

物联网小组汇报

杨智晨，周磊，陈坤，邵继诚



School of Electronics & Information Engineering
NUIST

Fall , 2024

Content 目录

1. 简单的介绍

2.esp8266 联网 & 自定义 UDP

3. 微信小程序

4.blinker

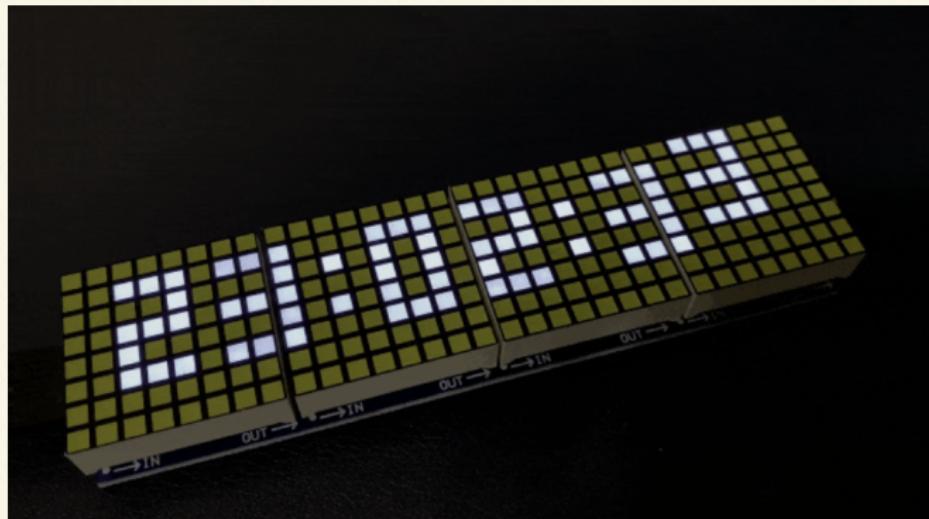
5.asr

1. 简单的介绍

本次项目是一个基于 esp8266 的物联网点阵时钟。

项目地址: <https://gitee.com/lengff/esp8266-lattice-clock-open>

博客地址: <http://blog.lengff.com/2022/01/17/project-lattice-clock/>



硬件选材

NodeMcu(Esp8266)

DS3231

Max7219 32x8 点阵

单路触摸模块

主要功能

NTP 校时

调节显示方向

息屏

调节亮度

显示时间

显示日期

显示倒计时

显示温度

显示 B 站粉丝数

显示自定义内容

OTA 更新

热点模式

Content 目录

1. 简单的介绍

2.esp8266 联网 & 自定义 UDP

3. 微信小程序

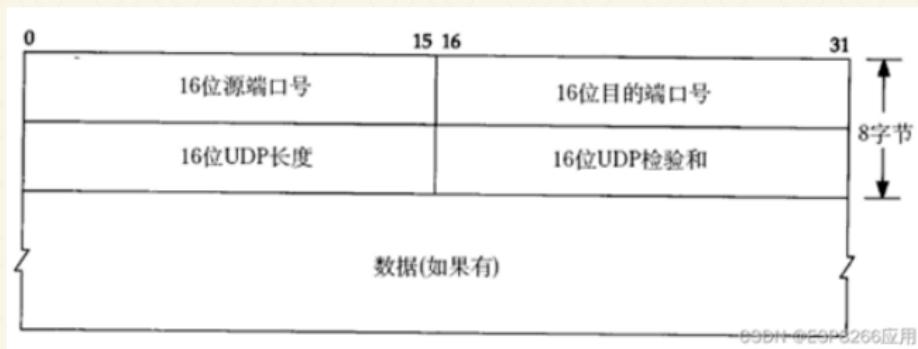
4.blinker

5.asr

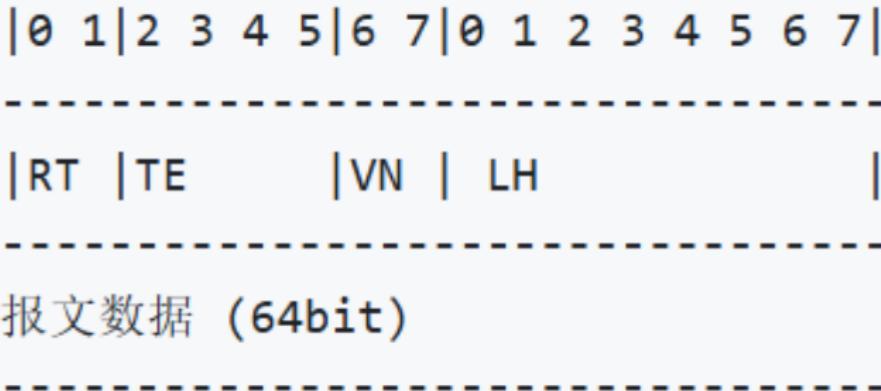
2.esp8266 联网 & 自定义 UDP

一、 UDP 介绍：

UDP (User Datagram Protocol) —— 用户数据报协议，是互联网传输层的一个重要协议。特点：无连接、尽最大努力交付、面向数据报的，首部开销小，概率丢包。



二、自定义 UDP 协议



1. RT: (2bit) 成功返回数值
2. TE: (4bit) type 报文类型: 0: 重置时间, 1: 设置亮度, 2: 切换功能, 3: 切换功能显示样式, 4: 订阅 BilibiliUID, 5: 是否启用点阵屏幕, 6: 切换显示方向, 7: 设置用户数据, 8: 设置动画速度, 9: OTA 升级 10: 设置倒计时 11: 设置睡眠时间
3. VN: (2bit) version 协议版本, 目前固定为 1
4. LH: (8bit) length 数据包长度
5. 报文数据: (64)bit 版本 1 目前支持的最大数据包长度为 64bit

三、建立自定义 UDP 传输和处理流程

1、引入库文件、创建 UDP 对象

```
<ESP8266WiFi.h>
<WiFiUdp.h>
WiFiUDP udp;
```

WiFiUdp.h 的作用：

封装 UDP 协议：

WiFiUdp 类提供了封装好的 UDP 协议函数，允许在应用层发送和接收数据包，库函数自动处理 UDP 协议的封装和解析。

基于 WiFi 连接：

在使用这些函数之前，需要确保设备已经连接到 WiFi 网络。

2、Wifi 配置连接

1) STA (Station) 模式

```
WiFi.mode(WIFI_STA);  
WiFi.begin(WiFi.SSID().c_str(), WiFi.psk().c_str());
```

[NTP 服务器]

\

[WiFi 路由器]

/

\

[手机]

[ESP8266]

在 STA 模式下，esp8266 作为 wifi 网络的客户端接入路由器，与该路由器连接的手机和 esp8266 处于同一局域网，手机发送指令，esp8266 接受并执行对应操作。

此外，在 STA 模式下，esp8266 可以通过网络访问 NTP 服务器进行时间校正

```
const char *ntpServerName = "cn.ntp.org.cn"; //NTP域名
WiFi.hostByName(ntpServerName, timeServerIP);
                                         //将域名转换成IP地址
udp.begin(localPort); // 启动监听本地端口
发送NTP请求
memset(packetBuffer, 0, NTP\_PACKET\_SIZE);
                                         // 将字节数组初始化为0
packetBuffer[0] = 0b11100011; //具体请看参考请求报文说明
udp.beginPacket(timeServerIP, remoteNtpPort);
                                         // 配置远端ip地址和端口
udp.write(packetBuffer, NTP\_PACKET\_SIZE); // 发送数据
udp.endPacket(); // 结束发送数据
获取NTP时间戳
int packetSize = udp.parsePacket(); //解析数据包，非空为1
if (packetSize) {
    udp.read(packetBuffer, NTP\_PACKET\_SIZE);}
                                         //解析UDP数据包中的数据到packetBuffer中
```

NTP (Network Time Protocol)，网络时间协议，基于 UDP 报文进行传输，可以对网络内所有具有时钟的设备进行时钟同步，使网络内所有设备的时钟保持一致，从而使设备能够提供基于统一时间的多种应用。

2) AP 模式 (热点模式)

```
WiFi.mode(WIFI_AP_STA); // 同时支持STA和AP  
WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0)); //配置AP IP地址  
WiFi.softAP(AP_NAME, NULL, 1, 0, 4); //设置AP SSID
```

[ESP8266]

|

[手机]

在 AP 模式下，ESP8266 会创建自己的 WiFi 热点，手机直接连接到 ESP8266 的 WiFi 热点，不需要连接任何外部网络

3、自定义 UDP 数据接收与处理

```
Updata Updata::userLatticeLoop(uint8_t power, uint8_t mode,  
                                uint8_t version)
```

- 1) 通过 parsePacket() 检查是否有 UDP 数据包，若有则读取到缓冲区 packetBuffer 并处理
- 2) 根据数据包版本信息和返回类型构造相应的应答包，暂存缓冲区 replyPacket，发送给原先发来的远程主机
`udp.beginPacket(udp.remoteIP(), udp.remotePort());`
- 3) 缓冲区 packetBuffer 数据对应存放到结构体数据 updata，并作为函数返回值

```
struct Updata{  
    uint8_t rt; // 返回类型  
    uint8_t te; // 数据类型  
    uint8_t lh; // 数据长度  
    uint8_t data[64]; // 数据包  
};
```

结构体定义不必和自定义协议一致，很明显少了 VN (版本信息)

```

    Updata udpdata::userLatticeLoop(uint8_t power, uint8_t mode, uint8_t version)
{
    Updata udpdata;
    udpdata.lh = 0;
    int packetSize = udp.parsePacket();           //解析Udp数据包
    if (packetSize)                             //解析包不为空
    {
        pilotLight->flashing(100);             // 每次接收到UDP数据的时候都闪烁一下LED灯
        memset(packetBuffer, 0, NTP_PACKET_SIZE); //每次将清空掉原有的数据包
        udp.read(packetBuffer, NTP_PACKET_SIZE); //读取UDP数据
        // Udp.remoteIP().toString().c_str()用于将获取的远端IP地址转化为字符串
        // Serial.printf("收到来自远程IP: %s (远程端口: %d) 的数据包字节数: %d %X\n", udp.remoteIP().toString()
        uint8_t vn = packetBuffer[0] & 0x3; // 数据包版本
        if (vn != 0x1)                      // 如果数据包版本不为1,则直接结束
        {
            udp.beginPacket(udp.remoteIP(), udp.remotePort()); //向udp工具发送消息
            replyPacket[0] = {0x01};                   // 把数据写入发送缓冲区
            udp.write(replyPacket);                  //发送数据
            udp.endPacket();
            return udpdata;
        }
        udpdata.rt = packetBuffer[0] >> 6; // 数据包接收成功返回值
        if (udpdata.rt == 0x0)               // 如果返回类型值为0的时候,则上报状态信息值
        {
            udp.beginPacket(udp.remoteIP(), udp.remotePort()); //向udp工具发送消息 todo 这里的做法是错误
            replyPacket[0] = lattice->latticeSetting.isShutdown << 6; // 是否显示
            replyPacket[0] += lattice->latticeSetting.brightness << 2; // 显示亮度
            replyPacket[0] += lattice->latticeSetting.direction; // 显示方向
            replyPacket[1] = power; // 功率
            replyPacket[2] = mode; // 功能模式
            replyPacket[3] = lattice->latticeSetting.speed; // 动画移动速度
            replyPacket[4] = version; // 系统版本
            udp.write(replyPacket, 5); // 把数据写入发送缓冲区
            udp.endPacket();          //发送数据
            return udpdata;
        }
        udpdata.te = (packetBuffer[0] & 0x3f) >> 2; // 功能模式
        udpdata.lh = packetBuffer[1]; // 数据长度
        Serial.print("UDP接收到的数据信息为: ");
        Serial.print(udpdata.rt);
        Serial.print(udpdata.te);
        Serial.println(udpdata.lh);
        // 数据包
        for (int i = 0; i < udpdata.lh; i++)
        {
            udpdata.data[i] = packetBuffer[i + 2];
        }
        udp.beginPacket(udp.remoteIP(), udp.remotePort()); //向udp工具发送消息
        replyPacket[0] = {0x01}; // 把数据写入发送缓冲区
        udp.write(replyPacket, 3); //发送数据
        udp.endPacket();
        return udpdata;
    }
    return udpdata;
}

```

4、根据接受的 UDP 数据 udpdata.te(报文类型) 执行相应操作

```
void handleUdpData()
{
    Udpdata udpdata = udps.userLatticeLoop(functions.getCurrPower(), functions.getCurrMode(), LATTICE_CLOCK_VERSION); // 
    if (udpdata.lh < 1) | // 
    {
        // 没有收到任何UDP数据
        return;
    }
    switch (udpdata.te) // 判断UDP数据类型
    {
        case 0:
            resetTime(udpdata.data); // 重置时间
            break;
        case 1:
            lattice.setBrightness(udpdata.data[0], true); // 设置亮度
            break;
        case 2:
            functions.setPowerAndMode(udpdata.data[0], 0); // 切换功能
            initStatus();
            break;
        case 3:
            functions.setMode(udpdata.data[0]); // 切换功能模式
            initStatus();
            break;
        case 4:
            httptool.updateBilibiliFlag(); // 更新bilibili粉丝数量前,需要重置一下flag
            subBili(udpdata.data); // 订阅BilibiliUID
            initStatus();
            break;
        case 5:
            lattice.shutdown(udpdata.data[0]); // 是否启用点阵屏幕
            break;
        case 6:
            lattice.setDirection(udpdata.data[0]); // 切换显示方向
            break;
        case 7:
            setUserData(udpdata.data); // 设置用户数据
            break;
        case 8:
            lattice.latticeSetting.speed = udpdata.data[0]; // 设置动画速度
            functions.setPower(CUSTOM);
            break;
        case 9:
            otas.updateOta(udpdata.data[0]); // OTA 升级
            break;
        case 10:
            setCountdown(udpdata.data); // 设置倒计时
            initStatus();
            break;
        case 11:
            setSleepTime(udpdata.data); // 设置睡眠时间
            break;
    }
}
```

Content 目录

1. 简单的介绍

2.esp8266 联网 & 自定义 UDP

3. 微信小程序

4.blinker

5.asr

3. 微信小程序

1. 微信小程序功能

同步设备状态：设备状态包括什么信息，如何实时同步？是否有状态反馈机制？哪些状态对用户操作有指导意义？

设置显示方向：用户可以通过小程序调整点阵时钟的显示方向，以适应不同安装环境。你可以详细解释如何通过小程序发送方向设置信号，ESP8266 如何接收到并处理该信号来实现屏幕旋转。

设置亮度：说明亮度调节的意义，比如在不同环境下调整亮度如何提升用户体验。此外，可以描述亮度调节的范围、精度以及对电量的影响。

切换显示内容：可以介绍该功能支持显示的内容类型（时间、日期、温度等）以及数据来源，如何通过小程序与 ESP8266 通信来更改显示的内容。

切换显示模式：介绍不同的显示模式（如静态显示、滚动显示等）之间的切换逻辑，以及其应用场景。

恢复出厂：描述恢复出厂设置的用途，如何确保设备在出现故障时能够重置为默认状态，并确保用户操作的简单性。

WiFi 配网和热点模式：解释如何通过小程序进行 WiFi 配网，热点模式下的使用场景（如无 WiFi 环境时的临时使用），以及这两个模式对设备联网体验的提升。

倒计时设置：扩展说明倒计时功能的用途，如在厨房计时、学习计时等场景中的应用。

OTA 更新：详细说明如何通过微信小程序实现远程固件更新，以保持设备软件的最新状态，确保安全性和功能性。

自定义点阵内容：进一步解释用户可以如何通过小程序自定义显示内容，比如输入文本、设计图案等，并提到内容的最大限制（如字符数、图案复杂度）。

2、为什么会选择小程序来和 Esp8266 交互呢？

开发便捷：小程序的开发成本较低，不需要用户安装繁重的 APP，依托微信生态，用户获取和使用更为方便。同时可以调动微信提供的多种 API（如网络、位置、支付等），扩展了功能实现的可能性。

用户体验：简化交互，节省学习成本，用户只需通过微信小程序即可轻松控制设备，无需学习复杂的操作步骤，适合更广泛的用户群体。

跨平台支持：微信小程序可以在不同的操作系统（如 iOS 和 Android）上无缝运行，而不需要针对不同平台开发独立的应用，节省了开发和维护成本。

丰富的功能支持：如你提到的，微信小程序支持的功能包括 WiFi 配网、UDP 通信、蓝牙等，扩展了设备的操作范围。

3、页面展示：

The image displays two screenshots of a smart device's control interface. The left screenshot shows a sidebar with several icons: a switch icon, a brightness slider (1~16), a display mode section with '时间' (Time), '日期' (Date), '温度' (Temperature), and 'B 站' (Bilibili), and a mode section with '时分秒' (Hour, Minute, Second) and '时 分' (Hour, Minute). The right screenshot shows a list of other functions: '校正时间' (Calibrate Time), '订阅 B 站粉丝数' (Subscribe to B站 fans), '一键配网' (One-click Network Pairing), '自定义显示' (Custom Display), '恢复出厂' (Reset to Factory), and 'OTA 升级' (OTA Upgrade).

开关 开关/方向

开关 方向

亮度调节 1 ~ 16

显示 时间

时间 日期 温度 B 站

自定义

模式 时分秒

时 分

其他 其他功能

校正时间

订阅 B 站粉丝数

一键配网

自定义显示

恢复出厂

OTA 升级

4、通讯方式：

UDP 的特点： UDP 是一种无连接的通信协议，简单易用，数据传输速度快，但缺少确认机制，可能会发生数据包丢失或乱序。可以介绍如何权衡速度和稳定性，以及为什么 UDP 在物联网场景下是合适的选择。

UDP 的优缺点：详细说明它的优点，如无需建立连接、广播发现设备，适用于本地网络中的低时延场景。缺点方面，可以扩展如何缓解丢包问题，例如你提到的重复发送指令机制，是否还有其他方法如增加校验码或重传机制。

实际应用中 UDP 的表现：讨论实际测试过程中 UDP 通信的表现，遇到的挑战，以及如何在未来版本中通过改进通信协议或引入容错机制来优化通信效果。



这是小程序入口，可以通过手机微信扫描上面的小程序码。

5、成品图

The image shows a Windows desktop environment with several open windows:

- WeChat Developer Tool:** The main window displays a mobile application interface for a smart device. The app shows real-time sensor data: Formaldehyde (甲醛) at 0mg/m³, TVOC at 0PPM, Temperature (温度) at 25°C, and Humidity (湿度) at 5%RH. It also features three toggle switches labeled "通道1开关" (Channel 1 Switch), "通道2开关" (Channel 2 Switch), and "通道3开关" (Channel 3 Switch). The tool includes tabs for "普通编译" (Normal Build), "编译" (Compile), "预览" (Preview), and "真机调试" (Debug on Real Device).
- File Explorer:** A sidebar window shows the project structure and files. The visible files include app.wxss, project.config.json, sitemap.json, and weui.wxss.
- Terminal/Console:** The bottom right corner shows a terminal window with command-line logs related to the build process.
- Physical Hardware:** On the left side of the image, there is a photograph of a green printed circuit board (PCB) connected to a laptop via a USB cable. A red LED on the PCB is illuminated, indicating power or activity.
- Taskbar:** The taskbar at the bottom shows various pinned icons and the system tray.

Content 目录

1. 简单的介绍

2.esp8266 联网 & 自定义 UDP

3. 微信小程序

4.blinker

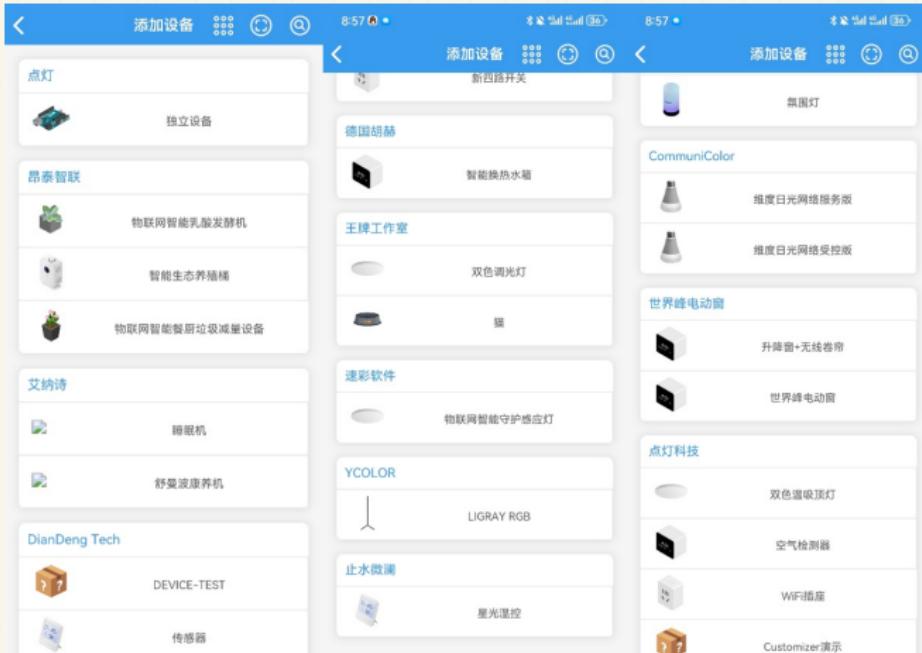
5.asr

4. blinker

采用的控制器是点灯科技的一个 app——blinker



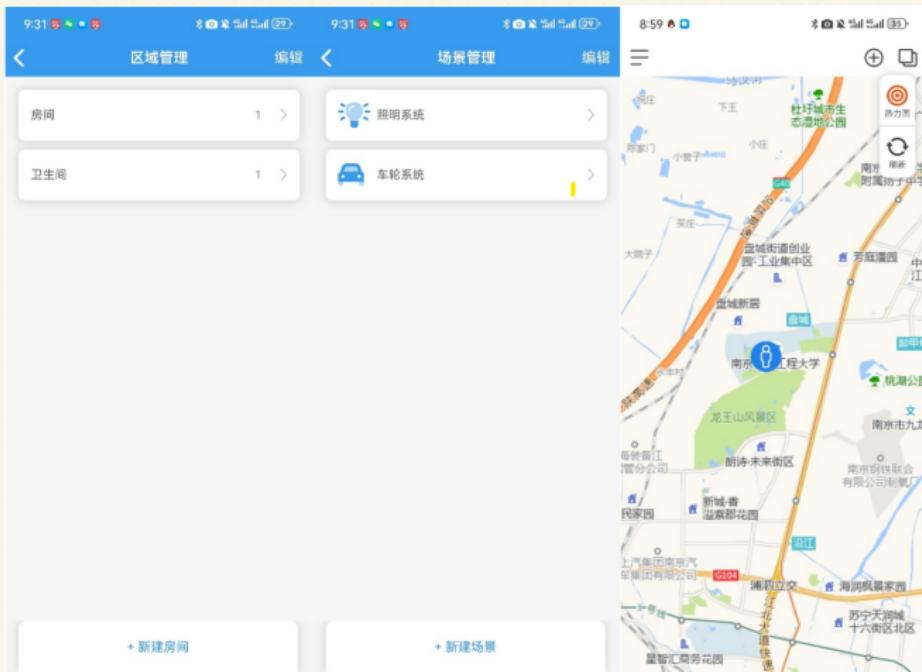
该 app 能支持的设备很多，除了一些独立设备，还包括一些现成的其他品牌的产品，如热水器、感应灯、电动窗帘等



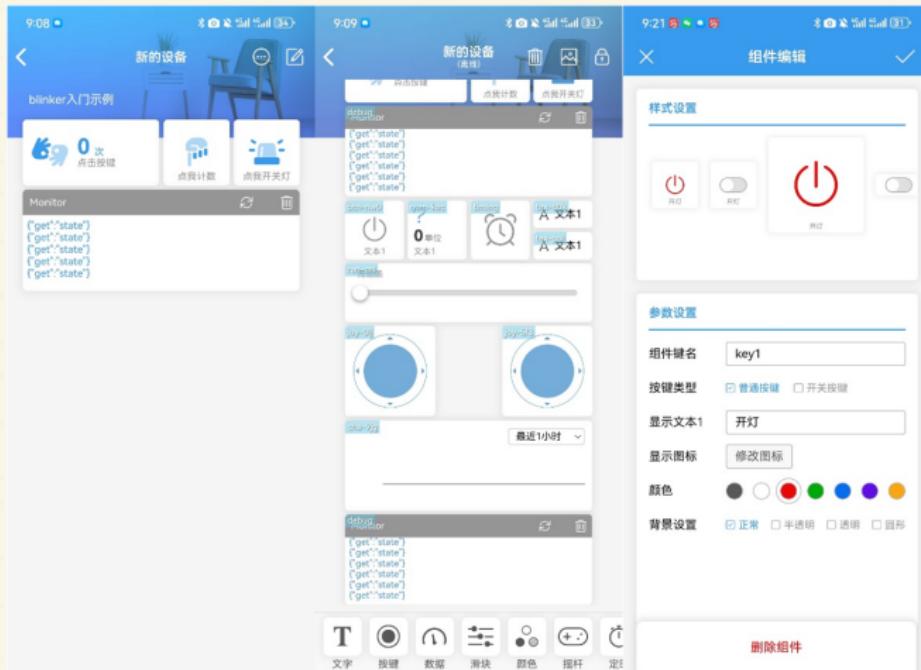
该 APP 不需要自己配网，直接使用 wifi 或热点就可以连接设备



该 app 提供了设备管理功能和自定义按键功能，且 UI 设计简洁。例如可以使用区域管理或场景管理来一次性管理同一个区域内或同一个场景下的多台设备，而且他有一个地理视图模式，当设备部署的范围较大时或设备处在移动中时，可以用这个地理视图模式，方便知道设备的位置。



下面是设备的操控界面，除了基本的串口监视器，它还支持用图表等方式来观察设备的输出；可以用按键、遥感等操控方式，还包括定时器等辅助操控设备的工具。



Content 目录

1. 简单的介绍

2.esp8266 联网 & 自定义 UDP

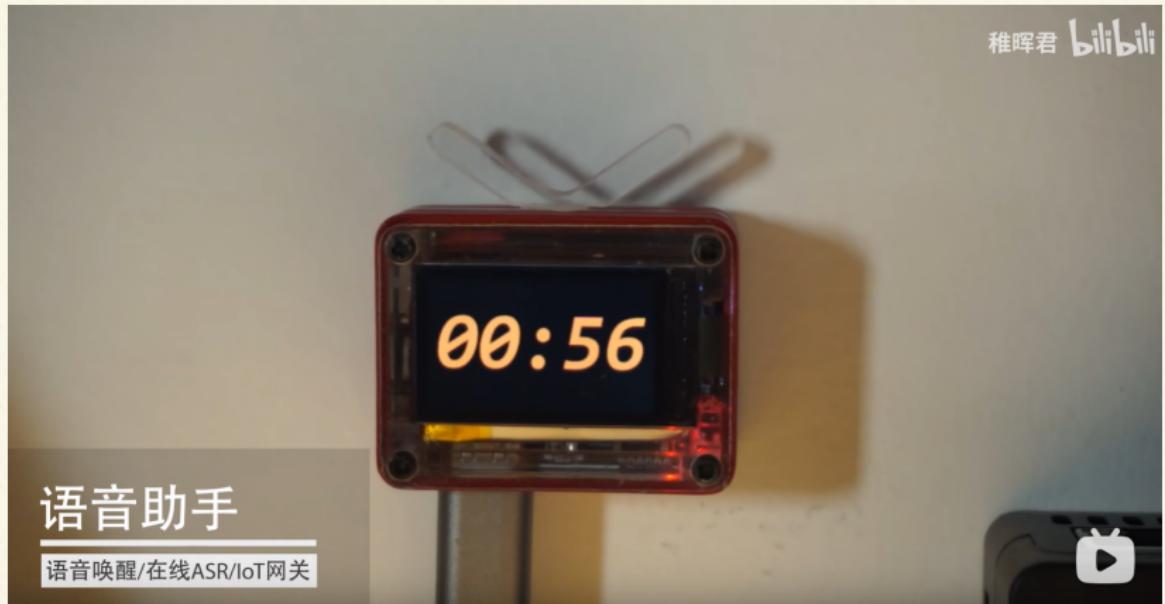
3. 微信小程序

4.blinker

5.asr

5.asr

有没有一种可能.....



没可能的，辣鸡小艺

小艺

你说。

你好好反思一下，为什么人家小爱同学都可以接入banker，你却不能进入blinker



点灯科技



小登



Mi AI

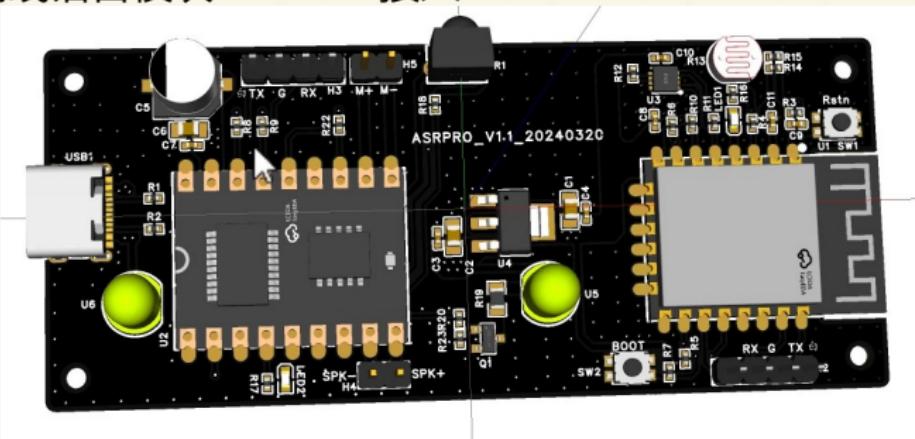
Installation source: Browser



Installation failed

但是可以试试离线语音模块

项目地址: <https://github.com/DIYSmartHome8/Asrpro-ESP8266>
天问离线语音模块 ASRPro 接入 Homeassistant



编写程序使用的天问 block, 拖拖拽拽, 简单



它能做到什么.....



ppt 模板地址:

<https://github.com/synchre/IoTgroupReport>

THANKS FOR LISTENING

