深圳技术大学

考试答题纸

(以论文、报告等形式考核专用)  
二○ 二四 ～二○ 二五 学年度第 一 学期

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 | SG00291 | | 课程名称 | 嵌入式系统设计 | 主讲教师 | 张智星 | 郑斯凯 | 评分 |  |
| 学 号 | 202200601099 | | 姓名 | 裴文成 | 班级 | 22级电子2班 | | | |
| 教师评语： | |  | | | | | | | |

|  |  |
| --- | --- |
| **小组题目：** | **基于STM32和ESP01S的智能家居系统** |

|  |  |
| --- | --- |
| **小组全部成员：** | **袁宏阳、裴文成、骆嘉怡** |

**基于STM2和ESP01S的智能家居系统**

**一、项目介绍（15分）**

**1. 项目背景**

随着物联网（IoT）技术的快速发展，智能家居系统逐渐成为人们提升生活质量的重要工具。传统的家居设备通常需要手动控制，而智能家居系统通过将电子设备、传感器、和通信模块相结合，实现了设备的自动化和远程控制。这种技术不仅能够优化能源使用，还能提升安全性和便捷性，符合现代人对智慧生活的追求。

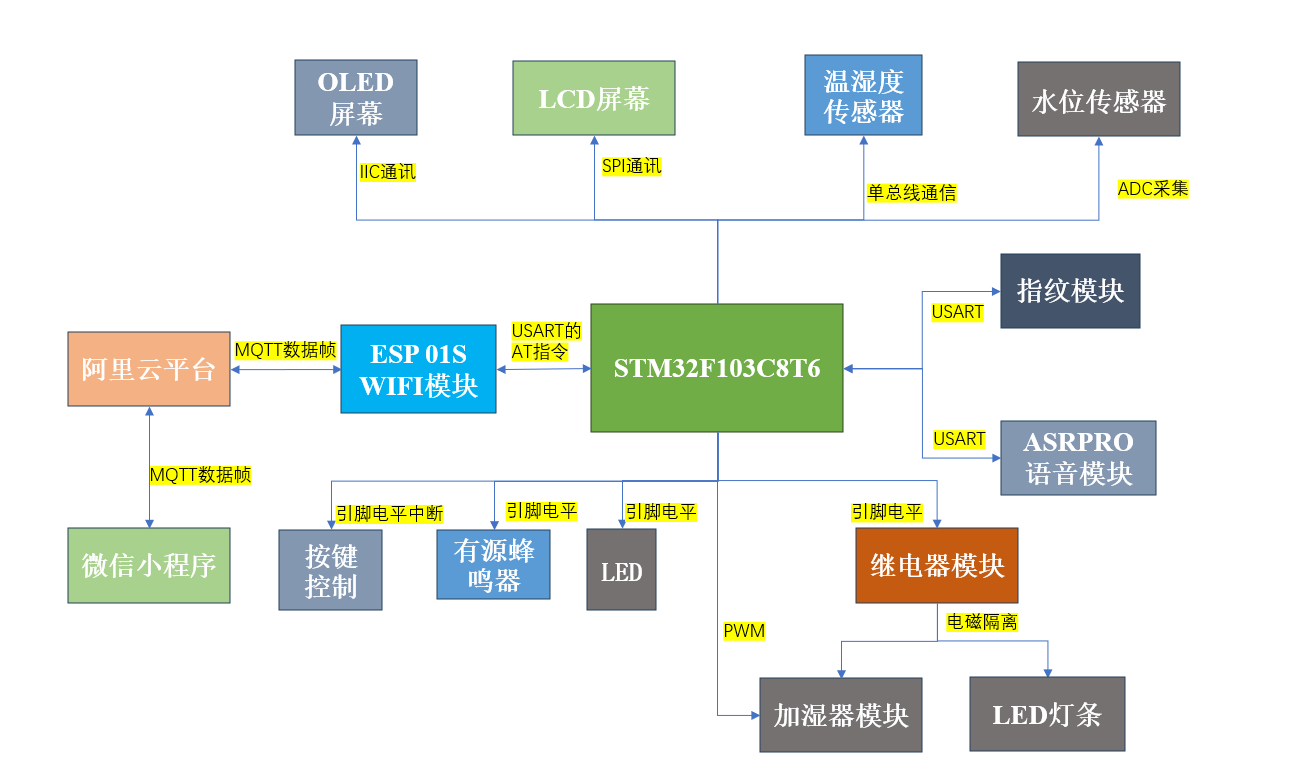
**2. 项目介绍**

本项目旨在基于 STM32 微控制器、 ESP01S Wi-Fi 模块、指纹识别模块和天问51语音模块，设计并实现一个简易的智能家居系统。STM32 作为主控芯片，负责连接传感器和执行设备，如温湿度传感器、继电器控制模块等；ESP01S提供无线通信功能，使用户可以通过手机小程序远程监控和控制家居设备; 指纹识别模块可以进行用户识别、用户注册、用户删除等功能；天问51语音模块提供简单高效的语音交互，用户在身份识别通过后可语音控制家居设备；

通过将低功耗、高性能的 STM32 微控制器与支持 Wi-Fi 的 ESP01S 模块相结合，本项目旨在打造一个高性价比且易于实现的智能家居系统，该系统集成了指纹识别和语音交互功能，为用户提供便捷、安全的远程监控和控制体验。

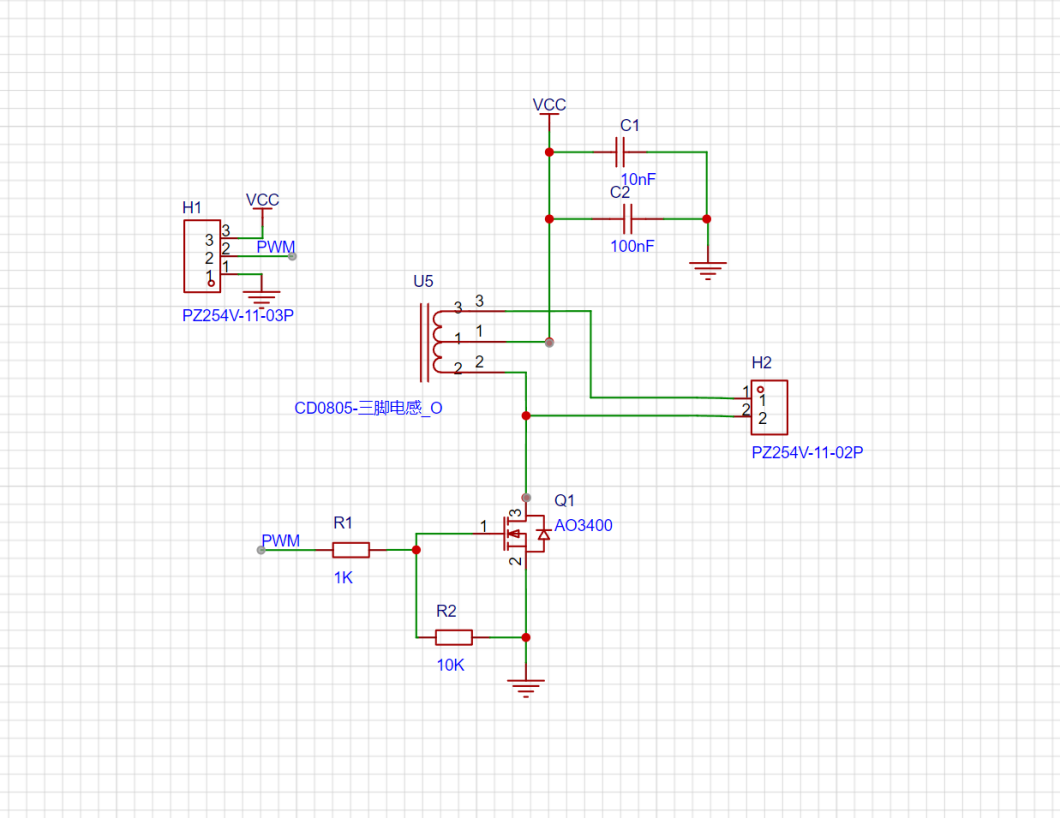
**二、设计（40分）**

**1. 嵌入式硬件设计**

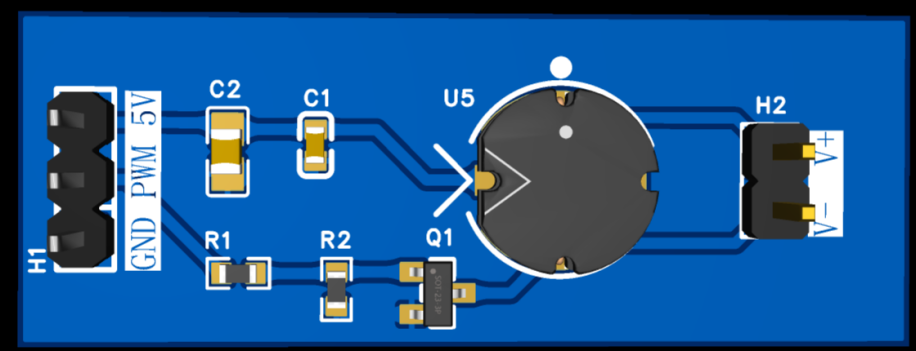


**图1 系统硬件结构图**

该项目所需硬件架构图如上，我们需要将WIFI模块、指纹传感器、LCD屏幕、OLED屏幕、按键、温湿度传感器、加湿雾化器、水位传感器等接入主控MCU。通过WIFI模块使用MQTT协议连接阿里云服务器从而实现数据的流转，并在上位机及手机微信小程序端进行数据的显示及控制。我们选用STM32F1O3C8T6作为我们的主控MCU，因为其价格便宜的同时，硬件、内存及IO口的数量也大致满足该项目的需求。

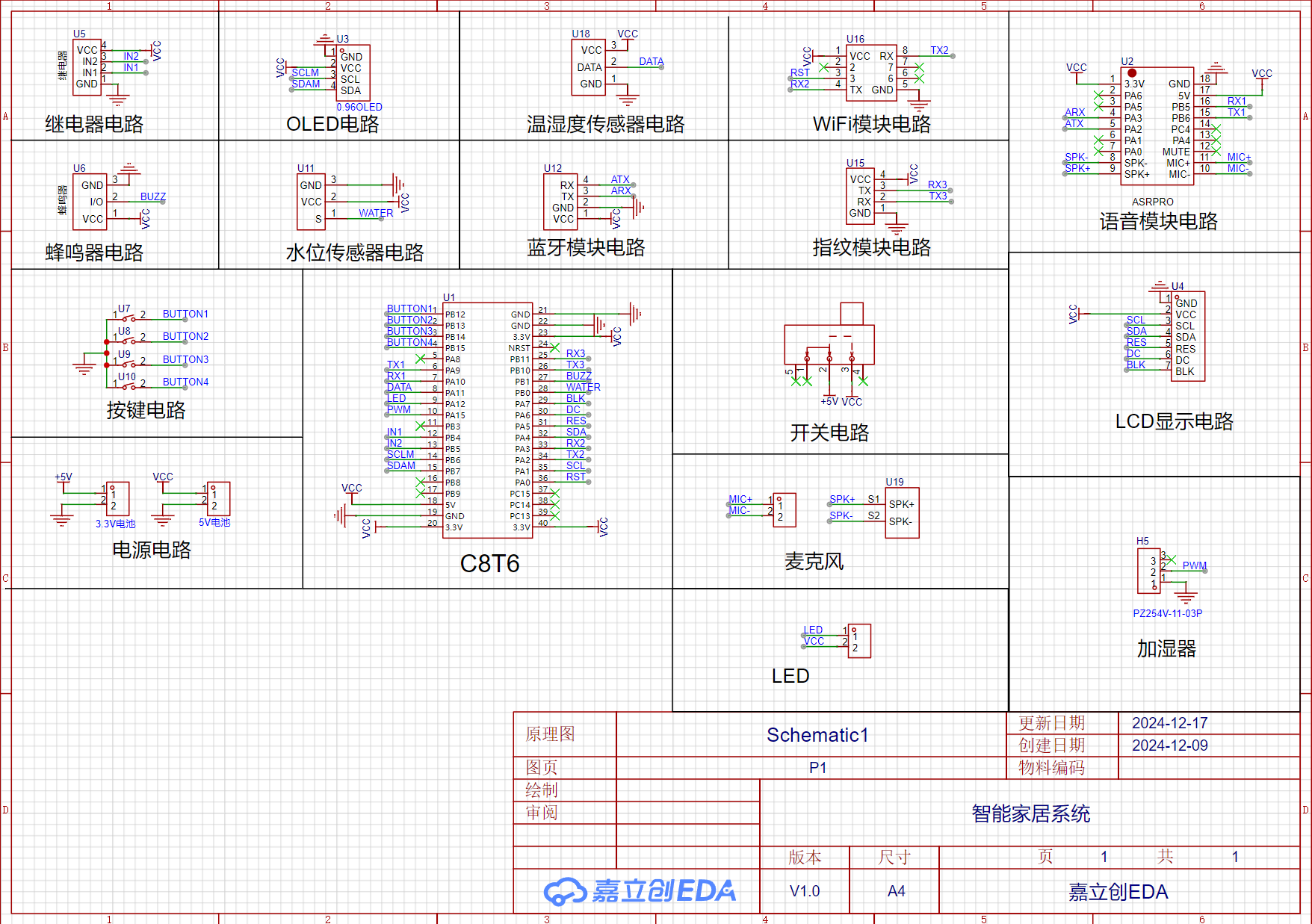


**图2 加湿器模块原理图**

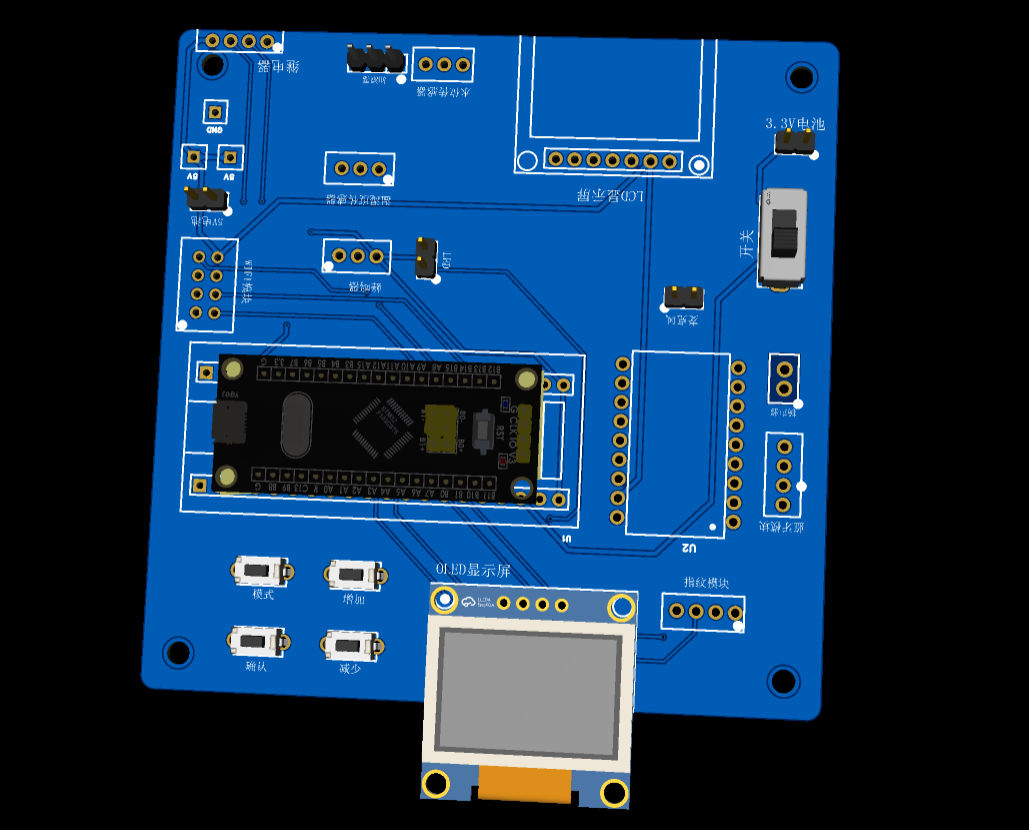


**图3 加湿器模块PCB的3D图**

因为现在市面上没有找到令我们满意的加湿器雾化模块，所以我们单独设计了一块加湿器雾化的电路，其使用NMOS管实现PWM输出110KHZ频率，外加三脚升压电感升压，利用电感的互感特性，将初级电能转化为磁能，磁能感应到次级时，次级将磁能再转换成电能，一系列的电磁转换过程，只要将初级设计成一个小感量，次级设计成一个大感量，当电感器流过一个交变的电流，就可以达到一个升压的效果，最终实现雾化片驱动。我们将其封装为模块，便于日后的插拔复用。



**图4 智能家居系统的原理图**

****

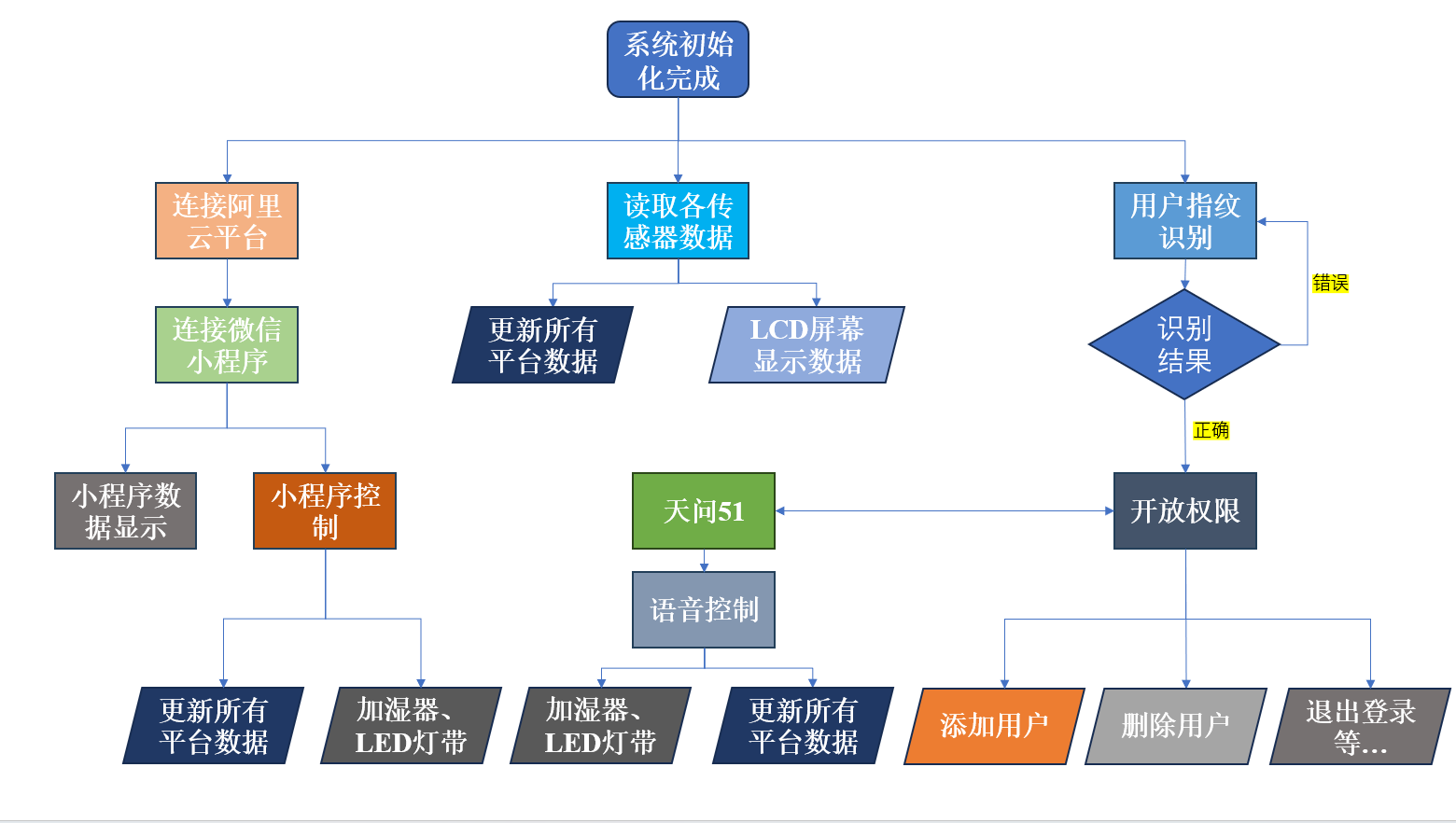
**图5 智能家居系统的PCB的3D图**

我们绘制该产品的原理图及PCB，通过焊接母座的方式可以将各元件直接插到母座上，从而实现了模块可复用的同时便于调试，并优化产品体积。

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **功能** | **引脚** | **复用功能** |
| **ESP01SWiFi模块** | 通过WIFI连接阿里云平台，实现手机互联，传递信息 | PA2、PA3 | TX2、RX2 |
| **ASRPRO语音模块** | 语音交互、控制系统 | PA9、PA10 | TX1、RX1 |
| **麦克风** | 语音交互的输入端 | MIC+、MIC- | MIC+、MIC- |
| **喇叭** | 语音交互的输出端 | SPK-、SPK+ | SPK-、SPK+ |
| **1.3寸LCD显示屏** | 显示温湿度等信息 | PA1、PA4、PA5、PA6、PA7 | SCL（时钟线）、SDA（数据线）、RES（复位线）、DC（数据/命令选择线）、CS（片选）、BLK（背光控制线） |
| **0.96寸OLED** | 用户身份识别的登录和操作系统界面 | PB6、PB7 | SCL（时钟线）、  SDA（数据线） |
| **指纹识别模块** | 用户的身份识别 | PB10、PB11 | TX3、RX3 |
| **蓝牙模块** | 通过串口与其他设备传递信息、互联 | PA2、PA3（ASRPRO） | TX、RX |
| **水位传感器模块** | 监测水位 | PB0 | ADC12\_IN8 |
| **加湿器模块** | 加湿空气 | PA15  COM2、NO2（继电器） | (重映射)TIM2\_CH1\_ETR  高压隔离 |
| **温湿度传感器** | 采集温度湿度信息 | PA11 | (重映射)DATA（单总线） |
| **继电器模块** | 控制电路通断 | PB4、PB5 | 普通IO口 |
| **红色LED** | 报警提示 | PA12 | 普通IO口 |
| **有源蜂鸣器** | 实现报警功能 | PB1 | 普通IO口 |
| **按键** | 用户身份识别的操作控制 | PB12、PB13、  PB14、PB15、 | 普通IO口 |
| **5V电源模块** | 供电 | 5V、VCC | 供电 |
| **LED灯链** | 模拟灯光控制效果 | COM1、NO1（继电器） | 高压隔离 |

该项目所用到的所有引脚对应及功能见上表。

**2. 嵌入式软件设计**

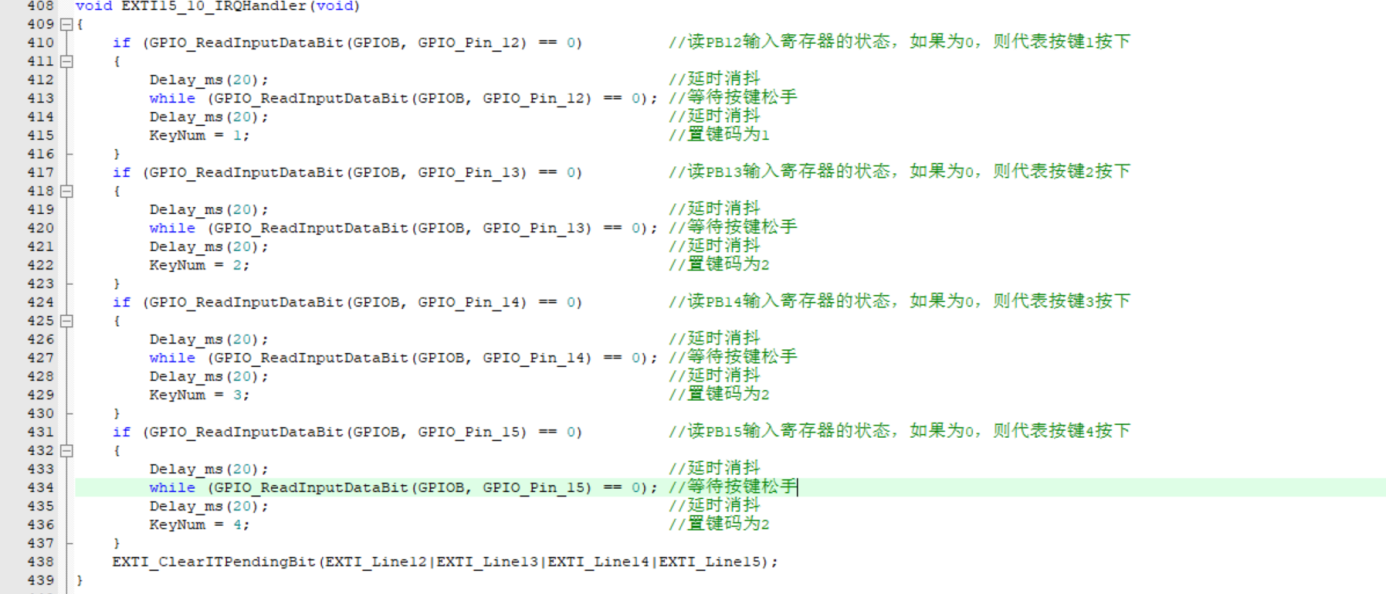


**图6系统软件流程图**

系统初始化完成之后，STM32通过ESP01S连接阿里云平台、连接小程序，小程序端可以显示更新后的数据并通过云平台控制继电器的开关；STM32不断读取传感器的数值，在LCD屏幕更新数据显示，同时定时更新所有平台数据。STM32通过指纹模块进行用户指纹识别，识别通过后开启权限，可以进行添加用户，删除用户，退出登录等操作，同时开放权限给天问模块进行语音控制。

**2.1 按键的EXIT中断读取及操作UI**





**图7按键初始化及中断读取**

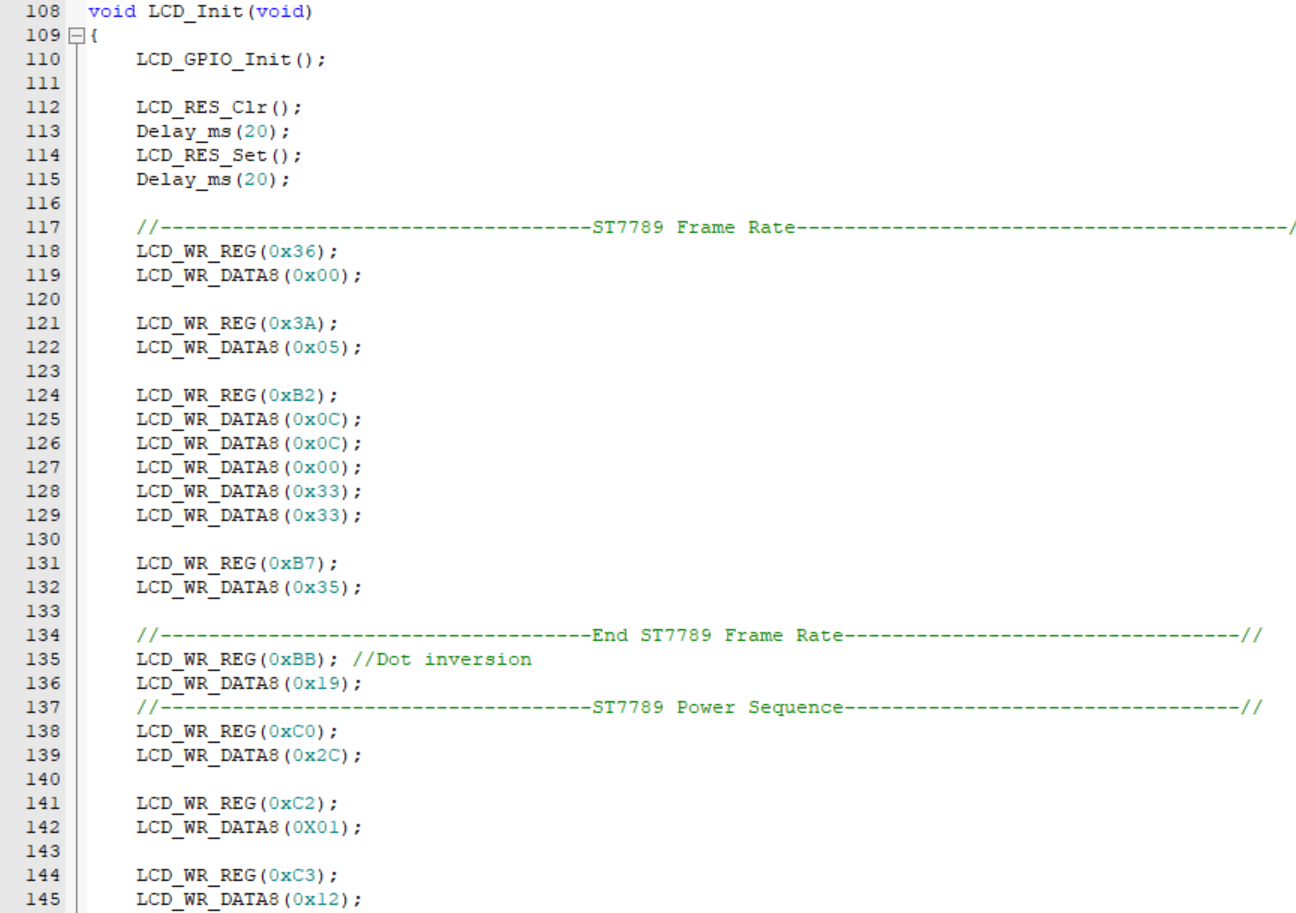
先开启GPIO、EXIT和AFIO的时钟，将四个按键配置为输入上拉模式，设置中断分组，开启EXIT的下降沿中断模式。当有按键按下时，这四个按键对应的端口会检测到下降沿，进而程序会进入 “EXTI15\_10\_IRQHandler”中断服务函数，在中断服务函数中，进行软件消抖并检测哪个端口的电平为低电平，并完成键码赋值。至此实现使用EXIT中断方式读取按键值，



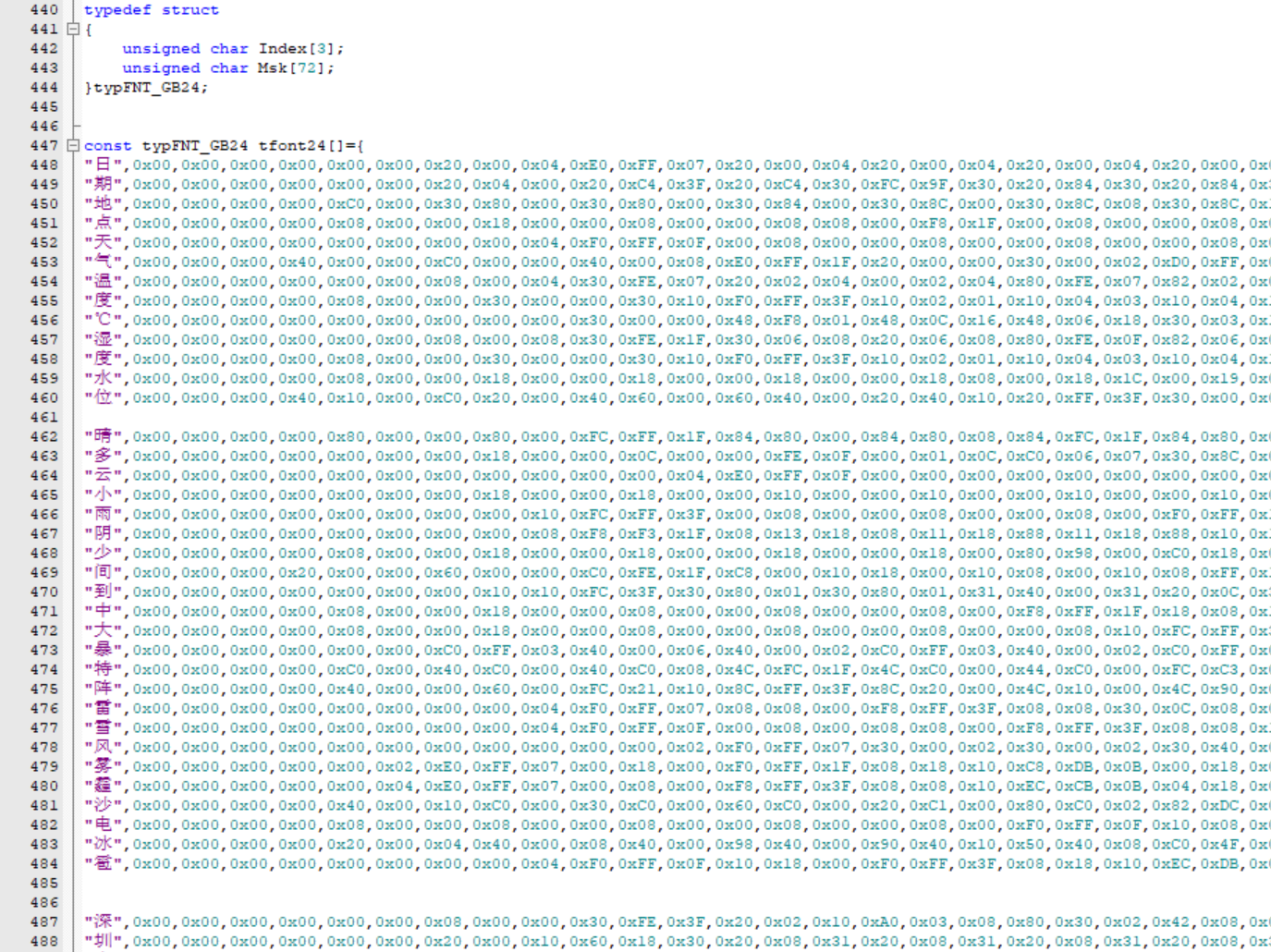
**图8按键控制界面UI的显示**

Key\_show 函数实现了一个基于按键输入的状态机，用于控制 OLED 屏幕上的显示内容和执行相应的功能。这个函数通过检查全局变量 KeyNum 和 KeyFlag 的值来决定显示哪些UI界面和执行哪些操作。

**2.2 LCD屏幕驱动及UI显示**



我们使用软件SPI的方式，完成了屏幕初始化驱动的编写。



**图9 LCD屏幕中文汉字的显示驱动**

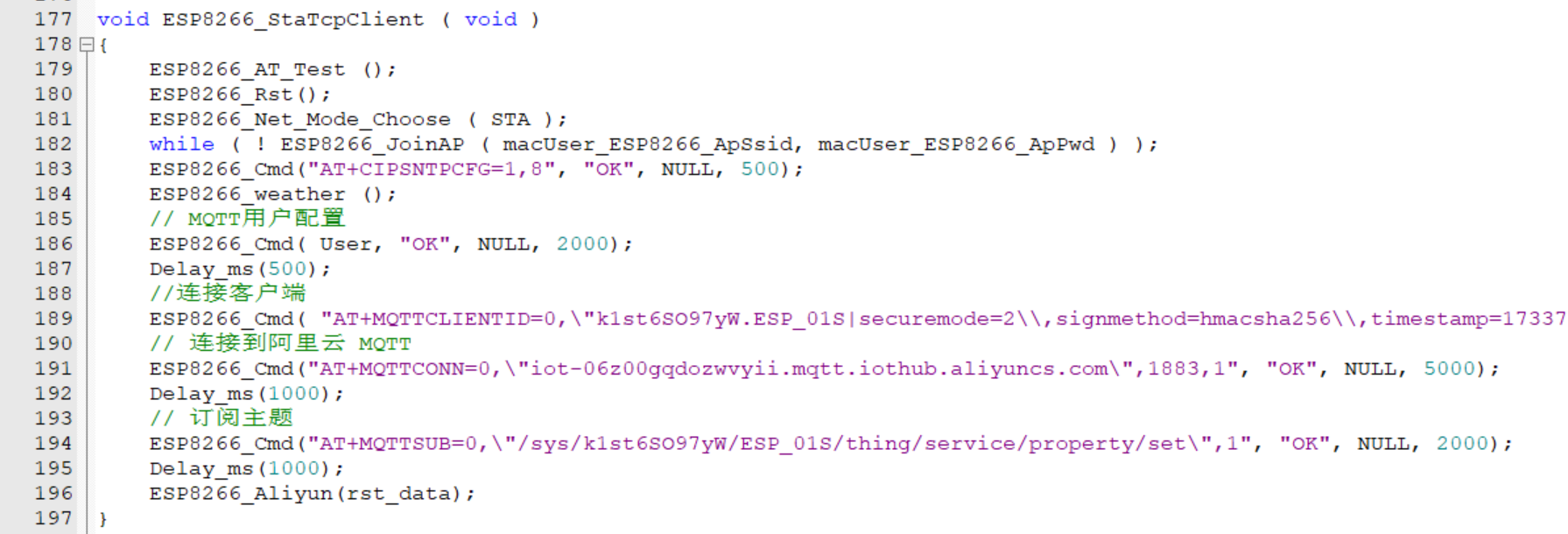
定义结构体typFNT\_GB24为24号大小得汉字，其索引大小为3\*8位的长度（因为我们使用的汉字编码为UTF-8，其为3个字节编码一个汉字），其字模数据长度为72\*8位的长度。再定义结构体数组tfont24[]其中包含所有使用的所有汉字索引及字模。在中文汉字的显示驱动中，使用4个for循环，完成对输入字符串的显示操作。



**图10 main函数中初始化LCD屏幕UI界面**

在系统初始化时刷新屏幕并显示图片和基础的UI界面，等待初始化完成，同时将图片显示函数放置与循环while(1)之外，避免一直刷新显示图片浪费系统资源。

**2.3 ESP01S的驱动及连接阿里云发送及接收数据**



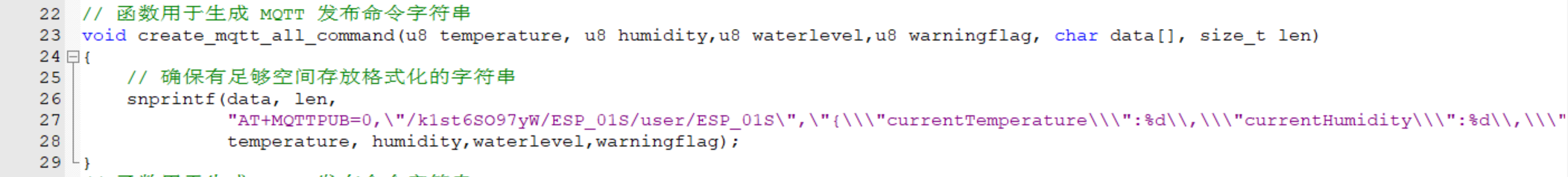
**图11 ESP01S的MQTT连接**

ESP01S烧录MQTT的固件库后才可以进行MQTT开发。先完成ESP01S对应串口引脚的初始化后， ESP8266\_StaTcpClient()使用，让ESP01S，连接WIFI，调用知心天气的API获取时间、天气、地点等数据，再连接阿里云平台的MQTT，并订阅对应的主题，完成数据传输链路的构建。



**图12 JSON数据包的解析界面**

通过解析串口接收到底JSON数据包，进行对应的变量赋值，完成数据的解析。



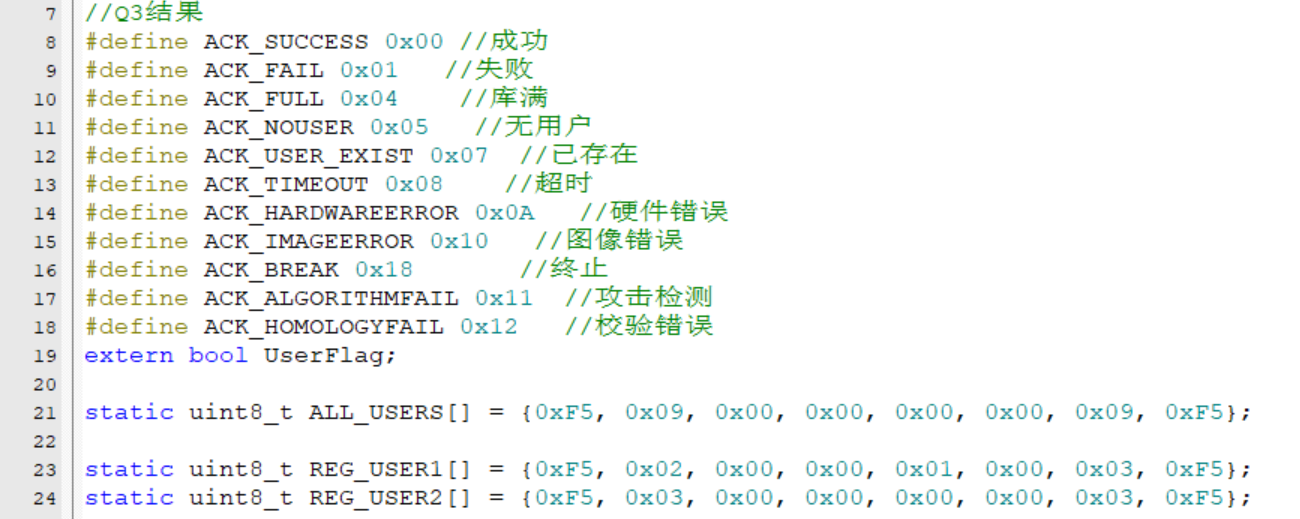
**图13 JSON数据包的构建**

通过构建对应的JSON数据包格式，并通过串口发送出去，完成数据发送。

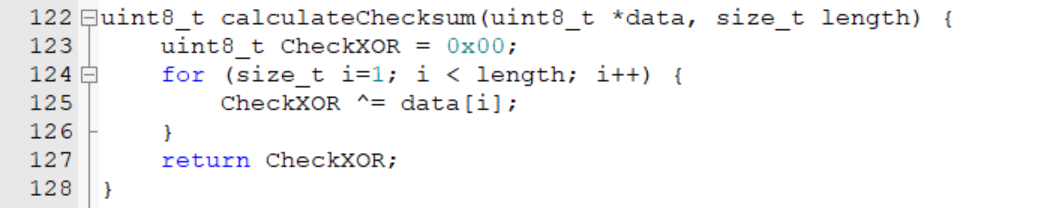
**2.4 指纹模块驱动代码的构建及编写**



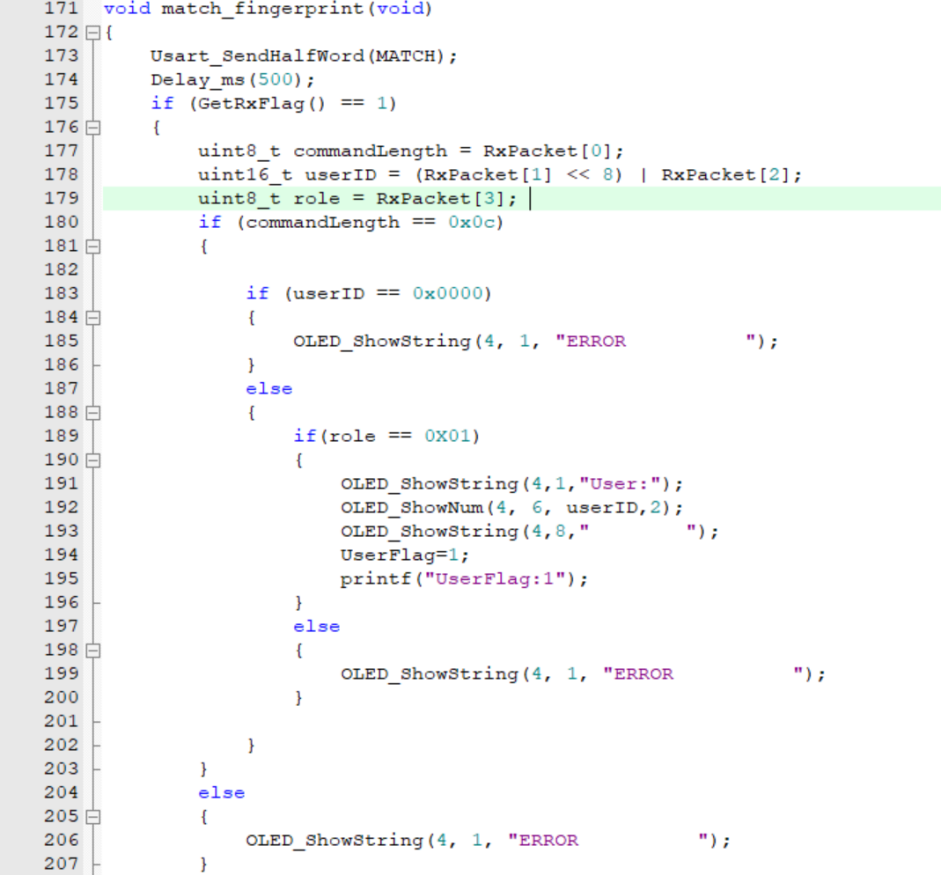
**图14 指纹某块的数据手册**

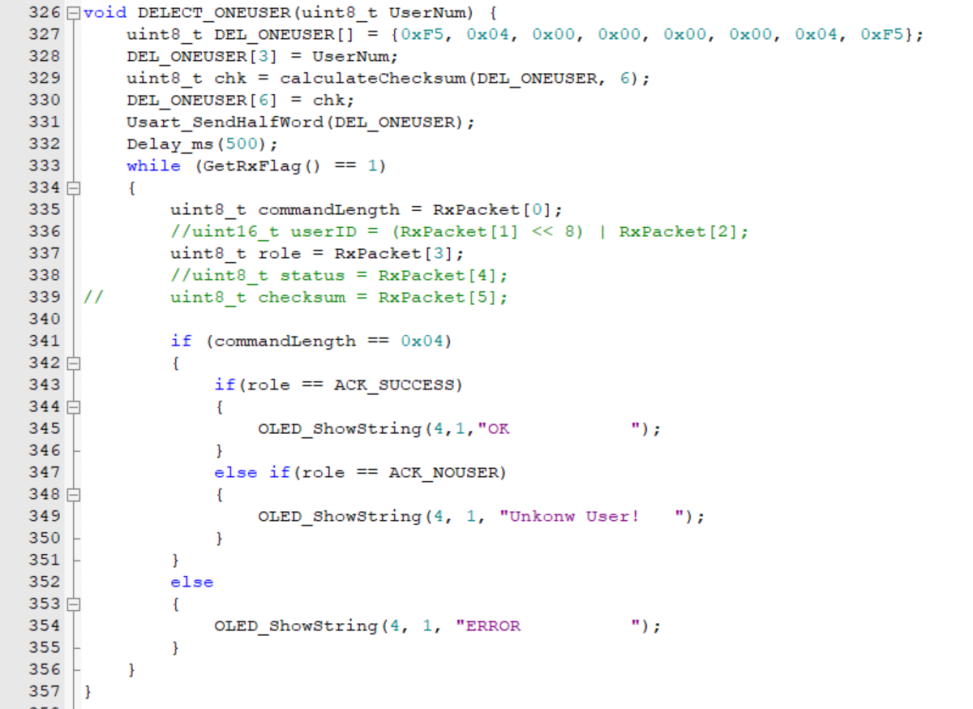


**图15 定义返回命令的类型**

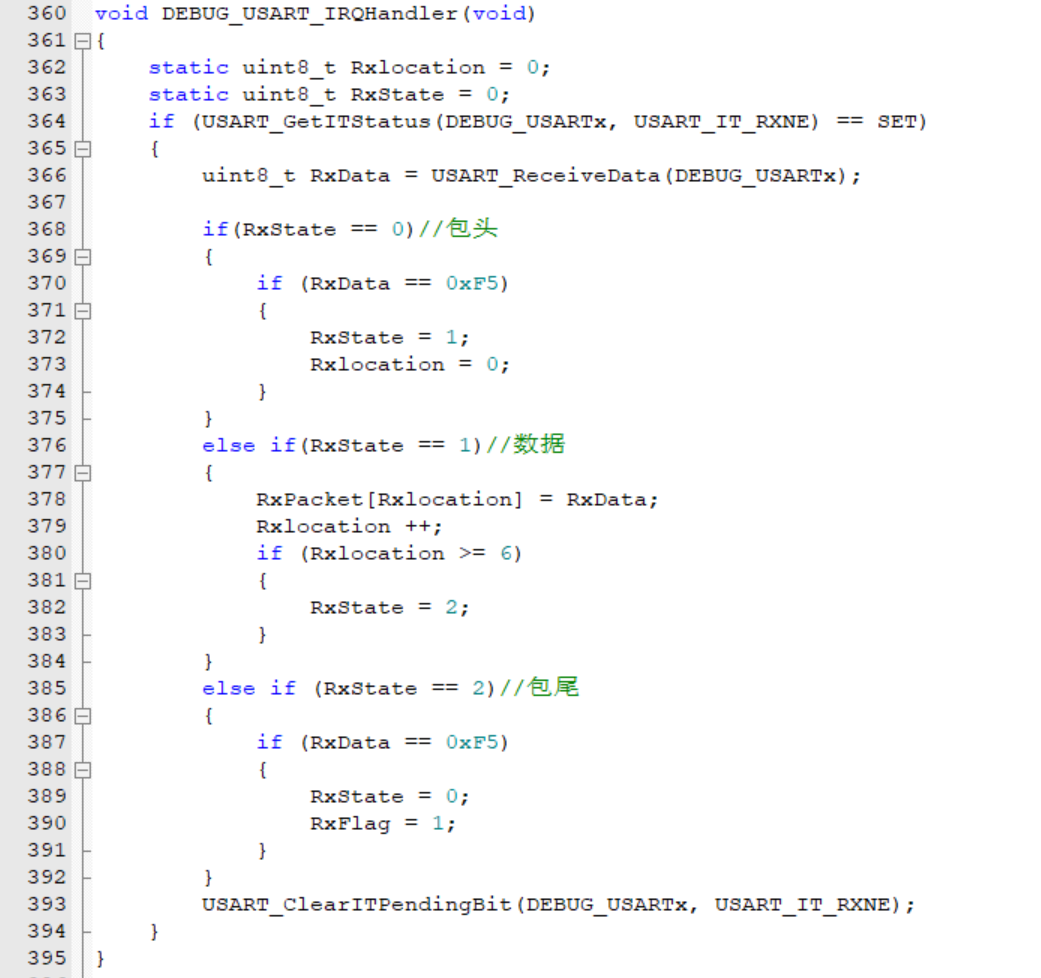


**图16 CHK异或校验和函数**





**图17 指纹匹配及删除用户函数**



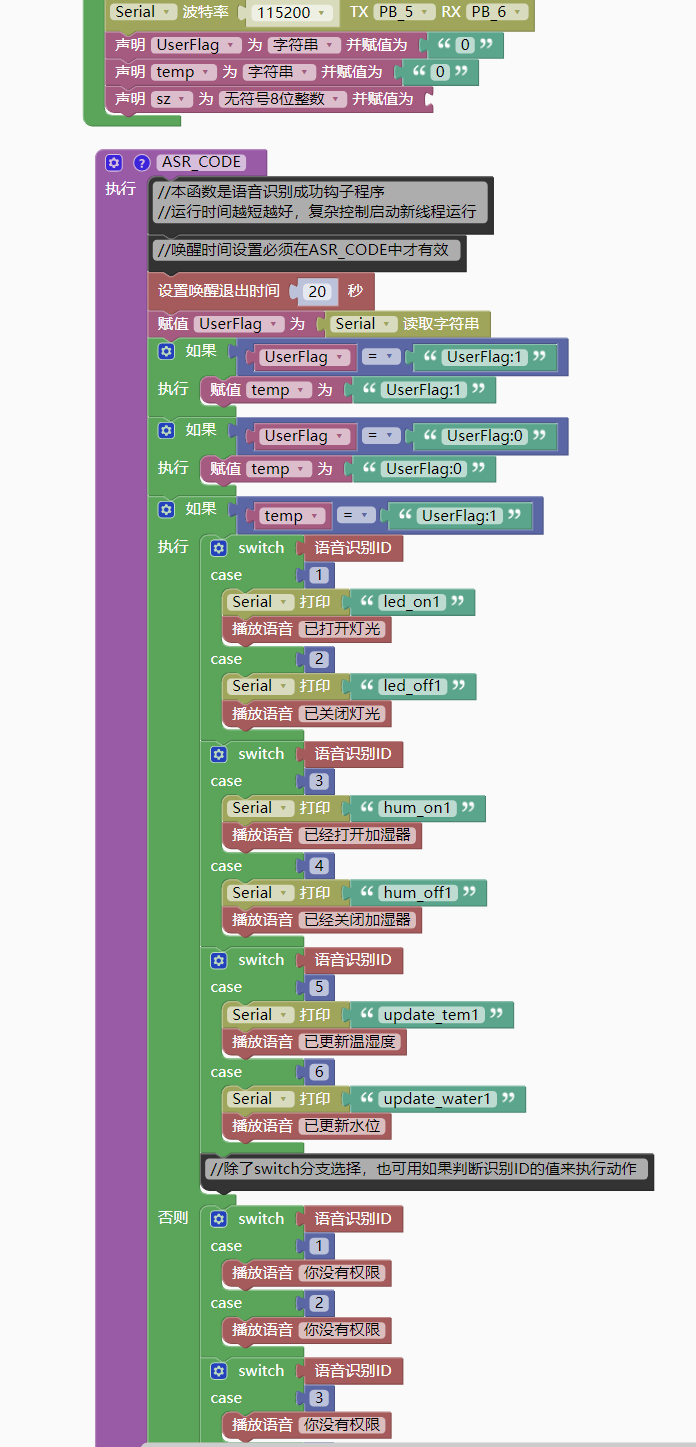
**图18 使用状态机接受数据**

按照数据手册中的数据，对指纹模块发送对应的函数，并解析返回的数据，从而判断指纹匹配及删除用户的状态。

**2.6 天问51语音模块驱动代码的构建及编写**



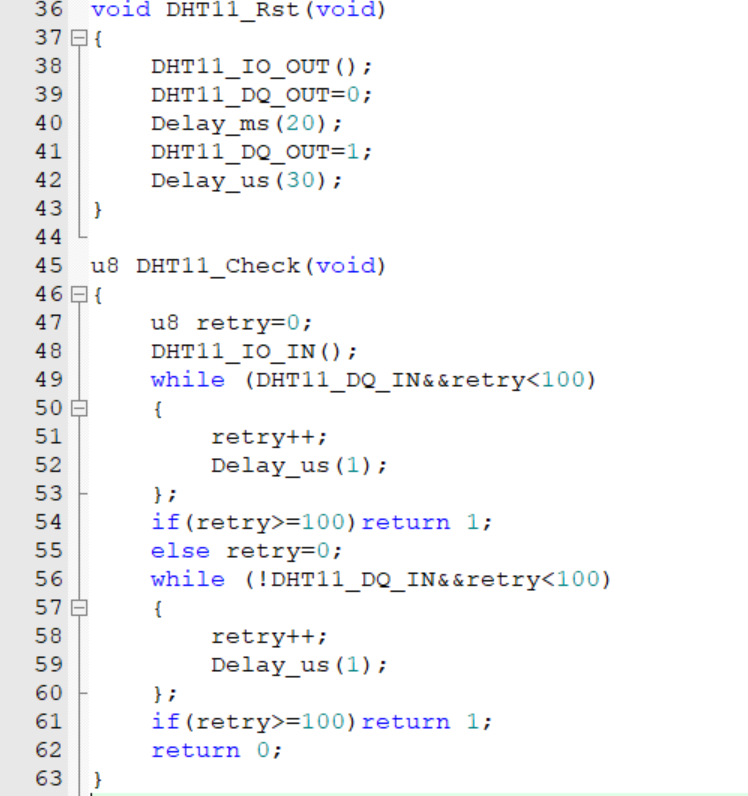
**图19 使用状态机处理天问51发送来的数据**



**图20 天问51语音模块对应的图形化编程**

使用串口完成天问51语音模块与STM32的信息交流，并同时向小程序端发送信息，同步各个平台的信息。

**2.8 温湿度传感器的构建及编写**

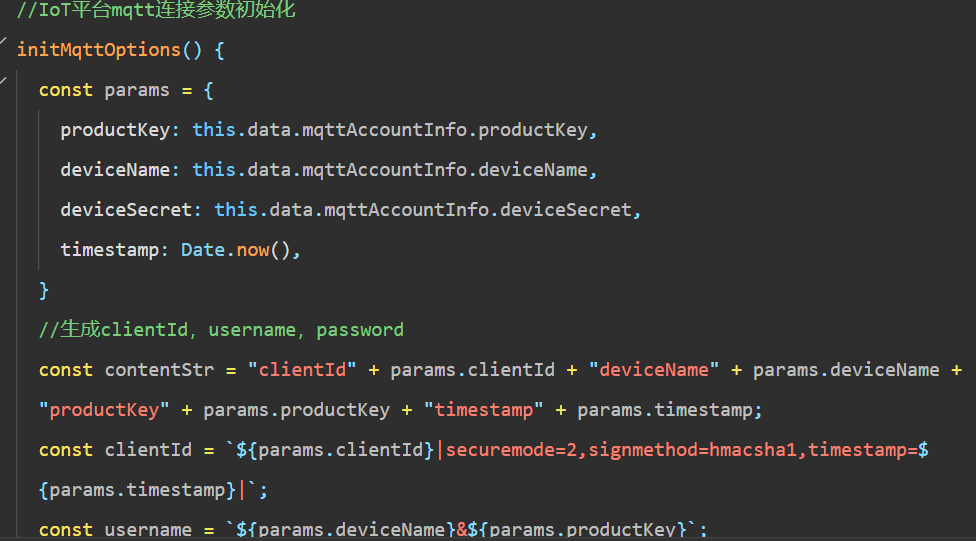




**图20 IO口模拟单总线通信**

初始化 DHT11 传感器，发送开始信号（Start Signal）以准备接收数据，再检测 DHT11 是否响应了启动信号，并确认是否准备好传输数据，最后从DHT11中读取5字节的数据，将读取5字节的数据验证校验和后，取整数赋值给温、湿度变量。

**2.7 微信小程序的构建及编写**



**图21 与云平台连接初始化**

该部分初始化首先获取阿里云连接的三元组，然后根据MQTT的相关要求生成身份信息、用户名称和密码，以通过平台验证与平台互联。



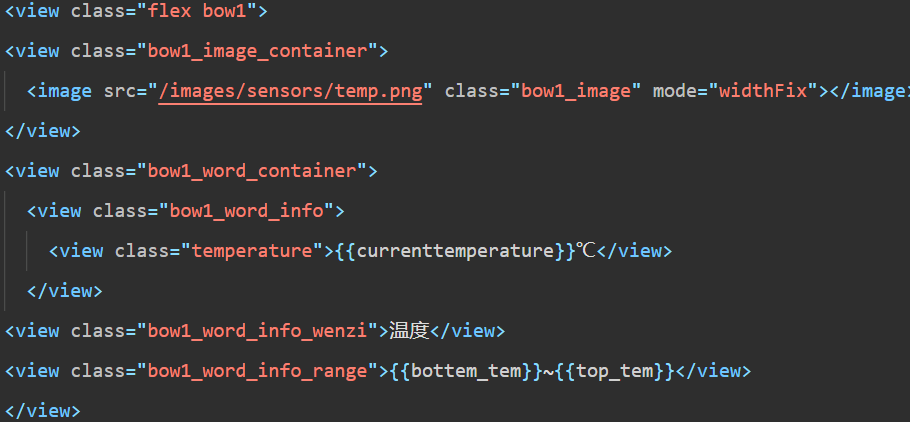
**图22 小程序处理接收消息**



**图23 小程序构建消息发送**

接收云平台信息，解析信息，如果关键字符不是未定义，即能在收到的信息找到该关键字（图中是currenttemperature），即将关键字部分的信息赋值给程序中所定义的变量，以便更新显示（if内代码即展示了赋值部分），其他传感器的接收解析同上。

发送消息部分，先构建sendData，然后进入判断与云平台是否相连。确认连接后将sendData的内容以JSON字符串格式发送，通过publish 方法向指定的主题（topic）发布消息。





**图24 小程序UI搭建代码**

小程序的UI搭建尤其需要注意其中包含关系，并且wxml中开的窗口和wxss的窗口格式设置也是需要一一对应，以上图为例简单说一下窗口搭建部分，设置父级窗口为bow1（flex bow1），在其下包含4个子级窗口（bow1\_image\_container等），子级窗口中填入需要显示的东西，上图即添加了图片，文字和变量显示。然后在wxss中设置每个窗口的大小宽度以及颜色和与其他窗口间的间距。

**三、调试及功能展示（25分）**

**1. 调试过程**

**1.1 硬件资源的冲突及引脚的重映射问题**

**TIM2的定时器中断和PWM的时钟输入使用了同一个信号源：**原定的1s中定时器中断速度非常快，导致程序移植卡在中断处理函数中。经过一系列排查之后，更换了定时器中断的信号源，完成了资源的冲突。

**温湿度传感器引脚未复用导致的数据无法接受问题：**代码编写无误但是一直接受不到数据。经过一系列排查之后，我们添加这两行代码，开启AFIO，并使能JTAG调试，完成引脚未复用问题。

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO,ENABLE);

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);

**1.2 LCD显示解码汉字时的乱码问题**

**阿里云平台和STM32对汉字编码方式不同：**导致接受的数据显示会出现乱码。经过一系列排查之后，锁定到汉字编码格式的问题。阿里云平台使用的汉字编码格式为UTF-8（三个字节编码一个汉字），开始时我们使用的LCD汉字索引为ANSI编码（两个字节编码一个汉字）。我们将汉字编码格式统一为UTF-8后，汉字的显示不再出现乱码问题。

**1.3 阿里云平台与微信小程序的连接问题**

**微信小程序的编写与阿里云平台的搭建使用不同的云平台流转规则：**数据可以从单片机发送至微信小程序，但单片机无法接受微信小程序发送的数据。我们一起修改了微信小程序的代码，使用了同一套云平台流转规则，解决了数据的互发和接收的问题。

**2. 功能展示**

****

**图25实物图**

****

**图26小程序图片**

**四、总结与心得（15分）**

**1. 项目总结**

这次基于STM32和ESP01S的智能家居系统的项目，从构思到最终的实现，每一个阶段都充满了挑战与惊喜，让我门对物联网（IoT）技术有了更深入的认知和了解。通过我们的努力，这次的简易智能家居系统项目已经基本实现并完善，但其仍然有着一些缺点和不足。因为时间紧迫，虽然目前的状态已经达到了我们的预期，但是后续时间我们还会继续深入开发修复BUG、增加更多的功能和实现更好的UI界面等。通过这次项目开发的，我们认识对于团队的任务分配及代码综合，依然存在着准备工作不足、沟通较少导致任务重复代码冲突等问题。但是这次项目的基本实现，也是对我们团队合作的认可。未来，我们会更加注重团队合作能力的综合锻炼。

**2. 项目不足与展望**

2.1 在OLED屏幕上移植u8g2库，实现中文显示并完善用户识别检测的UI界面。

2.2 完善指纹某块的用户注册时，可能出现的数据解码异常的问题。

2.3 完善ESP01S接受微信小程序传输的数据时，可能出现的数据解码异常的问题。

2.4 完善天问51语音模块实现更多的功能。如：温、湿度、水位状态和报警状态的播报等…

**3. 心得体会**

在这次基于 STM32 和 ESP8266 的简易智能家居系统项目中，我不仅掌握了多项嵌入式开发技能，也深刻体会到了团队合作的重要性。我们通过 STM32F103C8T6、指纹识别模块、ESP01S Wi-Fi 模块以及天问51的ASRPRO 语音模块等为核心，构建了一个完整的硬件平台。通过编写代码、驱动和通信协议，我们成功实现了远程监控和控制家居设备、用户识别和语音交互等功能。其中我完成了实现与云平台互联、屏幕驱动的编写、显示模块的编写、指纹模块的编写、原理图以及PCB设计、代码整合与调试、PCB焊接与测试和期末报告的编写。这些内容不但使我的自我学习能力得到了提升、综合提高了我的嵌入式开发能力，也让我对未来充满信心，激励我在嵌入式开发及物联网领域继续前行探索，创造更多的可能性。

**4. 小组分工**

**袁宏阳：**微信小程序UI搭建及小程序与云平台的互联、继电器模块的编写、原理图以及PCB设计、代码整合与调试、PCB焊接与测试、期末报告的编写。

**裴文成：**实现与云平台互联、屏幕驱动的编写、显示模块的编写、指纹模块的编写、原理图以及PCB设计、代码整合与调试、PCB焊接与测试、期末报告的编写。

**骆嘉怡：**传感器的驱动的实现，天问代码编写、期末PPT的编写。

**五、参考文献及主要代码（5分）**

**[1]** [**基于C语言从0开始手撸MQTT协议代码连接标准的MQTT服务器，完成数据上传和命令下发响应(华为云IOT服务器)\_mqtt具体实现 实例c-CSDN博客**](https://blog.csdn.net/xiaolong1126626497/article/details/134755897?spm=1001.2014.3001.5506)

[**https://xiaolong.blog.csdn.net/article/details/134755897**](https://xiaolong.blog.csdn.net/article/details/134755897)

**[2]** [**ESP8266模块（WIFI STM32）-CSDN博客**](https://blog.csdn.net/qq_44016222/article/details/142739842?ops_request_misc=%257B%2522request%255Fid%2522%253A%252280e8c0c0d70bc9878c21905ab8a9b563%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=80e8c0c0d70bc9878c21905ab8a9b563&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-142739842-null-null.142%5ev100%5epc_search_result_base3&utm_term=%E8%BE%B0%E5%93%A5%E5%8D%95%E7%89%87%E6%9C%BAesp8266&spm=1018.2226.3001.4187)

[**https://blog.csdn.net/qq\_44016222/article/details/142739842**](https://blog.csdn.net/qq_44016222/article/details/142739842)

**[3]**[**【3.微信小程序部分(已更完)】毕设：STM32+ESP8266-01S+阿里云+微信小程序 实现智能家居\_哔哩哔哩\_bilibili**](https://www.bilibili.com/video/BV1EC411z71i/?spm_id_from=333.1387.favlist.content.click&vd_source=da00025d02f9b0390b9931992f2b5174)

[**https://www.bilibili.com/video/BV1EC411z71i/**](https://www.bilibili.com/video/BV1EC411z71i/)

**[4]** [**微信小程序开发（超详细保姆式教程）-CSDN博客**](https://blog.csdn.net/m0_64875238/article/details/127796691?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522265e30a1674ac21743fce4e672e728dd%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=265e30a1674ac21743fce4e672e728dd&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-127796691-null-null.142)

[**https://blog.csdn.net/m0\_64875238/article/details/127796691**](https://blog.csdn.net/m0_64875238/article/details/127796691)

**部分未在上文展示的主要代码：**

1. **水位传感器代码：**

#include "Delay.h" // 延时函数头文件

#include "Water.h" // 水位相关的函数头文件

uint8\_t min\_water\_leavel = 30; // 最低水位阈值（可能用于水位检测）

void Adc\_Init(void)

{

ADC\_InitTypeDef ADC\_InitStructure; // 定义一个ADC初始化结构体，用于配置ADC的参数

GPIO\_InitTypeDef GPIO\_InitStructure; // 定义一个GPIO初始化结构体，用于配置引脚

// 启用ADC1和相关GPIO的时钟

RCC\_APB2PeriphClockCmd(WATER\_RCC | RCC\_APB2Periph\_ADC1, ENABLE); // 启用与ADC1和水位检测相关的外设时钟

// 配置ADC时钟源为PCLK2的1/6，PCLK2是72MHz，因此ADC时钟为12MHz

RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6); // 设置ADC时钟为12MHz

// 配置GPIO引脚（例如PA2）为模拟输入模式，供ADC采样使用

GPIO\_InitStructure.GPIO\_Pin = WATER\_IO; // 配置水位检测引脚

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AIN; // 设置为模拟输入模式

GPIO\_Init(WATER\_PORT, &GPIO\_InitStructure); // 初始化GPIO

// 重置ADC1，清除其内部设置，以确保新的配置不会受到之前设置的影响

ADC\_DeInit(ADC1); // 重置ADC1寄存器

// 配置ADC的参数

ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent; // 设置ADC工作模式为独立模式（只有ADC1工作）

ADC\_InitStructure.ADC\_ScanConvMode = DISABLE; // 禁用扫描转换模式（单通道转换）

ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE; // 禁用连续转换模式（手动触发转换）

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None; // 不使用外部触发

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right; // 设置数据右对齐

ADC\_InitStructure.ADC\_NbrOfChannel = 1; // 只配置1个通道（假设是水位检测通道）

ADC\_Init(ADC1, &ADC\_InitStructure); // 初始化ADC1并应用配置

// 启用ADC1

ADC\_Cmd(ADC1, ENABLE); // 启动ADC1工作

// 校准ADC1

ADC\_ResetCalibration(ADC1); // 重置校准寄存器

while (ADC\_GetResetCalibrationStatus(ADC1)); // 等待校准寄存器重置完成

ADC\_StartCalibration(ADC1); // 开始校准ADC1

while (ADC\_GetCalibrationStatus(ADC1)); // 等待校准完成

}

// ch: 通道号

u16 Get\_Adc(u8 ch)

{

// 配置ADC的通道和采样时间（这里使用了239.5个时钟周期的采样时间）

ADC\_RegularChannelConfig(ADC1, ch, 1, ADC\_SampleTime\_239Cycles5); // 配置ADC1进行通道ch的采样

// 启动ADC转换

ADC\_SoftwareStartConvCmd(ADC1, ENABLE); // 启动ADC的转换

// 等待转换完成，直到ADC数据寄存器就绪

while(!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC)); // 等待ADC转换结束（EOC标志位）

// 返回ADC转换的结果

return ADC\_GetConversionValue(ADC1); // 获取ADC1转换的值

}

// 获取ADC通道ch的平均值，通过重复多次读取并取平均值来减少噪声

// ch: 通道号

// times: 读取次数，通常可以设为8次、16次等

u16 Get\_Adc\_Average(u8 ch, u8 times)

{

u32 temp\_val = 0; // 存储累加的ADC值

u8 t;

// 循环多次读取ADC值并累加

for(t = 0; t < times; t++)

{

temp\_val += Get\_Adc(ch); // 累加每次读取的ADC值

Delay\_ms(5); // 延时5毫秒，确保每次读取之间有足够的时间间隔

}

// 计算并返回平均值

return temp\_val / times; // 返回ADC值的平均值

}

1. **while（1）中的代码：**

while (1)

{

// KeyNum =Key\_GetNum();

Key\_show();

TW\_show();

if(reflag)

{

reflag=!reflag;

DHT11\_Read\_Data(&temperature, &humidity);//读温、湿度

LCD\_ShowIntNum(70,144,temperature,2,RED,LIGHTBLUE,24); //更新温湿度

LCD\_ShowChinese24x24(70+get\_number\_length\_sprintf(temperature)\*12,144,(u8\*)"℃",RED,LIGHTBLUE,24,0);//94

LCD\_ShowIntNum(70,176,humidity,2,RED,LIGHTBLUE,24);

LCD\_ShowString(70+get\_number\_length\_sprintf(humidity)\*12,176,(u8\*)"% ",RED,LIGHTBLUE,24,0);//94

warning\_leavel=humidityandtemperature\_Check(temperature,humidity);

//读水位

water\_data = Get\_Adc\_Average(8,4);

water\_data = (double)water\_data / 4096 \* 100;//更新水位

waterlevel = (u8)water\_data;

LCD\_ShowIntNum(70,208,waterlevel,2,RED,LIGHTBLUE,24);

LCD\_ShowString(70+get\_number\_length\_sprintf(waterlevel)\*12,208,(u8\*)"% ",RED,LIGHTBLUE,24,0);//94

warning\_leavel = warning\_leavel+water\_leavel\_Check(waterlevel);

}

}

1. **定时器中断代码：**

// 定时器 TIM1 更新中断服务函数

void TIM1\_UP\_IRQHandler(void)

{

// 检查定时器 TIM1 的更新中断标志是否被设置

if (TIM\_GetITStatus(TIM1, TIM\_IT\_Update) != RESET)

{

// 清除定时器的更新中断标志

TIM\_ClearITPendingBit(TIM1, TIM\_IT\_Update);

++update\_flag;

if (update\_flag >= 15) // 假设每秒中断一次，这里表示 10 秒定时器

{

update\_flag = 0;

// 调用函数创建命令字符串

create\_mqtt\_all\_command(temperature,humidity,waterlevel,warning\_leavel,data, sizeof(data));

ESP8266\_Aliyun(data);

}

if(!(update\_flag % 5))//5s定时器

{

reflag = !reflag;

}

if(!(update\_flag%2))//2s定时器0.25\*8

{

warning\_reflag= !warning\_reflag;//报警

if(warning\_reflag)

{

if(warning\_leavel>=2)

{

LED1\_Turn();

Buzzer();

}

else if(2>warning\_leavel && warning\_leavel>0)//2

{

LED1\_Turn();

GPIO\_SetBits(WATER\_PORT, BUZZER\_IO);

//Buzzer();

}

else//01

{

GPIO\_SetBits(WATER\_PORT, BUZZER\_IO);

GPIO\_SetBits(GPIOA, GPIO\_Pin\_12);

}

}

}

}

}

void TIM3\_IRQHandler(void)

{

if(TIM\_GetITStatus(TIM3,TIM\_IT\_Update)==SET)

{

// Num=(Num+1)%8;

extract\_mqtt\_flags(strEsp8266\_Fram\_Record .Data\_RX\_BUF,&jsq\_switch,&LED\_switch);

if(jsq\_switch)

{

relay\_on2(); // 打开加湿器

}

else

{

relay\_off2(); // 关闭加湿器

}

if(LED\_switch)

{

relay\_on1(); // 打开灯光

}

else

{

relay\_off1(); // 关闭灯光

}

TIM\_ClearITPendingBit(TIM3,TIM\_IT\_Update);

}

}