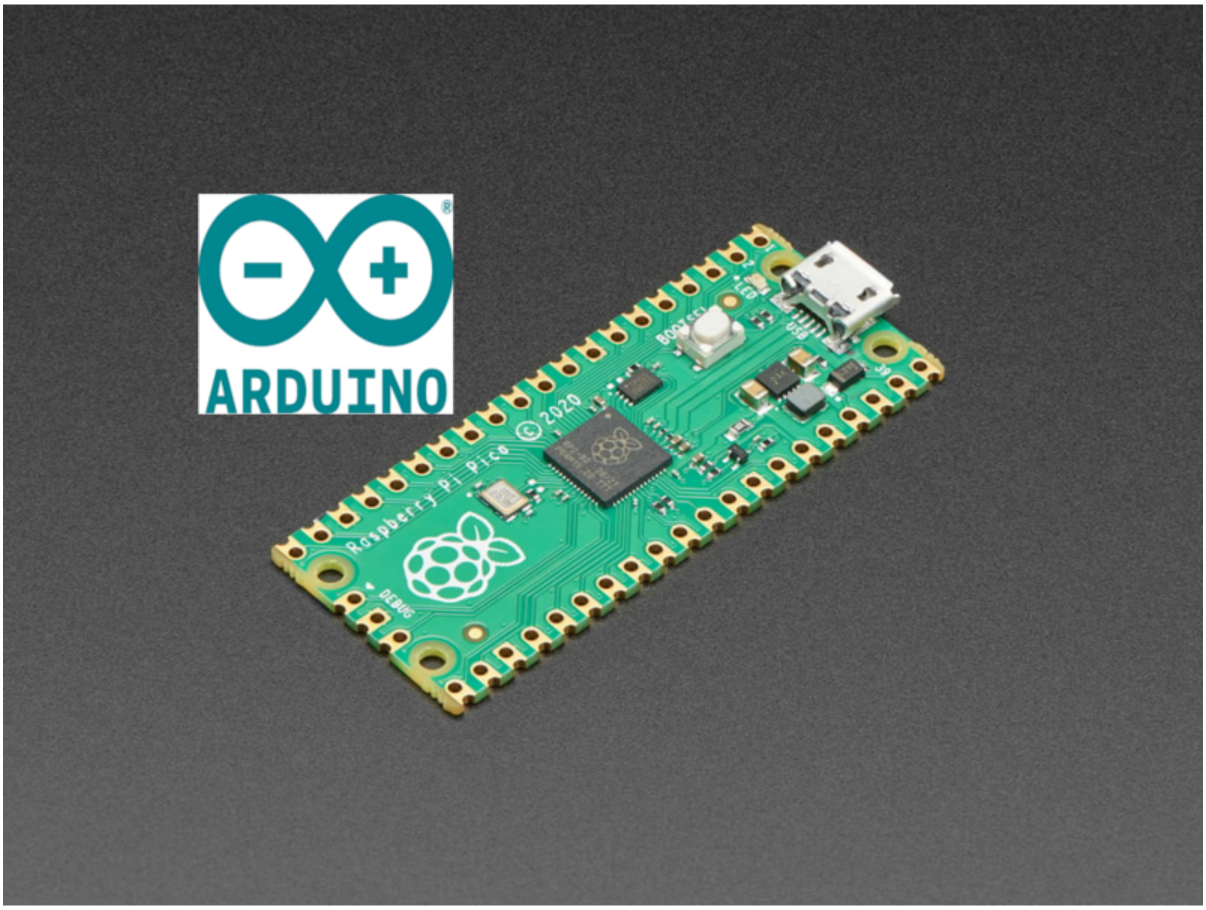
Using Raspberry Pi Pico and Pico 2 with Arduino

Created by Anne Barela

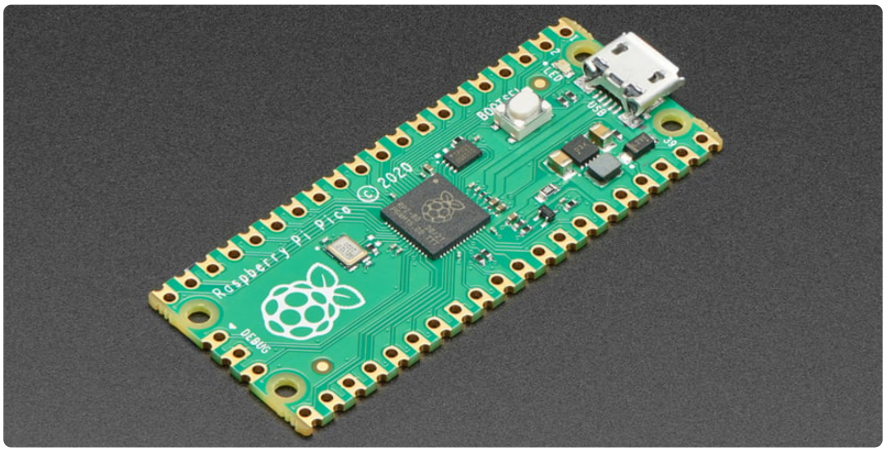


https://learn.adafruit.com/using-raspberry-pi-pico-pico-2-with-arduino

Last updated on 2025-10-14 03:25:14 PM EDT

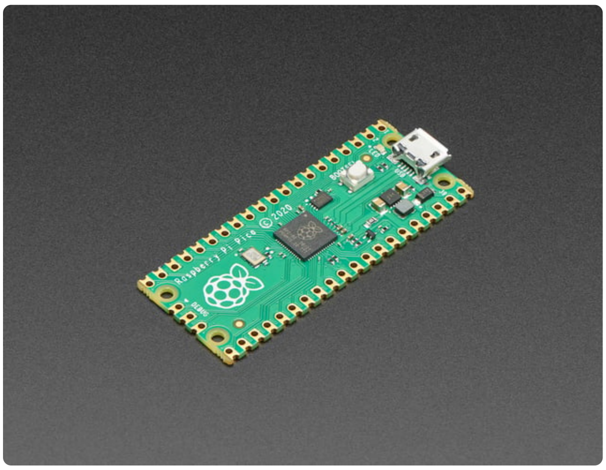
Table of Contents

Overview



The Raspberry Pi Pico and Pico 2 have revolutionized microcontroller design since their introduction. Inexpensive and powerful, they are a natural use for a variety of projects. They can be programmed in different languages to suit the designer.

If you're looking to program your Pico in CircuitPython. check out this guide by Adafruit:



Getting Started with Raspberry Pi Pico and CircuitPython
By Kattni Rembor
Overview

https://learn.adafruit.com/getting-started-with-raspberry-pi-pico-circuitpython/overview

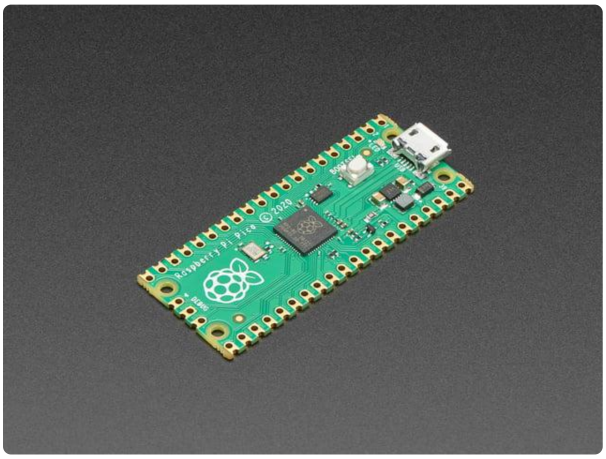
If you wish to use Arduino, that can be done also. But it is suggested you follow this guide. Why?

Back in 2021, when the first Raspberry Pi Pico boards arrived, Arduino provided support (https://adafru.it/1atn) through a version of their software coupled with mbed (https://adafru.it/1ato), a real-time operating system. While most other Arduino boards run the Arduino compiled code "bare metal" (without need of an operating system), the Arduino development environment (IDE) used mbed, adding a layer of complexity to programming.

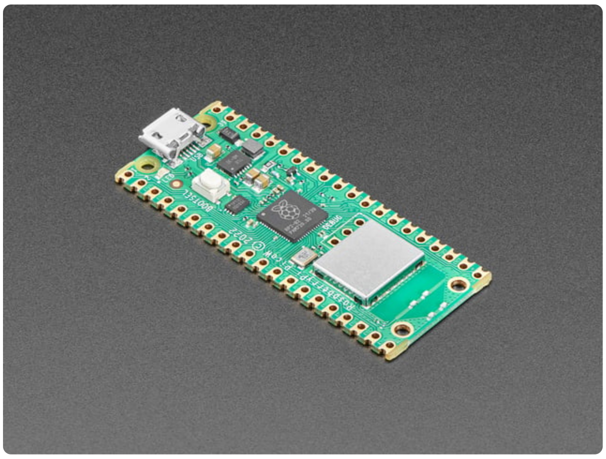
Earle F. Philhower, III designed software which places Arduino compiled code onto Raspberry Pi Pico boards without mbed. This "core" provides a programming paradigm like previous boards designers may have used.

This guide provides information on using the [Philhower core](https://adafru.it/ToC) (https://adafru.it/ToC) to make your Arduino programs.

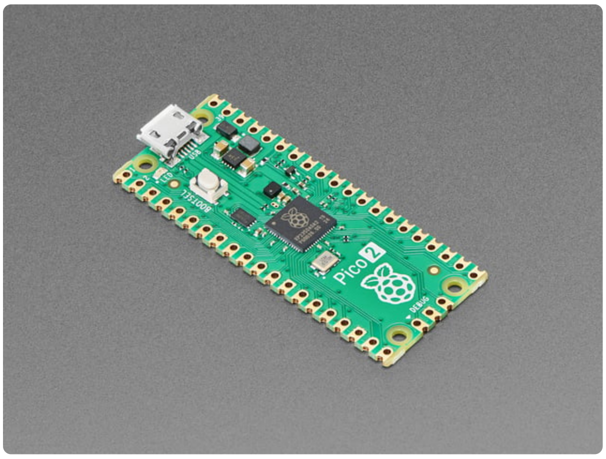
Parts



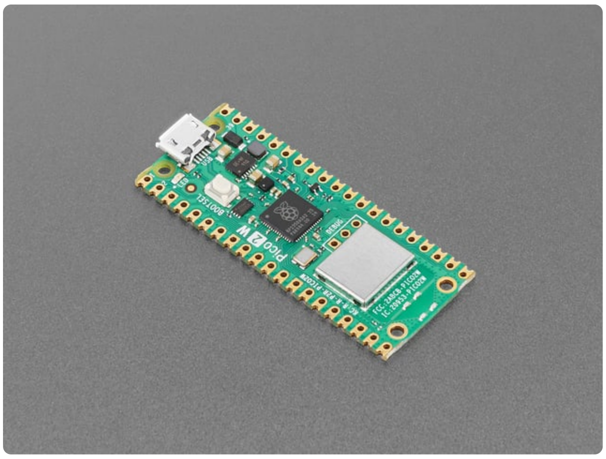
Raspberry Pi Pico RP2040
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...
https://www.adafruit.com/product/4864



Raspberry Pi Pico W
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...
https://www.adafruit.com/product/5526



Raspberry Pi Pico 2 - RP2350
Raspberry Pi Pico 2 is Raspberry Pi Foundation's update to their popular RP2040-based Pico board, now built on RP2350: their new...
https://www.adafruit.com/product/6006

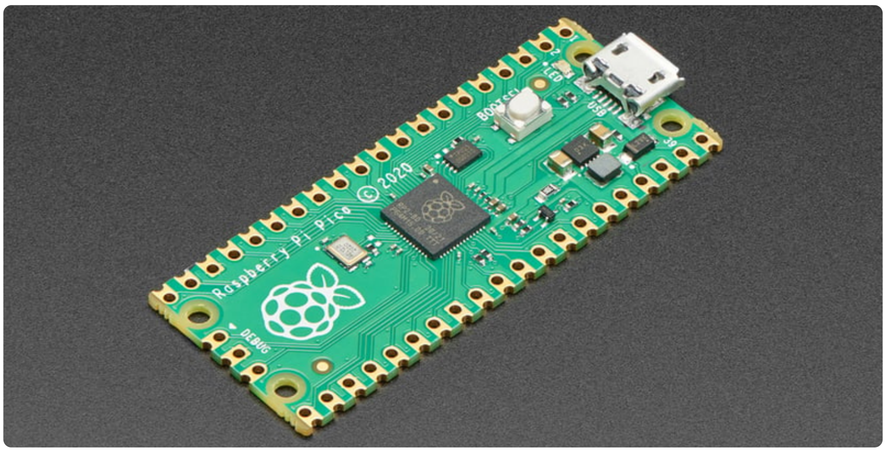


[Raspberry Pi Pico 2W](https://www.adafruit.com/product/6087)
Raspberry Pi Pico 2W is Raspberry Pi Foundation's update to their popular RP2040-based wireless ico board, now built on RP2350: their new...
https://www.adafruit.com/product/6087

Note the Pico and Pico 2 have a USB micro B connector, not USB-C like many other boards. If you need to, be sure to get a good USB data+power cable (not the tiny ones that come with power banks, they're power only).

[USB cable - USB A to Micro-B](https://www.adafruit.com/product/592)
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

Raspberry Pi Pico with RP2040



The original Raspberry Pi Pico uses an RP2040 dual Cortex M0+ microcontroller. When introduced, it provided significantly more computing power and RAM than other boards in the price range.

The Raspberry Pi Pico W adds a WiFi and Bluetooth capability.

The Philhower core for Arduino originally focused on RP2040-based boards. When the RP2350 arrived with the Raspberry Pi Pico 2 (and 2 W), the Arduino board support package was augmented with support for that chip also.

The following pages describe how to set things up for a Raspberry Pi Pico with RP2040. The process also applies, with appropriate changes, for other boards based on the RP2040 microcontroller.

RP2040 Arduino IDE Setup

The **Arduino Philhower core** (https://adafru.it/ToC) provides support for RP2040 microcontroller boards. This page covers getting your Arduino IDE set up to include your board.

Arduino IDE Download

The first thing you will need to do is to download the latest release of the Arduino IDE. The Philhower core requires **version 1.8** or higher.

Arduino IDE Download

https://adafru.it/Pd5

Download and install it to your computer.

Once installed, open the Arduino IDE.

Adding the Philhower Board Manager URL

In the Arduino IDE, navigate to the **Preferences** window. You can access it through **File** > **Preferences** on Windows or Linux, or **Arduino** > **Preferences** on OS X.

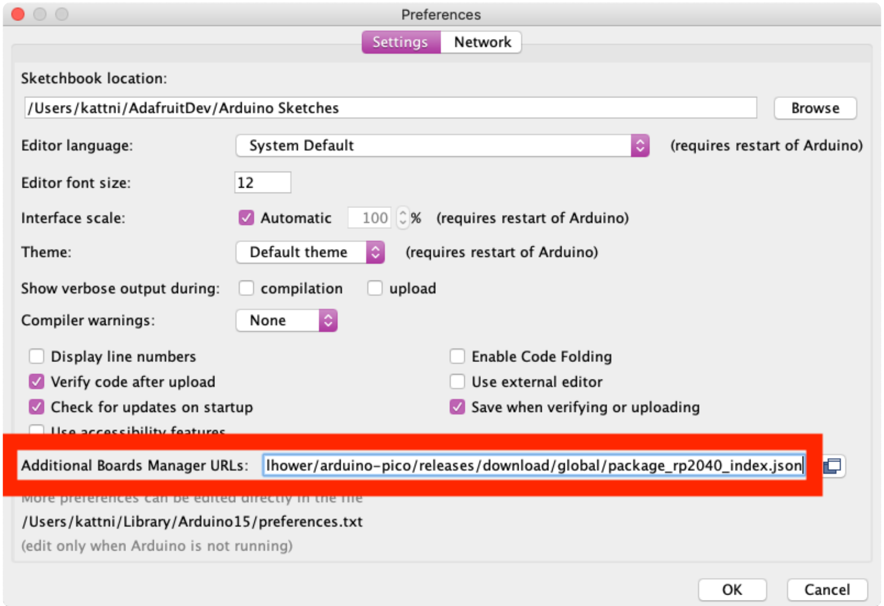
The **Preferences** window will open.

In the **Additional Boards Manager URLs** field, you'll want to add a new URL. The list of URLs is comma separated, and you will only have to add each URL once. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

Copy the following URL.

`https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json`

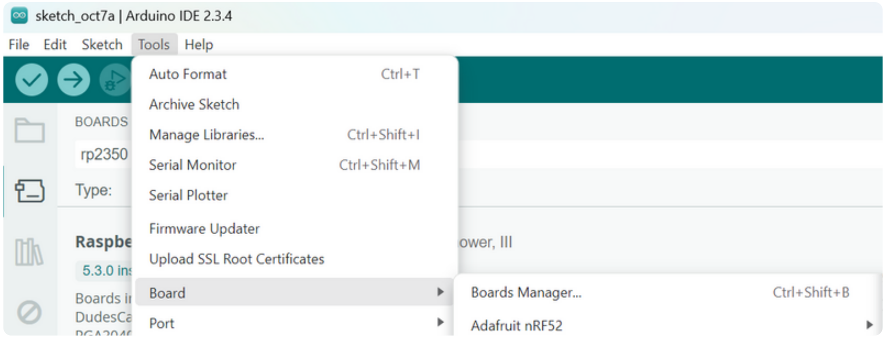
Add the URL to the the **Additional Boards Manager URLs** field (highlighted in red below).



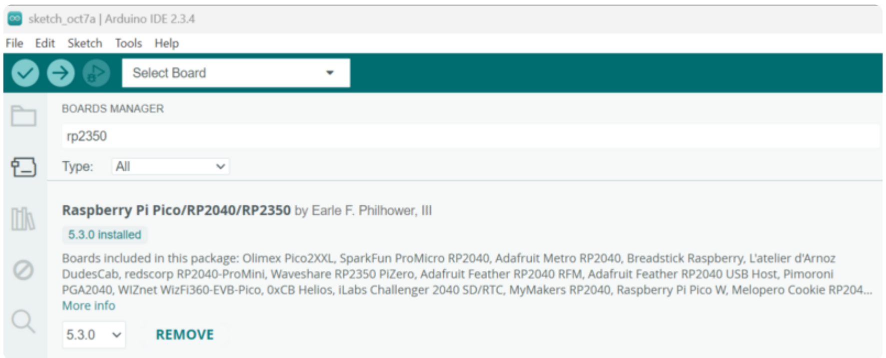
Click **OK** to save and close **Preferences**.

Add Board Support Package

In the Arduino IDE, click on **Tools** > **Board** > **Boards Manager**. If you have previously selected a board, the **Board** menu item may have a board name after it.



In the **Boards Manager**, search for RP2040. Scroll down to the **Raspberry Pi Pico/RP2040/RP2350 by Earle F Philhower, III** entry. Click **Install** to install it. If it was previously installed, look to make sure you have the latest version.

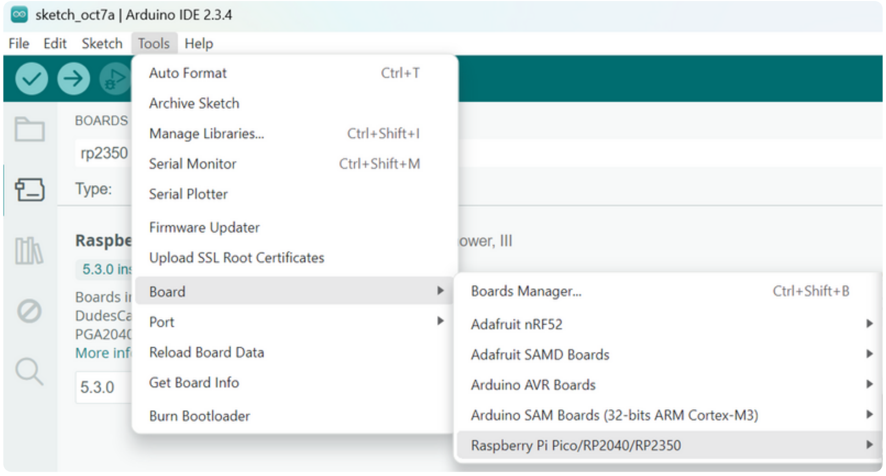


Installing a new board package can take a few minutes. Don't click Cancel!

Once installation is complete, click **Close** to close the Boards Manager.

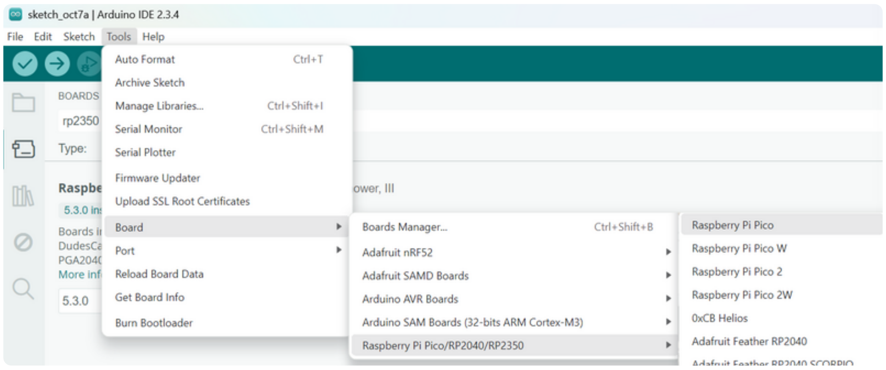
Choose Your Board

In the **Tools** > **Boards** menu, you should now see **Raspberry Pi RP2040 Boards** (possibly followed by a version number).



Navigate to the **Raspberry Pi Pico RP2040/RP2350** menu. You will see the available boards listed.

Navigate to the **Raspberry Pi RP2040/RP2350** menu and choose **Raspberry Pi Pico**.



If you are programming a Raspberry Pi Pico or Pico W, ensure you select the board you have and not a another board as the W boards have LED pinning different (though it should be transparent to the user if it is coded as shown).

Re. If you have a Pico board, do not select Pico W, Pico 2, or Pico 2 W. If you have a Pico W, select Raspberry Pi Pico W and don't select Pico or the Pico 2 boards.

If there is no serial Port available in the dropdown, or an invalid one appears - don't worry about it! The RP2040 does not actually use a serial port to upload, so its OK if it does not appear if in manual bootload mode. You will see a serial port appear after uploading your first sketch.

Now you're ready to begin using Arduino with your RP2040 board!

RP2040 Arduino Blink

The first and most basic program you can upload to your Arduino is the classic Blink sketch. This takes something on the board and makes it, well, blink! On and off. It's a great way to make sure everything is working and you're uploading your sketch to the right board and right configuration.

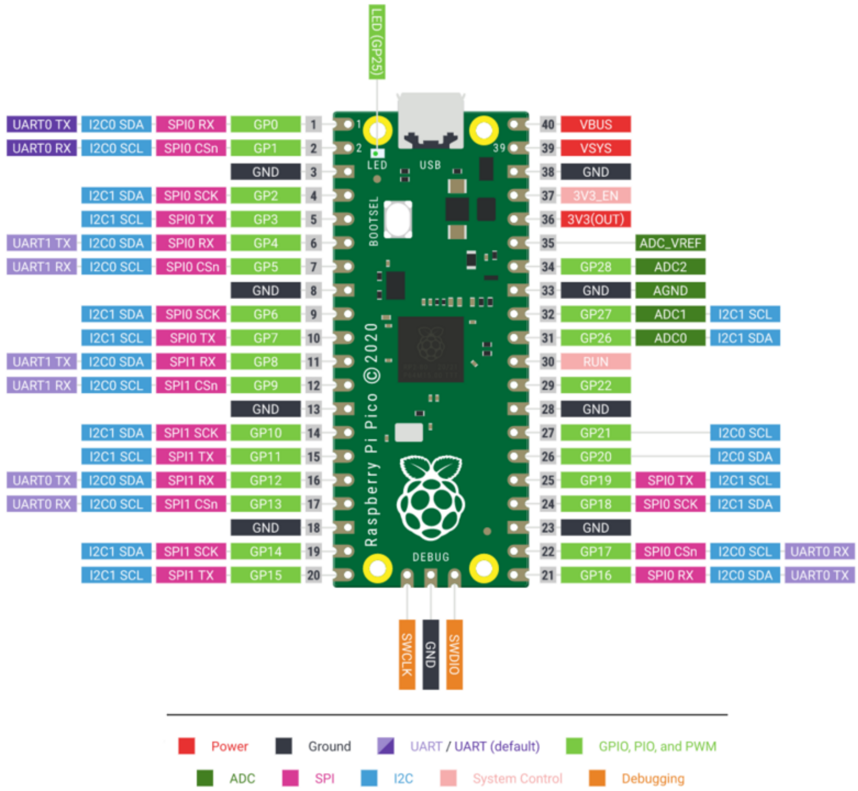
When all else fails, you can always come back to Blink!

Pre-Flight Check: Get Arduino IDE & Hardware Set Up

This lesson assumes you have Arduino IDE set up. This is a generalized checklist, some elements may not apply to your hardware. If you haven't yet, check the previous steps in the guide to make sure you:

- **Install the very latest Arduino IDE for Desktop** (not all boards are supported by the Web IDE so we don't recommend it).
- **Install any board support packages (BSP) required for your hardware.** Some boards are built in defaults on the IDE, but lots are not! You may need to install plug-in support which is called the BSP.

- **Get a Data/Sync USB cable for connecting your hardware.** A significant amount of problems folks have stem from not having a USB cable with data pins. Yes, these cursed cables roam the land, making your life hard. If you find a USB cable that doesn't work for data/sync, throw it away immediately! There is no need to keep it around, cables are very inexpensive these days.
- **Install any drivers required** - If you have a board with a FTDI or CP210x chip, you may need to get separate drivers. If your board has native USB, it probably doesn't need anything. After installing, reboot to make sure the driver sinks in.
- **Connect the board to your computer.** If your board has a power LED, make sure its lit. Is there a power switch? Make sure its turned On!



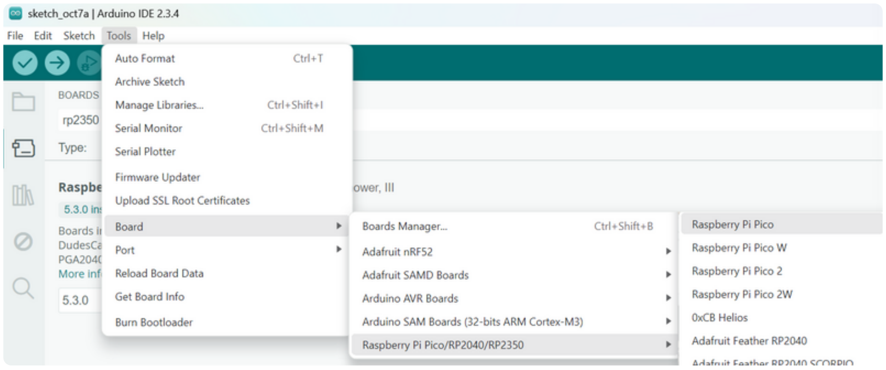
The Pico has the GPIO pin names listed on the bottom of the board as **GPx**, where **x** is the pin number. The Arduino pin name is the number following **GP**. So, for example, pin **GP0** would be Arduino pin `0`.

There is an onboard green LED for user use. Unlike many boards, the Pico lacks a power on LED.

Start up Arduino IDE and Select Board/Port

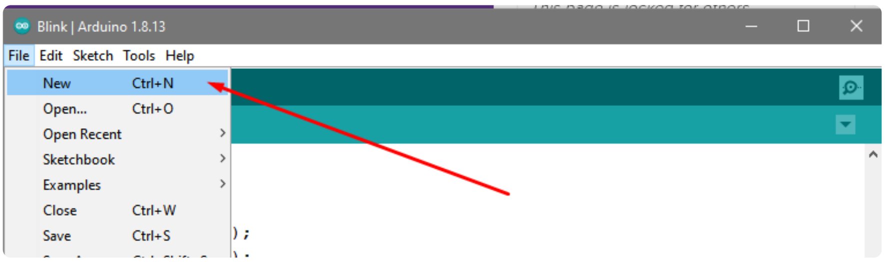
OK now you are prepared! Open the Arduino IDE on your computer. Now you have to tell the IDE what board you are using, and how you want to connect to it.

In the IDE find the **Tools** menu. You will use this to select the board. If you switch boards, you must switch the selection! So always double-check before you upload code in a new session.



New Blink Sketch

OK lets make a new blink sketch! From the **File** menu, select **New**



Then in the new window, copy and paste this text:

```
int led = LED_BUILTIN;

void setup() {
  // Some boards work best if we also make a serial connection
  Serial.begin(115200);

  // set LED to be an output pin
  pinMode(led, OUTPUT);
}

void loop() {
  // Say hi!
  Serial.println("Hello!");
```

```
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                // wait for a half second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                // wait for a half second
}
```

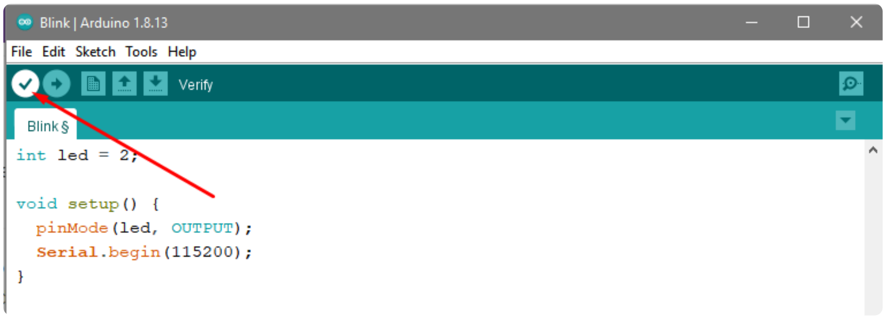
Note that in this example, we are not only blinking the LED but also printing to the Serial monitor, think of it as a little bonus to test the serial connection.

One note you'll see is that we reference the LED with the constant `LED_BUILTIN` rather than a number. That's because, historically, the built in LED was on pin 13 for Arduinos. But in the decades since, boards don't always have a pin 13, or maybe it could not be used for an LED. So the LED could have moved to another pin. It's best to use `LED_BUILTIN` so you don't get the pin number confused!

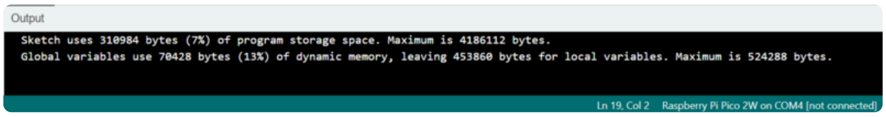
Verify (Compile) Sketch

OK now you can click the Verify button to convert the sketch into binary data to be uploaded to the board.

Note that Verifying a sketch is the same as Compiling a sketch - so we will use the words interchangeably

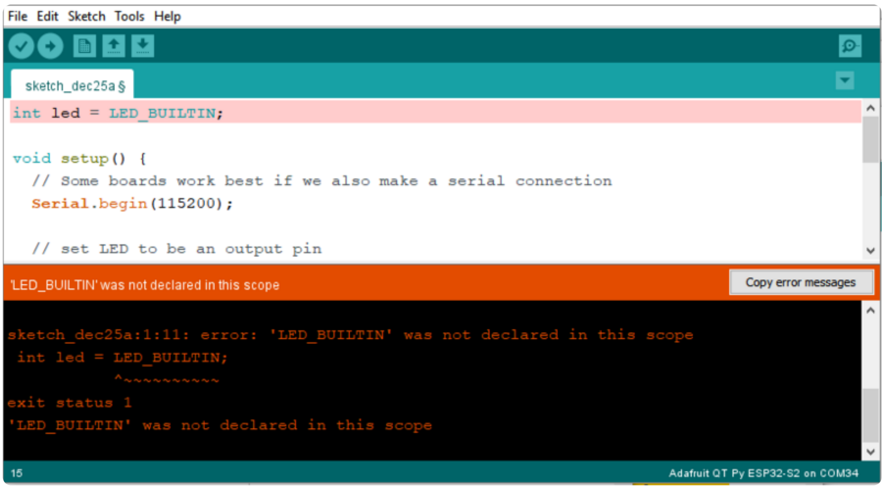


During verification/compilation, the computer will do a bunch of work to collect all the libraries and code and the results will appear in the bottom window of the IDE.

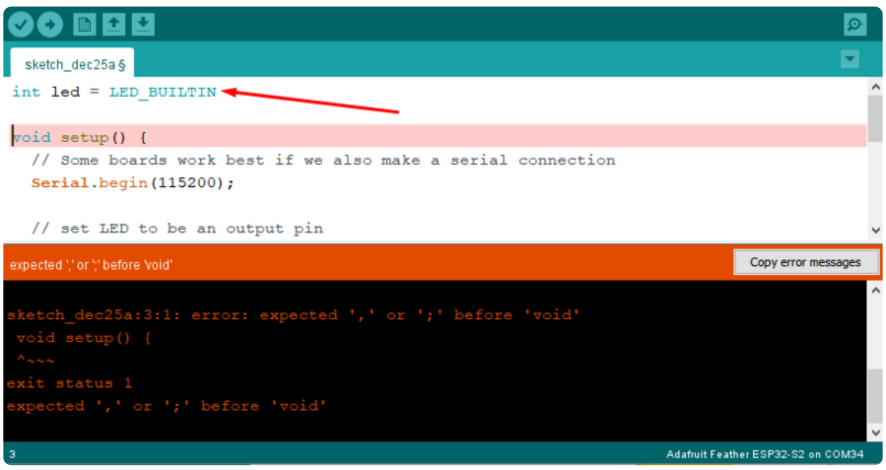


If something went wrong with compilation, you will get red warning/error text in the bottom window letting you know what the error was. It will also highlight the line with an error.

For example, here I had the wrong board selected - and the selected board does not have a built in LED!

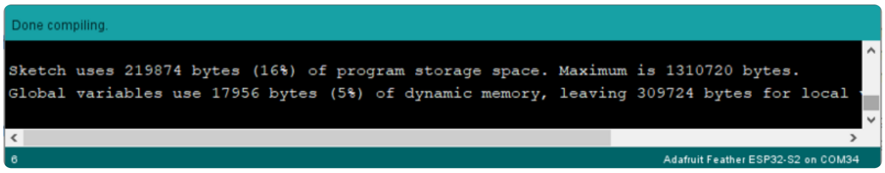


Here's another common error, in my haste I forgot to add a `;` at the end of a line. The compiler warns me that it's looking for one - note that the error is actually a few lines up!



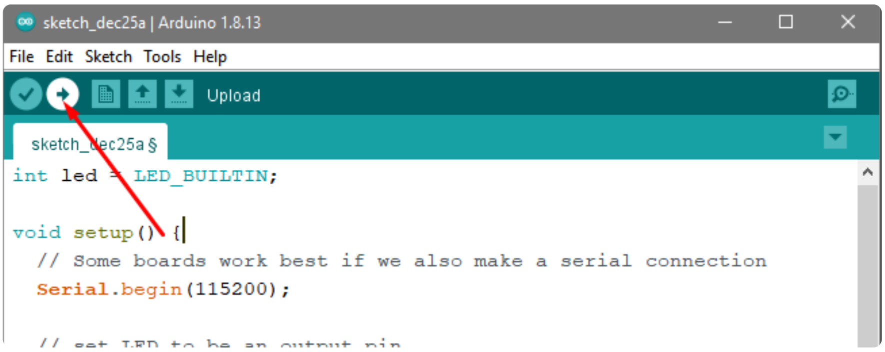
Turning on detailed compilation warnings and output can be very helpful sometimes - Its in Preferences under "Show Verbose Output During:" and check the Compilation button. If you ever need to get help from others, be sure to do this and then provide all the text that is output. It can assist in nailing down what happened!

On success you will see something like this white text output and the message **Done compiling.** in the message area.



Upload Sketch

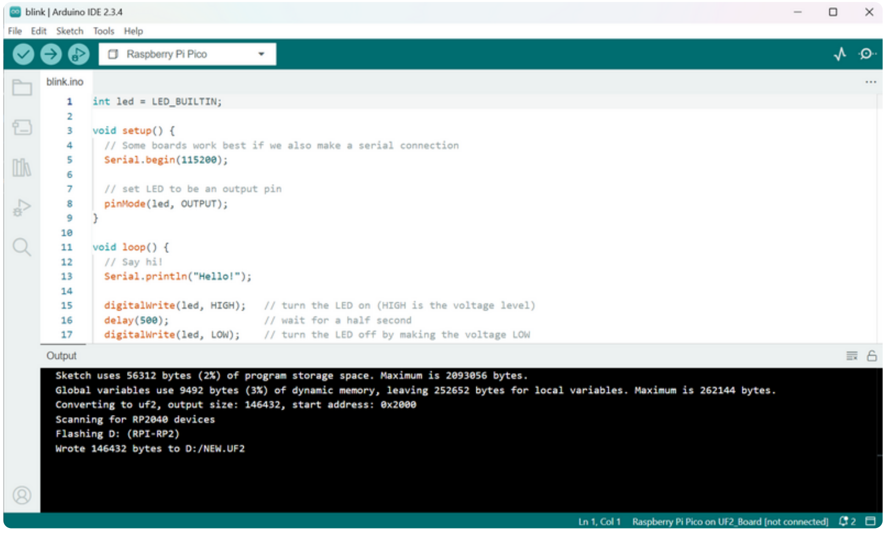
Once the code is verified/compiling cleanly you can upload it to your board. Click the **Upload** button.



The IDE will try to compile the sketch again for good measure, then it will try to connect to the board and upload a the file.

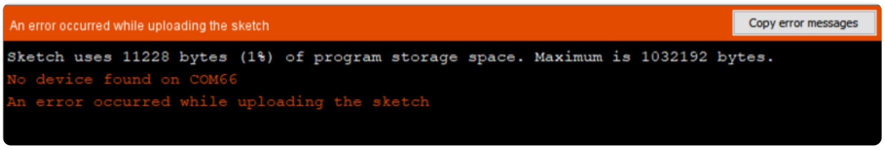
This is actually one of the hardest parts for beginners because it's where a lot of things can go wrong.

However, lets start with what it looks like on success! Here's what your board upload process looks like when it goes right:



Often times you will get a warning like this, which is kind of vague:

`No device found on COM66` (or whatever port is selected)
`An error occurred while uploading the sketch`



This could be a few things.

First up, check again that you have the correct board selected! Many electronics boards have very similar names or look, and often times folks grab a board different from what they thought.

If you're positive the right board is selected, we recommend the next step is to put the board into manual bootloading mode.

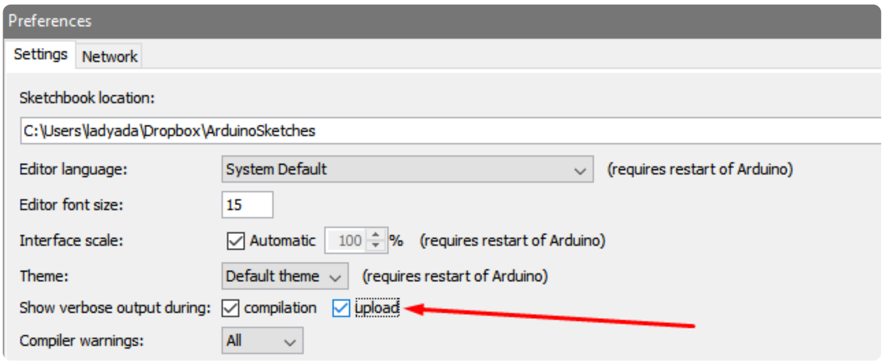
Native USB and manual bootloading

Historically, microcontroller boards contained two chips: the main micro chip (say, ATmega328 or ESP8266 or ESP32) and a separate chip for USB interface that would be used for bootloading (a CH430, FT232, CP210x, etc). With these older designs, the microcontroller is put into a bootloading state for uploading code by the separate chip. It allows for easier uploading but is more expensive as two chips are needed, and also the microcontroller can't act like a keyboard or disk drive.

Modern chips often have 'native' USB - that means that there is no separate chip for USB interface. It's all in one! Great for cost savings, simplicity of design, reduced size and more control. However, it means the chip must be self-aware enough to be able to put itself into bootload/upload mode on its own. That's fine 99% of the time but is very likely you will at some point get the board into an odd state that makes it too confused to bootload.

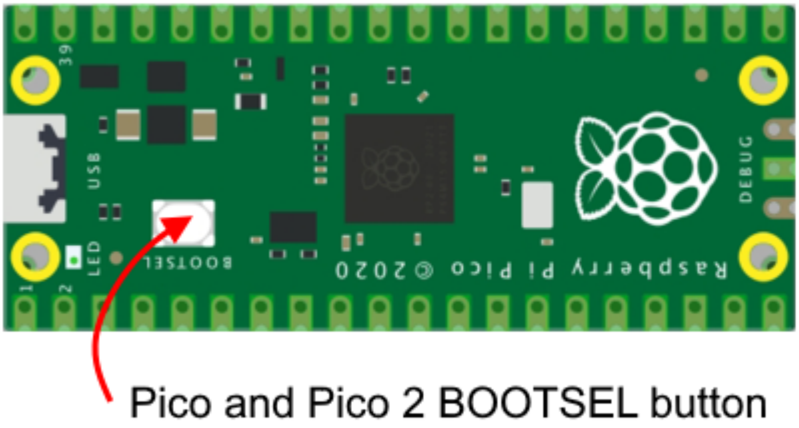
A lot of beginners have a little freakout the first time this happens, they think the board is ruined or 'bricked' - it's almost certainly not, it is just crashed and/or confused. You may need to perform a little trick to get the board back into a good state, at which point you won't need to manually bootload again.

Before continuing we really, really suggest turning on **Verbose Upload** messages, it will help in this process because you will be able to see what the IDE is trying to do. It's a checkbox in the **Preferences** menu.



Enter Manual Bootload Mode

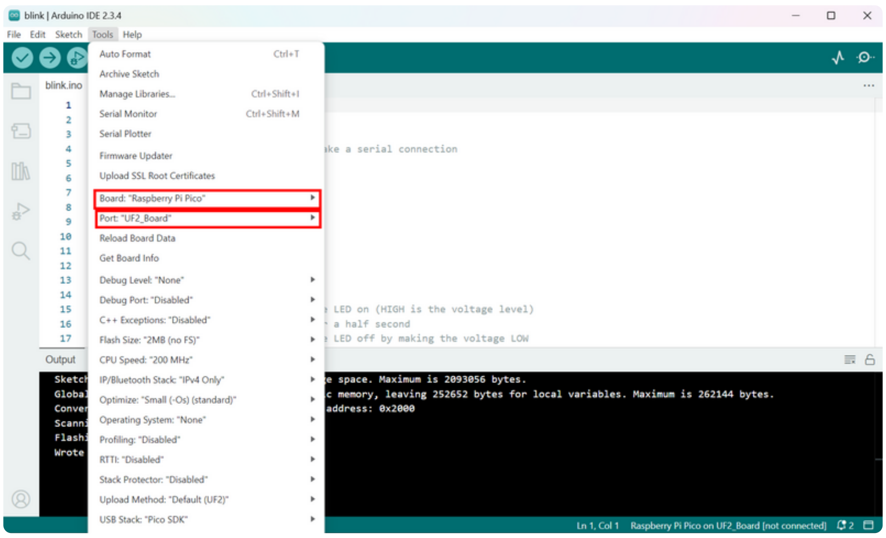
OK now you know it's probably time to try manual bootloading. No problem! Here is how you do that for this board:



Unplug your Raspberry Pi Pico from USB. Ensure the other end of your USB cable is connected to your computer running the Arduino IDE. Press the **BOOTSEL** button and continue holding and plug the USB cable back in to the Pico. Let go of the **BOOTSEL** button. **A new drive named RPI-RP2 should appear.** This is your sign that you can now load the blink sketch by pressing the right arrow button.

If you never get the RPI-RP2 drive, ensure your USB cable is plugged into the computer and not a USB hub and that the cable is a known good data+power USB cable.

Once you are in manual bootload mode, go to the Tools menu, and make sure you have selected the bootloader serial port. **It is almost certain that the serial port has changed now that the bootloader is enabled**



Now you can try uploading again!

Did you remember to select the new Port in the Tools menu since the bootloader port has changed?

This time, you should have success!

After uploading this way, be sure to **click the reset button** - it sort of makes sure that the board got a good reset and will come back to life nicely.

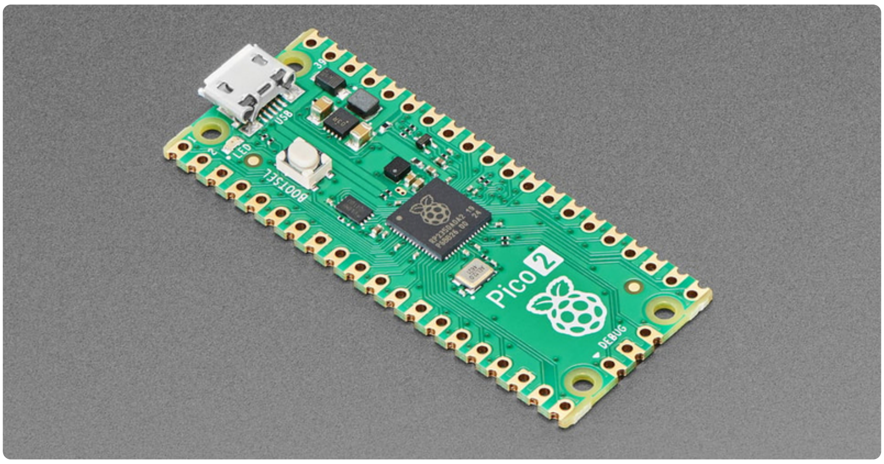
After uploading with Manual Bootloader - don't forget to re-select the old Port again

It's also a good idea to try to re-upload the sketch again now that you've performed a manual bootload to get the chip into a good state. It should perform an auto-reset the second time, so you don't have to manually bootload again.

Finally, a Blink!

OK it was a journey but now we're here and you can enjoy your blinking LED. Next up, try to change the delay between blinks and re-upload. It's a good way to make sure your upload process is smooth and practiced.

Raspberry Pi Pico 2 with RP2350



The upgraded Raspberry Pi Pico 2 uses an RP2350 dual Cortex M33 microcontroller. It increases the computing power by a factor of 2 over the previous RP2040 based boards.

The Raspberry Pi Pico 2 W adds a WiFi and Bluetooth capability.

The Philhower core was augmented to provide support for the RP2350 on the Raspberry Pi Pico 2 (and 2 W).

The following pages describe how to set things up for a Raspberry Pi Pico 2 with RP2350. The process also applies, with appropriate changes, for other boards based on the RP2350 microcontroller.

RP2350 Arduino IDE Setup

The [Arduino Philhower core](https://adafru.it/ToC) (https://adafru.it/ToC) provides support for RP2350 microcontroller boards. This page covers getting your Arduino IDE set up to include your board.

Arduino IDE Download

The first thing you will need to do is to download the latest release of the Arduino IDE. The Philhower core requires **version 1.8** or higher.

Arduino IDE Download

https://adafru.it/Pd5

Download and install it to your computer.

Once installed, open the Arduino IDE.

Adding the Philhower Board Manager URL

In the Arduino IDE, and navigate to the **Preferences** window. You can access it through **File** > **Preferences** on Windows or Linux, or **Arduino** > **Preferences** on OS X.

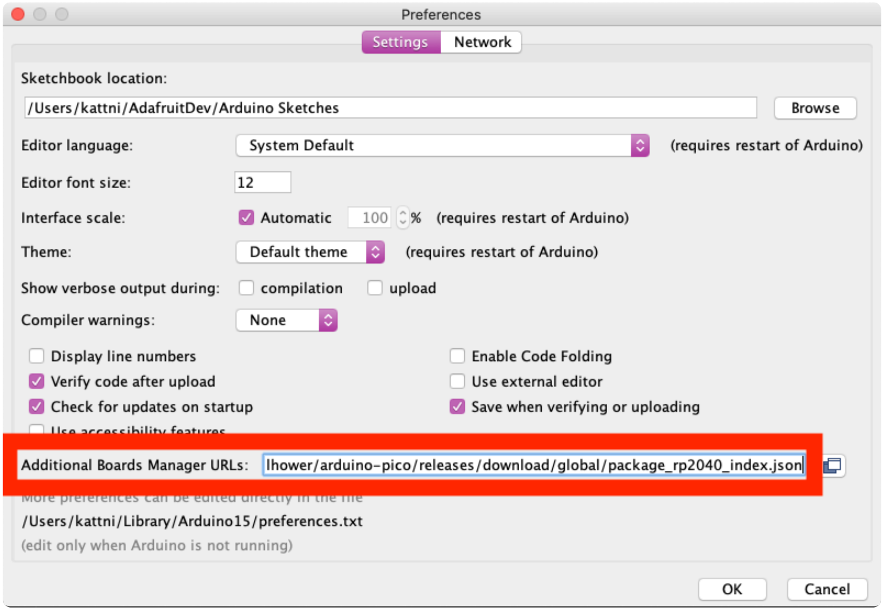
The **Preferences** window will open.

In the **Additional Boards Manager URLs** field, you'll want to add a new URL. The list of URLs is comma separated, and you will only have to add each URL once. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

Copy the following URL.

`https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json`

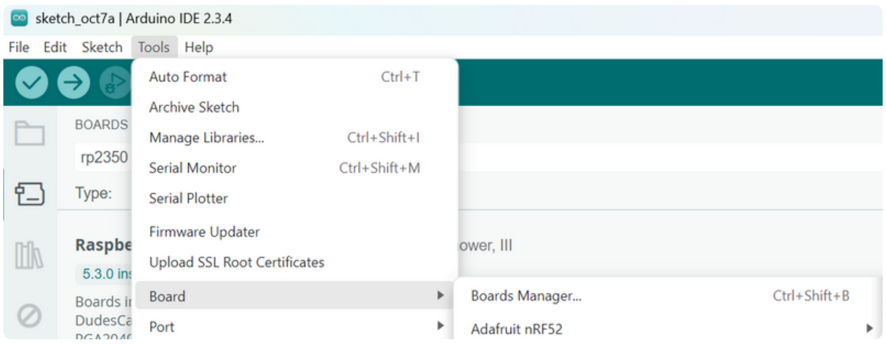
Add the URL to the the **Additional Boards Manager URLs** field (highlighted in red below). If you already have a board support package in that field, type a comma at the end then add the address above. Both will then be loaded by the Arduino IDE.



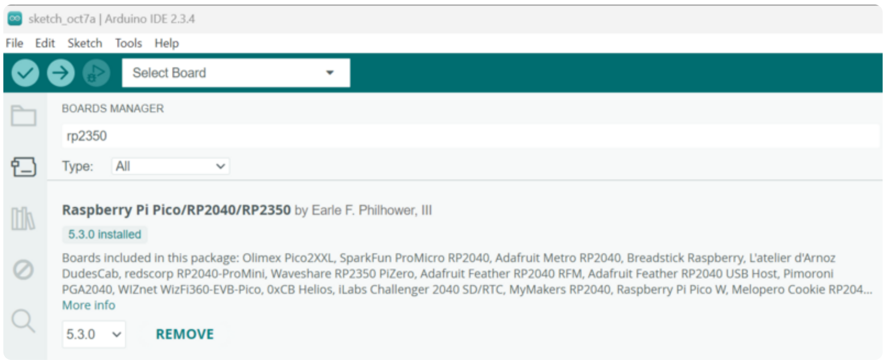
Click **OK** to save and close **Preferences**.

Add Board Support Package

In the Arduino IDE, click on **Tools** > **Board** > **Boards Manager**. If you have previously selected a board, the **Board** menu item may have a board name after it.



In the **Boards Manager**, search for RP2040. Scroll down to the **Raspberry Pi Pico/ RP2040/RP2350 by Earle F Philhower, III** entry. Click **Install** to install it.

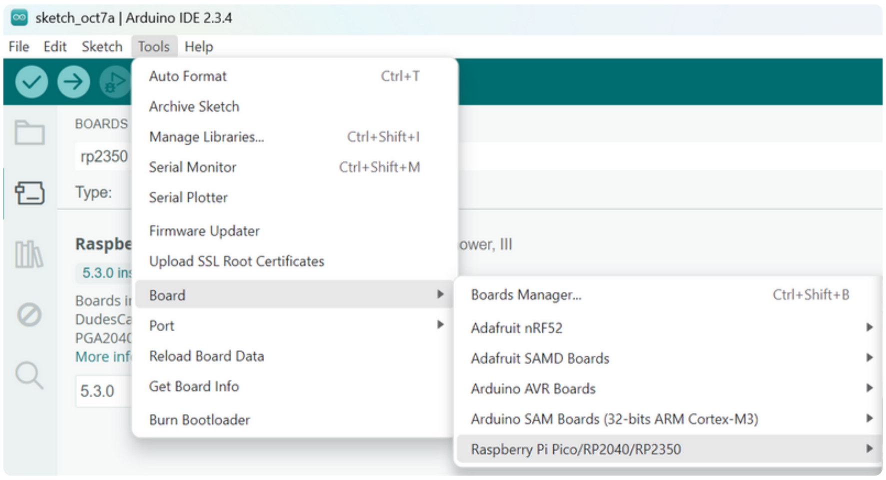


Installing a new board package can take a few minutes. Don't click Cancel!

Once installation is complete, click **Close** to close the Boards Manager.

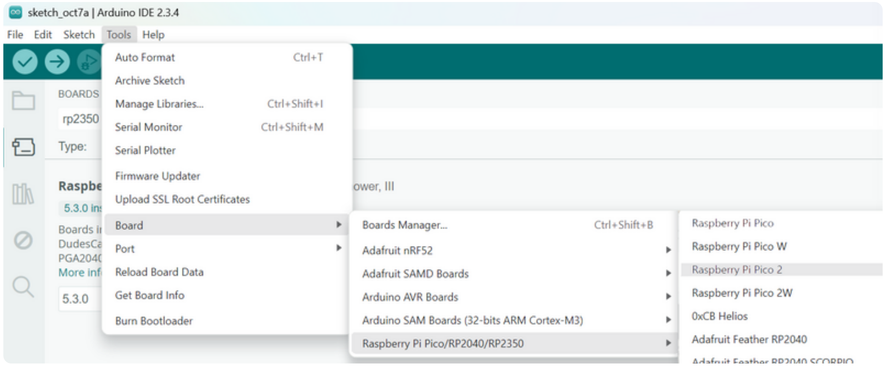
Choose Your Board

In the **Tools** > **Boards** menu, you should now see **Raspberry Pi RP2040/RP2350 Boards (in sketchbook)**.



Navigate to the **Raspberry Pi RP2040/RP2350 Boards** menu. You will see the available boards listed.

Choose **Raspberry Pi Pico 2** from the menu.



If you are programming a Raspberry Pi Pico 2 or Pico 2W, ensure you select the board you have and not a another board as the W boards have LED pinning different (though it should be transparent to the user if it is coded as shown).

Re. If you have a Pico 2 board, do not select Pico, Pico W, or Pico 2 W. If you have a Pico 2 W, select Raspberry Pi Pico 2W and don't select Pico 2 or the Pico boards.

If there is no serial Port available in the dropdown, or an invalid one appears - don't worry about it! The RP2350 does not actually use a serial port to upload, so its OK if it does not appear if in manual bootloader mode. You will see a serial port appear after uploading your first sketch.

Now you're ready to begin using Arduino with your RP2350 board!

RP2350 Arduino Blink

The first and most basic program you can upload to your Arduino is the classic Blink sketch. This takes something on the board and makes it, well, blink! On and off. It's a great way to make sure everything is working and you're uploading your sketch to the right board and right configuration.

When all else fails, you can always come back to Blink!

Pre-Flight Check: Get Arduino IDE & Hardware Set Up

This lesson assumes you have Arduino IDE set up. This is a generalized checklist, some elements may not apply to your hardware. If you haven't yet, check the previous steps in the guide to make sure you:

- **Install the very latest Arduino IDE for Desktop** (not all boards are supported by the Web IDE so we don't recommend it).
- **Install any board support packages (BSP) required for your hardware.** Some boards are built in defaults on the IDE, but lots are not! You may need to install plug-in support which is called the BSP.
- **Get a Data/Sync USB cable for connecting your hardware.** A significant amount of problems folks have stem from not having a USB cable with data pins. Yes, these cursed cables roam the land, making your life hard. If you find a USB cable that doesn't work for data/sync, throw it away immediately! There is no need to keep it around, cables are very inexpensive these days.
- **Install any drivers required** - If you have a board with a FTDI or CP210x chip, you may need to get separate drivers. If your board has native USB, it probably doesn't need anything. After installing, reboot to make sure the driver sinks in.
- **Connect the board to your computer.** If your board has a power LED, make sure its lit. Is there a power switch? Make sure its turned On!

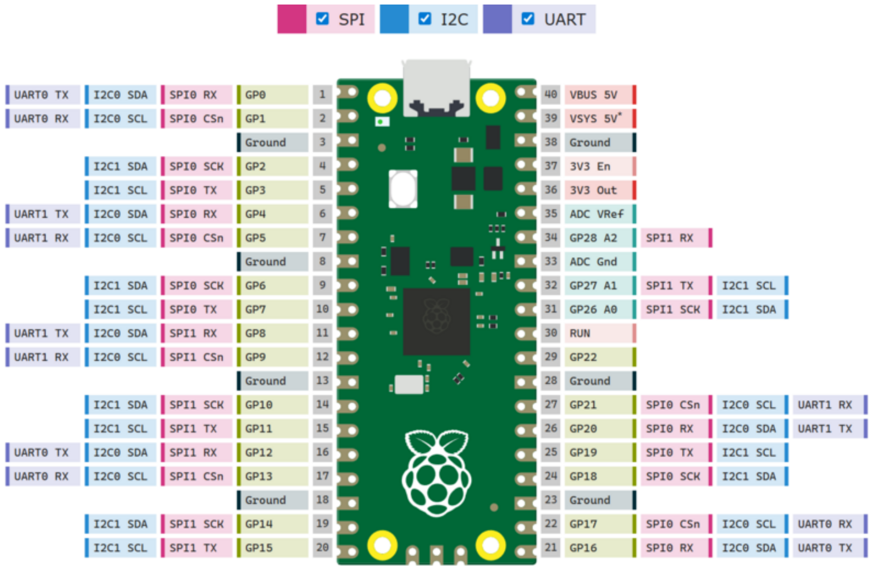


Diagram from pico2.pinout.xyz (https://adafru.it/1aty). Part of gadgetoid.com (https://adafru.it/1atz). License Creative Commons CC BY 3.0 (https://adafru.it/1atA), not remixed or altered.

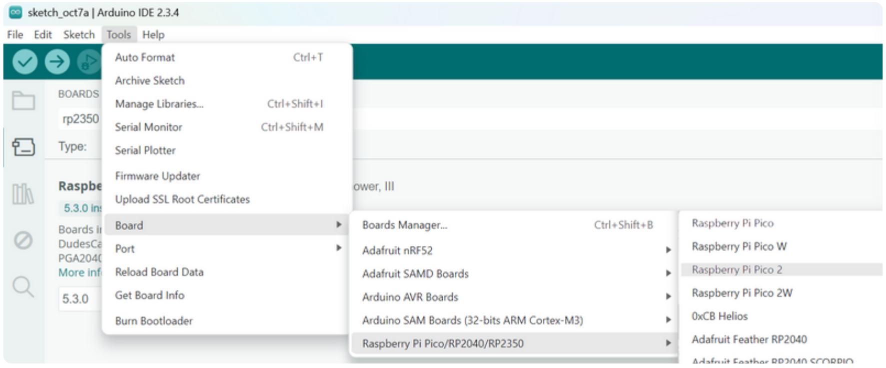
The Pico has the GPIO pin names listed on the bottom of the board as **GPx**, where **x** is the pin number. The Arduino pin name is the number following **GP**. So, for example, pin **GP0** would be Arduino pin `0`.

There is an onboard green LED for user use. Unlike many boards, the Pico lacks a power on LED.

Start up Arduino IDE and Select Board/Port

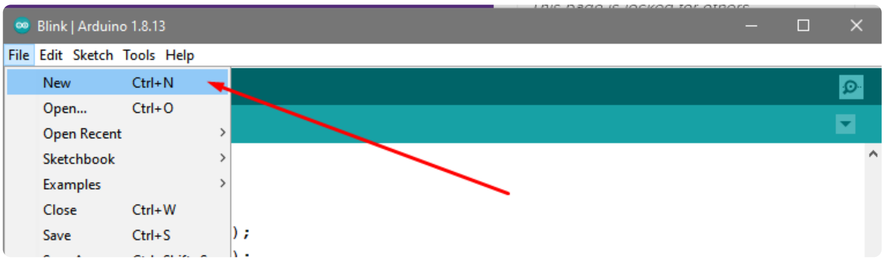
OK now you are prepared! Open the Arduino IDE on your computer. Now you have to tell the IDE what board you are using, and how you want to connect to it.

In the IDE find the **Tools** menu. You will use this to select the board. If you switch boards, you must switch the selection! So always double-check before you upload code in a new session.



New Blink Sketch

OK lets make a new blink sketch! From the **File** menu, select **New**



Then in the new window, copy and paste this text:

```
int led = LED_BUILTIN;

void setup() {
  // Some boards work best if we also make a serial connection
  Serial.begin(115200);

  // set LED to be an output pin
  pinMode(led, OUTPUT);
}

void loop() {
  // Say hi!
  Serial.println("Hello!");

  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                // wait for a half second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                // wait for a half second
}
```

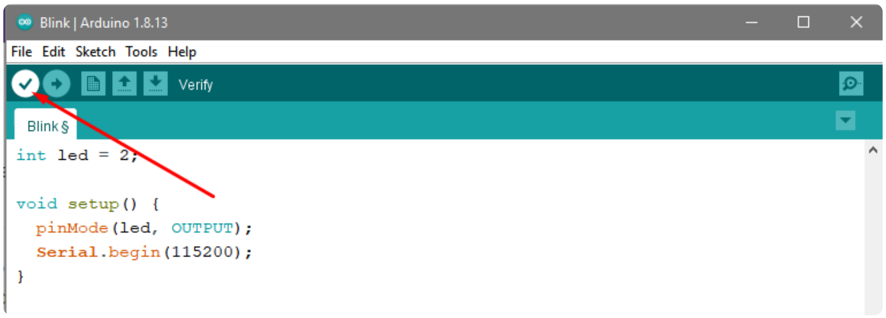
Note that in this example, we are not only blinking the LED but also printing to the Serial monitor, think of it as a little bonus to test the serial connection.

One note you'll see is that we reference the LED with the constant `LED_BUILTIN` rather than a number. That's because, historically, the built in LED was on pin 13 for Arduinos. But in the decades since, boards don't always have a pin 13, or maybe it could not be used for an LED. So the LED could have moved to another pin. It's best to use `LED_BUILTIN` so you don't get the pin number confused!

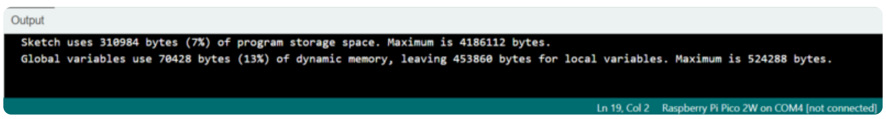
Verify (Compile) Sketch

OK now you can click the Verify button to convert the sketch into binary data to be uploaded to the board.

Note that Verifying a sketch is the same as Compiling a sketch - so we will use the words interchangeably

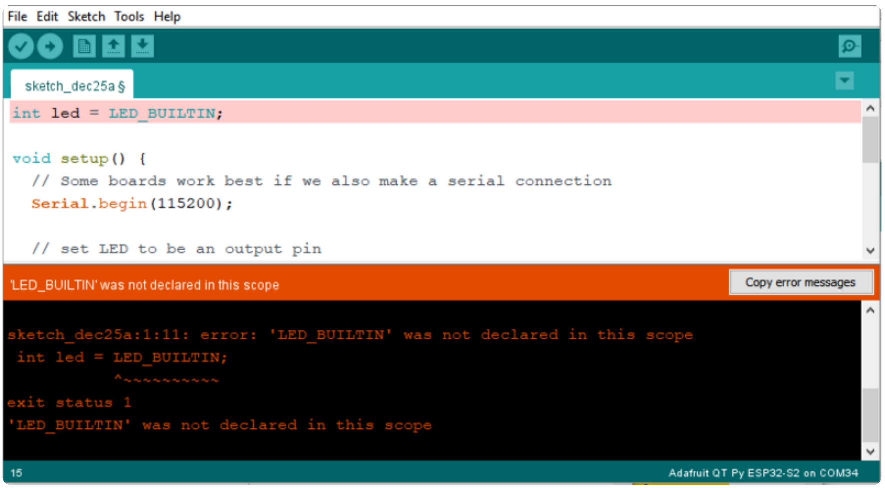


During verification/compilation, the computer will do a bunch of work to collect all the libraries and code and the results will appear in the bottom window of the IDE.

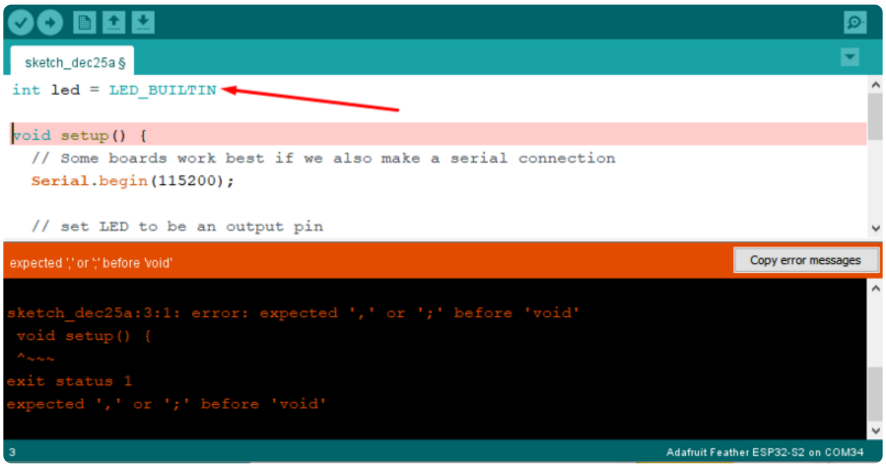


If something went wrong with compilation, you will get red warning/error text in the bottom window letting you know what the error was. It will also highlight the line with an error.

For example, here I had the wrong board selected - and the selected board does not have a built in LED!

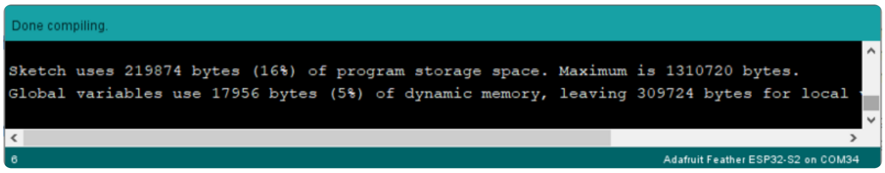


Here's another common error, in my haste I forgot to add a `;` at the end of a line. The compiler warns me that it's looking for one - note that the error is actually a few lines up!



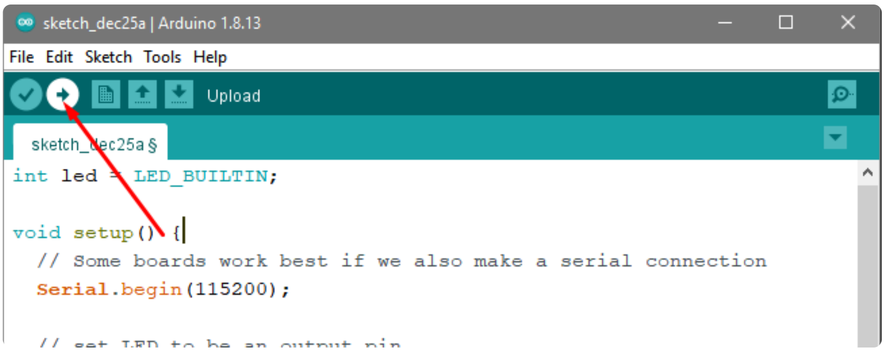
Turning on detailed compilation warnings and output can be very helpful sometimes - Its in Preferences under "Show Verbose Output During:" and check the Compilation button. If you ever need to get help from others, be sure to do this and then provide all the text that is output. It can assist in nailing down what happened!

On success you will see something like this white text output and the message **Done compiling.** in the message area.



Upload Sketch

Once the code is verified/compiling cleanly you can upload it to your board. Click the **Upload** button.



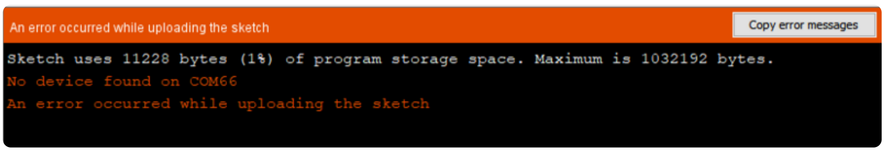
The IDE will try to compile the sketch again for good measure, then it will try to connect to the board and upload a the file.

This is actually one of the hardest parts for beginners because it's where a lot of things can go wrong.

However, lets start with what it looks like on success! Here's what your board upload process looks like when it goes right:

Often times you will get a warning like this, which is kind of vague:

`No device found on COM66` (or whatever port is selected)
`An error occurred while uploading the sketch`



This could be a few things.

First up, check again that you have the correct board selected! Many electronics boards have very similar names or look, and often times folks grab a board different from what they thought.

If you're positive the right board is selected, we recommend the next step is to put the board into manual bootloading mode.

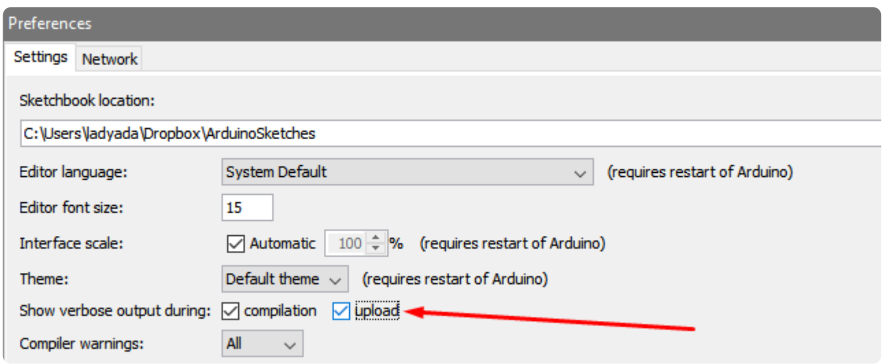
Native USB and manual bootloading

Historically, microcontroller boards contained two chips: the main micro chip (say, ATmega328 or ESP8266 or ESP32) and a separate chip for USB interface that would be used for bootloading (a CH430, FT232, CP210x, etc). With these older designs, the microcontroller is put into a bootloading state for uploading code by the separate chip. It allows for easier uploading but is more expensive as two chips are needed, and also the microcontroller can't act like a keyboard or disk drive.

Modern chips often have 'native' USB - that means that there is no separate chip for USB interface. It's all in one! Great for cost savings, simplicity of design, reduced size and more control. However, it means the chip must be self-aware enough to be able to put itself into bootload/upload mode on its own. That's fine 99% of the time but is very likely you will at some point get the board into an odd state that makes it too confused to bootload.

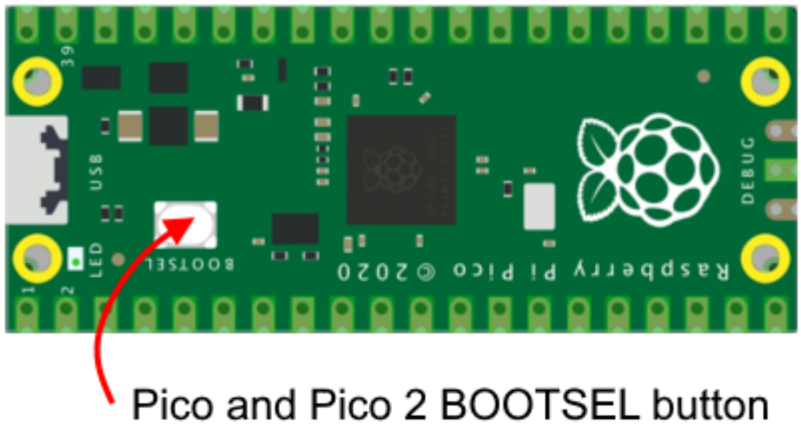
A lot of beginners have a little freakout the first time this happens, they think the board is ruined or 'bricked' - it's almost certainly not, it is just crashed and/or confused. You may need to perform a little trick to get the board back into a good state, at which point you won't need to manually bootload again.

Before continuing we really, really suggest turning on **Verbose Upload** messages, it will help in this process because you will be able to see what the IDE is trying to do. It's a checkbox in the **Preferences** menu.



Enter Manual Bootload Mode

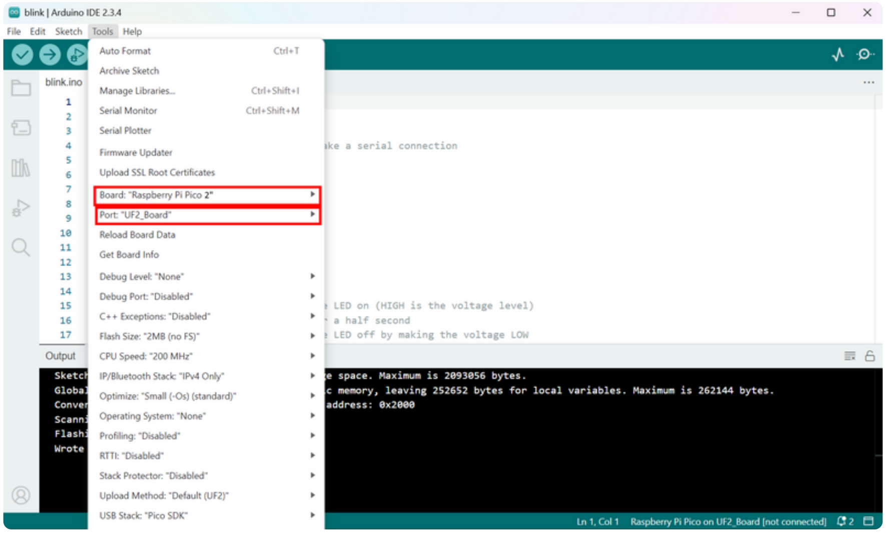
OK now you know it's probably time to try manual bootloading. No problem! Here is how you do that for this board:



Unplug your Raspberry Pi Pico from USB. Ensure the other end of your USB cable is connected to your computer running the Arduino IDE. Press the **BOOTSEL** button and continue holding and plug the USB cable back in to the Pico. Let go of the **BOOTSEL** button. **A new drive named RPI-RP2 should appear.** This is your sign that you can now load the blink sketch by pressing the right arrow button.

If you never get the RPI-RP2 drive, ensure your USB cable is plugged into the computer and not a USB hub and that the cable is a known good data+power USB cable.

Once you are in manual bootload mode, go to the Tools menu, and make sure you have selected the bootloader serial port. **It is almost certain that the serial port has changed now that the bootloader is enabled**



Now you can try uploading again!

Did you remember to select the new Port in the Tools menu since the bootloader port has changed?

This time, you should have success!

After uploading this way, be sure to **click the reset button** - it sort of makes sure that the board got a good reset and will come back to life nicely.

After uploading with Manual Bootloader - don't forget to re-select the old Port again

It's also a good idea to try to re-upload the sketch again now that you've performed a manual bootload to get the chip into a good state. It should perform an auto-reset the second time, so you don't have to manually bootload again.

Finally, a Blink!

OK it was a journey but now we're here and you can enjoy your blinking LED. Next up, try to change the delay between blinks and re-upload. It's a good way to make sure your upload process is smooth and practiced.