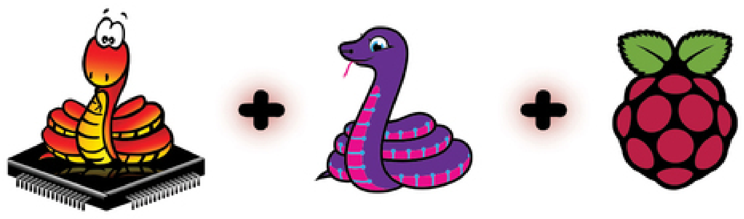
adafruit learning system

CircuitPython Libraries on MicroPython using the Raspberry Pi Pico

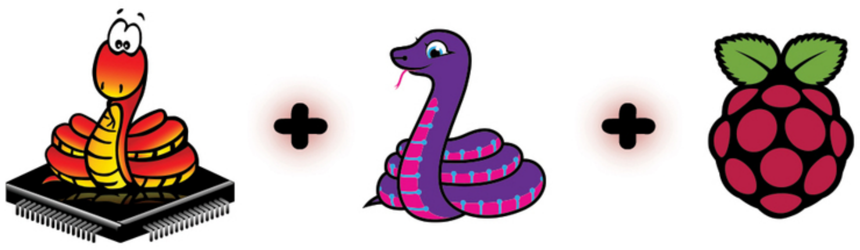
Created by Melissa LeBlanc-Williams



Last updated on 2021-06-07 05:49:37 PM EDT

Guide Contents

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) 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

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

Out of Stock

Out of Stock

Raspberry Pi Pico RP2040 with Loose Unsoldered Headers

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

Out of Stock

Out of Stock

Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor

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...

Out of Stock

Out of Stock

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...

$0.95

In Stock

Add to Cart

Thonny Setup

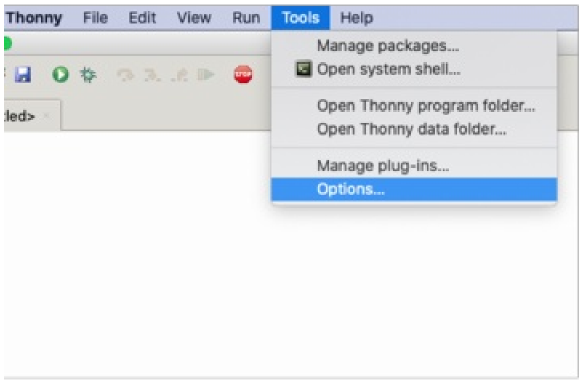
The first step is to get Thonny Setup on your system. If you already have Thonny installed, then great! You can continue onto configuring Thonny.

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.

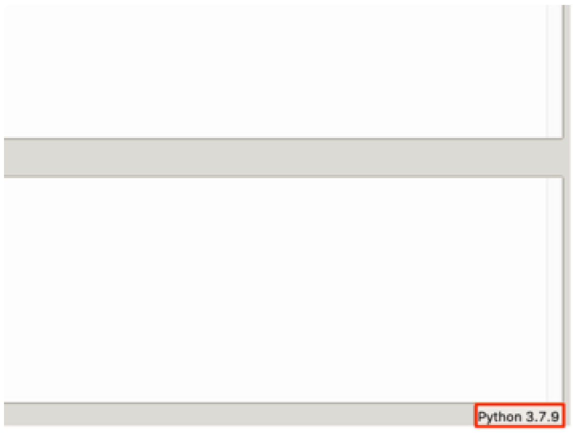
Configuring Thonny

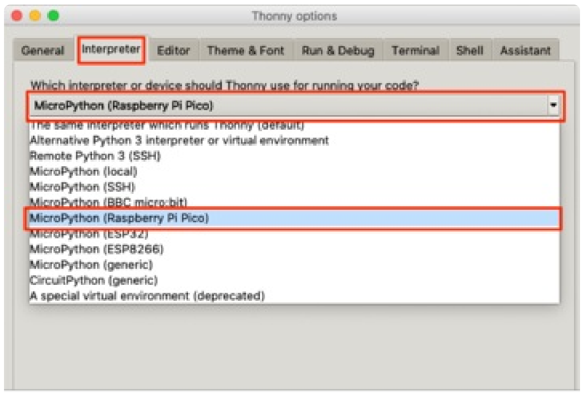
To use Thonny with the Raspberry Pi Pico, you will need to configure a few things. Go ahead and pen up the application if it isn't already running.



Thonny File Edit View Run Tools Help
Manage packages...
Open system shell...
Open Thonny program folder...
Open Thonny data folder...
Manage plug-ins...
Options...

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...**





Which interpreter or device should Thonny use for running your code?

MicroPython (Raspberry Pi Pico)

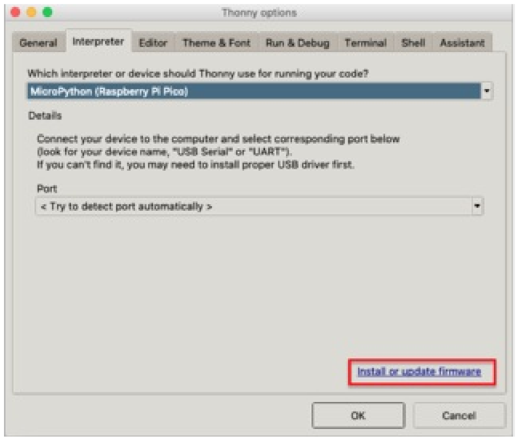
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
Remote Python 3 (SSH)
MicroPython (local)
MicroPython (SSH)
MicroPython (BBC micro:bit)
MicroPython (Raspberry Pi Pico)
MicroPython (ESP32)
MicroPython (ESP8266)
MicroPython (generic)
CircuitPython (generic)
A special virtual environment (deprecated)

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.



Which interpreter or device should Thonny use for running your code?
MicroPython (Raspberry Pi Pico)

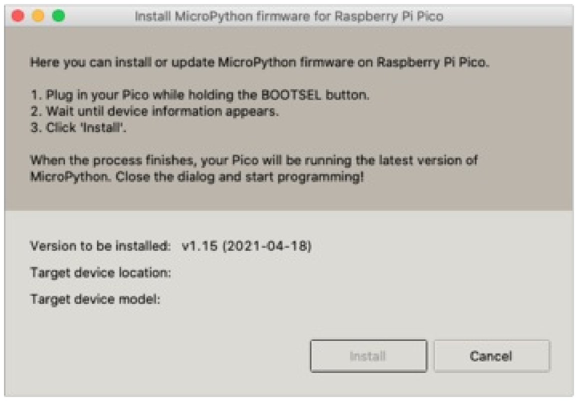
Details

Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port
< Try to detect port automatically >

Install or update firmware

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



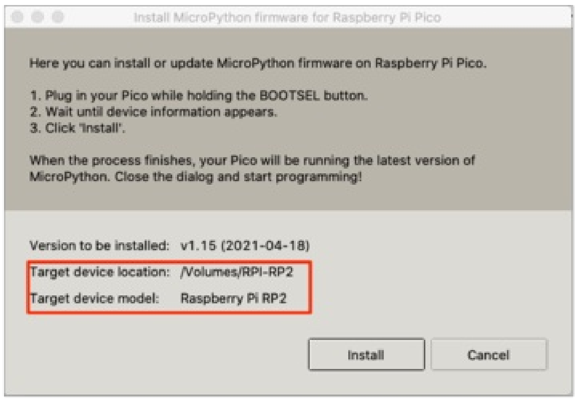
Here you can install or update MicroPython firmware on Raspberry Pi Pico.

1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

Version to be installed: v1.15 (2021-04-18)
Target device location:
Target device model:

This will take you to a Firmware Install Screen.


Install MicroPython firmware for Raspberry Pi Pico

Here you can install or update MicroPython firmware on Raspberry Pi Pico.

1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

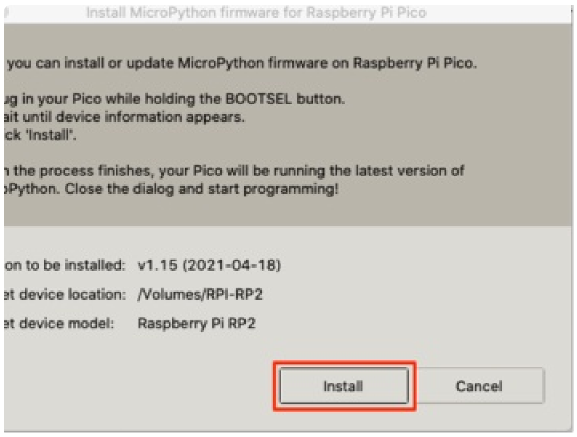
Version to be installed: v1.15 (2021-04-18)

Target device location: /Volumes/RPI-RP2

Target device model: Raspberry Pi RP2

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.

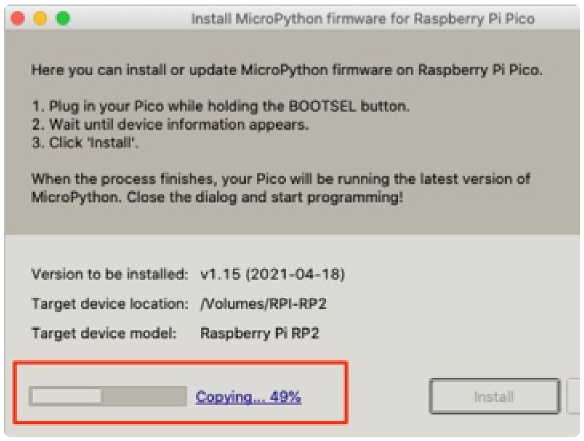

Install MicroPython firmware for Raspberry Pi Pico

you can install or update MicroPython firmware on Raspberry Pi Pico.

ug in your Pico while holding the BOOTSEL button.
ait until device information appears.
ck 'Install'.

n the process finishes, your Pico will be running the latest version of
oPython. Close the dialog and start programming!

on to be installed: v1.15 (2021-04-18)
et device location: /Volumes/RPI-RP2
et device model: Raspberry Pi RP2


Install MicroPython firmware for Raspberry Pi Pico

Here you can install or update MicroPython firmware on Raspberry Pi Pico.

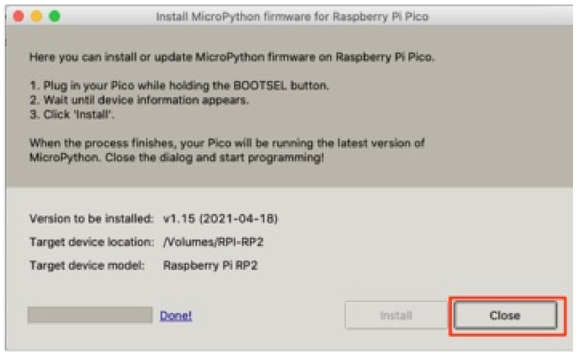
1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

Version to be installed: v1.15 (2021-04-18)
Target device location: /Volumes/RPI-RP2
Target device model: Raspberry Pi RP2

Copying... 49%

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


Install MicroPython firmware for Raspberry Pi Pico

Here you can install or update MicroPython firmware on Raspberry Pi Pico.

1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

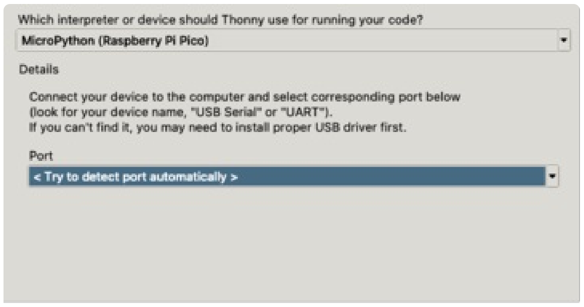
When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

Version to be installed: v1.15 (2021-04-18)
Target device location: /Volumes/RPI-RP2
Target device model: Raspberry Pi RP2

Done!

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

Selecting the Port



Which interpreter or device should Thonny use for running your code?

MicroPython (Raspberry Pi Pico)

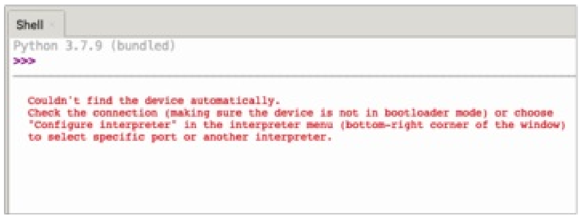
Details

Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port

< Try to detect port automatically >

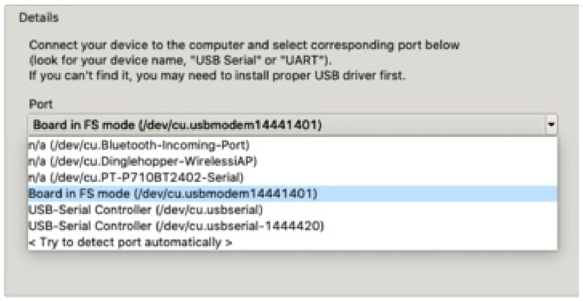
You can try having it detect the port automatically.



```
Python 3.7.9 (bundled)
>>>

Couldn't find the device automatically.
Check the connection (making sure the device is not in bootloader mode) or choose
"Configure interpreter" in the interpreter menu (bottom-right corner of the window)
to select specific port or another interpreter.
```

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.



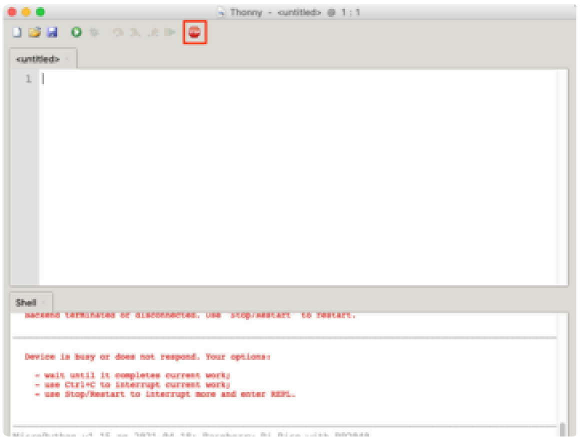
Details

Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port

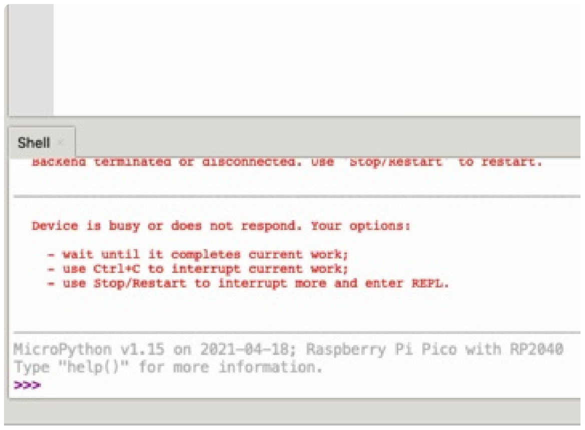
Board in FS mode (/dev/cu.usbmodem14441401)

n/a (/dev/cu.Bluetooth-Incoming-Port)
n/a (/dev/cu.Dinglehopper-WirelessiAP)
n/a (/dev/cu.PT-P710BT2402-Serial)
Board in FS mode (/dev/cu.usbmodem14441401)
USB-Serial Controller (/dev/cu.usbserial)
USB-Serial Controller (/dev/cu.usbserial-1444420)
< Try to detect port automatically >



```
Backend terminated or disconnected. Use 'Stop/Restart' to restart.

Device is busy or does not respond. Your options:

  - wait until it completes current work;
  - use Ctrl+C to interrupt current work;
  - use Stop/Restart to interrupt more and enter REPL.
```

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



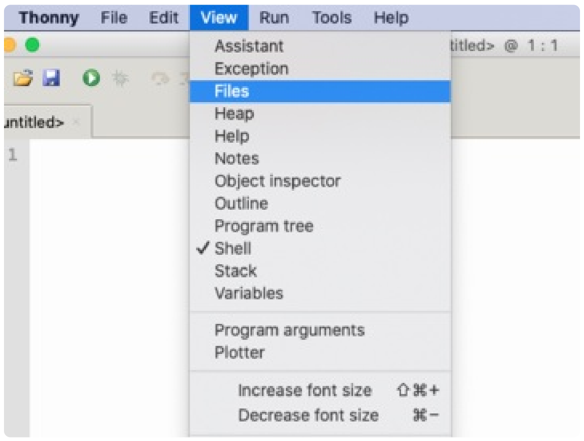
```
Backend terminated or disconnected. Use 'Stop/Restart' to restart.

Device is busy or does not respond. Your options:

  - wait until it completes current work;
  - use Ctrl+C to interrupt current work;
  - use Stop/Restart to interrupt more and enter REPL.

MicroPython v1.15 on 2021-04-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

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

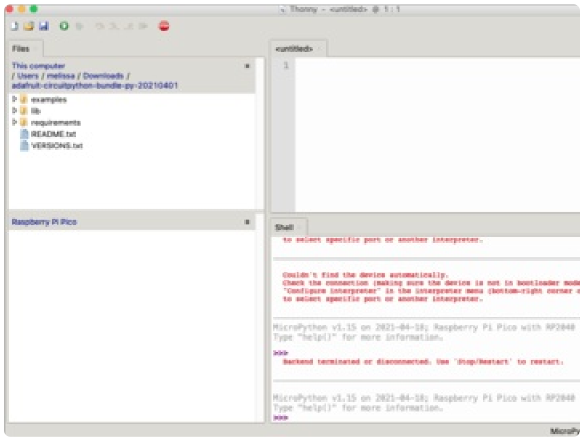
Installing Blinka and Libraries

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

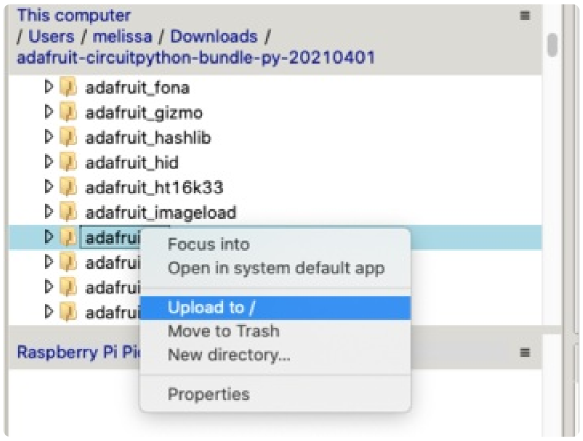


First you'll want to show the file transfer pane. You can get there by clicking View → Files.



```
to select specific port or another interpreter.

Couldn't find the device automatically.
Check the connection (making sure the device is not in bootloader mode
"Configure interpreter" in the interpreter menu (bottom-right corner o
to select specific port or another interpreter.

MicroPython v1.15 on 2021-04-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
Backend terminated or disconnected. Use 'Stop/Restart' to restart.

MicroPython v1.15 on 2021-04-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

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

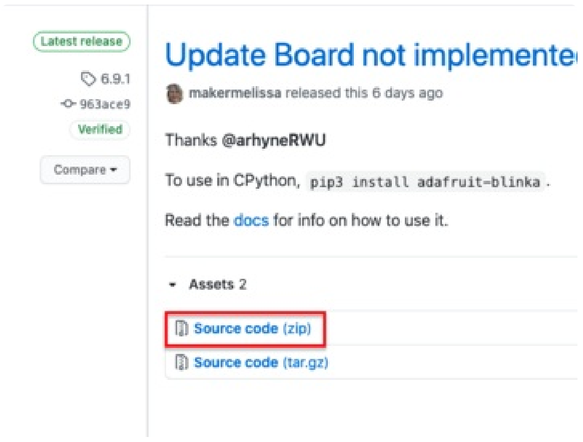
Blinka and PlatformDetect are both required to run Blinka on MicroPython. Neither of them are included in the bundle, so you will need to download them from GitHub. You can download the latest releases from GitHub by following these links:

https://adafru.it/Sjf

https://adafru.it/Sjf

https://adafru.it/SjA

https://adafru.it/SjA



Thanks @arhyneRWU

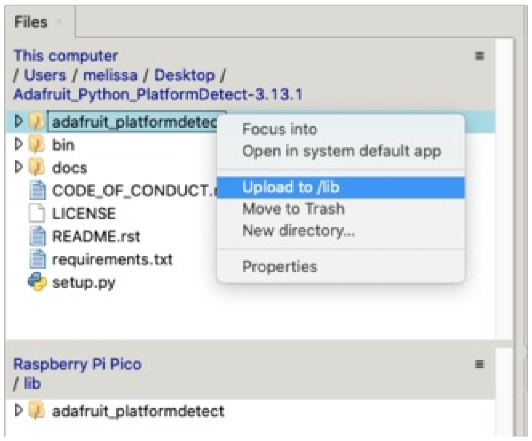
To use in CPython, `pip3 install adafruit-blinka`.

Read the docs for info on how to use it.

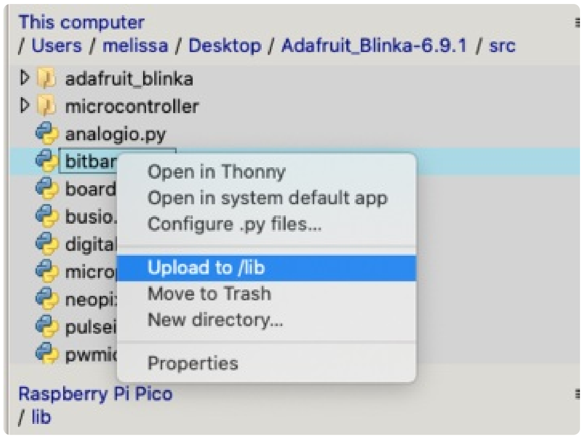
Click the **Source code (zip)** link to download.

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.



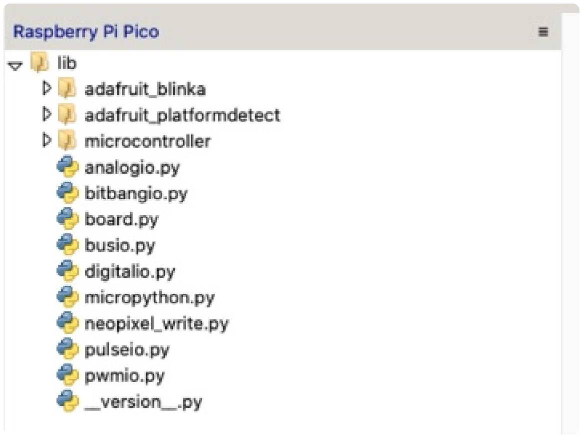
To install the PlatformDetect, use Thonny's Files pane to navigate to the location where you unzipped the files and upload the **adafruit_platformdetect** folder to the Pico.



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 can be removed under the adafruit_blinka folder. The only ones used by the Pico are the adafruit_blinka/microcontroller/rp2040.py 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://adafru.it/ENC

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.

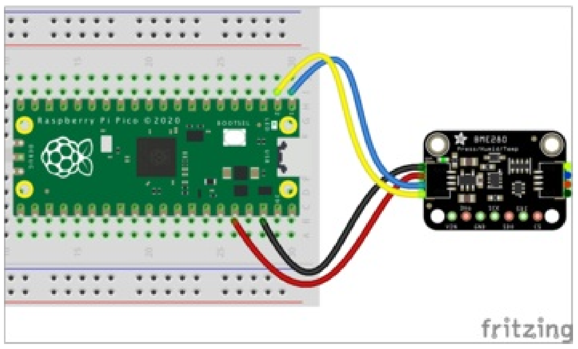
Instead, add each library as you need it, this will reduce the space usage but you'll need to put in a little more effort.

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.

Wiring

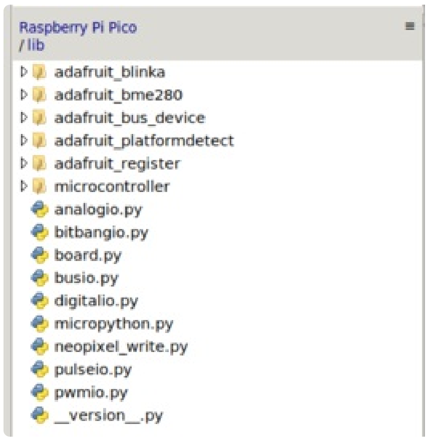


- Connect the **red wire** from the STEMMA QT connector on the BME280 to **pin 36** or **3.3V out** on the Pico.
- Connect the **black wire** from the STEMMA QT connector on the BME280 to **pin 38** or **Gnd** on the Pico.
- Connect the **blue wire** or SDA from the STEMMA QT connector on the BME280 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.

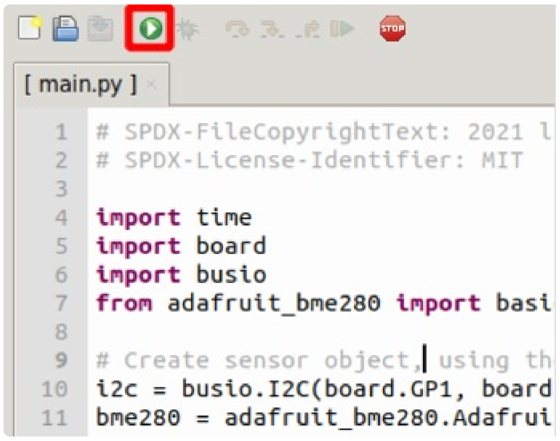
```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

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)
```



Raspberry Pi Pico
/ lib

```
adafruit_blinka
adafruit_bme280
adafruit_bus_device
adafruit_platformdetect
adafruit_register
microcontroller
analogio.py
bitbangio.py
board.py
busio.py
digitalio.py
micropython.py
neopixel_write.py
pulseio.py
pwmio.py
__version__.py
```

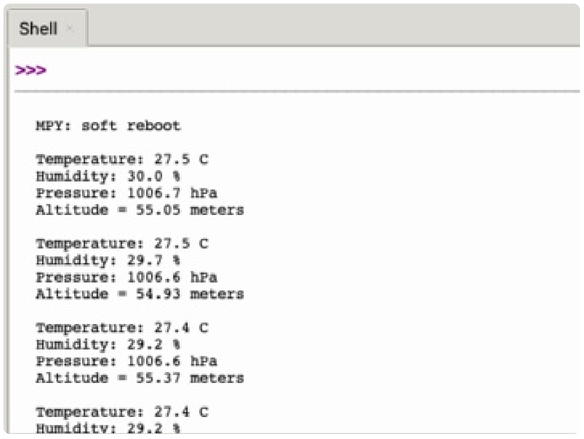
Here's what the files on your Raspberry Pi Pico should look like with the libraries added in.



```
# SPDX-FileCopyrightText: 2021 l
# SPDX-License-Identifier: MIT

import time
import board
import busio
from adafruit_bme280 import basi

# Create sensor object, using th
i2c = busio.I2C(board.GP1, board
bme280 = adafruit_bme280.Adafrui
```

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.



```
>>>

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.