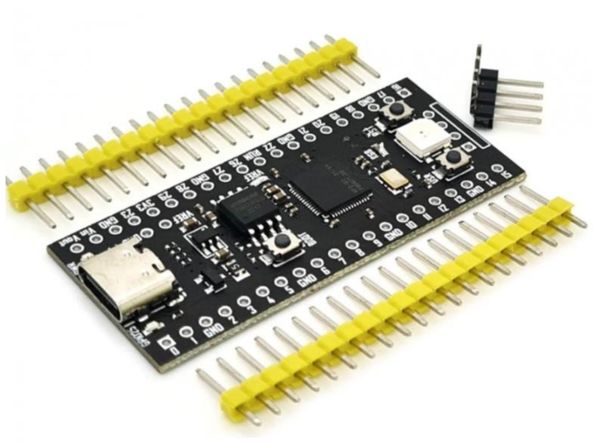
REF: B11-YD-RP2040-4MB

RPI PICO RP2040 DUAL CORE MICROCONTROLLER BOARD



Description

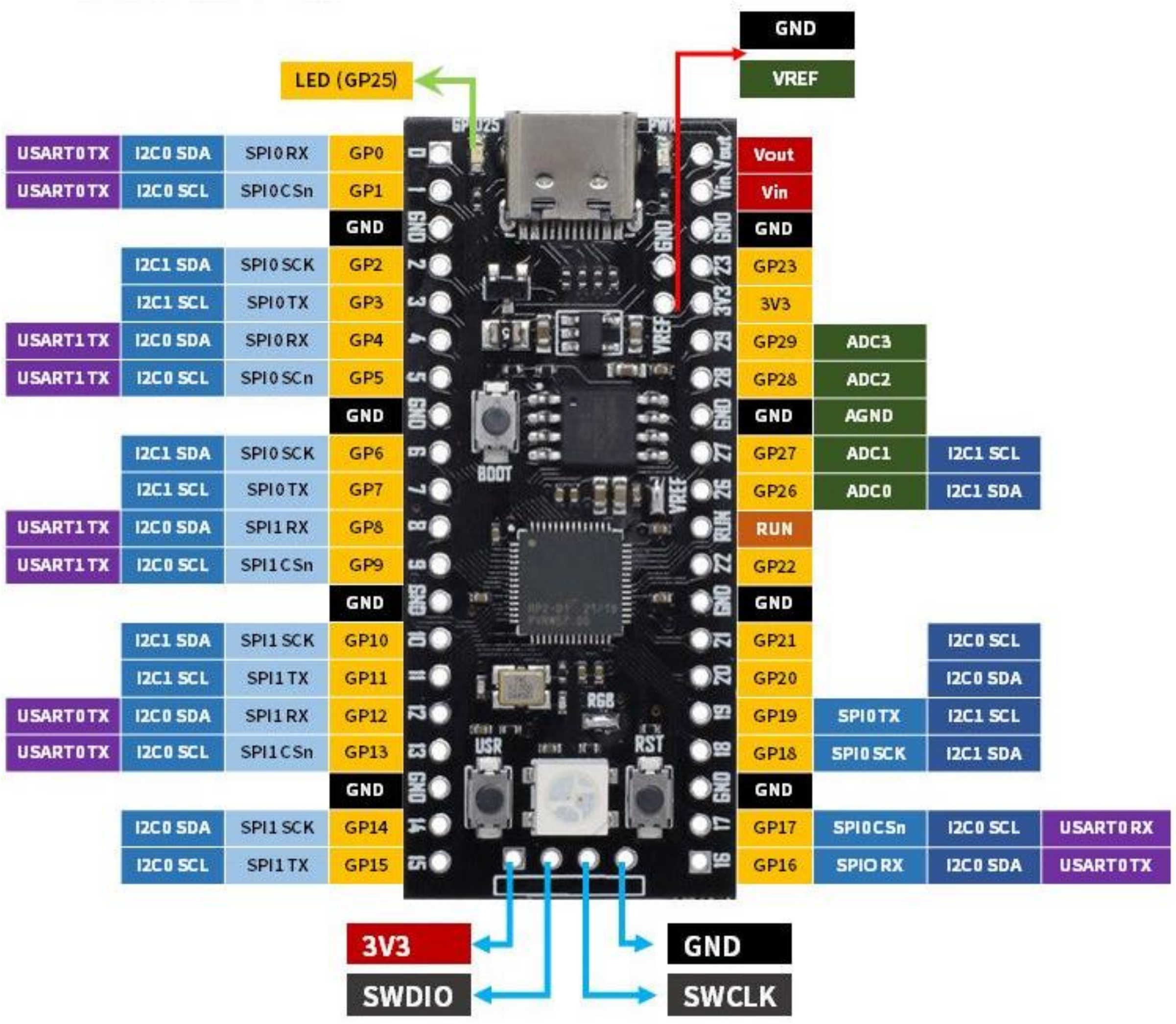
The **Raspberry Pi Pico** is a compact, high-performance **microcontroller development board** built around the **RP2040 chip**, designed by **Raspberry Pi**. Unlike the Raspberry Pi single-board computers (such as the Raspberry Pi 4 or 5), the Pico is a **microcontroller-based** board, meaning it does not run a full operating system like Linux but instead executes programs directly from flash memory. Powered by a **dual-core ARM Cortex-M0+ processor**, the RP2040 chip delivers impressive performance while maintaining **low power consumption**. The board features **26 multi-function GPIO pins**, **SPI, I2C, and UART interfaces**, **PWM support**, and a unique **Programmable I/O (PIO) subsystem**, making it ideal for embedded systems, robotics, IoT projects, and automation. Unlike some other microcontrollers like the ESP8266 or ESP32, the Raspberry Pi Pico does not have built-in Wi-Fi or Bluetooth. However, it can be connected to external communication modules to extend its capabilities.

Specifications

- Chipset: RP2040 (Designed by Raspberry Pi).
- CPU: Dual-core ARM Cortex-M0+ processor running at 133 MHz.
- SRAM: 264KB on-chip SRAM.
- Flash Storage: 2MB QSPI Flash memory for storing programs.
- Clock Management: On-chip PLL (Phase-Locked Loop) for adjusting clock speed.
- Total GPIO Pins: 26 multi-function GPIO pins.
- Analog Input: 3 ADC (Analog-to-Digital Converter) pins (12-bit resolution).
- Voltage Input: 1.8V – 5.5V (via VSYS pin).
- Power via USB: 5V supply from Micro-USB.
- Low Power Modes: Sleep and Deep Sleep modes for battery-powered applications.

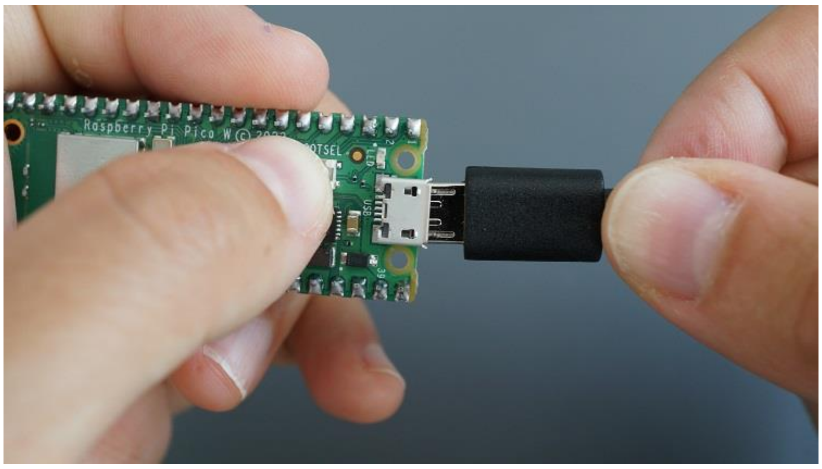
Pinout

The Arduino pins are based on the Arduino-Pico core by *Earle F. Philhower, III*. The pin assignments used by the Mbed-based RP2040 core are different and it is also somewhat limited. With the Arduino-Pico core, you can have 3 serial ports in total (including one USB-CDC and two hardware UARTs), two SPI ports, and two I2C ports. Raspberry Pi Pico supports PWM and interrupts on all GPIO pins. There are 16 PWM channels and you can assign them to any GPIO pins. The only two disadvantages of the Pico board are the lack of a reset button and a dedicated USB-Serial driver. Due to that, you have to always disconnect the board from power to reset it, and without a dedicated USB-Serial driver, most serial monitor applications will get disconnected from the board when you reset it and you won't be able to see any debug messages printed at the start of the connection.


GND
VREF
LED (GP25)
USART0 TX | I2C0 SDA | SPI0 RX | GP0 | Vout
USART0 TX | I2C0 SCL | SPI0 CSn | GP1 | Vin
GND | GND
I2C1 SDA | SPI0 SCK | GP2 | GP23
I2C1 SCL | SPI0 TX | GP3 | 3V3
USART1 TX | I2C0 SDA | SPI0 RX | GP4 | GP29 | ADC3
USART1 TX | I2C0 SCL | SPI0 SCn | GP5 | GP28 | ADC2
GND | GND | AGND
I2C1 SDA | SPI0 SCK | GP6 | GP27 | ADC1 | I2C1 SCL
I2C1 SCL | SPI0 TX | GP7 | GP26 | ADC0 | I2C1 SDA
USART1 TX | I2C0 SDA | SPI1 RX | GP8 | RUN
USART1 TX | I2C0 SCL | SPI1 CSn | GP9 | GP22
GND | GND
I2C1 SDA | SPI1 SCK | GP10 | GP21 | I2C0 SCL
I2C1 SCL | SPI1 TX | GP11 | GP20 | I2C0 SDA
USART0 TX | I2C0 SDA | SPI1 RX | GP12 | GP19 | SPI0 TX | I2C1 SCL
USART0 TX | I2C0 SCL | SPI1 CSn | GP13 | GP18 | SPI0 SCK | I2C1 SDA
GND | GND
I2C0 SDA | SPI1 SCK | GP14 | GP17 | SPI0 CSn | I2C0 SCL | USART0 RX
I2C0 SCL | SPI1 TX | GP15 | GP16 | SPI0 RX | I2C0 SDA | USART0 TX
BOOT
USR
RGB
RST
3V3
SWDIO
GND
SWCLK

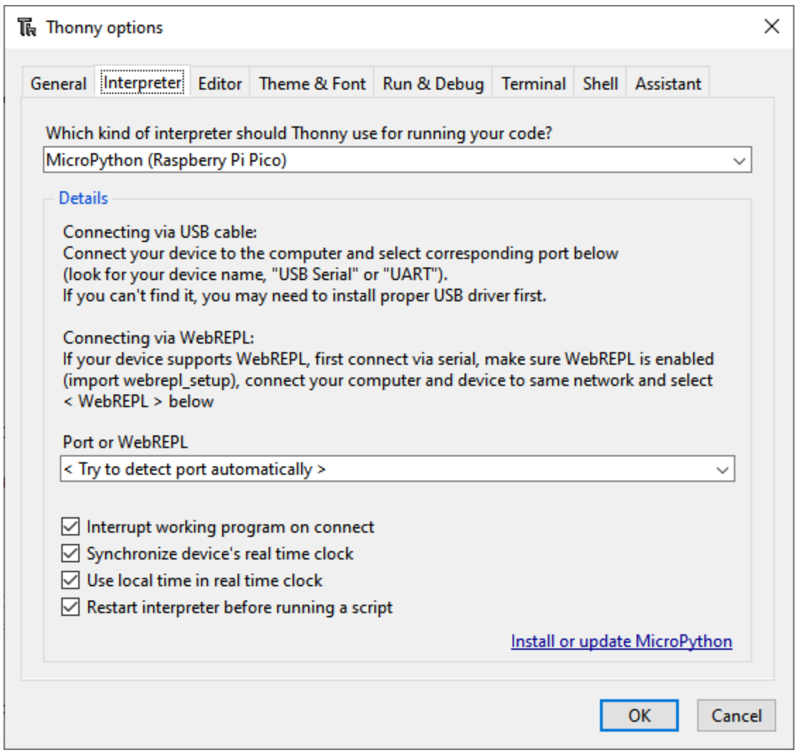
Connecting the Raspberry Pi Pico in BOOTLOADER mode

For you to be able to upload code to the Raspberry Pi Pico, it needs to be in bootloader mode. If the Raspberry Pi is currently running MicroPython firmware, you need to manually put it into bootloader mode. For that, connect the Raspberry Pi Pico to your computer while holding the BOOTSEL button at the same time. A new mass storage device window will open on your computer. You can ignore it and close that window.



Using Thonny

Open **Thonny**, go to **"Interpreter"**, and select **"MicroPython (Raspberry Pi Pico)"**. Then click **"Install or Update Micropython"** and install.



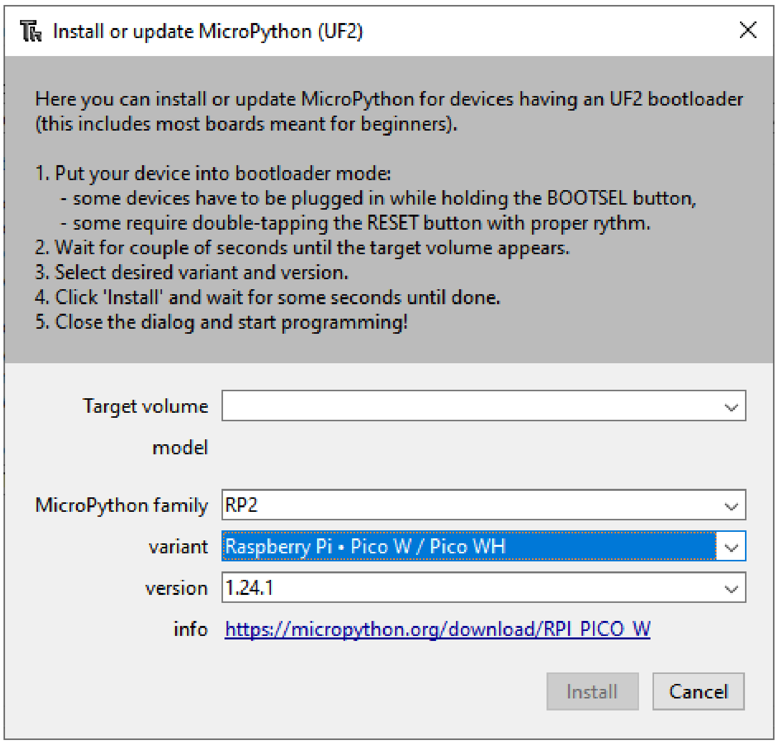
Which kind of interpreter should Thonny use for running your code?
MicroPython (Raspberry Pi Pico)

Connecting via USB cable:
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.

Connecting via WebREPL:
If your device supports WebREPL, first connect via serial, make sure WebREPL is enabled
(import webrepl_setup), connect your computer and device to same network and select
< WebREPL > below

Port or WebREPL
< Try to detect port automatically >

- [✓] Interrupt working program on connect
- [✓] Synchronize device's real time clock
- [✓] Use local time in real time clock
- [✓] Restart interpreter before running a script


Install or update MicroPython (UF2)

Here you can install or update MicroPython for devices having an UF2 bootloader (this includes most boards meant for beginners).

1. Put your device into bootloader mode:
   - some devices have to be plugged in while holding the BOOTSEL button,
   - some require double-tapping the RESET button with proper rythm.
2. Wait for couple of seconds until the target volume appears.
3. Select desired variant and version.
4. Click 'Install' and wait for some seconds until done.
5. Close the dialog and start programming!

Target volume

model

MicroPython family: RP2

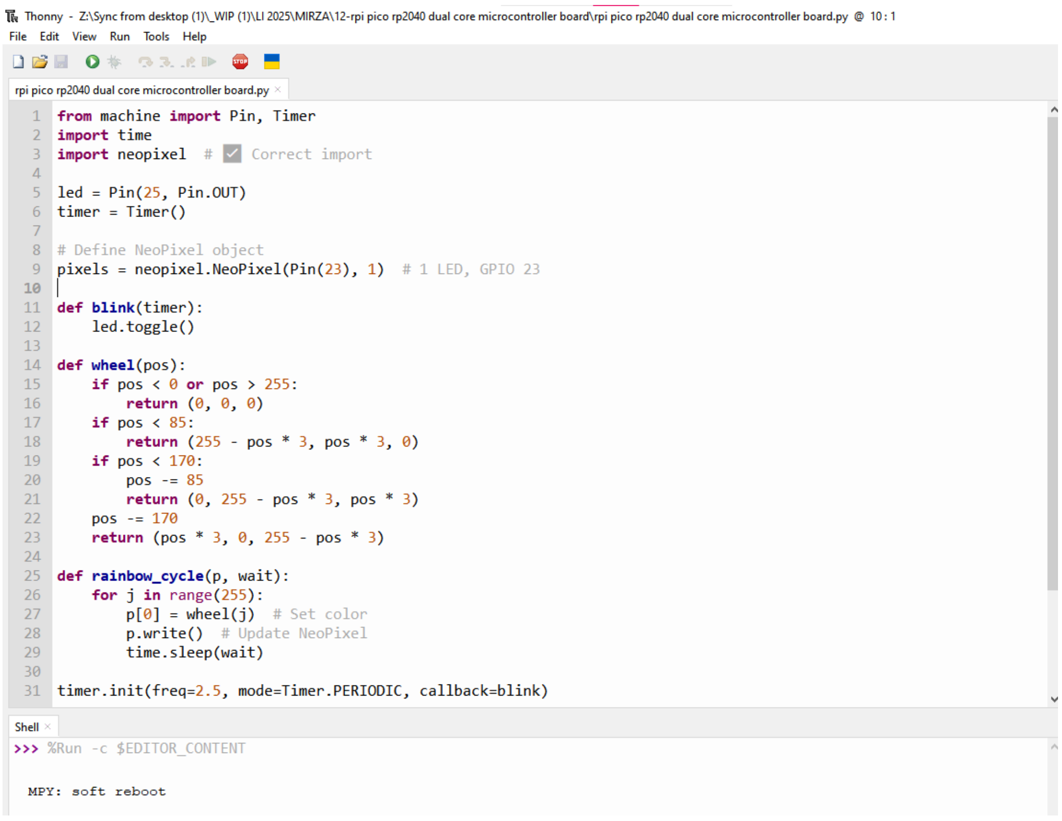
variant: Raspberry Pi • Pico W / Pico WH

version: 1.24.1

info: https://micropython.org/download/RPI_PICO_W

Coding

Click **"Run"** to execute your script.


Thonny - Z:\Sync from desktop (1)_WIP (1)\LI 2025\MIRZA\12-rpi pico rp2040 dual core microcontroller board\rpi pico rp2040 dual core microcontroller board.py @ 10 : 1

File Edit View Run Tools Help
rpi pico rp2040 dual core microcontroller board.py

```python
from machine import Pin, Timer
import time
import neopixel  # ✅ Correct import

led = Pin(25, Pin.OUT)
timer = Timer()

# Define NeoPixel object
pixels = neopixel.NeoPixel(Pin(23), 1)  # 1 LED, GPIO 23

def blink(timer):
    led.toggle()

def wheel(pos):
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)

def rainbow_cycle(p, wait):
    for j in range(255):
        p[0] = wheel(j)  # Set color
        p.write()  # Update NeoPixel
        time.sleep(wait)

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
```

Result