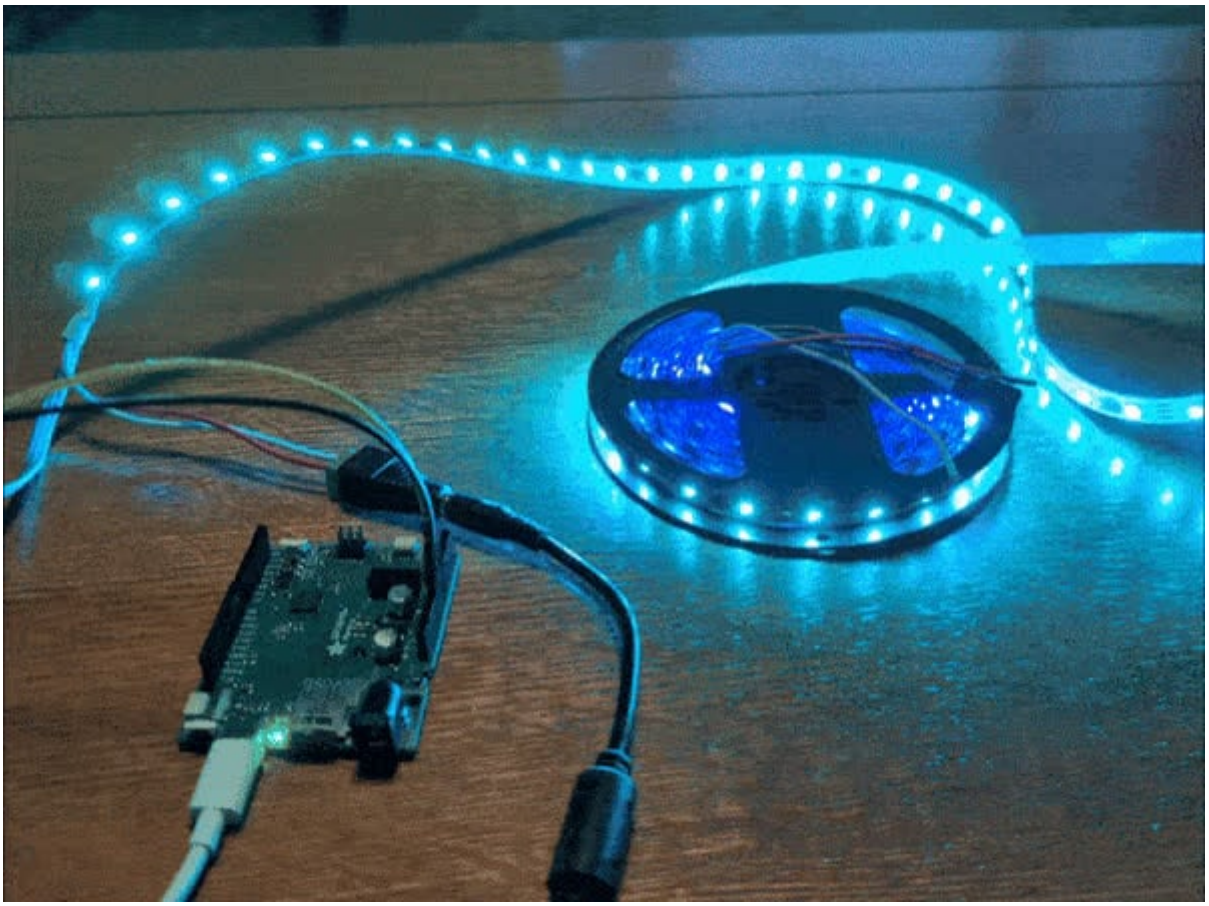
Driving TM1814 addressable LEDs

Created by Jeff Epler



https://learn.adafruit.com/driving-tm1814-addressable-leds

Last updated on 2024-10-23 07:39:09 PM EDT

Table of Contents

Overview 3

- Why require RP2040/RP2350
- adafruit_tm1814 library overview

Wiring & Power 6

- Wiring the TM1814 LED strip
- LED Strip Organization
- LED strip power estimation

CircuitPython Simpletest 8

- CircuitPython simpletest
- Copy the code to your CircuitPython device
- Code Walkthrough & Customization

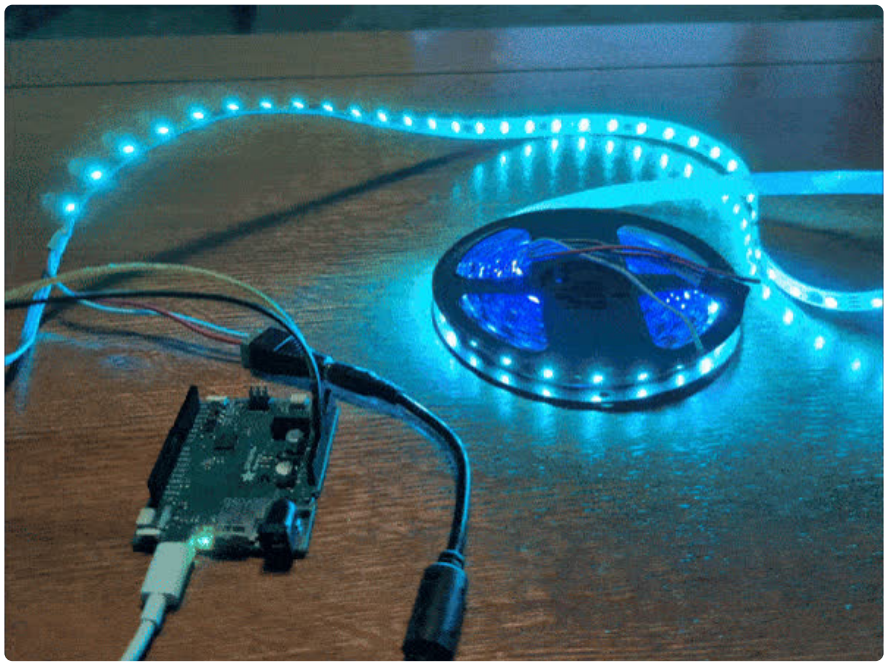
CircuitPython LED Animations 10

- Copy the code to your CircuitPython device
- Wiring the TM1814 LED strip
- Code Walkthrough & Customization

CircuitPython Docs 12

Downloads 12

Overview



TM1814 is a four-channel constant current LED driver. Similar to the NeoPixel's WS2812B, TM1814 uses one data line to control a string of daisy chained LEDs.

However, TM1814 is not compatible with the WS2812B, so it needs different software to drive it. Adafruit publishes Adafruit_CircuitPython_TM1814, a software library for CircuitPython that runs on RP2040 and RP2350 microcontrollers, including the Feather RP2040, Feather RP2350, Raspberry Pi Pico, and Raspberry Pi Pico 2.

Key facts about TM1814:

- TM1814 works with LED supply voltages up to 24V, vs about 5V for NeoPixels. A particular LED strip will have a design voltage, often 12V; refer to the LED strip supplier's details and do not exceed them.
- TM1814 often controls more than one LED package. For instance, a 12V strip is likely to have 3 RGBW LEDs for every 1 TM1814. These 3 LEDs will all have the same color and act as a "single pixel" in your code.
- There is an overall LED current control, selectable from 6.5mA to 38mA in 0.5mA steps, that applies to all TM1814 in a strip
- TM1814 is a standalone chip designed for RGBW designs, while WS281x chips are RGB only
- TM1814 will display a test pattern when not actively driven by a microcontroller
- The waveform is very similar to an inverted NeoPixel waveform, but not exactly

Why require RP2040/RP2350

The RP2 "PIO" peripheral can be used to create high precision waveforms constantly, without any intervention needed from your program's main code. The `background_write` function of CircuitPython's **StateMachine** object made this simple to implement. Doing the same thing on another microcontroller family, like ESP32, would have been more complicated and might have required changes in the CircuitPython core.

`adafruit_tm1814` library overview

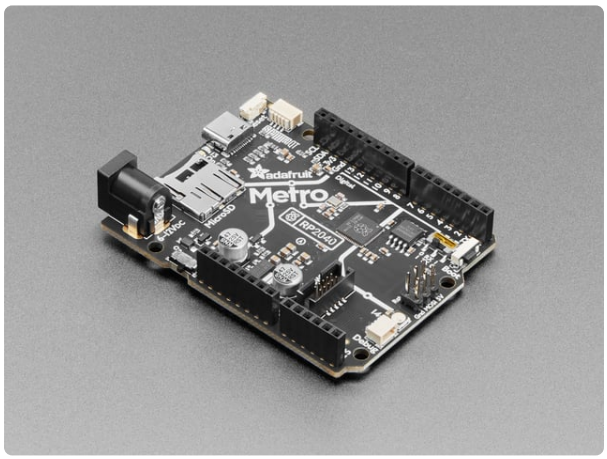
`TM1814PixelBackground` is an `adafruit_pixelbuf (https://adafru.it/1a9i)` object and supports all of that object's methods. So, your code can set all the LEDs with a line like `pixels.fill((255,0,0))` or set individual pixels with code like `pixels[0] = (255,0,0)`. It is also compatible with the extensive set of animations available through the Adafruit LED Animation library (https://adafru.it/LAg).

It also adds functionality specific to TM1814: the `current_control` property. The current is settable, per R/G/B/W channel, in the range of 6.5mA to 38mA.

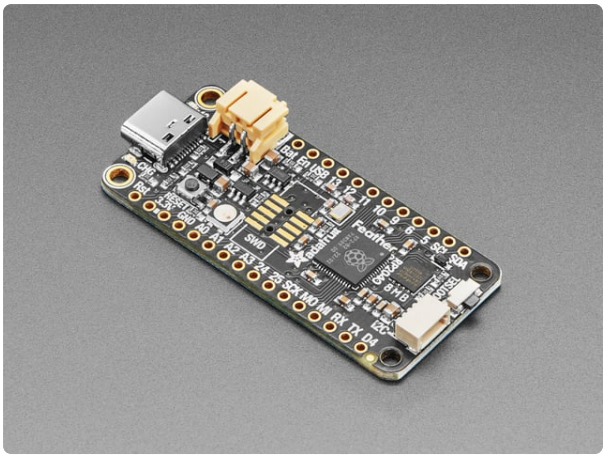
Additionally, the `show()` method exists, but does nothing, because the strip is constantly being updated. For similar reasons, the `auto_write` property exists, but any value assigned to it is ignored, and the property always reads back as `True`.

If you use an inverting signal conditioner between your microcontroller and the TM1814 strip, you can pass `inverted=True` to the constructor to give the signal the correct polarity.

You can check the full documentation for `adafruit_tm1814` via the link in the sidebar.



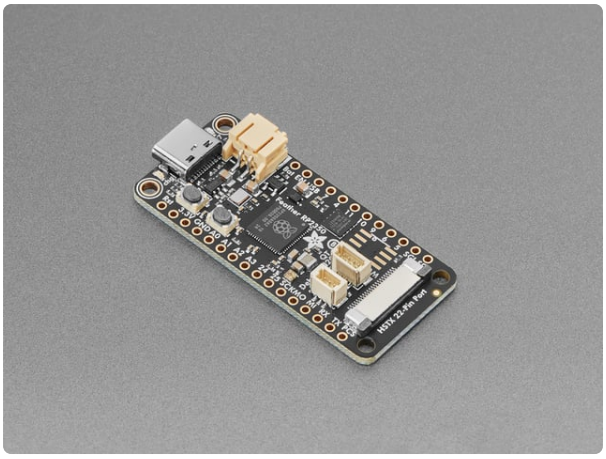
Adafruit Metro RP2040
Choo! Choo! This is the RP2040 Metro Line, making all station stops at "Dual Cortex M0+ mountain", "264K RAM round-about" and "16 Megabytes of Flash...
https://www.adafruit.com/product/5786



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



Adafruit Feather RP2350 with HSTX Port

RP2350 flies high with the Feather format - now you can use any FeatherWings with this battery-powered dev board. It comes with 8MB of flash, 22pin HSTX output port,...

https://www.adafruit.com/product/6000

Adafruit Metro RP2350

Coming soon! Sign up to be notified when these are in stock and shipping. We'll be updating this to use the 80-pin version of the RP2350.Choo! Choo! This is the...

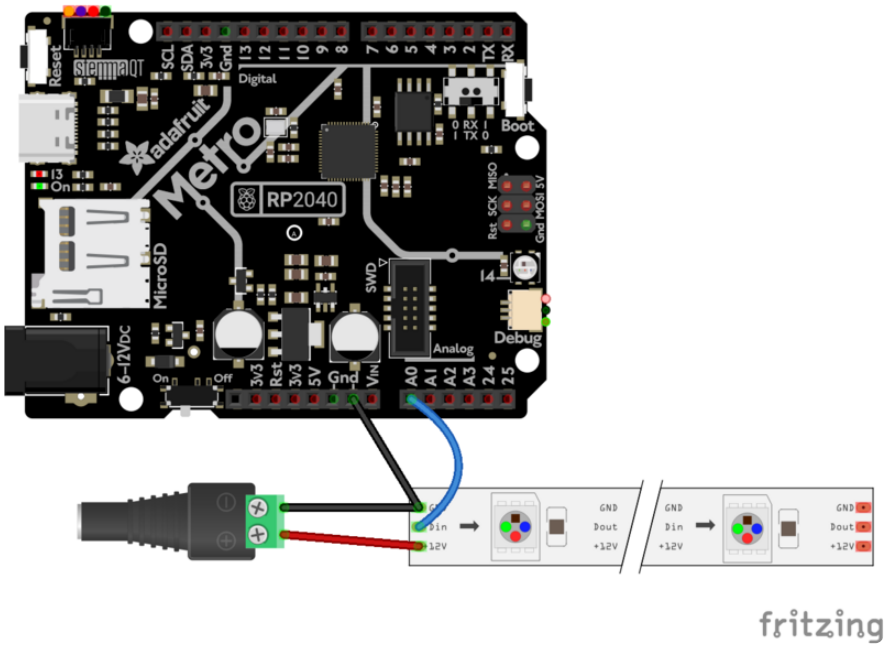
https://www.adafruit.com/product/6003

Pink and Purple Woven USB A to USB C Cable - 1 meter long

This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also made for USB C for our modernized breakout boards, Feathers, and...

https://www.adafruit.com/product/5153

Wiring & Power



Wiring the TM1814 LED strip

Refer to manufacturer instructions to identify the connections & connectors on your TM1814 LED strip.

Make the following connections:

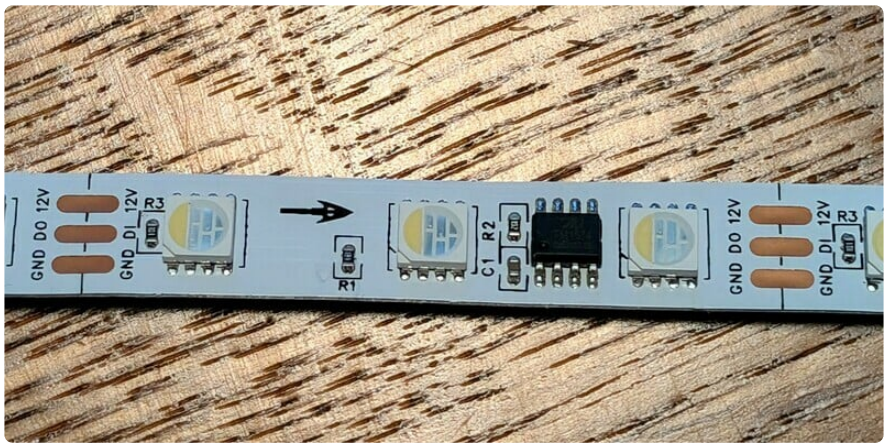
- TM1814 power and GND to dedicated power supply according to manufacturer directions (e.g., 12VDC 20A)
- TM1814 GND to microcontroller GND
- TM1814 Data In to microcontroller A0 (GP26 on Pico / Pico 2)

Do not connect TM1814 power to microcontroller power, as this will damage the microcontroller.

LED Strip Organization

As mentioned above, TM1812 is designed to work with higher supply voltages. This is useful because it allows a strip to provide more total light with less total current. Of course, there's a hitch: The LED strip has to be designed such that it's appropriate to drive the LEDs with a higher voltage. The simplest way to do this is to place several LEDs in series. For instance, a 12V strip may have 3 RGBW LED packages controlled by a single TM1814 controller chip. The typical forward voltage of a white or blue LED is around 3V, and so when 3 are placed in series the typical forward voltage is 3V+3V+3V=9V. Add a few volts for operating margin, and the manufacturer assigns 12V as the nominal voltage of the strip.

In the photo below you can see that there's 1 TM1814 chip (the black rectangle) togther with 3 RGBW LEDs (white rectangles with circles). The strip manufacturer has marked the voltage (12V) and the data direction (data inputs at the left and outputs at the right)



LED strip power estimation

The per-chip maximum current is the chip supply current plus the LED current. The LED current can be up to 4*38mA = 152mA. Add 10mA per chip to account for the TM1814's own power usage, and multiply by the number of chips. That comes out to 16.2A for a 5-meter strip with 100 TM1814s and 300 RGBW LEDs. So a 12V 20A DC power supply is not unreasonable, and it may be helpful to connect GND and VCC at multiple points on the strip to reduce power supply sag under load. Wiring high current LED strips is out of scope for this guide, but never fear: there is a dedicated page within the NeoPixel Überguide (https://adafru.it/DCq) just about how to power LED strips. This advice applies to TM1814, except that you have to observe the manufacturer's voltage rating, not the NeoPixel 5V advice.

If you set the current control registers to a lower value (see below) you can limit the overall strip current as well. For instance, at the lowest 6.5mA the maximum per chip current is probably 36mA, making the whole strip "only" 3.6A. However, you need to keep in mind that the current draw may be higher in self test mode.

In adafruit_tm1814, the default current is the lowest 6.5mA. The self test mode appears to use 12.5mA current.

CircuitPython Simpletest

The code in this guide is compatible with CircuitPython 9.2 and later, including 9.2.0-rc.0.

CircuitPython simpletest

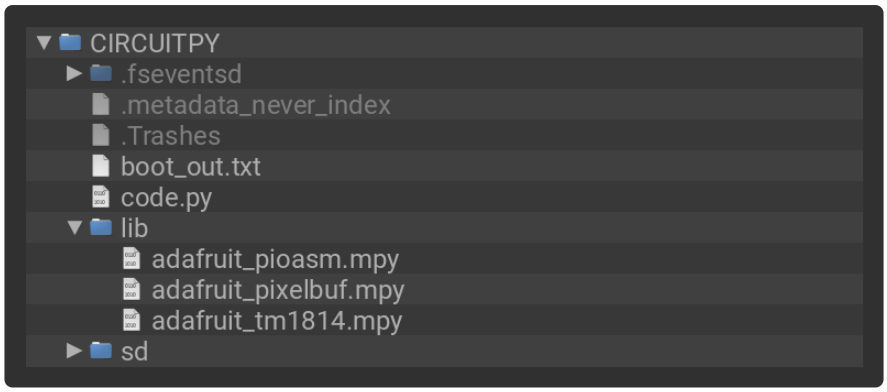
Click on the **Download Project Bundle** button in the window below. It will download to your computer as a zipped folder.

```
# SPDX-FileCopyrightText: Copyright (c) 2024 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import board
import rainbowio
import supervisor

from adafruit_tm1814 import TM1814PixelBackground

# The pin where the LED strip data line is connected
TM1814 = board.A0
# The number of TM1814 controllers. Note that sometimes one "pixel" controls
# more than one LED package.
#
# One common configuration is 3 LED packages & 1
# controller per 50mm, giving 100 TM1814 controllers (300 LED packages) in 5
# meters of LED strip.
NUM_PIXELS = 100
pixels = TM1814PixelBackground(TM1814, NUM_PIXELS, brightness=0.1)

# Cycle the rainbow at about 1 cycle every 4 seconds
while True:
    pixels.fill(rainbowio.colorwheel(supervisor.ticks_ms() // 16))
```

Copy the code to your CircuitPython device

After downloading the Project Bundle, plug your CircuitPython device into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Open up the zipped file, go to the the appropriate folder inside (e.g., **CircuitPython 9.x** if you are using CircuitPython 9.x) and copy the items in that folder directly onto your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should look like this after copying the sound effects and the **code.py** file.



Code Walkthrough & Customization

As usual, Python code begins by importing required modules:

```
import board
import rainbowio
import supervisor

from adafruit_tm1814 import TM1814PixelBackground
```

The next section supports some customization: you can change the pin connected to the TM1814 data line, and customize the number of pixels.

```
# The pin where the LED strip data line is connected
TM1814 = board.A0
# The number of TM1814 controllers. Note that sometimes one "pixel" controls
# more than one LED package.
#
# One common configuration is 3 LED packages & 1
# controller per 50mm, giving 100 TM1814 controllers (300 LED packages) in 5
# meters of LED strip.
NUM_PIXELS = 100
```

A TM1814 LED strip is initialized and set to all black with this line:

```
pixels = TM1814PixelBackground(TM1814, NUM_PIXELS, brightness=0.1)
```

The code's forever loop just sets the whole strip's color to a rainbow color in a pattern that repeats about every 4 seconds:

```
# Cycle the rainbow at about 1 cycle every 4 seconds
while True:
    pixels.fill(rainbowio.colorwheel(supervisor.ticks_ms() // 16))
```

CircuitPython LED Animations

The code in this guide is compatible with CircuitPython 9.2 and later, including 9.2.0-rc.0.

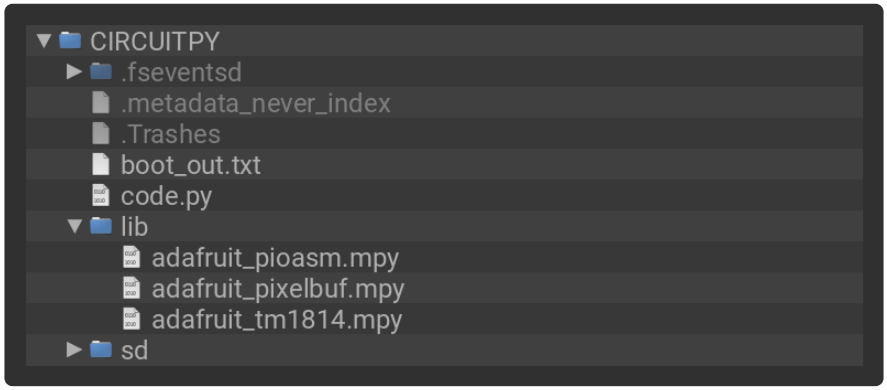
Click on the **Download Project Bundle** button in the window below. It will download to your computer as a zipped folder.

```
# SPDX-FileCopyrightText: Copyright (c) 2024 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import board
import rainbowio
import supervisor
from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.rainbowchase import RainbowChase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.rainbowsparkle import RainbowSparkle
from adafruit_led_animation.sequence import AnimationSequence

from adafruit_tm1814 import TM1814PixelBackground

# The pin where the LED strip data line is connected
TM1814 = board.A0
# The number of TM1814 controllers. Note that sometimes one "pixel" controls
# more than one LED package.
#
# One common configuration is 3 LED packages & 1 controller per 50mm, giving
# 100 TM1814 controllers (300 LED packages) in 5 meters of LED strip.
NUM_PIXELS = 100
pixels = TM1814PixelBackground(TM1814, NUM_PIXELS, brightness=0.1)

# Perform a rainbow animation sequence
rainbow = Rainbow(pixels, speed=0.1, period=2)
rainbow_chase = RainbowChase(pixels, speed=0.1, size=5, spacing=3)
rainbow_comet = RainbowComet(pixels, speed=0.1, tail_length=7, bounce=True)
rainbow_sparkle = RainbowSparkle(pixels, speed=0.1, num_sparkles=15)


animations = AnimationSequence(
    rainbow,
    rainbow_chase,
    rainbow_comet,
    rainbow_sparkle,
    advance_interval=5,
    auto_clear=True,
)

while True:
    animations.animate()
```

Copy the code to your CircuitPython device

After downloading the Project Bundle, plug your CircuitPython device into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Open up the zipped file, go to the the appropriate folder inside (e.g., **CircuitPython 9.x** if you are using CircuitPython 9.x) and copy the items in that folder directly onto your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should look like this after copying the sound effects and the **code.py** file.



Wiring the TM1814 LED strip

Refer to manufacturer instructions to identify the connections & connectors on your TM1814 LED strip.

Make the following connections:

- TM1814 power and GND to dedicated power supply according to manufacturer directions (e.g., 12VDC 2A)
- TM1814 GND to microcontroller GND
- TM1814 Data In to microcontroller A0 (GP26 on Pico / Pico 2)

Do not connect TM1814 power to microcontroller power, as this will damage the microcontroller.

Code Walkthrough & Customization

As usual, Python code begins by importing required modules:

```
import board
import rainbowio
import supervisor

from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.rainbowchase import RainbowChase
```

```
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.rainbowsparkle import RainbowSparkle
from adafruit_led_animation.sequence import AnimationSequence

from adafruit_tm1814 import TM1814PixelBackground
```

The next section supports some customization: you can change the pin connected to the TM1814 data line, and customize the number of pixels.

```
# The pin where the LED strip data line is connected
TM1814 = board.A0
# The number of TM1814 controllers. Note that sometimes one "pixel" controls
# more than one LED package.
#
# One common configuration is 3 LED packages & 1
# controller per 50mm, giving 100 TM1814 controllers (300 LED packages) in 5
# meters of LED strip.
NUM_PIXELS = 100
```

A TM1814 LED strip is initialized and set to all black with this line:

```
pixels = TM1814PixelBackground(TM1814, NUM_PIXELS, brightness=0.1)
```

Then the code creates a series of animations (as detailed in the guide for the Adafruit LED Animation library (https://adafru.it/LAg)) and runs the animations forever:

```
# Perform a rainbow animation sequence
rainbow = Rainbow(pixels, speed=0.1, period=2)
rainbow_chase = RainbowChase(pixels, speed=0.1, size=5, spacing=3)
rainbow_comet = RainbowComet(pixels, speed=0.1, tail_length=7, bounce=True)
rainbow_sparkle = RainbowSparkle(pixels, speed=0.1, num_sparkles=15)


animations = AnimationSequence(
    rainbow,
    rainbow_chase,
    rainbow_comet,
    rainbow_sparkle,
    advance_interval=5,
    auto_clear=True,
)

while True:
    animations.animate()
```

CircuitPython Docs

CircuitPython Docs (https://adafru.it/1a9h)

Downloads

TM1814 Datasheet

https://adafru.it/1a9j