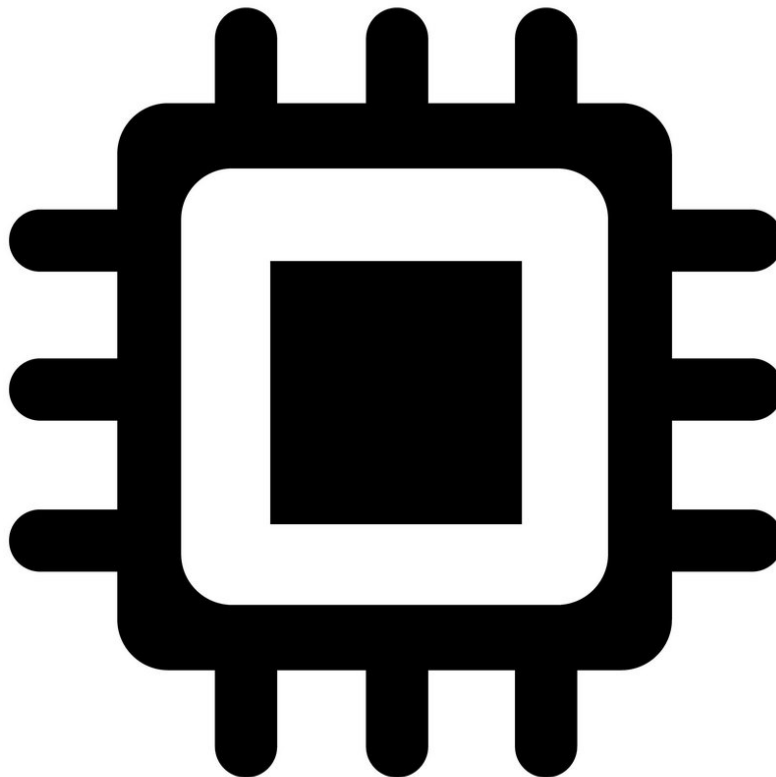




Learn to program a microcontroller with the Feather M4 Express

Get started with the Feather M4 and CircuitPython.

Written By: Eli Foster



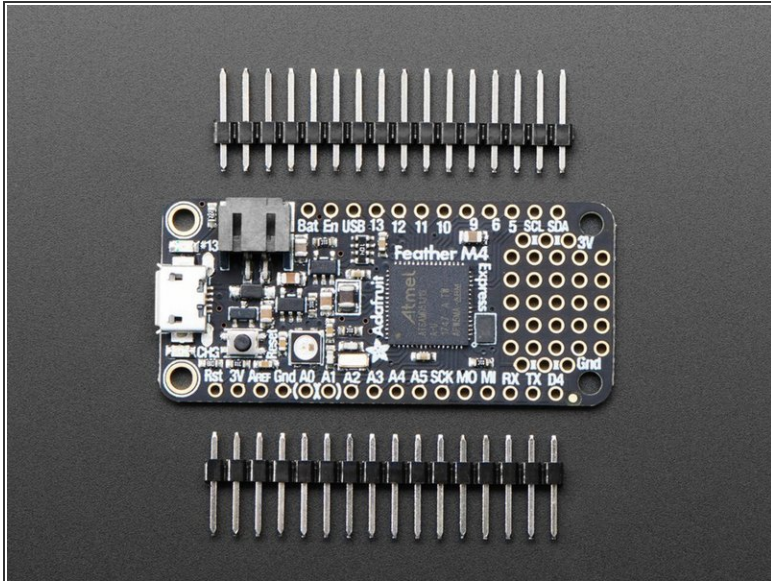
 TOOLS:

- [Micro-USB Cable](#) (1)

 PARTS:

- [Feather M4 Express](#) (1)

Step 1 — Bootloader



Update UF2 Bootloader

Latest version: v3.15.0

The bootloader allows you to load CircuitPython, MakeCode, and Arduino programs. The bootloader is not CircuitPython. You can check the current version of your bootloader by looking in the **INFO_UF2.TXT** file when the **BOOT** drive is visible (**FEATHERBOOT**, **CPLAYBOOT**, etc.).

It is not necessary to update your bootloader if it is working fine. Read the release notes on GitHub to see what has been changed. In general, we recommend you not update the bootloader unless you know there is a problem with it or a support person has asked you to try updating it.

To update, first save the contents of **CIRCUITPY**, just in case. Then double-click the reset button to show the **BOOT** drive. Drag the **update-bootloader.uf2** file to the **BOOT** drive. Wait a few tens of seconds for the bootloader to update; the **BOOT** drive will reappear. After you update, check **INFO_UF2.TXT** to verify that the bootloader version has been updated. Then you will need to reload CircuitPython.

DOWNLOAD UPDATER UF2 

BROWSE ON GITHUB 

- If using a factory Feather, the bootloader may need to be uploaded and CircuitPython installed.
- To do so, follow [Adafruit's guide on updating the bootloader](#): If your version is newer than v3.9.0, this step is not necessary.
- Download the appropriate bootloader from the [CircuitPython website](#), at the bottom of the page

Step 2 — CircuitPython

CircuitPython 9.0.4

This is the latest **stable** release of CircuitPython that will work with the Feather M4 Express.

Use this release if you are new to CircuitPython.

[Release Notes for 9.0.4](#)

ENGLISH (US) 

DOWNLOAD .UF2 NOW 

Built-in modules available: `_asyncio`, `_bleio`, `_pixelmap`, `adafruit_bus_device`, `adafruit_pixelbuf`, `aesio`, `alarm`, `analogio`, `array`, `atexit`, `audiobusio`, `audiocore`, `audioio`, `audiomixer`, `audiomp3`, `binascii`, `bitbangio`, `bitmaptools`, `board`, `builtins`, `builtins.pow3`, `busdisplay`, `busio`, `busio.SPI`, `busio.UART`, `codeop`, `collections`, `countio`, `digitalio`, `displayio`, `epaperdisplay`, `errno`, `fontio`, `fourwire`, `framebufferio`, `frequencyio`, `getpass`, `gifio`, `i2cdisplaybus`, `i2ctarget`, `io`, `json`, `keypad`, `keypad.KeyMatrix`, `keypad.Keys`, `keypad.ShiftRegisterKeys`, `locale`, `math`, `microcontroller`, `msgpack`, `neopixel_write`, `nvm`, `onewireio`, `os`, `os.getenv`, `paralleldisplaybus`, `ps2io`, `pulseio`, `pwmio`, `rainbowio`, `random`, `re`, `rgbmatrix`, `rotaryio`, `rtc`, `samd`, `sdcardio`, `select`, `sharpdisplay`, `storage`, `struct`, `supervisor`, `sys`, `terminalio`, `time`, `touchio`, `traceback`, `ulab`, `usb_cdc`, `usb_hid`, `usb_midi`, `vectorio`, `warnings`, `watchdog`, `zlib`

- To install CircuitPython, follow [Adafruit's guide](#) on installation.
- CircuitPython can be downloaded from the [same page](#), simply choose the first download instead of the last.
- Use your micro-USB cable to plug into the silver port on the Feather, and use the other end to interface with a personal computer or Lab PC.

Step 3 — Libraries

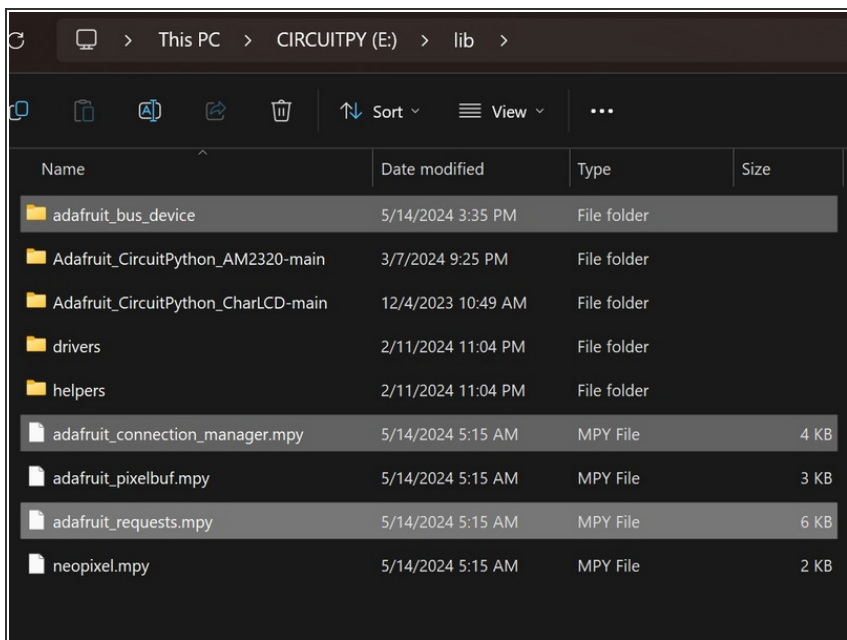
Bundle for Version 9.x

This bundle is built for use with CircuitPython 9.x.x. If you are using CircuitPython 9, please download this bundle. The .mpy format has changed as of CircuitPython 9, so 8.x libraries are *not* compatible.

adafruit-circuitpython-bundle-9.x-mpy-20240514.zip 

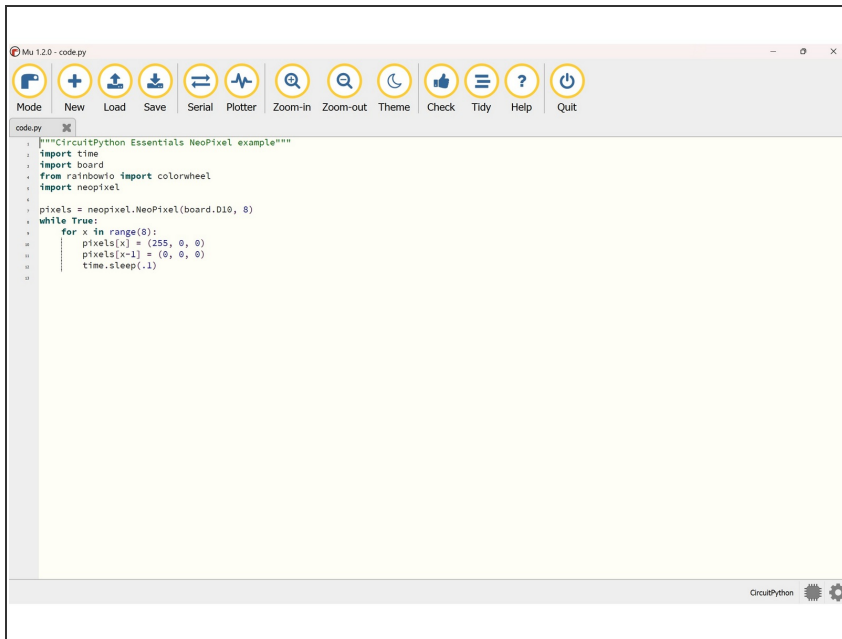
- To use external hardware or use some certain code, you need the specific libraries that will allow you to call functions, take sensor data, and use the onboard pins.
- CircuitPython hosts [library bundles](#), where you can download large lists of libraries for any number of applications. We will simply find the ones we need in the bundle and use them.
- Make sure that the libraries you download match the version of CircuitPython you downloaded in the last step. Version 9.x.x is not the first download link presented.
- Extract the .zip file anywhere convenient for you. If your Feather has enough storage space, it may be convenient to keep all the libraries onboard.

Step 4 — Starter Libraries



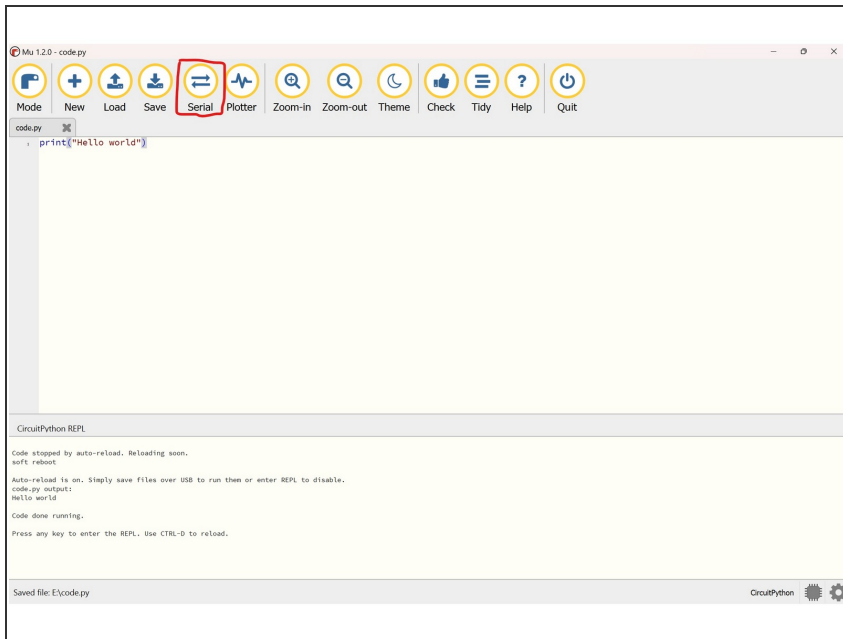
- To fully utilize the Feather's onboard capabilities, we will need a few specific libraries.
- If not uploading the entire bundle to the Feather, find the **adafruit_bus_device** folder, the **adafruit_connection_manager.mpy** file, and the **adafruit_requests.mpy** file.
- These need to be placed not in the root of your CIRCUIPTY drive, but in the **lib** folder. Copy or move the above files into the **lib** folder, and do *not* place them into the **drivers** folder.

Step 5 — Mu Editor



- The Feather with CircuitPython can use either Python or Arduino code in the typical formats.
- To use the Feather to it's full potential, make sure that you have an IDE installed that can write and read said languages. You may be familiar with Mu Editor, VSCode, or any other IDE that you prefer. For first-time users, we recommend [Mu Editor](#) as it is effective and lightweight.
- CircuitPython will only run code named **code.py**. Make sure that you name your file code.py and save it when you want to run it.

Step 6 — Python



- With the correct files on the Feather and an IDE to communicate with it, you can do anything your heart desires with a Feather.
- Most people choose to write code for it in Python, and it will compute anything you give to it.
- Typical Python syntax applies, so **print("Hello world")** will do exactly what you think it will. For more Python instruction, visit [W3Schools](https://www.w3schools.com/python/).
- CircuitPython does have one quirk, that it runs the code whenever it's updated. That is, when the Feather is plugged into power or code.py is saved onto the CIRCUITPY directory. It will *only* run the code saved as **code.py**.

Step 7 — Challenge



```
import digitalio
import board

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT
while True:
    led.value = True
```

- To earn your badge for the Feather, use the onboard LED to blink out some sort of pattern.
- The onboard RGB led looks like this on the left side of the USB port on the feather.
- Use the **digitalio** library to define an LED with the onboard red light (**board.D13**). The example code is in the second image.
- Now you can define the LED's direction by using the *variable.value* function, for example **led.value = True/False**.
- A simple **while** loop is currently making the LED stay on. To create a pattern of turning it on and off, use **time.sleep(number)** to wait for a set amount of time, *number* in seconds. If you use this syntax, remember that you must include **import time** at the beginning of your code.
- Set up a recurring pattern with the onboard LED.

The Feather M4 is the basis of many projects listed elsewhere on Dozuki and can be used for practically unlimited projects. Anything you can do with Python, you can do on a Feather.