

FireBeetle 2 ESP32-C5之大棚环境监测系统

作者：HonestQiao / 乔楚

版本：v1.0

日期：2025-10-31

首发：<https://mc.dfrobot.com.cn/thread-398562-1-1.html>

关键词1：温湿度检测、气压检测、光线强度检测、电池电量检测、低功耗、深度休眠、环境监测、物联网

关键词2：MQTT、Mind+、Arduino、SloT

关键词3：行空板M10、行空板K10、FireBeetle 2 ESP32-C5、DFRobot

〇、背景

今年北京的冬天，比以往来的更早一些，还没到11月，最低气温就降到了0度，让人猝不及防。我家阳台上向来种了不少花花草草，今年因为雨水充足，长得格外茂盛，不能像往年一样，搬进家里过冬。

于是，在阳台上，搭了一个小型大棚，为花花草草提供一个过冬的场所：



大棚搭好了，就想着监控大棚内的温湿度等环境参数，后面再配上自动加热装置，保持一定的温度，好方便植物们过冬。

于是，就结合手头有的开发板、传感器，做了个基础的大棚环境监测系统。

下面，把制作的步骤，给大家分享。

一、系统规划

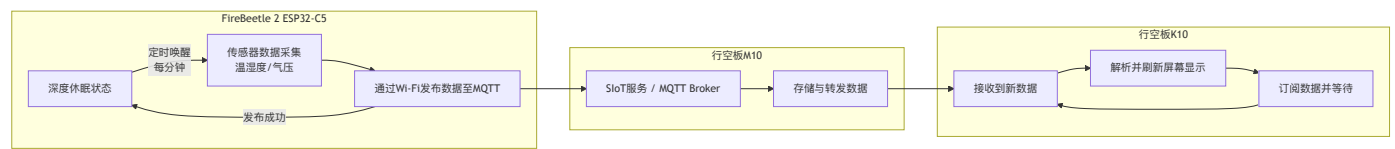
因为这次的大棚，是搭建在阳台上，家里的无线路由的信号可以覆盖，所以就选择了FireBeetle 2 ESP32-C5来读取传感器信息，然后通过WiFi发送信息。获取传感器的数据，然后发送出去，通过MQTT协议，是再好不过的。

而行空板M10，自带MQTT服务SIoT，可以轻松的建立MQTT服务环境，就用来做为本系统的中央服务环境。

数据有了，还需要方便查看，而行空板K10，核心芯片ESP32-S3，使用Mind+图形界面开发环境，可以很方便的接入WiFi，连接MQTT服务获取数据，并控制屏幕的显示，就用作手持终端了。

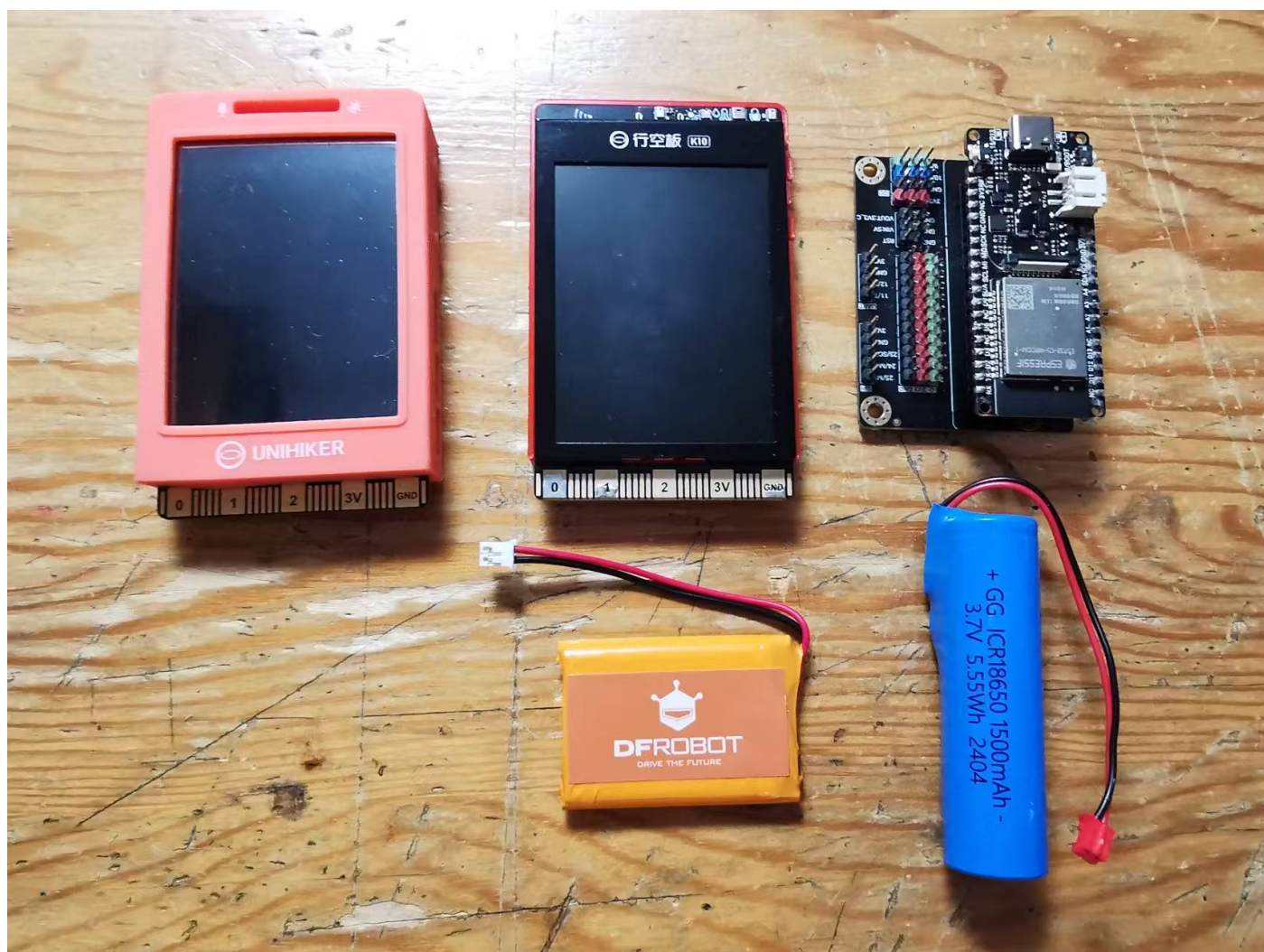
因为FireBeetle 2 ESP32-C5需要放到室外大棚中，暂时使用电池供电，为了提供更长时间的续航，在每次读取传感器数据并发布信息后，自动进入深度休眠模式，指定时间后，再自动唤醒进入下一次操作。

那么，规划的基本系统流程如下：



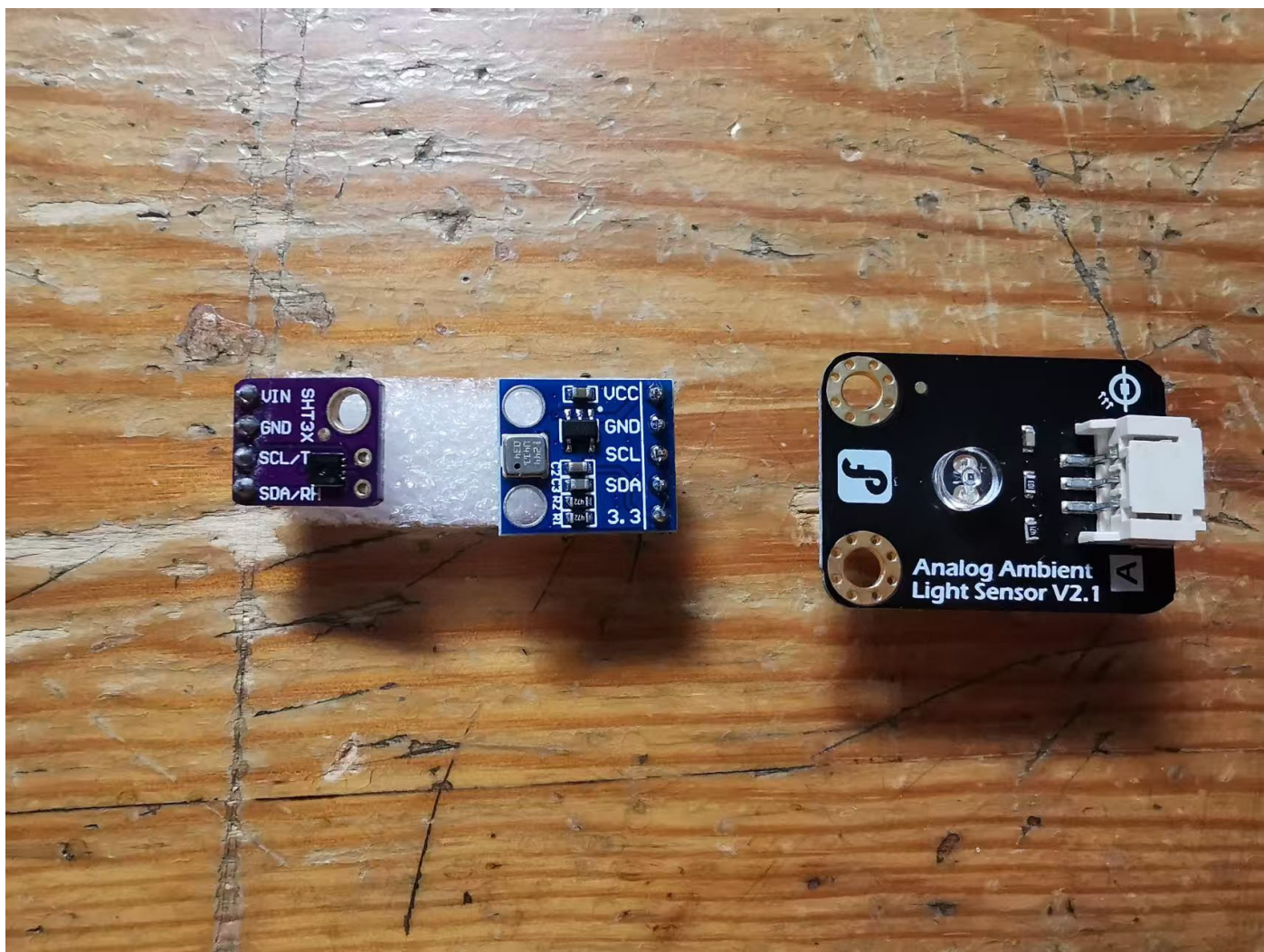
二、硬件准备

根据前面的系统规划，我用到如下的开发板：



- 行空板M10：使用USB供电
- 行空板K10：使用锂电池供电，是购买的DFRobot官方锂电池，3.7V 1000mAh。
- FireBeetle 2 ESP32-C5开发套件：使用18650充电电池供电，3.7V 1500mAh。

我还使用了如下传感器：



- 温湿度传感器SHT30：使用I2C接口
- 气压传感器BMP180：使用I2C接口
- 模拟光线传感器：使用ADC接口

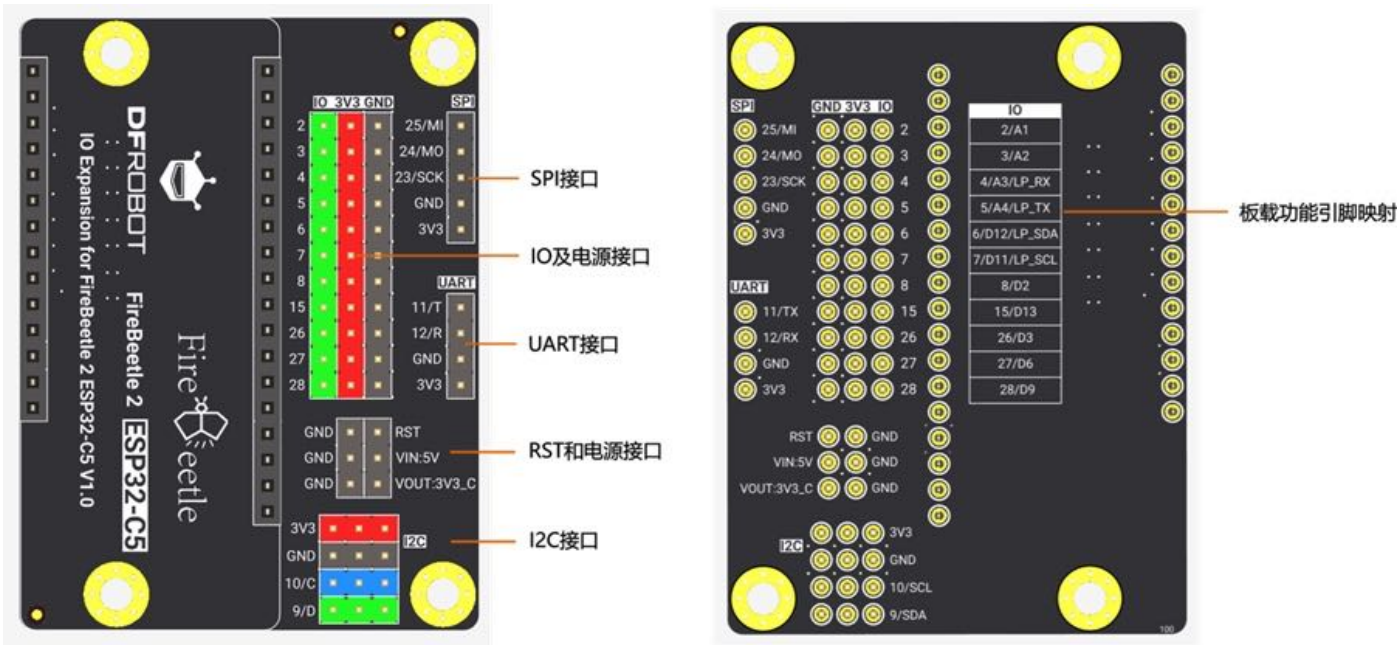
另外，我还准备了太阳能板：



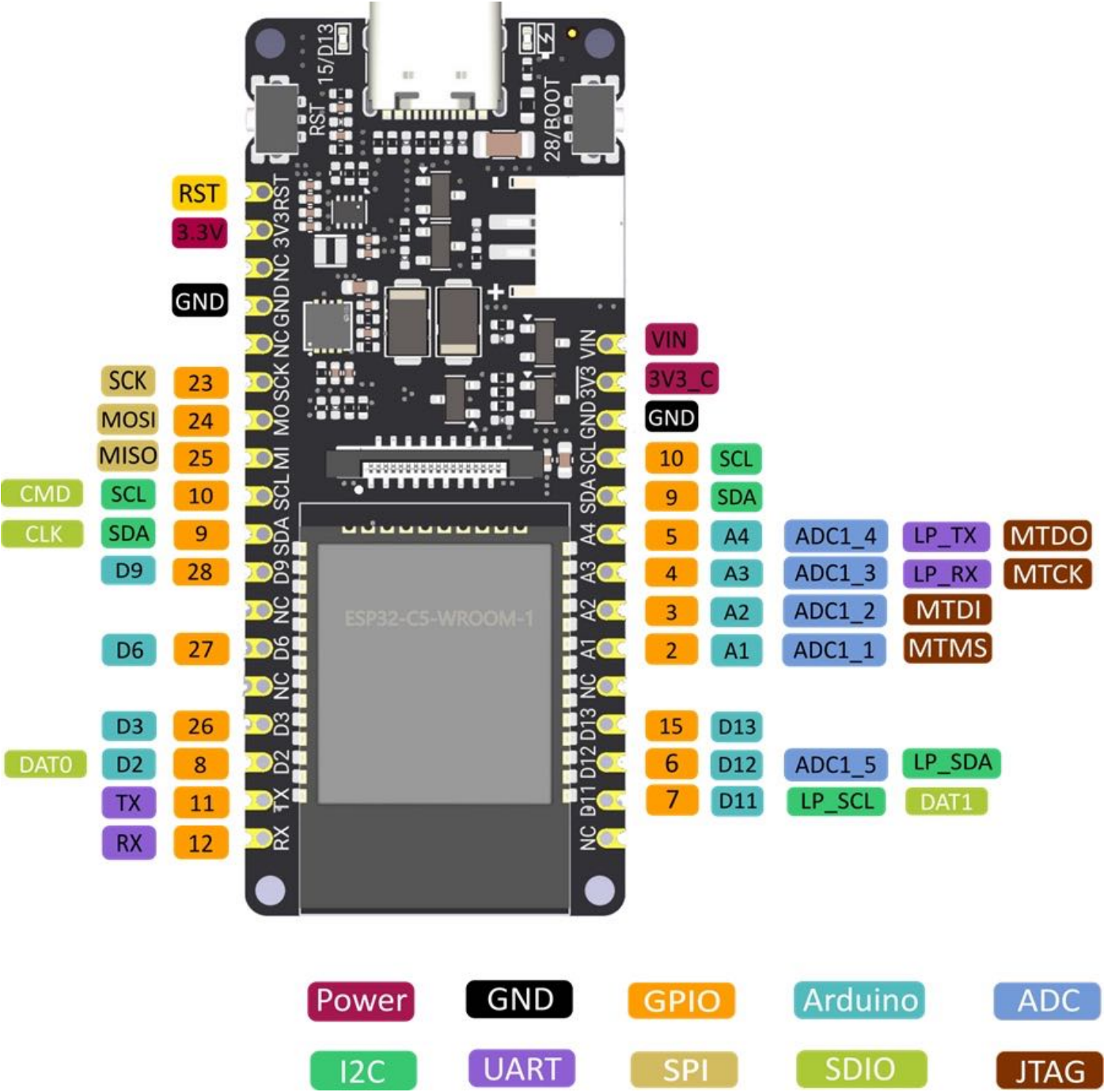
不过暂时太阳能板还没有用起来。在后续的更新中，将会把太阳能板用起来。

三、硬件连线

因为传感器是连接到FireBeetle 2 ESP32-C5扩展板上的，所以先了解FireBeetle 2 ESP32-C5扩展板的引脚安排：

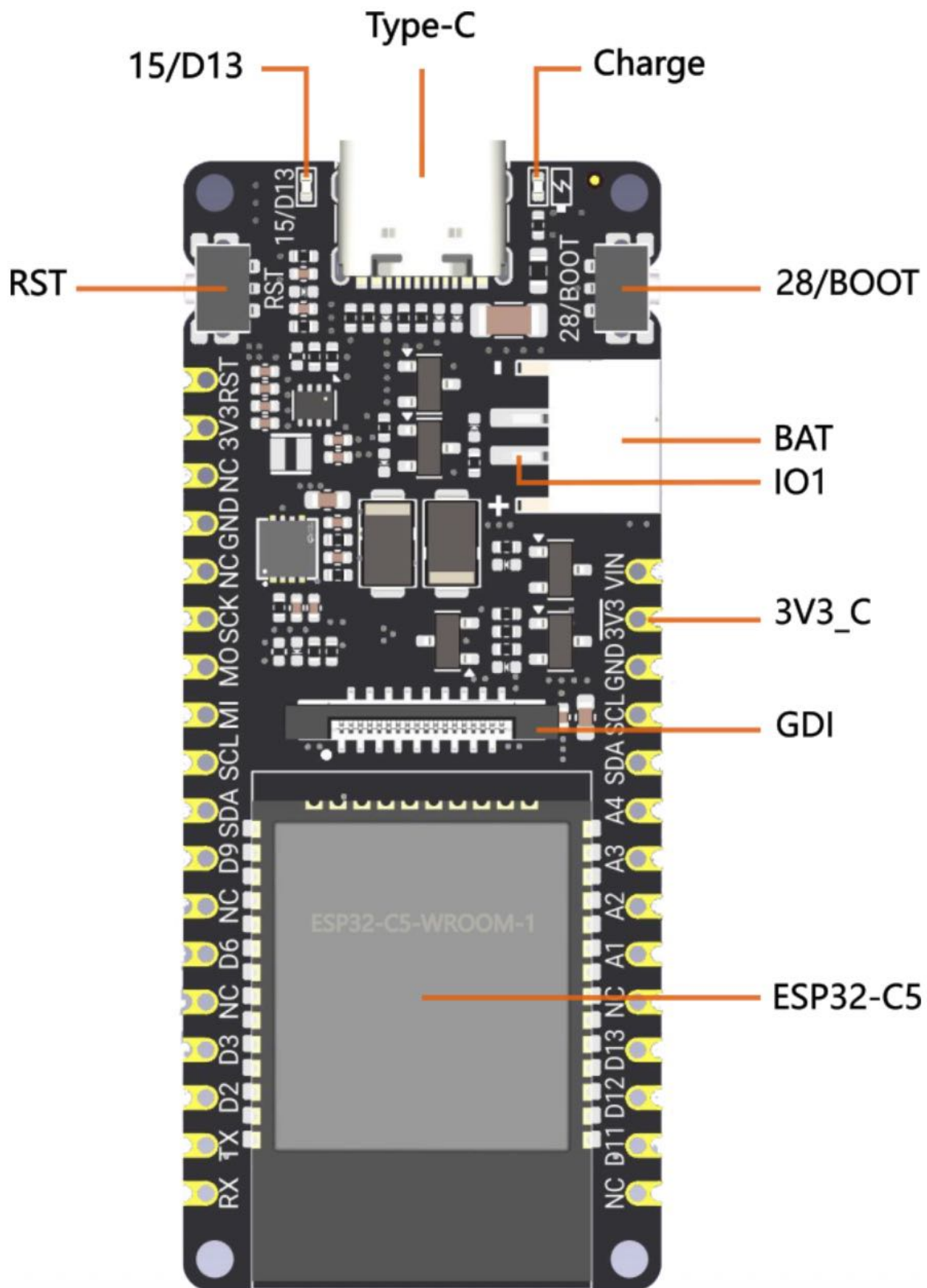


另外，还需要用到ADC引脚，所以也需要了解FireBeetle 2 ESP32-C5本身的引脚功能：

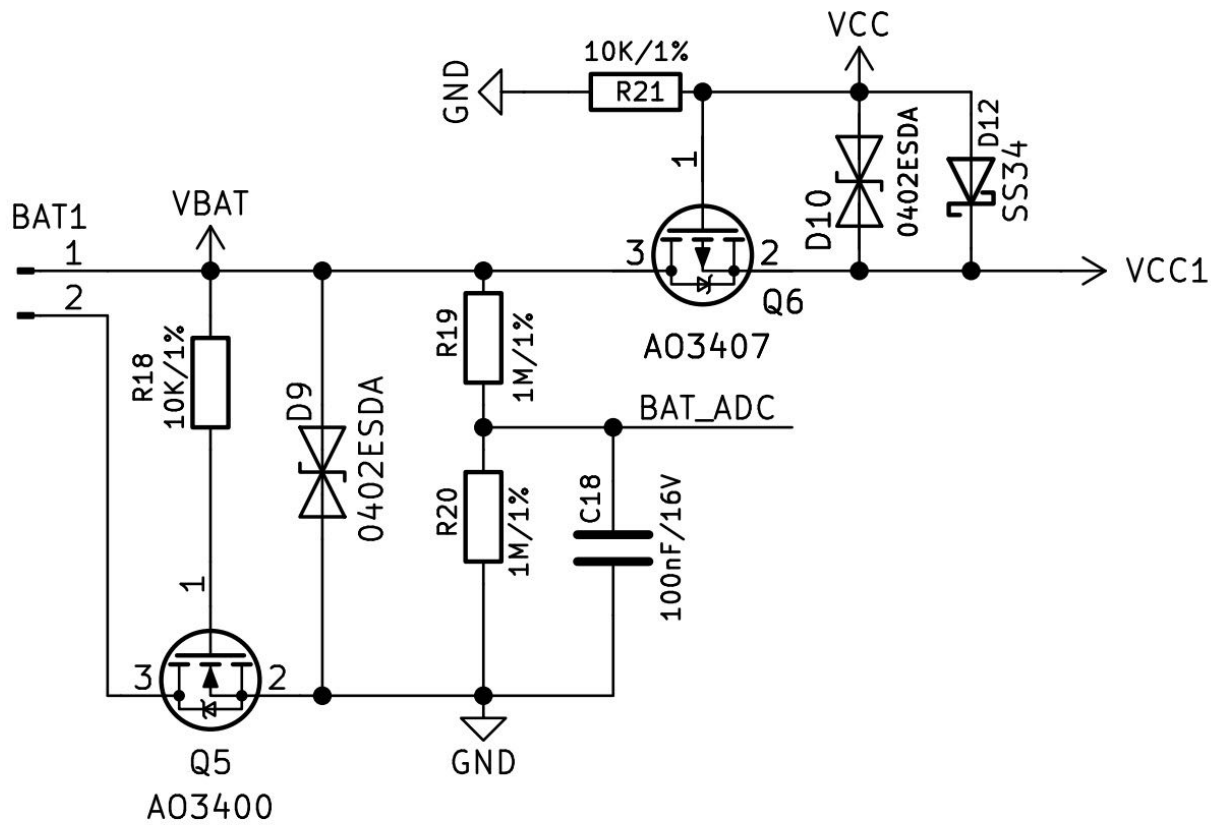


从以上两图，具体安排将I2C设备，连接到扩展板的I2C接口处，将模拟光线传感器接到扩展板的2号引脚。

另外，从DFRobot提供的资料，可以得知电池检测接口为IO1:



从DFRobot提供的原理图可以了解电池检测的具体电路：

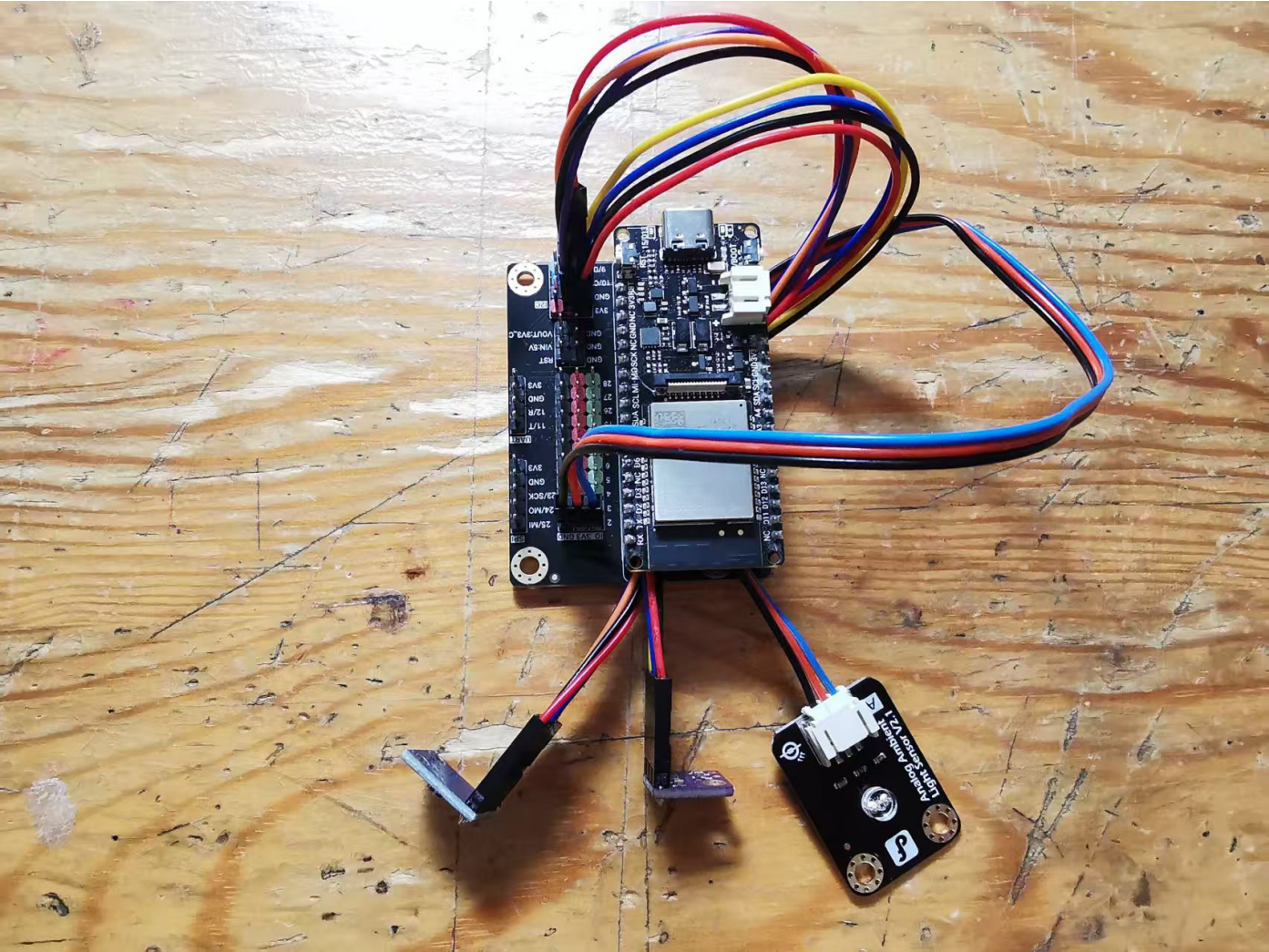


最终的接线如下：

FireBeetle 2 ESP32-C5扩展板	传感器引脚
3V	SHT30-VIN
GND	SHT30-GND
C-10	SHT30-SCL
D-9	SHT30-SDA
3V	BMP180-3.3
GND	BMP180-GND
C-10	BMP180-SCL
D-9	BMP180-SDA
3V	光线传感器-3.3
GND	光线传感器-GND
2	光线传感器-A

BAT+	锂电池+
BAT-	锂电池-
BAT_ADC-IO1	电池电压检测

FireBeetle 2 ESP32-C5与传感器的最终接线效果如下：



四、代码编写

这个大棚环境监测系统的代码，分为两个部分，分别为：

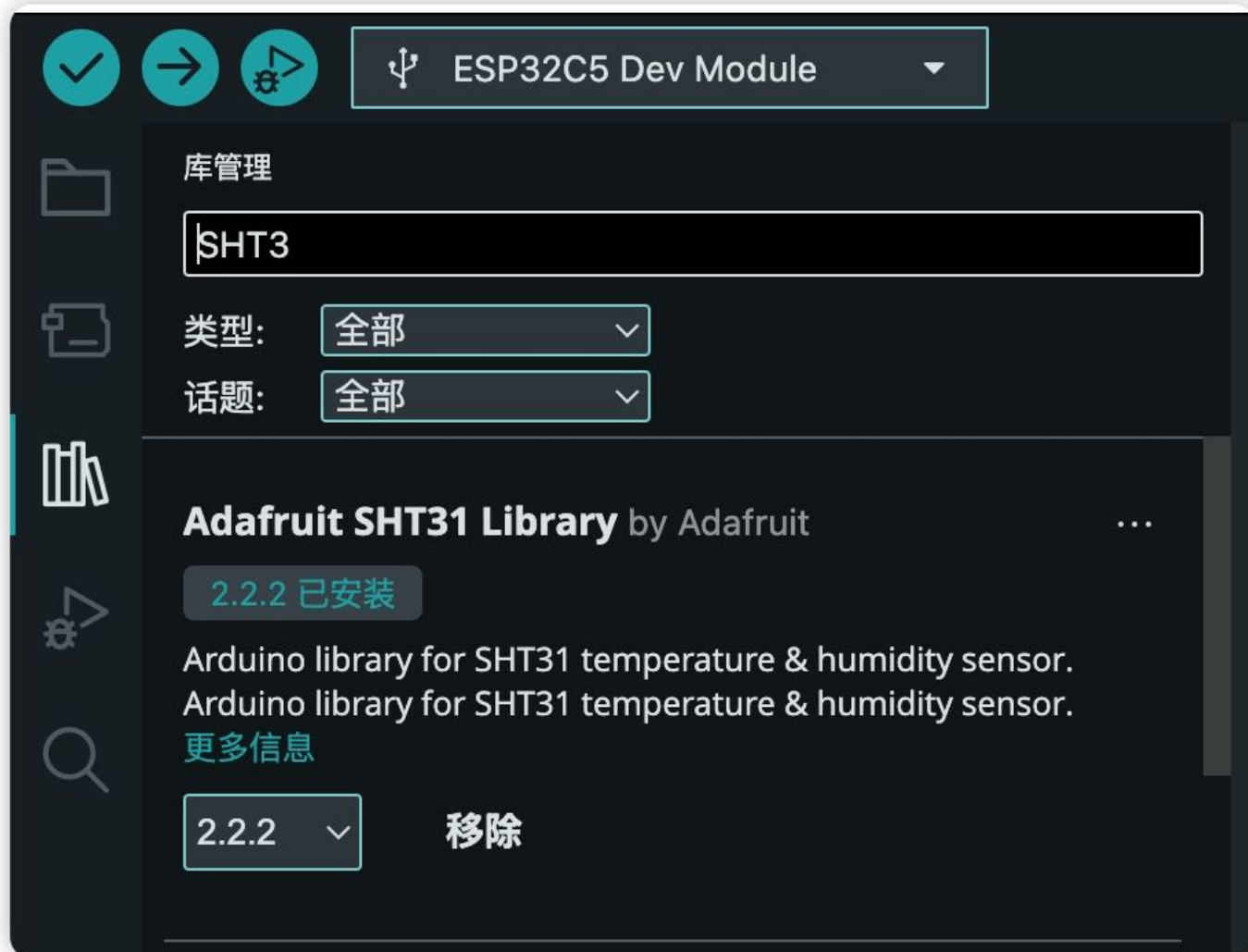
- 环境信息采集：使用Arduino IDE编写，运行在 FireBeetle 2 ESP32-C5 上
- 环境信息查看：使用Mind+ 编写，运行在 行空板K10 上

4.1 环境信息采集

4.1.1 扩展库安装

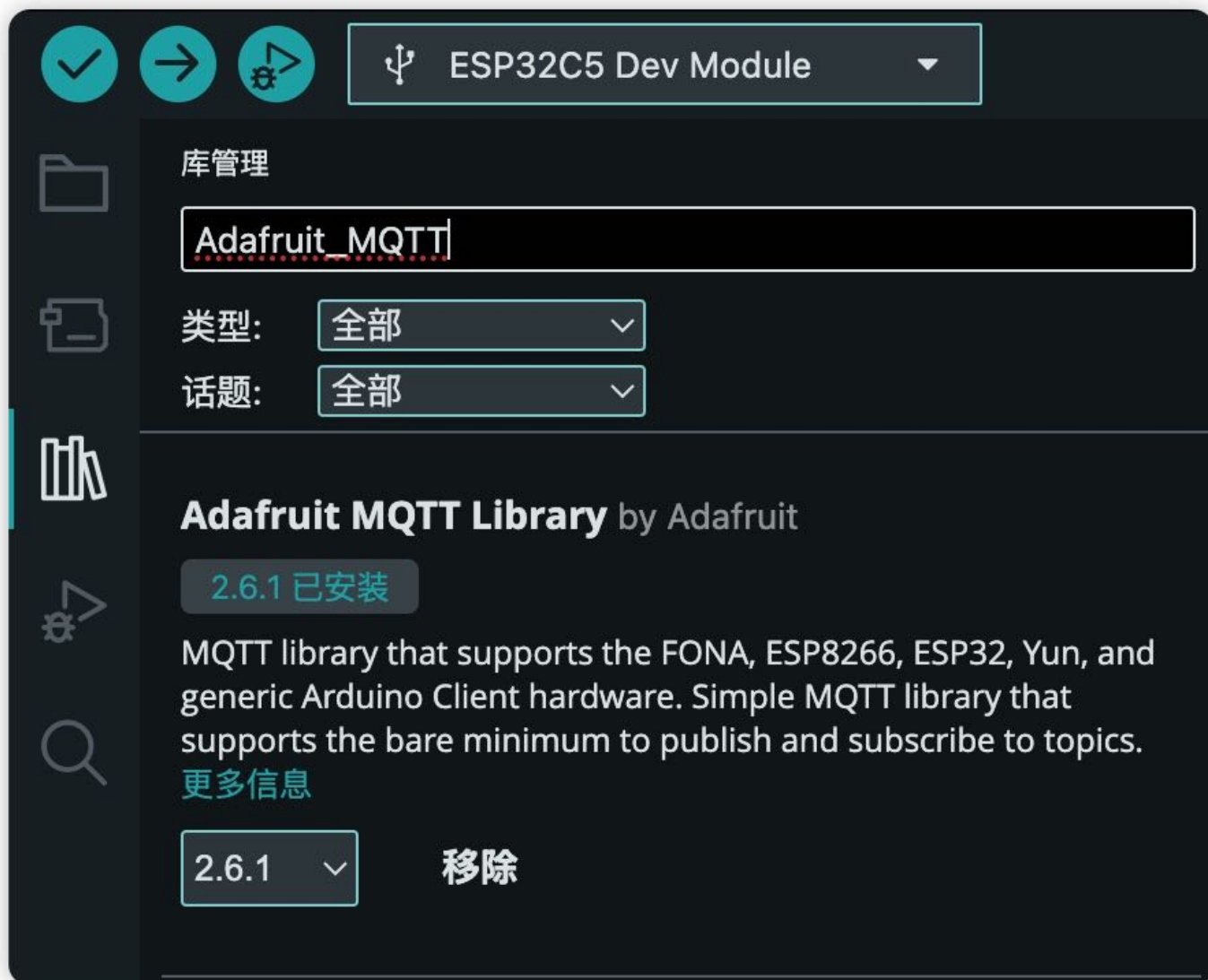
这部分的代码，需要读取传感器的数据，需要安装Adafruit_SHT31、Adafruit_BMP085这两个

扩展库：





另外，还要使用MQTT功能，需要安装Adafruit_MQTT扩展库：



4.1.2 实际代码

具体的代码，可以查看文章底部附件：[EnvMonitor_v1.0.zip](#)

这部分代码的逻辑流程如下：

代码中，关键配置如下：

4.1.2.1 调试和休眠配置

```

9 // =====
10 // 系统配置参数
11 // =====
12 #define DEBUG_MODE          0      // 调试模式：0-关闭 1-打开
13 #define ENABLE_SLEEP        1      // 允许休眠
14 #define ENABLE_MQTT         1      // 启用MQTT功能
15
16 #define uS_TO_S_FACTOR      1000000ULL // 微秒到秒转换因子
17 #if ( DEBUG_MODE == 1 )
18 #define TIME_TO_SLEEP       10      // ESP32休眠时间（秒）
19 #else
20 #define TIME_TO_SLEEP       60      // ESP32休眠时间（秒）
21 #endif

```

调试模式(DEBUG_MODE=1)时，TIME_TO_SLEEP设置为10，表示睡眠10秒再唤醒；
正式运行(DEBUG_MODE=0)时，TIME_TO_SLEEP设置为60，表示睡眠1分钟再唤醒。

4.1.2.2 WiFi和MQTT配置

```

25
26 // =====
27 // WiFi配置
28 // =====
29 #define WLAN_SSID           "WiFi名称"
30 #define WLAN_PASS           "WiFi密码"
31
32 // =====
33 // MQTT配置
34 // =====
35 #define AIO_SERVER          "192.168.1.185" // MQTT服务器地址
36 #define AIO_SERVERPORT      1883           // MQTT端口
37 #define AIO_USERNAME        "siot"         // MQTT用户名
38 #define AIO_KEY              "dfrobot"      // MQTT密码
39
40 // 气压传感器配置
41 #define BMP180_REFERENCE_PRESSURE 102500 // 参考海平面气压值
42

```

其中：

- WLAN：修改为对应的WiFi接入信息
- AIO：使用行空板M10的SiOT服务，端口、用户名、密码为默认值，请根据实际情况修改；IP需要在行空板M10上查看获取到的IP地址。

- 参考海平面气压值：这个需要根据实际情况调整

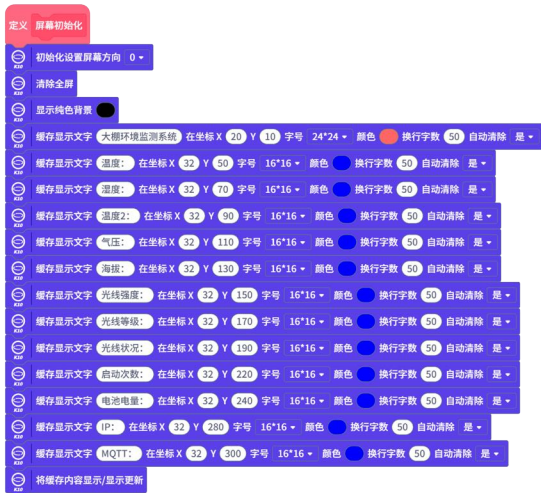
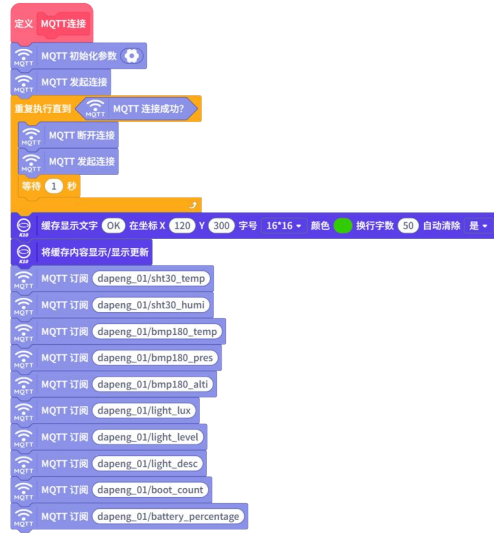
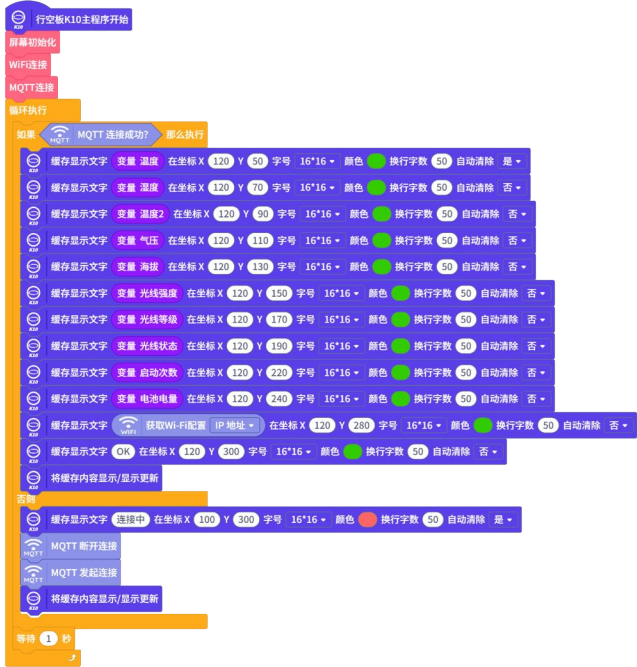
4.1.2.3 MQTT消息主题

另外，采集到的传感器信息，对应发布到如下的topic：

```
92 // =====
93 // MQTT对象声明
94 // =====
95 Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
96
97 // MQTT发布对象
98 Adafruit_MQTT_Publish sht30_temp = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/sht30_temp");
99 Adafruit_MQTT_Publish sht30_humi = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/sht30_humi");
100 Adafruit_MQTT_Publish bmp180_temp = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/bmp180_temp");
101 Adafruit_MQTT_Publish bmp180_pres = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/bmp180_pres");
102 Adafruit_MQTT_Publish bmp180_alti = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/bmp180_alti");
103 Adafruit_MQTT_Publish light_lux = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/light_lux");
104 Adafruit_MQTT_Publish light_level = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/light_level");
105 Adafruit_MQTT_Publish light_desc = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/light_desc");
106 Adafruit_MQTT_Publish boot_count = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/boot_count");
107 Adafruit_MQTT_Publish battery_voltage = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/battery_voltage");
108 Adafruit_MQTT_Publish battery_percentage = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/battery_percentage");
109 Adafruit_MQTT_Publish battery_status = Adafruit_MQTT_Publish(&mqtt, "dapeng_01/battery_status");
110
```

4.2 环境信息显示

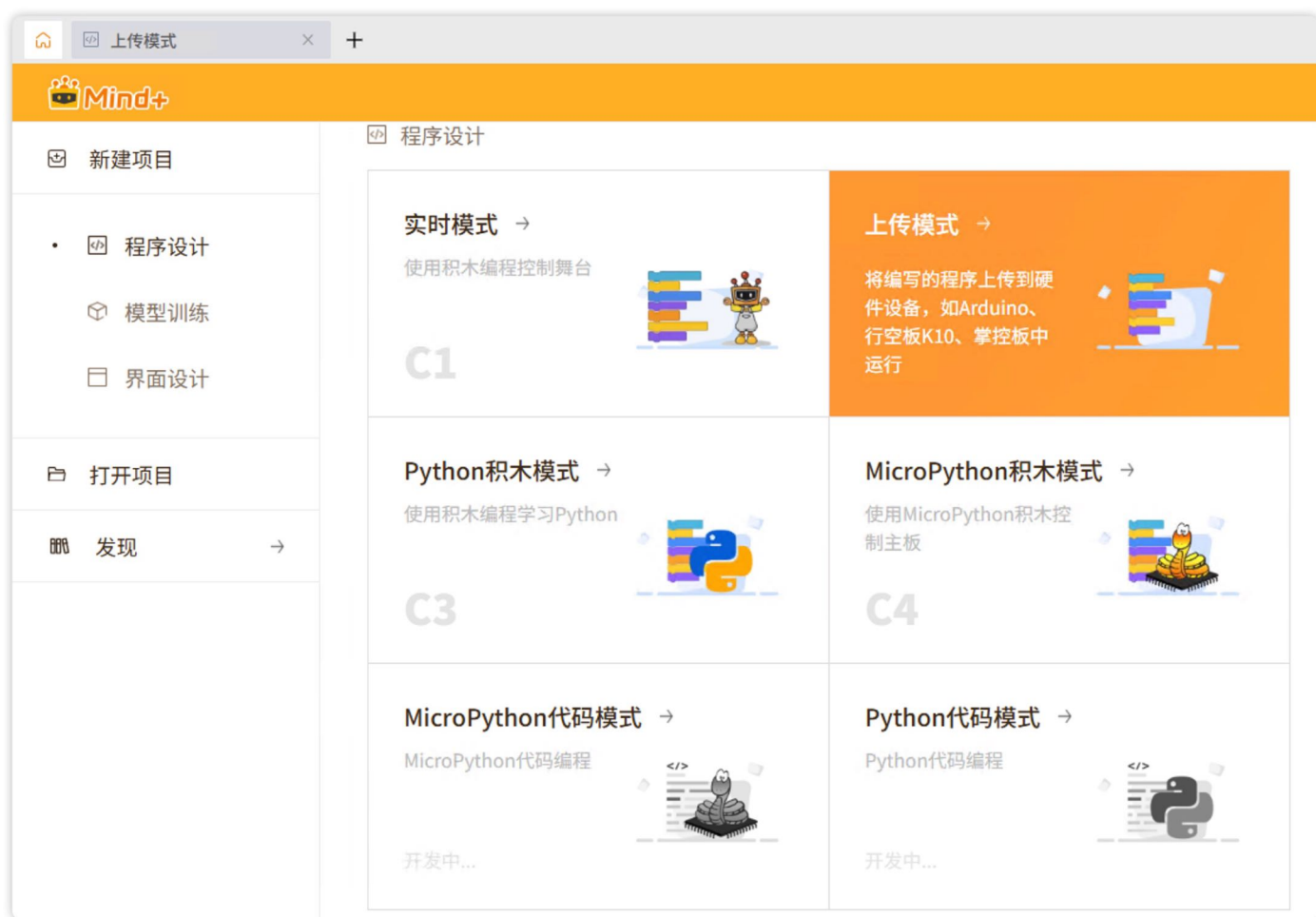
这部分的代码，使用Mind+ 2内测版本进行开发，完整代码如下：



具体的开发步骤如下。

4.2.1 开发模式

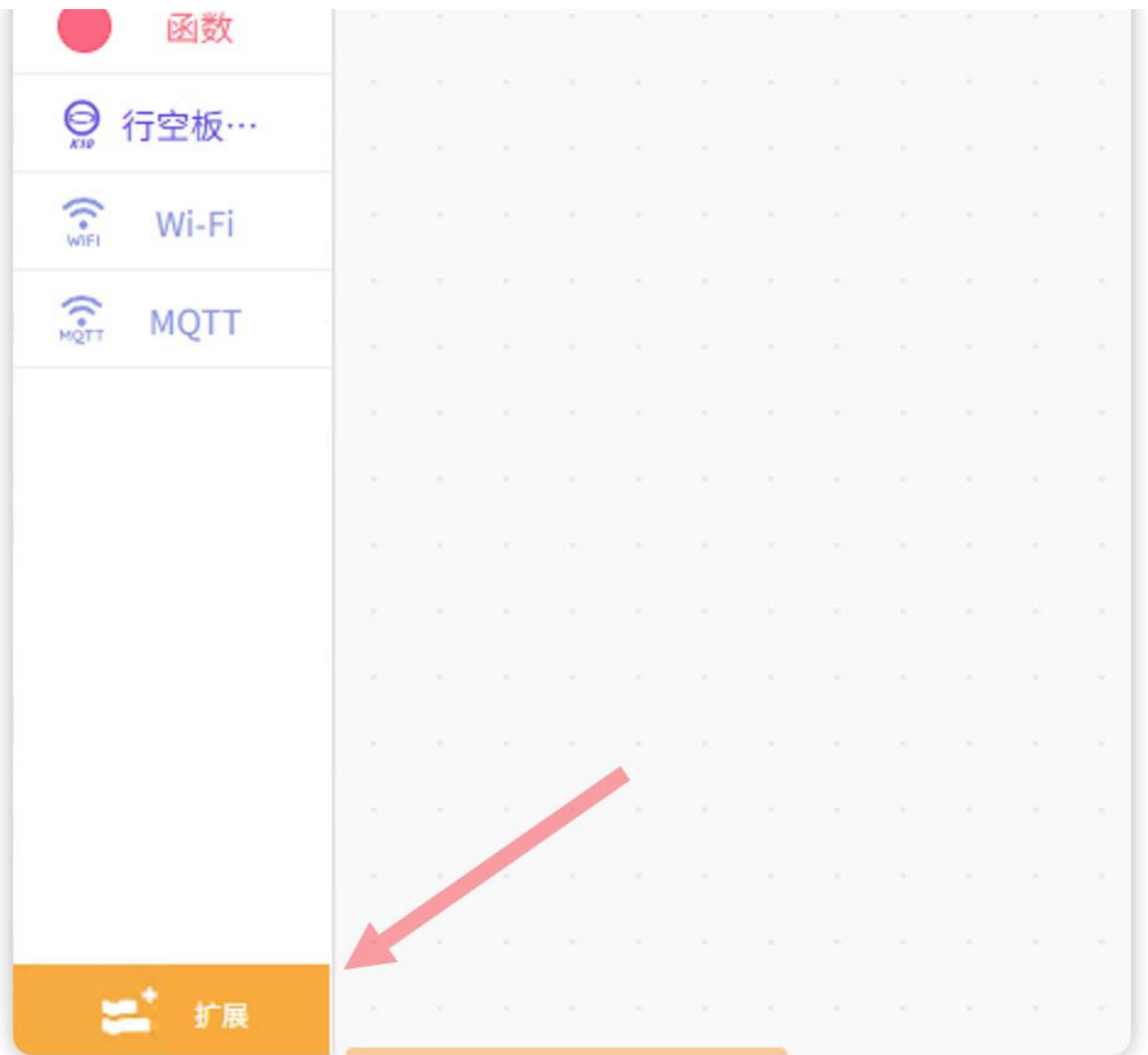
首先，打开Mind+内测版，进入程序设计中的上传模式：



4.2.2 添加主控板和扩展模块

进入上传模式开发环境后，从左下角，进入扩展设置：





先在主控板中添加 行空板K10：



再添加通信模块Wi-Fi和MQTT服务：





4.2.3 添加变量

添加完成后，回到主界面，添加需要的变量：





4.2.4 添加函数

为了方便在主程序中调用，先添加三个函数用于运行环境初始化处理：



- 屏幕初始化：设置屏幕背景颜色，显示数据的名称等
- WiFi连接：用于连接到WiFi网络
- MQTT连接：用于连接到MQTT网络

分别如下：

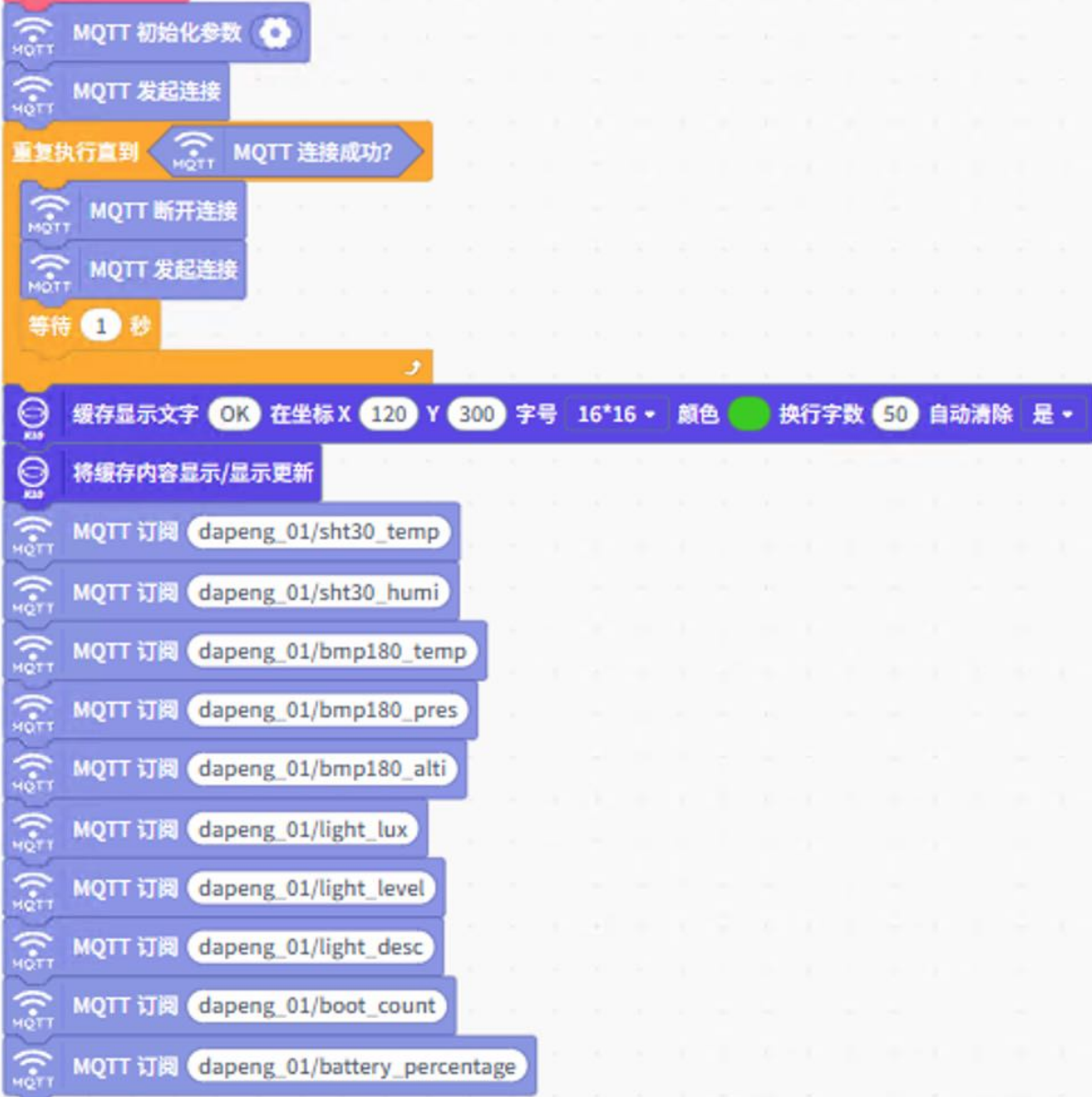
定义 屏幕初始化



定义 WiFi连接



定义 MQTT连接



在MQTT初始化参数功能块上，要点击齿轮图标进行设置：



上述设置，与**环境信息采集**部分代码中的设置保持一致。

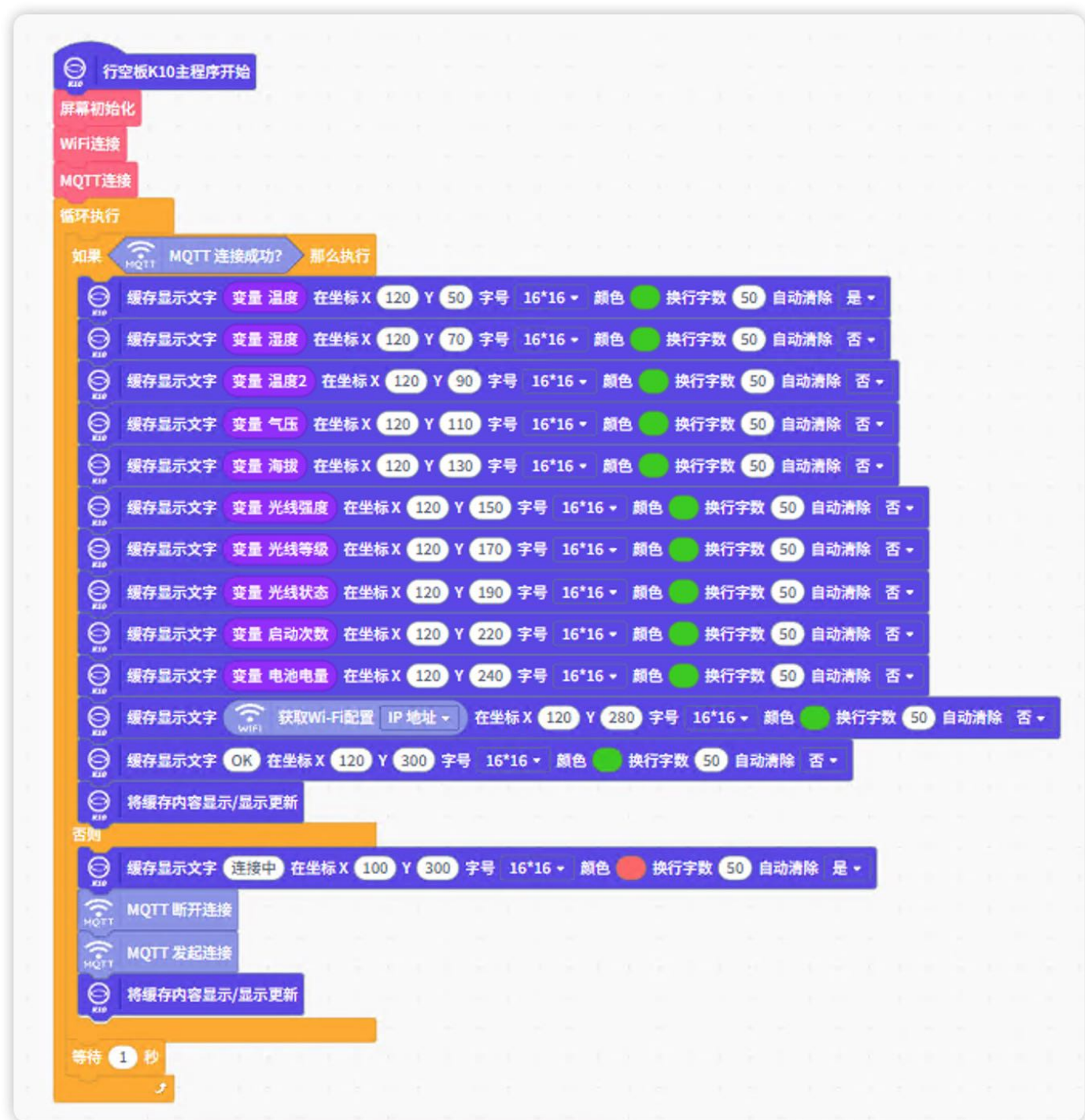
在 **MQTT连接** 函数中，连接成功后，会订阅相关的topic，与**环境信息采集**部分，发布的topic一致。

4.2.5 添加MQTT接收消息处理

当每个topic接收到消息的时候，就把获取到的值，设置到对应的变量之中，以便主程序进行显示：



4.2.6 主程序



在主程序，先进行运行环境的初始化，然后检测MQTT是否连接成功，连接成功的情况下，就将相关的信息，显示到屏幕上即可。

五、系统运行

因为涉及到三个开发板，以及MQTT服务，需要按照一定的顺序启动，才可以正常运行。

5.1 行空板M10

5.1.1 启动M10和Slot服务

将行空板M10，通过USB供电，系统启动后，先到**4-查看网络信息**中查看IP地址(如果没有联网先设置联网)：





上图中的**无线连接**后面的IP地址，就是行空板M10的网络IP地址，需要填写到之前的 **环境信息采集** 和 **环境信息显示** 中对应的MQTT连接IP部分。

然后，在**3-应用开关**中，打开Slot服务：



1-查看使用教程

2-切换运行程序

3-应用开关

4-查看网络信息

5-开关无线热点

6-查看系统信息





退出

Jupyter服务: 已禁用

Slot服务: 已启用

SMB文件共享: 已禁用

VNC/RDP屏幕共享: 已启用

程序开机自启: 已禁用

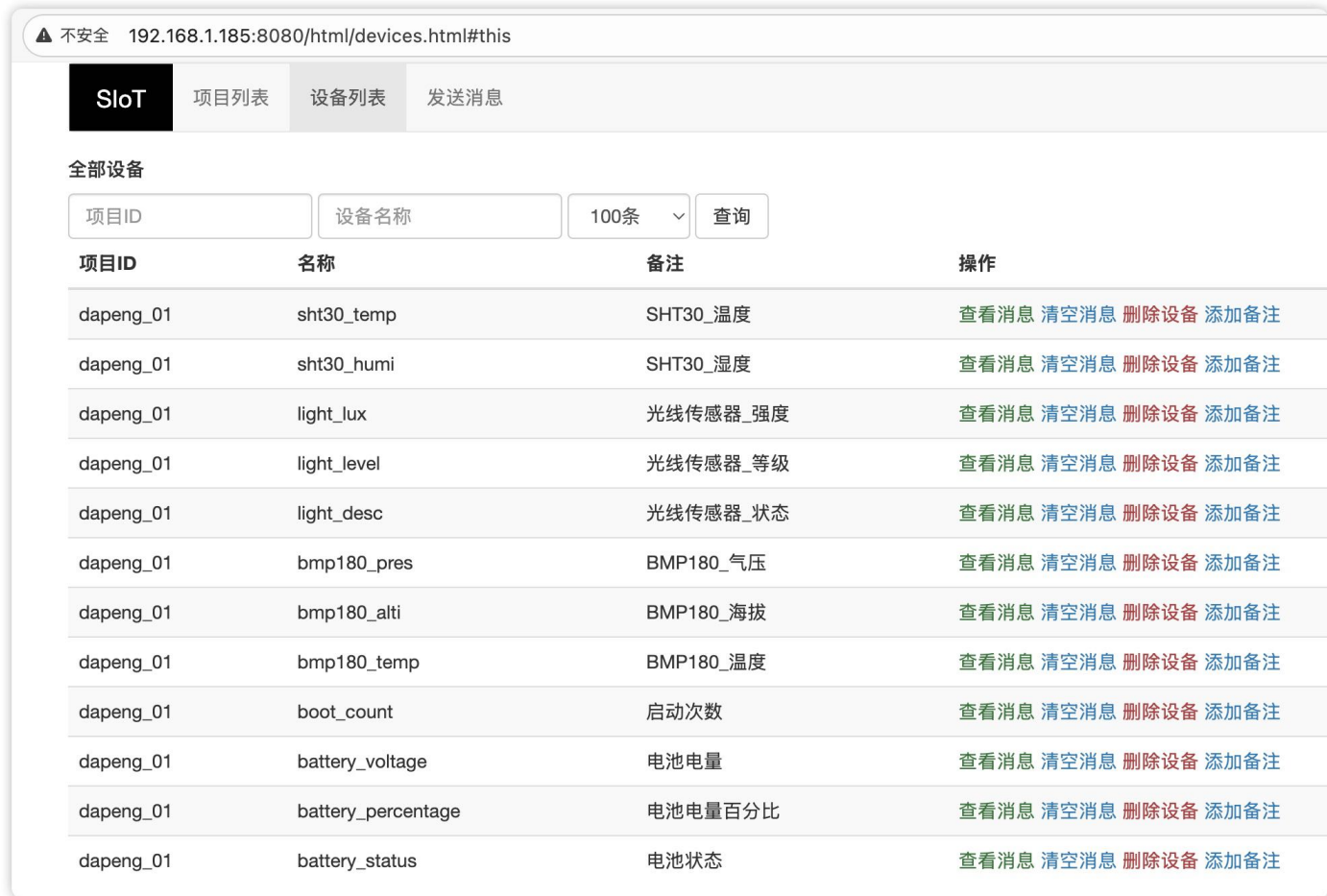


5.1.2 查看Slot管理界面



通过浏览器，访问行空板M10的IP地址，进入应用开关，确定 SloT服务正在运行。

然后，点击打开页面，进入SloT管理界面：



在管理界面中，就可以看到，环境信息采集部分发布的topic和数据。

再点击对应项目的查看消息，可以看到已经收到的数据：

⚠ 不安全 192.168.1.185:8080/html/messages.html?topic=dapeng_01/sht30_temp

SlOt

项目列表

设备列表

发送消息

当前主题 : dapeng_01/sht30_temp

发送消息

消息内容

发送

(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")

开始时间

结束时间

100条

▼

查询

导出查询结果

Topic	消息
dapeng_01/sht30_temp	20.3°C
dapeng_01/sht30_temp	20.3°C
dapeng_01/sht30_temp	20.2°C
dapeng_01/sht30_temp	20.3°C
dapeng_01/sht30_temp	20.3°C
dapeng_01/sht30_temp	20.2°C

注意，如果FireBeetle 2 ESP32-C5还没有正确运行的话，是看不到上述信息的。

5.2 FireBeetle 2 ESP32-C5

当行空板M10上启动SlOt服务后，就可以启动FireBeetle 2 ESP32-C5了。

首先，按照下图设置编译参数：

开发板 : "ESP32C5 Dev Module" >

端口 : "/dev/cu.usbmodem14201" >

Reload Board Data

获得开发板信息

USB CDC On Boot: "Enabled" >

CPU Frequency: "240MHz (WiFi)" >

Core Debug Level: "Info" >

Erase All Flash Before Sketch Upload: "Disabled" >

Flash Frequency: "80MHz" >

Flash Mode: "QIO" >

Flash Size: "4MB (32Mb)" >

JTAG Adapter: "Disabled" >

Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)" >

PSRAM: "Disabled" >

Upload Speed: "921600" >

Zigbee Mode: "Disabled" >

然后编译烧录到开发板，烧录完成后，串口监控会输出如下日志信息：

```
系统启动中...
🔥 正常启动或复位
启动次数: 1

正在连接WiFi...
SSID: OpenBSD
E (2272) wifi:can not get wifi protocol under Wifi band mode WIFI_BAND_MODE_AUTO, please use esp_wifi_get_protocols instead
[ 1900][W][STA.cpp:137] _onStaArduinoEvent(): Reason: 3 - AUTH_LEAVE
[ 1906][W][STA.cpp:543] disconnect(): STA already disconnected.
...
✅ WiFi连接成功!
IP地址: 192.168.1.167

[ 3377][I][esp32-hal-i2c-ng.c:105] i2cInit(): Initializing I2C Master: num=0 sda=9 scl=10 freq=100000
开始扫描I2C设备...
地址范围: 0x01 - 0x7F
发现I2C设备, 地址: 0x44
发现I2C设备, 地址: 0x77
共发现 2 个I2C设备
```

然后，将会检测传感器数据，并发布数据到MQTT服务：

初始化SHT31温湿度传感器...

[3405] [W] [Wire.cpp:296] begin(): Bus already started in Master Mode.

SHT31初始化成功

加热器初始状态: 关闭

初始化BMP180气压传感器...

[3423] [W] [Wire.cpp:296] begin(): Bus already started in Master Mode.

BMP180初始化成功

✅ BOOT按钮已配置为唤醒源

系统初始化完成, 开始监测环境数据...

🌿 我家大棚环境监测系统 - 1号棚

💡 光线: 1610 Lux | 档位: 6 | 强度: 中等偏强

🌡️ SHT31: 温度 19.7℃ | 湿度 57%

📊 BMP180: 温度 17.6℃ | 气压 101.882 kPa | 海拔 50.7米

🔋 电池: 电压 3.32V | 电量 6% | 状态: 电量不足请充电

💤 当前状态: 休眠已启用

正在连接MQTT服务器...✅ MQTT连接成功!

✅ 数据成功发布到MQTT

连续发布三次后, 将会进入休眠状态:

🌿 我家大棚环境监测系统 - 1号棚

💡 光线: 1604 Lux | 档位: 6 | 强度: 中等偏强

🌡️ SHT31: 温度 19.8℃ | 湿度 57%

📊 BMP180: 温度 17.6℃ | 气压 101.881 kPa | 海拔 51.1米

🔋 电池: 电压 3.32V | 电量 6% | 状态: 电量不足请充电

💤 当前状态: 休眠已启用

✅ 数据成功发布到MQTT

进入休眠状态, 60秒后自动唤醒

长按BOOT按钮可取消休眠

当指定的休眠时间到达后, 就会自动唤醒, 重新输出日志信息:

系统启动中...

🕒 由定时器唤醒

启动次数: 2

正在连接WiFi...

SSID: OpenBSD

...

✅ WiFi连接成功!

IP地址: 192.168.1.167

此时，如果查看 SIoT 管理界面对应的项目，就可以看到发布的数据了。

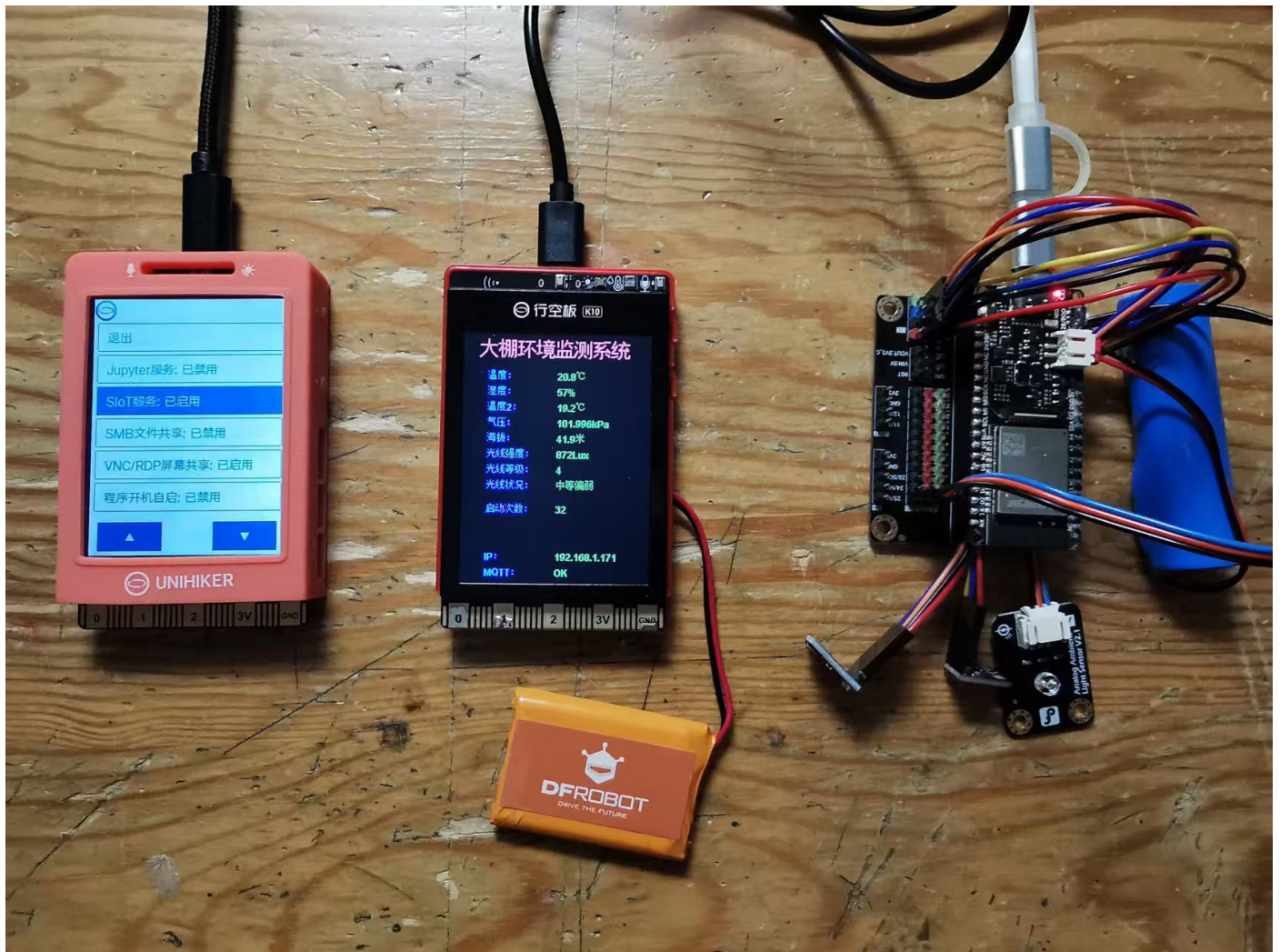
5.3 行空板K10

在Mind+中，将程序下载到行空板K10，下载成功运行后，行空板K10显示如下界面：

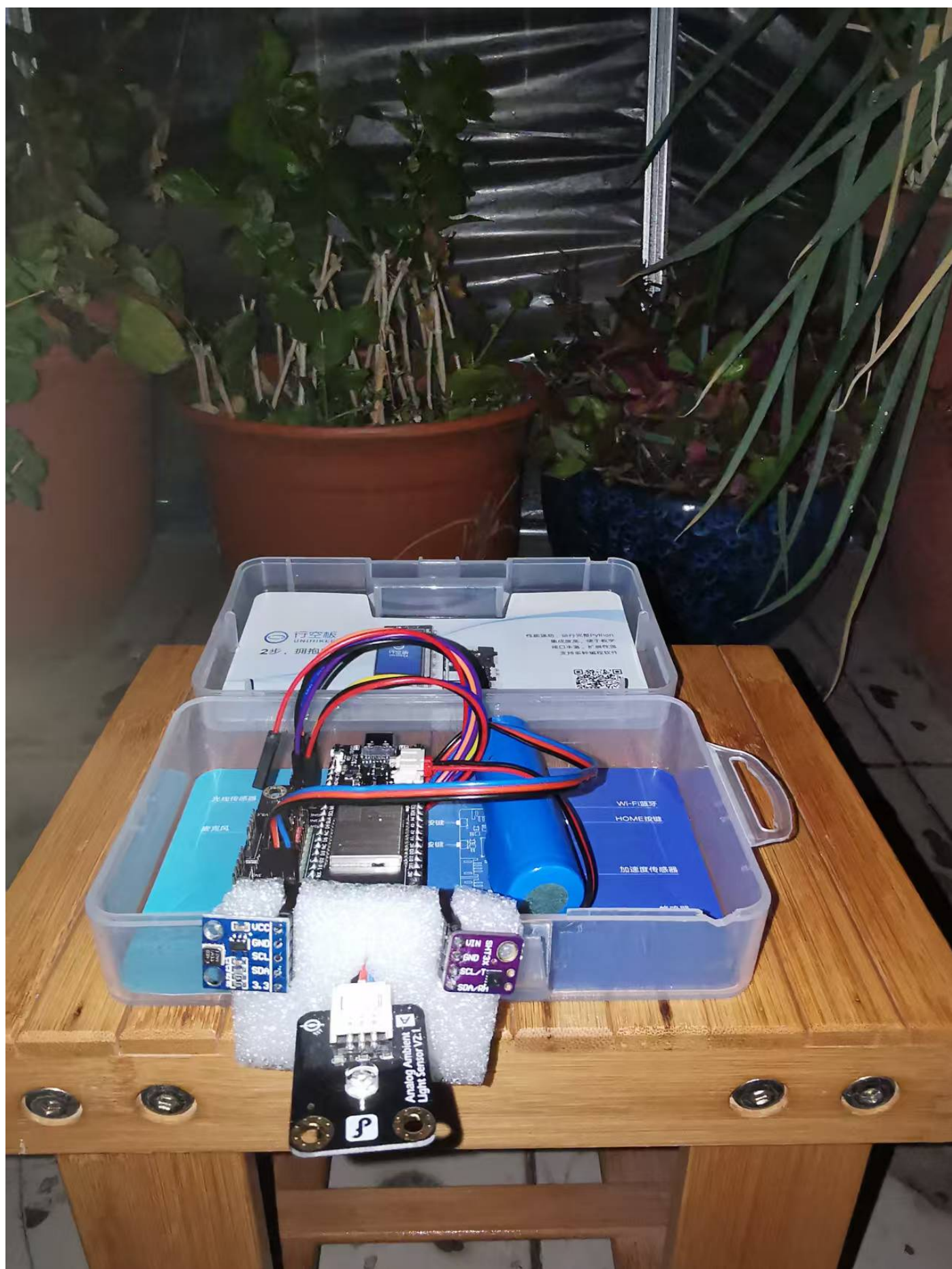


六、运行效果

全部调试运行以后，整体的运行效果如下：



现在，就可以把 FireBeetle 2 ESP32-C5 部署到大棚之中了：



拿出行空板K10，就能随时随地查看信息了：

大棚环境监测系统

温度:	14.0℃
湿度:	78%
温度2:	11.8℃
气压:	102.006kPa
海拔:	40.3米
光线强度:	9Lux
光线等级:	0
光线状况:	完全黑暗
启动次数:	11

IP:	192.168.1.171
MQTT:	OK

最新版本中，还添加了电池检测功能，能够随时查看查看FireBeetle 2 ESP32-C5所用锂电池的电量：



之前测试的时候，没有给锂电池充满电。充满电后，实际使用发现，过了4个小时，电量还有96%。

七、后续优化

现在，大棚环境监测系统已经能够稳定的运行起来了。

后续将继续进行优化，主要包括以下方面：

- 外壳设计：需要制作一个外壳，放置FireBeetle 2 ESP32-C5和传感器，并做好防水处理。
- 太阳能供电集成：将太阳能电池板给接上，就能全天候不间断运行了。
- 功耗和续航：需要测试FireBeetle 2 ESP32-C5的功耗数据，以及测试FireBeetle 2 ESP32-C5的实际续航时间。
- 报警功能添加：当环境温度出现异常时，及时报警。
- 远程控制功能：准备控制加热设备，自动加热，保持大棚里面的温度，不至于太低，让花花草草可以安然过冬。