Raspberry Pi Pico Servo Driver Board

Description

It is a servo control expansion board for Raspberry Pi Pico

Features

On-board Raspberry Pi Pico interface for Raspberry Pi Pico series boards supports up to 16-channel servo or PWM outputs. And each channel support 16-bit resolution On-board 5V voltage regulator chip. The output current is up to 3A.

It can be connected to on-board servo on the battery power supply board through the VIN terminal, and interface with common servos such as SG90, MG90S, MG996R for Pico for easy expansion. A complete supporting information manual is provided(example programs such as Raspberry Pi Pico C/C++ and MicroPython)

Product parameters

Working voltage 5V(Pico) or 6~12V(VIN terminal)

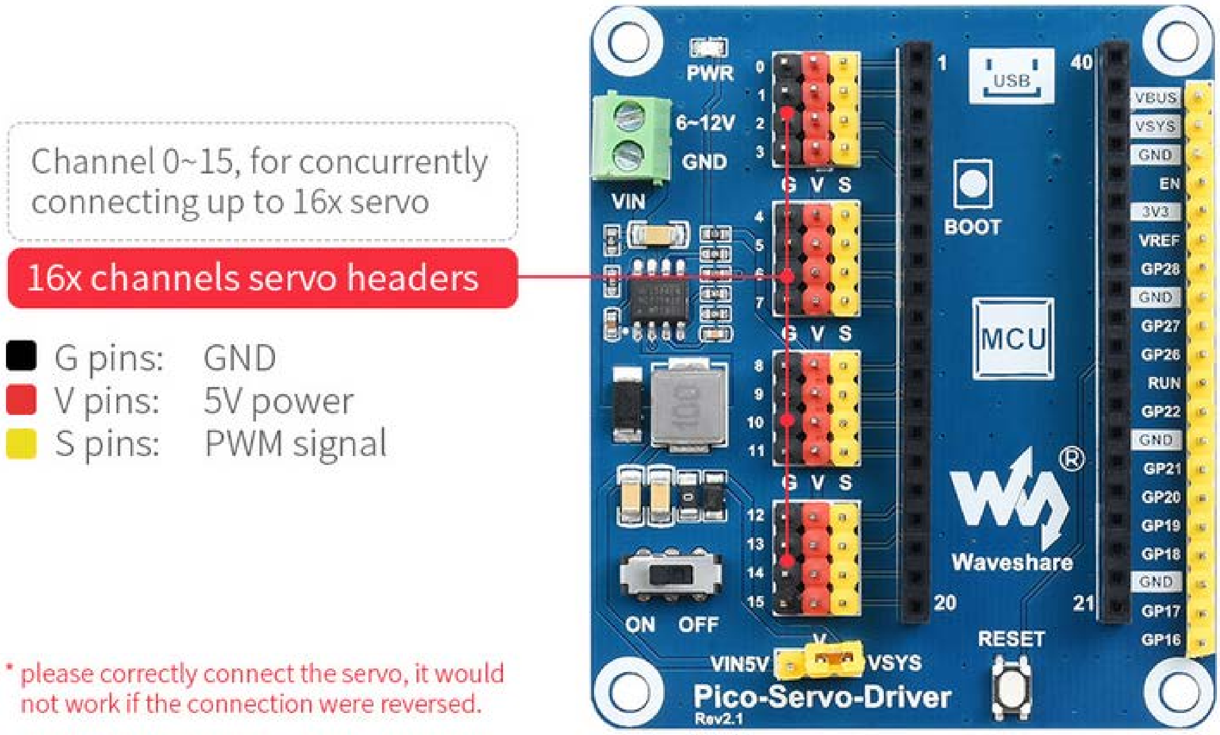
Servo voltage 5V

Logic voltage 3.3V

Control interface GPIO Via diameter 3.0mm

Product size 65 × 56mm

Pins



Wire up

Don’t connect the Pico reversely

Observe an end with silk prints on the module and an end of USB port to determine connection direction.

You can also depend on signals of pins and pins of Pico to determine connection direction.

Programming download

Download via Raspberry Pi, open the Raspberry Pi terminal:

And

```
sudo apt-get install p7zip-full
cd ~
sudo wget
https://www.waveshare.net/w/upload/3/31/Pico_Servo_Driver_Code.7z
7z Pico_Servo_Driver_Code.7z -o./Pico_Servo_Driver_Code.7z
cd ~/Pico_Servo_Driver_Code
```

Click the example program to download directly

Use C via Raspberry Pi

We use the Raspberry Pi. Because cnmake has multiple platforms and can be moved, you can compile on the PC.

Compile under the C directory

```
cd ~/Pico_Servo_Driver_Code/c/
```

Create and enter build directory in the folder and add SDK.

../../pico-sdk is the directory of the SDK.

The example program has build, jsut enter it.

```
cd build
export PICO_SDK_PATH=../../pico-sdk
```

(Note：write the correct pass of your own SDK)

Implement cmake and generate into Makefile files

```
cmake ..
```

Implement make and generate implement files，the first compile will take a while.

```
make -j9
```

After compiling, the uf2 file will be generated.

Press a key of the Pico board，connect the pico board to the Raspberry Pi via a

USB cable and release the key.

Then Raspberry Pi will recognize a drive(RPI-RP2), copy the main.uf2 from the build folder to the drive(RPI-RP2).

```
cp main.uf2 /media/pi/RPI-RP2/
```

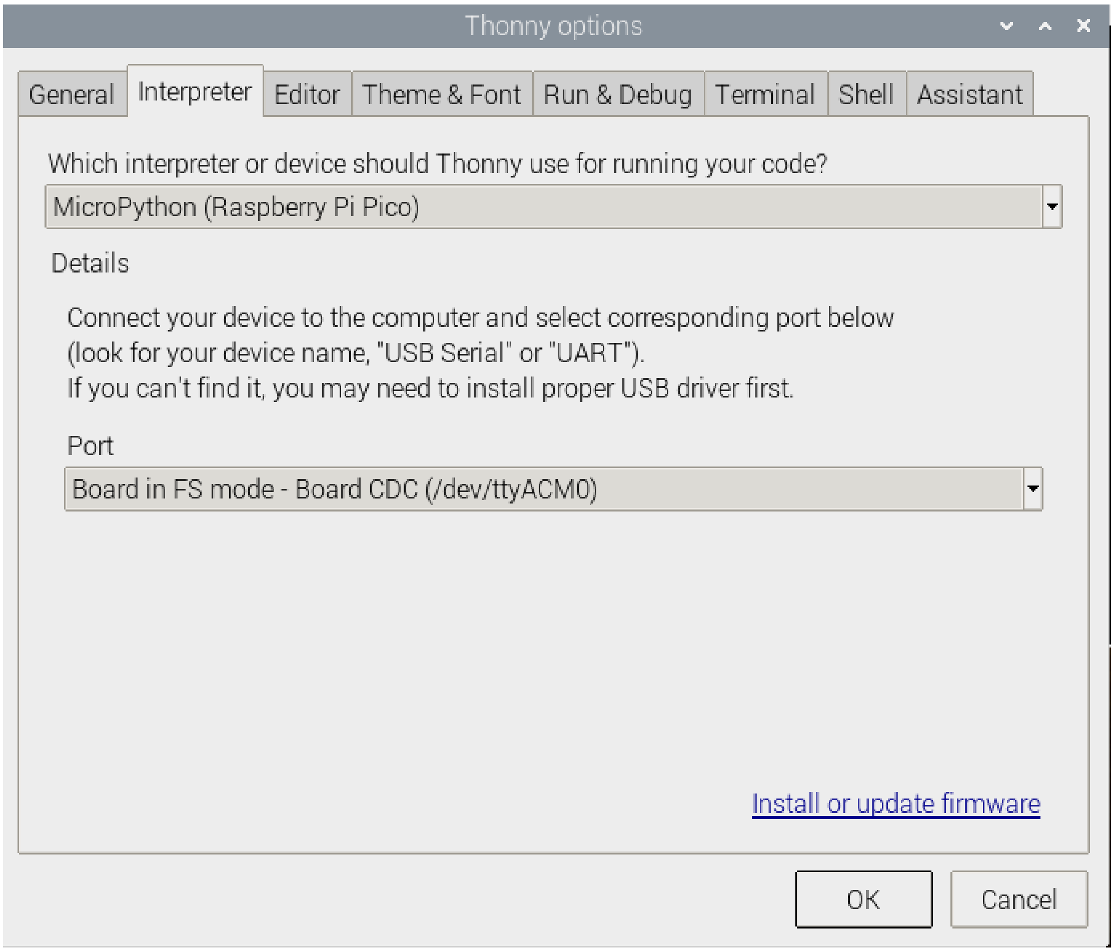
Python

1. Update the firmware of Micropython，copy the pico_micropython_xxxxx.uf2 file to the pico.

2. Open Thonny IDE on the Raspberry Pi(click Raspberry Pi-> Programming -> Thonny Python IDE), you can check the version information: Help->About Thonny

Make sure this version contain Pico support package, and click Tools -> Options... -> Interpreter, then select MicroPython(Raspberry Pi Pico and the ttyACM0 port

As shown below;



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

Board in FS mode - Board CDC (/dev/ttyACM0)

Install or update firmware

Pico-lcd-0.96-img-config2.png

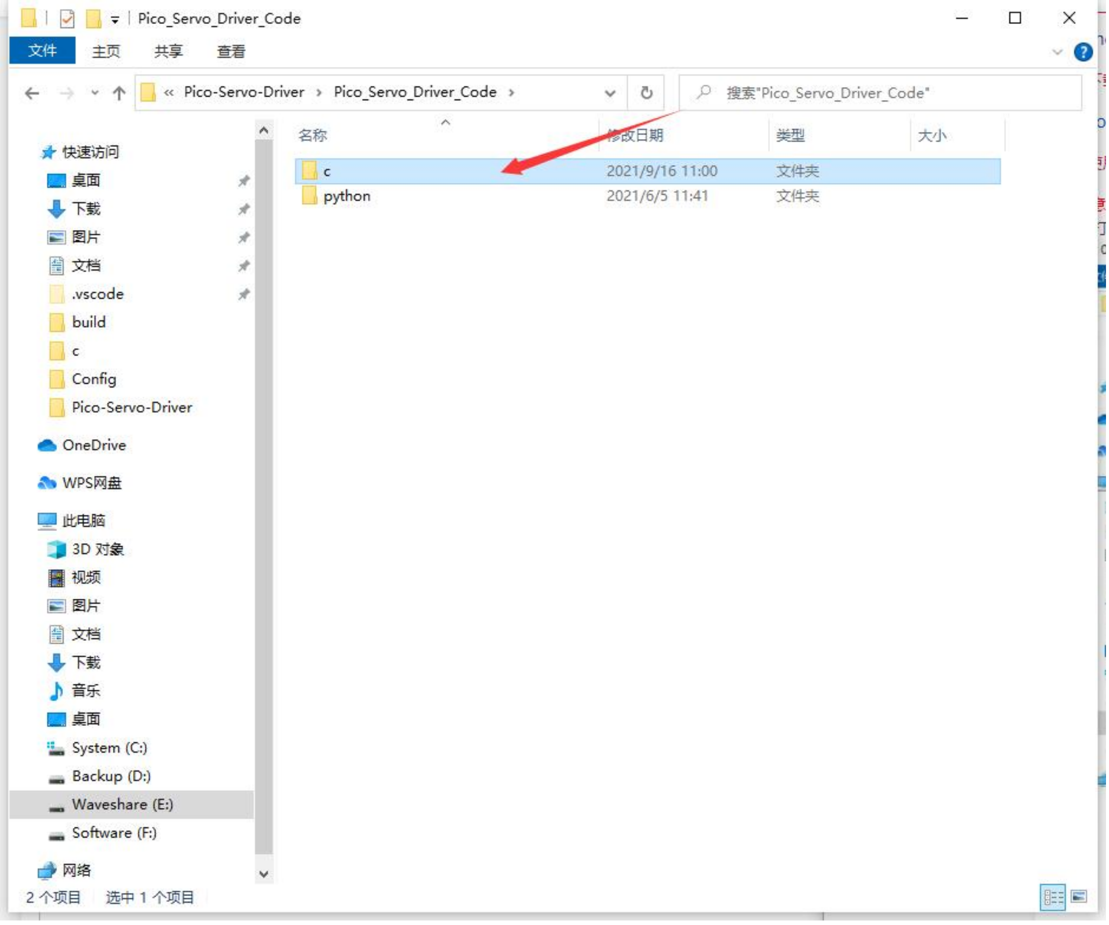
If the Thonny doesn’t have the pico support package, enter the following comender to update Thonny IDE sudo apt upgrade thonny

Click File->Open...->python/Pico_Servo_Driver_Code/python/servo.py then program the script

The servo will rotate from 0° to 180°when connected, repeat three times.

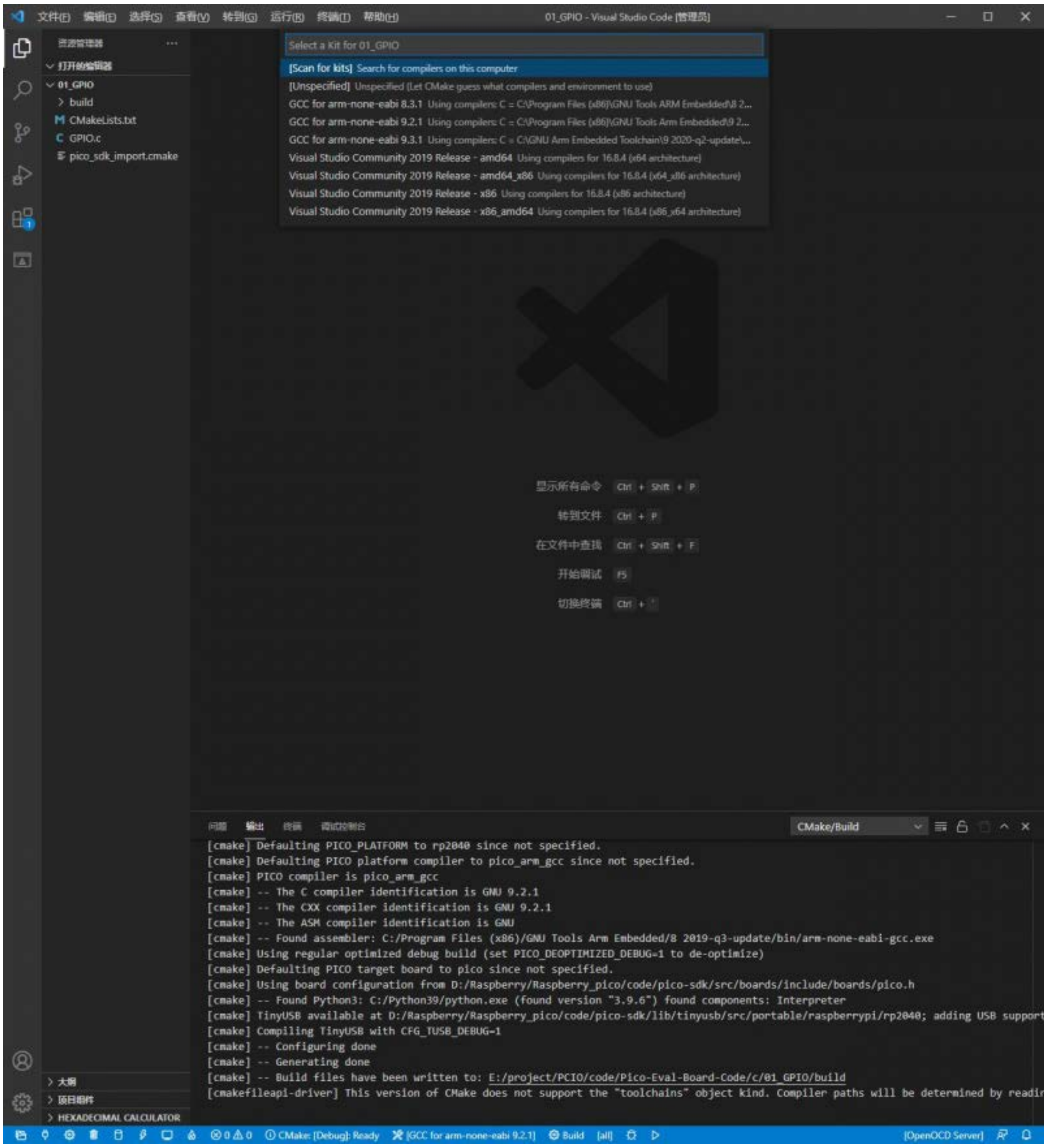
Windows

Open C folder



名称	修改日期	类型	大小
c	2021/9/16 11:00	文件夹	
python	2021/6/5 11:41	文件夹	

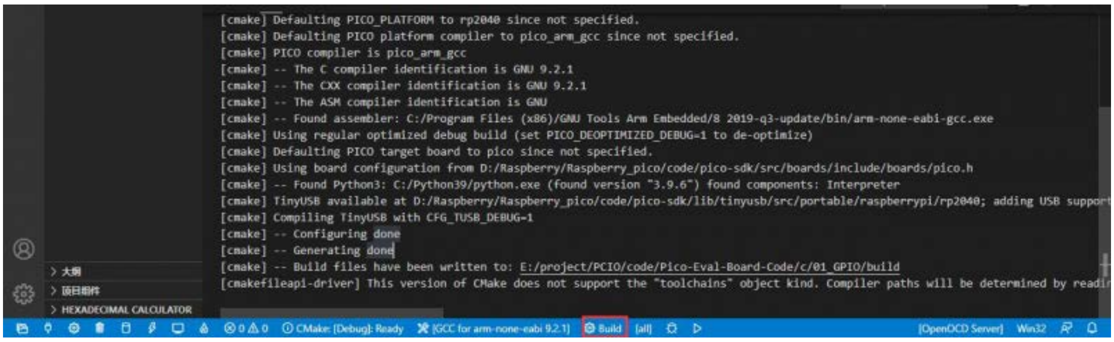
Open with the Vs coed and select the compiling tool,


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
01_GPIO - Visual Studio Code [管理员]
资源管理器
打开的编辑器
01_GPIO
build
CMakeLists.txt
GPIO.c
pico_sdk_import.cmake
显示所有命令 Ctrl + Shift + P
转到文件 Ctrl + P
在文件中查找 Ctrl + Shift + F
开始调试 F5
切换终端 Ctrl + `
问题 输出 终端 调试控制台
CMake/Build
大纲
项目组件
HEXADECIMAL CALCULATOR
CMake: [Debug]: Ready
[GCC for arm-none-eabi 9.2.1]
Build
[all]
[OpenOCD Server]

```
Select a Kit for 01_GPIO
[Scan for kits] Search for compilers on this computer
[Unspecified] Unspecified (Let CMake guess what compilers and environment to use)
GCC for arm-none-eabi 8.3.1 Using compilers: C = C:\Program Files (x86)\GNU Tools ARM Embedded\8 2...
GCC for arm-none-eabi 9.2.1 Using compilers: C = C:\Program Files (x86)\GNU Tools Arm Embedded\9 2...
GCC for arm-none-eabi 9.3.1 Using compilers: C = C:\GNU Arm Embedded Toolchain\9 2020-q2-update\...
Visual Studio Community 2019 Release - amd64 Using compilers for 16.8.4 (x64 architecture)
Visual Studio Community 2019 Release - amd64_x86 Using compilers for 16.8.4 (x64_x86 architecture)
Visual Studio Community 2019 Release - x86 Using compilers for 16.8.4 (x86 architecture)
Visual Studio Community 2019 Release - x86_amd64 Using compilers for 16.8.4 (x86_x64 architecture)
```

```
[cmake] Defaulting PICO_PLATFORM to rp2040 since not specified.
[cmake] Defaulting PICO platform compiler to pico_arm_gcc since not specified.
[cmake] PICO compiler is pico_arm_gcc
[cmake] -- The C compiler identification is GNU 9.2.1
[cmake] -- The CXX compiler identification is GNU 9.2.1
[cmake] -- The ASM compiler identification is GNU
[cmake] -- Found assembler: C:/Program Files (x86)/GNU Tools Arm Embedded/8 2019-q3-update/bin/arm-none-eabi-gcc.exe
[cmake] Using regular optimized debug build (set PICO_DEOPTIMIZED_DEBUG=1 to de-optimize)
[cmake] Defaulting PICO target board to pico since not specified.
[cmake] Using board configuration from D:/Raspberry/Raspberry_pico/code/pico-sdk/src/boards/include/boards/pico.h
[cmake] -- Found Python3: C:/Python39/python.exe (found version "3.9.6") found components: Interpreter
[cmake] TinyUSB available at D:/Raspberry/Raspberry_pico/code/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; adding USB support
[cmake] Compiling TinyUSB with CFG_TUSB_DEBUG=1
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: E:/project/PCIO/code/Pico-Eval-Board-Code/c/01_GPIO/build
[cmakefileapi-driver] This version of CMake does not support the "toolchains" object kind. Compiler paths will be determined by readir
```

Click compile

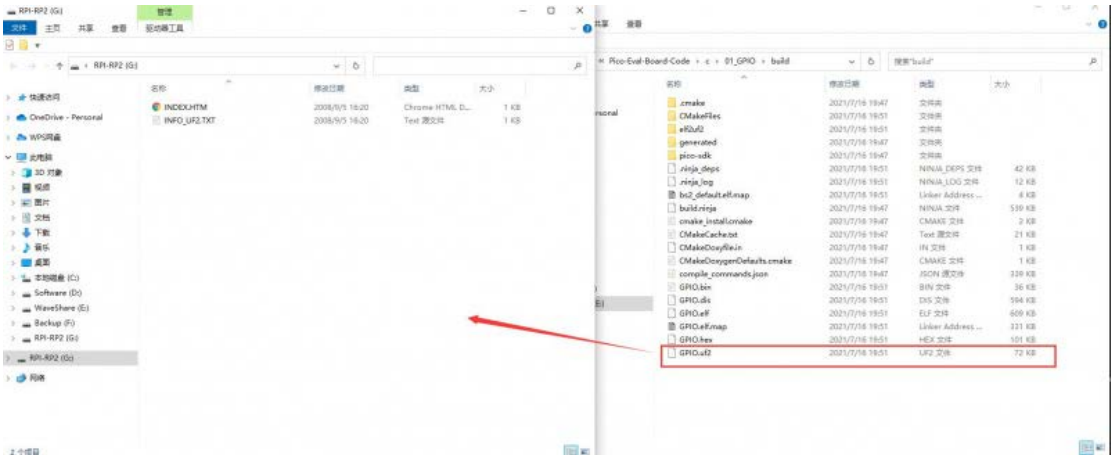


```
[cmake] Defaulting PICO_PLATFORM to rp2040 since not specified.
[cmake] Defaulting PICO platform compiler to pico_arm_gcc since not specified.
[cmake] PICO compiler is pico_arm_gcc
[cmake] -- The C compiler identification is GNU 9.2.1
[cmake] -- The CXX compiler identification is GNU 9.2.1
[cmake] -- The ASM compiler identification is GNU
[cmake] -- Found assembler: C:/Program Files (x86)/GNU Tools Arm Embedded/8 2019-q3-update/bin/arm-none-eabi-gcc.exe
[cmake] Using regular optimized debug build (set PICO_DEOPTIMIZED_DEBUG=1 to de-optimize)
[cmake] Defaulting PICO target board to pico since not specified.
[cmake] Using board configuration from D:/Raspberry/Raspberry_pico/code/pico-sdk/src/boards/include/boards/pico.h
[cmake] -- Found Python3: C:/Python39/python.exe (found version "3.9.6") found components: Interpreter
[cmake] TinyUSB available at D:/Raspberry/Raspberry_pico/code/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; adding USB support
[cmake] Compiling TinyUSB with CFG_TUSB_DEBUG=1
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: E:/project/PCIO/code/Pico-Eval-Board-Code/c/01_GPIO/build
[cmakefileapi-driver] This version of CMake does not support the "toolchains" object kind. Compiler paths will be determined by readir
```

1. Press the Reset button on the Pico-Eval-Board to reset the Pico, first press the BOOTSEL button then press the RUN button and release the Reset button. The

Pico will enter the disk mode directly.

2. Drag the UF2 file under the build file and drop to the RPI-RP2 drive letter



RPI-RP2 (G:)

名称	修改日期	类型	大小
INDEX.HTM	2008/9/5 16:20	Chrome HTML D...	1 KB
INFO_UF2.TXT	2008/9/5 16:20	Text 源文件	1 KB

Pico-Eval-Board-Code › c › 01_GPIO › build

名称	修改日期	类型	大小
.cmake	2021/7/16 19:47	文件夹	
CMakeFiles	2021/7/16 19:51	文件夹	
elf2uf2	2021/7/16 19:51	文件夹	
generated	2021/7/16 19:47	文件夹	
pico-sdk	2021/7/16 19:47	文件夹	
.ninja_deps	2021/7/16 19:51	NINJA_DEPS 文件	42 KB
.ninja_log	2021/7/16 19:51	NINJA_LOG 文件	12 KB
bs2_default.elf.map	2021/7/16 19:51	Linker Address ...	4 KB
build.ninja	2021/7/16 19:47	NINJA 文件	539 KB
cmake_install.cmake	2021/7/16 19:47	CMAKE 文件	2 KB
CMakeCache.txt	2021/7/16 19:47	Text 源文件	21 KB
CMakeDoxyfile.in	2021/7/16 19:47	IN 文件	1 KB
CMakeDoxygenDefaults.cmake	2021/7/16 19:47	CMAKE 文件	1 KB
compile_commands.json	2021/7/16 19:47	JSON 源文件	339 KB
GPIO.bin	2021/7/16 19:51	BIN 文件	36 KB
GPIO.dis	2021/7/16 19:51	DIS 文件	594 KB
GPIO.elf	2021/7/16 19:51	ELF 文件	609 KB
GPIO.elf.map	2021/7/16 19:51	Linker Address ...	331 KB
GPIO.hex	2021/7/16 19:51	HEX 文件	101 KB
GPIO.uf2	2021/7/16 19:51	UF2 文件	72 KB

3. Pico starts running the corresponding program

Code explanation

Hardware interfaces

Since hardware platforms and inner structure are different, you can check in the corresponding directories

You can check definitions in DEV_Config.c(.h), under the directory: ...\c\lib\Config

Date type：

```
#define UBYTE     uint8_t
#define UWORD     uint16_t
#define UDOUBLE uint32_t
```

module initialization and exit

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
```

PWM initialization：

```
void PWM_initialization()；
```

interrupt handler function：

```
void on_pwm_wrap()；
```

Define the channel used

```
#define CHANNE_N 0xFFFF        // 0x0001 means 0 channel is open, 0x0000 means all channels are closed
```

Angles of rotation

```
#define ROTATE_0          1700             //rotate to 0°
#define ROTATE_45         3300             //rotate to 45°
#define ROTATE_90         4940             //rotate to 90°
#define ROTATE_135        6600             //rotate to 135°
#define ROTATE_180        8250          //rotate to 180°
```

python

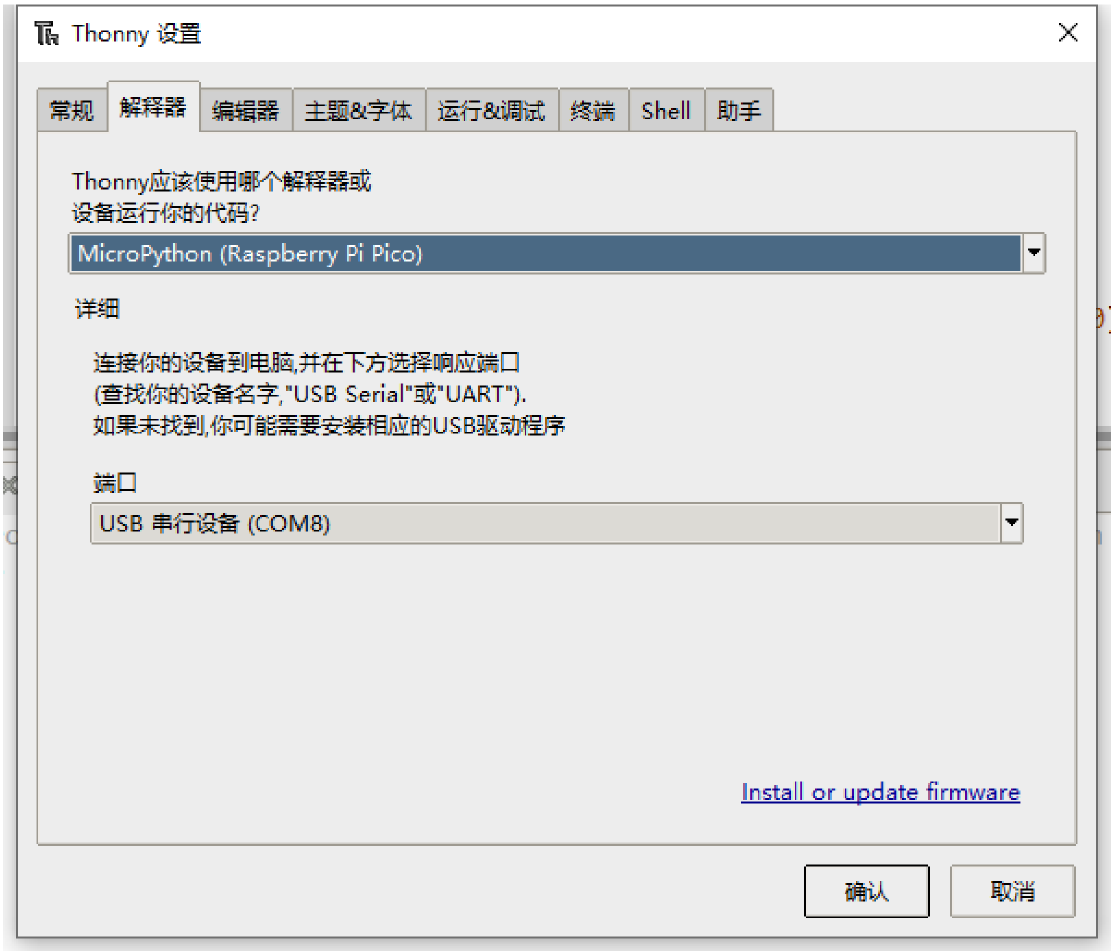
Windows environment

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

Download the pico_micropython_xxxxx.uf2, then copy it to the drive(RPI-RP2).

Open Thonny IDE(note: use the latest Thonny, otherwise, the Pico support package is not included. The latest version under Windows is v3.3.3

Click tool->setting->interpreter, then select the corresponding port of Pico



Thonny应该使用哪个解释器或
设备运行你的代码?

MicroPython (Raspberry Pi Pico)

详细

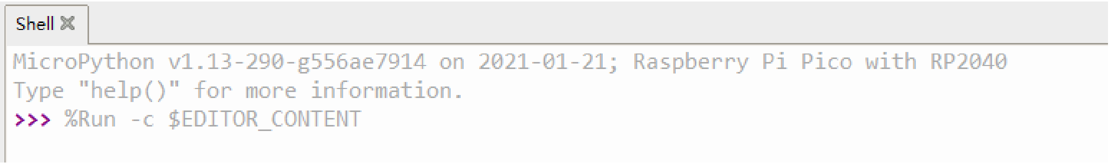
连接你的设备到电脑,并在下方选择响应端口
(查找你的设备名字,"USB Serial"或"UART").
如果未找到,你可能需要安装相应的USB驱动程序

端口

USB 串行设备 (COM8)

Click file->open->servo.py and click run

The following picture indicates that the program has run.



```
MicroPython v1.13-290-g556ae7914 on 2021-01-21; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
```

The experiment result is as same as the program C