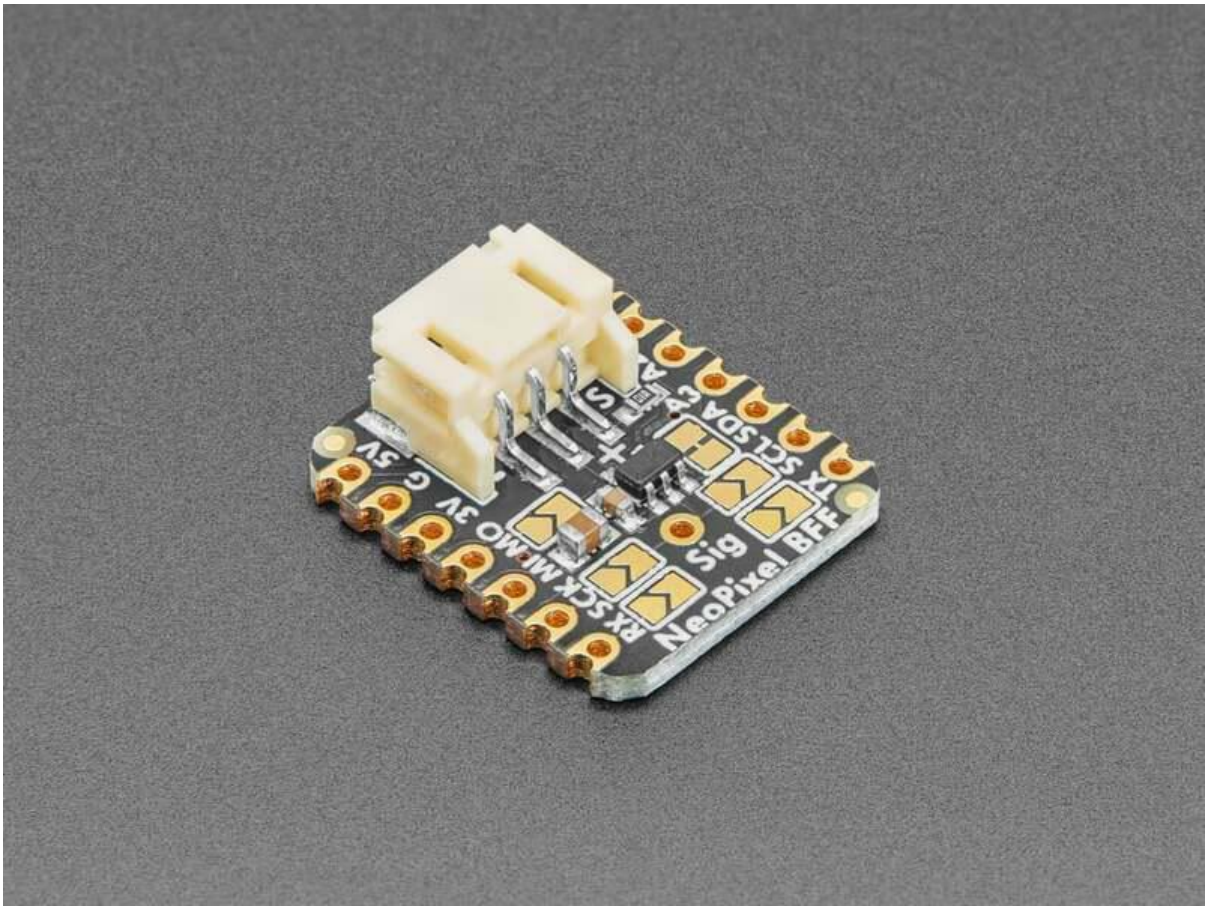
Adafruit NeoPixel Driver BFF

Created by Liz Clark

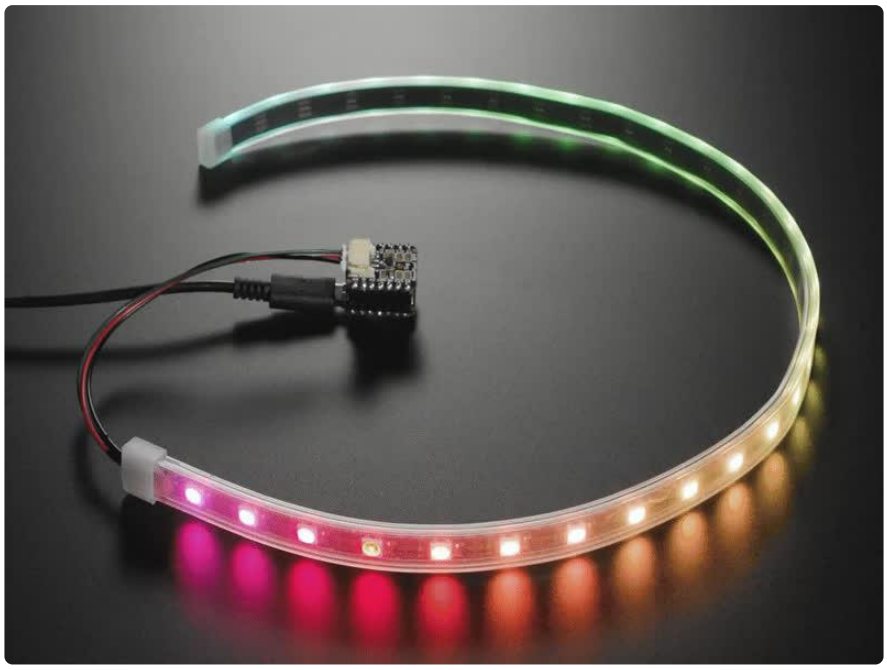


https://learn.adafruit.com/adafruit-neopixel-driver-bff

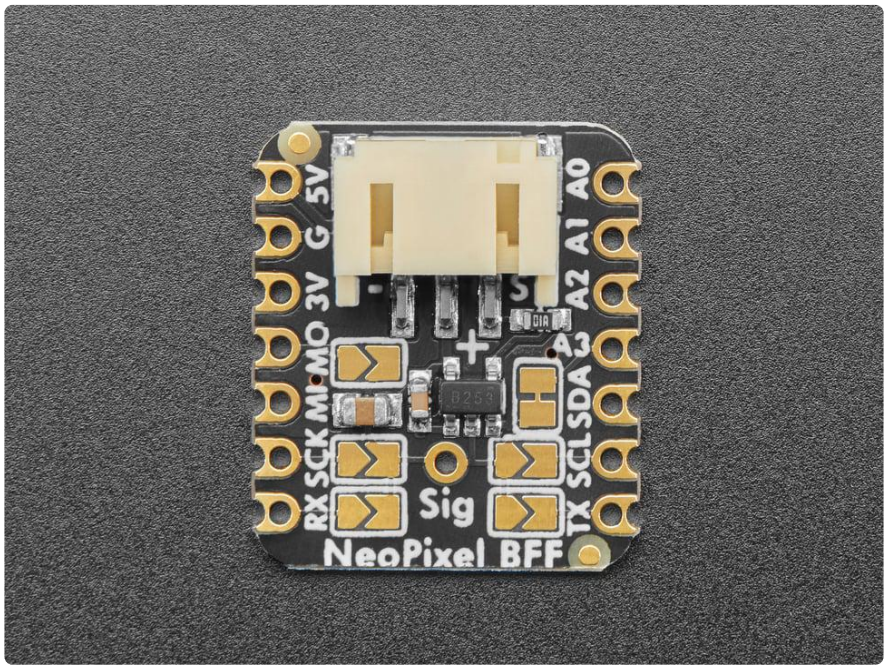
Last updated on 2022-12-01 04:14:57 PM EST

Table of Contents

Overview

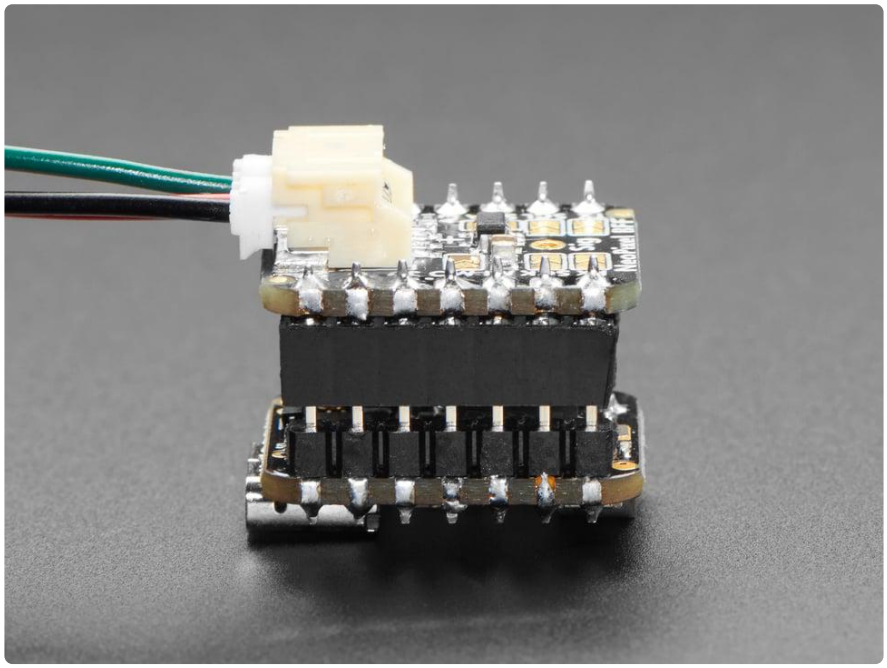


Adafruit's QT Py boards are a great way to make very small microcontroller projects that pack a ton of power - and now there is a way for you to quickly add a strand of NeoPixels () with a 5V level shifter and a detachable JST PH connector. It's an excellent way to make tiny wearable, cosplay or IoT projects with dazzling LEDs.

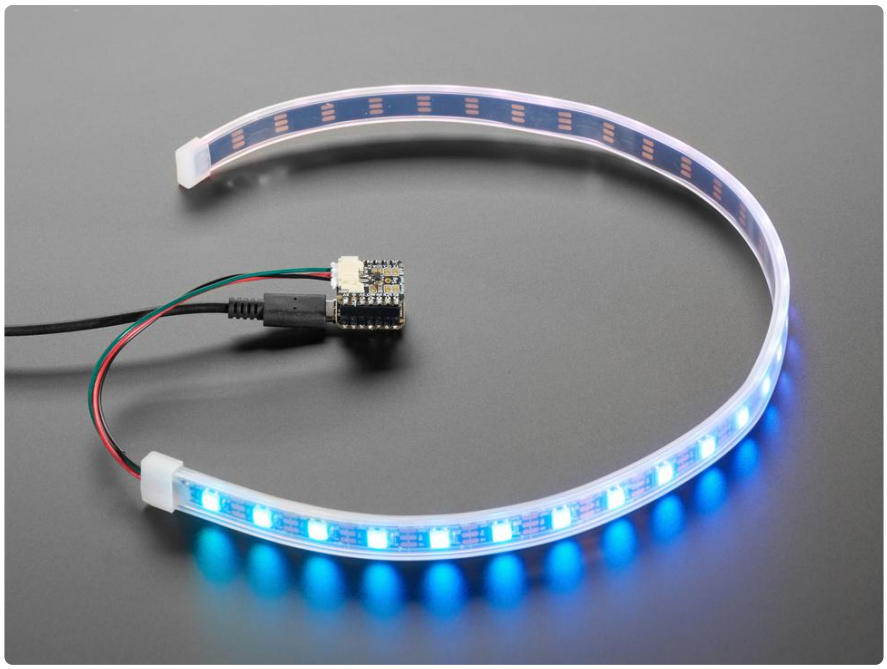


We call this the Adafruit NeoPixel Driver BFF - a "Best Friend Forever". When you were a kid, you may have learned about the "buddy" system. Well, this product is

kinda like that! A board that will watch your QT Py's back and give it the power and support to drive NeoPixel strips, rings, and panels.

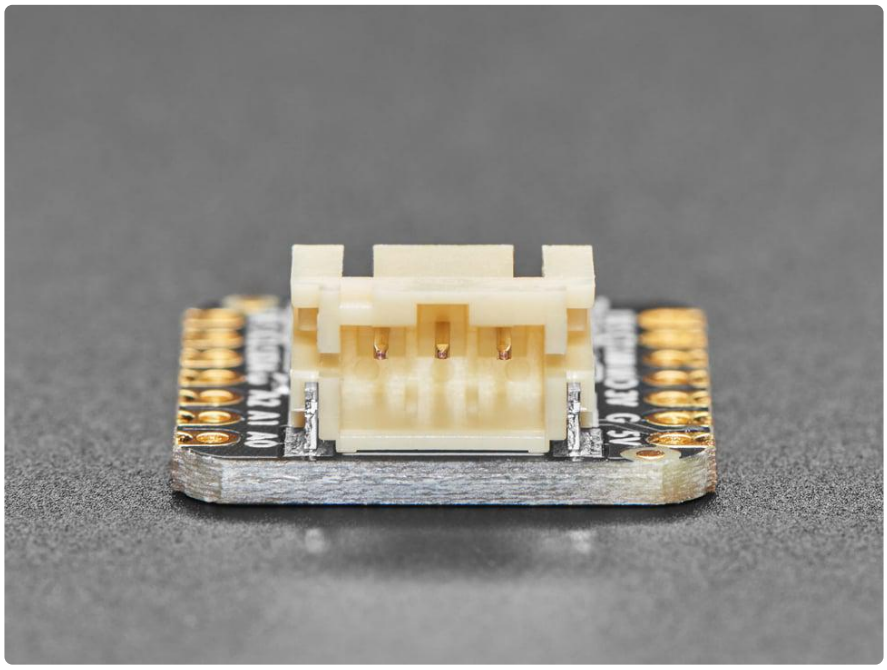


This PCB is designed to fit onto the back of any QT Py or Xiao board, it can be soldered into place or use pin and socket headers to make it removable. Onboard is a 74AHCT125 single-gate level shifter that will take pin A3's output, and shift it up to 5V using the USB power as a reference. The USB power, ground and shifted data is then piped out to a JST PH 2mm 3-pin connector.

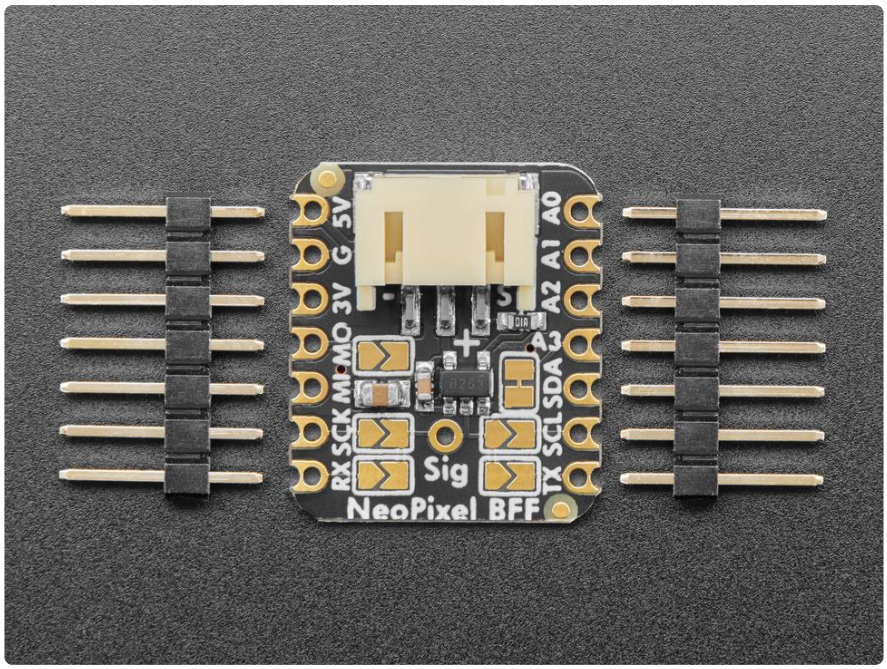


If using one of our NeoPixel strips with a JST-PH connector on them already (), you can just plug it right in for instant lights. If you're using a common JST SH (black in-

line) connector on the strips, you can use a JST-PH-to-female-header cable () and use that to plug into the pins on the input of the SH port and/or can jam the wires into the header sockets.



Note that we connect the USB 5V power directly out to the NeoPixel power pin on the JST PH connector and PH connectors are only rated for about 2 Amps, so it's not for powering directly more than maybe 30-60 NeoPixels (less if they're on full white brightness).

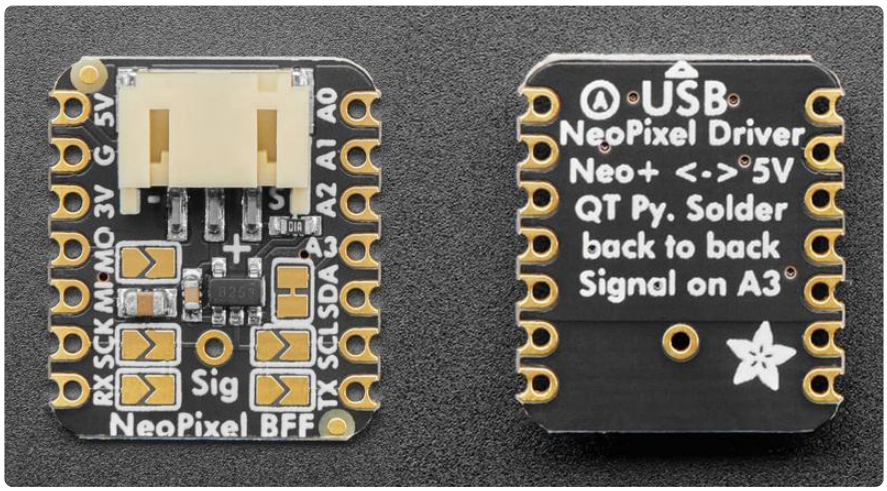


We include some header that you can solder to your QT Py. You can also pick up an Itsy Bitsy short female header kit to make it removable but compact (), you'll just need to trim down the headers to 7 pins long.

- Comes as an assembled and tested PCB
- For any QT Py or Xiao boards
- Uses the 5V input via USB Type-C connector on QT Py to Power 30ish NeoPixels
- Level-shifted data from pin A3 default, can solder-jumper select SCL, TX, RX, SCK or MOSI
- For driving any WS2812/WS2811/NeoPixel LEDs

NeoPixels and QT Py are not included.

Pinouts



The default input pin is A3.

STEMMA JST PH

- STEMMA JST PH () - 2mm pitch STEMMA JST port with connections for +, - and Sig to connect to a NeoPixel strip. Plug in a 3-pin STEMMA JST PH cable () with headers or a NeoPixel strip with a JST PH plug ().

Output Pin

- Sig - this is the signal output pin for the BFF. It is a through hole pad located in the center of the board. It is connected to pin A3 by default. It outputs the data signal to a connected NeoPixel strip.

A3 Jumper

- A3 jumper - This jumper is located on the front of the board behind the signal pin of the JST PH port. If cut, the Sig pin is no longer connected to A3 and you can solder one of the solder-jumpers closed to change the output pin.

Pin Select Solder-jumpers

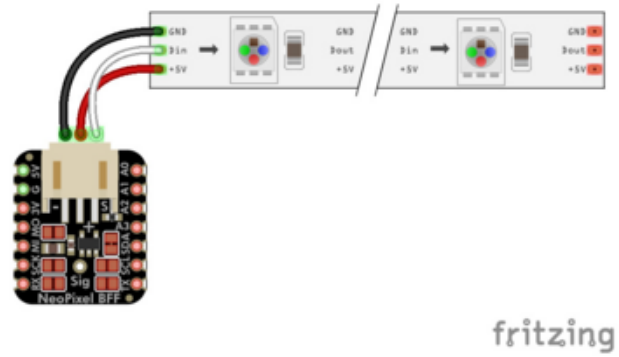
- Five solder-jumpers are available to change the signal output pin for the NeoPixel BFF. They are placed horizontally on the board and are located next to their corresponding pin names. You can choose to solder one of the solder-jumpers for one of the following pins:
    - SCL
    - TX
    - RX
    - SCK
    - MOSI

CircuitPython

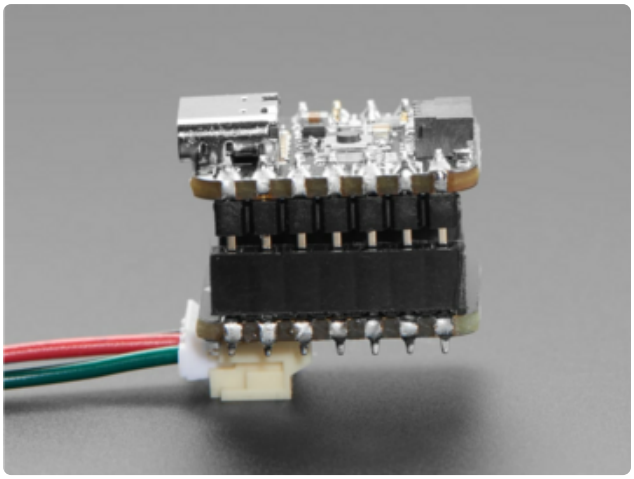
It's easy to use the NeoPixel Driver BFF with CircuitPython and the Adafruit_CircuitPython_NeoPixel () module. This module allows you to easily write Python code that lets you control NeoPixels.

CircuitPython Microcontroller Wiring

First, wire up a NeoPixel Driver BFF to your board exactly as shown below. Here's an example of wiring a QT Py RP2040 to the BFF with a strip of NeoPixels with a JST PH plug ():



Plug the JST PH NeoPixel strip into the NeoPixel Driver BFF JST PH port.



Connect the QT Py RP2040 with plug headers into the NeoPixel Driver BFF with socket headers. They should be plugged in with the backs of the boards facing each other.

For more information on soldering socket headers, check out this Learn Guide ().

How to Solder Headers Learn Guide

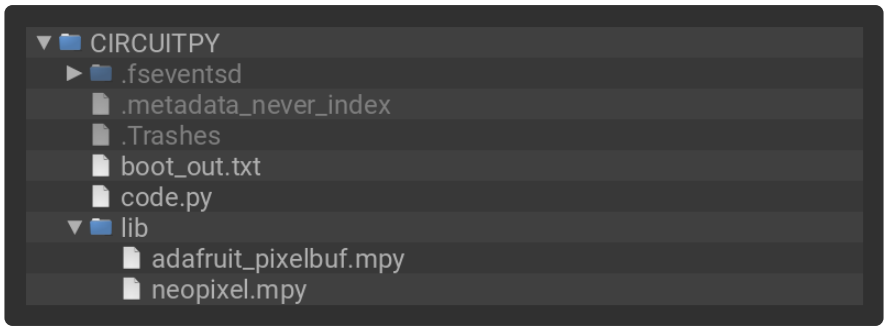
CircuitPython Usage

To use with CircuitPython, you need to first install the NeoPixel library, and its dependencies, 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 the example below, click the Download Project Bundle button below 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 c ode.py file to your CIRCUITPY drive.

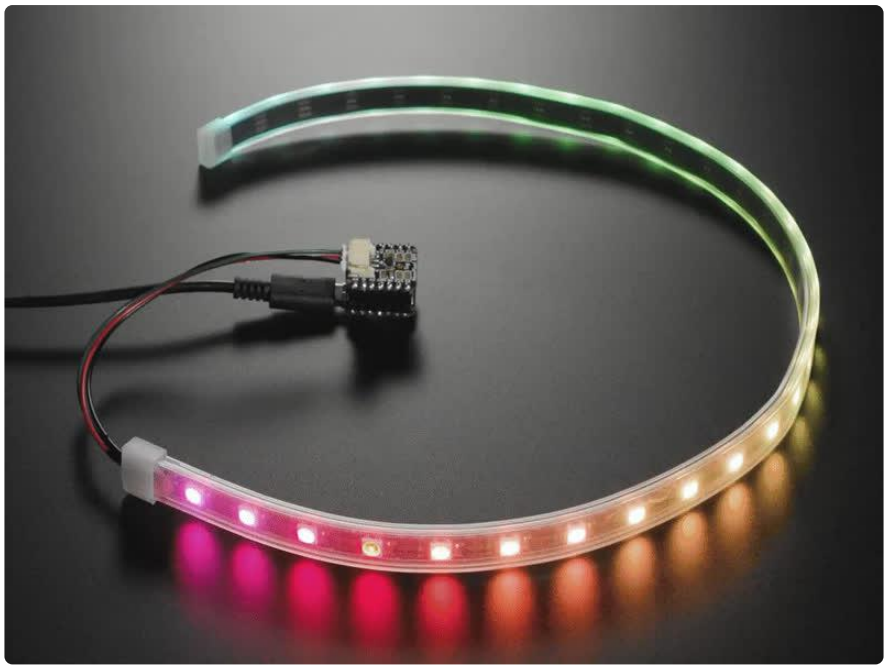
Your CIRCUITPY/lib folder should contain the following files:

- adafruit_pixelbuf.mpy
- neopixel.mpy



Example Code

```
# SPDX-FileCopyrightText: 2022 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
from rainbowio import colorwheel
import neopixel

NUMPIXELS = 12  # Update this to match the number of LEDs.
SPEED = 0.05  # Increase to slow down the rainbow. Decrease to speed it up.
BRIGHTNESS = 0.2  # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.
PIN = board.A3  # This is the default pin on the 5x5 NeoPixel Grid BFF.

pixels = neopixel.NeoPixel(PIN, NUMPIXELS, brightness=BRIGHTNESS, auto_write=False)


def rainbow_cycle(wait):
    for color in range(255):
        for pixel in range(len(pixels)):  # pylint: disable=consider-using-enumerate
            pixel_index = (pixel * 256 // len(pixels)) + color * 5
            pixels[pixel] = colorwheel(pixel_index & 255)
        pixels.show()
        time.sleep(wait)


while True:
    rainbow_cycle(SPEED)
```



Once everything is saved to the CIRCUITPY drive, you'll see your connected NeoPixel strip cycle through a rainbow animation on a loop.

You can edit the variables in the code for the number of pixels, speed, brightness and pin number depending on your needs.

```
NUMPIXELS = 12  # Update this to match the number of LEDs.
SPEED = 0.05  # Increase to slow down the rainbow. Decrease to speed it up.
BRIGHTNESS = 0.2  # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.
PIN = board.A3  # This is the default pin on the 5x5 NeoPixel Grid BFF.

pixels = neopixel.NeoPixel(PIN, NUMPIXELS, brightness=BRIGHTNESS, auto_write=False)
```

Python Docs

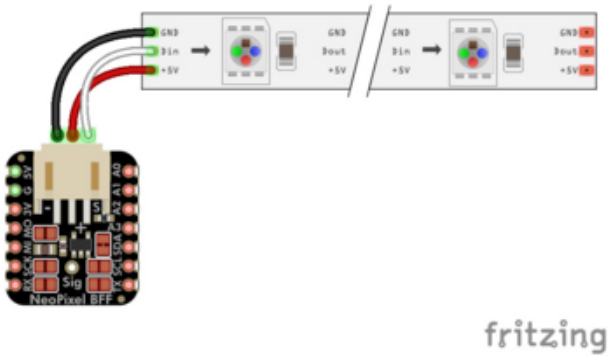
Python Docs ()

Arduino

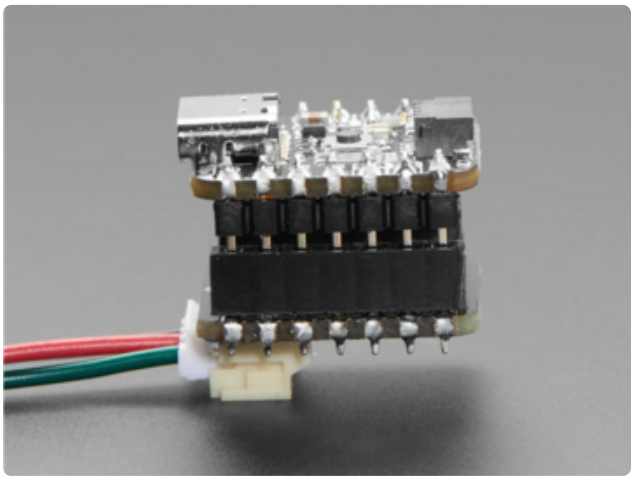
Using the NeoPixel Driver BFF with Arduino involves wiring up the breakout to your Arduino-compatible QT Py or Xiao form factor board, installing the Adafruit_NeoPixel ( ) library and running the provided example code.

Wiring

Wire up a NeoPixel Driver BFF to your board exactly as shown below. Here's an example of wiring a QT Py RP2040 to the BFF with a strip of NeoPixels with a JST PH plug ():



Plug the JST PH NeoPixel strip into the NeoPixel Driver BFF JST PH port.



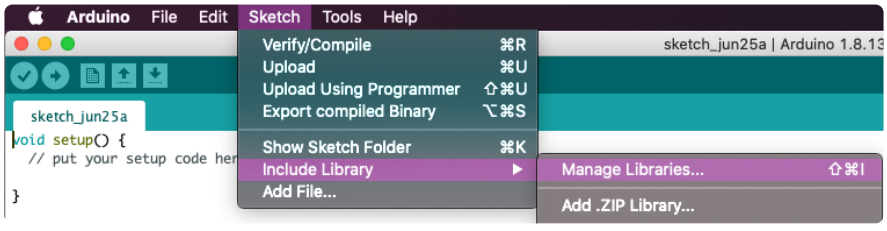
Connect the QT Py RP2040 with plug headers into the NeoPixel Driver BFF with socket headers. They should be plugged in with the backs of the boards facing each other.

For more information on soldering socket headers, check out this Learn Guide ().

How to Solder Headers Learn Guide

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 dependencies for the Adafruit NeoPixel library.

Example Code

```
// A basic everyday NeoPixel strip test program.

// NEOPIXEL BEST PRACTICES for most reliable operation:
// - Add 1000 uF CAPACITOR between NeoPixel strip's + and - connections.
// - MINIMIZE WIRING LENGTH between microcontroller board and first pixel.
// - NeoPixel strip's DATA-IN should pass through a 300-500 OHM RESISTOR.
// - AVOID connecting NeoPixels on a LIVE CIRCUIT. If you must, ALWAYS
//   connect GROUND (-) first, then +, then data.
// - When using a 3.3V microcontroller with a 5V-powered NeoPixel strip,
//   a LOGIC-LEVEL CONVERTER on the data line is STRONGLY RECOMMENDED.
// (Skipping these may work OK on your workbench but can fail in the field)

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1:
#define LED_PIN    6

// How many NeoPixels are attached to the Arduino?
#define LED_COUNT 60

// Declare our NeoPixel strip object:
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
// Argument 1 = Number of pixels in NeoPixel strip
// Argument 2 = Arduino pin number (most are valid)
// Argument 3 = Pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products)


// setup() function -- runs once at startup --------------------------------

void setup() {
  // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)
}


// loop() function -- runs repeatedly as long as board is on ---------------

void loop() {
```

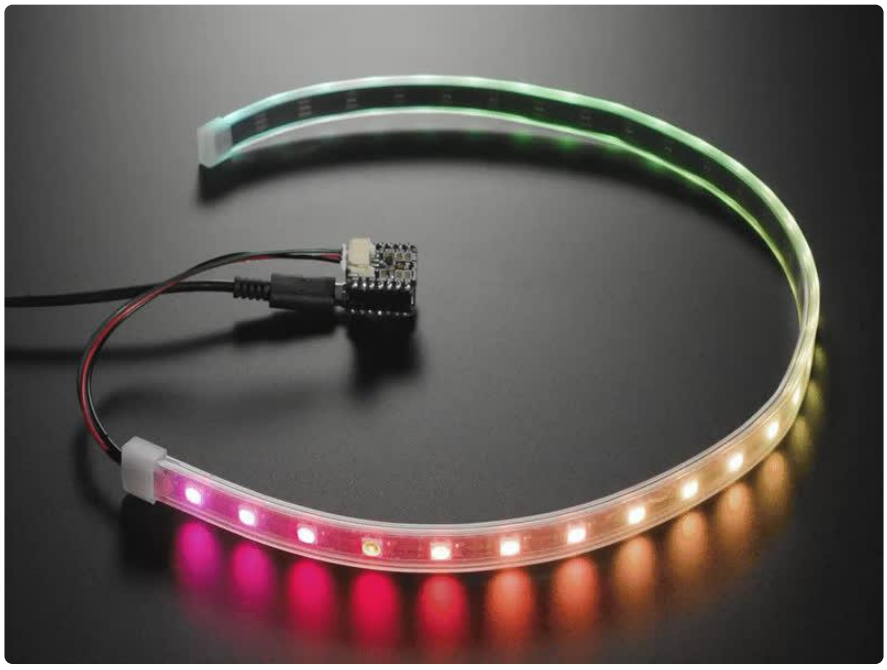
```
  // Fill along the length of the strip in various colors...
  colorWipe(strip.Color(255,   0,   0), 50); // Red
  colorWipe(strip.Color(  0, 255,   0), 50); // Green
  colorWipe(strip.Color(  0,   0, 255), 50); // Blue

  // Do a theater marquee effect in various colors...
  theaterChase(strip.Color(127, 127, 127), 50); // White, half brightness
  theaterChase(strip.Color(127,   0,   0), 50); // Red, half brightness
  theaterChase(strip.Color(  0,   0, 127), 50); // Blue, half brightness

  rainbow(10);             // Flowing rainbow cycle along the whole strip
  theaterChaseRainbow(50); // Rainbow-enhanced theaterChase variant
}


// Some functions of our own for creating animated effects -----------------

// Fill strip pixels one after another with a color. Strip is NOT cleared
// first; anything there will be covered pixel by pixel. Pass in color
// (as a single 'packed' 32-bit value, which you can get by calling
// strip.Color(red, green, blue) as shown in the loop() function above),
// and a delay time (in milliseconds) between pixels.
void colorWipe(uint32_t color, int wait) {
  for(int i=0; i<strip.numPixels(); i++) { // For each pixel in strip...
    strip.setPixelColor(i, color);         //  Set pixel's color (in RAM)
    strip.show();                          //  Update strip to match
    delay(wait);                           //  Pause for a moment
  }
}

// Theater-marquee-style chasing lights. Pass in a color (32-bit value,
// a la strip.Color(r,g,b) as mentioned above), and a delay time (in ms)
// between frames.
void theaterChase(uint32_t color, int wait) {
  for(int a=0; a<10; a++) {  // Repeat 10 times...
    for(int b=0; b<3; b++) { //  'b' counts from 0 to 2...
      strip.clear();         //   Set all pixels in RAM to 0 (off)
      // 'c' counts up from 'b' to end of strip in steps of 3...
      for(int c=b; c<strip.numPixels(); c += 3) {
        strip.setPixelColor(c, color); // Set pixel 'c' to value 'color'
      }
      strip.show(); // Update strip with new contents
      delay(wait);  // Pause for a moment
    }
  }
}

// Rainbow cycle along whole strip. Pass delay time (in ms) between frames.
void rainbow(int wait) {
  // Hue of first pixel runs 5 complete loops through the color wheel.
  // Color wheel has a range of 65536 but it's OK if we roll over, so
  // just count from 0 to 5*65536. Adding 256 to firstPixelHue each time
  // means we'll make 5*65536/256 = 1280 passes through this loop:
  for(long firstPixelHue = 0; firstPixelHue < 5*65536; firstPixelHue += 256) {
    // strip.rainbow() can take a single argument (first pixel hue) or
    // optionally a few extras: number of rainbow repetitions (default 1),
    // saturation and value (brightness) (both 0-255, similar to the
    // ColorHSV() function, default 255), and a true/false flag for whether
    // to apply gamma correction to provide 'truer' colors (default true).
    strip.rainbow(firstPixelHue);
    // Above line is equivalent to:
    // strip.rainbow(firstPixelHue, 1, 255, 255, true);
    strip.show(); // Update strip with new contents
    delay(wait);  // Pause for a moment
  }
}

// Rainbow-enhanced theater marquee. Pass delay time (in ms) between frames.
void theaterChaseRainbow(int wait) {
```

```
  int firstPixelHue = 0;     // First pixel starts at red (hue 0)
  for(int a=0; a<30; a++) {  // Repeat 30 times...
    for(int b=0; b<3; b++) { //  'b' counts from 0 to 2...
      strip.clear();         //   Set all pixels in RAM to 0 (off)
      // 'c' counts up from 'b' to end of strip in increments of 3...
      for(int c=b; c<strip.numPixels(); c += 3) {
        // hue of pixel 'c' is offset by an amount to make one full
        // revolution of the color wheel (range 65536) along the length
        // of the strip (strip.numPixels() steps):
        int      hue   = firstPixelHue + c * 65536L / strip.numPixels();
        uint32_t color = strip.gamma32(strip.ColorHSV(hue)); // hue -> RGB
        strip.setPixelColor(c, color); // Set pixel 'c' to value 'color'
      }
      strip.show();                // Update strip with new contents
      delay(wait);                 // Pause for a moment
      firstPixelHue += 65536 / 90; // One cycle of color wheel over 90 frames
    }
  }
}
```

Before running the code, change `LED_PIN` to A3.

```
#include &lt;Adafruit_NeoPixel.h&gt;
#ifdef __AVR__
 #include &lt;avr/power.h&gt; // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1:
//#define LED_PIN    6
  #define LED_PIN    A3
```

Then, upload the sketch to your board. You'll see the attached NeoPixel strip go through the strand test. The strand test includes the color wipe animation in red, green and blue, the theater chase animation in white, red and blue, the rainbow swirl animation and the rainbow theater chase animation.



Arduino Docs

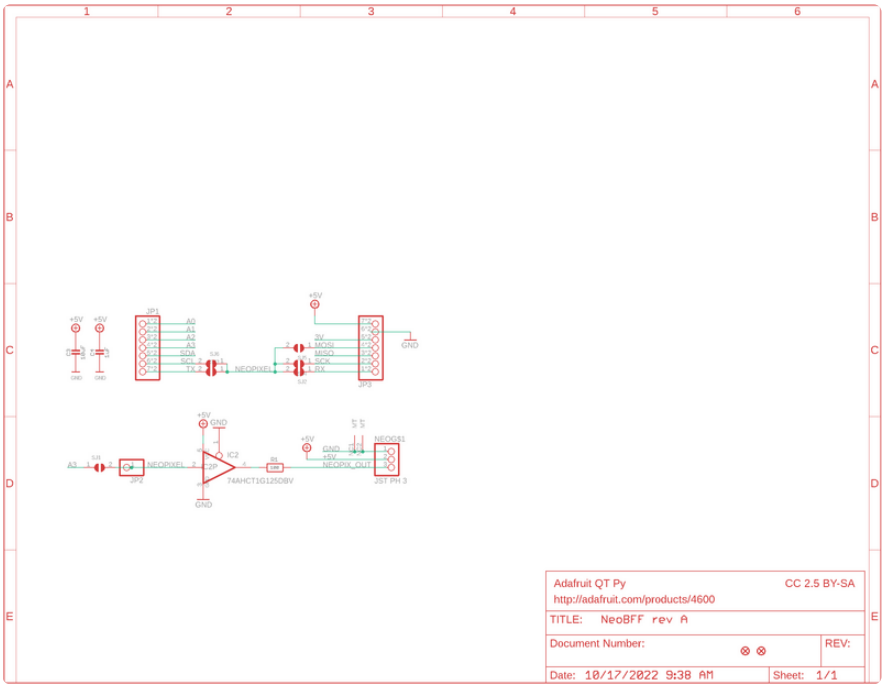
Arduino Docs ()

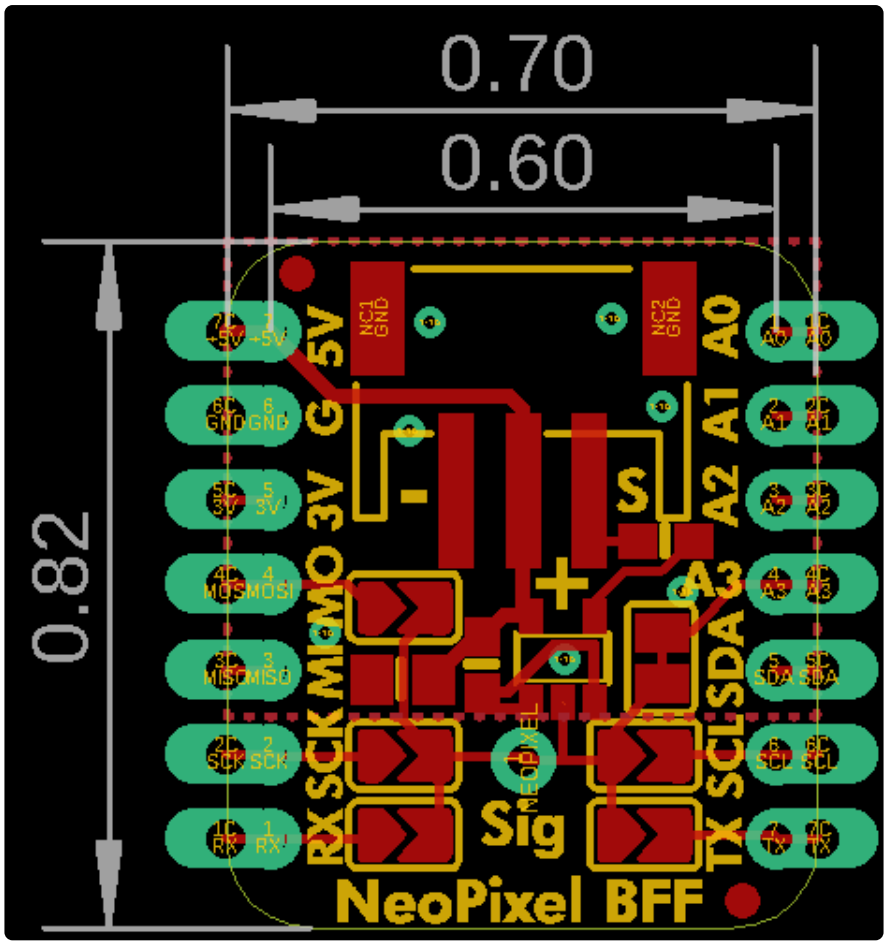
Downloads

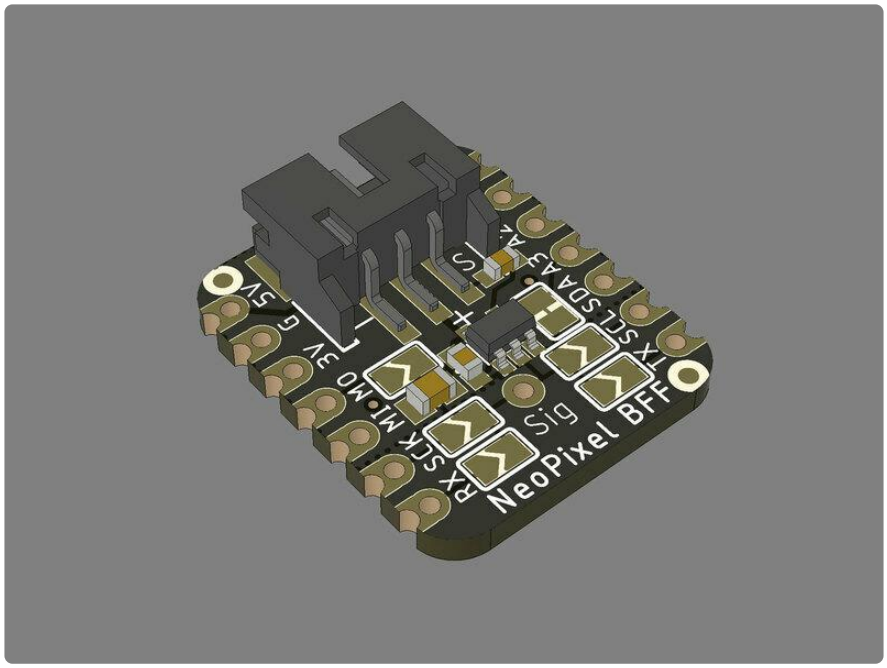
Files

- 74AHC125 single-gate level shifter datasheet ()
- EagleCAD PCB files on GitHub ()
- 3D Models on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic and Fab Print




0.70
0.60
0.82
NC1
GND
NC2
GND
5V
G
3V
MO
SCK
MI
RX
TX
SCL
SDA
A3
A2
A1
A0
Sig
NeoPixel BFF


NeoPixel BFF