RP2350-Relay-6CH

From Waveshare Wiki
Jump to: navigation, search

Overview

Introduction

RP2350-Relay-6CH and RP2350-Relay-6CH-W are industrial-grade 6-channel relay series products based on the RP2350B main controller, both equipped with peripheral interfaces such as RS485 and Pico, and featuring multiple protection circuits including power isolation and optocoupler isolation. Among them, the RP2350-Relay-6CH-W additionally supports WiFi and Bluetooth wireless communication functions. This series is designed with a focus on safety and stability, providing industrial-grade reliable protection.

RP2350-Relay-6CH

(https://www.waveshare.com/rp2350-relay-6ch.htm?sku=31790)

SPI / I2C / RS485

RP2350-Relay-6CH-W

(https://www.waveshare.com/rp2350-relay-6ch.htm?sku=32576)

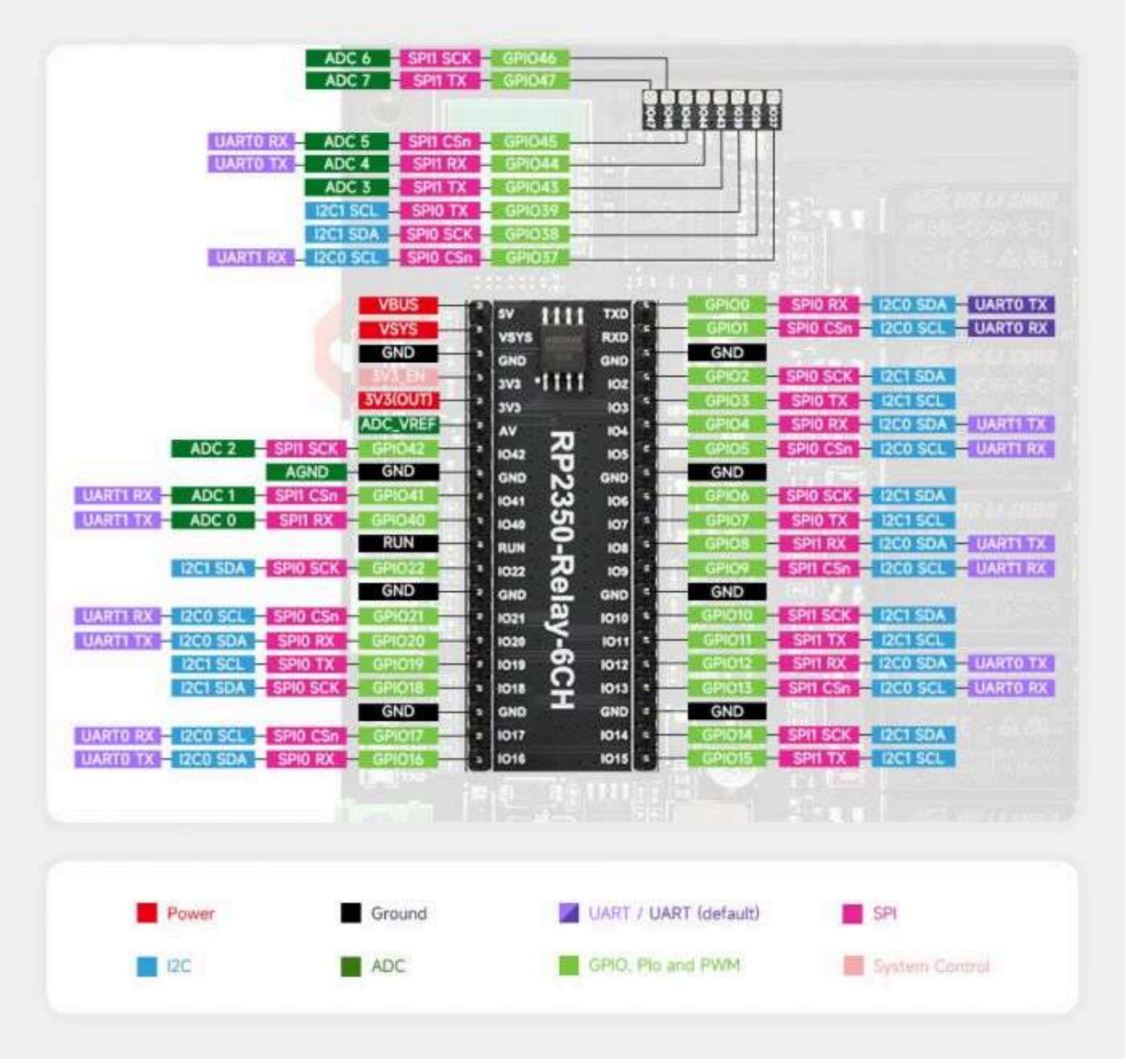
SPI / I2C / RS485

Features

- RP2350B microcontroller chip officially designed by Raspberry Pi
- Unique dual-core and dual-architecture design, equipped with dual-core ARM Cortex-M33 processor and dual-core Hazard3 RISC-V processor, flexible clock running up to 150 MHz, supporting flexible switching between the two architectures
- Equipped with Raspberry Pi Radio Module 2 module, supporting Wi-Fi 4 wireless network and Bluetooth 5.2 (Wi-Fi module version only)
- Built-in 520KB of SRAM and 16MB of on-chip Flash
- Type-C port, easier to use
- High quality relay, contact rating: ≤10A 250V AC or ≤10A 30V DC
- Onboard isolated RS485 interface, for connecting to various RS485 Modbus industrial modules or sensors
- Onboard Pico compatible interfaces can be adapted to some Raspberry Pi Pico HAT, for expanding more functions such as RTC / CAN / RS232 / LoRa / sensor, etc.
- Onboard power supply screw terminal, supports 7~36V wide voltage input, suitable for industrial applications

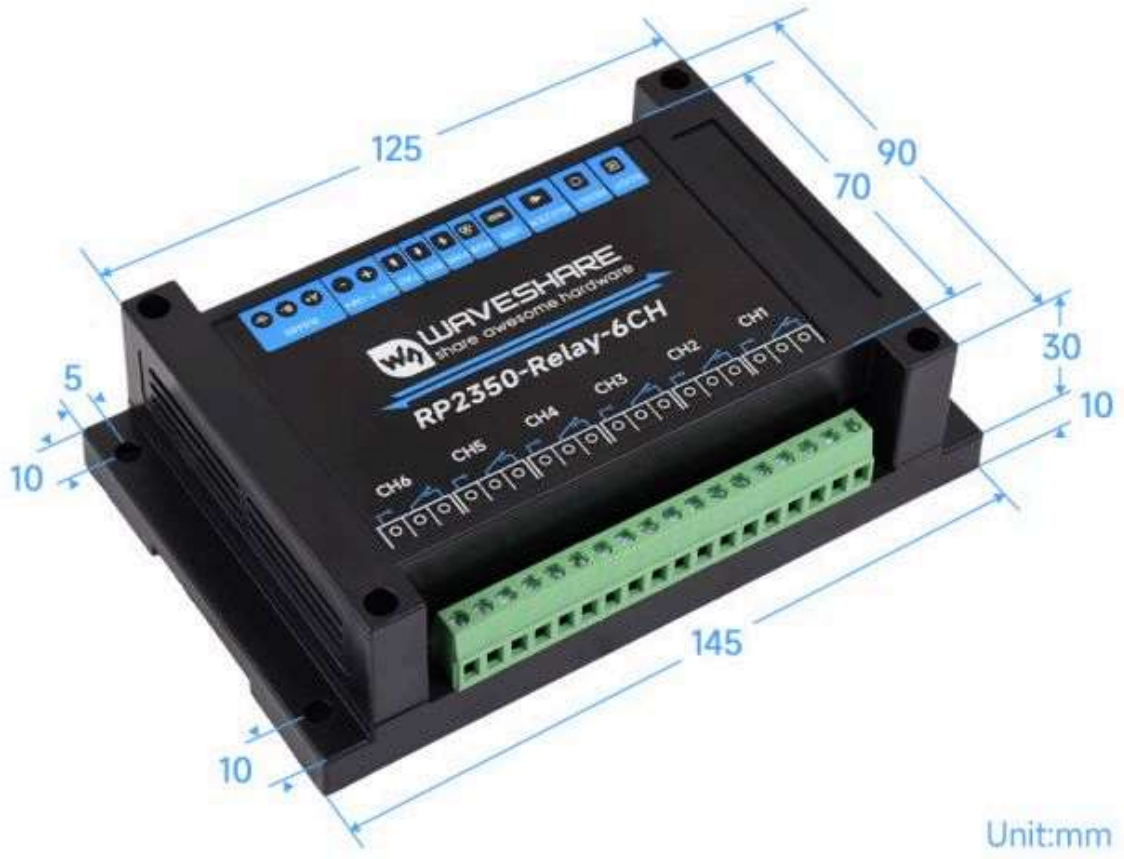
- Onboard optocoupler isolation to prevent interference with control chip from external high-voltage circuit connected to the relay
- Onboard digital isolation to avoid external signal interference with the control chip
- Onboard integrated power isolation, providing stable isolation voltage, no extra power supply required for the isolated terminal
- Built-in buzzer, RGB colorful LED, power supply and RS485 TX/RX indicators for monitoring the operating status of the module
- Rail-mounted ABS protective enclosure, easy to install, safe to use
- USB1.1 host and slave device support
- Low-power sleep and dormant modes
- Drag-and-drop programming via USB mass storage
- 34 × multi-functional GPIO pins
- 2 × I2C, 2 × UART, 8 × 12-bit ADC and 24 × controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- On-chip accelerated floating-point library
- 12 × Programmable I/O (PIO) state machines for custom peripheral support

Pinout Definition



(/wiki/File:700px-RP2350-Relay-6CH-details-inter_(1).jpg)

Dimensions



(/wiki/File:700px-RP2350-Relay-6CH-details-size_(1).jpg)

Electrical and Relay Safety Instructions

- This product must be operated by professional electricians or qualified personnel. During use, ensure electrical safety, leakage protection, and proper insulation.
- Before installing, maintaining, or replacing the relay device, always turn off the power and unplug the device.
- Do not attempt to disassemble the relay device to avoid damage or the risk of electric shock.
- Properly install and place the relay device. Do not use it in humid, overheated, flammable, or explosive environments to prevent accidents caused by improper installation or use.

1. Load Matching

- Ensure the relay's rated voltage and current match the load. Do not exceed the rated capacity.
- For inductive loads (motors, coils, lamps, etc.), the starting current may be much higher than the rated current. Choose a relay with sufficient current margin.

2. Short Circuit and Overcurrent Protection

- Install a **fuse** or **circuit breaker** in the relay circuit to prevent damage due to short circuits or accidental overcurrent.

- Ensure the load circuit has no short circuits during wiring, and select protection components with appropriate current ratings if necessary.

3. Arc and Switching Protection

- Relay switching generates arcs, which can cause contact wear or welding.
- For inductive loads, it is recommended to use **RC snubber circuits** or **varistors** for arc suppression.

4. Installation Environment

- Do not use the relay in humid, high-temperature, flammable, explosive, or dusty environments.
- Install the relay securely to avoid vibrations or shocks that may cause misoperation or damage.

5. Power-Off Operation

- Always cut off power before maintenance, wiring, or replacing the relay to ensure personnel and device safety.
- Latching relays are only powered when changing state. Avoid strong vibrations or strong magnetic fields while the relay is unpowered.

6. Status Confirmation

- After powering on, confirm or reset the relay status as needed to prevent abnormal operation caused by transportation, installation, or external disturbances.
- Avoid power interruption during relay operation to prevent uncertain status or contact damage.

7. Regular Inspection

- Periodically inspect relay contacts, terminals, and insulation to ensure proper operation.
- If abnormal heating, odor, or burn marks are detected, immediately cut off power and replace the relay.

Pico Getting Started

Firmware Download

- MicroPython Firmware Download [Expand]

- C_Blink Firmware Download [Expand]

Basic Introduction

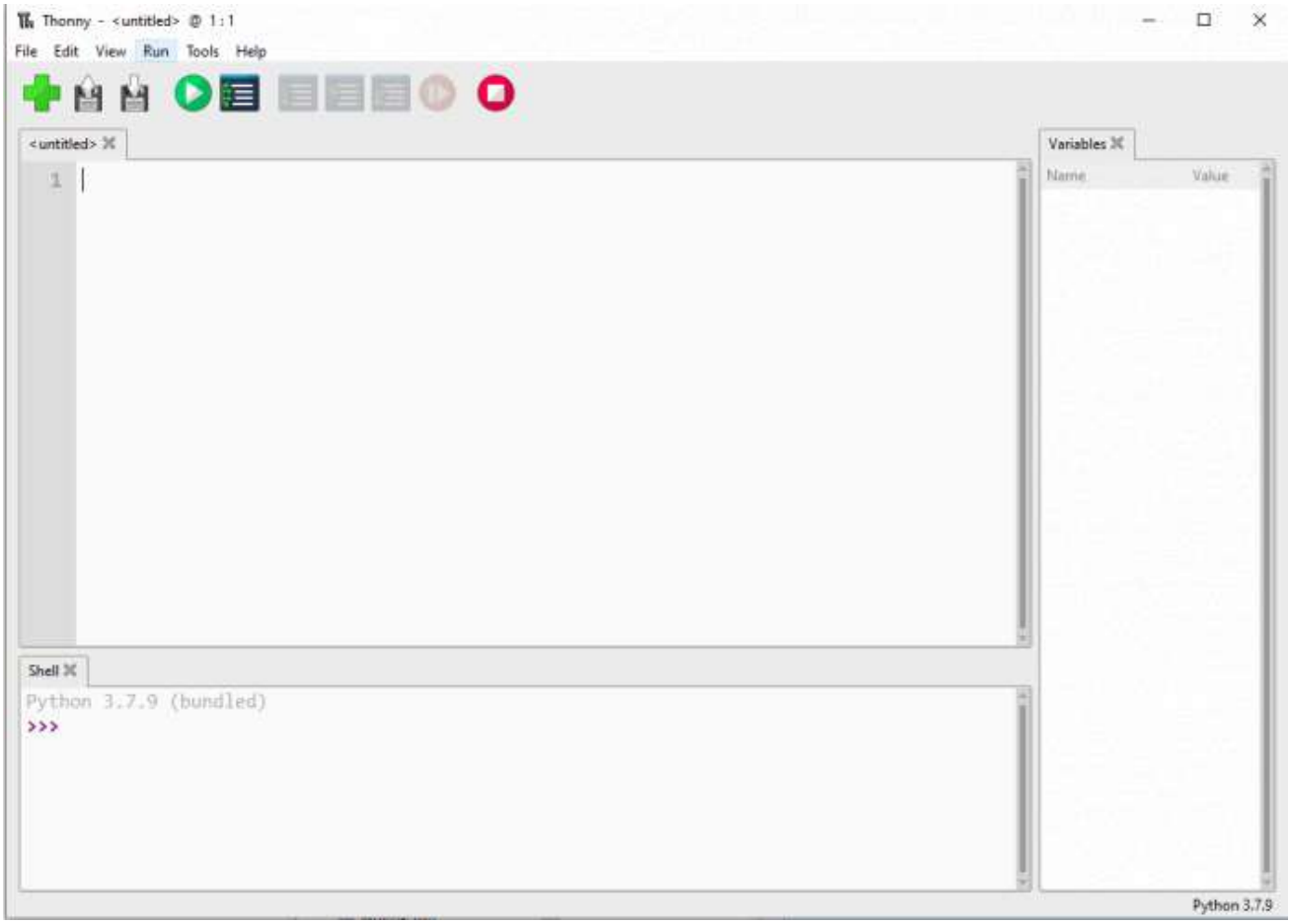
Raspberry Pi Pico Basics (https://www.waveshare.com/wiki/Pico_Basic_Introduction)

MicroPython Series

Install Thonny IDE

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

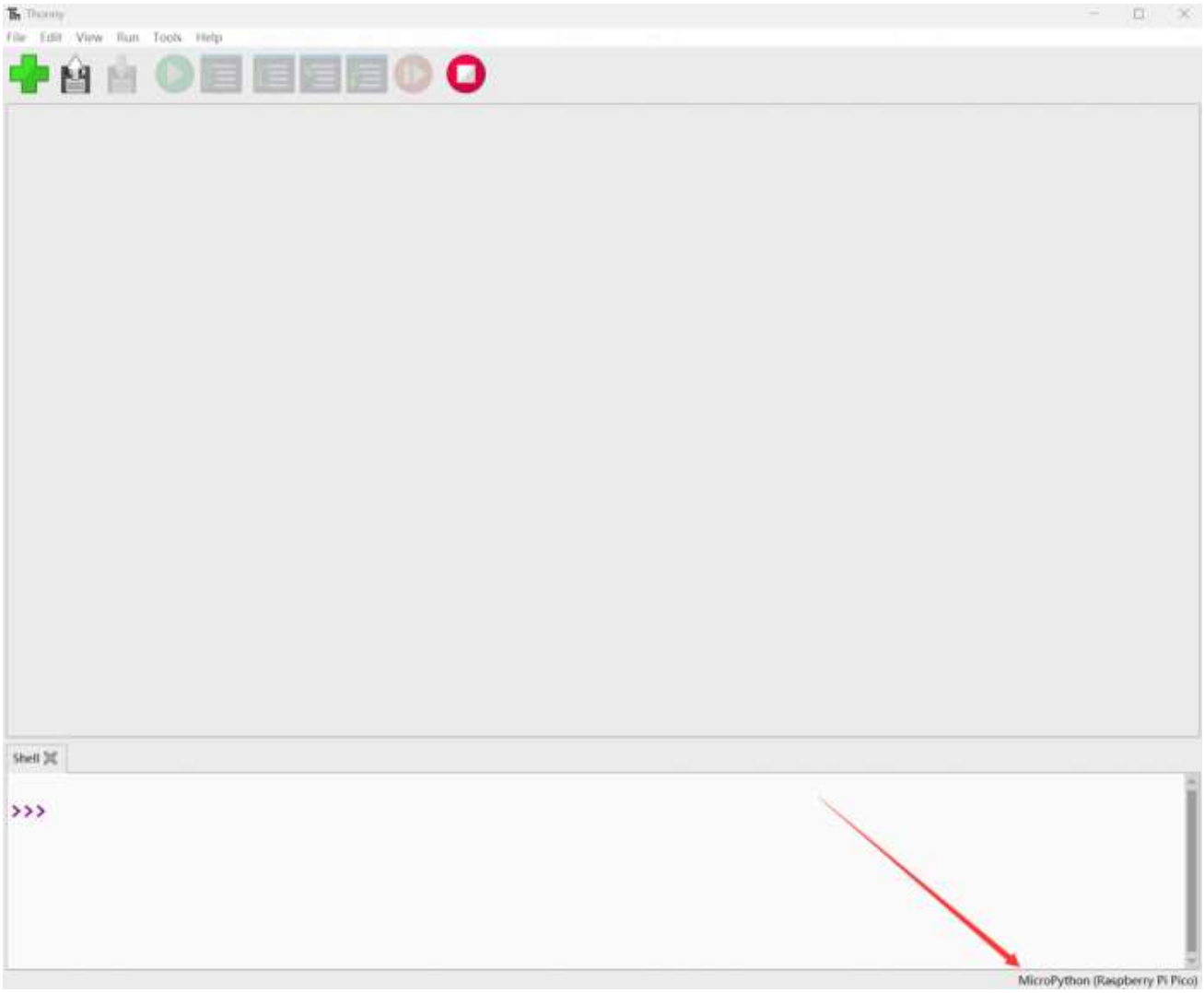
- Download Thonny IDE and follow the steps to install, the installation packages are all Windows versions, please refer to Thonny's official website for other versions
  - Thonny IDE official download link (https://github.com/thonny/thonny/releases/download/v3.3.3/thonny-3.3.3.exe)
  - Thonny IDE download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



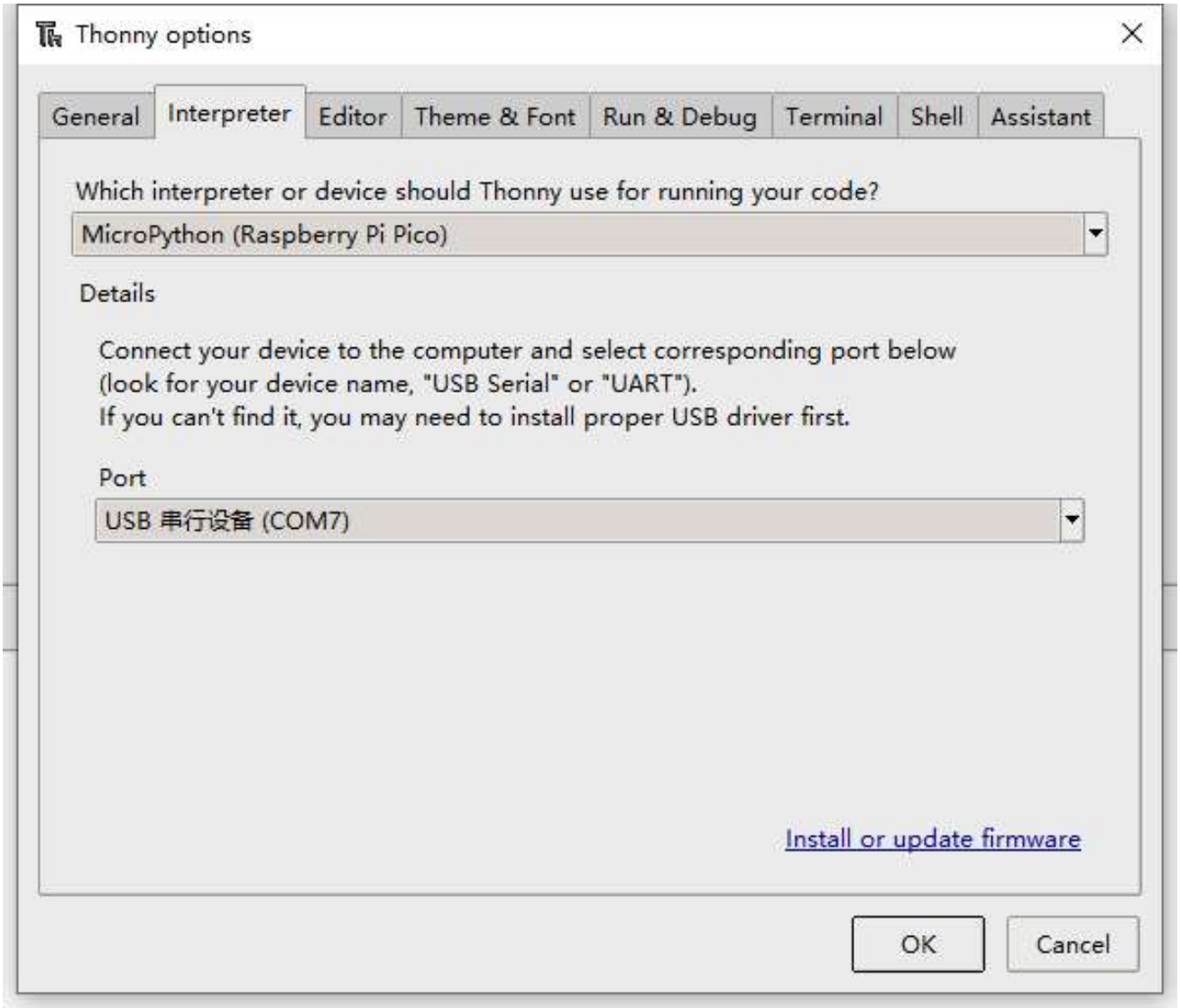
(/wiki/File:Pico-R3-Tonny1.png)

- 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 Configure interpreter

- In the pop-up window, select MicroPython (Raspberry Pi Pico), and choose the corresponding port



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



(/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 flash 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)
  - Pico2 16MB firmware library (https://files.waveshare.com/wiki/common/WAVESHARE-RP2350-20250807-v1.26.0-16MB.zip)
  - RP2350B firmware library (https://files.waveshare.com/wiki/common/WAVESHARE_RP2350B.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)

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

MicroPython Series Tutorials

【MicroPython】machine.Pin class function details (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91_Machine.Pin_Functions)

【MicroPython】machine.PWM class function details (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.PWM_Function)

【MicroPython】machine.ADC class function details (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.ADC_Function)

【MicroPython】machine.UART class function details (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.UART_Function)

【MicroPython】machine.I2C class function details (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.I2C_Function)

【MicroPython】machine.SPI class function details (https://www.waveshare.com/wiki/%E3%8

0%90MicroPython%E3%80%91machine.SPI_Function)

【MicroPython】rp2.StateMachine class function details (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91PIO_Function)

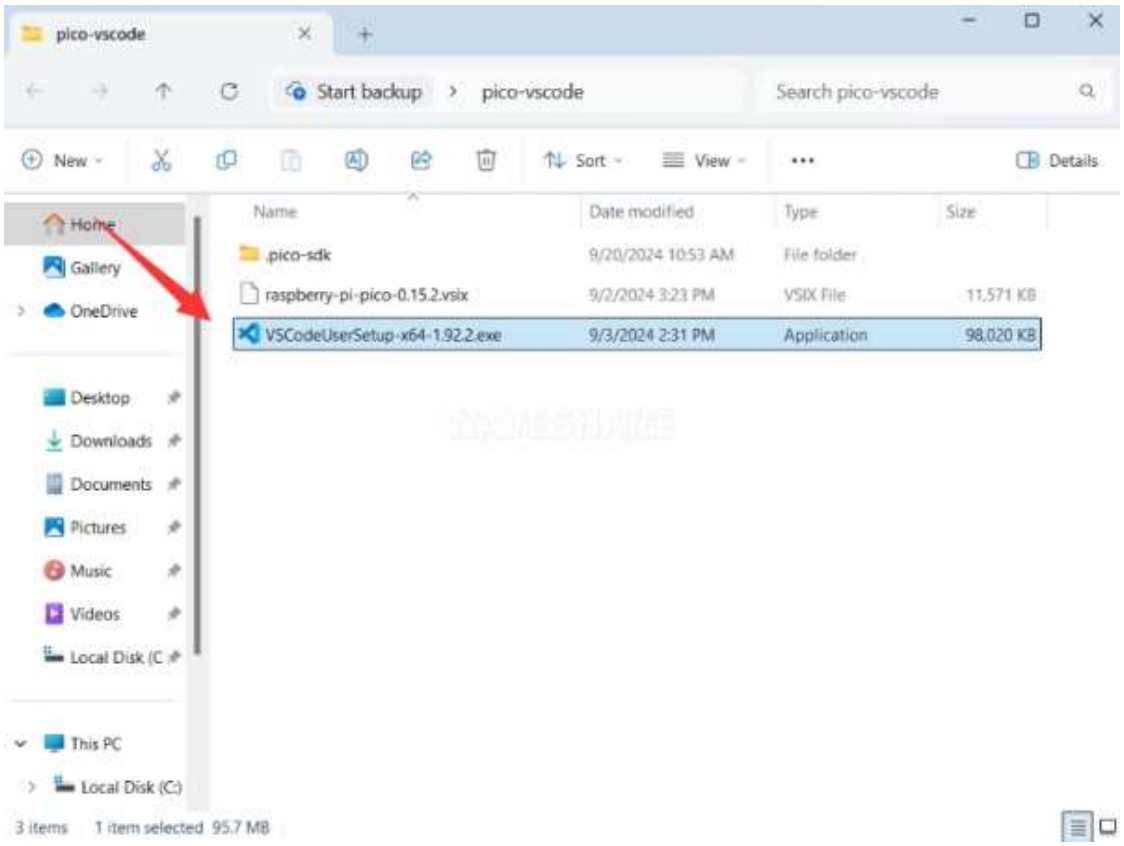
C/C++ Series

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

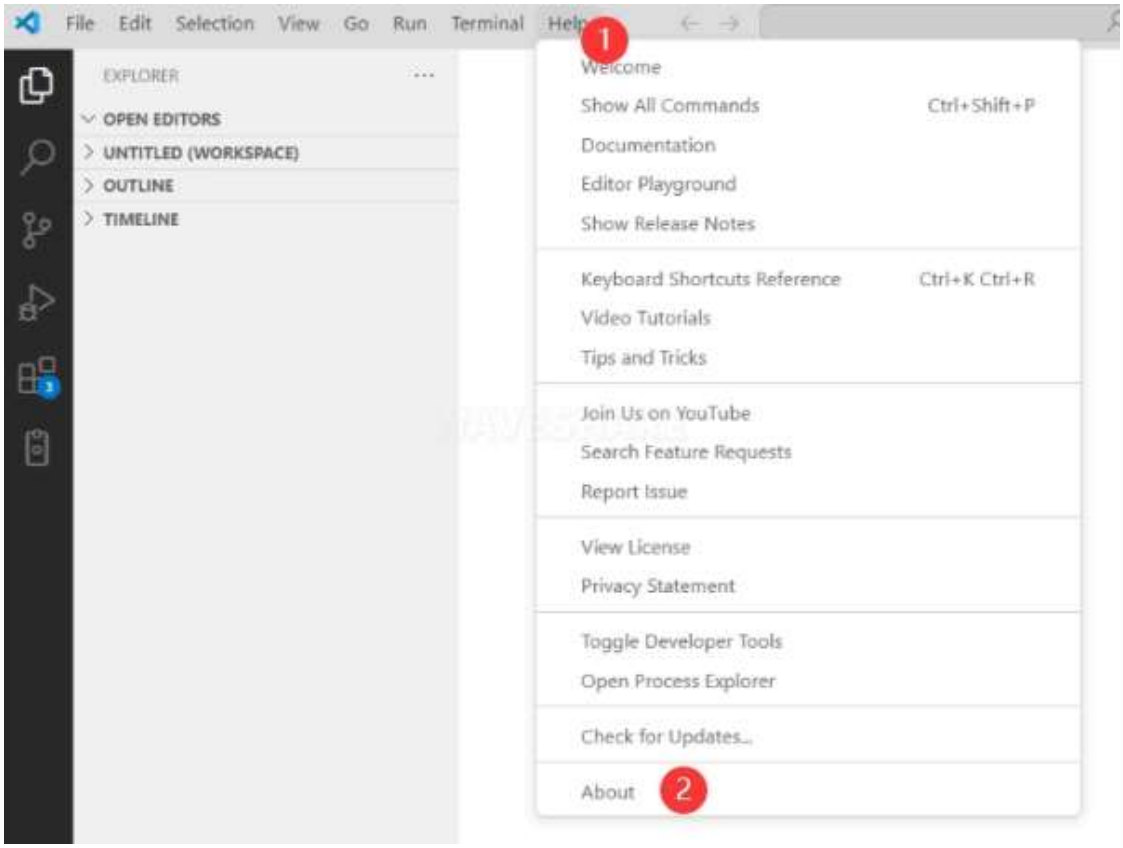
Install VSCode

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

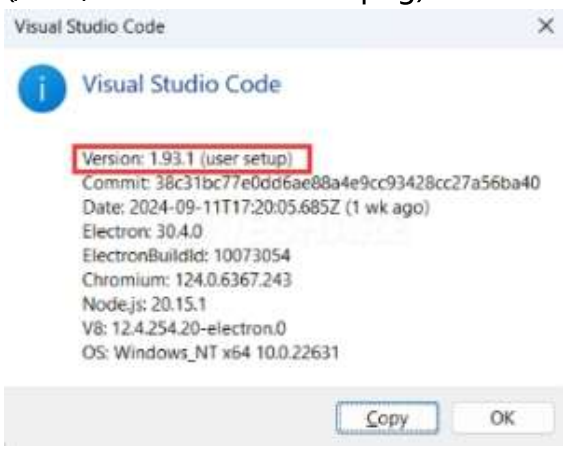


(/wiki/File:Pico-vscode-1.png)

Note: If vscode is installed, check if the version is v1.87.0 or later



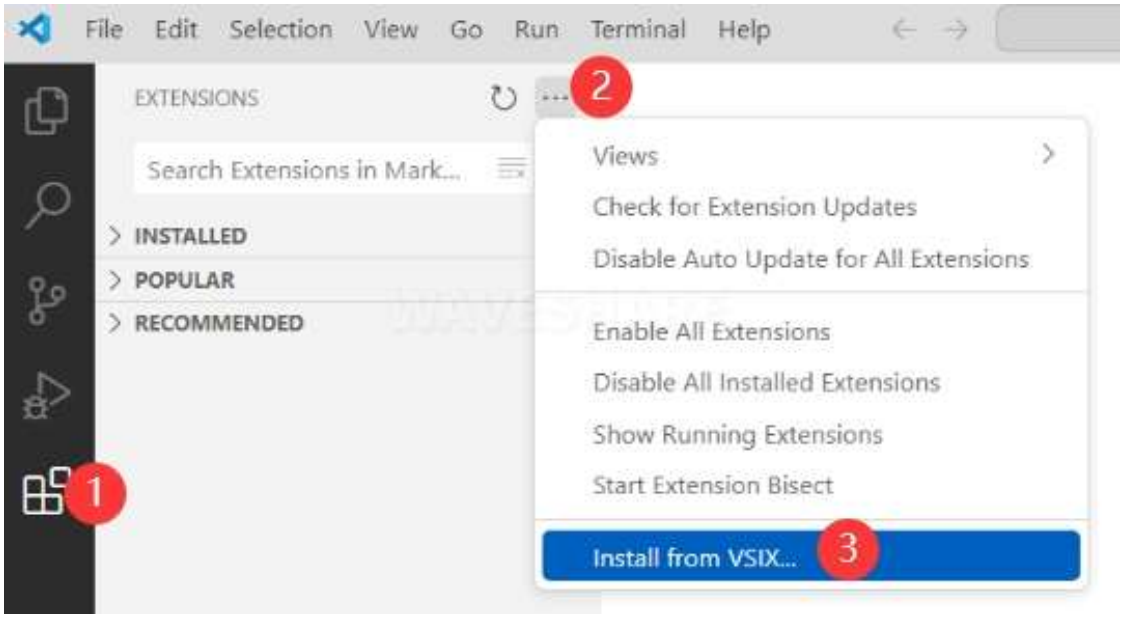
(/wiki/File:Pico-vscode-2.png)



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

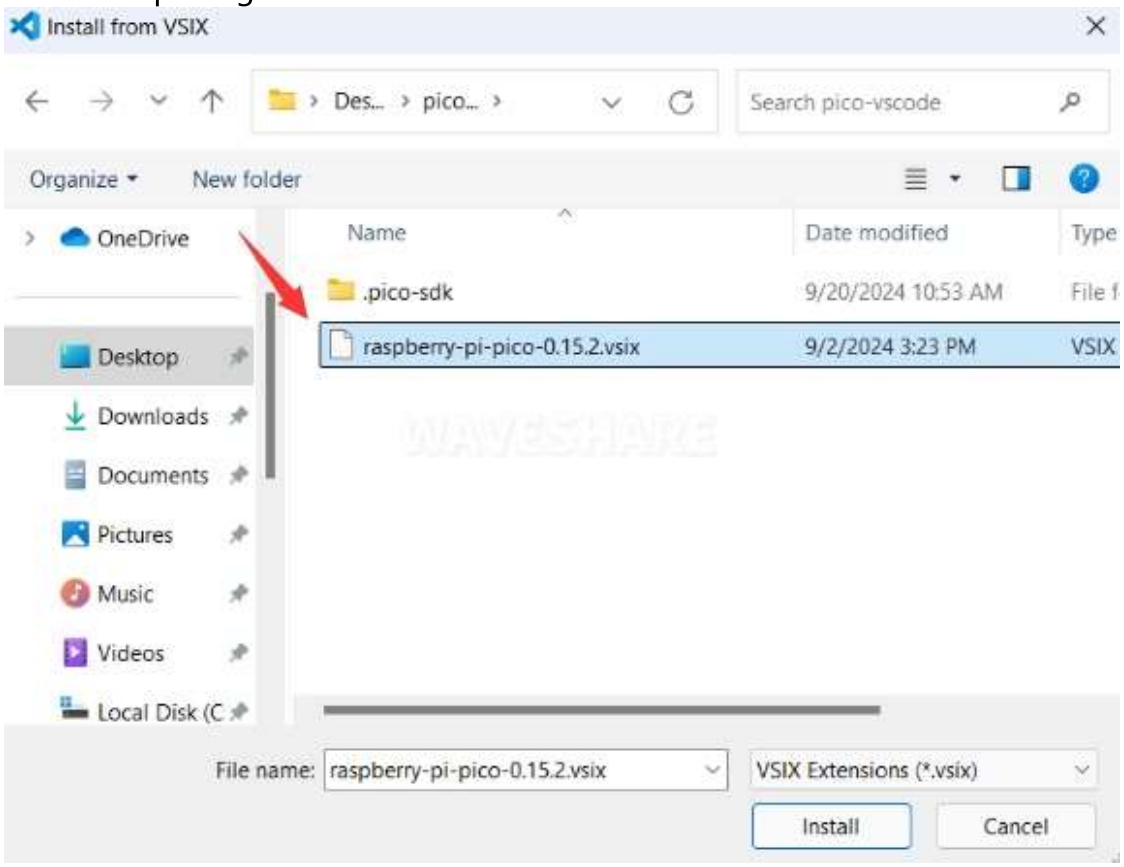
Install Extension

1. Click Extensions and select Install from VSIX



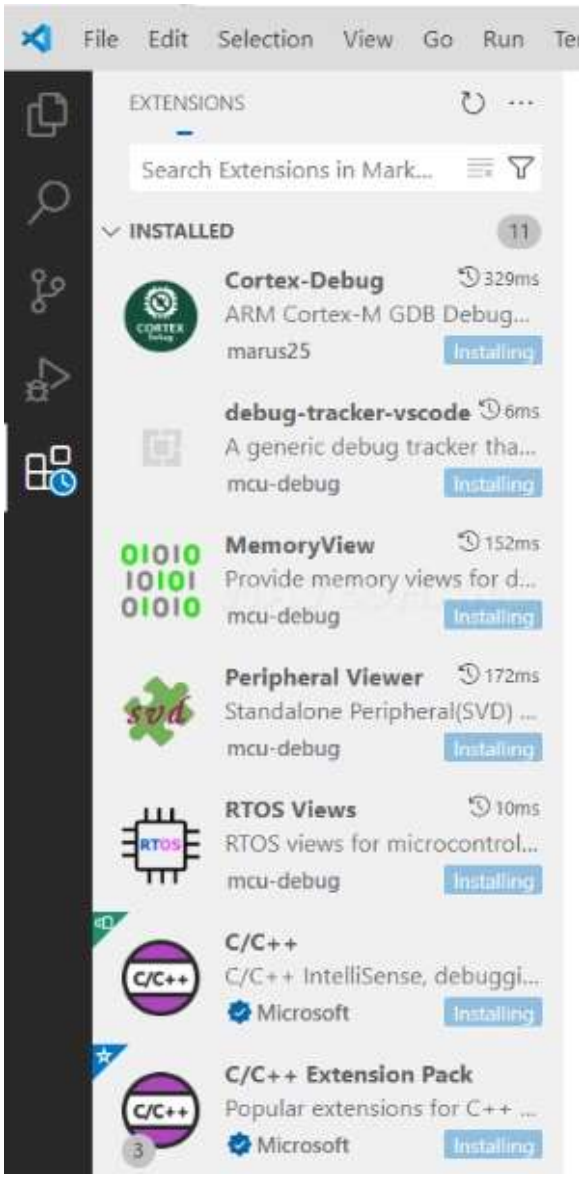
(/wiki/File:Pico-vscode-4.png)

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



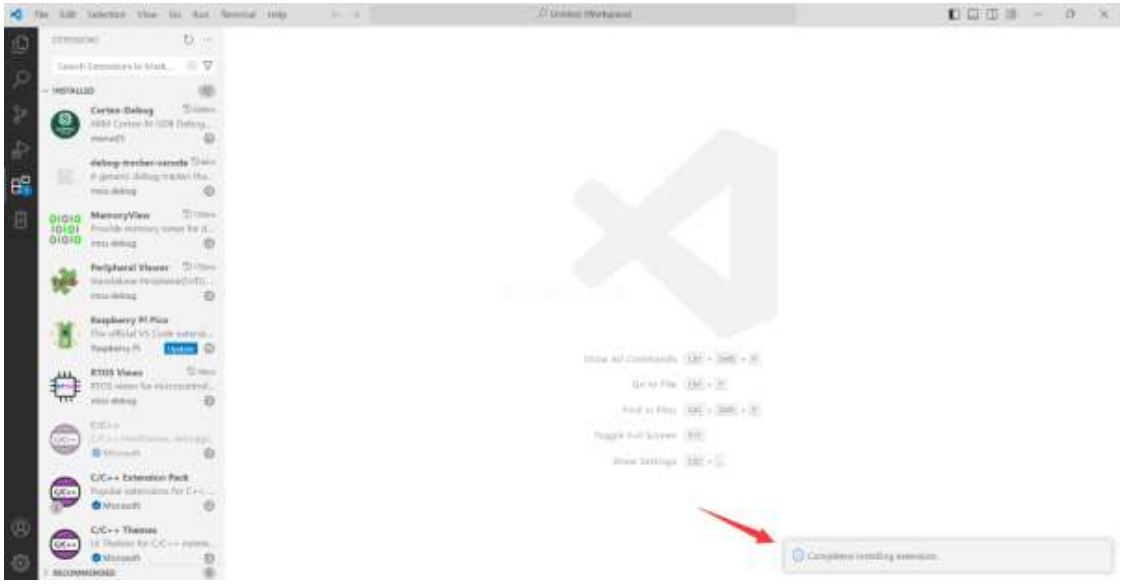
(/wiki/File:Pico-vscode-5.png)

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



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

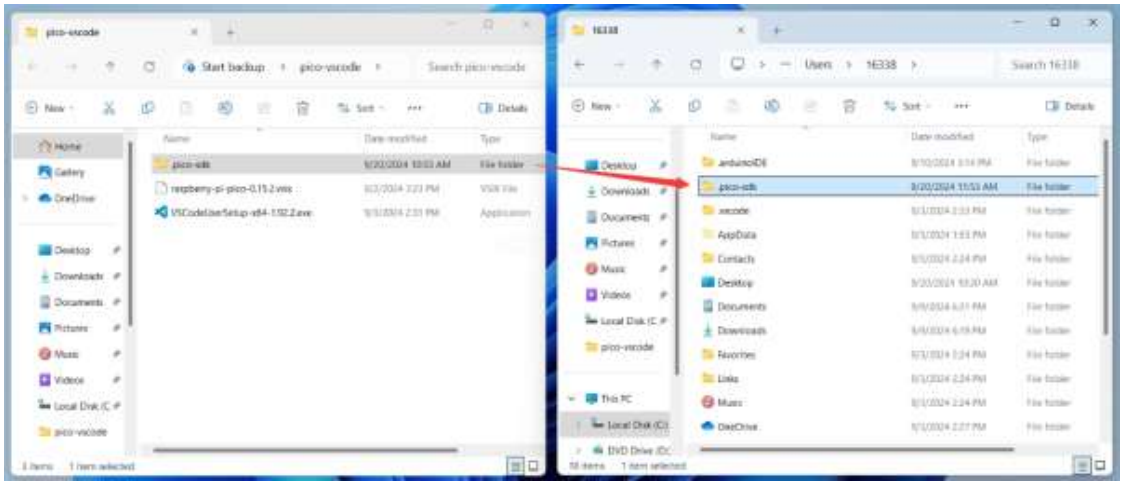
4. The text in the right lower corner shows that the installation is complete. Close VSCode



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

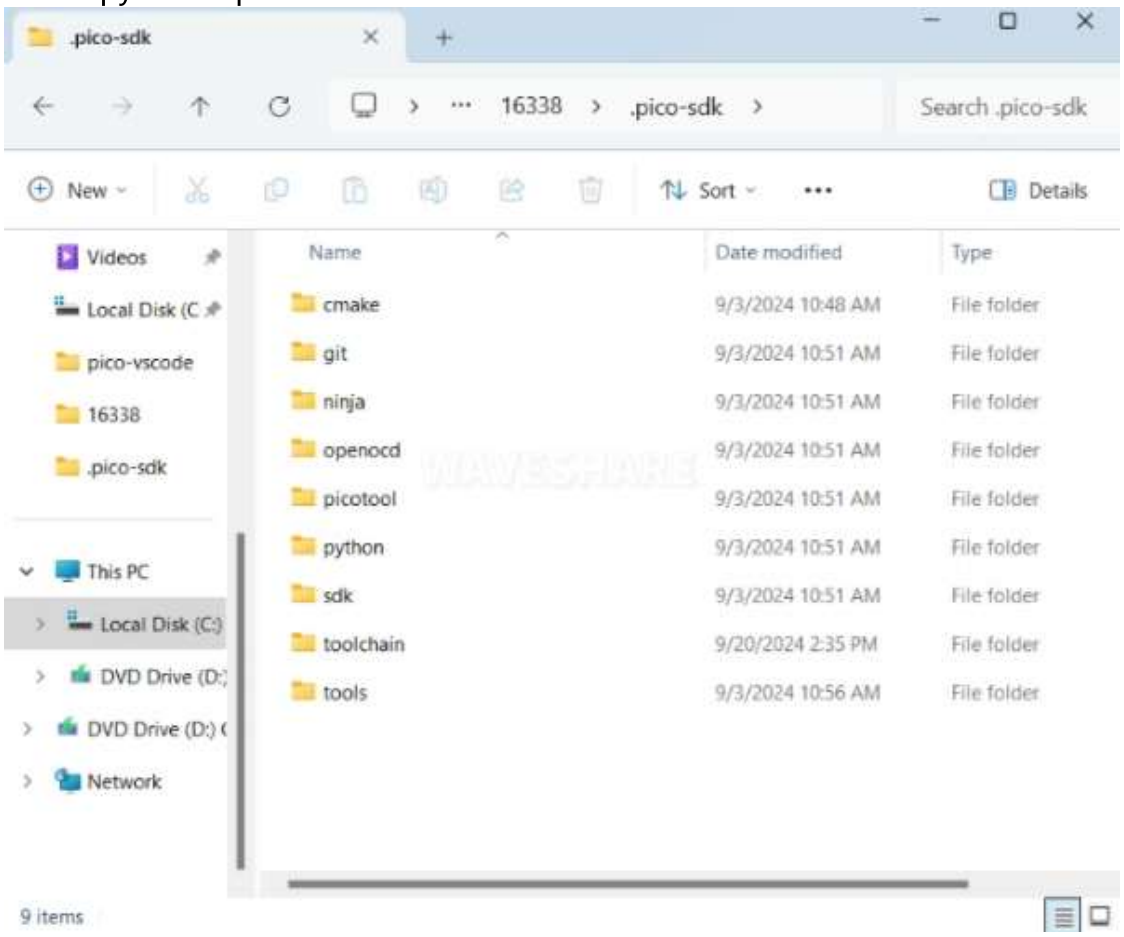
Configure Extension

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



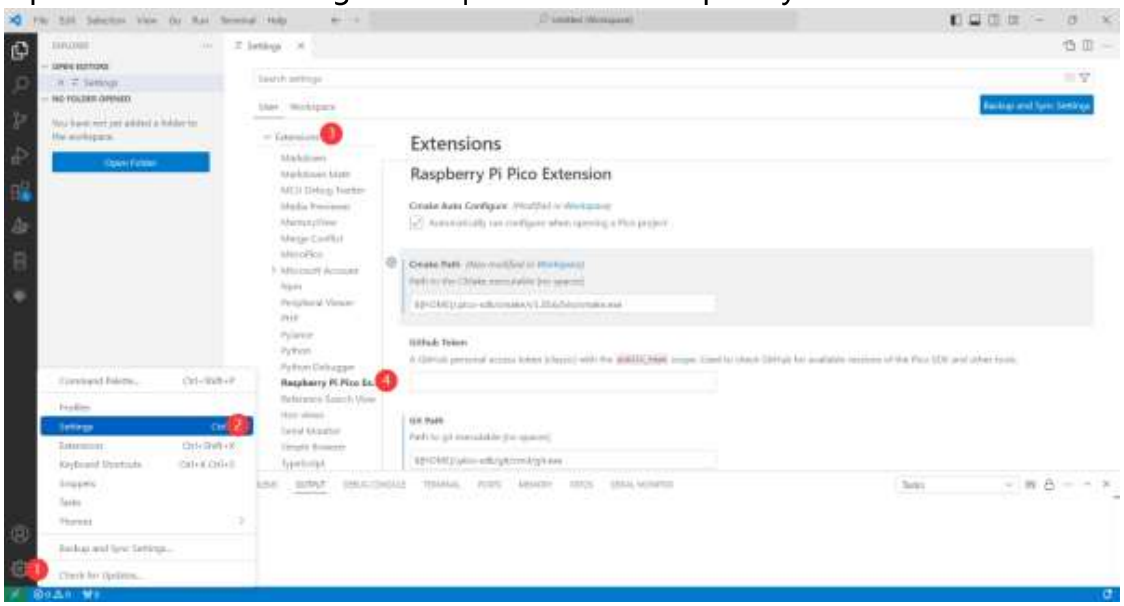
(/wiki/File:Pico-vscode-8.png)

2. The copy is completed



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

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

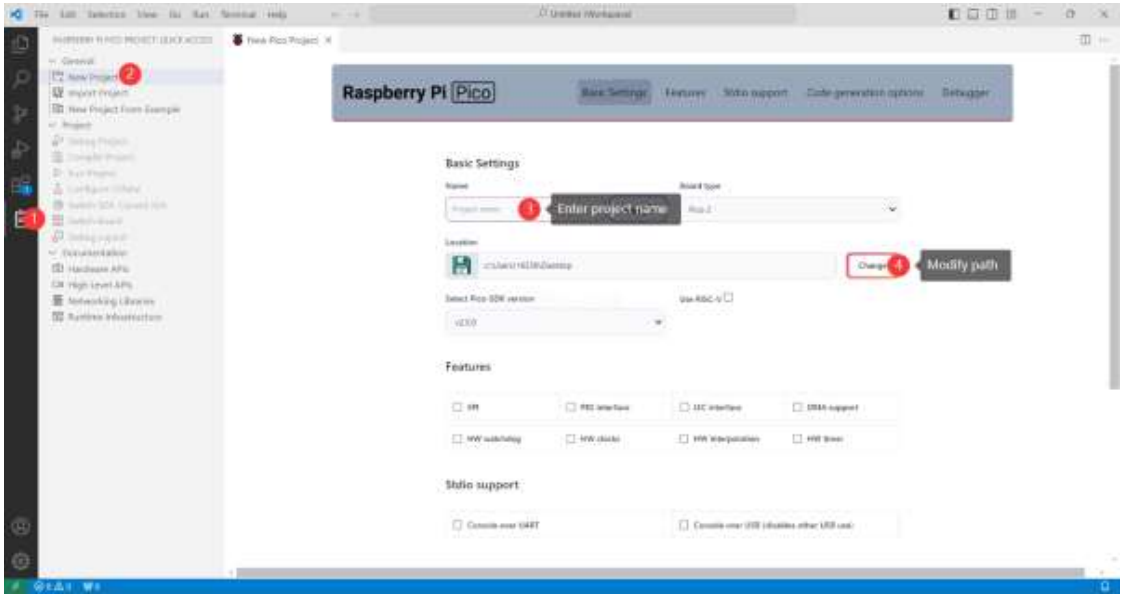


(/wiki/File:Pico-vscode-10.png)
The configuration is as follows:

```
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
```

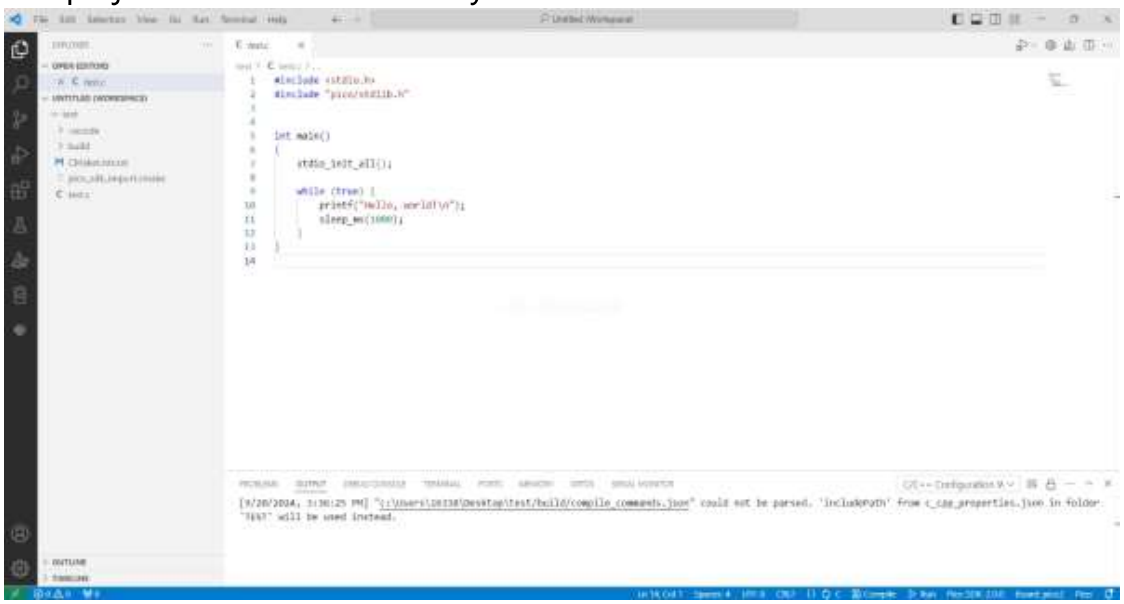
New Project

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



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

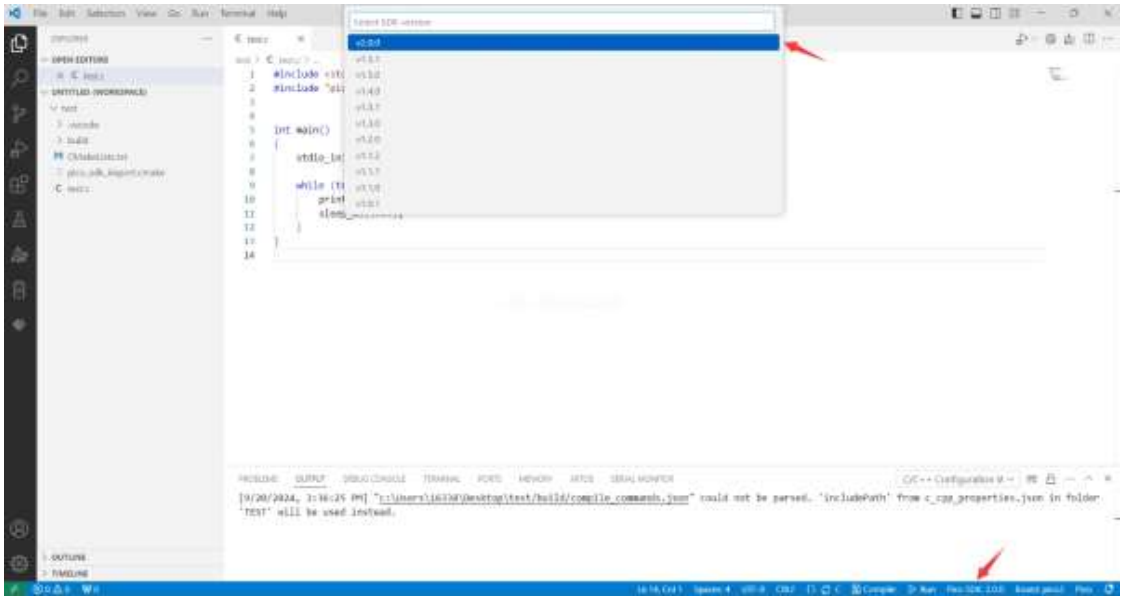
2. The project is created successfully



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

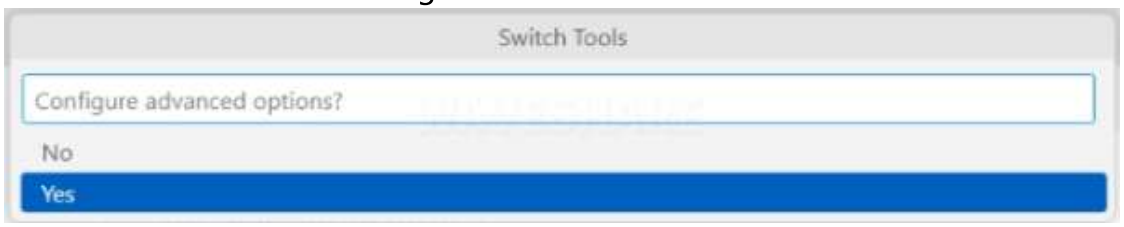
Compile Project

1. Select the SDK version



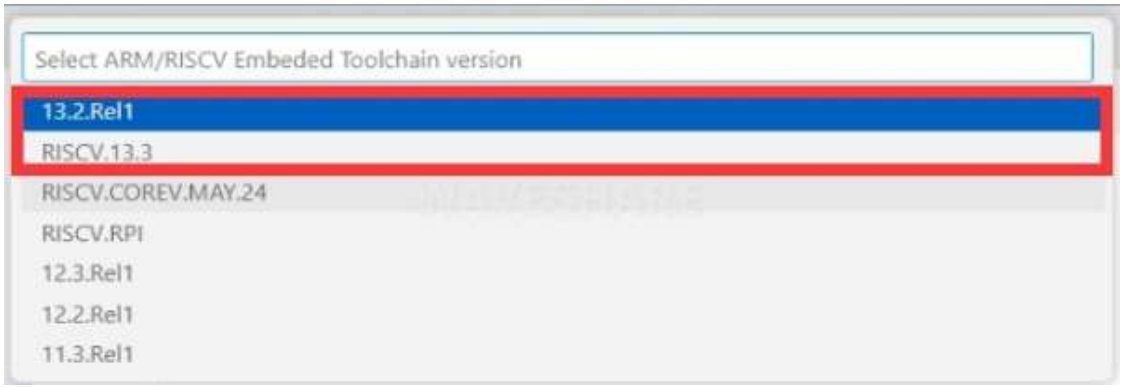
(/wiki/File:Pico-vscode-13.png)

2. Select Yes for advanced configuration



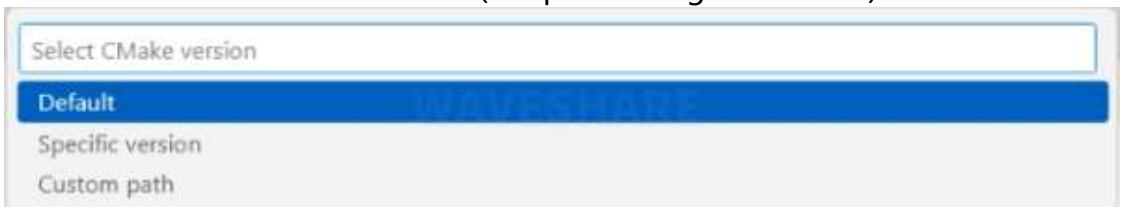
(/wiki/File:Pico-vscode-14.png)

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



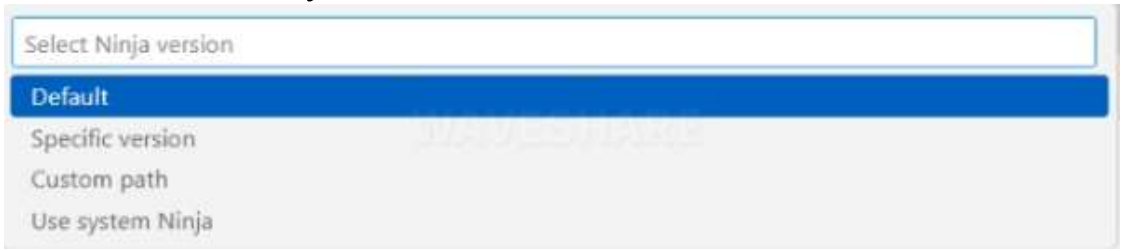
(/wiki/File:Pico-vscode-15.png)

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



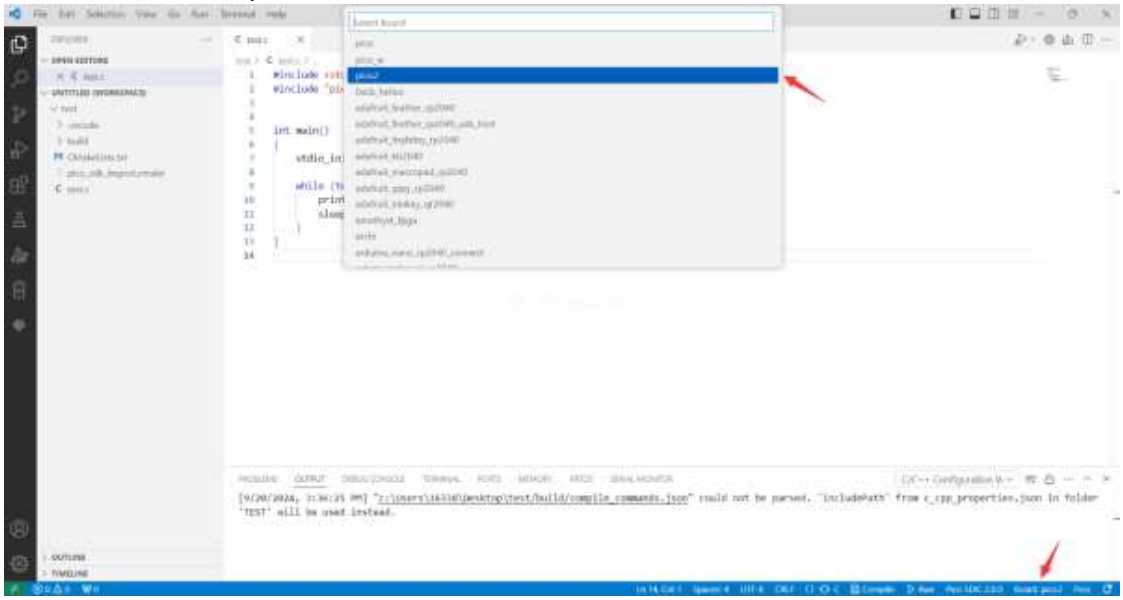
(/wiki/File:Pico-vscode-16.png)

5. Select Default for Ninja version



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

6. Select the development board

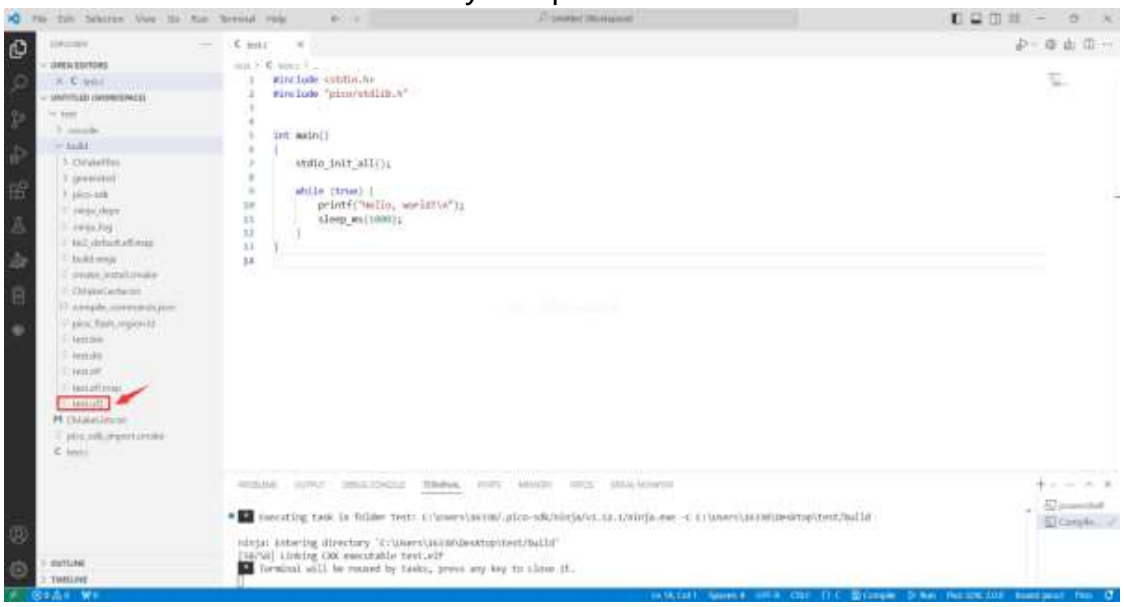


(/wiki/File:Pico-vscode-18.png)

7. Click Compile to compile

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

8. The .uf2 format file is successfully compiled

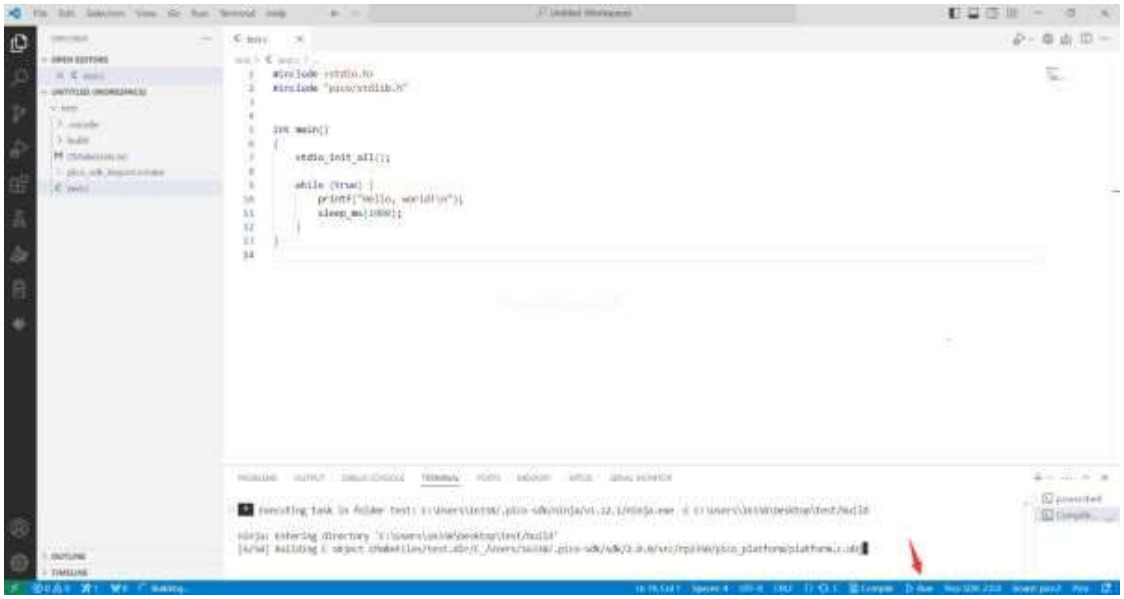


(/wiki/File:Pico-vscode-20.png)

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



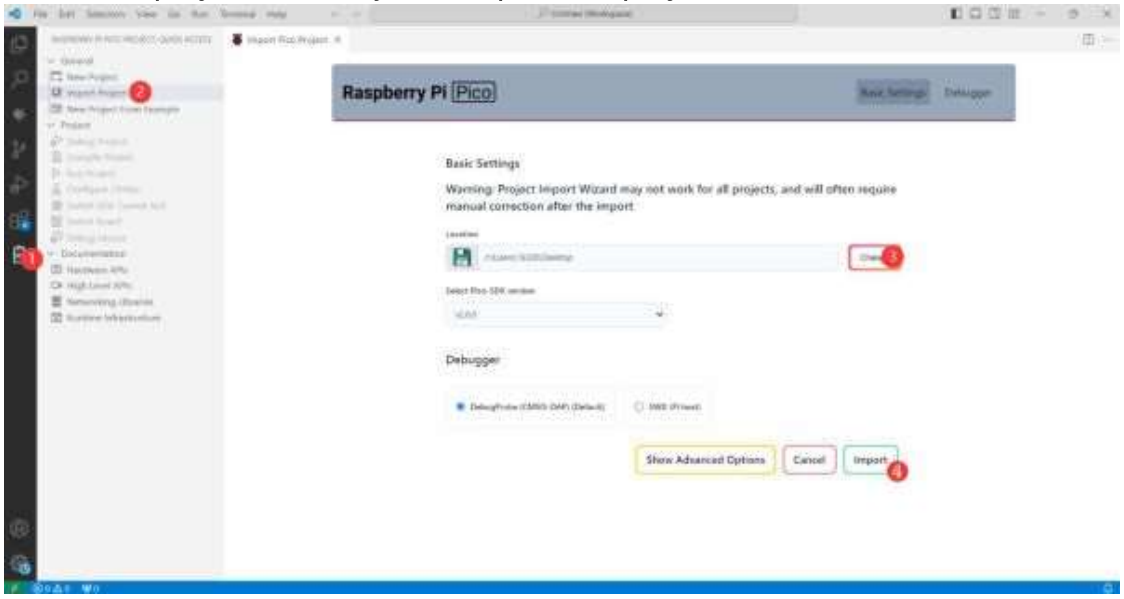
(/wiki/File:Pico-vscode-24.jpg)

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, ind
icating successful program flashing.
```

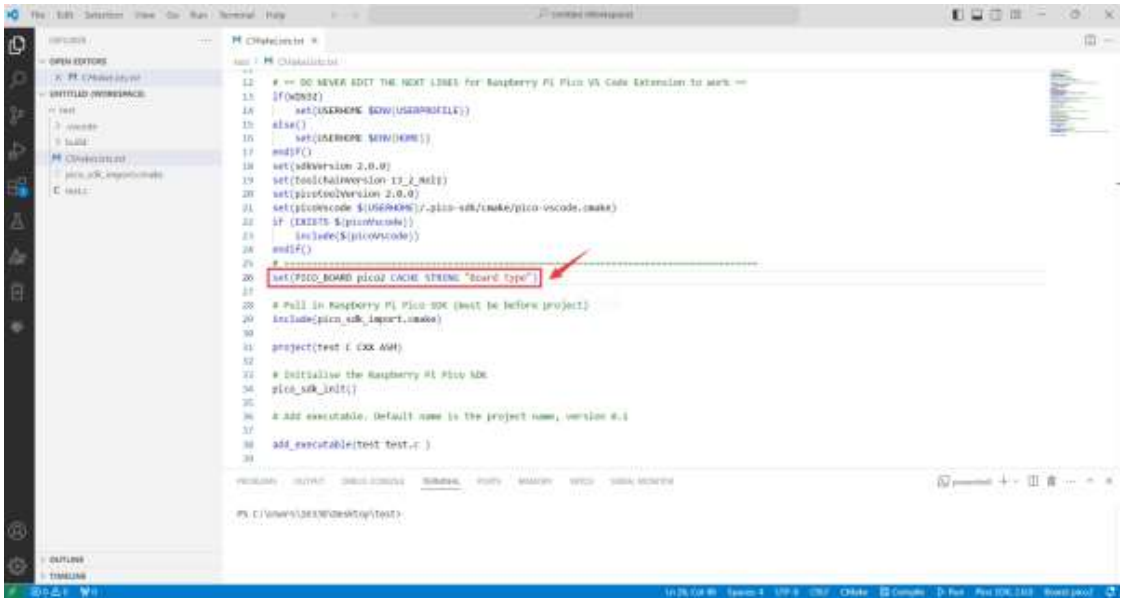
Import Project

1. Select the project directory and import the project



(/wiki/File: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

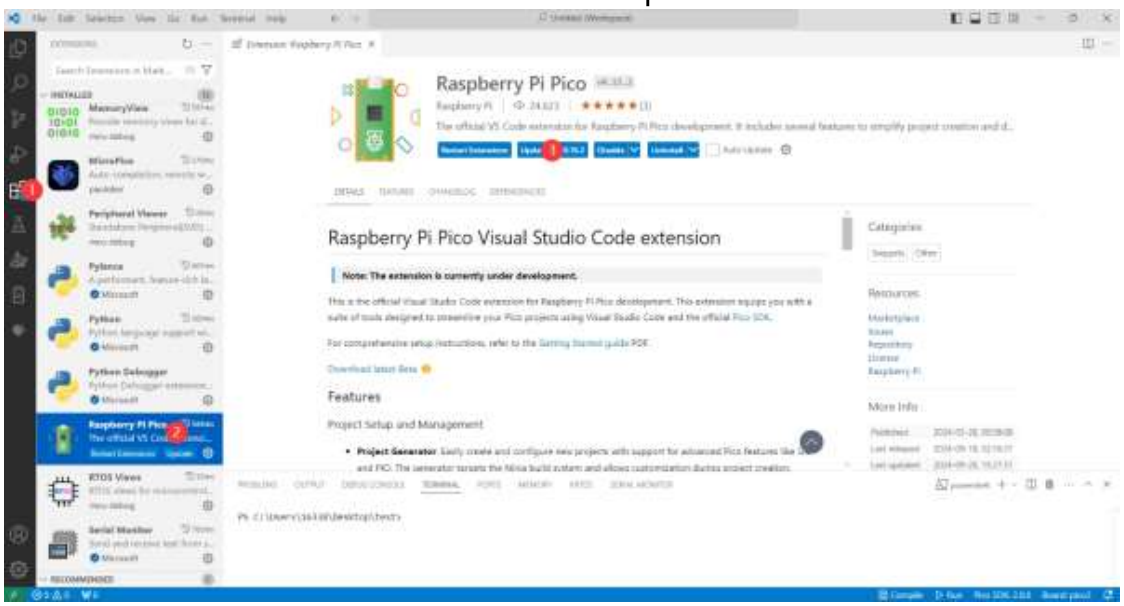


(/wiki/File:Pico-vscode-21.png)

```
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

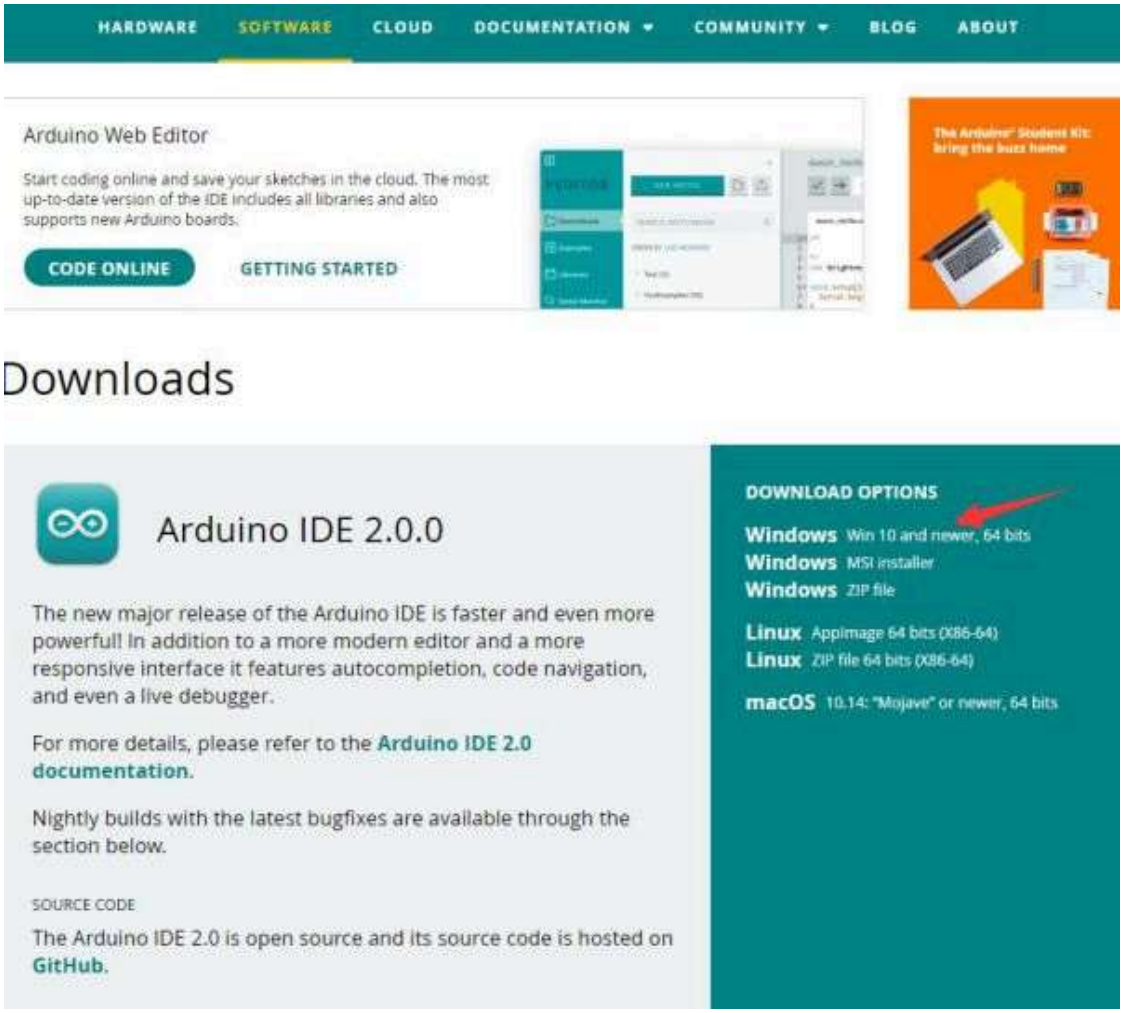


(/wiki/File:Pico-vscode-22.png)

Arduino IDE Series

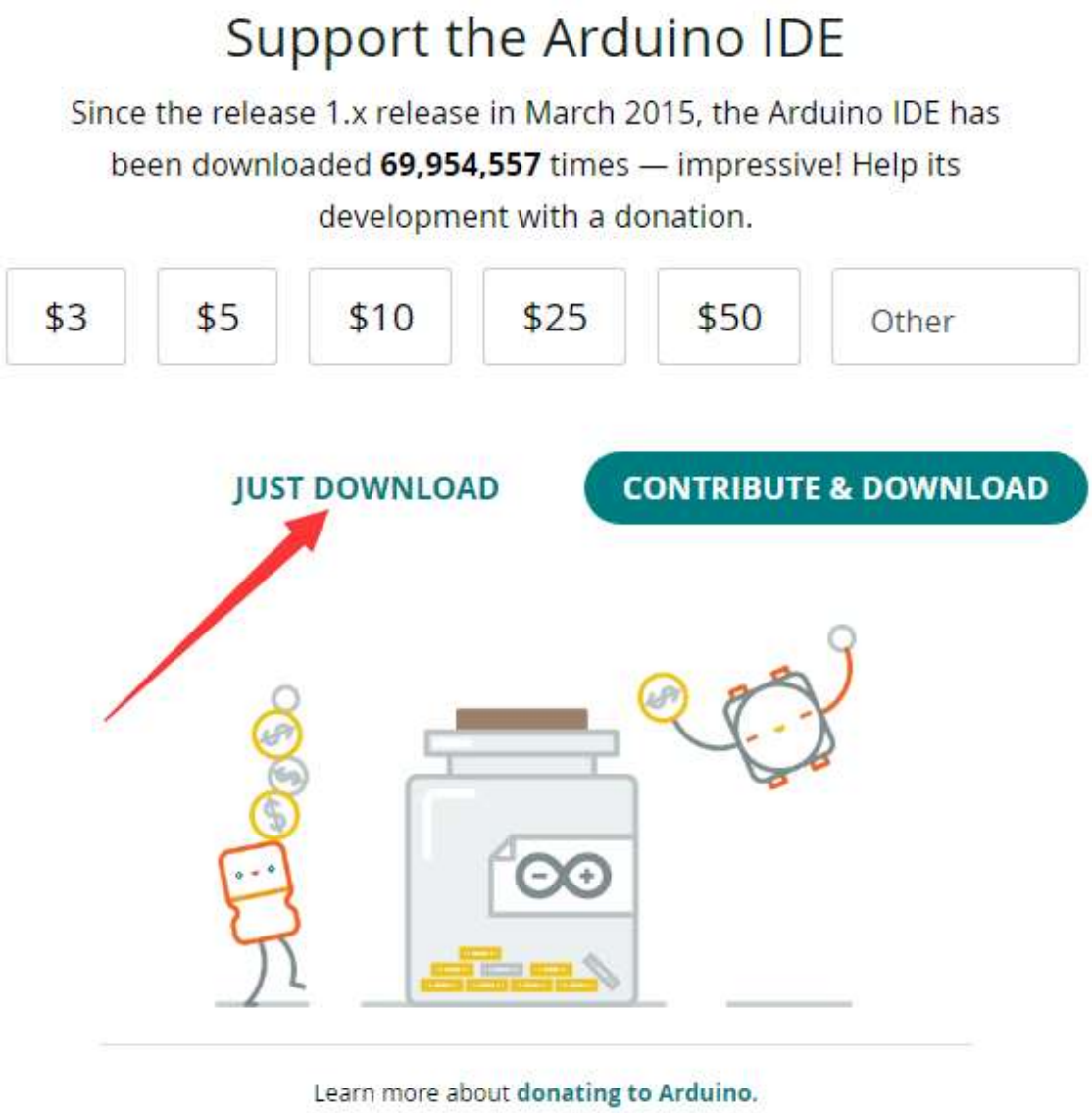
Install Arduino IDE

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



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

2. Here, you can select Just Download.



(/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.

(/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

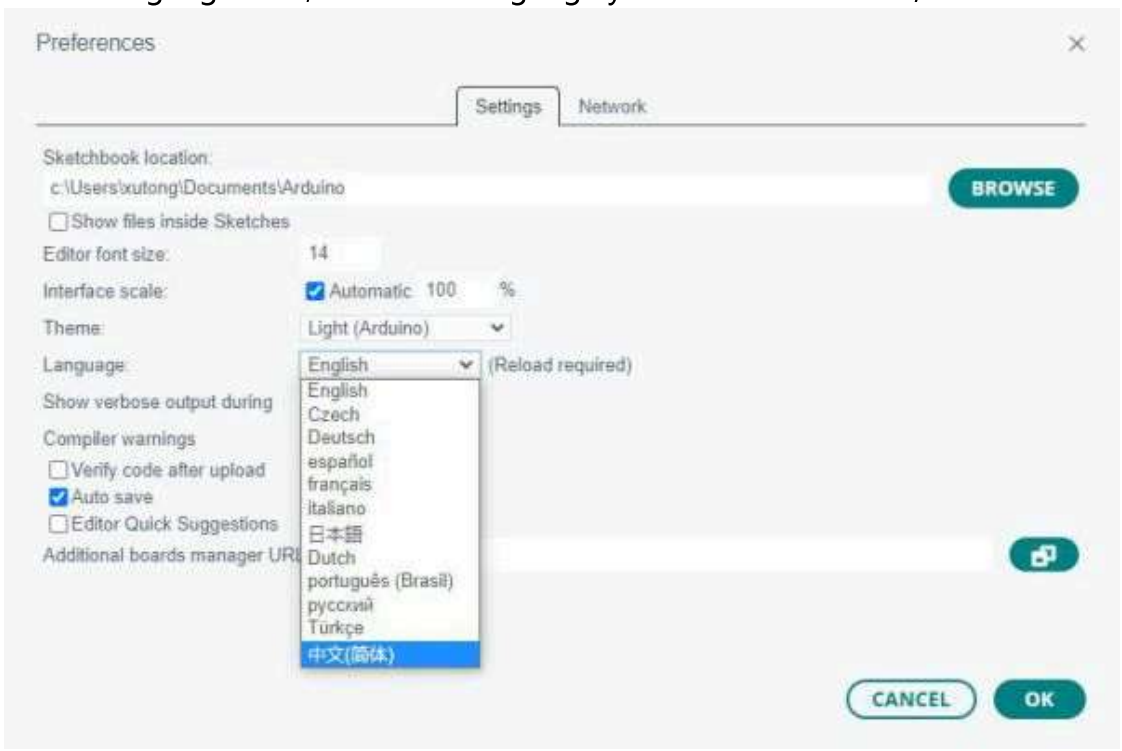
Arduino IDE Interface

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.



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

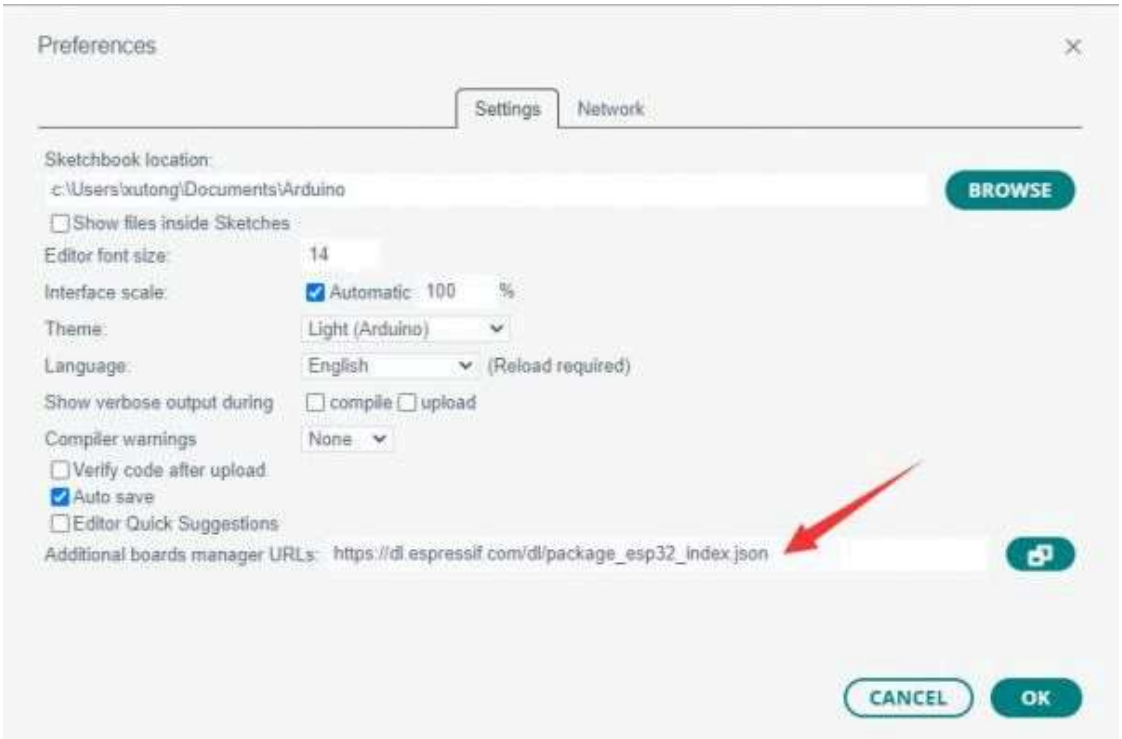
Install Arduino-Pico Core in Arduino IDE

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
```

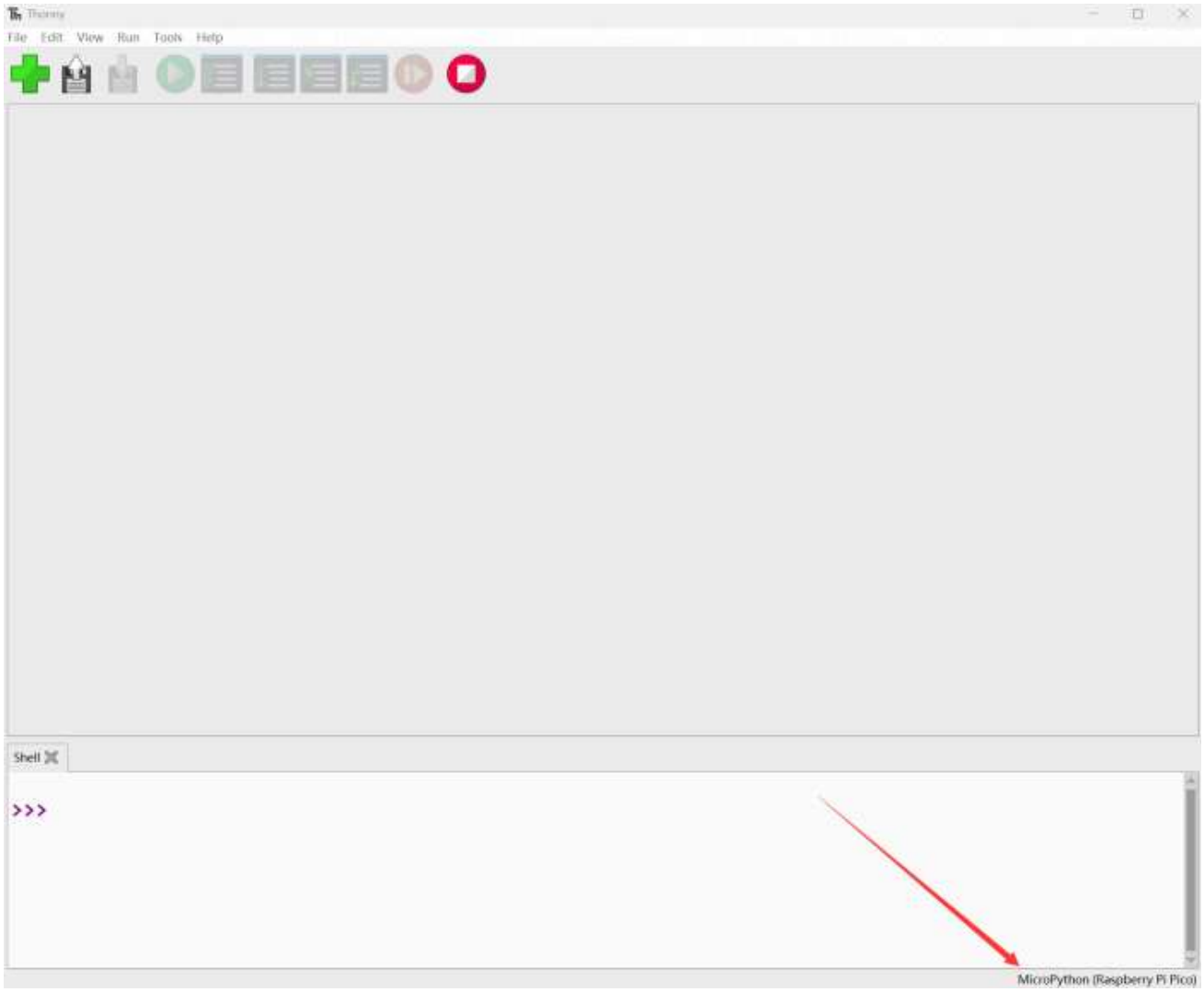


(/wiki/File:RoArm-M1_Tutorial_II05.jpg)

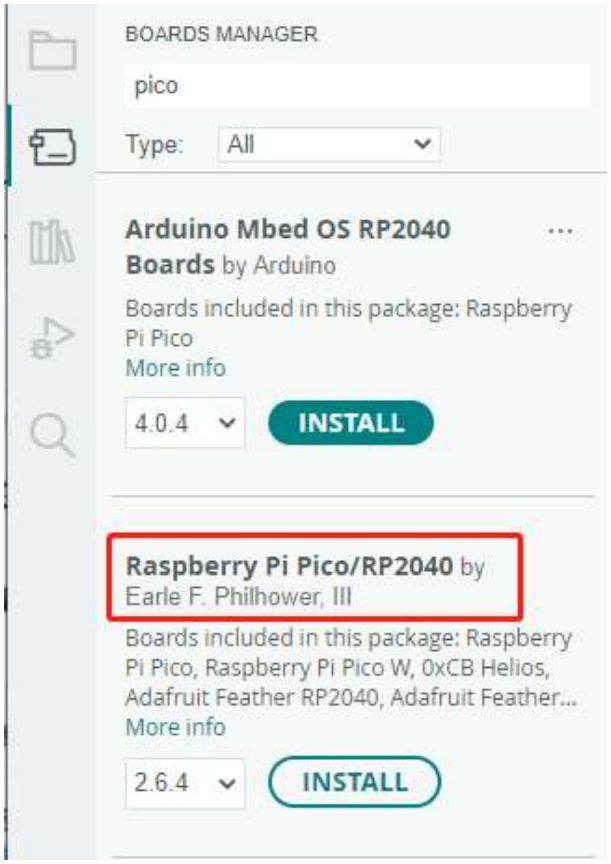
Note: If you already have an ESP32 board URL, you can use a comma to separate the URLs as follows:

```
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



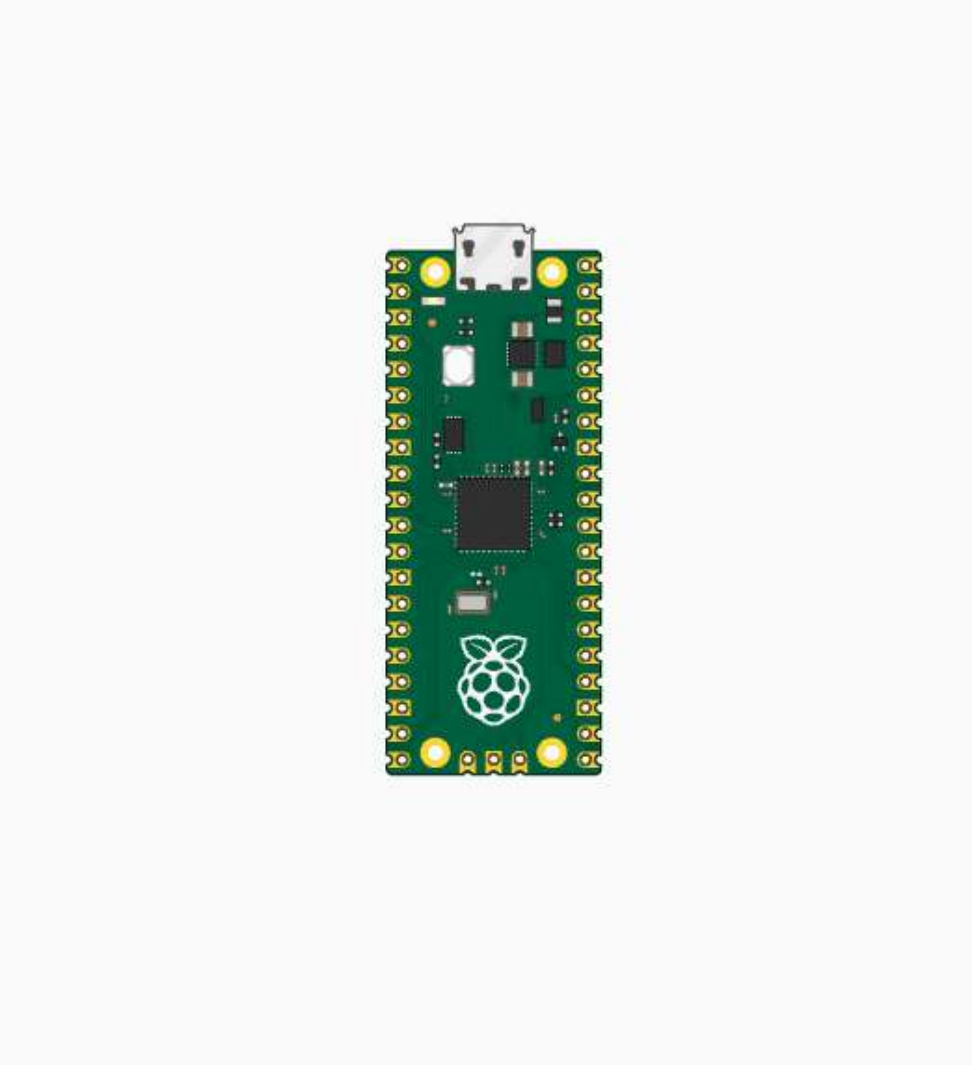
(/wiki/File:Pico_Get_Start_05.png)



(/wiki/File:Pico_Get_Start_06.png)

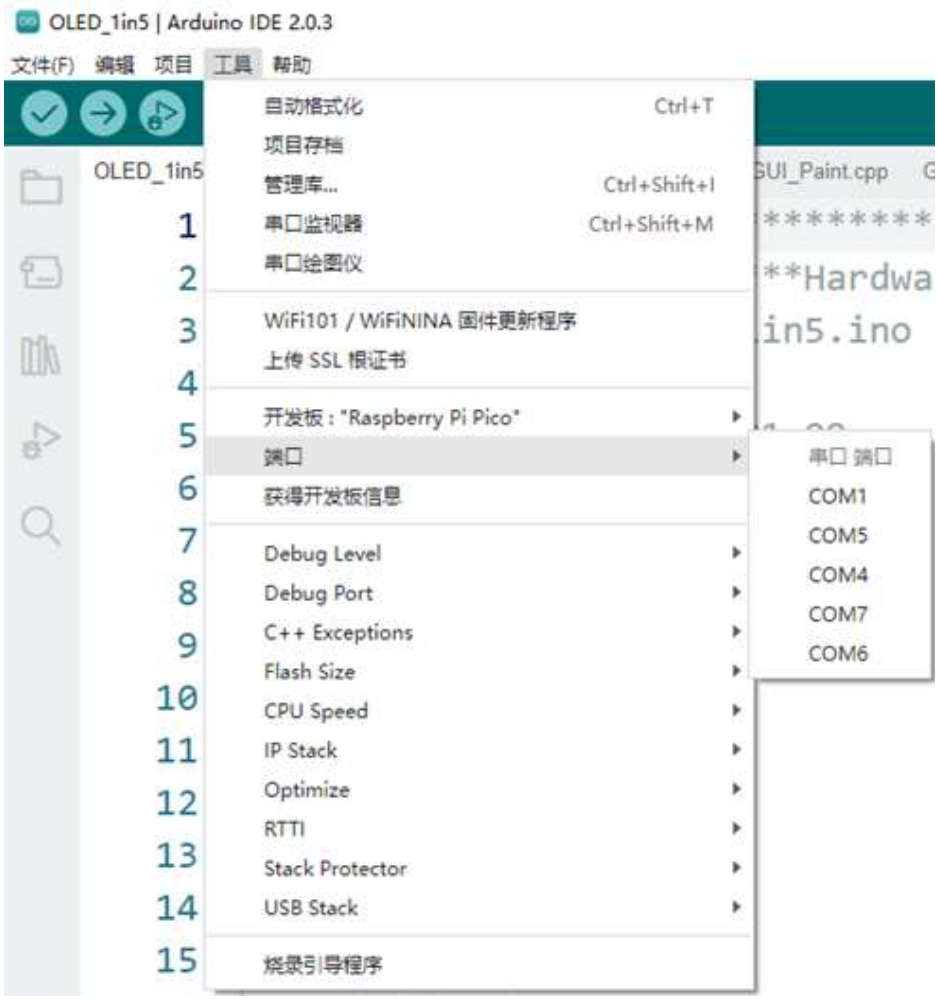
Upload Demo at the First Time

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)

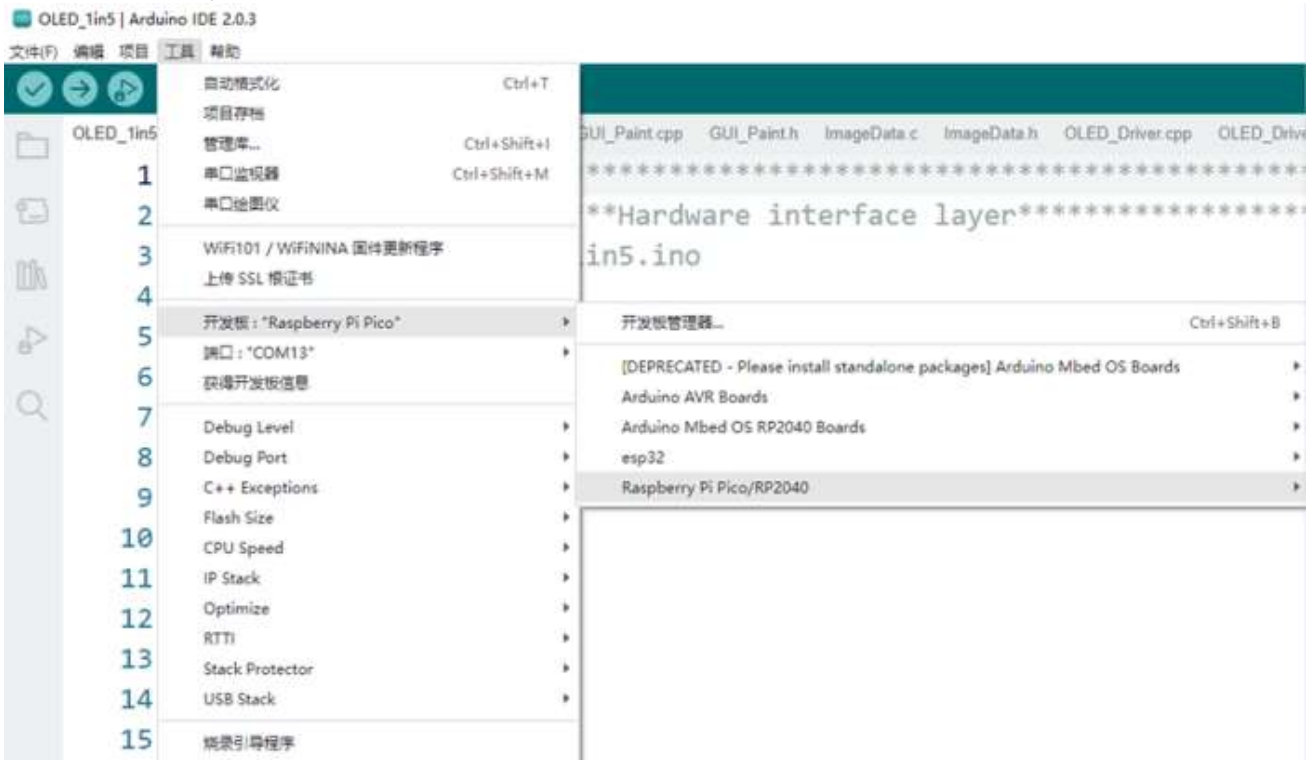


(/wiki/File:Pico%E8%BF%9E%E6%8E%A5%E5%89%8D%E7%AB%AF%E5%8F%A3.png)

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 > Corresponding models (Raspberry Pi Pico, Raspberry Pi Pico 2, etc.)



(/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)

Demo

C/C++ Demo

01_RS485

Demo description

- This example implements the control of six relay switches through RS485

Code analysis

- **Relay_Control()**: This function mainly executes the corresponding relay control operation and outputs the corresponding status prompt information according to the array index passed through the parameters

- **Parameter analysis**
  - `uint8_t index`: In the `main` function, the program will parse the received RS485 data and match it with the preset control command array. When a match is found, the `Relay_Control` function is called, and the index position of the matching command in the array is passed via `index` parameter
- **Logical flow**
  - Use the `if else` statement to perform different operations based on the value of `index`:
    - In the case of `index < 6`, that is, for the `CH1~CH6` commands, the `DEV_Digital_Write` function is used to switch the level status of the corresponding GPIO pin (such as `RELAY1_PIN`, etc.), and at the same time, the corresponding `relay_status` array element is updated to record the change of the relay status, and the corresponding on or off prompt information is output according to the final state of the relay. Finally, in the `main` function, the `Beep` function is called to control the buzzer.
    - For the case of `index = 6`, i.e., for the `ALL_ON` command, all GPIO pins (corresponding to 6 channel relays) are set to high level (on state). Using the `memset` function, all elements of the `relay_status` array are set to 1, indicating that all relays are turned on. A message indicating that all relays are turned on is then output. Finally, in the `main` function, the `Beep` function is called to control the buzzer.
    - For the case of `index = 7`, i.e., for the `ALL_OFF` command, all GPIO pins (corresponding to 6 channel relays) will be set to low level (off state). Using the `memset` function, all elements of the `relay_status` array will be set to 0, indicating that all relays are off. A message indicating that all relays are off will be output. Finally, in the `main` function, the `Beep` function will be called to control the buzzer.
    - If the value of `index` does not match the above commands, the prompt message of receiving non-command data will be output

RS485 Control

- Jump to RS485 Control Tutorial (https://www.waveshare.com/wiki/ESP32-S3-Relay-6CH-RS485)

02_MQTT

Demo description

- This demo only supports the RP2350-Relay-6CH-W
- This demo achieves network communication through a WiFi connection and remotely controls 6 relays via the MQTT protocol

Precautions

- Devices need to be created in Waveshare Cloud
- The following key parameters need to be modified:
  - WiFi connection parameters:
    - `WIFI_SSID` : WiFi network name
    - `WIFI_PASSWORD` : WiFi password
  - MQTT core parameters:
    - `MQTT_CLIENT_ID` : Device ID
    - `MQTT_SERVER` : Server address
    - `MQTT_PUB_TOPIC` : Release a topic
    - `MQTT_SUB_TOPIC` : Subscribe to a topic

Code analysis

- Function description
  - `Relay_Init()` : Initialize relay control GPIO
  - `start_client()` : Create an MQTT client and connect to the server
  - `sub_unsub_topics()` : Subscribe/unsubscribe from the topic
  - `mqtt_incoming_data_cb()` : JSON message parsing and relay control core logic
- Logical flow
  - **Device initialization**:
    - Call `Relay_Init()` to initialize GPIO
    - Call `cyw43_arch_wifi_connect_timeout_ms()` to connect to WiFi
  - **MQTT connection**:
    - Call `mqtt_client_connect()` to establish an MQTT connection
    - Call `sub_unsub_topics()` to subscribe to a control topic
  - **Control logic**:
    - Receive JSON formatted messages (e.g., {data:{CH1:1}}) through the callback function `mqtt_incoming_data_cb()`
    - Parse channel name (CH1~CH6 or ALL) and control value (0/1)
    - Call `gpio_put()` to control the corresponding relay
    - After successfully controlling the relay, call `mqtt_publish()` to publish the update status to the MQTT server

Waveshare Cloud Control

- Jump to Waveshare Cloud Control (https://www.waveshare.com/wiki/ESP32-S3-POE-ETH-8DI-8RO-Waveshare_Cloud)

03_BLE

Demo description

- This demo only supports the RP2350-Relay-6CH-W
- This demo achieves wireless communication via Bluetooth Low Energy (BLE) to remotely control 6 relays

Precautions

- Device broadcast name: RP2350-Relay-6CH-W
- Use custom UUID service: 0000FF01-0000-1000-8000-00805F9B34FB
- Control command format: <channel name> <control value>

Code analysis

- Function description
  - `Relay_Init()` : Initialize relay control GPIO
  - `hci_power_control()` : Turn on Bluetooth
  - `packet_handler()` : Bluetooth protocol stack event handling core
  - `att_write_callback()` : Command parsing and execution core logic
- Logical flow
  - **Device initialization**:
    - Call `Relay_Init()` to initialize relay GPIO
    - Call `hci_power_control()` to start the BLE protocol stack
  - **Establish connection**:
    - Open the Bluetooth serial port tool, find the device broadcast name and UUID
    - Enable notification feature after client connection
  - **Control logic**:
    - Receive text commands (e.g., CH1 1) in the callback function `att_write_callback()`
    - Parse channel name (CH1~CH6 or ALL) and control value (0/1)
    - Call `gpio_put()` to perform GPIO control
    - Return the operation result to the client through `att_server_request_can_send_now_event()`

Example of control commands

- Open channel 1: `CH1 1`
- Close channel 2: `CH2 0`
- Open all channels: `ALL 1`
- Close all channels: `ALL 0`

Example of error feedback

- Invalid command: `ERR: Cmd Invalid`
- Invalid channel: `ERR: CH 1-6 Only`

- Invalid value: `ERR: Need Value 0/1`

MicroPython Demo

01_RS485

Demo description

- This example implements the control of six relay switches through RS485

Code analysis

- **relay_control()**: This function mainly executes the corresponding relay control operation and outputs the corresponding status prompt information according to the array index passed through the parameters
  - **Parameter analysis**
    - `index`: In the `main` function, the program will parse the received RS485 data and match it with the preset control command array. When a match is found, the `relay_control` function is called, and the index position of the matching command in the array is passed via `index` parameter
  - **Logical flow**
    - Use the `if else` statement to perform different operations based on the value of `index`:
      - In the case of `index < 6`, that is, for the `CH1~CH6` commands, the `relays[index].value` function is used to switch the level status of the corresponding GPIO pin, and at the same time, the corresponding `relay_status` array element is updated to record the change of the relay status, and the corresponding on or off prompt information is output according to the final state of the relay. Finally, the `beep` function is called to control the buzzer.
      - For the case of `index = 6`, i.e., for the `ALL_ON` command, all GPIO pins (corresponding to 6 channel relays) are set to high level (on state), and all elements of the `relay_status` array are set to 1 through the `for` loop, indicating that all relays are turned on. A message indicating that all relays are turned on is then output. Finally, the `beep` function is called to control the buzzer.
      - For the case of `index = 7`, i.e., for the `ALL_OFF` command, all GPIO pins (corresponding to 6 channel relays) will be set to low level (off state). Using the `for` loop, all elements of the `relay_status` array will be set to 0, indicating that all relays are off. A message indicating that all relays are off will be output. Finally, the `beep` function will be called to control the buzzer.
      - If the value of `index` does not match the above commands, the prompt message of receiving non-command data will be output

RS485 Control

- Jump to RS485 Control Tutorial (https://www.waveshare.com/wiki/ESP32-S3-Relay-6CH-RS485)

02_MQTT

Demo description

- This demo only supports the RP2350-Relay-6CH-W
- This demo achieves network communication through a WiFi connection and remotely controls 6 relays via the MQTT protocol

Precautions

- Devices need to be created in Waveshare Cloud
- Modify the following parameters in config.py:
  - `wifi_ssid` : WiFi name
  - `wifi_password` : WiFi password
  - `mqtt_client_id` : Device ID
- Modify the following parameters in main.py:
  - `MQTT_SUB_TOPIC` : Subscribe to a topic
  - `MQTT_PUB_TOPIC` : Release a topic

Code analysis

- Function description
  - `initialize_wifi()` : Connect to WiFi
  - `mqtt_connect()` : Connect to MQTT
  - `mqtt_subscribe()` : Subscribe to a topic
  - `mqtt_publish()` : Publish message
  - `mqtt_recv_callback()` : JSON message parsing and execution core
  - `init_relays()` : Initialize all relays to the off state
  - `set_relay()` : Control a single relay
  - `set_all_relays()` : Control all relays
- Logical flow
  - **Device initialization**:
    - Call `init_relays()` to initialize all relays
  - **WiFi connection**:
    - Call `initialize_wifi()` to connect to the specified network
  - **MQTT connection**:

        - Call `mqtt_connect()` to connect to MQTT server
        - Call `client.set_callback()` to set the callback function
        - Subscribe to the topic MQTT_SUB_TOPIC via `mqtt_subscribe()`
    - **Control logic**:
        - Call `client.check_msg()` in the main loop to wait for messages
        - Receive JSON formatted messages (e.g., {data:{CH1:1}}) through the callback function `mqtt_recv_callback()`
        - Parse channel name (CH1~CH6 or ALL) and control value (0/1)
        - Call `set_relay()` and `set_all_relays()` to achieve relay control
        - After successfully controlling the relay, call `mqtt_publish()` to publish the update status to the MQTT server

Waveshare Cloud Control

- Jump to Waveshare Cloud Control (https://www.waveshare.com/wiki/ESP32-S3-POE-ETH-8DI-8RO-Waveshare_Cloud)

03_BLE

Demo description

- This demo only supports the RP2350-Relay-6CH-W
- This demo achieves wireless communication via Bluetooth Low Energy (BLE) to remotely control 6 relays

Precautions

- Device broadcast name: RP2350-Relay-6CH-W
- Use custom UUID service: 0000FF01-0000-1000-8000-00805F9B34FB
- Control command format: <channel name> <control value>

Code analysis

- Function description
    - `init_relays()` : Initialize relay control GPIO
    - `bluetooth.BLE()` : Initialize Bluetooth object
    - `BLERelayControl()` : Bluetooth implements relay control
    - `_process_command()` : Command parsing and execution core logic
- Logical flow
    - **Device initialization**:
        - Call `init_relays()` to initialize relay GPIO
        - Call `bluetooth.BLE()` to initialize Bluetooth object
        - Call `BLERelayControl()` to initialize Bluetooth controller

- **Establish connection**:
  - Open the Bluetooth serial port tool, find the device broadcast name and UUID
  - Enable notification feature after client connection
- **Control logic**:
  - Determine Bluetooth events through the callback function `_irq()`
  - Call the function `_process_command()` to receive text commands (e.g., CH1 1)
  - Parse channel name (CH1~CH6 or ALL) and control value (0/1)
  - Call `_set_relay()` and `_set_all_relays()` to achieve relay control
  - Return the operation result to the client through `_send_response()`

Example of control commands

- Open channel 1: `CH1 1`
- Close channel 2: `CH2 0`
- Open all channels: `ALL 1`
- Close all channels: `ALL 0`

Example of error feedback

- Invalid command: `ERR: Cmd Invalid`
- Invalid channel: `ERR: CH 1-6 Only`
- Invalid value: `ERR: Need Value 0/1`

External Expansions

RS485 Extended Relay Channels [Expand]

Extend Timer Switch Function with Pico Port [Expand]

Extend CAN Port with Pico Port [Expand]

Expand Environmental Monitoring Function [Expand]

Extend RS485 Port with Pico Port [Expand]

Resources

Supporting Resources

Demo

- Onboard RGB LED Test Demo (https://files.waveshare.com/wiki/RP2350-Relay-6CH/RP2350-Relay-6CH-RGB.zip)
- RP2350-Relay-6CH Demo (https://files.waveshare.com/wiki/RP2350-Relay-6CH/RP2350-Relay-6CH.zip)
- RP2350-Relay-6CH-W Demo (https://files.waveshare.com/wiki/RP2350-Relay-6CH/RP2350-Relay-6CH-W.zip)
- RP2350-Relay-6CH External Expansion Demo (https://files.waveshare.com/wiki/RP2350-Relay-6CH/RP2350-Relay-6CH-Expand.zip)

Schematic Diagram

- Schematic (https://files.waveshare.com/wiki/RP2350-Relay-6CH/RP2350-Relay-6CH.pdf)

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)
- Pico2 Schematic diagram (https://files.waveshare.com/wiki/common/Pico-2-schematic.pdf)
- Pico2 Pinout definition (https://files.waveshare.com/wiki/common/Pico-2-Pinout.pdf)
- Pico2 Getting Started (https://files.waveshare.com/wiki/common/Getting-started-with-pico-2.pdf)
- Pico2 C SDK User Manual (https://files.waveshare.com/wiki/common/Raspberry-pi-pico-2-c-sdk.pdf)
- Pico2 Python SDK User Manual (https://files.waveshare.com/wiki/common/Raspberry-pi-pico-2-python-sdk.pdf)
- Pico2 Datasheet (https://files.waveshare.com/wiki/common/Pico-2-datasheet.pdf)
- RP2350 Datasheet (https://files.waveshare.com/wiki/common/Rp2350-datasheet.pdf)
- RP2350 Hardware Design Reference Manual (https://files.waveshare.com/wiki/common/Hardware-design-with-rp2350.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)

Development Software

- Thonny Python IDE (Windows version V3.3.3) (https://files.waveshare.com/wiki/common/Thonny-3.3.3.zip)
- Pico environment building related software (https://drive.google.com/file/d/110wdNeJrX-NhCtEoGBAZgOCin9_VkGH3/view?usp=sharing)

- pico-vscode package (https://drive.google.com/file/d/18-KDNrQlI0KuTMdS6W5iblUGaGm3FbVJ/view?usp=sharing)
- SSCOM (https://files.waveshare.com/wiki/RP2350-Relay-6CH/SSCOM5.13.1_For_RP2350_Relay_6CH.zip)
- USB-CAN-A_TOOL (https://files.waveshare.com/wiki/common/USBCANV2.10.zip)

FAQ

Question: Raspberry Pi Pico 2 GPIO is configured as a pull-down input, why does reading IO show a high voltage level when the pin is left unconnected?

Answer:

You can refer to the RP2350-E9 section in the RP2350 Datasheet (https://files.waveshare.com/wiki/common/Rp2350-datasheet.pdf)

Question: When controlling other devices using RS485, it is not sensitive or communication fails?

Answer:

Please move the jumper cap to 120R and try again. Some RS485 devices require a 120R resistor to be connected in series

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

Submit Now (https://service.waveshare.com/)

issue.

Working Time: 9 AM - 6 PM GMT+8

(Monday to Friday)