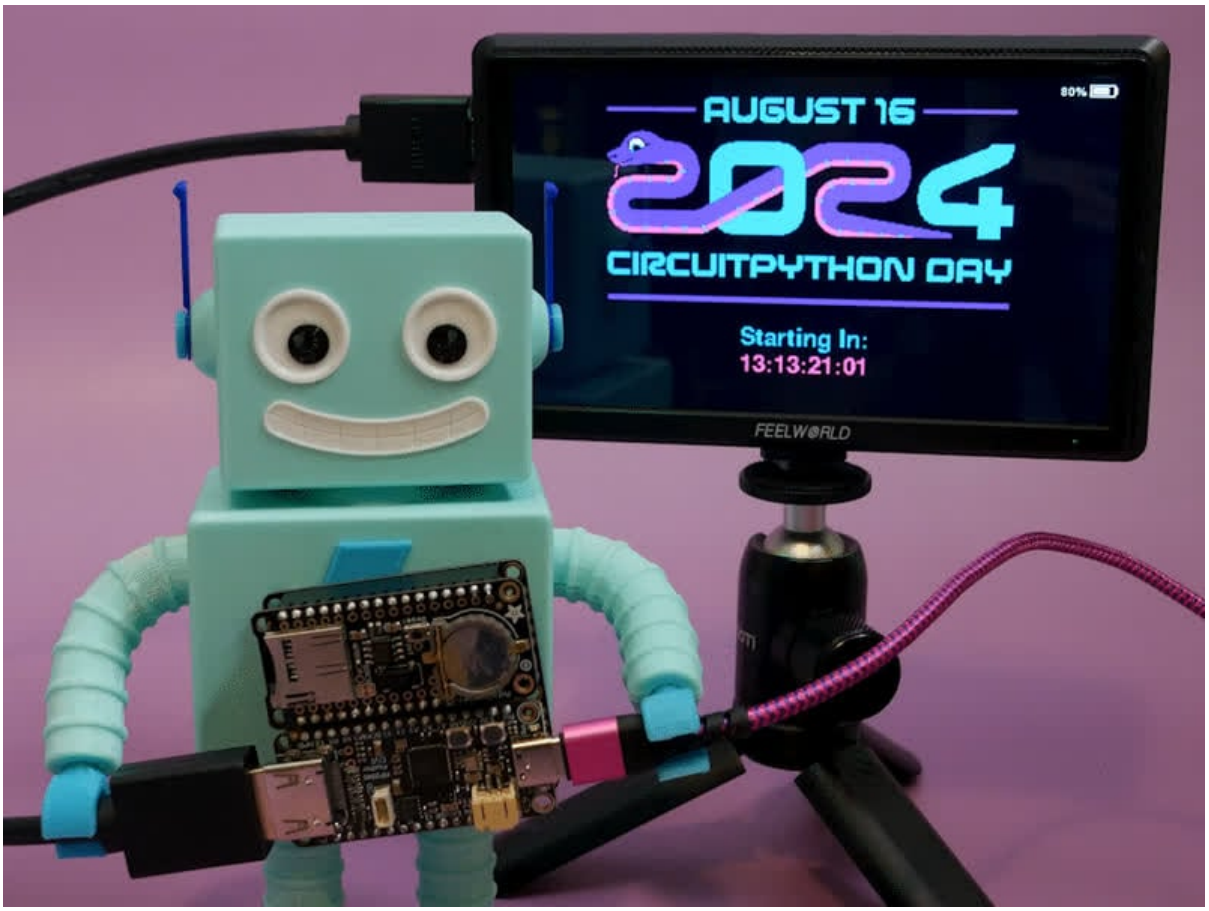
Feather RP2040 DVI CircuitPython Day 2024 Countdown Clock

Created by Liz Clark

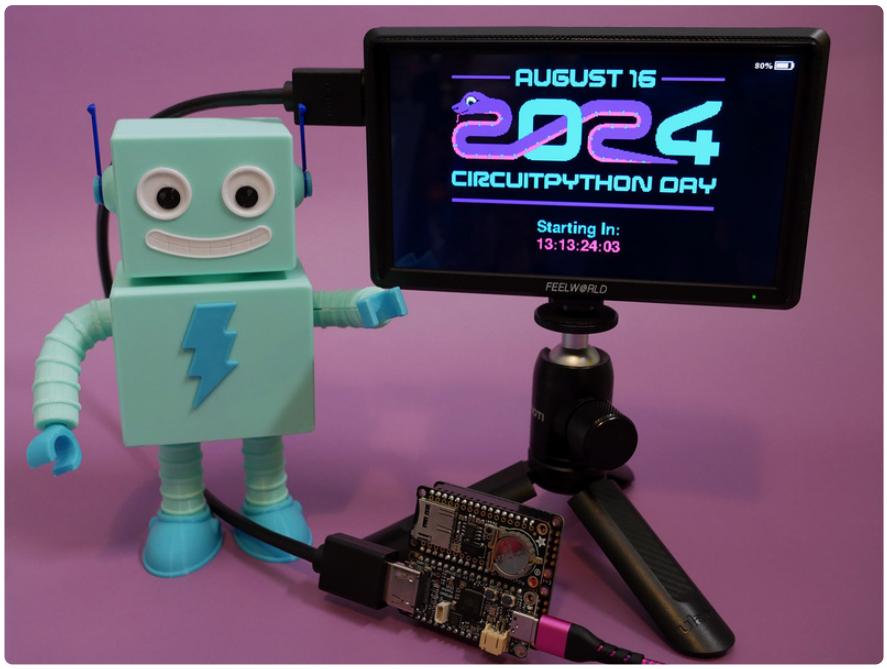


https://learn.adafruit.com/feather-rp2040-dvi-circuitpython-day-2024-countdown-clock

Last updated on 2024-08-02 01:49:51 PM EDT

Table of Contents

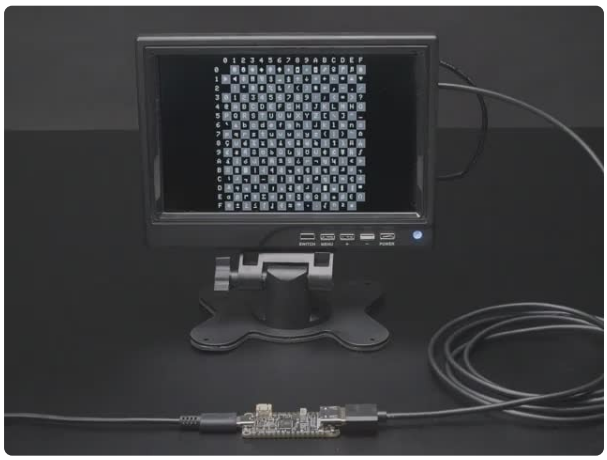
Overview



In this year of chaos and calamity, one day on the horizon fills us with hope: CircuitPython Day 2024! Countdown to the snakiest day of the year (August 16, 2024) with, what else? A CircuitPython project!

In this project, you'll use a Feather RP2040 DVI board, along with an Adalogger FeatherWing, to calculate down to the second how much longer you have to wait for this year's festivities. Plug the DVI port on the Feather into an idle TV, computer monitor, capture card or other HDMI input to admire your CircuitPython coded pixels.

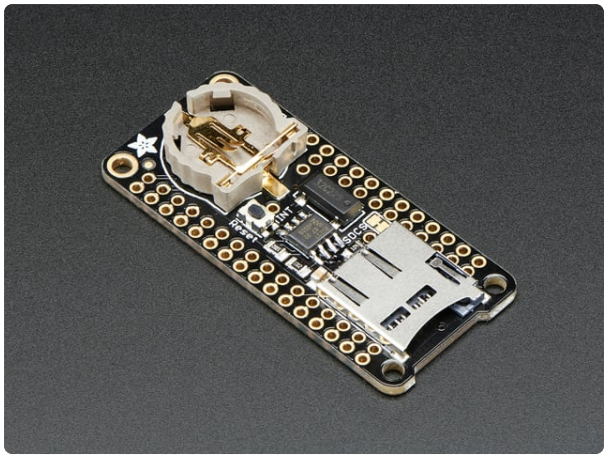
Parts



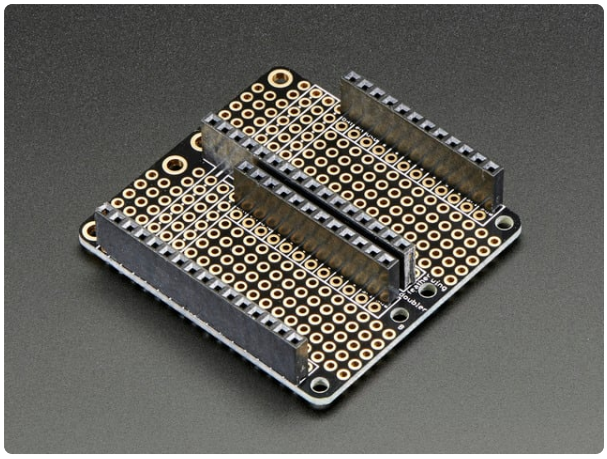
Adafruit Feather RP2040 with DVI Output Port - Works with HDMI

Wouldn't it be cool if you could display images and graphics from a microcontroller directly to an HDMI monitor or television? We think so! So we designed this RP2040 Feather that...

https://www.adafruit.com/product/5710



[Adalogger FeatherWing - RTC + SD Add-on For All Feather Boards]
A Feather board without ambition is a Feather board without FeatherWings! This is the Adalogger FeatherWing: it adds both a battery-backed Real Time Clock and micro SD...
https://www.adafruit.com/product/2922



[FeatherWing Doubler - Prototyping Add-on For All Feather Boards]
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our
https://www.adafruit.com/product/2890

[Pink and Purple Woven USB A to USB C Cable - 2 meters 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 more.
https://www.adafruit.com/product/5044

[Slim HDMI Cable - 1820mm / 6 feet long]
Connect two HDMI devices together and save space with this slim, high quality HDMI cable. It has nice molded grips for easy installation and is 1820mm long (~6 feet)....
https://www.adafruit.com/product/2422

7" Display 1280x800 (720p) IPS + Speakers - HDMI/VGA/NTSC/PAL
Yes, this is an adorable small HDMI television with incredibly high resolution and built in 3W stereo speakers! We tried to get the smallest possible HDMI/VGA display with...
https://www.adafruit.com/product/1667

1 x CR1220 Battery https://www.adafruit.com/product/380
CR1220 12mm Diameter - 3V Lithium Coin Cell Battery

CircuitPython

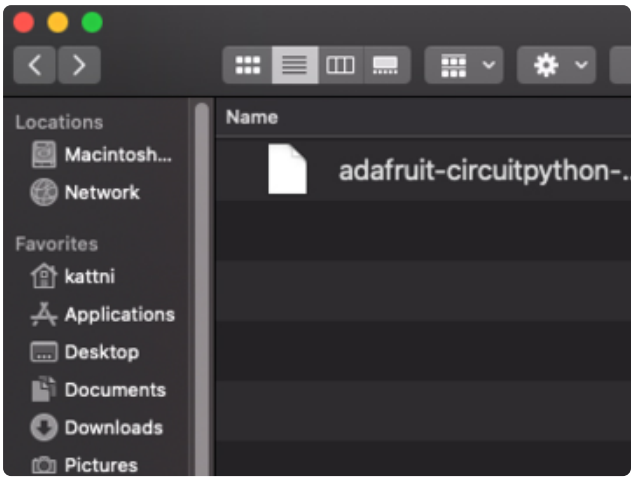
CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

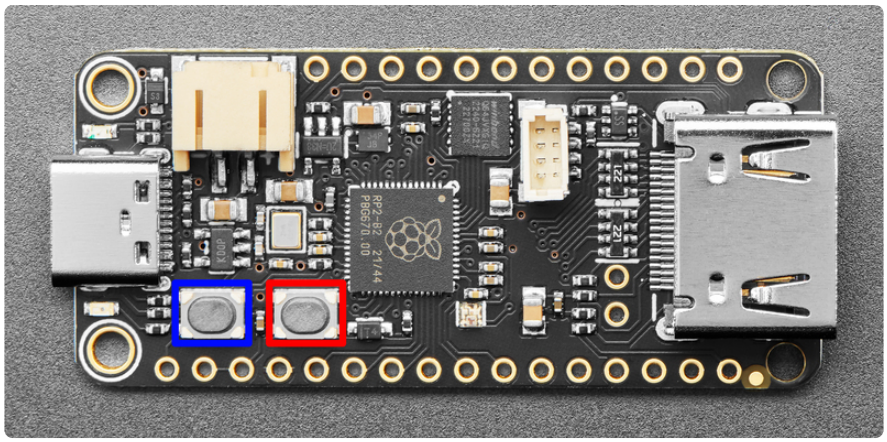
Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/19cl



Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

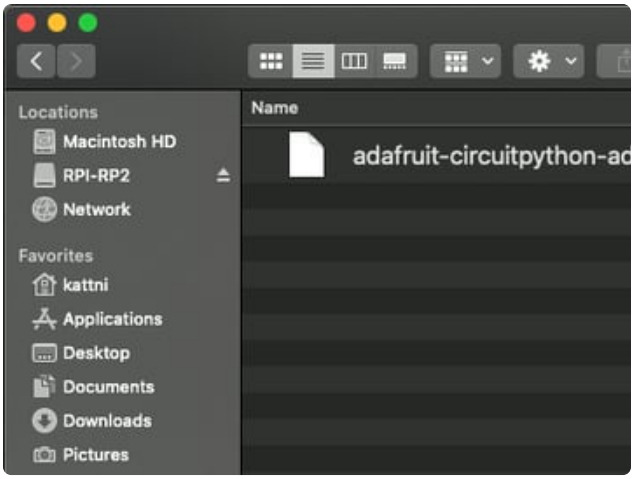


To enter the bootloader, hold down the **BOOT/BOOTSEL button** (highlighted in red above), and while continuing to hold it (don't let go!), press and release the **reset button** (highlighted in blue above). **Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!**

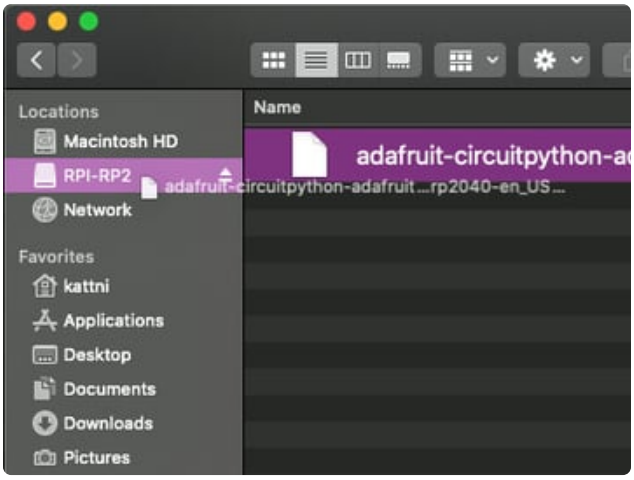
If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

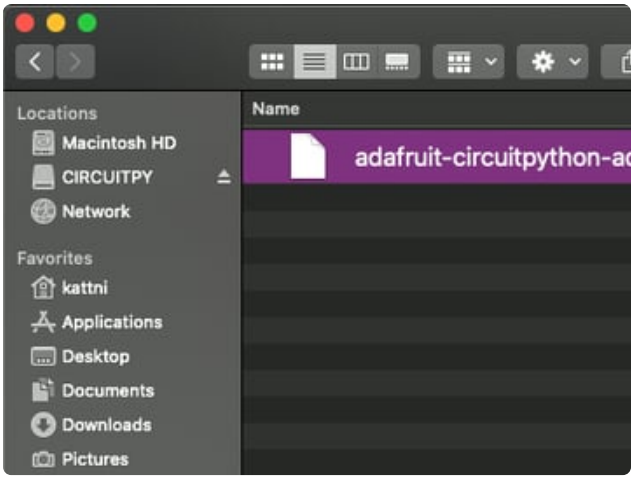
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



You will see a new disk drive appear called **RPI-RP2**.



Drag the **adafruit_circuitpython_etc.uf2** file to **RPI-RP2.**



The **RPI-RP2** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Safe Mode

You want to edit your **code.py** or modify the files on your **CIRCUITPY** drive, but find that you can't. Perhaps your board has gotten into a state where **CIRCUITPY** is read-only. You may have turned off the **CIRCUITPY** drive altogether. Whatever the reason, safe mode can help.

Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in **boot.py** (where you can set **CIRCUITPY** read-only or turn it off completely). Second, it does not run the code in **code.py**. And finally, it does not automatically soft-reload when data is written to the **CIRCUITPY** drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the **CIRCUITPY** drive.

Entering Safe Mode

To enter safe mode when using CircuitPython, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

If you successfully enter safe mode on CircuitPython, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.

CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.

Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the **CIRCUITPY** drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and CIRCUITPY doesn't show up as a disk drive after installing CircuitPython, try loading this 'nuke' UF2 to RPI-RP2. which

will do a 'deep clean' on your Flash Memory. **You will lose all the files on the board**, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

https://adafru.it/RLE

Code the Countdown Clock

Once you've finished setting up your Feather RP2040 DVI with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download to your computer as a zipped folder.

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

import time
import displayio
import picodvi
import board
import framebufferio
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label
from adafruit_pcf8523.pcf8523 import PCF8523
from adafruit_ticks import ticks_ms, ticks_add, ticks_diff

EVENT_YEAR = 2024
EVENT_MONTH = 8
EVENT_DAY = 16
EVENT_HOUR = 0
EVENT_MINUTE = 0
# we'll make a python-friendly structure
event_time = time.struct_time((EVENT_YEAR, EVENT_MONTH, EVENT_DAY,
                               EVENT_HOUR, EVENT_MINUTE, 0,  # we don't track
seconds
                               -1, -1, False))  # we dont know day of week/year or
DST

# check for DVI Feather with built-in display
if 'DISPLAY' in dir(board):
    display = board.DISPLAY

# check for DVI feather without built-in display
elif 'CKP' in dir(board):
    displayio.release_displays()
    fb = picodvi.Framebuffer(320, 240,
        clk_dp=board.CKP, clk_dn=board.CKN,
        red_dp=board.D0P, red_dn=board.D0N,
        green_dp=board.D1P, green_dn=board.D1N,
        blue_dp=board.D2P, blue_dn=board.D2N,
        color_depth=8)
    display = framebufferio.FramebufferDisplay(fb)
# otherwise assume Pico
else:
    displayio.release_displays()
```

```
    fb = picodvi.Framebuffer(320, 240,
        clk_dp=board.GP12, clk_dn=board.GP13,
        red_dp=board.GP10, red_dn=board.GP11,
        green_dp=board.GP8, green_dn=board.GP9,
        blue_dp=board.GP6, blue_dn=board.GP7,
        color_depth=8)
    display = framebufferio.FramebufferDisplay(fb)

i2c = board.I2C()
rtc = PCF8523(i2c)
set_clock = False
if set_clock:
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2024,  8,   1,   16,  26,  00,    0,   -1,    -1))
    print("Setting time to:", t)
    rtc.datetime = t
    print()
#  variable to hold RTC datetime
t = rtc.datetime

pink = 0xf1078e
purple = 0x673192
aqua = 0x19beed
group = displayio.Group()
my_font = bitmap_font.load_font("/Helvetica-Bold-16.pcf")
clock_area = label.Label(my_font, text="00:00:00:00", color=pink)
clock_area.anchor_point = (0.0, 1.0)
clock_area.anchored_position = (display.width / 2 - clock_area.width / 2,
                                display.height - (clock_area.height + 20))
text1 = label.Label(my_font, text="Starting In:", color=aqua)
text1.anchor_point = (0.0, 0.0)
text1.anchored_position = (display.width / 2 - text1.width / 2,
                           display.height - (clock_area.height + text1.height + 35))

blinka_bitmap = displayio.OnDiskBitmap("/cpday_dvi.bmp")
blinka_grid = displayio.TileGrid(blinka_bitmap,
pixel_shader=blinka_bitmap.pixel_shader)
group.append(blinka_grid)
group.append(text1)
group.append(clock_area)
display.root_group = group

clock_clock = ticks_ms()
clock_timer = 1000
while True:
    if ticks_diff(ticks_ms(), clock_clock) >= clock_timer:
        t = rtc.datetime
        remaining = time.mktime(event_time) - time.mktime(t)
        secs_remaining = remaining % 60
        remaining //= 60
        mins_remaining = remaining % 60
        remaining //= 60
        hours_remaining = remaining % 24
        remaining //= 24
        days_remaining = remaining
        clock_area.text = (f"{days_remaining:0>2}:{hours_remaining:0>2}" +
                           f":{mins_remaining:0>2}:{secs_remaining:0>2}")
        clock_clock = ticks_add(clock_clock, clock_timer)
```

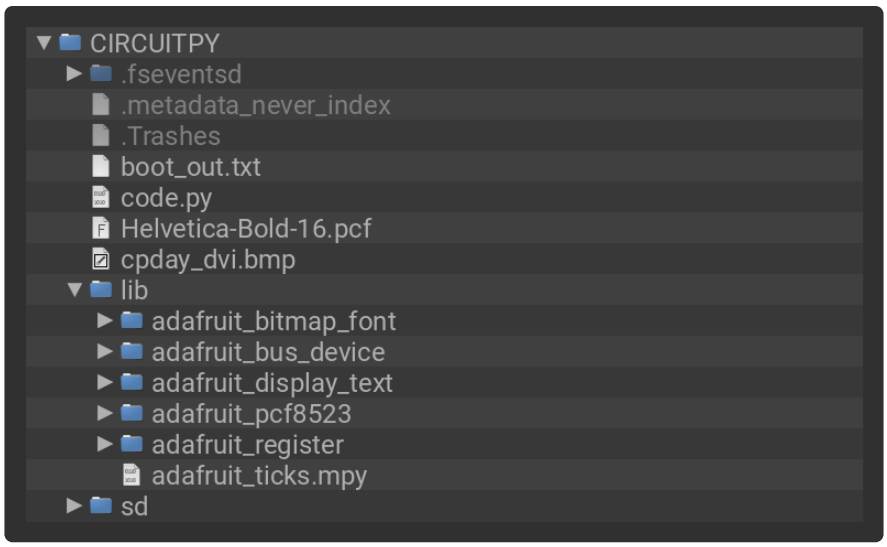
Upload the Code and Libraries to the Feather RP2040 DVI

After downloading the Project Bundle, plug your Feather RP2040 DVI 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**. Unzip the folder and copy the following items to the Feather RP2040 DVI's **CIRCUITPY** drive.

- **lib** folder
- **code.py**
- **cpday_dvi.bmp**
- **Helvetica-Bold-16.pcf**

Your Feather RP2040 DVI **CIRCUITPY** drive should look like this after copying the **lib** folder, **cpday_dvi.bmp** file, **Helvetica-Bold-16.pcf** file and the **code.py** file.



How the CircuitPython Code Works

At the top of the code, the event time is set up. In this case, it's August 16, 2024 at midnight. Then, the RTC on the Adalogger FeatherWing is instantiated over I2C.

If you need to set the time on the RTC, you can edit the `struct_time` to match your current time and then change `set_clock` to `True`. Once the time is set, change `set_clock` to `False` and save the code again.

```
EVENT_YEAR = 2024
EVENT_MONTH = 8
EVENT_DAY = 16
EVENT_HOUR = 0
EVENT_MINUTE = 0
# we'll make a python-friendly structure
event_time = time.struct_time((EVENT_YEAR, EVENT_MONTH, EVENT_DAY,
                               EVENT_HOUR, EVENT_MINUTE, 0,  # we don't track
seconds
                               -1, -1, False))  # we dont know day of week/year or
DST
```

```
i2c = board.I2C()
rtc = PCF8523(i2c)
set_clock = False

if set_clock:
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2024,  7,   30,   15,  59,  00,    0,   -1,    -1))
    print("Setting time to:", t)
    rtc.datetime = t
    print()
```

Next, the framebuffer for the DVI output is created.

```
# check for DVI Feather with built-in display
if 'DISPLAY' in dir(board):
    display = board.DISPLAY

# check for DVI feather without built-in display
elif 'CKP' in dir(board):
    displayio.release_displays()
    fb = picodvi.Framebuffer(320, 240,
        clk_dp=board.CKP, clk_dn=board.CKN,
        red_dp=board.D0P, red_dn=board.D0N,
        green_dp=board.D1P, green_dn=board.D1N,
        blue_dp=board.D2P, blue_dn=board.D2N,
        color_depth=8)
    display = framebufferio.FramebufferDisplay(fb)
# otherwise assume Pico
else:
    displayio.release_displays()
    fb = picodvi.Framebuffer(320, 240,
        clk_dp=board.GP12, clk_dn=board.GP13,
        red_dp=board.GP10, red_dn=board.GP11,
        green_dp=board.GP8, green_dn=board.GP9,
        blue_dp=board.GP6, blue_dn=board.GP7,
        color_depth=8)
    display = framebufferio.FramebufferDisplay(fb)
```

Graphics

Next are the display objects for the DVI output. These take care of the background bitmap graphic and text elements.

```
pink = 0xf1078e
purple = 0x673192
aqua = 0x19beed
group = displayio.Group()
my_font = bitmap_font.load_font("/Helvetica-Bold-16.pcf")
clock_area = label.Label(my_font, text="00:00:00:00", color=pink)
clock_area.anchor_point = (0.0, 1.0)
clock_area.anchored_position = (display.width / 2 - clock_area.width / 2,
                                display.height - (clock_area.height + 20))
text1 = label.Label(my_font, text="Starting In:", color=aqua)
text1.anchor_point = (0.0, 0.0)
text1.anchored_position = (display.width / 2 - text1.width / 2,
                           display.height - (clock_area.height + text1.height + 35))

blinka_bitmap = displayio.OnDiskBitmap("/cpday_dvi.bmp")
blinka_grid = displayio.TileGrid(blinka_bitmap,
pixel_shader=blinka_bitmap.pixel_shader)
group.append(blinka_grid)
group.append(text1)
```

```
group.append(clock_area)
display.root_group = group
```

Time is Ticking

Finally, a `ticks` timers is created for timekeeping in the loop.

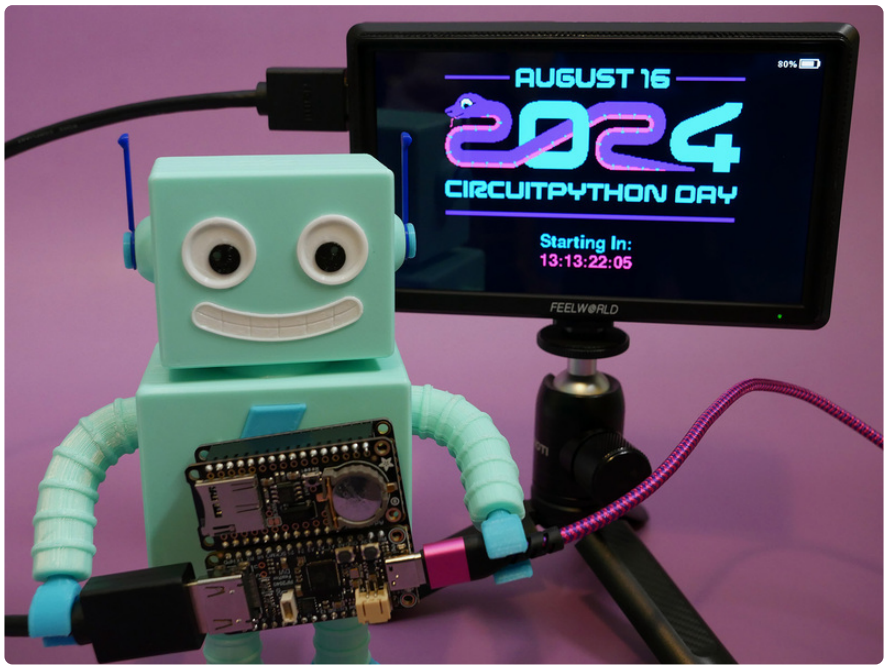
```
clock_clock = ticks_ms()
clock_timer = 1000
```

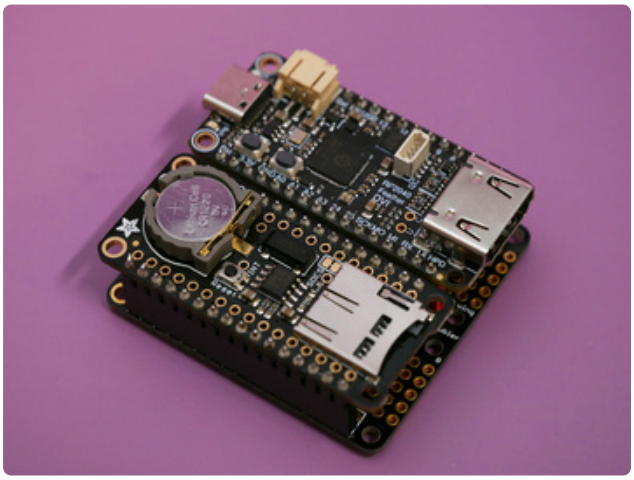
The Loop

In the loop, the time is read from the RTC. The time remaining until the event is calculated by using `mktime()` to convert the event time and time reading from the RTC into seconds. This is stored in `remaining` and is converted to days, hours, minutes and seconds. These values are added to the `clock_area` text to display via DVI.

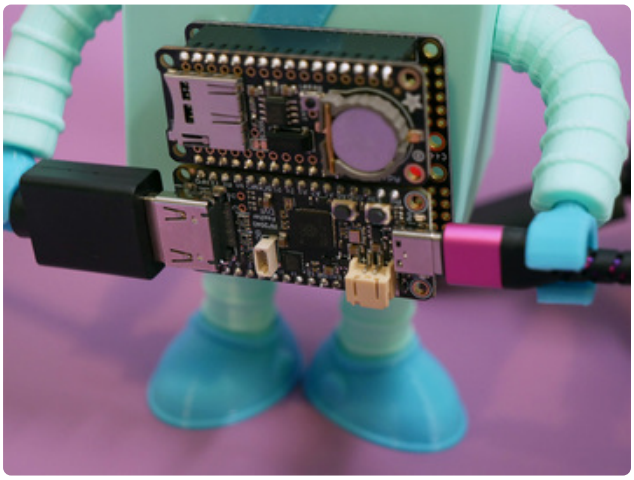
```
while True:
    if ticks_diff(ticks_ms(), clock_clock) &gt;= clock_timer:
        t = rtc.datetime
        remaining = time.mktime(event_time) - time.mktime(t)
        secs_remaining = remaining % 60
        remaining //= 60
        mins_remaining = remaining % 60
        remaining //= 60
        hours_remaining = remaining % 24
        remaining //= 24
        days_remaining = remaining
        clock_area.text = (f"{days_remaining:0&gt;2}:{hours_remaining:0&gt;2}" +
                           f":{mins_remaining:0&gt;2}:{secs_remaining:0&gt;2}")
        clock_clock = ticks_add(clock_clock, clock_timer)
```

Assembly and Use





Plug the Feather and FeatherWing into the FeatherWing Doubler.



Power the Feather via USB. Then plug in an HDMI cable into the DVI port on the Feather and your HDMI display. You'll see the countdown clock appear on the display.