED to shine through. The input and output of the pixel are broken out so you can 'chain' these boards together and control them as one NeoPixel strand.



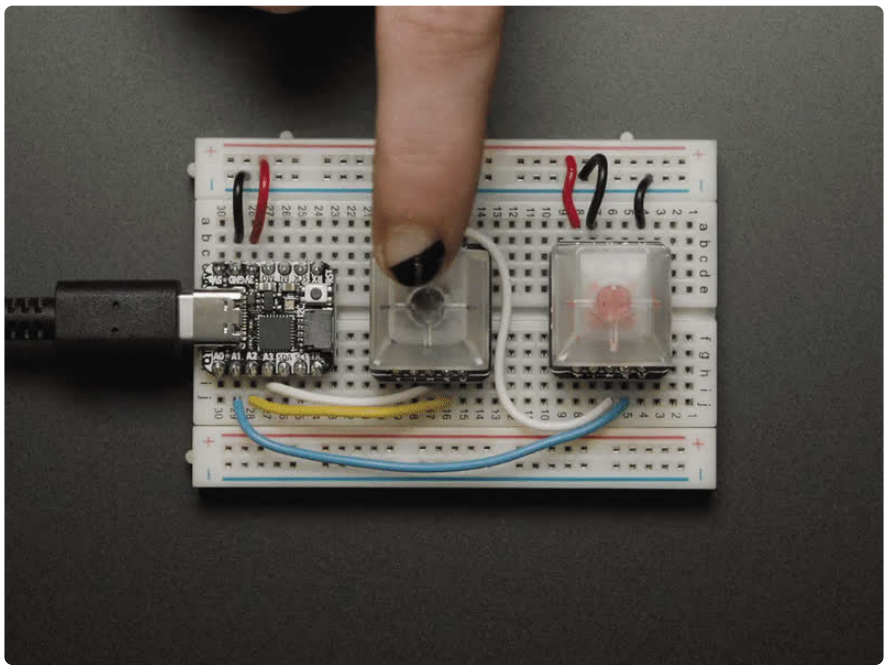
There's plenty of flexibility for any kind of use; each board has the following pin out:

- **VDD (+)** The power pin for the NeoPixel; provide 3 to 5VDC
- **GND (-)** The ground power pin for the NeoPixel, connect to ground
- **In (I)** and **Out (O)** The input and output to/from the NeoPixel, for chaining
- **Switch Anode (A**) - The 'positive' side of the switch+diode. If you're using the switch with a pull-up resistor, connect this pin to your microcontroller. If you're using a pull-down, connect this pin to your logic level power pin.
- **Switch Cathode (C)** - The 'negative' side of the switch+diode. If you're using the switch with a pull-up resistor, connect this pin to ground. If you're using a pull-down, connect this pin to your microcontroller.

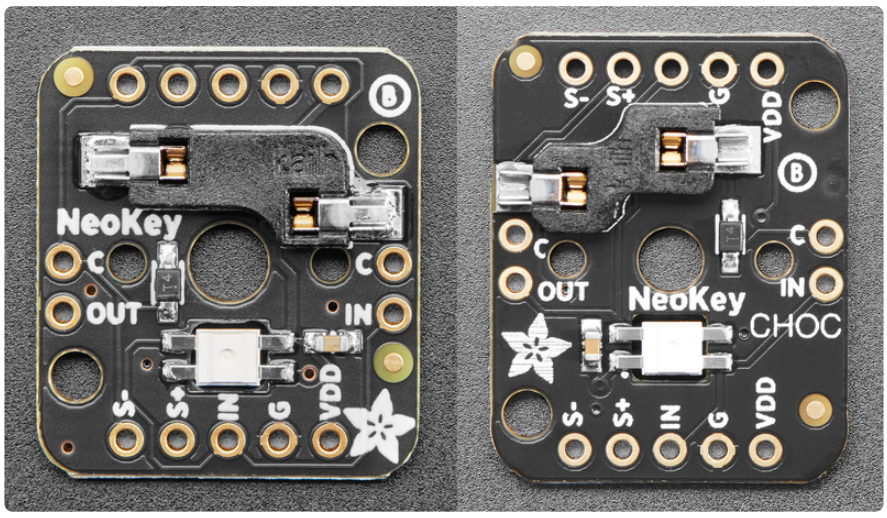


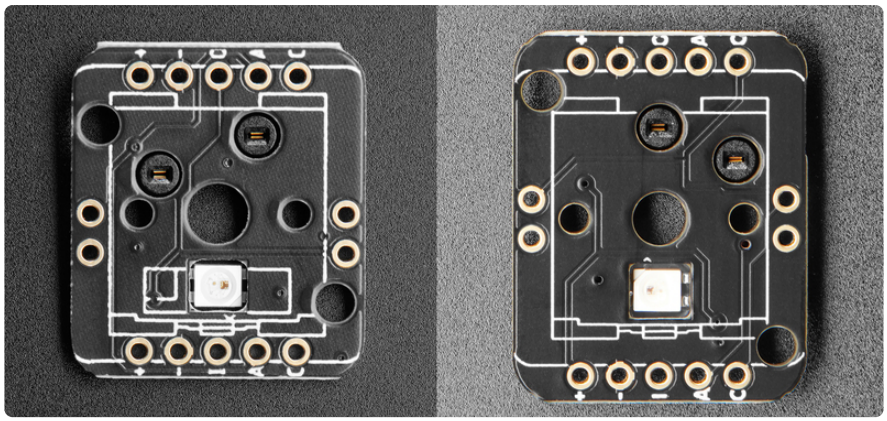
There're two rows of 5-pin contacts on a 0.1" grid on both sides. Solder in both sides for mechanical stability; the two sides share the same pins except for one side is NeoPixel in and the opposite side has the out pin. For side-by-side wiring, we also have the cathode pin broken out on the sides, and another set of NeoPixel In/Out pins.

Please note, each order comes with one assembled PCB and a small stick of break-off header. Soldering is required to attach the header for breadboard use. A mechanical switch and key cap are not included!



Pinouts





Key Switch Pins

- **S+ (A)** - This is the switch anode, the 'positive' side of the switch+diode. If you're using the switch with a pull-up resistor, connect this pin to your microcontroller. If you're using a pull-down, connect this pin to your logic level power pin.
- **S- (C)** - This is the switch cathode, the 'negative' side of the switch+diode. If you're using the switch with a pull-up resistor, connect this pin to ground. If you're using a pull-down, connect this pin to your microcontroller.

NeoPixel Power Pins

- **VDD** - This is the power pin for the NeoPixel. Connect it to 3VDC-5VDC power.
- **GND** - This is the ground pin for the NeoPixel. Connect it to ground.

NeoPixel Data Pins

- **IN (I)** - This is the data-in pin for the NeoPixel. Connect this to a microcontroller GPIO pin.
- **OUT (O)** - This is the data-out pin for the NeoPixel. If you are connecting more than one breakout together, you would connect this to the **IN** pin on the next breakout in the sequence.

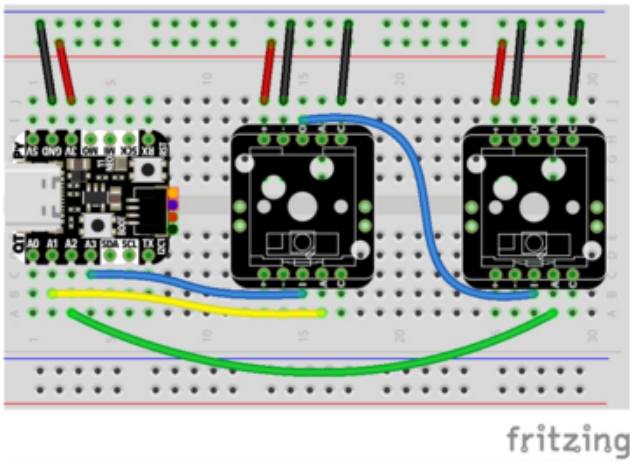
CircuitPython & Python

It's easy to use your NeoKey socket breakouts with CircuitPython and the built-in `keypad` module. You can use the key switches to control the NeoPixel LEDs using the Adafruit CircuitPython NeoPixel (https://adafru.it/yew) library. You can create a tiny keyboard using the Adafruit CircuitPython HID (https://adafru.it/xid) library. This page covers how to do both!

The wiring is identical for the MX and CHOC NeoKey breakouts. The difference in breakout size has no effect on how to wire it up.

CircuitPython Microcontroller Wiring

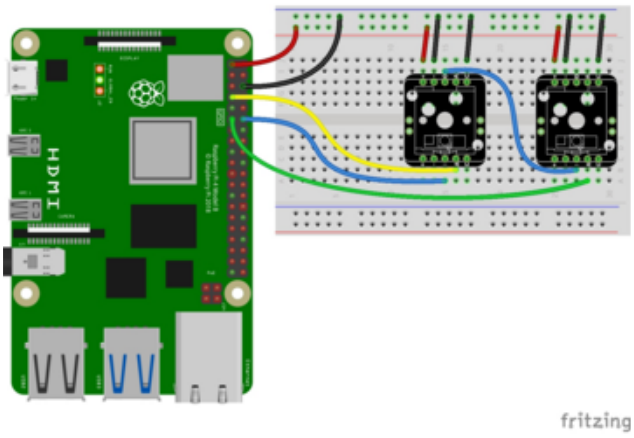
The first thing you need to do is wire up your NeoKey socket breakouts. The diagram below uses the MX version, but the wiring is identical for the CHOC version.



QT Py GND to **ground rail on breadboard**
QT Py 3V to **power rail on breadboard**
Both NeoKey breakouts S+ to **power rail on breadboard**
Both NeoKey breakouts S- to **ground rail on breadboard**
Both NeoKey breakouts C to **ground rail on breadboard**
QT Py A1 to **left breakout A**
QT Py A2 to **right breakout A**
QT Py A3 to **left breakout I**
Left breakout O to **right breakout I**

Python Computer Wiring

If you're using an SBC other than the Raspberry Pi, you may not have NeoPixel support. To fully run the demos on this page, you'll want to use a Pi.



Pi 3.3V to **power rail on breadboard**
Pi GND to **ground rail on breadboard**
Both NeoKey breakouts S+ to **power rail on breadboard**
Both NeoKey breakouts S- to **ground rail on breadboard**
Both NeoKey breakouts C to **ground rail on breadboard**
Pi GPIO4 to **left breakout A**
Pi GPIO17 to **right breakout A**
Pi GPIO18 to **left breakout I**
Left breakout O to **right breakout I**

Python Installation of NeoPixel Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python.

From your command line run the following commands:

- `pip3 install adafruit-circuitpython-neopixel`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython Usage

To run all of the CircuitPython demos, you need to first install the **Adafruit_CircuitPython_NeoPixel** library and its dependency. To run the HID demo, you will also need to install the **Adafruit_CircuitPython_HID** library. To install a library, you copy it into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In each of the demos, click the **Download Project Bundle** button to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

You'll find these instructions and details about necessary libraries for each specific demo with all three of the examples below.

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

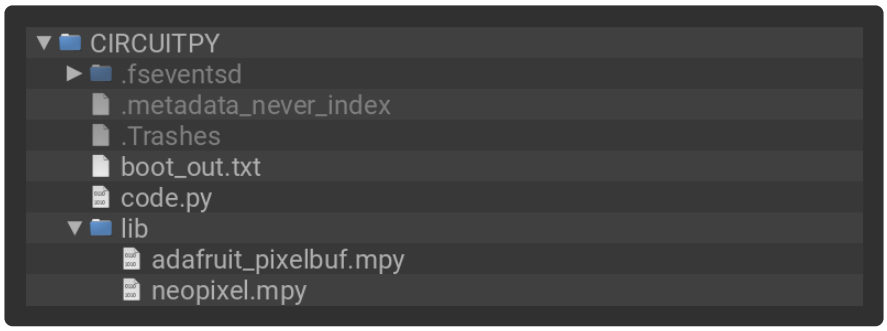
```
python3 code.py
```

Key Press NeoPixel Random Color Demo

Click the **Download Project Bundle** button to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following files:

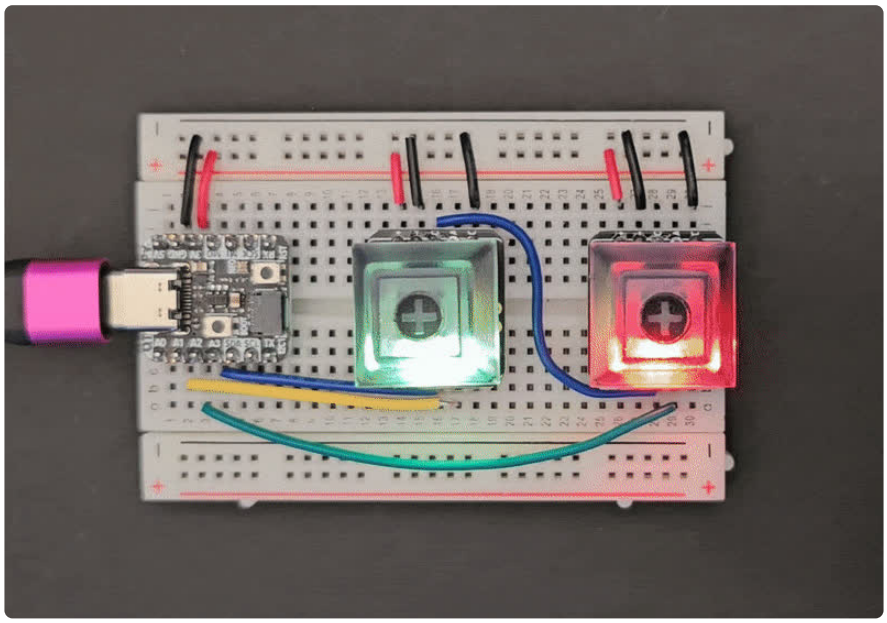
- **adafruit_pixelbuf.mpy**
- **neopixel.mpy**




```
# SPDX-FileCopyrightText: 2023 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""NeoKey Breakout Random NeoPixel Color Demo"""

import random
import board
import keypad
import neopixel
from rainbowio import colorwheel

# --- CONFIGURATION ---
# NeoPixel brightness. Must be a float or integer between 0.0 and 1.0, where 0 is
off and
# 1 is maximum brightness. Defaults to 0.3.
BRIGHTNESS = 0.3

# NeoPixel and key switch pins. If using different pins, update to match your
wiring setup.
# pylint: disable=simplifiable-condition
# Check to see if a Raspberry Pi is present, and set the appropriate pins.
if "CE0" and "CE1" in dir(board):  # These pins are Pi-specific.
    PIXEL_PIN = board.D18
    KEY_PINS = (
        board.D4,
        board.D17,
    )
# Otherwise, assume a microcontroller, and set the appropriate pins.
else:
    PIXEL_PIN = board.A3
    KEY_PINS = (
        board.A1,
        board.A2,
    )
```

```
# --- SET UP AND CODE ---
# Number of NeoPixels. This will always match the number of breakouts and
# therefore the number of key pins listed.
NUM_PIXELS = len(KEY_PINS)

# Create NeoPixel object.
pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=BRIGHTNESS)

# Create keypad object.
keys = keypad.Keys(KEY_PINS, value_when_pressed=False, pull=True)

while True:
    # Begin getting key events.
    event = keys.events.get()
    # If there is a key press event, run this block.
    if event and event.pressed:
        # Print the key number of the pressed key.
        print(f"Key {event.key_number} pressed!")
        # Light up the corresponding NeoPixel to a random color.
        pixels[event.key_number] = colorwheel(random.randint(0, 255))
```

Ensure you have everything copied to your **CIRCUITPY** drive. Once that is complete, try pressing each key to see the NeoPixel light up a random color!



Customisations

Each of the examples on this page have code at the beginning that includes the things that you can customise. There are three items that are present in all of the demos.

NeoPixel Brightness

Here you can customise the brightness of the NeoPixel LED. The value must be a float or integer between 0.0 and 1.0, where 0 is off, and 1 is maximum brightness. It defaults to `0.3` because NeoPixels at max are very bright!

```
BRIGHTNESS = 0.3
```

Key Switch and NeoPixel Pins

If you wired your breakouts to your microcontroller or Raspberry Pi as shown above, you do not need to make any changes here.

If you followed the CircuitPython Microcontroller Wiring or Python Computer Wiring diagrams above, you will not need to make any changes to this section of code. However, if you changed any of the connected pins, you will need to update this to match.

Raspberry Pi Pins

If you made changes to the wiring on a Raspberry Pi, you will need to **update the first set of pins**, nested under `if "CE0" and "CE1" in dir(board):`.

If you moved the NeoPixel IN connection to another pin on your Pi, you would update `PIXEL_PIN` to the Pi pin to which you connected the left breakout **I** pin.

If you moved either breakout **A** connection, you would update `KEY_PINS` to match.

Microcontroller Pins

If you made changes to the wiring on a microcontroller, you will need to **update the second set of pins**, nested under `else:`.

If you moved the NeoPixel IN connection to another pin on your microcontroller, you would change `PIXEL_PIN` to the microcontroller pin to which you connected the left breakout **I** pin.

If you moved either breakout **A** connection, you would update `KEY_PINS` to match.

```
if "CE0" and "CE1" in dir(board):
    PIXEL_PIN = board.D18
    KEY_PINS = (
        board.D4,
        board.D17,
    )
else:
    PIXEL_PIN = board.A3
    KEY_PINS = (
        board.A1,
        board.A2,
    )
```

Set Up

Each of the examples on this page include set up code. The code snippets below are included in all three demos.

Number of NeoPixels

The number of NeoPixels will always be equal to the number of breakouts, and therefore, to the number of key pins. To make this example extensible to more breakouts, `NUM_PIXELS` is set to the number of key pins. You will not need to update this if you add more NeoKey breakouts.

```
NUM_PIXELS = len(KEY_PINS)
```

NeoPixel and keypad Objects

Next you create the NeoPixel and keypad objects to enable access to the LEDs and the keys.

```
pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=BRIGHTNESS)
keys = keypad.Keys(KEY_PINS, value_when_pressed=False, pull=True)
```

Loop

Inside the loop, you first begin getting the key events. This allows you to complete actions based on a specific key event.

```
event = keys.events.get()
```

If a key press event happens, two thing will occur.

The code prints the number of the key pressed to the serial console. Remember, Python begins counting at 0!

Finally, it sets the LED associated with the key that you pressed to a random color. `colorwheel` expects a color value between 0 and 255, which covers the entire rainbow. Using the built-in `random` module, you can set the color to a random value within that range.

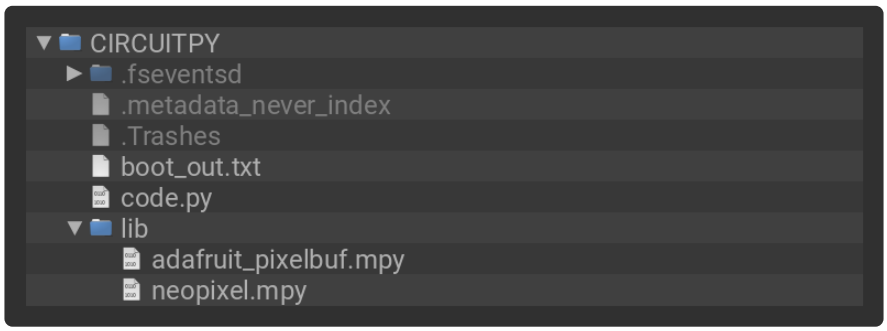
```
if event and event.pressed:
    print(f"Key {event.key_number} pressed!")
    pixels[event.key_number] = colorwheel(random.randint(0, 255))
```

Key Press NeoPixel Rainbow Demo

Click the **Download Project Bundle** button to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following files:

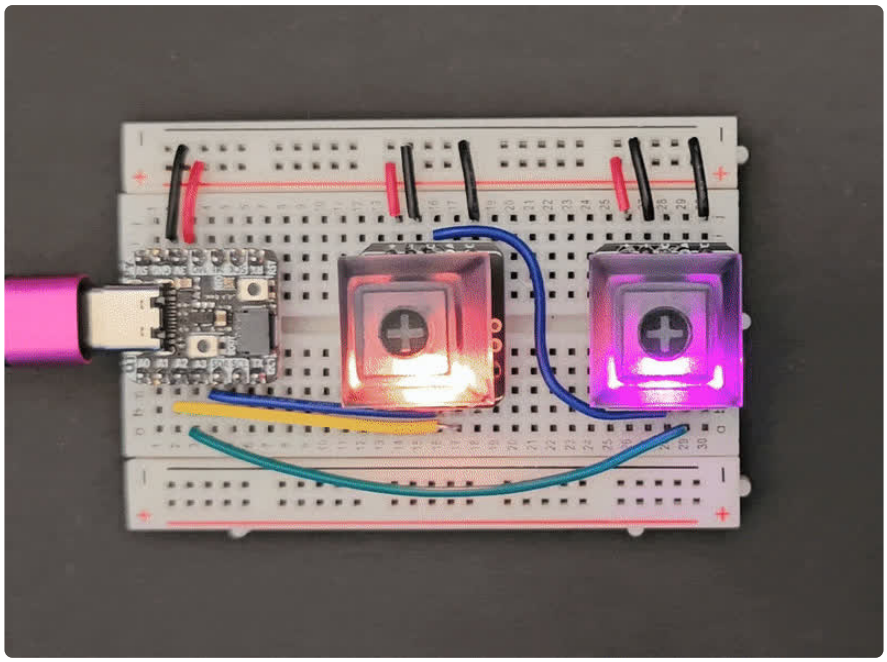
- **adafruit_pixelbuf.mpy**
- **neopixel.mpy**




```
# SPDX-FileCopyrightText: 2023 Kattni Rembor for Adafruit Industries
# SPDX-FileCopyrightText: 2023 Dan Halbert for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""NeoKey Breakout NeoPixel Rainbow Cycle Demo"""

import time
import board
import keypad
import neopixel
from rainbowio import colorwheel

# --- CONFIGURATION ---
# NeoPixel brightness. Must be a float or integer between 0.0 and 1.0, where 0 is
off and
# 1 is maximum brightness. Defaults to 0.3.
```

```
BRIGHTNESS = 0.3

# NeoPixel and key switch pins. If using different pins, update to match your
wiring setup.
# pylint: disable=simplifiable-condition
# Check to see if a Raspberry Pi is present, and set the appropriate pins.
if "CE0" and "CE1" in dir(board):  # These pins are Pi-specific.
    PIXEL_PIN = board.D18
    KEY_PINS = (
        board.D4,
        board.D17,
    )
# Otherwise, assume a microcontroller, and set the appropriate pins.
else:
    PIXEL_PIN = board.A3
    KEY_PINS = (
        board.A1,
        board.A2,
    )

# --- SETUP AND CODE ---
# Number of NeoPixels. This will always match the number of breakouts and
# therefore the number of key pins listed.
NUM_PIXELS = len(KEY_PINS)

# Create NeoPixel object.
pixels = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, brightness=BRIGHTNESS)

# Create keypad object.
keys = keypad.Keys(KEY_PINS, value_when_pressed=False, pull=True)

# Create two lists.
# The `pressed` list is used to track the key press state.
pressed = [False] * NUM_PIXELS
# The `color_value` list is used to track the current color value for a specific
NeoPixel.
color_value = [0] * NUM_PIXELS

while True:
    # Begin getting key events.
    event = keys.events.get()
    if event:
        # Remember the last state of the key when pressed.
        pressed[event.key_number] = event.pressed

    # Advance the rainbow for the key that is currently pressed.
    for index in range(NUM_PIXELS):
        if pressed[index]:
            # Increase the color value by 1.
            color_value[index] += 1
            # Set the pixel color to the current color value.
            pixels[index] = colorwheel(color_value[index])

    time.sleep(0.01)
```

Ensure you have everything copied to your **CIRCUITPY** drive. Once that is complete, try pressing each key to see the NeoPixel light up in a rainbow cycle while the key is pressed! Try releasing the key to stop the rainbow cycle.



Customisations

See the Customisations section (https://adafru.it/18Fu) above for details.

Set Up

For the first part of the set up in this demo, see the Set Up (https://adafru.it/18Fu) section above for details.

Create Two Lists

The final part of the code set up creates two lists.

The `pressed` list is used to track the key press state.

The `color_value` list is used to track the current color for a specific NeoPixel.

```
pressed = [False] * NUM_PIXELS
color_value = [0] * NUM_PIXELS
```

Loop

Inside the loop, you first begin getting the key events. This allows you to complete actions based on a specific key event.

```
event = keys.events.get()
```

If there is a key event, used the `pressed` list to remember the state of each key when pressed.

```
if event:
    pressed[event.key_number] = event.pressed
```

Rainbow Cycle

This code block advances the rainbow for the key that is currently pressed. Each time a specific key is pressed, the `color_value` is increased by 1 for that NeoPixel LED, and then the associated pixel is set to the color using `colorwheel`.

The last line in the file is a `time.sleep()` to keep things running smoothly.

```
for index in range(NUM_PIXELS):
  if pressed[index]:
    color_value[index] += 1
    pixels[index] = colorwheel(color_value[index])

time.sleep(0.01)
```

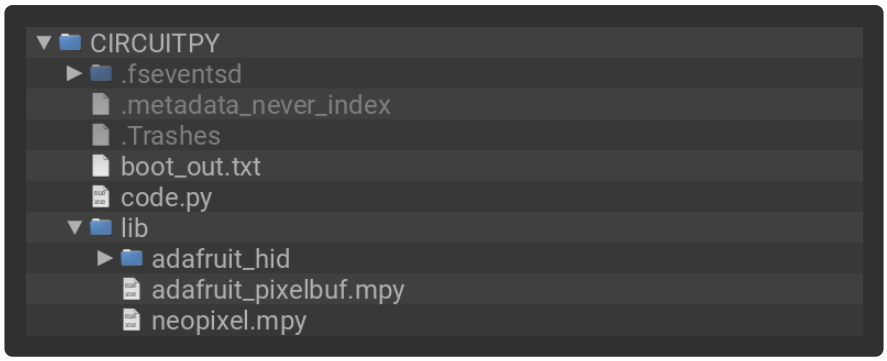
HID Keyboard Demo

This demo will not work on a Raspberry Pi! It requires a microcontroller.

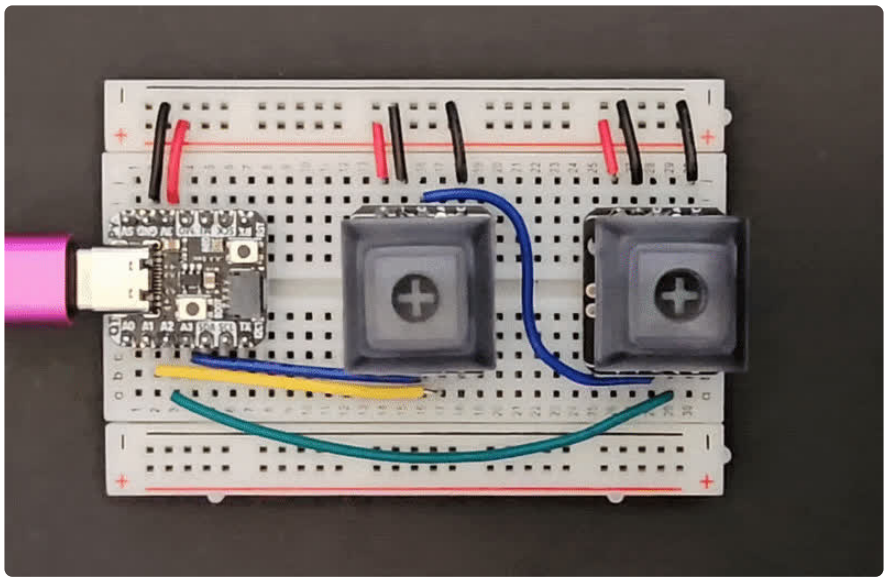
Click the **Download Project Bundle** button to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folder and files:

- **adafruit_hid/**
- **adafruit_pixelbuf.mpy**
- **neopixel.mpy**



Ensure everything is copied to your **CIRCUITPY** drive. Once that is complete, open up a text editor. Press the left breakout key switch to type **b** and see the LED light up yellow. Press the right breakout key switch to type **3** and see the LED light up cyan.



Customisations

There are two customisable tuples at the beginning of this demo.

The first contains the keycodes to assign to each key, and is saved to `SEND_ON_PRESS`. You can update what each key sends by changing either member of this tuple to a different keycode.

The second contains the colors to light up each LED when the key is pressed, and is saved to `COLORS`. You can make the LEDs light up different colors by changing either of the color tuples.

```
SEND_ON_PRESS = (
    Keycode.B,
    Keycode.THREE,
)
COLORS = (
    (255, 255, 0),
```

```
    (0, 255, 255),
)
```

For the last two options in this code, see the Customisations section (https://adafru.it/18Fu) above for details.

Set Up

For the first part of the set up in this demo, see the Set Up (https://adafru.it/18Fu) section above for details.

The final set up is to create the keyboard object. It includes a delay to avoid a race condition on some systems.

```
time.sleep(1)
keyboard = Keyboard(usb_hid.devices)
```

Loop

Inside the loop, you first begin getting the key events. This allows you to complete actions based on a specific key event.

```
event = keys.events.get()
```

The Key Press Event

If a key is pressed, get the key number and save it to `key_number` as it is used multiple times in the rest of this block. Light up the NeoPixel associated with the key to the color associated with that switch's pixel from the `COLORS` list. Finally, send the keycode associated with that key from the `SEND_ON_PRESS` list.

```
if event and event.pressed:
    key_number = event.key_number
    pixels[key_number] = COLORS[key_number]
    keyboard.press(SEND_ON_PRESS[key_number])
```

The Key Release Event

If a key is released, turn off the LED associated with that key, and report that the key switch has been released.

```
if event and event.released:
    pixels[event.key_number] = (0, 0, 0)
    keyboard.release_all()
```

CircuitPython keypad Docs

CircuitPython keypad Docs (https://adafru.it/18Fx)

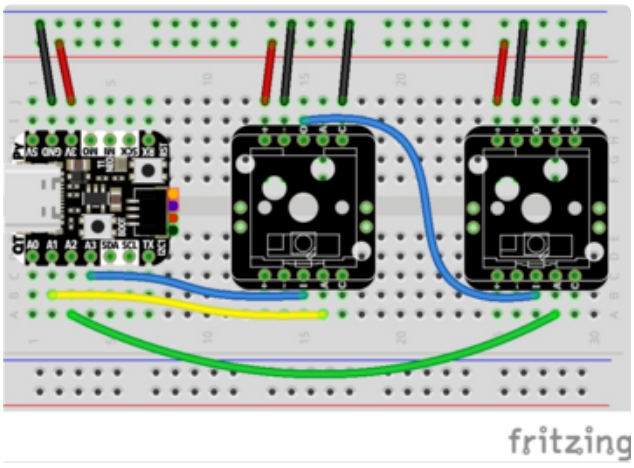
CircuitPython NeoPixel Docs

CircuitPython NeoPixel Docs (https://adafru.it/18gA)

Arduino

Using the NeoKey Breakouts with Arduino involves wiring the breakouts up to your Arduino-compatible board, installing the Adafruit_NeoPixel (https://adafru.it/aZU) library, and running the provided example code.

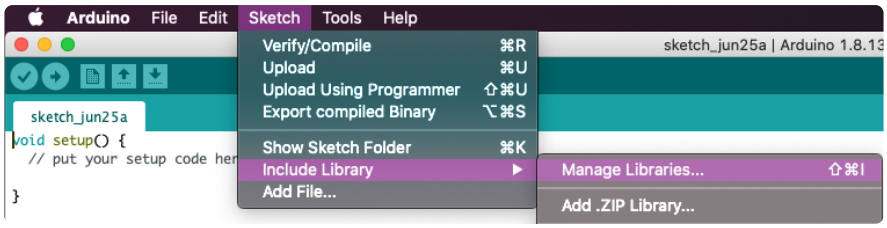
Wiring



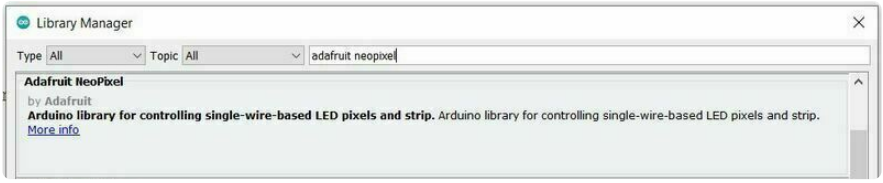
QT Py GND to **ground rail on breadboard**
QT Py 3V to **power rail on breadboard**
Both NeoKey breakouts S+ to **power rail on breadboard**
Both NeoKey breakouts S- to **ground rail on breadboard**
Both NeoKey breakouts C to **ground rail on breadboard**
QT Py A1 to **left breakout A**
QT Py A2 to **right breakout A**
QT Py A3 to **left breakout I**
Left breakout O to **right breakout I**

Library Installation

You can install the **Adafruit NeoPixel** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit NeoPixel** and select the **Adafruit NeoPixel** library:



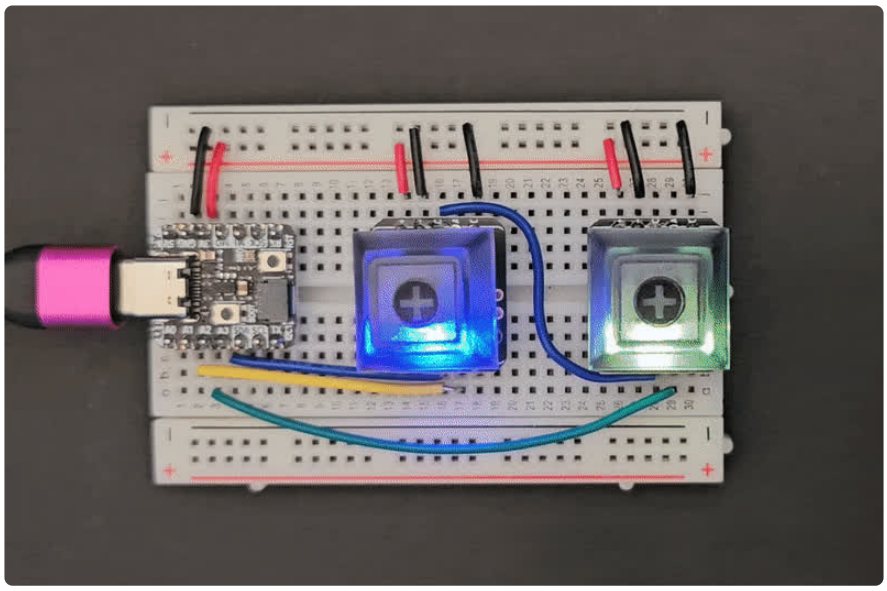
There are no additional library dependencies for the NeoPixel library.

Example Code

```
// SPDX-FileCopyrightText: 2023 Kattni Rembor for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Adafruit_NeoPixel.h>

#define NEOPIXEL_PIN A3
#define NUM_PIXELS 2

#define SWITCHA_PIN A1
#define SWITCHB_PIN A2

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_PIXELS, NEOPIXEL_PIN, NEO_GRB +
NEO_KHZ800);

void setup() {
  Serial.begin(115200);
  //while (!Serial);
  strip.begin();
  strip.setBrightness(25);
  strip.show(); // Initialize all pixels to 'off'
  pinMode(SWITCHA_PIN, INPUT_PULLUP);
  pinMode(SWITCHB_PIN, INPUT_PULLUP);
}

uint8_t j=0;
void loop() {
  for(int i=0; i< strip.numPixels(); i++) {
    strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
  }

  // turn off LED if not pressed!
  if (!digitalRead(SWITCHA_PIN)) {
    Serial.println("A");
    strip.setPixelColor(0, 0);
  }
  if (!digitalRead(SWITCHB_PIN)) {
    Serial.println("B");
```

```
    strip.setPixelColor(1, 0);
  }

  strip.show();

  j++; // go to next color
  delay(10);
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {

}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Upload the sketch to your board. The NeoPixel LEDs on the NeoKey breakouts will light up in a rainbow swirl pattern. If you press a key, the associated LED will turn off. When you release it, the LED will begin rainbowing again!



Arduino NeoPixel Docs

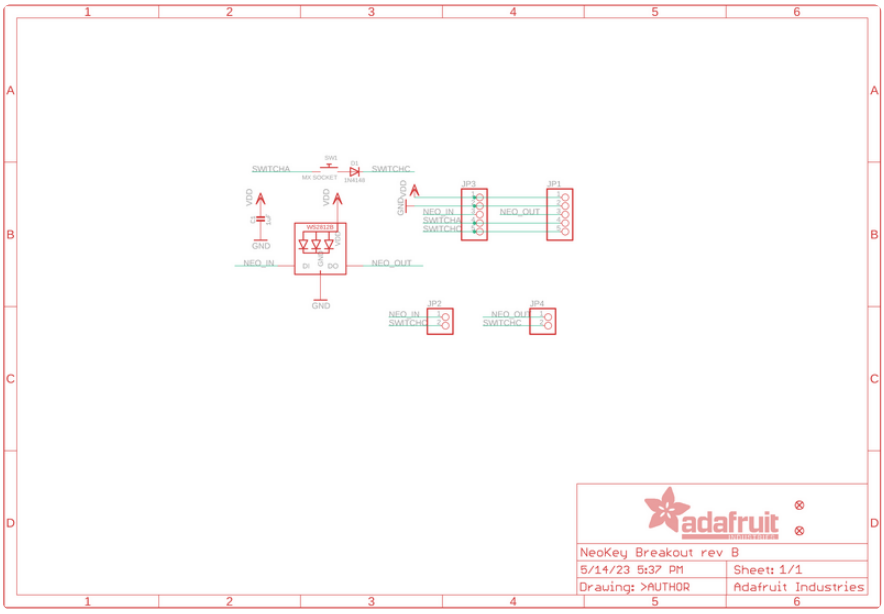
Arduino NeoPixel Docs (https://adafru.it/Etk)

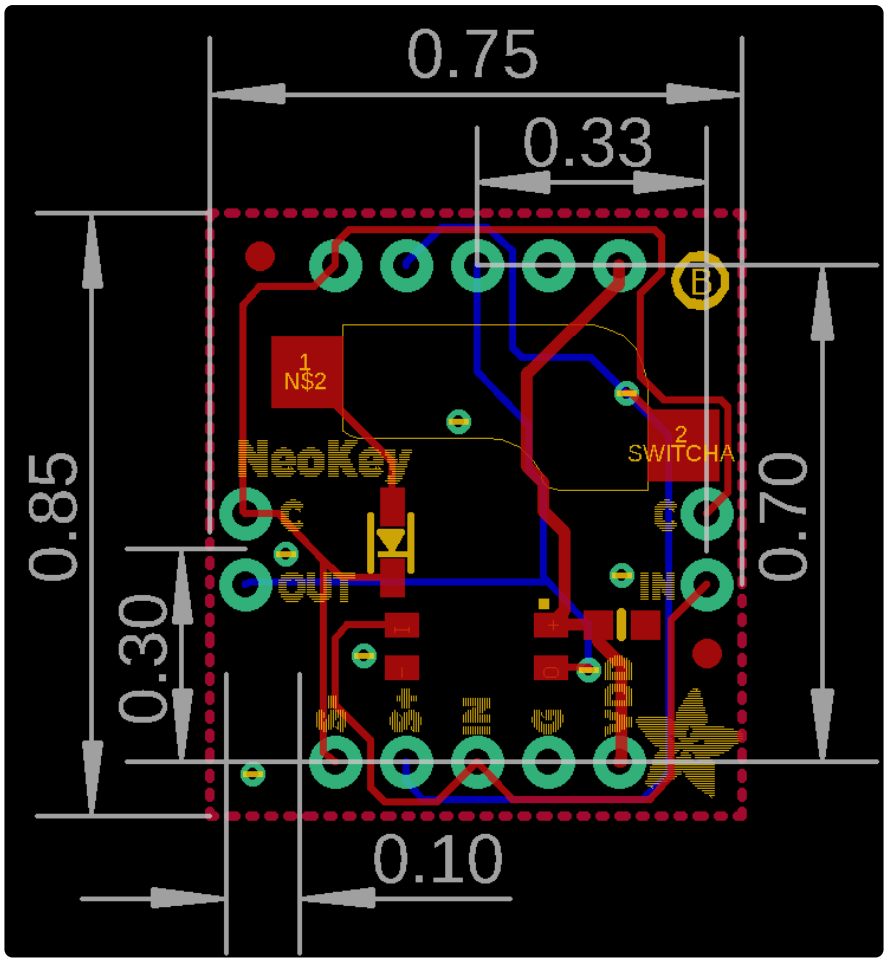
Downloads

Files

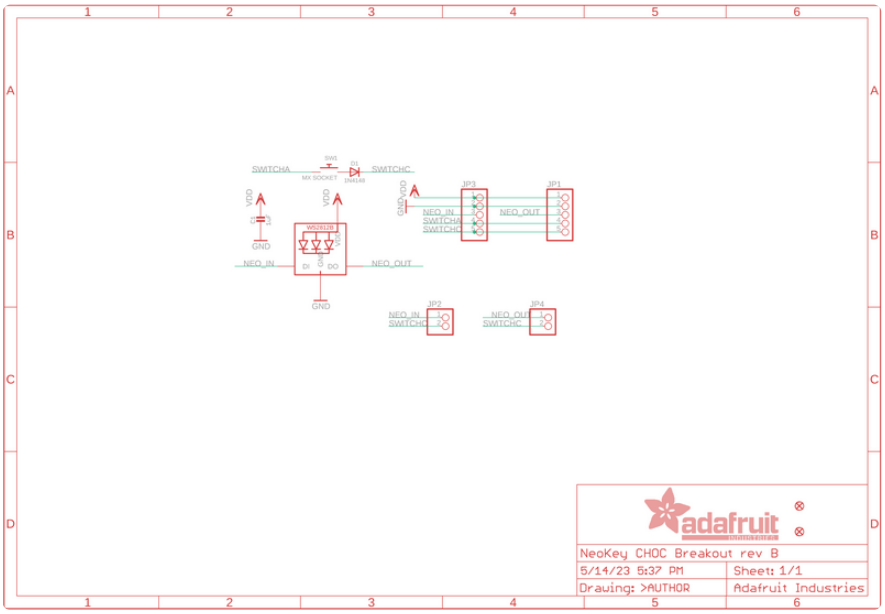
- MX key switch socket datasheet (https://adafru.it/18FA)
- CHOC key switch socket datasheet (https://adafru.it/18FC)
- MX Breakout Fritzing object in the Adafruit Fritzing Library (https://adafru.it/18FG)
- CHOC Breakout Fritzing object in the Adafruit Fritzing Library (https://adafru.it/18FI)
- MX Breakout EagleCAD PCB files on GitHub (https://adafru.it/18FL)
- CHOC Breakout EagleCAD PCB files on GitHub (https://adafru.it/18FN)

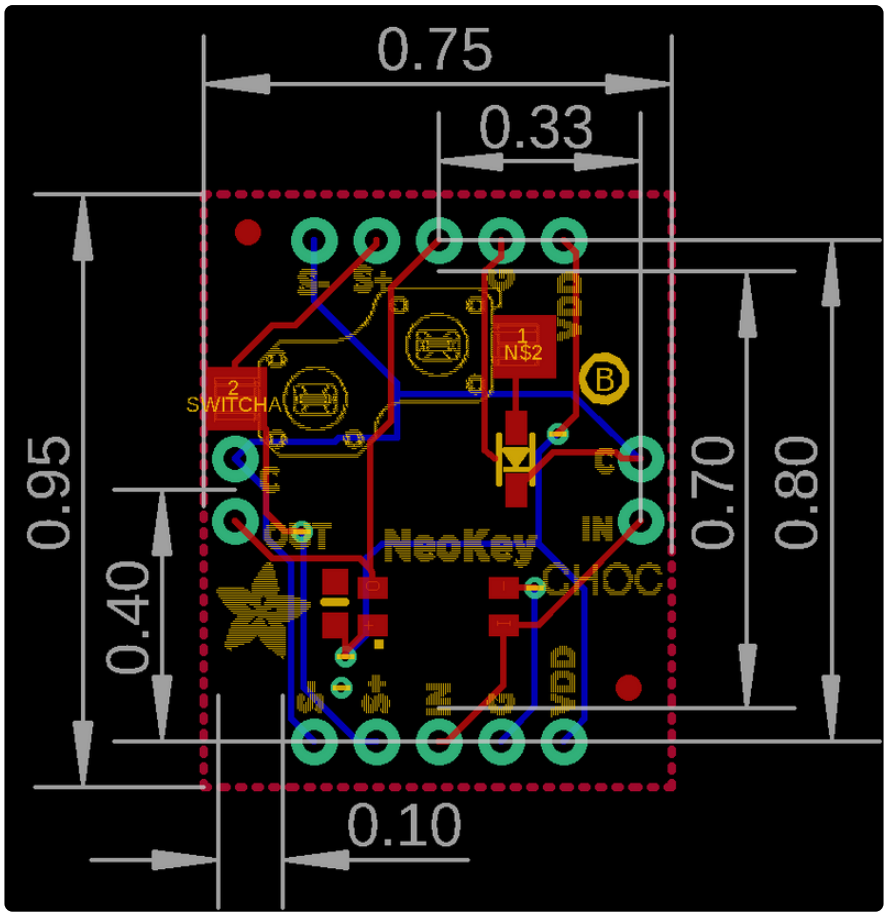
MX Breakout Schematic and Fab Print





CHOC Breakout Schematic and Fab Print




0.75
0.33
0.95
0.40
0.70
0.80
0.10
SWITCHA
N$2
VDD
OUT
NeoKey
IN
CHOC