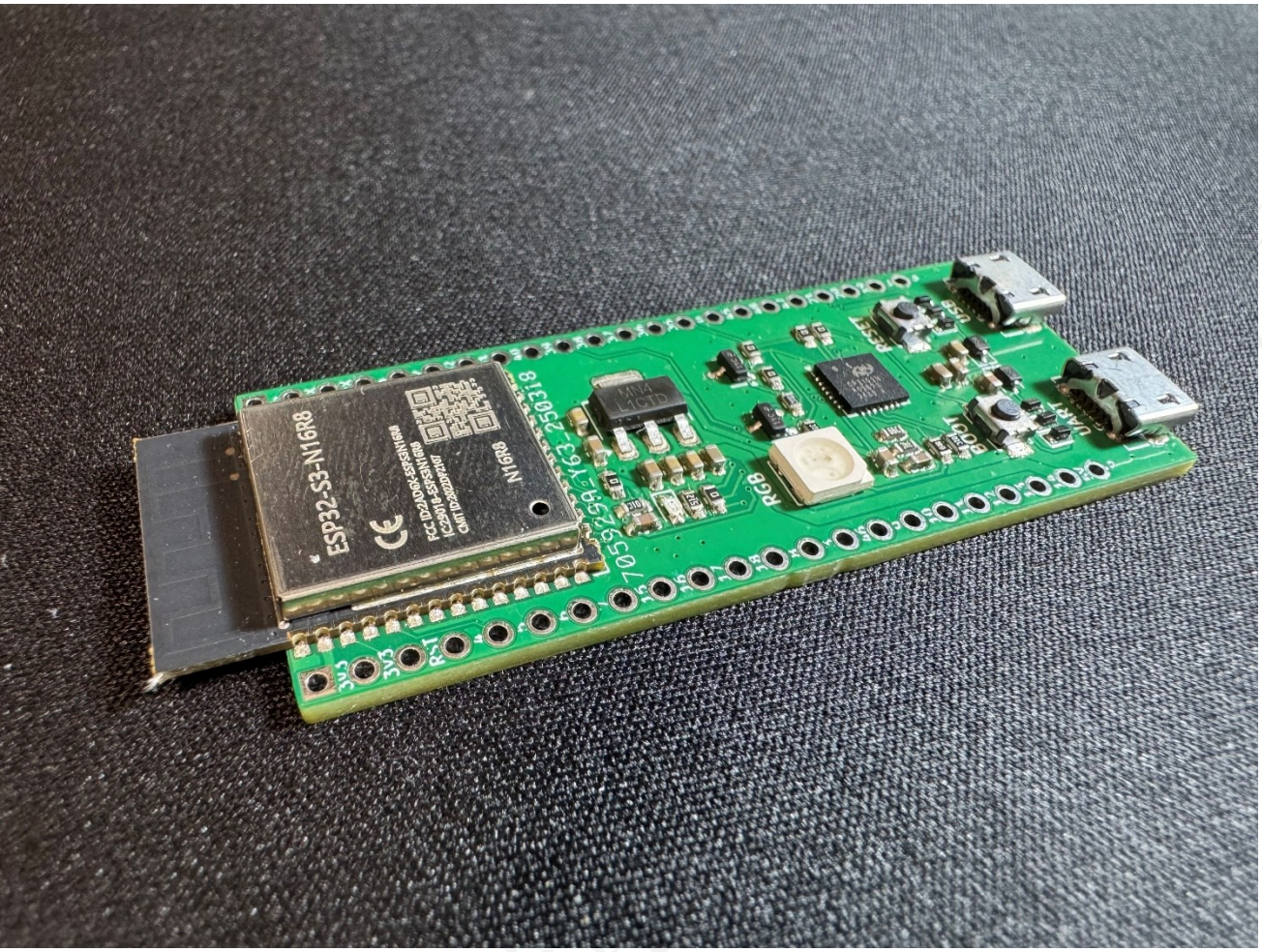
2025



南京宇微电子科技有限公司

ESP32-S3-Mini 开发板 V1.0

2025 年 10 月 22 日修订

作者：付坤

目录

版本修订

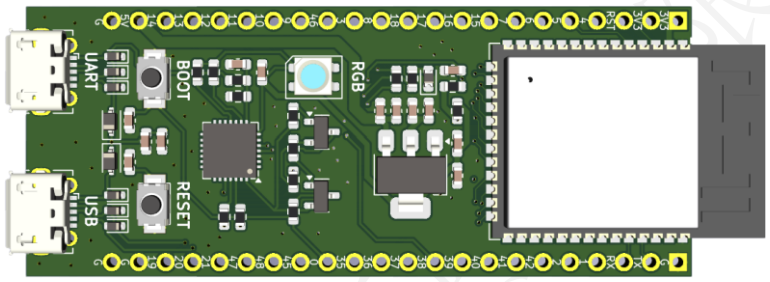
客户须知

本文档为产品使用参考所编写，文档版本可能随时更新，恕不另行通知。本文中提供的所有使用方法、说明及建议仅供参考，不构成任何承诺或保证。使用本产品及本文档内容所产生的结果，由用户自行承担风险。本公司对因使用本文档或产品而导致的任何直接或间接损失，不承担任何责任。

一、概览

ESP32-S3-DevKitC-1 是 Espressif（乐鑫）推出的 ESP32-S3 系列开发板，专为 AIoT 和边缘计算设计，我们根据其开源的资料复刻了这块开发板。核心采用 ESP32-S3-N16R8 模组，搭载双核 Xtensa LX7 处理器（主频最高 240MHz）、512KB SRAM、8MB PSRAM、16MB Flash。支持 2.4GHz Wi-Fi（802.11 b/g/n）和蓝牙 5（LE），新增 AI 加速引擎（向量指令）和 USB OTG 接口，提升机器学习任务效率。

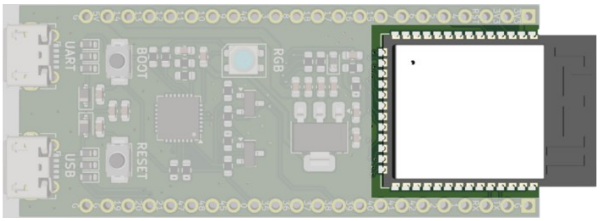
板载 36 个 GPIO，支持丰富外设，体积紧凑（约 69×25.5mm），支持 Arduino IDE、ESP-IDF、MicroPython 等环境。通过 MicroUSB 接口连接 CP2102 USB 转串口芯片，通过 MicroUSB 接口直接连接 ESP32 模块的 USB OTG 接口，均可用于对系统供电。双排 2×22 针脚设计，22.86mm 间距兼容面包板，板载复位/引导按键和 RGB LED。



参数	
核心主控	ESP32-S3
天线规格	2.4Ghz ISM 波段 PIFA 天线
输入电压	3.3~5.5V
引脚数量	2×22, 22.86mm 间距
引脚间距	2.54mm
板载外设	USB 转串口、256 级 LED
物理接口	MicroUSB 接口、杜邦插针
无线接口	WIFI/BLE
工作温度	-40℃~+85℃
存储温度	-40℃~+85℃

二、板卡分区介绍

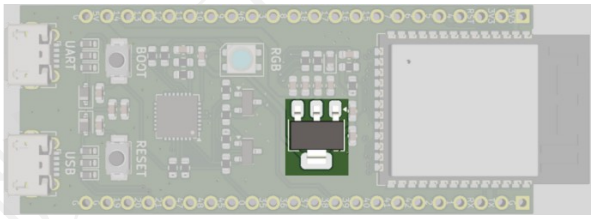
2.1 ESP32-S3 模组



ESP32-S3 模组是基于 Espressif（乐鑫）推出的高性能 Wi-Fi + Bluetooth LE MCU，专为 AIoT（AI + 物联网）的应用而设计。核心搭载 ESP32-S3R8 芯片，采用双核 Xtensa 32 位 LX7 处理器（主频高达 240 MHz），内置 512 KB SRAM 和 384 KB ROM，支持神经网络计算和信号处理加速（如向量指令）。

- 无线能力：2.4 GHz Wi-Fi（802.11 b/g/n）和 Bluetooth 5（LE），集成 PCB 天线，传输速率高达 150 Mbps。
- 硬件规格：尺寸 18 × 25.5 × 3.1 mm，供电 3.0–3.6 V，工作温度 –40 ~ +85 °C。安全特性包括 RSA 安全启动、AES-XTS Flash 加密、数字签名和 HMAC 外设。

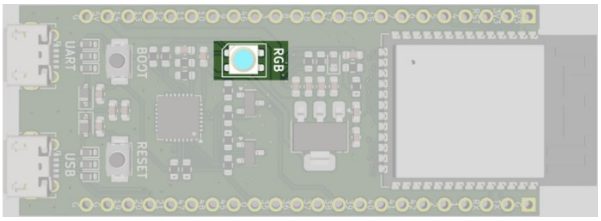
2.2 电源



通过 MicroUSB 接口供电，标识为 UART 和 USB 的 USB 接口均可接受 5V 直流供电，经过板载 5V 转 3.3V LDO 稳压后供给模块和板载其它外设。两个 MicroUSB 接口的输入电源 VBUS 经由一对防倒灌二极管连接，因此可以同时连接这两个 MicrosUSB 接口使用，而不必担心电流倒灌问题。

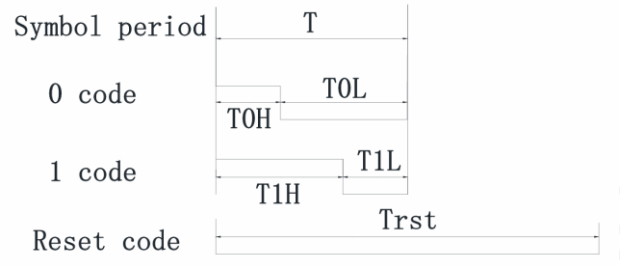
开发板两侧接口有 5V 标识的，约 4.4V 可供取电，同时也可作为 5V 的输入端口。开发板两侧接口有 3.3V 标识的，约 3.3V 可供取电，同时也可以作为 3.3V 的输入端口。

2.3 LED



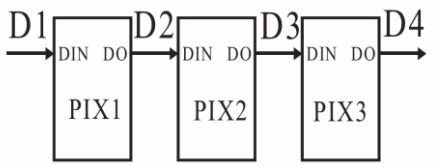
SK68XXMINI-HS 是由 P&A（OPSC）公司推出的智能可寻址 SMD LED 芯片，形状为 3535（3.5×3.7×1.95 mm），功率 0.1/0.2W，MSL 5a 级别。集成 HD107S IC 驱动电路，支持数字 RGB 控制（24 位灰度，每通道 256 级），工作电压 DC 5V，PWM 刷新率 >26kHz，数据传输速率 >30MHz。采用单线串行协议（DIN/DOUT），兼容 WS2812/SK6812，支持级联（最多 1024 像素）。

每个 LED 接收 24 位 RGB 数据（8 位红 +8 位绿 +8 位蓝），共 256 级灰度。RZ 编码（Return-to-Zero），即高电平表示“1”，短高电平+低电平表示“0”。

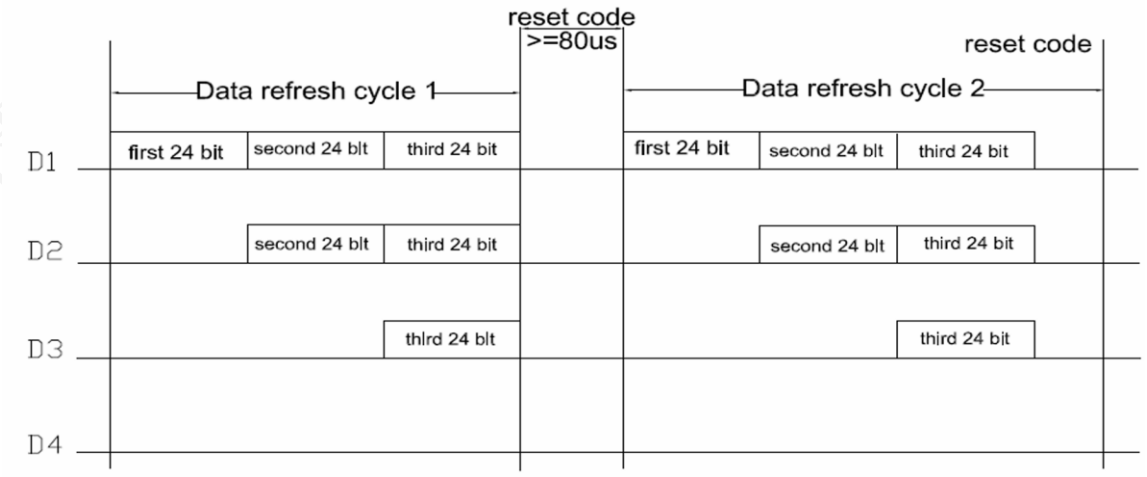


- 逻辑“1”：高电平 ~0.8μs + 低电平 ~0.4μs（总周期 ~1.2μs）
- 逻辑“0”：高电平 ~0.4μs + 低电平 ~0.8μs
- 复位信号：低电平持续 >50μs（RESET），表示一帧数据结束。

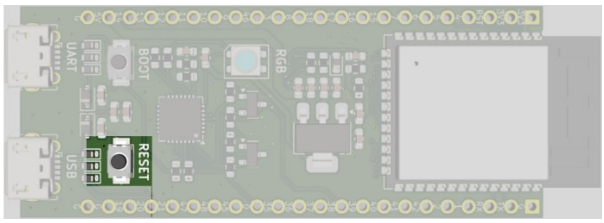
这颗 LED 可以级联，如灯带场景使用（本开发板是单颗）。第一颗 LED 接收前 24 位数据，锁存为自己颜色。剩余数据通过 DOUT 引脚 转发给下一颗。所有 LED 串联，数据像“接力”一样传递。



MCU 输出一段控制代码，经过第一个 LED 时取第一组 24bit 数据转为色彩显示，然后输出剩余所有数据，以此类推控制一串 LED。

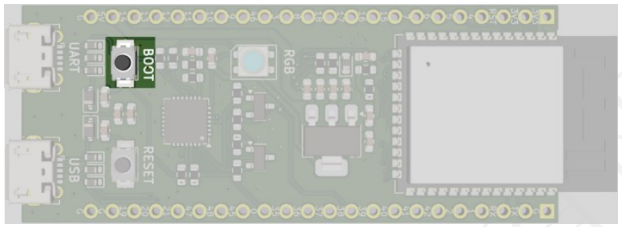


2.4 RESET (EN)



芯片使能/复位引脚。低电平有效，拉低 ＞50μs 触发复位。

2.5 BOOT (GPIO0)



启动模式选择引脚。上电时采样电平决定 ROM 启动方式。

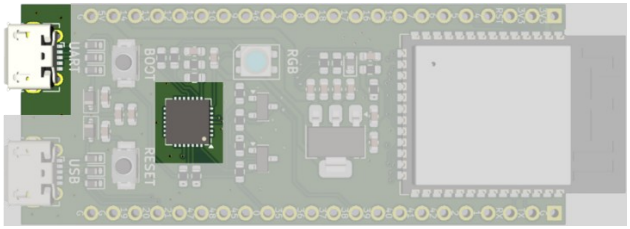
模式	引脚电平（上电时）	启动行为
SPI Boot（正常运行）	高电平（按键保持松开状态）	从 SPI Flash 启动用户固件（默认）
Download Boot（下载模式）	低电平（按键保持按下状态）	进入 USB/UART 下载模式，等待 esptool.py 烧录
保留/调试	其他	特殊用途（不常用）

进入正常运行的条件是 RESET 上升沿，或者系统上电时 BOOT 为高电平，对于本开发板来说就是不做任何操作。

进入下载模式的条件是 RESET 上升沿时 BOOT 为低电平，鉴于按键松开为高电平，按下为低电平。所以手动下载时，需要按键配合

1. 按下 RESET 和 BOOT 按键，均为低电平
2. 松开 RESET，保持 BOOT 按下状态，等待一定时间，MS 级
3. 松开 BOOT，全部为高电平

2.6 UART

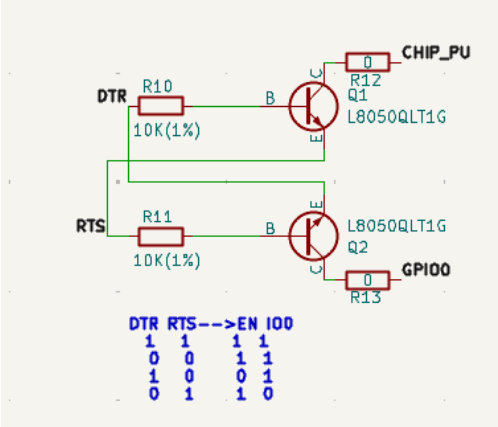


自动下载功能，其一对关键的输入引脚是 DTS 和 RTS，均由 CP2102 芯片提供，ESP32 之固件通

过芯片的串口引脚进行通信下载。

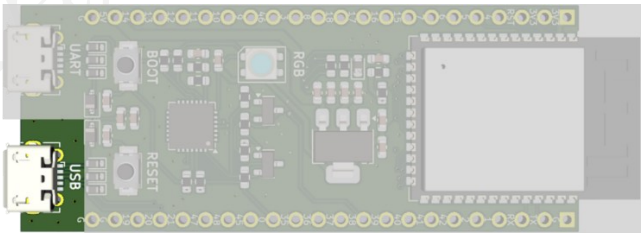
板载有自动下载电路（CP2102 + 晶体管），通过 DTR 和 RTS 引脚控制 CHIP_PU（RESET）和 BOOT(GPIO0)，模拟与手动按键相同的下载时序。下图是部分原理图，CHIP_PU 和 GPIO0 均通过电阻上拉至 VCC，我们列出 DTR 与 RTS 配合输入和 CHIP_PU(RESET)与 GPIO0 输出的映射表为

DTR	RTS	CHIP_PU (RESET)	BOOT(GPIO0)
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0



CP2102 除了负责串口固件下载，还可单独作为串口调试端口，在代码端通过类似“printf”等打印输出命令执行串口的打印指令。

2.7 USB



ESP32-S3 支持 USB OTG（On-The-Go），是其 USB 功能中的核心亮点，允许芯片在主机（Host）和设备（Device）两种角色间动态切换，无需外挂 USB 控制器芯片，即可实现 U 盘读写、键盘/鼠标输入、串口通信、充电检测 等高级功能。

内置即插即用的 USB 串口下载功能，无需外部 USB-to-UART 芯片（如 CP2102），即可实现高速串口通信、固件下载、调试输出。其内置 USB 2.0 全速控制器（12 Mbps），支持 CDC-ACM（虚拟串口）和 JTAG 调试，极大简化开发板设计。

引脚	功能	备注

GPIO19	USB D-	必须连接 USB 数据线
GPIO20	USB D+	必须连接 USB 数据线
VBUS	5V 检测	自动检测 USB 插入
GND	地	共地

与 2.6 传统的 UART 串口下载和调试的对比

项目	**USB CDC(ESP32-S3)**	**UART+CP2102**
芯片需求	无需外挂	需要 CP2102
引脚占用	2 个（D+/D-）	2 个（D+/D-）+电源
最高速率	3Mbps	2Mbps
下载方式	插入即用	需按键或自动电路
驱动兼容	需要 inf（Win7）	通用
成本	更低	更高

三、应用

3.1 AI 视觉识别

ESP32-S3 是 Espressif 首款带有向量加速指令的芯片，能够在本地执行轻量级神经网络模型，因此非常适合做人脸识别、手势检测或物体识别等任务。搭配 OV2640、OV5640 等摄像头模块，它可以在边缘侧快速识别人脸、手势或运动目标，并通过 Wi-Fi 将结果上传到云端，常用于门禁系统、智能考勤、家庭安防摄像头等。

3.2 语音唤醒与智能音频

凭借 I²S 音频接口、PDM 麦克风支持和 AI 指令加速，ESP32-S3 能够在本地完成语音唤醒和关键词识别。它可以作为智能语音助手的核心，识别如“打开灯”或“播放音乐”等指令；也能作为 BLE/Wi-Fi 音频终端，在无线音箱或语音控制面板中实现实时语音交互。

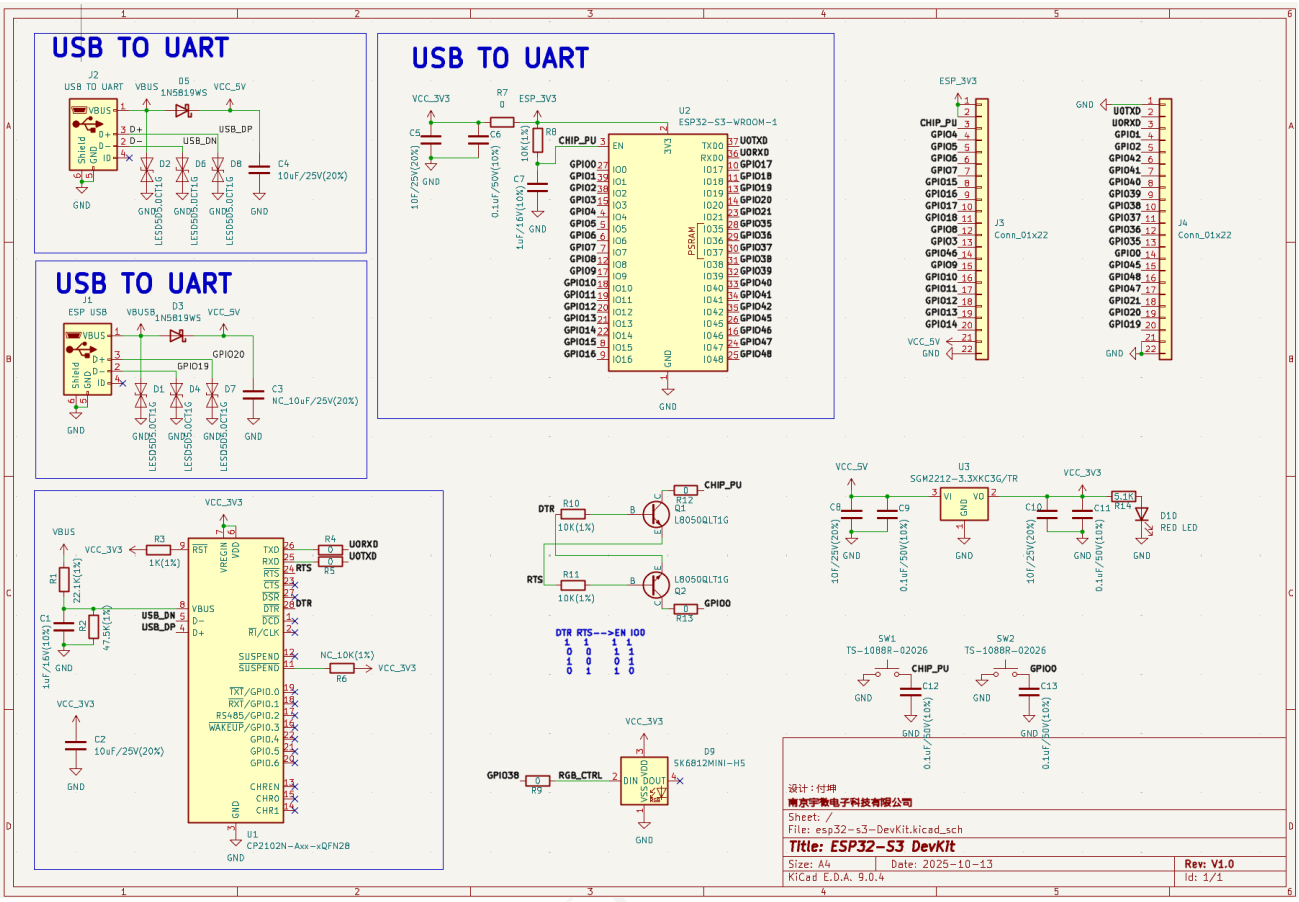
3.3 图形界面与人机交互设备

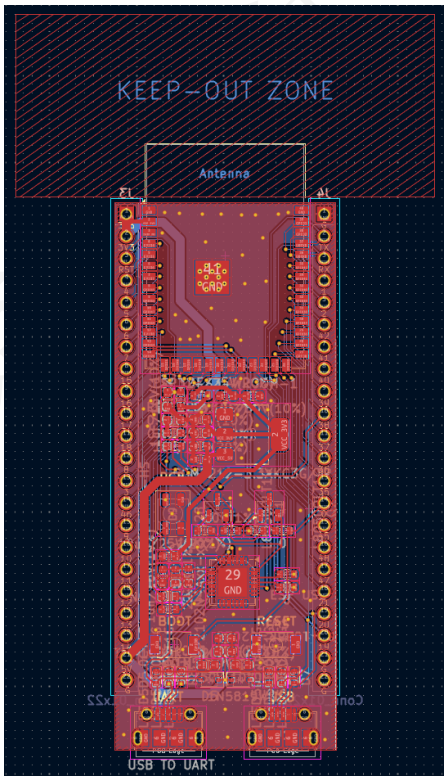
凭借丰富的 GPIO 与 LCD 接口支持，ESP32-S3 能驱动彩色显示屏与触控面板，运行基于 LVGL 的轻量级图形界面。它常被用作智能家电控制面板、便携终端、3D 打印机屏幕控制器等设备的主控芯片，提供流畅、低功耗的图形交互体验。

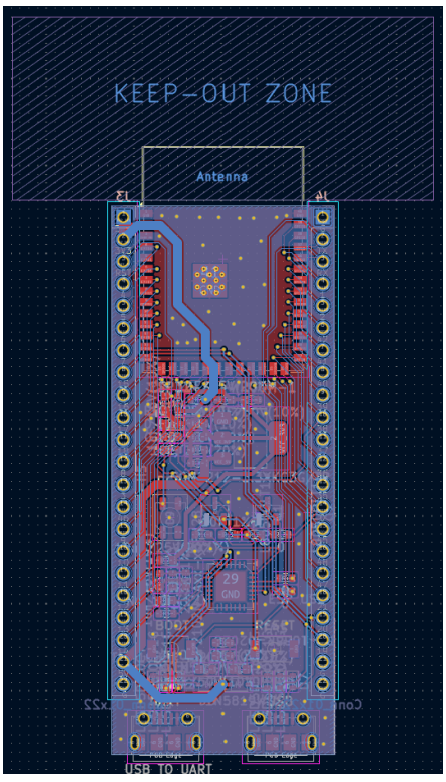
3.4 无线物联网与边缘网关

集成 Wi-Fi 与 BLE 双模无线通信能力，使 ESP32-S3 成为理想的 IoT 中枢节点。它可以将多个蓝牙传感器的数据汇聚后，通过 Wi-Fi 上传至云平台，也能作为家庭自动化或工业监控系统的边缘网关，连接温湿度、光照、气体等各种传感器，构建可靠的无线监控网络。

四、原理图与 PCB 布线


USB TO UART
USB TO UART
USB TO UART
ESP32-S3-WROOM-1
CP2102N-Axx-xQFN28
SGM2212-3.3XKC3G/TR
L8050QLT1G
SK6812MINI-H5
TS-1088R-02026
Title: ESP32-S3 DevKit
Size: A4
Date: 2025-10-13
Rev: V1.0
KiCad E.D.A. 9.0.4
Sheet: /
File: esp32-s3-DevKit.kicad_sch


KEEP-OUT ZONE
Antenna
USB TO UART


KEEP-OUT ZONE
Antenna
USB TO UART

五、编程指南

5.1 Hello World

这段代码是 ESP-IDF 官方“Hello World”示例程序的主函数 app_main()，用于验证 ESP32/ESP32-S3 等芯片的基本运行环境。

5.1.1 代码总览

```
/*
 * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
 *
 * SPDX-License-Identifier: CC0-1.0
 */

#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"
#include "esp_system.h"

void app_main(void)
{
    printf("Hello world!\n");

    /* Print chip information */
    esp_chip_info_t chip_info;
    uint32_t flash_size;
    esp_chip_info(&chip_info);
    printf("This is %s chip with %d CPU core(s), %s%s%s%s, ",
           CONFIG_IDF_TARGET,
           chip_info.cores,
           (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
           (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
           (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
           (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15.4 (Zigbee/Thread)" : "");

    unsigned major_rev = chip_info.revision / 100;
    unsigned minor_rev = chip_info.revision % 100;
    printf("silicon revision v%d.%d, ", major_rev, minor_rev);
    if(esp_flash_get_size(NULL, &flash_size) != ESP_OK) {
        printf("Get flash size failed");
```

```
        return;
    }

    printf("%" PRIu32 "MB %s flash\n", flash_size / (uint32_t)(1024 * 1024),
           (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "embedded" : "external");

    printf("Minimum free heap size: %" PRIu32 " bytes\n", esp_get_minimum_free_heap_size());

    for (int i = 10; i >= 0; i--) {
        printf("Restarting in %d seconds...\n", i);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    printf("Restarting now.\n");
    fflush(stdout);
    esp_restart();
}
```

5.1.2 代码简析

功能	函数代码	注解
主体结构	void app_main(void)	ESP-IDF 中，app_main()是程序的入口函数（相当于传统 C 程序的 main()）。RTOS 启动后会自动创建一个任务来运行 app_main()。
打印欢迎信息	printf("Hello world!\n")	打印 "Hello world!"，测试串口输出是否正常。
获取芯片信息	esp_chip_info_t chip_info; uint32_t flash_size; esp_chip_info(&chip_info);	esp_chip_info_t 是结构体类型，保存芯片核心数、支持特性（Wi-Fi、BT、BLE 等） esp_chip_info(&chip_info)调用 SDK 函数获取当前芯片的详细信息。
打印芯片信息	printf("This is %s chip with %d CPU core(s), %s%s%s%s, ", CONFIG_IDF_TARGET, chip_info.cores, (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "", (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "", (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",	根据 chip_info.features 打印出当前芯片的功能

	(chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15.4 (Zigbee/Thread)" : "");	

打印版本号	unsigned major_rev = chip_info.revision / 100;
	unsigned minor_rev = chip_info.revision % 100;
	printf("silicon revision v%d.%d, ", major_rev, minor_rev);
		解析芯片版本号

读取Flash容量	if(esp_flash_get_size(NULL, &flash_size) != ESP_OK) {
	printf("Get flash size failed");
	return;
	}
		esp_flash_get_size(NULL, &flash_size) 读取 Flash 大小（以字节为单位）
		若失败则打印错误并退出。

打印Flash信息	printf("%" PRIu32 "MB %s flash\n", flash_size / (uint32_t)(1024 * 1024), (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "embedded" : "external");	输出 flash 的容量和类型： "embedded" 表示片上 flash "external" 表示外挂 flash
打印堆内存状态	printf("Minimum free heap size: %" PRIu32 " bytes\n", esp_get_minimum_free_heap_size());	打印当前最小空闲堆大小，用于检测内存是否充足。

倒计时并重启	for (int i = 10; i >= 0; i--) {
	printf("Restarting in %d seconds...\n", i);
	vTaskDelay(1000 / portTICK_PERIOD_MS);
	}
	printf("Restarting now.\n");
	fflush(stdout);
	esp_restart();
		每秒打印一次倒计时
		vTaskDelay() 是 FreeRTOS 的延时函数
		倒计时结束后刷新输出缓冲区（fflush）
		调用 esp_restart() 复位芯片

六、联系我们

若需任何帮助，请邮件联系我们：info@fukunlab.com

样品购买：淘宝店铺-宇微电子

产品介绍：