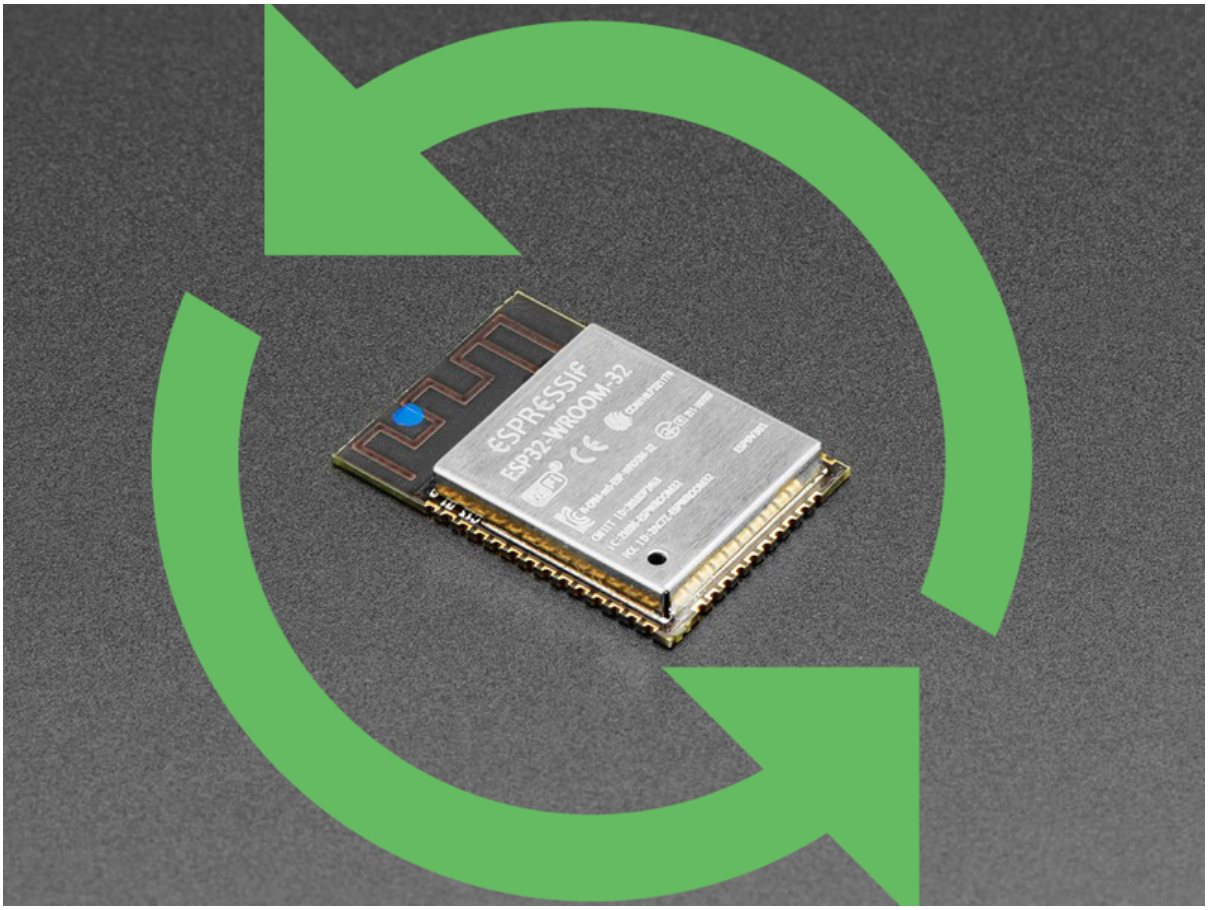
Upgrading ESP32 Firmware

Created by Brent Rubell



https://learn.adafruit.com/upgrading-esp32-firmware

Last updated on 2024-06-03 02:55:34 PM EDT

Table of Contents

Overview

If you want to keep the firmware on your ESP32 WiFi-BLE co-processor up-to-date, you'll need to update the firmware on the ESP32.

You're going to to **turn your board into a USB-to-Serial converter** to **flash new firmware to your ESP32 - no extra hardware required**!

This process is mostly setup and should take from 10 to 20 minutes.

This guide is not for when you are running Arduino/MicroPython/FreeRTOS/etc *directly* on the ESP32, this is only for using the ESP32 as an AirLift/WiFi co-processor!

To support BLE on the ESP32 AirLift, you'll need NINA_W102-1.7.1.bin or later.

Why would I update my ESP32's firmware?

Using an ESP32 as a WiFi-BLE co-processor is a way to connect your CircuitPython and Arduino projects to the internet. Having WiFi managed by a separate chip means your code is simpler, you don't have to cache socket data, or compile in & debug an SSL library.

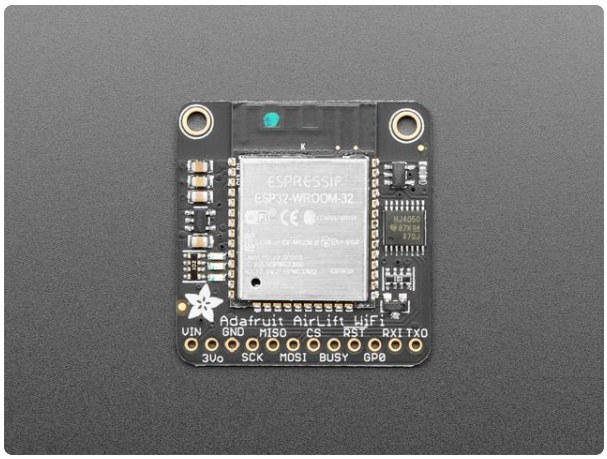
Adafruit ships a variety of products which use the ESP32 as a WiFi-BLE co-processor with a variant of the Arduino nina-fw core (https://adafru.it/FWj). This firmware is programmed to the ESP32 at the Adafruit factory. If you wish to update to a newer version of nina-fw, you'll need to program it to the ESP32.

BLE is supported on the ESP32 co-processor only with version NINA_W102-1.7.1.bin or later of the firmware (released in October 2020). If you want BLE support, it is quite likely you'll need to upgrade

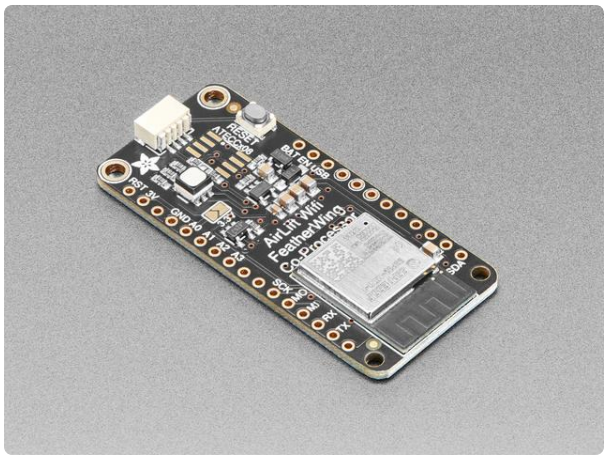
Parts

External ESP32 Co-Processors

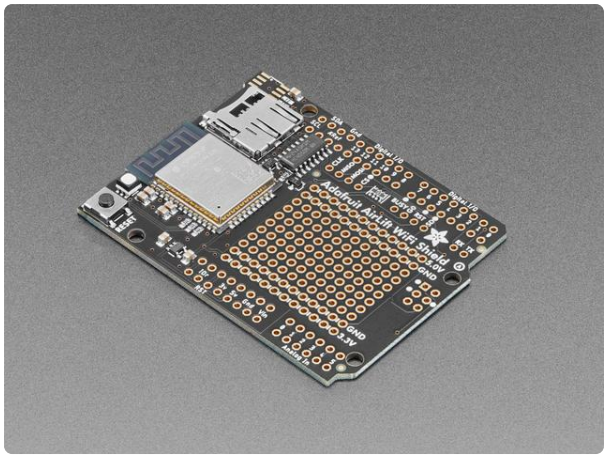
If you already have a project which uses a popular microcontroller (like the ATMega328 or ATSAMD51), you can easily add WiFi by using an externally connected ESP32 module.



[Adafruit AirLift – ESP32 WiFi Co-Processor Breakout Board](https://www.adafruit.com/product/4201)
Give your plain ol' microcontroller project a lift with the Adafruit AirLift - a breakout board that lets you use the powerful ESP32 as a WiFi co-processor. You probably...
https://www.adafruit.com/product/4201



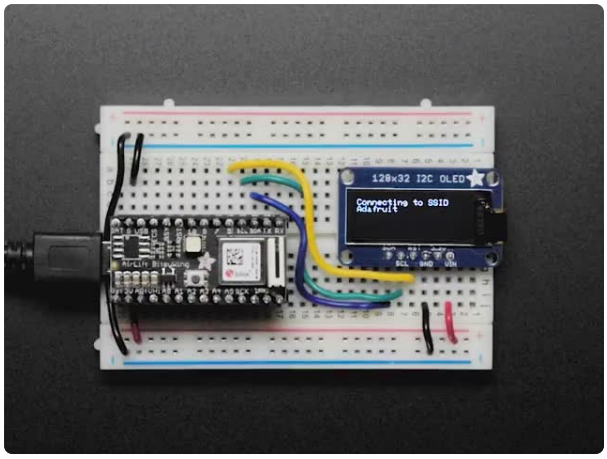
[Adafruit AirLift FeatherWing – ESP32 WiFi Co-Processor](https://www.adafruit.com/product/4264)
Give your Feather project a lift with the Adafruit AirLift FeatherWing - a FeatherWing that lets you use the powerful ESP32 as a WiFi co-processor. You probably have your...
https://www.adafruit.com/product/4264



[Adafruit AirLift Shield - ESP32 WiFi Co-Processor](https://www.adafruit.com/product/4285)

Give your Arduino project a lift with the Adafruit AirLift Shield - a shield that lets you use the powerful ESP32 as a WiFi co-processor. You probably have your favorite...

https://www.adafruit.com/product/4285



[Adafruit AirLift Bitsy Add-On – ESP32 WiFi Co-Processor](https://www.adafruit.com/product/4363)

Give your ItsyBitsy project a lift with the Adafruit AirLift Bitsy Add-On - a daughterboard that lets you use the powerful ESP32 as a WiFi co-processor. You probably have your...

https://www.adafruit.com/product/4363

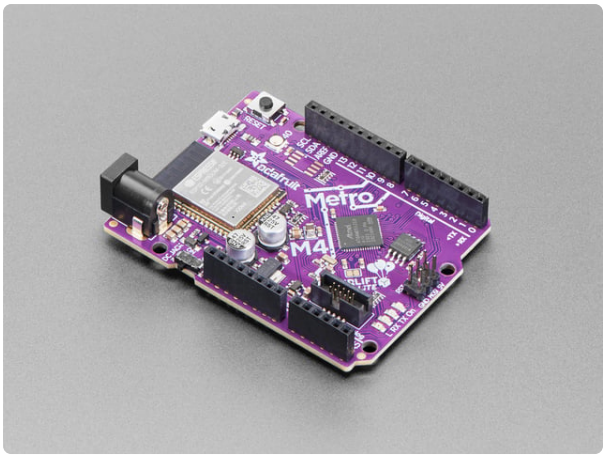
ESP32 Co-Processor All-in-One Boards

Don't want to add extra hardware to your project? Consider grabbing a board which has an ESP32 WiFi co-processor built-in.

[Adafruit PyPortal - CircuitPython Powered Internet Display](https://www.adafruit.com/product/4116)

PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...

https://www.adafruit.com/product/4116



Adafruit Metro M4 Express AirLift (WiFi) - Lite

Give your next project a lift with AirLift - our witty name for the ESP32 co-processor that graces this Metro M4. You already know about the Adafruit Metro... https://www.adafruit.com/product/4000

Materials

1 x USB Cable https://www.adafruit.com/product/592

USB cable - USB A to Micro-B - 3 foot long

Upgrade All-in-One ESP32 AirLift Firmware

Upload Passthrough Code

First, you'll need to upload the code below to allow your board to act as a programmer for the ESP32 AirLift module.

Back up any code and files on your CIRCUITPY drive. The code will overwrite the drive's contents. You should not end up losing any files on the QSPI flash, but it's a good idea to back them up anyways.

Download the **UF2** file for your board and save it to your computer's Desktop.

Matrix Portal M4 ESP32 Passthrough TinyUSB 2023-07-30 UF2

https://adafru.it/19EJ

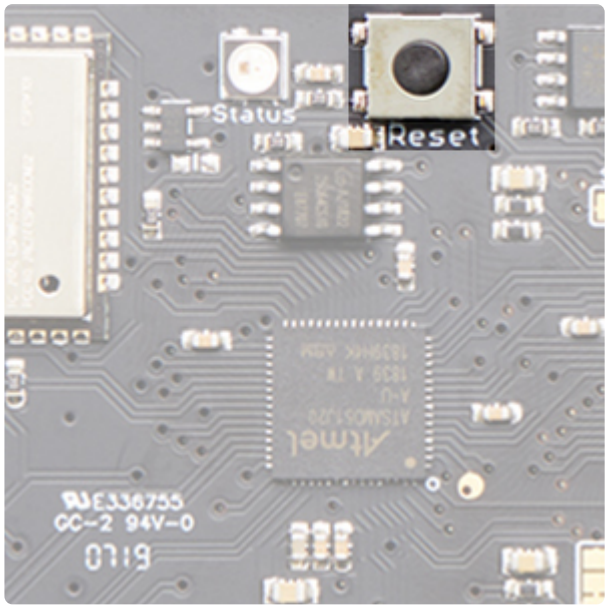
Metro M4 AirLift ESP32 Passthrough TinyUSB 2023-07-30 UF2

https://adafru.it/19EK

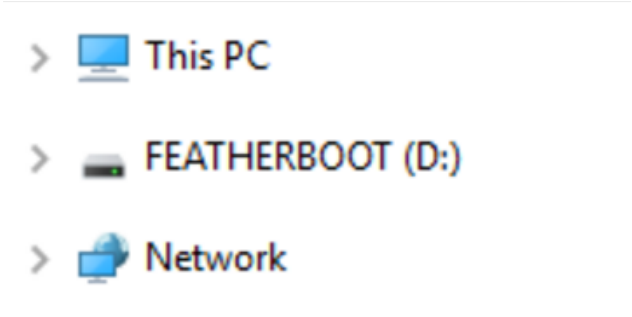
PyPortal M4 ESP32 Passthrough TinyUSB 2023-07-30 UF2

https://adafru.it/19EL

To enter bootloader mode, start with your board unplugged from USB. Next, find the reset button on your board. It's a small, black button, and on most of the boards, it will be the only button available.

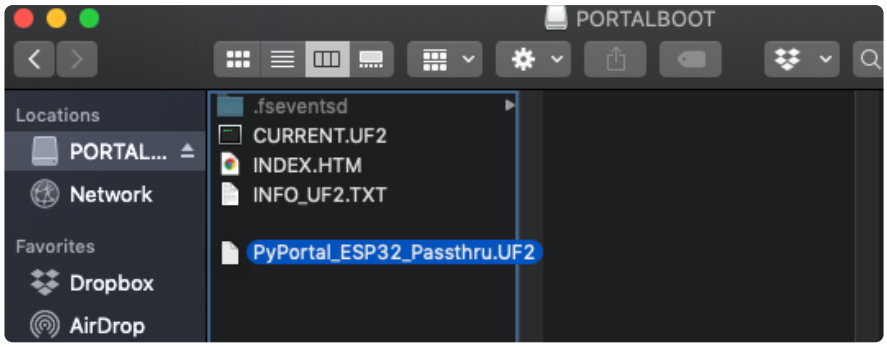


Once successful, the RGB LED on the board will flash red and then stay green. A new drive will show up on your computer. The drive will be called **boardnameBOOT** where boardname is a reference to your specific board. For example, a Feather will have **FEATHERBOOT** and a Trinket will have **TRINKETBOOT** etc.



You will see a new disk drive appear called **boardnameBOOT**. The board is now in bootloader mode.

Find the **.UF2** file you downloaded and drag that file to the new drive on your computer.



The board's LED should flash and the drive will disappear. Your board should re-enumerate USB and appear as a COM or Serial port on your computer. Make a note of the serial port by checking the **Device Manager** (Windows) or **typing `ls /dev/cu*` or `/dev/tty*` (Mac or Linux) in a terminal.**

```
$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port  /dev/cu.usbmodem1432201
```

```
$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port  /dev/cu.usbmodem1432201
```

Download NINA Firmware

Click the link below to download the latest version of the NINA firmware. **Unzip** it **and save the .bin file to your desktop**.

To support BLE on the ESP32 AirLift, you'll need to download NINA firmware version 1.7.1, or later.

Download the latest nina-fw .bin file

https://adafru.it/G3D

Next, you'll need to flash the firmware to your ESP32 AirLift module.

If you're using the Google Chrome browser or Microsoft Edge (version 89 or later), you may follow the instructions below for programming using your board.

For advanced users who have **esptool.py** installed, skip to the bottom of the page.

Upload NINA Firmware

Next, you'll need to upload the new version of NINA firmware to your ESP32 AirLift. To do this, we'll use the web-based implementation of the flasher tool for Espressif chips,

ESPTool. You will need to be running Google Chrome or Microsoft Edge (version 89 or later) to follow the steps below.

Safari and Firefox, etc. are not supported because we need Web Serial and only Chrome is supporting it to the level needed. **If you're using an unsupported browser**, you'll need to either switch to Google Chrome or upload NINA firmware using the Python **esptool.py** program from your computer (Scroll down to Upload NINA Firmware with esptool.py,)

Please ensure you are running Google Chrome or Microsoft Edge (version 89 or later) before following the steps below. Esptool-js is based on Web Serial API and ONLY works for Google Chrome and Microsoft Edge, version 89 or later.

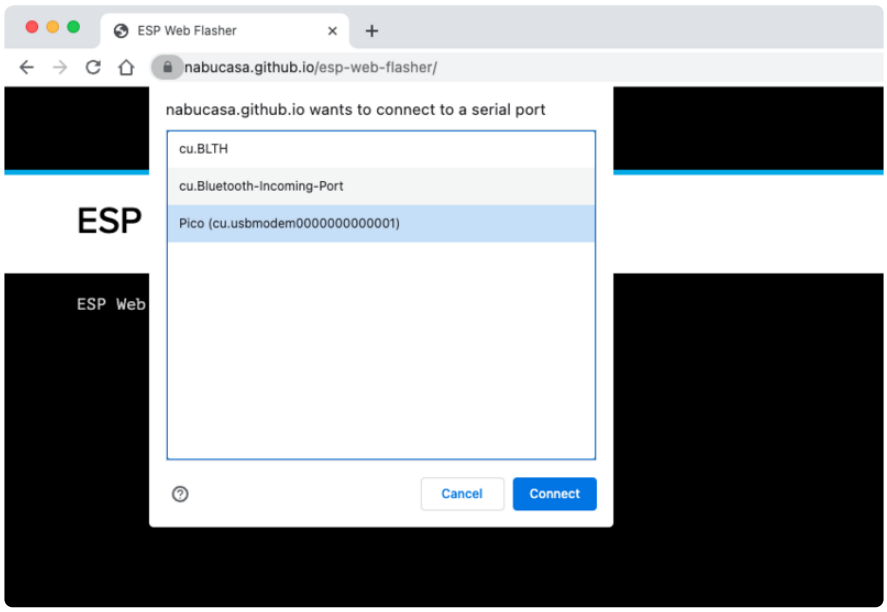
On your Google Chrome browser, navigate to https://adafruit.github.io/Adafruit_WebSerial_ESPTool/ (https://adafru.it/PMB)

In the top-right corner, select 115200 as the baud rate and click the **Connect** button.

You will get a pop-up asking you to select the board's COM or Serial port.

- If there are a lot of boards and ports appearing in this list and you're not sure what to select - remove all other USB devices so only your board is attached, that way there's no confusion over multiple ports!

Click **Connect**.

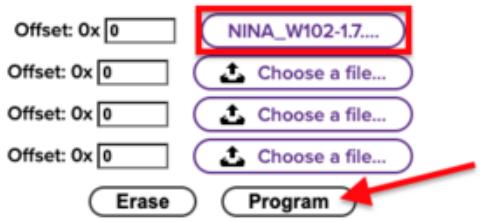



```
ESP Web Flasher loaded.
Connecting...
Connected successfully.
Try hard reset.
Chip type ESP32
Connected to ESP32
MAC Address: C4:4F:33:0E:A1:29
Uploading stub...
Running stub...
Stub is now running...
Detecting Flash Size
FlashId: 0x164020
Flash Manufacturer: 20
Flash Device: 4016
Auto-detected Flash size: 4MB
```

Upon success, you will see that it is connected and will print out a unique MAC address identifying the board.

Once you have successfully connected, a command toolbar will appear at the top of the screen.

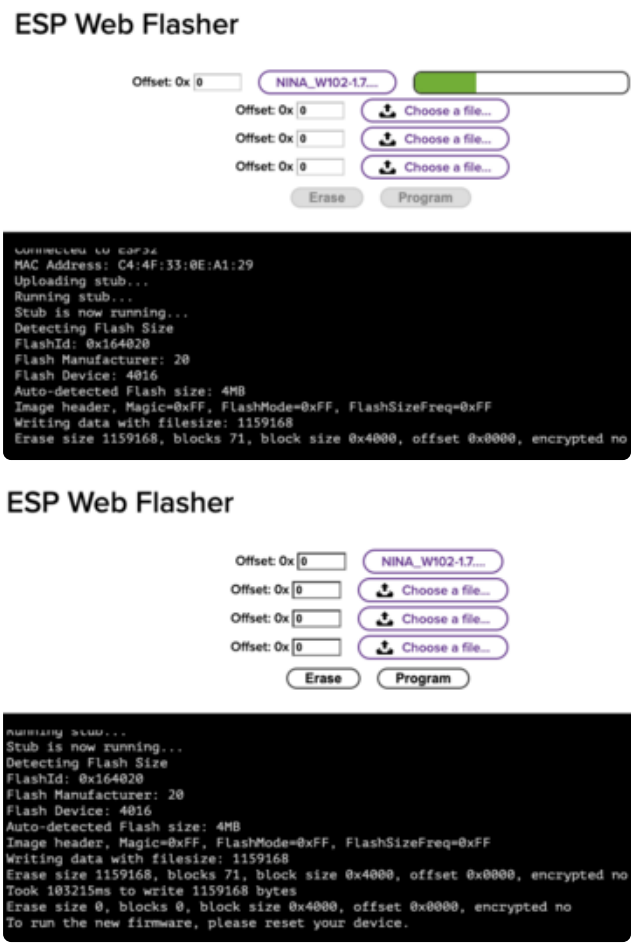
sher




```
d.
y.
```

Verify that the offset is **0x0** and choose the **NINA_....bin** file you downloaded above.

Click the **program** button to flash the firmware to your ESP32 AirLift.



ESPTool will take a few minutes to write firmware to your device. After it's complete, the progress bar will disappear and the console will print "To run the new firmware,..."

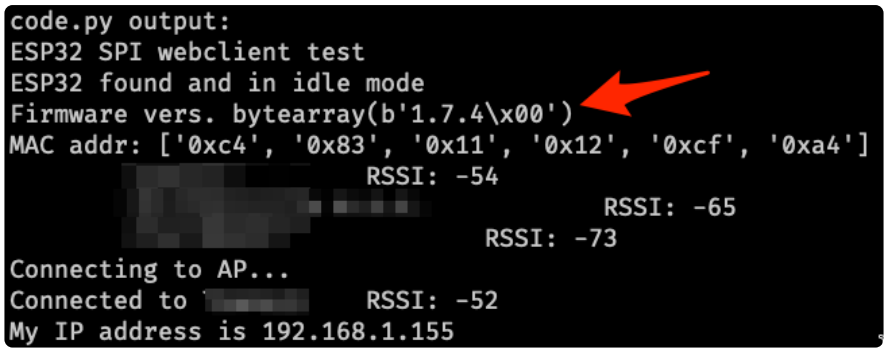
Press the Reset button (or, on the RP2040 Pico, unplug your device from USB power) to get out of the ROM bootloader.

Verify the New Firmware Version

To verify everything is working correctly, we'll load up some CircuitPython code.

If you were previously using your ESP32 project with CircuitPython, you'll need to first reinstall CircuitPython firmware for your board. The QSPI flash should have retained its contents. If you don't see anything on the **CIRCUITPY** volume, copy files from the backup you made earlier to **CIRCUITPY**.

To verify the new ESP32 WiFi firmware version is correct, follow the Connect to WiFi step in this guide (https://adafru.it/Eao) and come back here when you've successfully run the code. The REPL output should display the firmware version you flashed.



(Advanced) Upload NINA Firmware with ESPTool.py

For advanced users who have **esptool.py** installed, run the following commands on your command line:

If you're using macOS or Linux - run the following command, replacing `/dev/ttyACM0` with the serial port of your board and `NINA_W102-1.6.0` with the binary file you're flashing to the ESP32.

```
esptool.py --port /dev/ttyACM0 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.6.0.bin
```

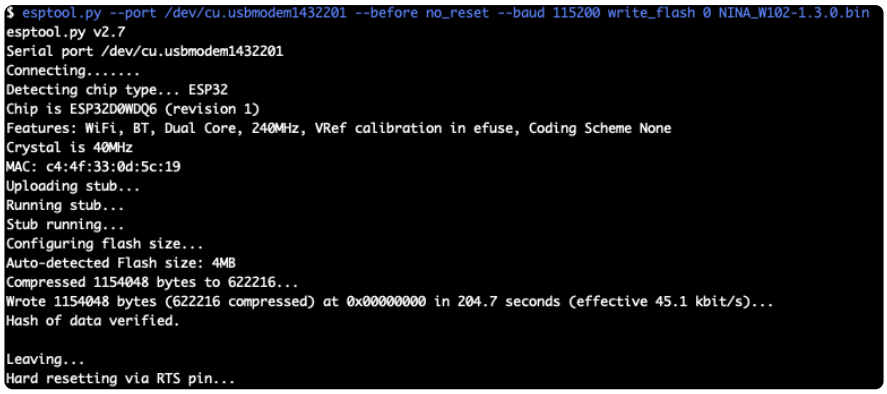
If you're using Windows - run the following command, replacing `COM7` with the serial port of your board and `NINA_W102-1.6.0` with the binary file you're flashing to the ESP32

```
esptool.py --port COM7 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.6.0.bin
```

The command should detect the ESP32 and will take a minute or two to upload the firmware.

- If ESPTool doesn't detect the ESP32, make sure you've uploaded the correct **.UF2** file to the bootloader and are using the correct serial port.

Once the firmware is fully uploaded, press the Reset button (or, on the RP2040 Pico, unplug your device from USB power) to get out of the ROM bootloader mode.

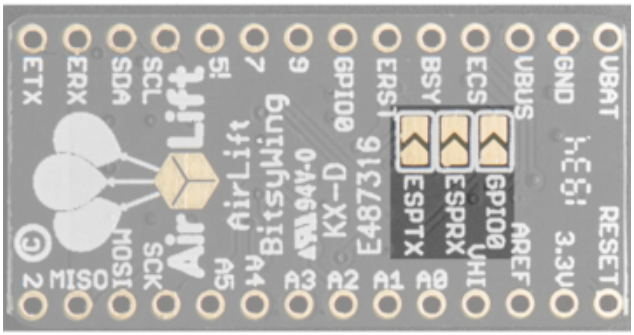


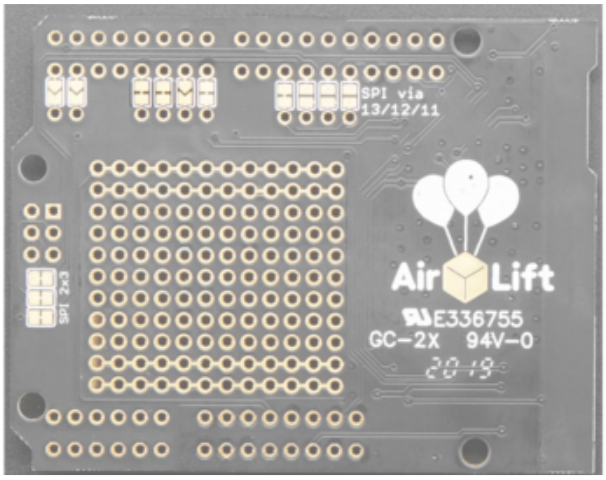
Upgrade External ESP32 AirLift Firmware

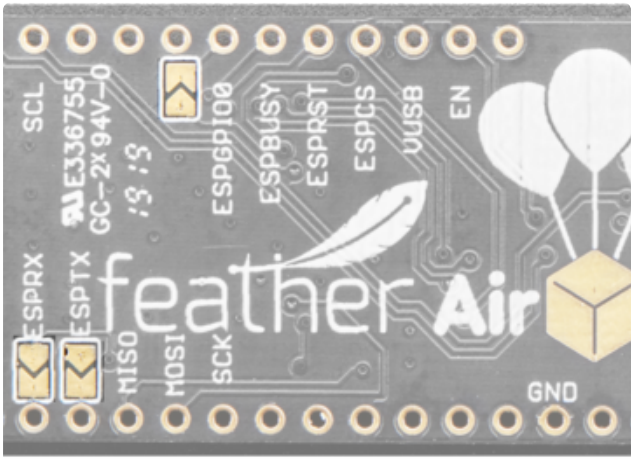
Upgrading AirLift FeatherWing, AirLift Shield or AirLift ItsyWing

External AirLift boards, like the AirLift FeatherWing, AirLift Shield or AirLift ItsyWing, have three optional ESP32 control pins which are not connected by default:

- ESPGPIO0
- ESPRX
- ESPTX







Make sure to solder each of these pads together. You will not be able to upload firmware to your ESP32 if they are not connected.

Upload Passthrough Code

First, you'll need to upload the code below to allow your board to act as a programmer for the ESP32 AirLift module.

Back up any code and files on your CIRCUITPY drive. The code will overwrite the drive's contents. You should not end up losing any files on the QSPI flash, but it's a good idea to back them up anyways.

Download the **UF2** file for your board and save it to your computer's Desktop.

This section is only for an AirLift FeatherWing with a Feather M4, or an AirLift BitsyWing with an ItsyBitsy M4. If you are using a different hardware combination - scroll down to the "Upgrading an External AirLift Breakout" section.

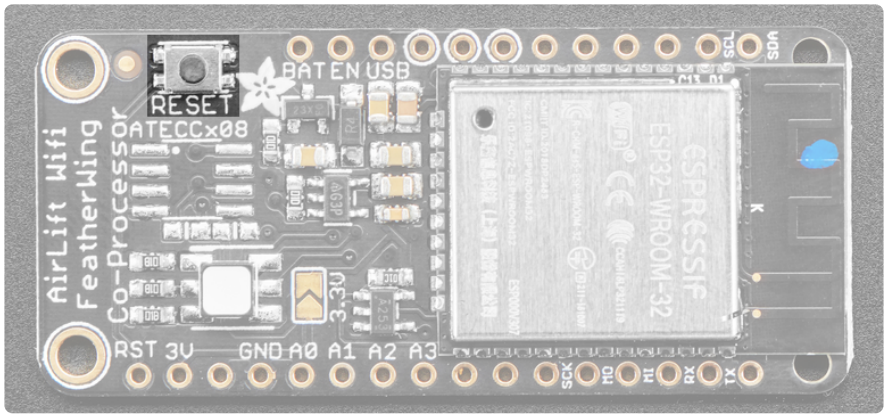
Feather M0/M4 Passthrough UF2

https://adafru.it/OYF

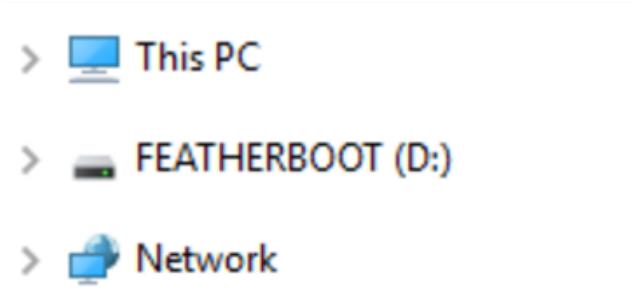
Feather NRF52840 Passthrough UF2

https://adafru.it/PTE

To enter bootloader mode, start with your board unplugged from USB. Next, find the reset button on your board. It's a small, black button, and on most of the boards, it will be the only button available.

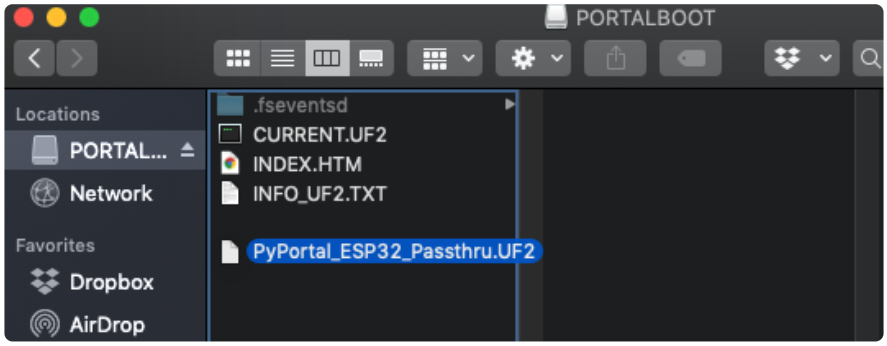


Tap this button twice to enter the bootloader. If it doesn't work on the first try, don't be discouraged. The rhythm of the taps needs to be correct and sometimes it takes a few tries. Once successful, the RGB LED on the board will flash red and then stay green. A new drive will show up on your computer. The drive will be called **boardnameBOOT** where boardname is a reference to your specific board. For example, a Feather will have **FEATHERBOOT** and a Trinket will have **TRINKETBOOT** etc. Going forward, we'll just call the boot drive **BOOT**.



You will see a new disk drive appear called **boardnameBOOT**. The board is now in bootloader mode.

Find the **.UF2** file you downloaded and drag that file to the new drive on your computer.



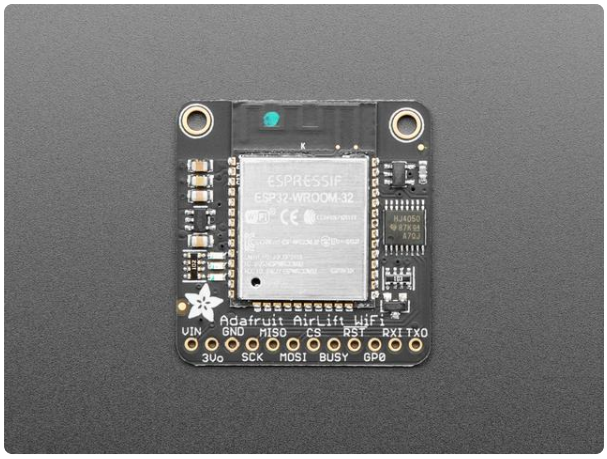
The board's LED should flash and the drive will disappear. Your board should re-enumerate USB and appear as a COM or Serial port on your computer. Make a note of the serial port by checking the **Device Manager** (Windows) or **typing `ls /dev/cu*` or `/dev/tty*` (Mac or Linux) in a terminal.**

```
$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port  /dev/cu.usbmodem1432201
```

```
$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port  /dev/cu.usbmodem1432201
```

Upgrading an External AirLift Breakout

If you have an AirLift ESP32 Breakout Board, you will be turning your existing Arduino-compatible board into a USB-to-Serial-Converter. To do this, you'll need a special Arduino sketch named **SerialESPPassthrough.ino** and an Arduino-compatible board with Native USB support such as the Adafruit Metro M4.



Adafruit AirLift – ESP32 WiFi Co-Processor Breakout Board

Give your plain ol' microcontroller project a lift with the Adafruit AirLift - a breakout board that lets you use the powerful ESP32 as a WiFi co-processor. You probably...

https://www.adafruit.com/product/4201

First, make the following connections between the board and the AirLift Breakout:

- **Board Pin 12** to **ESP32_ResetN**
- **Board Pin 10** to **ESP32 GPIO0**
- **Board TX** to **RXI**
- **Board RX** to **TX0**

Next, download the Arduino sketch below.

```
// SPDX-FileCopyrightText: 2018 Arduino SA
//
// SPDX-License-Identifier: LGPL-2.1-or-later
/*
  SerialNINAPassthrough - Use esptool to flash the ESP32 module
  For use with PyPortal, Metro M4 WiFi...

  Copyright (c) 2018 Arduino SA. All rights reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/

#include <Adafruit_NeoPixel.h>

unsigned long baud = 115200;

#if defined(ADAFRUIT_FEATHER_M4_EXPRESS) || \
  defined(ADAFRUIT_FEATHER_M0_EXPRESS) || \
  defined(ARDUINO_AVR_FEATHER32U4) || \
  defined(ARDUINO_NRF52840_FEATHER) || \
  defined(ADAFRUIT_ITSYBITSY_M0) || \
  defined(ADAFRUIT_ITSYBITSY_M4_EXPRESS) || \
  defined(ARDUINO_AVR_ITSYBITSY32U4_3V) || \
  defined(ARDUINO_NRF52_ITSYBITSY) || \
```

```
  defined(ARDUINO_PYGAMER_M4_EXPRESS)
  // Configure the pins used for the ESP32 connection
  #define SerialESP32   Serial1
  #define SPIWIFI       SPI  // The SPI port
  #define SPIWIFI_SS    13   // Chip select pin
  #define ESP32_RESETN  12   // Reset pin
  #define SPIWIFI_ACK   11   // a.k.a BUSY or READY pin
  #define ESP32_GPIO0   10
  #define NEOPIXEL_PIN   8
#elif defined(ARDUINO_AVR_FEATHER328P)
  #define SerialESP32   Serial1
  #define SPIWIFI       SPI  // The SPI port
  #define SPIWIFI_SS     4   // Chip select pin
  #define ESP32_RESETN   3   // Reset pin
  #define SPIWIFI_ACK    2   // a.k.a BUSY or READY pin
  #define ESP32_GPIO0   -1
  #define NEOPIXEL_PIN   8
#elif defined(TEENSYDUINO)
  #define SerialESP32   Serial1
  #define SPIWIFI       SPI  // The SPI port
  #define SPIWIFI_SS     5   // Chip select pin
  #define ESP32_RESETN   6   // Reset pin
  #define SPIWIFI_ACK    9   // a.k.a BUSY or READY pin
  #define ESP32_GPIO0   -1
  #define NEOPIXEL_PIN   8
#elif defined(ARDUINO_NRF52832_FEATHER )
  #define SerialESP32   Serial1
  #define SPIWIFI       SPI  // The SPI port
  #define SPIWIFI_SS    16  // Chip select pin
  #define ESP32_RESETN  15  // Reset pin
  #define SPIWIFI_ACK    7  // a.k.a BUSY or READY pin
  #define ESP32_GPIO0   -1
  #define NEOPIXEL_PIN   8
#elif !defined(SPIWIFI_SS)  // if the wifi definition isnt in the board variant
  // Don't change the names of these #define's! they match the variant ones
  #define SerialESP32   Serial1
  #define SPIWIFI       SPI
  #define SPIWIFI_SS    10   // Chip select pin
  #define SPIWIFI_ACK    7   // a.k.a BUSY or READY pin
  #define ESP32_RESETN   5   // Reset pin
  #define ESP32_GPIO0   -1   // Not connected
  #define NEOPIXEL_PIN   8
#endif

#if defined(ADAFRUIT_PYPORTAL)
  #define PIN_NEOPIXEL   2
#elif defined(ADAFRUIT_METRO_M4_AIRLIFT_LITE)
  #define PIN_NEOPIXEL   40
#endif

Adafruit_NeoPixel pixel = Adafruit_NeoPixel(1, PIN_NEOPIXEL, NEO_GRB + NEO_KHZ800);

void setup() {
  Serial.begin(baud);
  pixel.begin();
  pixel.setPixelColor(0, 10, 10, 10); pixel.show();

  while (!Serial);
  pixel.setPixelColor(0, 50, 50, 50); pixel.show();

  delay(100);
  SerialESP32.begin(baud);

  pinMode(SPIWIFI_SS, OUTPUT);
  pinMode(ESP32_GPIO0, OUTPUT);
  pinMode(ESP32_RESETN, OUTPUT);

  // manually put the ESP32 in upload mode
  digitalWrite(ESP32_GPIO0, LOW);
```

```
  digitalWrite(ESP32_RESETN, LOW);
  delay(100);
  digitalWrite(ESP32_RESETN, HIGH);
  pixel.setPixelColor(0, 20, 20, 0); pixel.show();
  delay(100);
}

void loop() {
  while (Serial.available()) {
    pixel.setPixelColor(0, 10, 0, 0); pixel.show();
    SerialESP32.write(Serial.read());
  }

  while (SerialESP32.available()) {
    pixel.setPixelColor(0, 0, 0, 10); pixel.show();
    Serial.write(SerialESP32.read());
  }
}
```

Unzip the file, and open the **SerialESPPassthrough.ino** file in the Arduino IDE.

Change the following pin definitions in the sketch to match your wiring:

```
#elif !defined(SPIWIFI_SS)  // if the wifi definition isnt in the board variant
  // Don't change the names of these #define's! they match the variant ones
  #define SerialESP32   Serial1
  #define SPIWIFI       SPI
  #define SPIWIFI_SS    10   // Chip select pin
  #define SPIWIFI_ACK    7   // a.k.a BUSY or READY pin
  #define ESP32_RESETN   5   // Reset pin
  #define ESP32_GPIO0   -1   // Not connected
  #define NEOPIXEL_PIN   8
#endif
```

Using the Arduino IDE, **upload the code to your board** (Sketch->Upload).

After uploading, the board should enumerate USB and appear as a COM or Serial port on your computer.

Make a note of the serial port by checking the **Device Manager** (Windows) or typing in `ls /dev/cu*` or `/dev/tty*` (Mac or Linux) in a terminal

Download NINA Firmware

Click the link below to download the latest version of the NINA firmware. **Unzip** it **and save the .bin file to your desktop**.

To support BLE on the ESP32 AirLift, you'll need to download NINA firmware version 1.7.1, or later.

Download the latest nina-fw .bin file

https://adafru.it/G3D

Next, you'll need to flash the firmware to your ESP32 AirLift module.

If you're using the Google Chrome browser or Microsoft Edge (version 89 or later), you may follow the instructions below for programming using your board.

For advanced users who have **esptool.py** installed, skip to the bottom of the page.

Upload NINA Firmware

Next, you'll need to upload the new version of NINA firmware to your ESP32 AirLift. To do this, we'll use the web-based implementation of the flasher tool for Espressif chips, ESPTool. You will need to be running Google Chrome or Microsoft Edge (version 89 or later) to follow the steps below.

Safari and Firefox, etc. are not supported because we need Web Serial and only Chrome is supporting it to the level needed. **If you're using an unsupported browser**, you'll need to either switch to Google Chrome or upload NINA firmware using the Python **esptool.py** program from your computer (Scroll down to Upload NINA Firmware with esptool.py,)

Please ensure you are running Google Chrome or Microsoft Edge (version 89 or later) before following the steps below. Esptool-js is based on Web Serial API and ONLY works for Google Chrome and Microsoft Edge, version 89 or later.

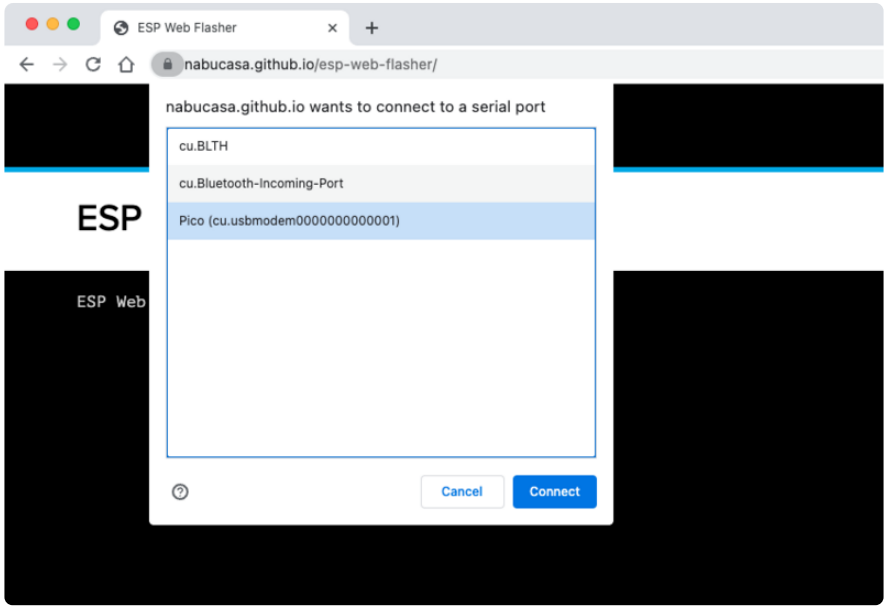
On your Google Chrome browser, navigate to https://adafruit.github.io/Adafruit_WebSerial_ESPTool/ (https://adafru.it/PMB)

In the top-right corner, select 115200 as the baud rate and click the **Connect** button.

You will get a pop-up asking you to select the board's COM or Serial port.

- If there are a lot of boards and ports appearing in this list and you're not sure what to select - remove all other USB devices so only your board is attached, that way there's no confusion over multiple ports!

Click **Connect**.

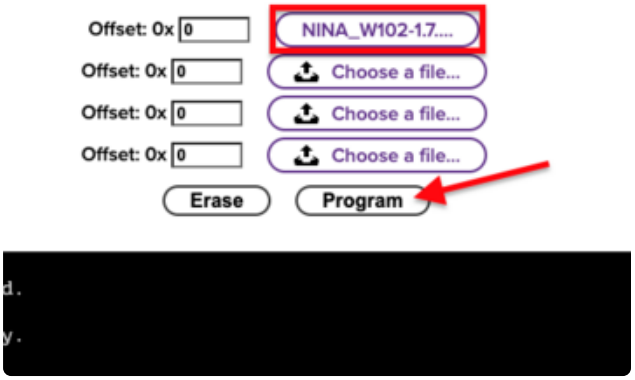



```
ESP Web Flasher loaded.
Connecting...
Connected successfully.
Try hard reset.
Chip type ESP32
Connected to ESP32
MAC Address: C4:4F:33:0E:A1:29
Uploading stub...
Running stub...
Stub is now running...
Detecting Flash Size
FlashId: 0x164020
Flash Manufacturer: 20
Flash Device: 4016
Auto-detected Flash size: 4MB
```

Upon success, you will see that it is connected and will print out a unique MAC address identifying the board.

Once you have successfully connected, a command toolbar will appear at the top of the screen.

sher



Verify that the offset is **0x0** and choose the **NINA_....bin** file you downloaded above.

Click the **program** button to flash the firmware to your ESP32 AirLift.

ESPTool will take a few minutes to write firmware to your device. After it's complete, the progress bar will disappear and the console will print "To run the new firmware,..."

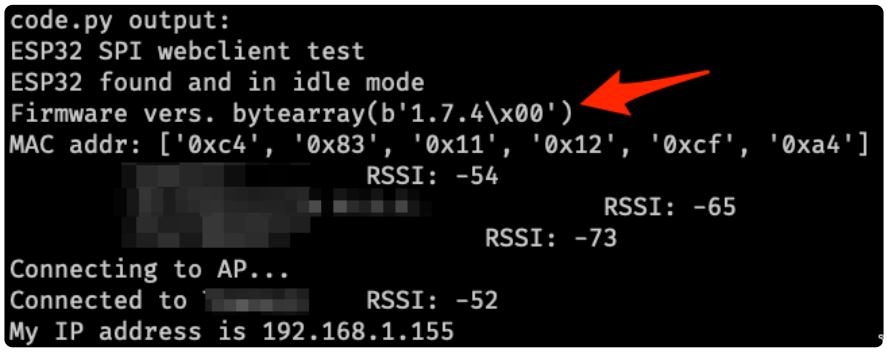
Press the Reset button (or, on the RP2040 Pico, unplug your device from USB power) to get out of the ROM bootloader.

Verify the New Firmware Version

To verify everything is working correctly, we'll load up some CircuitPython code.

If you were previously using your ESP32 project with CircuitPython, you'll need to first reinstall CircuitPython firmware for your board. The QSPI flash should have retained its contents. If you don't see anything on the **CIRCUITPY** volume, copy files from the backup you made earlier to **CIRCUITPY**.

To verify the new ESP32 WiFi firmware version is correct, follow the Connect to WiFi step in this guide (https://adafru.it/Eao) and come back here when you've successfully run the code. The REPL output should display the firmware version you flashed.



(Advanced) Upload NINA Firmware with ESPTool.py

For advanced users who have **esptool.py** installed, run the following commands on your command line:

If you're using macOS or Linux - run the following command, replacing `/dev/ttyACM0` with the serial port of your board and `NINA_W102-1.6.0` with the binary file you're flashing to the ESP32.

```
esptool.py --port /dev/ttyACM0 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.6.0.bin
```

If you're using Windows - run the following command, replacing `COM7` with the serial port of your board and `NINA_W102-1.6.0` with the binary file you're flashing to the ESP32

```
esptool.py --port COM7 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.6.0.bin
```

The command should detect the ESP32 and will take a minute or two to upload the firmware.

- If ESPTool doesn't detect the ESP32, make sure you've uploaded the correct **.UF2** file to the bootloader and are using the correct serial port.

Once the firmware is fully uploaded, press the Reset button (or, on the RP2040 Pico, unplug your device from USB power) to get out of the ROM bootloader mode.

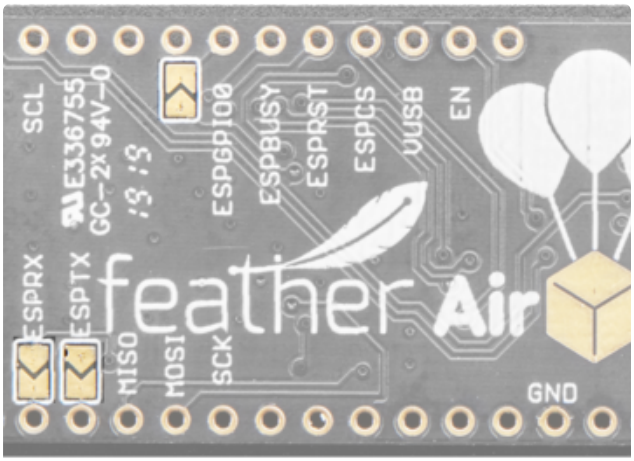
```
$ esptool.py --port /dev/cu.usbmodem1432201 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.3.0.bin
esptool.py v2.7
Serial port /dev/cu.usbmodem1432201
Connecting.....
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: c4:4f:33:0d:5c:19
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1154048 bytes to 622216...
Wrote 1154048 bytes (622216 compressed) at 0x00000000 in 204.7 seconds (effective 45.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Upgrade AirLift Firmware using RP2040

Externally-connected AirLift FfeatherWing and Itsy breakouts have three optional ESP32 control pins which are not connected by default:

- **ESPGPIO0**
- **ESPRX**
- **ESPTX**



Make sure to solder each of these pads together. You will not be able to upload firmware to your ESP32 if they are not connected.

Other AirLift external breakouts have these pins already available.

Upload Passthrough Code

First, you'll need to upload the code below to allow your board to act as a programmer for the ESP32 AirLift module.

Back up any code and files on your CIRCUITPY drive. The code will overwrite the drive's contents. You should not end up losing any files on the QSPI flash, but it's a good idea to back them up anyways.

Download the **UF2** file for your board and save it to your computer's Desktop.

Feather RP2040 Passthrough UF2

https://adafru.it/TE0

For Adafruit Feather RP2040, if you are not stacking a FeatherWing AirLift directly, connect these pins together:

- Feather D13 to Airlift CS
- Feather D12 to AirLift RST
- Feather D10 to AirLift GPIO0
- Feather TX to Airlift RX
- Feather RX to AirLift TX

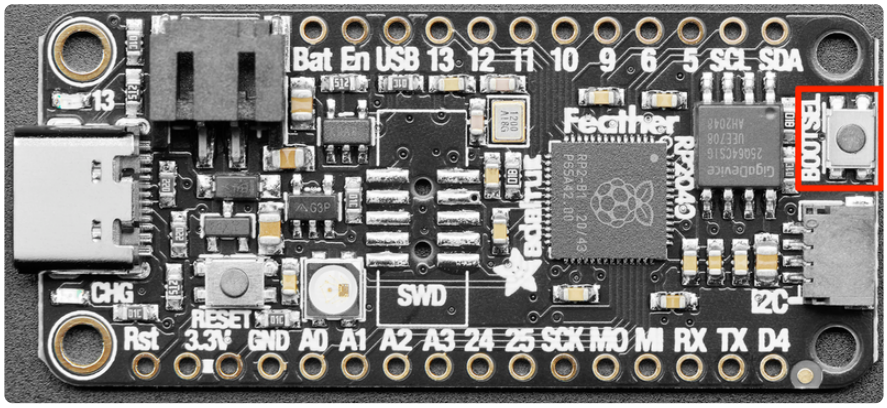
Raspberry Pi Pico RP2040 Passthrough UF2

https://adafru.it/TE1

For Raspberry Pi Pico, connect these pins together:

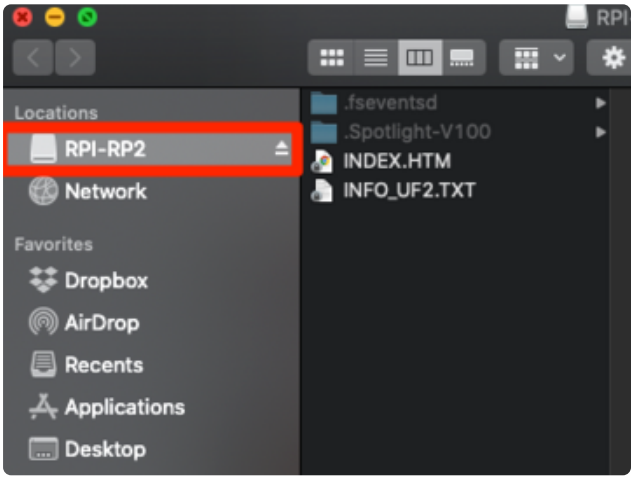
- Pico GPIO13 to Airlift CS
- Pico GPIO16 to AirLift RST
- Pico GPIO9 to AirLift GPIO0
- Pico GPIO0 (TX) to Airlift RX
- Pico GPIO1 (RX) to AirLift TX

To enter bootloader mode, start with your board unplugged from USB. Next, find the reset button on your board. It's a small, black button, and on most of the boards, it will be the only button available.



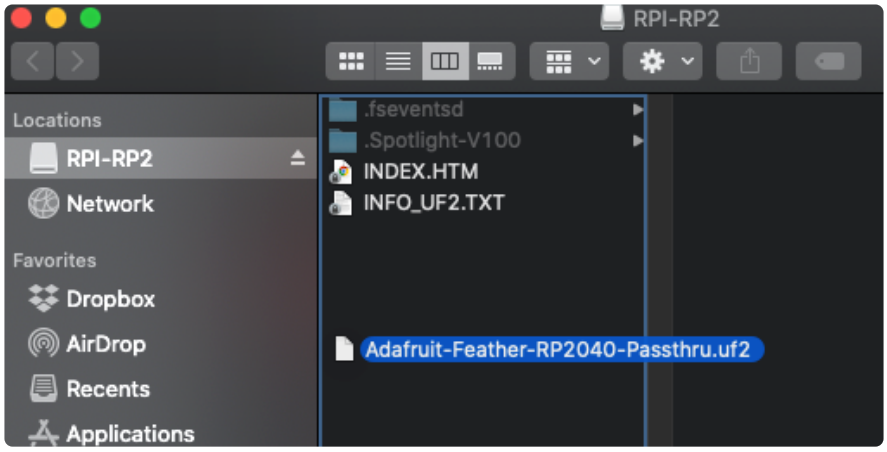
To enter bootloader mode, start with your RP2040 board unplugged from USB.

Press and hold the **BOOTSEL button** (highlighted in red in the image of the Feather RP2040, but all RP2040 boards should include this button), continue to hold it while plugging it into USB, and wait for the **RPI-RP2** drive to appear before releasing the button.



You will see a new disk drive appear called **RPI-RP2**. The board is now in bootloader mode.

Find the **.UF2** file you downloaded and drag that file to the new drive on your computer.



The board's LED should flash and the drive will disappear. Your board should re-enumerate USB and appear as a COM or Serial port on your computer. Make a note of the serial port by checking the **Device Manager** (Windows) or **typing** `ls /dev/cu*` **or** `/dev/tty*` (Mac or Linux) **in a terminal.**

```
$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port  /dev/cu.usbmodem1432201
```

```
$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port  /dev/cu.usbmodem1432201
```

Download NINA Firmware

Click the link below to download the latest version of the NINA firmware. **Unzip** it **and save the .bin file to your desktop**.

To support BLE on the ESP32 AirLift, you'll need to download NINA firmware version 1.7.1, or later.

Download the latest nina-fw .bin file

https://adafru.it/G3D

Next, you'll need to flash the firmware to your ESP32 AirLift module.

If you're using the Google Chrome browser or Microsoft Edge (version 89 or later), you may follow the instructions below for programming using your board.

For advanced users who have **esptool.py** installed, skip to the bottom of the page.

Upload NINA Firmware

Next, you'll need to upload the new version of NINA firmware to your ESP32 AirLift. To do this, we'll use the web-based implementation of the flasher tool for Espressif chips, ESPTool. You will need to be running Google Chrome or Microsoft Edge (version 89 or later) to follow the steps below.

Safari and Firefox, etc. are not supported because we need Web Serial and only Chrome is supporting it to the level needed. **If you're using an unsupported browser**, you'll need to either switch to Google Chrome or upload NINA firmware using the Python **esptool.py** program from your computer (Scroll down to Upload NINA Firmware with esptool.py,)

Please ensure you are running Google Chrome or Microsoft Edge (version 89 or later) before following the steps below. Esptool-js is based on Web Serial API and ONLY works for Google Chrome and Microsoft Edge, version 89 or later.

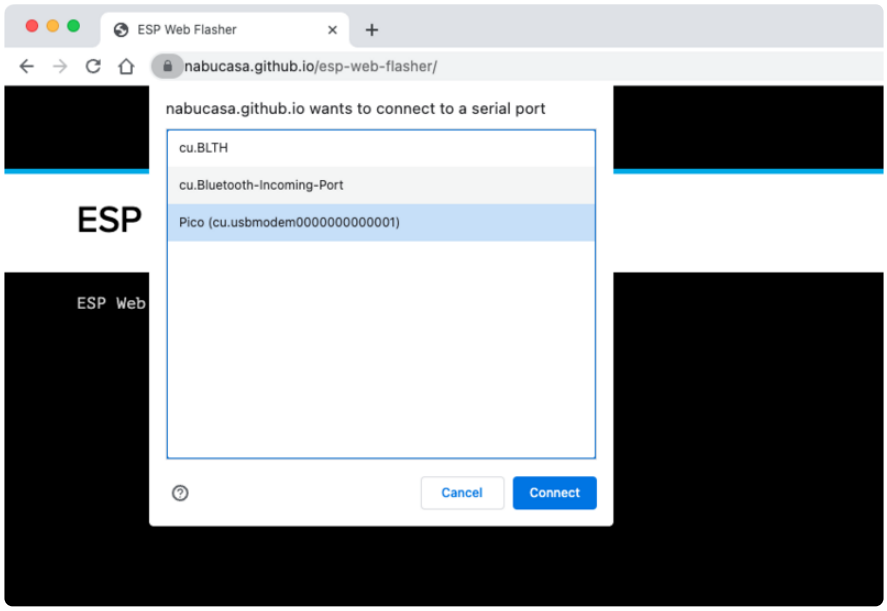
On your Google Chrome browser, navigate to https://adafruit.github.io/Adafruit_WebSerial_ESPTool/ (https://adafru.it/PMB)

In the top-right corner, select 115200 as the baud rate and click the **Connect** button.

You will get a pop-up asking you to select the board's COM or Serial port.

- If there are a lot of boards and ports appearing in this list and you're not sure what to select - remove all other USB devices so only your board is attached, that way there's no confusion over multiple ports!

Click **Connect**.

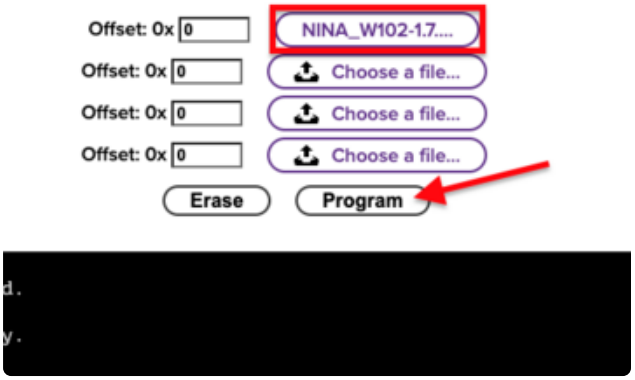



```
ESP Web Flasher loaded.
Connecting...
Connected successfully.
Try hard reset.
Chip type ESP32
Connected to ESP32
MAC Address: C4:4F:33:0E:A1:29
Uploading stub...
Running stub...
Stub is now running...
Detecting Flash Size
FlashId: 0x164020
Flash Manufacturer: 20
Flash Device: 4016
Auto-detected Flash size: 4MB
```

Upon success, you will see that it is connected and will print out a unique MAC address identifying the board.

Once you have successfully connected, a command toolbar will appear at the top of the screen.

sher



Verify that the offset is **0x0** and choose the **NINA_....bin** file you downloaded above.

Click the **program** button to flash the firmware to your ESP32 AirLift.

ESPTool will take a few minutes to write firmware to your device. After it's complete, the progress bar will disappear and the console will print "To run the new firmware,..."

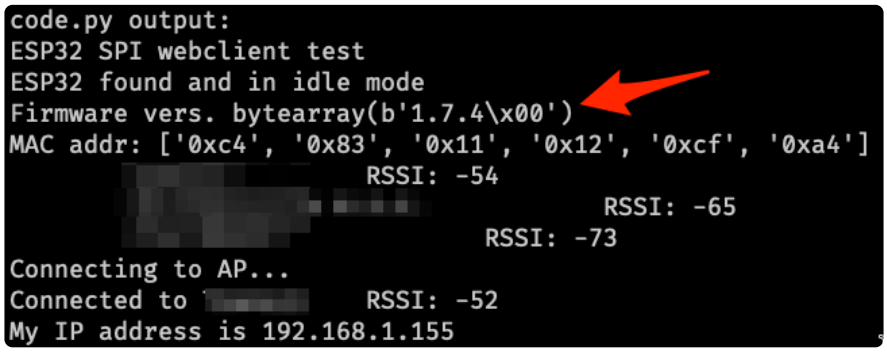
Press the Reset button (or, on the RP2040 Pico, unplug your device from USB power) to get out of the ROM bootloader.

Verify the New Firmware Version

To verify everything is working correctly, we'll load up some CircuitPython code.

If you were previously using your ESP32 project with CircuitPython, you'll need to first reinstall CircuitPython firmware for your board. The QSPI flash should have retained its contents. If you don't see anything on the **CIRCUITPY** volume, copy files from the backup you made earlier to **CIRCUITPY**.

To verify the new ESP32 WiFi firmware version is correct, follow the Connect to WiFi step in this guide (https://adafru.it/Eao) and come back here when you've successfully run the code. The REPL output should display the firmware version you flashed.



(Advanced) Upload NINA Firmware with ESPTool.py

For advanced users who have **esptool.py** installed, run the following commands on your command line:

If you're using macOS or Linux - run the following command, replacing `/dev/ttyACM0` with the serial port of your board and `NINA_W102-1.6.0` with the binary file you're flashing to the ESP32.

```
esptool.py --port /dev/ttyACM0 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.6.0.bin
```

If you're using Windows - run the following command, replacing `COM7` with the serial port of your board and `NINA_W102-1.6.0` with the binary file you're flashing to the ESP32

```
esptool.py --port COM7 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.6.0.bin
```

The command should detect the ESP32 and will take a minute or two to upload the firmware.

- If ESPTool doesn't detect the ESP32, make sure you've uploaded the correct **.UF2** file to the bootloader and are using the correct serial port.

Once the firmware is fully uploaded, press the Reset button (or, on the RP2040 Pico, unplug your device from USB power) to get out of the ROM bootloader mode.

```
$ esptool.py --port /dev/cu.usbmodem1432201 --before no_reset --baud 115200 write_flash 0 NINA_W102-1.3.0.bin
esptool.py v2.7
Serial port /dev/cu.usbmodem1432201
Connecting.......
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: c4:4f:33:0d:5c:19
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1154048 bytes to 622216...
Wrote 1154048 bytes (622216 compressed) at 0x00000000 in 204.7 seconds (effective 45.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```