RP2040-Zero

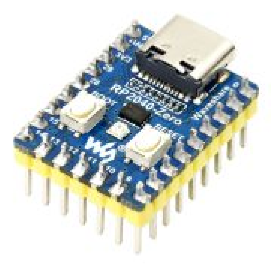
From Waveshare Wiki
Jump to: navigation, search

Introduction

The RP2040-Zero mini development board is equipped with a Type-C female port, utilizes the RP2040A developed by Raspberry Pi as the core, and leading out all unoccupied pins in a very small form factor. Using castellated processing technology, it can be welded onto your custom board

RP2040-Zero

with pre-soldered pin header

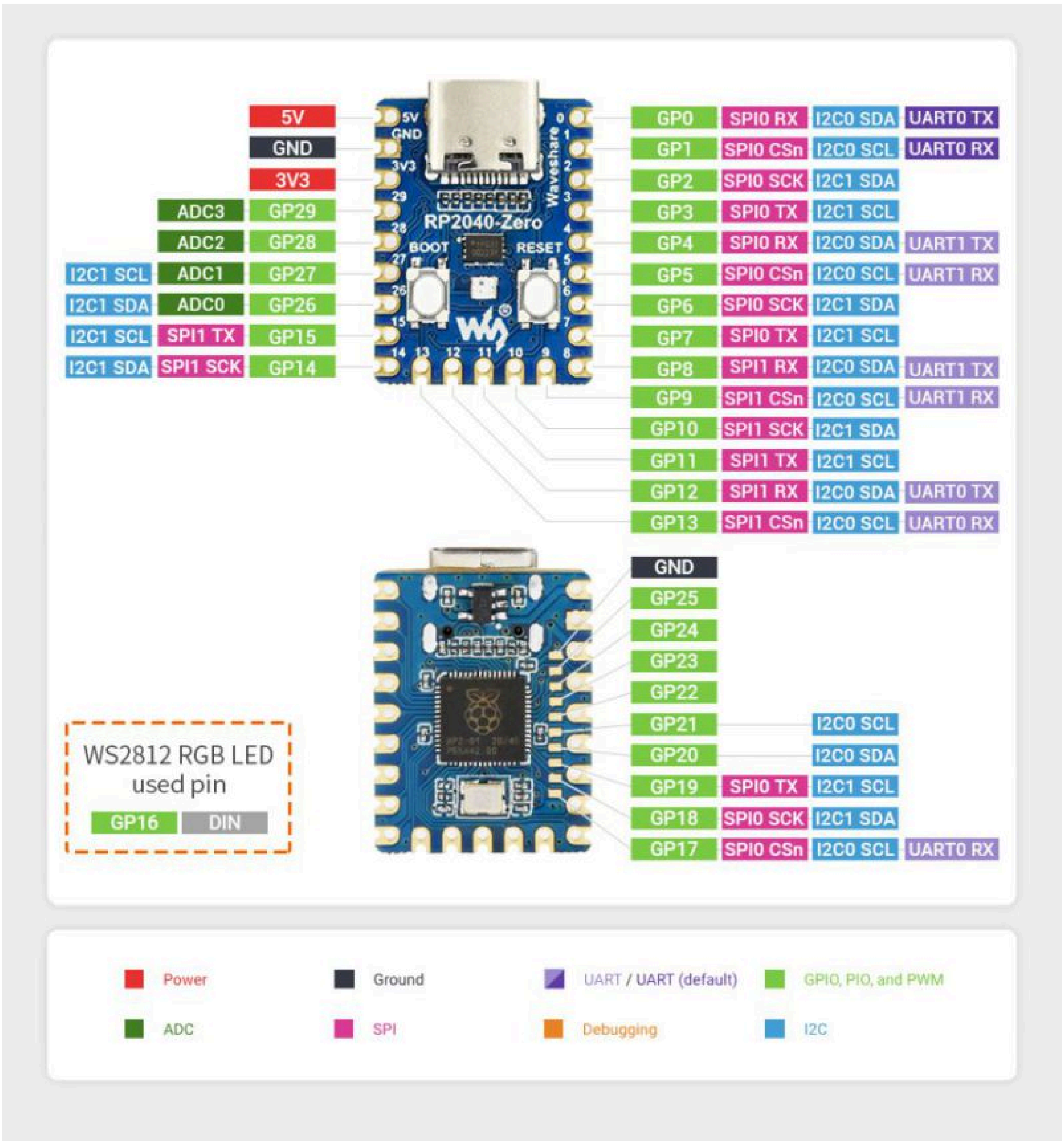


Features

- RP2040 microcontroller chip officially designed by Raspberry Pi
- Dual-core ARM Cortex M0+ processor, flexible clock running up to 133 MHz
- Built-in 264KB SRAM and 2MB Flash
- Type-C connector, keeps it up to date, easier to use
- Castellated module allows soldering directly to carrier boards
- USB1.1 host and device support
- Low-power sleep and dormant modes
- Drag-and-drop programming using mass storage over USB
- 29 GPIO pins of RP2040 (20 can be led out through pin headers, the rest can be led out only by soldering)
- Multiple hardware peripherals
  - SPI × 2
  - I2C × 2
  - UART × 2
  - 12-bit ADC × 4
  - Controllable PWM channel × 16
- Accurate clock and timer on-chip
- Temperature sensor
- On-chip accelerated floating-point library

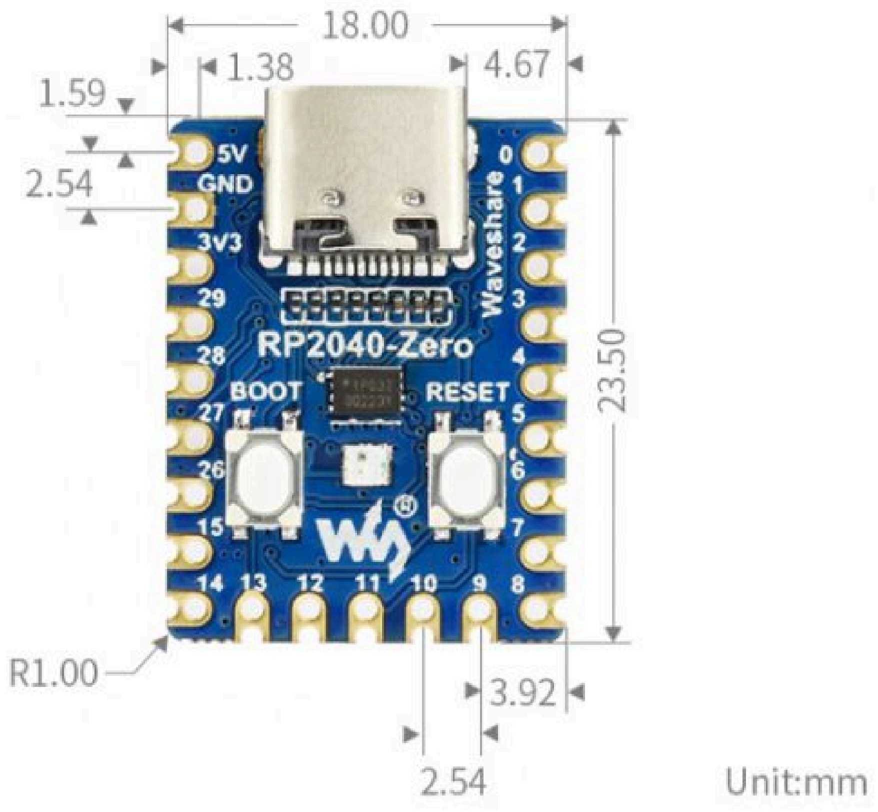
- 8 × Programmable I/O (PIO) state machines for custom peripheral support

Pinout Definition



(/wiki/File:900px-RP2040-Zero-details-7.jpg)

Dimensions



(/wiki/File:900px-RP2040-Zero-details-size.jpg)

Anti-piracy Statement

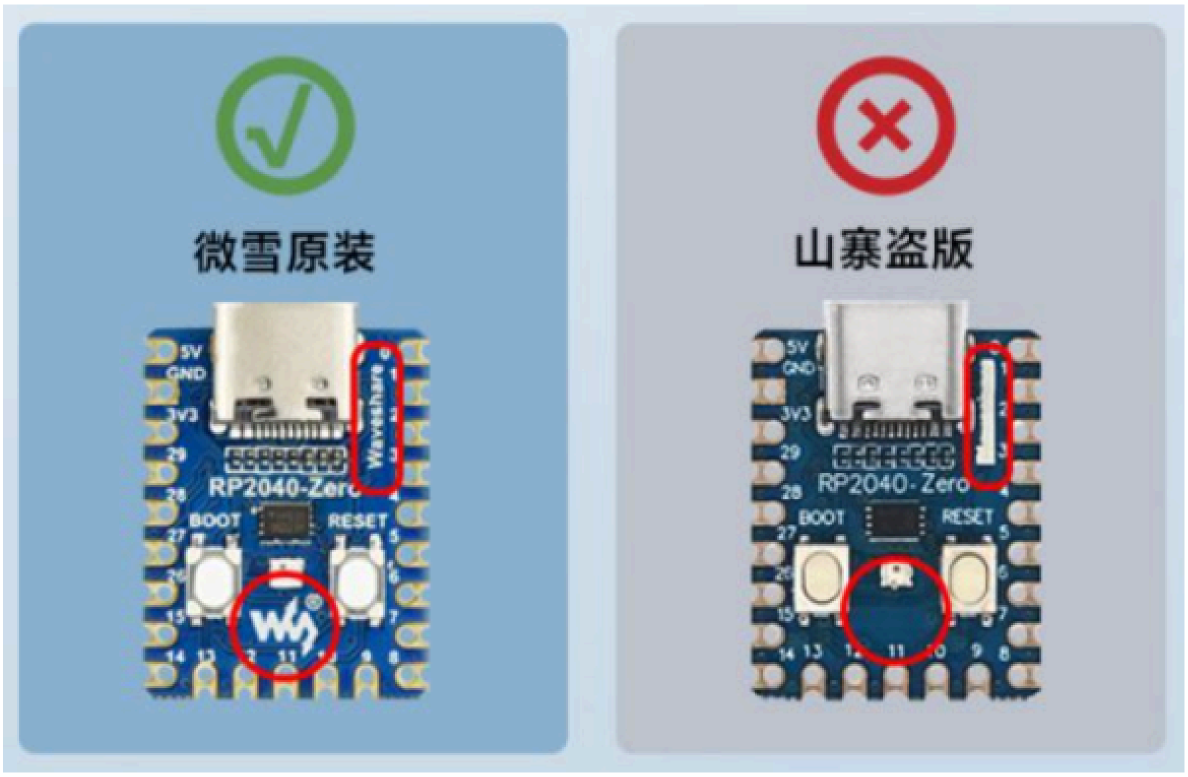
There are many unscrupulous merchants in the market who plagiarize Waveshare products. These merchants typically:

1. Copying Waveshare's webpage descriptions, product images, and product information;
2. Using low-quality components can result in unstable operation at best and, at worst, may lead to short circuits or equipment damage (to avoid property loss, please avoid counterfeit products);
3. Neglecting product quality and lacking the ability to handle after-sales service (we not only produce reliable boards but also provide a strong after-sales team to safeguard your products and creations);

For genuine Waveshare products, please recognize the following features in terms of configuration and appearance:

1. With Waveshare logo (certificate). (For counterfeit products, they generally dare not add the Waveshare trademark.)
2. Using gold immersion process (only some models, see product description) (you can see that counterfeit products have rough PCB edges and poor soldering quality).

Waveshare original	Counterfeit product



(/wiki/File:RP2040-%E6%89%93%E5%87%BB%E7%9B%97%E7%89%88%E5%A3%B0%E6%98%8E-750.png)

Pico Getting Started

Firmware Download

- MicroPython Firmware Download [Expand]
- C_Blink Firmware Download [Expand]

Basic Introduction

Raspberry Pi Pico Basics ()

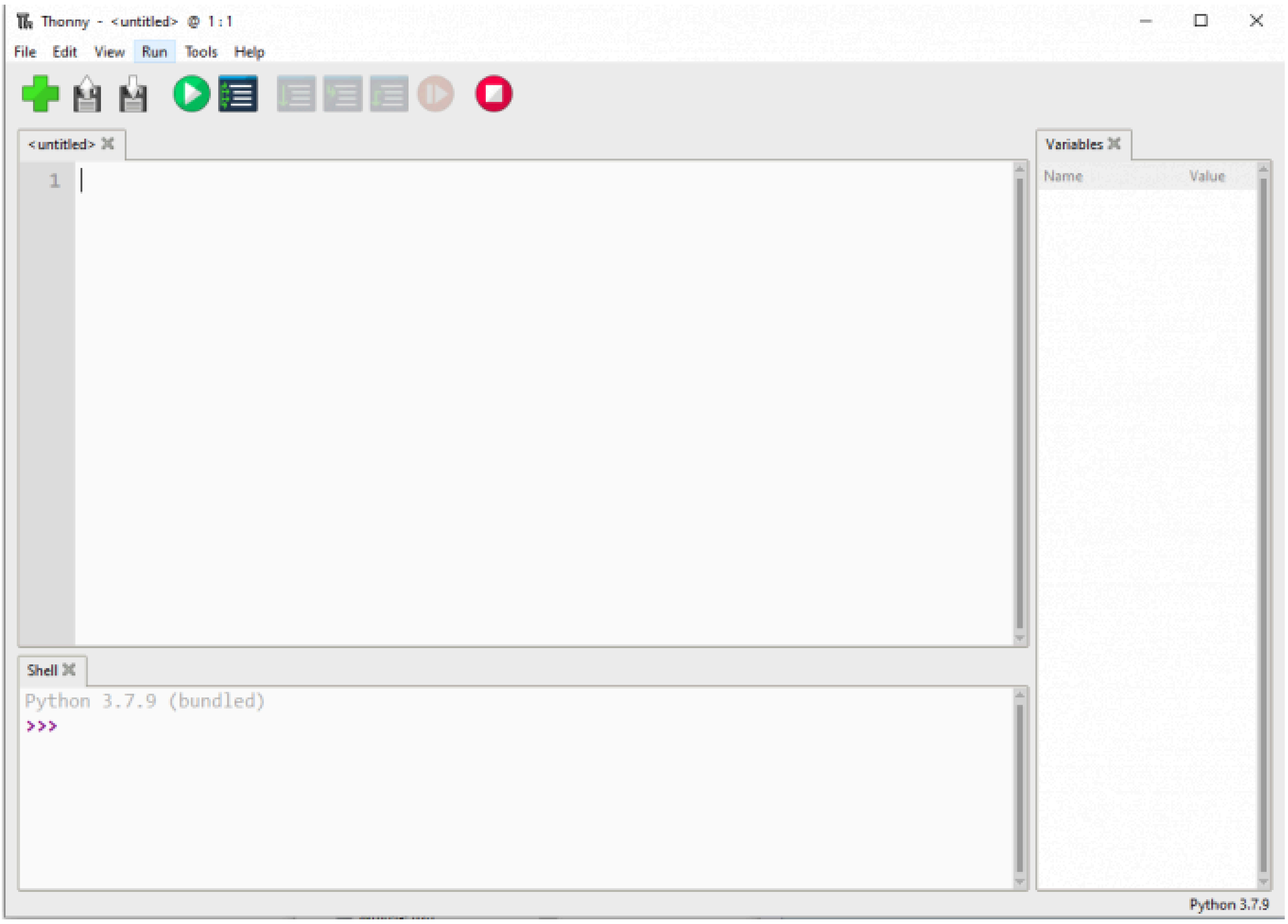
MicroPython Series

Install Thonny IDE

In order to facilitate the development of Pico/Pico2 boards using MicroPython on a computer, it is recommended to download the Thonny IDE

- Download Thonny IDE and follow the steps to install, the installation packages are all Windows versions, please refer to Thonny's official website for other versions
  - Thonny IDE official download link (https://github.com/thonny/thonny/releases/download/v3.3.3/thonny-3.3.3.exe)
  - Thonny IDE download link (https://files.waveshare.com/wiki/common/Thonny-3.3.3.zip)
  - Thonny official website (https://thonny.org/)
- After installation, the language and motherboard environment need to be configured for the first use. Since we are using Pico/Pico2, pay attention to selecting the Raspberry Pi option for

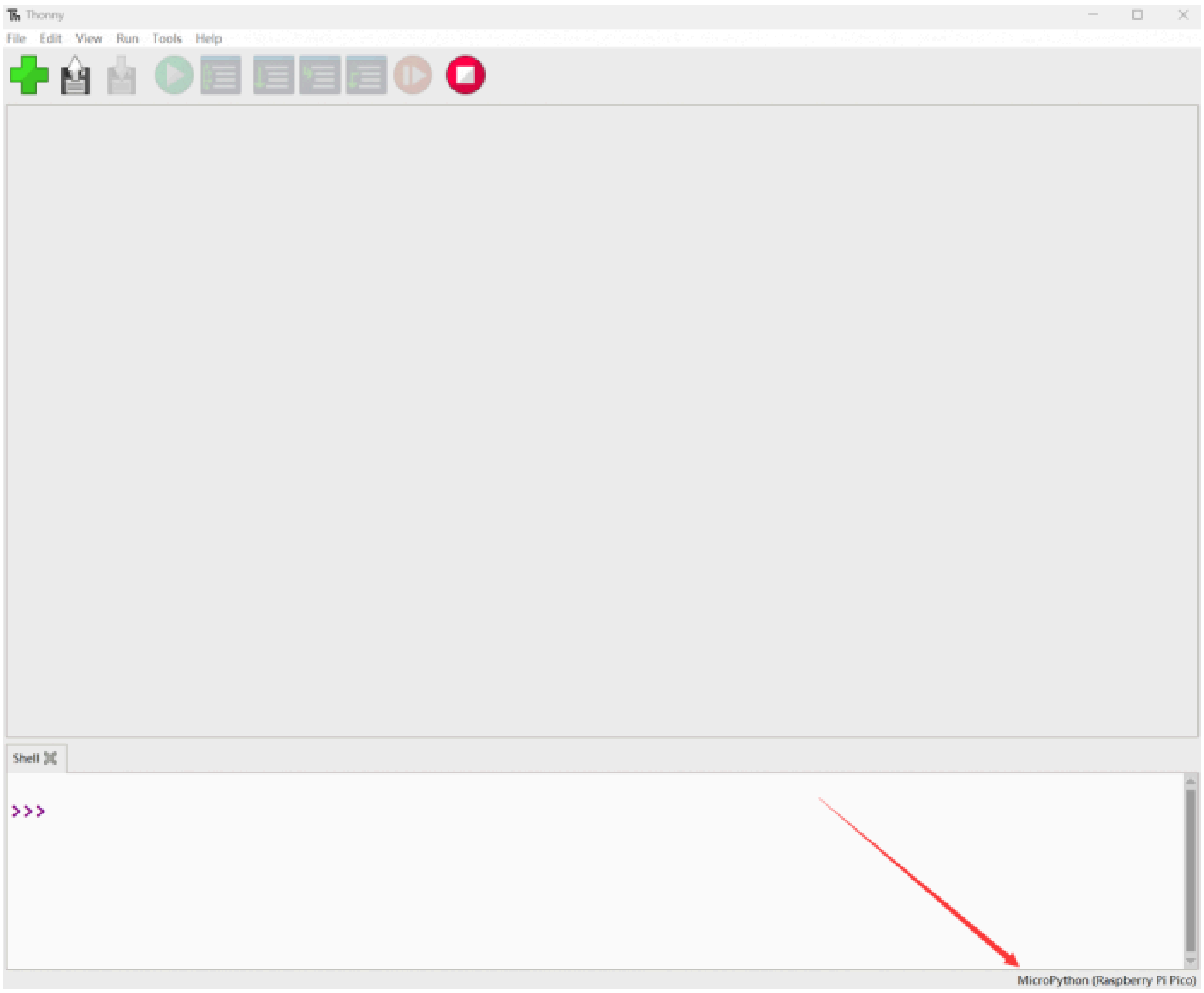
the motherboard environment



```
Python 3.7.9 (bundled)
>>>
```

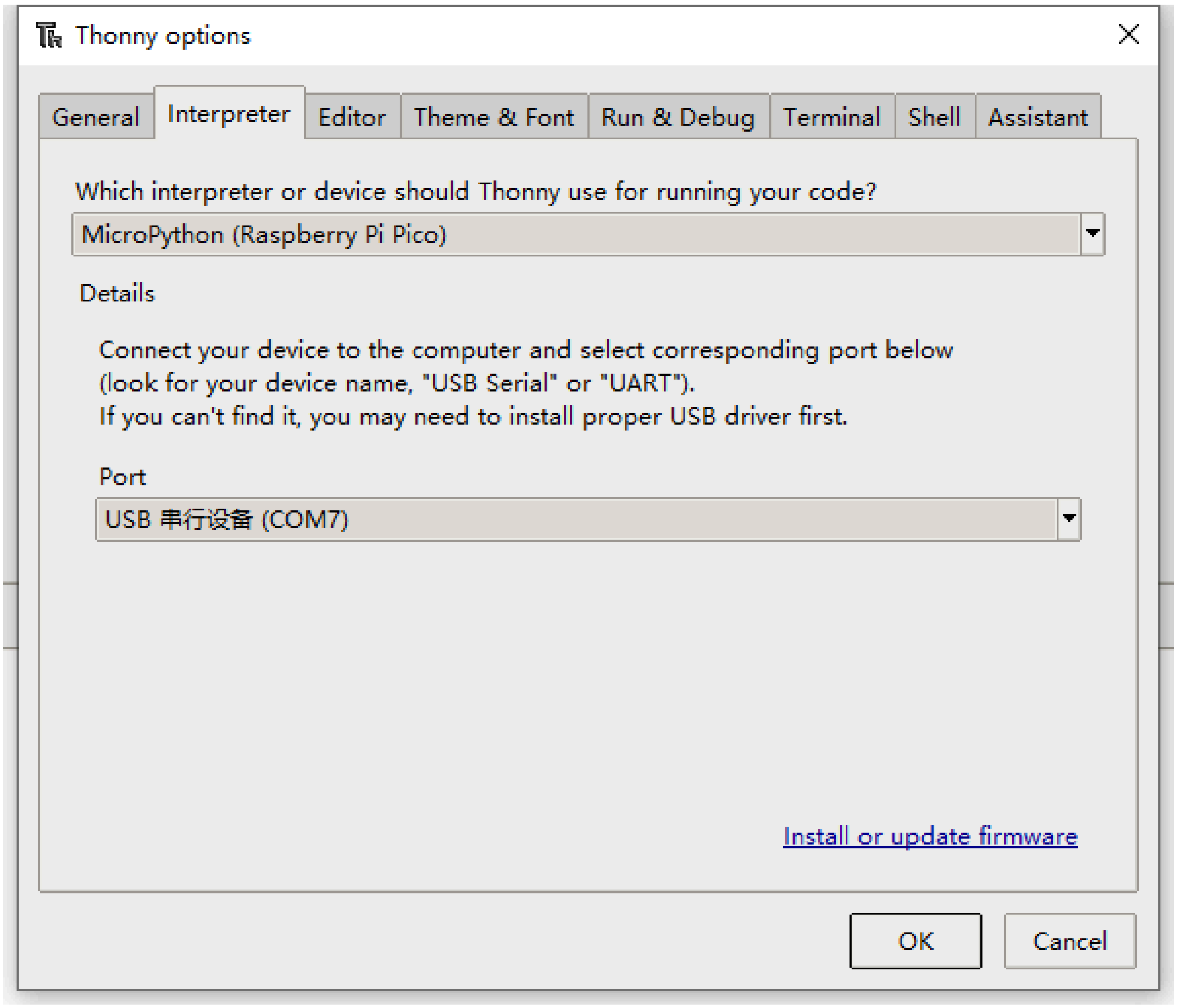
(/wiki/File:Pico-R3-Tonny1.png)

- Configure MicroPython environment and choose Pico/Pico2 port
  - Connect Pico/Pico2 to your computer first, and in the lower right corner of Thonny left-click on the configuration environment option --> select Configture interpreter
  - In the pop-up window, select MicroPython (Raspberry Pi Pico), and choose the corresponding port


Thonny
File Edit View Run Tools Help
Shell
MicroPython (Raspberry Pi Pico)

```
>>>
```

(/wiki/File:1050px-Raspberry-Pi-Pico-Basic-Kit-M-2.png)



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.

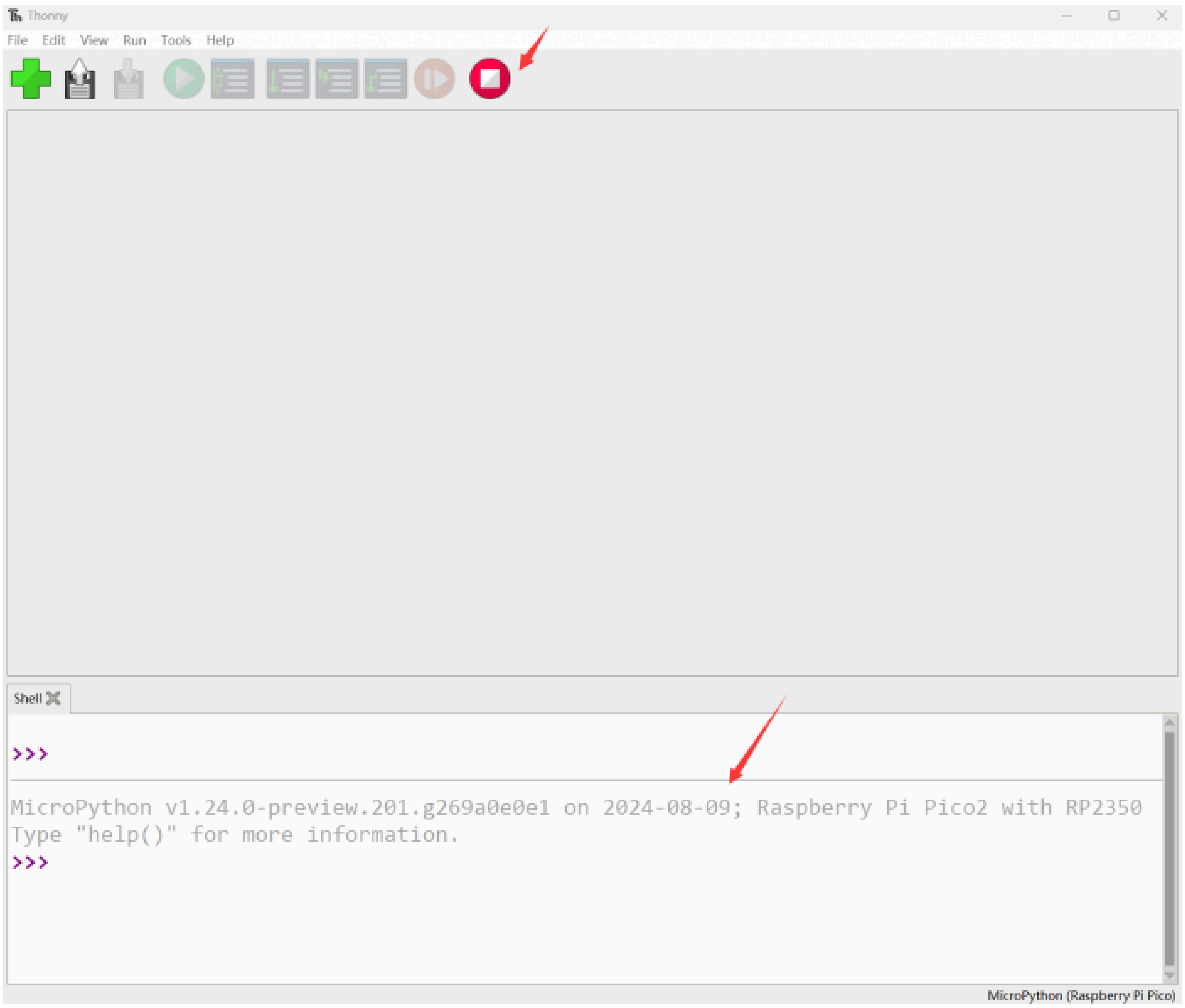
Port

USB 串行设备 (COM7)

(/wiki/File:Raspberry-Pi-Pico-Basic-Kit-M-3.png)

Flash Firmware

- Click OK to return to the Thonny main interface, download the corresponding firmware library and burn it to the device, and then click the Stop button to display the current environment in the Shell window
- **Note: Flashing the Pico2 firmware provided by Micropython may cause the device to be unrecognized, please use the firmware below or in the package**
  - Pico firmware library (https://files.waveshare.com/wiki/Raspberry-Pi-Pico-2/RPI_PICO-20250415-v1.25.0.zip)
  - Pico2 firmware library (https://files.waveshare.com/wiki/common/WAVESHARE-RP2350-20250711-v1.26.0.zip)
- How to download the firmware library for Pico/Pico2 in windows: After holding down the BOOT button and connecting to the computer, release the BOOT button, a removable disk will appear on the computer, copy the firmware library into it
- How to download the firmware library for RP2040/RP2350 in windows: After connecting to the computer, press the BOOT key and the RESET key at the same time, release the RESET key first and then release the BOOT key, a removable disk will appear on the computer, copy the firmware library into it (you can also use the Pico/Pico2 method)


Thonny
File Edit View Run Tools Help
Shell
MicroPython (Raspberry Pi Pico)

```
>>>
MicroPython v1.24.0-preview.201.g269a0e0e1 on 2024-08-09; Raspberry Pi Pico2 with RP2350
Type "help()" for more information.
>>>
```

(/wiki/File:Raspberry-Pi-Pico2-Python.png)

MicroPython Series Tutorials

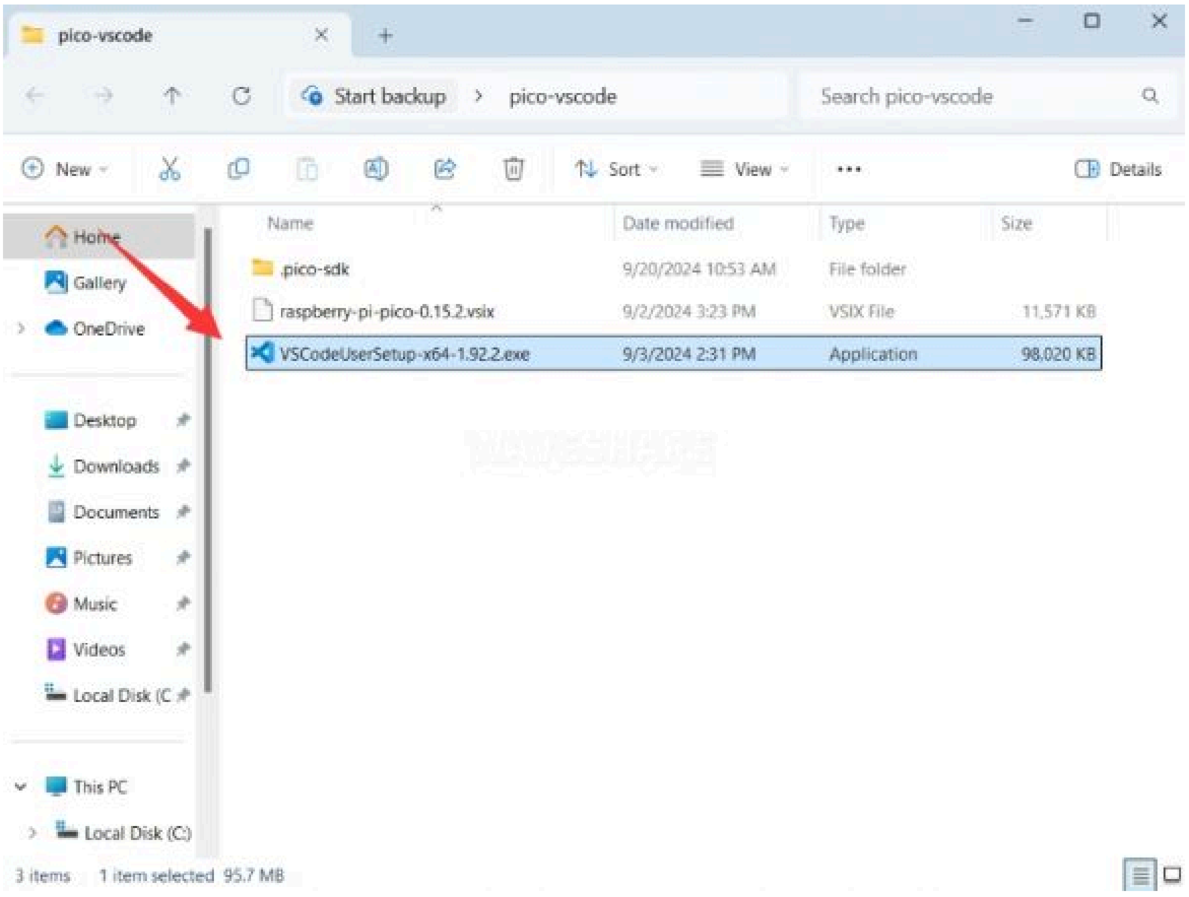
【MicroPython】 machine.Pin class function details ()
【MicroPython】machine.PWM class function details ()
【MicroPython】machine.ADC class function details ()
【MicroPython】machine.UART class function details ()
【MicroPython】machine.I2C class function details ()
【MicroPython】machine.SPI class function details ()
【MicroPython】rp2.StateMachine class function details ()

C/C++ Series

For C/C++, it is recommended to use Pico VSCode for development. This is a Microsoft Visual Studio Code extension designed to make it easier for you to create, develop, and debug projects for the Raspberry Pi Pico series development boards. No matter if you are a beginner or an experienced professional, this tool can assist you in developing Pico with confidence and ease. Here's how to install and use the extension.

- Official website tutorial: https://www.raspberrypi.com/news/pico-vscode-extension/ (https://www.raspberrypi.com/news/pico-vscode-extension/)
- This tutorial is suitable for Raspberry Pi Pico, Pico2 and the RP2040 and RP2350 series development boards developed by Waveshare
- The development environment defaults to Windows11. For other environments, please refer to the official tutorial for installation

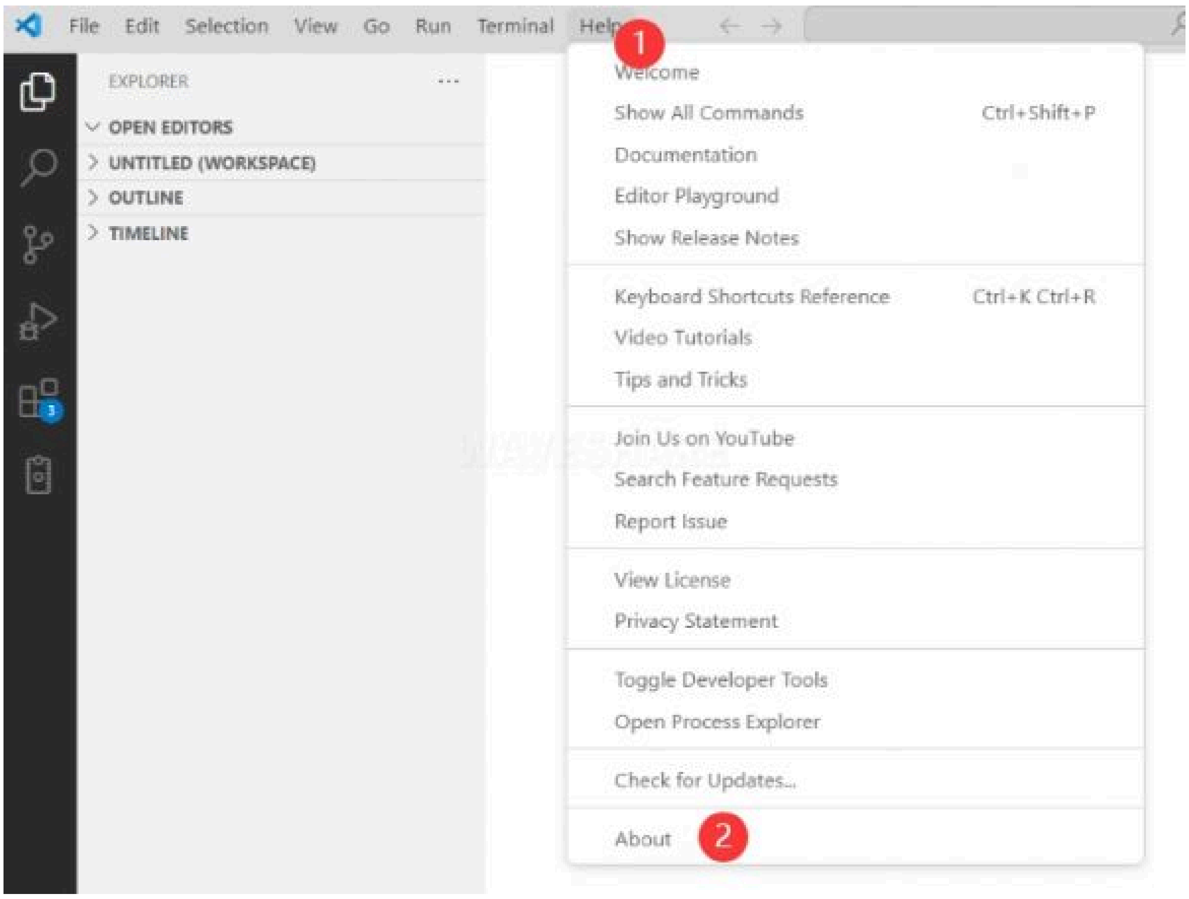
Install VSCode

1. First, click to download pico-vscode package (https://drive.google.com/file/d/18-KDNrQlI0KuTMdS6W5iblUGaGm3FbVJ/view?usp=sharing), unzip and open the package, double-click to install VSCode


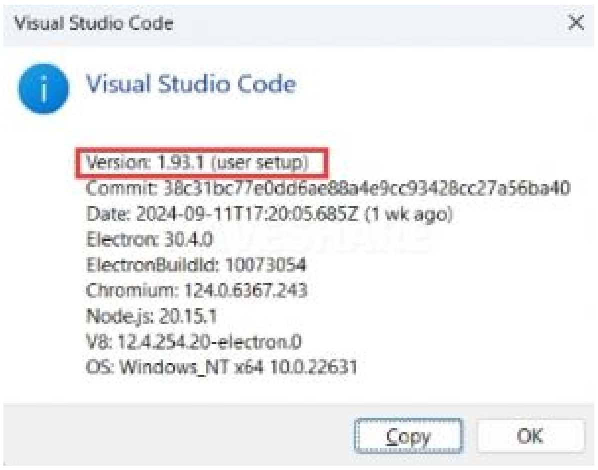
Name	Date modified	Type	Size
.pico-sdk	9/20/2024 10:53 AM	File folder	
raspberry-pi-pico-0.15.2.vsix	9/2/2024 3:23 PM	VSIX File	11,571 KB
VSCodeUserSetup-x64-1.92.2.exe	9/3/2024 2:31 PM	Application	98,020 KB

3 items 1 item selected 95.7 MB

(/wiki/File:Pico-vscode-1.JPG)
Note: If vscode is installed, check if the version is v1.87.0 or later



(/wiki/File:Pico-vscode-2.JPG)


Visual Studio Code

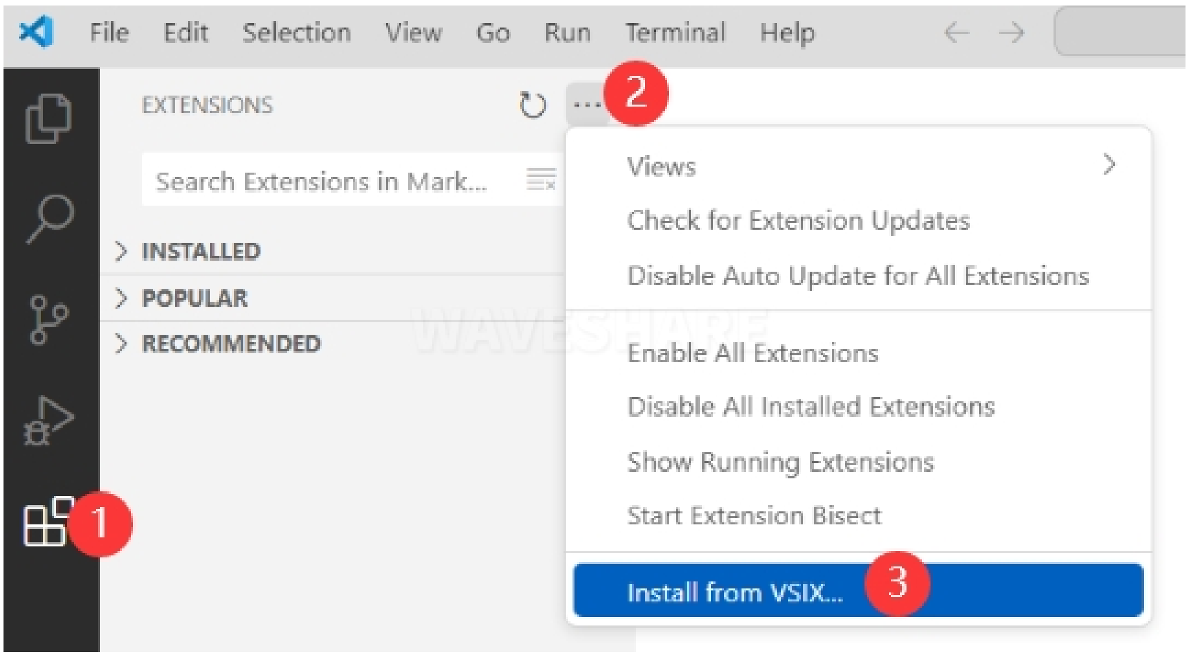
Visual Studio Code

```
Version: 1.93.1 (user setup)
Commit: 38c31bc77e0dd6ae88a4e9cc93428cc27a56ba40
Date: 2024-09-11T17:20:05.685Z (1 wk ago)
Electron: 30.4.0
ElectronBuildId: 10073054
Chromium: 124.0.6367.243
Node.js: 20.15.1
V8: 12.4.254.20-electron.0
OS: Windows_NT x64 10.0.22631
```

(/wiki/File:Pico-vscode-3.JPG)

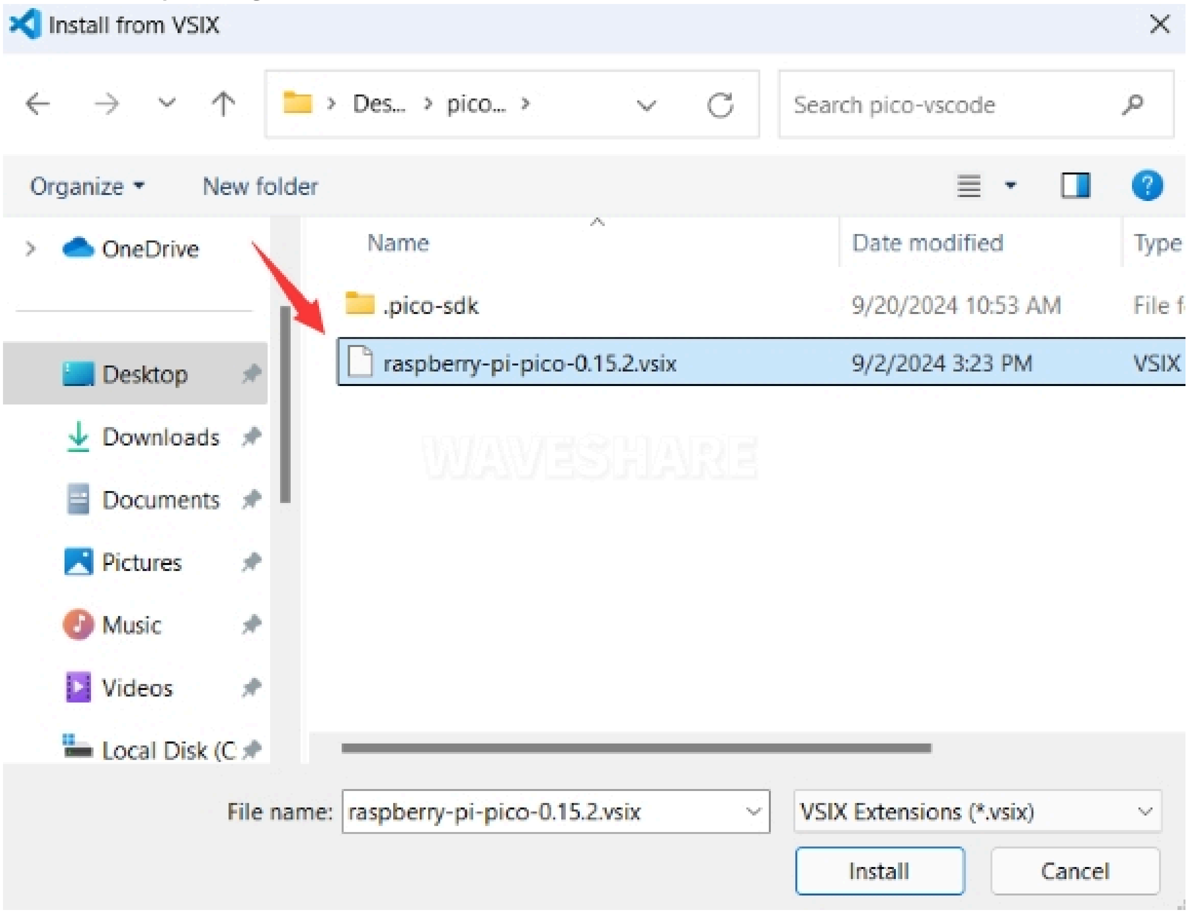
Install Extension

1. Click Extensions and select Install from VSIX



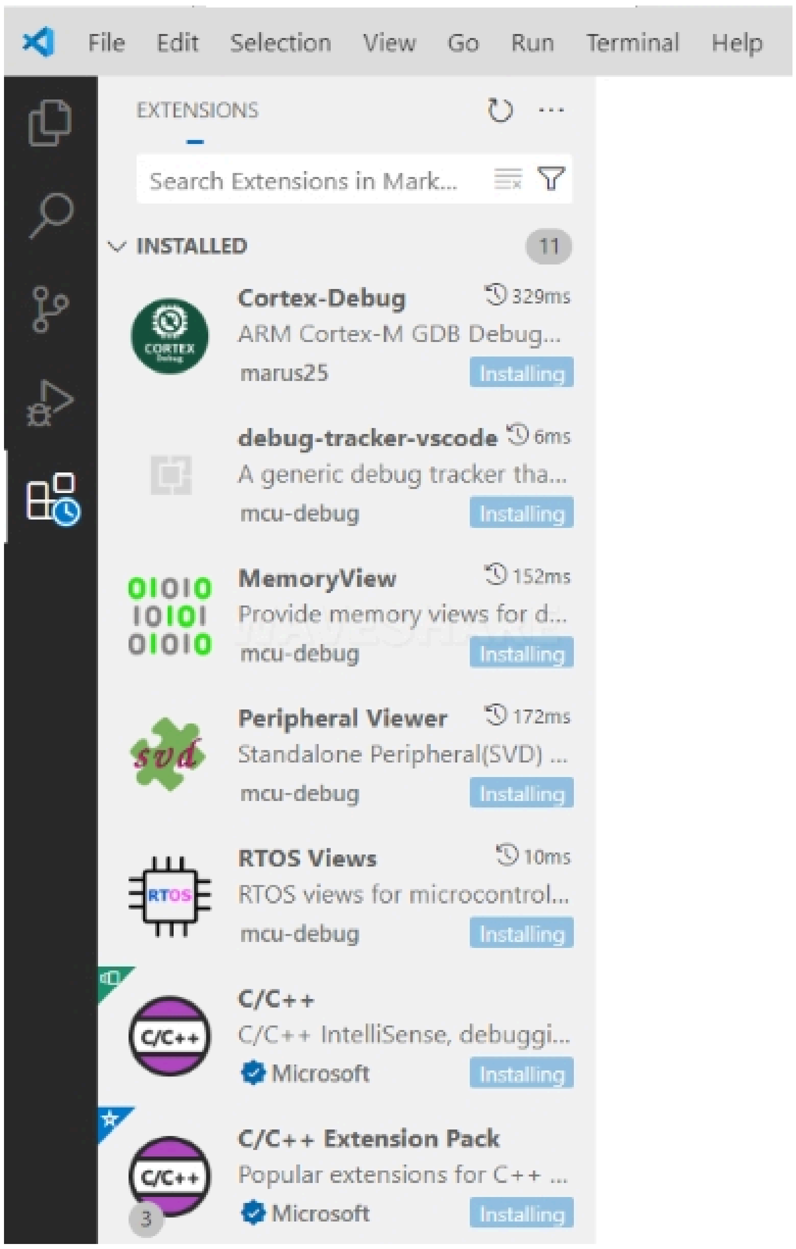
(/wiki/File:Pico-vscode-4.JPG)

2. Select the package with the vsix suffix and click Install



(/wiki/File:Pico-vscode-5.JPG)

3. Then vscode will automatically install raspberry-pi-pico and its dependency extensions, you can click Refresh to check the installation progress



Cortex-Debug 329ms
ARM Cortex-M GDB Debug...
marus25 Installing

debug-tracker-vscode 6ms
A generic debug tracker tha...
mcu-debug Installing

MemoryView 152ms
Provide memory views for d...
mcu-debug Installing

Peripheral Viewer 172ms
Standalone Peripheral(SVD) ...
mcu-debug Installing

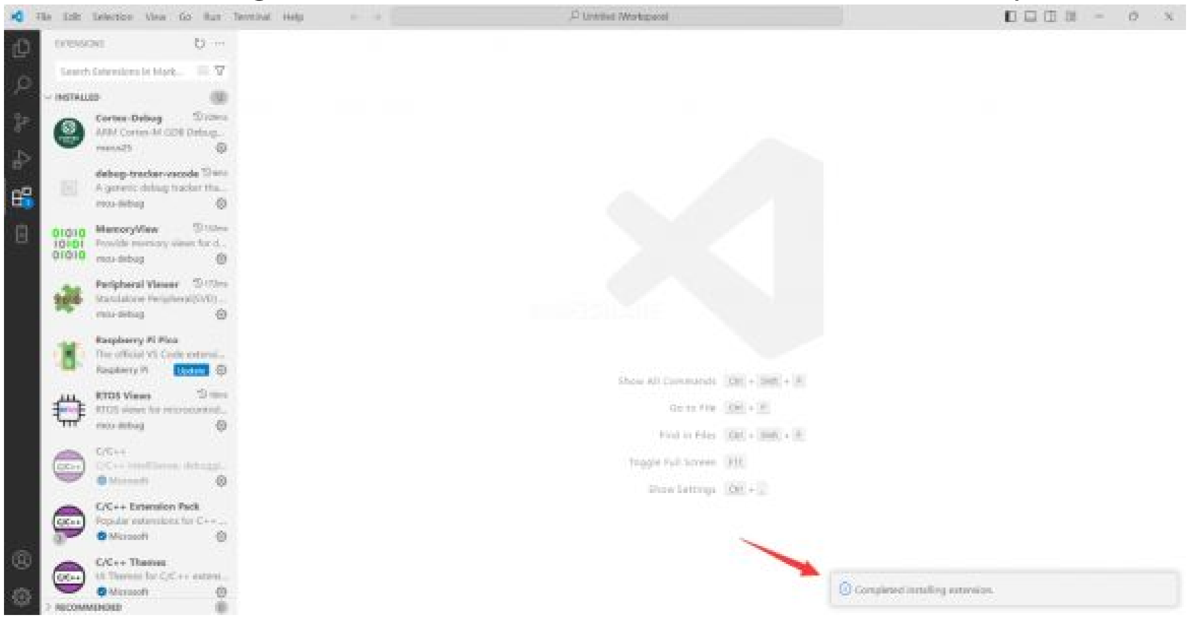
RTOS Views 10ms
RTOS views for microcontrol...
mcu-debug Installing

C/C++
C/C++ IntelliSense, debuggi...
Microsoft Installing

C/C++ Extension Pack
Popular extensions for C++ ...
Microsoft Installing

(/wiki/File:Pico-vscode-6.JPG)

4. The text in the right lower corner shows that the installation is complete. Close VSCode

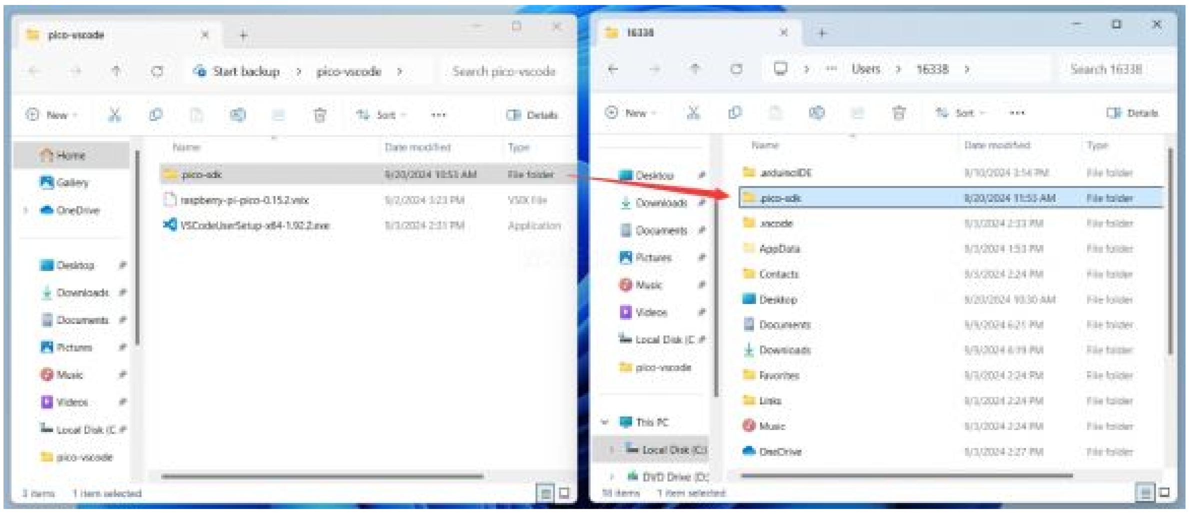


Completed installing extension.

(/wiki/File:Pico-vscode-7.JPG)

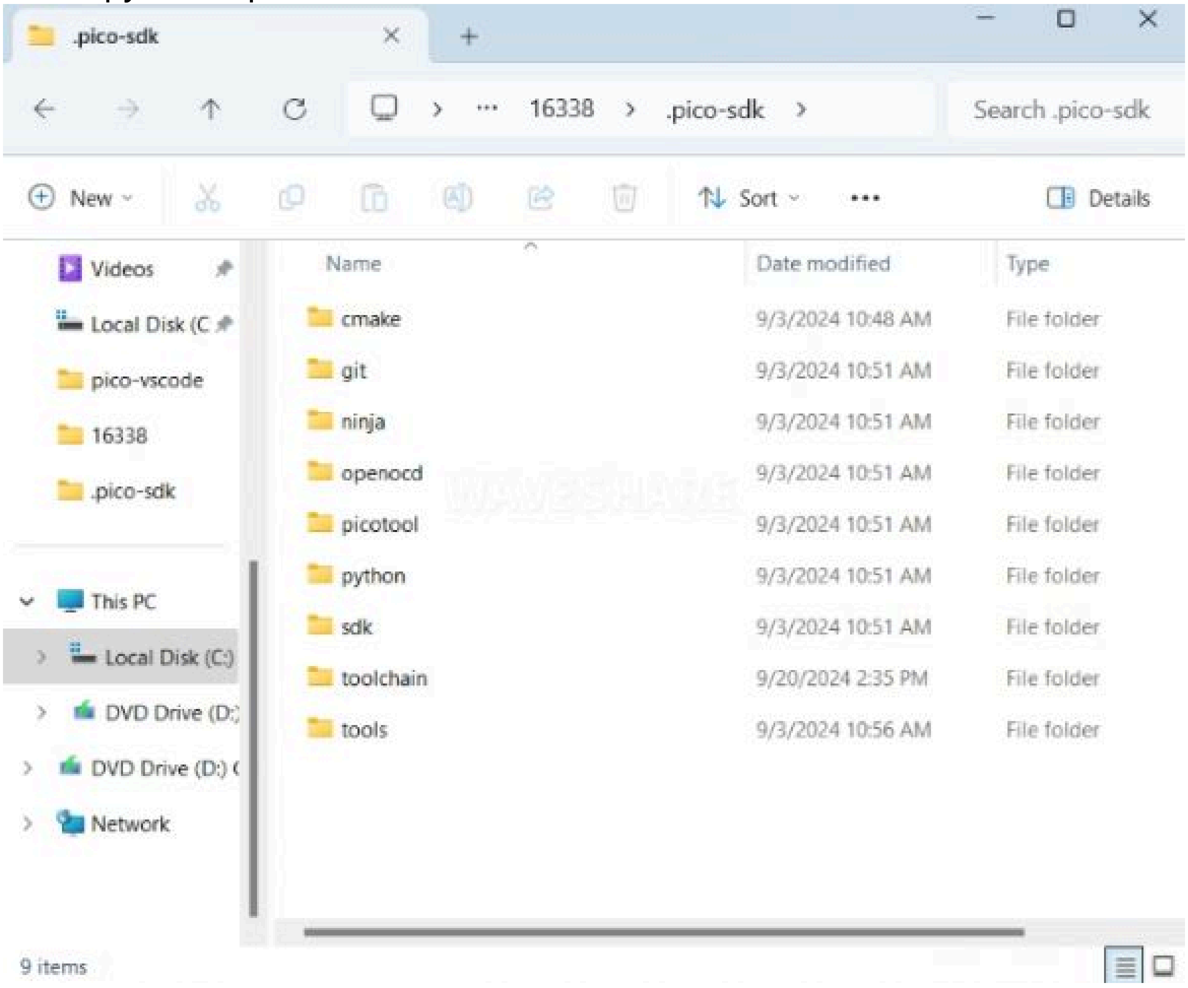
Configure Extension

1. Open directory C:\Users\username and copy the entire .pico-sdk to that directory



(/wiki/File:Pico-vscode-8.JPG)

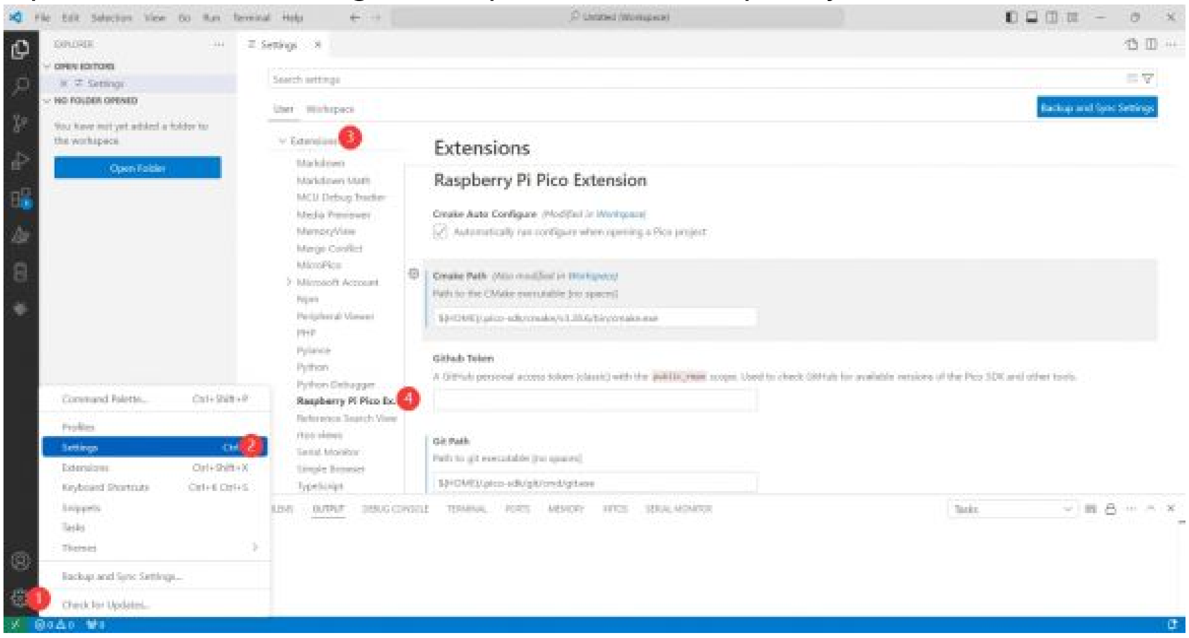
2. The copy is completed



Name	Date modified	Type
cmake	9/3/2024 10:48 AM	File folder
git	9/3/2024 10:51 AM	File folder
ninja	9/3/2024 10:51 AM	File folder
openocd	9/3/2024 10:51 AM	File folder
picotool	9/3/2024 10:51 AM	File folder
python	9/3/2024 10:51 AM	File folder
sdk	9/3/2024 10:51 AM	File folder
toolchain	9/20/2024 2:35 PM	File folder
tools	9/3/2024 10:56 AM	File folder

(/wiki/File:Pico-vscode-9.JPG)

3. Open vscode and configure the paths for the Raspberry Pi Pico extensions



Extensions

Raspberry Pi Pico Extension

Cmake Auto Configure (Modified in Workspace)

☑ Automatically run configure when opening a Pico project

Cmake Path (Also modified in Workspace)

Path to the CMake executable (no spaces)

${HOME}/.pico-sdk/cmake/v3.28.6/bin/cmake.exe

Github Token

A GitHub personal access token (classic) with the public_repo scope. Used to check GitHub for available versions of the Pico SDK and other tools.

Git Path

Path to git executable (no spaces)

${HOME}/.pico-sdk/git/cmd/git.exe

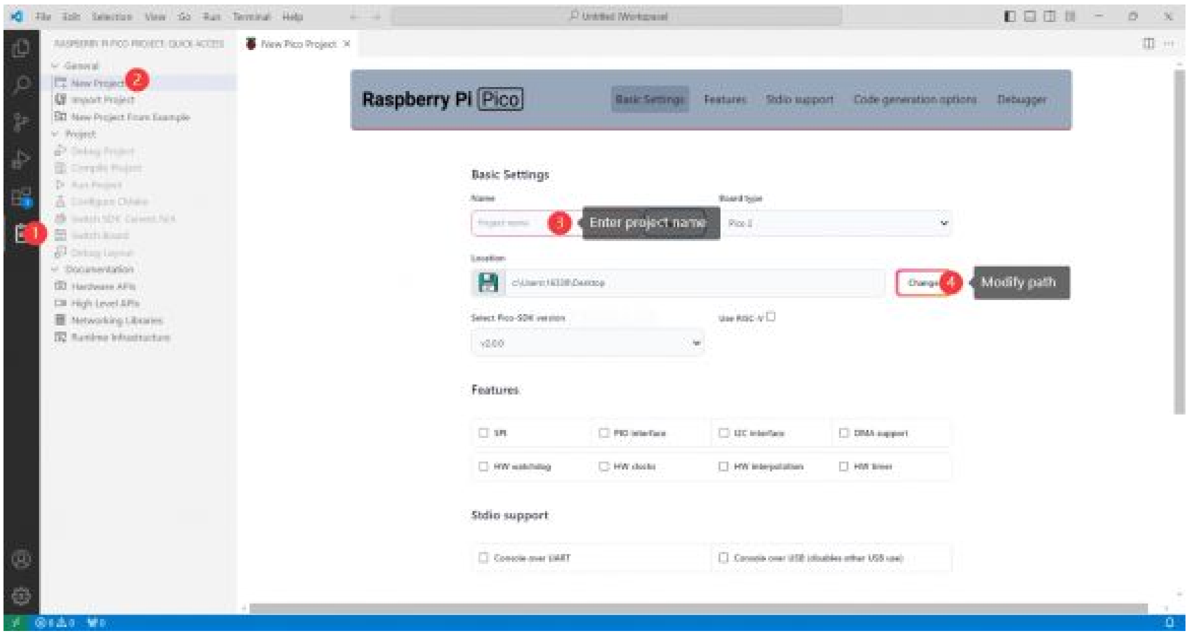
(/wiki/File:Pico-vscode-10.JPG)

The configuration is as follows:

```
Cmake Path:
${HOME}/.pico-sdk/cmake/v3.28.6/bin/cmake.exe

Git Path:
${HOME}/.pico-sdk/git/cmd/git.exe

Ninja Path:
${HOME}/.pico-sdk/ninja/v1.12.1/ninja.exe

Python3 Path:
${HOME}/.pico-sdk/python/3.12.1/python.exe
```

New Project

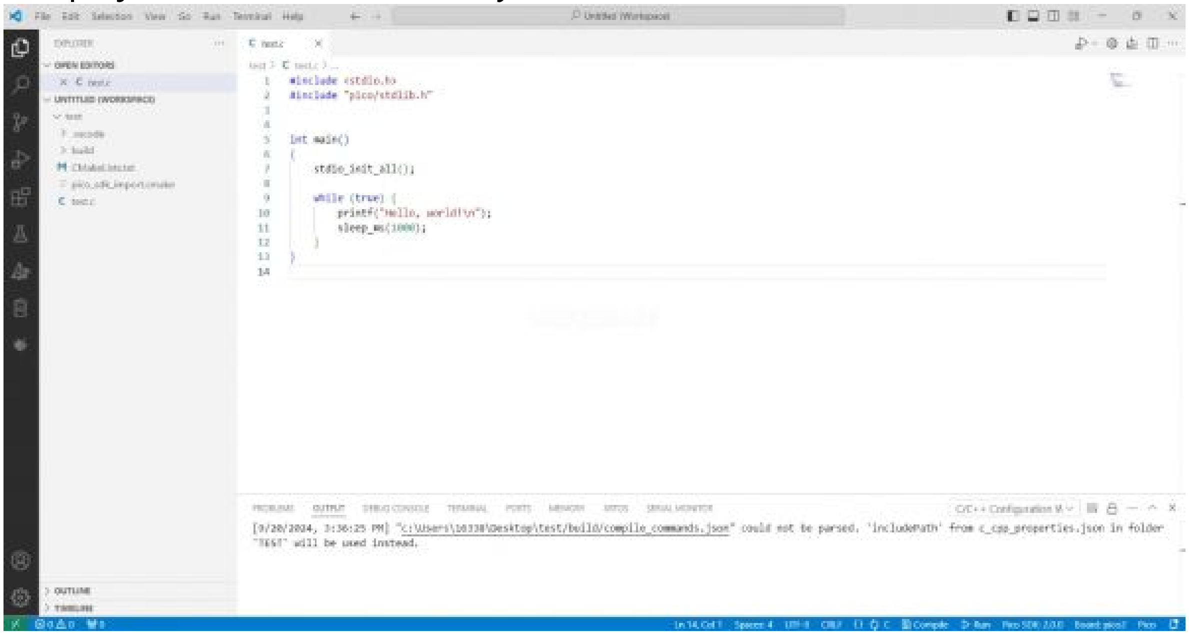
1. The configuration is complete, create a new project, enter the project name, select the path, and click Create to create the project
   To test the official example, you can click on the Example next to the project name to select



Name: Project name
Board type: Pico 2
Location: C:\Users\16338\Desktop
Select Pico-SDK version: v2.0.0
Use RISC-V ☐

(/wiki/File:Pico-vscode-11.JPG)

2. The project is created successfully



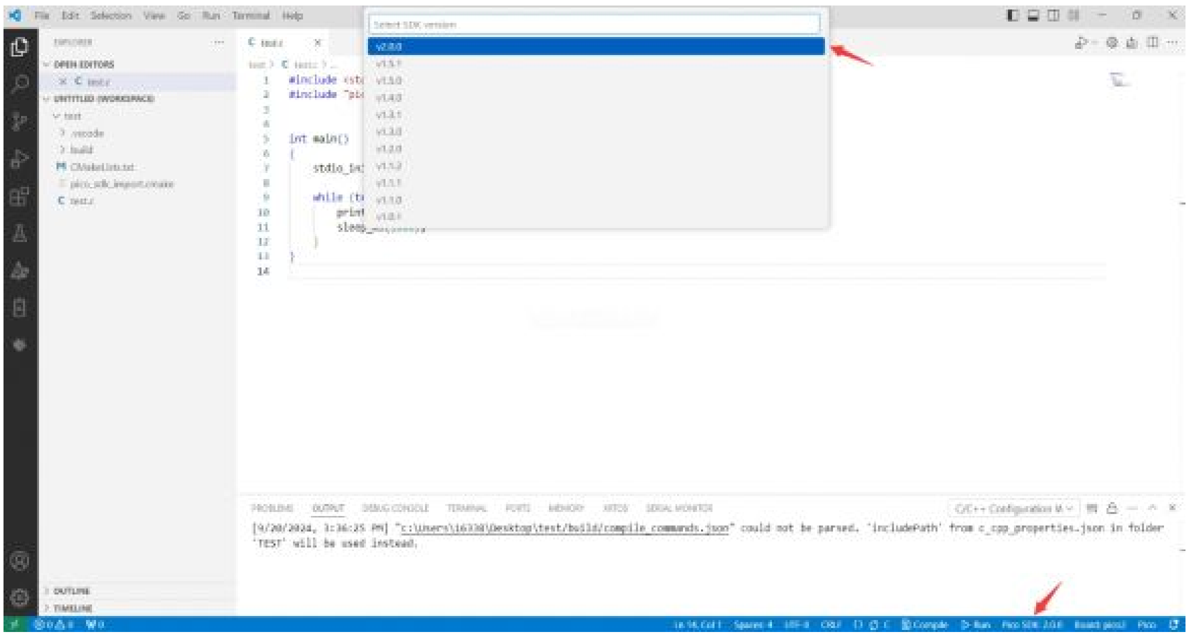
```
#include <stdio.h>
#include "pico/stdlib.h"


int main()
{
    stdio_init_all();

    while (true) {
        printf("Hello, world!\n");
        sleep_ms(1000);
    }
}
```

```
[9/20/2024, 3:36:25 PM] "c:\Users\16338\Desktop\test/build/compile_commands.json" could not be parsed. 'includePath' from c_cpp_properties.json in folder "TEST" will be used instead.
```

(/wiki/File:Pico-vscode-12.JPG)

Compile Project

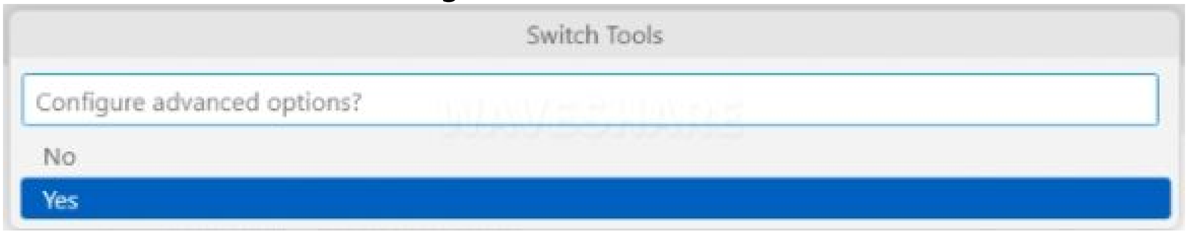
1. Select the SDK version



```
[9/20/2024, 1:36:25 PM] "c:\users\16338\Desktop\test/build/compile_commands.json" could not be parsed. 'includePath' from c_cpp_properties.json in folder 'TEST' will be used instead.
```

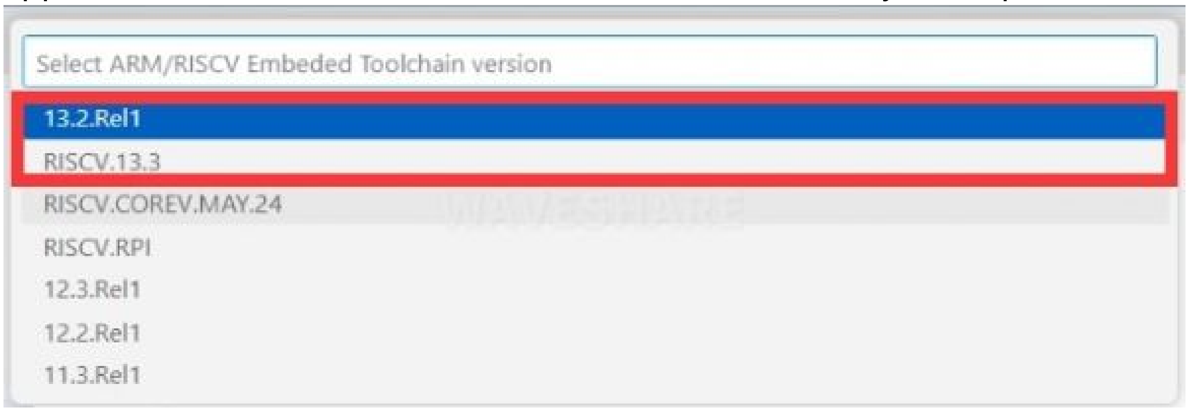
(/wiki/File:Pico-vscode-13.JPG)

2. Select Yes for advanced configuration



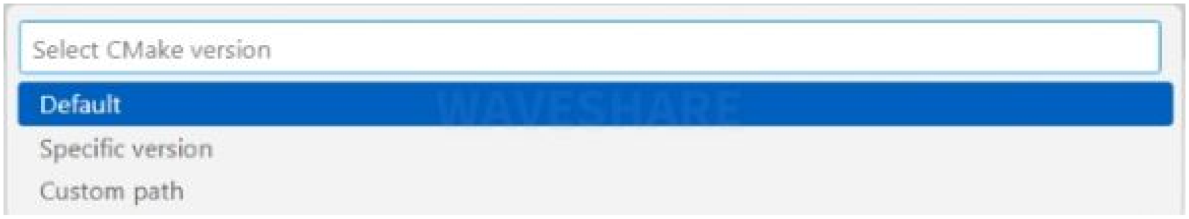
(/wiki/File:Pico-vscode-14.JPG)

3. Choose the cross-compilation chain, 13.2.Rel1 is applicable for ARM cores, RISCV.13.3 is applicable for RISCV cores. You can select either based on your requirements



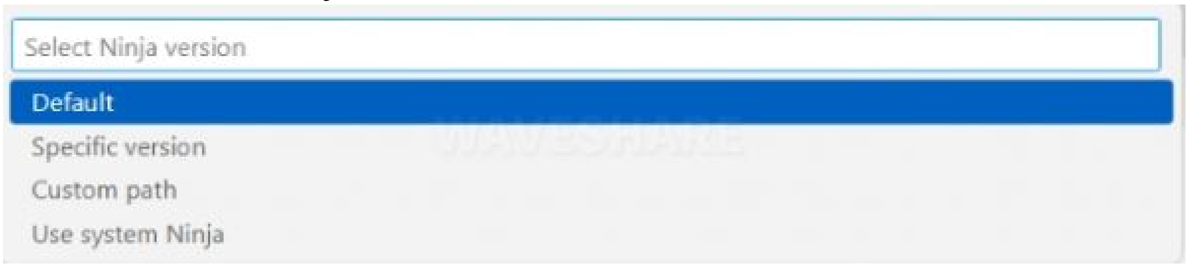
(/wiki/File:Pico-vscode-15.JPG)

4. Select Default for CMake version (the path configured earlier)



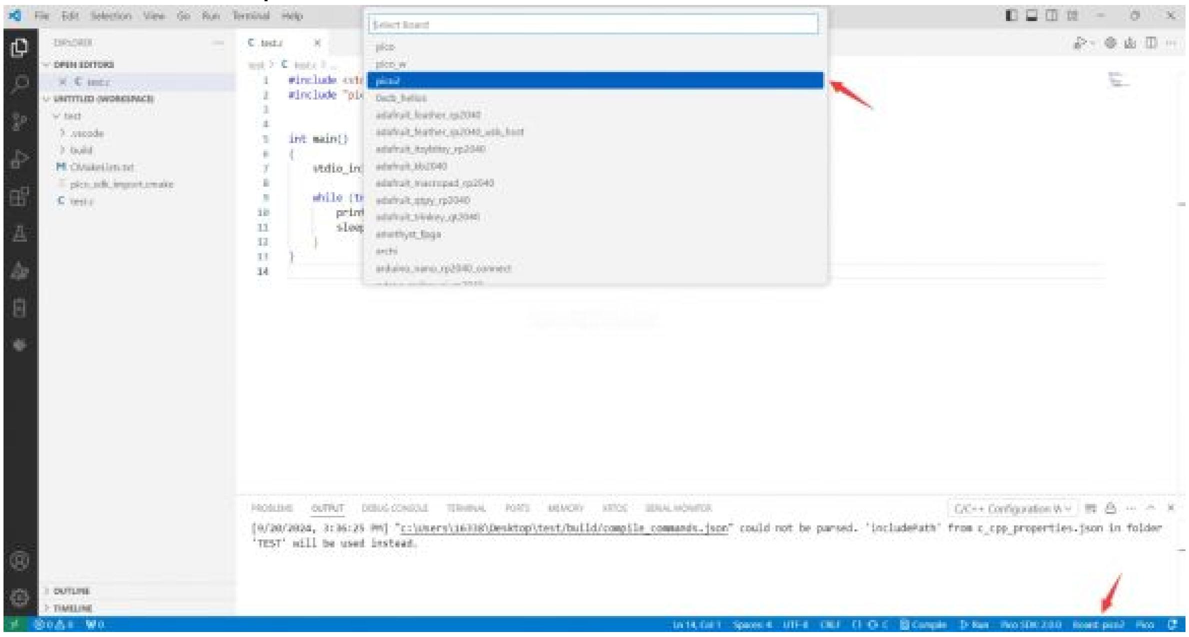
(/wiki/File:Pico-vscode-16.JPG)

5. Select Default for Ninja version



(/wiki/File:Pico-vscode-17.JPG)

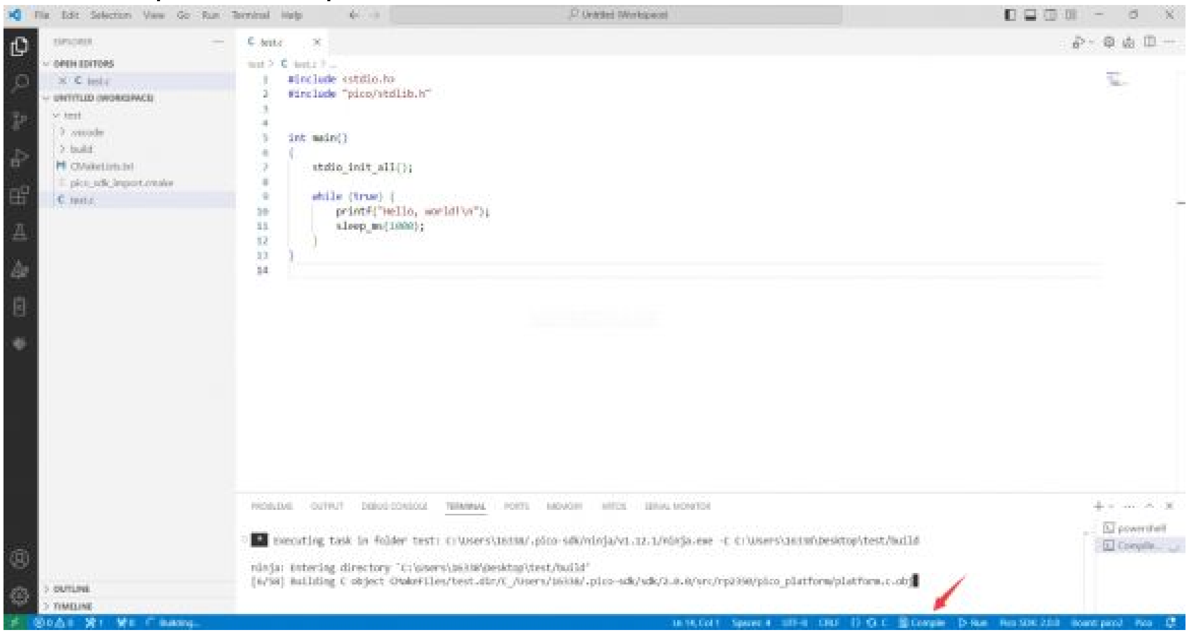
6. Select the development board



```
[9/20/2024, 3:36:25 PM] "c:\users\16338\Desktop\test/build/compile_commands.json" could not be parsed. 'includePath' from c_cpp_properties.json in folder
'TEST' will be used instead.
```

(/wiki/File:Pico-vscode-18.JPG)

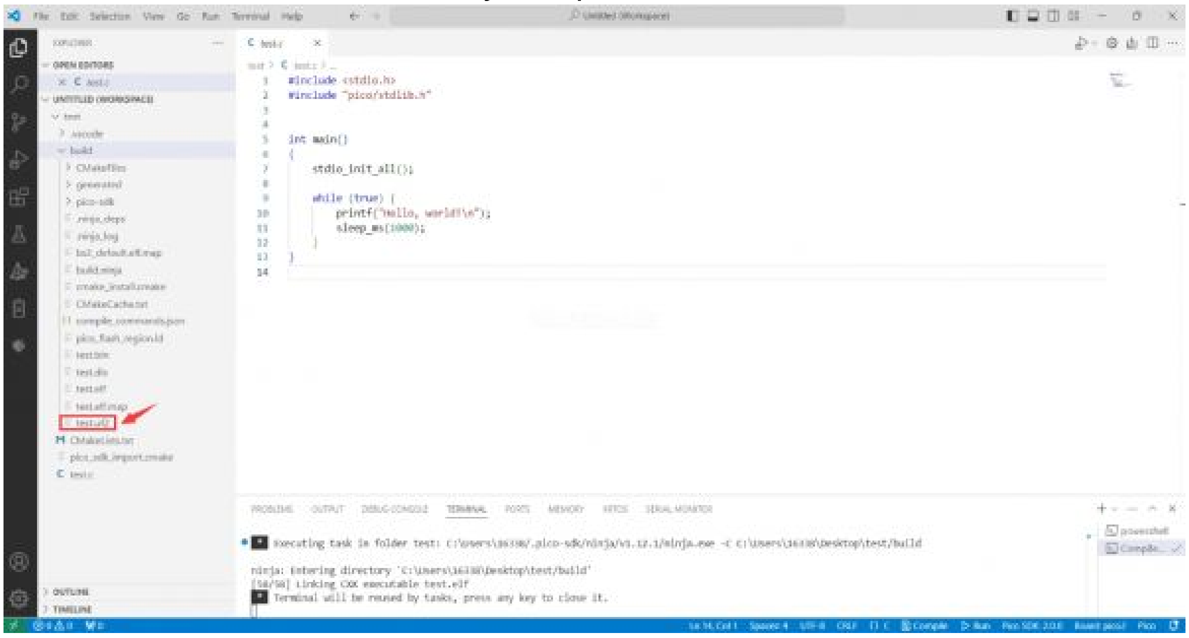
7. Click Complile to compile



```c
#include <stdio.h>
#include "pico/stdlib.h"


int main()
{
    stdio_init_all();

    while (true) {
        printf("Hello, world!\n");
        sleep_ms(1000);
    }
}
```

```
 *  Executing task in folder test: C:\Users\16338/.pico-sdk/ninja/v1.12.1/ninja.exe -C C:\Users\16338\Desktop\test/build

ninja: Entering directory `C:\Users\16338\Desktop\test/build'
[6/58] Building C object CMakeFiles/test.dir/C_/Users/16338/.pico-sdk/sdk/2.0.0/src/rp2350/pico_platform/platform.c.obj
```

(/wiki/File:Pico-vscode-19.JPG)

8. The uf2 format file is successfully compiled


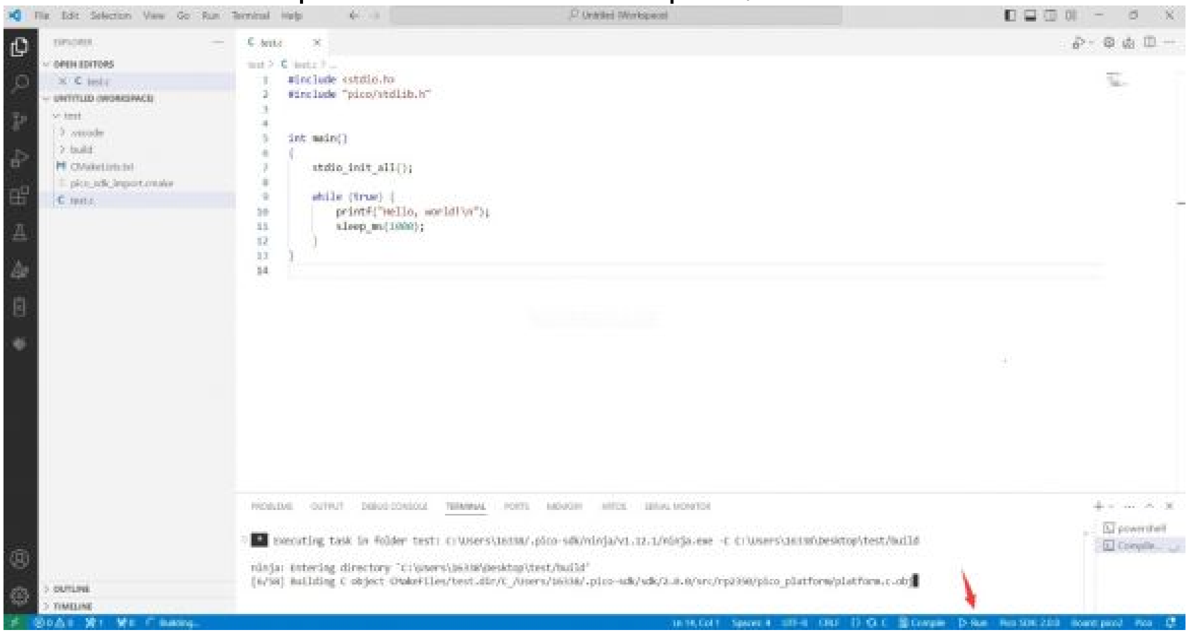
```c
#include <stdio.h>
#include "pico/stdlib.h"


int main()
{
    stdio_init_all();

    while (true) {
        printf("Hello, world!\n");
        sleep_ms(1000);
    }
}
```

```
Executing task in folder test: C:\Users\16138/.pico-sdk/ninja/v1.12.1/ninja.exe -C C:\Users\16138\Desktop\test/build

ninja: Entering directory `C:\Users\16138\Desktop\test/build'
[58/58] Linking CXX executable test.elf
Terminal will be reused by tasks, press any key to close it.
```

(/wiki/File:Pico-vscode-20.JPG)

Flash Firmware

Here are two methods for flashing firmware

1. Flash firmware using the pico-vscode plugin
   Connect the development board to the computer, click Run to flash the firmware directly


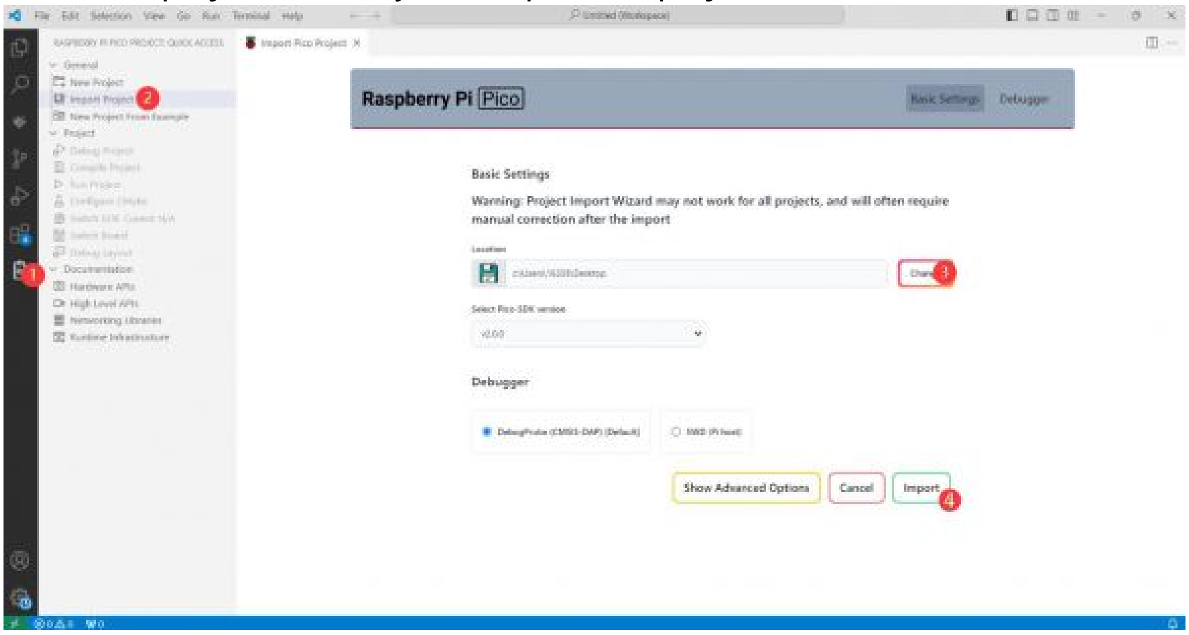
```c
#include <stdio.h>
#include "pico/stdlib.h"


int main()
{
    stdio_init_all();

    while (true) {
        printf("Hello, world!\n");
        sleep_ms(1000);
    }
}
```

```
Executing task in folder test: C:\Users\16138/.pico-sdk/ninja/v1.12.1/ninja.exe -C C:\Users\16138\Desktop\test/build

ninja: Entering directory `C:\Users\16138\Desktop\test/build'
[6/58] Building C object CMakeFiles/test.dir/C_/Users/16138/.pico-sdk/sdk/2.0.0/src/rp2350/pico_platform/platform.c.obj
```

(/wiki/File:600px-Pico-vscode-24.jpg)
2. Flash the firmware manually

```
1. Press and hold the Boot button
2. Connect the development board to the computer
3. Then the computer will recognize the development board as a USB device.
4. Copy the .uf2 file to the USB drive, and the device will automatically restart, indi
cating successful program flashing.
```

Import Project

1. Select the project directory and import the project


Basic Settings

Warning: Project Import Wizard may not work for all projects, and will often require manual correction after the import

Location: c:/Users/.../Desktop

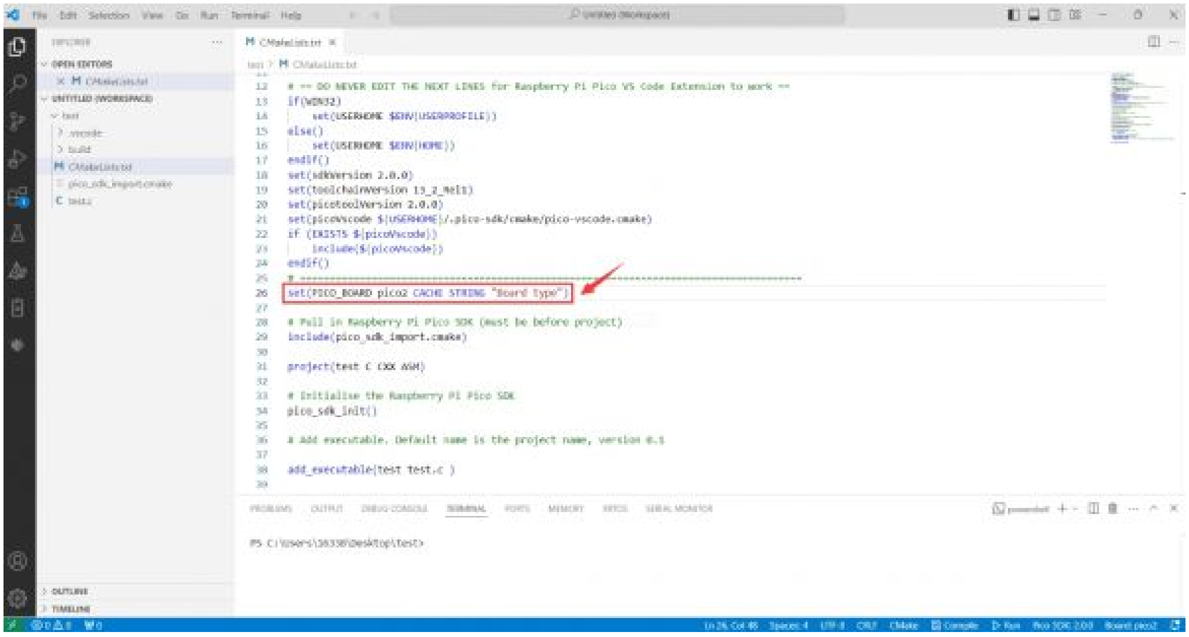
Select Pico-SDK version: v2.0.0

Debugger

DebugProbe (CMSIS-DAP) (Default)

SWD (Pi host)

Show Advanced Options | Cancel | Import

(/wiki/File:600px-Pico-vscode-23.jpg)
2. The Cmake file of the imported project cannot have Chinese (including comments), otherwise the import may fail
3. To import your own project, you need to add a line of code to the Cmake file to switch between pico and pico2 normally, otherwise even if pico2 is selected, the compiled firmware will still be suitable for pico


```
# == DO NEVER EDIT THE NEXT LINES for Raspberry Pi Pico VS Code Extension to work ==
if(WIN32)
    set(USERHOME $ENV{USERPROFILE})
else()
    set(USERHOME $ENV{HOME})
endif()
set(sdkVersion 2.0.0)
set(toolchainVersion 13_2_Rel1)
set(picotoolVersion 2.0.0)
set(picoVscode ${USERHOME}/.pico-sdk/cmake/pico-vscode.cmake)
if (EXISTS ${picoVscode})
    include(${picoVscode})
endif()
# ====================================================================================
set(PICO_BOARD pico2 CACHE STRING "Board type")

# Pull in Raspberry Pi Pico SDK (must be before project)
include(pico_sdk_import.cmake)

project(test C CXX ASM)

# Initialise the Raspberry Pi Pico SDK
pico_sdk_init()

# Add executable. Default name is the project name, version 0.1

add_executable(test test.c )
```

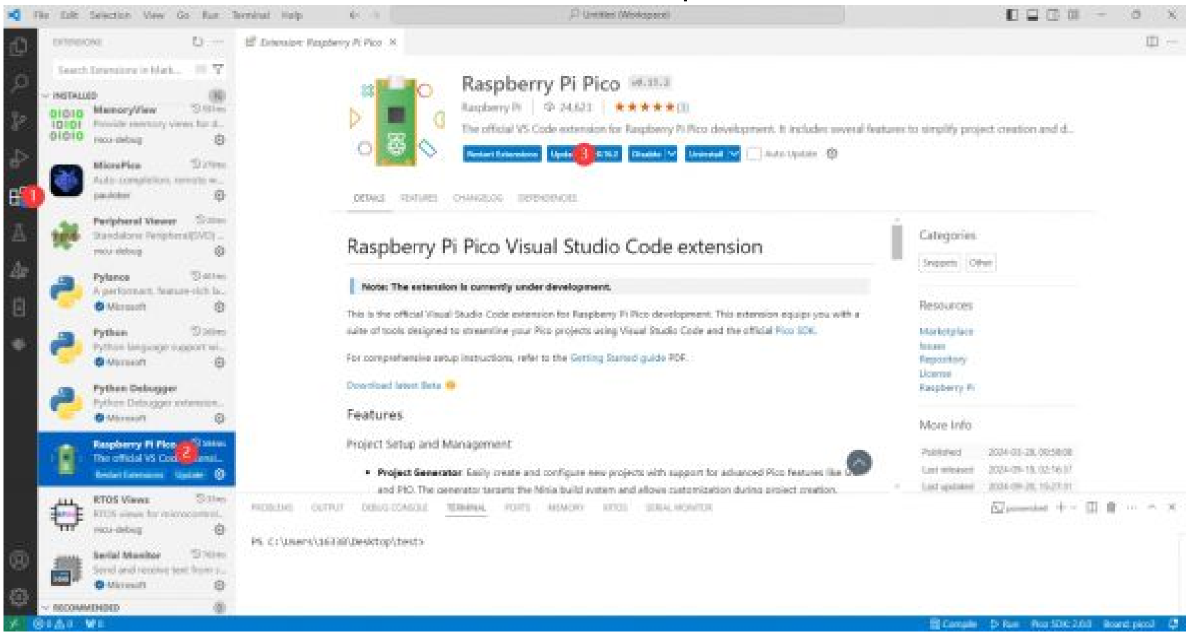
```
PS C:\Users\...\Desktop\test>
```

(/wiki/File:Pico-vscode-21.JPG)

```
set(PICO_BOARD pico CACHE STRING "Board type")
```

Update Extension

1. The extension version in the offline package is 0.15.2, and you can also choose to update to the latest version after the installation is complete



Raspberry Pi Pico Visual Studio Code extension

Note: The extension is currently under development.

This is the official Visual Studio Code extension for Raspberry Pi Pico development. This extension equips you with a suite of tools designed to streamline your Pico projects using Visual Studio Code and the official Pico SDK.

For comprehensive setup instructions, refer to the Getting Started guide PDF.

Features

Project Setup and Management

- **Project Generator**: Easily create and configure new projects with support for advanced Pico features like I2C and PIO. The generator targets the Ninja build system and allows customization during project creation.

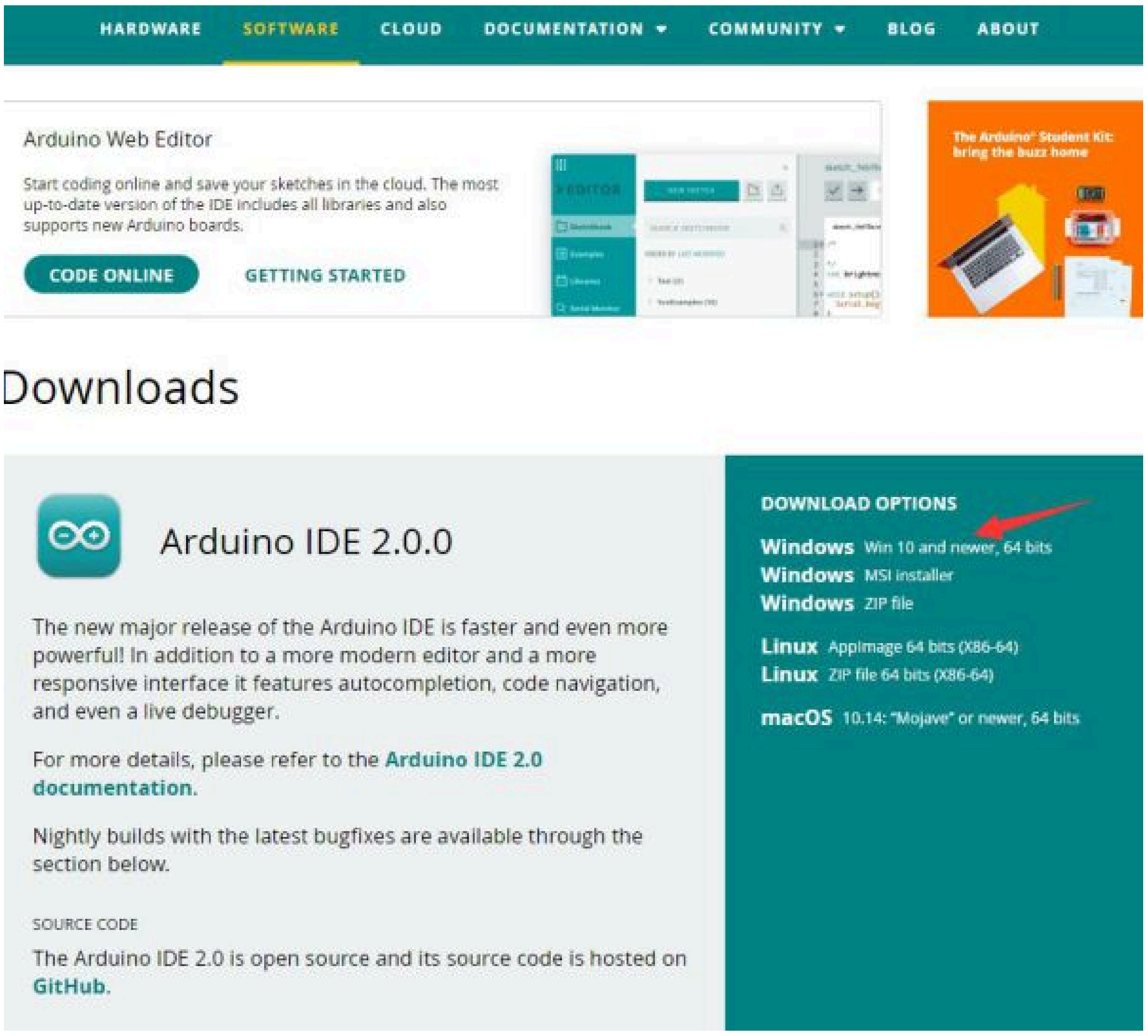
```
PS C:\Users\16138\Desktop\test>
```

(/wiki/File:Pico-vscode-22.JPG)

Arduino IDE Series

Install Arduino IDE

1. First, go to Arduino official website (https://www.arduino.cc/) to download the installation package of the Arduino IDE.



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

Downloads

Arduino IDE 2.0.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the Arduino IDE 2.0 documentation.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on GitHub.

DOWNLOAD OPTIONS

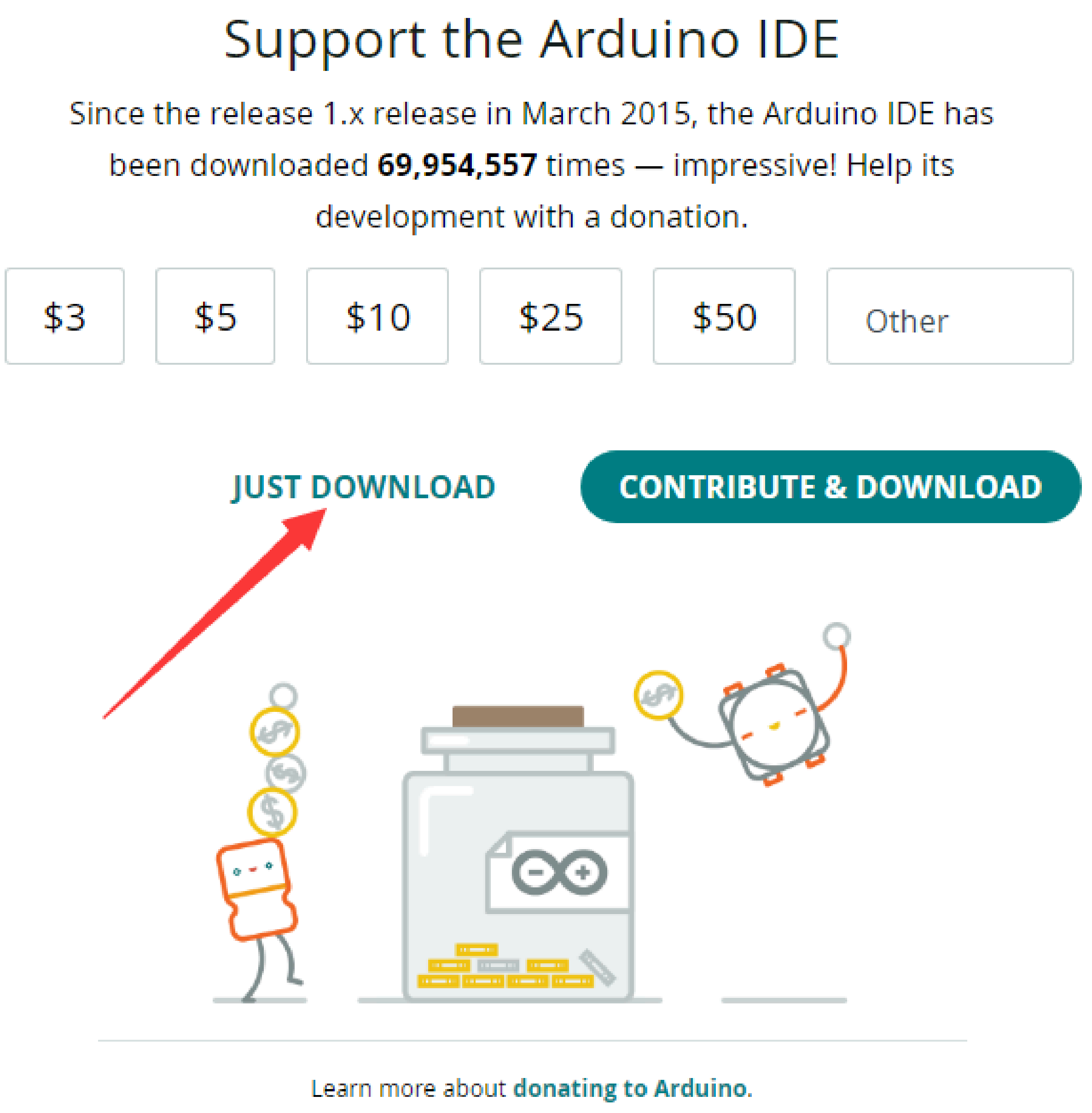
Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS 10.14: "Mojave" or newer, 64 bits

(/wiki/File:Arduino%E4%B8%8B%E8%BD%BD2.0%E7%89%88%E6%9C%AC.jpg)

2. Here, you can select Just Download.

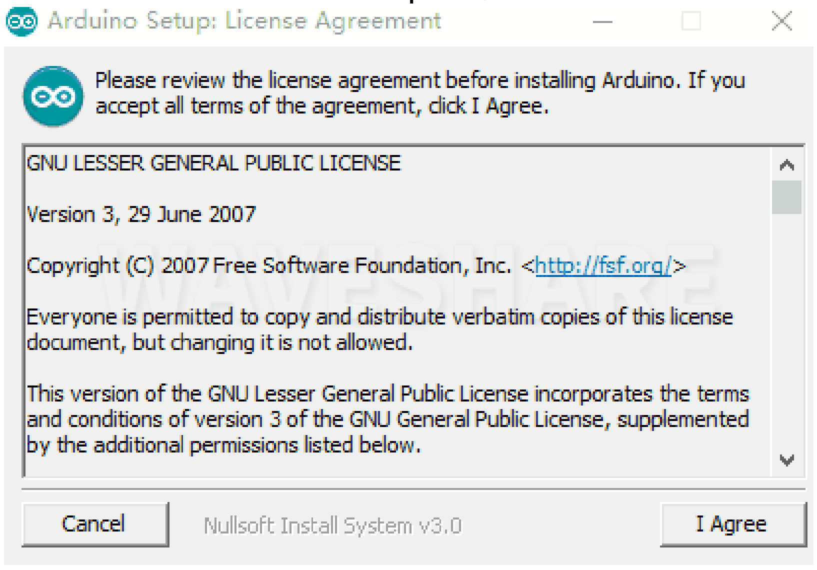

Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **69,954,557** times — impressive! Help its development with a donation.

Learn more about **donating to Arduino.**

(/wiki/File:%E4%BB%85%E4%B8%8B%E8%BD%BD%E4%B8%8D%E6%8D%90%E8%B5%A0.png)

3. Once the download is complete, click Install.


Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

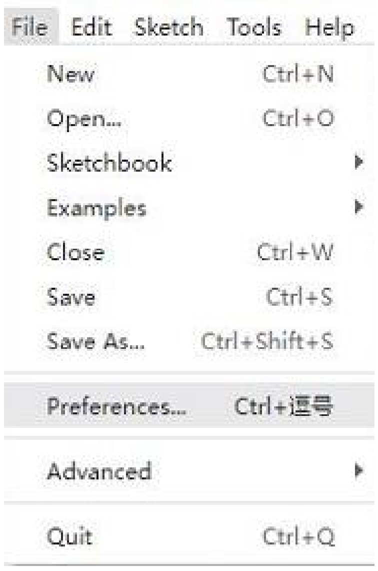
This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

(/wiki/File:IDE%E5%AE%89%E8%A3%85%E6%B0%B4%E5%8D%B0-1.gif)

Notice: During the installation process, it will prompt you to install the driver, just click Install

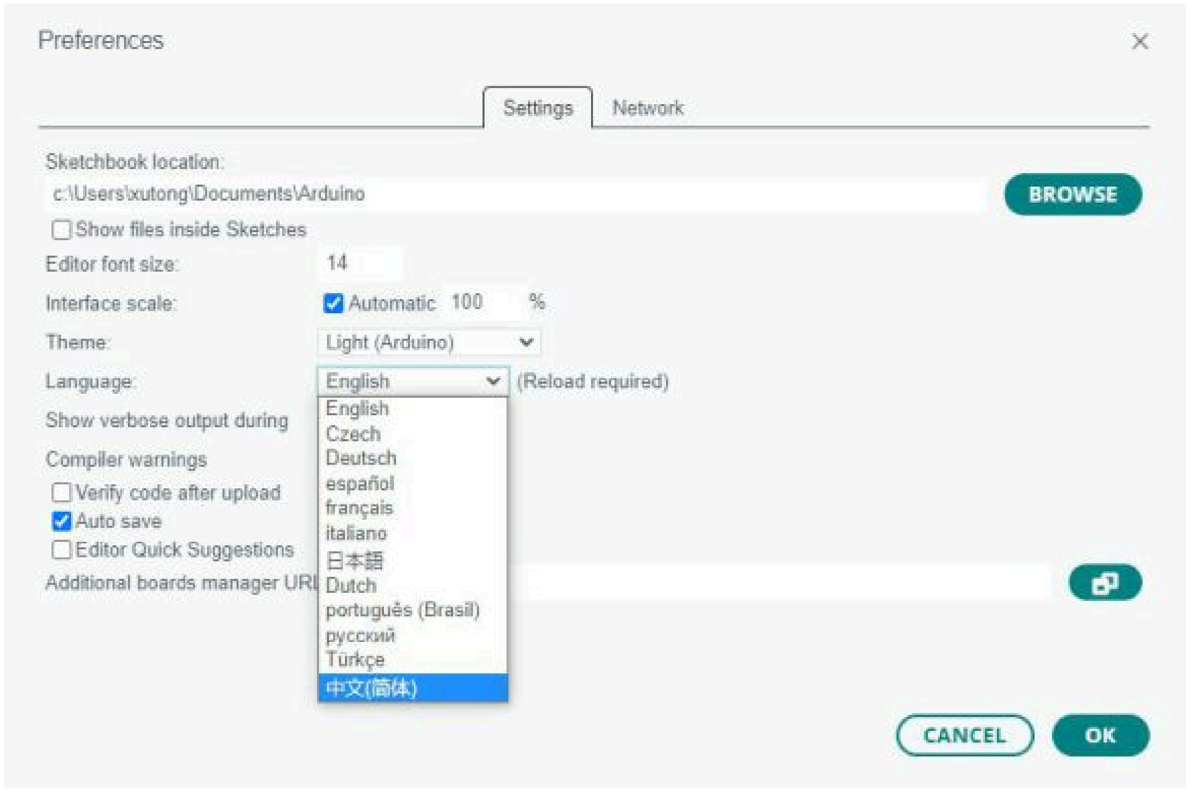
Arduino IDE Interface

1. After the first installation, when you open the Arduino IDE, it will be in English. You can switch to other languages in File --> Preferences, or continue using the English interface.



(/wiki/File:%E9%A6%96%E9%80%89%E9%A1%B9-%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87.jpg)

2. In the Language field, select the language you want to switch to, and click OK.



Preferences

Settings | Network

Sketchbook location:
c:\Users\xutong\Documents\Arduino
BROWSE
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during
Compiler warnings
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs

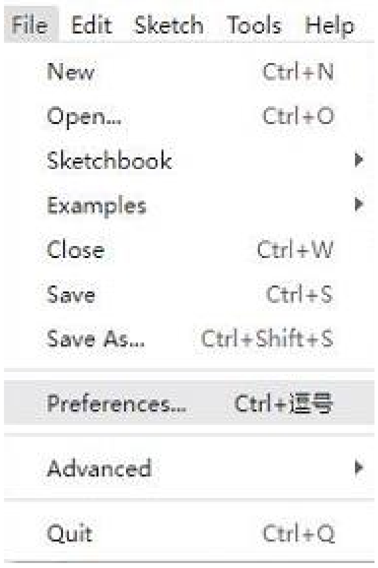
English
Czech
Deutsch
español
français
italiano
日本語
Dutch
português (Brasil)
русский
Türkçe
中文(简体)

CANCEL OK

(/wiki/File:%E9%A6%96%E9%80%89%E9%A1%B9-%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87ok.jpg)

Install Arduino-Pico Core in Arduino IDE

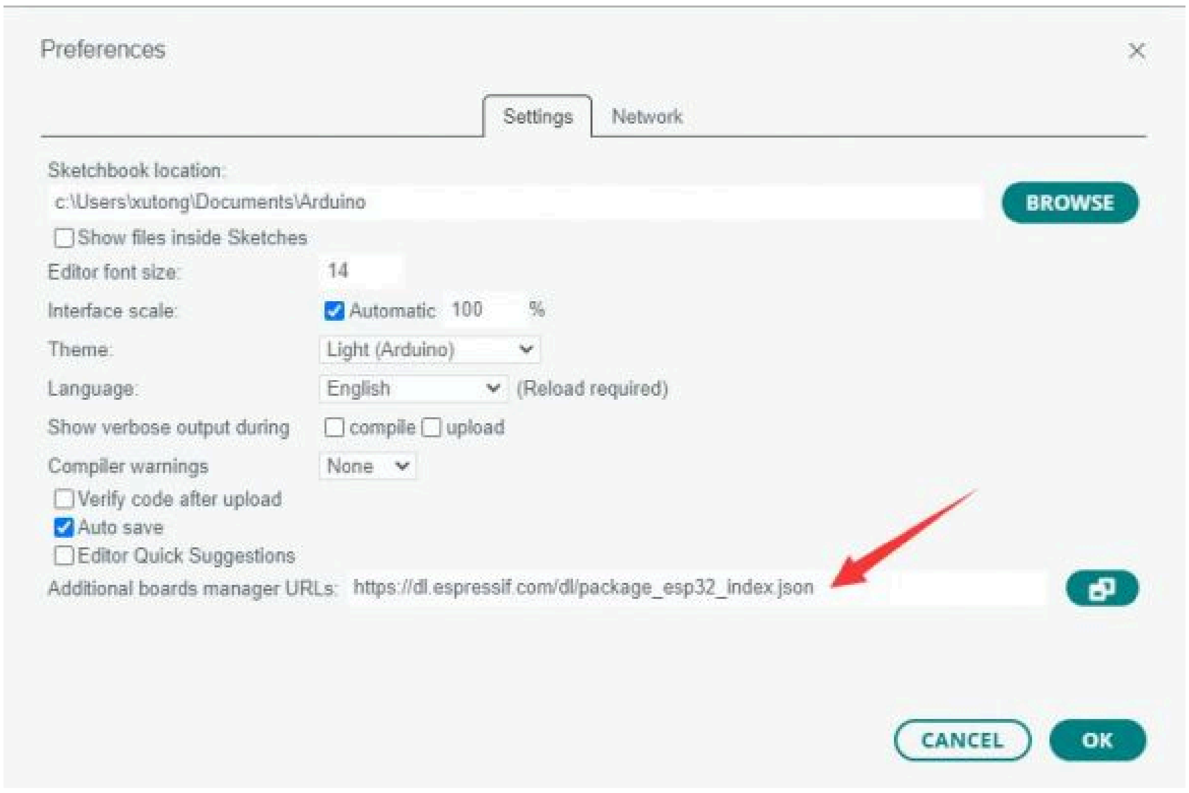
1. Open the Arduino IDE, click on the file in the top left corner, and select Preferences



(/wiki/File:RoArm-M1_Tutorial04.jpg)

2. Add the following link to the attached board manager URL, and then click OK
This link already includes board versions such as RP2040 and RP2350. Please visit arduino-pico (https://github.com/earlephilhower/arduino-pico) for the latest version files

```
https://github.com/earlephilhower/arduino-pico/releases/download/4.5.2/package_rp2040_index.json
```



Preferences

Settings | Network

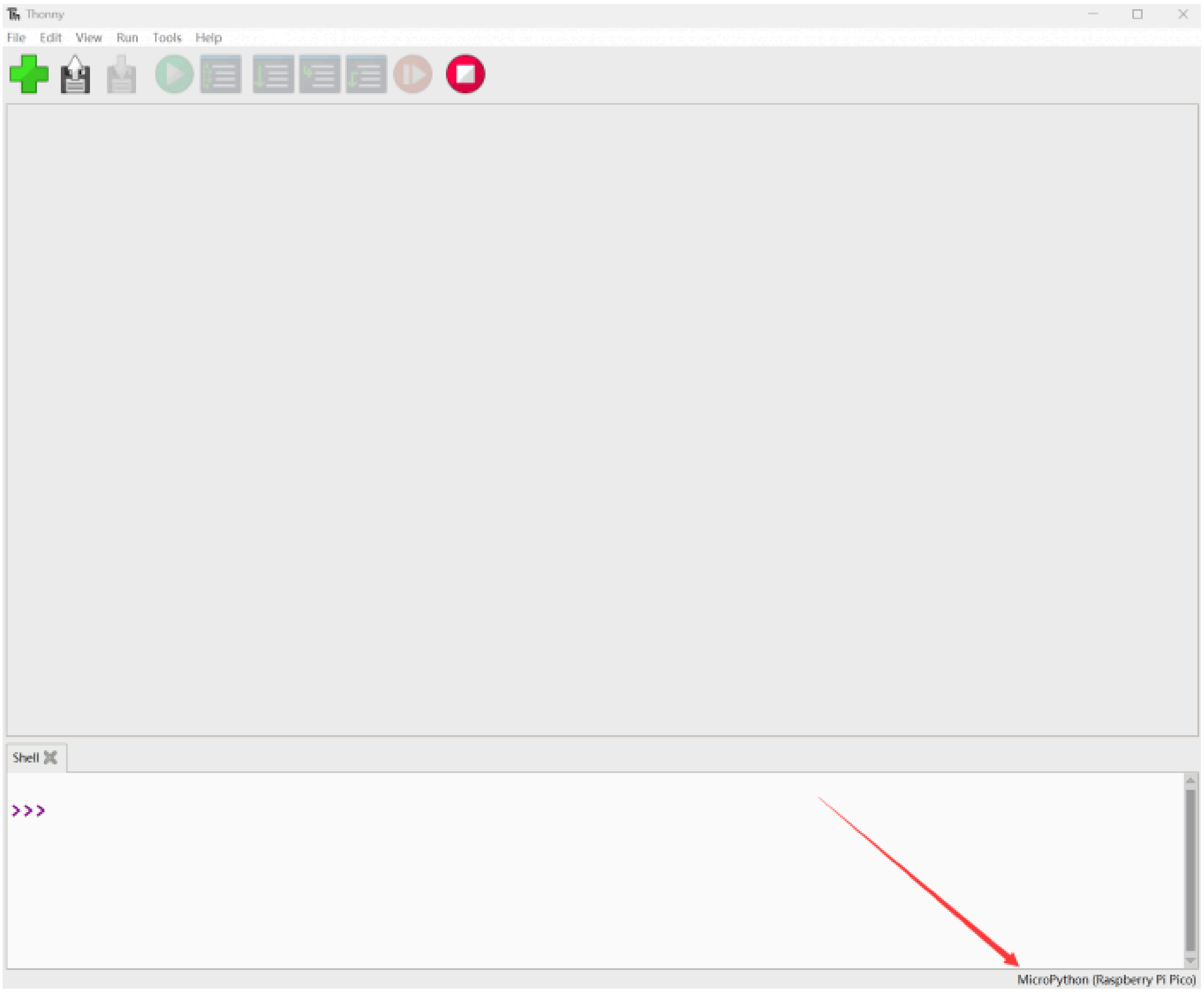
Sketchbook location: c:\Users\xutong\Documents\Arduino
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during ☐ compile ☐ upload
Compiler warnings None
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

(/wiki/File:RoArm-M1_Tutorial_II05.jpg)

Note: If you already have an ESP32 board URL, you can use a comma to separate the URLs as follows:

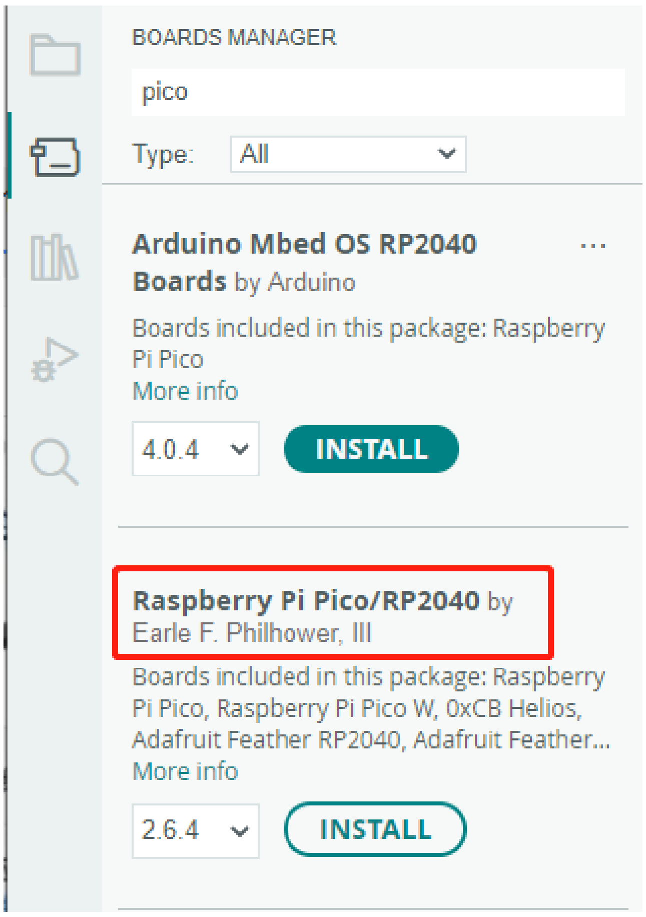
```
https://dl.espressif.com/dl/package_esp32_index.json,https://github.com/earlephilhower/
arduino-pico/releases/download/4.5.2/package_rp2040_index.json
```

3. Click Tools > Development Board > Board Manager > Search pico, as my computer has already been installed, it shows that it is installed


Thonny
File Edit View Run Tools Help
Shell
MicroPython (Raspberry Pi Pico)

```
>>>
```

(/wiki/File:Pico_Get_Start_05.png)


BOARDS MANAGER

pico

Type: All

Arduino Mbed OS RP2040 Boards by Arduino

Boards included in this package: Raspberry Pi Pico

More info

4.0.4 INSTALL

Raspberry Pi Pico/RP2040 by Earle F. Philhower, III

Boards included in this package: Raspberry Pi Pico, Raspberry Pi Pico W, 0xCB Helios, Adafruit Feather RP2040, Adafruit Feather...

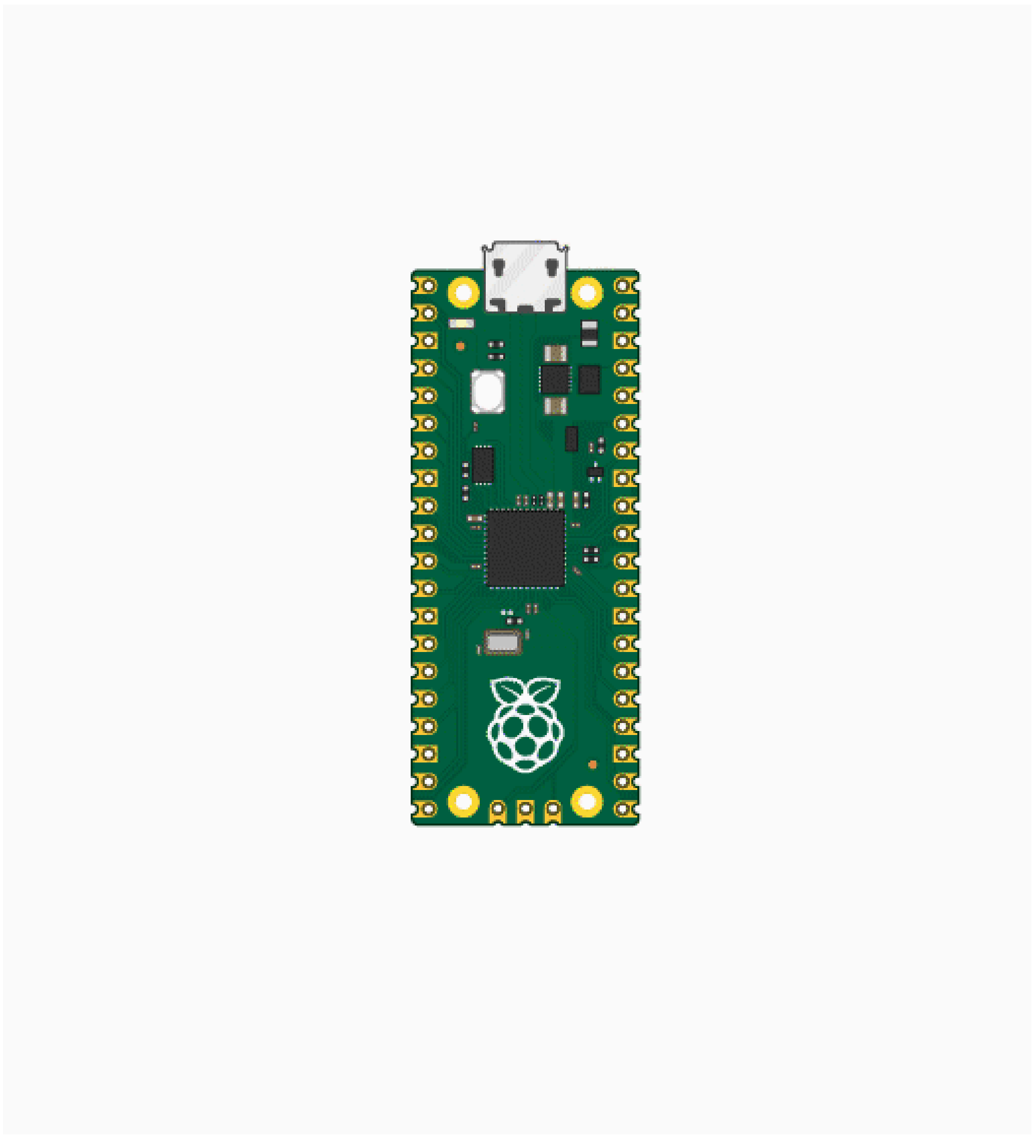
More info

2.6.4 INSTALL

(/wiki/File:Pico_Get_Start_06.png)

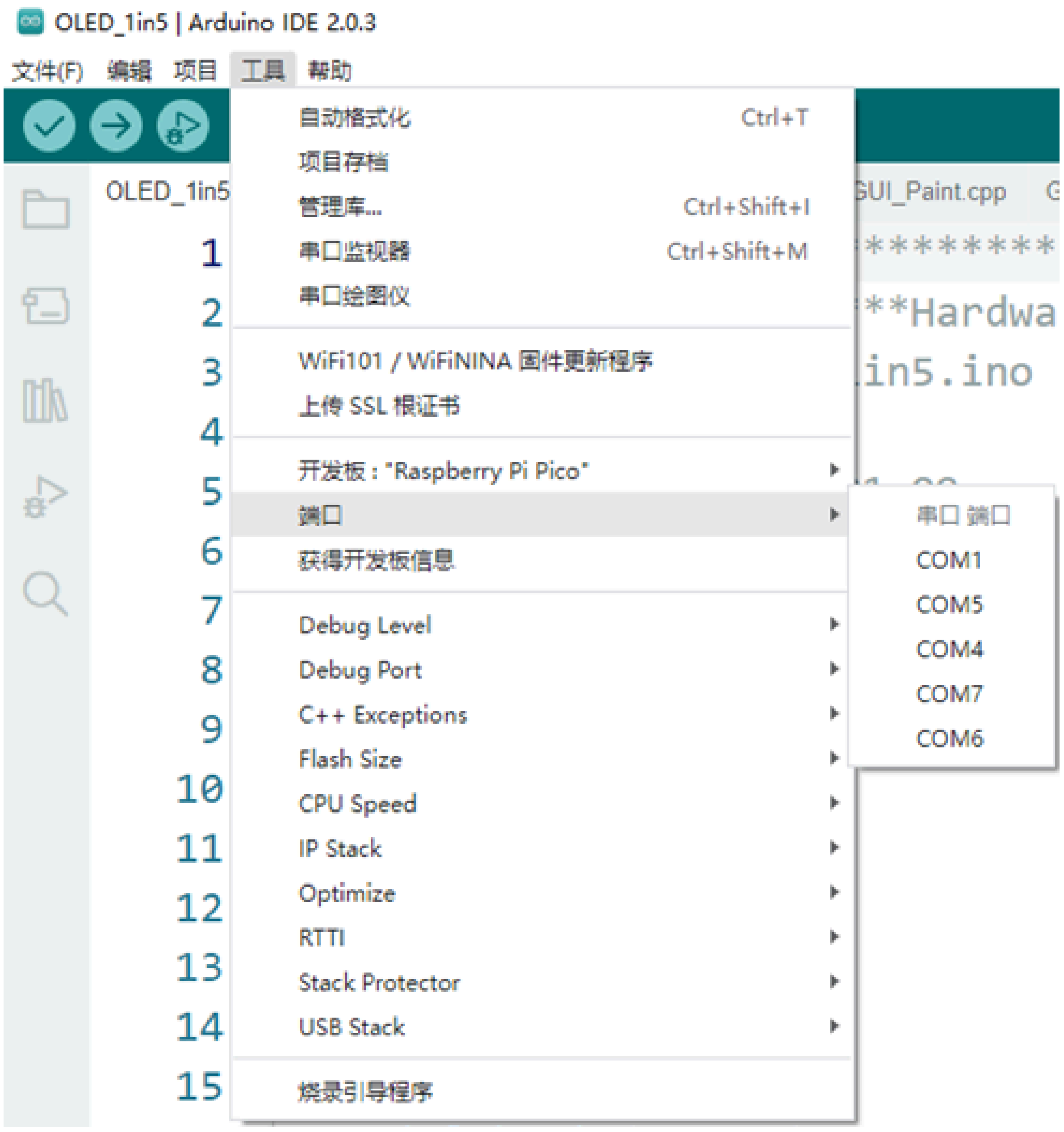
Upload Demo at the First Time

1. Press and hold the BOOTSET button on the Pico board, connect the pico to the USB port of the computer via the Micro USB cable, and release the button after the computer recognizes a removable hard disk (RPI-RP2).



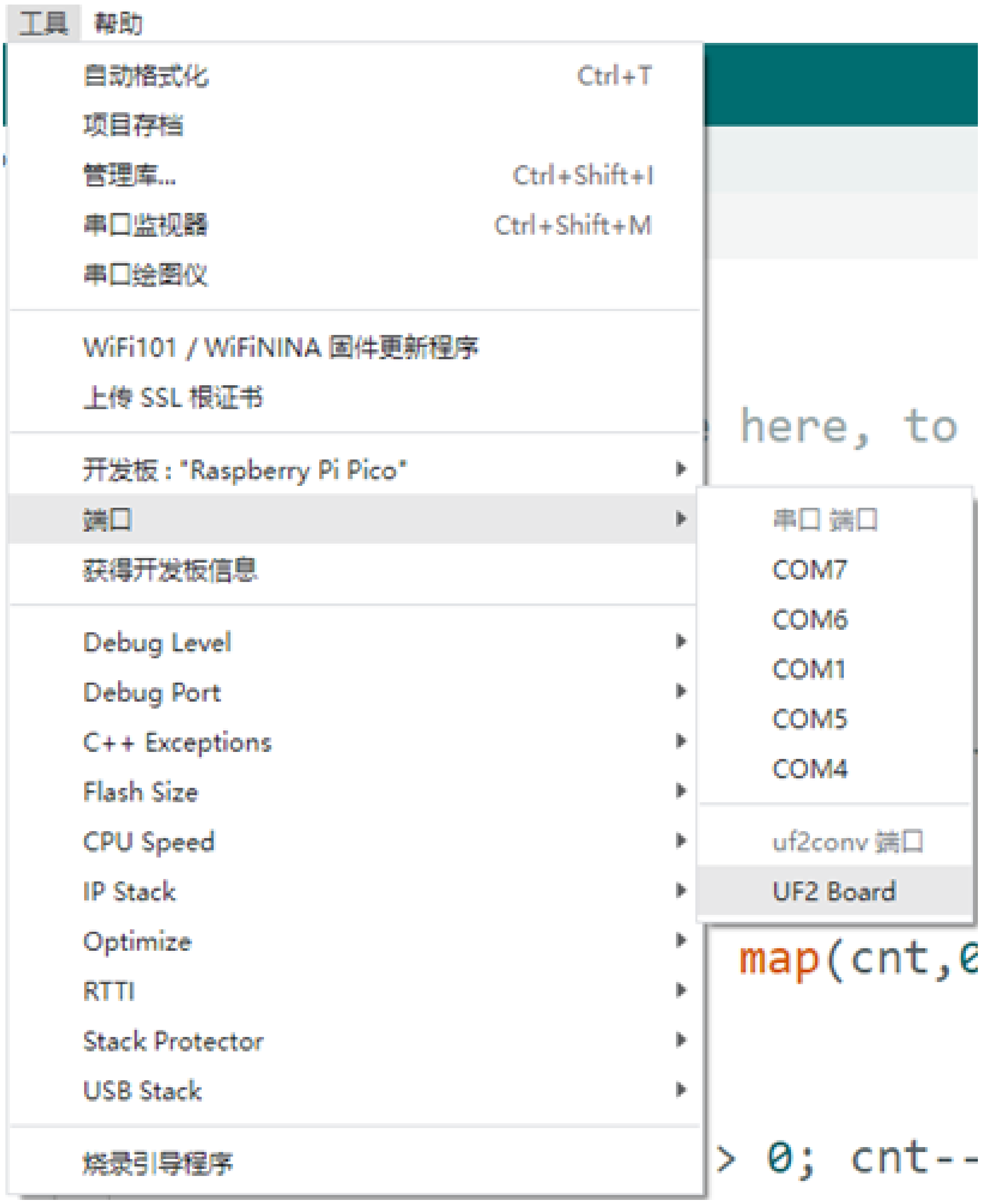
(/wiki/File:Pico%E8%BF%9E%E6%8E%A5%E6%95%B0%E6%8D%AE%E7%BA%BF.gif)

2. Download the program and open D1-LED.ino under the arduino\PWM\D1-LED path
3. Click Tools --> Port, remember the existing COM, do not click this COM (the COM displayed is different on different computers, remember the COM on your own computer)



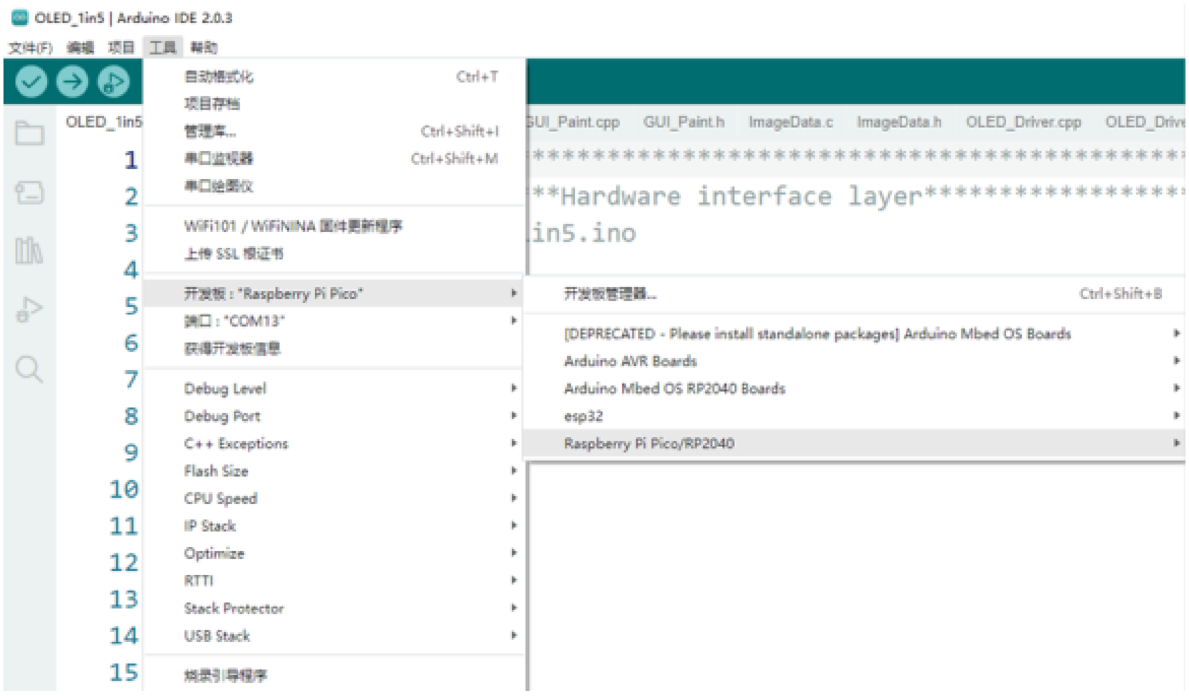
(/wiki/File:Pico%E8%BF%9E%E6%8E%A5%E5%89%8D%E7%AB%AF%E5%8F%A3.png)

4. Connect the driver board to the computer using a USB cable. Then, go to Tools > Port. For the first connection, select uf2 Board. After uploading, when you connect again, an additional COM port will appear



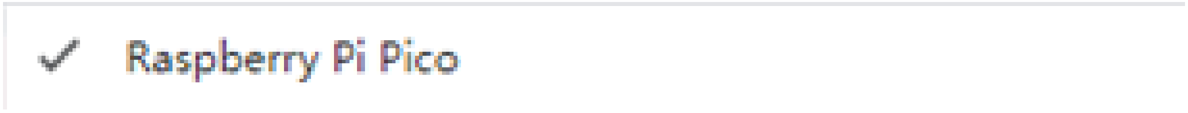
(/wiki/File:Pico%E8%BF%9E%E6%8E%A5%E5%90%8Euf2.png)

5. Click Tools > Development Board > Raspberry Pi Pico > Compatible versions (Raspberry Pi Pico, Raspberry Pi Pico 2, etc.)


OLED_1in5 | Arduino IDE 2.0.3
文件(F) 编辑 项目 工具 帮助
自动格式化 Ctrl+T
项目存档
管理库... Ctrl+Shift+I
串口监视器 Ctrl+Shift+M
串口绘图仪
WiFi101 / WiFiNINA 固件更新程序
上传 SSL 根证书
开发板："Raspberry Pi Pico"
端口："COM13"
获得开发板信息
Debug Level
Debug Port
C++ Exceptions
Flash Size
CPU Speed
IP Stack
Optimize
RTTI
Stack Protector
USB Stack
烧录引导程序
开发板管理器... Ctrl+Shift+B
[DEPRECATED - Please install standalone packages] Arduino Mbed OS Boards
Arduino AVR Boards
Arduino Mbed OS RP2040 Boards
esp32
Raspberry Pi Pico/RP2040
GUI_Paint.cpp GUI_Paint.h ImageData.c ImageData.h OLED_Driver.cpp OLED_Drive

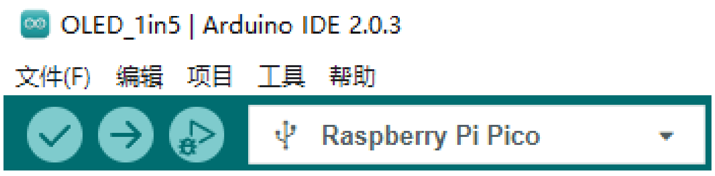
```
***********************************************
**Hardware interface layer*****************
in5.ino
```

(/wiki/File:%E5%B7%A5%E5%85%B7pico%E5%BC%80%E5%8F%91%E6%9D%BF.png)



(/wiki/File:Arduono-Raspberrypi_pico.png)

6. After setting it up, click the right arrow to upload the program



(/wiki/File:Pico%E4%B8%8A%E4%BC%A0%E7%A8%8B%E5%BA%8F.png)

- **If issues arise during this period, and if you need to reinstall or update the Arduino IDE version, it is necessary to uninstall the Arduino IDE completely. After uninstalling the software, you need to manually delete all contents within the C:\Users\[name]\AppData\Local\Arduino15 folder (you need to show hidden files to see this folder). Then, proceed with a fresh installation.**

Open Source Demos

MircoPython video demo (github) (https://github.com/waveshareteam/Pico_MircoPython_Examples)

MicroPython firmware/Blink demos (C) (https://files.waveshare.com/wiki/common/Raspberry_Pi_Pico_Demo.zip)

Raspberry Pi official C/C++ demo (github) (https://github.com/raspberrypi/pico-examples/)

Raspberry Pi official micropython demo (github) (https://github.com/raspberrypi/pico-micropython-examples)

Arduino official C/C++ demo (github) (https://github.com/earlephilhower/arduino-pico)

Resources

Supporting Resources

Documents

- RP2040-Zero Schematic diagram (https://files.waveshare.com/upload/4/4c/RP2040_Zero.pdf)
- RP2040-Zero-STEP file (3D drawing) (https://files.waveshare.com/upload/f/f7/RP2040_Zero_stp.zip)

Demos

- WS2812B Test Code (https://files.waveshare.com/upload/5/58/RP2040-Zero.zip)

Application

- JustUSB Project (shared by Waveshare users) (https://github.com/Alwinator/JustUSB)

Official Resources

Raspberry Pi Official Documents

- Get Started with MicroPython on Raspberry Pi Pico (https://hackspace.raspberrypi.org/books/micropython-pico)
- Raspberry Pi related books download (https://magpi.raspberrypi.org/books)
- Raspberry Pi Pico Schematic (https://files.waveshare.com/upload/e/ed/RPI-PICO-R3-PUBLIC-SCHEMATIC.pdf)
- Pico Pinout definition (https://files.waveshare.com/upload/5/52/Pico-R3-A4-Pinout.pdf)
- Pico (https://files.waveshare.com/upload/3/30/Getting_started_with_pico.pdf)
- Pico C SDK User Manual (https://files.waveshare.com/upload/5/5f/Pico_c_sdk.pdf)
- Pico Python SDK User Manual (https://files.waveshare.com/upload/b/b0/Pico_python_sdk.pdf)
- Pico Datasheet (https://files.waveshare.com/upload/1/11/Pico_datasheet.pdf)
- RP2040 Datasheet (https://files.waveshare.com/upload/f/fd/Rp2040_datasheet.pdf)
- RP2040 Hardware Design Manual (https://files.waveshare.com/upload/9/9d/Hardware_design_with_rp2040.pdf)

Raspberry Pi Open Source Demos

- Raspberry Pi official C/C++ Demos (github) (https://github.com/raspberrypi/pico-examples/)
- Raspberry Pi official micropython Demos (github) (https://github.com/raspberrypi/pico-micropython-examples)

FAQ

Question: Why do I put a file on a disk and the disk disappears and I don't see the previous file when I re-enter the disk?

Answer:

1. The Bootrom of RP2040 provides a standard USB boot loader, which can be recognized as a writable drive for using UF2 files to copy code to the RP2040. The UF2 file copied to the drive is downloaded and written to Flash or RAM, and the device is automatically restarted, allowing code to be downloaded and run on the RP2040 solely via USB connection.
2. Any type of file can be written from the host to a USB drive, but these files are typically not stored; it appears so only due to the host's cache. Only when a UF2 file is written to the device, special content is recognized, and data is written to a designated position in RAM or Flash. After downloading the complete and valid UF2 file, the RP2040 will automatically reboot to run the newly downloaded code.
3. UF2 files will not be stored, but instead, the firmware is flashed to the specified location according to the corresponding file format, the specific file format can refer to Microsoft's open source project at https://github.com/microsoft/uf2 (https://github.com/microsoft/uf2)

Question: How to flash the firmware?

Answer:

Press RESET first, then press BOOT. Release RESET first, then release BOOT to enter the flashing mode. Just drag and drop or copy the firmware into it

Question:How do I debug this design when the debugger pins are not brought out?

Answer:

Debugging is not possible. You can program on a board that can be debugged and then directly burn the firmware into the RP2040 Zero.

Question:Is there any way to connect a battery pack to the WS RP2040-Zero using GPIO pins or anything else?

Answer:

The VSYS pin of the RP2040 is connected to the VUSB pin directly in RP2040-zero (named Pin23), If you want to connect the battery directly to the VSYS pin, you need to add a diode to avoid backflow. You can also directly connect the battery to Pin 21 (the 3V3) of the RP2040-zero if the voltage of the battery is 3.3V.
RP2040 zero itself has no battery protection function, you need to ensure that your battery will not be overcharged or over-discharged, causing safety accidents.

Question:Is that none of the debugging pins are exposed for SWD debugging?

Answer:

This board doesn't pin out the SWD pins.

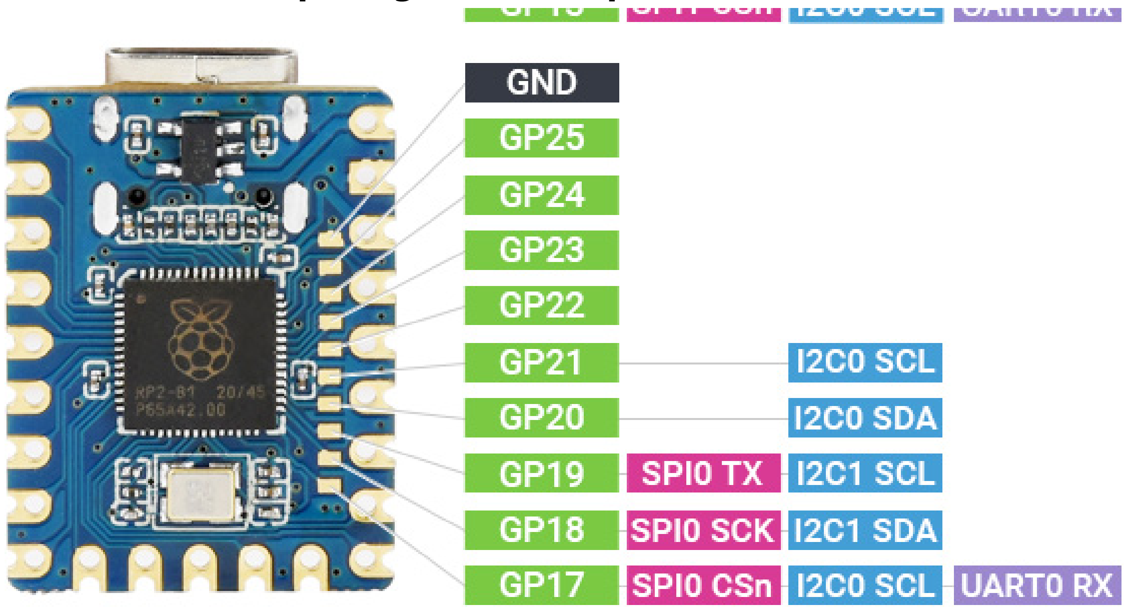
Question:Any advice on using the RP2040-Zero in USB host mode? For the pico, it recommends an OTG cable but since the RP2040-Zero has a USB-C connector there is no sense of pin?

Answer:

Just use an OTG adapter like this or use a C2C cable:

(/wiki/File:RP2040-Zero-FAQ05.png)

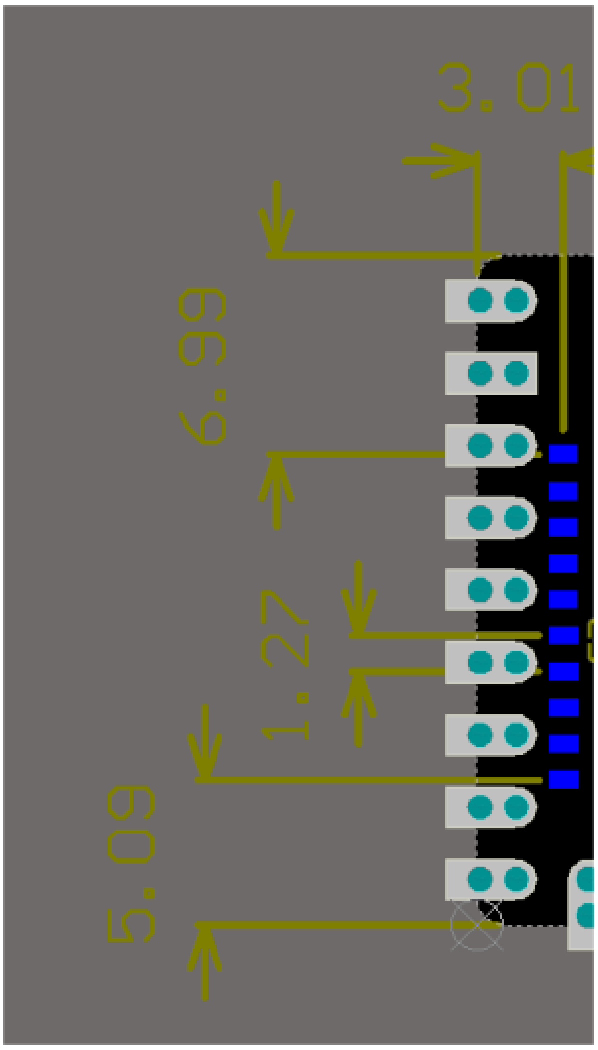
Question:What is the spacing on the 10 pads underneath the RP2040-Zero?



(/wiki/File:RP2040-Zero-FAQ01.png)

Answer:

Please check this:

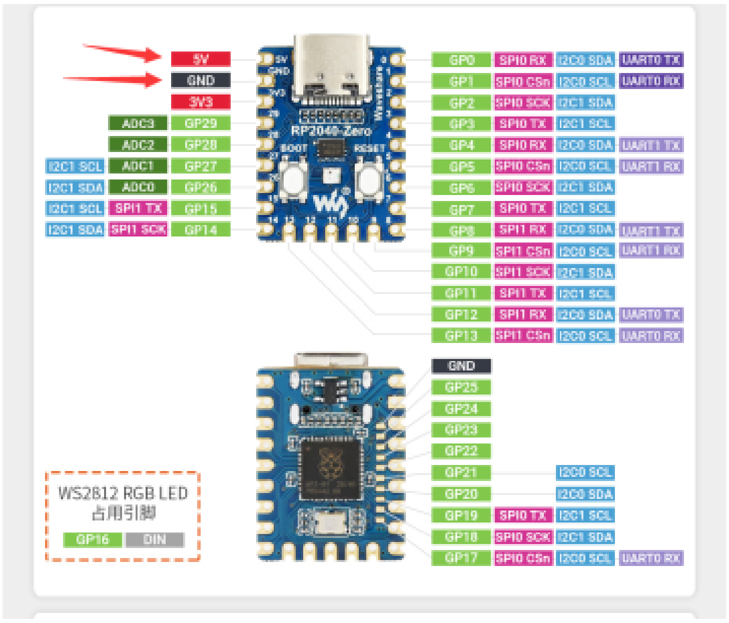


(/wiki/File:RP2040-Zero-FAQ02.png)

Question:In P2040-Zero to which pin must the 5V input voltage must be given to power up the board?

Answer:

If you do not use USB for power supply, you can use the 5V pin for power supply:



(/wiki/File:RP2040-zero-09.png)

Question:I want to power RP2040-Zero using 1s lipo battery.. can I connect the battery directly to the 5v pin mentioned in the pinout document?

Answer:

Due to the limited space, the power management part is omitted, resulting in zero can only be powered by 5V/3.3V. But zero itself has no battery protection function, you need to ensure that your battery will not be overcharged or over-discharged, which will cause safety accidents.

Question:How can I power it using a battery power source such as a lipo3.7v?

Answer:

The VSYS pin of the RP2040 is connected to the VUSB pin directly in RP2040-zero (named Pin23), If you want to connect the battery directly to the VSYS pin, you need to add a diode to avoid backflow. You can also directly connect the battery to Pin 21 (the 3V3) of the RP2040-zero if the voltage of the battery is 3.3V.

Question:Could it be powered via a regulated 3.3V supply but not the NOT USBC - 5V?

Answer:

The VSYS pin of the RP2040 is connected to the VUSB pin directly in RP2040-zero (named Pin23), if you do not need to use the USB port, you can connect a 3.3V power to the VSYS pin, we still recommend you to add a diode to it to avoid backflow.

Question:What is the lowest sleep current that can be achieved and how do I achieve this?

Answer:

The RP2040 microcontroller, which is used in the RP2040 Zero, has the potential to achieve very low sleep currents, making it ideal for low-power applications.
The power consumption is 2mA.

Implementation example: https://github.com/raspberrypi/pico-playground/tree/master?tab=readme-ov-file#sleep (https://github.com/raspberrypi/pico-playground/tree/master?tab=readme-ov-file#sleep)

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now ()