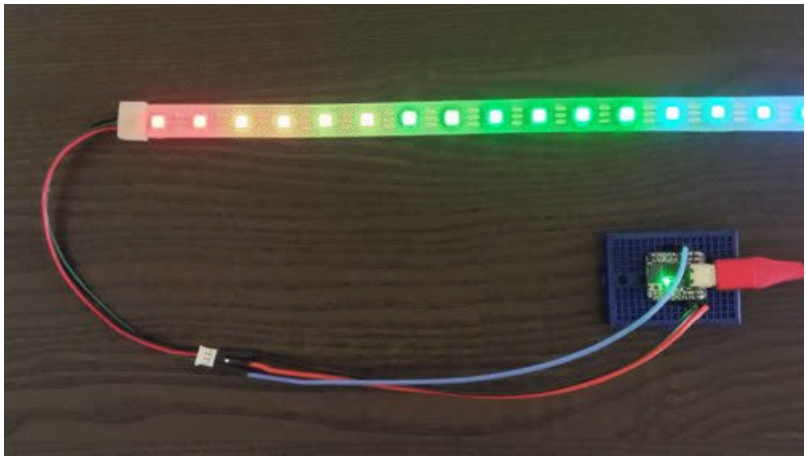
adafruit learning system

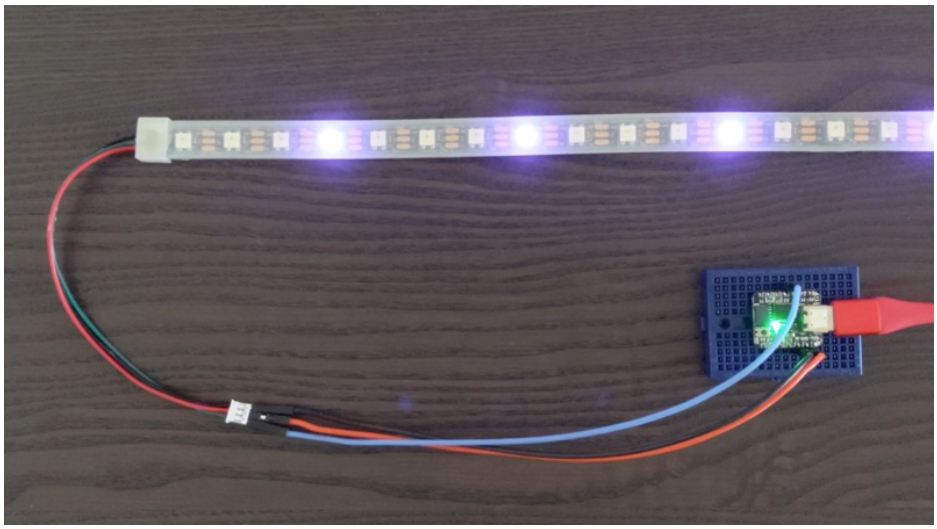
Adafruit QT Py and NeoPixel LEDs

Created by Kattni Rembor



Last updated on 2020-10-14 01:59:42 PM EDT

Overview



The QT Py is a tiny little development board capable of big things. Big things are great, but LEDs are the best. QT Py plus LEDs is even better! This guide will help you get started using NeoPixel LEDs with your QT Py and QT Py Haxpress (A QT Py with a soldered-on flash chip). You'll learn how to wire up the board and LEDs, and how to get the software set up.

For the QT Py, you'll be provided with an example that includes some simple animations, such as blink and chase, with basic customisations so you can modify them to fit your needs.

If you've soldered up a QT Py Haxpress, you have the option to use it with the CircuitPython LED Animation library. This guide includes a page on how to use the LED Animation library with your QT Py Haxpress, with three examples to get you started, as well as links to details and more animations.

Let's get blinking!

Parts

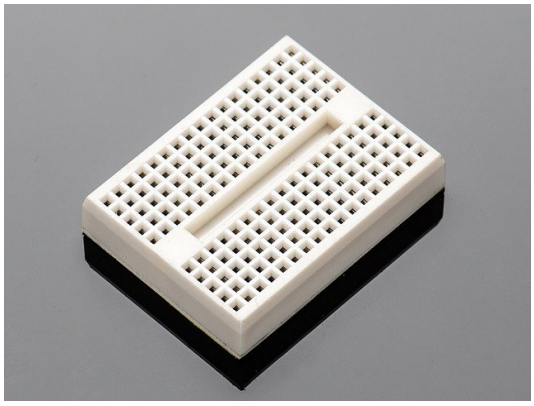
Your browser does not support the video tag. Adafruit QT Py - SAMD21 Dev Board with STEMMA QT

OUT OF STOCK

Out Of Stock

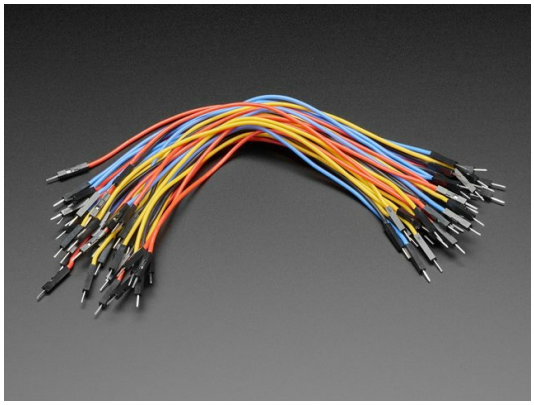
Adafruit NeoPixel LED Strip with 3-pin JST Connector

$12.50
IN STOCK
Add To Cart



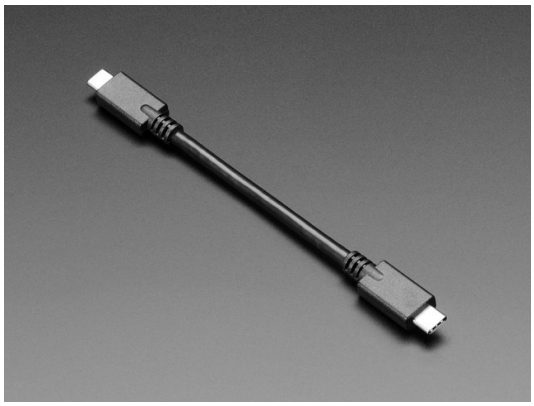
Tiny breadboard

$4.00
IN STOCK
Add To Cart



Premium Silicone Covered Male-Male Jumper Wires - 200mm x 40

$9.95
IN STOCK
Add To Cart



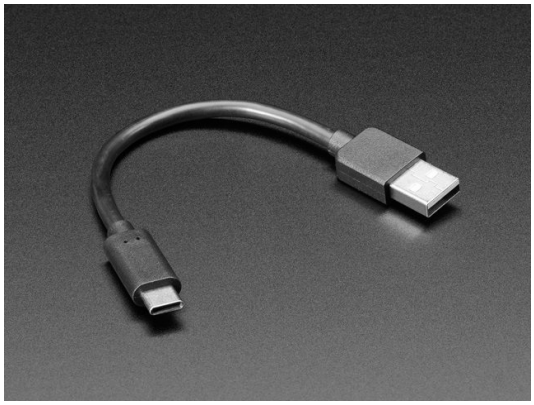
USB C to USB C Cable - USB 3.1 Gen 4 with E-Mark - 6" long

$8.50
IN STOCK
Add To Cart

USB C to USB C Cable - USB 3.1 Gen 4 with E-Mark - 1 meter long

OUT OF STOCK

Out Of Stock

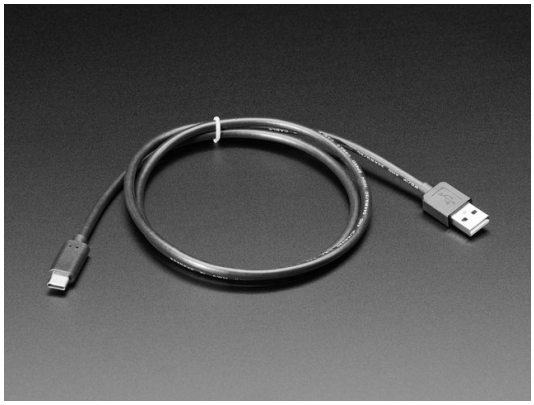


USB Type A to Type C Cable - 6" long

$2.95

IN STOCK

Add To Cart



USB Type A to Type C Cable - approx 1 meter / 3 ft long

OUT OF STOCK

Out Of Stock

QT Py NeoPixel Wiring

The first thing you need to do is connect NeoPixels smart color LEDs to your QT Py. NeoPixels come in many form factors (https://adafru.it/dYn) but connecting them is basically the same. There are three pins: **ground**, **5v power** and **data**. You'll connect these to the appropriate pins on your QT Py.

For an in depth look at NeoPixels, check out this guide (https://adafru.it/dhw).

There are options for connecting NeoPixels to the QT Py. You can solder headers to your QT Py and use a breadboard, or you can solder wires directly to your QT Py. It's up to you!

To demonstrate the examples in this guide, I have connected a NeoPixel LED Strip with 3-pin JST Connector (https://adafru.it/Cup) to the QT Py on a tiny breadboard (https://adafru.it/kft) using Silicone Covered Male-Male Jumper Wires (https://adafru.it/NXB). You can use any form of NeoPixels and connect them however you like, but if you want the examples to run without modification, ensure that you've connected them to the pin listed below.

If you're using a form of NeoPixels that are chainable, e.g. a bare strip or a ring, there will be DATA IN and DATA OUT. You must connect your QT Py to DATA IN!

Wiring

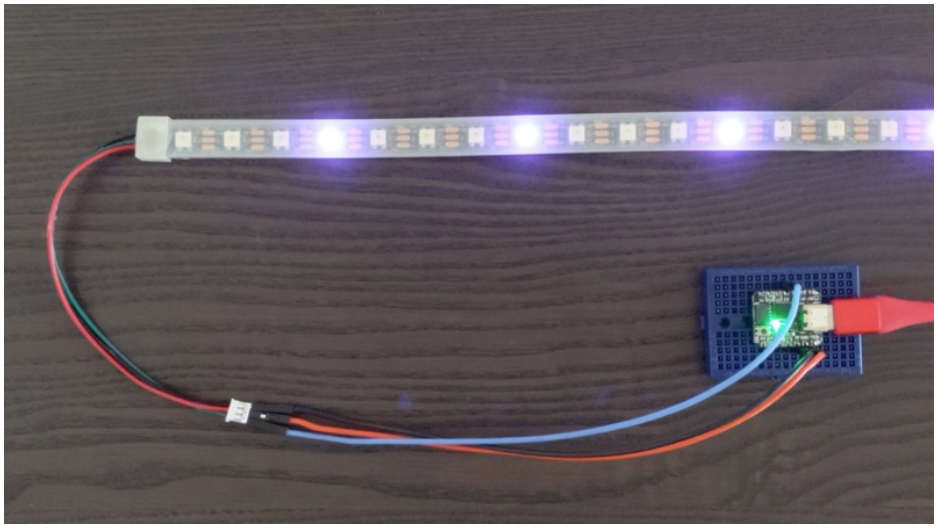
Connect the NeoPixels to the QT Py as follows:

- **QT Py A3** to **NeoPixel data / DIN (data in)**
- **QT Py GND** to **NeoPixel ground**
- **QT Py 5V** to **NeoPixel power (5V)**

You must connect your NeoPIxels to 5V power, on the QT Py or externally, for the LEDs to work as expected. The example may not run properly if the LEDs are connected to 3V power!

Once you've got everything connected, it's time to get coding and light it up!

Basic Animations for QT Py



CircuitPython makes using NeoPixels with a QT Py super simple. Simply load the necessary libraries and save the example to your board. This example includes some basic animations to use with a QT Py and NeoPixels, such as blinking, theatre chase, and color wipe. Each animation has some customisations you can do to fit with your project. Let's take a look!

Step 1 - Install CircuitPython

This guide requires CircuitPython be installed, click the button below to learn how to do that and install the latest version of CircuitPython

https://adafru.it/O2D

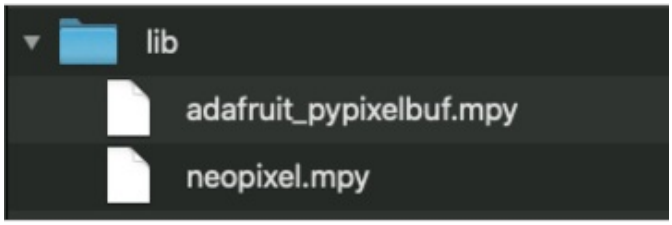
https://adafru.it/O2D

CircuitPython NeoPixel Library Installation

First make sure you are running the latest version of Adafruit CircuitPython for the QT Py (https://adafru.it/NCB).

You'll need to install the Adafruit CircuitPython NeoPixel (https://adafru.it/yew) library and its dependency on your QT Py.

Next you'll need to install the necessary libraries to use the LEDs -- carefully follow the steps to find and install these libraries from Adafruit's CircuitPython Library Bundle (https://adafru.it/ENC). Our CircuitPython starter guide has a great page on how to install libraries from the bundle (https://adafru.it/ABU).



You'll want to manually install the following libraries by copying the files to the **lib** folder on your **CIRCUITPY** drive:

- **adafruit_pypixelbuf.mpy**
- **neopixel.mpy**

Before continuing make sure your board's **lib** folder or root filesystem has the **neopixel.mpy**, and **adafruit_pypixelbuf.mpy** files copied over.

Light It Up!

Save the following example to your **CIRCUITPY** drive as **code.py**:

```
"""Basic NeoPixel LED animations for the QT Py."""
import time
import board
import neopixel
import adafruit_pypixelbuf

# Update this to match the pin to which you connected the NeoPixels
pixel_pin = board.A3
# Update this to match the number of NeoPixels connected
num_pixels = 30

pixels = neopixel.NeoPixel(pixel_pin, num_pixels, auto_write=False)
# Set to 0-1 to change the brightness of the NeoPixels
pixels.brightness = 0.2


def blink(color, wait):
    """Blink animation. Blinks all pixels."""
    pixels.fill(color)
    pixels.show()
    time.sleep(wait)
    pixels.fill((0, 0, 0))
    pixels.show()
    time.sleep(wait)


def chase(color, spacing=3, iteration_step=1):
    """Theatre chase animation. Chases across all pixels."""
```

```
    if spacing < 2:
        raise ValueError("Spacing must be greater than 1 to show chase pattern.")

    # Use modulo division to create the spacing between pixels.
    chase_pixel = iteration_step % spacing
    # Loop over pixels and turn on expected pixels to provided color.
    for pixel in range(0, len(pixels), spacing):
        # If the pixel is outside the total pixel range, break.
        if pixel + chase_pixel > len(pixels) - 1:
            break
        pixels[pixel + chase_pixel] = color
    pixels.show()

    # Loop over pixels and turn off expected pixels.
    for pixel in range(0, len(pixels), spacing):
        # If the pixel is outside the total pixel range, break.
        if pixel + chase_pixel > len(pixels) - 1:
            break
        pixels[pixel + chase_pixel] = (0, 0, 0)


def color_wipe(color, wait):
    """Color wipe animation. Wipes across all pixels."""
    for pixel in range(num_pixels):
        pixels[pixel] = color
        time.sleep(wait)
        pixels.show()
    time.sleep(0.5)


def rainbow_cycle(wait):
    """Rainbow cycle animation. Cycles across all pixels."""
    for color_index in range(255):
        for pixel in range(num_pixels):
            pixel_index = (pixel * 256 // num_pixels) + color_index
            pixels[pixel] = adafruit_pypixelbuf.colorwheel(pixel_index & 255)
        pixels.show()
        time.sleep(wait)


RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)

while True:
    # Blink 5 times. Increase or decrease the range for more or less blinking.
    for blinks in range(5):
        blink(RED, 0.5)  # Increase number to slow down blinking, decrease to speed up.

    # Chase. Increase or decrease the range for longer or shorter chase animation.
    for step in range(50):
        chase(PURPLE, spacing=4, iteration_step=step)
        time.sleep(0.05)

    # Fill all pixels.
    pixels.fill(RED)
    pixels.show()
```

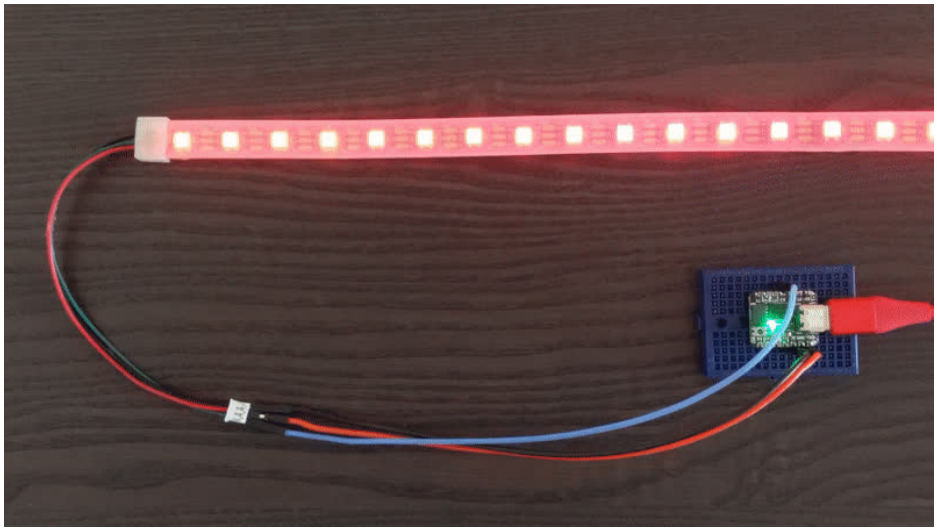
```
    # Increase or decrease the time to change the speed of the solid color change in seconds.
    time.sleep(0.5)
    pixels.fill(GREEN)
    pixels.show()
    time.sleep(0.5)
    pixels.fill(BLUE)
    pixels.show()
    time.sleep(0.5)

    # Color wipe.
    color_wipe(YELLOW, 0.01)  # Increase the number to slow down the color chase.
    color_wipe(CYAN, 0.01)
    color_wipe(PURPLE, 0.01)

    # Rainbow cycle.
    rainbow_cycle(0)  # Increase the number to slow down the rainbow.
```

Your LEDs should begin blinking red! The blinking will be followed by a purple chase animation, three solid colors (red, green and blue), three color wipes (yellow, cyan and purple), and a rainbow cycle. Each of the animations is simple to use. Let's take a look!

Blink

The blink animation turns on the LEDs for a specified period of time, and turns them off for the same amount of time. To use blink, specify a `color`, and the `wait` time interval you'd like to use.

```
[...]
    for blinks in range(5):
        blink(RED, 0.5)
```



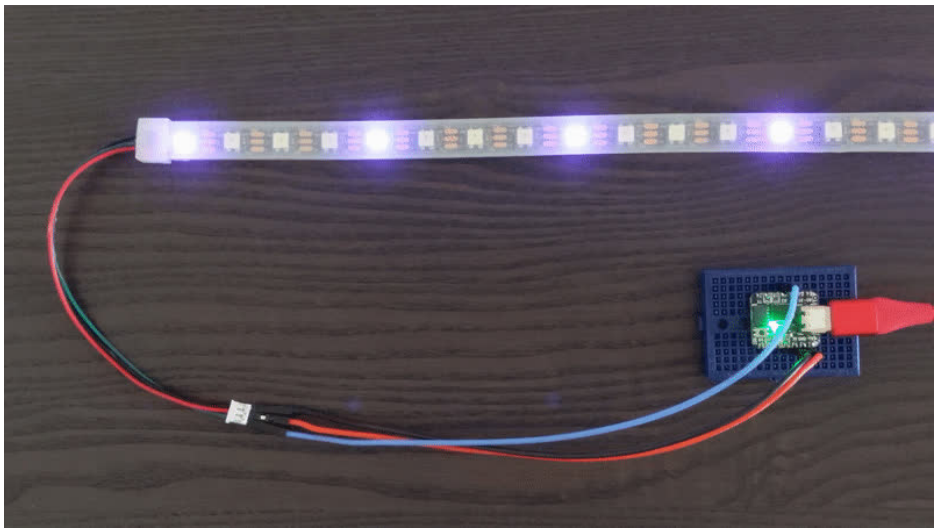
If you use `blink()` on its own, it will blink only one time. So, the example includes code to blink it for a a specified number of times - in this case 5 times. If you want to increase or decrease the number of blinks, change `5` on the line `for blinks in range(5):` to a larger or smaller number.

To change the blinking itself, you have the option to specify a color and a time interval. For example, try changing `blink(red, 0.5)` to `blink(blue, 1)`. Slower blue blinking!

Chase

The chase animation lights up a single LED spaced every x-number of pixels and creates a theatre chase animation. To use chase, specify a `color`, `spacing`, and `iteration step`.

```
[...]
    for step in range(50):
        chase(PURPLE, spacing=4, iteration_step=step)
        time.sleep(0.05)
```



For chase to work, the `iteration_step` should be the `step` in a range. Increase the number in `range(number)` to increase the length of time the chase animation runs, decrease it to shorten the time it runs.

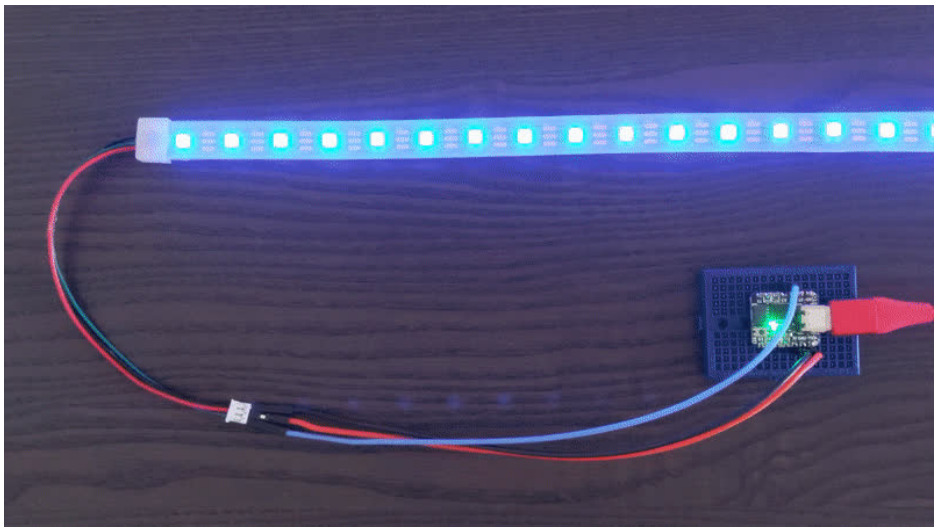
To change the chase itself, you have the option to specify a color and the spacing. For example, try changing `chase(PURPLE, spacing=4, iteration_step=step)` to `chase(GREEN, spacing=2, iteration_step=step)`. Tighter green chase!

If you use `chase()` without a `time.sleep()`, the animation runs very quickly. So, there is a `time.sleep(0.05)` included to slow it down a bit. You can increase this number to slow down the chase animation further, or set it to `0` for super fast chasing!

Color Wipe

The color wipe fills the LEDs a specified color beginning with the first pixel and wiping to the last pixel. To use color wipe, specify a `color` and `wait` to determine speed.

```
[...]
    color_wipe(YELLOW, 0.01)
```

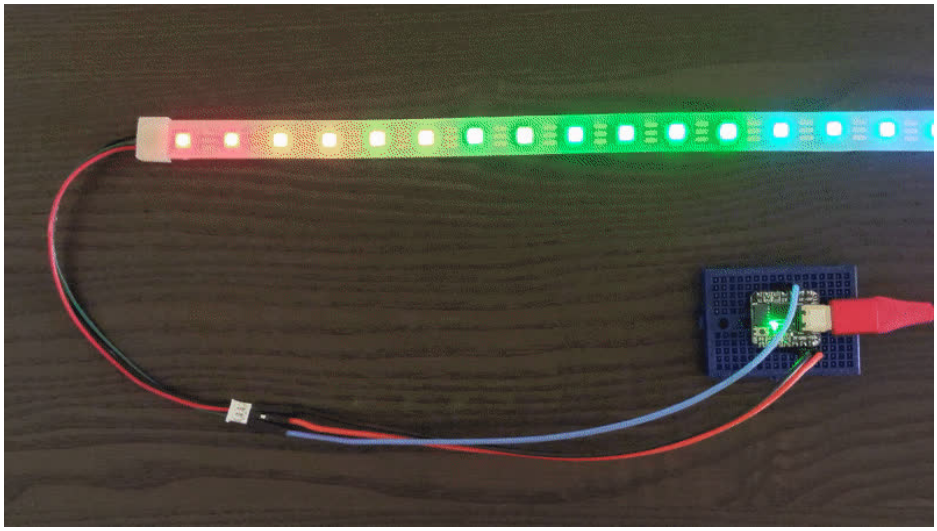


Color wipe is used standalone. To change it, you have the option to specify a color and the speed. For example, try changing `color_wipe(YELLOW, 0.01)` to `color_wipe(BLUE, 0.2)`. Slower blue color wipe!

Rainbow Cycle

The rainbow cycle spreads a rainbow across all pixels and then cycles it across all pixels for once. To use, specify a `wait` time to determine the speed.

```
[...]
    rainbow_cycle(0)
```



The rainbow cycle is used standalone. To change it, you have the option to specify the speed. For example, try changing the `rainbow_cycle(0)` to `rainbow_cycle(0.2)`. Slower rainbow cycle!

To cycle the rainbow more than once, you can use range in the same way you did with `blink`. Add the following code to cycle the rainbow three times. Change `3` to any number to cycle that number of times.

```
[...]
    for cycles in range(3):
        rainbow_cycle(0)
```

That's all there is to using basic animations with QT Py and NeoPixels!

LED Animations with QT Py Haxpress

The LED Animation library does not work with the base QT Py board. You MUST solder a flash chip to the bottom of your QT Py to create a QT Py Haxpress to use the LED Animation library. This section assumes you have soldered a flash chip to your QT Py and loaded the appropriate version of CircuitPython: https://learn.adafruit.com/adafruit-qt-py/circuitpython

The CircuitPython LED Animation library makes animating your LEDs super simple with a wide variety of easily customisable animations. Most of the available animations run on the SAMD21 microcontroller (https://adafru.it/O2c) found on the QT Py. This page provides examples of using blink, chase and comet animations. You can use any of the other valid animations in the same way.

For more details about these animations and the others available in the LED Animation library, check out the CircuitPython LED Animations guide (https://adafru.it/NKa).

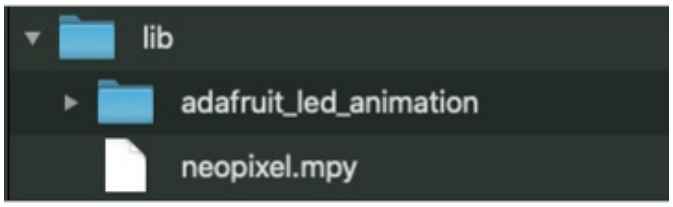
CircuitPython LED Animation Library Installation

You'll need to install the Adafruit CircuitPython LED Animation (https://adafru.it/O2d) library and the NeoPixel library on your QT Py.

For more details on getting started with this library, check out the Import and Setup page of the CircuitPython LED Animations guide (https://adafru.it/LfT).

First make sure you are running the latest version of Adafruit CircuitPython for the QT Py Haxpress (https://adafru.it/NCC).

Next you'll need to install the necessary libraries to use the LEDs -- carefully follow the steps to find and install these libraries from Adafruit's CircuitPython Library Bundle (https://adafru.it/ENC). Our CircuitPython starter guide has a great page on how to install libraries from the bundle (https://adafru.it/ABU).



You'll want to manually install the following libraries by copying the files and folders to the **lib** folder on your **CIRCUITPY** drive:

- **adafruit_led_animation**
- **neopixel.mpy**

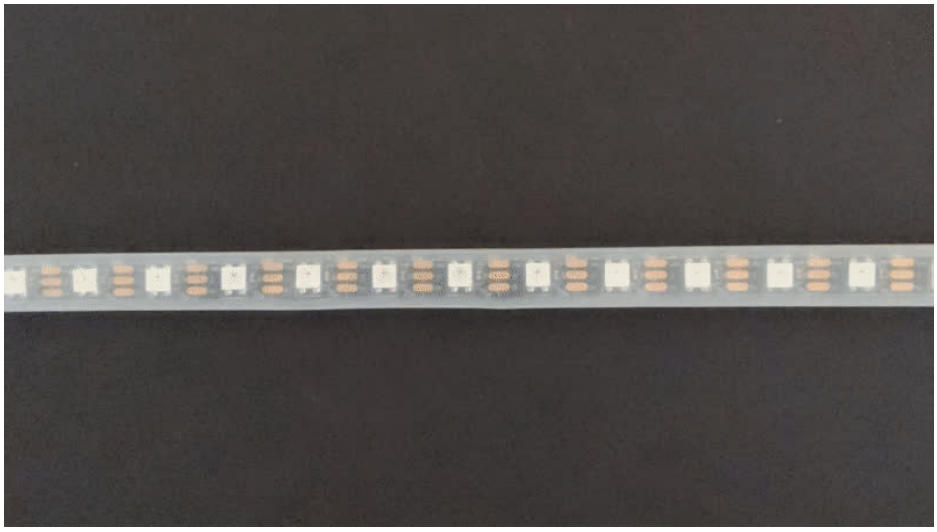
Before continuing make sure your board's **lib** folder or root filesystem has the **adafruit_led_animation**, and **neopixel.mpy** files and folders copied over.

Blink

This is a blinking animation that lights up and turns off all the LEDs at a specified interval.

Save the following example as **code.py** on your **CIRCUITPY** drive:

```
"""
This example blinks the LEDs purple at a 0.5 second interval.

For QT Py Haxpress and a NeoPixel strip. Update pixel_pin and pixel_num to match your wiring if
using a different board or form of NeoPixels.

This example will run on SAMD21 (M0) Express boards (such as Circuit Playground Express or QT Py
Haxpress), but not on SAMD21 non-Express boards (such as QT Py or Trinket).
"""
import board
import neopixel

from adafruit_led_animation.animation.blink import Blink
from adafruit_led_animation.color import PURPLE

# Update to match the pin connected to your NeoPixels
pixel_pin = board.A3
# Update to match the number of NeoPixels you have connected
pixel_num = 30

pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=0.5, auto_write=False)

blink = Blink(pixels, speed=0.5, color=PURPLE)

while True:
    blink.animate()
```



The LEDs should begin to blink purple!

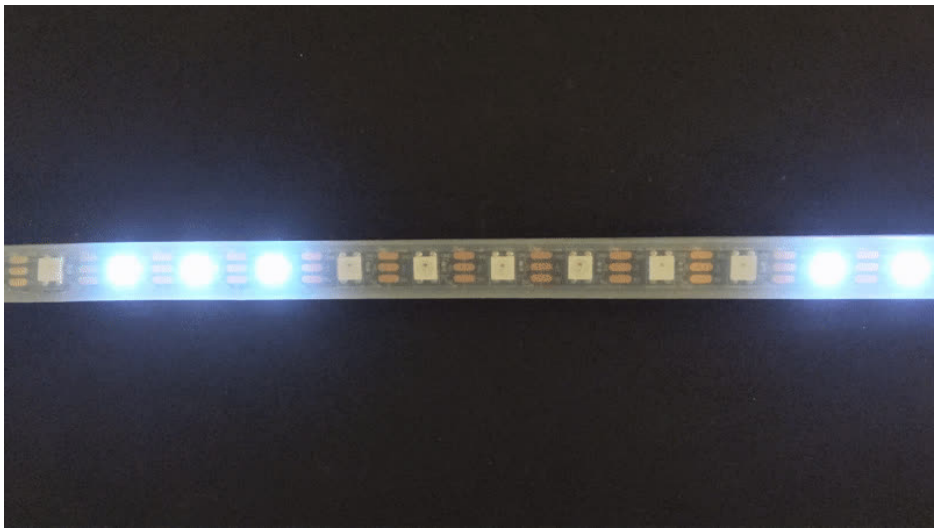
For details on how to customise the blink animation, check out the Blink section of the CircuitPython LED Animations guide (https://adafru.it/O2e).

Chase

This is a theatre chase style animation that lights up blocks of LEDs evenly spaced and chases them along the strip.

Save the following example as **code.py** on your **CIRCUITPY** drive:

```
"""
This example animates a theatre chase style animation in white with a repeated 3 LEDs lit up at a
spacing of six LEDs off.

For QT Py Haxpress and a NeoPixel strip. Update pixel_pin and pixel_num to match your wiring if
using a different board or form of NeoPixels.

This example will run on SAMD21 (M0) Express boards (such as Circuit Playground Express or QT Py
Haxpress), but not on SAMD21 non-Express boards (such as QT Py or Trinket).
"""
import board
import neopixel

from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.color import WHITE

# Update to match the pin connected to your NeoPixels
pixel_pin = board.A3
# Update to match the number of NeoPixels you have connected
pixel_num = 30

pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=0.5, auto_write=False)

chase = Chase(pixels, speed=0.1, size=3, spacing=6, color=WHITE)

while True:
    chase.animate()
```



Groups of three LEDs lit up white should chase along the strip spaced out by six LEDs turned off!

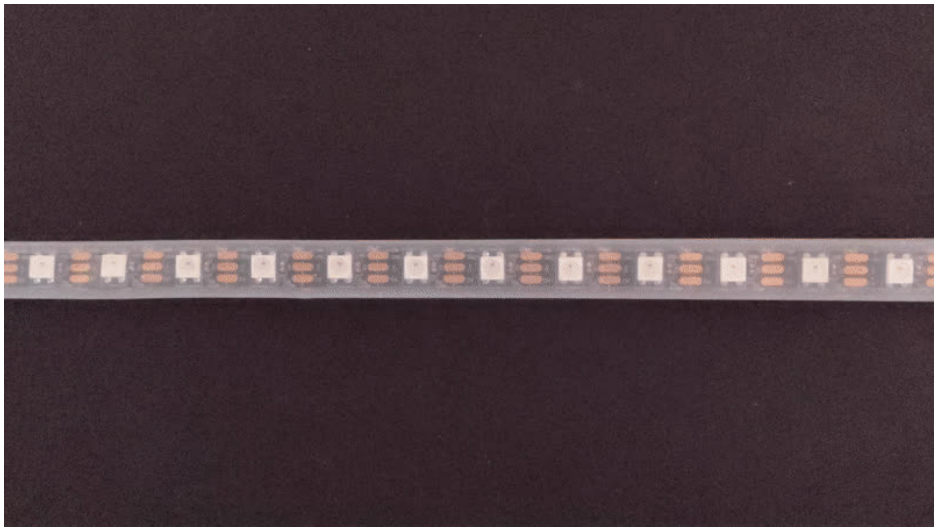
For details on how to customise the chase animation, check out the Chase section of the CircuitPython LED Animations guide (https://adafru.it/O2e).

Comet

This is a comet animation with a bright LED moving along the strip with a progressively dimming tail of a specified size following it.

Save the following example as **code.py** on your **CIRCUITPY** drive:

```
"""
This example animates a jade comet that bounces from end to end of the strip.

For QT Py Haxpress and a NeoPixel strip. Update pixel_pin and pixel_num to match your wiring if
using a different board or form of NeoPixels.

This example will run on SAMD21 (M0) Express boards (such as Circuit Playground Express or QT Py
Haxpress), but not on SAMD21 non-Express boards (such as QT Py or Trinket).
"""
import board
import neopixel

from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.color import JADE

# Update to match the pin connected to your NeoPixels
pixel_pin = board.A3
# Update to match the number of NeoPixels you have connected
pixel_num = 30

pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=0.5, auto_write=False)

comet = Comet(pixels, speed=0.02, color=JADE, tail_length=10, bounce=True)

while True:
    comet.animate()
```



A jade colored comet should begin bouncing back and forth across the strip!

For details on how to customise the comet animation, check out the Comet section of the CircuitPython LED Animations guide (https://adafru.it/O2e).

More Animations

The CircuitPython LED Animation library has many more animations available, most of which will work (https://adafru.it/O2c) on your QT Py Haxpress. For more information on the rest of the features of the LED Animation library, check out the CircuitPython LED Animations guide (https://adafru.it/LZF).