东華理工大學

EAST CHINA UNIVERSITY OF TECHNOLOGY

《嵌入式应用实践》报告

学　　期：2025-2026 学年第二学期

课题名称：基于STM32的人脸识别门禁系统

姓　　名：邹宇航

班　　级：2022605

学　　号：2025400015

指导老师：马善农、王新水（工程师）

东华理工大学电子信息现代产业学院

2026 年 6 月

目录

1.系统功能描述

为了提高智能门禁的安全性和控制设计成本,本文提出了基于 STM32 的人脸识别门禁系统的设计方案。系统以 STM32 单片机为控制核心,以串口触摸屏为人机交互窗口,以家用USB 摄像头为图像采集设备,以WiFi 模块传输图像数据和匹配结果,实现了刷脸开锁。通过测试,系统功能稳定,且易于实现,具有一定的应用价值[1]。系统整体分为以下功能模块，具体功能如下：

1.1 FM225 人脸注册和识别模块

人脸注册模块点击屏幕录入按钮后锁定触控、刷新引导提示，STM32 通过 UART 串口向 FM225 模组下发注册指令，模组引导采集正脸、抬头、低头、左右偏头等多方位人脸图像并提取特征本地存储，自带重复检测功能，已登记人脸不可二次录入，录入成功则更新人脸数量并上报云端，人脸重复、库存已满、活体校验异常等情况均在屏幕给出提示。人脸识别模块点击识别按钮后，人脸识别服务器调用摄像头获取照片之后，对照片进行人脸检测，通过之后的照片被认定为有效并与人脸信息库进行人脸对比，返回结果和与之匹配照片的 image_token 值，并根据相应的 image_token 值在数据库中获取用户详细信息，同时调用 FM225 模组完成人脸特征比对。

1.2 电磁锁驱动模块

电磁锁驱动模块提供开锁、上锁底层控制函数，门禁执行机构采用电磁锁模拟门锁，UI 触控点击门锁按钮或人脸识别全部校验通过时，STM32 输出控制电平调用开锁接口置位门锁状态并启动 3~10 秒自动关锁计时，超时后自动执行上锁。手动切换门锁状态或自动落锁时都会同步上报门锁状态至 MQTT 云端，指令执行后实时刷新界面门锁显示标识；若人脸验证失败、活体检测异常，门锁始终保持锁闭状态，系统会自动记录每一次通行验证记录。

1.3 GT1151 电容触摸模块

GT1151 电容触摸模块依靠标准 I2C 通信总线与主控 STM32 完成数据交互，主控周期性读取模块上传的屏幕触控坐标数据，以此精准捕捉用户的各类触摸操作；当检测到左右连续滑动触摸动作时，系统会自动切换硬件控制、门禁主页、设备监控三大显示页面，实现多界面便捷切换，若识别到屏幕定点点击行为，则通过匹配触控坐标与界面按键区域判定用户所需执行的对应功能，可完成人脸注册、人脸识别、门锁开关、蜂鸣器启停等各类控制操作；系统设置了触控屏蔽机制，在人脸采集、门锁驱动、云端数据交互等指令执行阶段临时屏蔽多余触控信号，有效规避多步操作叠加带来的误触问题，保障系统运行稳定；所有触摸滑动、按键点击动作均可做到低延迟实时响应，操作完成后主控同步刷新 LCD 显示屏内容，即时更新设备运行状态、操作结果与系统提示信息，形成流畅闭环的人机交互逻辑。

1.4 ESP8266 无线网络通信模块

ESP8266 无线网络通信模块通过 UART 串口与主控 STM32 实现高速数据交互，依托 MQTT 轻量级物联网协议接入 OneNET 云端平台，构建稳定可靠的双向通信链路：一方面，模块可实时采集并上传门锁开关状态、蜂鸣器工作状态、已录入人脸数量、陌生人识别次数、设备在线时长等核心运行数据至云端数据库，便于管理员远程查看门禁运行日志与统计信息；另一方面，模块能够实时监听并接收云端下发的远程控制指令，实现对电磁锁开关、蜂鸣器启停、人脸模组注册 / 识别模式切换等功能的远程操控，同时将设备离线、在线状态同步至 LCD 监控页面实时刷新展示。

1.5 蜂鸣器模块

蜂鸣器采用有源蜂鸣器作为声光提示器件，由 STM32 单片机 PB8 引脚完成电路驱动，依靠定时器 10ms 定时扫描驱动，不会占用主循环资源。系统设置两种报警提示模式，人脸识别开锁成功时，提示通行正常；人脸识别失败和人脸录入超时等故障时，蜂鸣器持续长鸣 5 秒且搭配 LED 灯同步闪烁报警。同时支持屏幕触控手动控制蜂鸣器启停，设备运行状态实时上传至 MQTT 云端，LCD 界面实时同步显示蜂鸣器工作状态，报警时长结束后自动恢复初始静音状态。

1.6 LCD 屏幕和人机交互

GT1151 电容触摸模块通过 I2C 总线与主控通信，主控循环读取触控坐标以捕捉用户操作；检测到左右滑动时，系统切换硬件控制、门禁主页、设备监控三大界面，识别到点击操作则匹配坐标对应按键，执行人脸注册、人脸识别、门锁及蜂鸣器控制等功能。系统在指令执行过程中屏蔽多余触控，防止误操作，滑动、点击均可实时响应，并同步刷新 LCD 界面展示最新状态与操作反馈。

1.7 OneNET 云端远程管控

ESP8266 搭建无线网络链路，基于轻量 MQTT 协议实现 STM32 与 OneNET 云平台低时延双向通信。设备定时上报电磁锁状态、蜂鸣器工况、人脸数量、陌生人通行记录等门禁数据，在云端网页可视化呈现，便于管理员查看统计。平台可下发指令远程开关电磁锁、触发蜂鸣器报警，支持调取出入日志、监测设备在线状态；本地人脸录入、识别、清空人脸库等操作结果同步上传云端留存，实现门禁状态远程查看、硬件远程控制、通行记录统一管理的一体化远程管控方案。

2.设计方案

2.1 系统硬件分析

本设计的目的是设计基于 STM32 单片机的人脸识别智能门禁系统，具体实现以下功能：

1、能够通过 FM225 摄像头模块实现人脸注册和验证功能。

2、能够通过 ESP8266 WiFi 模块实现物联网功能，连接 OneNet 云平台。

3、能够通过 LCD 显示屏实时显示系统状态、人脸信息和事件日志。

4、能够通过电磁锁继电器实现门锁的自动控制。

5、能够通过云端实现远程控制功能，包括开锁、验证、注册等操作。

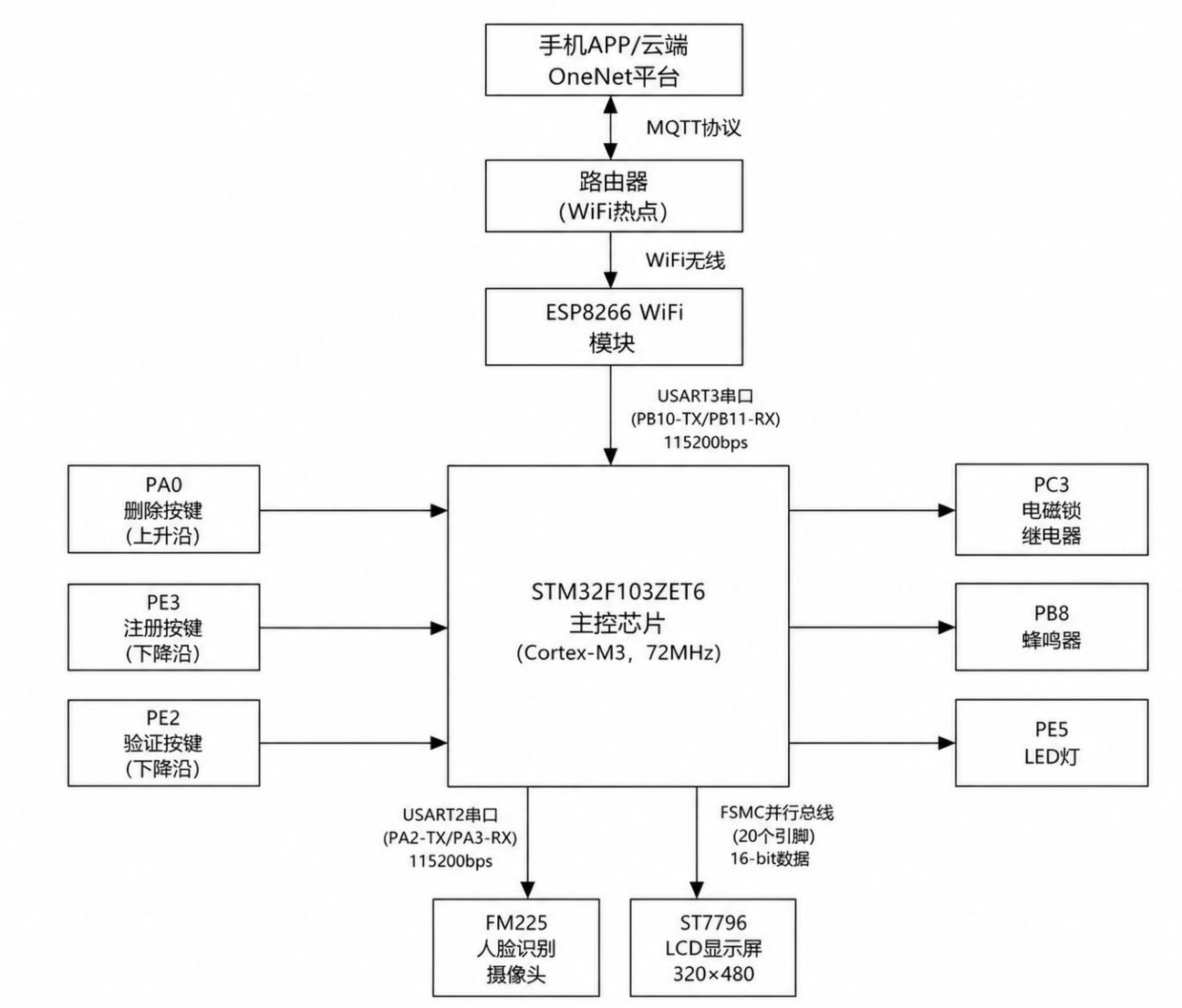

图 2-1 系统框图

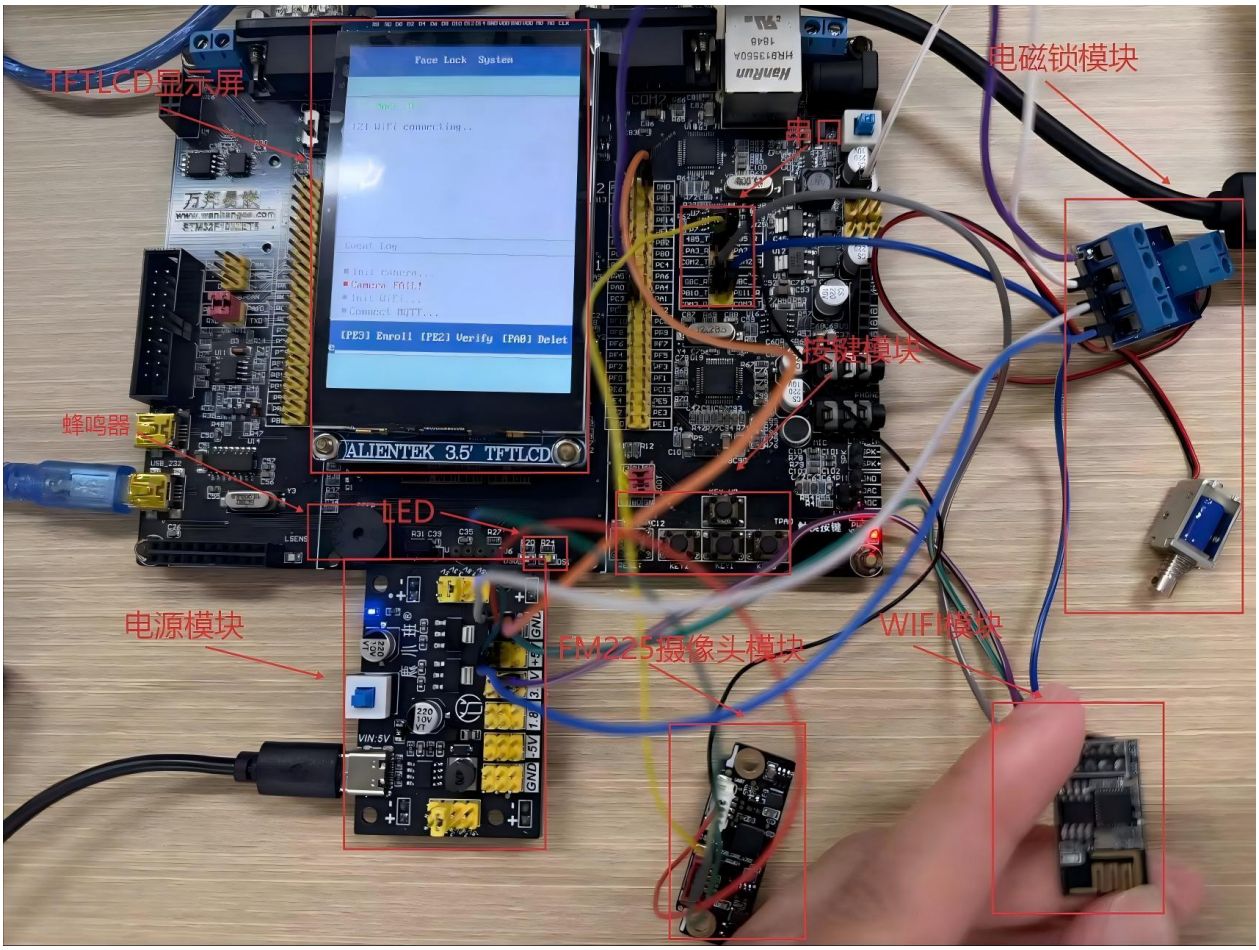

图 2-2 系统实物图

2.2 整体方案设计

为了实现人脸识别功能，需要选择合适的摄像头模块。本设计采用 FM225 人工智能人脸识别摄像头模块，该模块内部集成人脸检测、特征提取、模板存储和 1:N 比对算法，STM32 通过串口发送命令、接收结果，不运行任何图像处理算法，大大降低了主控芯片的计算负担。

为了实现物联网功能，需要选择合适的 WiFi 模块。本设计采用 ESP8266-01S WiFi 模块，该模块支持 802.11 b/g/n 协议，内置 TCP/IP 协议栈，支持 Station/SoftAP/混合模式，通过 AT 指令集进行配置和控制，工作电压 3.3V，适合嵌入式系统应用。

为了实现数据可视化显示，需要选择合适的显示器。本设计采用 ST7796 驱动的 320×480 像素 TFT LCD 显示屏，通过 FSMC 并行总线与 STM32 连接，支持内存映射方式访问，实现快速图形显示。相比 OLED 显示器，TFT LCD 具有更大的显示面积和更丰富的色彩表现。

为了实现云平台对接，本设计选择中国移动 OneNet 物联网平台，采用 MQTT 协议进行数据通信。OneNet 平台提供完善的设备管理、数据存储和指令下发功能，支持通过手机 APP 实时查看设备状态和远程控制。

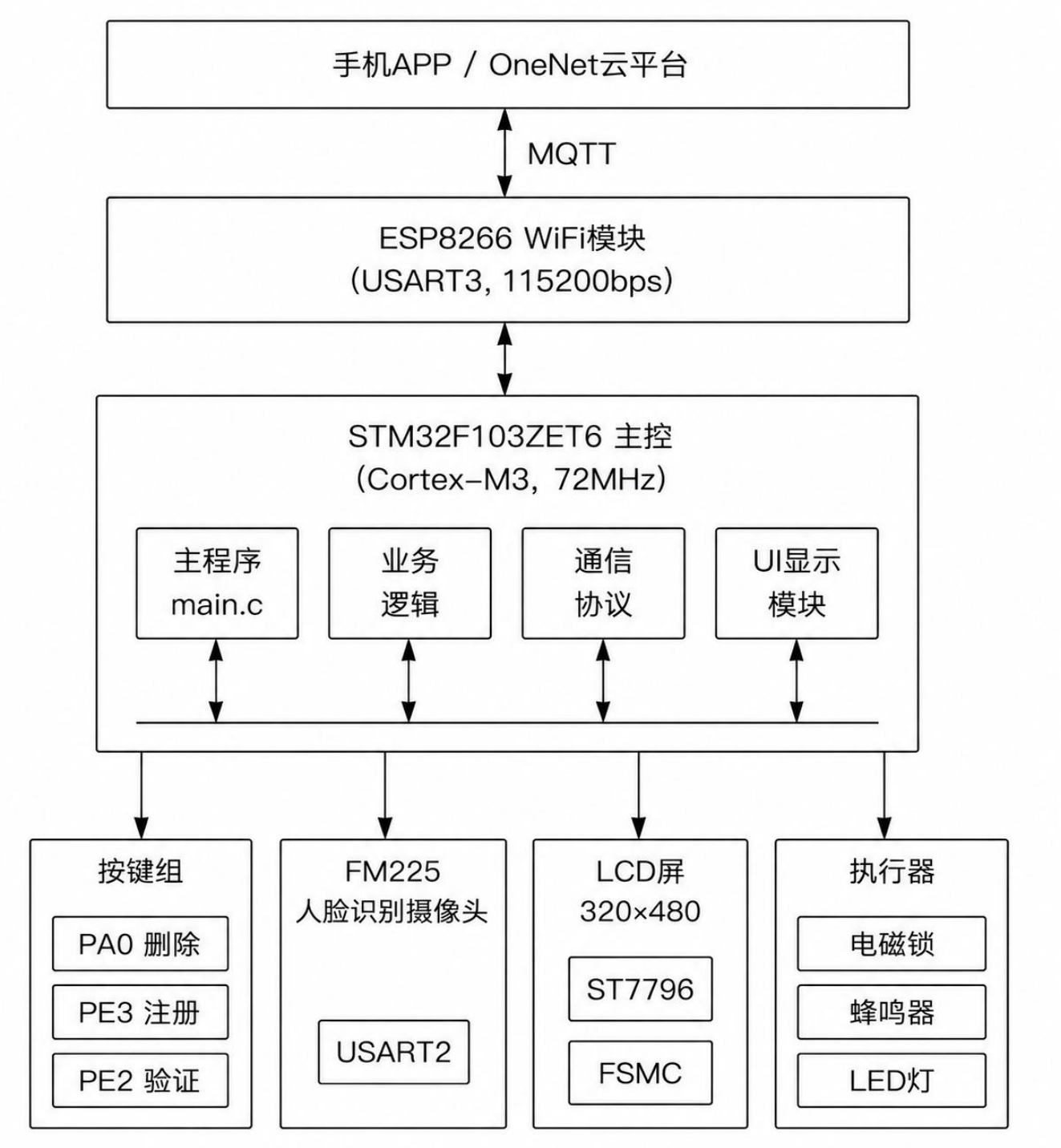

图 2-3 总体结构框图

2.3 硬件部分描述

2.3.1 微控制器电路模块

该电路模块负责数据的接收、处理、显示、传输与异常报警控制。具体的原理图如图 2-4 所示。

STM32F103ZET6 微控制器是意法半导体推出的基于 ARM Cortex-M3 内核的 32 位微控制器。主频 72MHz，具有 512KB Flash 和 64KB RAM，提供 112 个 GPIO 引脚、5 个 USART、2 个 I2C、3 个 SPI 等丰富外设资源。本项目使用 USART2 连接 FM225 摄像头（115200bps），USART3 连接 ESP8266 WiFi 模块（115200bps），FSMC 驱动 LCD 显示屏（16-bit 并行总线）。

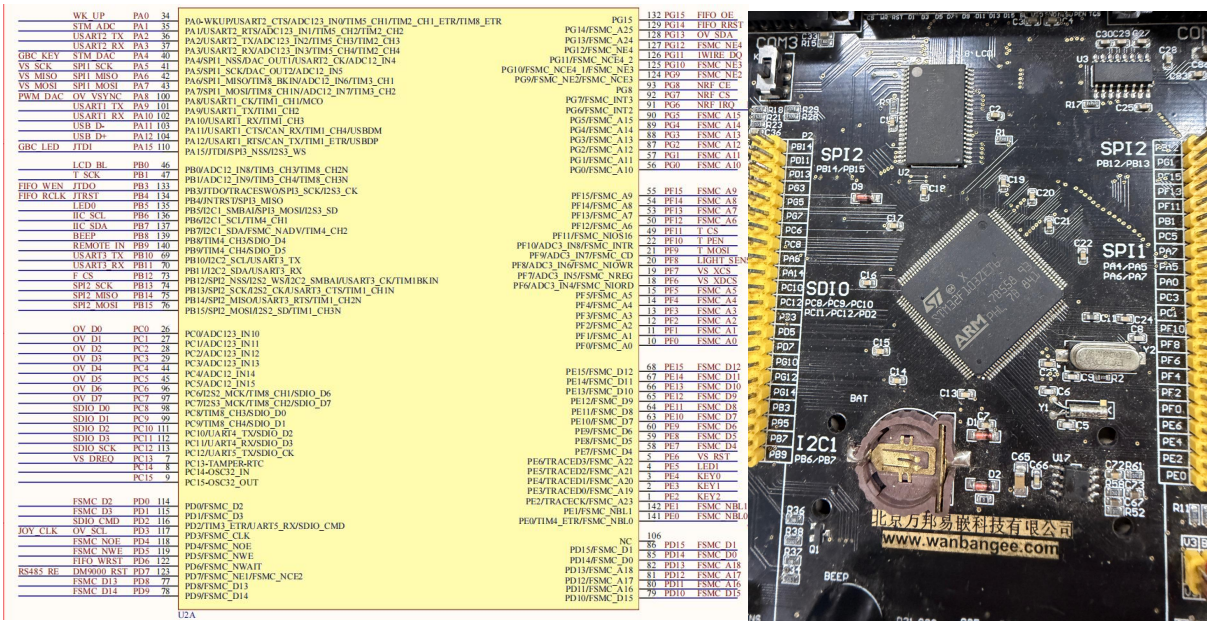

图 2-4 微控制器原理图

2.3.2 人脸识别模块

FM225（THFaceCropper）是一颗独立的 AI 视觉模块，内部集成人脸检测、特征提取、模板存储和 1:N 比对算法。STM32 通过串口发送命令、接收结果，不运行任何图像处理算法。

模块支持 5 方向人脸采集（上、下、左、右、正前方），内部 Flash 存储人脸模板，采用自定义二进制帧协议通信，波特率 115200bps。通信协议格式为：发送帧 EF AA | MsgID | Size(大端) | Data | Parity(XOR)，接收帧格式相同。

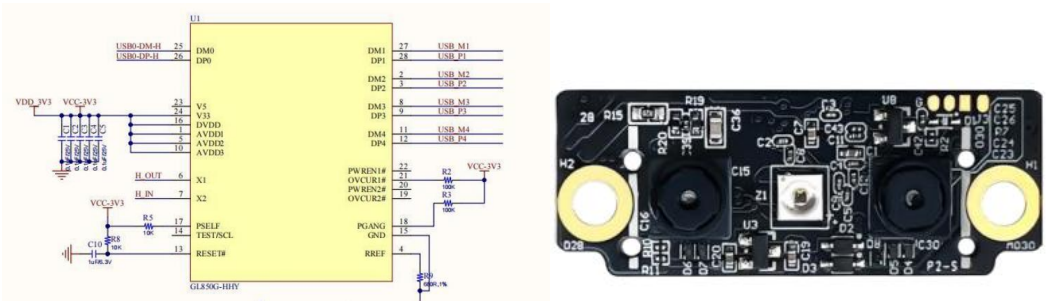

图 2-5 人脸识别模块原理图

2.3.3 WiFi 通信模块

ESP8266-01S 是一款高性能、低成本的 WiFi 模块，支持 802.11 b/g/n 协议，内置 TCP/IP 协议栈。支持 Station/SoftAP/混合模式，支持 TCP/UDP/MQTT 协议，工作电压 3.3V，通过串口 AT 指令控制。

本项目中 ESP8266 工作在 Station 模式，通过 AT 指令连接 WiFi 热点和 OneNet MQTT 服务器。主要使用的 AT 指令包括：AT+CWMODE=1 设置 Station 模式，AT+CWJAP 连接 WiFi 热点，AT+MQTTUSERCFG 配置 MQTT 客户端，AT+MQTTCONN 连接 MQTT 服务器，AT+MQTTSUB 订阅主题，AT+MQTTPUB 发布消息。

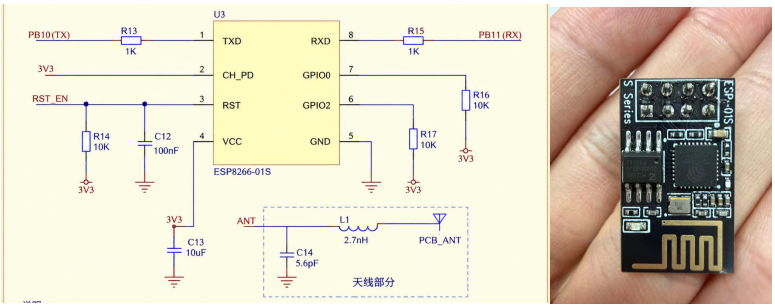

图 2-6 WiFi 通信模块原理图

2.3.4 液晶显示模块

ST7796 是一款 TFT LCD 驱动芯片，支持最大 320×480 像素分辨率，16 位 RGB565 色彩格式。通过 FSMC 并行总线与 STM32 连接，支持内存映射方式访问，实现快速图形显示。

本项目中 LCD 用于显示人脸识别状态、系统日志、操作提示等信息。屏幕布局分为 6 个区域：标题栏（y=0, 高 44px）显示"Face Lock System"标题；状态条（y=44, 高 28px）显示用户数量和锁状态；人脸信息区（y=75, 高 195px）显示人脸检测状态、坐标、角度；日志区（y=275, 高 110px）显示系统事件日志；按键提示栏（y=390, 高 40px）显示按键功能提示；统计栏（y=434, 高 46px）显示操作统计信息。

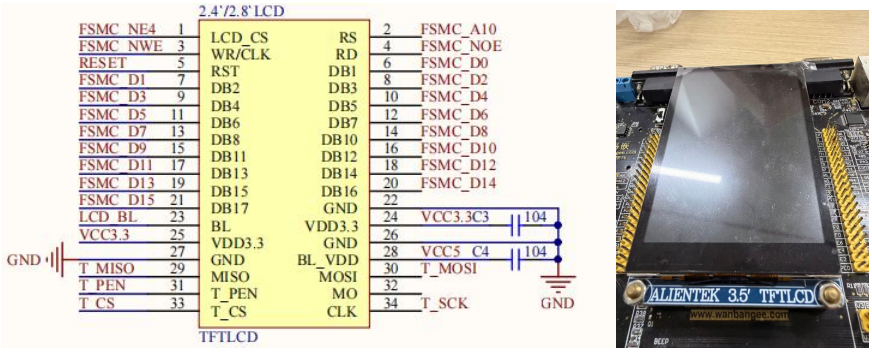

图 2-7 液晶显示模块原理图

2.3.5 电磁锁与继电器模块

系统采用继电器驱动电磁锁，实现门锁控制。继电器控制引脚为 PC3，高电平时继电器吸合电磁锁打开，低电平时继电器断开电磁锁关闭，响应时间小于 100ms。

电磁锁采用 12V 供电，通过继电器实现弱电控制强电。当 STM32 输出高电平到 PC3 引脚时，三极管导通，继电器线圈通电，触点吸合，电磁锁得电开锁。当 STM32 输出低电平时，继电器断开，电磁锁失电关锁。

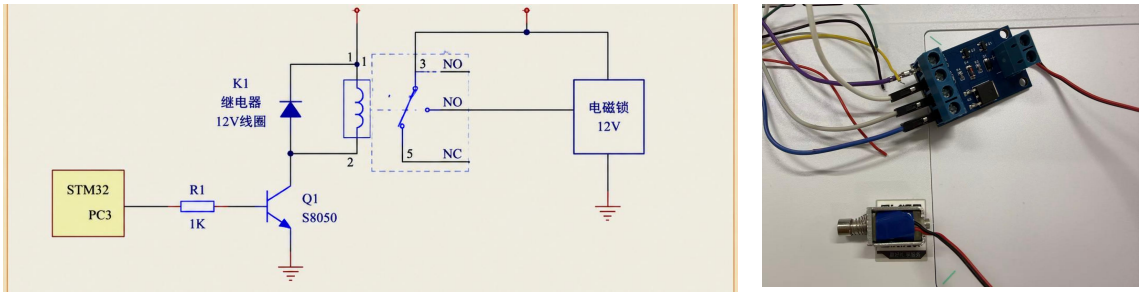

图 2-8 电磁锁与继电器模块原理图

2.3.6 蜂鸣器与 LED 指示灯模块

蜂鸣器连接 PB8 引脚，为有源蜂鸣器，高电平响。LED 指示灯连接 PE5 引脚，低电平亮。这两个模块用于操作反馈、状态指示和告警提示。

操作反馈：注册成功 LED 慢闪 3 次，注册失败 LED 快闪 1 次；验证成功 LED 慢闪 3 次，验证失败 LED 快闪 1 次。告警提示：陌生人验证失败时蜂鸣器响 300ms；云端远程报警时蜂鸣器连续响 3 次。

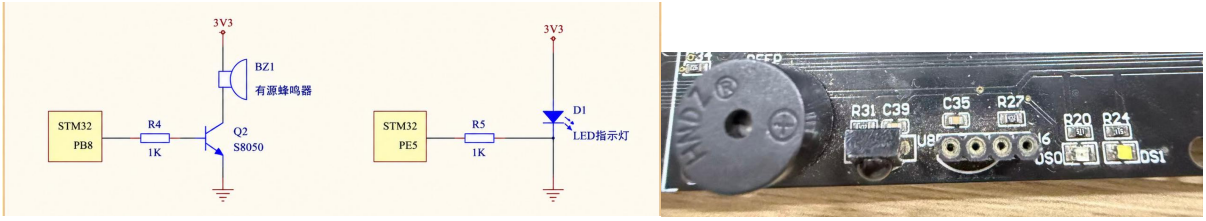

图 2-9 蜂鸣器与 LED 指示灯模块原理图

2.4 软件部分描述

2.4.1 软件结构组成

根据本设计的总体方案，该系统主要由人脸识别模块、WiFi 通信模块、LCD 显示模块、门锁控制模块和用户交互模块组成。因此，软件系统组成分为主程序模块、FM225 通信程序模块、ESP8266 驱动程序模块、OneNet 云平台对接程序模块、UI 界面显示程序模块和按键控制程序模块。具体的软件组成结构图如图 2-11 所示。

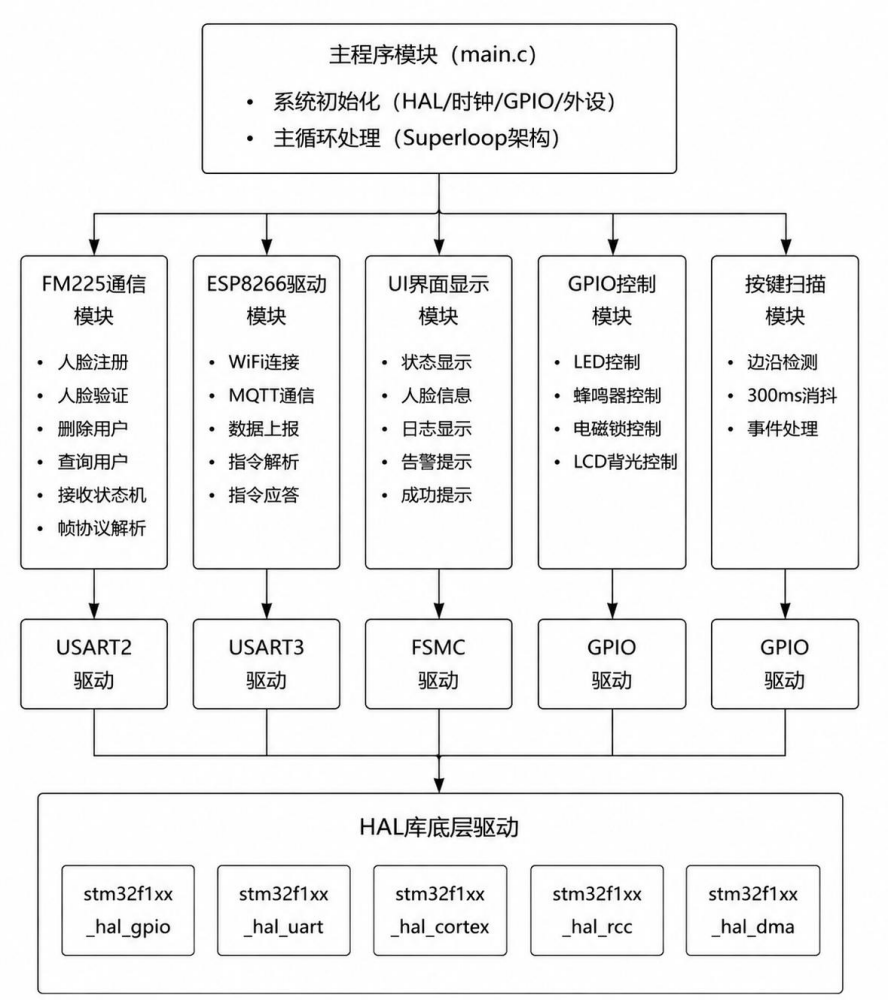

图 2-11 软件组成结构框架图

2.4.2 主程序设计

主程序采用前后台（Superloop）架构。前台为中断服务程序，负责 UART 接收和 SysTick 时基；后台为主循环轮询处理。主循环周期 100ms（约 10Hz），依次处理人脸状态更新、命令结果处理、告警清除、云端指令检查、心跳上报、按键扫描等任务。具体的程序流程如图 2-11 所示。

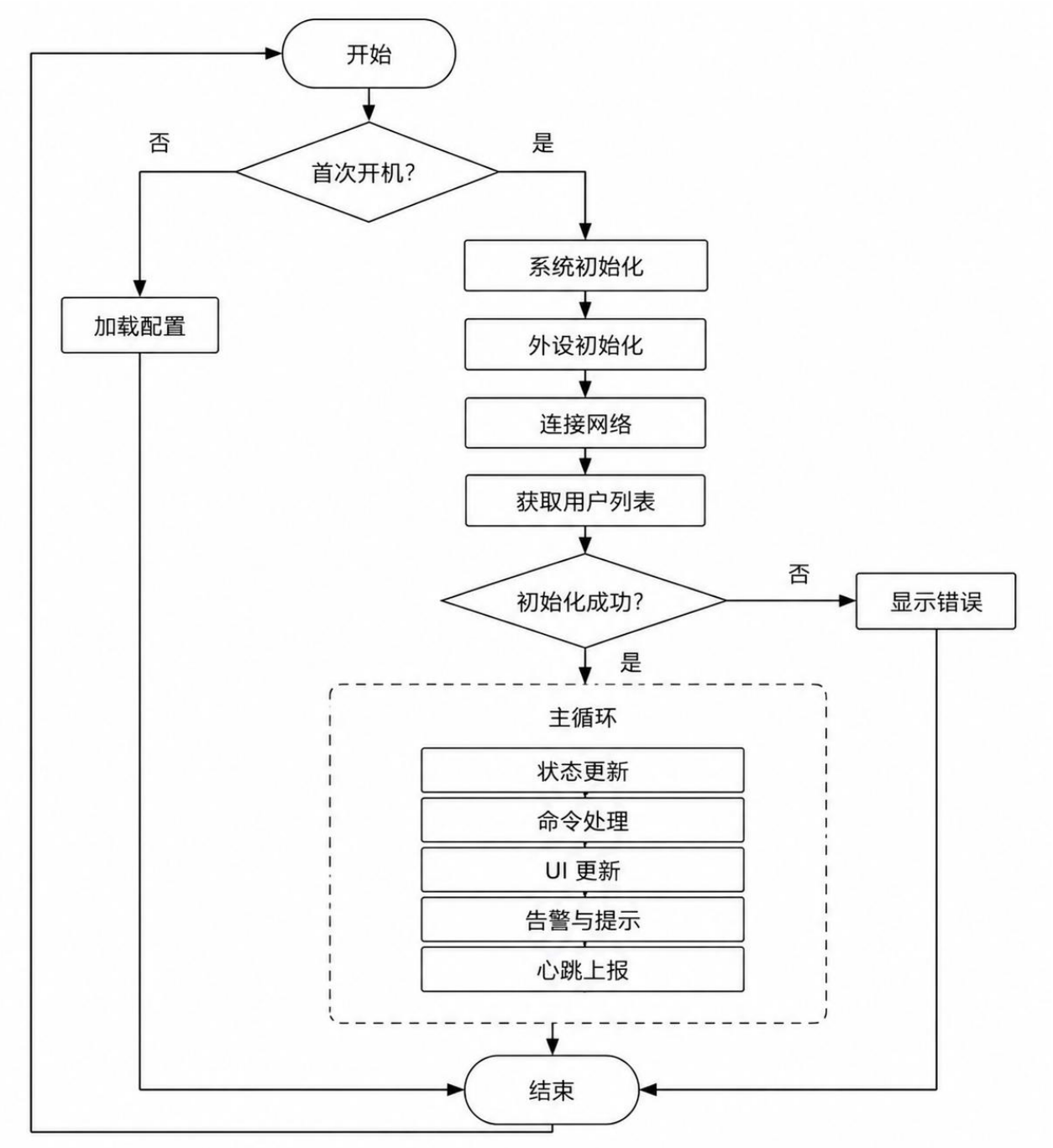

图 2-12 主程序流程图

2.4.3 FM225 通信程序设计

该程序模块负责与 FM225 人脸识别摄像头模块的通信。具体的程序流程如图 2-12 所示。

FM225 通信采用自定义二进制帧协议，通过 USART2 中断接收实现。接收状态机包含 7 个状态：等待帧头 0xEF、等待帧头 0xAA、接收消息 ID、接收长度高字节、接收长

度低字节、接收数据段、接收校验位。校验算法为 XOR 校验，对消息 ID、长度和数据段逐字节异或。

当接收状态机收到完整帧后，调用 ProcessReceivedFrame()函数处理。处理两种消息类型：MID_REPLY（命令应答）包含命令结果码和用户 ID；MID_NOTE（异步通知）包含模块就绪信号和人脸状态数据。

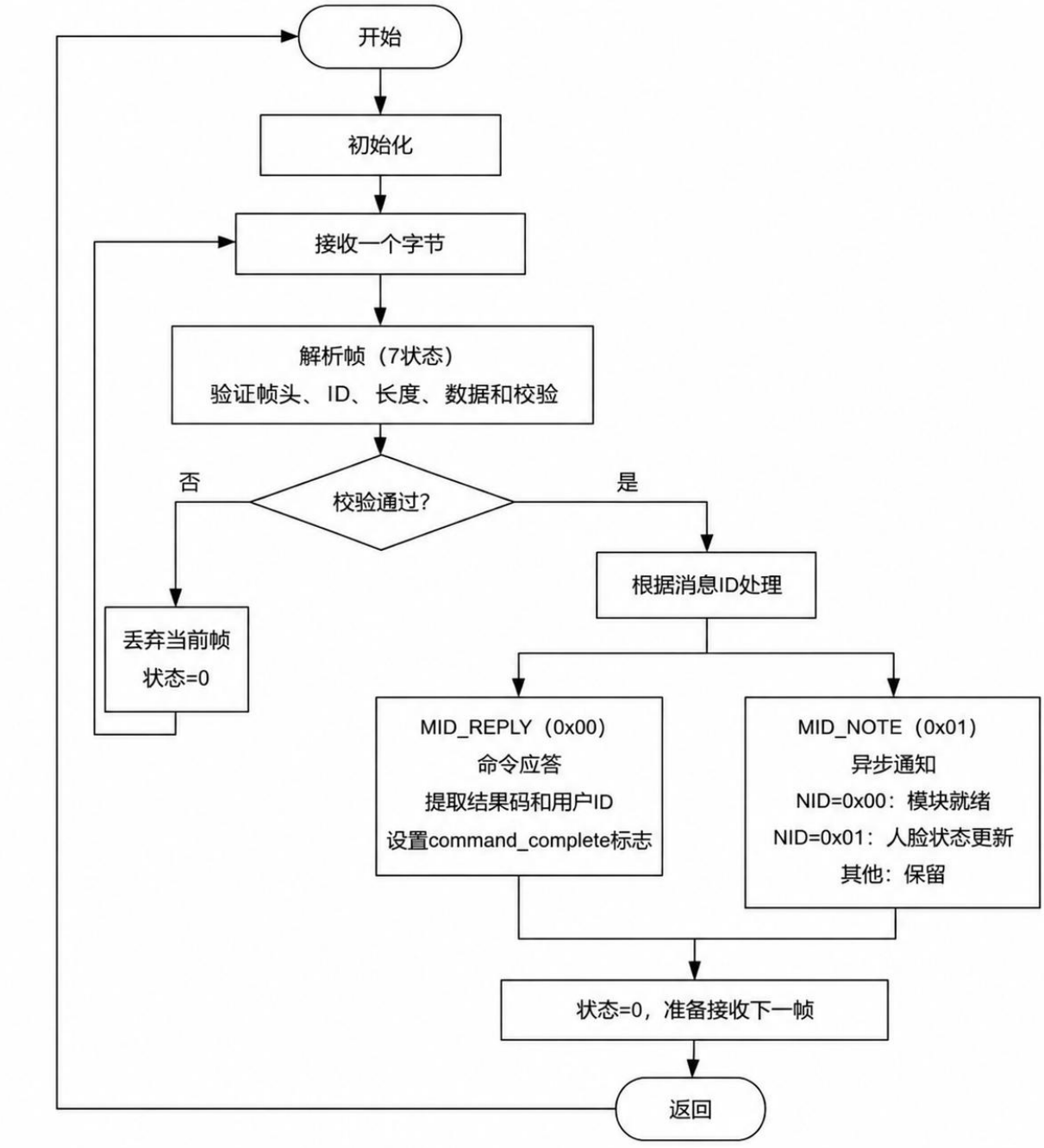

图 2-13 FM225 通信程序流程图

2.4.4 ESP8266 WiFi 通信程序设计

该程序模块负责与 ESP8266 WiFi 模块的通信以及 OneNet 云平台的对接。具体的程序流程如图 2-13 所示。

ESP8266 通过 USART3 串口通信，采用 AT 指令集进行控制。核心函数 ESP8266_SendCmd()实现发送 AT 指令并等待期望回复的功能，工作流程为：清空接收缓冲区；发送指令加回车换行；每 10ms 检查一次回复；包含期望回复返回成功；包含"ERROR"或"FAIL"返回失败；超时返回失败。

OneNet 连接包括 5 个步骤：设置 WiFi 模式为 Station；连接 WiFi 热点；配置 MQTT 客户端参数；连接 MQTT 服务器；订阅属性上报回复和属性设置两个主题。

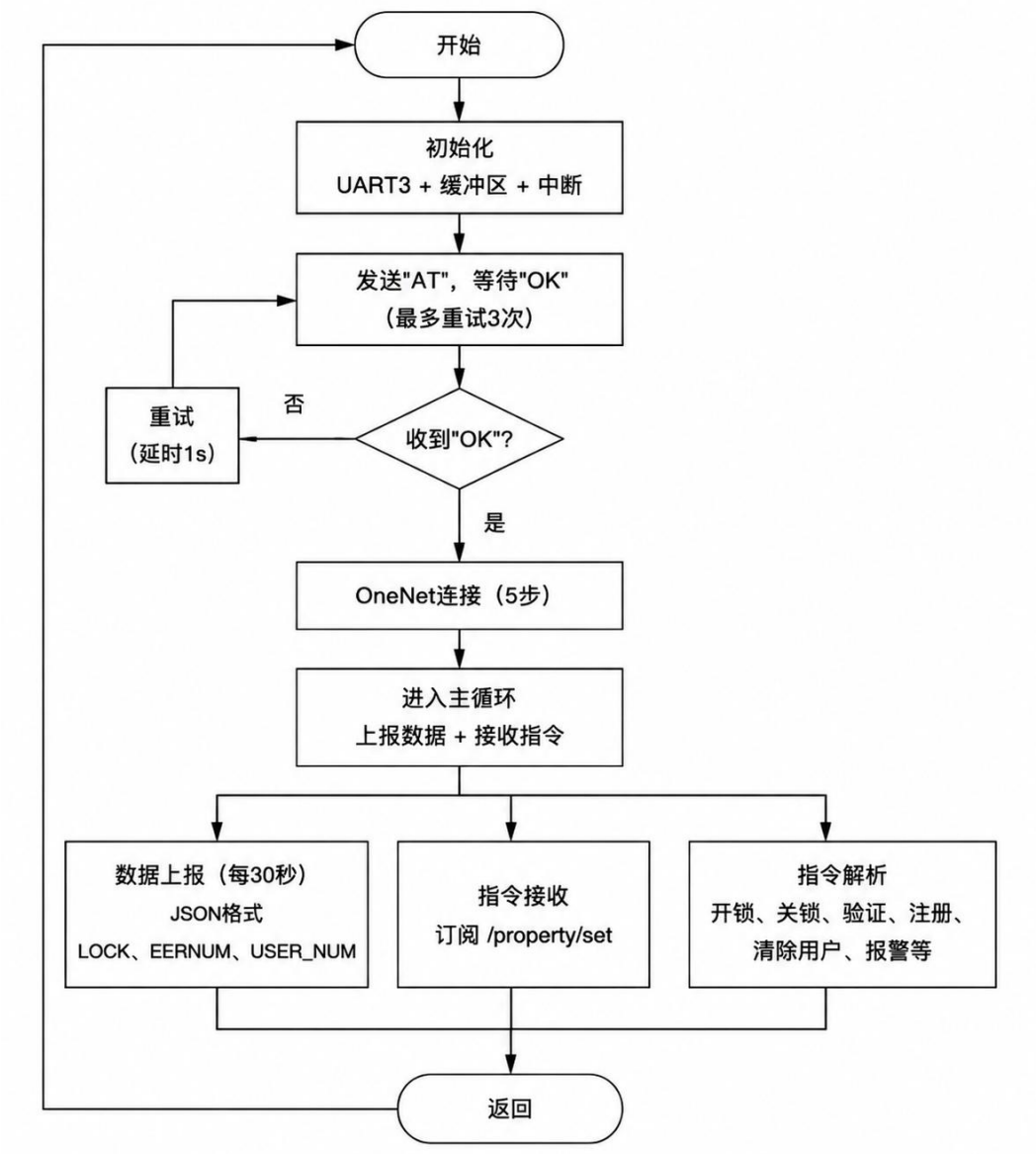

图 2-14 ESP8266 WiFi 通信程序流程图

2.4.5 云端指令解析程序设计

该程序模块负责解析云平台下发的控制指令。具体的程序流程如图 2-14 所示。

ESP8266 收到 MQTT 订阅消息后格式为"+MQTTSUBRECV:0,topic,len,payload"。系统通过 strstr()函数查找"+MQTTSUBRECV:"标记，提取 payload 内容，然后逐个检查指令关键字：OPEN_JC（人脸验证）、OPEN_LR（人脸注册）、Clear_user（清除用户）、YC_BJ（远程报警）、LOCK（开锁/关锁）。匹配成功后清空缓冲区并返回对应指令代码。

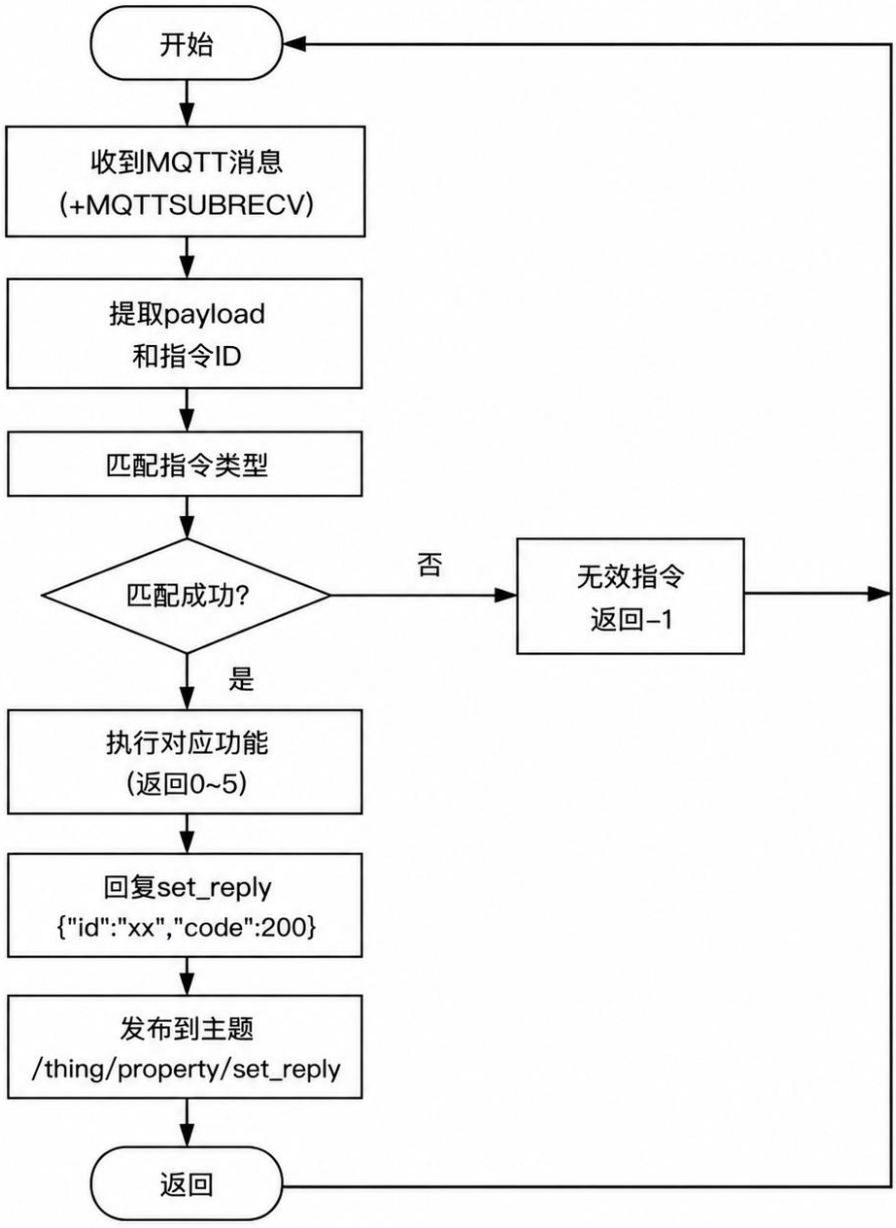

图 2-15 云端指令解析程序流程图

2.4.6 UI 界面显示程序设计

该模块负责 LCD 屏幕的界面显示功能。具体的程序流程如图 2-15 所示。

UI 模块在 LCD 驱动之上提供面向业务的界面绘制函数。主要功能包括：UI_Init()初始化界面绘制所有静态区域；UI_UpdateStatus()更新状态条显示锁状态和用户数量；UI_UpdateFace()更新人脸信息区显示状态、坐标、角度；UI_Log()添加日志到环形缓冲区，5 行日志用彩色圆点标识；UI_ShowAlarm()显示红色告警信息；UI_ShowSuccess()显示绿色成功提示。

图 2-16 UI 界面显示程序流程图

3.调试心得和收获

3.1 设计中出现的问题

在进行系统设计时，曾遇到下列问题：

（1）FM225 通信协议问题：官方文档中字段顺序与实际固件行为不一致，导致人脸状态数据解析错误，人脸框坐标和角度数据显示异常。

（2）ESP8266 AT 指令超时问题：WiFi 连接和 MQTT 连接耗时较长（5-15 秒），期间主循环被阻塞，影响系统实时性，用户可能以为系统卡死。

（3）MQTT 消息格式问题：OneNet 平台下发的 JSON 指令需要正确解析，字符串匹配容易出错，指令 ID 提取不准确，导致应答失败。

（4）LCD 显示刷新问题：频繁刷新整个屏幕导致画面闪烁，用户体验较差，特别是在人脸信息频繁更新时。

（5）按键消抖问题：机械按键存在抖动现象，一次按下可能触发多次响应，导致重复注册或验证操作。

3.2 解决办法

（1）协议问题：通过实际抓取 UART 数据，交叉验证确认正确的字段顺序，修改 ParseFaceState()函数中的数据解析逻辑，确保与实际固件行为一致。具体方法是使用逻辑分析仪捕获 FM225 发送的原始数据，逐字节分析字段含义。

（2）超时问题：在 WiFi/MQTT 连接期间，在 LCD 上显示进度提示（如"[1] Set mode..."、"[2] WiFi connecting..."），让用户了解当前状态，避免误以为系统卡死。同时在连接失败时显示具体失败原因。

（3）JSON 解析：使用 strstr()函数进行关键字匹配，通过查找""id":"字符串定位指令 ID 字段，使用 strncpy()提取 ID 值，确保应答正确。增加调试日志输出接收到的原始数据，便于排查问题。

（4）显示优化：采用局部刷新策略，只更新变化的区域（如状态条、人脸信息区），避免全屏刷新，消除闪烁现象。使用脏标志位记录需要更新的区域，减少不必要的重绘。

（5）按键消抖：采用 300ms 定时消抖，记录上次按键触发时间（HAL_GetTick()），间隔不足 300ms 则忽略当前按键，有效消除抖动影响。

3.3 心得和体会

通过这次嵌入式应用实践，我收获了很多。在之前的学习中，虽然学习过 C 语言和单片机原理，但很少有这种理论和实践相结合的课程，所以我格外珍视这次自己动手实践的机会。

在项目开发过程中，我系统地学习了 STM32 单片机的开发流程，从 GPIO、USART 等基础外设的配置，到 FSMC 并行总线、MQTT 协议等高级功能的实现。通过 10 几天的学习和在一次次的错误、调试中，我学习到了很多，这都使我受益良多。

在 STM32 单片机实验中，团队协作同样给我留下了深刻的印象。在小组项目中，与同学们共同探讨方案、分工协作，在这个过程中，我学会了倾听他人的意见和建议，学会了如何在团队中发挥自己的作用，也明白了团队的力量远大于个人自身的力量。

通过本次项目实训，我深入理解了嵌入式系统软硬件协同设计的重要性。硬件接口决定了软件架构，软件设计影响系统性能。同时，我也掌握了串口通信协议设计、物联网开发、LCD 显示等实用技术，为今后的学习和工作打下了坚实的基础。

回顾这段 STM32 单片机实验经历，我深感收获颇丰。它不仅让我掌握了一门实用的电子技术，更培养了我解决问题的能力、创新思维以及团队协作精神。我将以此为起点，继续保持对嵌入式技术的热情和探索精神，不断学习和实践，努力提升自己的专业水平。

4. 课题设计成果和展望

4.1 设计成果

（1）人脸识别功能

通过 FM225 人工智能人脸识别摄像头模块，实现了人脸注册和验证功能。注册时支持 5 个方向（上、下、左、右、正前方）的人脸图像采集，用户需在 30 秒内缓慢转头完成采集。验证时执行 1:N 比对算法，将当前人脸与所有已注册用户模板进行比对，超时时间为 10 秒。FM225 模块内部完成人脸检测、特征提取、模板存储和比对计算，STM32 通过串口发送命令和接收结果，大大降低了主控芯片的计算负担。

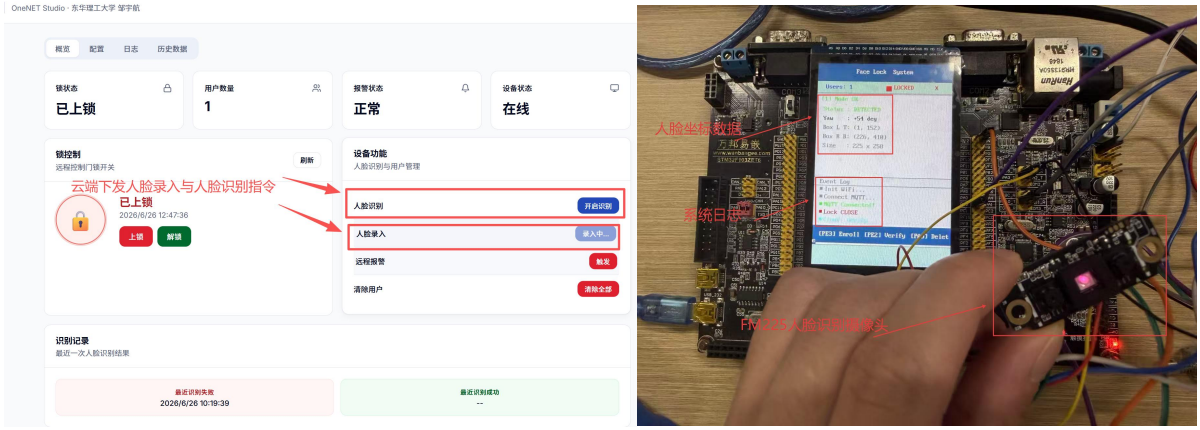

图 4-1 人脸识别注册功能实验结果

（2）WiFi 物联网功能

通过 ESP8266-01S WiFi 模块，实现了物联网功能。系统采用 AT 指令集配置 ESP8266，连接 WiFi 热点后通过 MQTT 协议接入中国移动 OneNet 物联网平台。系统每 30 秒自动上报一次心跳数据，包括锁状态和已注册用户数量，保持设备在线状态，让手机 APP 能实时查看设备状态。

图 4-2 WiFi 连接功能实验结果

（3）LCD 显示界面功能

通过 ST7796 驱动的 320×480 像素 TFT LCD 显示屏，实现了丰富的图形界面显示。屏幕分为 6 个功能区域：标题栏显示"Face Lock System"标题；状态条显示用户数量和锁状态；人脸信息区显示人脸检测状态、偏航角和框坐标；日志区显示系统事件日志，采用 5 行环形缓冲区设计；按键提示栏显示三个按键的功能提示；统计栏显示操作统计信息。

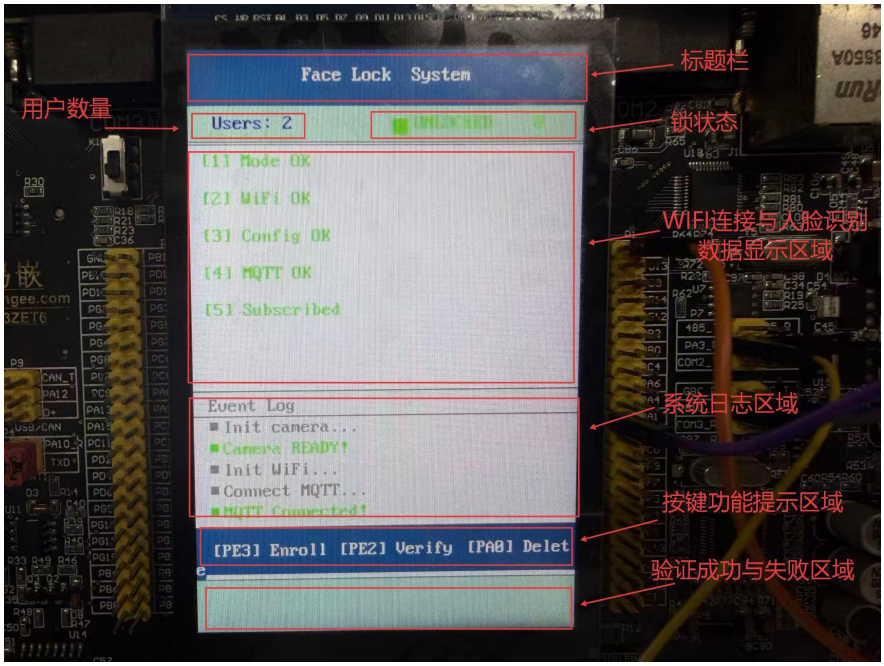

图 4-3 LCD 显示界面实验结果

（4）门锁自动控制功能

通过继电器驱动电磁锁，实现了门锁的自动控制。验证成功时，STM32 输出高电平到 PC3 引脚，三极管导通，继电器吸合，电磁锁得电开锁。验证失败时，立即关锁并触发告警。同时支持云端远程开锁和关锁功能。

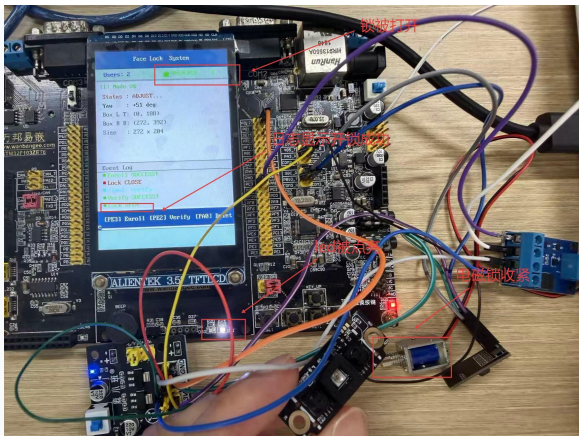

图 4-4 人门锁自动控制功能实验结果

（5）声光告警功能

通过有源蜂鸣器和 LED 指示灯，实现了完善的声光告警机制。陌生人验证失败时，蜂鸣器响 300ms 发出声音告警，LCD 显示红色"Stranger!"提示，LED 快闪 1 次。云端远程报警时，蜂鸣器连续响 3 次。告警信息显示 2 秒后自动清除，恢复正常界面。

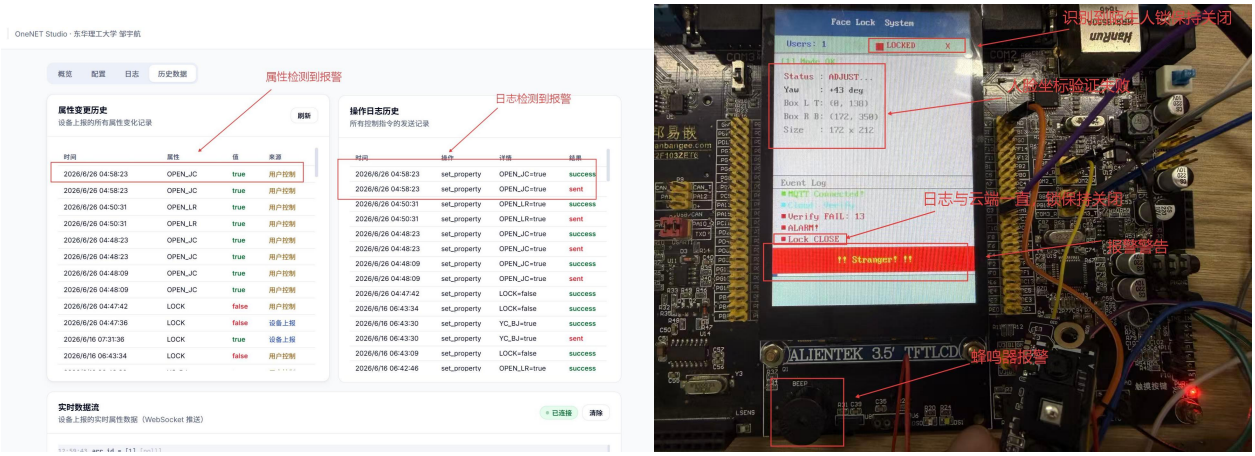

图 4-5 告警功能实验结果

4.2 设计过程

在本次课题设计过程中，我经历了从迷茫探索到逐步清晰、从框架搭建到细节填充的复杂历程。

项目初期（第 1-3 天）：学习 STM32 单片机基础知识，包括 GPIO、USART、FSMC 等外设的配置和使用方法。通过查阅数据手册和参考例程，逐步掌握了 STM32 的开发流程。同时学习 Keil5 开发环境的使用，熟悉工程创建、编译、下载等基本操作。

硬件调试阶段（第 4-7 天）：逐一测试各个外设模块，确保每个模块都能正常工作。首先调试 LCD 显示，实现基本的字符和图形显示；然后调试 FM225 摄像头，实现串口通信和命令收发；接着调试 ESP8266 WiFi 模块，实现 AT 指令配置和 MQTT 连接；最后整合所有模块，实现完整的系统功能。

软件设计阶段（第 8-12 天）：采用前后台（Superloop）架构，将系统分为应用层、业务模块层、硬件抽象层和硬件层四个层次。主循环采用 100ms 周期，依次处理人脸状态更新、命令结果处理、告警清除、云端指令检查、心跳上报、按键扫描等任务。编写各个功能模块的驱动程序，实现人脸识别、WiFi 通信、LCD 显示、门锁控制等功能。

联调测试阶段（第 13-15 天）：将所有模块整合到一起，进行系统联调测试。测试人脸识别注册和验证功能，确保 FM225 通信正常；测试 WiFi 连接和 MQTT 通信，确保云端数据收发正常；测试 LCD 界面显示，确保各个区域显示正确；测试按键控制和告警功能，确保系统响应及时。

在整个设计过程中，我不断遇到问题、分析问题、解决问题，逐步完善系统功能，积累了丰富的实践经验。

4.3 改进方法

对于课题设计的完善，我有以下几种改进方法：

（1）模块化封装

可以将每个功能模块都封装成独立的头文件和源文件，无论功能的大小。这能使文件的可读性和可修改性提高，便于后期维护和功能扩展。

（2）引入 RTOS 实时操作系统

可以引入 FreeRTOS 实时操作系统，将人脸识别、WiFi 通信、LCD 显示、按键扫描等功能模块作为独立任务运行。RTOS 可以提供更好的任务调度和资源管理，提高系统的实时性和可维护性。

（3）增加活体检测功能

可以在现有功能的基础上新增活体检测功能，防止照片攻击，提高系统的安全性。FM225 模块支持活体检测功能，可以通过配置参数启用。

（4）优化断线重连机制

可以优化 MQTT 通信的断线重连机制，当 WiFi 断开或 MQTT 连接断开时，系统能自动尝试重新连接，提高系统的稳定性和可靠性。

（5）增加用户权限管理

可以增加多个用户权限级别，实现管理员和普通用户的分级管理。管理员可以进行用户注册和删除操作，普通用户只能进行验证操作。

（6）增加数据存储功能

可以增加本地数据存储功能，将操作日志、用户信息等数据存储到 W25Q64 Flash 芯片中，便于后期查询和分析。

参考文献

[1] STMicroelectronics. STM32F103xx 参考手册 [Z]. 2018.

[2] 意法半导体. STM32F10x 中文参考手册 [M]. 2010.

[3] emtech. SX1276/77/78 Datasheet [Z]. 2015.

[4] 王宜怀. 计算机控制系统——基于 ARM Cortex-M [M]. 北京: 清华大学出版社, 2020.

[5]张毅刚. 单片机原理及接口技术 [M]. 北京: 人民邮电出版社, 2020.

[6] 刘军. 例说 STM32 [M]. 北京: 北京航空航天大学出版社, 2018.

[7] OmniVision. OV2640 Datasheet [Z]. 2006.

[8] 张弘. 数字图像处理与压缩技术 [M]. 北京: 电子工业出版社, 2019.

5.附录一——系统电路展示

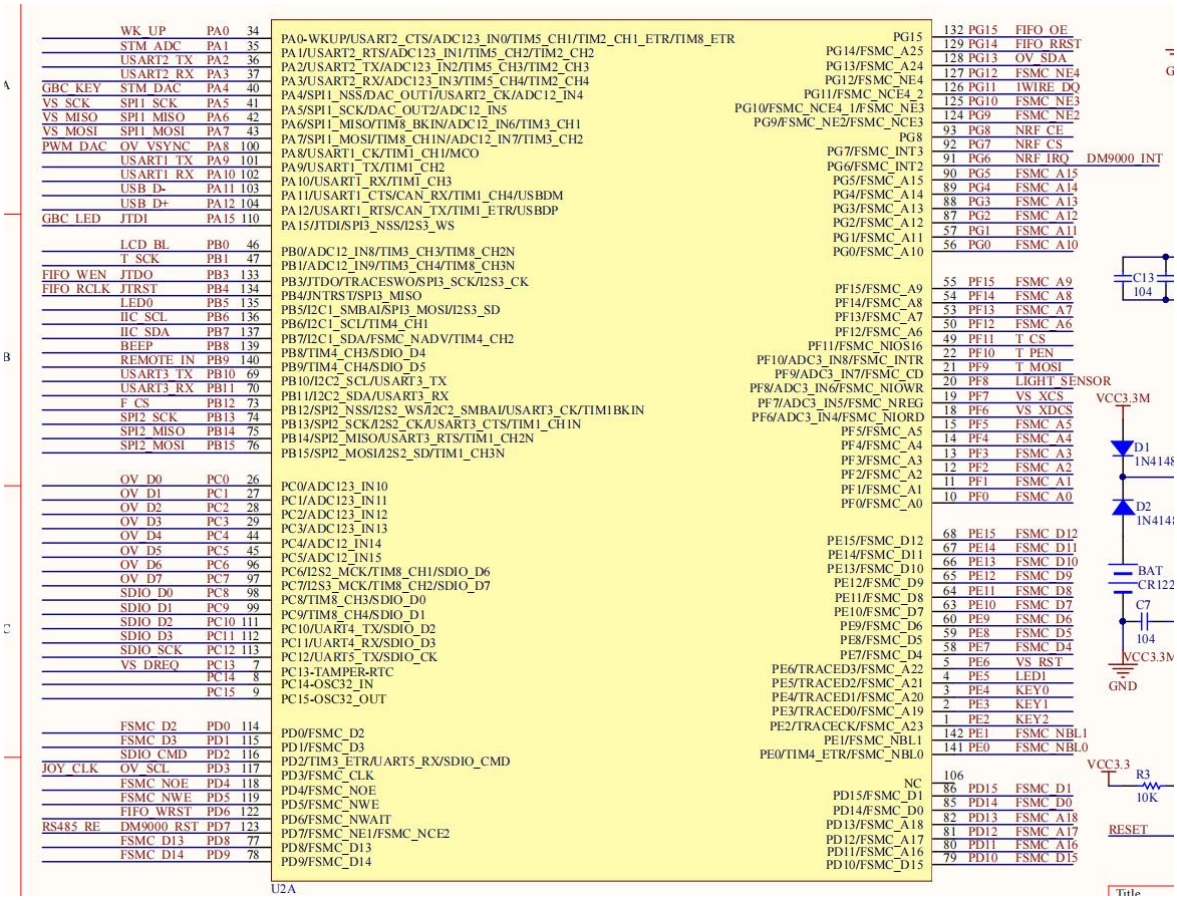

图 5-1 系统总体电路 1

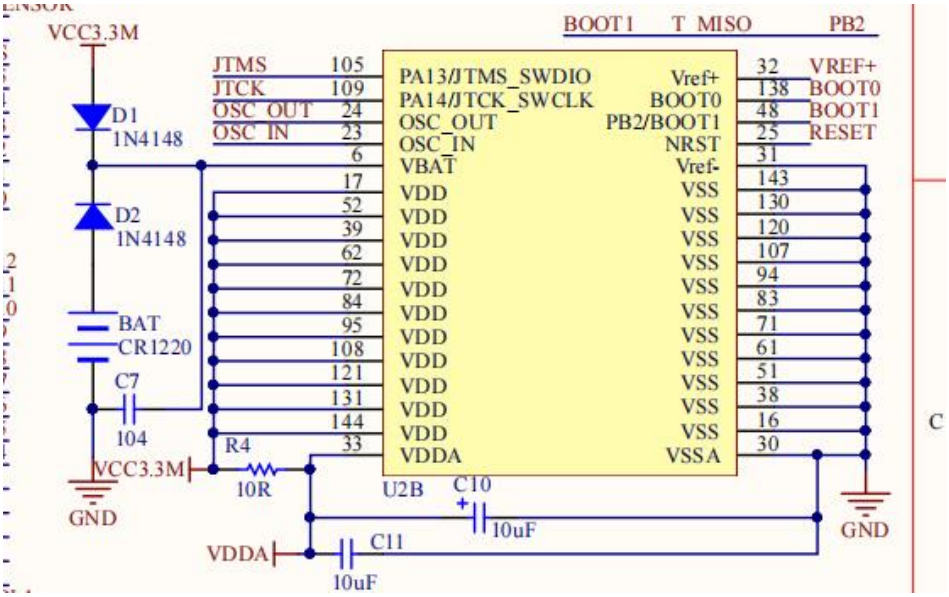

图 5-2 系统总体电路 2

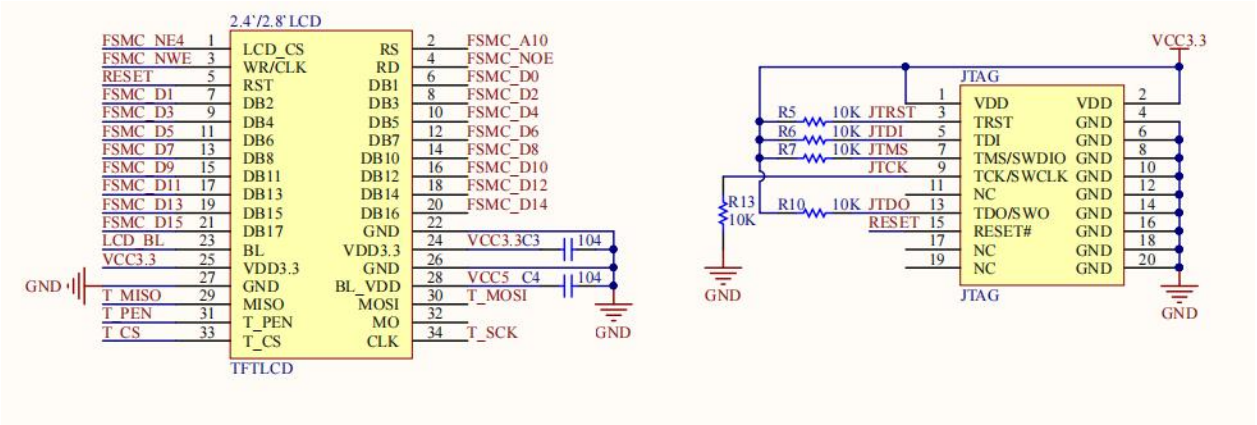

图 5-3 系统总体电路 3

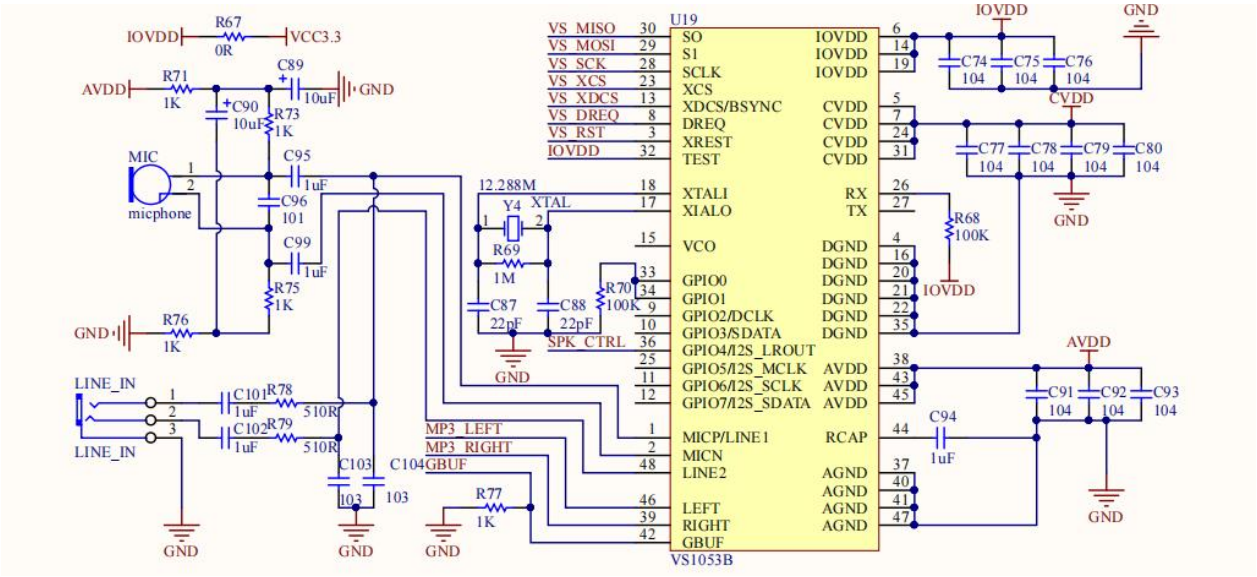

图 5-4 系统总体电路 4

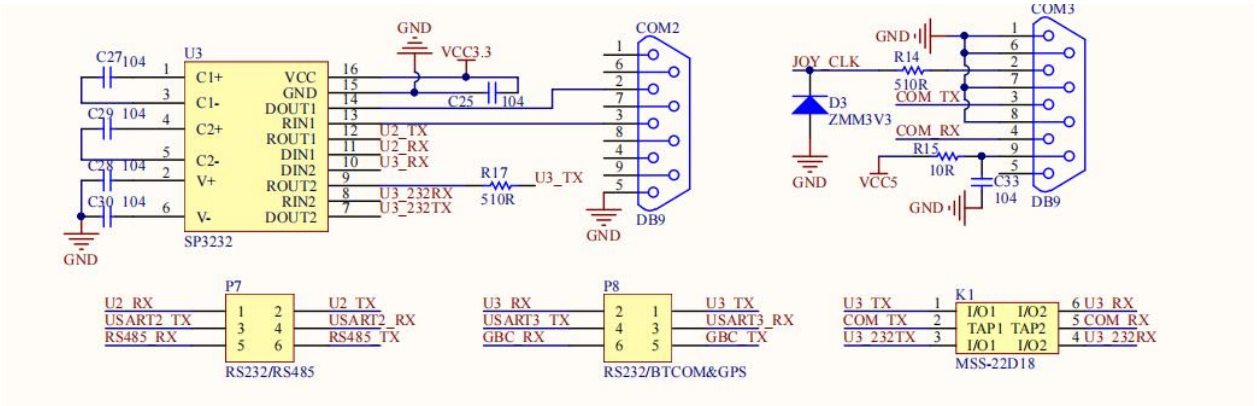

图 5-5 系统总体电路 5

6. 附录二——关键引脚接线

系统接线对照表

一、电源模块

电源输出	电压	连接对象	说明
3.3V	3.3V	STM32F103ZET6、FM225摄像头、ESP8266 WiFi、ST7796 LCD逻辑电源等	给数字电路供电
5V	5V	继电器模块、蜂鸣器、LED指示灯、LCD背光等	给外设供电
12V	12V	电磁锁	给电磁锁供电

二、主控芯片 STM32F103ZET6 接口连接

STM32引脚	功能/信号	连接对象	连接对象接口	说明
PA0	删除按键	删除按键	—	按键输入（下拉）
PE2	验证按键	验证按键	—	按键输入（下拉）
PE3	注册按键	注册按键	—	按键输入（下拉）
PA2 (TX)	UART1_TX	FM225摄像头	RX	USART1 发送
PA3 (RX)	UART1_RX	FM225摄像头	TX	USART1 接收
PB10 (TX)	UART3_TX	ESP8266 WiFi	RX	USART3 发送
PB11 (RX)	UART3_RX	ESP8266 WiFi	TX	USART3 接收
PC3	控制信号	继电器模块	IN	控制继电器吸合/断开（开锁）
PB8	控制信号	蜂鸣器	IN	蜂鸣器驱动信号
PE5	控制信号	LED指示灯	IN	LED状态指示
FSMC (20引脚)	并行总线	ST7796 LCD	DB[15:0], RS, WR,RD, CS, RST, BLK等	LCD数据/控制总线
PB0	控制信号	LCD背光	BLK	LCD背光控制（PWM/高电平）

三、外设接口说明

外设名称	供电电压	连接方式	主要作用
FM225摄像头（人脸识别）	3.3V	UART1 (PA2/PA3)	人脸采集与识别数据通信
ESP8266 WiFi（物联网）	3.3V	UART3 (PB10/PB11)	WiFi连接与MQTT通信
继电器模块	5V	PC3控制	驱动电磁锁，开/关锁
电磁锁	12V	继电器输出	执行开锁/上锁动作
蜂鸣器	5V	PB8控制	声音提示
LED指示灯	5V	PE5控制	状态指示
ST7796 LCD (320×480)	3.3V	FSMC并行总线 + 控制信号	显示界面
LCD背光	5V	PB0控制	LCD背光亮度控制

四、按键接口说明

按键名称	连接引脚	功能	说明
删除按键	PA0	删除全部用户	低电平有效
验证按键	PE2	人脸验证	低电平有效
注册按键	PE3	人脸注册	低电平有效

7.附录三——核心模块代码

主程序初始化代码

```
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* ========== MCU 基础配置 ========== */
  HAL_Init();                    // HAL 库初始化（SysTick、中断优先级等）
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  SystemClock_Config();          // 时钟配置：HSE 8MHz → PLL x9 → 72MHz
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */

  /* ========== 外设初始化 ========== */
  MX_GPIO_Init();                // GPIO 引脚初始化（按键、LED、锁、蜂鸣器）
  MX_USART2_UART_Init();         // USART2 初始化（FM225 摄像头通信，115200bps）
  MX_FSMC_Init();                // FSMC 初始化（LCD 并行总线，16-bit）

  /* USER CODE BEGIN 2 */

  /* ========== LCD 初始化 ========== */
  LCD_Init();                    // ST7796 LCD 屏幕初始化（初始化序列 + 清屏白色）
  HAL_GPIO_WritePin(LCD_BL_GPIO_Port, LCD_BL_Pin, GPIO_PIN_SET);  // 开背光
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_RESET);           // 初始上锁
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_5, GPIO_PIN_SET);             // LED 灭
  HAL_Delay(300);  // 等待 LCD 稳定

  /* ========== UI 界面初始化 ========== */
  UI_Init();  // 绘制标题栏、状态条、人脸区域、日志区、按键栏、统计栏

  /* ========== 按键初始状态读取 ========== */
  /* 读取当前按键电平，作为边沿检测的初始值 */
  pa0_last = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);  // PA0 删除按键
  pe3_last = HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_3);  // PE3 注册按键
  pe2_last = HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_2);  // PE2 验证按键

  /* ========== FM225 摄像头初始化 ========== */
  UI_Log("Init camera...", UI_GRAY);
  FM22x_Init();                                  // 初始化驱动状态变量
  HAL_UART_Receive_IT(&huart2, &fm22x_rx_byte, 1);  // 开启 USART2 中断接收
```

FM225摄像头代码

```
 */
static void ProcessReceivedFrame(uint8_t msgid, uint16_t size, uint8_t* data)
{
    switch(msgid)
    {
        /* -------- 命令应答消息 -------- */
        case MID_REPLY:
            if(size >= 2)
            {
                uint8_t cmd = data[0];       // 命令码（标识这是哪个命令的回复）
                uint8_t result = data[1];    // 结果码（0=成功）

                last_result = result;        // 保存结果码供主循环使用
                command_complete = true;      // 设置完成标志，通知主循环

                /* 如果是验证命令的回复，设置锁+蜂鸣器标志 */
                if(cmd == MID_VERIFY)
                    lock_beep_state = 1;

                /* 解析结果码的副作用（开锁/蜂鸣器） */
                ParseResult(result);

                /* -------- 处理用户 ID -------- */
                /* 注册或验证成功时，FM225 会在 data[2..3] 返回用户 ID（大端序） */
                if((cmd == MID_ENROLL || cmd == MID_ENROLL_SINGLE || cmd == MID_VERIFY)
                    && result == MR_SUCCESS && size >= 4)
                {
                    /* 解析用户 ID（大端序：高字节在前） */
                    g_last_uid = ((uint16_t)data[2] << 8) | data[3];

                    /* 注册成功时，将新用户 ID 添加到本地缓存数组 */
                    if (cmd == MID_ENROLL || cmd == MID_ENROLL_SINGLE) {
                        uint8_t found = 0;
                        /* 检查是否已存在（避免重复添加） */
                        for (uint8_t i = 0; i < g_user_id_count; i++) {
                            if (g_user_ids[i] == g_last_uid) { found = 1; break; }
                        }
                        /* 如果不存在且未满，添加到数组 */
                        if (!found && g_user_id_count < 50) {
                            g_user_ids[g_user_id_count++] = g_last_uid;
                        }
```

电磁锁代码

```
if (mqtt_connected) {
    char cmd_id[32] = {0};
    int8_t cloud_cmd = OneNet_CheckCommand(cmd_id);

    if (cloud_cmd == 0 && lock_state == 1) {
        /* 云端指令：关锁（LOCK=false） */
        do_lock();
        if (cmd_id[0]) OneNet_SendSetReply(cmd_id);  // 回复云端
    } else if (cloud_cmd == 1 && lock_state == 0) {
        /* 云端指令：开锁（LOCK=true） */
        do_unlock();
```

esp8266核心代码

```
uint8_t OneNet_Connect(void)
{
  /* 清空 WiFi 状态显示区域（LCD 上 y=80~260） */
  LCD_FillRect(10, 80, 310, 260, WHITE);

  /* Step 1: 设置 WiFi 模式为 Station（客户端模式） */
  LCD_ShowString(10, 80,  "[1] Set mode...", BLACK, WHITE);
  if (!ESP8266_SendCmd("AT+CWMODE=1", "OK", 3000))
  {
    LCD_ShowString(10, 80,  "[1] Mode FAIL        ", RED, WHITE);
    return 0;
  }
  LCD_ShowString(10, 80,  "[1] Mode OK          ", GREEN, WHITE);

  /* Step 2: 连接 WiFi 热点 */
  LCD_ShowString(10, 110, "[2] WiFi connecting..", BLACK, WHITE);
  if (!ESP8266_ConnectWiFi("onenet", "123456789"))
  {
    LCD_ShowString(10, 110, "[2] WiFi FAIL        ", RED, WHITE);
    return 0;
  }
  LCD_ShowString(10, 110, "[2] WiFi OK          ", GREEN, WHITE);
  HAL_Delay(2000);  // WiFi 连接后等待 2 秒稳定

  /* Step 3: 配置 MQTT 客户端参数（产品ID + Token 认证） */
  LCD_ShowString(10, 140, "[3] MQTT config...", BLACK, WHITE);
  if (!ESP8266_MQTTConfig(
      "onenet_one",                                  // 客户端 ID（设备名称）
      "u9gXgMhYR1",                                  // 用户名（产品 ID）
      "version=2018-10-31&res=products%2Fu9gXgMhYR1"     // 密码（Token 认证字符串）
      "%2Fdevices%2Fonenet_one&et=1812503220"
      "&method=md5&sign=ARYnb3UiX1TGHWqufrpm6A%3D%3D"))
  {
    LCD_ShowString(10, 140, "[3] Config FAIL      ", RED, WHITE);
    return 0;
  }
```

LCD屏幕核心代码

```
switch(msgid)
{
    /* -------- 命令应答消息 -------- */
    case MID_REPLY:
        if(size >= 2)
        {
            uint8_t cmd = data[0];       // 命令码（标识这是哪个命令的回复）
            uint8_t result = data[1];    // 结果码（0=成功）

            last_result = result;        // 保存结果码供主循环使用
            command_complete = true;      // 设置完成标志，通知主循环

            /* 如果是验证命令的回复，设置锁+蜂鸣器标志 */
            if(cmd == MID_VERIFY)
                lock_beep_state = 1;

            /* 解析结果码的副作用（开锁/蜂鸣器） */
            ParseResult(result);

            /* -------- 处理用户 ID -------- */
            /* 注册或验证成功时，FM225 会在 data[2..3] 返回用户 ID（大端序） */
            if((cmd == MID_ENROLL || cmd == MID_ENROLL_SINGLE || cmd == MID_VERIFY)
                && result == MR_SUCCESS && size >= 4)
            {
                /* 解析用户 ID（大端序：高字节在前） */
                g_last_uid = ((uint16_t)data[2] << 8) | data[3];

                /* 注册成功时，将新用户 ID 添加到本地缓存数组 */
                if (cmd == MID_ENROLL || cmd == MID_ENROLL_SINGLE) {
                    uint8_t found = 0;
                    /* 检查是否已存在（避免重复添加） */
                    for (uint8_t i = 0; i < g_user_id_count; i++) {
                        if (g_user_ids[i] == g_last_uid) { found = 1; break; }
                    }
                    /* 如果不存在且未满，添加到数组 */
                    if (!found && g_user_id_count < 50) {
                        g_user_ids[g_user_id_count++] = g_last_uid;
                    }
                }
            }

            /* -------- 处理 GetAllUser 响应 -------- */
            /* 命令码 0x24 = GetAllUser，成功时返回用户数量和 ID 列表 */
            if(cmd == 0x24 && result == MR_SUCCESS && size >= 3)
            {
                g_user_count = data[2];  // 用户总数

                /* 确保数据段足够容纳所有用户 ID（每个2字节） */
                if (size >= (uint16_t)(3 + g_user_count * 2)) {
                    g_user_id_count = 0;
```

串口核心代码

```
/* 使能 GPIOA 时钟 */
__HAL_RCC_GPIOA_CLK_ENABLE();

/**USART2 GPIO Configuration
PA2     ------> USART2_TX  (推挽复用输出)
PA3     ------> USART2_RX  (浮空输入)
*/
GPIO_InitStruct.Pin = GPIO_PIN_2;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;        // 复用推挽输出
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH; // 高速
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

GPIO_InitStruct.Pin = GPIO_PIN_3;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;        // 浮空输入
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* 配置 USART2 接收中断：优先级 0,0（最高） */
HAL_NVIC_SetPriority(USART2_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(USART2_IRQn);

/* USER CODE BEGIN USART2_MspInit 1 */

/* USER CODE END USART2_MspInit 1 */
}
```