

# RP2040-Zero

来自Waveshare Wiki

跳转至： [导航](#)、 [搜索](#)

## 说明

### 产品概述

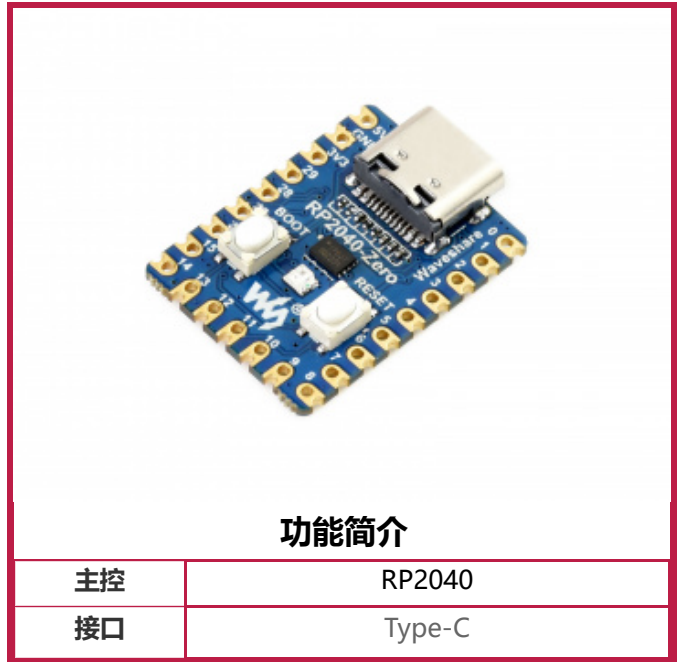
RP2040-Zero-Zero开发板

采用 Raspberry Pi 研发的 RP2040作为核心,并在极小板型下引出全部未占用引脚

采用半孔工艺，可以焊接接在你自定义的板子上

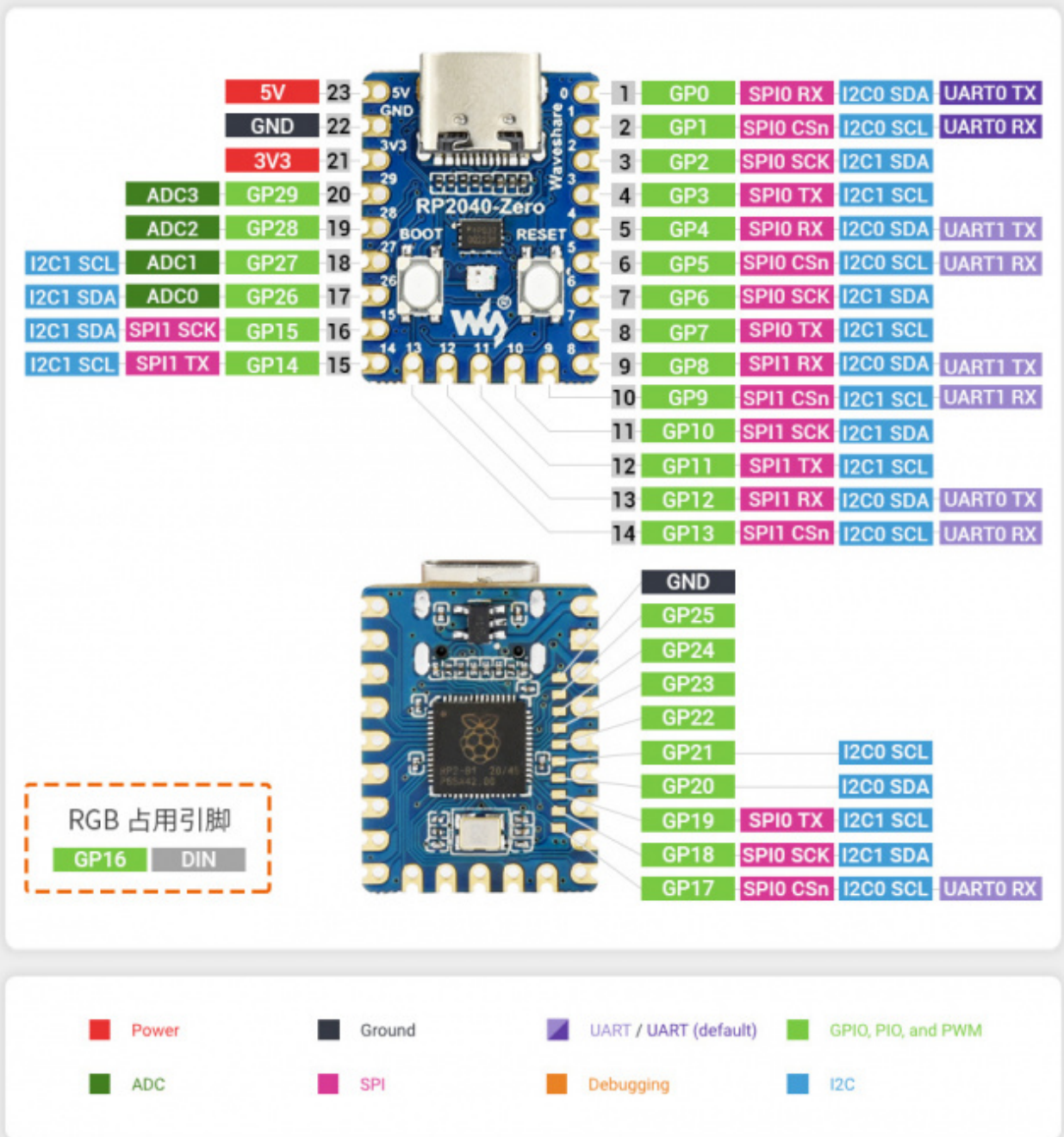
### 产品特性

- 采用了 Raspberry Pi 官方设计的 RP2040 微控制器芯片
- 搭载了双核 ARM Cortex M0 + 处理器，运行频率高达 133MHz 灵活时钟
- 内置了 264KB 的 SRAM 和 2MB 的板载 Flash
- 采用Type-C接口，紧跟时代潮流，无需纠结正反插
- 邮票孔设计，可直接焊接集成到用户自主设计的底板上
- USB1.1 主机和设备支持
- 支持低功耗睡眠和休眠模式
- 可通过 USB 识别为大容量存储器进行拖放式下载程序
- RP2040 的 29 个 GPIO 引脚引出（20 个可通过排针引出，其余仅可焊接引出）
- 多种硬件外设
  - 2 个 SPI
  - 2 个 I2C
  - 2 个 UART
  - 4 个 12 位 ADC
  - 16 个可控 PWM 通道
- 精确的片上时钟和定时器
- 温度传感器
- 片上加速浮点库

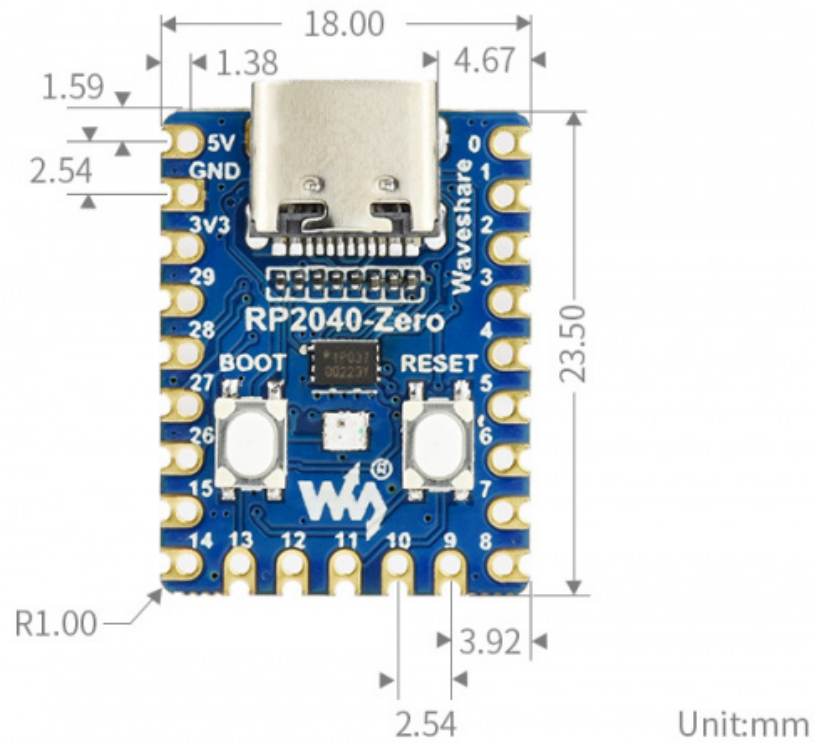


- 8 个可编程 I/O (PIO) 状态机，用于自定义外设支持

## 引脚分布



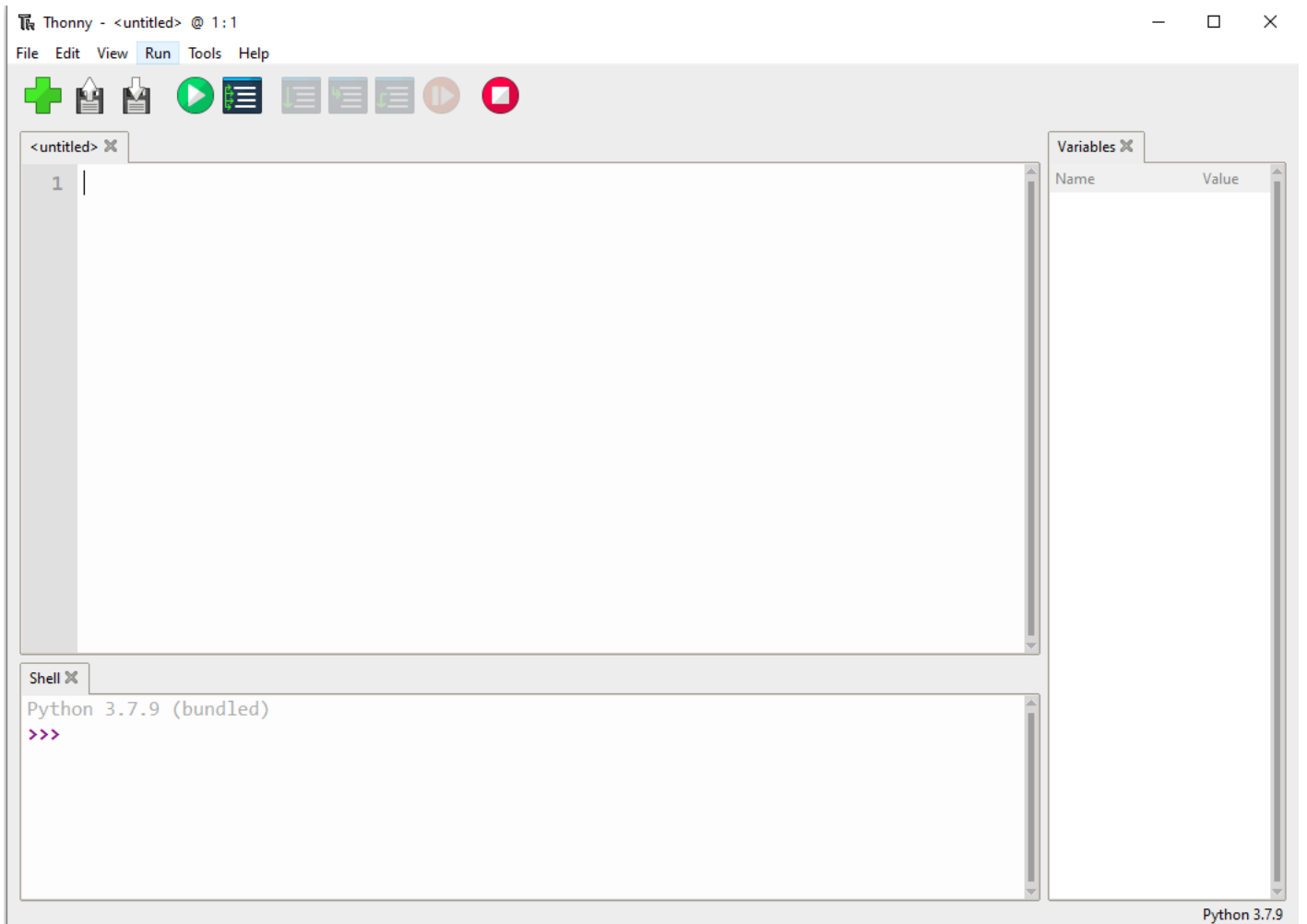
## 尺寸图



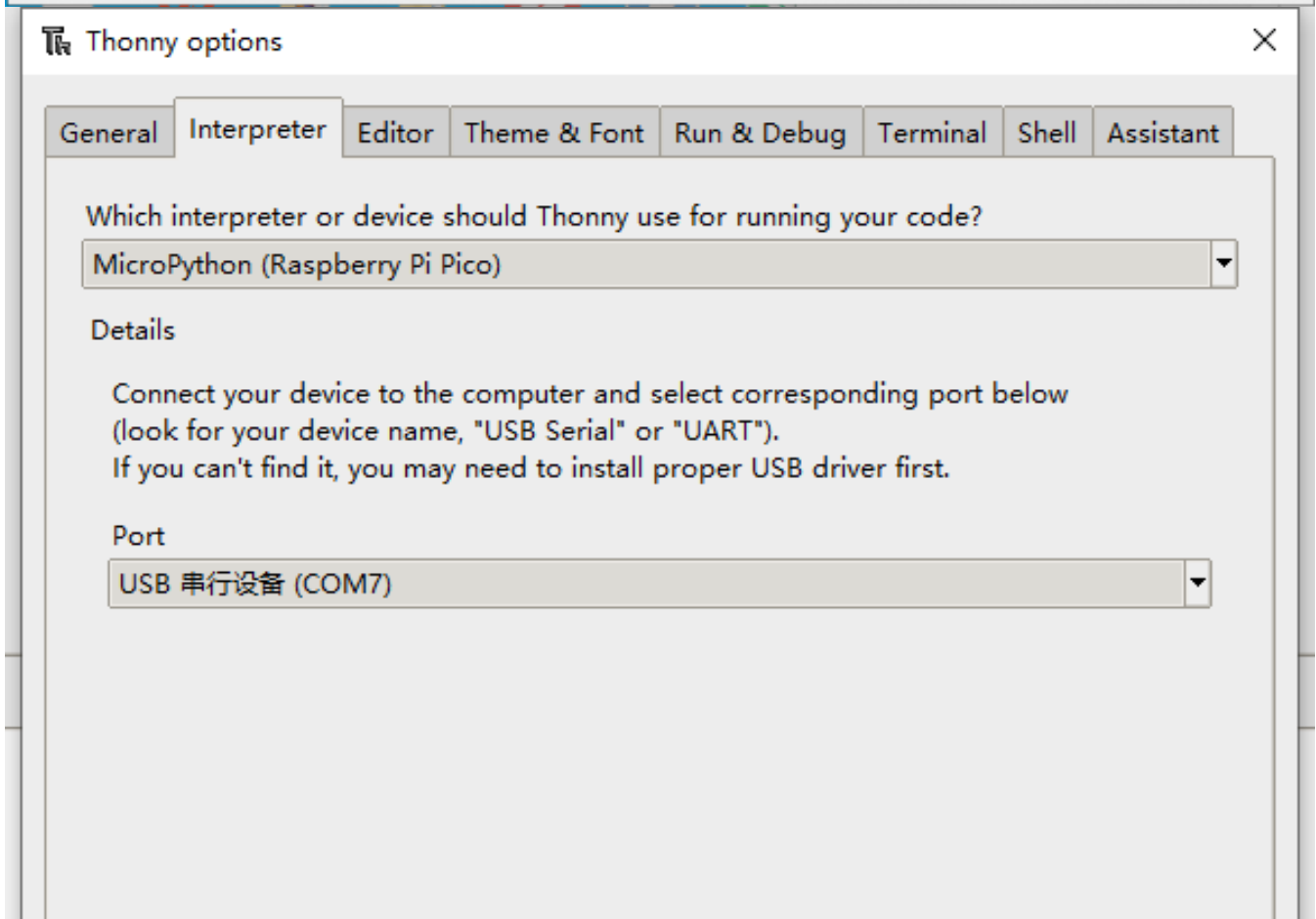
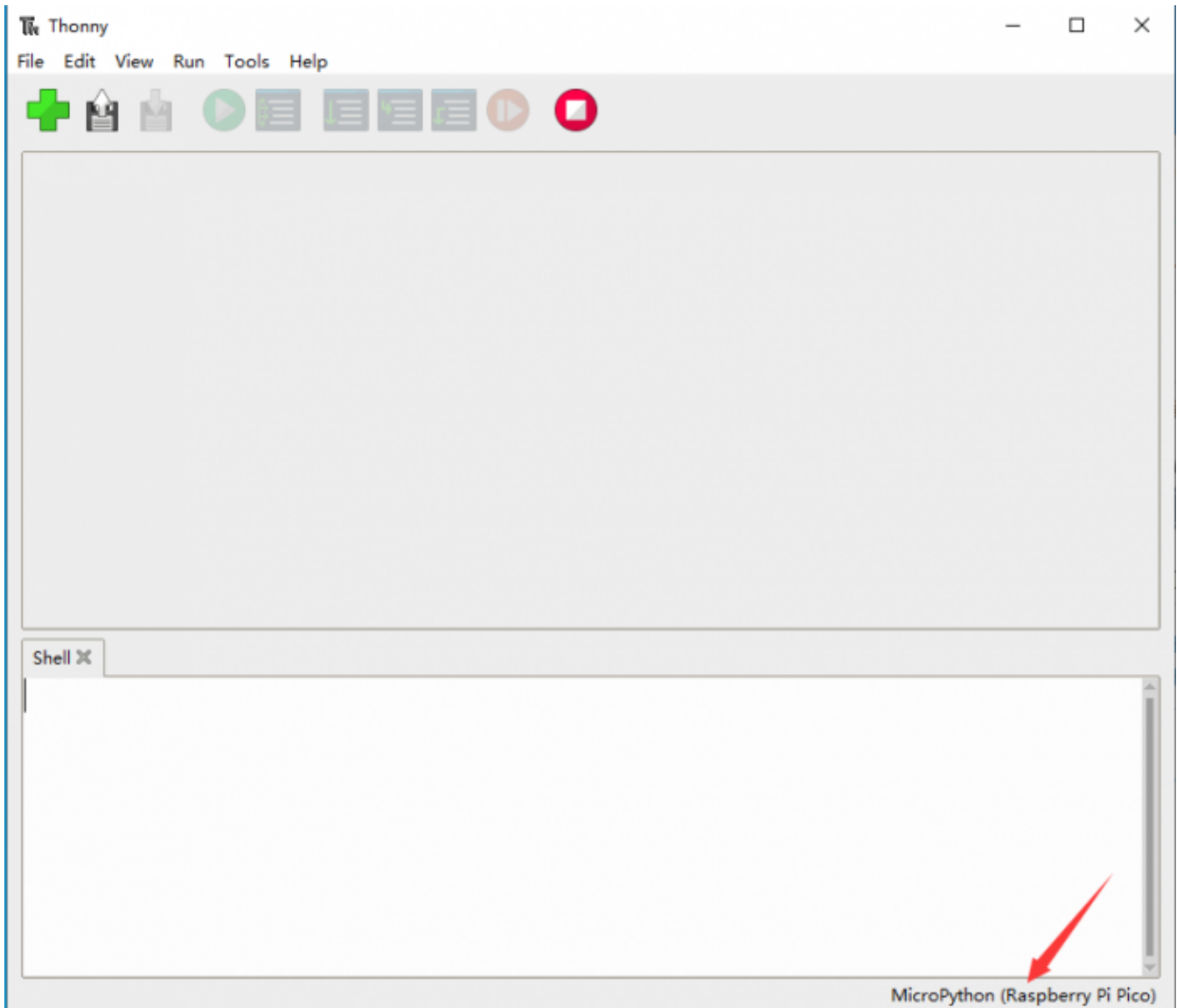
## 软件环境配置

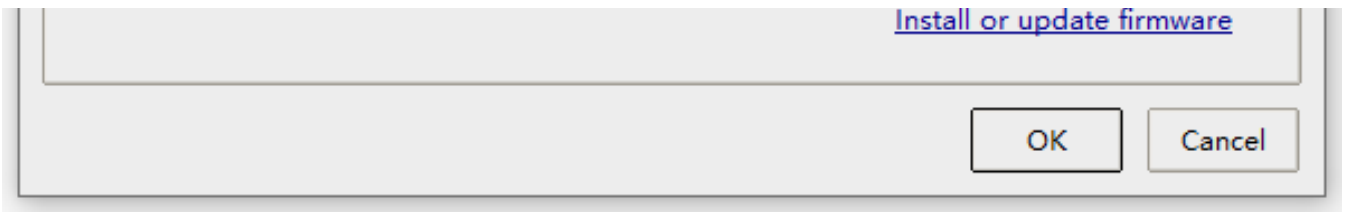
为了方便在电脑上使用MicroPython开发Pico板，建议下载Thonny IDE.

- 下载Thonny IDE 并按照步骤安装
  - Thonny IDE下载链接 (Windows版本)
  - Thonny 官网
- 安装完成之后，第一次要配置语言和主板环境，由于我们是为了使用Pico，所以注意主板环境选择Raspberry Pi 选项。



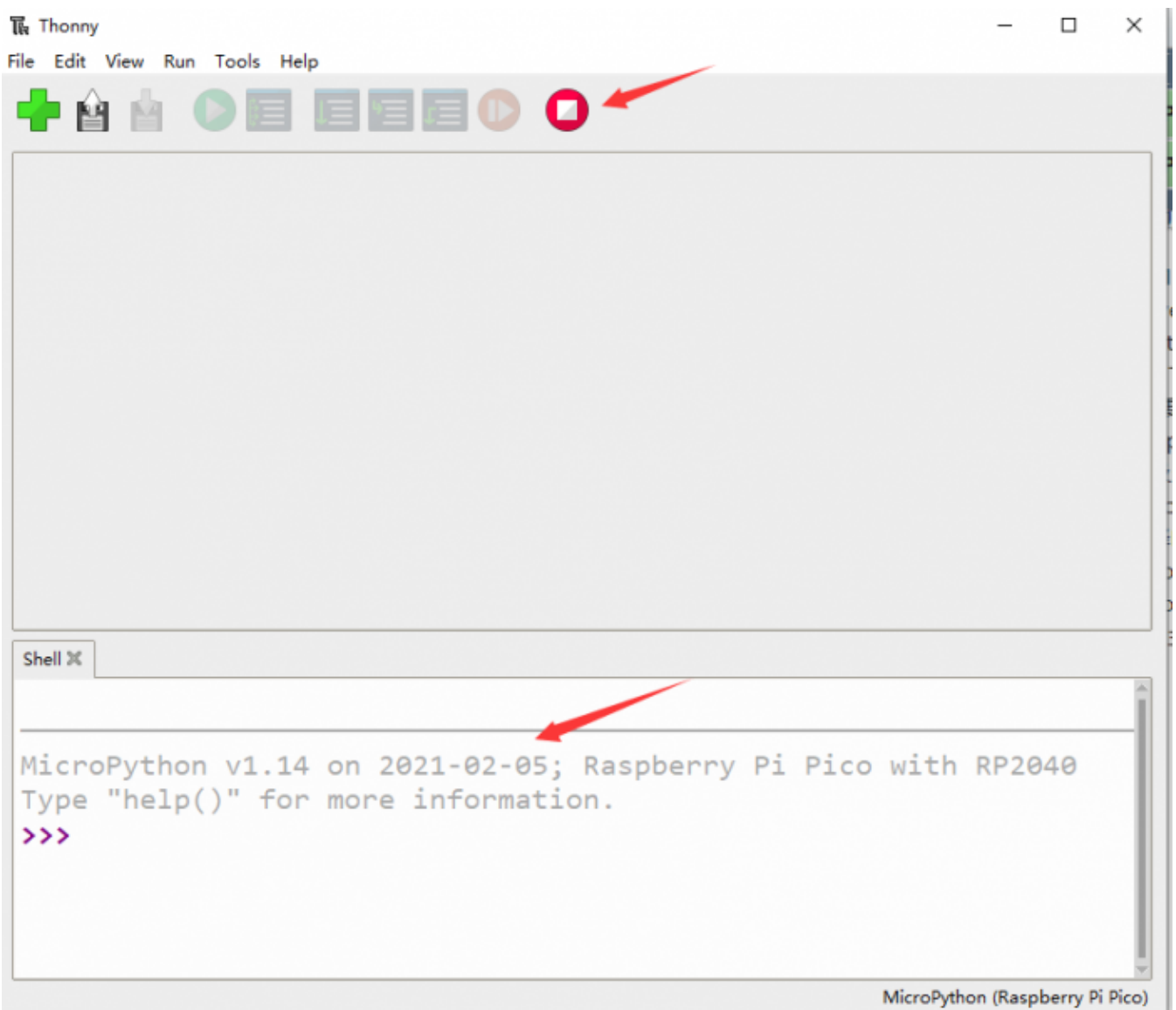
- 配置Micrpython环境及选择Pico端口。
  - 先将Raspberry Pi Pico 接入电脑，左键点击Thonny右下角的配置环境选项--》选择 configure interpreter
  - 在弹出的窗口栏中选择MicroPython(Raspberry Pi Pico),同时选择对应的端口。





- 点击ok后返回到Thonny主界面，下载固件库 到Pico里面，然后点击停止按钮，在Shell窗口中即可显示当前使用到的环境。
- Pico在windows下载固件库方法: 按住BOOT键后连接电脑后，松开BOOT键，电脑会出现一个可移动磁盘，将固件库复制进去即可。
- RP2040在windows下载固件库方法: 连接电脑后，同时按下BOOT键跟RESET键,先松开RESET键再松开BOOT键，电脑会出现一个可移动磁盘，

将固件库复制进去即可（用Pico的方式也可以）。



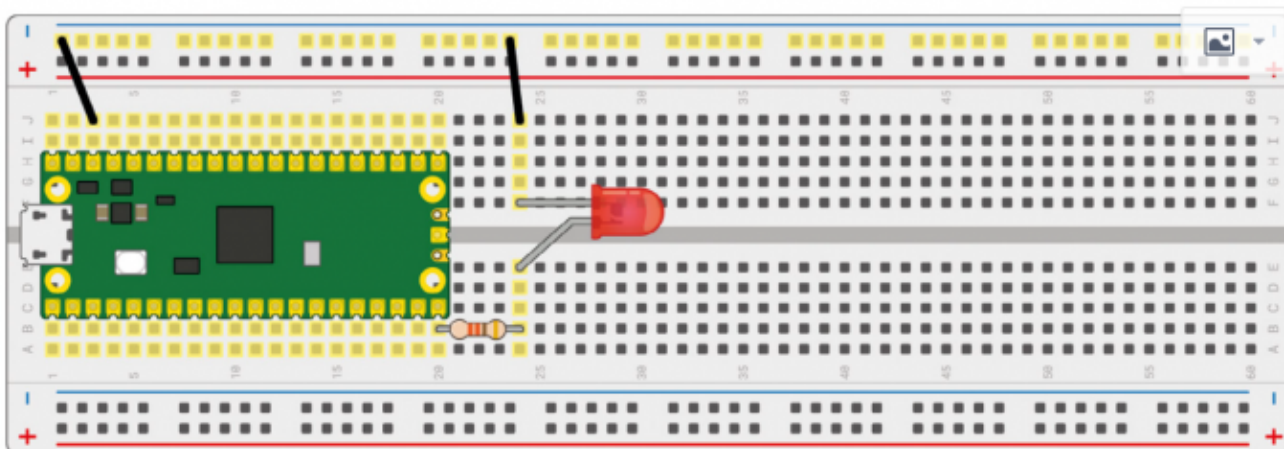
## 示例实验

- 下载示例程序 到电脑桌面即可进行一些几个有趣的实验。

## External LED 实验

- 按照下图连接好硬件，连接好接入电脑的Micro USB，在Thonny打开示例程序Lesson-5 External LED中的python文件，运行示例程序可以看到红灯有在闪烁的现象。
- 使用注意事项：LED较长的引脚为正极，较短的为负极，负极应该接GND，正极应该和GPIO输出口相连，使用时必须接上电阻。

External LED blink

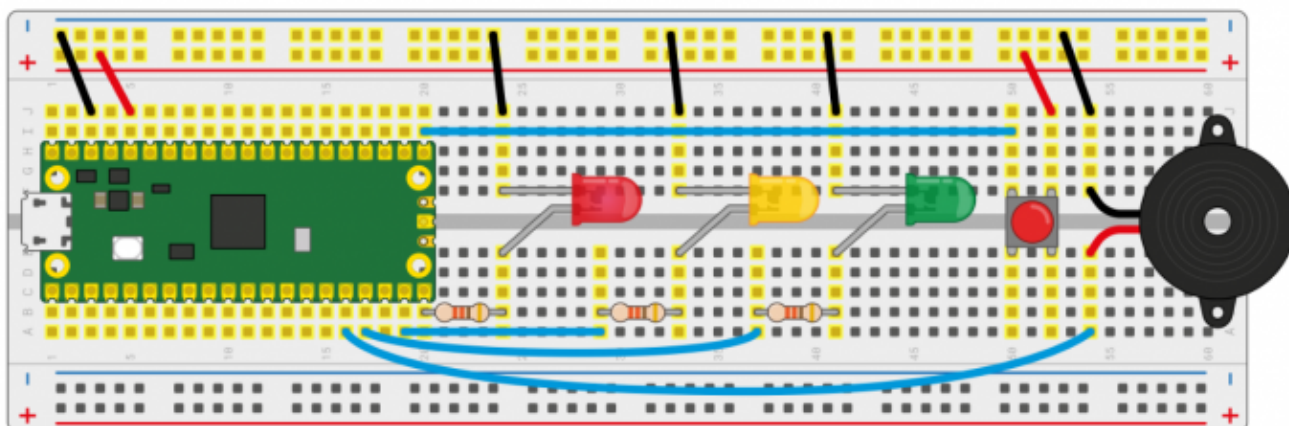


#### 代码解析

```
led_external = machine.Pin(15, machine.Pin.OUT) #设置GP15为输出模式
while True:
    led_external.toggle() #每过5秒钟让LED灯的状态改变一次
    utime.sleep(5)
```

## Traffic Light System 实验

- 按照下图连接好硬件，连接好接入电脑的Micro USB，在Thonny打开示例程序Lesson-9 Traffic-Light-System中的python文件，运行程序可以看到交通灯带正常的运行，当按下按键时会触发蜂鸣器。
- 使用注意事项：LED较长的引脚为正极，较短的为负极，负极应该接GND，正极应该和GPIO输出口相连，使用时必须接上电阻;蜂鸣器的红线接GPIO口输出，黑线接GND。



#### 代码解析

```

def button_reader_thread(): #检测按键是否被按下
    global button_pressed
    while True:
        if button.value() == 1:
            button_pressed = True

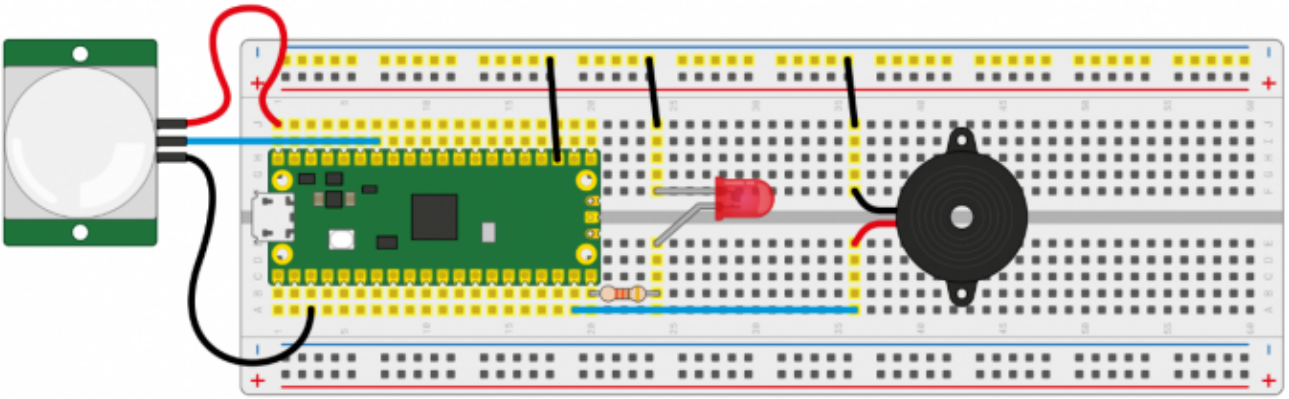
_thread.start_new_thread(button_reader_thread, ()) #用开启线程的方式去检测按键
while True:
    if button_pressed == True: #如果按键被按下, 红灯亮起, 蜂鸣器响闹
        led_red.value(1)
        for i in range(10):
            buzzer.value(1)
            utime.sleep(0.2)
            buzzer.value(0)
            utime.sleep(0.2)
        global button_pressed
        button_pressed = False
    led_red.value(1) #正常情况下红灯边绿灯时黄灯会亮两秒, 然后黄灯和红灯灭, 绿灯亮
    utime.sleep(5) #由绿灯边红灯时, 绿灯先灭, 黄色亮两秒, 然后红灯亮
    led_amber.value(1)
    utime.sleep(2)
    led_red.value(0)
    led_amber.value(0)
    led_green.value(1)
    utime.sleep(5)
    led_green.value(0)
    led_amber.value(1)
    utime.sleep(5)
    led_amber.value(0)

```

## Burglar Alarm LED Buzzer 实验

- 按照下图连接好硬件, 连接好接入电脑的Micro USB, 在Thonny打开示例程序Lesson-14 Burglar Alarm LED Buzzer中的python文件,运行程序可以看到, 当人为的在Passive infrared sensor前晃动时, LED灯闪亮的同时蜂鸣器也会报警。
- 使用注意事项: Passive infrared sensor 的中间引脚为数据输出引脚, 两边的引脚分别接入VCC和GND即可。



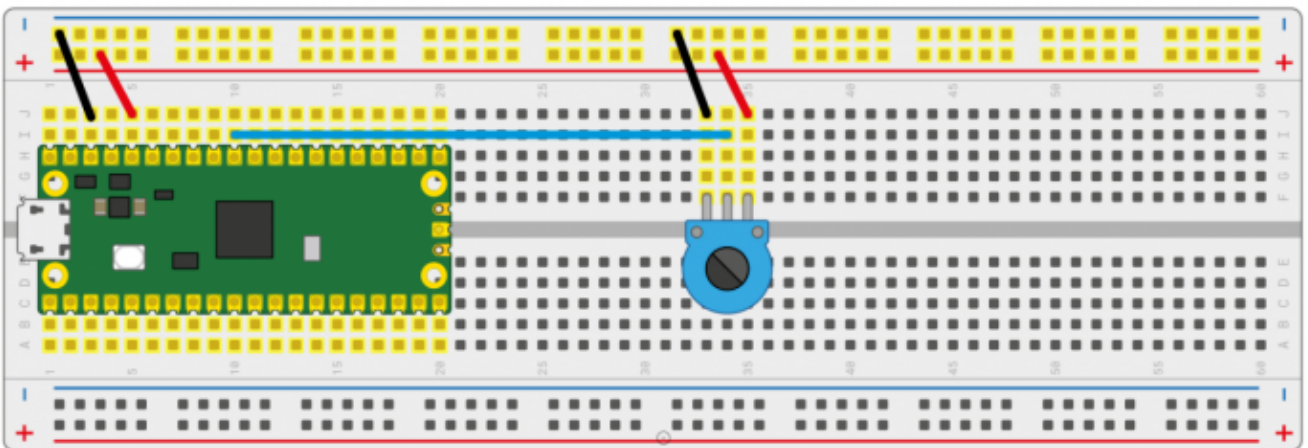


## ■ 代码解析

```
def pir_handler(pin): #中断处理函数，蜂鸣器响，led快速闪烁
    print("ALARM! Motion detected!")
    for i in range(50):
        led.toggle()
        buzzer.toggle()
        utime.sleep_ms(100)
sensor_pir.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_handler)#开启中断，当人体
传感器检测到异常时就会今天中断处理函数处理
while True: #无异常状态下会每隔5秒改变一次LDE的状态
    led.toggle()
    utime.sleep(5)
```

## Potentiometer 实验

- 按照下图连接好硬件，连接好接入电脑的Micro USB，在Thonny打开示例程序Lesson-16 Potentiometer中的python文件,运行程序，旋转电位器可以看到Sheel窗口中打印出来的电压值也在改变。
- 使用注意事项：Potentiometer的中间引脚为数据输出口，两边的引脚分别接上GND和VCC即可。

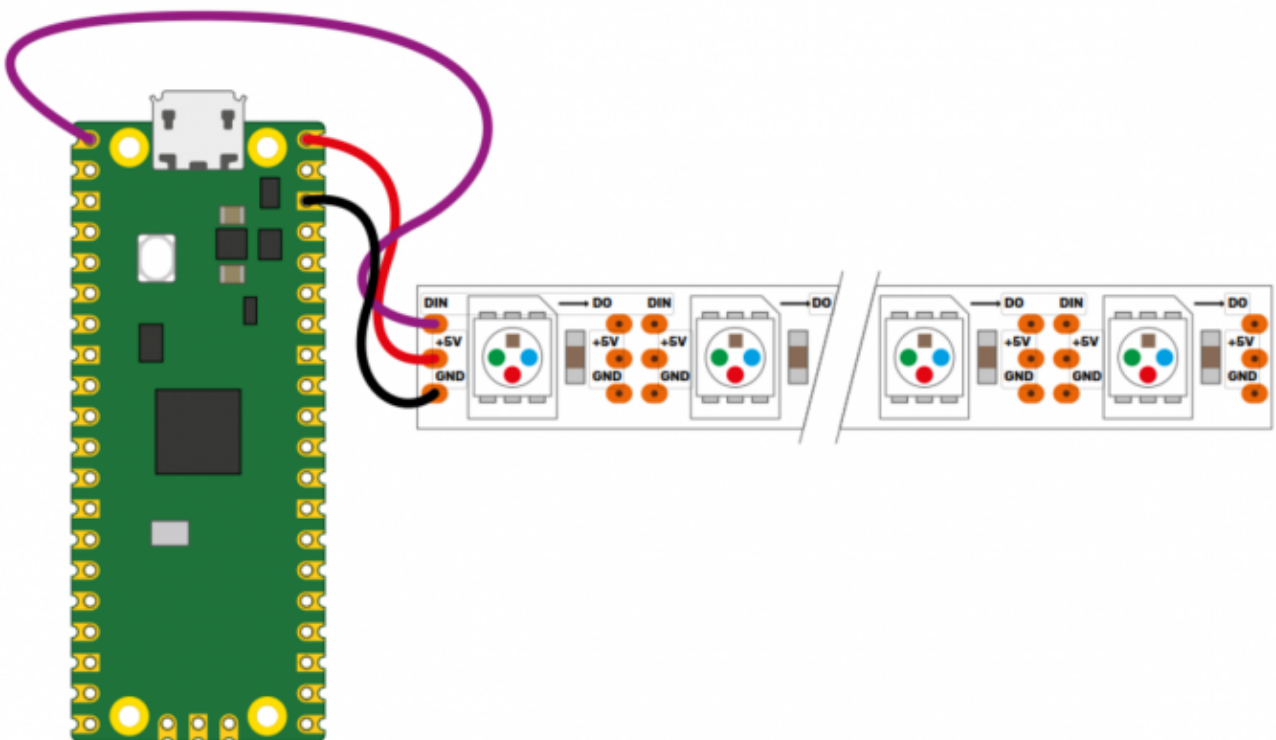


## ■ 代码解析

```
potentiometer = machine.ADC(26) #将GP26作为模拟信号采集引脚
conversion_factor = 3.3 / (65535)
while True:
    voltage = potentiometer.read_u16() * conversion_factor #将采集到的数据进行格式化转换
    换成电压值
    print(voltage) #打印电压信息，电压值会随着滑动变阻器旋转而变化
    utime.sleep(2)
```

## WS2812 实验

- 按照下图连接好硬件，连接好接入电脑的Micro USB，在Thonny打开示例程序Lesson-25 WS2812中的WS2812\_RGB\_LED.py文件,运行程序可以一次看到蓝、红、绿、白的RGB颜色。



- 代码解析

#这一段代码使用到的是状态机机制,如下代码是一个装饰器,在装饰器中我们可以硬件进行初始化、设定引脚的电平等。

#label("bitloop") 我们可以在代码中定义一下标记,方便我们通过跳转的方式跳到他们这里执行。

#jmp(not\_x,"do\_zero") 当x=0时,我们就调整到标签“do\_zero”。

#nop().set(0) [T2 - 1] 当x=0时,会跳转到这里执行。

```
@asm_pio(sideset_init=PIO.OUT_LOW, out_shiftdir=PIO.SHIFT_LEFT, autopull=True, pull_thresh=24)
```

```
def ws2812():
```

```
    T1 = 2
```

```
    T2 = 5
```

```
    T3 = 1
```

```
    label("bitloop")
```

```
    out(x, 1)                .side(0)    [T3 - 1]
```

```
    jmp(not_x, "do_zero")    .side(1)    [T1 - 1]
```

```
    jmp("bitloop")          .side(1)    [T2 - 1]
```

```
    label("do_zero")
```

```
    nop()                    .side(0)    [T2 - 1]
```

```
# Create the StateMachine with the ws2812 program, outputting on Pin(22).
```

```
sm = StateMachine(0, ws2812, freq=800000, sideset_base=Pin(0)) #创建状态机
```

```
# Start the StateMachine, it will wait for data on its FIFO.
```

```
sm.active(1) #开始状态机
```

```
# Display a pattern on the LEDs via an array of LED RGB values.
```

```
ar = array.array("I", [0 for _ in range(NUM_LEDS)])
```

```
print(ar)
```

```
print("blue")
```

```
for j in range(0, 255):
```

```
    for i in range(NUM_LEDS):
```

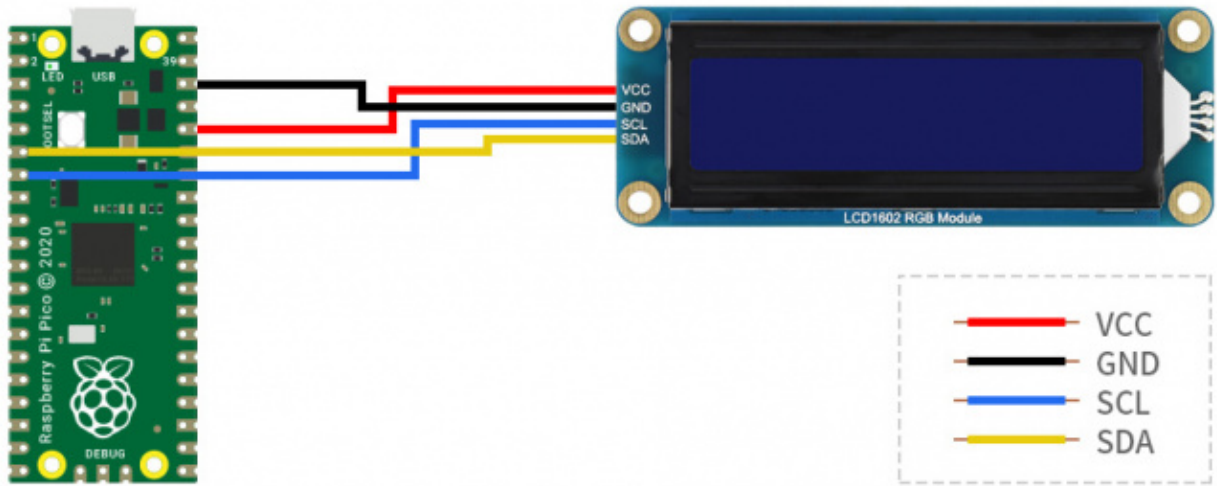
```
        ar[i] = j
```

```
    sm.put(ar,8) #put()的方法是将数据放入状态机的输出FIFO
```

```
    time.sleep_ms(5)
```

## LCD1602 I2C 实验

- 按照下图连接好硬件,连接好接入电脑的Micro USB,在Thonny打开示例程序Lesson-21 LCD1602 I2C中的python文件,先将RGB1602.py文件另存为Raspberry Pi Pico中,运行Choose\_Color.py可以看到每5秒切换一种不同的颜色;运行Discoloration.py文件可以看到RGB颜色渐变的效果。



## ■ 代码解析

### Choose\_Color.py

```
#定义颜色
rgb9 = (0,255,0) #青色'
lcd.setCursor(0, 0) #设置光标位置
# print the number of seconds since reset:
lcd.printout("Waveshare") #写入字符
lcd.setCursor(0, 1) #设置光标位置到第二行第零列
lcd.printout("Hello,World!")#写入字符
lcd.setRGB(rgb1[0],rgb1[1],rgb1[2]); #设置背光
```

### Discoloration.py

```
t=0
while True:

    r = int((abs(math.sin(3.14*t/180)))*255); #RGB随着时间的变化而变化
    g = int((abs(math.sin(3.14*(t+60)/180)))*255);
    b = int((abs(math.sin(3.14*(t+120)/180)))*255);
    t = t + 3;
    lcd.setRGB(r,g,b);#重新设置RGB的值
# set the cursor to column 0, line 1
    lcd.setCursor(0, 0) #定位到第一行第零列
# print the number of seconds since reset:
    lcd.printout("Waveshare")#写入字符
    lcd.setCursor(0, 1) #定位到第二行第零列
    lcd.printout("Hello,World!")#写入字符
    time.sleep(0.3)
```

## 配套资料

### 文档

---

RP2040-Zero原理图

RP2040-Zero-STEP文件 (3D模型)

### 程序

---

WS2812B测试程序

## 应用程序

- JustUSB项目 (微雪用户分享)

## 官方资料

### 树莓派官方文档

---

- Raspberry Pi Pico入门学习MicroPython编程书籍 (英文版)
- 树莓派相关书籍下载
- Raspberry Pi Pico原理图
- Pico引脚分布图
- Pico入门使用手册
- Pico C SDK使用手册
- Pico Python SDK使用手册
- Pico数据手册
- RP2040数据手册
- RP2040硬件设计参考手册

### 树莓派开源例程

---

- 树莓派官方C/C++示例程序 (github)
- 树莓派官方micropython示例程序 (github)

## 开发软件

- Thonny Python IDE (Windows版本 V3.3.3)
- Pico环境搭建相关软件 (百度网盘提取码: prgc )

## Pico系列教程

树莓派Pico 视频教程 [展开](#)

树莓派Pico C/C++ SDK 入门教程 [展开](#)

## Pico开源例程

树莓派官方Pico资料链接

MircoPython视频例程(github)

MicroPython固件/Blink例程 (C)

树莓派官方C/C++示例程序 (github)

树莓派官方micropython示例程序 (github)