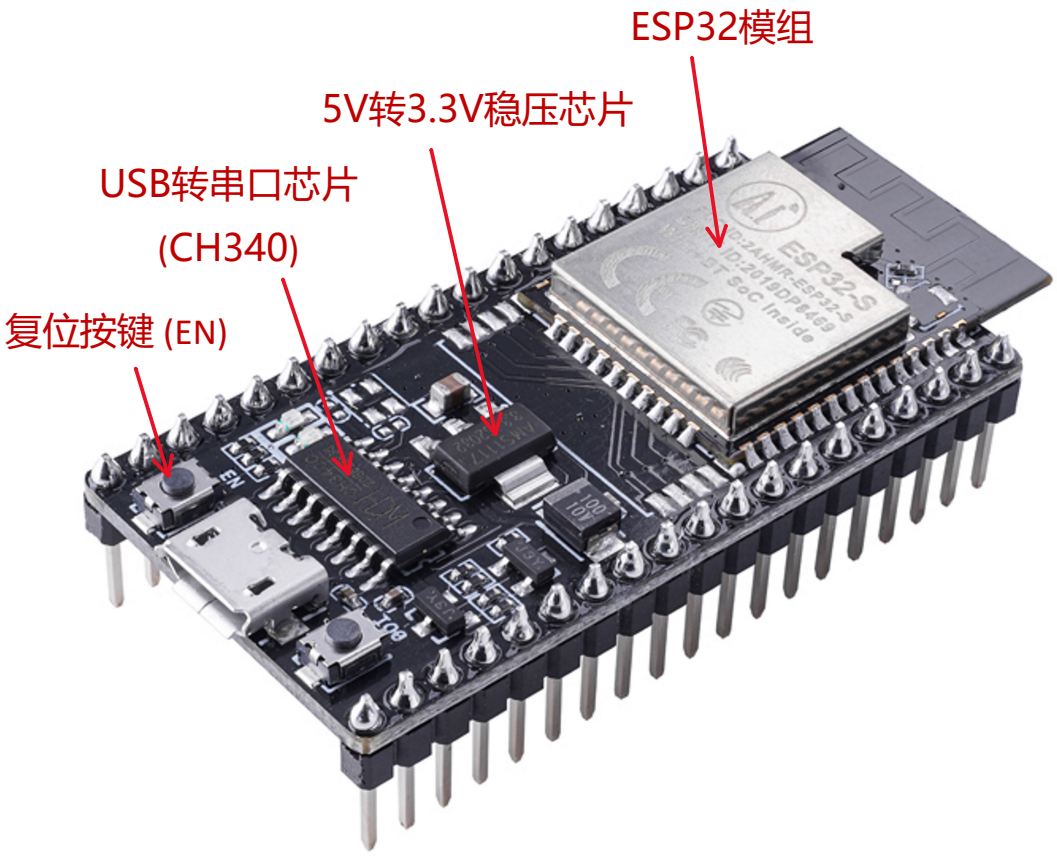
01 - Intro

2022年1月27日 10:41

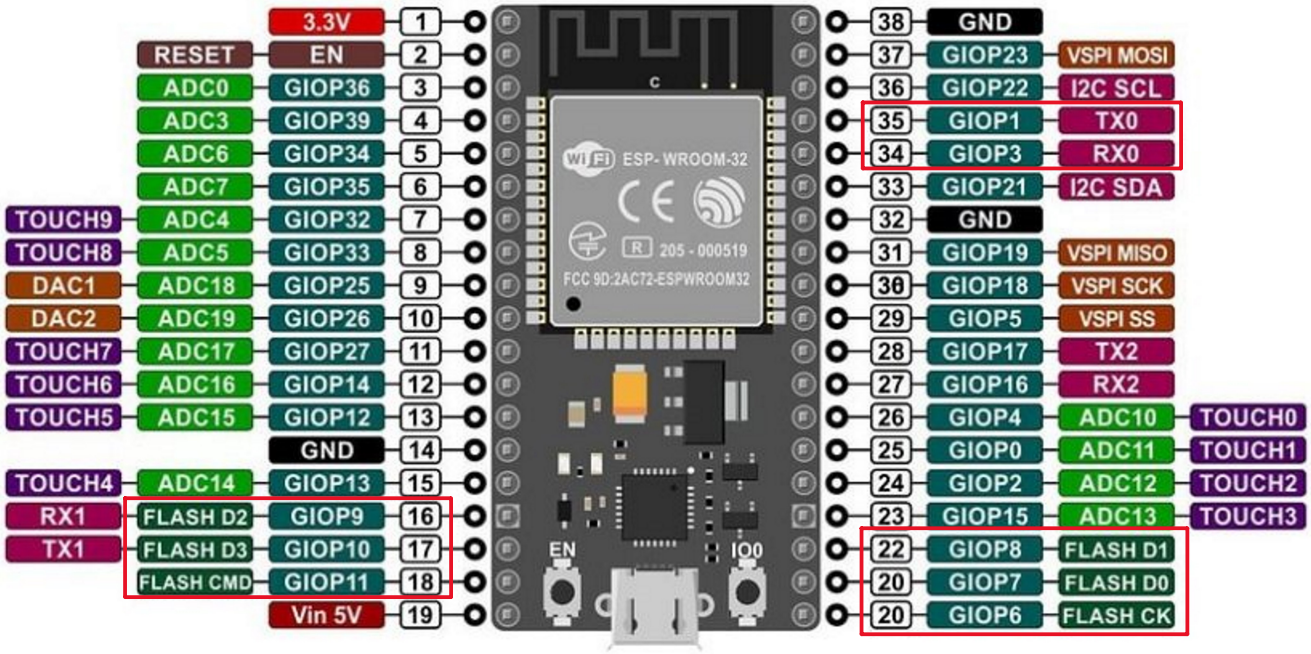
MicroPython与嵌入式系统基础

1. 开发环境搭建，Python入门
2. GPIO：LED，按键，红外传感器，超声波传感器，步进电机
3. PWM：直流电机/舵机
4. ADC：红外传感器模拟输出，电位器
5. UART：UART发送接收数据，ESP32-PC, ESP32-ESP32
6. I^2C：传感器（TMP102，MPU6050）
7. wifi：网页控制GPIO

ESP32开发板 (NodeMCU-32S)



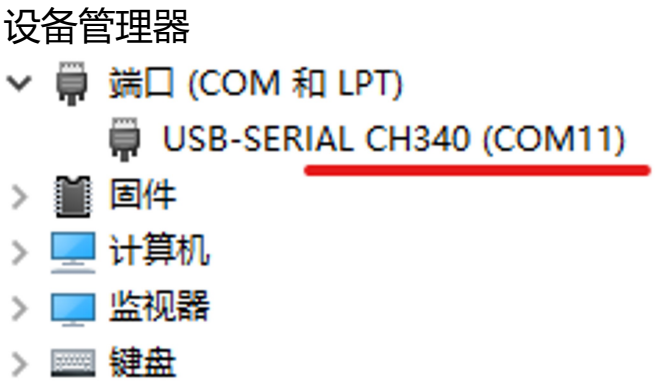
引脚功能



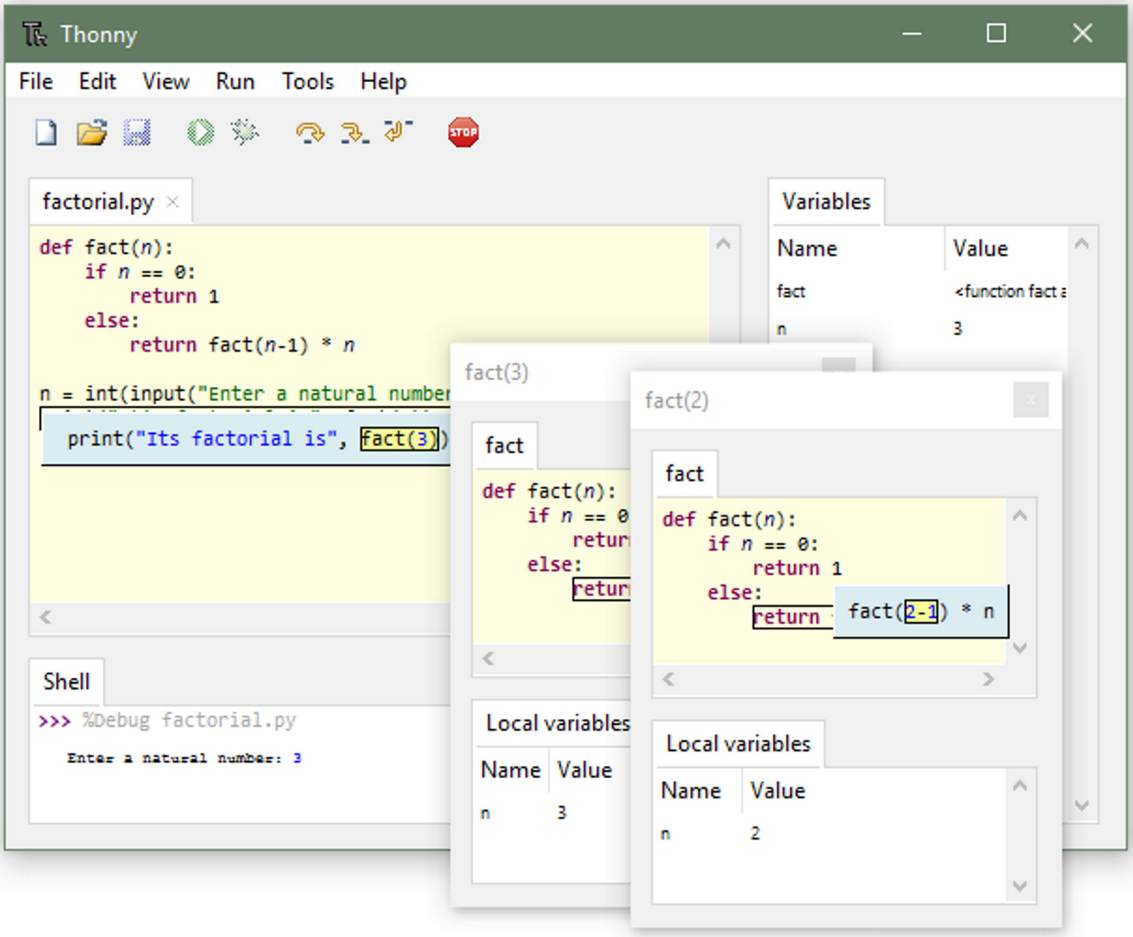
GPIO1和GPIO3为UART0，MicroPython与电脑通信专用。

GPIO6 至 GPIO11 用于连接模组上集成的 SPI flash，不建议用于其他功能。

安装USB转串口驱动（CH340）



Thonny IDE



https://thonny.org/

下载MicroPython

https://micropython.org/download/esp32/

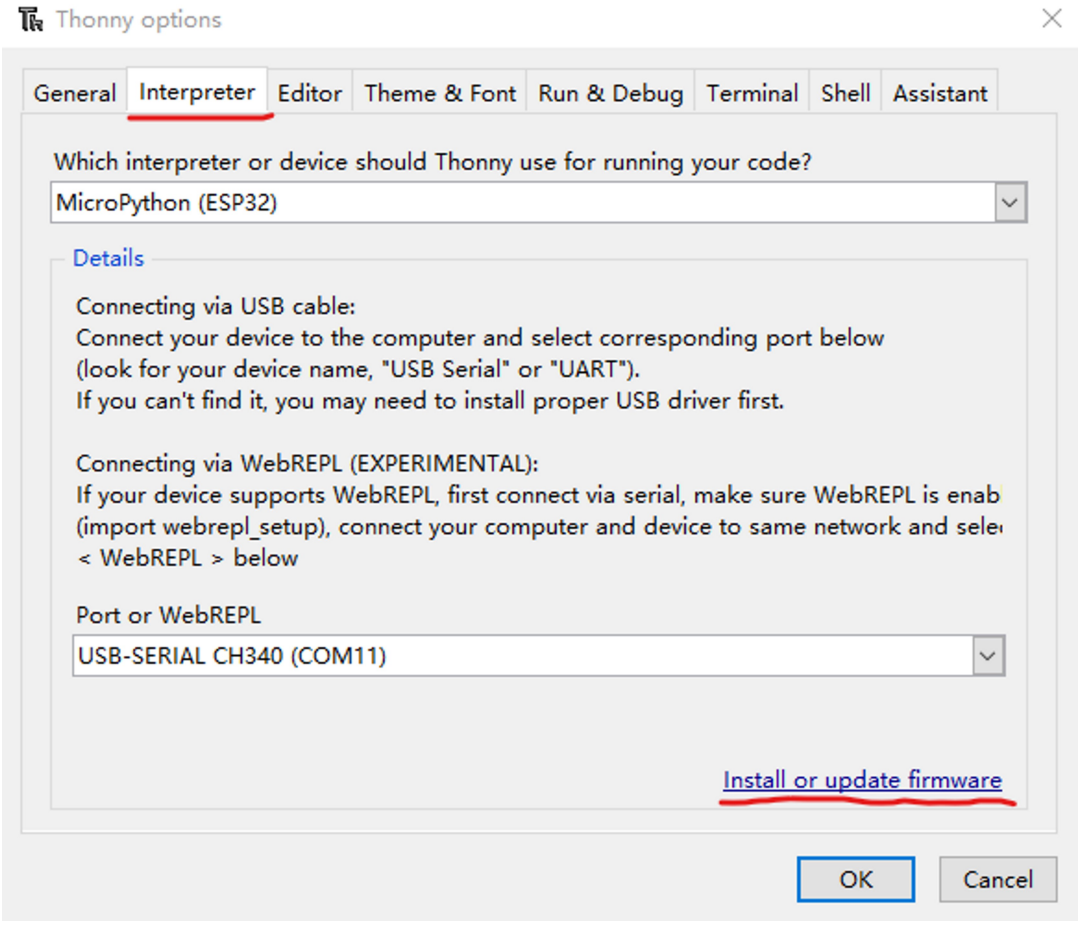
Firmware

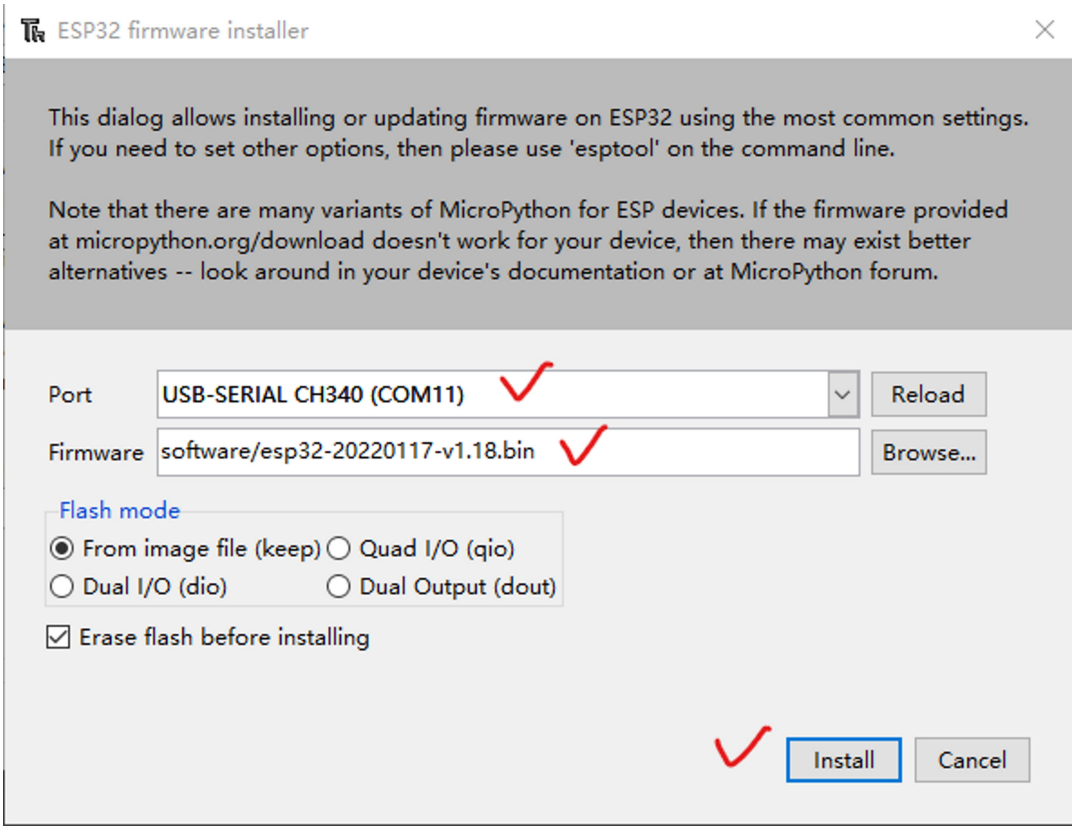
Releases

v1.18 (2022-01-17) .bin [.elf] [.map] [Release notes] (latest)
v1.17 (2021-09-02) .bin [.elf] [.map] [Release notes]
v1.16 (2021-06-23) .bin [.elf] [.map] [Release notes]
v1.15 (2021-04-18) .bin [.elf] [.map] [Release notes]
v1.14 (2021-02-02) .bin [.elf] [.map] [Release notes]
v1.13 (2020-09-02) .bin [.elf] [.map] [Release notes]
v1.12 (2019-12-20) .bin [.elf] [.map] [Release notes]

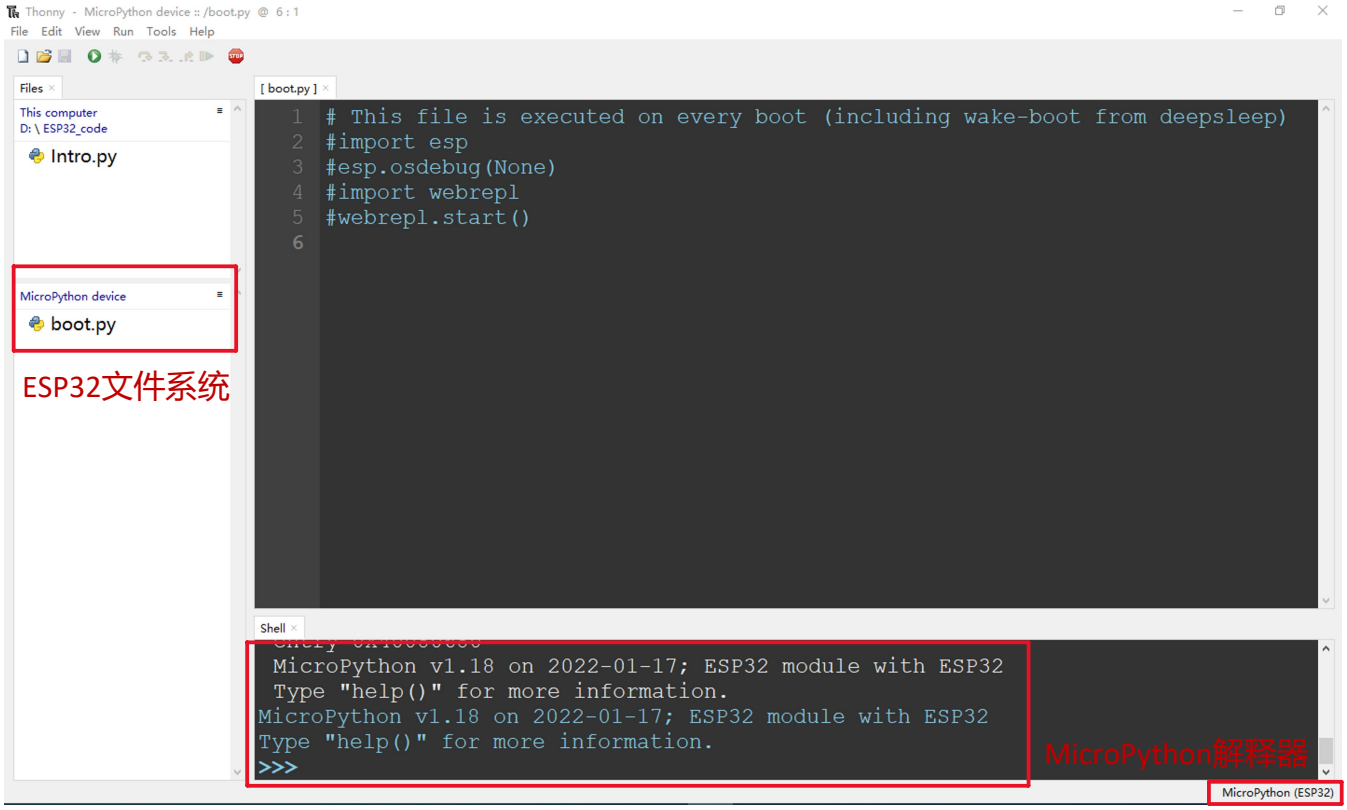
安装MicroPython

Thonny IDE -> Tools -> Options





安装完毕



ESP32重启后，自动执行boot.py和main.py

MicroPython官方文档，源码

https://docs.micropython.org/en/latest/

https://docs.micropython.org/en/latest/esp32/quickref.html#

https://github.com/micropython/micropython/

MicroPython/ESP32教程

https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/

Python参考书 - Think Python

英文版

http://greenteapress.com/thinkpython2/thinkpython2.pdf
https://greenteapress.com/thinkpython2/html/index.html

中文版

https://tairraos.github.io/ThinkPython/index.html
https://pan.baidu.com/s/1pLiwSAn%E3%80%82

Python入门教程

intro.py

```
"""
Python Tutorial

- Hello world
- Python as a caculator
- variables
- strings
- arithmetic operators
- comparison operators
- if statements
- logical operators
- lists
- tuples
- dictionaries
- loops
- functions
- classes
- modules
"""

# Ctrl-l    Clear command line
# Ctrl-c    Interrupt current Python command
# Ctrl-d    Exit Python session
# up and down arrow to cycle through past commands

# ---------------- Hello World ----------------
print("Hello World!")
name = input("What's your name? ")
print("Hello ", name)

# ---------------- Python as caculator ----------------
# Type on command line
1+2
0.3*5
4**2

# ---------------- variables ----------------
distance = 10         # integer, type()
speed = 4.9           # float
message = 'Hello'     # string
is_cool = True        # boolean  True/False

# ---------------- strings ----------------
# We can define strings using single (' ') or double (" ") quotes.
# To define a multi-line string, use triple quotes (""").
empty_string = ''      # empty string
course = 'Python Tutorial'
print(course[0])       # first character
print(course[1])       # second character
print(course[-1])      # first character from the end
```

```
print(course[-2])      # second character from the end
print(course[1:5])     # string slicing (5th not included)

# string concatenation
message = 'Hello ' + 'World'
print(message)

# formatted strings
name = 'Python'
message = f'Hello {name}!'
print(message)

# string methods/functions
message.upper()               # to convert to uppercase
message.lower()               # to convert to lowercase
message.find('P')             # index of first occurrence of 'P'
message.replace('P', 'Q')
is_contained = 'Python' in message     # in operator

# ---------------- arithmetic operators ----------------
# +, -, *, /, //, %, **
print(5/2)         # / returns a float
print(5//2)        # // returns an int
print(5%2)         # % returns the remainder
print(5**2)        # x**y returns x to the power of y

# ---------------- comparison operators ----------------
a > b
a >= b (greater than or equal to)
a < b
a <= b
a == b (equal)
a != b (not equal)

# ---------------- if statements ----------------
age = 15

if age < 18:                # colon
    print("Too young!")     # indentation
    print("No pass!")
elif age > 30:
    print("Too old!")
    print("No pass!")
else:
    print("Right age!")
    print("Pass!")

# ---------------- logical operators ----------------
# and, or, not
is_hot = False
is_cold = False
if not is_hot and not is_cold:
    print("beautiful day")
```

```
if is_hot or is_cold:
    print("not a good day")

# ---------------- list ----------------
empty_list = []
numbers = [1, 2, 3, 4, 5]
numbers[0]         # first item
numbers[1]         # second item
numbers[-1]        # first item from the end
numbers[-2]        # second item from the end
numbers[:]         # returns a copy of the whole list
numbers[::2]       # returns [1, 3, 5]
numbers[::-1]      # reverse the list

# list methods/functions
numbers.append(6)         # adds 6 to the end
numbers.insert(0, 6)      # adds 6 at index position of 0
numbers.remove(6)         # removes 6
numbers.pop()             # removes the last item
numbers.clear()           # removes all the items
numbers.index(8)          # index of first occurrence of 8
numbers.sort()            # sorts the list
numbers.reverse()         # reverses the list
numbers.copy()            # returns a copy of the list

# ---------------- tuple ----------------
# tuples are like lists, but immutable (read-only).
coordinates = (1, 2, 3)
x, y, z = coordinates      # unpack a list or a tuple
coordinates[1] = 0         # error

# ---------------- dictionary ----------------
# key/value pairs
student = {
    "name": "John Smith",      # key = "name", value = "John Smith"
    "age": 20,
    "is_registered": True
}

student["name"]            # "John Smith"
student.get("name")        # "John Smith"
student["name"] = "Steve Jobs"

# ---------------- loop ----------------
# while loop
i = 1
while i < 5:
    print(i)
    i += 1

# for loop
range(5)         # 0, 1, 2, 3, 4
list(range(5))   # [0, 1, 2, 3, 4]
range(1, 5)      # 1, 2, 3, 4
```

```
range(1, 5, 2)   # 1, 3
for i in range(1, 5):
    print(i)

# in operator
for c in "hello world!":
    print(c)

numbers = [1, 2, 3, 4, 5]    # list
for n in numbers:
    print(n)

coordinates = (1, 2, 3)      # tuple
for n in coordinates:
    print(n)

student = {                  # dictionary
"name": "John Smith",
"age": 20,
"is_registered": True
}
for key in student:
    print(key, student[key])


# ---------------- function ----------------
def greet_user(name):
    print(f"Hi {name}")      # return None

greet_user("John")

def square(number):
    return number * number   # return value

result = square(2)
print(result)                # prints 4

# ---------------- class ----------------
# Classes are user-defined data types.
# An object consists of data and operations.
# Functions of a class are called methods.
# A class is a blueprint, an object is an instance of a class.

class Point:
 def __init__(self, x, y):  # a special method called constructor.
     self.x = x
     self.y = y
 def get_coord(self):
     return self.x, self.y
 def set_coord(self, x, y):
     self.x = x
     self.y = y
 def distance(self, point):
     return ((self.x - point.x)**2 + (self.y - point.y)**2)**0.5
```

```
p1 = Point(0, 0)  # An object is an instance of a class.
p2 = Point(1, 1)
print(p1.get_coord())
p2.set_coord(3, 4)
print(p2.get_coord())
print("Distance between p1 and p2 is: ", p1.distance(p2))

# Class inheritance
class Person:
    def __init__(self, name):
        self.name = name
    def set_name(self, name):
        self.name = name
    def get_name(self):
        return self.name

class Student(Person):          # Student inherits from Person
    def __init__(self, name, major):
        Person.__init__(self, name)
        self.major = major
        self.grades = {}        # empty dict

    def get_major(self):
        return self.major

    def add_grades(self, course, grade):
        self.grades[course] = grade

    def get_grades(self):
          return self.grades

s1 = Student("Xiao Wang", "Mechanical Engineering")
s1.add_grades("Programming", 100)
s1.add_grades("Robotics", 95)

name = s1.get_name()          # method inherited from Person class
major = s1.get_major()
grades = s1.get_grades()

print(" Student information ")
print(f" Name: {name}\n Major: {major}\n Grades: {grades} ")

# ---------------- modules ----------------
# A module is a file with some Python code.
# import the entire time module
import time         # dir() to list all functions in a module
print("Start sleeping for 5 seconds")
time.sleep(5)
print("Sleep ended")

# import one function from the time module
from time import sleep
print("Start sleeping for 5 seconds")
```

```
sleep(5)
print("Sleep ended")
```

Python练习题

1. Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). Put the result in a list. (Hint: %, list.append())
   https://github.com/zhiwehu/Python-programming-exercises

2. 生成斐波纳契数列的前n个数，结果为一个列表（list)。
   斐波纳契数列：第一个数等于1, 第二个数等于1，从第三个数开始，任意一个数等于前两个数之和。

3. 计程车计价
   按区间计价，总价等于各个区间费用之和：
   - 前5公里，一口价20元
   - 大于5公里，小于50公里，3元/公里
   - 大于等于50公里，2元/公里；逢10公里倍数，免费，如第50公里，第60公里等免费
   - 大于等于100公里，1元/公里；逢5公里倍数，免费，如第100公里，第105公里等免费

   完成计价函数

   ```
   def taxi_meter(distance):
       # your code here

       return total
   ```

   验证

   ```
   distance = eval(input("distance = "))
   fare = taxi_meter(distance)
   print(f"fare = {fare}")
   ```

   输入：
   distance = 108
   输出：
   fare = 249

4. 写一个Car类
   要求：

1. 属性：brand, mileage
2. 方法：get_brand(), update_mileage(km), get_mileage()

验证：

```
my_car = Car('BYD', 100)
car_brand = my_car.get_brand()
print(car_brand)     # BYD
distace_traveled = my_car.get_mileage()
print(distace_traveled)      # 100
my_car.update_mileage(200)
print(my_car.get_mileage())      # 300
```

02 - GPIO

2022年1月27日 10:45

- Blink LED
- 按键开关灯
- 红外循迹/避障传感器开关灯
- 超声波传感器
- 步进电机驱动

machine模块

machine — functions related to the hardware
https://docs.micropython.org/en/latest/library/machine.html

Classes

- class Pin – control I/O pins
- class Signal – control and sense external I/O devices
- class ADC – analog to digital conversion
- class ADCBlock – control ADC peripherals
- class PWM – pulse width modulation
- class UART – duplex serial communication bus
- class SPI – a Serial Peripheral Interface bus protocol (controller side)
- class I2C – a two-wire serial protocol
- class I2S – Inter-IC Sound bus protocol
- class RTC – real time clock
- class Timer – control hardware timers
- class WDT – watchdog timer
- class SD – secure digital memory card (cc3200 port only)
- class SDCard – secure digital memory card

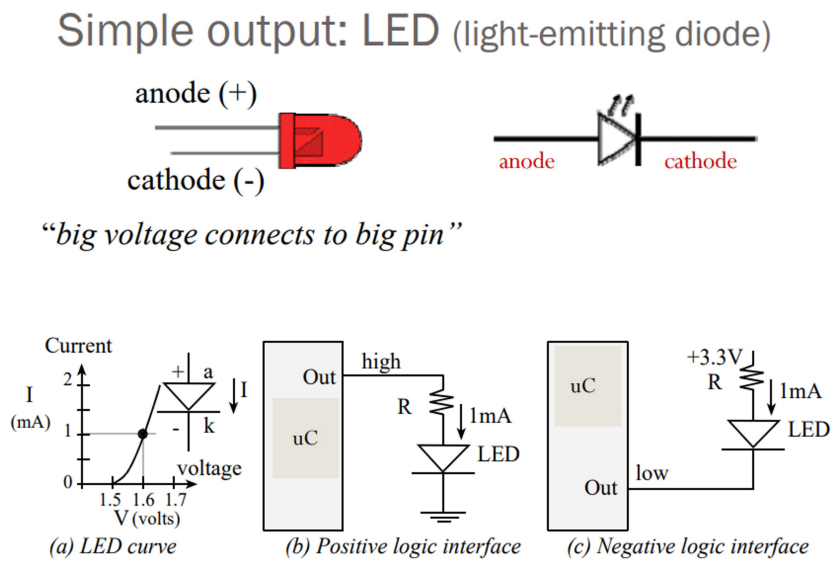
Pin类 - 控制I/O引脚

https://docs.micropython.org/en/latest/library/machine.Pin.html

```
from machine import Pin

# create an output pin on pin #0
p0 = Pin(0, Pin.OUT)

# set the value low then high
p0.value(0)
p0.value(1)

# input pin on pin #2, with a pull up resistor
p2 = Pin(2, Pin.IN, Pin.PULL_UP)

# read and print the pin value
print(p2.value())
```

```
# configure an irq callback
p0.irq(lambda p:print(p))
```

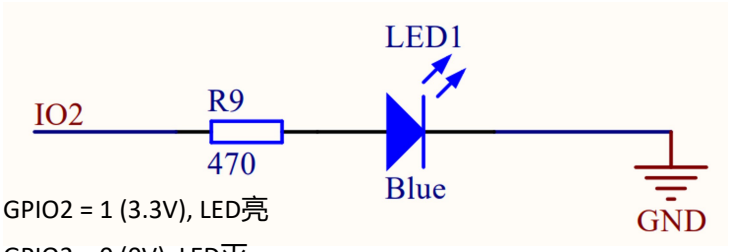
GPIO输出 - LED



https://www.eng.auburn.edu/~nelsovp/courses/elec3040_3050/

ESP32开发板 (NodeMCU-32S) 上的LED

NodeMCU_32S_V1.1.pdf



GPIO2 = 1 (3.3V), LED亮

GPIO2 = 0 (0V), LED灭

实验一：LED

闪灯1

blink1.py

```
"""
Blink LED - sleep()
Function sleep() is blocking.
On-board LED is connected to GPIO2.
"""

import time
from machine import Pin

led = Pin(2, Pin.OUT)  # Pin 2 as output
```

```
while True:
  led.value(1)      # LED on
  time.sleep(0.5)   # delay for 0.5 second
  led.value(0)      # LED off
  time.sleep(0.5)   # delay for 0.5 second
```

闪灯2

blink2.py

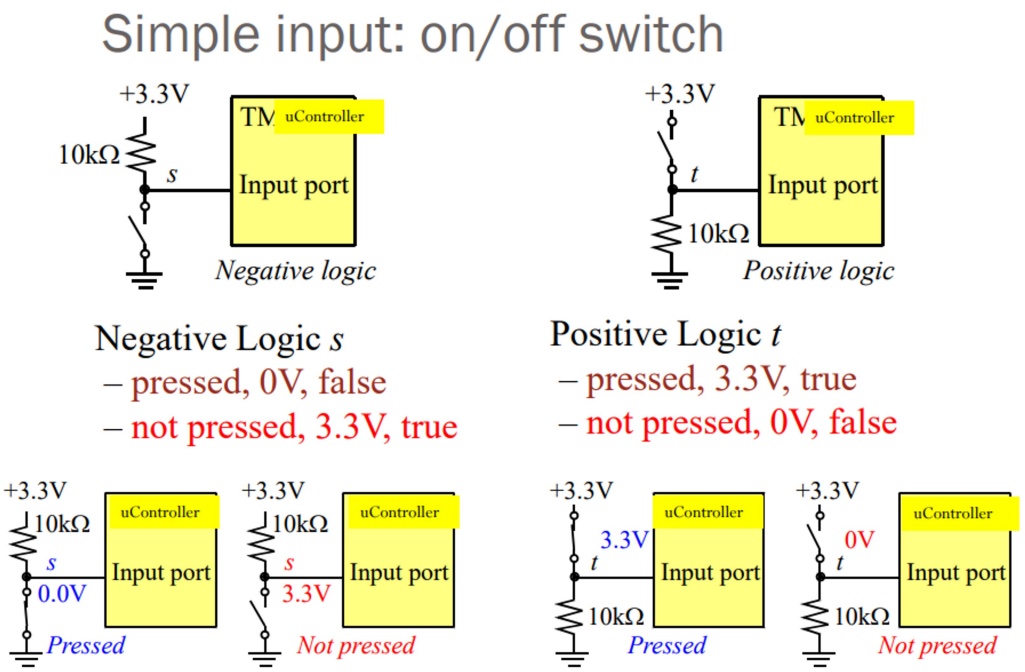
```
"""
Blink LED - ticks_ms(), ticks_diff()
On-board LED is connected to GPIO2.
"""

import time
from machine import Pin

led = Pin(2, Pin.OUT)  # Pin 2 as output
led.value(1)

old_time = time.ticks_ms() # Old time (ms)

while True:
    new_time = time.ticks_ms()  # New time (ms)
    duration = time.ticks_diff(new_time, old_time)  # duration (ms)
    if (duration >= 1000):
        if (led.value() == 1):
            led.value(0)
        else:
            led.value(1)
        old_time = new_time  # update old time
```

闪灯3

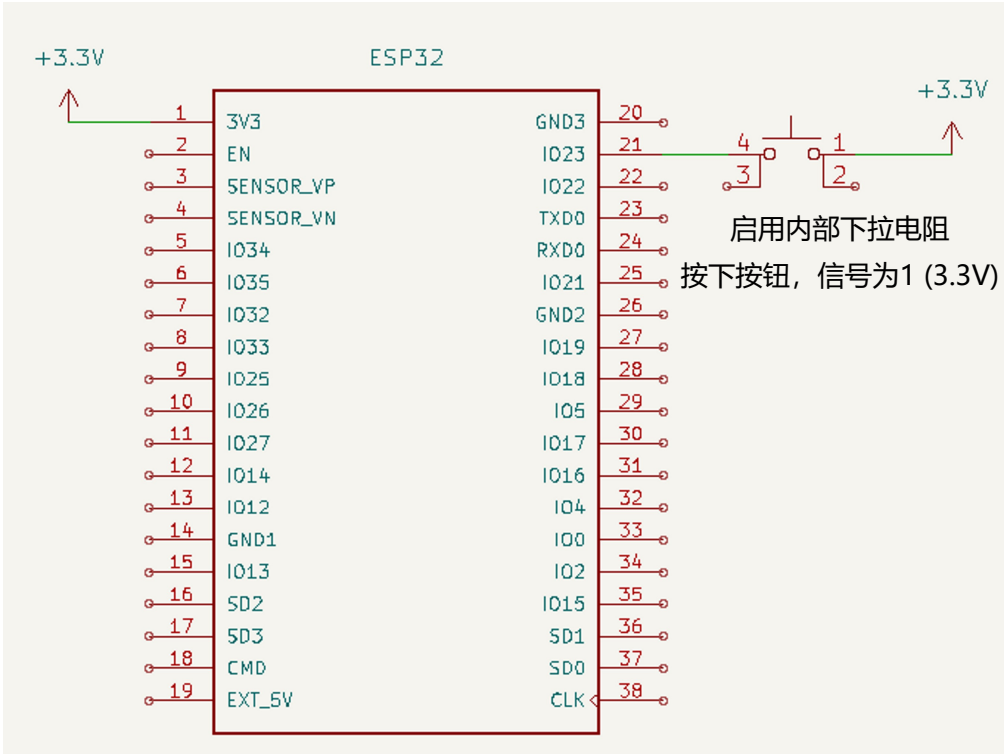
blink3.py

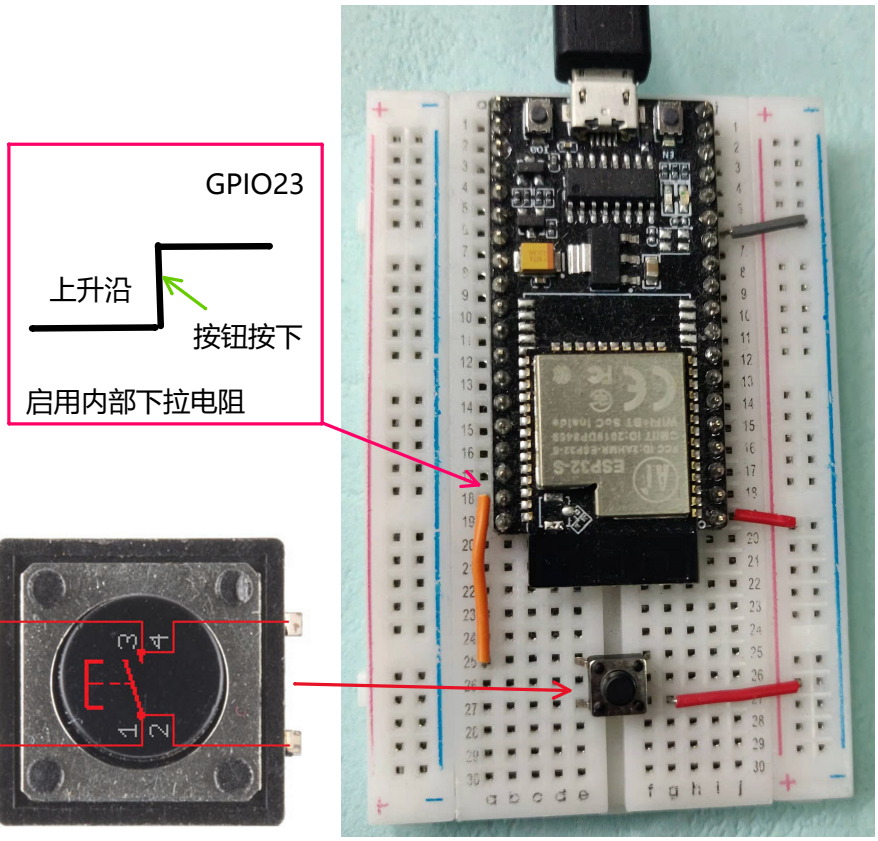
```
"""
Blink LED - timer callback
On-board LED is connected to GPIO2.
"""

from machine import Pin, Timer

led = Pin(2, Pin.OUT)
my_timer = Timer(0)  # using timer 0

# define callback function
def toggle_led(self):
    led.value(not led.value()) # reverse led pin state

my_timer.init(period = 1000, callback = toggle_led)  # 1000ms

while True:
    pass    # do nothing
```

GPIO输入 - 按钮

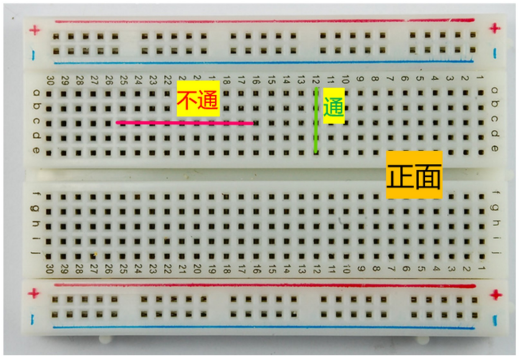


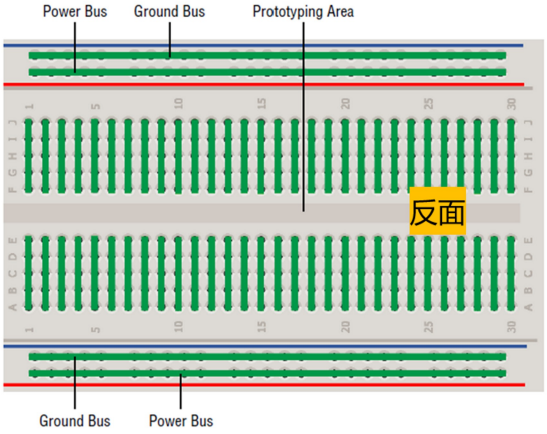
https://www.eng.auburn.edu/~nelsovp/courses/elec3040_3050/




GPIO23
上升沿
按钮按下
启用内部下拉电阻

面包板


不通
通
正面


Power Bus
Ground Bus
Prototyping Area
反面
Ground Bus
Power Bus

实验二：按钮

按钮开关灯 - 轮询模式

buttton_led.py

```
"""
Use button to toggle LED - polling
"""

from machine import Pin

led = Pin(2, Pin.OUT)
# Pin 23 as input, activate pull-down resistor
button = Pin(23, Pin.IN, Pin.PULL_DOWN)

led_state = 0
old_button_state = 0
new_button_state = 0

while True:
    new_button_state = button.value()
    # Button is pressed: old state is 0, new state is 1.
    if old_button_state == 0 and new_button_state == 1:
        led_state = led.value()     # read led pin state
        led_state = not led_state   # reverse led pin state
        led.value(led_state)        # write new led pin state
    old_button_state = new_button_state  # update button pin state
```

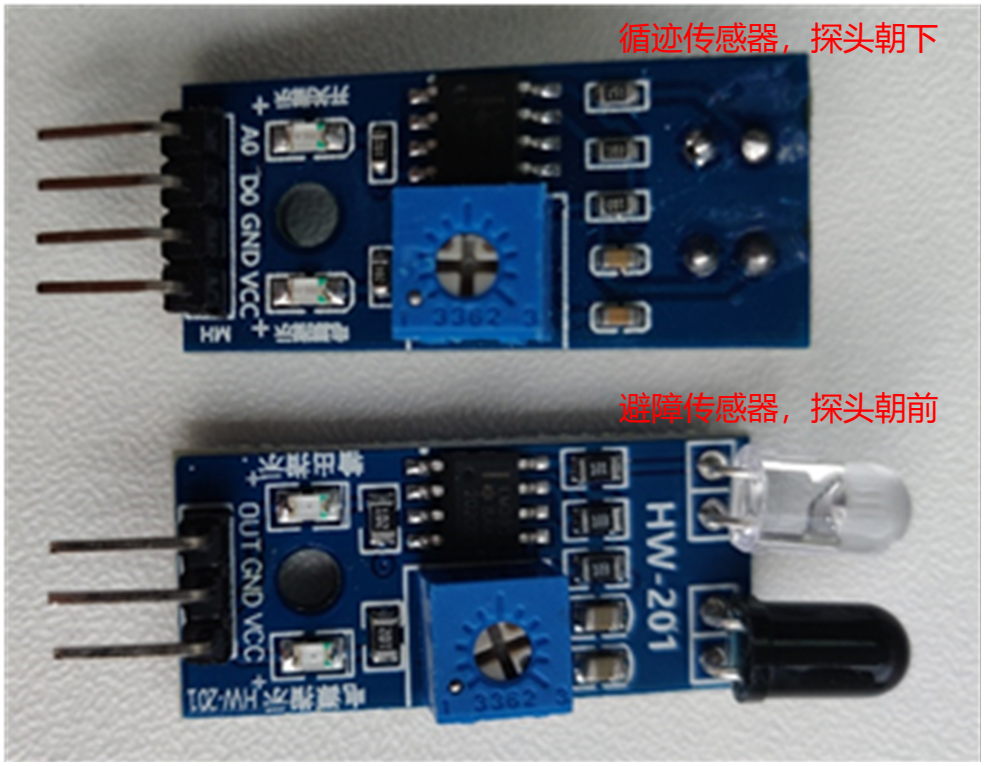
按钮开关灯 - 中断模式

buttton_led_irq.py

```
"""
Use button to toggle LED - interrupt
"""

from machine import Pin

led = Pin(2, Pin.OUT)
# Pin 23 as input, activate pull-down resistor
button = Pin(23, Pin.IN, Pin.PULL_DOWN)

# define interrupt callback function
def toggle_led(p):
    led_state = led.value()     # read led pin state
    led_state = not led_state   # reverse led pin state
    led.value(led_state)        # write led pin state

# trigger interrupt on rising edge - from 0 to 1
button.irq(handler = toggle_led, trigger = Pin.IRQ_RISING)

while True:
    pass   # do nothing
```

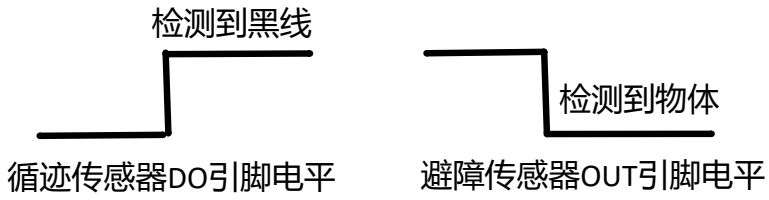
练习一： 红外传感器控灯

当循迹传感器检测到黑线或避障传感器检测到物体时，反转LED的状态。



循迹传感器： output = 1 (检测到黑线， 传感器开关指示LED灭)

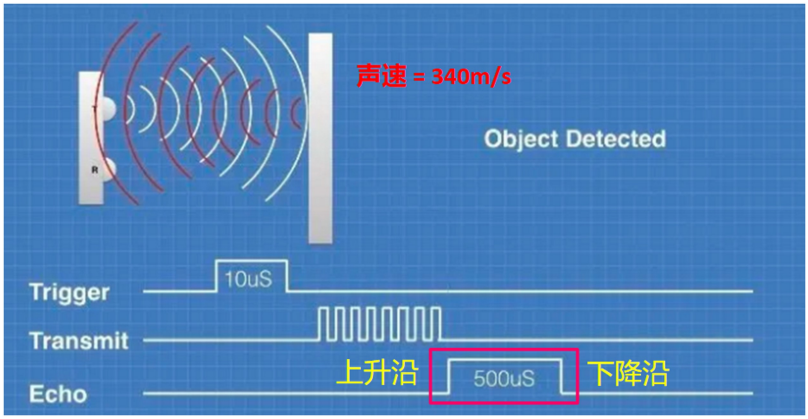
避障传感器： output = 0 (检测到物体， 传感器输出指示LED亮)



接线

传感器		开发板
VCC	-	3.3V
GND	-	GND
OUT	-	GPIO
DO	-	GPIO

超声波传感器

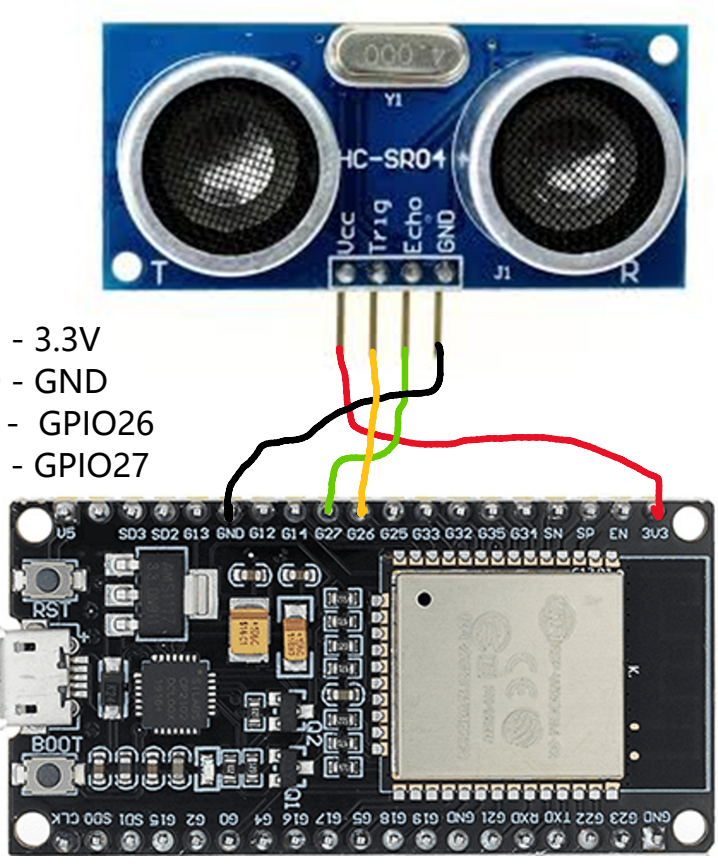


VCC - 3.3V
GND - GND
Trig - 触发（10μs高电平）
Echo - 回波

距离 = 回波引脚高电平时间 × 声速 / 2
例：回波引脚高定平时间为500μs
距离 $= 500 \times 10^{-6} \times 340 \div 2 = 0.085m = 8.5cm$

程序流程：

- 将Trig引脚置为高电平 - trig_pin.value(1)
- 延时10微秒 - time.sleep_us(10)
- 将Trig引脚置为低电平 - trig_pin.value(0)
- 捕获Echo引脚高电平时长 - machine.time_pulse_us(echo_pin, 1, timeout)
- 计算距离



VCC - 3.3V
GND - GND
Trig - GPIO26
Echo - GPIO27

HC-SR04库

https://github.com/rsc1975/micropython-hcsr04/blob/master/hcsr04.py

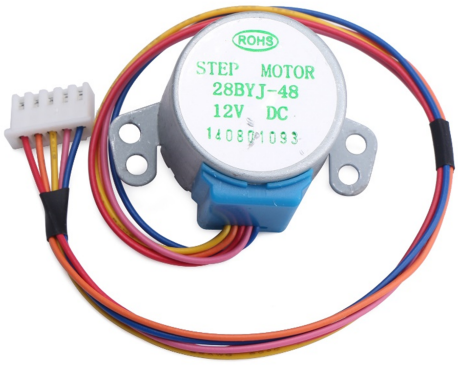
```
from hcsr04 import HCSR04
from time import sleep

sensor = HCSR04(trigger_pin = 26, echo_pin = 27)

while True:
    distance = sensor.distance_cm()
    print('Distance:', distance, 'cm')
    sleep(1)
```

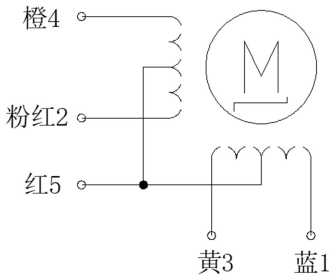
练习二：LED状态显示障碍物距离

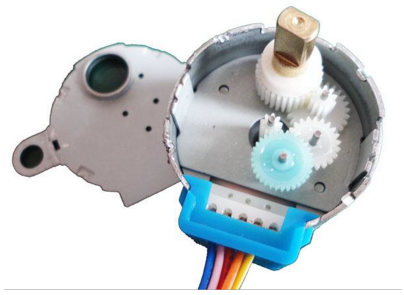
- 超声波传感器检测障碍物
- 小于5cm内或大于50cm，LED灭
- 5-50cm范围内，LED闪灯，频率与距离成反比，sleep_ms(50) - sleep_ms(500)

步进电机

28BYJ-48步进电机







驱动电源：12V

步距角：5.625°

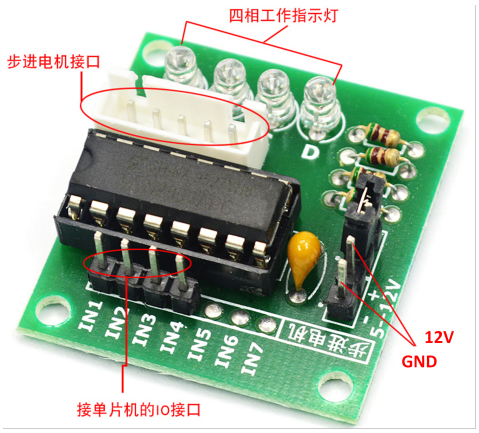
减速比：64

驱动方式：四相八拍

输出轴旋转一圈需要的步数：

(360/5.625) x 64 = 4096

ULN2003步进电机驱动



驱动相序 - 四相八拍

顺时针				逆时针			
IN1	IN2	IN3	IN4	IN1	IN2	IN3	IN4
1	0	0	0	1	0	0	1
1	1	0	0	0	0	0	1
0	1	0	0	0	0	1	1
0	1	1	0	0	0	1	0
0	0	1	0	0	1	1	0
0	0	1	1	0	1	0	0
0	0	0	1	1	1	0	0
1	0	0	1	1	0	0	0

实验三：步进电机

步进电机正反转

stepper.py

```
"""
Drive stepper motor 28BYJ-48 using ULN2003
"""

from machine import Pin
from time import sleep_ms

# define pins for ULN2003
IN1 = Pin(16, Pin.OUT)
IN2 = Pin(17, Pin.OUT)
IN3 = Pin(5,  Pin.OUT)
IN4 = Pin(18, Pin.OUT)

# half-step mode
# clockwise step sequence
seq_cw = [[1, 0, 0, 0],
          [1, 1, 0, 0],
          [0, 1, 0, 0],
          [0, 1, 1, 0],
          [0, 0, 1, 0],
          [0, 0, 1, 1],
          [0, 0, 0, 1],
          [1, 0, 0, 1]]

# counterclockwise step sequence
seq_ccw = seq_cw[::-1]

delay = 1  # ms, delay between steps

# one clockwise revolution (4096 steps)
for i in range(4096):
```

```
        step = i % 8
        IN1.value(seq_cw[step][0])
        IN2.value(seq_cw[step][1])
        IN3.value(seq_cw[step][2])
        IN4.value(seq_cw[step][3])
        sleep_ms(1)

    # one counterclockwise revolution (4096 steps)
    for i in range(4096):
        step = i % 8
        IN1.value(seq_ccw[step][0])
        IN2.value(seq_ccw[step][1])
        IN3.value(seq_ccw[step][2])
        IN4.value(seq_ccw[step][3])
        sleep_ms(1)
```

练习三： 写一个Stepper类

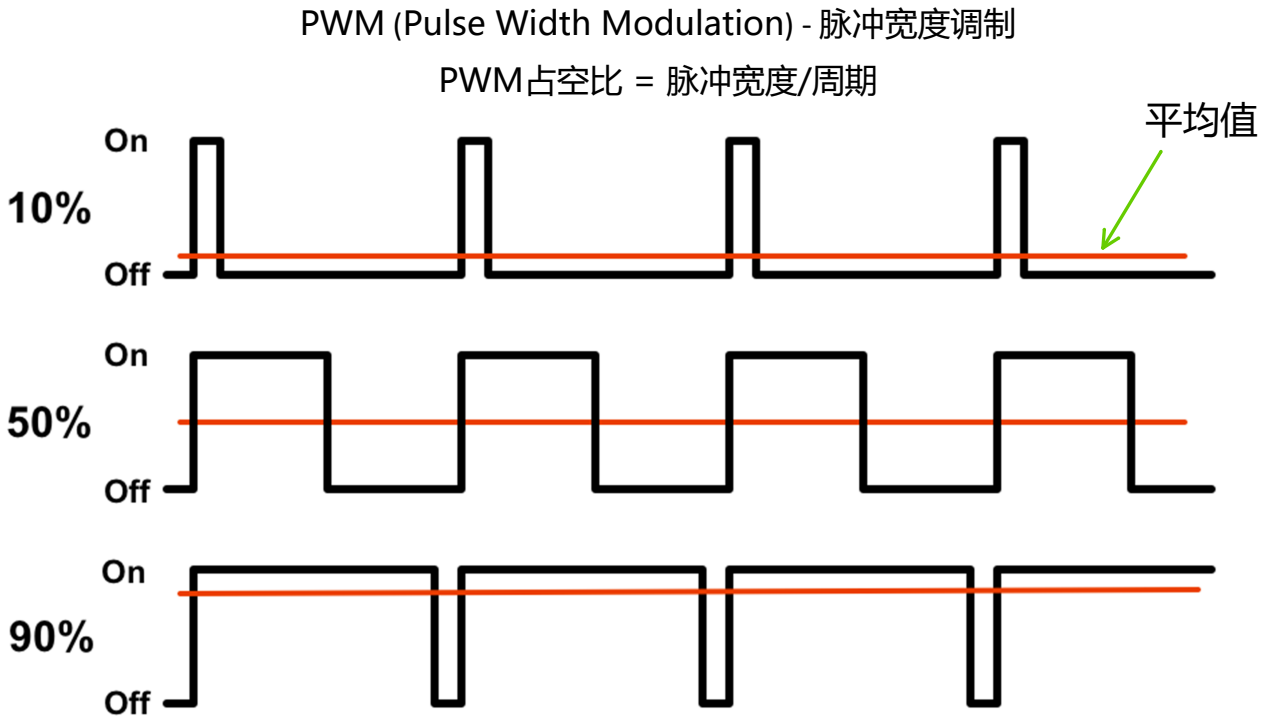
```
my_stepper = Stepper(16, 17, 5, 18)   # use pins 16, 17, 5, 18
my_stepper.rotate_angle(90)   # rotate 90 degrees clockwise
my_stepper.rotate_angle(-90)  # rotate 90 degrees counterclockwise
```

03 - PWM

2022年1月27日 10:45

- **PWM - 脉冲宽度调制**
- **直流电机**
- **舵机**

PWM - 脉冲宽度调制



MicroPython库

PWM (pulse width modulation)

https://docs.micropython.org/en/latest/esp32/quickref.html#pwm-pulse-width-modulation

```
from machine import Pin, PWM

pwm0 = PWM(Pin(0))          # create PWM object from a pin
freq = pwm0.freq()          # get current frequency (default 5kHz)
pwm0.freq(1000)             # set PWM frequency from 1Hz to 40MHz
```

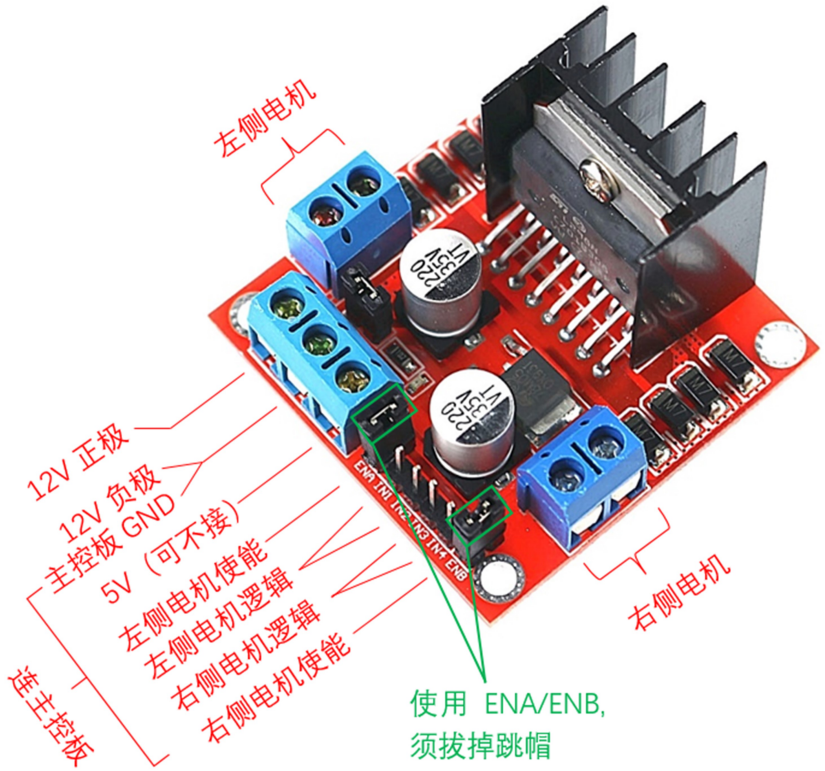
```
duty = pwm0.duty()         # get current duty cycle, range 0-1023 (default 512, 50%)
pwm0.duty(256)             # set duty cycle from 0 to 1023 as a ratio duty/1023 (25%)

duty_u16 = pwm0.duty_u16() # get current duty cycle, range 0-65535
pwm0.duty_u16(2**16*3//4)  # set duty cycle as a ratio duty_u16/65535 (75%)

duty_ns = pwm0.duty_ns()   # get current pulse width in ns
pwm0.duty_ns(250000)      # pulse width in nanoseconds from 0 to 1000000000/freq (25%)

pwm0.deinit()              # turn off PWM on the pin

pwm2 = PWM(Pin(2), freq=20000, duty=512)  # create and configure in one go
print(pwm2)                               # view PWM settings
```

PWM控制LED亮度 - 呼吸灯

led_fade.py

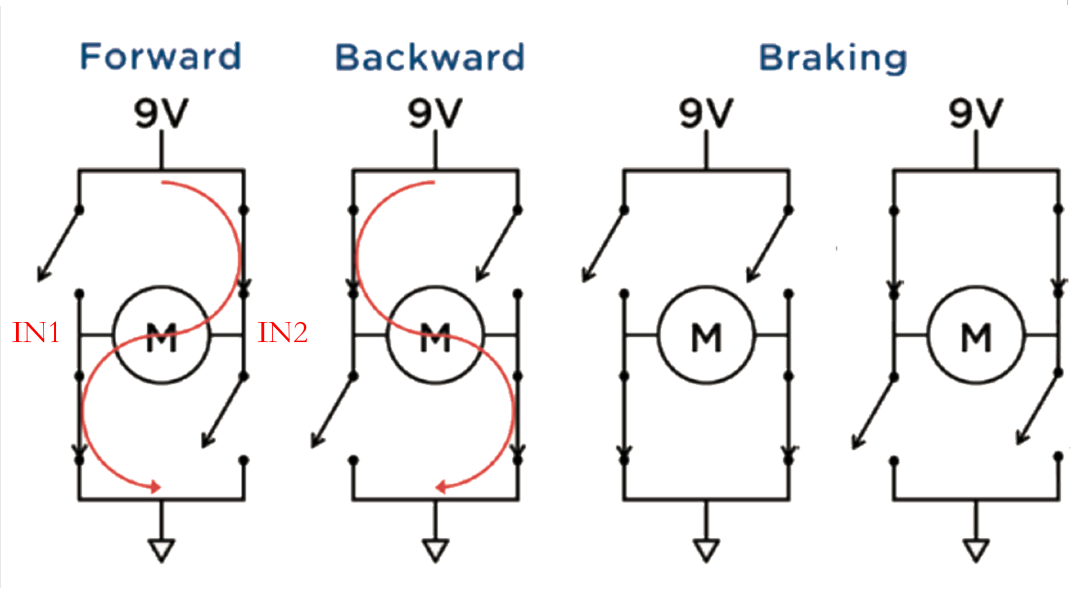
```
"""
PWM实现呼吸灯
NODEMCU-32S开发板上的LED与GPIO2相连
"""

from machine import Pin, PWM
led = Pin(2, Pin.OUT)

from machine import Pin, PWM
from time import sleep_ms
# 引脚2输出1000Hz PWM
led = PWM(Pin(2), freq = 1000)

while True:
    # 脉冲宽度由最小值0增加到最大值1023
    for i in range(1024):
        led.duty(i)
        sleep_ms(1)
    # 脉冲宽度由最大值1023减小到最小值0
    for i in range(1023, -1, -1):
        led.duty(i)
        sleep_ms(1)
```

直流电机控制 - L298N驱动模块

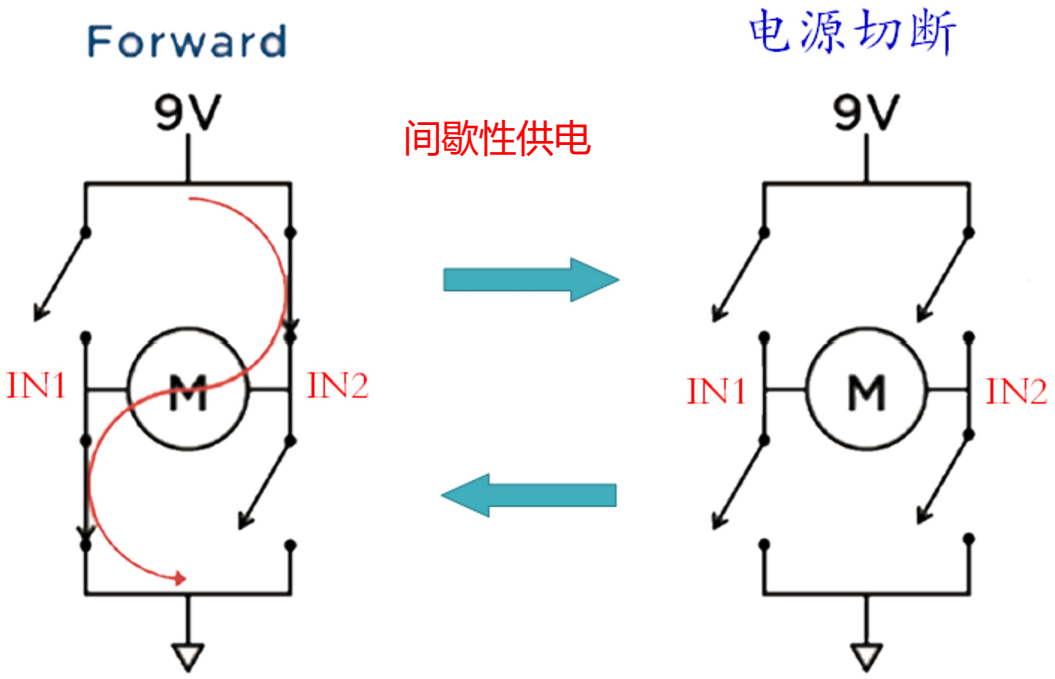


IN1	IN2	Motor Direction
0	0	Stopped (braking)
0	1	Clockwise
1	0	Counter-clockwise
1	1	Stopped (braking)

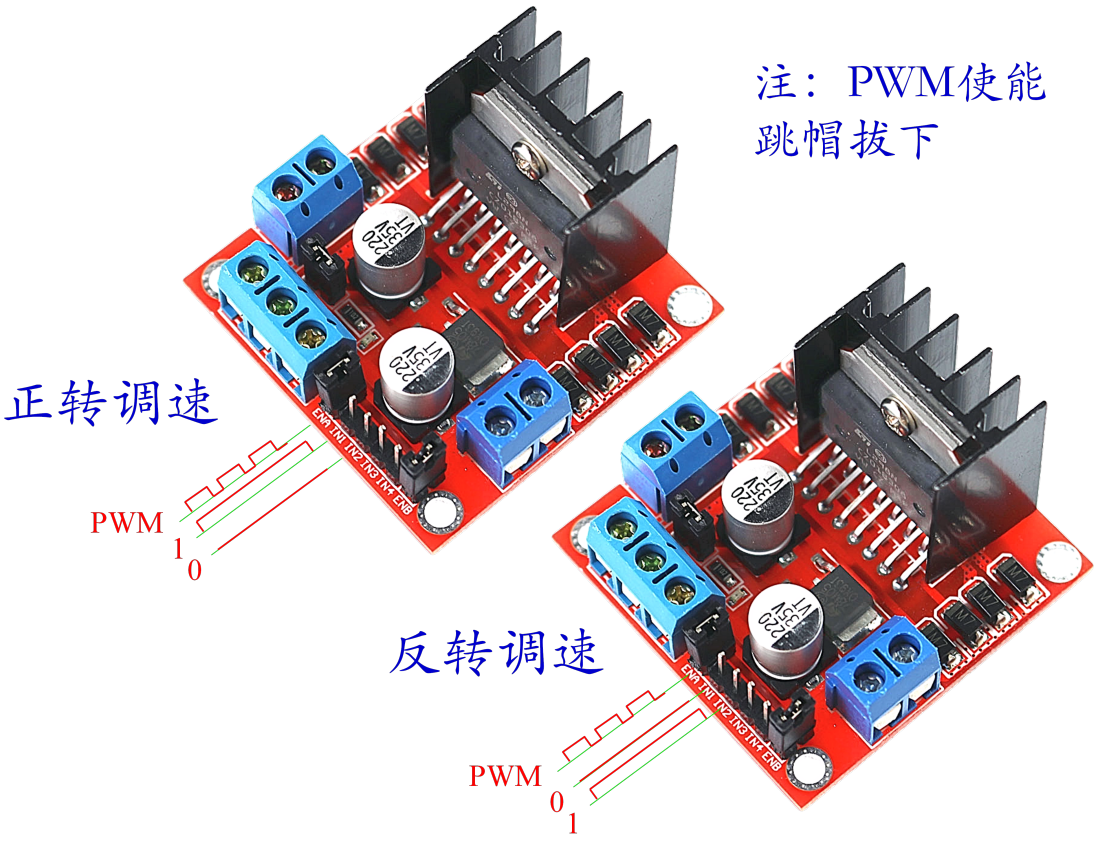
H桥电路 - 正反转

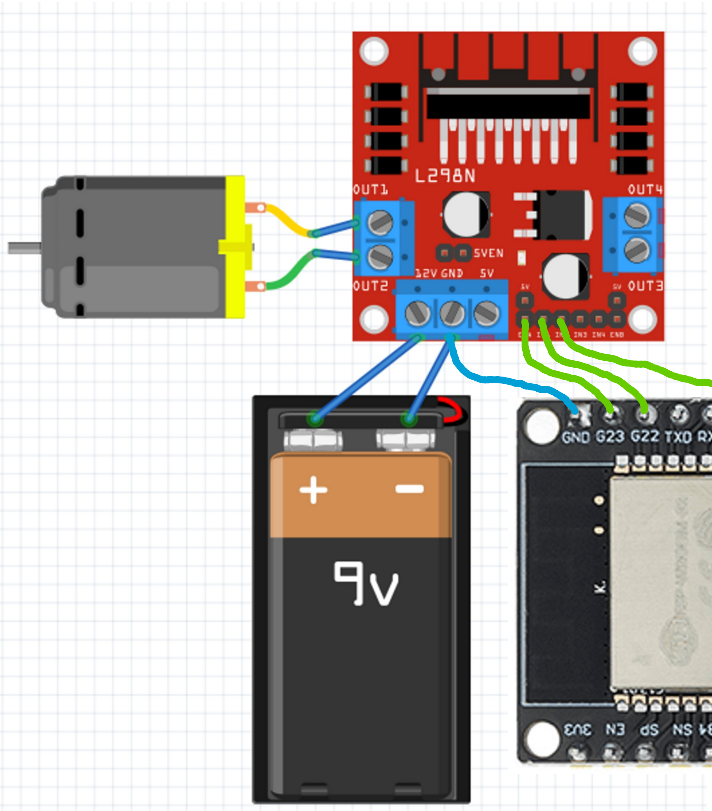


H桥电路 - 调速



用法1 - 三引脚（IN1/IN2/ENA)



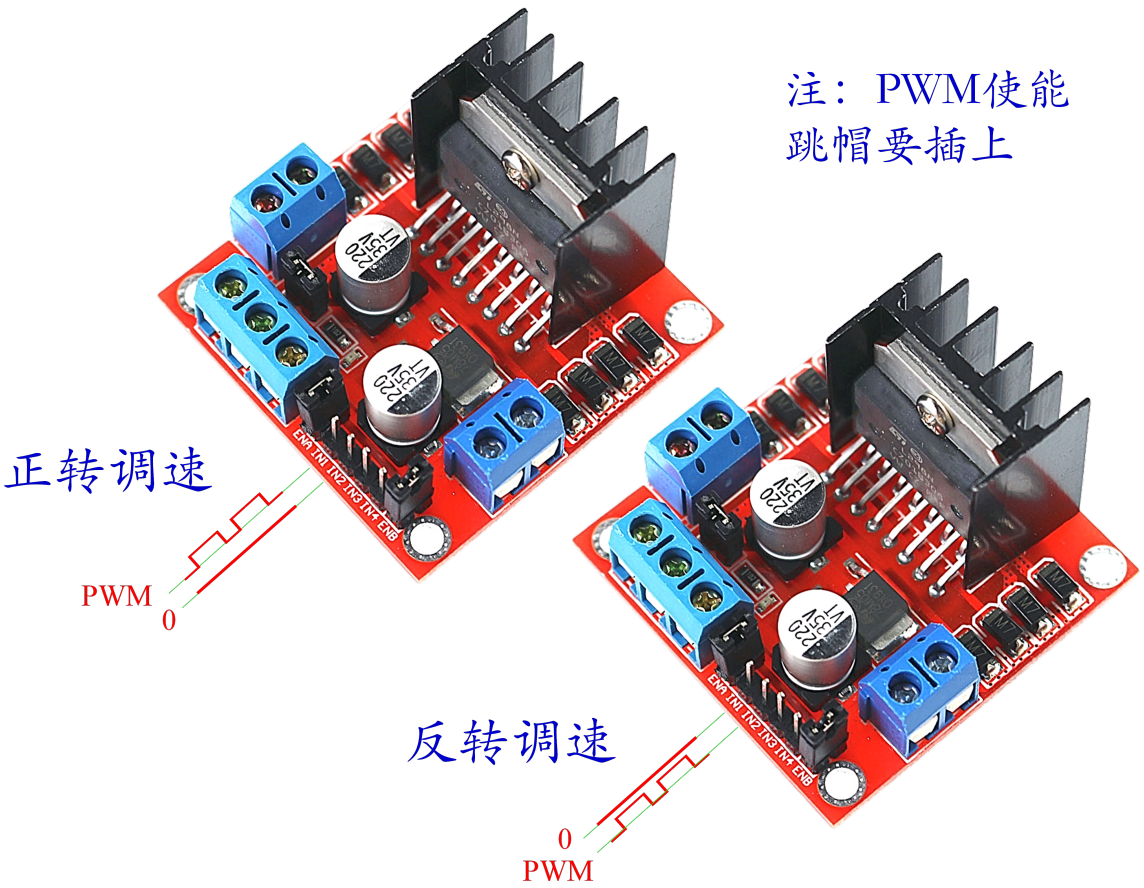


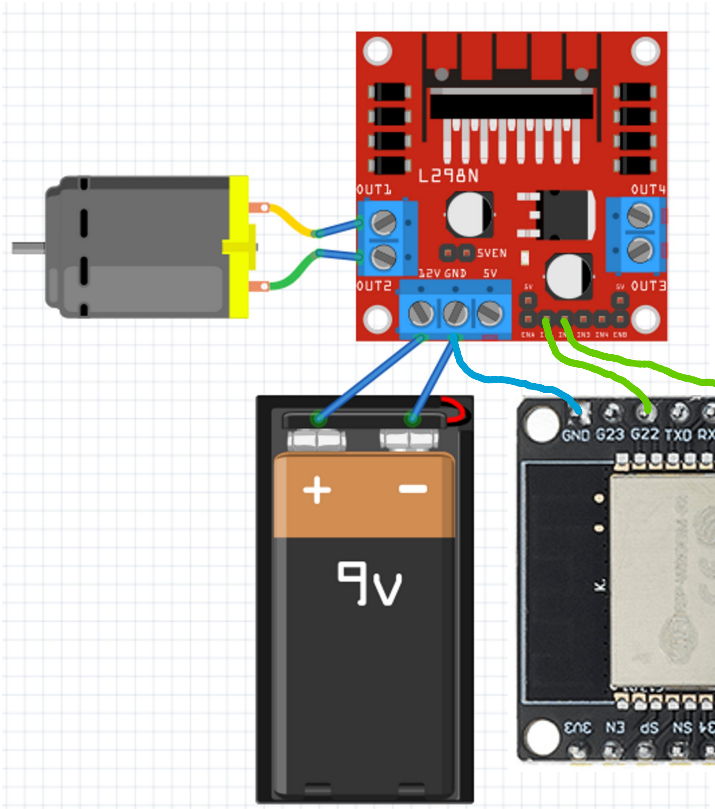
L298N_three_pin.py

```
"""
DC motor control using L298N module
Three-pin mode: ENA - pwm, IN1/IN2 - digital output
"""

from machine import Pin, PWM
import time

ENA = PWM(Pin(23), freq = 1000)
IN1 = Pin(22, Pin.OUT)
IN2 = Pin(21, Pin.OUT)

while True:
    # 全速正转（脉冲宽度1023）
    IN1.value(1)
    IN2.value(0)
    ENA.duty(1023)
    time.sleep(2)
    # 停止
    IN1.value(0)
    IN2.value(0)
    time.sleep(1)
    # 半速反转（脉冲宽度512）
    IN1.value(0)
```

```
IN2.value(1)
ENA.duty(512)
time.sleep(2)
# 停止
IN1.value(0)
IN2.value(0)
time.sleep(1)
```

用法2 - 两引脚 (IN1/IN2)





GND - GND
IN1 - GPIO22
IN2 - GPIO21
ENA跳帽要插上

L298N_two_pin.py

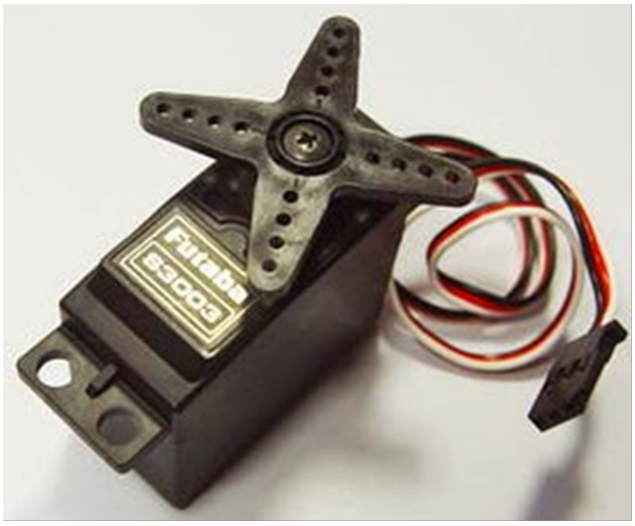
```
"""
Two-pin mode: IN1/IN2 - pwm
"""

from machine import Pin, PWM
import time

IN1 = PWM(Pin(22), freq = 1000)
IN2 = PWM(Pin(21), freq = 1000)

while True:
    # 全速正转（脉冲宽度1023）
    IN1.duty(1023)
    IN2.duty(0)
    time.sleep(2)
    # 停止
    IN1.duty(0)
    IN2.duty(0)
    time.sleep(1)
    # 半速反转（脉冲宽度512）
    IN1.duty(0)
    IN2.duty(512)
    time.sleep(2)
    # 停止
```

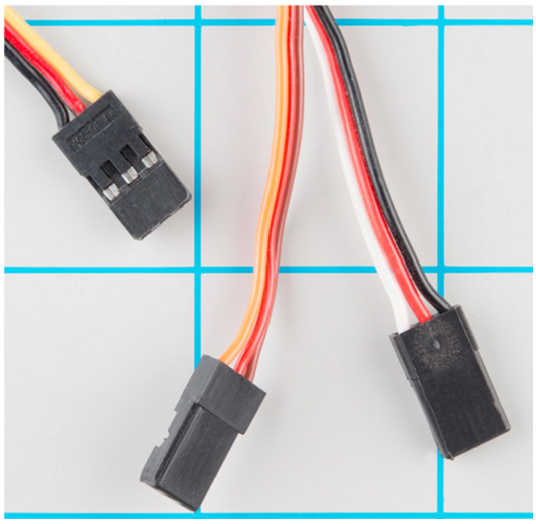
```
        IN1.duty(0)
        IN2.duty(0)
        time.sleep(1)
```

练习 - 编写电机驱动函数

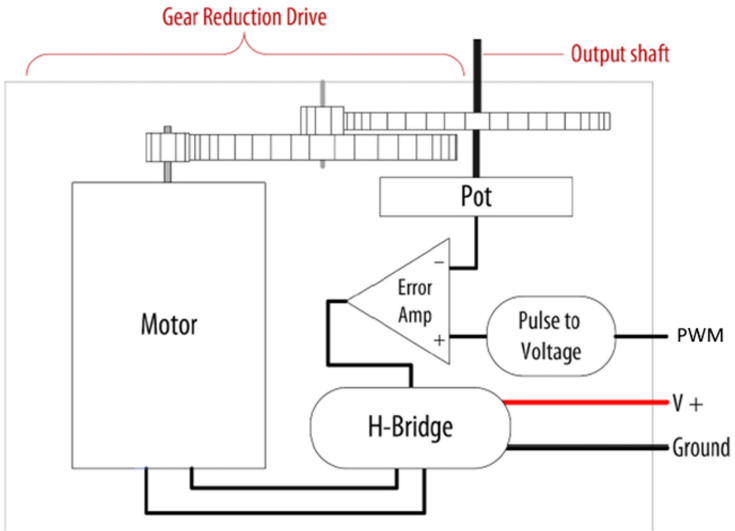
```
def motor_forward(speed):
    # your code here

def motor_backward(speed):
    # your code here

def motor_stop():
    # your code here
```

舵机控制

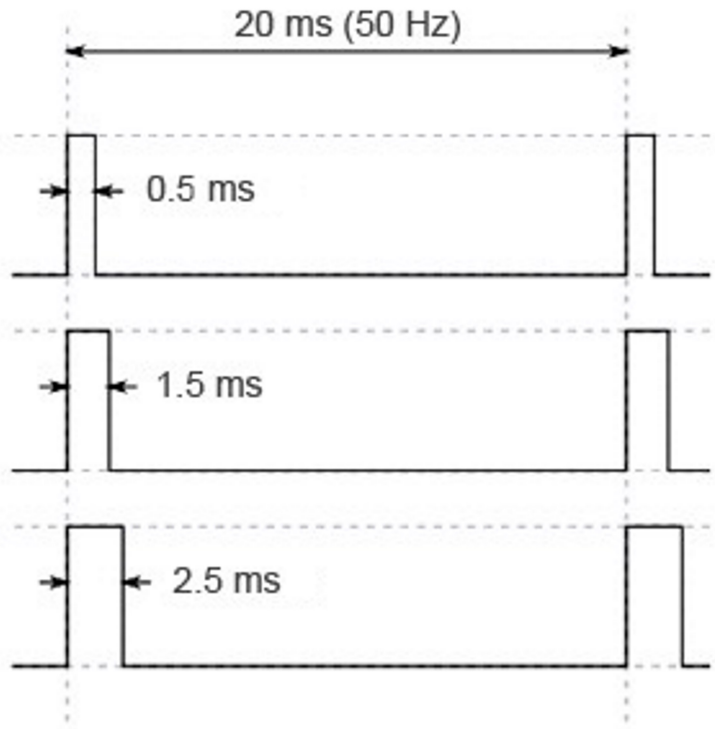




红色：5V　　黑色/棕色：GND　　黄色/白色：控制信号



舵机的控制信号是 PWM 信号，周期不变，高电平的时间决定舵机的位置。
PWM周期为20ms，脉冲宽度为0.5ms-2.5ms，对应舵盘的位置为0 - 180度。

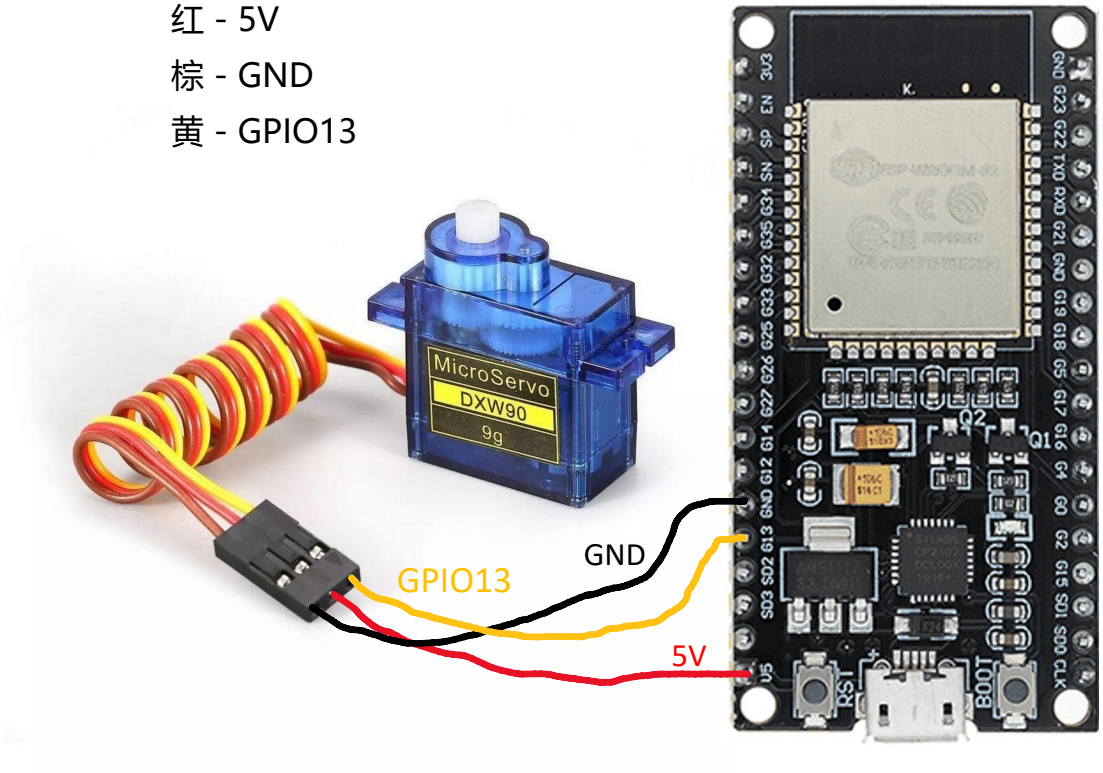


PWM周期： 20ms
PWM脉冲宽度时长： 0-20ms
PWM脉冲宽度数值： 0-1023
0度角脉冲宽度时长： 0.5ms
0度角脉冲宽度数值： $\text{int}(0.5 \times \frac{1023}{20}) = 25$
90度角脉冲宽度时长： 1.5ms
90度角脉冲宽度数值： 75
180度角脉冲宽度时长： 2.5ms
180度角脉冲宽度数值： 125
任意角度*x*脉冲宽度时长： $t = \frac{2.5-0.5}{180} x + 0.5$
任意角度*x*脉冲宽度数值：$\text{int}(t \times \frac{1023}{20})$

红 - 5V

棕 - GND

黄 - GPIO13



servo.py

```
"""
Servo motor
"""

import time
from machine import Pin, PWM

# 舵机控制信号线连接引脚13，PWM频率为50Hz
servo = PWM(Pin(13), freq = 50)

pulse_time_0 = 0.5          # 0度角，脉冲宽度时长为0.5ms
pulse_time_180 = 2.5        # 180度角，脉冲宽度时长为2.5ms
pwm_period = 20             # PWM周期为20ms
pwm_max_width = 1023        # PWM周期（20ms）对应的脉冲宽度数值

# 舵机角度转换为PWM脉冲宽度数值
def angle_to_pwm(x):
    # 角度x对应的脉冲宽度时长
    pulse_time = x * (pulse_time_180 - pulse_time_0) / 180 +
pulse_time_0
```

```
        # 脉冲宽度时长对应的脉冲宽度数值
        pulse_width = int(pulse_time * pwm_max_width / pwm_period)
        return pulse_width

    pwm0 = angle_to_pwm(0)      # 0度角脉冲宽度数值
    pwm90 = angle_to_pwm(90)    # 90度角脉冲宽度数值
    pwm180 = angle_to_pwm(180) # 180度角脉冲宽度数值

    while True:
        # 舵机位置0度
        servo.duty(pwm0)
        time.sleep(1)
        # 舵机位置90度
        servo.duty(pwm90)
        time.sleep(1)
        # 舵机位置180度
        servo.duty(pwm180)
        time.sleep(1)
        # 舵机位置90度
        servo.duty(pwm90)
        time.sleep(1)
```

练习 - 编写舵机库

servo.py

```
"""
servo library
"""

from machine import PWM, Pin

class Servo:
    def __init__(self, pin):
    # your code here


    def set_angle(self, x):
    # your code here
```

---

main_servo.py

```
"""
```

```
Demo of servo library
 - create a servo object on Pin 13
 - sweep from left to right and right to left
"""

import time, servo

my_servo = servo.Servo(13)

angle = 0.0
while True:
    while angle <= 180:
        my_servo.set_angle(angle)
        angle += 0.1
        time.sleep_ms(1)


    while angle >= 0:
        my_servo.set_angle(angle)
        angle -= 0.1
        time.sleep_ms(1)
```

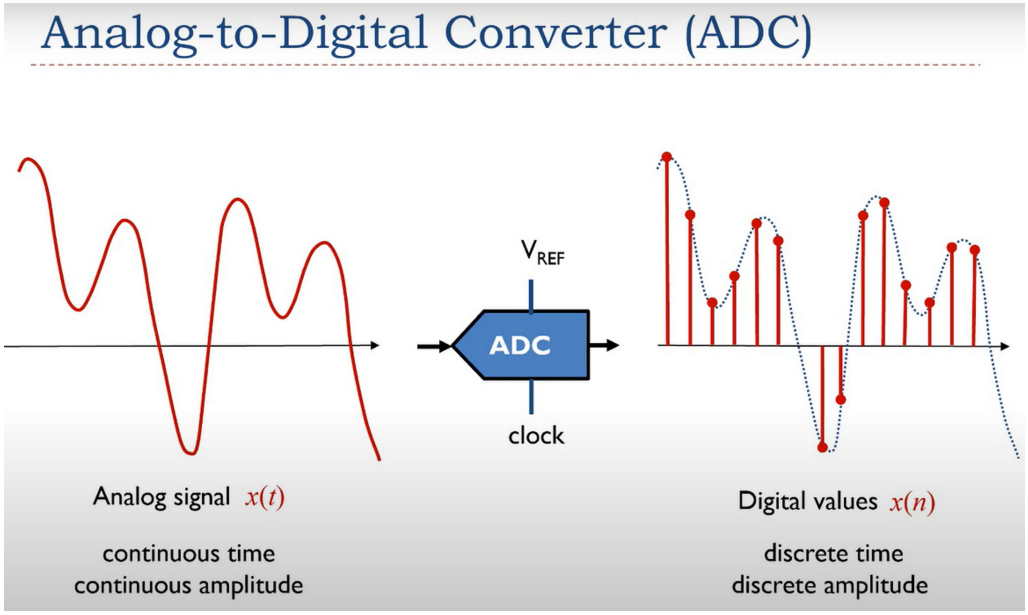
04 - ADC

2022年1月27日 10:46

- 红外循迹传感器模拟输出
- 电位器

ADC（模数转换器）

https://docs.micropython.org/en/latest/esp32/quickref.html#adc-analog-to-digital-conversion



ADC functionality is available on pins 32-39 (ADC block 1) and pins 0, 2, 4, 12-15 and 25-27 (ADC block 2).

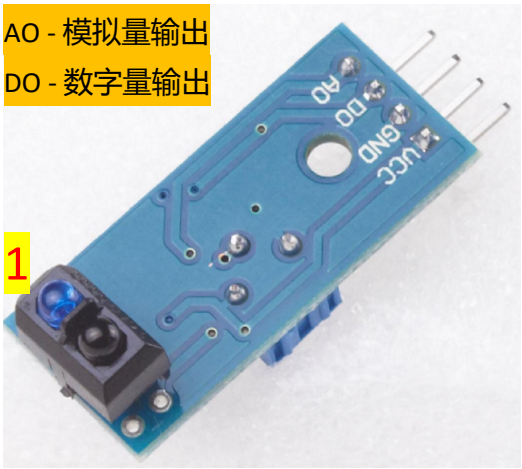
```
from machine import ADC
adc = ADC(Pin(32))          # create ADC object for pin 32
adc.read()                  # read raw value, 0-4095  (default 12-bit)
adc.read_u16()              # read raw value, 0-65535 (16-bit)
```

Note that the ESP32 uses an internal ADC reference voltage of 1.0v. To read voltages above this value, input attenuation can be applied with the optional atten keyword argument to the constructor. Valid values are:

- ADC.ATTN_0DB: No attenuation, this is the default
- ADC.ATTN_2_5DB: 2.5dB attenuation, gives a maximum input voltage of approximately 1.33v
- ADC.ATTN_6DB: 6dB attenuation, gives a maximum input voltage of approximately 2.00v
- ADC.ATTN_11DB: 11dB attenuation, gives a maximum input voltage of approximately 3.55v

```
adc = ADC(Pin(32), atten=ADC.ATTN_11DB)      # 0.0v - 3.55v range
```

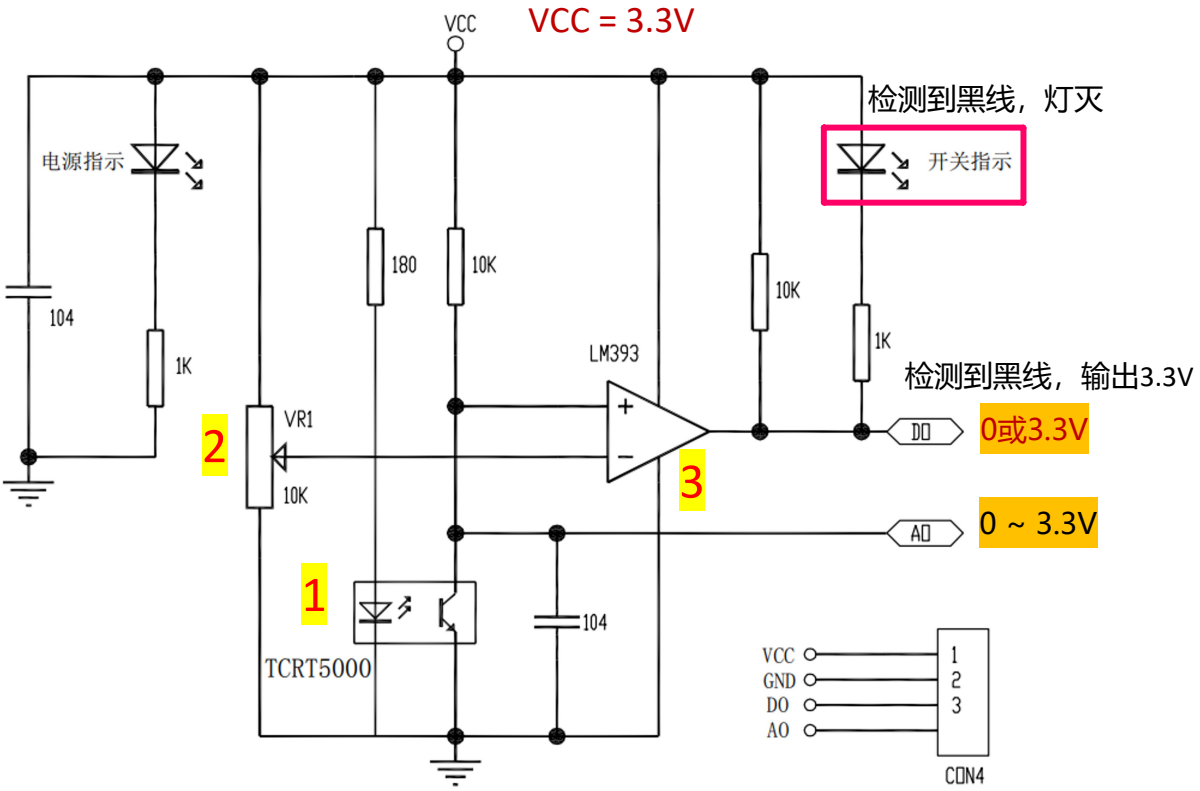
实验一：红外循迹传感器

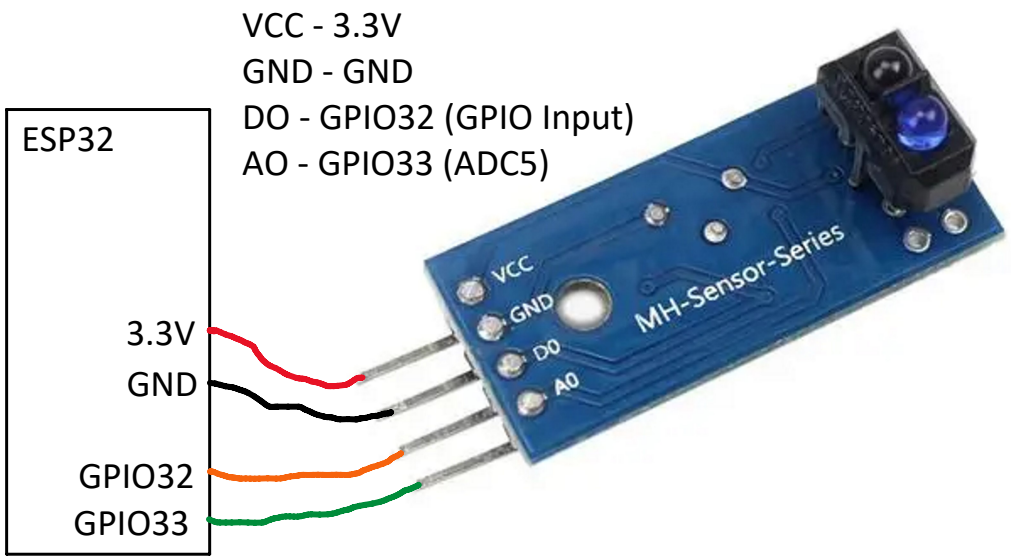


1 - 红外发射/接收管

2 - 电位器

3 - 比较器



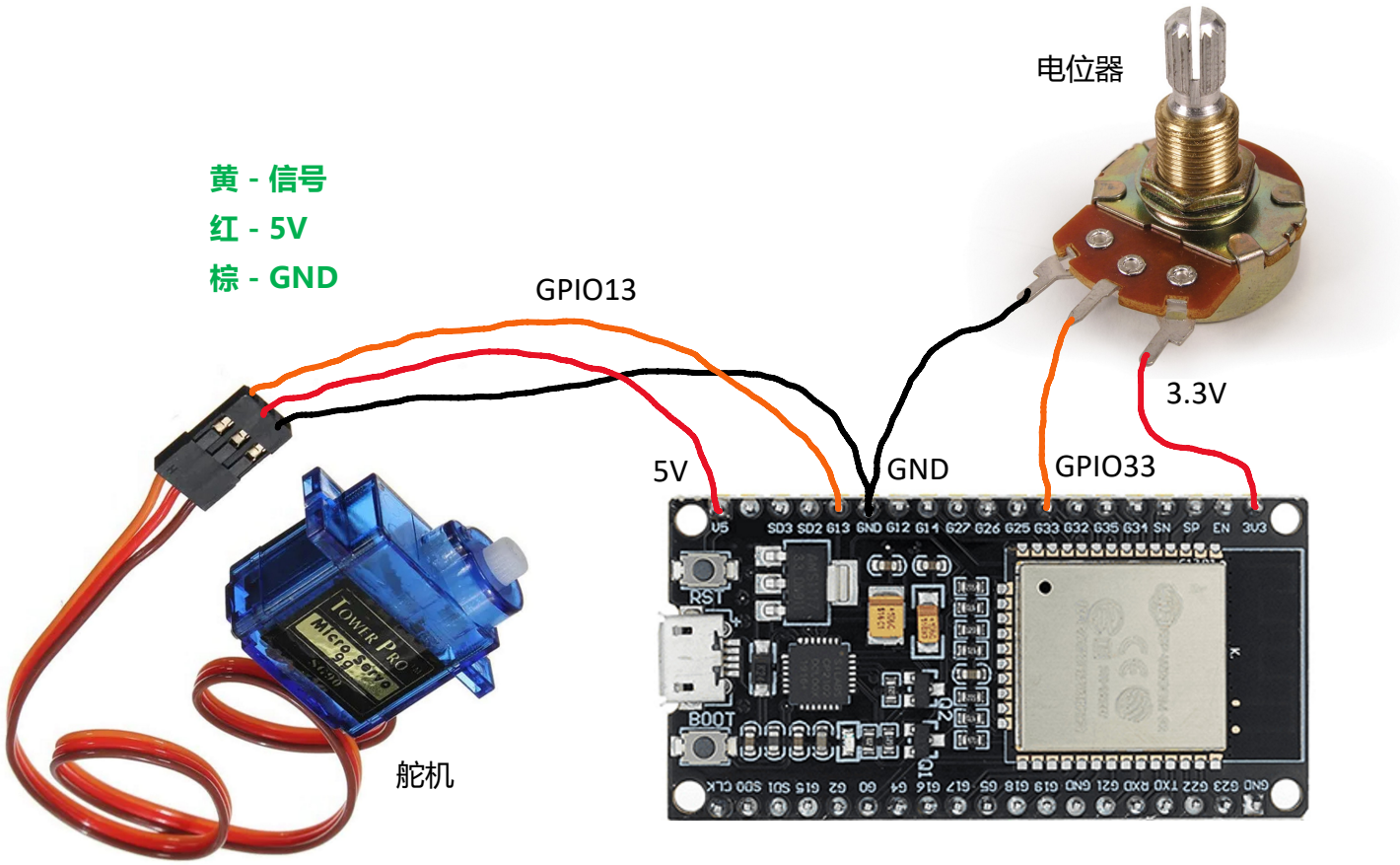


adc_sensor.py

```
"""
Infrared sensor outputs both digital and analog signals
"""

from machine import Pin, ADC
from time import sleep

# DO数字量输入, AO模拟量输入
DO = Pin(32, Pin.IN)
AO = ADC(Pin(33), atten = ADC.ATTN_11DB)  # 0.0v - 3.55v range

while True:
    print('DO = ', DO.value())
    print('AO = ', AO.read())
    sleep(1)
```

实验二：电位器控制舵机角度

电位器输出 --> ADC --> PWM脉冲宽度数值 --> 舵机角度



舵机角度与脉冲宽度数值

PWM周期： 20ms

PWM脉冲宽度时长： 0-20ms

PWM脉冲宽度数值： 0-1023

0度角脉冲宽度时长： 0.5ms

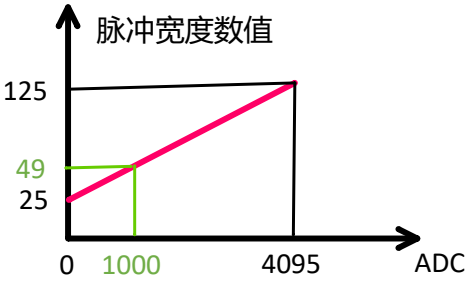
0度角脉冲宽度数值： $\text{int}(0.5 \times \frac{1023}{20}) = 25$

180度角脉冲宽度时长： 2.5ms

180度角脉冲宽度数值：125

电位器输出电压转换为舵机角度

电位器电压 (V)	ADC输出值	舵机角度	脉冲宽度数值
0	0	0	25
3.3	4095	180	125



$$脉冲宽度数值 = \frac{125-25}{4095} \times ADC + 25$$

Example:
ADC = 1000

$$脉冲宽度数值 = \frac{125-25}{4095} \times 1000 + 25 = 49$$

adc_pot_servo.py

```
"""
Control servo using potentiometer
"""

from machine import Pin, ADC, PWM

pot = ADC(Pin(32), atten = ADC.ATTN_11DB)  # 电位器 - ADC
servo = PWM(Pin(33), freq = 50)            # 舵机

while True:
    adc_value = pot.read()
    pulse_width_value = (125 - 25)/4095 * adc_value + 25
    servo.duty(int(pulse_width_value))
```

练习：电位器控制LED亮度

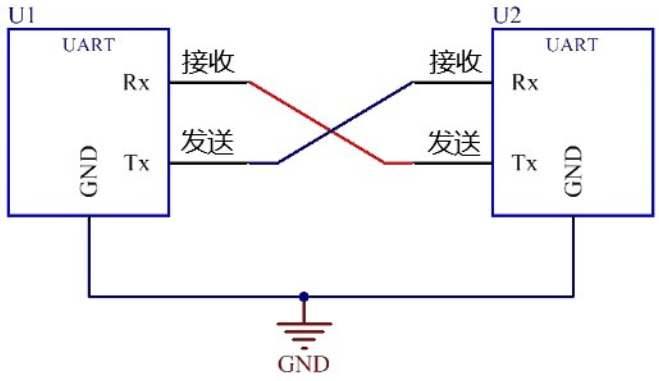
ADC测量电位器电压，输出值与LED引脚PWM占空比成正比。

05 - UART

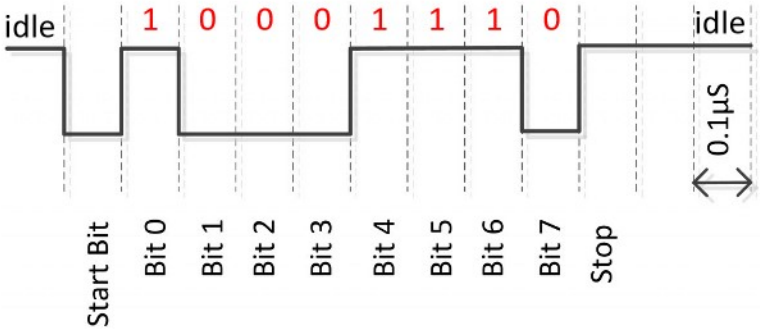
2022年1月27日 10:45

UART通信协议

UART - Universal Asynchronous Receiver/Transmitter



0x71, 8N1 (8 Data bits, No Parity, 1 Stop)



- 空闲时保持高电平（1）
- 准备发送，拉低电平（0），即start bit
- 开始发送数据，共8位（8 bits）
- 发送完毕，拉高电平（1），即stop bit
- 回到空闲状态，保持高电平（1）

常用波特率 (baud rate)：

9600, 14400, 19200, 38400, 57600, 115200, 128000，256000

The ESP32 has three hardware UARTs: UART0, UART1 and UART2.

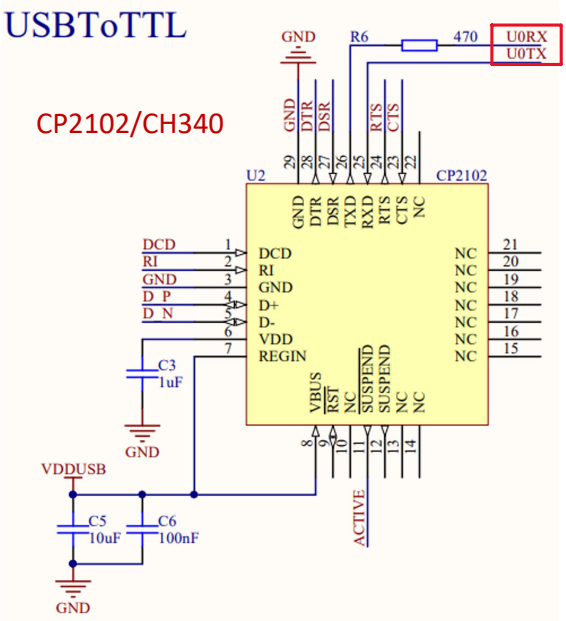
Any GPIO can be used for hardware UARTs.

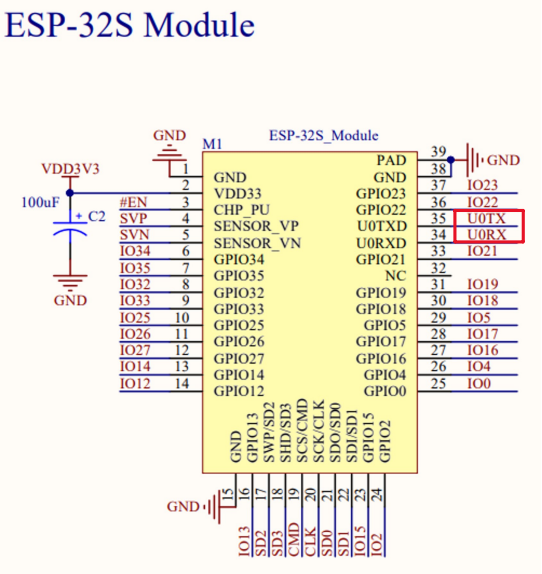
https://docs.micropython.org/en/latest/esp32/quickref.html#uart-serial-bus
https://docs.micropython.org/en/latest/library/machine.UART.html#machine-uart

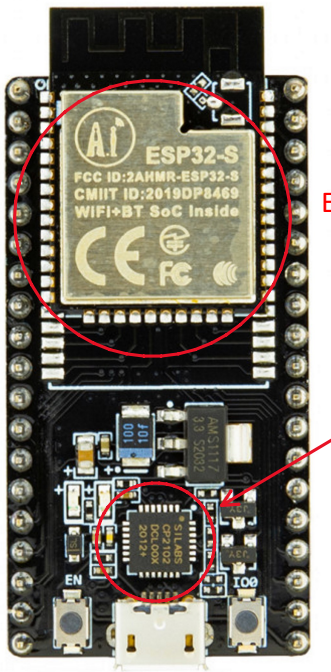
```
from machine import UART

uart1 = UART(1, baudrate=9600, tx=33, rx=32)
uart1.write('hello')  # write 5 bytes
uart1.read(5)         # read up to 5 bytes

uart2 = UART(2, baudrate=115200, tx=17, rx=16)
uart2.read()          # read all available characters
uart2.readline()      # read a line, terminate on '\n' or timeout
uart2.readinto(buf)   # read and store into the given buffer
```

UART0 - MicroPython和PC通信

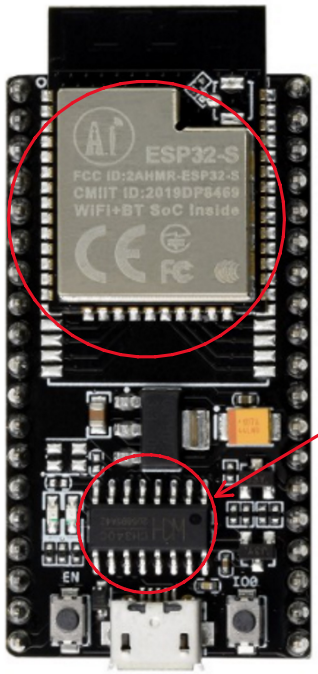
UART0与串口转USB芯片（CP2102或CH340)连接，MicroPython通过UART0与PC通信。







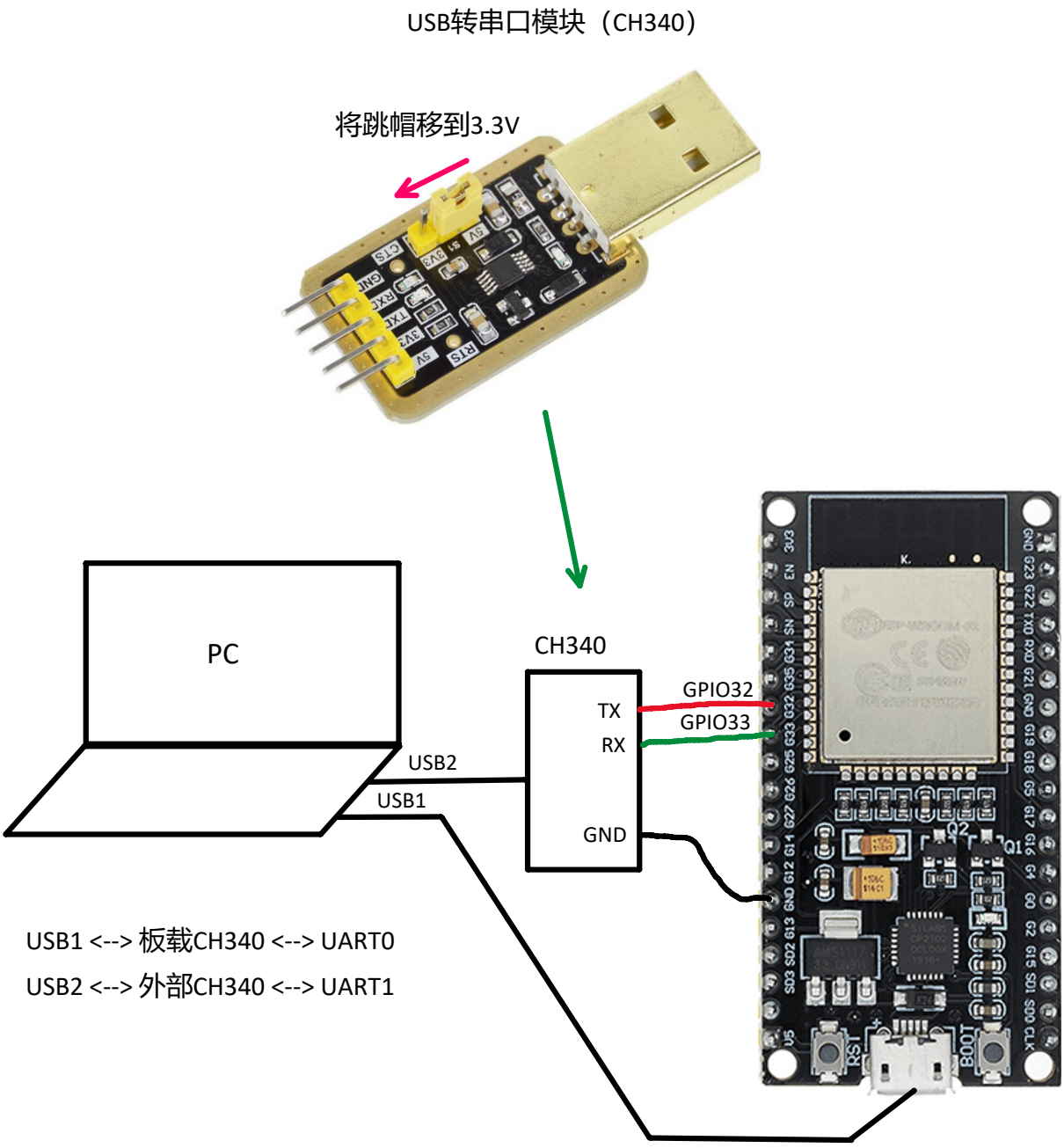
CP2102版



CH340版

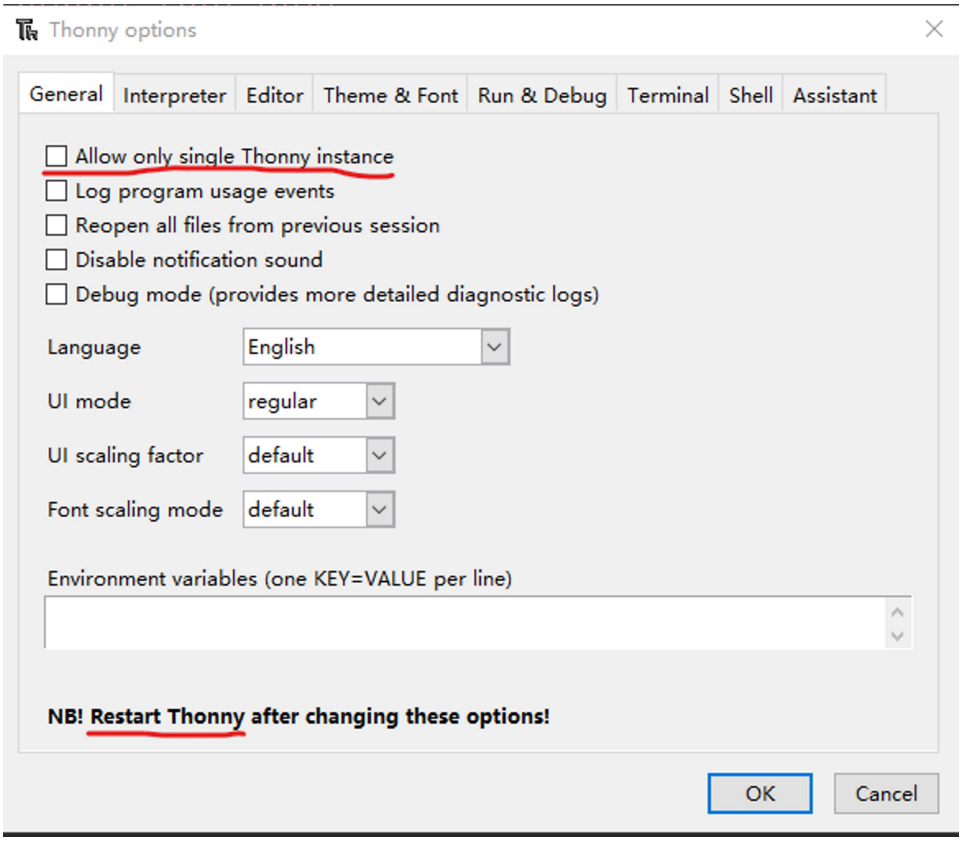
CP2102和CH340作用相同，
将USB信号转换为UART信号

实验：PC和ESP32发送接收数据

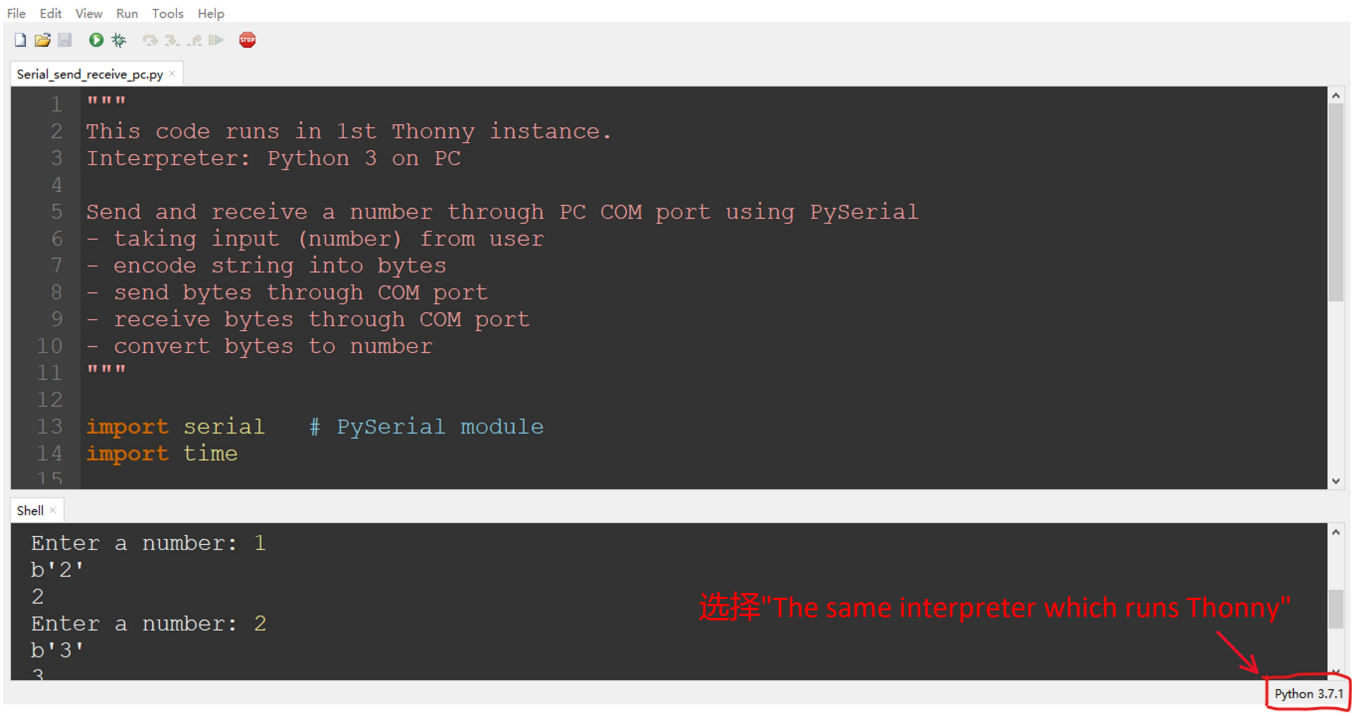


1. 发送接收一个整数
   - 电脑端用户输入一个整数
   - 电脑通过串口（CH340）发送给ESP32
   - ESP32将接收到的整数加1，并发送给电脑
   - 电脑接收并显示被修改的整数

启用两个Thonny界面



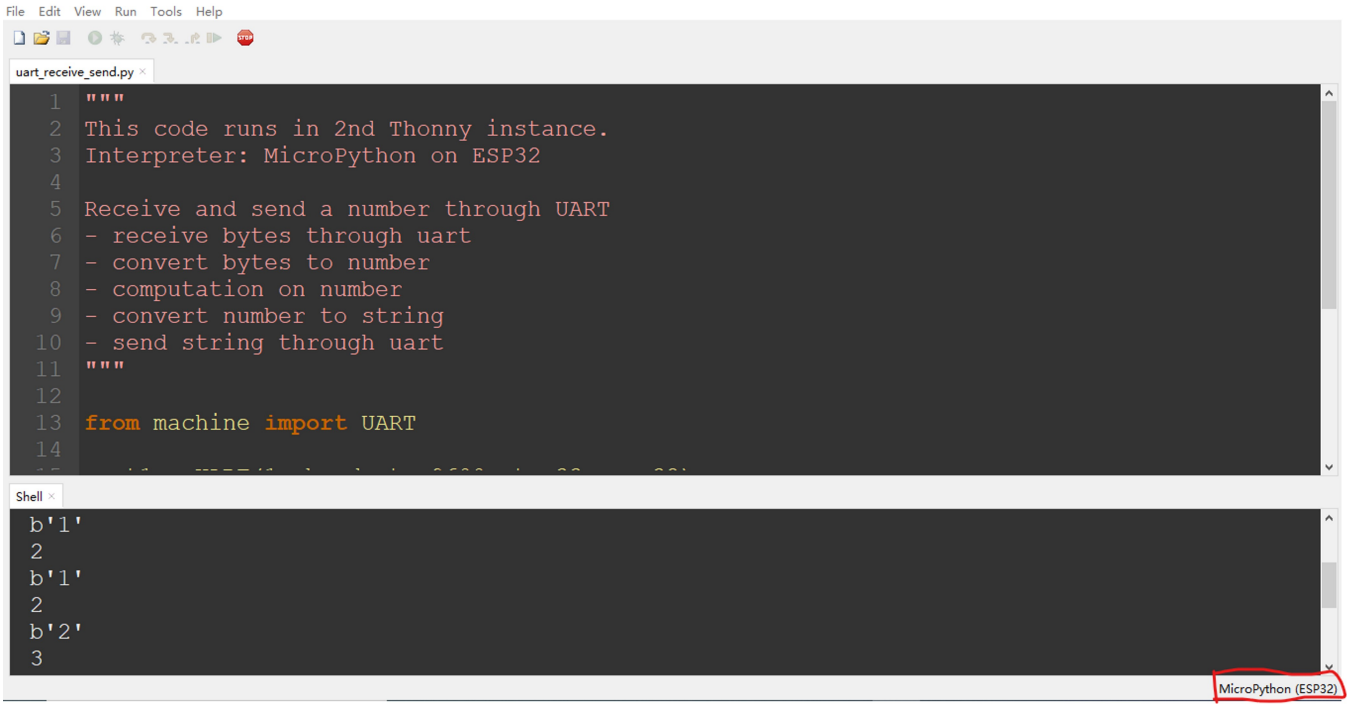
第一个Thonny界面，解释器为Python3.7，运行Serial_send_receive_pc.py



serial_send_receive_pc.py

```
"""
This code runs in 1st Thonny instance.
Interpreter: Python 3 on PC

Send and receive an integer through PC COM port using PySerial
- taking input (integer) from user
- encode string into bytes
- send bytes through COM port
- receive bytes through COM port
- convert bytes to integer
"""

import serial   # PySerial module
import time

esp32 = serial.Serial(port='COM3', baudrate=9600, timeout=0.1)

while True:
    number_string = input("Enter an integer number: ")  # taking input
from user
    number_bytes = number_string.encode('utf-8')  # convert string to
bytes
    esp32.write(number_bytes)    # send bytes
    time.sleep(0.05)
    if esp32.in_waiting:      # incoming data available
        data_bytes = esp32.readline()   # receive bytes
        print(data_bytes)
        number = int(data_bytes)    # convert bytes to integer
        print(number)
```

第二个Thonny界面，解释器为MicroPython，运行uart_send_receive.py



uart_send_receive.py

```
"""
This code runs in 2nd Thonny instance.
Interpreter: MicroPython on ESP32

Receive and send an integer through UART
- receive bytes through uart
- convert bytes to integer
- computation on integer
- convert integer to string
- send string through uart
"""

from machine import UART

uart1 = UART(1, baudrate=9600, tx=33, rx=32)

while True:
    if uart1.any():    # incoming data available
        number_bytes = uart1.read() # bytes
        print(number_bytes)
        number = int(number_bytes) + 1  # convert bytes to integer and
add 1
        print(number)
        number_string = str(number)  # convert integer to string
        uart1.write(number_string)   # send string
```

2. 发送/接收浮点数 - 打包解包数据

- struct.pack('f', float_number)，发送端将浮点数打包为bytes
- struct.unpack('f', raw_bytes)，接收端将bytes解包为浮点数

PC发送端代码

serial_send_receive_struct_pc.py

```
"""
This code runs on PC.
Send and receive through COM port
Pack and unpack floats using struct
"""

import serial, time, random, struct

esp32 = serial.Serial(port='COM3', baudrate=9600, timeout=.1)

# code for sender
while True:
    float_1 = random.uniform(1, 100)  # generate random float between 1 and 100
    float_2 = random.uniform(1, 100)
    packet_bytes = struct.pack('ff', float_1, float_2)  # 'ff': two floats
    esp32.write(packet_bytes)   # send 8 bytes, one float is 4 bytes
    time.sleep(0.01)
```

ESP32端接收代码

uart_send_receive_struct.py

```
"""
This code runs on ESP32.
UART send and receive
Pack and unpack floats using struct
"""

import machine, time, random, struct

uart1 = machine.UART(1, baudrate=9600, tx=33, rx=32)

# code for sender
while True:
    float_1 = random.uniform(1, 100)
    float_2 = random.uniform(1, 100)
    packet_bytes = struct.pack('ff', float_1, float_2)  # 'ff': two floats
    uart1.write(packet_bytes)   # send 8 bytes, one float is 4 bytes
    time.sleep_ms(500)

# code for receiver
while True:
    if uart1.any():
        packet_bytes = uart1.read(8)   # receive 8 bytes
        float_1, float_2 = struct.unpack('ff', packet_bytes)   # unpack into two floats
        print(float_1, float_2)
```

1. 发送/接收浮点数 - 标记数据包

   防止接收错误数据包的措施：

- 发送端将数据打包为'<数据>'， '<' 和 '>'为数据包开始结束标记
- 接收端检验数据包大小以及开始结束标记

PC发送端代码

serial_send_markers_pc.py

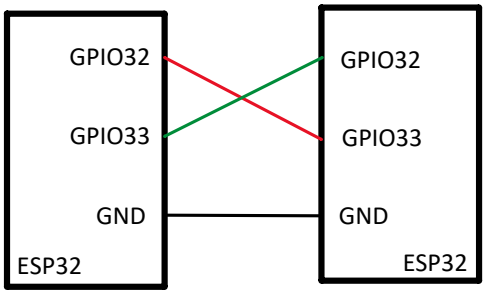
```
"""
This code runs on PC.
Send floats through COM port
Float bytes are guarded by markers: b'<bytes>'
Pack and unpack floats using struct
"""

import serial, time, random, struct

esp32 = serial.Serial(port='COM3', baudrate=115200, timeout=.1)

while True:
    float_1 = random.uniform(1, 100)  # generate random float between 1 and 100
    float_2 = random.uniform(1, 100)
    raw_bytes = b'<' + struct.pack('ff', float_1, float_2) + b'>'
    esp32.write(raw_bytes)   # send bytes
    time.sleep(0.1)
```

ESP32端接收代码

uart_receive_markers.py

```
"""
This code runs on ESP32.
UART receiving bytes guarded by markers: b'<bytes>'
"""

import time, machine, struct

uart1 = machine.UART(1, baudrate=115200, tx=33, rx=32)

while True:
    if uart1.any():
        raw_bytes = uart1.read()  # '<ff>'
        # float = 4 bytes, char = 1 byte
        # two floats + two chars = 10 bytes
        # '<'==60, '>'==62 in ASCII
        if len(raw_bytes) == 10 and raw_bytes[0] == 60 \
                                and raw_bytes[-1] == 62:
            float_1, float_2 = struct.unpack('ff', raw_bytes[1:-1])
            print(float_1, float_2)
```

练习：ESP32与ESP32通信



1. 发送接收浮点数

   第一块ESP32发送，第二块ESP32接收

2. 控制LED

   第一块ESP32

   - 提示用户输入指令，Please Enter Option (0-Off, 1-On, 2-Blink):
   - 发送指令
   - 接收第二块ESP32的确认信息， LED off， LED on， LED blinking

   第二块ESP32

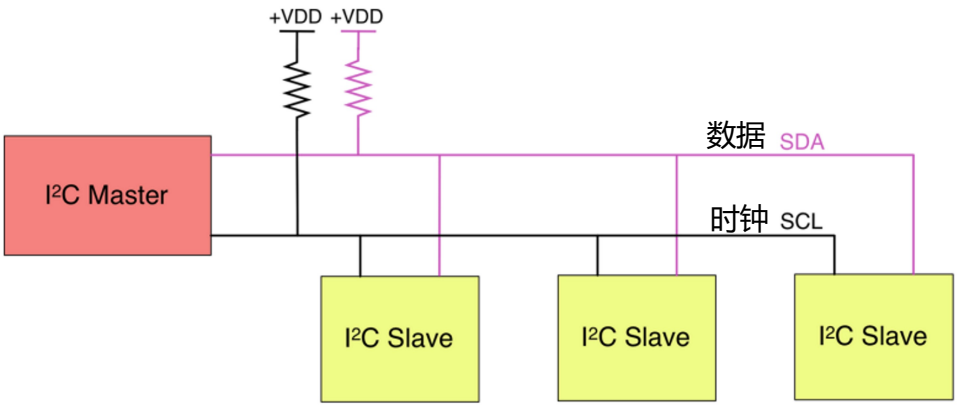
   - 接收指令，如果为'0'，熄灭板载LED，如果为'1'，点亮板载LED，如果为'2'，闪灯
   - 发送相应的确认信息：LED off， LED on， LED blinking

06 - I²C

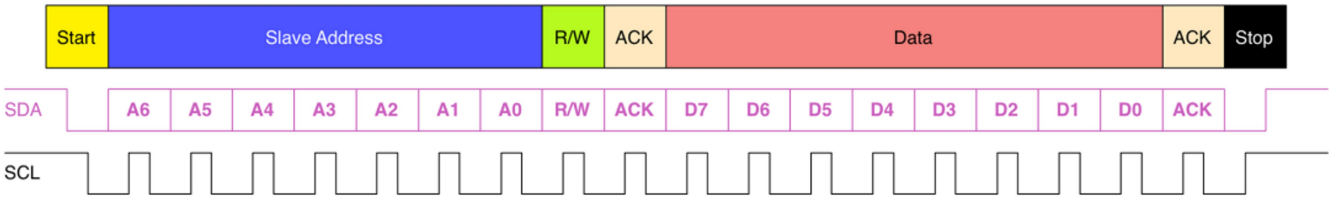
2021年9月21日 21:09

- 温度传感器TMP102
- 惯性传感器MPU6050

I²C总线：一主多从



I²C通信协议



- 主机发送开始信号，拉低SDA，并开始发送时钟信号SCL
- 主机发送从机地址以及读写位（7个地址位，1个读写位R/W，共8位）
- 地址匹配的从机发送确认信号（ACK）
- 主机或从机发送数据（写/读）
- 主机或从机发送确认信号（ACK）(读/写)
- 传送结束，主机发送结束信号（释放SDA），并停止发送时钟信号SCL

ESP32 I²C

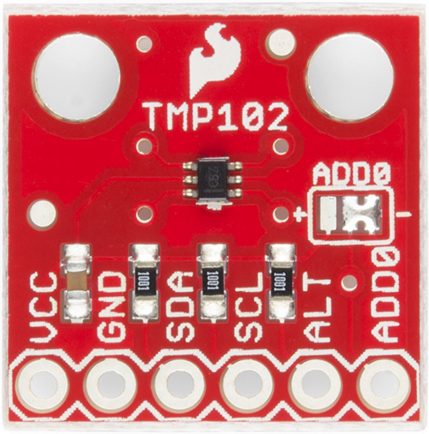
https://docs.micropython.org/en/latest/esp32/quickref.html#hardware-i2c-bus
https://docs.micropython.org/en/latest/library/machine.I2C.html#machine-i2c

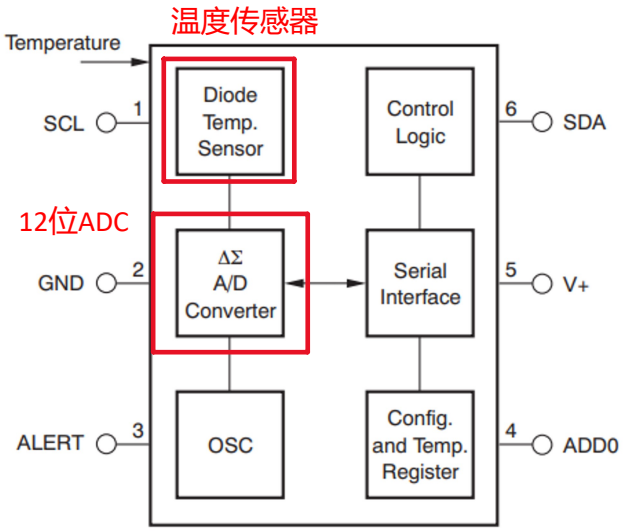
There are two hardware I2C peripherals with identifiers 0 and 1. Any available output-capable pins can be used for SCL and SDA but the defaults are given below.

	I2C(0)	I2C(1)
scl	18	25
sda	19	26

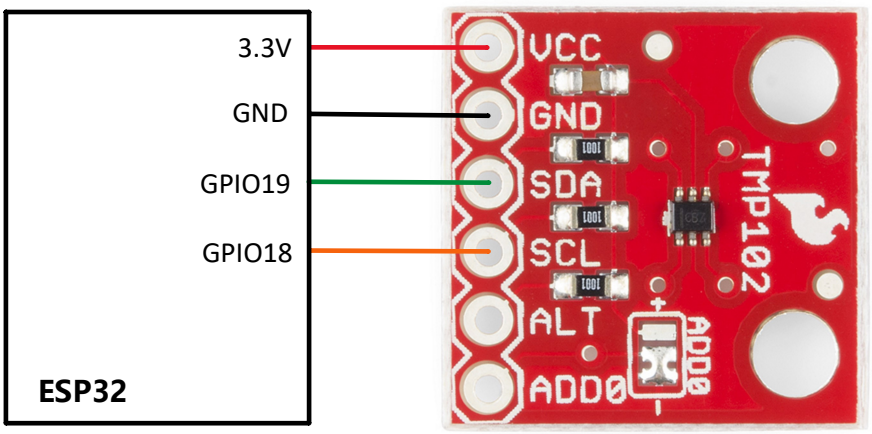
```
from machine import Pin, I2C

i2c = I2C(0)
i2c = I2C(1, scl=Pin(5), sda=Pin(4), freq=400000)

i2c.scan()              # scan for peripherals, returning a list of 7-bit addresses

i2c.writeto(42, b'123')         # write 3 bytes to peripheral with 7-bit address 42
i2c.readfrom(42, 4)             # read 4 bytes from peripheral with 7-bit address 42

i2c.readfrom_mem(42, 8, 3)      # read 3 bytes from memory of peripheral 42,
                                #   starting at memory-address 8 in the peripheral
i2c.writeto_mem(42, 2, b'\x10') # write 1 byte to memory of peripheral 42
                                #   starting at address 2 in the peripheral
```

温度传感器TMP102

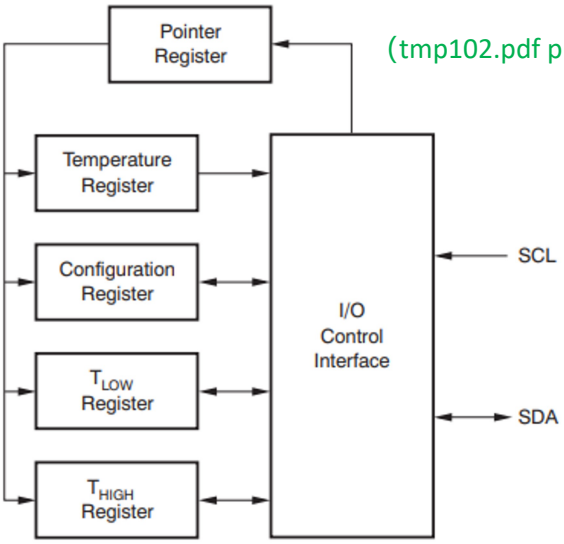




传感器模块和开发板的连接



内部寄存器



(tmp102.pdf p16)

Figure 12. Internal Register Structure

温度寄存器包含两个字节

Table 8. Byte 1 of Temperature Register(1) (tmp102.pdf p16)

D7	D6	D5	D4	D3	D2	D1	D0
T11	T10	T9	T8	T7	T6	T5	T4
(T12)	(T11)	(T10)	(T9)	(T8)	(T7)	(T6)	(T5)

(1) Extended mode 13-bit configuration shown in parenthesis.

左移四位：T11T10T9T8T7T6T5T40000

Table 9. Byte 2 of Temperature Register(1) (tmp102.pdf p17)

D7	D6	D5	D4	D3	D2	D1	D0
T3	T2	T1	T0	0	0	0	0
(T4)	(T3)	(T2)	(T1)	(T0)	(0)	(0)	(1)

(1) Extended mode 13-bit configuration shown in parenthesis.

右移四位：0000T3T2T1T0

做或运算：T11T10T9T8T7T6T5T40 0 0 0
0 0 0 0 0 0 0 0 T3T2T1T0
T11T10T9T8T7T6T5T4T3T2T1T0

ADC输出值转换为温度

(tmp102.pdf p9)

- 温度为正

To convert a positive digital data format to temperature:

1. Convert the 12-bit, left-justified binary temperature result, with the MSB = 0 to denote a positive sign, to a decimal number.
2. Multiply the decimal number by the resolution to obtain the positive temperature.

   Example: 0011 0010 0000 = 320h = 800 × (0.0625°C / LSB) = 50°C

- 温度为负

To convert a negative digital data format to temperature: 补码转换（two's complement）

1. Generate the twos compliment of the 12-bit, left-justified binary number of the temperature result (with MSB = 1, denoting negative temperature result) by complementing the binary number and adding one. This represents the binary number of the absolute value of the temperature.
2. Convert to decimal number and multiply by the resolution to get the absolute temperature, then multiply by –1 for the negative sign.

   Example: 1110 0111 0000 has twos compliment of 0001 1001 0000 = 0001 1000 1111 + 1

   Convert to temperature: 0001 1001 0000 = 190h = 400; 400 × (0.0625°C / LSB) = 25°C = (|–25°C|); (|–25°C|) × (–1) = –25°C

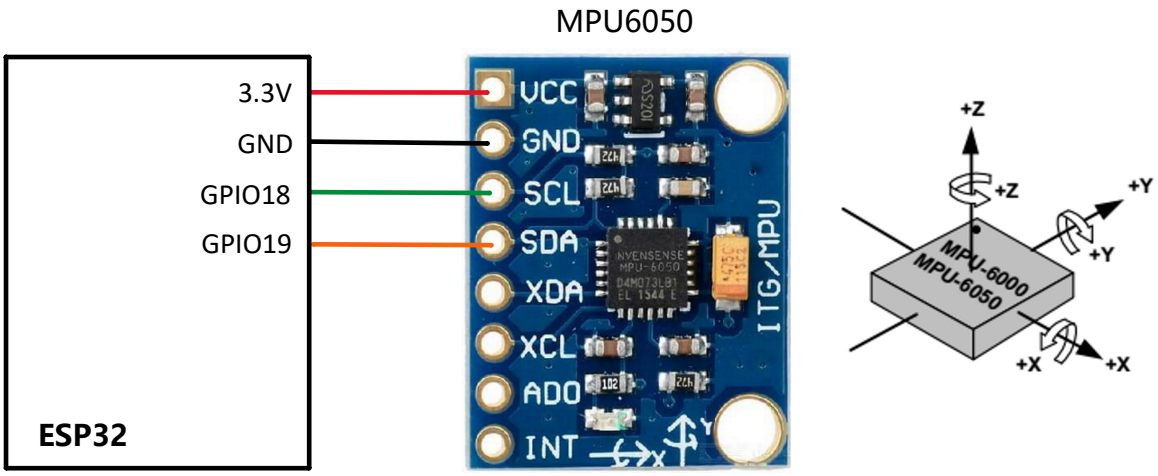
读取温度

tmp102.py

```
"""
TMP102 Temperature Sensor

Temperature register is 12-bit.
Throw away D3-D0 of byte 2 and combine with byte 1.

Byte 1 of Temperature Register
D7  D6  D5 D4 D3 D2 D1 D0
T11 T10 T9 T8 T7 T6 T5 T4

Byte 2 of Temperature Register
D7 D6 D5 D4 D3 D2 D1 D0
T3 T2 T1 T0 0  0  0  0
"""

from machine import I2C, Pin
import time

i2c = I2C(0, scl=Pin(18), sda=Pin(19))

TMP102_ADDR = 0x48     # TMP102 address on the I2C bus
TEMP_REG    = 0x00     # temperature register address

while True:
    # Read two bytes from temperature register
    raw_bytes = i2c.readfrom_mem(TMP102_ADDR, TEMP_REG, 2)
    value = (raw_bytes[0] << 4) | (raw_bytes[1] >> 4)   # get 12 bits from two bytes
    if value & (1 << 11) != 0:     # bit 11 == 1, value is negative
        value = value - (1 << 12)  # two's complement
    temp_c = value * 0.0625        # temperature (C)
    print(round(temp_c , 2), "C")
    time.sleep(1)
```

MPU6050

- 三轴加速度计：XYZ加速度
- 三轴陀螺仪：XYZ角速度



读取z轴加速度

mpu6050_simple.py

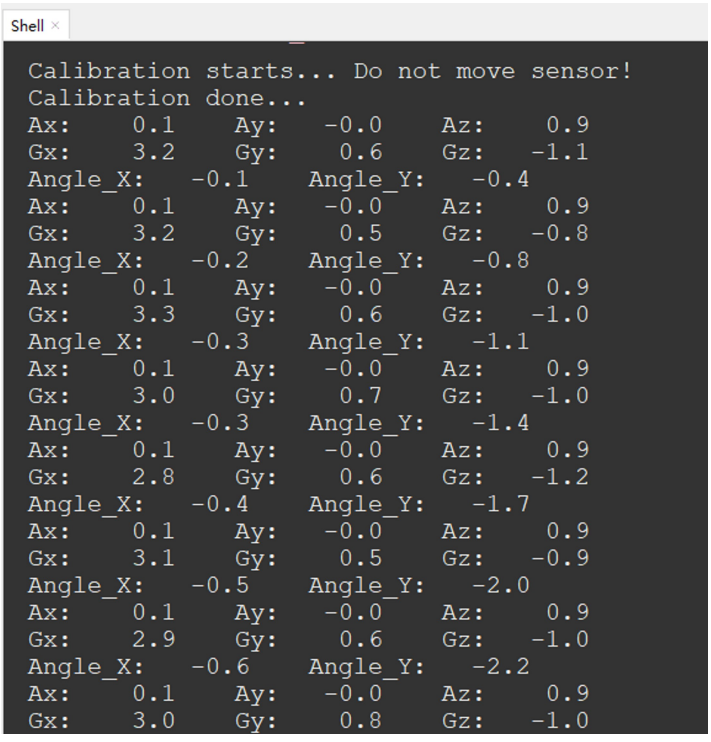
```
"""
Simple demo to read data from MPU6050

- wake up MPU6050
- read raw bytes from register
- convert raw bytes to number
- scale number to physical unit
"""

from machine import Pin, I2C
from time import sleep
import struct

i2c = I2C(0, scl=Pin(18), sda=Pin(19))

MPU_ADDR    =  0x68      # mpu6050 i2c address
PWR_MGMT    =  0x6B      # register of power management
ACCEL_ZOUT  =  0x3F      # register of acceleration along z-axis

i2c.writeto_mem(MPU_ADDR, PWR_MGMT, bytearray([0])) # wake up mpu6050

# Read acceleration along z-axis
while True:
    raw_bytes = i2c.readfrom_mem(MPU_ADDR, ACCEL_ZOUT, 2)   # read two raw bytes
    print(raw_bytes)
    print(len(raw_bytes))
    raw_accel_z = struct.unpack('>h', raw_bytes)[0]   # bytes to number: big endian, short
    accel_z = raw_accel_z / 16384    # number to acceleration (g)
    print("AccZ =", accel_z, 'g')
    print("***************")
    sleep(1)
```

使用库获取x轴/y轴转角

Mpu6050_main.py

```
"""
Calculate tilt angles using complementary filter
"""

from machine import Pin, I2C
from mpu6050 import MPU6050
from time import sleep

i2c = I2C(0, scl = Pin(18), sda = Pin(19))

mpu = MPU6050(i2c)
mpu.calibrate()

while True:
    angle_x, angle_y = mpu.get_angle_xy()   # tilt angles, deg
    print(f"Angle_X: {angle_x: 6.1f}    Angle_Y: {angle_y: 6.1f}")

    sleep(0.01)
```

练习：使用库读取三轴加速度/角速度



07 - Wifi

2022年2月5日 9:37

network模块

https://docs.micropython.org/en/latest/esp32/quickref.html#networking

```
import network

wlan = network.WLAN(network.STA_IF) # create station interface
wlan.active(True)       # activate the interface
wlan.scan()             # scan for access points
wlan.isconnected()      # check if the station is connected to an AP
wlan.connect('essid', 'password') # connect to an AP
wlan.config('mac')      # get the interface's MAC address
wlan.ifconfig()         # get the interface's IP/netmask/gw/DNS addresses

ap = network.WLAN(network.AP_IF) # create access-point interface
ap.config(essid='ESP-AP') # set the ESSID of the access point
ap.config(max_clients=10) # set how many clients can connect to the network
ap.active(True)         # activate the interface
```

wifi连接流程

```
# connect to wifi (station mode)
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect("ssid", "password")
while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())
```

socket模块

https://docs.micropython.org/en/latest/library/socket.html#module-socket

建立wifi连接后，可以使用socket进行通信，接收/发送数据。

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # 创建socket
s.bind(('', 80))  # 将ESP32 ip绑定到端口80
s.listen(5)     # 将连接数限制为5


conn, addr = s.accept()  # 等待连接
request = conn.recv(1024)  # 接收请求，最多1024字节
conn.sendall(response)   # 发送回复字符串
conn.close()    # 发送完毕，关闭连接
```

回复字符串可以是html网页代码

```
response = """<!DOCTYPE html>
    <html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
    </head>

    <body>
        <p>Hello from ESP32<p>
    </body>

    </html>"""
```

ESP32服务器与客户端通信流程

- 建立wifi连接
- 开启socket通信
- 等待连接
- 接收请求
- ESP32通过socket发送回复
- 关闭连接

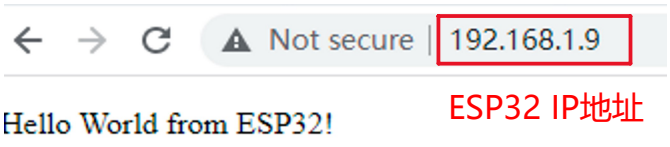
实验一：发送Hello World网页到客户端浏览器

hello_web.py

```
"""
Display 'Hello World from ESP32!' on web page
"""

try:
  import usocket as socket
except:
  import socket

import network, esp, gc

esp.osdebug(None)  # no debug info print
gc.collect()       # garbage collector

# connect to wifi (station mode)
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect("your-ssid", "your-password")  # 替换为实际的wifi名称/密码
```

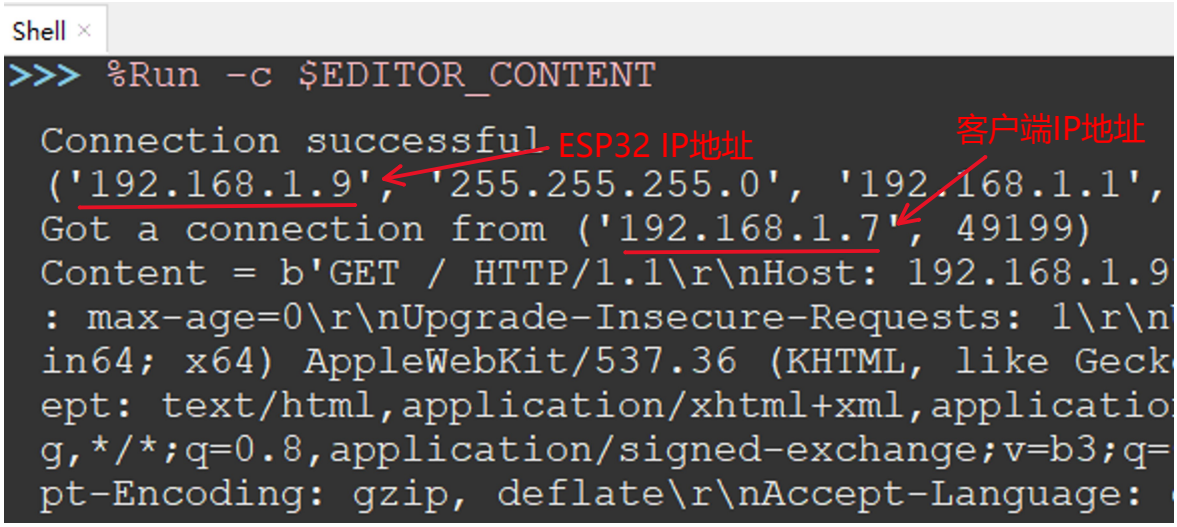
```
while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())


# open socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
    # stops at  “conn, addr = s.accept()”  waiting for a request
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    request = str(request)
    print('Content = %s' % request)

    response = """<!DOCTYPE html>
        <html lang="en">

        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-width, initial-scale=1">
        </head>

        <body>
            <p>Hello World from ESP32!</p>
        </body>

        </html>"""

    conn.send('HTTP/1.1 200 OK\n')
    conn.send('Content-Type: text/html\n')
    conn.send('Connection: close\n\n')
    conn.sendall(response)
    conn.close()
```

浏览器网页



Hello World from ESP32!

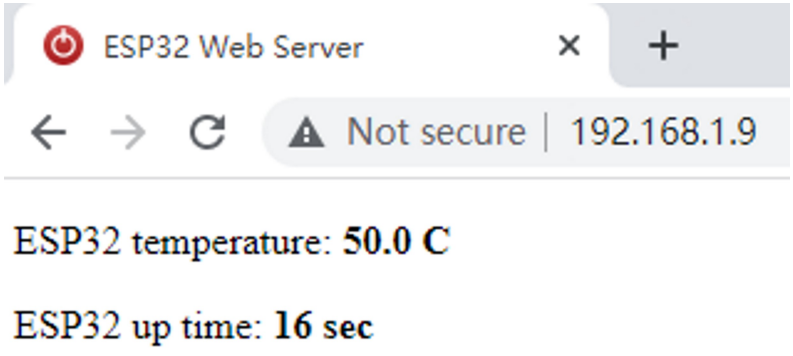
ESP32 IP地址



实验二：网页显示ESP32温度和运行时间

temp_web_main.py
temp_web_page.py (save this file to ESP32)

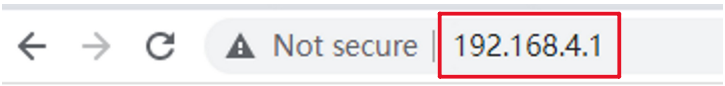
```
# 导入函数web_page()
from temp_web_page import web_page
# 读取温度
temp = esp32.raw_temperature() # Fahrenheit
temp = (temp - 32)/1.8   # Celsius
#计算运行时间
duration = time.ticks_ms() - start_time  # ms
duration = round(duration / 1000)  # sec
#将temp, duration嵌入到html代码中
response = web_page(temp, duration)
```



实验三：网页控制LED

- Save led_web_boot.py as boot.py to ESP32
- Save led_web_main.py as main.py to ESP32
- Save led_web_page.py to ESP32
- 重启ESP32（按EN键）

- ESP32 wifi 模式为access point
- 将电脑/手机连接 ESP32-AP (热点名称)/123456789 (密码)
- 可在led_web_boot.py中将wifi 模式改为station



ESP32 IP地址

Shell

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:5656
load:0x40078000,len:12696
load:0x40080400,len:4292
entry 0x400806b0
network config: ('192.168.4.1', '255.255.255.0', '192.168.4.1', '0.0.0.0')
Got a connection from ('192.168.4.2', 52992)
Content = b'GET /led_on HTTP/1.1\r\nHost: 192.168.4.1\r\nConnection: keep-al
ive\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent: Mozilla/5.0 (Windows NT 1
0.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82
Safari/537.36\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=
b3;q=0.9\r\nReferer: http://192.168.4.1/led_on\r\nAccept-Encoding: gzip, def
late\r\nAccept-Language: en-US,en;q=0.9\r\n\r\n'
LED ON
```