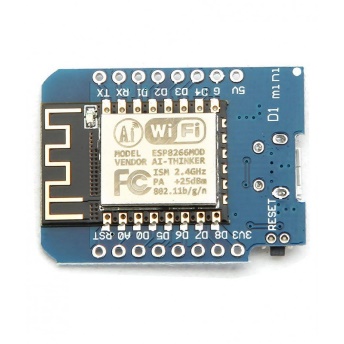
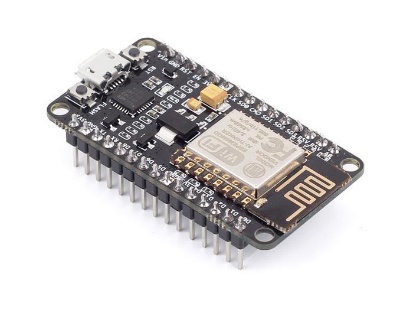
ESP8266——固件烧录与连接

【硬件准备】

* NodeMCU/WEMOS D1 mini/PiliBoard（多选一）



* USB数据线



【操作步骤】

1. 安装esptool工具
2. 下载MicroPython固件
3. 在NodeMCU/D1 mini上烧录MicroPython
4. 使用Putty进行连接
5. 在NodeMCU/D1 mini上点亮板载LED
6. 在PiliBoard上进行操作

【参考】

* MicroPython固件下载地址

<http://micropython.org/download#esp8266>

* NodeMCU通讯驱动（cp2102）

<https://cn.silabs.com/products/development-tools/software.page=0#interface>

* D1 mini/PiliBoard通讯驱动（ch340g）

<http://www.wch.cn/download/CH341SER_ZIP.html>

* 烧入固件命令

安装esptool工具（linux上加上sudo执行）：

pip install esptool

清空固件（在linux上，使用esptool.py，而不是esptool）：

esptool --port COMx erase\_flash

写入固件：

esptool --port COMx --baud 115200 write\_flash 0 esp8266-20180511-v1.9.4.bin

（部分型号的NodeMCU，需增加命令参数 -fm dio）

* 点亮LED灯

import machine

p2 = machine.Pin(2,machine.Pin.OUT)

p2.value(1) # 熄灭

p2.value(0) # 点亮

* MicroPython官网

<http://www.micropython.org/>

* PiliBoard与MicroPython

<https://www.hachina.io/docs/6807.html>

ESP8266上的MicroPython使用

【操作步骤】

1. REPL
2. 文件系统
3. 连接wifi网络
4. WebREPL
5. 常用的REPL操作技巧
6. 8266上的AP网卡

【参考】

* 文件系统常用命令

import os

os.statvfs('/') # 查看文件系统状态

os.listdir() # 列出当前目录下文件

os.mkdir('xxxx') # 创建目录

os.rmdir('xxxx') # 删除目录

os.stat('test.py') # 列出文件状态

os.rename('test.py','test.py.bak') # 修改文件名

os.remove('test.py.bak') # 删除文件

# 打印文件内容

f=open('boot.py','r')

f.read()

f.close()

* 网卡操作常用命令

import network

# 工作站网卡

sta\_if = network.WLAN(network.STA\_IF) # 设置工作站网卡sta\_if

sta\_if.active(True) # 开启网卡

sta\_if.connect('xxx','yyy') # 连接wifi网络，名称为xxx，密码为yyy

sta\_if.active() # 查询网卡是否激活

sta\_if.isconnected() # 查询网络是否连接

sta\_if.ifconfig() # 查询网络连接信息

sta\_if.active(False) # 关闭网卡

# AP热点

ap\_if = network.WLAN(network.AP\_IF) # 设置热点网卡ap\_if

ap\_if.config(essid="abc", password="12345678") # 设置热点与密码

ap\_if.config("essid") # 查看设置的热点

# active()与ifconfig()，用法与工作站网卡相同

* WEBRepl

WEBRepl开启与设置命令：import webrepl\_setup

WEBRepl页面：<http://micropython.org/webrepl/>

* 闪烁LED灯

import machine

import time

p2 = machine.Pin(2, machine.Pin.OUT)

while True:

print("flashing……")

p2.value(0)

time.sleep(1)

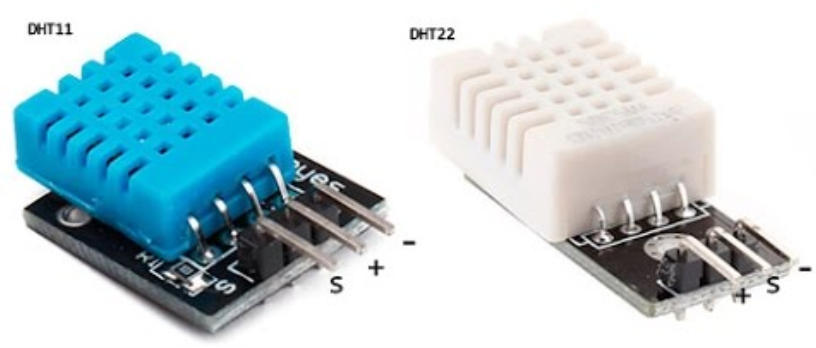
p2.value(1)

time.sleep(1)

连接ESP8266的DHT温湿度传感器

【硬件准备】

温湿度传感器：dht11/dht22

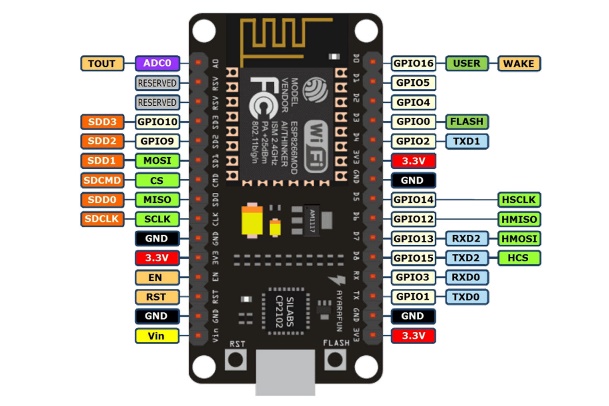
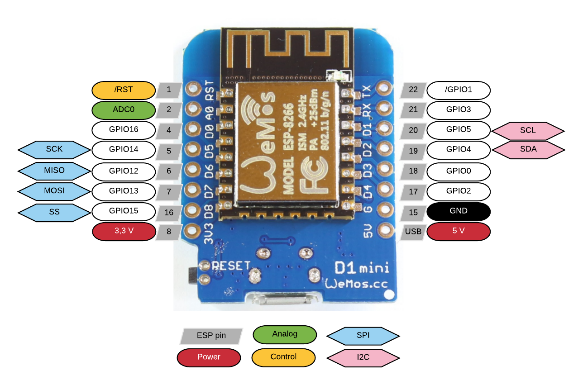


【操作步骤】

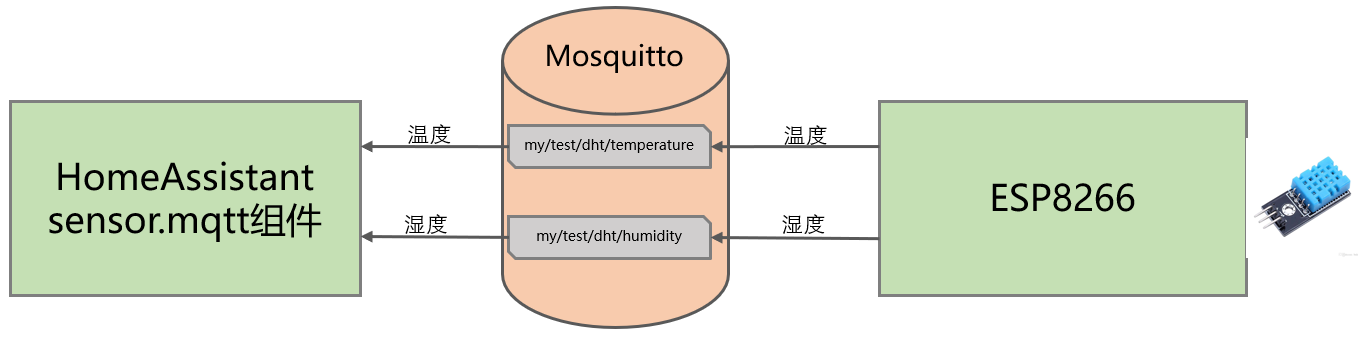
1. 硬件连接
2. 读取温湿度
3. 向MQTT服务器发送温度值
4. HomeAssistant中的配置
5. 8266中的自启动运行

【参考】

* 开发板管脚

* mqtt连接逻辑



* 程序

import dht

import machine

import time

from ubinascii import hexlify

from umqtt.simple import MQTTClient

# 初始化一个温湿度传感器, 温湿度传感器连接4号GPIO

DHT = dht.DHT11(machine.Pin(4))

# CLIENT\_ID: 每个mqtt客户端有自己独立的client\_id标识

# 无所谓是什么，但不同客户端不能相同。此处我们使用8266的unique\_id

client\_id = hexlify(machine.unique\_id()).decode()

mqtt\_broker = "192.168.31.193"

mqtt\_user = 'pi'

mqtt\_password = 'hachina'

# 温度和湿度在mqtt服务器上的主题位置

TEMPERATURE\_TOPIC = "my/test/dht/temperature"

HUMIDITY\_TOPIC = "my/test/dht/humidity"

# 等待15秒，wifi自动连接上再运行

time.sleep(15)

mqtt = MQTTClient( client\_id, mqtt\_broker, 1883, mqtt\_user, mqtt\_password )

mqtt.connect()

print("连接到服务器：{s}".format(s=mqtt\_broker))

while True:

# 不断循环，每10秒测量一次温度和湿度，发布到mqtt服务器上

try:

DHT.measure()

except:

print("No dht sensor connected to Pin(%d)"%(PIN\_NO))

break

print("测量到温度：%d；湿度：%d"%(DHT.temperature(),DHT.humidity()))

# mqtt发布信息

mqtt.publish( TEMPERATURE\_TOPIC, str(DHT.temperature()).encode(), retain=True)

mqtt.publish( HUMIDITY\_TOPIC, str(DHT.humidity()).encode(), retain=True)

time.sleep(10)

* HomeAssistant中配置

sensor mqtt:

- platform: mqtt

name: "dht-T"

state\_topic: "my/test/dht/temperature"

unit\_of\_measurement: "°C"

- platform: mqtt

name: "dht-H"

state\_topic: "my/test/dht/humidity"

unit\_of\_measurement: "%"

ESP8266完成各种功能

【操作步骤】

1. 从INTERNET同步时间
2. WEB服务器
3. PWM舵机控制
4. ds18b20温度传感器
5. tsl2561光照传感器
6. ws2812灯带控制

【参考】

* ESP8266功能与常用应用领域

|  |  |
| --- | --- |
| 功能 | 常用应用领域 |
| 网络 | NTP客户端、MQTT客户端、WEB服务器端/客户端 |
| 常规GPIO输入输出 | 状态量读取与输出、继电器控制 |
| GPIO PWM | 可调亮度LED、可控硅控制、舵机控制 |
| 1-wire总线通讯 | 连接ds18b20等各种1-wire总线器件 |
| I2C总线通讯 | 连接am2320等各种i2c总线器件 |
| SPI总线通讯 | 连接各种SPI总线器件 |
| ADC模拟信号输入 | 测量输入电压、模拟传感器连接 |
| apa102、neopixel驱动 | 各种灯带控制 |
| 中断 | 按钮输入 |
| 计时器 | 定时控制 |
| …… | …… |

* MicroPython参考文档网站

<http://docs.micropython.org/en/latest/>

* PiliBoard各种用例与程序

<https://github.com/zhujisheng/piliboard/tree/master/examples>

* NTP访问

import ntptime

import time

ntptime.settime()

t = time.localtime()

print("当前国际标准时间：%d年%d月%d日 %d:%d:%d"%(t[0],t[1],t[2],t[3],t[4],t[5]))

* 一个最简单的WEB服务器

response = """<!DOCTYPE html>

<html>

<head> <title>ESP 8266</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

</head>

<body><h1>欢迎来到MicroPython的世界</h1></body>

</html>

"""

import socket

addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]

s = socket.socket()

s.bind(addr)

s.listen(1)

print('listening on', addr)

while True:

cl, addr = s.accept()

print('client connected from', addr)

cl\_file = cl.makefile('rwb', 0)

while True:

line = cl\_file.readline()

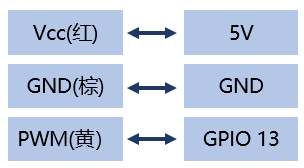
if not line or line == b'\r\n':

break

cl.send(response)

cl.close()

* PWM舵机控制



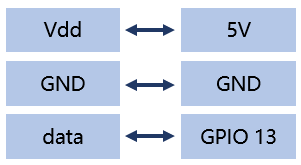
import machine

pin\_no = 13

servo = machine.PWM(machine.Pin(pin\_no), freq=50)

servo.duty(30)

* DS18b20温度传感器



import time

import machine

import onewire, ds18x20

dat = machine.Pin(13)

ds = ds18x20.DS18X20(onewire.OneWire(dat))

roms = ds.scan()

print('找到设备：', roms)

# 循环10次，每次打印所有设备测量的温度

while True:

print('温度：', end=' ')

ds.convert\_temp()

time.sleep\_ms(750)

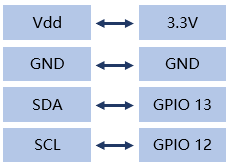
for rom in roms:

print(ds.read\_temp(rom), end=' ')

print()

* tsl2561光照传感器

<https://github.com/adafruit/micropython-adafruit-tsl2561/blob/master/docs/tsl2561.rst>



import tsl2561

import time

from machine import I2C, Pin

i2c = I2C(scl=Pin(12), sda=Pin(13))

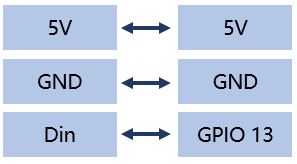
sensor = tsl2561.TSL2561(i2c)

while True:

print(sensor.read())

time.sleep\_ms(750)

* ws2812灯带控制



from machine import Pin

from neopixel import NeoPixel

leds\_num = 30

dp = Pin(13, Pin.OUT)

np = NeoPixel(dp, leds\_num)

for i in range(leds\_num):

np[i] = (0, 0, 0)

np.write()

# 按红、绿、蓝、白显示灯带

color = [(255,0,0),(0,255,0),(0,0,255),(255,255,255)] #红、绿、蓝、白

for i in range(0,np.n):

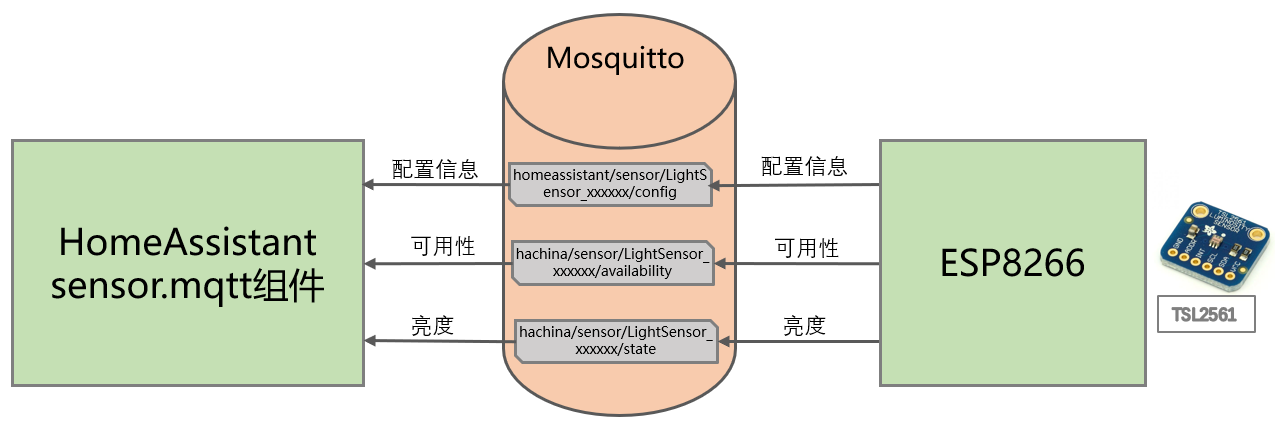
np[i] = color[i%4]

np.write()

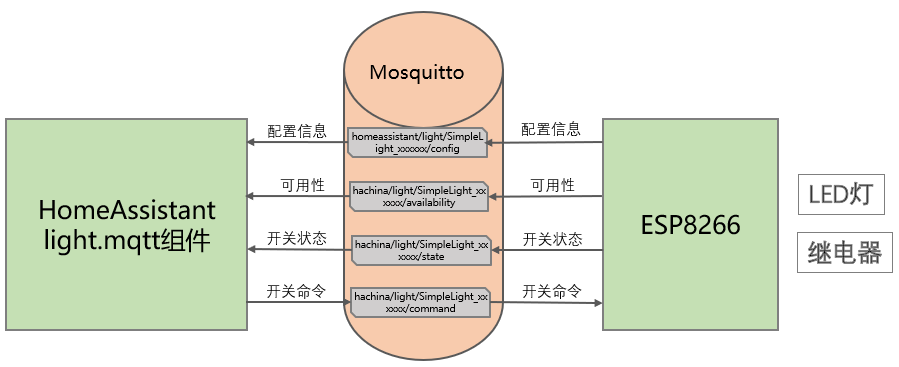
典型样例讲解：光照传感器与智能灯

【操作步骤】

1. 光照传感器

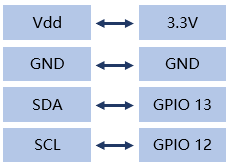


1. 智能灯



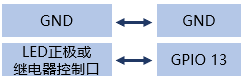
【参考】

* 光照传感器：样例example\_17\_5\_1.zip



|  |  |  |
| --- | --- | --- |
| 文件 | 说明 | 备注 |
| config.json | wifi和mqtt的配置信息 | 你需要修改为你自己的环境 |
| main.py | 启动时自动运行的文件 | 也可以不上传这个文件，手工运行其中命令 |
| tsl2561.py | tsl2561驱动程序 | 如果使用其它传感器，不需要此文件 |
| sensor\_tsl2561.py | 主程序 | 可以修改GPIO口为你实际的连接口；连接其它传感器，修改对应硬件操作部分程序 |

* 智能灯：样例example\_17\_5\_2.zip



|  |  |  |
| --- | --- | --- |
| 文件 | 说明 | 备注 |
| config.json | 同上 | |
| main.py | 同上 | |
| StateMQTTClient | 基于umqtt.simple的一个自己实现的mqtt类 | 其中增加了对ping返回的判断，并增加了是否连接的状态 |
| light\_gpio.py | 主程序 | 可以修改GPIO口为你实际的连接口 |

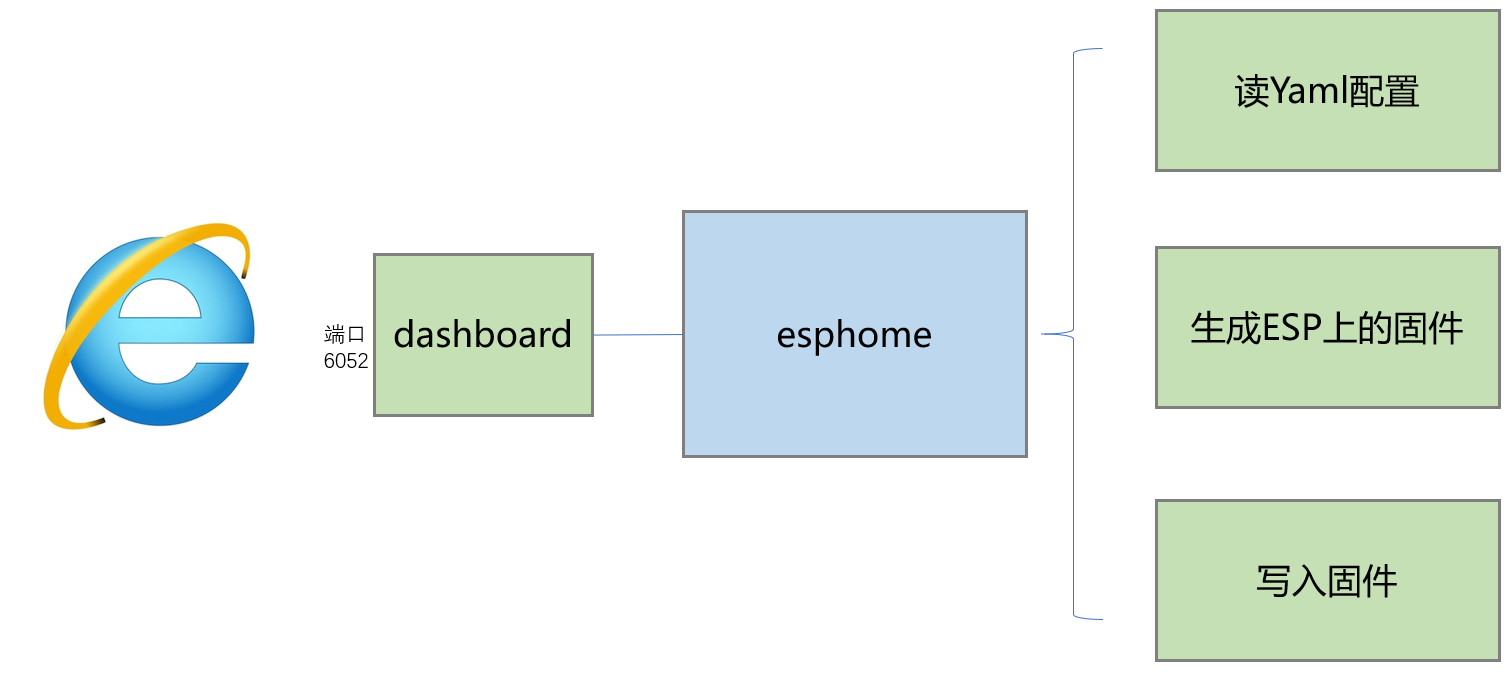
ESPHome——不编程，集成ESP8266

【操作步骤】

1. 安装esphome
2. 打开web配置界面
3. 配置与上传
4. 配置一些常见设备

【参考】

* ESPHome的功能



* ESPHome官网

<https://esphomelib.com/>

* 相关命令

安装：sudo pip2 install esphome

打开web端配置：esphome esphome\_config/ dashboard

注：如果在安装esphome时出现错误：*tornado requires Python '>= 3.5' but the running Python is 2.7.13*，请先使用sudo pip2 install tornado==5.1.1命令安装tornado，然后再安装esphome

* ESPHome组件配置

<https://esphome.io/index.html#core-components>

* 板载GPIO2号口LED的配置

output:

- platform: esp8266\_pwm

pin: GPIO2

frequency: 200 Hz

id: led\_onboard

inverted: True

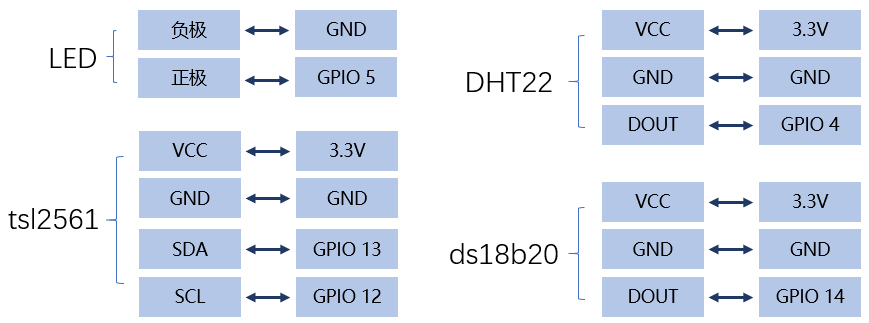
light:

- platform: monochromatic

name: "LED On NodeMCU Board"

output: led\_onboard

* 连接多个外围设备的配置样例



output:

# 输出，板载LED

- platform: esp8266\_pwm

pin: GPIO2

frequency: 200 Hz

id: led\_onboard

inverted: True

# 输出，外接LED

- platform: gpio

pin: GPIO5

id: output\_gpio5

# ds18b20

dallas:

- pin: GPIO14

i2c:

sda: 13

scl: 12

light:

# 板载LED

- platform: monochromatic

name: "LED On NodeMCU Board"

output: led\_onboard

# 外接LED

- platform: binary

name: "LED Link to GPIO5"

output: output\_gpio5

sensor:

# DHT22温湿度传感器

- platform: dht

pin: GPIO4

model: DHT22

temperature:

name: "Temperature DHT22"

humidity:

name: "Humidity DHT22"

update\_interval: 30s

# DS18b20温度传感器

- platform: dallas

index: 0

name: "Temperature ds18b20"

# tsl2561光亮度传感器

- platform: tsl2561

name: "Lux TSL2561"

update\_interval: 30s