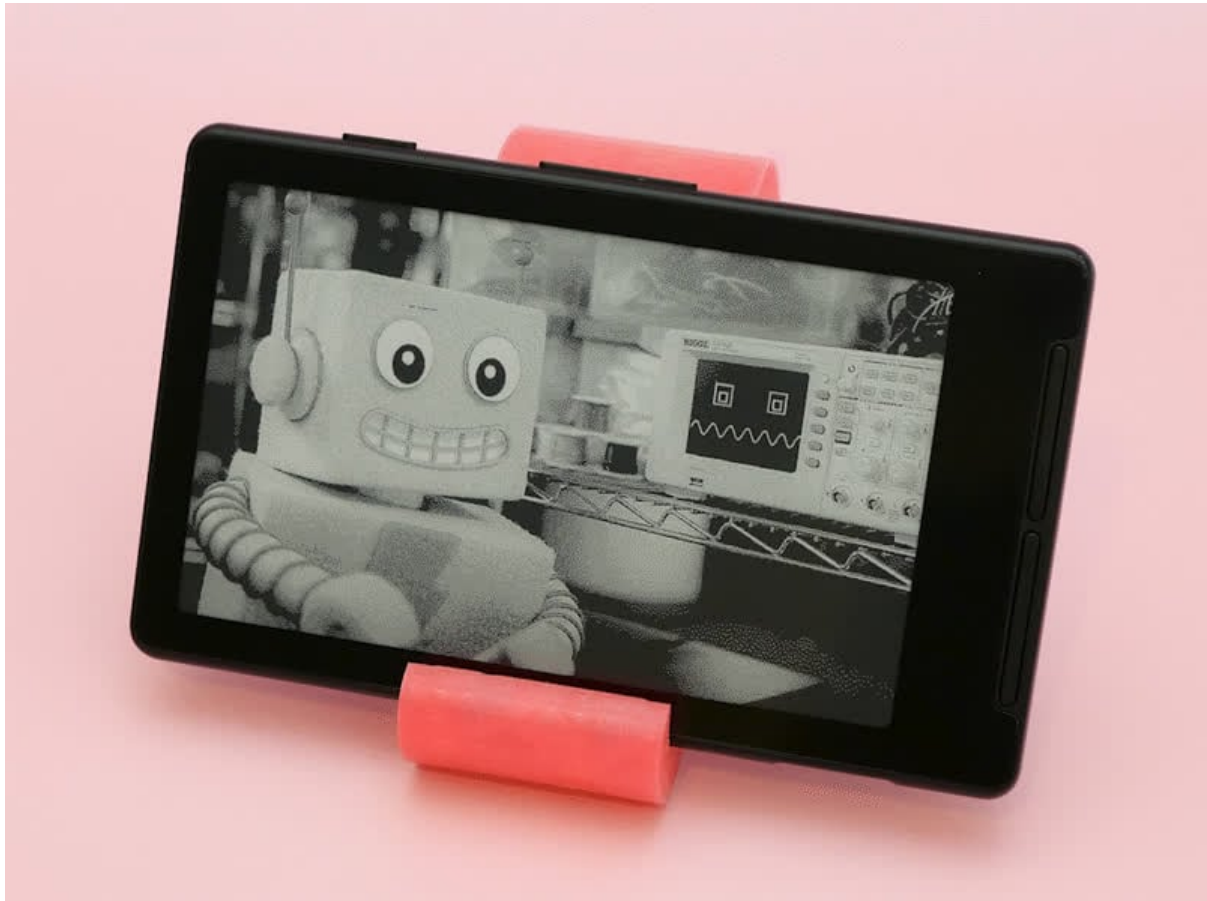




# CircuitPython on the Xteink X4 eReader

Created by Liz Clark



<https://learn.adafruit.com/circuitpython-on-the-xteink-x4-ereader>

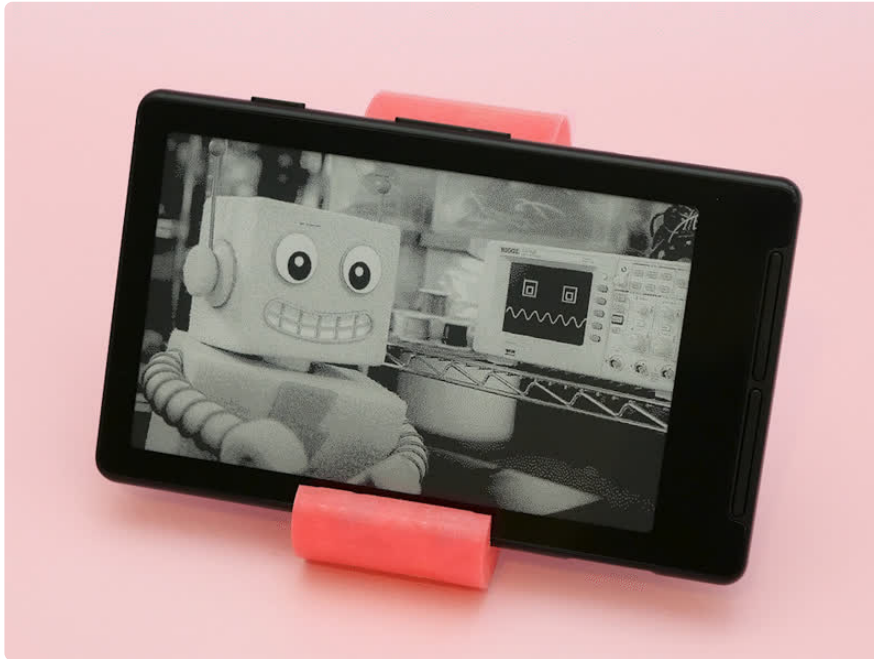
Last updated on 2026-06-03 08:54:41 PM UTC

# Table of Contents

<b>Overview</b>	<b>3</b>
<ul style="list-style-type: none"><li>• <a href="#">OpenX4 E-Paper Community SDK</a></li><li>• <a href="#">Parts</a></li></ul>	
<b>Pinouts</b>	<b>5</b>
<ul style="list-style-type: none"><li>• <a href="#">ESP32-C3</a></li><li>• <a href="#">eInk Display</a></li><li>• <a href="#">Buttons</a></li><li>• <a href="#">microSD Card</a></li><li>• <a href="#">Battery Monitoring</a></li></ul>	
<b>Back-up the Firmware</b>	<b>6</b>
<ul style="list-style-type: none"><li>• <a href="#">ESPTool</a></li></ul>	
<b>Install CircuitPython</b>	<b>7</b>
<ul style="list-style-type: none"><li>• <a href="#">CircuitPython Download</a></li><li>• <a href="#">Connecting to the Web Flasher</a></li><li>• <a href="#">Erasing the Board Contents</a></li><li>• <a href="#">Programming the Board</a></li></ul>	
<b>Setting up Web Workflow</b>	<b>12</b>
<ul style="list-style-type: none"><li>• <a href="#">Option 1: Create settings.toml file via REPL</a></li><li>• <a href="#">Option 2: Create settings.toml file using Thonny</a></li></ul>	
<b>Connecting via Web Browser</b>	<b>18</b>
<ul style="list-style-type: none"><li>• <a href="#">Connect using MDNS Address</a></li><li>• <a href="#">Connect using IP Address</a></li></ul>	
<b>Using Web Workflow</b>	<b>20</b>
<ul style="list-style-type: none"><li>• <a href="#">Welcome! Page</a></li><li>• <a href="#">Password Entry</a></li><li>• <a href="#">Serial Terminal</a></li><li>• <a href="#">File Browser</a></li></ul>	
<b>CircuitPython Helper Library</b>	<b>23</b>
<ul style="list-style-type: none"><li>• <a href="#">Library Usage with Web Workflow</a></li><li>• <a href="#">Upload the Library</a></li><li>• <a href="#">Upload code.py</a></li><li>• <a href="#">Run the Code</a></li></ul>	
<b>eInk Display</b>	<b>29</b>
<ul style="list-style-type: none"><li>• <a href="#">Library Usage with Web Workflow</a></li><li>• <a href="#">Upload the Library File</a></li><li>• <a href="#">Upload the Code and Bitmaps</a></li><li>• <a href="#">Run the Code</a></li></ul>	
<b>Weather Demo</b>	<b>34</b>
<ul style="list-style-type: none"><li>• <a href="#">Library Usage with Web Workflow</a></li><li>• <a href="#">Upload the Libraries</a></li><li>• <a href="#">Upload the Bitmaps and Fonts</a></li><li>• <a href="#">Upload code.py</a></li><li>• <a href="#">Run the Code</a></li></ul>	

---

# Overview



Many folks in our maker community find an intriguing device with a familiar microcontroller inside and think, "Will it run DOOM?". Here at Adafruit, of course, that thought crosses our minds, but the more pertinent question is always "Can it run CircuitPython?".

The latest object to make us wonder this is the Xteink X4 eReader, a super slim pocket-sized eReader. It has an ESP32-C3 inside with serial over JTAG peripheral available through the USB port, making it easy to interface with, to load alternative firmware; no teardown required ([but here's one if you're curious](https://adafru.it/1aBz)) (<https://adafru.it/1aBz>).

## OpenX4 E-Paper Community SDK

There is a fantastic project for the Xteink X4 called the [OpenX4 E-Paper Community SDK](https://adafru.it/1aBA) (<https://adafru.it/1aBA>). It is written for use with ESP-IDF and PlatformIO. The work that's been done with that project made porting the Xteink X4 to CircuitPython super easy.



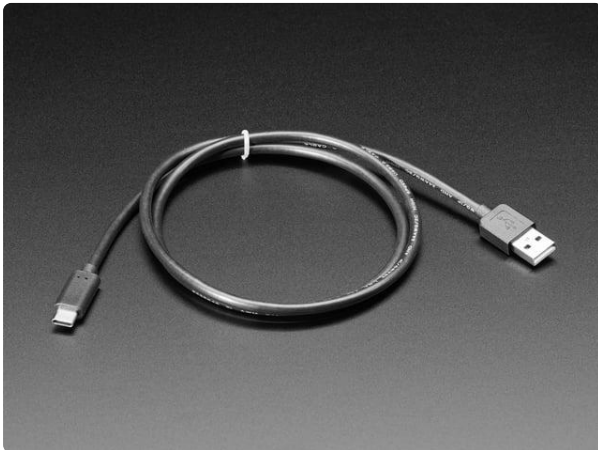
For working with the Xteink X4 and CircuitPython, Web Serial support in your browser is needed, and requires at least Firefox 151 or a Chrome-89-based browser.

## Parts

1 x [Xteink X4 eReader](#)

Xteink X4 eReader

<https://www.xteink.com/products/xteink-x4>



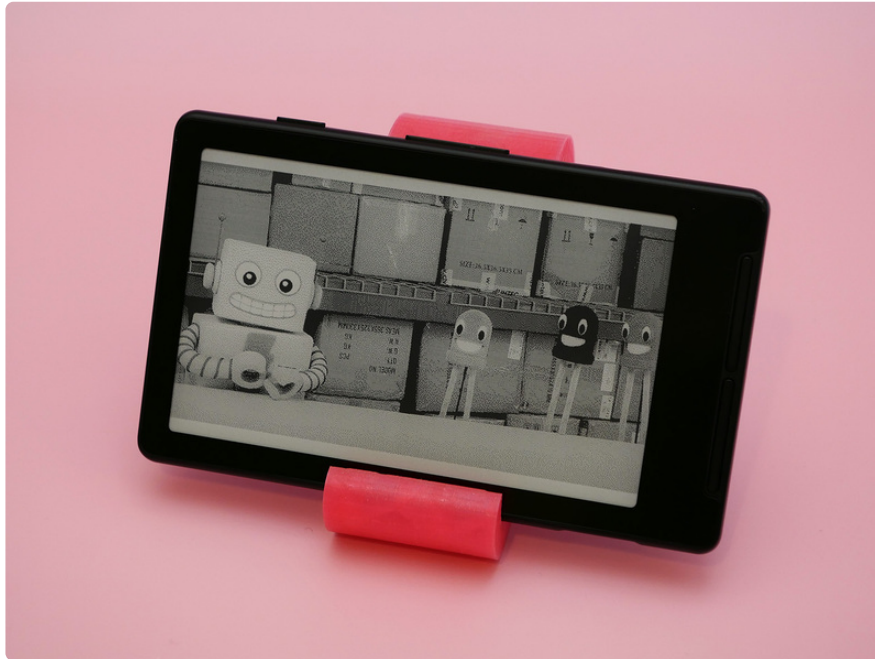
[USB Type A to Type C Cable - approx 1 meter / 3 ft long](#)

As technology changes and adapts, so does Adafruit. This USB Type A to Type C cable will help you with the transition to USB C, even if you're still...

<https://www.adafruit.com/product/4474>

---

# Pinouts



## ESP32-C3

The [ESP32-C3](https://adafru.it/1aBB) (<https://adafru.it/1aBB>) is the brains of the Xteink X4. It's an ultra low power chip from Espressif and is really popular in IoT devices. The only downside is it doesn't have any PSRAM and only has 400 KB of SRAM. However, the Xteink X4 has 16 MB of flash for a ton of storage.

## eInk Display

The Xteink X4 uses an SSD1677 controller with a [4.26" 800x480 GDEQ0426T82 display](https://adafru.it/1aBC) (<https://adafru.it/1aBC>). It is available as a built-in monochrome eInk display with the CircuitPython firmware. It is connected via SPI:

- SCLK: GPIO8 ( `board.EPD_SCK` )
- MOSI: GPIO10 ( `board.EPD_MOSI` )
- CS: GPIO21 ( `board.EPD_CS` )
- DC: GPIO4 ( `board.EPD_DC` )
- RST: GPIO5 ( `board.EPD_RST` )
- BUSY: GPIO6 ( `board.EPD_BUSY` )

## Buttons

There are 7 buttons available on the Xteink X4: the power button, up button, down button, right button, left button, confirm button and back button.

There is a bit of a twist though: only the power button is connected directly to a GPIO pin (**GPIO3** / `board.BUTTON`). The rest of the six buttons are read with a resistor ladder that are connected to two ADC pins (**GPIO1** / `board.BUTTON_ADC_1` and **GPIO2** / `board.BUTTON_ADC_1`). Each button reads a different set of analog values when pressed and that's how they are read in software. The [CircuitPython helper library](https://adafru.it/1aBD) (<https://adafru.it/1aBD>) takes care of reading the two ADC inputs.

## microSD Card

On the right side of the eReader is a microSD card. It uses the same SPI bus as the display and uses **GPIO12** (`board.SD_CS`) as its CS pin and **GPIO7** for MISO (`board.SD_MISO`).

## Battery Monitoring

You can monitor the battery voltage with **GPIO0** (`board.A0`). That pin is connected to the battery with a voltage divider. **GPIO20** (`board.D20`) also detects if the battery is charging via USB.

---

## Back-up the Firmware

Although this guide is all about installing new firmware on the Xteink X4, what about the stock firmware? You can use ESPTool to dump the flash contents of the ESP32-C3 and save it as a .BIN file. You can then reinstall this .BIN file later if you ever want to go back to the default firmware.

If you got too excited, though, and already installed CircuitPython, you can download the .BIN file below:

<https://adafru.it/1aBE>

## ESPTool

You can use ESPTool to read the flash over USB serial. After plugging in the Xteink X4 into your computer with a known data USB cable, use this command:

```
python -m esptool --chip esp32c3 --port COM4 read_flash 0x0 0x1000000 firmware_backup.bin
```

Replace `COM4` with the serial port for the Xteink X4 on your system. You'll see the `read_flash` program start running. There are 16 MB of flash on the eReader, so this will take a few minutes.

When it's done, you'll be prompted to reset the board. You should see the .BIN file in your file directory. Now you have a backup of the stock firmware that you can refer back to.

---

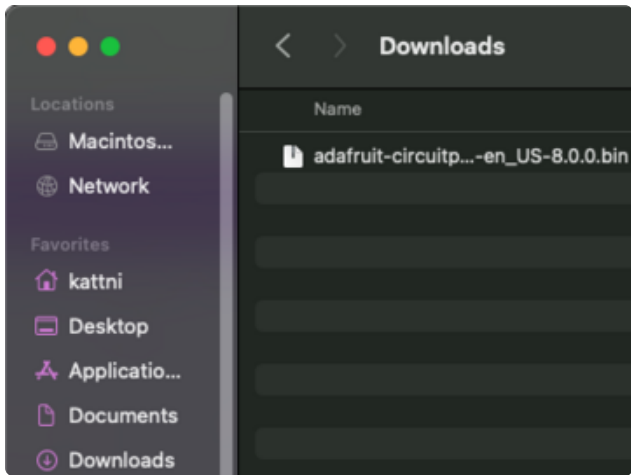
## Install CircuitPython

[CircuitPython \(https://adafru.it/tB7\)](https://adafru.it/tB7) is a derivative of [MicroPython \(https://adafru.it/f9W\)](https://adafru.it/f9W) 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. ESP32-C3 CircuitPython firmware is uploaded to the board via the USB JTAG serial port.

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

## CircuitPython Download

<https://adafru.it/1aBF>



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

Save it wherever is convenient for you.

## Connecting to the Web Flasher



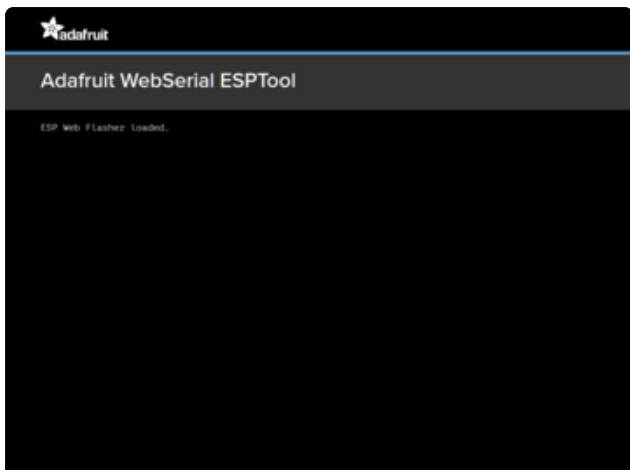
You do not need to put the Xteink X4 eReader into bootloader mode to upload a new .BIN file.

To begin, **plug your board into your computer via USB**, using a known-good data-sync cable, directly, or via an adapter if needed.

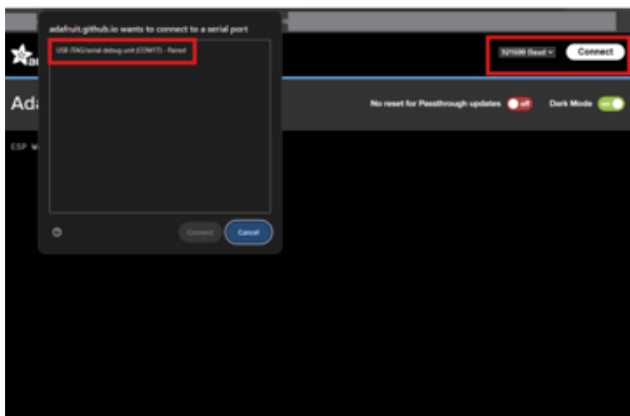
You will have to use Firefox 151 or later, Chrome or a Chromium-based browser to install CircuitPython. [For example, Edge and Opera are Chromium based \(https://adafru.it/10BL\)](https://adafru.it/10BL).

Safari and some other browsers etc. are not supported - [they have not implemented Web Serial \(https://adafru.it/10BM\)](https://adafru.it/10BM)!

In the **Chrome browser** visit [https://adafruit.github.io/Adafruit\\_WebSerial\\_ESPTool/](https://adafruit.github.io/Adafruit_WebSerial_ESPTool/) (<https://adafru.it/PMB>)



The main page of the ESP Web Flasher should look something like this.

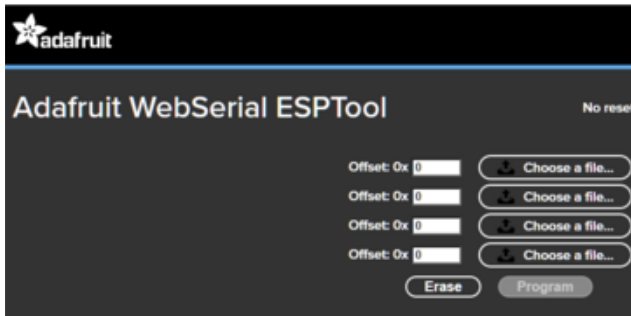


Press the **Connect** button in the top right of the web browser. You will get a pop up asking you to select the COM or Serial port. Look for **USB JTAG/Serial debug**.

On some systems, such as MacOS, there may be additional system ports that appear in the list.

```
esptool.js
Serial port WebSerial VendorID 0x303a ProductID 0x1001
Connecting... Detecting chip type... ESP32-C3
Chip is ESP32-C3 (revision 4)
Features: Wi-Fi,BLE
Crystal is 40MHz
MAC: 38:44:be:98:2f:04
Uploading stub...
Running stub...
Stub running...
Changing baudrate to 921600
Changed
```

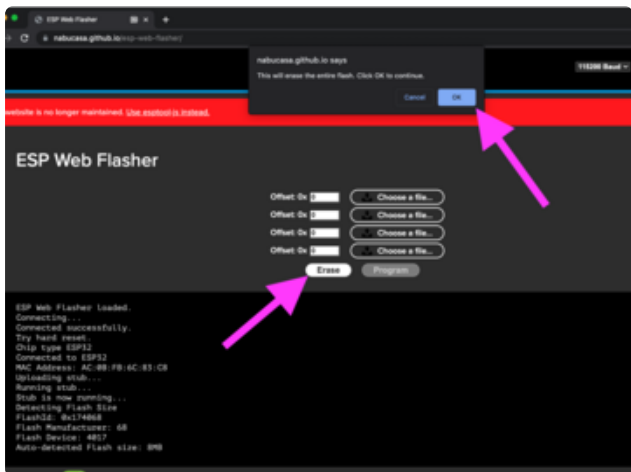
The Javascript code will now try to connect to the board. It may timeout for a bit until it succeeds. On success, you will see that it is **Connected** and will print out a unique **MAC address** identifying the board along with other information that was detected.



Once you have successfully connected, the file upload GUI will appear.

## Erasing the Board Contents

If you would like to erase the entire flash area so that you can start with a clean slate, you can use the erase feature. We recommend doing this every time before installing or updating CircuitPython.



To erase the contents, click the **Erase** button. You will be prompted as to whether you want to continue. Click **OK** to continue. If you do not wish to continue, click **Cancel**.

```

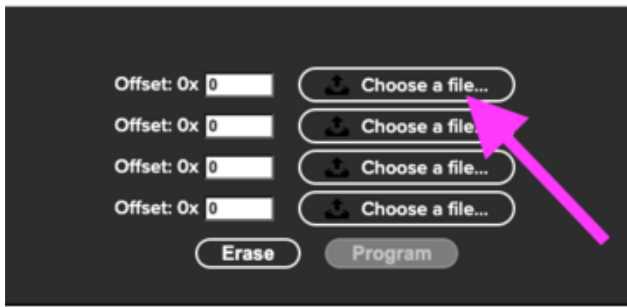
MAC Address: AC:0B:FB:6C:83:C8
Uploading stub...
Running stub...
Stub is now running...
Detecting Flash Size
FlashId: 0x174068
Flash Manufacturer: 68
Flash Device: 4017
Auto-detected Flash size: 8MB
Erasing flash memory. Please wait...
Finished. Took 5965ms to erase.

```

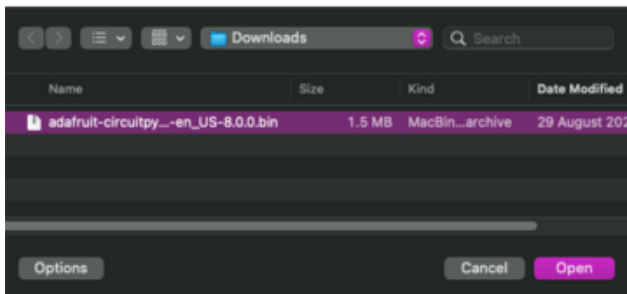
You'll see "Erasing flash memory. Please wait..." This will eventually be followed by "Finished." and the amount of time it took to erase.

**Do not disconnect!** Immediately continue on to Programming the Board.

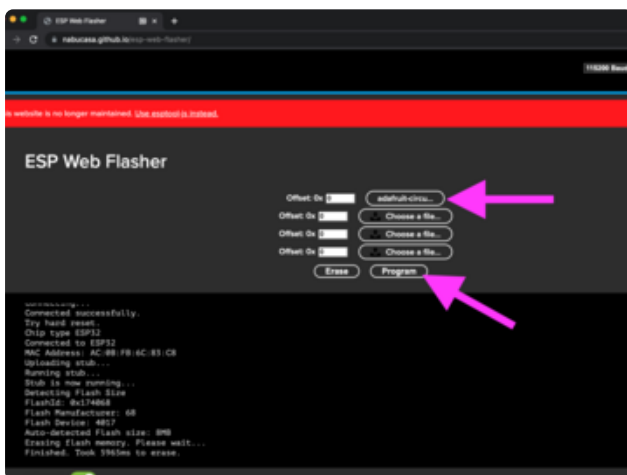
# Programming the Board



You can click on **Choose a file...** from any of the available buttons. It will only attempt to program buttons with a file and a unique location. Select the **.BIN** file you downloaded at the beginning of this page from the file chooser dialogue.



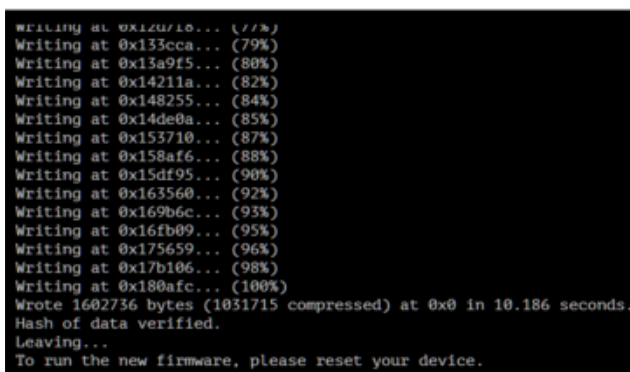
Verify that the **Offset** box next to the file location you used is 0x0. The offset defaults to 0x0, so unless you changed it manually, it should be good to go.



Once you choose a file, the button text will change to match your filename. You can then click the **Program** button to start flashing.



A progress bar will appear and after a minute or two, you will have written the firmware.



You've now successfully programmed CircuitPython onto your board! As suggested in the output, press reset to run the new firmware.

As the ESP32-C3 does not have native USB, no USB drive will show up on your computer when you reset. With CircuitPython firmware loaded, the REPL can be accessed over a serial/COM port.

Don't worry though! We have the [CircuitPython USB Workflow Code Editor \(https://adafru.it/1a6F\)](https://adafru.it/1a6F) or the [CircuitPython Web Workflow Code Editor \(https://adafru.it/18e8\)](https://adafru.it/18e8) so that you can access the board via USB or WiFi in your Chromium-based browser.

## Setting up Web Workflow

Web workflow is enabled when a file named **settings.toml** is added to the root folder of the CircuitPython file system. This file contains local wifi info and other settings. [More info here \(https://adafru.it/10Bp\)](https://adafru.it/10Bp).

Now, normally creating a file is super easy when the board shows up as a disk drive...but as mentioned before, that's not possible on the original ESP32 because it does not have native USB so it cannot show up as a disk drive. So we have to be a little more creative!

To start out, the minimal contents of this file are:

```
1 CIRCUITPY_WIFI_SSID = "wifissid"
2 CIRCUITPY_WIFI_PASSWORD = "wifipassword"
3 CIRCUITPY_WEB_API_PASSWORD= "webpassword"
```

with each item updated with local specifics.

- `wifissid` - replace with local wifi network name
- `wifipassword` - replace with local wifi network password
- `webpassword` - used when connecting to the board via web browser, set to whatever

There are a couple of options for creating this file.



Creating the `settings.toml` file will also enable the WiFi hardware, which may be off by default.

## Option 1: Create `settings.toml` file via REPL

The `settings.toml` file is simple enough that it can be created with a few simple Python commands directly via the REPL.

<https://adafru.it/Awz>

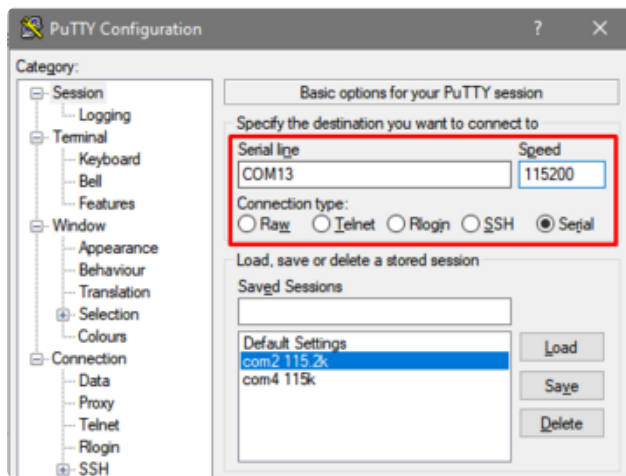
Here are the basic commands to use. **Note these need to be updated with local WiFi specifics. Note also where double quotes and single quotes are used.**

```

1 f = open('settings.toml', 'w')
2 f.write('CIRCUITPY_WIFI_SSID = "wifissid"\n')
3 f.write('CIRCUITPY_WIFI_PASSWORD = "wifipassword"\n')
4 f.write('CIRCUITPY_WEB_API_PASSWORD = "webpassword"\n')
5 f.close()

```

- Replace `wifissid` with the name of your local WiFi network
- Replace `wifipassword` with your local WiFi password
- The other password, `webpassword`, is used when you access the board via a web browser. Set this to whatever you want.



If you have a terminal program like PuTTY, minicom, screen or similar and you know how to use them - connect to the ESP32 board at the correct port name and 115200 baud

Once connected, you can press the **Reset** button to kick the firmware, then hit return a few times to get to the REPL prompt. Now you can copy and paste the lines above (with your correct SSID/password!)

Don't forget, ESP32 does not support 5 GHz networks, so use your 2.4 GHz SSID if you have two.

```
Board ID:adafruit_feather_esp32_v2
UID:C45752EBF2CA

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Hello World!

Code done running.

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

Adafruit CircuitPython 8.0.0-beta.6 on 2022-12-21; Adafruit Feather ESP32 V2 with ESP32
>>> f = open('settings.toml', 'w')
>>> f.write('CIRCUITPY_WIFI_SSID = "██████████"\n')
31
>>> f.write('CIRCUITPY_WIFI_PASSWORD = "██████████"\n')
37
>>> f.write('CIRCUITPY_WEB_API_PASSWORD = "esp32"\n')
37
>>> f.close()
>>>
```

Now press the **Reset** button again, you will see the ESP32 reboot. This time, it should get an IP address. If you have a fairly smart terminal program, the IP address will appear in the title bar.

```
192.168.0.123 | REPL | 8.0.0-beta.6
Adafruit CircuitPython 8.0.0-beta.6 on 2022-12-21; Adafruit Feather ESP32 V2 with ESP32
Board ID:adafruit_feather_esp32_v2
UID:C45752EBF2CA

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Hello World!

Code done running.

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

Adafruit CircuitPython 8.0.0-beta.6 on 2022-12-21; Adafruit Feather ESP32 V2 with ESP32
>>>
```

Alternatively, or if you want more details - you can always query the board over the REPL to ask it the MAC address and IP address:

```
1 import wifi
2
3 print("My MAC addr: %02X:%02X:%02X:%02X:%02X:%02X" % tuple(wifi.radio.mac_address))
4 print("My IP address is", wifi.radio.ipv4_address)
```

```

code.py output:
Hello World!

Code done running.

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

Adafruit CircuitPython 8.0.0-beta.6 on 2022-12-21; Adafruit Feather ESP32 V2 with ESP32
>>> import wifi
>>> print("My MAC addr: %02X:%02X:%02X:%02X:%02X:%02X" % tuple(wifi.radio.mac_address))
My MAC addr: 4C:75:25:BE:2F:AC
>>> print("My IP address is", wifi.radio.ipv4_address)
My IP address is 192.168.0.123
>>>

```

If that's not working either, try asking what SSIDs it thinks it's seeing to make sure you have no typos!

```

1 import wifi
2
3 print("Available WiFi networks:")
4 for n in wifi.radio.start_scanning_networks():
5     print("\t%s\t\tRSSI: %d\tChannel: %d" % (str(n.ssid, "utf-8"), n.rssi, n.channel))
6 wifi.radio.stop_scanning_networks()

```

Note that since this code has an indented part in it, you shouldn't just paste it in directly into the REPL because it won't handle the tab/spaces correctly. Instead, once you hit **Return** to get to the REPL, type in **Control-E** to enter paste mode . Then paste in the text and type **Control-D** to finish and run

```

code.py output:
Hello World!

Code done running.

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

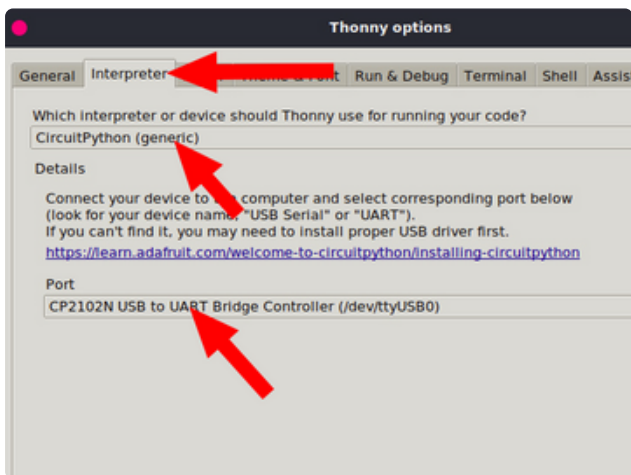
Adafruit CircuitPython 8.0.0-alpha.1-147-g05d2013b4 on 2022-08-18; Adafruit Feather ESP32 V2 with ESP32
>>>
paste mode; Ctrl-C to cancel, Ctrl-D to finish
=== import wifi
===
=== print("Available WiFi networks:")
=== for n in wifi.radio.start_scanning_networks():
===     print("\t%s\t\tRSSI: %d\tChannel: %d" % (str(n.ssid, "utf-8"), n.rssi, n.channel))
=== wifi.radio.stop_scanning_networks()
===
Available WiFi networks:
adafruit          RSSI: -58          Channel: 6
patrick's Network RSSI: -76          Channel: 6
Fios-N2VnJ        RSSI: -70          Channel: 1
VVCBR            RSSI: -88          Channel: 1
ohana            RSSI: -93          Channel: 1
MySpectrumWiFi173-2G RSSI: -54          Channel: 11
Sally            RSSI: -82          Channel: 11
Patrick          RSSI: -86          Channel: 11
Sonos            RSSI: -92          Channel: 11
SNET             RSSI: -93          Channel: 11
Verizon_9DWHL4   RSSI: -96          Channel: 11
linksys_SES_2868 RSSI: -74          Channel: 3
Meatlocker       RSSI: -85          Channel: 5
Fios-K97GI       RSSI: -73          Channel: 7
Fios-K97GI       RSSI: -88          Channel: 7
>>>

```

Check that your SSID appears properly - if not, maybe its 5 GHz, maybe its not typed correctly, etc!

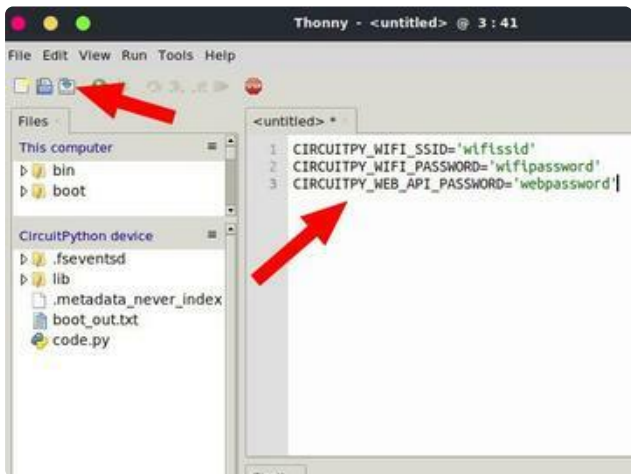
## Option 2: Create **settings.toml** file using Thonny

This option requires installing additional software - the [Thonny Python IDE \(https://adafruit.it/Qb6\)](https://adafruit.it/Qb6). Thonny provides file access and a text editor which makes creating the **settings.toml** file a little more streamlined than using direct REPL commands. **This technique requires Thonny 4.0.0 or later.**



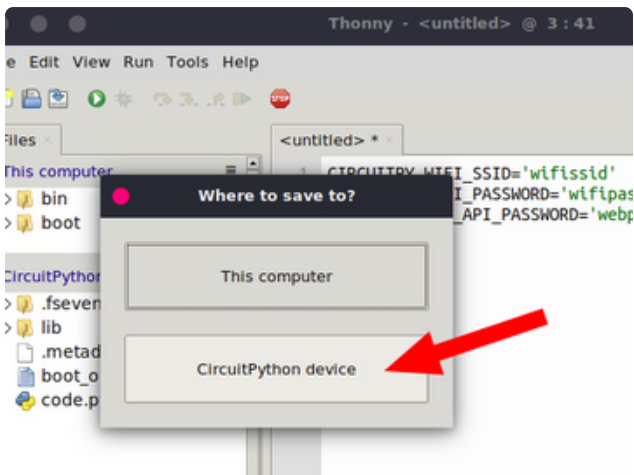
In Thonny, open the **Tools -> Options** dialog and select the **Interpreter** tab.

Set interpreter to **CircuitPython (generic)** and the COM port as needed.



Enter the file contents in the editor window. Update **wifissid** etc. as needed.

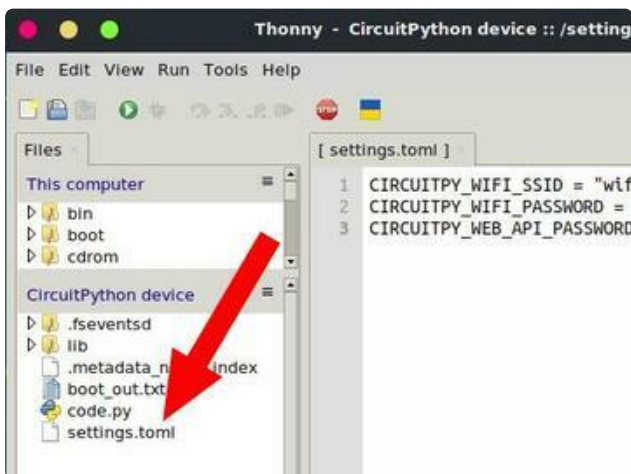
Click save.



Select CircuitPython Device



Save the file as **settings.toml**.



Now the **settings.toml** file shows up on the CircuitPython device.

## Connecting via Web Browser

Once the web workflow is enabled and the board connects to the local wifi network with an IP address, the board can be accessed by opening a web browser and entering the board's network address. But what is the address to enter?

## Connect using MDNS Address

The CircuitPython web workflow uses [MDNS \(https://adafru.it/10BW\)](https://adafru.it/10BW) so that connecting to the board can be as simple as using the address `circuitpython.local`. Try using that first and if it works, great. Open a web browser and navigate to:

```
1 http://circuitpython.local
```

Or just click the button below:

<https://adafru.it/10BX>

## Connect using IP Address

If MDNS does not work for some reason, the fallback approach is to use the actual IP address, like `192.168.0.121`, the CircuitPython board was assigned when it connected to the local wifi network.

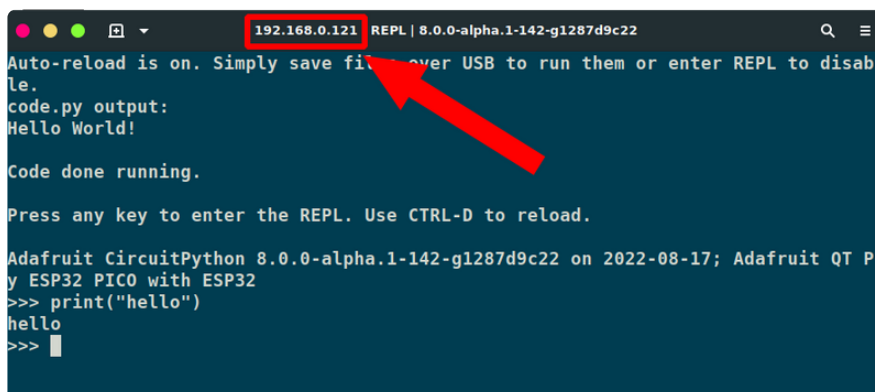
```
1 http://192.168.0.121
```

There are several options for determining this address.

### Terminal Window Title Bar

The web workflow serial output includes some special escape sequences that are intended to update the window title bar of the terminal program being used.

For example, here's what a terminal window running screen to connect to the board looks like:

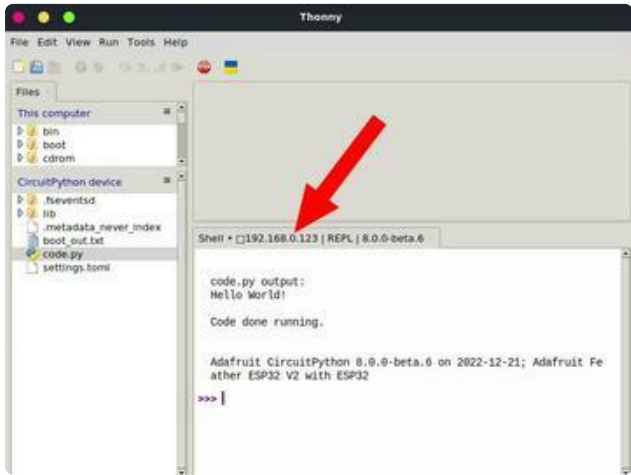


## Via REPL

The IP address can be obtained fairly easily via the REPL with a few commands:

```
1 >>> import wifi
2 >>> wifi.radio.ipv4_address
3 192.168.0.121
4 >>>
```

## Serial output in Thonny



In Thonny, look for the IP address in the banner text of the serial output in the Shell window.

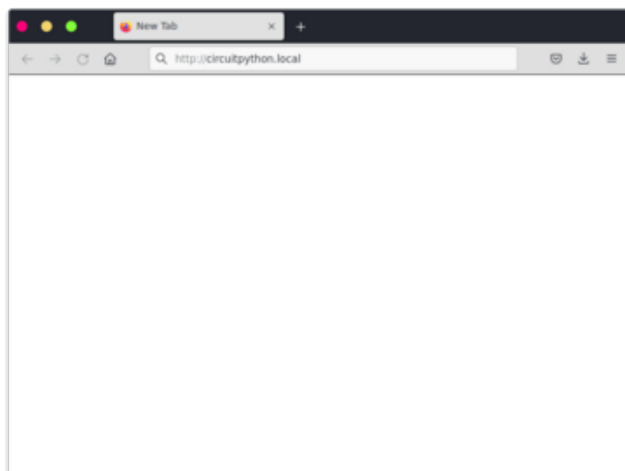
Note the **settings.toml** file has been added to the CircuitPython device.

---

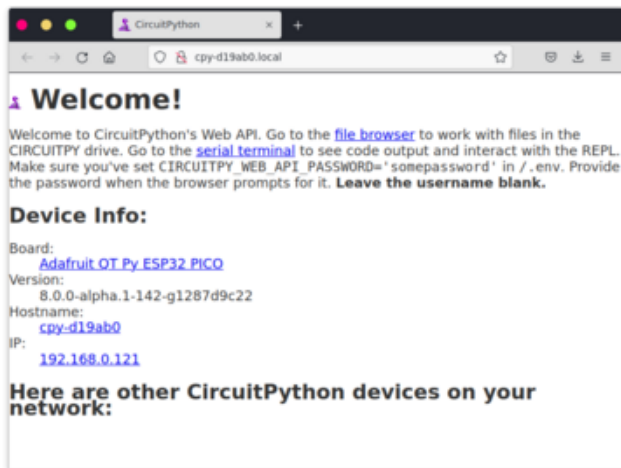
## Using Web Workflow

### Welcome! Page

This is the initial page seen when first connecting.



Launch a web browser and navigate to **circuitpython.local** OR the numeric IP address like **192.168.0.121**



The browser should find the board and show the **Welcome!** screen.

## Password Entry

When first accessing things from the **Welcome!** page, a password must be entered.

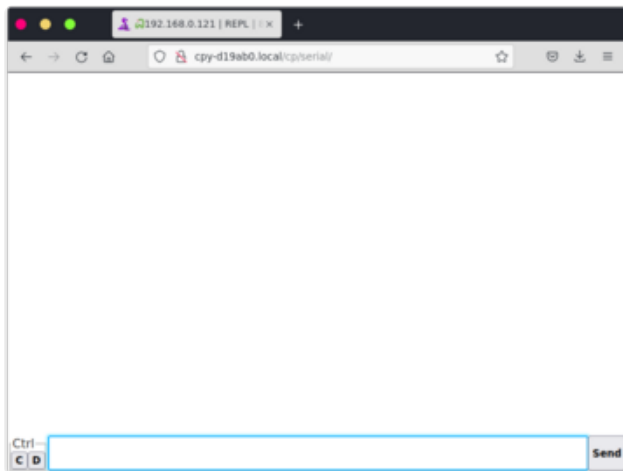


Enter the password that was setup for `CIRCUITPY_WEB_API_PASSWORD` in the `settings.toml` file.

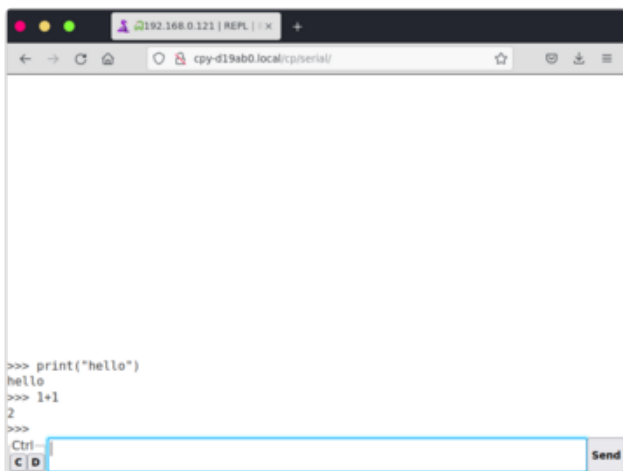
Leave Username blank.

## Serial Terminal

From the Welcome! page, click the **serial terminal** link to access the serial output as well as REPL for entering commands.



The serial window initially looks blank.

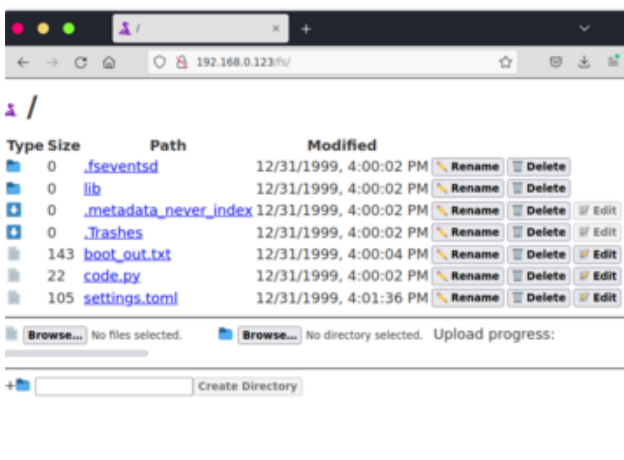


Commands can be entered in the input field at the bottom.

The results will be shown above and scroll up.

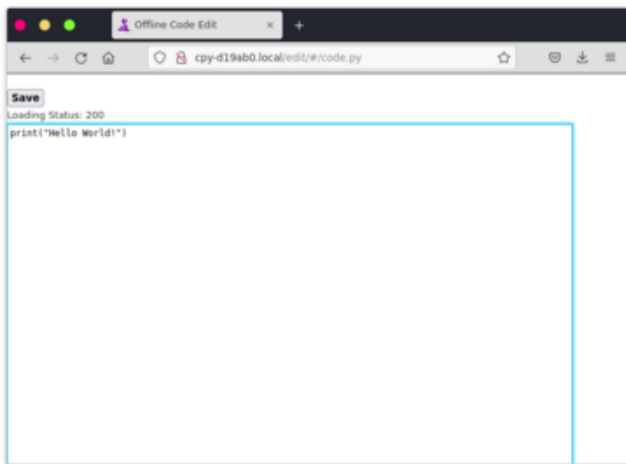
## File Browser

From the Welcome! page, click the **file browser** link on the Welcome! page to access files and folders.

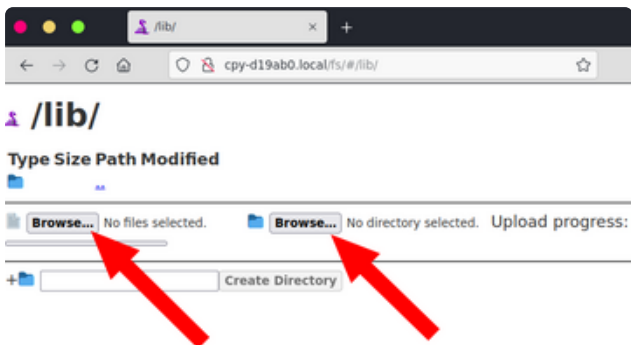


The files and folders should show up.

There are buttons for various actions.



For example, click the **Edit** button for `code.py` bring up a simple editor to allow changing the contents.



Library files and folders from the [Bundle \(https://adafru.it/ENC\)](https://adafru.it/ENC) can be uploaded to the `/lib` folder using the **Browse...** buttons.

There's a separate button for files and folders (directories).

---

## CircuitPython Helper Library



As mentioned on the Pinouts page, six of the seven buttons on the Xteink X4 are connected to a resistor ladder read by two analog pins. The Xteink X4 helper library makes reading those buttons a lot easier. There's also a few helper functions for reading the battery voltage.

## Library Usage with Web Workflow

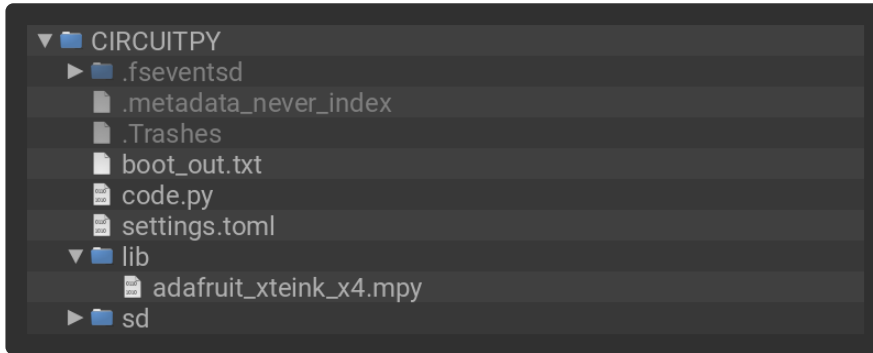
To use with CircuitPython, you need to first install the Xteink X4 library into the `lib` folder on the Xteink X4. Then you need to update `code.py` with the example script.

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.

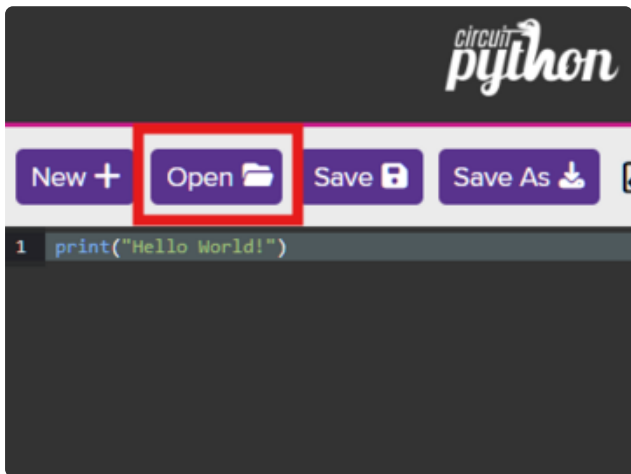
```
1 # SPDX-FileCopyrightText: Copyright (c) 2026 Liz Clark for Adafruit Industries
2 #
3 # SPDX-License-Identifier: MIT
4
5 """
6 Xteink X4 Helper Demo
7
8 Uses button inputs and battery monitor
9 """
10
11 from adafruit_xteink_x4 import BatteryMonitor, InputManager
12
13 battery = BatteryMonitor()
14 buttons = InputManager()
15
16 print(f"Battery: {battery.percentage}% ({battery.volts:.2f}V)")
17 print()
18
19 while True:
20     buttons.update()
21
22     if buttons.any_pressed:
23         for i in range(7):
24             if buttons.was_pressed(i):
25                 print(f"Pressed: {buttons.button_name(i)}")
26
27     if buttons.any_released:
28         for i in range(7):
29             if buttons.was_released(i):
30                 held = buttons.held_time
31                 print(f"Released: {buttons.button_name(i)} (held {held:.2f}s)")
32                 print(f"Battery: {battery.percentage}% ({battery.volts:.2f}V)")
```

[https://github.com/adafruit/Adafruit\\_CircuitPython\\_Xteink\\_X4/blob/main/examples/xteink\\_x4\\_simpletest.py](https://github.com/adafruit/Adafruit_CircuitPython_Xteink_X4/blob/main/examples/xteink_x4_simpletest.py)

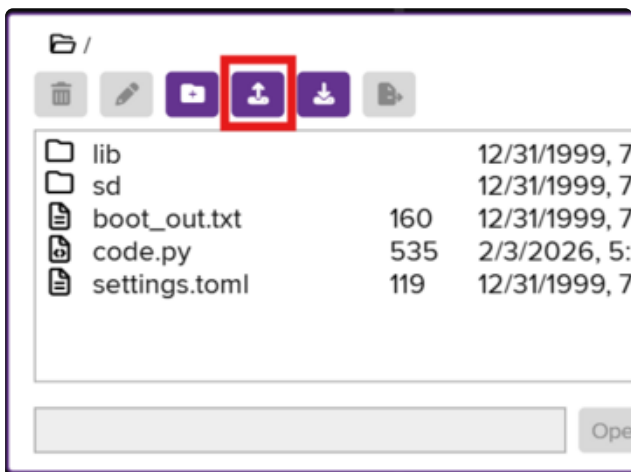
Extract the contents of the zip file. You'll see the following contents in the extracted folder:



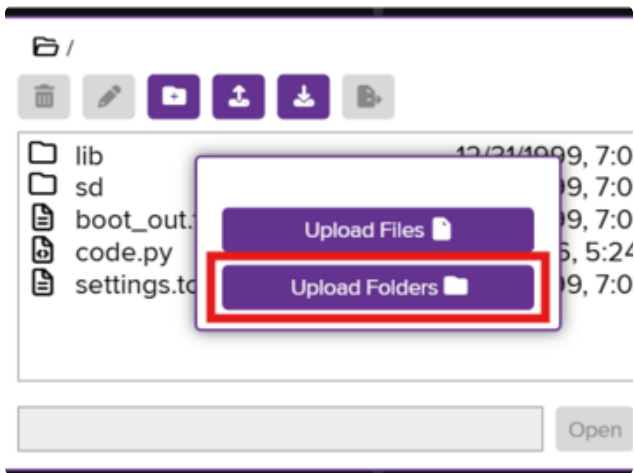
## Upload the Library



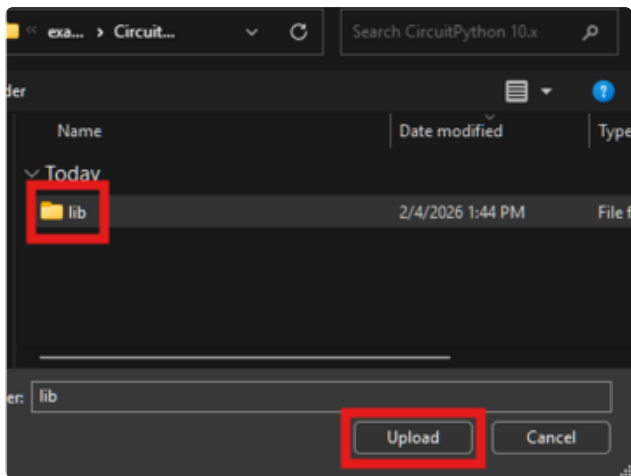
In the [Web Workflow Code Editor \(https://adafru.it/10QF\)](https://adafru.it/10QF), click the **Open** button.



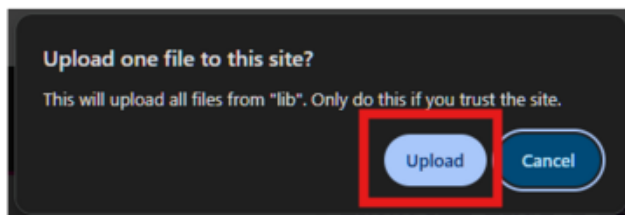
This opens the file browser. Click the **Upload** button.



Click on **Upload Folders**.



Select the `/lib` folder in the Project Bundle and click **Upload**.

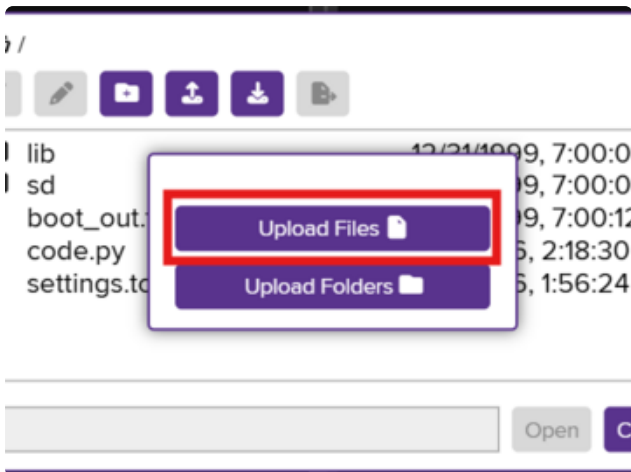


You'll be asked to confirm the folder upload. Click **Upload** to continue.

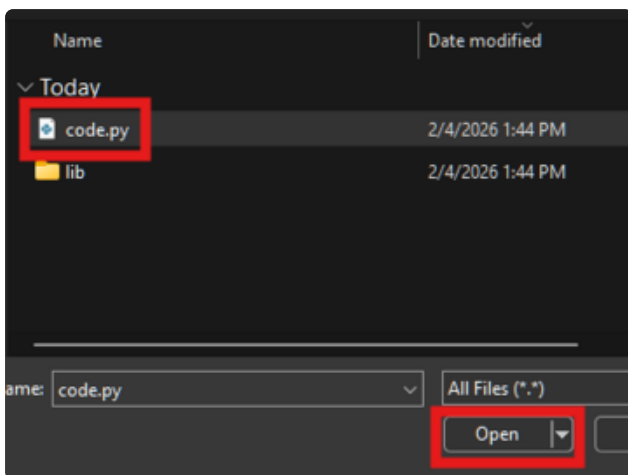


After the upload finishes, you'll be able to see the library in the `/lib` folder on the device.

## Upload code.py

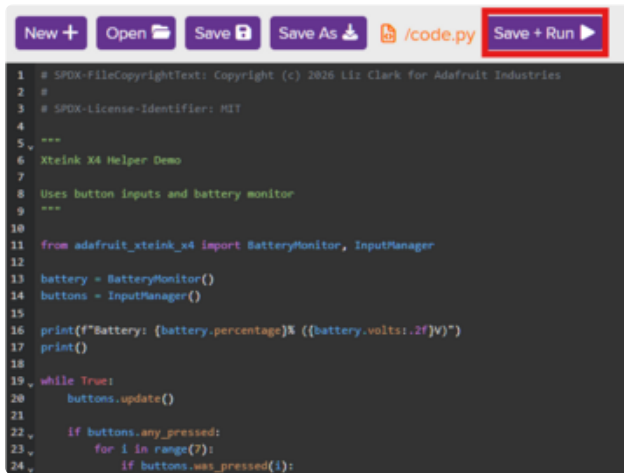


Next, you'll upload the `code.py` file to the Xteink X4. Click the **Upload** button and then **Upload Files**.



Select the `code.py` file from the Project Bundle and then click **Open**.

## Run the Code



```
1 # SPDX-FileCopyrightText: Copyright (c) 2026 Liz Clark for Adafruit Industries
2 #
3 # SPDX-License-Identifier: MIT
4
5 """
6 Xteink X4 Helper Demo
7
8 Uses button inputs and battery monitor
9 """
10
11 from adafruit_xteink_v4 import BatteryMonitor, InputManager
12
13 battery = BatteryMonitor()
14 buttons = InputManager()
15
16 print(f"Battery: {battery.percentage}% ({battery.volts:.2f}V)")
17 print()
18
19 while True:
20     buttons.update()
21
22     if buttons.any_pressed:
23         for i in range(7):
24             if buttons.was_pressed(i):
```

To run the new `code.py` file, click **Save + Run**.

After running the code, you'll see the battery voltage print out before going into the loop. In the loop, the button you press will print out to the serial monitor along with the battery voltage. You'll see the CircuitPython REPL on the elnk display.

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Battery: 100% (4.17V)

Pressed: Back
Released: Back (held 0.20s)
Battery: 100% (4.18V)
Pressed: Confirm
Released: Confirm (held 0.21s)
Battery: 100% (4.17V)
Pressed: Left
Released: Left (held 0.21s)
Battery: 100% (4.17V)
Pressed: Right
Released: Right (held 0.37s)
Battery: 100% (4.17V)
Pressed: Down
Released: Down (held 0.21s)
Battery: 100% (4.17V)
Pressed: Up
Released: Up (held 1.09s)
Battery: 100% (4.16V)
Pressed: Right
Released: Right (held 0.96s)
Battery: 100% (4.16V)
```

---

# eInk Display



The Xteink X4 has a lovely 800x480 eInk display. That's a lot of pixels! The display can be accessed as `board.DISPLAY` in CircuitPython or you can use the [SSD1677 driver](https://adafru.it/1aBG) (<https://adafru.it/1aBG>). The demo code below lets you show a different dithered image starring the Circuit Playground characters every time you press one of the buttons on the eReader.

## Library Usage with Web Workflow

To use with CircuitPython, you need to first install the libraries into the `lib` folder and upload the bitmap images onto the Xteink X4. Then you need to update `code.py` with the example script.

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.

```
1 # SPDX-FileCopyrightText: 2026 Liz Clark for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """
5 Xteink X4 bitmap test
6 """
7 import os
8 import board
9 import displayio
10 from adafruit_xteink_x4 import InputManager
```

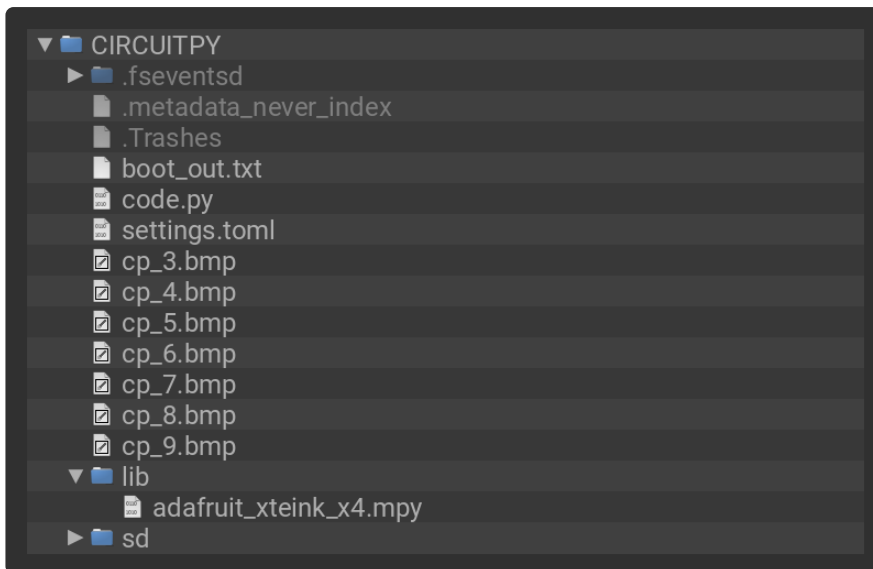
```

11
12 display = board.DISPLAY
13
14 groups = []
15 images = []
16 for filename in os.listdir('/'):
17     if filename.lower().endswith('.bmp') and not filename.startswith('.'):
18         images.append("/"+filename)
19 print(images)
20
21 for i in range(len(images)):
22     splash = displayio.Group()
23     bitmap = displayio.OnDiskBitmap(images[i])
24     tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
25     splash.append(tile_grid)
26     groups.append(splash)
27
28 index = 0
29
30 display.root_group = groups[index]
31 display.refresh()
32
33 buttons = InputManager()
34
35 while True:
36     buttons.update()
37
38     if buttons.any_pressed:
39         for i in range(7):
40             if buttons.was_pressed(i):
41                 index = i
42
43     if buttons.any_released:
44         for i in range(7):
45             if buttons.was_released(i):
46                 held = buttons.held_time
47                 print(f"Released: {buttons.button_name(i)} (held {held:.2f}s)")
48                 print("updating display..")
49                 display.root_group = groups[index]
50                 display.refresh()
51                 print(f"showing {images[index]}")

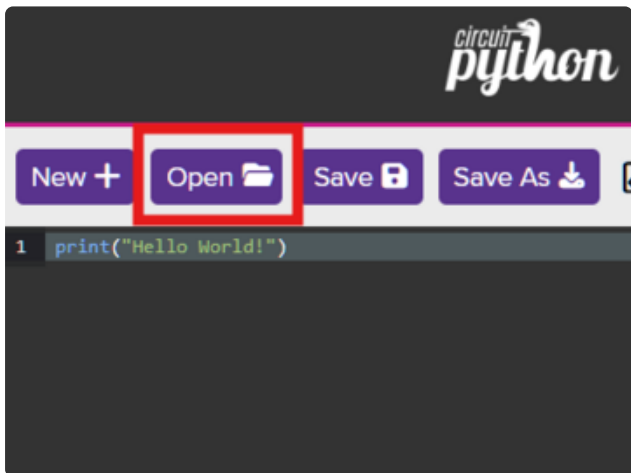
```

[https://github.com/adafruit/Adafruit\\_Learning\\_System\\_Guides/blob/main/Xteink\\_X4\\_Examples/Xteink\\_X4\\_eInk\\_Demo/code.py](https://github.com/adafruit/Adafruit_Learning_System_Guides/blob/main/Xteink_X4_Examples/Xteink_X4_eInk_Demo/code.py)

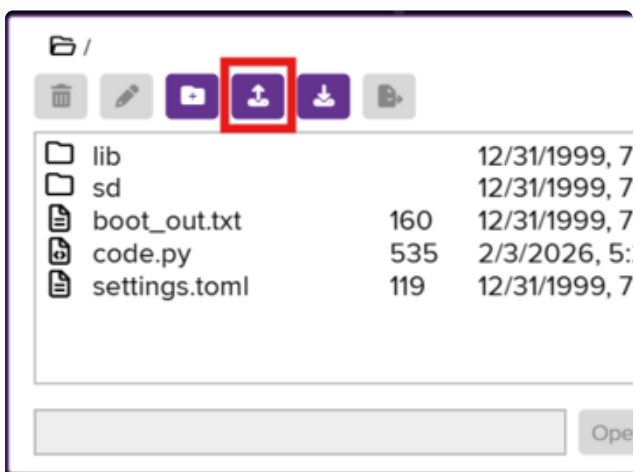
Extract the contents of the zip file. You'll see the following contents in the extracted folder:



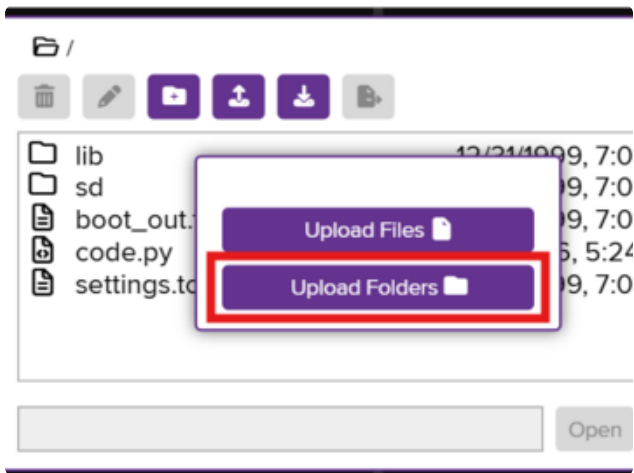
## Upload the Library File



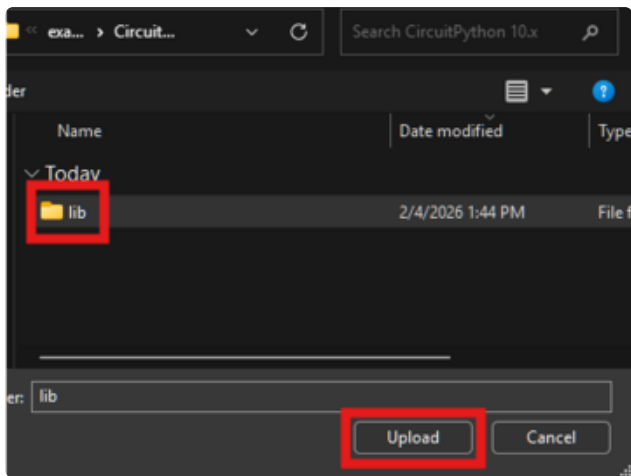
In the [Web Workflow Code Editor \(https://adafru.it/10QF\)](https://adafru.it/10QF), click the **Open** button.



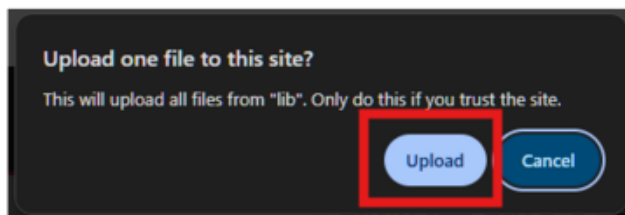
This opens the file browser. Click the **Upload** button.



Click on **Upload Folders**.



Select the `/lib` folder in the Project Bundle and click **Upload**.

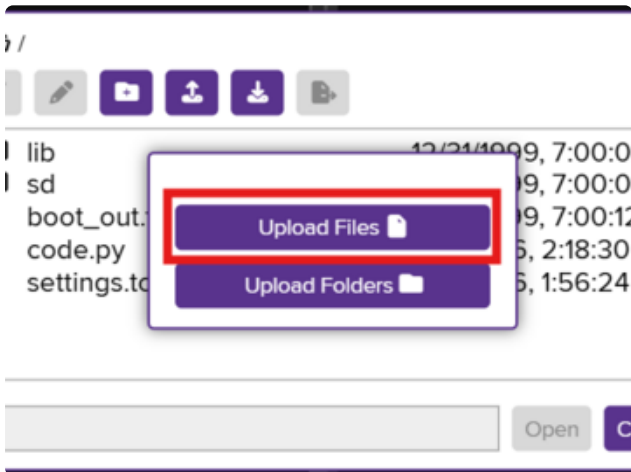


You'll be asked to confirm the folder upload. Click **Upload** to continue.

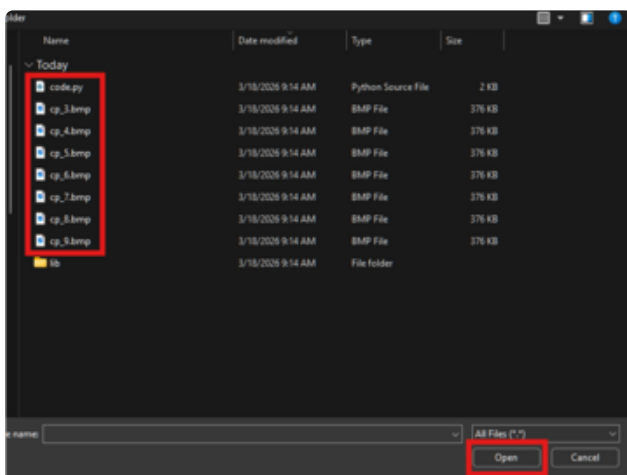


After the upload finishes, you'll be able to see the library in the `/lib` folder on the device.

## Upload the Code and Bitmaps

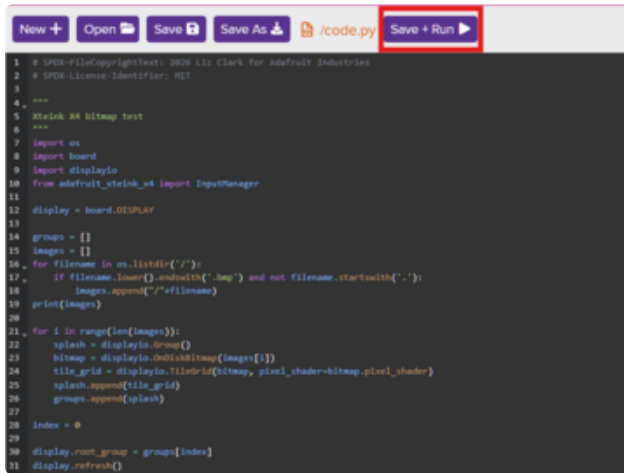


Next, you'll upload the `code.py` file and bitmap files to the Xteink X4. Click the **Upload** button and then **Upload Files**.



Select the `code.py` file and the **.BMP** files from the Project Bundle and then click **Open**.

## Run the Code



```
1 # SPDX-FileCopyrightText: 2024 LLC Clark for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 ---
5 Xteink X4 bitmap test
6 ---
7 import os
8 import board
9 import displayio
10 from adafruit_xteink_x4 import InputManager
11
12 display = board.DISPLAY
13
14 groups = []
15 images = []
16 for filename in os.listdir('/'):
17     if filename.lower().endswith('.bmp') and not filename.startswith('.'):
18         images.append("/"+filename)
19 print(images)
20
21 for i in range(len(images)):
22     splash = displayio.Group()
23     bitmap = displayio.on_disk_bitmap(images[i])
24     tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
25     splash.append(tile_grid)
26     groups.append(splash)
27
28 index = 0
29 display.root_group = groups[index]
30 display.refresh()
31
```

To run the new `code.py` file, click **Save + Run**.

The seven bitmap images are loaded as `OnDiskBitmap` s. In the loop, the button you press will update the display to show the associated bitmap image.

---

## Weather Demo



Everyone loves a weather display and especially one on a chonky elnk display with low power usage. This weather project is based on the [MagTag Daily Weather Forecast Display Learn Guide by Carter Nelson](https://adafru.it/PB-) (<https://adafru.it/PB->) with a few adjustments to better suit the Xteink X4.

Since the display is so big, the graphics have to match in size and multiple sprite sheets will quickly get you into `MemoryError` territory on an ESP32-C3. In this version, only today's weather is shown but it is still using the [OpenMeteo API \(https://adafru.it/18Aa\)](https://adafru.it/18Aa) and [deep\\_sleep \(https://adafru.it/Pyc\)](https://adafru.it/Pyc) to save battery.

## Library Usage with Web Workflow

To use with CircuitPython, you need to first install the libraries into the `lib` folder and the `bmp` folder onto the Xteink X4. Then you need to update `code.py` with the example script.

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.

```
1 # SPDX-FileCopyrightText: 2026 Liz Clark for Adafruit Industries
2 #
3 # SPDX-License-Identifier: MIT
4 # pylint: disable=redefined-outer-name, eval-used, wrong-import-order, unsubscriptable-object
5
6 """
7 Xteink X4 Weather Display Demo
8 Based on MagTag Weather by Carter Nelson
9 """
10
11 import time
12 import os
13 import board
14 import alarm
15 import displayio
16 import adafruit_imageload
17 import ssl
18 import wifi
19 import socketpool
20 import adafruit_requests
21 from adafruit_bitmap_font import bitmap_font
22 from adafruit_display_text import label
23 import gc
24
25 gc.collect()
26
27 display = board.DISPLAY
28 display.rotation = 270
29 DISPLAY_WIDTH = display.width
30
31 try:
32     wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'), os.getenv('CIRCUITPY_WIFI_PASSWORD'))
33 except TypeError:
34     print("Could not find WiFi info. Check your settings.toml file!")
35     raise
36
37 # --| USER CONFIG |-----
38 LAT = 40.7128 # latitude
```

```

39 LON = -74.0060 # longitude
40 TMZ = "America/New_York" # https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
41 METRIC = False # set to True for metric units
42 CITY = "New York, NY" # optional
43 # -----
44
45 pool = socketpool.SocketPool(wifi.radio)
46 requests = adafruit_requests.Session(pool, ssl.create_default_context())
47 URL = f"https://api.open-meteo.com/v1/forecast?latitude={LAT}&longitude={LON}&"
48 URL += "daily=weather_code,temperature_2m_max,temperature_2m_min"
49 URL += ",sunrise,sunset"
50 URL += "&timeformat=unixtime"
51 URL += f"&timezone={TMZ}"
52 gc.collect()
53 resp_data = requests.get(URL)
54 #resp_data = get_forecast()
55 print("got url")
56 forecast_data = resp_data.json()
57
58 # -----
59 # Define various assets
60 # -----
61 gc.collect()
62 font_file = "/fonts/Arial-Bold-24.bdf"
63 font = bitmap_font.load_font(font_file)
64 BACKGROUND_BMP = "/bmps/weather_bg_vert.bmp"
65 ICONS_LARGE_FILE = "/bmps/weather-icons.bmp"
66 DAYS = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
67 MONTHS = (
68     "January",
69     "February",
70     "March",
71     "April",
72     "May",
73     "June",
74     "July",
75     "August",
76     "September",
77     "October",
78     "November",
79     "December",
80 )
81
82 # Weather Code Information from https://open-meteo.com/en/docs
83 # Code Description
84 # 0 Clear sky
85 # 1, 2, 3 Mainly clear, partly cloudy, and overcast
86 # 45, 48 Fog and depositing rime fog
87 # 51, 53, 55 Drizzle: Light, moderate, and dense intensity
88 # 56, 57 Freezing Drizzle: Light and dense intensity
89 # 61, 63, 65 Rain: Slight, moderate and heavy intensity
90 # 66, 67 Freezing Rain: Light and heavy intensity
91 # 71, 73, 75 Snow fall: Slight, moderate, and heavy intensity
92 # 77 Snow grains
93 # 80, 81, 82 Rain showers: Slight, moderate, and violent
94 # 85, 86 Snow showers slight and heavy

```

```

95 # 95 * Thunderstorm: Slight or moderate
96 # 96, 99 * Thunderstorm with slight and heavy hail
97
98 # Map the above WMO codes to index of icon in 3x3 spritesheet
99 WMO_CODE_TO_ICON = (
100     (0,), # 0 = sunny
101     (1,), # 1 = partly sunny/cloudy
102     (2, 3, 45, 48,), # 2 = cloudy/very cloudy/fog
103     (61, 63, 65, 51, 53, 55, 80, 81, 82), # 4 = rain/showers
104     (95, 96, 99), # 6 = storms
105     (56, 57, 66, 67, 71, 73, 75, 77, 85, 86), # 7 = snow
106 )
107
108 # -----
109 # Background bitmap
110 # -----
111 gc.collect()
112 splash = displayio.Group()
113 bitmap = displayio.OnDiskBitmap(BACKGROUND_BMP)
114 tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
115 splash.append(tile_grid)
116 display.root_group = splash
117 print("got background")
118
119 # -----
120 # Weather icons sprite sheet
121 # -----
122 gc.collect()
123 icons_large_bmp, icons_large_pal = adafruit_imageload.load(ICONS_LARGE_FILE)
124 print("got icon sheet")
125
126 # //////////////////////////////////////
127 # helper functions
128
129 def temperature_text(tempC):
130     if METRIC:
131         return "{:3.0f}C".format(tempC)
132     else:
133         return "{:3.0f}F".format(32.0 + 1.8 * tempC)
134
135
136 def update_today(data):
137     """Update today weather info."""
138     # date text
139     s = data["daily"]["time"][0] + data["utc_offset_seconds"]
140     t = time.localtime(s)
141     today_day.text = "{}".format(
142         DAYS[t.tm_wday].upper()
143     )
144     print(today_day.text)
145     today_date.text = "{} {}, {}".format(
146         MONTHS[t.tm_mon - 1].upper(), t.tm_mday, t.tm_year
147     )
148     # weather icon
149     w = data["daily"]["weather_code"][0]
150     today_icon[0] = next(i for i, t in enumerate(WMO_CODE_TO_ICON) if w in t)
151     # temperatures

```

```

151     today_temp.text = f"H: {temperature_text(data['daily']['temperature_2m_max'][0])} "
152     today_temp.text += f"L: {temperature_text(data['daily']['temperature_2m_min'][0])}"
153     # sunrise/set
154     sr = time.localtime(data["daily"]["sunrise"][0] + data["utc_offset_seconds"])
155     ss = time.localtime(data["daily"]["sunset"][0] + data["utc_offset_seconds"])
156     today_sunrise.text = "{:2d}:{:02d} AM".format(sr.tm_hour, sr.tm_min)
157     today_sunset.text = "{:2d}:{:02d} PM".format(ss.tm_hour - 12, ss.tm_min)
158
159     # =====
160     # U I
161     # =====
162     print("making ui")
163     today_day = label.Label(font, text="?" * 30, color=0x000000)
164     today_day.anchor_point = (0.5, 0)
165     today_day.anchored_position = (DISPLAY_WIDTH / 2, 106)
166     today_date = label.Label(font, text="?" * 30, color=0x000000)
167     today_date.anchor_point = (0.5, 0)
168     today_date.anchored_position = (DISPLAY_WIDTH / 2, 140)
169
170     location_name = label.Label(font, color=0x000000)
171     if CITY:
172         location_name.text = f"{CITY[:16]}"
173     else:
174         location_name.text = f"({LAT},{LON})"
175     location_name.anchor_point = (0.5, 0)
176     location_name.anchored_position = (DISPLAY_WIDTH / 2, 210)
177
178     today_icon = displayio.TileGrid(
179         icons_large_bmp,
180         pixel_shader=icons_large_pal,
181         x=203,
182         y=275,
183         width=1,
184         height=1,
185         tile_width=74,
186         tile_height=74,
187     )
188     today_icon.x = int(DISPLAY_WIDTH / 2 - today_icon.tile_width / 2)
189
190     today_temp = label.Label(font, text="H: +100F", color=0x000000)
191     today_temp.anchor_point = (0, 0)
192     today_temp.anchored_position = (163, 415)
193
194     today_sunrise = label.Label(font, text="12:12 PM", color=0x000000)
195     today_sunrise.anchor_point = (0, 0)
196     today_sunrise.anchored_position = (202, 520)
197
198     today_sunset = label.Label(font, text="12:12 PM", color=0x000000)
199     today_sunset.anchor_point = (0, 0)
200     today_sunset.anchored_position = (202, 614)
201     today_banner = displayio.Group()
202     today_banner.append(today_day)
203     today_banner.append(today_date)
204     today_banner.append(location_name)
205     today_banner.append(today_icon)
206     today_banner.append(today_temp)

```

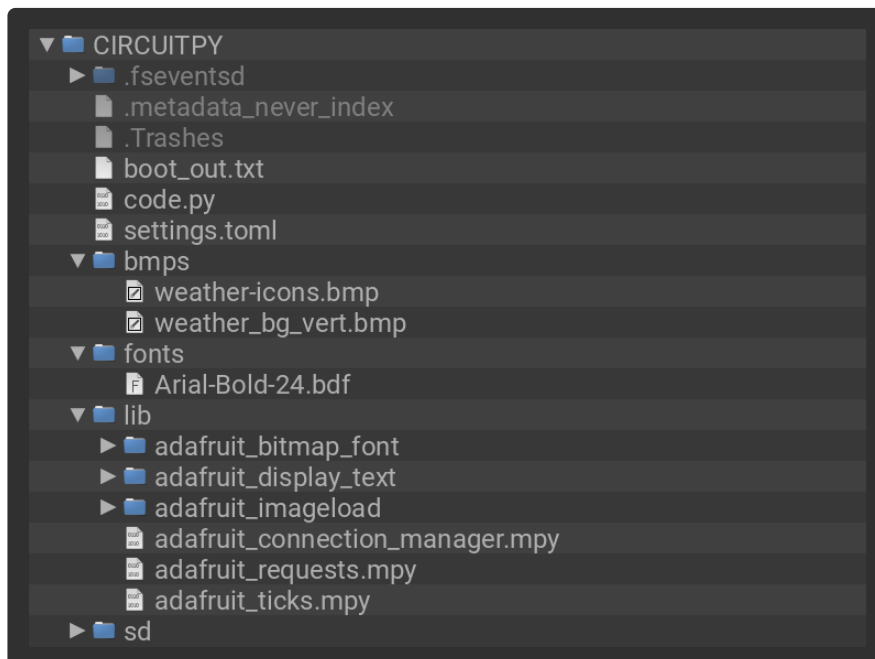
```

207 today_banner.append(today_sunrise)
208 today_banner.append(today_sunset)
209
210 display.root_group.append(today_banner)
211
212 # =====
213 # M A I N
214 # =====
215 gc.collect()
216 print("Updating...")
217 update_today(forecast_data)
218
219 print("Refreshing...")
220 time.sleep(display.time_to_refresh + 1)
221 display.refresh()
222 time.sleep(display.time_to_refresh + 1)
223
224 print("Sleeping...")
225 wake_alarm = alarm.wake_alarm
226 pin_alarm = alarm.pin.PinAlarm(pin=board.BUTTON, value=False, pull=True)
227 alarm.exit_and_deep_sleep_until_alarms(pin_alarm)
228 # entire code will run again

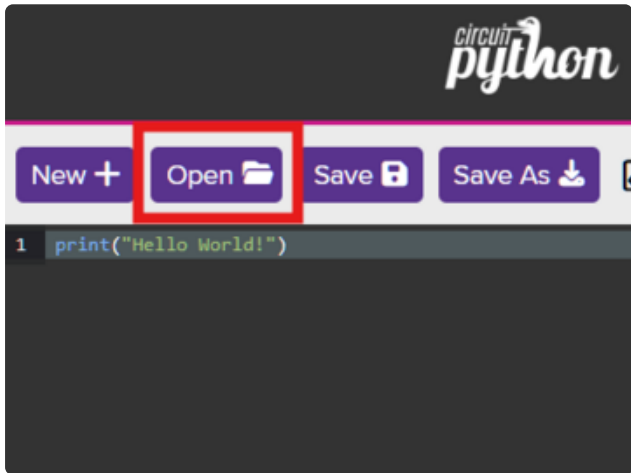
```

[https://github.com/adafruit/Adafruit\\_Learning\\_System\\_Guides/blob/main/Xteink\\_X4\\_Examples/Xteink\\_X4\\_Weather/code.py](https://github.com/adafruit/Adafruit_Learning_System_Guides/blob/main/Xteink_X4_Examples/Xteink_X4_Weather/code.py)

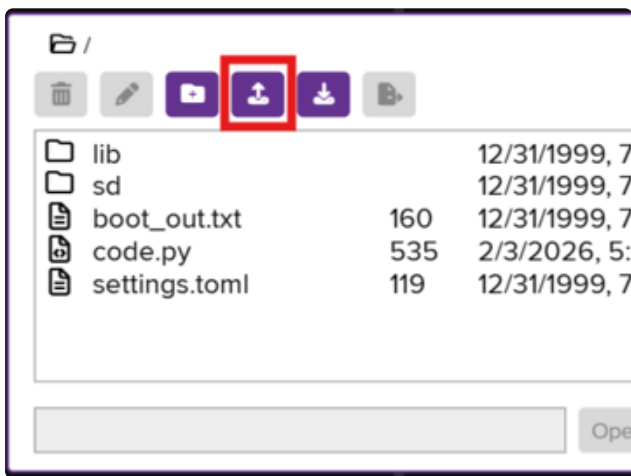
Extract the contents of the zip file. You'll see the following contents in the extracted folder:



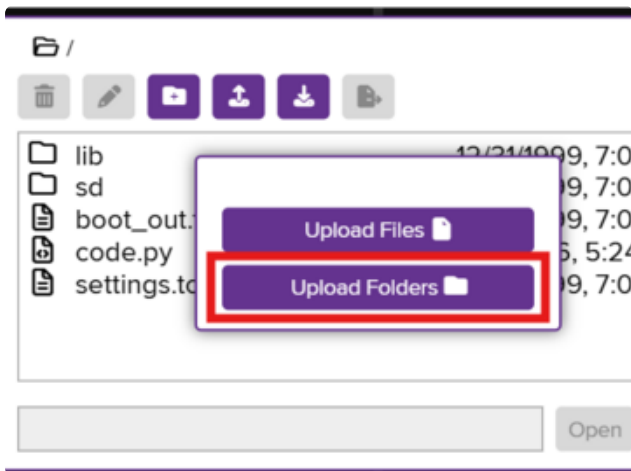
## Upload the Libraries



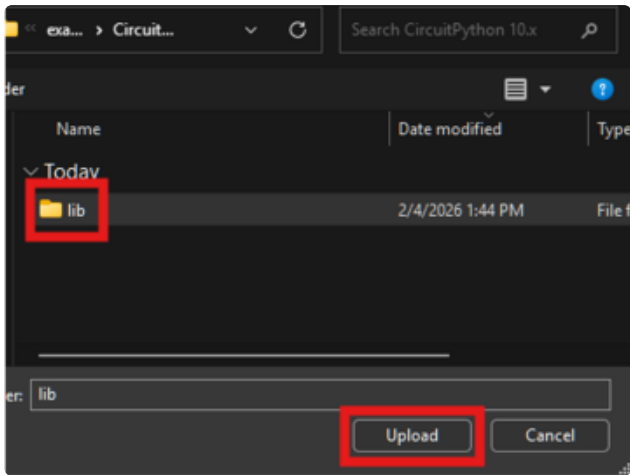
In the [Web Workflow Code Editor](https://adafru.it/10QF) (<https://adafru.it/10QF>), click the **Open** button.



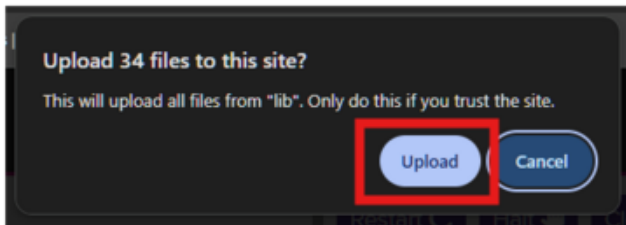
This opens the file browser. Click the **Upload** button.



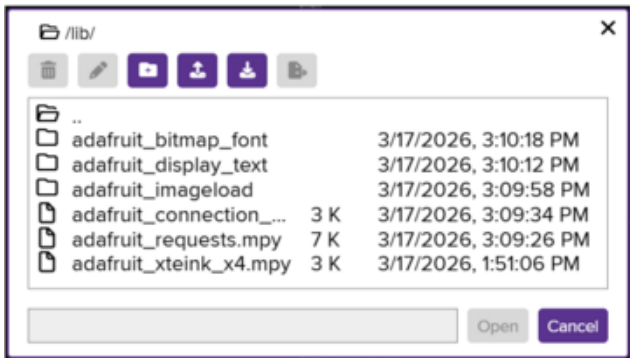
Click on **Upload Folders**.



Select the `/lib` folder in the Project Bundle and click **Upload**.



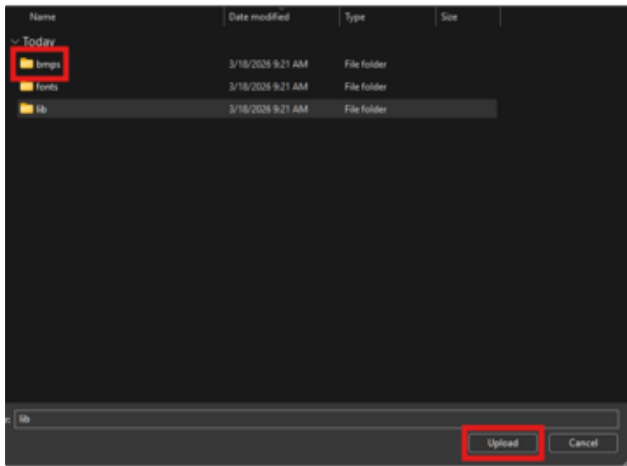
You'll be asked to confirm the folder upload. Click **Upload** to continue.



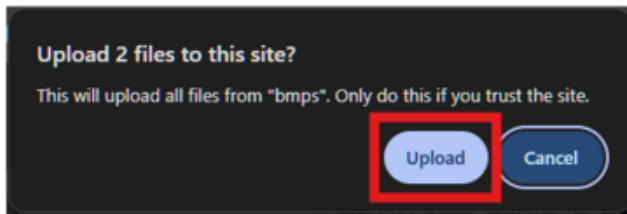
After the upload finishes, you'll be able to see the library in the `/lib` folder on the device.

## Upload the Bitmaps and Fonts

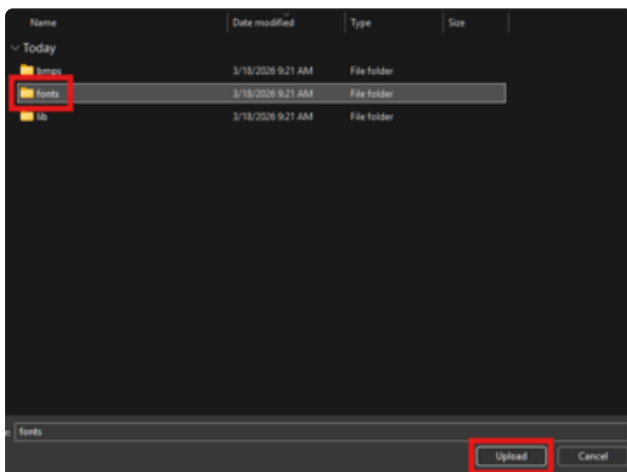
Next, you'll upload the `/fonts` and `/bmps` folders.



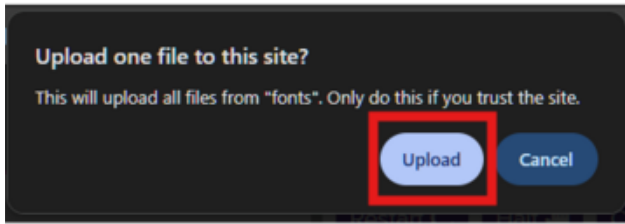
Select **Upload Folders** and then select the **/bmps** folder in the Project Bundle. Click **Upload**.



You'll be asked to confirm the folder upload. Click **Upload** to continue.

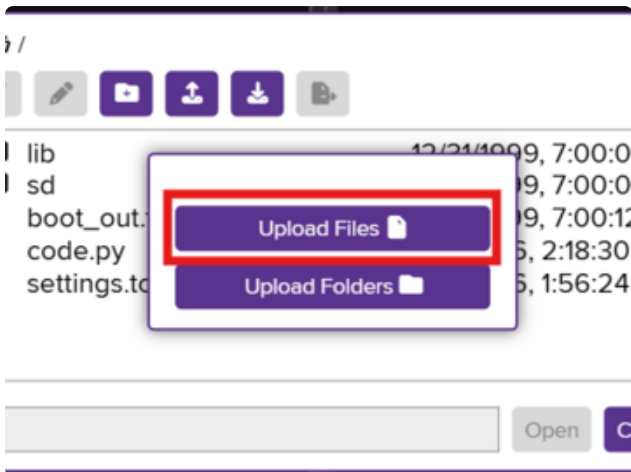


Select **Upload Folders** and then select the **/fonts** folder in the Project Bundle. Click **Upload**.

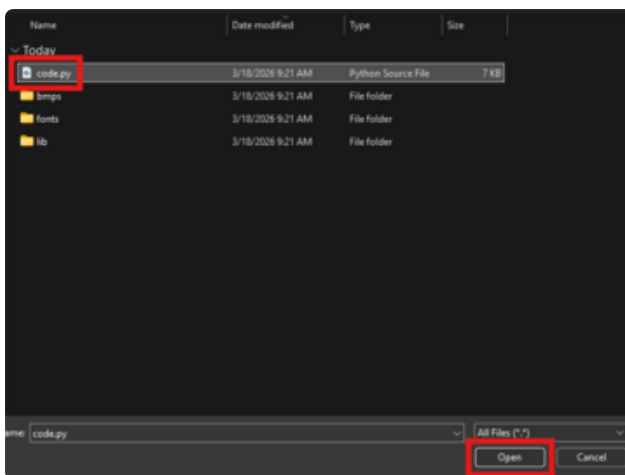


You'll be asked to confirm the folder upload. Click **Upload** to continue.

## Upload code.py

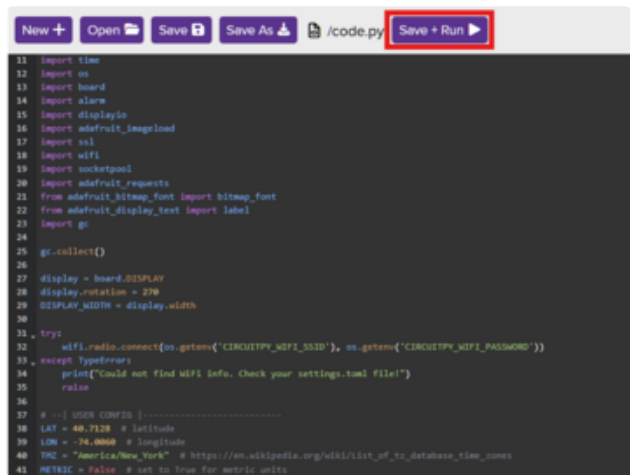


Next, you'll upload the **code.py** file to the Xteink X4. Click the **Upload** button and then **Upload Files**.



Select the **code.py** file from the Project Bundle and then click **Open**.

## Run the Code



```
11 import time
12 import os
13 import board
14 import alarm
15 import displayio
16 import adafruit_imageload
17 import ssl
18 import wifi
19 import socketpool
20 import adafruit_requests
21 from adafruit_bitmap_font import bitmap_font
22 from adafruit_display_text import label
23 import gc
24
25 gc.collect()
26
27 display = board.DISPLAY
28 display.rotation = 270
29 DISPLAY_WIDTH = display.width
30
31 try:
32     wifi.radio.connect(os.getenv("CIRCUITPY_WIFI_SSID"), os.getenv("CIRCUITPY_WIFI_PASSWORD"))
33 except TypeError:
34     print("Could not find WiFi info. Check your settings.toml file!")
35     raise
36
37 # --[ USER CONFIG ]-----
38 LAT = 40.7128 # latitude
39 LONG = -74.0060 # longitude
40 TIME = "America/New_York" # https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
41 METRIC = False # set to True for metric units
```

To run the new `code.py` file, click **Save + Run**.

The code connects to WiFi, requests weather information from Open-Meteo, populates the different text elements with the information from the API and then creates a display buffer with the bitmap images and text elements to show on the eInk display. After this, the Xteink X4 goes into deep sleep. You can wake up the Xteink X4 by pressing the power button.