

Getting started with the Raspberry Pi Pico 2 in MicroPython



Online guide,
always up to date

From installing MicroPython to your first LED, a button, a fading light and the chip temperature, with Thonny.

By **Ada Intelisano** · informaticait.it technical team

IN SHORT

- To install MicroPython, hold down BOOTSEL and drag a .uf2 file onto the board.
- Thonny sends programs to the board and shows what they print.
- LED on GP15 with 220 Ω , button on GP16 with the internal resistor.
- The pins work at 3.3 V; the onboard LED is called "LED".

Time: one hour

Level: start here

You need: a computer with Thonny

The Raspberry Pi Pico 2 costs a few euros and you program it in Python, the language many people learn at school. Nothing to compile: write a line, press a key and the board runs it right away. In an hour you will blink an LED, read a button, dim a light and measure the chip temperature.

1 The parts

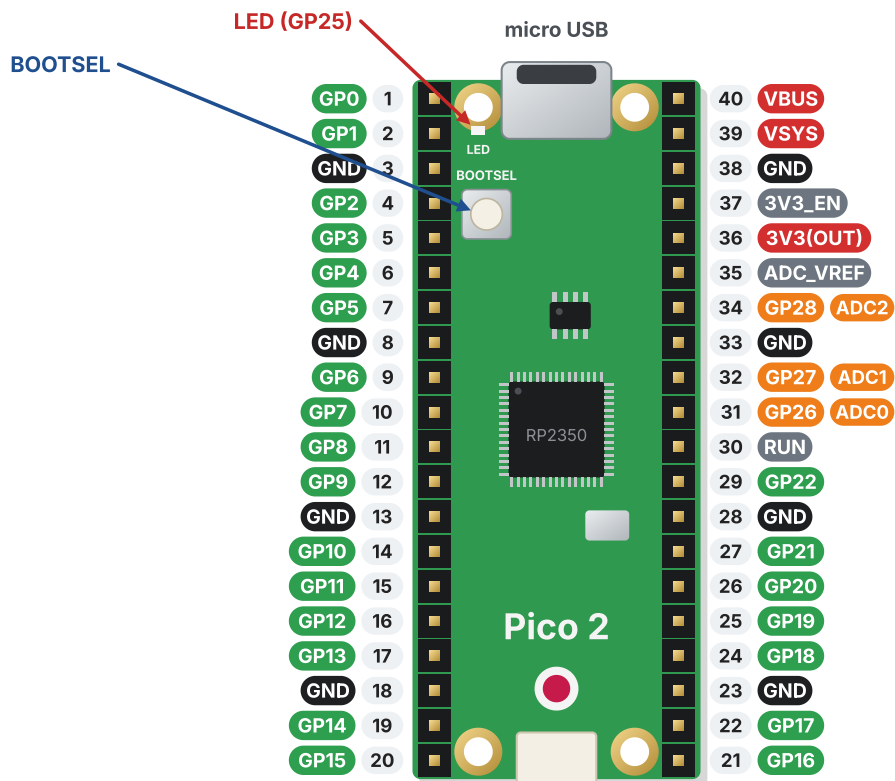
- a **Raspberry Pi Pico 2** with the pins already soldered (a Pico H works too) and a micro USB cable that also carries data;
- a breadboard, a red LED, a **220 Ω** resistor, a breadboard pushbutton and three jumper wires;
- a computer with **Thonny**, a free program for writing Python.

New word: MicroPython

A version of Python made for microcontrollers. You install it on the board just once; after that you send it programs from Thonny and it runs them line by line.

2 Meet the board

Raspberry Pi Pico 2: the 40 pins



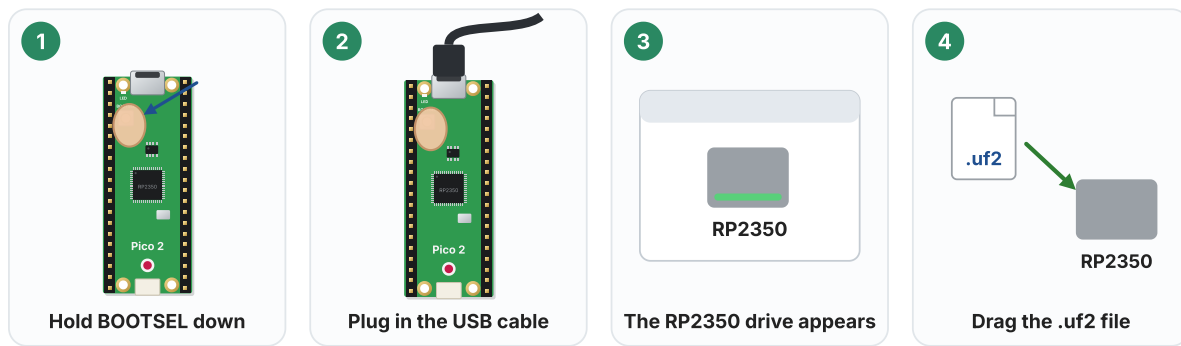
Pins work at 3.3 V. 3V3(OUT) is an output:
the data sheet recommends staying below 300 mA.

[informaticait.it](https://www.informaticait.it)

Forty pins, numbered 1 to 40: 1 to 20 down the left side from the top, 21 to 40 up the right side from the bottom. The GPIOs are called GP0, GP1... and work at **3.3 V**. Next to the USB connector you'll find the onboard **LED** and the **BOOTSEL** button, which you'll need in a moment.

3 Install MicroPython

Installing MicroPython



the board restarts by itself and now runs MicroPython (with a Pico, RPI-RP2 appears instead of RP2350)

informaticait.it

1. Download the MicroPython **.uf2** file for your model from micropython.org (Pico 2: "RPI_PICO2"; Pico: "rp2-pico").
2. Hold down BOOTSEL and plug the USB cable into the computer, then let go of the button.
3. A drive shows up, like a USB stick, called **RP2350** (on the Pico it's RPI-RP2). Drag the file onto it: the board restarts by itself.
4. In Thonny, at the bottom right, choose the interpreter "MicroPython (Raspberry Pi Pico)".

4 Your first program

Type these lines in Thonny and press the green triangle (Run).

```
# The first program: the LED on the board turns on and off every half second.
from machine import Pin
import time

led = Pin("LED", Pin.OUT)    # "LED" works on the Pico 2 and on the W models

while True:
    led.toggle()              # if it was on it turns off, and the other way around
    time.sleep(0.5)
```

✓ What you should see

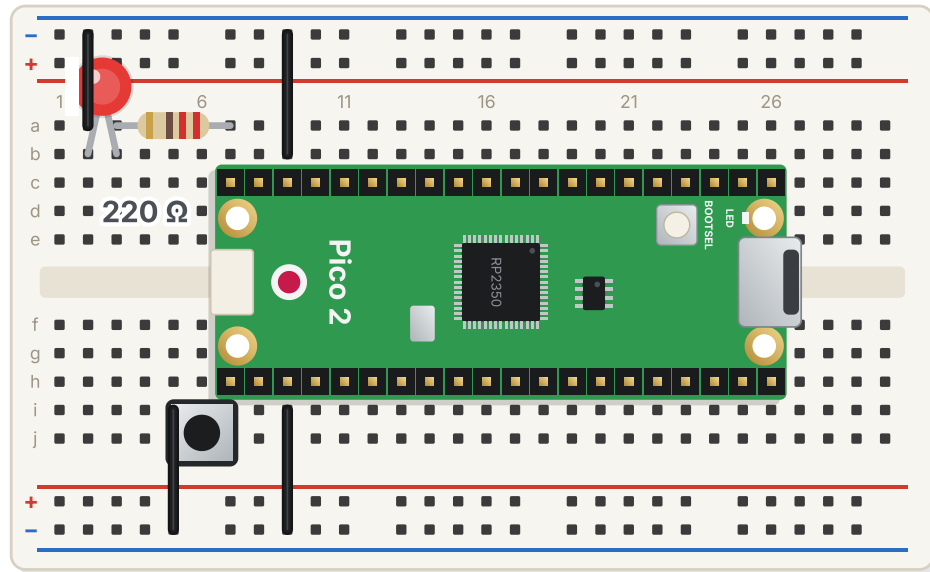
The LED next to the USB connector turns on and off every half second. Press the red button (Stop) to stop it.

New word: **main.py**

If you save the program on the board as **main.py** ("Save as" → "MicroPython device"), it starts by itself every time the board gets power, even without a computer.

5 LED and button on the breadboard

Pico 2 on the breadboard



GP15 (pin 20) → 220 Ω → LED → - rail

GP16 (pin 21) → button → - rail

GND (pins 18 and 23) → - rails

informaticait.it

- The Pico straddles the center channel, with the USB on the right.
- **LED:** from pin 20 (GP15) through the 220 Ω resistor to the long leg of the LED; the short leg goes to the - rail.
- **Button:** one leg to pin 21 (GP16), the other to the - rail. The two - rails go to the Pico GND pins (pins 18 and 23).

```
# The button turns on the red LED: when pressed, pin GP16 goes to GND and reads 0.
from machine import Pin
import time

led = Pin(15, Pin.OUT)                # GP15, physical pin 20
# GP16, physical pin 21, with the internal resistor to 3.3 V
pulsante = Pin(16, Pin.IN, Pin.PULL_UP)

while True:
    if pulsante.value() == 0:
        led.on()
    else:
        led.off()
    time.sleep_ms(10)
```

✓ What you should see

Hold the button down: the red LED turns on. Let go: it turns off.

New word: PULL_UP

A resistor inside the chip that keeps the pin at 3.3 V while the button is open. When you press it, the pin goes to GND and reads 0. That way you don't need an extra resistor on the breadboard.

6 The breathing light

PWM turns the pin on and off a thousand times a second. By changing how long it stays on, you make the LED look brighter or dimmer. In MicroPython that “how long” goes from 0 to 65535.

```
# The LED slowly fades on and off:
# PWM changes how much of the time the pin stays on.
from machine import Pin, PWM
import time

led = PWM(Pin(15))
led.freq(1000)                                # 1000 times a second: the eye only sees the average

while True:
    for livello in range(0, 65536, 512):      # from off (0) to fully on (65535)
        led.duty_u16(livello)
        time.sleep_ms(8)
    for livello in range(65535, -1, -512):
        led.duty_u16(livello)
        time.sleep_ms(8)
```

✓ What you should see

The red LED slowly brightens and then fades out, over and over.

7 The hidden thermometer

Inside the chip there is a temperature sensor, connected to channel 4 of the analog converter. The formula comes from the datasheet. It measures the chip, not the room, but touch it with a finger and you'll see the number go up.

```
# The thermometer inside the chip: converter channel 4 and the datasheet formula.
from machine import ADC
import time

sensore = ADC(4)
fattore = 3.3 / 65535                                # from number (0-65535) to volts

while True:
    volt = sensore.read_u16() * fattore
    gradi = 27 - (volt - 0.706) / 0.001721
    print("Chip temperature: %.1f °C" % gradi)
    time.sleep(1)
```

✓ What you should see

In the bottom part of Thonny a new line appears every second, for example "Chip temperature: 26.3 °C".

If it doesn't work

- Thonny doesn't see the board: the cable is charge-only, or the interpreter isn't selected.
- The external LED stays off: it's in backwards, or the resistor isn't in the same column.
- No RP2350 drive: you let go of BOOTSEL too early, try again.

8 Challenge yourself

Try it yourself

- Make the red LED blink three times every time you press the button.
- Turn on the external LED when the chip temperature goes above 30 °C.
- Build the traffic light from guide no. 39 again, with three LEDs on the Pico.

✓ What you learned

- you install MicroPython by dragging a .uf2 file;
- pins, PWM and the ADC take just a few lines of Python;
- main.py starts by itself at power-up;
- the Pico pins work at 3.3 V.

Sources: Raspberry Pi, Raspberry Pi Pico 2 datasheet · Raspberry Pi, MicroPython documentation for microcontrollers and "Pico-series Python SDK" · Raspberry Pi, RP2040 and RP2350 datasheets (temperature sensor) · MicroPython, rp2 port documentation

