软件应用指南

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.

目录

免责申明和版权公告

本文中的信息，如有变更，恕不另行通知。 文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反 言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

文中所得测试数据均为亿佰特实验室测试所得，实际结果可能略有差异。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

最终解释权归成都亿佰特电子科技有限公司所有。

注 意 ：

由于产品版本升级或其他原因，本手册内容有可能变更。亿佰特电子科技有限公司保留在没有任何通知或者提示的情况下对本手册的内容进行修改的权利。本手册仅作为使用指导，成都亿佰特电子科技有限公司尽全力在本手册中提供准确的信息，但是成都亿佰特电子科技有限公司并不确保手册内容完全没有错误，本手册中的所有陈述、信息和建议也不构成任何明示或暗示的担保。

1. 概述

软件评估指南用于介绍在亿佰特的开发板上运行系统下的核心资源与外设资源的测试步骤与评估方法。本文可作为前期评估指导使用，也可以作为通用系统开发的测试指导书使用。

1.1.软件资源

ECM50 基于 ESP32-S3 芯片设计的全场景通信工业级可编程工业计算机，支持 4G、以太网，WIFI，蓝牙 4 种通信方式，支持 TCP、HTTP、MQTT 等多种网络协议，用户可以通过 MicroPython 语言编程，在 RTU 端执行定制化的业务逻辑与数据处理；适用于各种物联网数据采集、传输、控制场景，使用官方提供的丰富案例源码快速开发自己的业务功能。大大降低开发周期，节省项目开发成本。

ECM50 搭载 MicroPython 的操作系统，整机出厂附带操作系统，用户可以使用提供的例程来运行一些代码做测试。包含 LED，GPIO，DI，DO，AI，ETH，4G，Lora，WIFI，BLE，232/485 串口和 SD 卡。

2. 快速开始

2.1.开发软件安装

ECM50 出厂自带 micropython 固件，用户到手之后可以直接开始测试，如果需要重新烧录固件，请参考 2.4 小节。

我们使用 micropython 官方推荐的 Thonny Python IDE，Thonny IDE 是一款开源软件，以极简方式设计，对 MicroPython 的兼容性非常友善。而且支持 Windows、MacOS、Linux、树莓派。

安装 CH340 驱动：03_Tools/ CH340 驱动.rar

Thonny 软件安装包：03_Tools/ thonny-4.1.7.exe。

安装完成后打开如下所示。

2.2.接线和启动

使用 TYPE-C 数据线连接 ECM50 此面的接口，此接口兼具供电和串口的功能。如下图所示。

如果需要长期稳定的供电，或者需要使用 4G 等用电需求大的外设时，需要连接 DC 供电，位置在连接端子的最左侧，丝印为 DC8~28V。

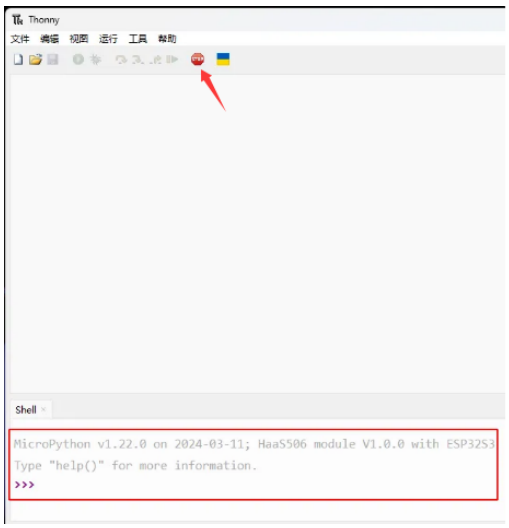
按照第一种方式之后，ECM50 就会自动启动了。

2.3.点亮 LED

在保证设备供电和串口连接的状态下，打开 Thonny 软件，点击运行，配置解释器。

选择 ESP32 解释器，端口选择自动探测，点击确认。

点击重启终端，如果出现下面 micropython 的提示符则说明软件和设备连接成功。



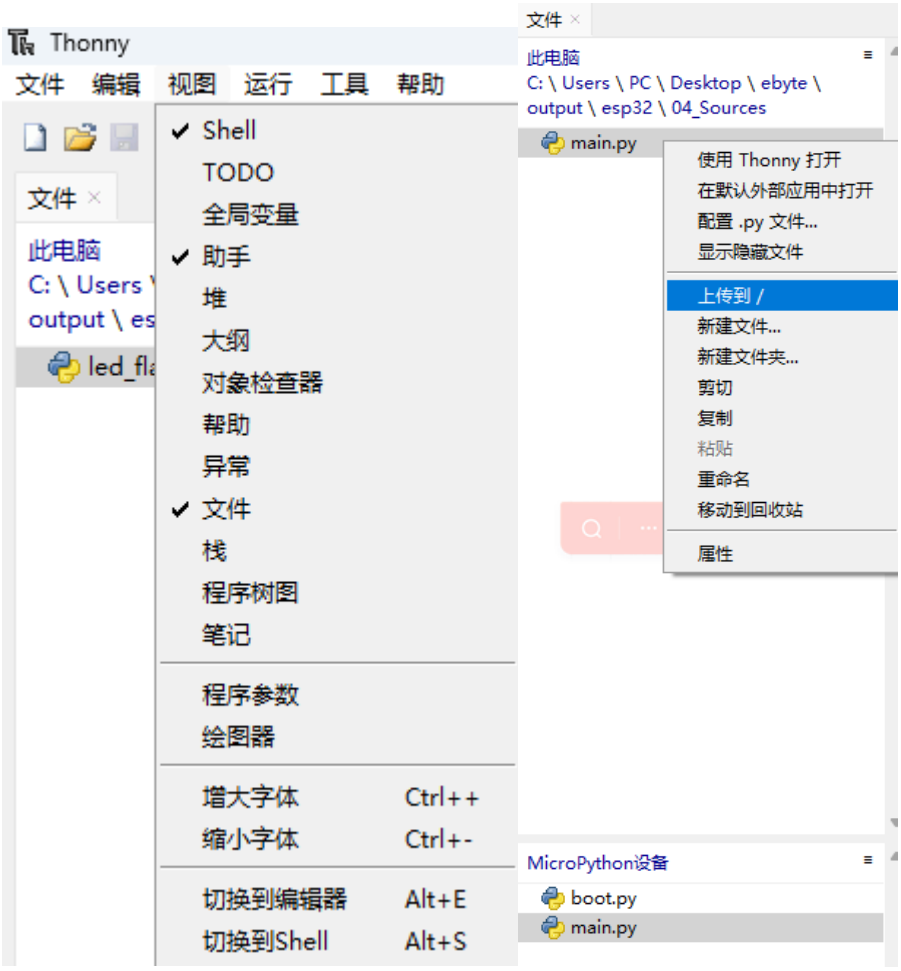
点击左上角文件→打开，选择此电脑，然后打开 04_Sources/led_flash.py。

然后点击运行。

可以看到板子的 LED 在闪烁，到这里就大功告成了。

如果想开机自动启动，则需要将此程序重命名为 main.py，然后上传到工业计算机：

1. 打开视图→文件
2. 右键点击文件，然后选择上传到 / 。
3. 重启即可开机自动闪烁 LED 灯。

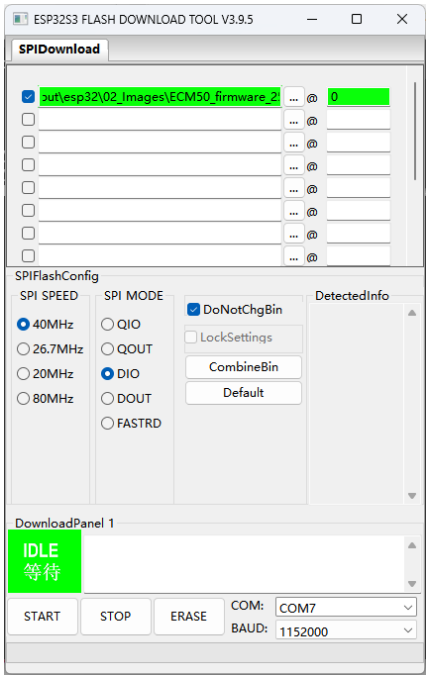


2.4.烧录固件

使用 typc 数据线连接我们 ECM50 的 typec 接口。

需要使用到 Tools 目录下的 flash_download_tool_3.9.5，解压之后直接打开 flash_download_tool_3.9.5.exe，选择 ESP32-S3。

选择 Images 目录下的 ECM50 固件。然后再选择下面的串口，波特率直接选择 1152000。点击 START 开始烧录。烧录完成之后退出，按下复位或者重新上电即可开始测试。

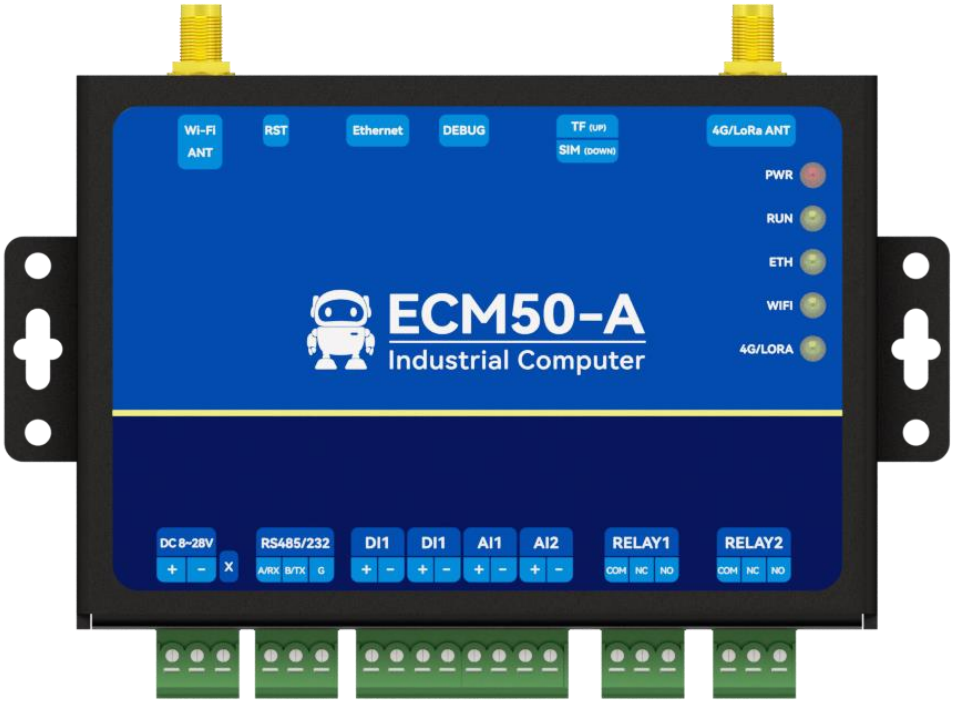


3. 功能测试

我们将对工业计算机的每一项功能使用 micropython 的代码来测试，用户只需要打开 04_Sources 的对应 python 文件，然后点击运行即可，和上面测试 LED 的方法一样。

3.1.外设接口

3.1.1. LED



ECM50 一共有五个灯，分别是 4G/Lora，WIFI，ETH，RUN，PWR。

PWR 是通电后亮起。

WIFI 灯和 RUN 灯是用户可以用 IO 来编程控制的灯。

4G/Lora 灯是用户在使用 4G 或者 Lora 时会自动闪烁的灯，无法用 IO 控制。

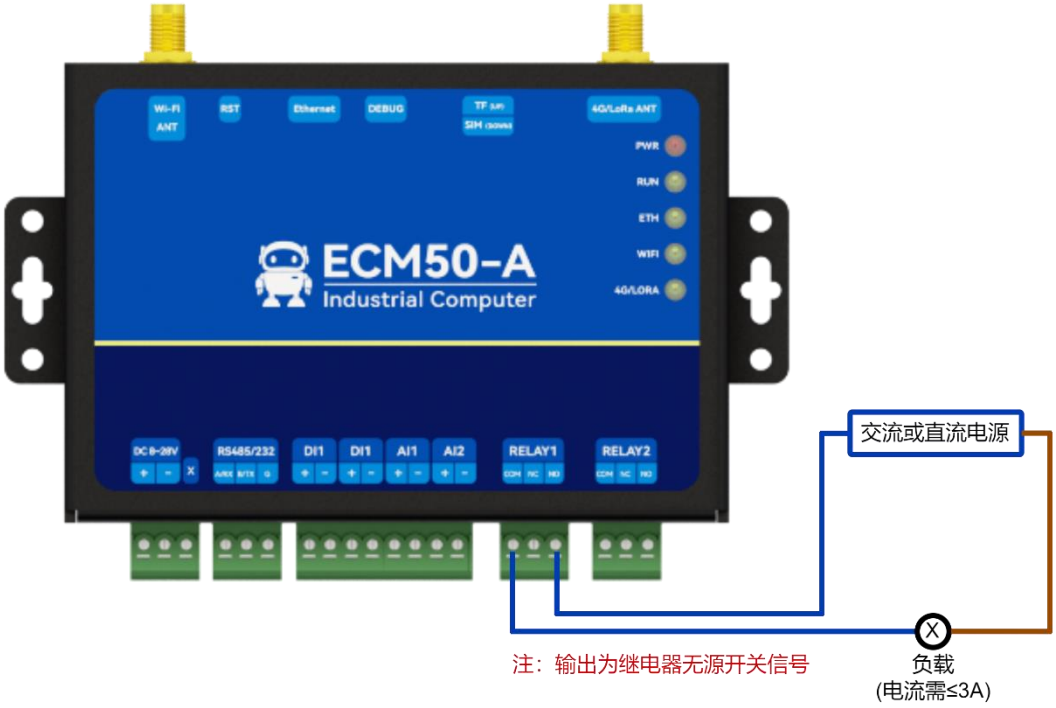
ETH 是用户在使用以太网时会自动闪烁的灯，无法用 IO 控制。

下面的代码我们控制 WIFI 灯和 RUN 灯的闪烁。

```
import time
from machine import Pin


def main():
    # RTU 一共 2 个可控指示灯
    led_wifi = Pin(45, Pin.OUT) # LED2 on GPIO 39 wifi
    led_sta = Pin(21, Pin.OUT) # LED4 on GPIO 21  运行
    while True:
        led_wifi.value(1)
        led_sta.value(1)
        print("led high")
        time.sleep(1)
        led_wifi.value(0)
        led_sta.value(0)
        print("led low")
        time.sleep(1)


#程序启动入口
if __name__ == '__main__':
    main()
```

3.1.2. DO

ECM50 有 2 路 10A/277VACNC+NO。都是使用 IO 来控制打开或者关闭通路。



如上图所示，用户可以在一个回路中接上电源和用电器，那么直接控制 DO 信号就可以来控制改回路的通断。

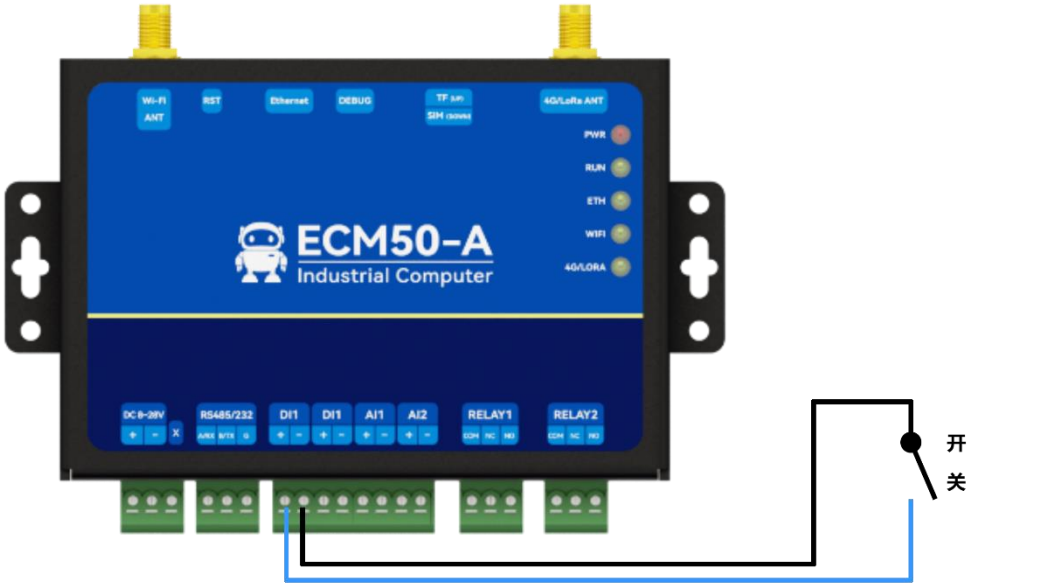
```
import time
import machine


def main():
    # RTU 一共 2 个 DO
    DO1 = machine.Pin(15, machine.Pin.OUT)
    DO2 = machine.Pin(16, machine.Pin.OUT)
    while True:
        DO1.value(1)
        time.sleep(0.5)
        DO2.value(1)
        print("DO output high")
        time.sleep(1)
        DO1.value(0)
        time.sleep(0.5)
        DO2.value(0)
        print("DO output low")
        time.sleep(1)


#程序启动入口
if __name__ == '__main__':
    main()
```

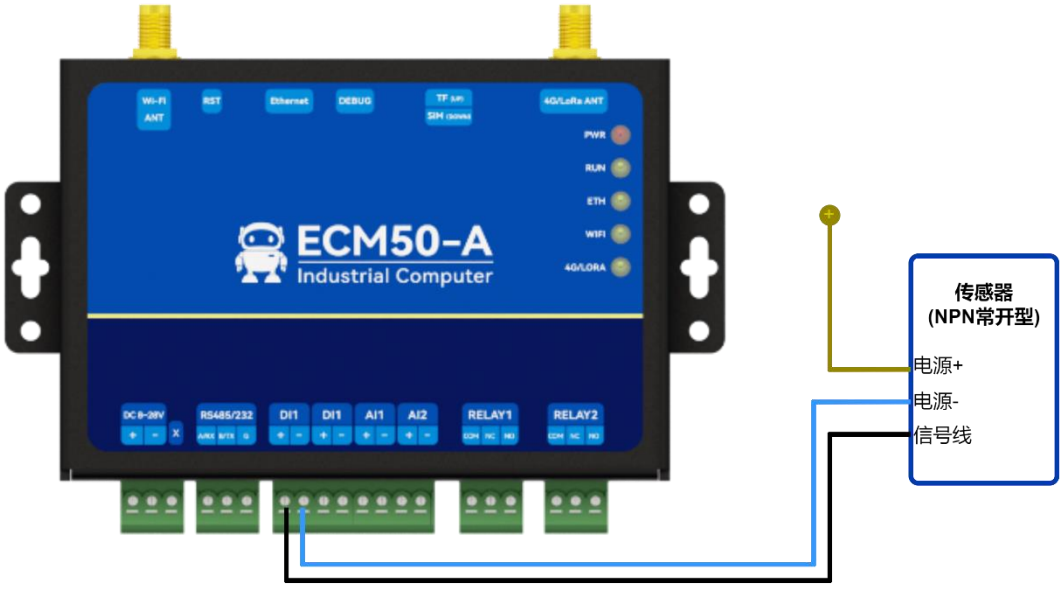
3.1.3. DI

ECM50 有 2 路默认 NPN 型干接点，可自行改湿节点，我们还是使用 IO 设置为 input，并设置内部下拉，来读取 IO 的值。

图一，如果开关闭合，那么 ECM50 测得值为低，断开，测得值就为高。

图二是传感器来控制信号线，信号线决定了测得值的高低。


ECM50-A
Industrial Computer
开
关

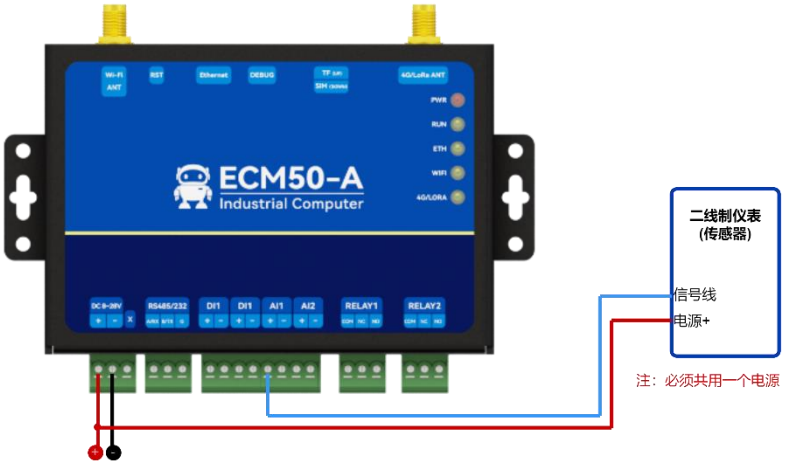

ECM50-A
Industrial Computer
传感器
(NPN常开型)
电源+
电源-
信号线

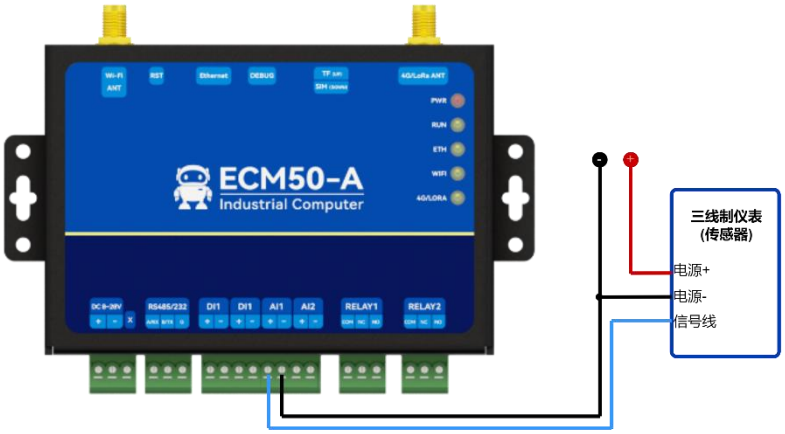
```
import time
import machine


def main():
    DI1 = machine.Pin(4, machine.Pin.IN,machine.Pin.PULL_DOWN)
    DI2 = machine.Pin(5, machine.Pin.IN,machine.Pin.PULL_DOWN)
    while True:
        di1_value = DI1.value()
        di2_value = DI2.value()
        print(f'[1]di1_value = {di1_value}, di2_value = {di2_value}')
        time.sleep(1)


if __name__ == '__main__':
    main()
```

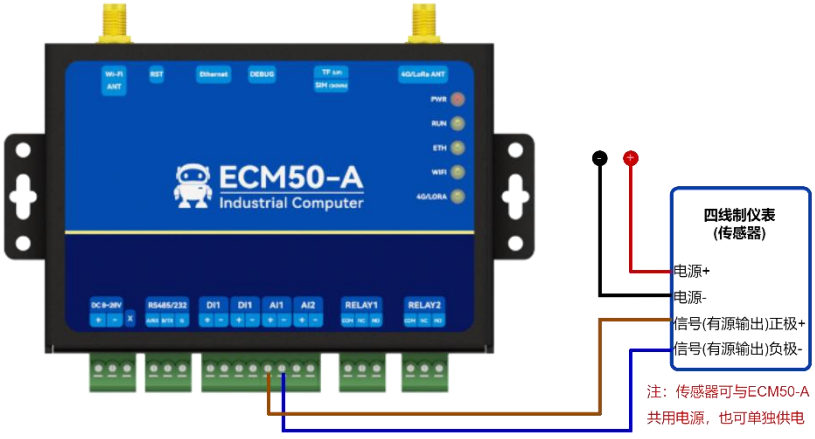
3.1.4. AI

ECM50 有 2 路默认 4-20mA 电流采集，可自行改电压采集（精度 12 位）。这里实际上读取的值还是电压，经过换算得到了电流，使用电流发生器来测试。

AI 有下面几种接线方法，供参考。


ECM50-A
Industrial Computer
二线制仪表
(传感器)
信号线
电源+
注：必须共用一个电源


ECM50-A
Industrial Computer
三线制仪表
(传感器)
电源+
电源-
信号线

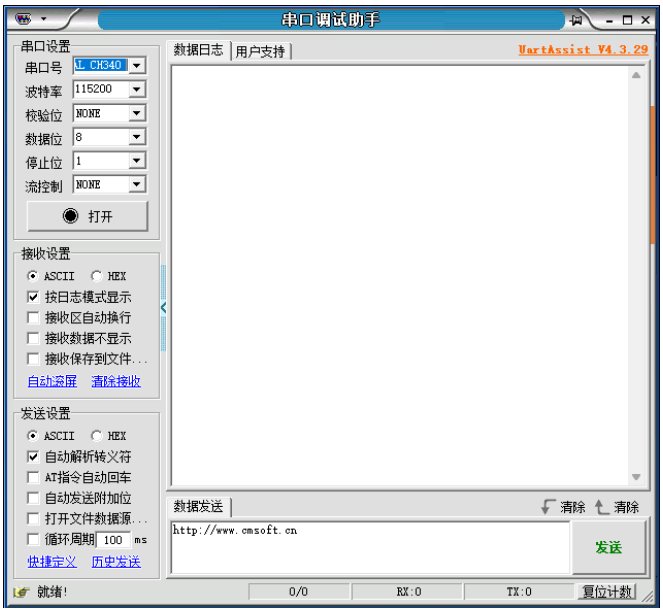

ECM50-A
Industrial Computer
四线制仪表
(传感器)
电源+
电源-
信号(有源输出)正极+
信号(有源输出)负极-
注：传感器可与ECM50-A
共用电源，也可单独供电

```
# ai 测试案例
# ai1-gpio1,ai2-gpio11  电流测量范围 4-20ma
from machine import ADC
import time
import machine


def main():
    ai1 = ADC(machine.Pin(6)) #ai1
    ai1.atten(ADC.ATTN_11DB)  #使用 0 衰减，精度最高，输入电压范围：0-950mv
    ai2 = ADC(machine.Pin(7)) #ai2
    ai2.atten(ADC.ATTN_11DB)
    while True:
        val_u16_1 = ai1.read_u16()  # read a raw analog value in the range 0-65535
        val_uv_1 = ai1.read_uv()  # read an analog value in microvolts
        val_raw1 = ai1.read() #获取原始 ADC 值
        print(f'val_u16_1 = {val_u16_1}, val_uv_1 = {val_uv_1}',f'val_raw1 = {val_raw1}',f'ai1 = {val_uv_1/150/1000}ma')
        time.sleep(0.5)
        val_u16_2 = ai2.read_u16()  # read a raw analog value in the range 0-65535
        val_uv_2 = ai2.read_uv()  # read an analog value in microvolts
        value2 = ai2.read() #获取原始 ADC 值
        print(f'val_u16_2 = {val_u16_2}, val_uv_2 = {val_uv_2}',f'val_raw2 = {val_raw2}',f'ai2 = {val_uv_2/150/1000}ma')
        time.sleep(0.5)


if __name__ == '__main__':
    main()
```

3.1.5. 串口

ECM50 的串口 1 要么挂载 RS232，要么挂载 RS485，不同型号搭载不同通信。我们找到 ECM50 的 RS485/232 端口，使用 USB 转 485（自动收发）或者 USB 转 232 连接 PC 来进行测试。

打开 03_Tools/UartAssist.exe，打开 USB 串口，选择 115200 波特率，无校验，数据为 8，停止位 1，无流控制。点击打开串口。



在使用 Thonny 打开 04_Sources/RS485.py，点击运行。

```
import machine
import utime
from machine import Pin


def main():
    uart = machine.UART(1, baudrate=115200, tx=17, rx=18)
    while True:
        if uart.any():
            # 读取串口数据
            data = uart.read()


            # 打印接收到的数据
            print("Received data:", data)
        # 暂停 100 毫秒
        utime.sleep_ms(100)
        uart.write('test')
if __name__ == '__main__':
    main()
```

即可看到串口数据的正常收发，232 的测试代码一样，只需要更换为 USB 转 232 连接 PC 即可。

3.1.6. SD 卡

ECM50 在 TF 卡槽插入一张 SD 卡，可以通过 SPI 来读写 SD 卡的文件，注意 SD 卡需要被格式化为 FAT32 格式。

我们的代码中挂载了 SD 卡，尝试打开了一个 test.txt 并写入一行数据，然后再读取这个数据，最后卸载掉 SD 卡。

```
import machine
import os
import utime
def main():
    # 初始化 SD 卡
    sd = machine.SDCard(slot=2)
    # 挂载 SD 卡
    try:
        os.mount(sd, "/sd")
        print("SD card mounted.")
    except OSError as e:
        print("Error mounting SD card:", e)
    ret = os.listdir('/sd')
    print("SD card:", ret)
    #写入文件
    try:
        with open("/sd/test.txt", "w") as f:
            f.write("Hello, SD Card!")
        print("File written to SD card.")
        f.close()
    except OSError as e:
        print("Error writing to file:", e)
    ret = os.listdir('/sd')
    print("SD card:", ret)
    utime.sleep(1)
    # 读取文件
    try:
        with open("/sd/test.txt", "r") as f:
```

```
                content = f.read()
            print("File content:", content)
        except OSError as e:
            print("Error reading from file:", e)

        # 卸载 SD 卡
        try:
            os.umount("/sd")
            print("SD card unmounted.")
        except OSError as e:
            print("Error unmounting SD card:", e)

if __name__ == '__main__':
    main()
```

3.2.网络接口

3.2.1. Ethernet

以太网采用 SPI 转网芯片 W5500，RJ4510/100Mbps，连接至 SPI2 接口上。

将网线插入 ECM50 和路由器，我们来测试一下 ECM50 自动获取 IP 地址，并且和同样连接到路由器的 PC 进行 TCP 通信收发测试。

PC 打开 03_Tools/netassist5.15.exe。

选择 TCP server，将主机地址选择为为 PC 的 IP 地址，这里是 192.168.20.5，端口号设置为 8080，最后单击打开。

ECM50 这边我们直接运行 eth.py 即可，我们在代码中指定了 TCP 服务端的 IP 地址和端口号，用户需要根据自己的实际情况来进行修改方能正常进行 TCP 通信。

```
import network
import machine
import time

def main():
    # 初始化 SPI 和引脚
    spi = machine.SPI(1,
                      baudrate=20_000_000,  # 20MHz 时钟
                      sck=machine.Pin(12),
                      mosi=machine.Pin(11),
                      miso=machine.Pin(13))
    cs_pin = machine.Pin(47, machine.Pin.OUT)  # 片选引脚
    reset_pin = machine.Pin(48, machine.Pin.OUT)
    int_pin = machine.Pin(14, machine.Pin.IN)  # 中断引脚（可选）

    # 创建 LAN 接口
    lan = network.LAN(
        phy_type=network.PHY_W5500,  # 指定 W5500 驱动
        reset=reset_pin,
        spi=spi,                          # SPI 总线对象
        cs=cs_pin,                        # 片选引脚
        int=int_pin,                      # 中断引脚（可省略）
        phy_addr=0,                       # PHY 地址（SPI 模式下通常为 0）
        ref_clk_mode=1,                   # 参考时钟模式（0=输入，1=输出）
        ref_clk=25                        # 参考时钟引脚（根据硬件设计）
    )
    print(spi)
    # 激活接口
    lan.active(True)

    # 等待连接（最多等待 10 秒）
    start_time = time.time()
    while not lan.isconnected():
        if time.time() - start_time > 10:
            print("连接超时！")
            break
        time.sleep_ms(500)
        print("等待连接...")
    # 打印网络配置
    if lan.isconnected():
        print("连接成功！")
        print("IP 地址:", lan.ifconfig()[0])
        print("MAC 地址:", ":".join("{:02x}".format(b) for b in lan.config("mac")))
        do_tcp_client() #tcp 测试
```

```
def do_tcp_client():
    import socket
    import time

    # 服务器的 IP 地址和端口
    server_ip = '192.168.20.5'
    server_port = 8080   # 选择一个合适的端口号

    # 创建 socket 对象
    addr_info = socket.getaddrinfo(server_ip,server_port)
    addr = addr_info[0][-1]
    # 连接到服务器
    s = socket.socket()
    s.connect(addr)
    # 发送数据
    try:
        while True:
            print('send data')
            # 发送数据
            s.send(b'GET / HTTP/1.1\r\nHost: micropython.org\r\n\r\n')
            data = s.recv(1000)
            print(f"Received: {data.decode('utf-8')}")
            # 每隔 1 秒发送一次数据
            time.sleep(1)


    except KeyboardInterrupt:
        print("Client stopped by user.")
    finally:
        # 关闭连接
        client_socket.close()

 #程序启动入口
if __name__ == '__main__':
    main()
```

3.2.2. 4G 模块

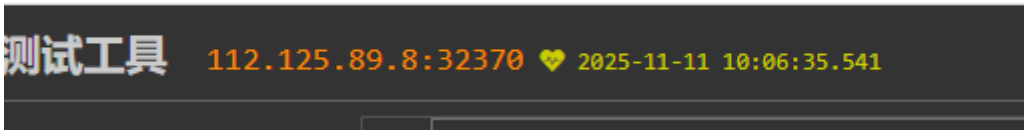
ECM50 的 Lora 模块和 4G 模块是二选一的，不同型号搭载不同通信。4G 模块采用中移 EC801E-CN 模块，使用串口来进行数据交互。我们需要一张 4G 卡，放入 4G 卡槽中。

我们 4G 模块的测试方法是使用 AT 指令去测试的 TCP 服务，用户如果有 MQTT 或者 UDP 等其他需求，可以自行查看移远相关手册，按照我们的教程去操作即可。

我们使用一个 TCP 测试工具，相当于使用 ECM50 的 4G 模块的 TCP 功能和工具建立连接，并发送信息。

https://netlab.luatos.com/

选择打开 TCP，就会看到一个 IP 地址和一个端口号，这里我们看到的 IP 是 112.125.89.8，端口是 32370。



将上面获取到的 IP 地址填入到 4g.py 的做 TCP 测试的函数 do_tcp_test 中，上电后执行 4G 测试代码：4g.py。注意 4G 模块耗电量比较大，需要另外接入 DC 电源到 ECM50 的 DC8~28V 才能正常工作。

```
64  def do_tcp_test(uart):
65      host = "112.125.89.8"
66      port = "32370"
67      cmd = 'AT+QIOPEN=1,0,"TCP","%s",%s,0,2' % (host, port)
68      resp = send_at(uart, cmd, 3)  # 增加超时时间，等待连接建立
69      print("QIOPEN response:", resp)
70
71      # 等待连接建立
72      utime.sleep(2)
73
```

```
Shell
Response: b'AT+CPIN?\r\r\n+CPIN: READY\r\n\r\nOK\r\n'
Sending Set APN: AT+CGDCONT=1,"IP","CMNET"
Send: AT+CGDCONT=1,"IP","CMNET"
Response: b'AT+CGDCONT=1,"IP","CMNET"\r\r\nOK\r\n'
Sending Activate PDP context: AT+CGACT=1,1
Send: AT+CGACT=1,1
Response: b'AT+CGACT=1,1\r\r\nOK\r\n'
Sending 检查状态: AT+CGACT?
Send: AT+CGACT?
Response: b'AT+CGACT?\r\r\n+CGACT: 1,1\r\n\r\nOK\r\n'
Sending 获取IP地址: AT+CGPADDR=1
Send: AT+CGPADDR=1
Response: b'AT+CGPADDR=1\r\r\n+CGPADDR: 1,"10.0.192.76"\r\n\r\nOK\r\n'
Send: AT+QIOPEN=1,0,"TCP","112.125.89.8",32370,0,2
QIOPEN response: b'AT+QIOPEN=1,0,"TCP","112.125.89.8",32370,0,2\r\r\nCONNECT\r\n'
Sent: test
Received data: b'test'
Sent: test
Received data: b'test'
Sent: test
```

可以看到 4G 模块可以正常获取 IP 地址并且使用 TCP 和网络服务器通讯。

```
import machine
import utime
import time
import network
from machine import Pin
import usocket as socket
import sys
import ubinascii
import uos

GL_PWR = Pin(38, Pin.OUT)

def send_at(uart, cmd, timeout=2, wait_for_response=True):
    print(f"Send: {cmd}")
    uart.write(cmd + "\r\n")
    if not wait_for_response:
        return None

    start = utime.time()
    response = b""
    while (utime.time() - start) < timeout:
        if uart.any():
            response += uart.read()
            # 检查是否收到完整响应
            if b"OK" in response or b"ERROR" in response:
                break
        utime.sleep_ms(100)
    return response if response else None

def setup_modem():
    # 硬件初始化
    GL_PWR = Pin(38, Pin.OUT)
    GL_PWR.value(1)  # 使能供电
    uart = machine.UART(2, baudrate=115200, tx=39, rx=40)

    # 等待模块初始化
    print("Waiting for modem initialization...")
    utime.sleep(5)

    # 清空缓冲区
    while uart.any():
        uart.read()
```

```
    # AT 指令序列
        responses = {}
        commands = [
            ('AT', "Basic test"),
            ('AT+CPIN?', "if is ready"),
            ('AT+CGDCONT=1,"IP","CMNET"', "Set APN"),
            ('AT+CGACT=1,1', "Activate PDP context"),
            ('AT+CGACT?', "检查状态"),
            ('AT+CGPADDR=1', "获取 IP 地址"),
        ]

        for cmd, desc in commands:
            print(f"Sending {desc}: {cmd}")
            resp = send_at(uart, cmd, 5)
            responses[cmd] = resp
            print("Response:", resp)
            utime.sleep(1)

        return uart

def do_tcp_test(uart):
        host = "112.125.89.8"
        port = "32370"
        cmd = 'AT+QIOPEN=1,0,"TCP","%s",%s,0,2' % (host, port)
        resp = send_at(uart, cmd, 3)   # 增加超时时间，等待连接建立
        print("QIOPEN response:", resp)

        # 等待连接建立
        utime.sleep(2)

        count = 3
        while count > 0:
            if uart.any():
                # 读取串口数据
                data = uart.read()
                # 打印接收到的数据
                print("Received data:", data)

            # 发送消息并减少计数器
            uart.write('test')
            print("Sent: test")
            count -= 1

            utime.sleep(1)
```

```
# 退出透传模式
    utime.sleep(1)  # 确保没有数据传输
    uart.write(b'+++')  # 发送退出序列，注意使用字节
    utime.sleep(1)  # 等待模块响应

    resp = send_at(uart, "AT+QICLOSE=0", 5)  # 断开连接



def main():
    try:
        uart = setup_modem()
        do_tcp_test(uart)
    except Exception as e:
        print("严重错误:", e)
        import traceback
        traceback.print_exc()
    finally:
        GL_PWR.value(0)

if __name__ == '__main__':
    main()
```

3.2.3. Lora 模块

ECM50 的 Lora 模块和 4G 模块是二选一的，不同型号搭载不同通信。ECM50 搭载的亿佰特的 E22-400T22D 模块。这里我们使用 ECM50 和单独的另一块 E22-400T22D 进行通信测试，也可以使用另一块 ECM50 来进行 Lora 通信测试。

首先我们准备好单独的 E22-400T22D 模块的接线，使用 USB 转 TTL 连接好 PC。根据手册得知 M0 和 M1 分别对应了 4 种工作模式，此模块默认是配置好的，所以不需要再次配置，我们直接选择传输模式将 M0 和 M1 接地。

模块有四种工作模式，由引脚 M1、M0 设置；详细情况如下表所示：

模式（0-3）	M1	M0	模式介绍	备注
0 传输模式	0	0	串口打开，无线打开，透明传输	支持特殊指令空中配置
1 WOR 模式	0	1	可以定义为 WOR 发送方和 WOR 接收方	支持空中唤醒
2 配置模式	1	0	用户可通过串口对寄存器进行访问，从而控制模块工作状态	
3 深度休眠	1	1	模块进入休眠	

连接好串口之后，打开串口调试助手，设置波特率为 9600。

PC 打开 Thonny，然后运行 lora.py 代码，代码每秒接收一次数据和发送一次数据。

串口助手应该可以接收到 ECM50 发送的数据，并且串口助手发送的数据 ECM50 也可以接收到。

```
import machine
import utime
from machine import Pin

def main():
    M1 = Pin(41, Pin.OUT)
    M1.value(0)
    utime.sleep_ms(1000)
    uart = machine.UART(2, baudrate=9600, tx=39, rx=40)
    #uart.write('hello')
    #uart.write('AT+UART=?')

    while True:

        if uart.any():
            # 读取串口数据
            data = uart.read()

            # 打印接收到的数据
            print("Received data:", data)

        # 暂停 1000 毫秒
        utime.sleep_ms(1000)
        uart.write('test')

if __name__ == '__main__':
    main()
```

3.2.4. WIFI

ECM50 支持 2.4GHzWi-Fi(IEEE802.11b/g/n)，我们输入 WIFI 名称和密码来加入路由器的 WIFI，并自动获取 IP 地址，然后和局域网的设备进行 TCP 的通信，TCP 的通信设置网络调试助手和 ETH 章节的一样。同样注意 IP 地址和端口号即可。

```
def do_tcp_client():
    import socket
    import time

    # 服务器的 IP 地址和端口
    server_ip = '192.168.20.5'
    server_port = 8080   # 选择一个合适的端口号

    # 创建 socket 对象
    addr_info = socket.getaddrinfo(server_ip,server_port)
    addr = addr_info[0][-1]
    # 连接到服务器
    s = socket.socket()
    s.connect(addr)
    # 发送数据
    try:
        while True:
            print('send data')
            # 发送数据
            s.send(b'GET / HTTP/1.1\r\nHost: micropython.org\r\n\r\n')
            data = s.recv(1000)
            print(f"Received: {data.decode('utf-8')}")
            # 每隔 1 秒发送一次数据
            time.sleep(1)


    except KeyboardInterrupt:
        print("Client stopped by user.")
    finally:
```

```
            # 关闭连接
            client_socket.close()


def do_connect():
    import network
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('connecting to network...')
        wlan.connect('CMCC-AbPL', 'EC12345678')
        while not wlan.isconnected():
            pass
    print('network config:', wlan.ifconfig())
    print(wlan.isconnected())
    do_tcp_client()

#程序启动入口
if __name__ == '__main__':
    do_connect()
```

3.2.5. 蓝牙

蓝牙可以作为主设备和从设备，代码分别是 ble_simple_peripheral.py - 蓝牙外设（从设备），ble_simple_central.py - 蓝牙中央设备（主设备），ble_advertising.py - 蓝牙广播数据包生成器。代码附在 Sources 里面。

ble_simple_peripheral.py 用途是创建一个 BLE 外设设备，提供 UART 服务，作为服务端，等待中央设备连接，提供 UART 服务，包含 TX（发送）和 RX（接收）特征，可以接收数据和发送通知。

ble_simple_central.py 用途是创建一个 BLE 中央设备，搜索并连接外设，作为客户端，主动扫描和连接外设，发现 UART 服务并建立通信，可以发送数据和接收通知。

3.2.5.1. 可以使用两个 ESP32 设备测试。

在第一个 ESP32 上运行 ble_simple_peripheral.py，在第二个 ESP32 上运行 ble_simple_central.py

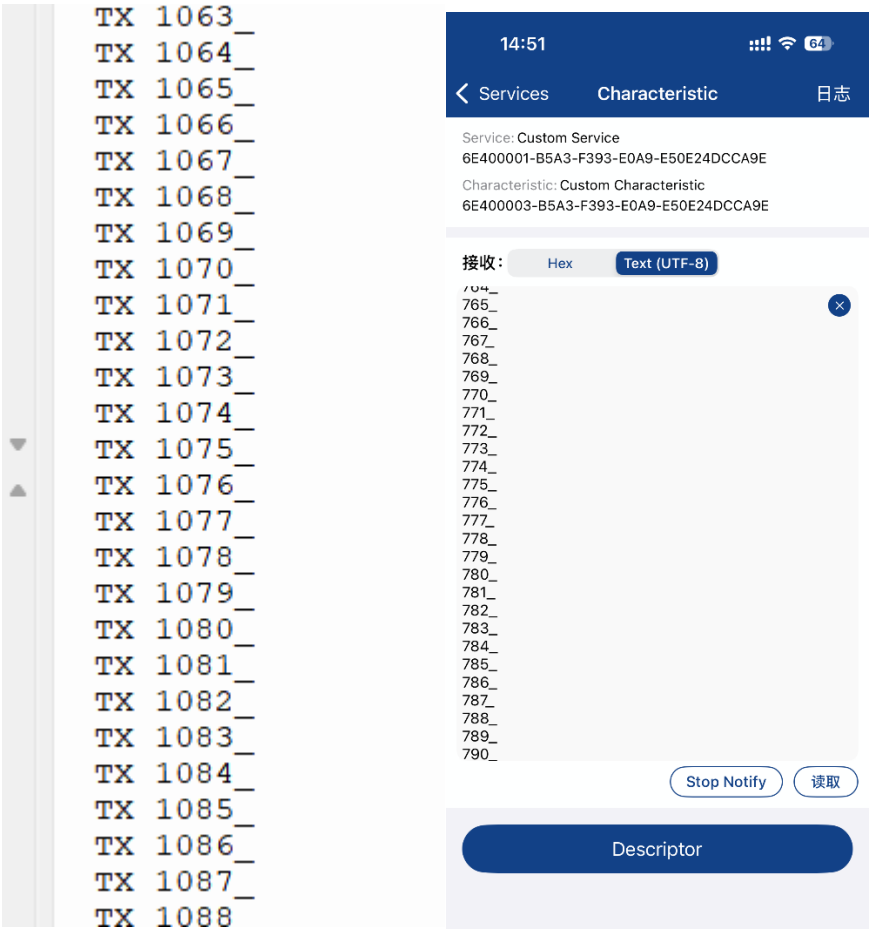
中央设备会扫描并找到外设，建立连接后，中央设备会每秒发送递增的数字，外设收到数据后会打印"RX"信息，外设也会每 100ms 发送 3 个数据包。

将 ble_advertising.py 生成器右键点击上传到/，才可以正常运行。

3.2.5.2. 使用手机 APP 测试

手机上下载一个蓝牙调试助手。

在 ESP32 上上传 ble_advertising.py，然后运行 ble_simple_peripheral.py，手机扫描蓝牙设备，找到"mpy-uart"，连接后查看 UART 服务，向 RX 特征写入数据，观察 ESP32 控制台输出，订阅 TX 特征，接收 ESP32 发送的数据。



4. 扩展应用

4.1.ModBus TCP

Modbus TCP 是 Modbus 协议在以太网上的一种实现，它保留了 Modbus 协议的核心功

能和数据模型，但对消息封装进行了调整，使其符合 TCP/IP 的要求。

我们通过 W5500 以太网模块与局域网的 PC 进行通信，ECM50 分别实现了 ModBus TCP 的 Poll 端和 Slave 端。可以简单理解为 slave 端是被动的，被 Poll 端拿数据。在上述的过程中，PC 分别使用 modbus poll 和 modbus slave 软件进行配合，没有的可以去网上下载。

uModBusConst.py - 定义了 Modbus 协议的所有标准常量和参数

uModBusFunctions.py - 构建符合 Modbus 协议格式的请求数据包

uModBusTCPPoll.py - 实现 Modbus TCP 客户端协议的完整通信流程

main.py-调用了 uModBusTCPPoll 的接口，负责连接网络和激活客户端。

uModBusTCPSlave.py-依赖 uModBusConst 的常量，完成了连接网络和 modbus 服务端的实现。

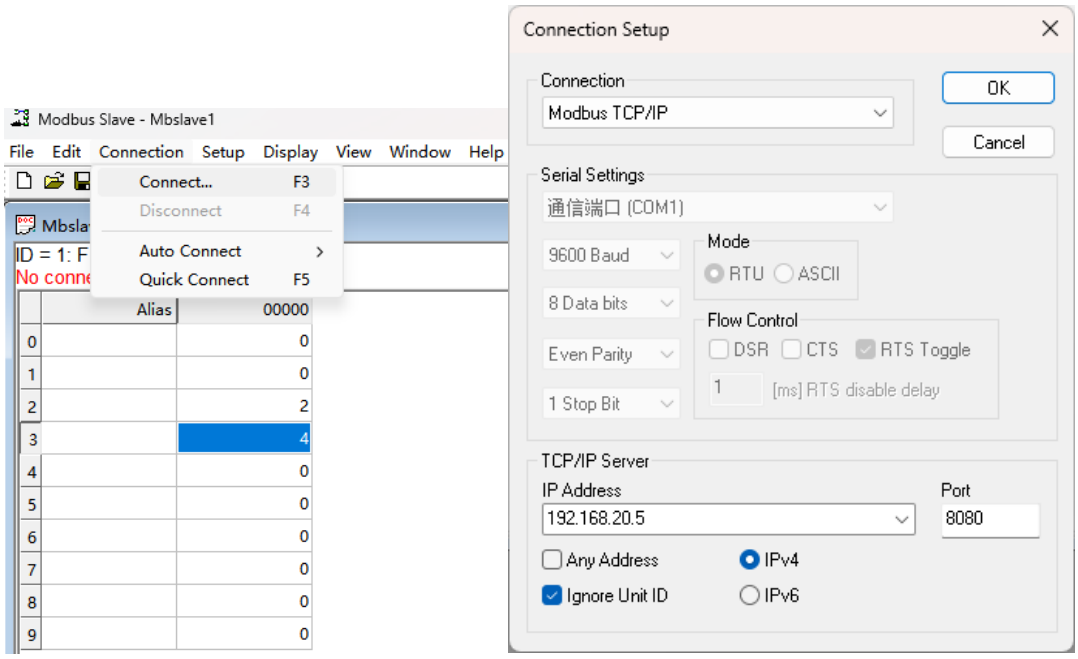
代码放在 04_Sources/modbus 下。

4.1.1. ECM50 Poll

我们 ECM50 作为客户端，需要使用 Poll 的相关代码，PC 则使用模拟软件 modbus slave。

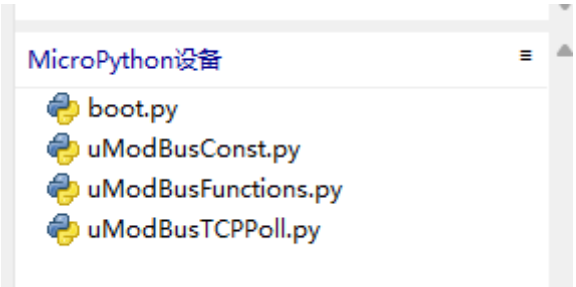
我们在 PC 先打开 modbus slave 软件，这个软件可以模拟 modbus 设备，我们的代码就是通过网络和这个虚拟设备进行通信的。

打开后设置连接，然后选择 TCP/IP 连接，关闭 any address，设置为 PC 的 IP 地址，选择端口号为 8080，然后点击 OK。



我们使用 Thonny 打开 modbus 文件夹，将 uModBusConst.py，uModBusFunctions.py，

uModBusTCPPoll.py 上传到 ECM50，因为我们的测试代码依赖这三个文件。然后双击打开 main.py 设置 IP 地址和端口号，是上述在 modbus slave 中设置的 IP 地址和端口号，最后点击绿色按钮执行。



ECM50 就会每秒读取寄存器的值并打印，可以动态修改 modbus slave 中寄存器的值，ECM50 这边也会动态读取到相应的值。

```
等待连接...
连接成功！
IP地址: 192.168.20.48
MAC地址: dc:b4:d9:10:78:a3
网络连通性正常
尝试连接到Modbus服务器 (1/5)...
成功连接到Modbus服务器
读取的值: (0, 0, 5, 0, 0)
读取的值: (0, 0, 5, 0, 0)
读取的值: (0, 0, 5, 0, 0)
读取的值: (0, 0, 5, 0, 0)
读取的值: (0, 0, 5, 0, 0)
```

4.1.2. ECM50 Slave

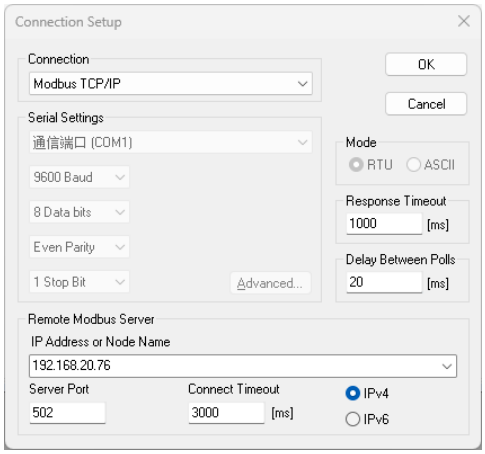
ECM50 作为服务端，而 PC 作为客户端，需要 ECM50 先执行代码作为服务器，然后 PC 使用 modbus poll 软件拉取数据。

我们先将依赖上传到 ECM50，uModBusTCPSlave.py 把数据包和通信流程写到一起了，所以只依赖 uModBusConst.py，将此文件上传到 ECM50。

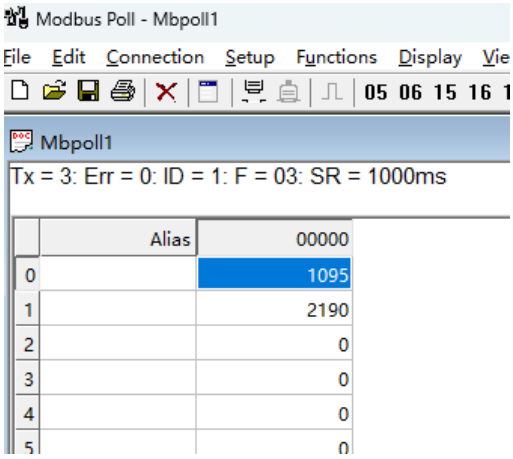
然后双击打开 uModBusTCPSlave.py，点击绿色执行按钮执行代码。

```
MPY: soft reboot
=== Modbus TCP Slave 启动 ===
目标IP: 192.168.20.76
Modbus端口: 502
从站ID: 1
等待网络连接...
等待网络连接...
等待网络连接...
等待网络连接...
网络连接成功！
IP地址: 192.168.20.76
MAC地址: dc:b4:d9:10:78:a3
数据更新定时器已启动
Modbus服务器开始运行...
Modbus TCP Slave started on port 502, Slave ID: 1
Data sizes - Coils: 1000, Registers: 1000
```

然后我们打开 modbus poll 软件，点击 connect，设置为 TCP/IP 连接，并输入我们代码中设置的 IP 地址和端口号，最后点击 OK。



可以看到我们的寄存器 0 和 1 在每秒更新中，因为我们的代码使用定时器每秒在写这两个寄存器。



4.2.MQTT

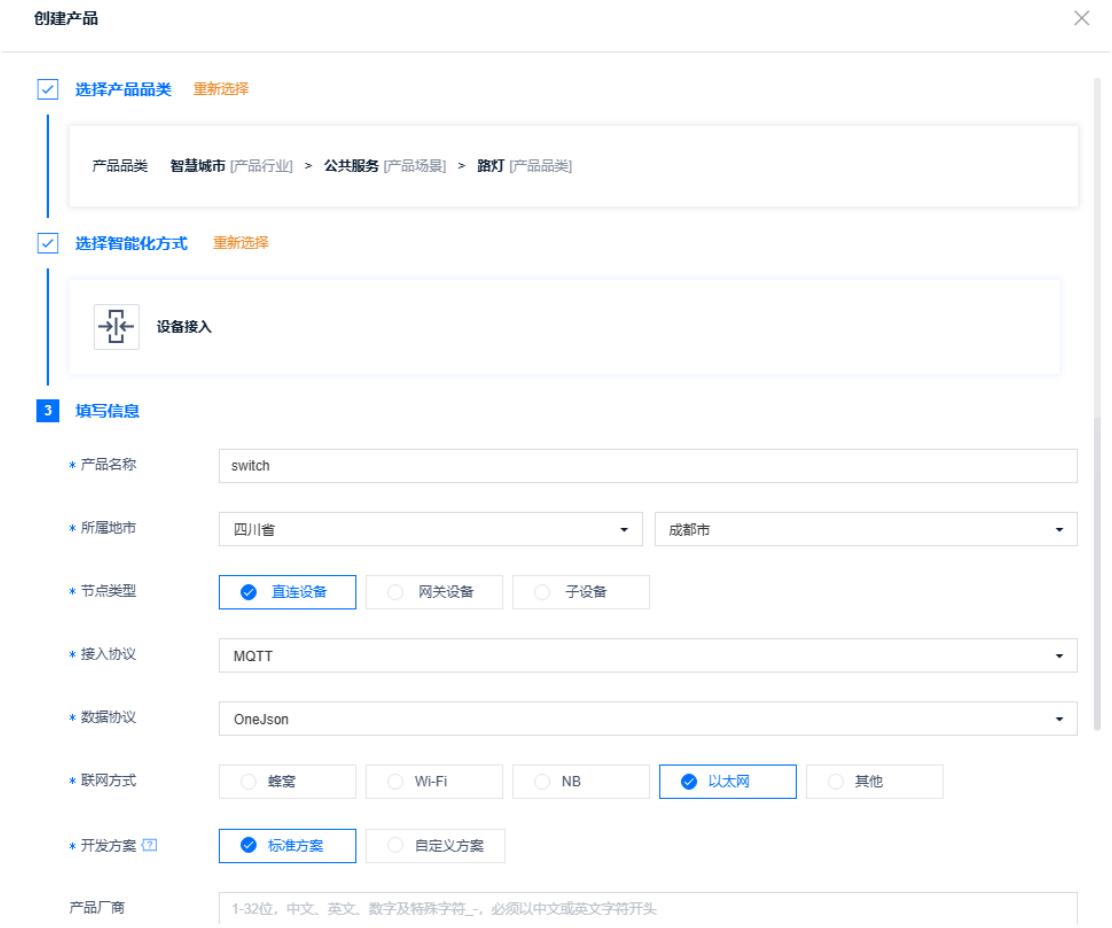
在物联网领域，传感器与服务器的通信、信息的收集以及 MQTT 协议都是可以考虑的方案之一。MQTT 通常用于需要低功耗和低带宽的场景，如物联网设备、传感器网络、移动应用等。它的轻量级和高效特性使其非常适合在资源受限的设备和不可靠网络环境中使用。

接下来我们演示在 ECM50 平台上使用代码来访问 ONENET 物联网开发平台。从创建一个产品类到创建一个设备，最后使用 ECM50 来控制该设备的属性。

4.2.1. 创建产品

我们首先注册登录好之后，进入控制台。点击产品开发，创建产品。

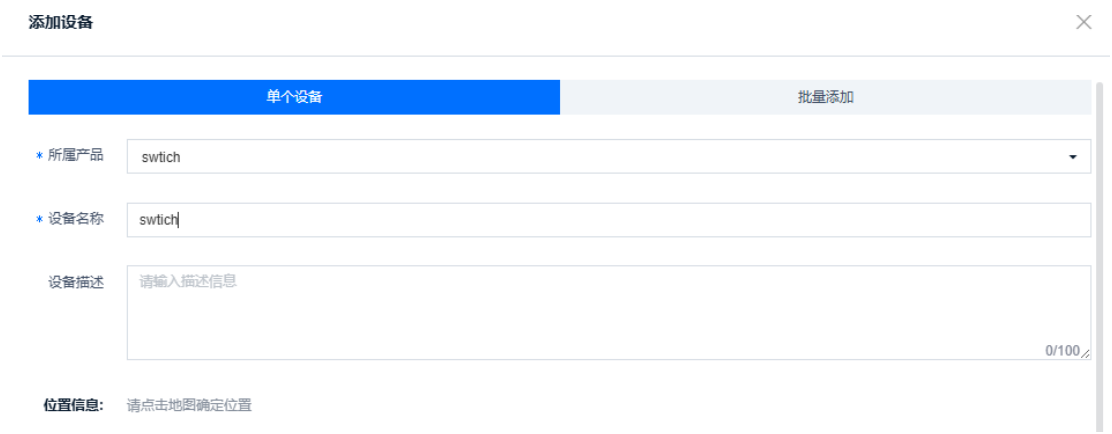
创建产品填入必要的信息。



点击产品开发，直接点击下一步，下一步，然后点击发布。

4.2.2. 添加设备

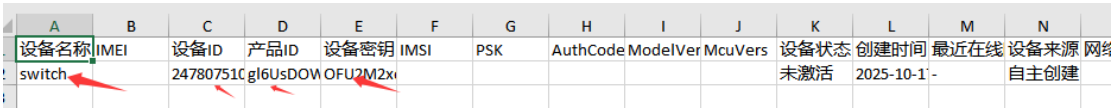
接下来我们添加设备，点击添加设备。



创建好设备之后，我们点击选择设备然后导出设备。

4.2.3. 生成 token

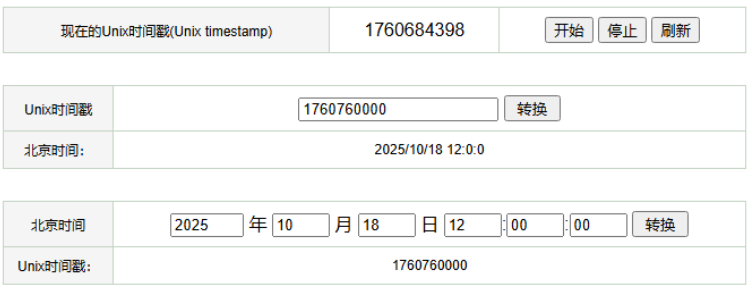
得到了一个 excel 是我们的设备的所需信息，我们需要用这些数据来生成密码。



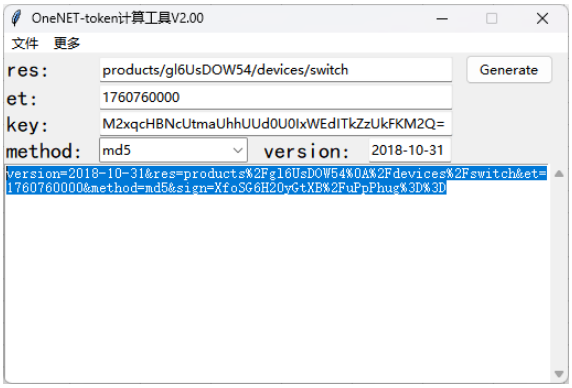
我们还需要获取一个时间戳，这个时间戳就是有效时间，我们直接从现在的时间往后推几个小时来填写即可。时间戳的意义是在此使用时间前有效

使用时间戳转换工具

https://unixtime.bmcx.com/



接下来我们打开 onenet 提供的密码生成器，我们随附件放到了 03_Tools/token.exe



将我们前面记录下来的信息填入上面的程序。

将设备名称，产品 ID 填入 res，将时间戳填入 et，将设备密钥填入 key，点击生成，我们就得到了一个密码（token）。

4.2.4. 修改 ECM50 代码

我们打开 mqtt_onenet_eth.py

```
import ujson
import _thread
import network
from umqtt.simple import MQTTClient
import time
import machine
import ubinascii


Broker_Address="mqtts.heclouds.com"
password="version=2018-10-31&res=products%2Fgl6UsDOW5
User_Name="gl6UsDOW54"
client_ID ="switch"
Broker_Port=1883
# 服务质量
qos = 0
# topic
get_topic = "$sys/gl6UsDOW54/switch/thing/property/post/reply"
post_topic = "$sys/gl6UsDOW54/switch/thing/property/post"  # 上报属性

# 固定设备ID
device_id = "2478075109"

BrightValue = 75
```

修改对应的信息即可，运行代码后，代码将会每一段时间上传 brightvalue 亮度值。

5. 参考资料

- Micropython 官方 guide：

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

- ESP-IDP 编程指南

  https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32s3/index.html

6. 修订说明

修订说明表

版本	修改内容	修改时间	编制	校对	审批
V1.0	初稿	25-10-14	HSL	LJQ	WFX
V1.1	修改 4G 模块相关内容	25-11-11	HSL	LJQ	WFX

7.关于我们

销售热线：4000-330-990
技术支持：support@cdebyte.com 官方网站：https://www.ebyte.com
公司地址：四川省成都市高新西区西区大道 199 号 B5 栋

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.