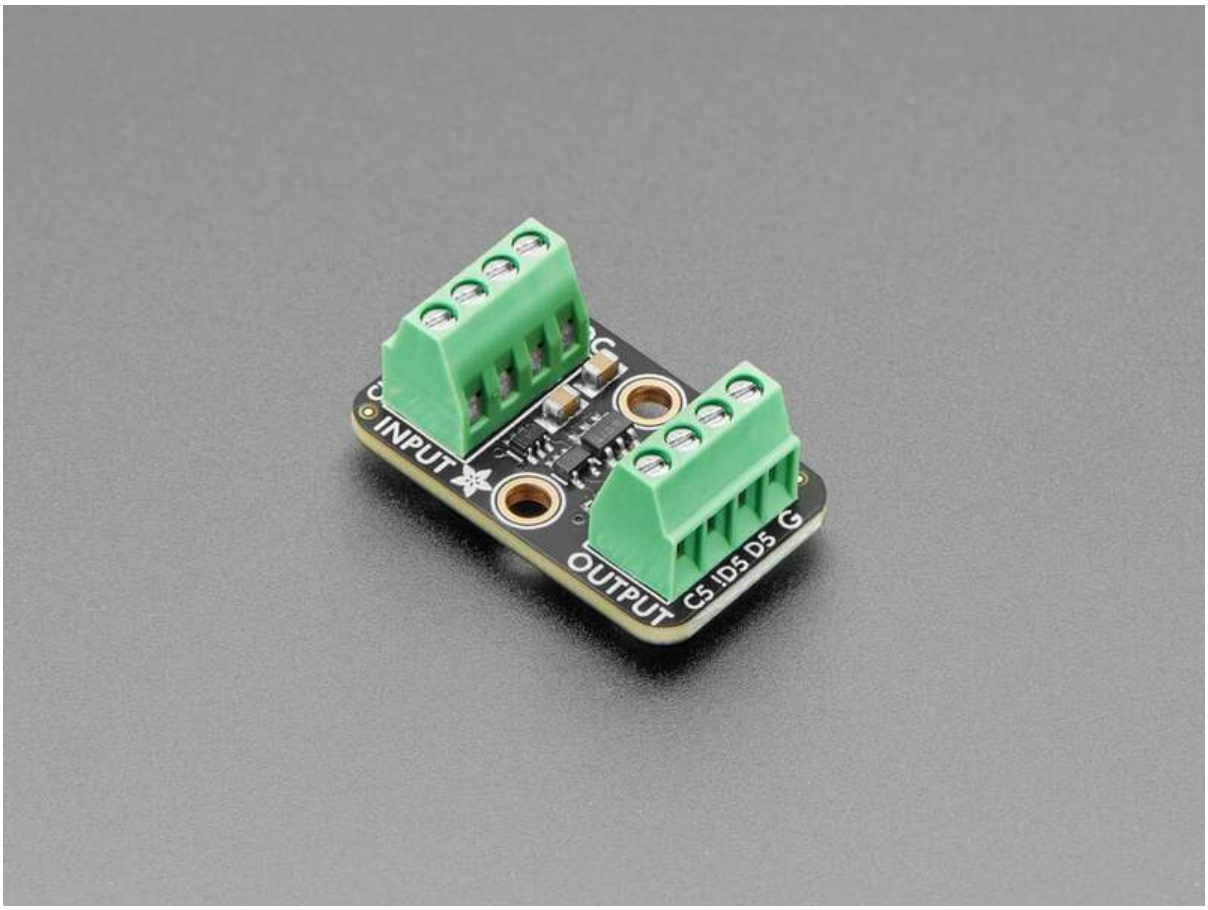
Adafruit Pixel Shifter

Created by Liz Clark



https://learn.adafruit.com/adafruit-pixel-shifter

Last updated on 2024-12-30 12:39:17 PM EST

Table of Contents

Overview 3

Pinouts 5

- Power Pins
- Input Logic
- Output Logic
- NeoPixel and NeoPixel Jumper

CircuitPython and Python 6

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of NeoPixel Library
- CircuitPython Usage
- Python Usage
- Example Code

Python Docs 10

Arduino 10

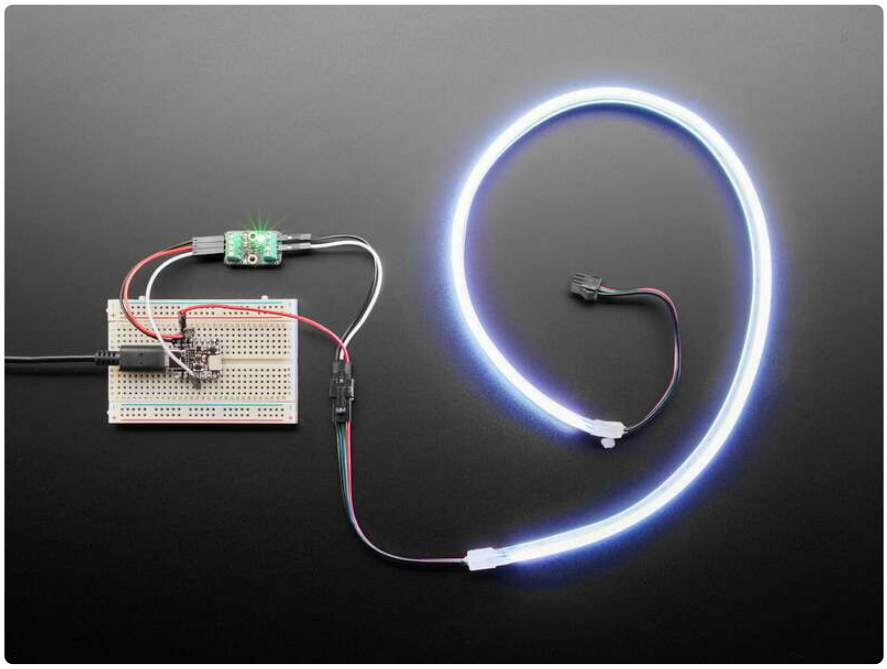
- Wiring
- Library Installation
- Example Code

Arduino Docs 13

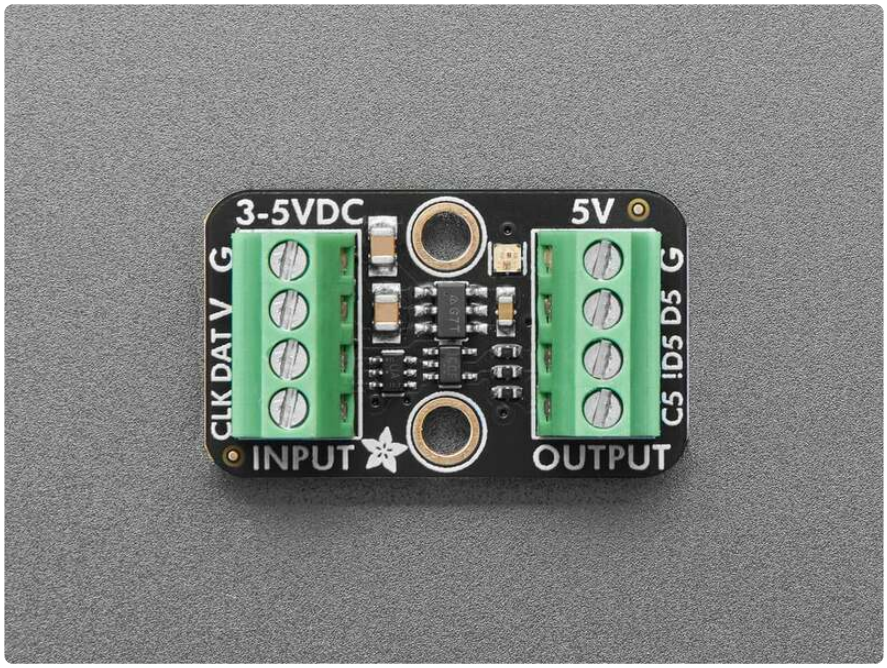
Downloads 13

- Files
- Schematic and Fab Print

Overview

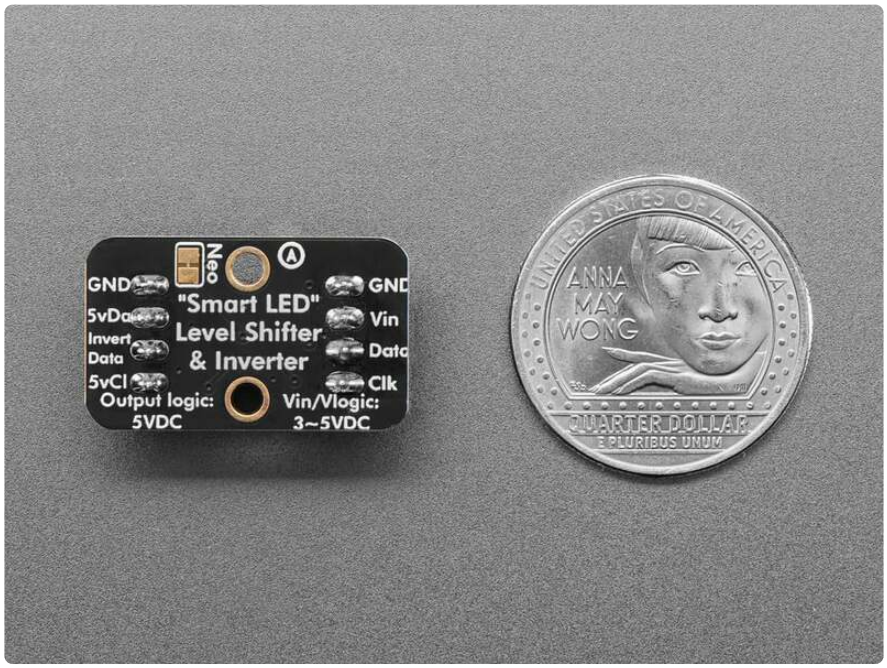


We've been stocking WS2811-n-friends (http://adafru.it/1260) for a long time, enough to see many iterations and versions of the "one-wire-control" addressable LED pixel. We call them NeoPixels, since the part number itself can change quite a bit, but all have the same idea: send color data to lights and they'll change on their own without having to constantly PWM three or four diodes. Despite this simplicity, we've seen folks struggle with them because of voltage level expectations: some NeoPixel-compatible pixels are very picky and want 5V logic level. Without the right voltage, you get flickering or weird behavior. There's also some funky variants like the TM1814 that want inverted signal levels.

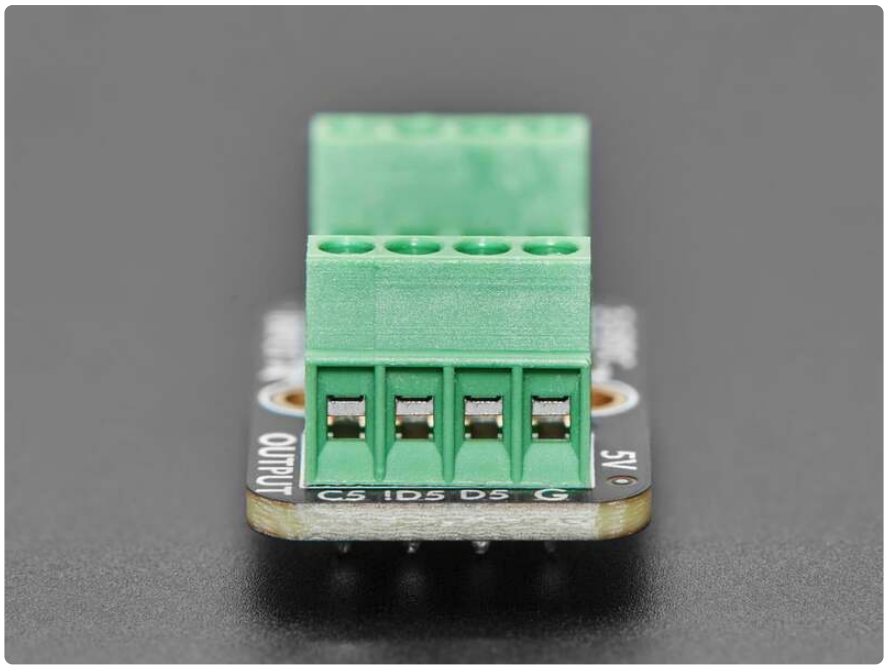


To make wiring easy, so you don't need a separate chip or breadboard, we made this **Adafruit Pixel Shifter** which can be easily wired in-line to your LED strips or grids or any other shape configuration they come in.

On-board is a tiny 5V voltage generator, two shifters and one inverter. Provide it 3-5V power (it only needs a few milliamps) and it will shift up to two signal lines, with one of them also available as an inverted output. All pins are available on 0.1" terminal blocks. Use a small flathead screw to attach your solid or stranded core wire, 18AWG to 26AWG.

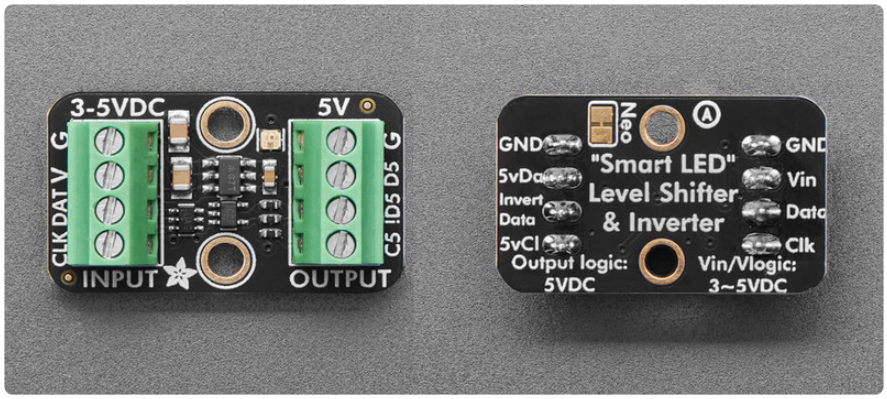


It's a simple, low-cost, and solder-free way to make your finicky LED setups work with 3.3V logic chips like the RP2040, ESP32 / ESP8266, STM32, SAMD, Raspberry Pi or other modern 3.3V-logic devices. You can also use it for other 5V-logic-needing IC's that require one or two inputs. The onboard 74HCT2G34 shifter can easily handle 10MHz signals, most LED strips use a signal frequency of about 800KHz. A perfect companion to WLED!



You can also use this board as a quick signal debugger: there's a single tiny NeoPixel on the Data line, so you can verify that you're getting valid signal on that pin. If you aren't using a NeoPixel protocol, or want to keep it dark, disable it by cutting the trace on the back.

Pinouts



Power Pins

- **V** - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V. It can be powered from either 3V to 5V.
- **G** - common ground for power and logic.

Input Logic

- **DAT** - This is the data line input from your microcontroller to your addressable RGB LED.

- **CLK** - This is the clock line input from your microcontroller to your addressable RGB LED. Some RGB LED variants such as Dotstars (ex: SK9822) utilize a clock signal.

Output Logic

- **D5** - This is the 5V level shifted data line output to your addressable RGB LED.
- **!D5** - This is the inverted 5V level shifted data line output to your addressable RGB LED. This output should be used for funky variants like the TM1814 that want inverted signal levels.
- **C5** - This is the 5V level shifted clock line output to your addressable RGB LED. Some RGB LED variants such as Dotstars (ex: SK9822) utilize a clock signal.

NeoPixel and NeoPixel Jumper

- **NeoPixel LED** - On the front of the board, to the left of the output terminal block, is the NeoPixel LED. This LED is connected to the data line, so you can verify that you're getting valid signal on that pin to check against the output you are getting with your connected addressable RGB LEDs.
- **Neo** - On the back of the board, to the left of the top mounting hole, is the NeoPixel LED jumper. It is labeled **Neo** and outlined in white on the board silk. If you do not want to power the onboard NeoPixel LED, you can cut this jumper to disable it.

CircuitPython and Python

It's easy to use the **Pixel Shifter** with CircuitPython and the Adafruit_CircuitPython_NeoPixel (https://adafru.it/yew) module. This module allows you to easily write Python code for NeoPixels.

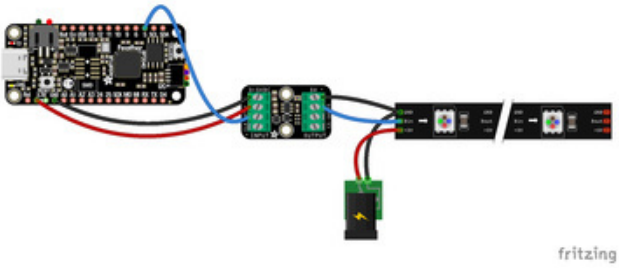
You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN). You should note though that many single board computers (SBCs) don't have NeoPixel/other addressable RGB LED support due to the precision timing required to send data.

Many single board computers don't have NeoPixel/addressable RGB LED support due to the precision timing required to send data.

[Adafruit NeoPixel Digital RGB LED Strip - Black 30 LED]
You thought it couldn't get better than our world-famous 32-LED-per-meter Digital LED strip but we will prove you wrong! These...
https://www.adafruit.com/product/1460

CircuitPython Microcontroller Wiring

Here is how you'll wire the breakout to a Feather RP2040 and NeoPixel strip:



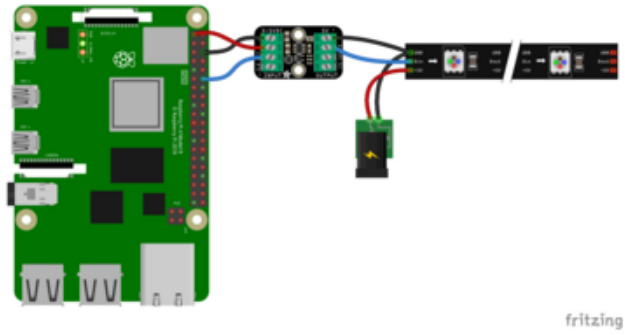
Feather 3.3V to **breakout V (red wire)**
Feather GND to **breakout G (black wire)**
Feather D5 to **breakout DAT (blue wire)**
Breakout G to **NeoPixel GND (black wire)**
Breakout D5 to **NeoPixel DATA IN (blue wire)**
External Power 5V to **NeoPixel PWR (red wire)**
External Power GND to **NeoPixel GND (black wire)**

Python Computer Wiring

Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the breakout and a NeoPixel strip:

On the Raspberry Pi, NeoPixels must be connected to GPIO10, GPIO12, GPIO18 or GPIO21 to work!



Pi 3.3V to **breakout V (red wire)**
Pi GND to **breakout G (black wire)**
Pi GPIO18 to **breakout DAT (blue wire)**
Breakout G to **NeoPixel GND (black wire)**
Breakout D5 to **NeoPixel DATA IN (blue wire)**
External Power 5V to **NeoPixel PWR (red wire)**
External Power GND to **NeoPixel GND (black wire)**

Python Installation of NeoPixel Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `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!

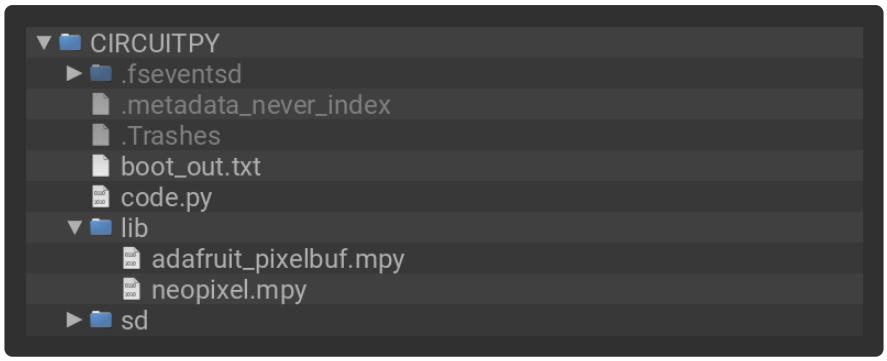
CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_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 **code.py** file to your **CIRCUITPY** drive.

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

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



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
```

Example Code

If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

If you are using LEDs that require a different color order than the GRB and RGB that are available in the neopixel library, you can pass along a string instead, e.g.: `'pixels = neopixel.NeoPixel(pixel_pin, num_pixels, pixel_order = "RBG"'` This is what works for these cafe lights. You can pass along any order you need, `BGR`, `GRB`, `RBG`, etc.

```
# SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
from rainbowio import colorwheel
import neopixel

num_pixels = 1
# pylint: disable=simplifiable-condition
# check to see if its a raspberry pi
if "CE0" and "CE1" in dir(board):  # pi only zone
    pixel_pin = board.D18
# otherwise assume a microcontroller
else:
    pixel_pin = board.D5

pixels = neopixel.NeoPixel(pixel_pin, num_pixels)

color_offset = 0
```

```
while True:
    for i in range(num_pixels):
        rc_index = (i * 256 // num_pixels) + color_offset
        pixels[i] = colorwheel(rc_index & 255)
    pixels.show()
    color_offset += 1
    time.sleep(0.01)
```

The code has a check to determine if you are running the code on a Raspberry Pi or not. If you are, the NeoPixel pin is set as **GPIO18**. Otherwise, the NeoPixel pin is set as **D5**. Once the loop starts, you'll see your NeoPixel cycle through the colors of the rainbow.

Python Docs

Python Docs (https://adafru.it/18gA)

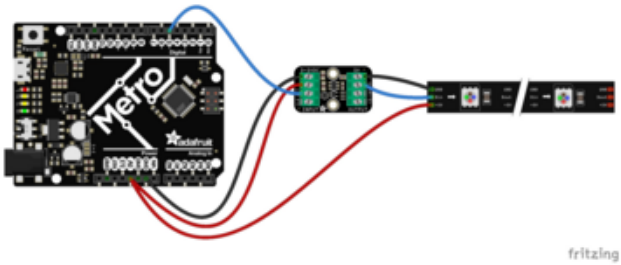
Arduino

Using the Pixel Shifter with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_NeoPixel (https://adafru.it/aZU) library and running the provided example code.

Adafruit NeoPixel Digital RGB LED Strip - Black 30 LED
You thought it couldn't get better than our world-famous 32-LED-per-meter Digital LED strip but we will prove you wrong! These...
https://www.adafruit.com/product/1460

Wiring

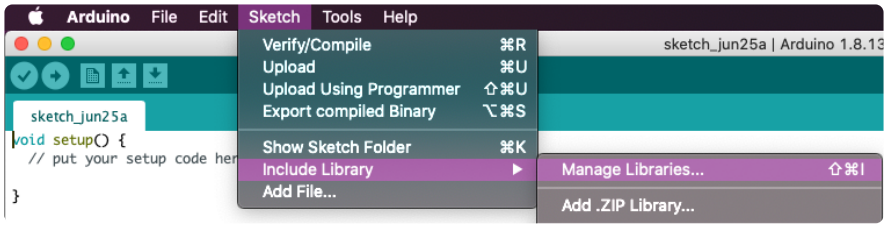
Here is how you'll wire the breakout to an Adafruit Metro and NeoPixel strip:



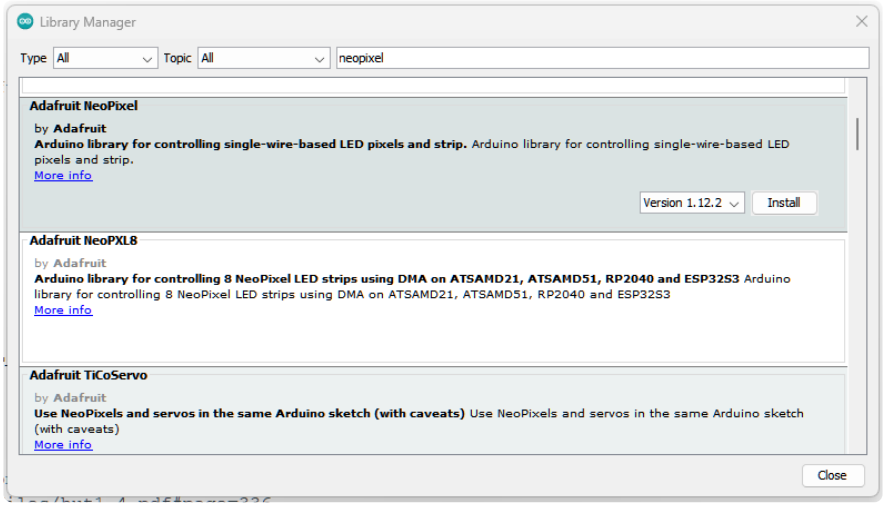
Metro 5V to **breakout V (red wire)**
Metro GND to **breakout G (black wire)**
Metro D5 to **breakout DAT (blue wire)**
Metro 5V to **NeoPixel PWR (red wire)**
Breakout G to **NeoPixel GND (black wire)**
Breakout D5 to **NeoPixel DATA IN (blue wire)**

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

```
// SPDX-FileCopyrightText: 2024 Limor Fried for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#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 1

// 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

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)
}
void loop() {

  rainbow(10);             // Flowing rainbow cycle along the whole strip
}

// 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
  }
}
```

Upload the sketch to your board. You'll see your NeoPixel strip cycle through a rainbow swirl animation.

Arduino Docs

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

Downloads

Files

- EagleCAD PCB Files on GitHub (https://adafru.it/1a9H)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a9I)

Schematic and Fab Print