RP2040-GEEK

From Waveshare Wiki
Jump to: navigation, search

Overview

Introduction

RP2040-GEEK is a geeks development board designed by Waveshare, onboard USB-A interface, 1.14-inch LCD screen, TF card slot, and other peripherals. Provides different firmware for SWD port, UART port, and I2C port.

RP2040-GEEK

(https://www.waveshare.com/rp2040-geek.htm)

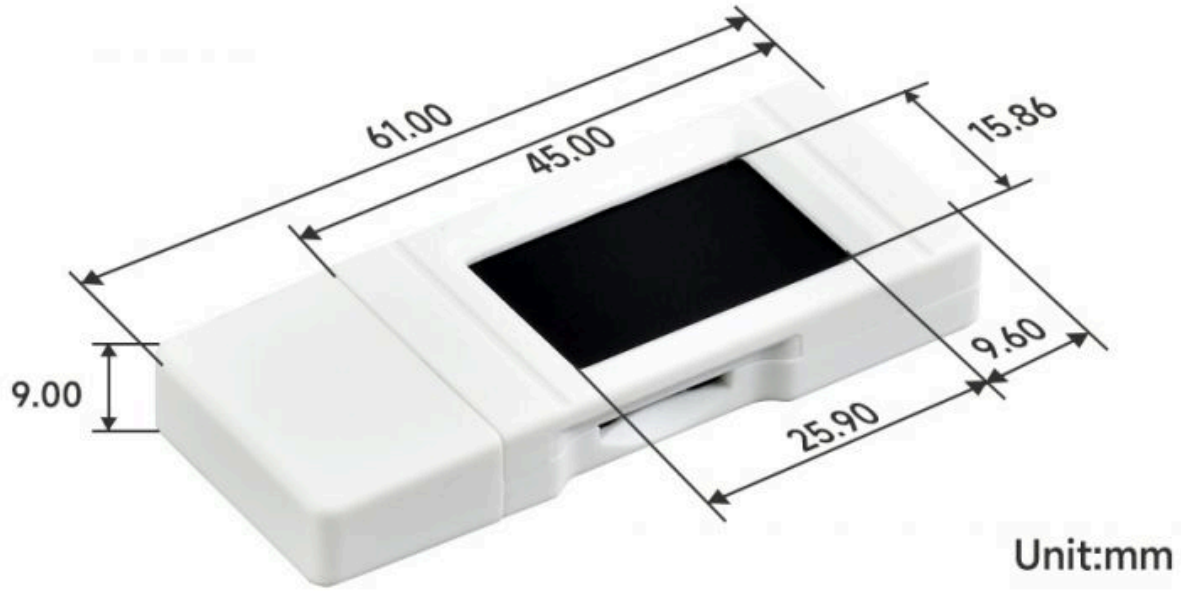
SWD, UART, USB-A

Feayures

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom.
- Dual-core ARM Cortex M0+ processor, with a high operating frequency of up to 133MHz and flexible clock.
- Built-in 264KB of SRAM and 4MB of onboard Flash.
- Onboard 1.14-inch 240×135 pixel 65K color IPS LCD display.
- Onboard 3PIN SWD port for connecting the debugged target board.
  - Standard CMSIS-DAP interface can be used to debug most ARM-based microcontrollers.
  - Works with OpenOCD and other tools supporting CMSIS-DAP.
  - Adopts the Raspberry Pi 3PIN Debug Connector Specification.
- Onboard 3PIN USB to UART bridge.
- Onboard 4PIN I2C port for the testing target board.
- Equipped with plastic case and cables.

- Open-source firmware, more convenient to upgrade.

Dimensions



(/wiki/File:RP2-4-_geek_09.jpg)

Pico Quick Start

Firmware Download

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

Firmware Download

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

Text Tutorial

Introduction

- Raspberry Pi Pico (https://www.waveshare.com/wiki/Raspberry_Pi_Pico)

MicroPython Series

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

- 【MicroPython】 machine.PWM Function (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.PWM_Function)
- 【MicroPython】 machine.ADC Function (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.ADC_Function)
- 【MicroPython】 machine.UART Function (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.UART_Function)
- 【MicroPython】 machine.I2C Function (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.I2C_Function)
- 【MicroPython】 machine.SPI Function (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91machine.SPI_Function)
- 【MicroPython】 rp2.StateMachine (https://www.waveshare.com/wiki/%E3%80%90MicroPython%E3%80%91PIO_Function)

C/C++ Series

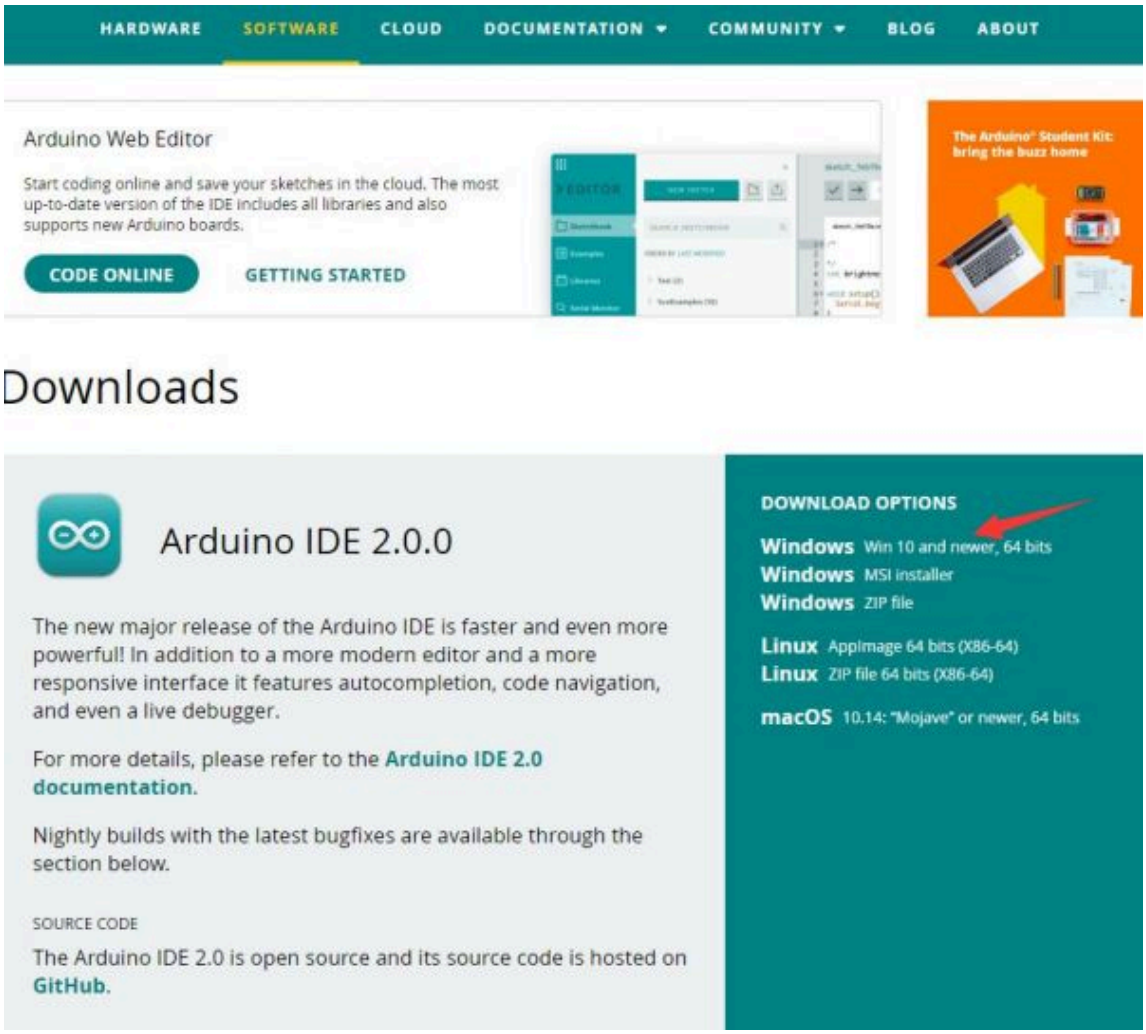
For C/C++, it is recommended to use Pico VS Code for development. This is a Microsoft Visual Studio Code extension designed to make it easier for you to create, develop, and debug projects for the Raspberry Pi Pico series development board. Whether you are a beginner or an experienced professional, this tool can help you confidently and easily develop Pico. Below we will introduce 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 applicable to Raspberry Pi Pico, Pico2, and our company's RP2040 and RP2350 series development boards.
- The development environment defaults to Windows as an example. For other environments, please refer to the official website tutorial for installation.

Arduino IDE Series

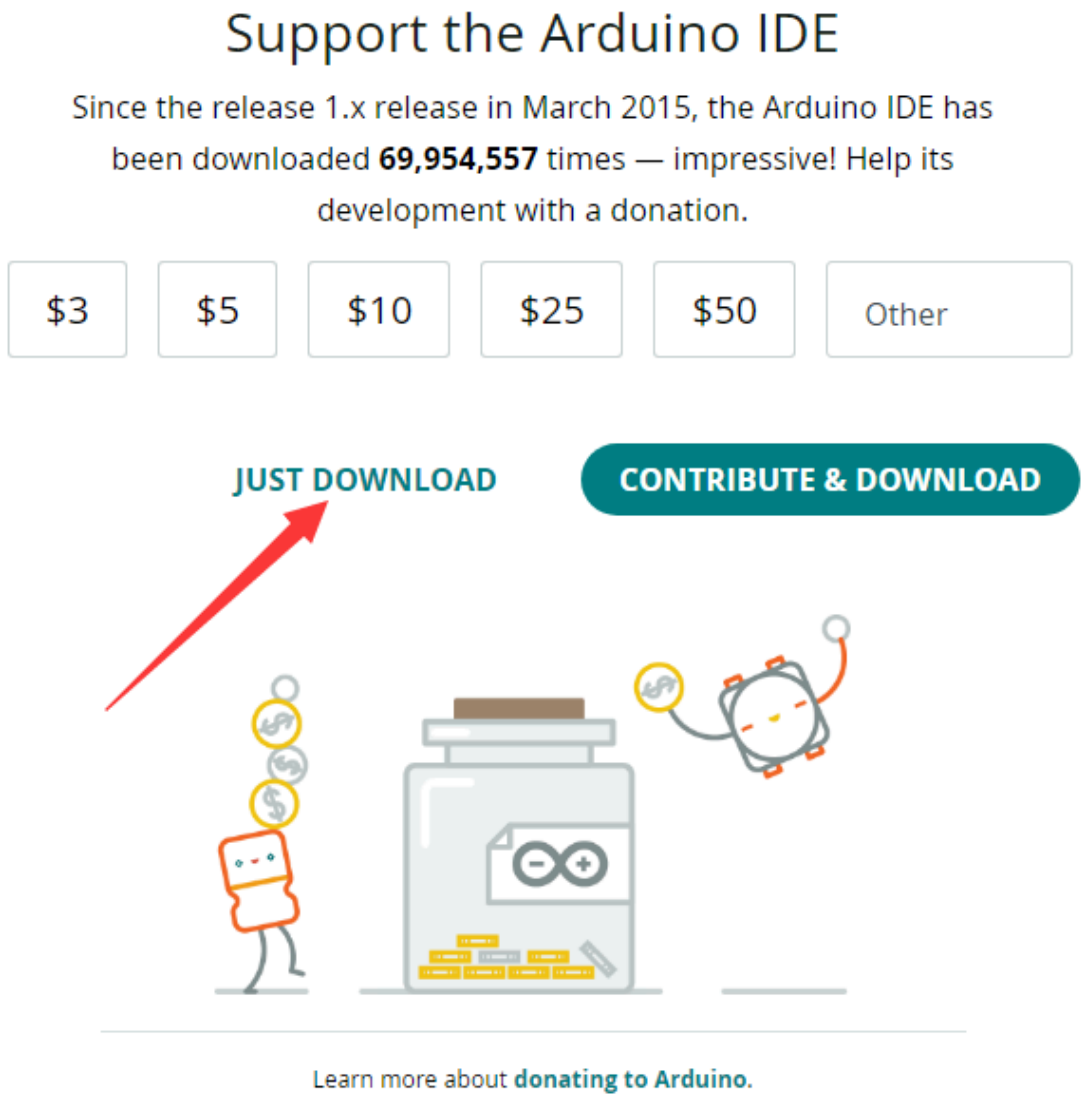
Install Arduino IDE

1. Download the Arduino IDE installation package from Arduino website (https://www.arduino.cc/).



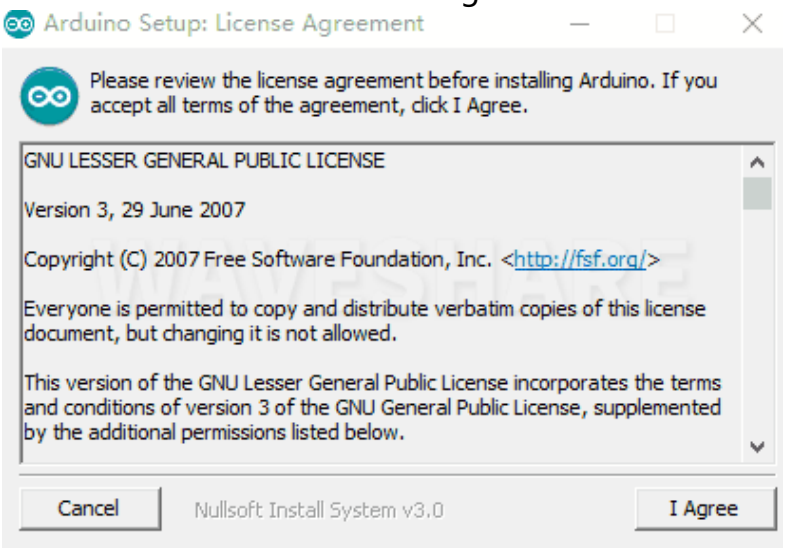
(/wiki/File:RoArm-M1_Tutorial_II01.jpg)

2. Just click on "JUST DOWNLOAD".



(/wiki/File:Arduino_IDE_Pico.png)

3. Click to install after downloading.



(/wiki/File:RoArm-M1_Tutorial_II02.gif)

4. **Note: You will be prompted to install the driver during the installation process, we can click Install.**

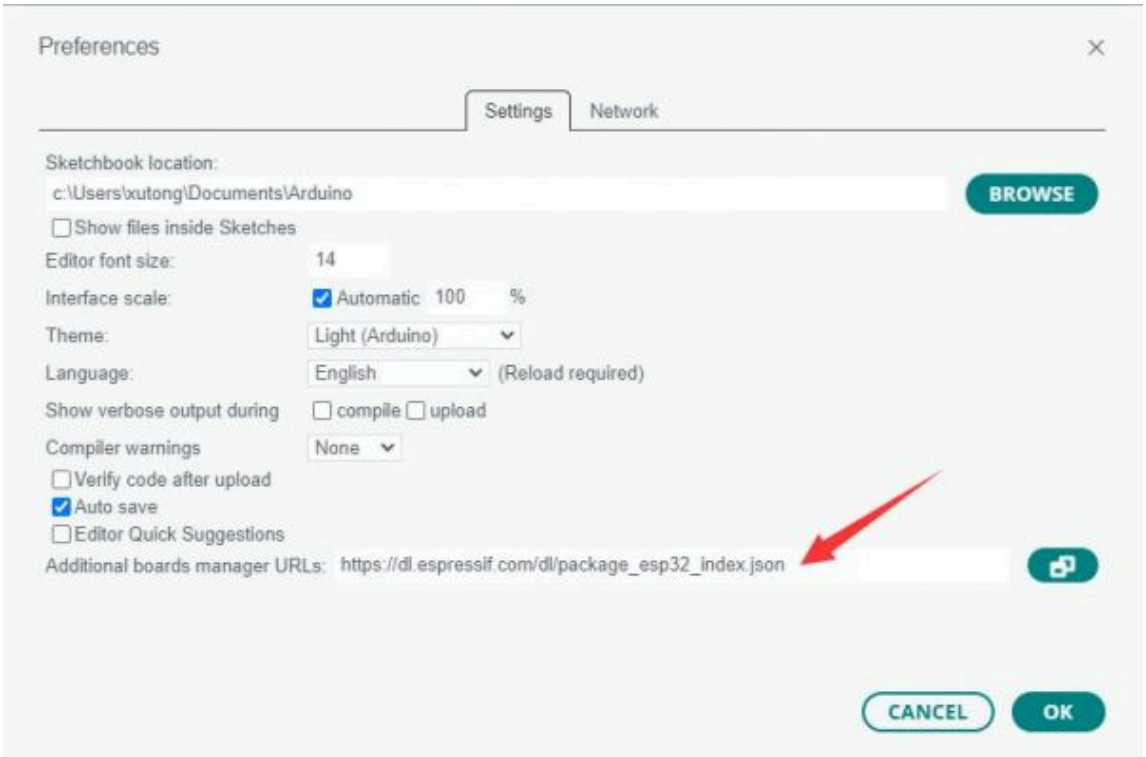
Install Arduino-Pico Core on Arduino IDE

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".

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

2. Add the following link in "Additional boards manager URLs", then click OK.

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

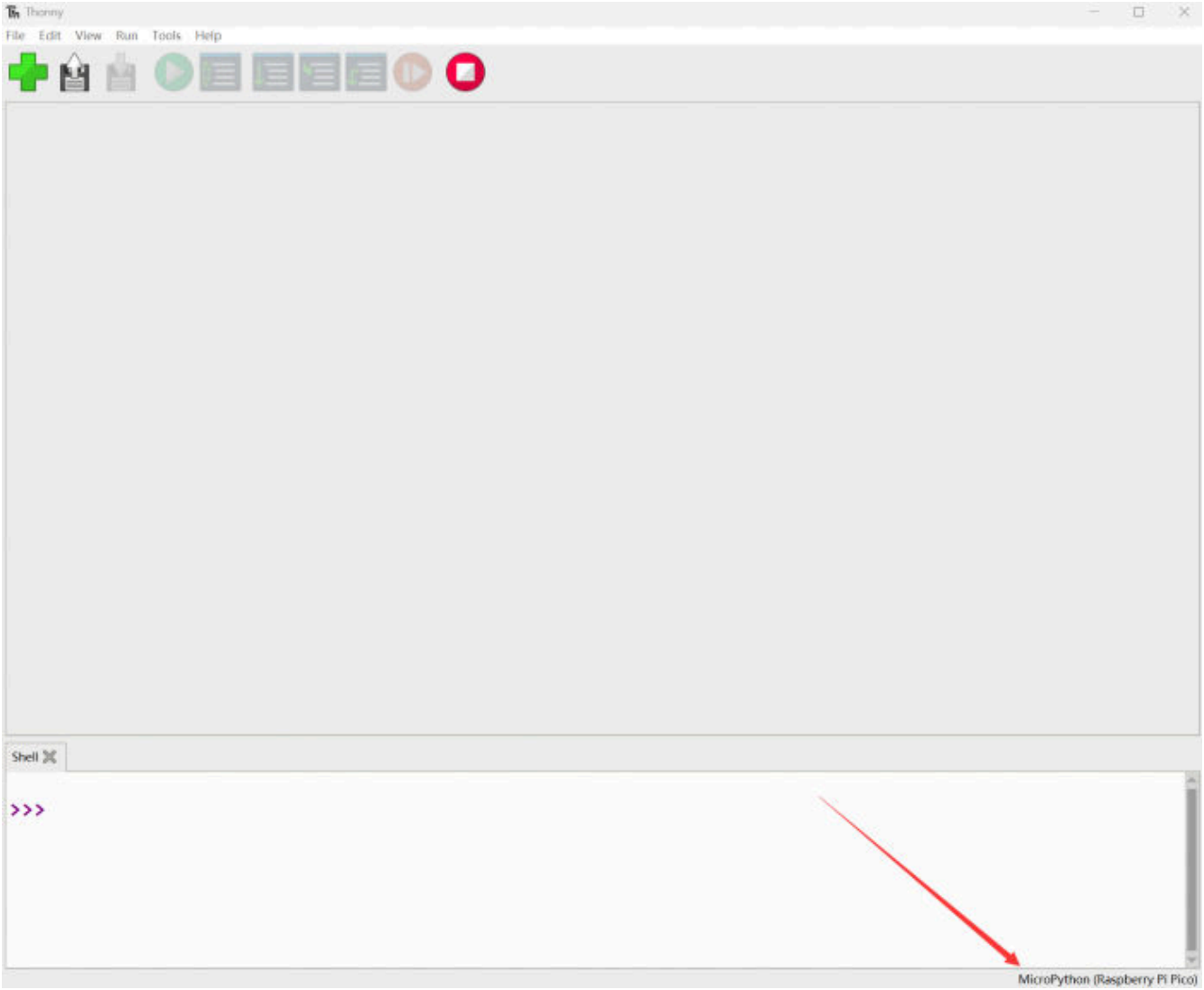


(/wiki/File:RoArm-M1_Tutorial_II05.jpg)

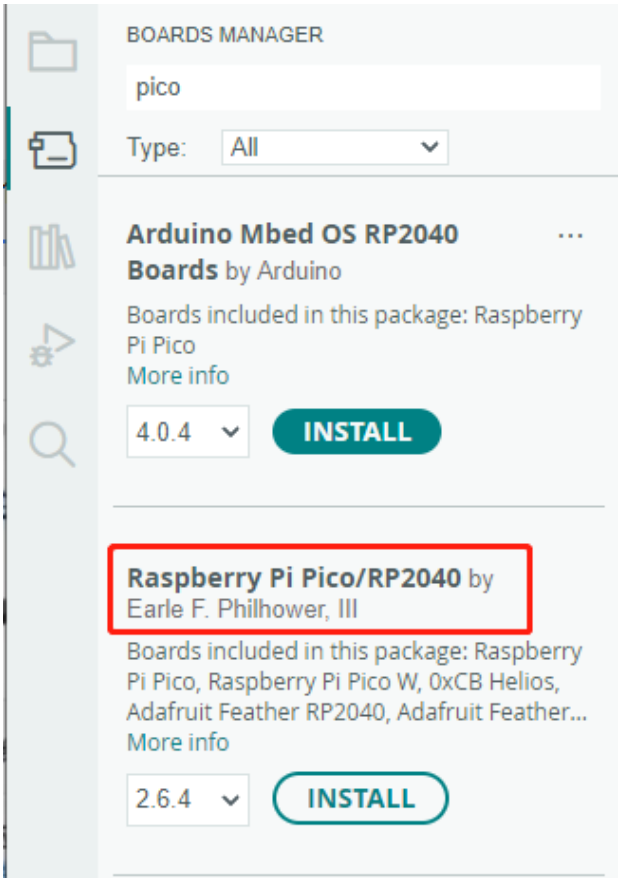
Note: If you already have the ESP32 board URL, you can separate the URLs with commas like this:

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

3. Click on Tools -> Board -> Board Manager -> Search for pico, it shows installed since my computer has already installed it.



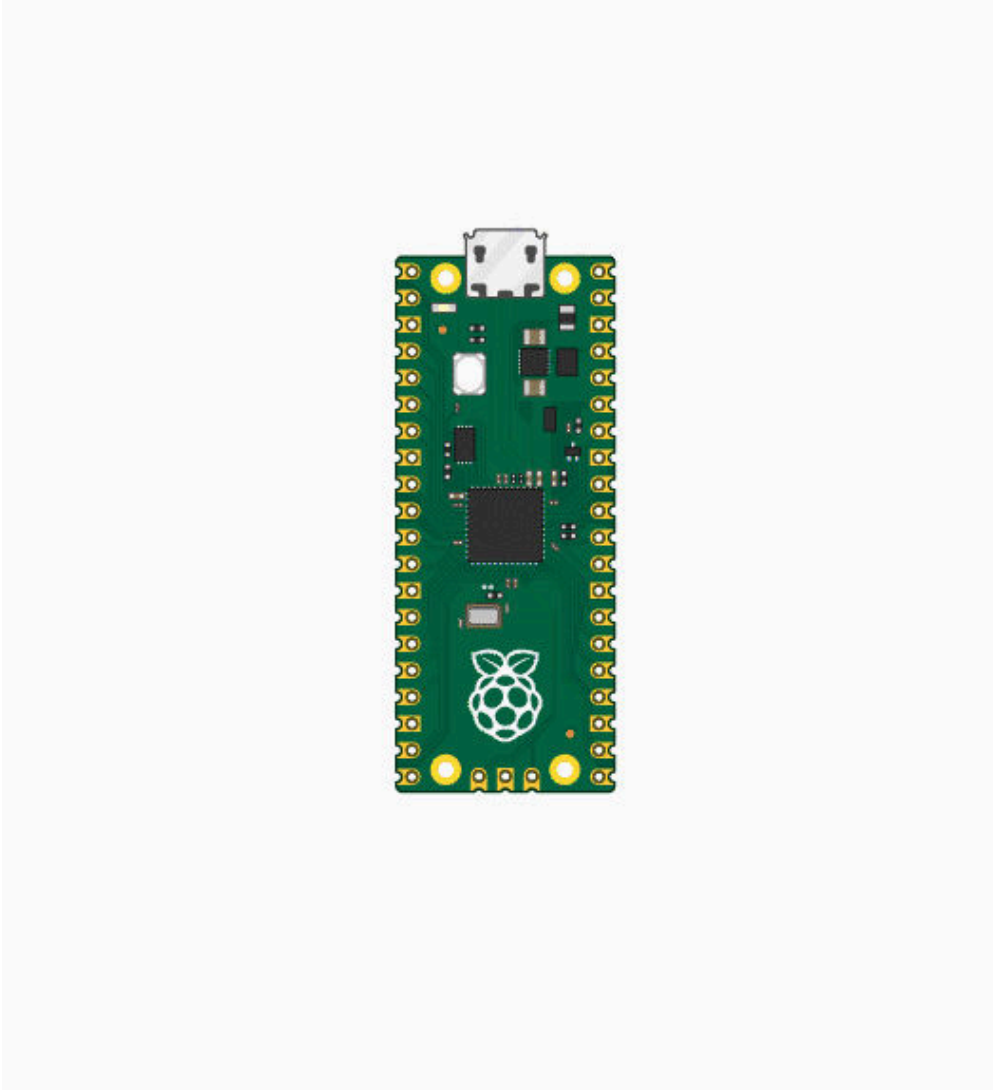
(/wiki/File:Pico_Get_Start_05.png)



(/wiki/File:Pico_Get_Start_06.png)

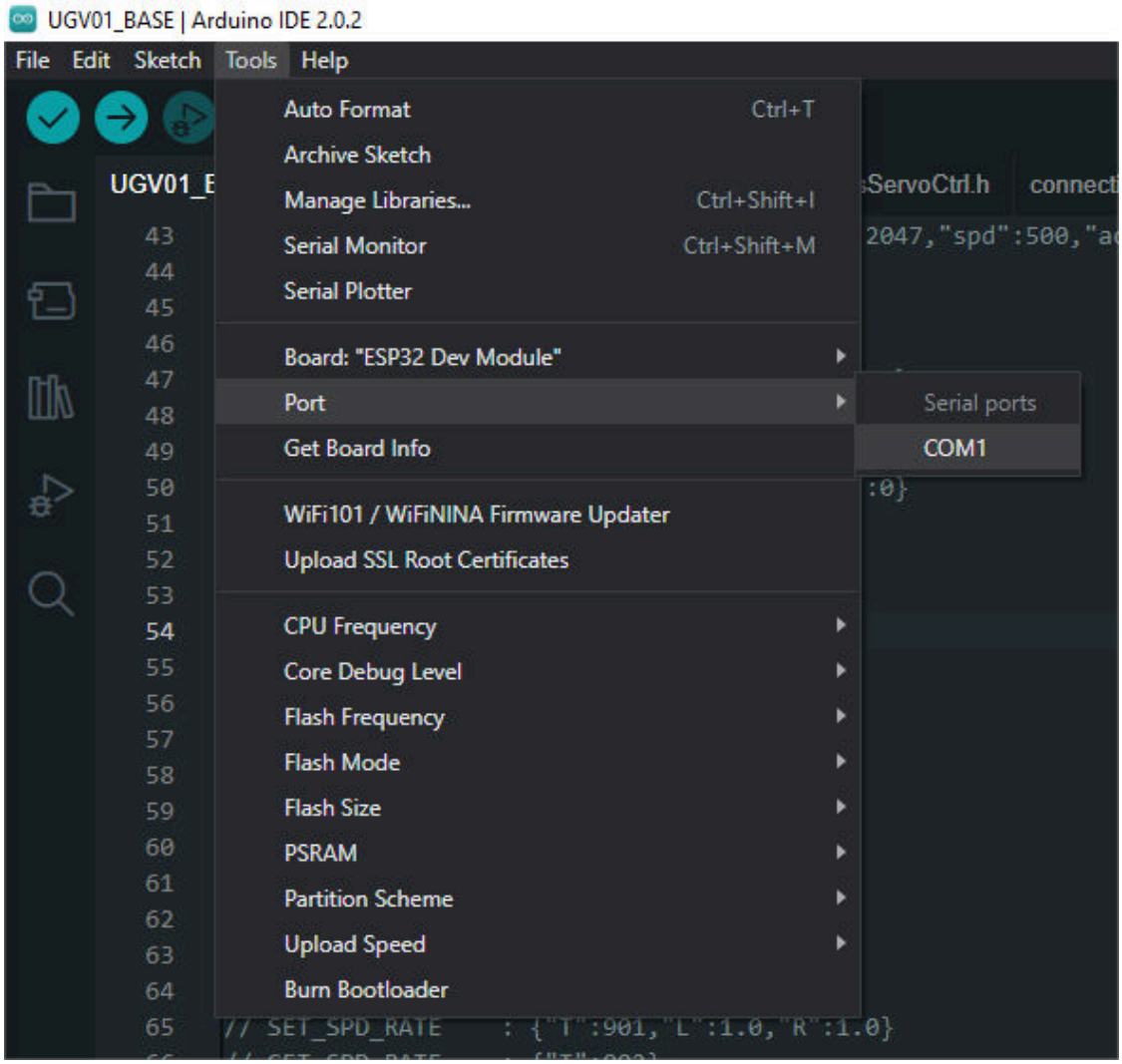
Upload Demo At the First Time

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 when the computer recognizes a removable hard drive (RPI-RP2).



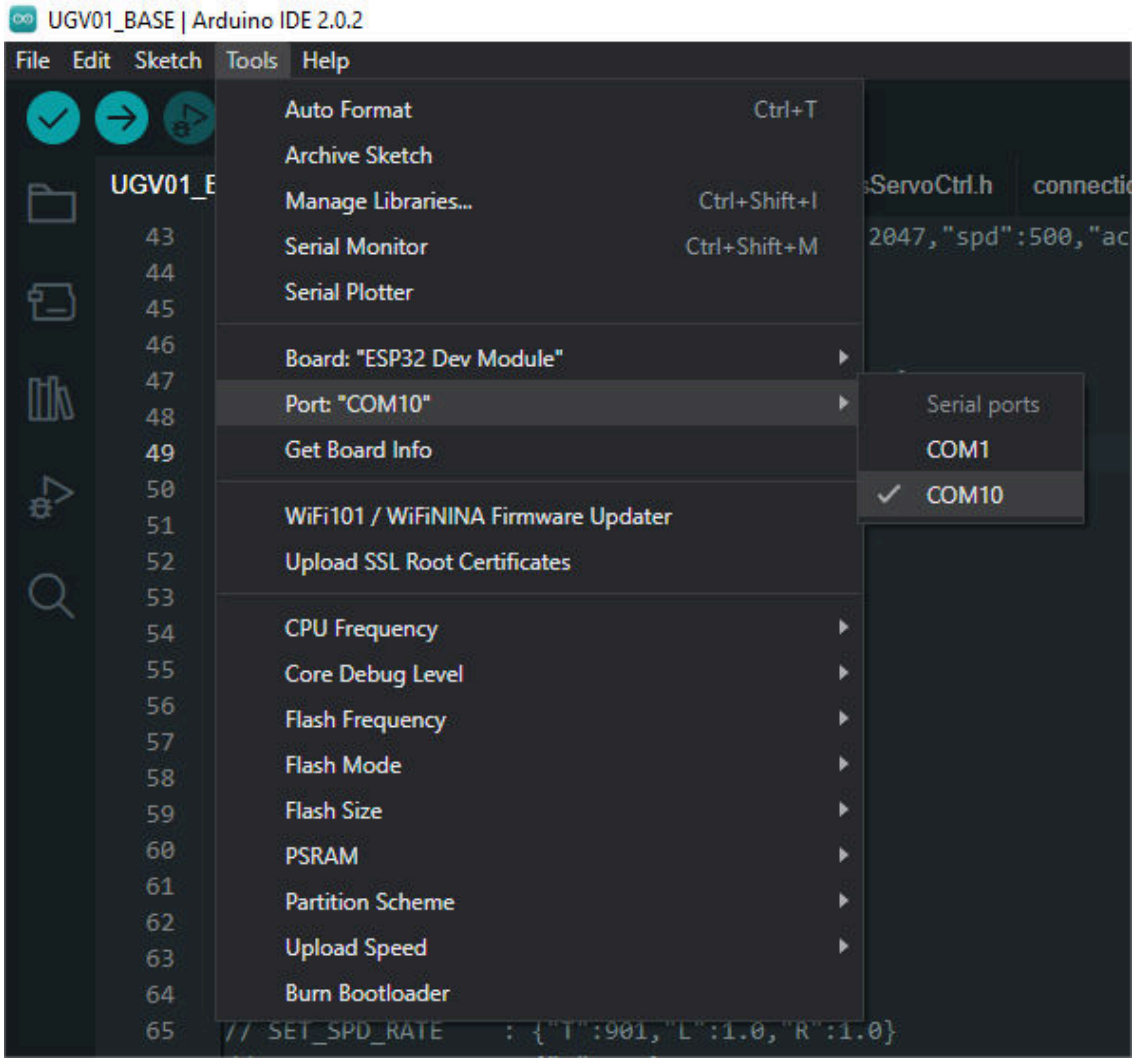
(/wiki/File:Pico_Get_Start.gif)

2. Download the demo from #Resource, open the D1-LED.ino under arduino\PWM\D1-LED path.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, remember the existing COM on your computer).



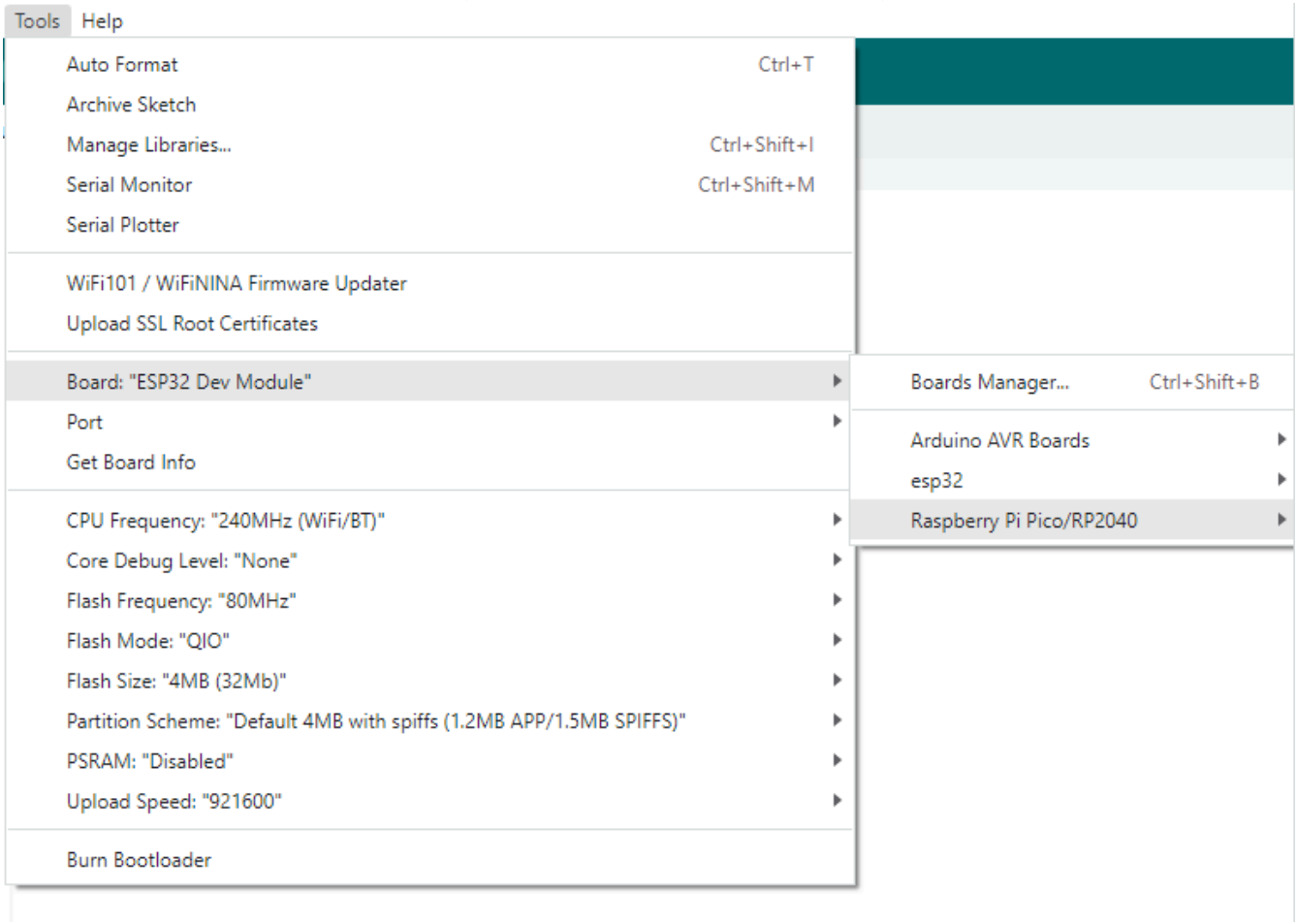
(/wiki/File:UGV1_doenload02EN.png)

4. Connect the driver board to the computer with a USB cable, then click Tools -> Ports, select uf2 Board for the first connection, and after the upload is complete, connecting again will result in an additional COM port.



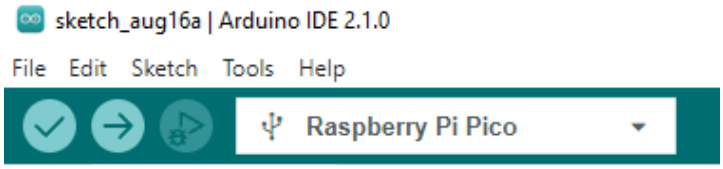
(/wiki/File:UGV1_doenload03EN.png)

5. Click Tools -> Board -> Raspberry Pi Pico/RP2040 -> Raspberry Pi Pico.



(/wiki/File:Pico_Get_Start02.png)

6. After setting, click the right arrow to upload.



(/wiki/File:Pico_Get_Start03.png)

- **If you encounter problems during the period, you need to reinstall or replace the Arduino IDE version, uninstall the Arduino IDE clean, after uninstalling the software you need to manually delete all the contents of the folder C:\Users\[name]\AppData\Local\Arduino15 (you need to show the hidden files in order to see it) and then reinstall.**

Open Source Demo

- MicroPython Demo (GitHub) (https://github.com/waveshareteam/Pico_MircoPython_Examples)
- MicroPython Firmware/Blink Demo (C) (https://files.waveshare.com/upload/b/b2/Raspberry_Pi_Pico_Demo.zip)
- Official Raspberry Pi C/C++ Demo (https://github.com/raspberrypi/pico-examples/)
- Official Raspberry Pi MicroPython Demo (https://github.com/raspberrypi/pico-micropython-examples)

- Arduino Official C/C++ Demo (https://github.com/earlephilhower/arduino-pico)

Sample Demo

C/C++ Demo

01-LCD

1. It takes 2.5s to display a picture on the LCD.
2. The example for LCD displaying the GUI.

02-picoprobe

- This demo is based on the open-sourced picoprobe (https://github.com/raspberrypi/picoprobe) demo.
- Run the picoprobe demo on the RP2040-GEEK, and it will analog a USB TO SWD and USB TO UART device.

1. Use UART as the tool for USB to UART for communication with the device.
2. Use the SWD interface as the debug tool for openocd to debug most of the arm chips.

- For more information on how to use picoprobe, please refer to the following "picoprobe Tutorial".

03-FATFS

- This demo is based on no-OS-FatFS-SD-SPI-RPi-Pico (https://github.com/carlk3/no-OS-FatFS-SD-SPI-RPi-Pico).

1. This demo repeatedly attempts to mount an SD card and read its root directory, with the option to get its runtime information using a USB serial port.

```
**********RUN MOUNT**********

sd_spi_go_low_frequency: Actual frequency: 398089
V2-Version Card
R3/R7: 0x1aa
R3/R7: 0x40ff8000
R3/R7: 0xc0ff8000
Card Initialized: High Capacity Card
SD card initialized
SDHC/SDXC Card: hc_c_size: 30199
Sectors: 30924800
Capacity:    15100 MB
sd_spi_go_high_frequency: Actual frequency: 12500000



**********RUN LS**********

Directory Listing: 0:/
System Volume Information [directory] [size=0]
compile_commands.json [writable file] [size=589200]
cmake_install.cmake [writable file] [size=2073]
CMakeCache.txt [writable file] [size=22707]
numbers.txt [writable file] [size=63]
```

(/wiki/File:RP2040-GEEK-DEMO.jpg)

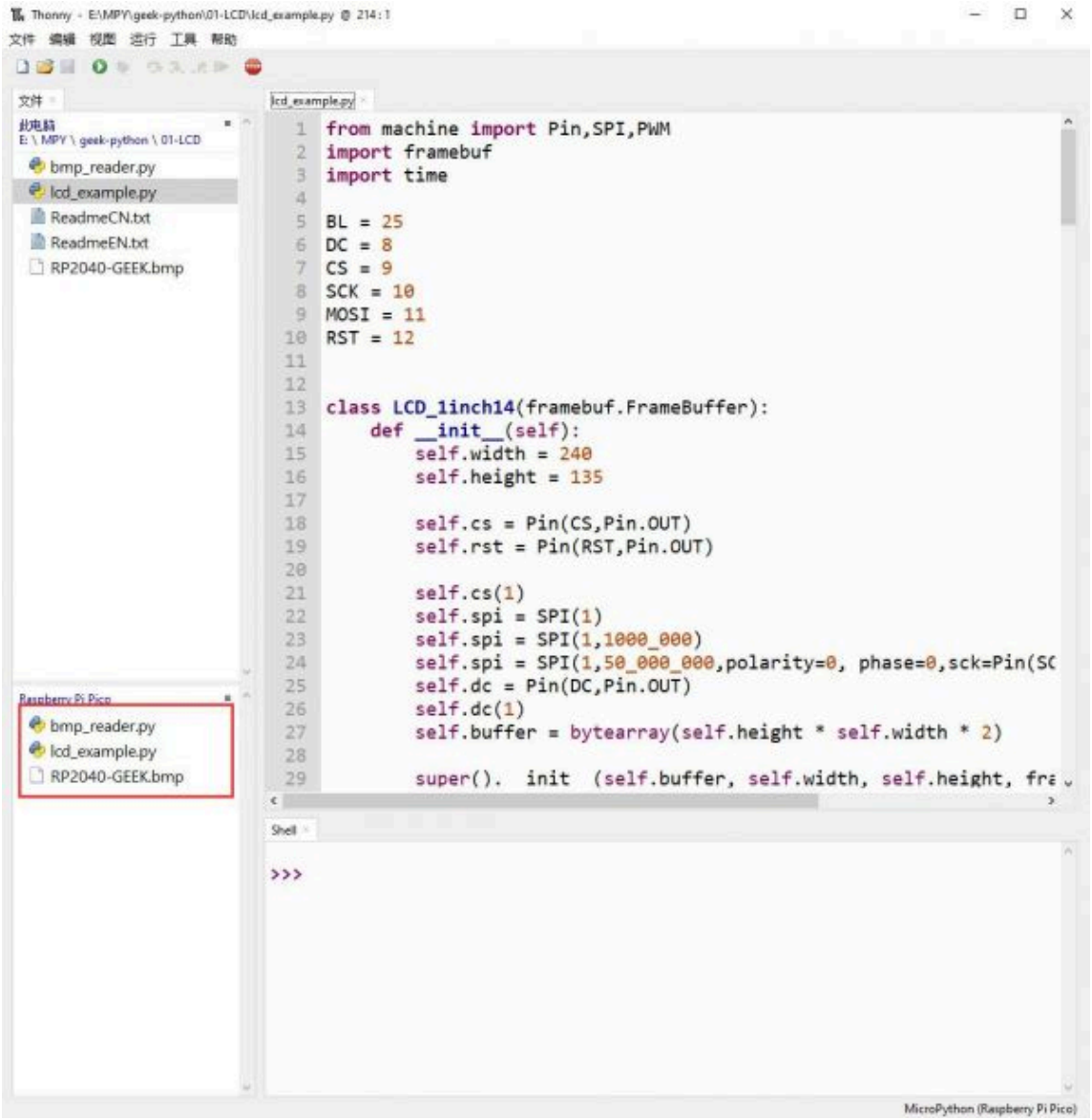
- Note that the SD card file format is FAT32.

Micropython Demo

01-LCD

How to Use

1. Upload all py and BMP files to RP2040-GEEK via thonny.



(/wiki/File:RP2040-Thonny.jpg)

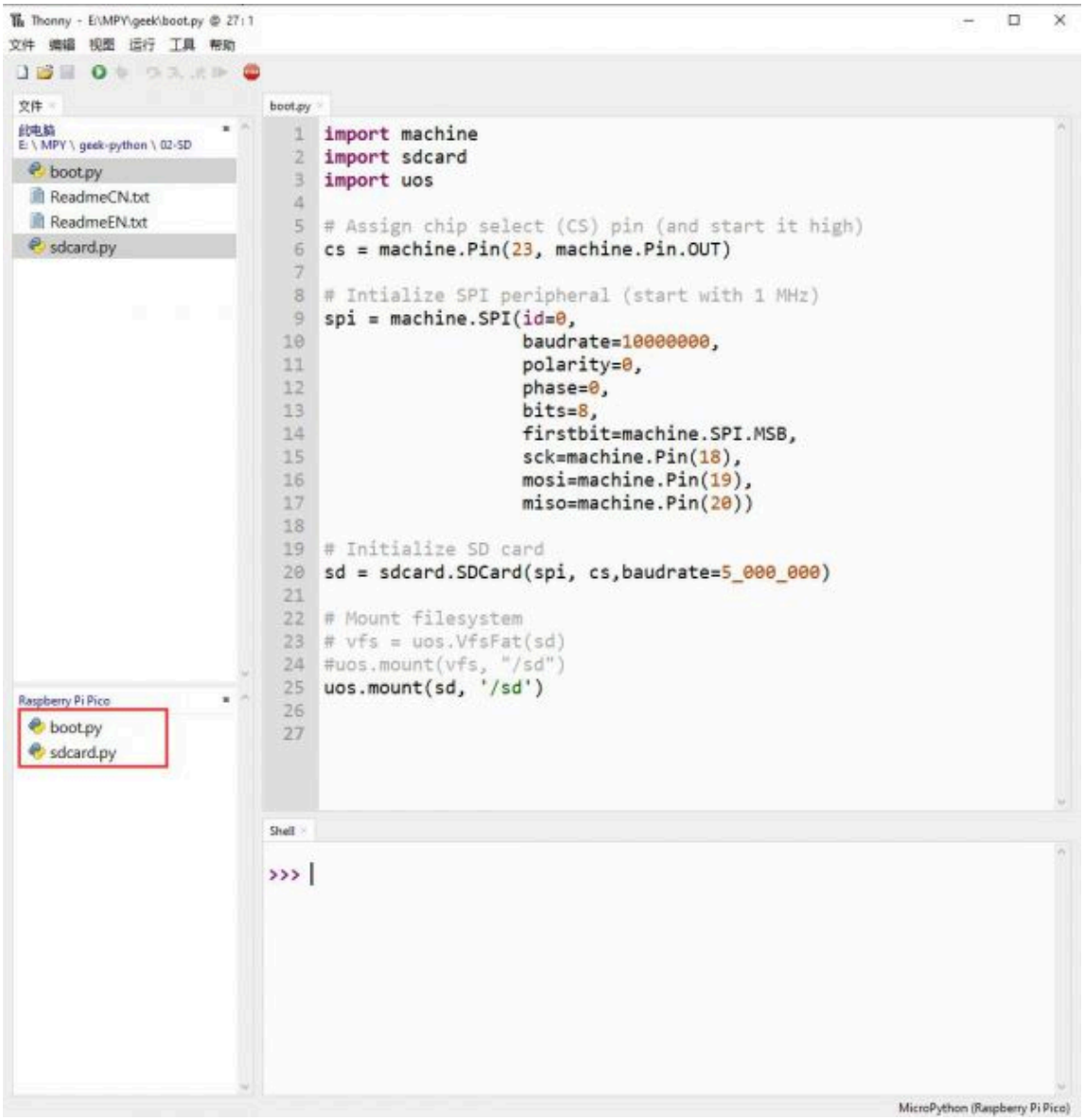
Demo Effect

- LCD displays the GUI, and then bmp picture after a few seconds.

02-SD

How to Use

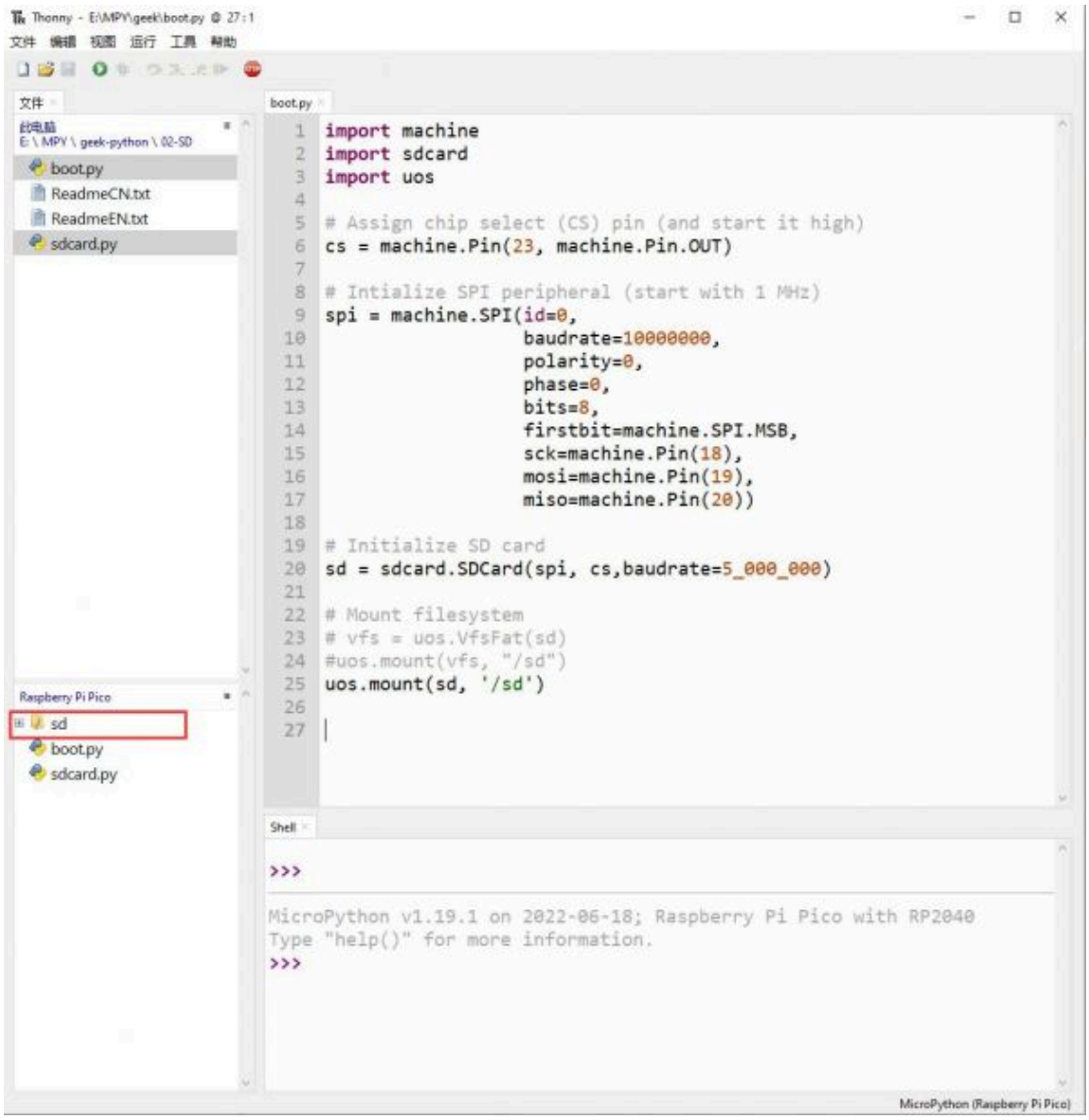
- Upload all the py files to the RP2040-GEEK via thonny, and reset.



(/wiki/File:RP2040-Thonny02.jpg)

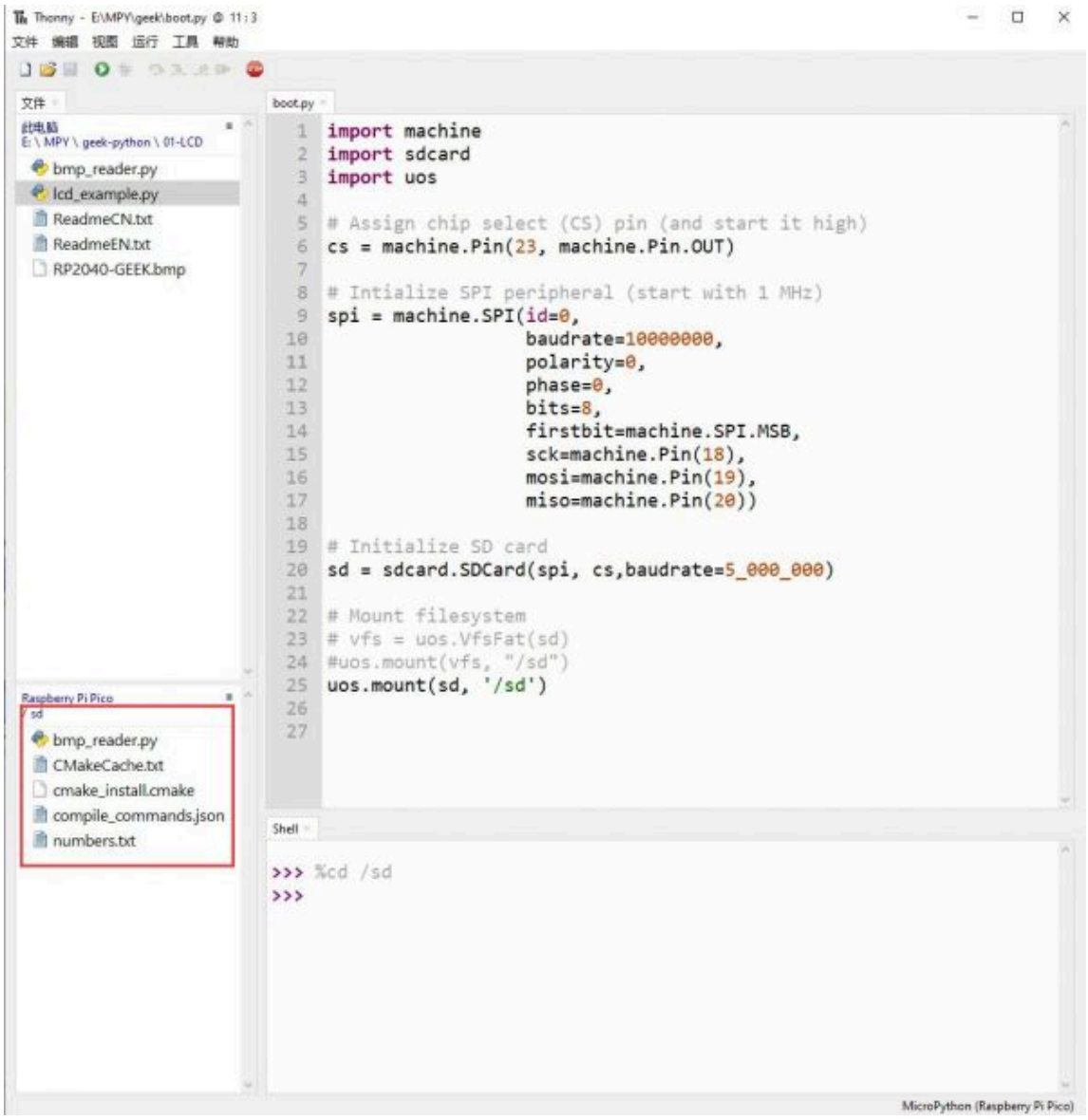
Demo Effect

- After RP2040-GEEK resets, it will automatically mount the SD card to the sd file according to the "boot.py" demo.



(/wiki/File:RP2040-Thonny03.jpg)

- Double-click the SD file folder and you can see the file stored on the SD card.



(/wiki/File:RP2040-Thonny04.jpg)

Picoprobe User Guide

Install OpenOCD

Linux (and Raspberry Pi)

Download the Dependency Library

```
sudo apt install automake autoconf build-essential texinfo libtool libftdi-dev libusb-1.0
-0-dev
```

Get and Compile

```
git clone https://github.com/raspberrypi/openocd.git --branch rp2040 --depth=1 --no-singl
e-branch
cd openocd
./bootstrap
```

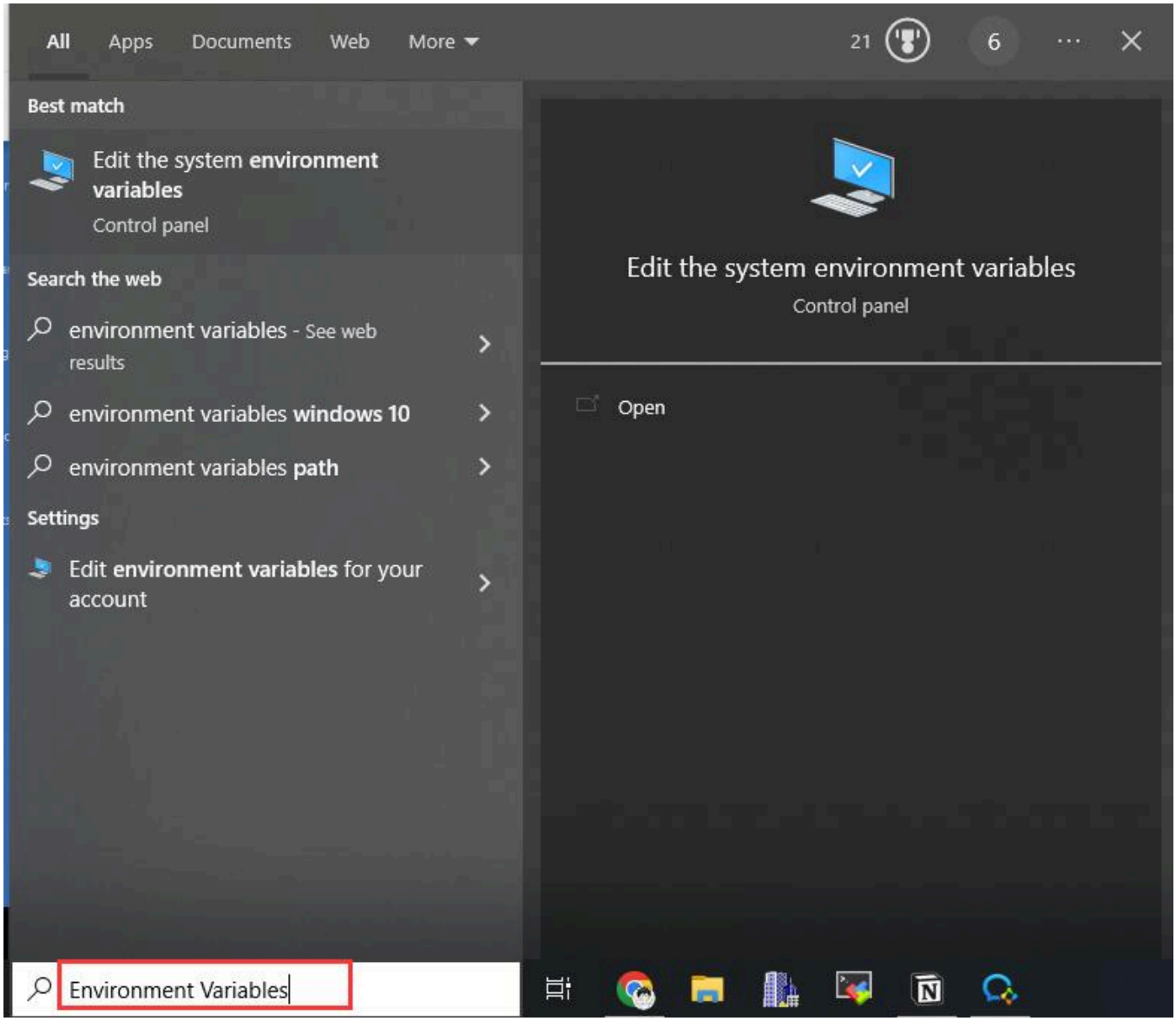
```
./configure
make -j4
sudo make install
```

Windows

1. As OpenOCD self-compilation is complicated in the windows environment, it is recommended to use the already compiled version.
   - Click here to download. (https://files.waveshare.com/upload/4/4b/OpenOCD-0.12.0.zip)
2. Unzip and store in a relatively short directory, e.g. directly in the C drive.

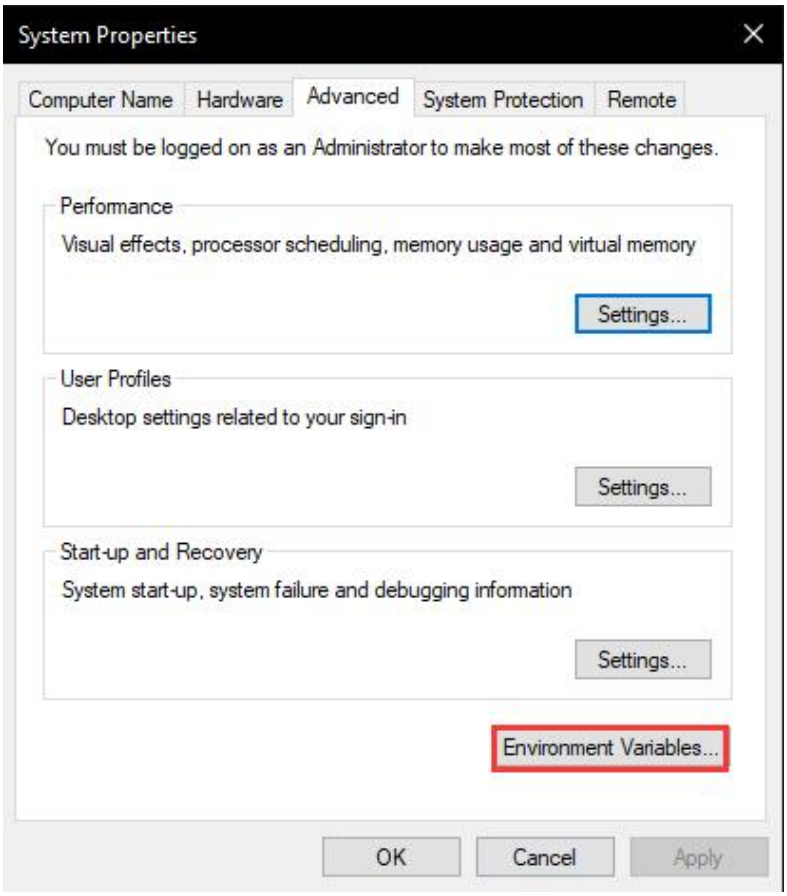
Add Environment Variables

1. Click the menu, and search "environment variables".



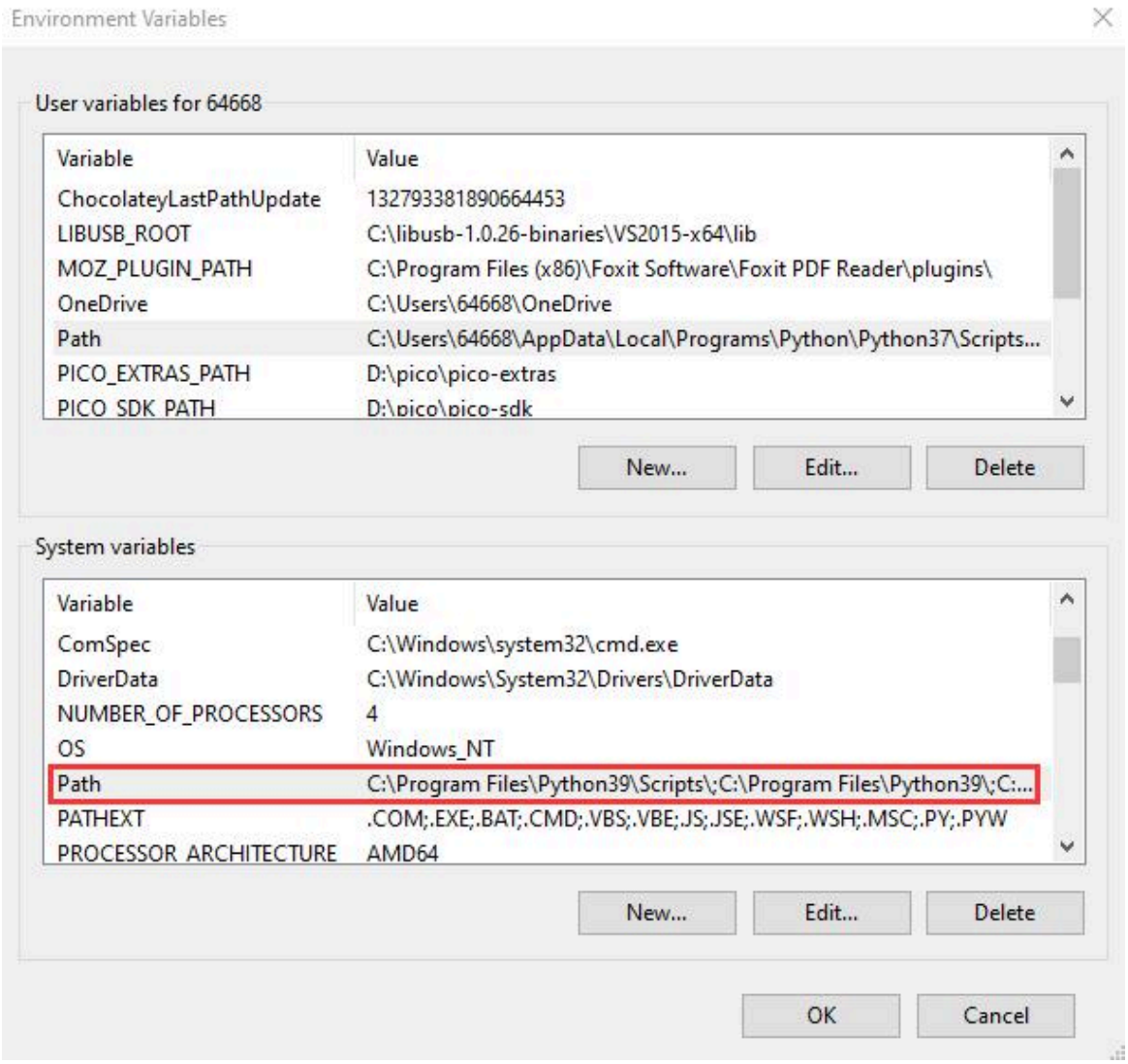
(/wiki/File:Raspberry_Pi_Debug_Probe05.jpg)

2. Click "Edit the system environment variables":



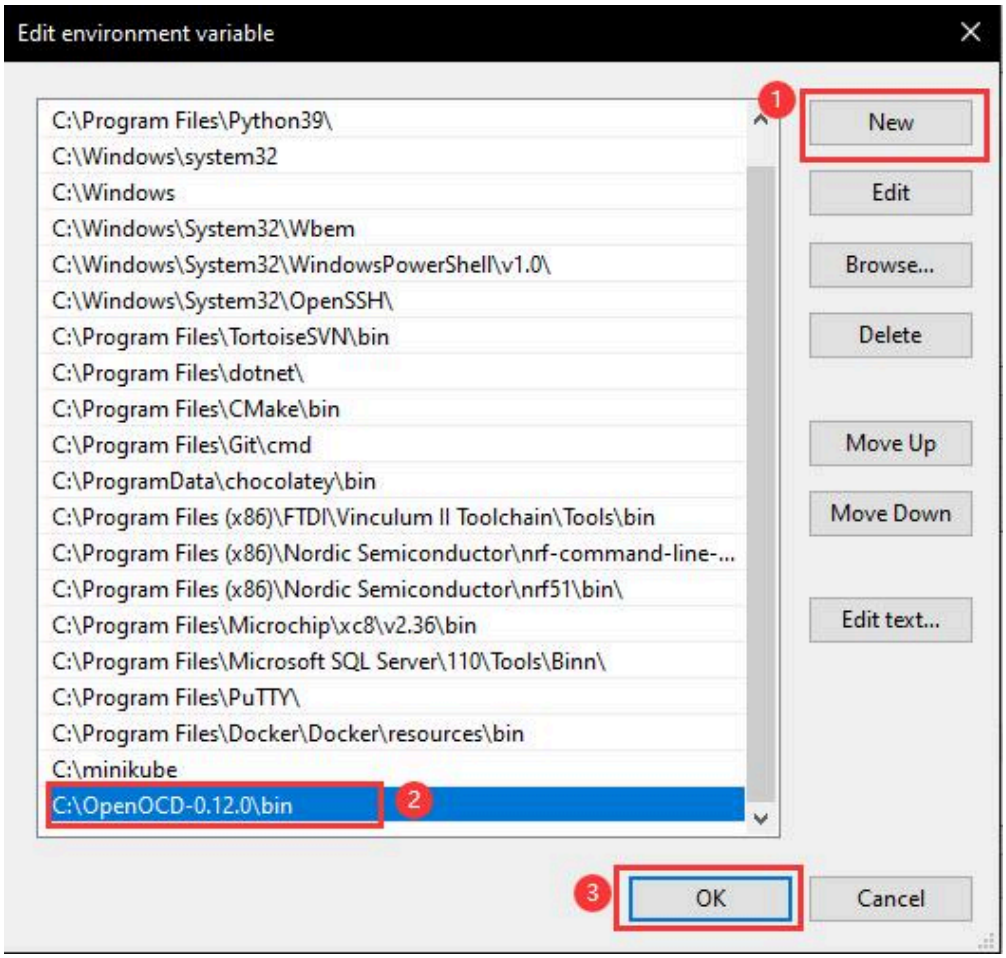
(/wiki/File:Raspberry_Pi_Debug_Probe06.jpg)

3. Double-click the "Path" variable, and then enter the edit interface.



(/wiki/File:Raspberry_Pi_Debug_Probe07.jpg)

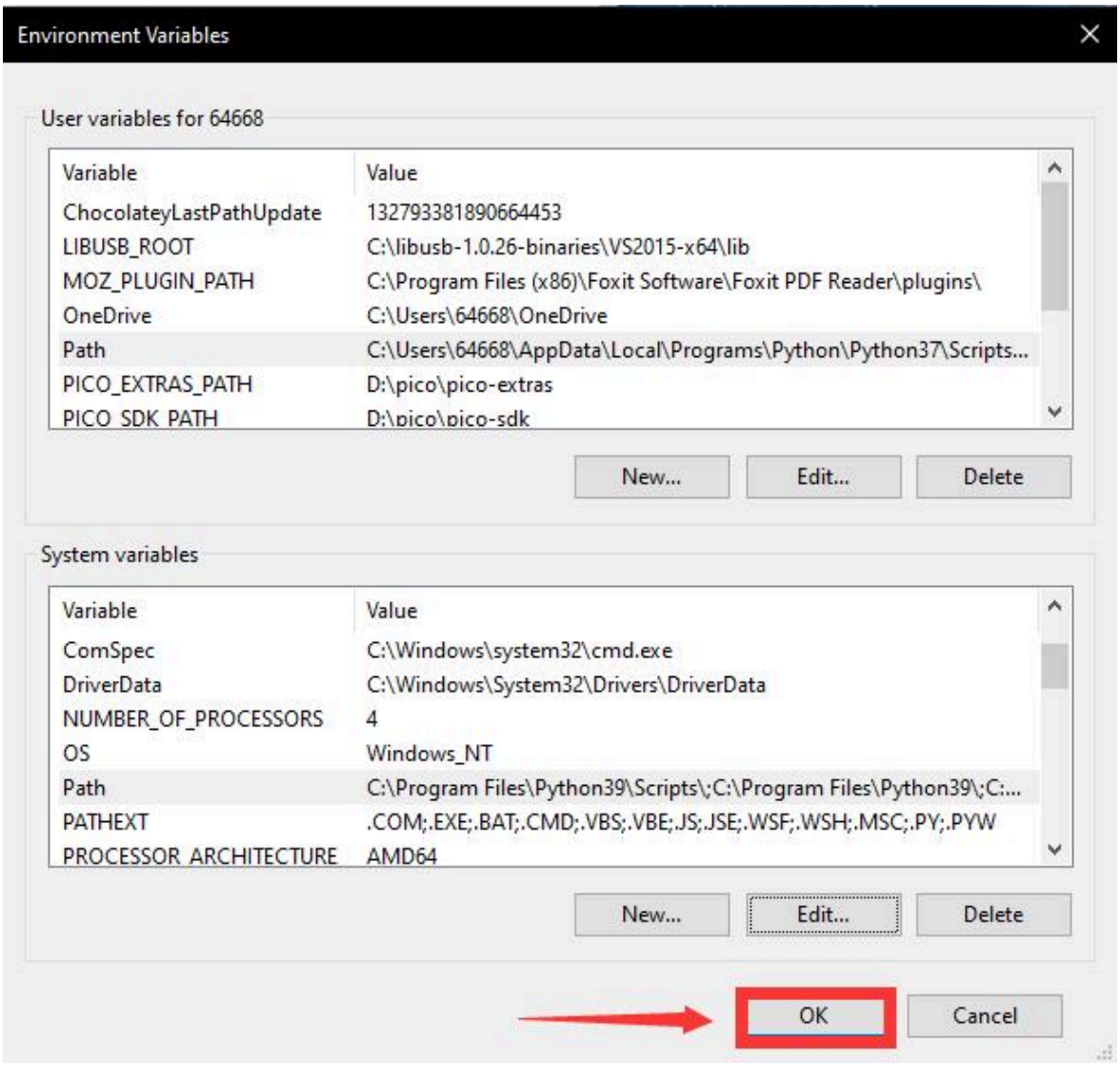
4. Add a new path.



(/wiki/File:Raspberry_Pi_Debug_Probe08.jpg)

- Create a new variable address.
- Enter the OpenOCD storage address.
- Click OK to save.

5. Click "OK" and save the change.



(/wiki/File:Raspberry_Pi_Debug_Probe09.jpg)

6. Reboot the computer.

Install GDB

Linux (and Raspberry Pi)

1. Install gdb-multiarch.

```
sudo apt install gdb-multiarch
```

Windows

1. If you have installed the related pico-sdk environment, you can skip this step as the GDB is included in Arm GNU Toolchain.

2. If you have not installed the related pico-sdk environment, it is recommended to install it with the official pico demo.

- Official website (https://github.com/raspberrypi/pico-setup-windows/releases/tag/v0.5.1)

- Package (https://www.aliyundrive.com/s/78gDFbrSk9P)

Program the Demo with Raspberry Pi Debug Probe

- With Pico Debug Probe, you can load the binaries via the SWD port and OpenOCD.
- You do not need to unplug and hold the BOOTSEL button every time you push a new binary to Pico.

1. Take RP2040 usage as an example, the commands for programming the demo:

```
sudo openocd -f interface/cmsis-dap.cfg -f target/rp2040.cfg -c "adapter speed 5000" -c
"program {your elf file name}.elf verify reset exit"
```

2. If there is a blink.elf file in your current file folder.

```
sudo openocd -f interface/cmsis-dap.cfg -f target/rp2040.cfg -c "adapter speed 5000" -c
"program blink.elf verify reset exit"
```

Debug the Demo with Raspberry Pi Debug Probe

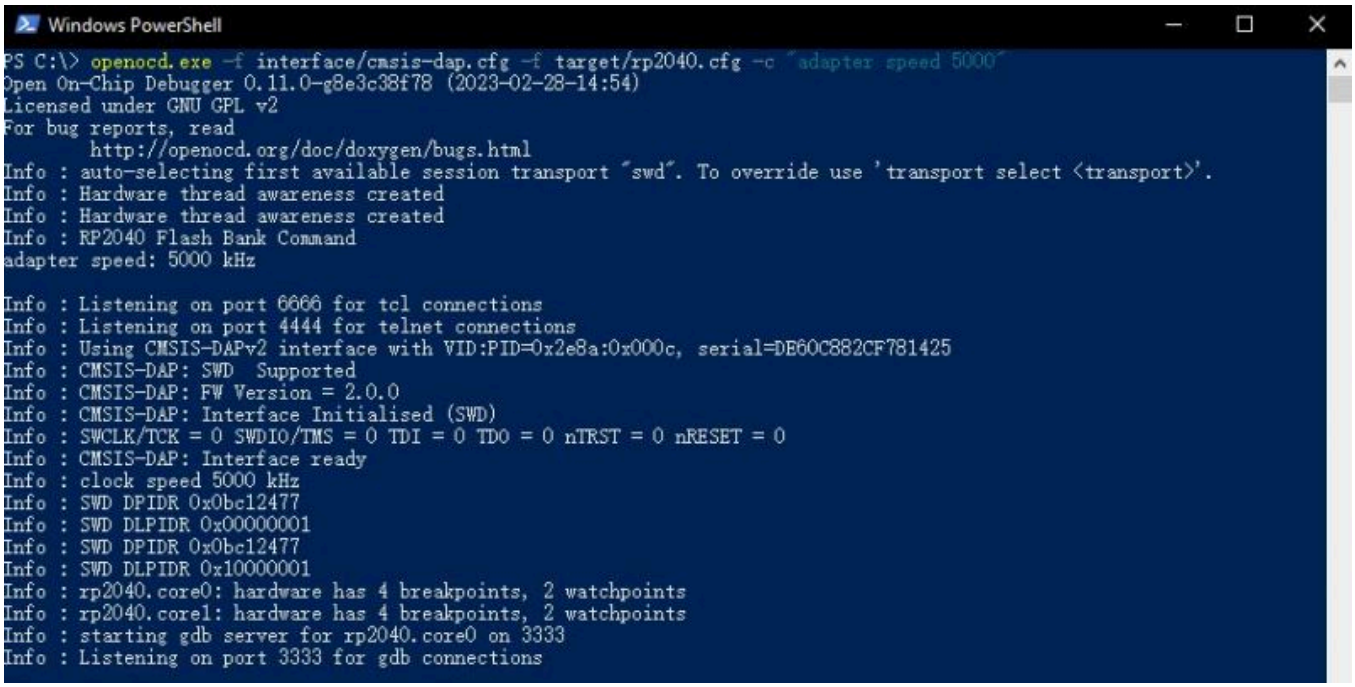
Open OpenOCD Server

- You can let openocd be used in server mode and connect to GDB, thus providing you with breakpoints and "correct" debugging.

1. Here is also an example of debugging the rp2040 by entering the following command:

```
sudo openocd -f interface/cmsis-dap.cfg -f target/rp2040.cfg -c "adapter speed 5000"
```

- Take using PowerShell in windows as an example:



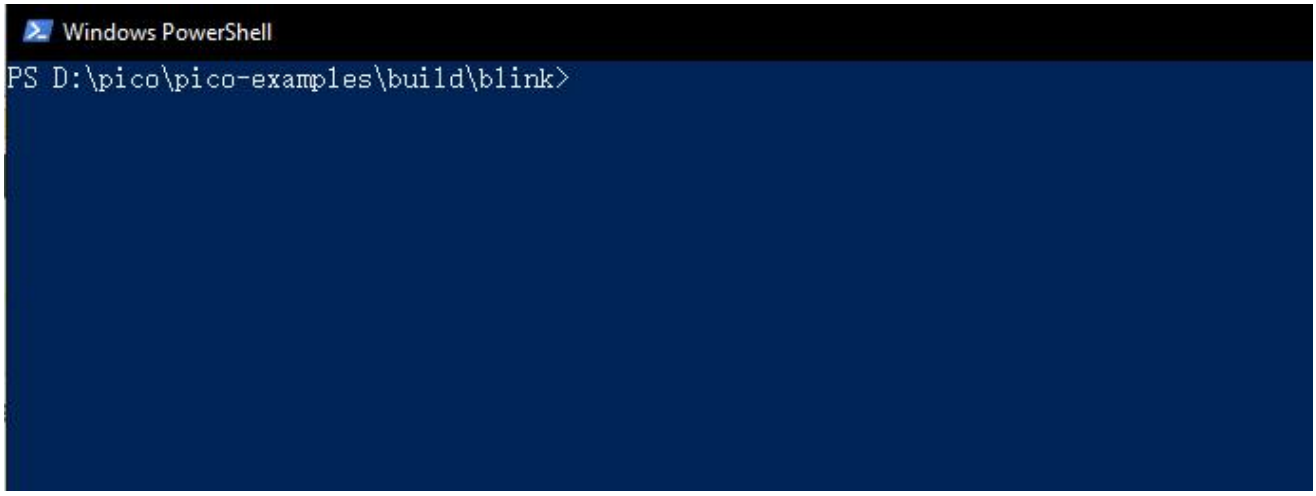
(/wiki/File:Raspberry_Pi_Debug_Probe010.jpg)

2. *If you start listening on the local 3333 port at this point, the OpenOCD server has been successfully turned on.

Use GDB Command Line

- This demo is set up based on pico-sdk, and pico-example was compiled.

1. Open PowerShell and enter the corresponding "build" file folder, take the blink demo as an example.



(/wiki/File:Raspberry_Pi_Debug_Probe011.jpg)

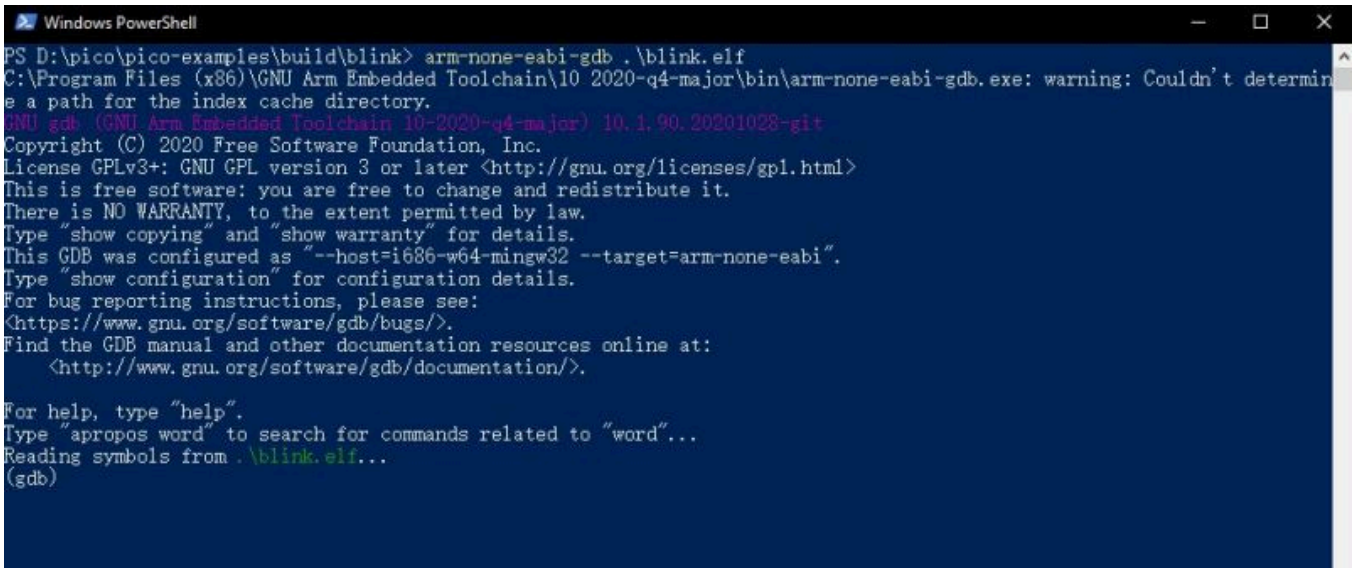
2. Open GBD and enter the following commands:

- If it is windows, you can input the following commands:

```
arm-none-eabi-gdb blink.elf
```

- If it is Linux, you can input the following commands:

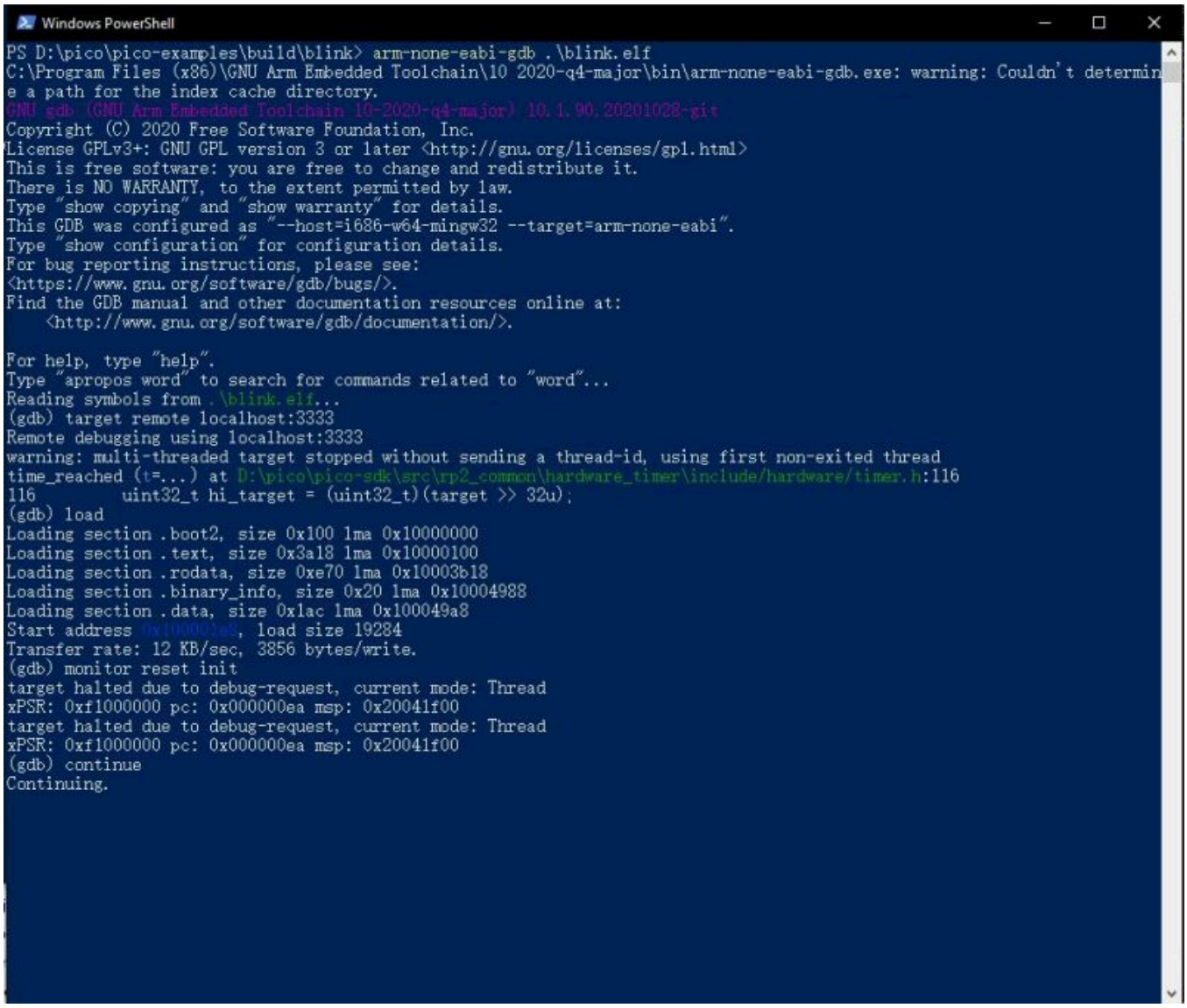
```
gdb blink.elf
```



(/wiki/File:Raspberry_Pi_Debug_Probe012.jpg)

3. Input the following commands in order:

```
target remote localhost:3333
load
monitor reset init
continue
```



(/wiki/File:Raspberry_Pi_Debug_Probe013.jpg)

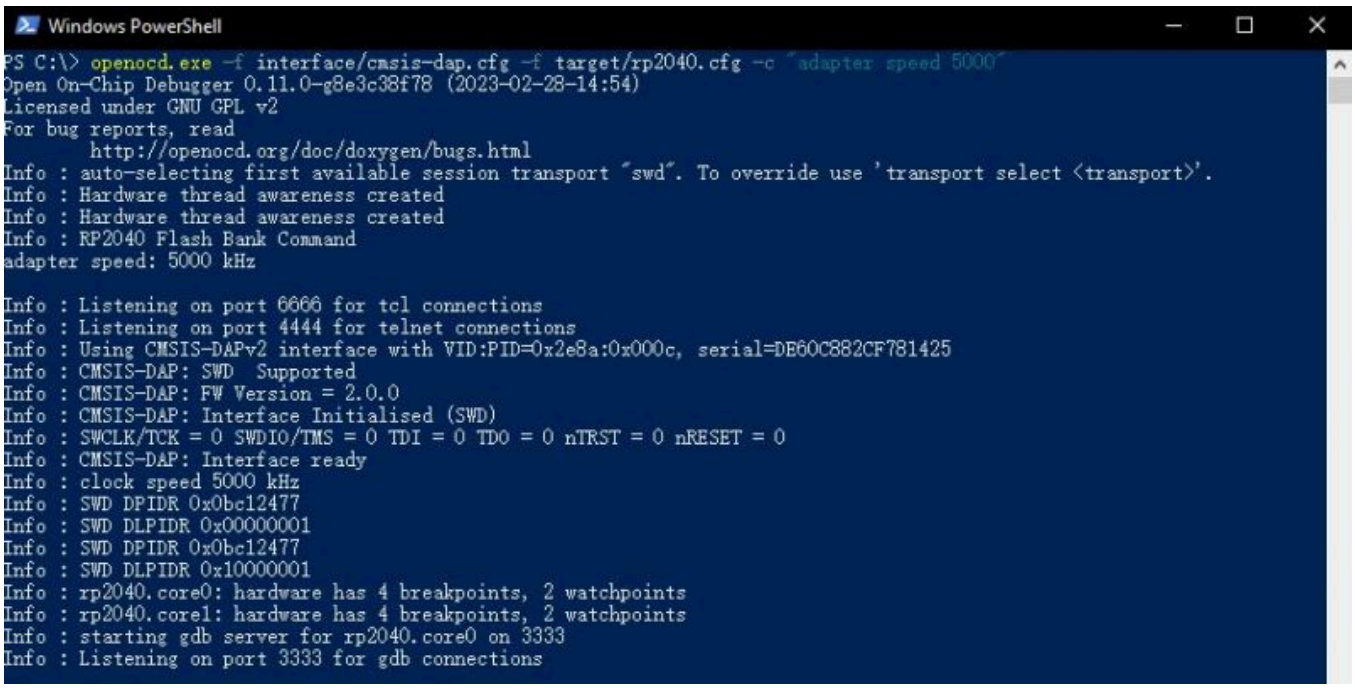
- You can see Pico executes blink, and LED blinks.

Use VSCode to Debug (Advanced)

- Please make sure that the #Open OpenOCD Server and the #Use GDB Command Line are running properly.
- Please make sure the Pico compilation environment is set up normally.
- Please make sure your VSCode can install the following plug-in.

1. Cortex-Debug ()
2. Cmake-tools
3. C/C++

1. First, open the OpenOCD server:



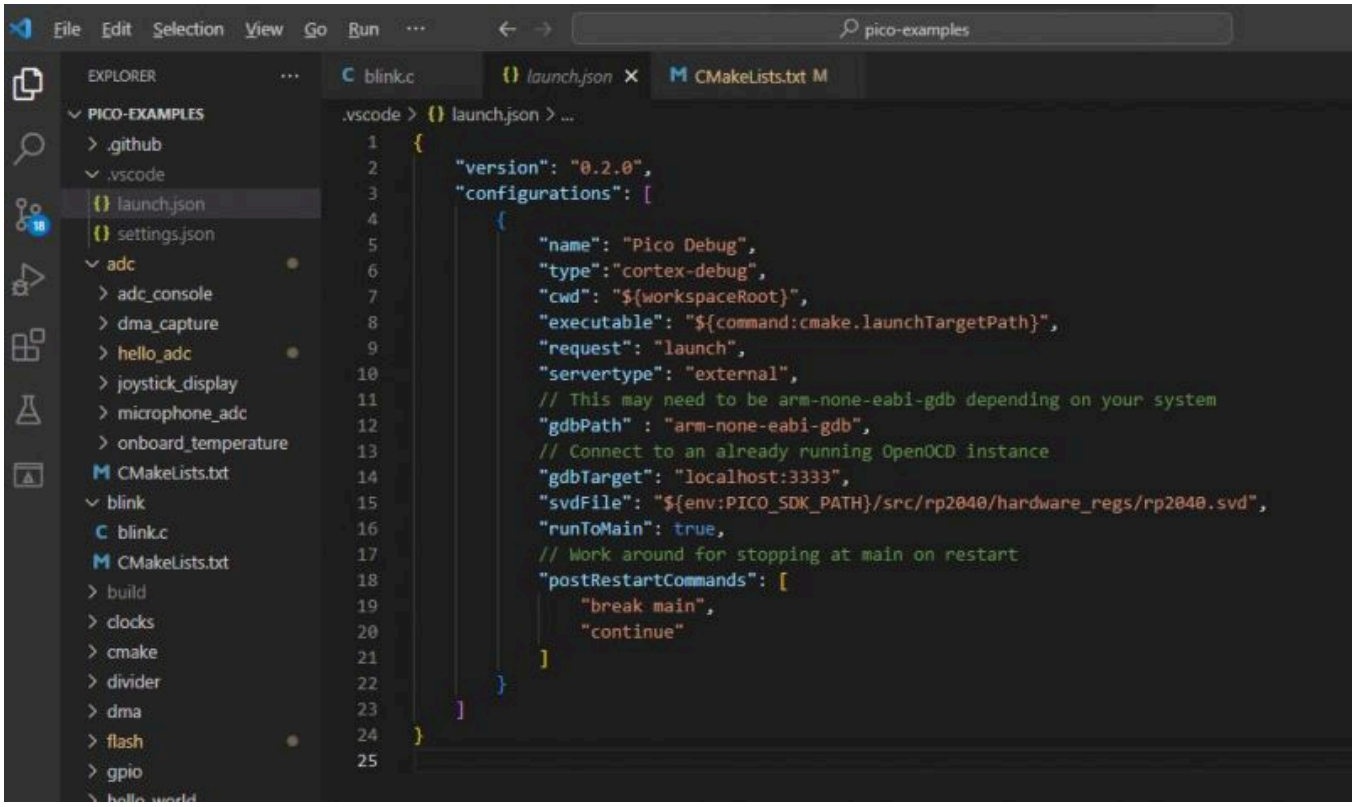
(/wiki/File:Raspberry_Pi_Debug_Probe010.jpg)

2. Use VSC to open the pico-example file folder and open the blink demo.

3. Use F1 to input the following commands:

```
open 'launch.json'
```

4. Put the following content in after opening.



(/wiki/File:Raspberry_Pi_Debug_Probe014.jpg)

- If it is Windows, please input:

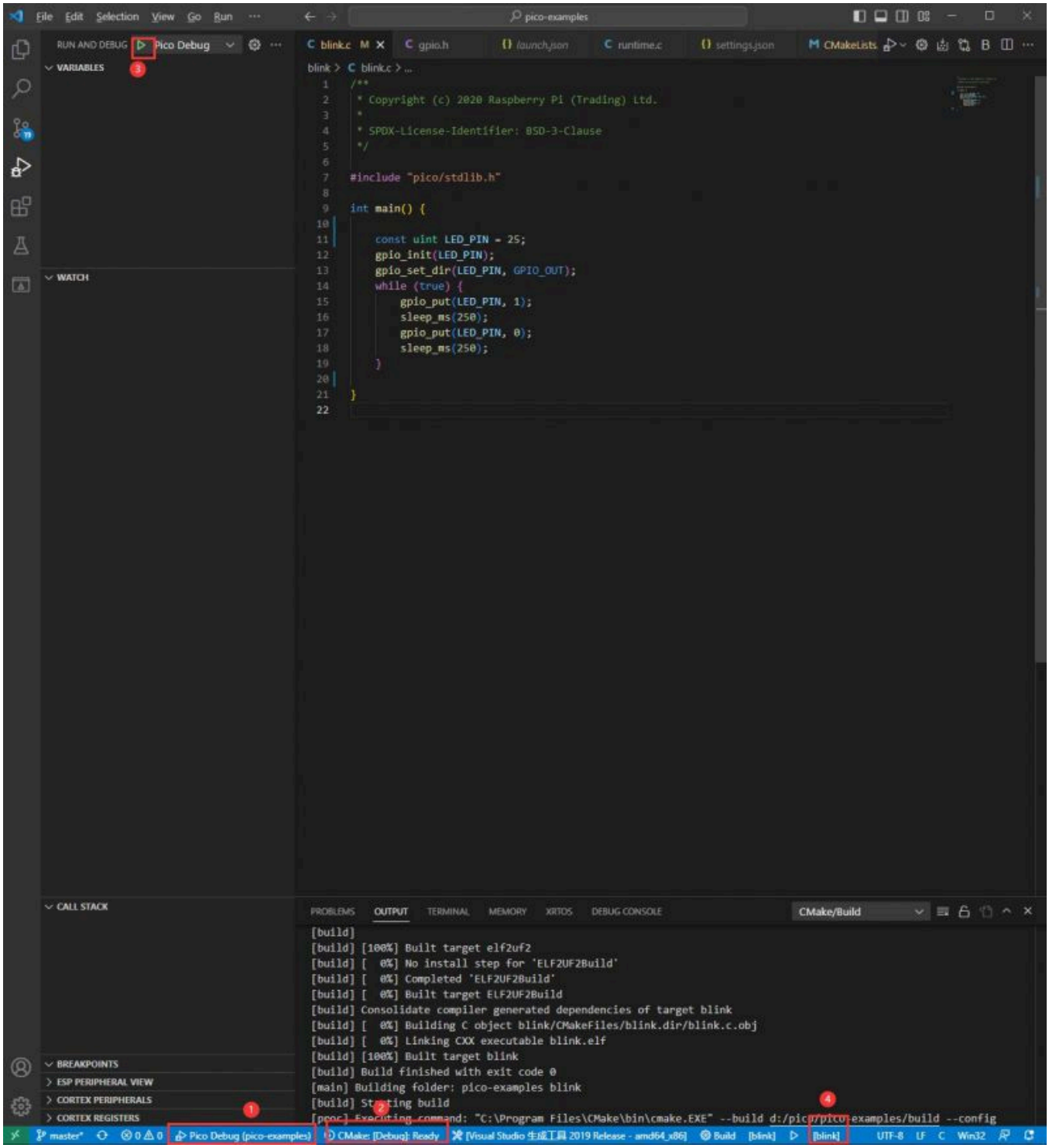
```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Pico Debug",
            "type":"cortex-debug",
            "cwd": "${workspaceRoot}",
            "executable": "${command:cmake.launchTargetPath}",
            "request": "launch",
            "servertype": "external",
            // This may need to be arm-none-eabi-gdb depending on your system
            "gdbPath" : "gdb",
            // Connect to an already running OpenOCD instance
            "gdbTarget": "localhost:3333",
            "svdFile": "${env:PICO_SDK_PATH}/src/rp2040/hardware_regs/rp2040.svd",
            "runToMain": true,
            // Work around for stopping at main on restart
            "postRestartCommands": [
                "break main",
                "continue"
            ]
        }
    ]
}
```

- If it is a Linux system, please input:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Pico Debug",
            "type":"cortex-debug",
            "cwd": "${workspaceRoot}",
            "executable": "${command:cmake.launchTargetPath}",
            "request": "launch",
            "servertype": "external",
            // This may need to be arm-none-eabi-gdb depending on your system
            "gdbPath" : "arm-none-eabi-gdb",
            // Connect to an already running OpenOCD instance
            "gdbTarget": "localhost:3333",
            "svdFile": "${env:PICO_SDK_PATH}/src/rp2040/hardware_regs/rp2040.svd",
            "runToMain": true,
            // Work around for stopping at main on restart
            "postRestartCommands": [
                "break main",
                "continue"
            ]
        }
    ]
}
```

- The difference between them is the gdb calls are different.

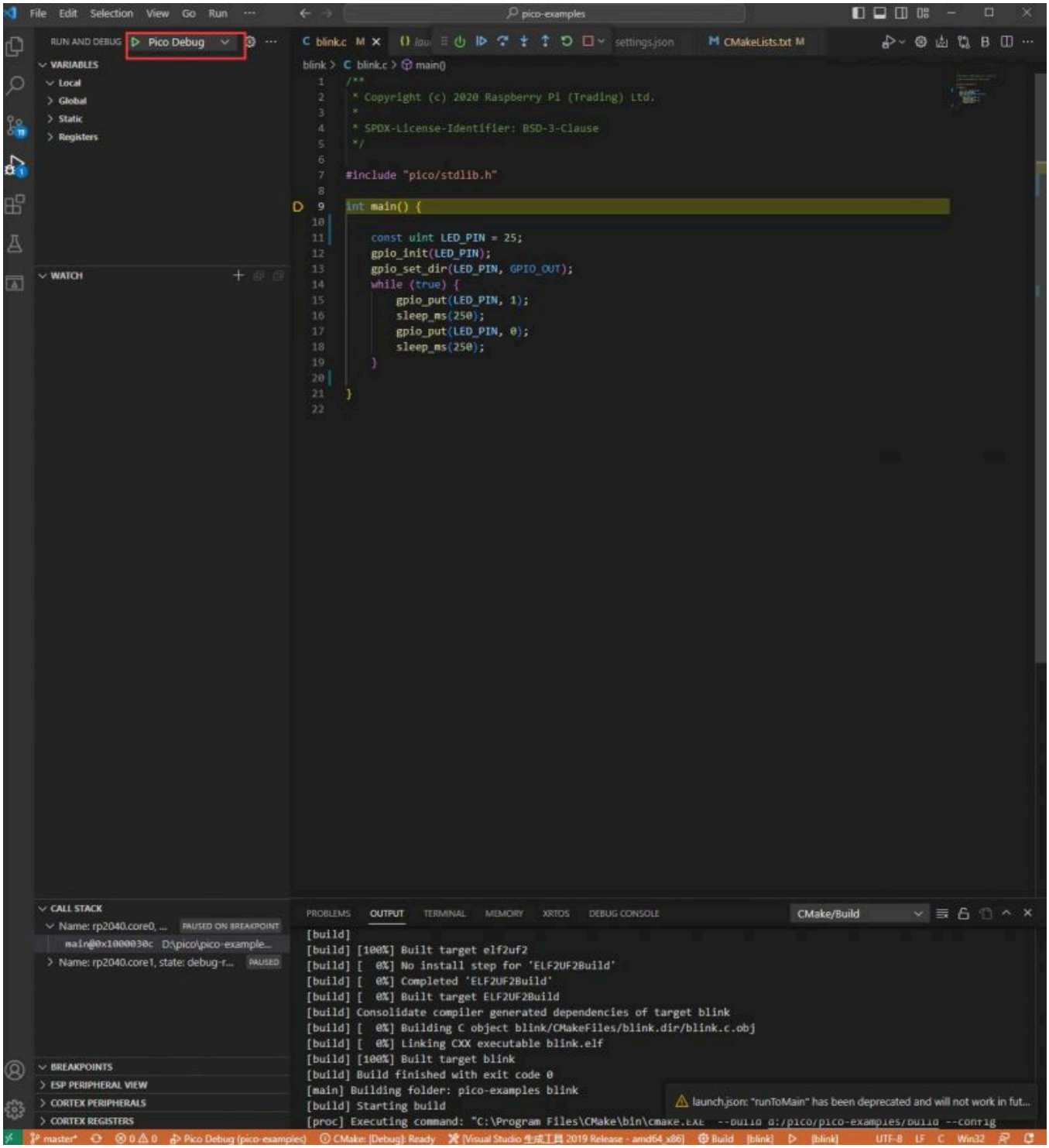
5. Enter the run and debug interface shortcut key Ctrl + Shift + D.



(/wiki/File:Raspberry_Pico_Debug_Probe07.jpg)

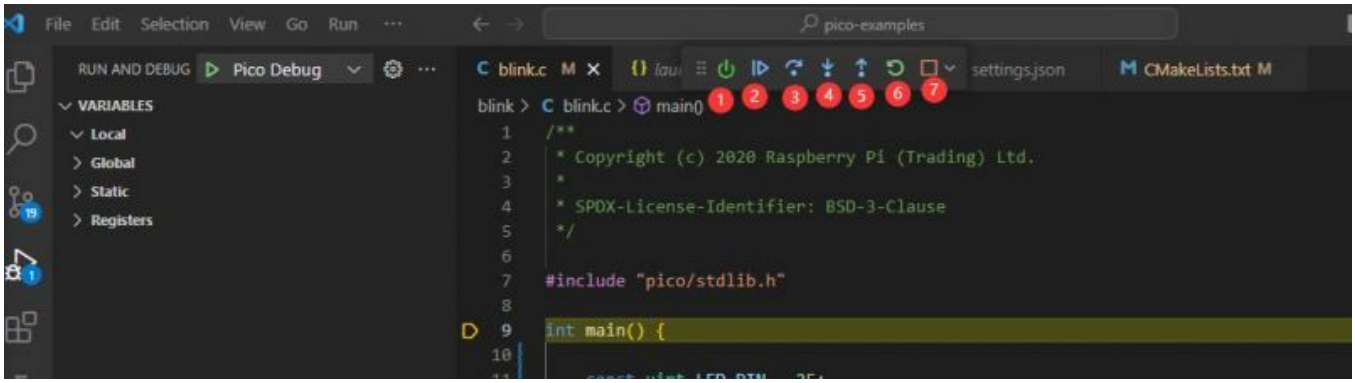
- ①Select Pico Debug as the debugger.
- ②Select CMake as the debug mode.
- ③Start to debug the key, the shortcut key F5.
- ④Choose the debug object as blink.

6. Click debug key to enter the debug mode, the shortcut key is F5.



(/wiki/File:Raspberry_Pi_Debug_probe90.jpg)

7. The Debug Tools show:



(/wiki/File:Raspberry_Pi_Debug_Probe91.jpg)

- ①Restart the device.
- ②Continue to run the demo.
- ③Execute.
- ④Enter the function to run.
- ⑤Jumping out of function runs.
- ⑥Stop debugging.

8. Continue to run the demo, press F5 and you can see Pico runs the blink demo.

Resource

Document

- Schematic (https://files.waveshare.com/wiki/RP2040-GEEK/RP2040-GEEK-Schematic.pdf)

Demo

- Demo code (https://drive.google.com/file/d/1xVKmKce5kZ9awlOl6vQy6H-rzDhEXHrH/view?usp=sharing)

Official Resources

Raspberry Pi Official Datasheet

- Get Started With Pico Manual (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)
- RP2040 Datasheet (https://files.waveshare.com/upload/f/fd/Rp2040_datasheet.pdf)

Raspberry Pi Open-source Demo

- Raspberry Pi Official C/C++ demo (github) (https://github.com/raspberrypi/pico-examples/)
- Raspberry Pi micropython demo (github) (https://github.com/raspberrypi/pico-micropython-examples)
- picoprobe source code (github) (https://github.com/raspberrypi/picoprobe)

Development Software

- Zimo221.7z (https://files.waveshare.com/upload/c/c6/Zimo221.7z)
- Image2Lcd.7z (https://files.waveshare.com/upload/3/36/Image2Lcd.7z)
- Font Library Tutorial (https://www.waveshare.com/wiki/Ink_Screen_Font_Library_Tutorial)
- Image Extraction Tutorial (https://www.waveshare.com/wiki/Image_Extraction)

- Thonny Python IDE (Windows V3.3.3) (https://files.waveshare.com/upload/7/73/Thonny-3.3.3.zip)

FAQ

Question:What type of controller was used for the display?

Answer:

RP2040-GEEK LCD controller is ST7789VW.

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