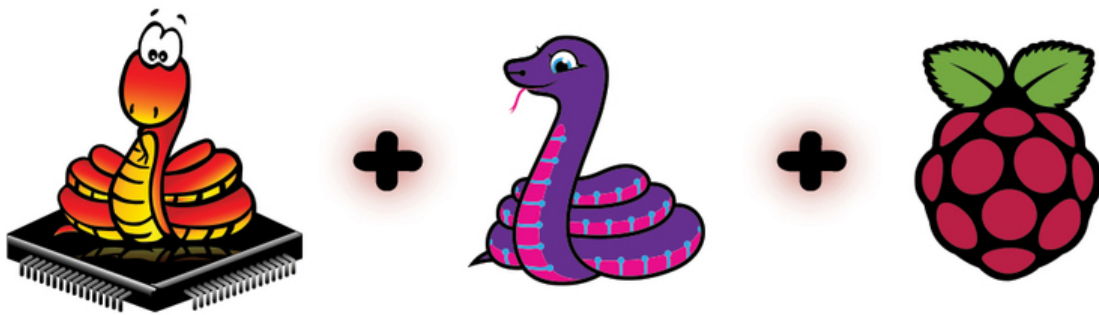




# CircuitPython Libraries on MicroPython using the Raspberry Pi Pico

Created by Melissa LeBlanc-Williams



<https://learn.adafruit.com/circuitpython-libraries-on-micropython-using-the-raspberry-pi-pico>

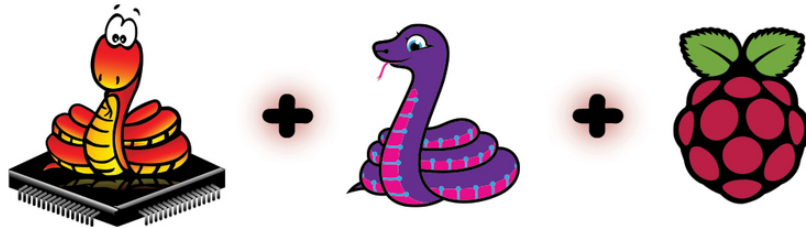
Last updated on 2025-03-01 05:33:49 PM EST

# Table of Contents

Overview	3
• Parts	
Running CircuitPython Code without CircuitPython	5
• Adafruit Blinka: a CircuitPython Compatibility Library	
• Raspberry Pi and Other Single-Board Linux Computers	
• Desktop Computers	
• MicroPython	
• Installing Blinka	
• Installing CircuitPython Libraries	
• Linux Single-Board Computers	
• Desktop Computers using a USB Adapter	
• MicroPython	
Thonny Setup	8
• Download and Install Thonny	
• Configuring Thonny	
MicroPython Installation	9
• Selecting the Port	
Installing Blinka and Libraries	13
• Using the File Transfer Interface	
• Install Blinka and PlatformDetect	
• Install CircuitPython Libraries	
BME280 Library Example	17
• Wiring	
• Example Code	

---

# Overview



Blinka is our CircuitPython compatibility layer. It was originally written as a compatibility layer to run on top of MicroPython to work on boards such as the PyBoard and allow libraries written for CircuitPython to work. This allowed more library re-use and fewer libraries that needed to be maintained.

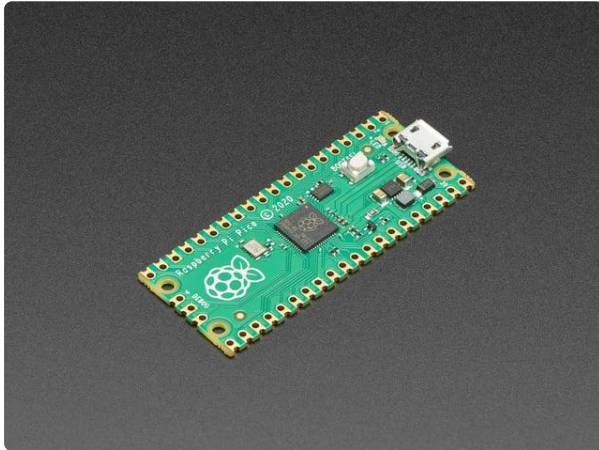
Blinka was soon adapted to work on Linux-based Single Board Computers to allow those same libraries to work on boards such as the Linux-based Raspberry Pi computers (not to be confused with the Raspberry Pi Pico microcontrollers). It gained a lot of popularity there was expanded to run on many more boards. It wasn't used much on MicroPython at that point and there weren't many guides that showed how to run Blinka over MicroPython.

When we recently tried to run Blinka over MicroPython, there were a few things that needed to be fixed up to get it working. While we were at it, we decided to add support for the Raspberry Pi Pico when running MicroPython.

There's already another way to run Blinka on the Raspberry Pi Pico using the U2IF firmware, but the difference is that Blinka is running on the host computer and the Raspberry Pi Pico is acting as a passthrough. If you're interested in running it that way, be sure to check out our [CircuitPython Libraries on any Computer with Raspberry Pi Pico \(https://adafru.it/Sje\)](https://adafru.it/Sje) guide.

Additionally, you can always run CircuitPython natively on Raspberry Pi Pico. However, if you would like to run Blinka on a Raspberry Pi Pico or any other supported board running MicroPython, then this is the guide for you.

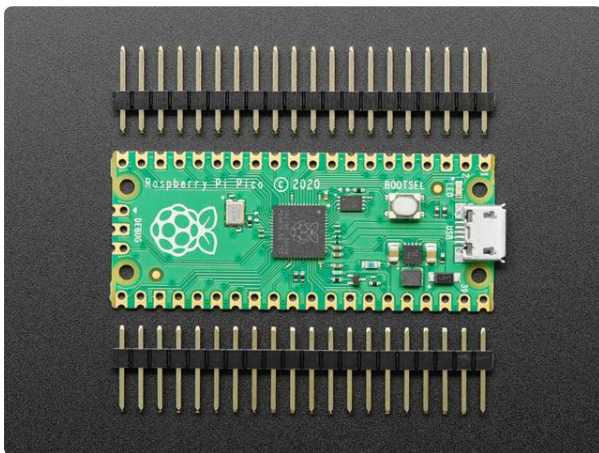
## Parts



### [Raspberry Pi Pico RP2040](https://www.adafruit.com/product/4864)

The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...

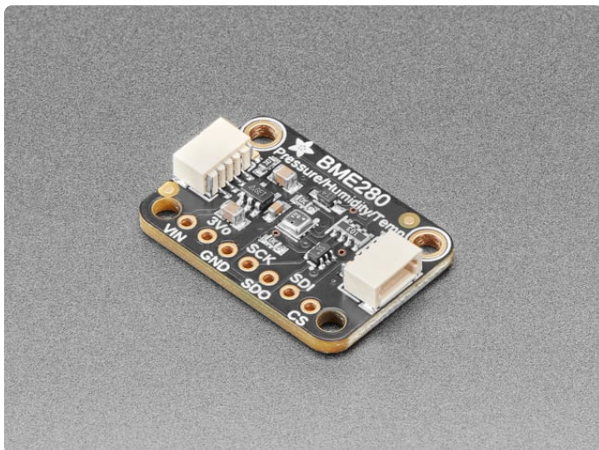
<https://www.adafruit.com/product/4864>



### [Raspberry Pi Pico RP2040 with Loose Unsoldered Headers](https://www.adafruit.com/product/4883)

The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're...

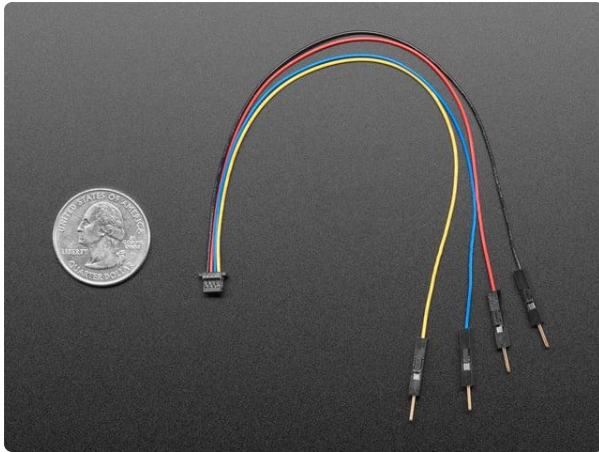
<https://www.adafruit.com/product/4883>



### [Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor](https://www.adafruit.com/product/2652)

Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts...

<https://www.adafruit.com/product/2652>



### STEMMA QT / Qwiic JST SH 4-pin to Premium Male Headers Cable

This 4-wire cable is a little over 150mm / 6" long and fitted with JST-SH female 4-pin connectors on one end and premium Dupont male headers on the other.

Compared with the...

<https://www.adafruit.com/product/4209>

---

## Running CircuitPython Code without CircuitPython

There are two parts to the CircuitPython ecosystem:

- **CircuitPython firmware**, written in C and built to run on various microcontroller boards (not PCs). The firmware includes the CircuitPython interpreter, which reads and executes CircuitPython programs, and chip-specific code that controls the hardware peripherals on the microcontroller, including things like USB, I2C, SPI, GPIO pins, and all the rest of the hardware features the chip provides.
- **CircuitPython libraries**, written in Python to use the native (built into the firmware) modules provided by CircuitPython to control the microcontroller peripherals and interact with various breakout boards.

But suppose you'd like to use CircuitPython **libraries** on a board or computer that does not have a native CircuitPython **firmware** build. For example, on a PC running Windows or macOS. Can that be done? The answer is yes, via a separate piece of software called **Blinka**. Details about Blinka follow, however it is important to realize that the **CircuitPython firmware is never used**.

CircuitPython firmware is NOT used when using Blinka.

# Adafruit Blinka: a CircuitPython Compatibility Library

Enter **Adafruit Blinka**. Blinka is a software library that emulates the parts of CircuitPython that control hardware. Blinka provides non-CircuitPython implementations for `board`, `busio`, `digitalio`, and other native CircuitPython modules. You can then write Python code that looks like CircuitPython and uses CircuitPython libraries, without having CircuitPython underneath.

There are multiple ways to use Blinka:

- Linux based Single Board Computers, for example a Raspberry Pi
- Desktop Computers + specialized USB adapters
- Boards running MicroPython

More details on these options follow.

## Raspberry Pi and Other Single-Board Linux Computers

On a Raspberry Pi or other single-board Linux computer, you can use Blinka with the regular version of Python supplied with the Linux distribution. Blinka can control the hardware pins these boards provide.

## Desktop Computers

On Windows, macOS, or Linux desktop or laptop ("host") computers, you can use special USB adapter boards that provide hardware pins you can control. These boards include [MCP221A \(https://adafru.it/IfV\)](https://adafru.it/IfV) and [FT232H \(https://adafru.it/xia\)](https://adafru.it/xia) breakout boards, and [Raspberry Pi Pico boards running the u2if software \(https://adafru.it/Sje\)](https://adafru.it/Sje). These boards connect via regular USB to your host computer, and let you do GPIO, I2C, SPI, and other hardware operations.

## MicroPython

You can also use Blinka with MicroPython, on [MicroPython-supported boards \(https://adafru.it/SBi\)](https://adafru.it/SBi). Blinka will allow you to import and use CircuitPython libraries in your MicroPython program, so you don't have to rewrite libraries into native MicroPython code. Fun fact - this is actually the original use case for Blinka.

## Installing Blinka

Installing Blinka on your particular platform is covered elsewhere in this guide. The process is different for each platform. Follow the guide section specific to your

platform and make sure Blinka is properly installed before attempting to install any libraries.

Be sure to install Blinka before proceeding.

## Installing CircuitPython Libraries

Once Blinka is installed the next step is to install the CircuitPython libraries of interest. How this is done is different for each platform. Here are the details.

### Linux Single-Board Computers

On Linux single-board computers, such as Raspberry Pi, you'll use the Python `pip3` program (sometimes named just `pip`) to install a library. The library will be downloaded from [pypi.org](https://pypi.org) (<https://adafru.it/19ff>) automatically by `pip3`.

How to install a particular library using `pip3` is covered in the guide page for that library. For example, [here is the `pip3` installation information](https://adafru.it/OkF) (<https://adafru.it/OkF>) for the library for the LIS3DH accelerometer.

The library name you give to `pip3` is usually of the form `adafruit-circuitpython-libraryname`. This is not the name you use with `import`. For example, the LIS3DH sensor library is known by several names:

- The GitHub library repository is [Adafruit\\_CircuitPython\\_LIS3DH](https://adafru.it/uBs) (<https://adafru.it/uBs>).
- When you import the library, you write `import adafruit_lis3dh`.
- The name you use with `pip3` is `adafruit-circuitpython-lis3dh`. This the name used on [pypi.org](https://pypi.org) (<https://adafru.it/19ff>).

Libraries often depend on other libraries. When you install a library with `pip3`, it will automatically install other needed libraries.

### Desktop Computers using a USB Adapter

When you use a desktop computer with a USB adapter, like the MCP2221A, FT232H, or u2if firmware on an RP2040, you will also use `pip3`. However, **do not install the library with `sudo pip3`**, as mentioned in some guides. Instead, just install with `pip3`.



# MicroPython

For MicroPython, you will not use `pip3`. Instead you can get the library from the CircuitPython bundles. See [this guide page \(https://adafru.it/ABU\)](https://adafru.it/ABU) for more information about the bundles, and also see the [Libraries page on circuitPython.org \(https://adafru.it/ENC\)](https://adafru.it/ENC).

---

## Thonny Setup

The first step is to get Thonny Setup on your system. If you already have Thonny installed, then great! You can continue onto configuring Thonny.

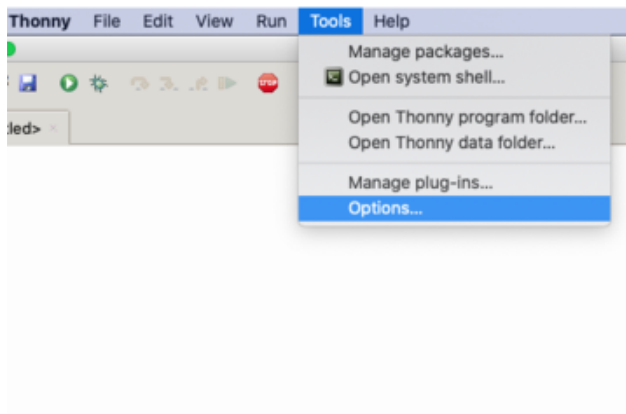
### Download and Install Thonny

Thonny runs on a variety of systems including Windows, MacOS, and Linux. It is the easiest way to copy files over to MicroPython. Head over to <https://thonny.org/> (<https://adafru.it/Qb6>) and download the latest version of Thonny. Once that's downloaded, run the installer and go through the steps.

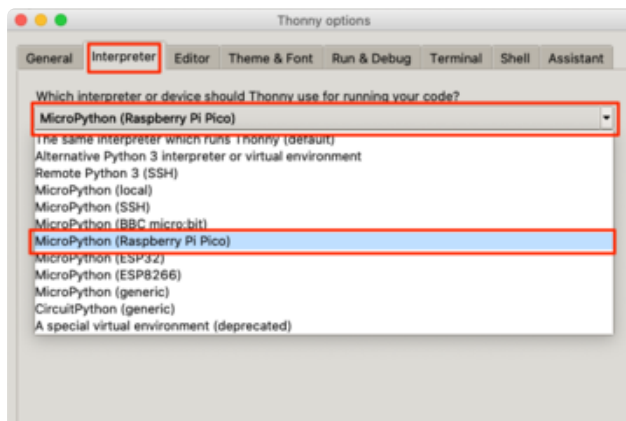
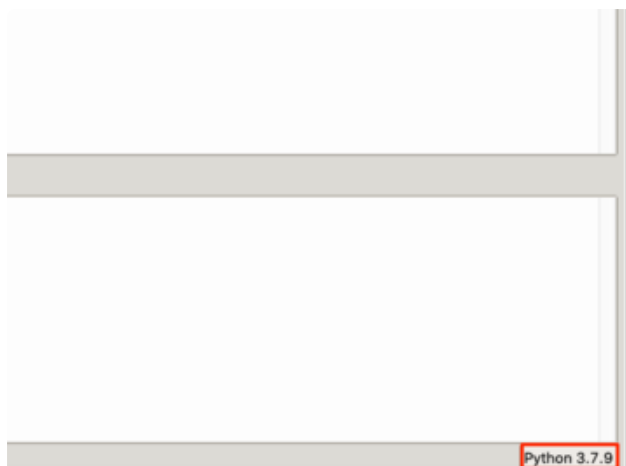
### Configuring Thonny

To use Thonny with the Raspberry Pi Pico, you will need to configure a few things. Go ahead and pen up the application if it isn't already running.





You'll first want to go to **Tools → Options**. Additionally, you can get here by selecting the current interpreter in the corner and choosing **Configure Interpreter...**

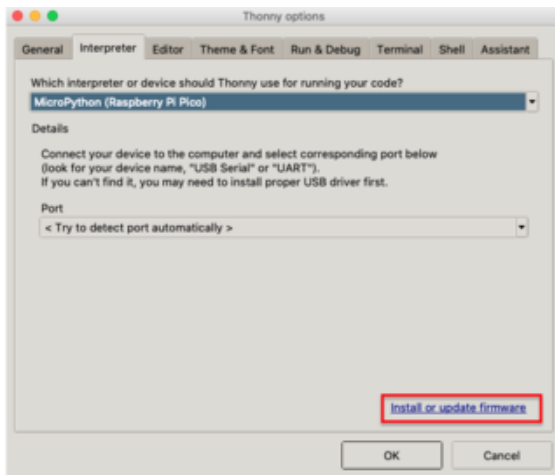


On the options screen, go to the **Interpreter** tab.

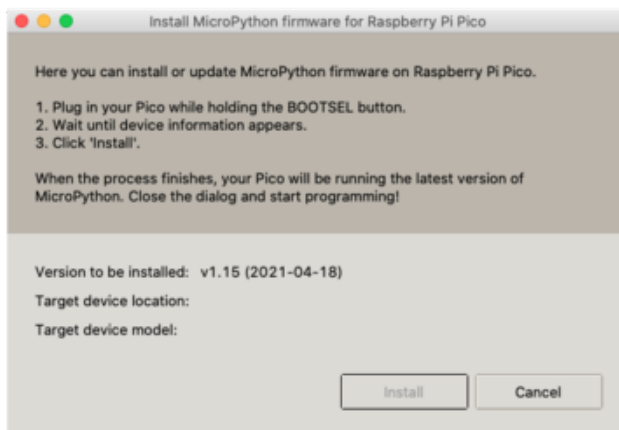
From the first dropdown, make sure **MicroPython (Raspberry Pi Pico)** is selected.

## MicroPython Installation

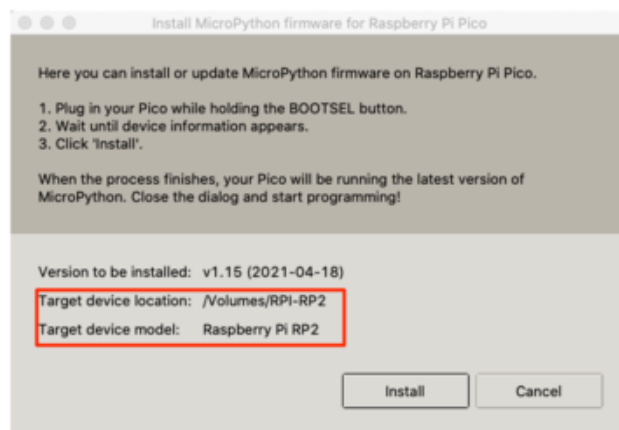
If you already have your board in Bootloader Mode and you have **MicroPython (Raspberry Pi Pico)** selected, it will automatically come up with the installation screen. If it does not, follow the steps below to install.



On the Interpreter Tab screen, there is an **Install or Update Firmware** link. Go ahead and click it.

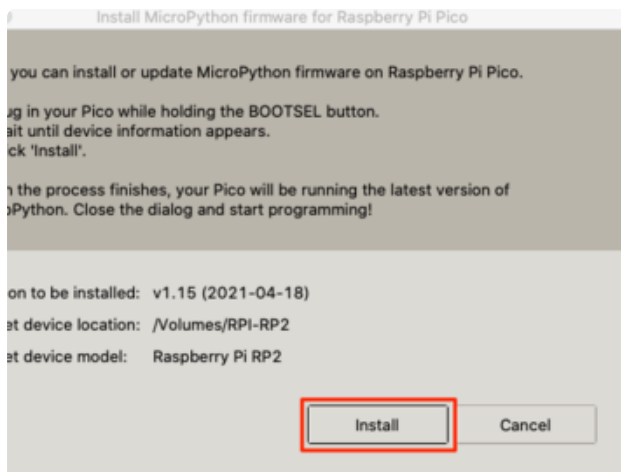


This will take you to a Firmware Install Screen.

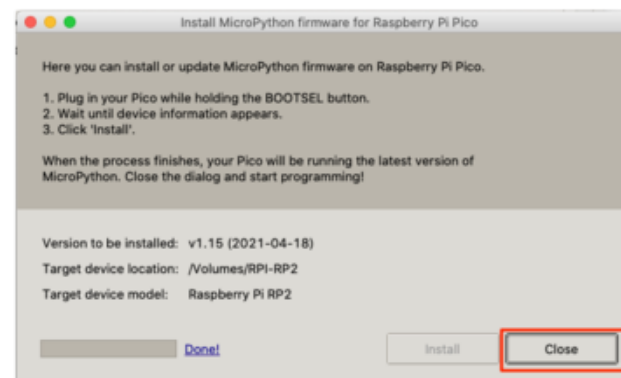
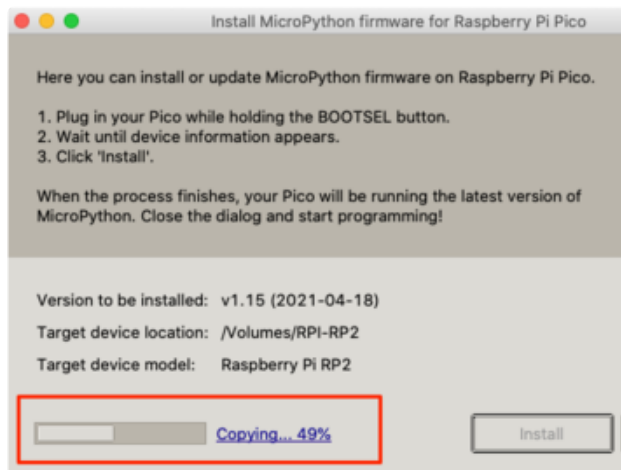


While holding the **BOOTSEL** button, plug in your Pico. It should show up under the Target device fields.

If the board is new out of the package, you do not need to hold **BOOTSEL** while plugging it in.

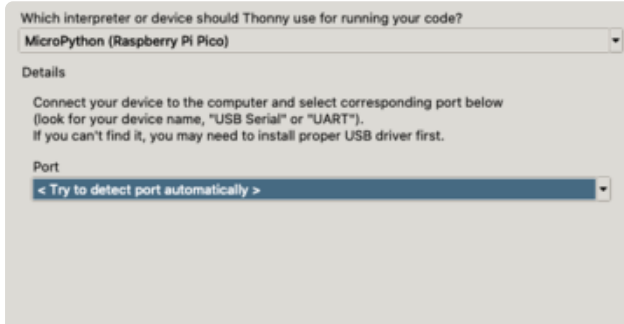


Click the **Install** button to begin installation. It will take a few seconds for it to copy the firmware onto the board.

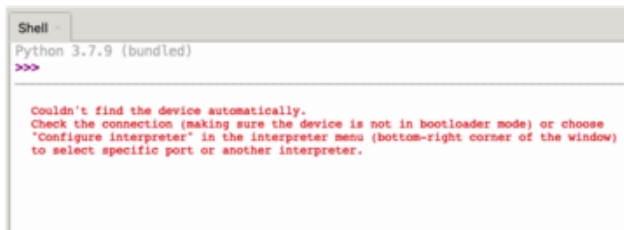


When it has finished copying, click **Close**.

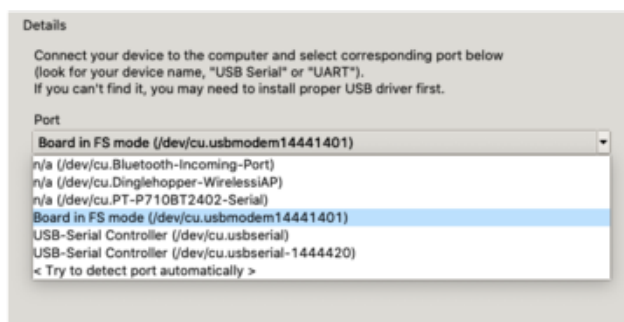
# Selecting the Port



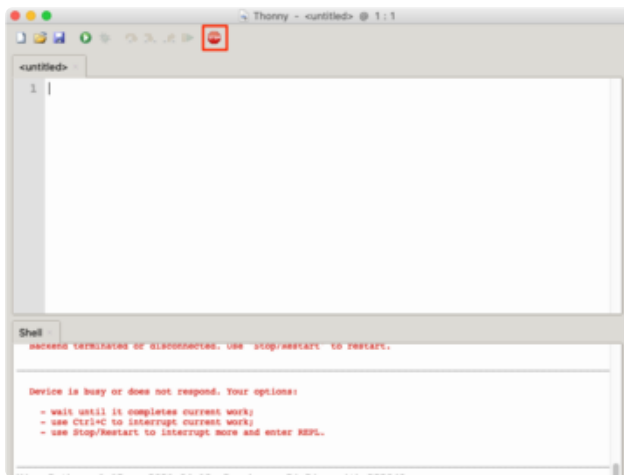
You can try having it detect the port automatically.



If it comes up with a message that it couldn't find the device, try hitting the Stop/Restart button at the top a few times.



You can always select your port manually. Select the your port from the Dropdown menu. These will vary from system to system.



If it comes up with an error that the device is busy, try clicking on the **Stop/Restart Button** at the top.

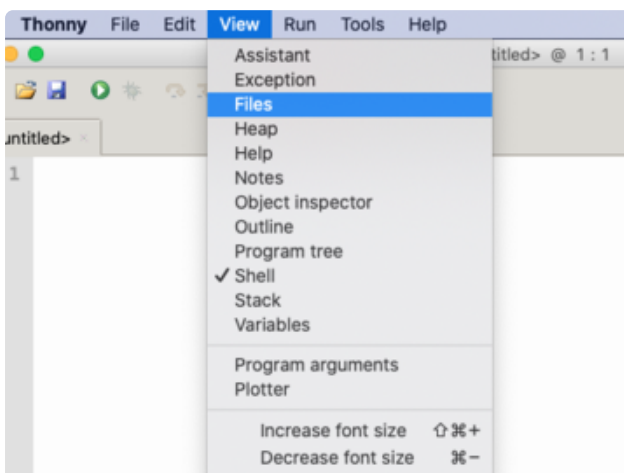


Once it connects successfully, the Shell window should show MicroPython running on the Raspberry Pi Pico.

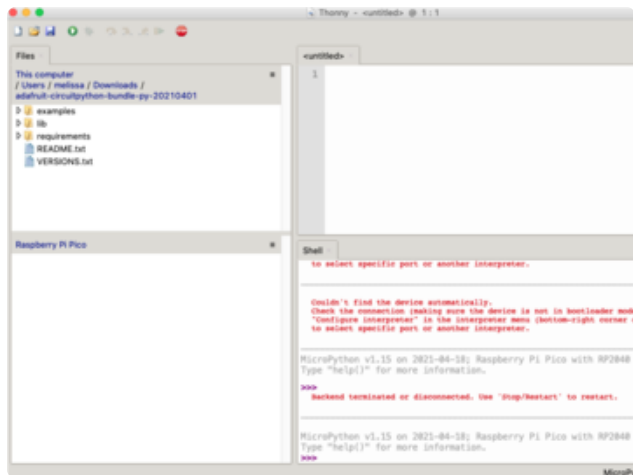
## Installing Blinka and Libraries

Installing Blinka and CircuitPython libraries is very easy using Thonny. It has a built-in file transfer feature that makes copying the files very easy.

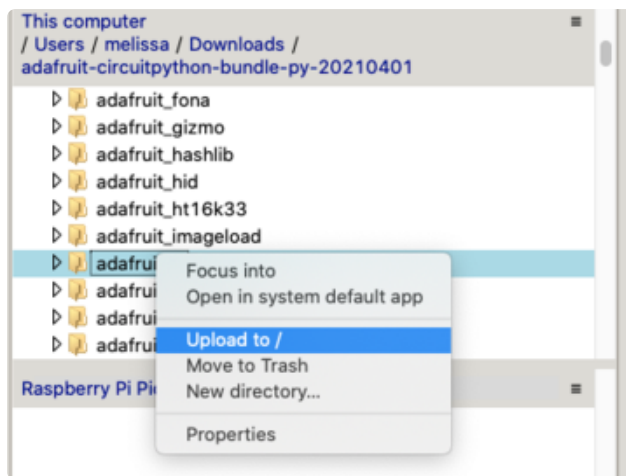
### Using the File Transfer Interface



First you'll want to show the file transfer pane. You can get there by clicking View → Files.



A files pane should appear on the left side of the main window. You can resize the elements to make it easier to see. Your local files should be at the top while the files on your Raspberry Pi Pico are on the bottom.



To upload, right-click on a folder or file, choose **Upload to** and it will upload to the current folder you are in on the Pico.

## Install Blinka and PlatformDetect

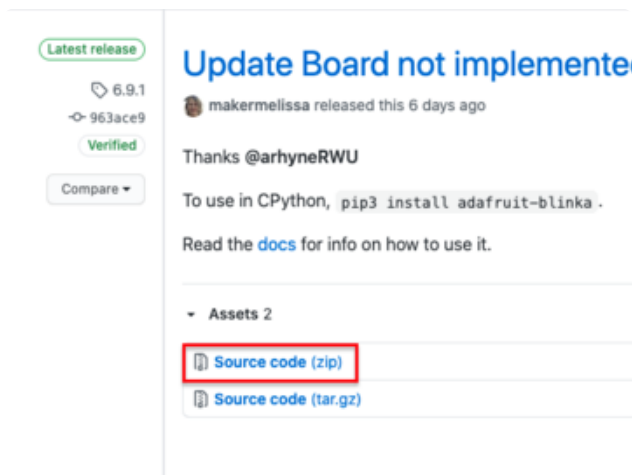
Blinka and PlatformDetect are both required to run Blinka on MicroPython. Neither of them are included in the bundle, so you will need to download them from GitHub. You can download the latest releases from GitHub by following these links:

Download the latest Blinka from  
GitHub

<https://adafru.it/Sjf>

Download the latest PlatformDetect  
from GitHub

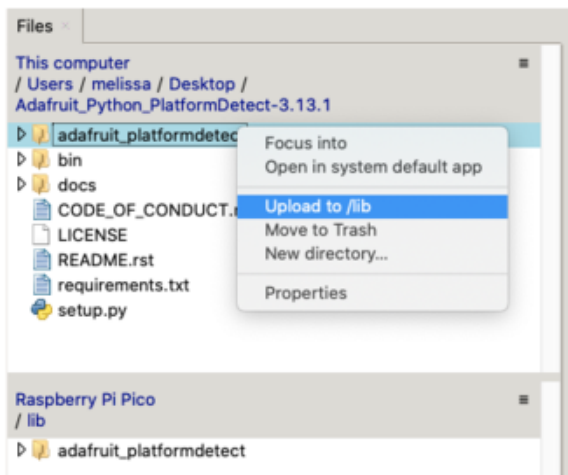
<https://adafru.it/SjA>



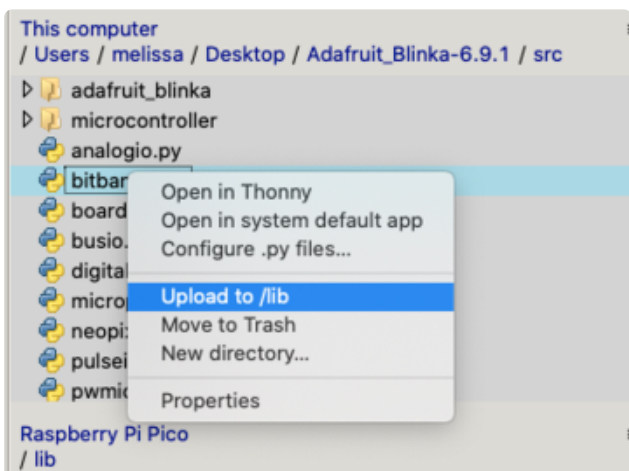
Click the **Source code (zip)** link to download.

Once you have downloaded the release files, go ahead and unzip them using your favorite zip utility software and place the files in a safe location such as your desktop.

Blinka, PlatformDetect, and libraries can be installed in either the root of the Pico or you can create a lib folder and copy the libraries inside of there.



To install the PlatformDetect, use Thonny's Files pane to navigate to the location where you unzipped the files and upload the **adafruit\_platformdetect** folder to the Pico.

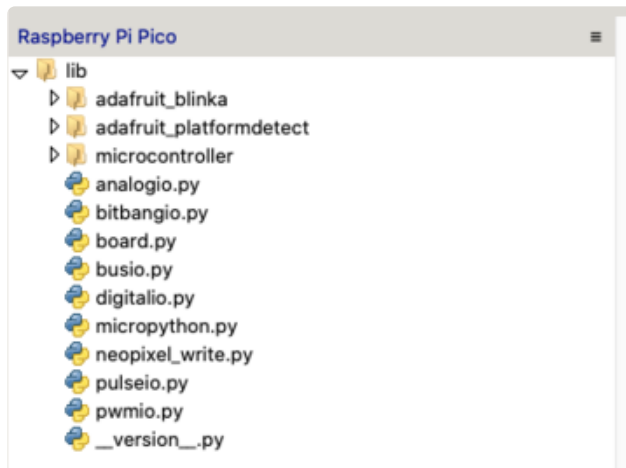


To install the Blinka, navigate to the location where you unzipped the Blinka files and upload everything **inside of the src** folder to the Pico.

The easiest way to do that is to go inside of the src folder, highlight everything, right-click, and choose **Upload to**.



If you want to free up some more room, many of the linux board and microcontroller files and folders can be removed under the **adafruit\_blinka** folder. The only ones used by the Pico are the **adafruit\_blinka/microcontroller/generic\_micropython** folder, the **adafruit\_blinka/microcontroller/rp2040** folder, and **adafruit\_blinka/board/raspberrypi/pico.py**.



After you finish copying all of the files, this is what your Raspberry Pi Pico should look like.

Libraries in MPY format will not work in MicroPython. You'll need to use the libraries ending in .py.

## Install CircuitPython Libraries

First start by downloading the latest CircuitPython library bundle from [circuitpython.org](https://circuitpython.org).

**Download the latest CircuitPython library bundle**

<https://adafru.it/ENC>

Download the **adafruit-circuitpython-bundle-py-\*.zip** bundle zip file, and unzip a folder of the same name. Inside you'll find a **lib** folder. The entire collection of libraries is too large to fit on the Raspberry Pi Pico. Instead, add each library as you need it, this will reduce the space usage but you'll need to put in a little more effort.

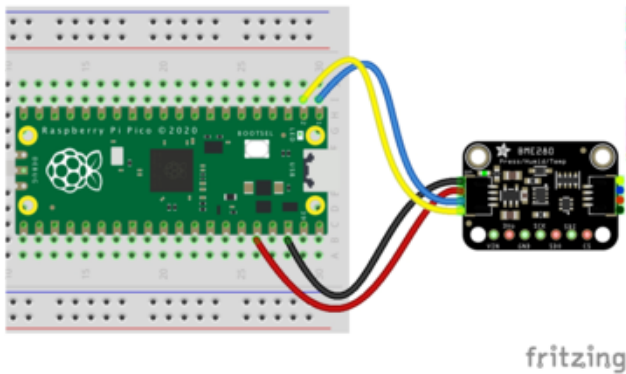
Like you did with Blinka and PlatformDetect, use Thonny's File Transfer system to navigate to the bundle folder. Select any library files or folders you want to transfer, and select **Upload to**.

# BME280 Library Example

For this example, we're going to use the Adafruit BME280 sensor, which uses I2C since there are so many sensors that use I2C.

The space left by MicroPython is quite limited. We recommend removing any unnecessary files or you may run into memory errors.

## Wiring



Connect the **red wire** from the STEMMA QT connector on the BME280 to **pin 36** or **3.3V out** on the Pico.

Connect the **black wire** from the STEMMA QT connector on the BME280 to **pin 38** or **Gnd** on the Pico.

Connect the **blue wire** or SDA from the STEMMA QT connector on the BME280 to **pin 1** or **GP0** on the Pico.

Connect the **yellow wire** or SCL from the STEMMA QT connector on the BME280 to **pin 2** or **GP1** on the Pico.

## Example Code

Go ahead and click **Download Project Bundle** to download the examples and dependent libraries. Unzip the bundle and upload the libraries as described on the **Install Blinka and Libraries** page.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import busio
from adafruit_bme280 import basic as adafruit_bme280

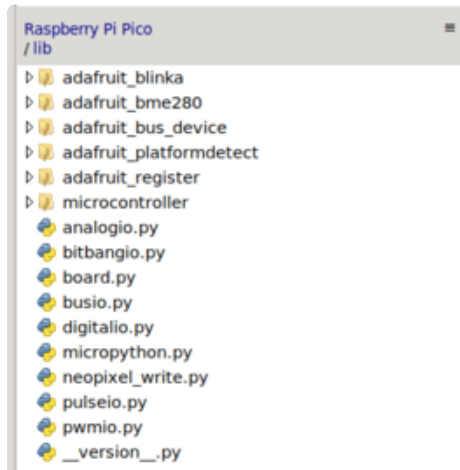
# Create sensor object, using the board's default I2C bus.
i2c = busio.I2C(board.GP1, board.GP0) # SCL, SDA
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# OR create sensor object, using the board's default SPI bus.
# spi = busio.SPI(board.GP2, MISO=board.GP0, MOSI=board.GP3)
# bme_cs = digitalio.DigitalInOut(board.GP1)
```

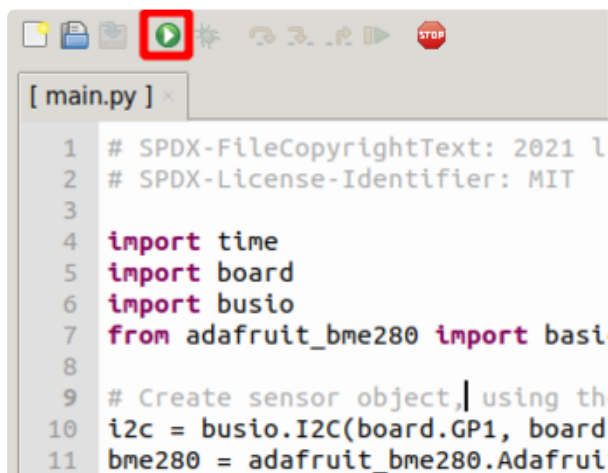
```
# bme280 = adafruit_bme280.Adafruit_BME280_SPI(spi, bme_cs)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bme280.temperature)
    print("Humidity: %0.1f %" % bme280.relative_humidity)
    print("Pressure: %0.1f hPa" % bme280.pressure)
    print("Altitude = %0.2f meters" % bme280.altitude)
    time.sleep(2)
```

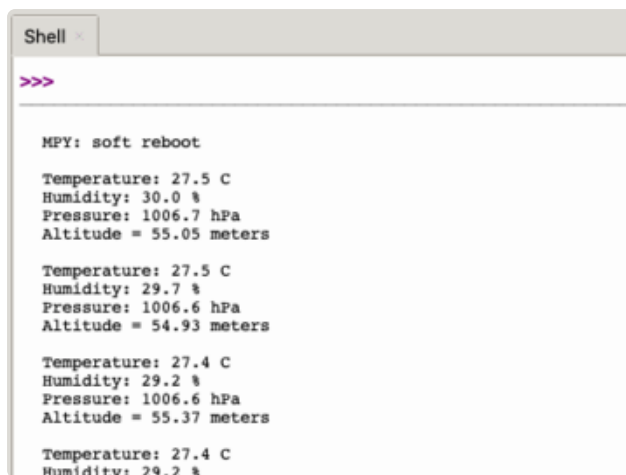


Here's what the files on your Raspberry Pi Pico should look like with the libraries added in.



Press the **green play button** to run the script from the editor.

You can also copy the code onto the pico using Thonny, save it onto the Raspberry Pi Pico as **main.py** in the **root folder** if you would like it to run each time. You will need to press Control+D to perform a soft restart if you save it to the Pico.



```
Shell <
>>>

MPY: soft reboot

Temperature: 27.5 C
Humidity: 30.0 %
Pressure: 1006.7 hPa
Altitude = 55.05 meters

Temperature: 27.5 C
Humidity: 29.7 %
Pressure: 1006.6 hPa
Altitude = 54.93 meters

Temperature: 27.4 C
Humidity: 29.2 %
Pressure: 1006.6 hPa
Altitude = 55.37 meters

Temperature: 27.4 C
Humidity: 29.2 %
```

It will show you the temperature and humidity information in the Shell pane.