物联网智能照明系统

摘要

本文设计并实现了一种基于 ESP32-C3 芯片的物联网智能照明系统。该系统采用 2.0 寸 IPS 液晶显示屏和电容触摸屏提供人机交互界面，通过 WiFi 连接阿里云平台实现远程控制。系统在 ESP-IDF 开发平台上开发，移植了 LCD 驱动、LVGL 图形库和阿里云 C Link SDK 等关键组件。在软件实现上，系统采用 SPI 接口驱动 LCD 显示，I2C 接口控制触摸屏，并使用 MQTT 协议与云平台进行数据交互。通过 Gui Guider 工具设计的用户界面，系统可以实时显示环境温湿度数据并支持照明设备的远程控制。实验结果表明，该系统具有界面友好、功能完善、成本适中等特点，为智能家居领域提供了一种实用的解决方案。

关键词：物联网；esp32；阿里云；lvgl

1 引言

现在人们已经进入了互联网时代，很多传统的生活设备已经被一一淘汰。在未来，传统的有线家庭设备将逐渐被取代取而代之的是由网络集成的智能家居系统。的智能家居可以通过网络将家中的各种设备连接起来利用物联网技术为用户提供服务如照明、家电控制、防盗报警等环境监测。与普通家庭相比，智能家居可以让用户获得更高效和智能生活经验，同时保持原有的工作习惯。具有机器学习功能的智能家居也可以提供为用户提供互动体验，启用客房设施根据用户的使用习惯，规划出更加舒适和为用户提供环保生活。然而，这种智能家用设备在中国不受欢迎。这份报告的设计是智能家居中的智能照明系统。整个系统的设计是为了尝试和实现不需要的技术在中国普通家庭中很常见。为了获得能量节约和智能控制照明及其强度。在这个项目中，开发了一个原型，利用模型进行仿真。

2 相关工作

使用带有 wifi 功能的 esp32-c3 芯片作为主控芯片，在 esp-idf5.3.1 开发平台上移植 lcd 屏幕驱动以及 lvgl 轻量化图形显示框架，使用 guider gui 软件绘制用户交互 ui 界面，编写温湿度传感器驱动以及 wifi 驱动程序，移植阿里云的 c link sdk 来让 esp32-c3 与阿里云平台使用 mqtt 协议进行通信。边端可以把温湿度传感器上传至阿里云服务器，以及接受阿里云上下达的指令。

2.1 硬件框架

硬件框图如图1所示：

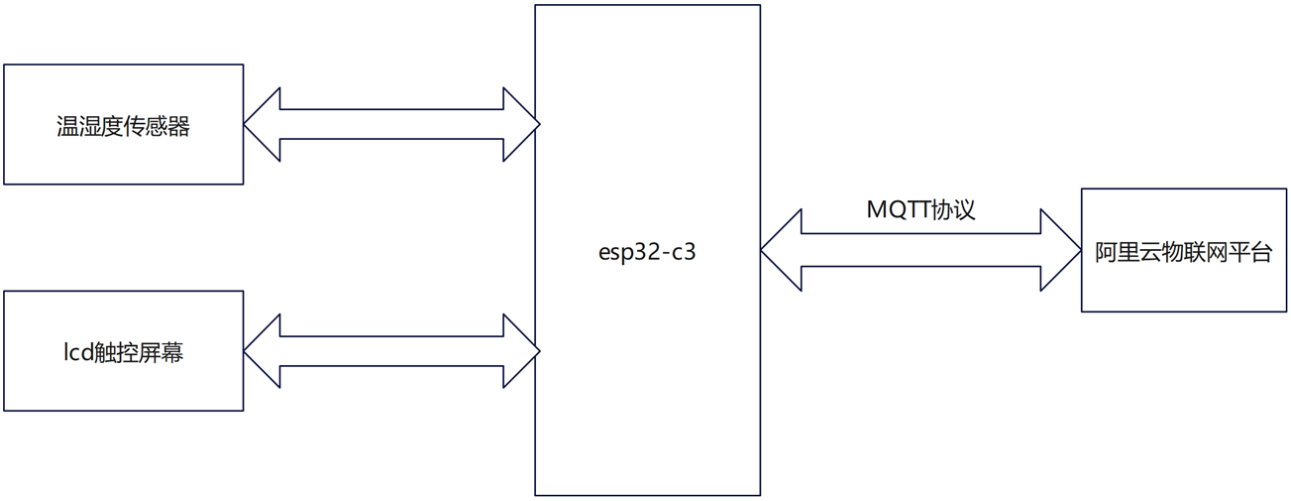

图 1. 硬件框图

2.2 软件框图

软件框图如图2所示：

2.3 lcd 驱动移植

ESP 芯片可以生成市面上常见的 LCD 所需的各种时序，如 SPI LCD、I2C LCD、并行 LCD （Intel 8080）、RGB/SRGB LCD、MIPI DSI LCD 等。esp_lcd 组件提供了一个抽象的驱动框架，以统一的方式支持它们 [2]。

开发板上的液晶屏是 2.0 寸的 IPS 高清液晶屏，分辨率 240*320，显示非常清晰。液晶屏驱动芯片 ST7789，采用 SPI 通信方式与 ESP32-C3 连接。开发板上的触摸屏是电容触摸屏，用手指就可以触摸，支持双指触摸。触摸芯片型号是 FT6336，使用 I2C 接口与 ESP32-C3 连接，I2C 地址为 0x38。

移植步骤如下：

1. spi 总线初始化

```
    esp_err_t ret;
    spi_bus_config_t buscfg = {
        //spi总线使用的引脚
        .miso_io_num = PIN_NUM_MISO,
        .mosi_io_num = PIN_NUM_MOSI,
        .sclk_io_num = PIN_NUM_CLK,
        .quadwp_io_num = -1,
        .quadhd_io_num = -1,
        //在一次spi传输中最多传输80行像素
    .max_transfer_sz = EXAMPLE_LCD_H_RES * 80 * sizeof(uint16_t)
    };
    //使用dma的传输方式
    ret = spi_bus_initialize(LCD_HOST, &buscfg, SPI_DMA_CH_AUTO);
```

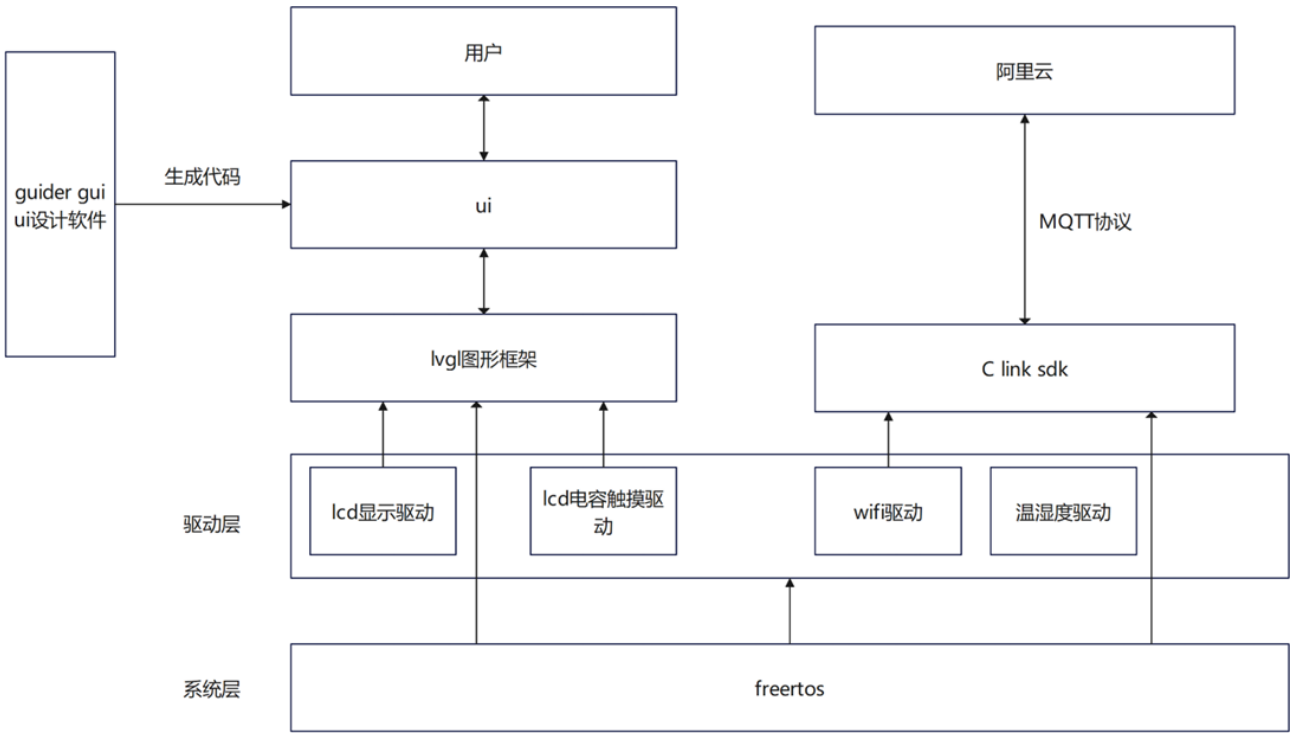

图 2. 软件框图

2. SPI 总线分配 LCD IO 设备句柄

```
    esp_lcd_panel_io_spi_config_t io_config = {
        .dc_gpio_num = LCD_PIN_NUM_LCD_DC, // 数据/命令选择引脚
        .cs_gpio_num = LCD_PIN_NUM_LCD_CS, // 片选引脚
        .pclk_hz = LCD_LCD_PIXEL_CLOCK_HZ, // 像素时钟频率
        .lcd_cmd_bits = LCD_LCD_CMD_BITS, // 命令位数
        .lcd_param_bits = LCD_LCD_PARAM_BITS, // 参数位数
        .spi_mode = 0,//spi模式CPOL, CPHA
        .trans_queue_depth = 10, // 传输队列深度
    };
    // 将 LCD 连接到 SPI 总线
    esp_lcd_new_panel_io_spi((esp_lcd_spi_bus_handle_t)LCD_HOST
    , &io_config, io_handle);
```

3. 安装 LCD 控制器驱动

配置参数调用 esp_lcd_new_panel_st7789() 来创建 LCD 面板的句柄，这个句柄后续会用于控制显示屏的显示内容。io_handle 是上一步骤中获得的从 spi 总线分配的 lcd io 句柄。

```

esp_err_t esp_spi_lcd_panel_init(esp_lcd_panel_handle_t
*panel_handle,const esp_lcd_panel_io_handle_t io_handle)
{
```

```
    esp_lcd_panel_dev_config_t panel_config = {
    // 复位引脚，设为-1表示不使用硬件复位
        .reset_gpio_num = LCD_PIN_NUM_RST,
    // RGB颜色元素的顺序
        .rgb_ele_order = LCD_RGB_ELEMENT_ORDER_RGB,
    // 每像素16位色深，即RGB565格式
        .bits_per_pixel = 16,
    };
    // 为 ST7789 创建 LCD 面板句柄，并指定 SPI IO 设备句柄
    return esp_lcd_new_panel_st7789(io_handle,
    &panel_config, panel_handle);
```

2.4 lcd 触摸驱动移植

下载 FT5x06 控制器组件。

```
idf.py add-dependency "espressif/esp_lcd_touch_ft5x06^1.0.4"
```

由于使用的是 idf5.3.1，有些 api 不适用，需要对 FT5x06 控制器组件代码进行修改。managed_components 中的代码，需要先把 managed_components 中的组件放到自己的 components 中才可以修改，否则你改了之后，一编译，就会恢复原来的样子，就白改了。所以我们先在 spi_lcd_touch 文件夹中新建一个文件，命名为 components。然后把 managed_components 文件夹中的 espressif__esp_lcd_touch_ft5x06 文件夹剪切到 components 文件夹中。注意，是剪切，不是复制。

修改触摸读写函数为如下：

```

static esp_err_t touch_ft5x06_i2c_write(
esp_lcd_touch_handle_t tp, uint8_t reg, uint8_t data)
{
    assert(tp != NULL);

    // *INDENT-OFF*
    /* 写入数据 */
    uint8_t write_buf[2] = {reg, data};
    return i2c_master_transmit(i2c_ft5x06_dev
    , write_buf, sizeof(write_buf), -1);
    // *INDENT-ON*
}

static esp_err_t touch_ft5x06_i2c_read(
esp_lcd_touch_handle_t tp, uint8_t reg,
uint8_t *data, uint8_t len)
```

```
{
    assert(tp != NULL);
    assert(data != NULL);
    return i2c_master_transmit_receive(
    i2c_ft5x06_dev,&reg,1,data,len,-1);
}
```

触摸配置为如下:

```

    /* 配置触摸屏参数 */
    esp_lcd_touch_config_t tp_cfg = {
        .x_max = 320,                       /* X轴最大分辨率 */
        .y_max = 240,                       /* Y轴最大分辨率 */
        .rst_gpio_num = -1,                 /* 复位引脚,不使用 */
        .int_gpio_num = -1,                 /* 中断引脚,不使用 */
        .levels = {
            .reset = 0,                     /* 复位电平 */
            .interrupt = 0,                 /* 中断电平 */
        },
        .flags = {
            .swap_xy = 0,                   /* 是否交换XY轴 */
            .mirror_x = 0,                  /* 是否镜像X轴 */
            .mirror_y = 0,                  /* 是否镜像Y轴 */
        },
    };

    /* 创建触摸屏句柄并初始化FT5x06触摸芯片 */
    esp_lcd_touch_handle_t tp;
    //esp_lcd_panel_io_handle_t io_handle;
    esp_lcd_touch_new_i2c_ft5x06(io_handle,
    &tp_cfg,
    &tp,i2c_master_handle);
```

2.5 lvgl 移植

下载组件

```
idf.py add-dependency "espressif/esp_lvgl_port^1.4.0"
```

配置如下 [1]:

```
/* 显示屏宽度 */
#define DISP_WIDTH 240
```

```c
/* 显示屏高度 */
#define DISP_HEIGHT 320
/* LVGL显示设备句柄 */
static lv_disp_t * disp_handle;

/**
 * @brief 初始化LVGL端口
 * @param io_handle LCD面板IO句柄
 * @param panel_handle LCD面板句柄
 * @param _disp_handle LVGL显示设备句柄指针
 * @return esp_err_t 错误码
 */
esp_err_t u_esp_lvgl_port_init(esp_lcd_panel_io_handle_t
io_handle,esp_lcd_panel_handle_t panel_handle
,lv_disp_t **_disp_handle)
{
    /* 使用默认配置初始化LVGL */
    const lvgl_port_cfg_t lvgl_cfg =
    ESP_LVGL_PORT_INIT_CONFIG();
    esp_err_t err = lvgl_port_init(&lvgl_cfg);
    if (err != ESP_OK) {
        return err;
    }

    /* 配置LVGL显示设备参数 */
    const lvgl_port_display_cfg_t disp_cfg = {
        .io_handle = io_handle,                  /* LCD面板IO句柄 */
        .panel_handle = panel_handle,            /* LCD面板句柄 */
        .buffer_size = 320*sizeof(uint16_t)*14*2,
        .double_buffer = true,                   /* 使用双缓冲 */
        .hres = DISP_WIDTH,                      /* 水平分辨率 */
        .vres = DISP_HEIGHT,                     /* 垂直分辨率 */
        .monochrome = false,                     /* 非单色显示 */
        //.color_format = LV_COLOR_FORMAT_RGB565,
        .rotation = {
            .swap_xy = false,                    /* 不交换XY轴 */
            .mirror_x = false,                   /* 不镜像X轴 */
            .mirror_y = false,                   /* 不镜像Y轴 */
        },
        .flags = {
```

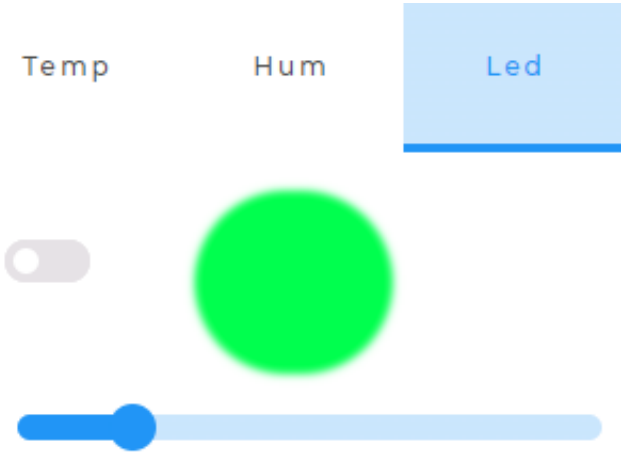

图 3. ui 界面

```
43                 .buff_dma = true,          /* 使用DMA传输显示缓冲 */
44                 //.swap_bytes = false,           /* 不交换字节序 */
45             }
46     };
47     /* 添加显示设备并获取句柄 */
48     disp_handle = lvgl_port_add_disp(&disp_cfg);
49     *_disp_handle = disp_handle;
50     return ESP_OK;
51 }
```

main 函数中执行:

```
1     lv_disp_t *disp_handle;
2     ESP_ERROR_CHECK(u_esp_lvgl_port_init(io_handle,
3     panel_handle,&disp_handle));
```

lvgl 触摸输入配置

```
1     /* 配置LVGL触摸设备参数 */
2     const lvgl_port_touch_cfg_t touch_cfg = {
3         .disp = disp_handle,   /* LVGL显示设备句柄 */
4         .handle = tp,          /* 触摸屏句柄 */
5     };
6     /* 添加触摸设备并获取句柄 */
7     lv_indev_t* touch_handle =
8     lvgl_port_add_touch(&touch_cfg);
```

2.6 Gui Guider 绘制 ui 界面

绘制如下图3所示，表示 led 灯，temp 和 hum 分别表示温度和湿度界面。需要把生成的代码添加进工程的编译路径。

2.7 移植 C Link sdk

将 C-SDK 作为 idf 自定义组件引入到 idf 的 components 目录中 [3]。在 link sdk 目录下添加 CmakeLists.txt 文件。CMakeLists.txt 内容如下:

```
set(include_dirs core core/sysdep core/utils components/data-model )
file(GLOB c_sdk_srcs
    "core/*.c"
    "core/utils/*.c"
    "core/sysdep/*.c"
    "portfiles/aiot_port/*.c"
    "external/*.c"
    "components/data-model/*.c"
)
idf_component_register(SRCS ${c_sdk_srcs}
                       INCLUDE_DIRS "${include_dirs}"
                       REQUIRES mbedtls)
```

LinkSDK 与 idf 中都有 mbedtls 库,避免引用库冲突,修改文件 idf 目录下的/components/C-SDK/core/sysdep/core_adapter.c，关闭 CORE_ADAPTER_MBEDTLS_ENABLED 宏定义。

下载官方附件的station_example_main.c替代 examples\wifi\getting_started\station\main 下的/station_example_main.c 文件.

2.8 阿里云创建物模型

创建图4所示，此处，创建的产品功能为设备上报温湿度数据到阿里云物模型 [4]，和接收阿里云服务器数据进行设备属性设置，既 led 等的亮灭。

2.9 设备代码编写

1. wifi 名称与密码修改

```
#define EXAMPLE_ESP_WIFI_SSID      "1133"
#define EXAMPLE_ESP_WIFI_PASS      "ltf12345678"
#define EXAMPLE_ESP_MAXIMUM_RETRY  10
```

2. 设备三元组

```
/* TODO: 替换为自己设备的三元组 */
char *product_key        = "k22e0WCtkgb";
char *device_name        = "esp32";
char *device_secret      = "82e2705b2a5b3bf37daa3c48bd786769";
```

3. 端口号修改为 1883 4.link_main 函数修改订阅设备属性设置主题

```
  /* MQTT 订阅 topic 功能示例，请根据自己的业务需求进行使用 */
```


新建产品 （设备模型）
新建产品
从设备中心新建产品
* 产品名称
请输入产品名称
* 所属品类
标准品类
自定义品类
请选择标准品类
查看功能
* 节点类型
直连设备
网关子设备
网关设备
连网与数据
* 连网方式
Wi-Fi
* 数据格式
ICA 标准数据格式（Alink JSON）
校验类型
认证方式
更多信息
确认
取消
CSDN @L_ss

图 4. 创建产品

```
    {
        char *sub_topic =
        "/sys/k22e0WCtkgb/esp32/thing/service/property/set";

        res = aiot_mqtt_sub(mqtt_handle, sub_topic, NULL, 1, NULL);
        if (res < 0) {
            printf("aiot_mqtt_sub failed, res: -0x%04lX\n", -res);
            return -1;
        }
    }
```

5. 修改 demo_mqtt_process_thread 线程，来发送温湿度数据

```
/* 执行aiot_mqtt_process的线程，包含心跳发送和QoS1消息重发 */
void *demo_mqtt_process_thread(void *args)
{
    int32_t res = STATE_SUCCESS;

    while (g_mqtt_process_thread_running) {
        res = aiot_mqtt_process(args);
         {
        //添加的温湿度发送代码
        char *pub_topic =
        "/sys/k22e0WCtkgb/esp32/thing/event/property/post";
        //添加数据进入json格式
        cJSON*param =cJSON_GetObjectItem(mqtt_cjson.root,"params");
        cJSON_ReplaceItemInObject(param,
        "temperature",cJSON_CreateNumber(temperature));
        cJSON_ReplaceItemInObject(param,"hum",
        cJSON_CreateNumber(humidity));
        temperature+=0.1;
        humidity+=0.1;

        char*pub_payload = cJSON_PrintUnformatted(
        mqtt_cjson.root);
        // ESP_LOGI(TAG, "%s", pub_payload);
        //发送数据
        res = aiot_mqtt_pub(args, pub_topic,
        (uint8_t *)pub_payload, strlen(pub_payload), 0);
        if (res < 0) {
            printf("aiot_mqtt_sub
            failed, res: -0x%04lX\n", -res);
```

```
30
31          }
32          cJSON_free(pub_payload);
33          }
34          if (res == STATE_USER_INPUT_EXEC_DISABLED) {
35              break;
36          }
37          sleep(10);
38      }
39      return NULL;
40  }
```

6. 修改接收回调函数 demo_mqtt_default_recv_handler 。aiot_mqtt_recv 函数会调用接受回调函数 demo_mqtt_default_recv_handler。

```
1   case AIOT_MQTTRECV_PUB: {
2       printf("pub, qos: %d, topic: %.*s\n",
3       packet->data.pub.qos, packet->data.pub.topic_len
4       , packet->data.pub.topic);
5       printf("pub, payload: %.*s\n",
6       (int16_t)packet->data.pub.payload_len,
7       packet->data.pub.payload);
8       /* TODO: 处理服务器下发的业务报文 */
9       if(strstr((char*)packet->data.pub.topic,
10      "/property/set")){
11          //判断下发的指令是关灯还是开
12          if(strstr((char*)packet->data.pub.payload,
13          "{\"LEDSwitch\":1}")){
14              lv_custom_set_light(1);
15              // lv_led_on(guider_ui.screen_1_led_1);
16              ESP_LOGI(TAG, "light is on");
17          }
18          else{
19              lv_custom_set_light(0);
20              // lv_led_off(guider_ui.screen_1_led_1);
21              ESP_LOGI(TAG, "light is off");
22          }
```

3 本文方法

3.1 本文方法概述

本文设计并实现了一种基于物联网技术的智能照明系统。该系统的核心是使用 ESP32-C3 芯片作为主控单元，结合 LCD 显示屏和触摸屏，实现用户友好的交互界面。系统通过 WiFi 连接到阿里云平台，利用 MQTT 协议进行数据通信，实现远程监控和控制。

首先，硬件部分包括 ESP32-C3 芯片、2.0 寸 IPS 液晶屏和电容触摸屏。液晶屏通过 SPI 接口与 ESP32-C3 连接，而触摸屏则通过 I2C 接口连接。软件部分在 ESP-IDF 开发平台上进行，移植了 LCD 驱动和 LVGL 轻量级图形库，以实现图形界面的显示和触摸交互。

其次，系统通过移植阿里云的 C Link SDK，实现了与阿里云平台的连接。设备能够将采集到的温湿度数据上传至云端，并接收来自云端的控制指令。用户可以通过阿里云平台远程控制家中的照明设备，实现智能化的灯光管理。

最后，本文还使用 Gui Guider 工具设计了用户界面，提供了直观的温湿度显示和灯光控制功能。通过对比现有的智能家居解决方案，本文的方法在硬件成本、系统稳定性和用户体验方面具有一定的优势。

4 复现细节

4.1 与已有开源代码对比

本项目在实现过程中参考了以下开源代码：

- esp-idf5.3.1 的官方示例：在 wifi 模块中参考了其中的 wifi 连接实现，具体查看相关工作章节中的设备代码编写小节
- c link sdk: 使用阿里云官方提供的 link sdk 来与阿里云进行通信，具体查看相关工作章节中的移植 C link sdk 小节

4.2 实验环境搭建

esp-idf 下载地址：https://dl.espressif.com/dl/esp-idf/，选择图5版本。

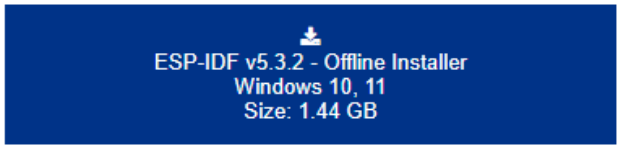

图 5. esp-idf 版本

下载安装后，打开 vscode，安装 ESP-IDF 扩展。然后把下载好的 ESP-IDF v5.3.2 导入 vscode, 点击下图6中的 USE EXISTING SETUP.

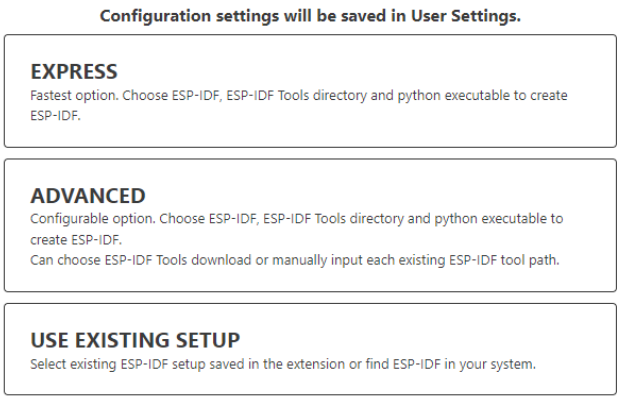

图 6. 配置

点击下图7中的下选项，即可完成环境配置。

图 7. 配置 2

4.3 界面分析与使用说明

在 vscode 的右下角，会出现如下图8所示图案。

图 8. 界面分析

其中，从左往右分别为选择烧录方式，此处是串口烧录；串口通信端口选择，此处为 com3；芯片型号选择，此处为 esp32-c3；编译配置选项，与 make menuconfig 功能一样；删除编译生成的文件；编译按键；烧录按钮；监视按钮。

4.4 创新点

原论文使用的是 stm32 加上 esp8266 模组来实现，本文使用的是 esp32c3 芯片来实现，成本低，不同的芯片平台，代码需要重新编写；原论文使用的显示是 oled 屏，本文使用的是彩色 lcd 屏加 gui guider 软件绘制更加美观的用户交互界面。在与阿里云服务器通信中，本文采用了阿里云官方提供的 c link sdk，更加方便，可移植性更高。

5 实验结果分析

在阿里云平台上接收到边端设备发送的温湿度信息，如下图9、图10、图11所示。

图 9. 阿里云接收数据结果

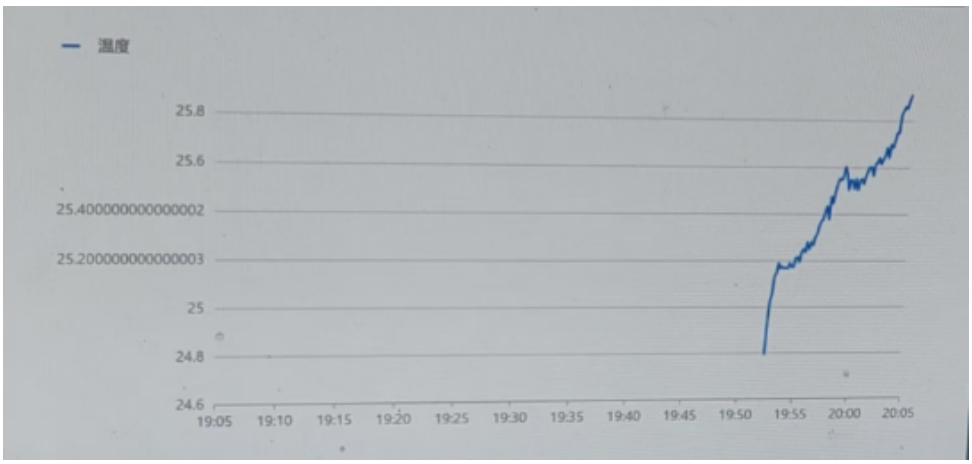

图 10. 阿里云温度曲线图

图 11. 阿里云湿度曲线图

阿里云可以发送命令控制边端设备 led 灯的亮灭, 如下图12所示:

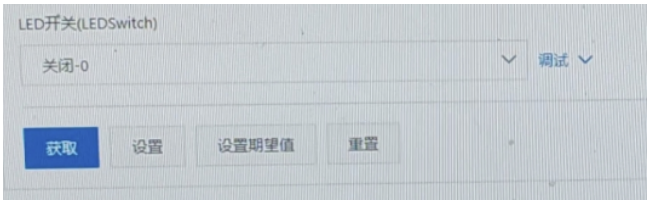

图 12. 阿里云控制 led

6 总结与展望

本文设计并实现了一个基于 ESP32-C3 的智能照明系统，通过整合 LCD 显示、触摸控制和物联网技术，实现了本地和远程的智能照明控制。系统采用 LVGL 图形库构建友好的用户界面，并通过阿里云物联网平台实现远程数据监控和设备控制。实验结果表明，该系统具有较好的实用性和可靠性。

然而，当前系统仍存在一些不足：首先，系统的智能化程度有待提高，可以引入机器学习算法来实现自适应照明控制；其次，系统的安全性需要进一步加强，特别是在物联网通信方面；最后，系统的功能相对单一，未来可以整合更多传感器和控制模块，扩展至全屋智能家居系统。未来的研究方向包括引入深度学习算法优化控制策略、增强系统安全性、扩展多设备协同控制等方面。

参考文献

[1] LVGL. Lvgl documentation. https://docs.lvgl.io/, 2024. 访问时间: 2024-11-20.

[2] 乐鑫信息科技. Esp32 lcd 开发指南. https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32/api-reference/peripherals/lcd.html, 2023. 访问时间: 2024-11-20.

[3] 阿里云. 物联网平台. https://help.aliyun.com/product/30520.html, 2024. 访问时间: 2024-11-20.

[4] 阿里云 IoT. Link sdk 文档中心. https://help.aliyun.com/document_detail/163755.html, 2024. 访问时间: 2024-11-20.