重庆邮电大学

学生课程设计报告册

学年学期：2023-2024 学年 ■春□秋学期

课程名称：复杂物联网工程系统设计

学生学院：自动化学院/工业互联网学院

专业班级：08052103

学生学号：2021213198

学生姓名：简年杰

学生成绩：

指导教师：胡坤志

重庆邮电大学教务处

物联网节点设计　课程设计报告
成绩评定参考说明

A. 是否根据要解决的问题进行了理论分析。

B. 是否根据所要解决的问题建立数学模型。

C. 是否有具体的设计过程，是否对设计方案进行分析比较。

D. 是否有具体的仿真平台搭建过程或硬件制作及软件编程过程。

E. 是否有具体的仿真参数选定过程或实物参数调整过程。

F. 仿真结果或硬件效果是否丰富。

G. 是否有对仿真结果或硬件输出结果的具体分析。

H. 是否很好地解决了本次课程设计提出的问题。

I. 是否有对本次课程设计的心得体会以及对相关问题的展望。

J. 答辩情况。

K. 整个设计报告格式是否规范，目录要自动链接，公式编号及引用要自动链接。

L. 参考文献至少 8 篇(可以参考 CNKI、IEEE/IEL 和 ElsevierSD 数据库)。

M. 参考文献格式是否规范，是否在正文中引用。

N. 在指导教师出题及设计要求的大框架下，根据自己的具体设计情况（扩展、就某一方面的独特研究、就某一方面的创新），自己拟定设计题目、设计内容及要求的学生有加分奖励。

O. 课程设计最终成绩按等级制计算并提交，百分制与等级制的换算见附表。

附表：成绩与等级的换算关系如下：

成绩	≥90	80~89	70~79	60~69	＜60
等级	A（优）A 的人数比例≤15%	B+（良）	B-（中）	C（及格）	F（不及格）

一、 课程设计任务及要求

设计题目	基于 RFID 和人脸识别的智慧门禁系统设计

设计任务：

智能出入管理系统隶属于智能建筑领域，是新型的现代化安全管理系统，它的设计应用对于校园、小区和房屋的安全管理与信息采集起了很大的作用，是解决重要部门出入口实现安全防范管理的有效措施，并有利于提高社会治安治理水平。通过查询相关资料，设计智能出入管理系统，实现门禁控制，以及数据库存储相关信息与基于数据分析的警报提醒功能。

本次设计主要涵盖边缘数据采集、数据网络传输以及后台数据分析处理三大部分。其中，边缘层数据采集包括 RFID 身份认证、人脸识别、智慧门禁等内容；网络传输层借助 WiFi 网络等，实现边缘层采集数据的上报和后台分析数据发送等；平台层可基于远端服务器或者网络云平台等方式建立后台数据库，实时存储和分析边缘层的采集数据。功能模块方面，整个系统需具备边缘分析、后台分析以及报警推送能，对于简单数据的分析如按键控制和门禁开关等直接由边缘层进行处理和执行，对于人脸识别、身份认证、以及风险预警等操作由远端数据库进行分析与处理。

1、硬件要求

RFID 读写器、门禁闸机、门禁摄像、WIFI、手机、语音模块、传输网关，物联网开发板等。

2、软件要求

使用 MDK 集成开发环境或者 IAR 集成开发环境等开发软件编写数据采集以及相关控制程序。利用 android studio 或者 eclipse 软件开发相关手机 APP。开发后台运行数据库，通过数据分析对门禁信息进行评估，一旦发现风险及时报警。

3、功能与性能指标

1）具备基于人脸识别的门禁管理功能，可以有效监控出入人员身份信息，识别准确率不低于 85%；

2）具备 RFID 读取，以及密码按键、指纹识别、手机 APP 远程开关门禁等中的至少一种功能，可以有效监控出入人员身份信息；

3）具备闸机开关、语音播报等功能，能对人员进出的各种情况进行响应和提示；

4）开发门禁智能看板，显示日期、天气及小区通知公告等内容；

5）基于 WIFI 或者 4G、5G 等搭建数据传输网络，实现边缘节点间以及与数据管理中心之间的交互，数据丢包率不高于 1%，响应时间应控制在 3 秒以内；

6）基于云平台或者远端服务器搭建数据库，对人员进出权限进行集中管理，同时对人员出入方式进行分类统计；
课程对培养目标的支撑： 1、能够通过系统优化和单元集成，进行传感器与智能终端、网络传输、智能信息处理、工业物联网等部件或系统的设计，体现创新意识； 2、能够用图纸、报告或实物的形式，呈现设计成果； 3、能够采用计算机仿真、实物或半实物仿真等方法，设计实验方案，搭建实验系统，进行实验； 4、具有组建、协调和负责团队的能力，能够承担团队成员以及负责人的角色； 5、具备一定的外语运用能力，通过阅读国内外技术文献、理解技术行为之间的差异，能够进行有效技术交流。

指导教师（签字）	

二、 人员及分工

姓名	班级	学号	设计分工	分工系数	联系电话
简年杰	2021213198	08052103	小程序、云平台	0.4	18225460871
陈中印	2021213240	08052104	下位机开发	0.4	19922237052
王俊贤	2021213264	08052104	数据库	0.2	18381904036

目录

摘　要

在过去的生活中，我们经常会遇到因为门锁引发的安全问题，为了更好地应对未来可能出现的类似事件，我们小组设计了一款智能门禁系统，结合了 RFID 和人脸识别技术。这款系统不仅具备了人脸识别功能，还支持指纹解锁、刷卡解锁等多种解锁方式，为用户提供了更多便利和安全选择。通过微信小程序，用户可以实时查看门锁的状态，实现了安全性和便利性的双重提升。在这一设计中，我们采用了多种先进硬件设备，包括 AS608 指纹模块、K210 核心板、RFID-RC522 模块、OLED 模块、ESP8266-01S 模块和舵机模块等。主控芯片选用 STM32f103C8T6 最小系统板，通过控制 AS608 指纹模块和 RFID-RC522 模块实现指纹和 RFID 的识别功能，同时利用 K210 加载 YOLOv2 模型进行人脸识别，成功识别后与 STM32 进行通信。使用 ESP8266 模块将数据通过阿里云传输至微信小程序，同时在阿里云中进行数据库管理，实现了系统的完整功能和高效运作。通过这一设计方案，我们希望为未来的生活和生产领域提供更加安全和便利的解决方案，减少类似事件可能带来的不利影响。本文详细介绍了设计思路、器件选型、器件原理、软件设计以及测试分析，为智慧门禁领域的发展贡献了一份力量。

关键词：智慧门禁、阿里云、人脸识别、指纹识别、数据库

第一章 设计说明

1.1 背景

随着科学技术的不断更新发展，人民生活水平不断提高，安全意识也在不断提升。而锁作为第一道防线，自然有着其不可或缺的作用。锁具有重要的必要性，无论是在家庭、商业还是公共场所，它们都承担着保护财产、保障安全和维护隐私的重要责任。锁是保护财产和人员安全的第一道防线。它们阻止未经授权的人员进入私人空间或受限区域，从而降低盗窃、入室抢劫和其他犯罪行为的风险。锁提供了个人和机构的隐私保护。在家庭和办公场所，锁可以防止未经授权者进入个人房间或办公室，确保个人隐私和机密信息的安全。锁可以控制人员的出入，管理者可以根据需要授权特定人员进入特定区域，同时限制其他人员的访问权限。这在商业、政府和公共机构中尤其重要，有助于维护秩序和安全。锁能够给人们带来心理上的安全感和舒适感。知道家中或工作场所有锁保护，人们会感到更加安心和放心，从而提高生活和工作的质量。锁还可以预防意外事件的发生。例如，安装门锁可以防止儿童无意间进入危险区域，从而降低意外伤害的风险。

而科学进步的今天，智慧门禁系统承接了锁的义务与责任，也是我们小组的设计初心。本次设计主要涵盖边缘数据采集、数据网络传输以及后台数据分析处理三大部分。其中，边缘层数据采集包括 RFID 身份认证、人脸识别、智慧门禁等内容；网络传输层借助 WiFi 网络等，实现边缘层采集数据的上报和后台分析数据发送等；平台层可基于远端服务器或者网络云平台等方式建立后台数据库，实时存储和分析边缘层的采集数据。功能模块方面，整个系统需具备边缘分析、后台分析以及报警推送能，对于简单数据的分析如按键控制和门禁开关等直接由边缘层进行处理和执行，对于人脸识别、身份认证、以及风险预警等操作由远端数据库进行分析与处理。

1.2 国内外现状

智能门锁作为智能家居领域的重要组成部分，近年来在国内外市场都呈现出蓬勃发展的趋势。随着人们对家居安全和便利性要求的不断提高，智能门锁逐渐成为了消费者关注的焦点。在技术方面，智能门锁不断创新，采用了指纹识别、密码、手机 APP 远程开锁、人脸识别等先进技术，提升了门锁的安全性和便利性。据市场研究机构预测，智能门锁市场规模正在逐年扩大，未来几年仍将保持较快增长。在国外市场，智能门锁已相对成熟，一些产品在欧美、日本、韩国等地具有较高的知名度和市场份额；而在国内市场，随着互联网和物联网的发展，智能门锁也在快速崛起。国内企业纷纷推出各类智能门锁产品，通过线上线下渠道进行销售和推广，加速了智能门锁市场的发展。因此，智能门锁作为智能家居领域的重要组成部分，具有广阔的市场前景和应用潜力，值得进一步深入研究和探讨。

1.3 主要研究内容和意义

RFID 识别模块：将课程内容用于实践，深入了解 RFID 的原理，掌握 RC522 模块的使用方法，进

行 RFID 识别。

人脸图像识别模块：本模块是作为智能门禁系统中不可缺少的模块。通过将录入的人脸与 RFID 识别到的信息比对，可以追溯到是谁在某时某刻激活了门禁系统，从而使得信息可以追溯源头。同时需要制作程序对人脸识别的准确率和响应速度做优化，从而满足课程设计需求。

指纹识别模块：作为对人脸识别模块的补充，同样可以确定激活门禁系统的人选，时的信息可以追溯源头。同时需要制作程序对指纹识别的准确率和响应速度做优化，从而满足课程设计需求。

WI-FI 消息传输： 将门禁的人员出入信息和门状态数据进行传输，传输到云服务器。从而减少本地负担，以及每次重新开机后，由于有云端服务器的存在，数据不会丢失。

云平台和数据库：阿里云物联网平台可以对消息进行中转，实现门禁和微信小程序的通信。阿里云 RDS 数据库可以存储人员出入信息，方便人员管理。

第二章 总体设计

2.1 概述

我们小组设计的基于 RFID 和人脸识别的智慧门禁系统分为采集数据（RFID，人脸识别，指纹识别）、数据传输和服务器分析处理数据。通过 RFID 读写器，K210 核心板模块，AS608 指纹识别模块进行数据的采集。再通过 ESP8266-01S 实现对与远程云服务器的数据传输。服务端为阿里云服务器的？服务端提供了以下模块：用户和设备的访问控制模块，用户管理模块，RFID 门禁数据处理模块，人脸识别数据处理模块，指纹识别数据处理模块。从而实现对于智慧门禁系统的管理。

2.2 需求分析

2.2.1 功能需求

1）门禁系统搭载 RFID 识别和门禁控制的功能，在用户进行刷卡以后，通过 RFID 进行识别后，即可控制开启门禁。

2）门禁系统同时搭载了人脸图像识别技术和指纹数据识别技术。实现会将用户的人脸数据和指纹数据导入到数据库并进行学习，当用户进入摄像头范围内时，或者用户将指纹置于指纹检测器上时会识别被测用户的身份，将用户的人脸数据或指纹数据上传到云服务器，与云服务器内的数据进行对比识别，识别成功后，下发门禁系统的控制指令。

3）门禁系统会通过 WI-FI 网络进行数据传输，通过 WI-FI 信号实现边缘曾数据采集的上报，将测得的数据上传到云平台，云平台将数据存储到数据库，以供实时对边缘曾采集的数据进行存储和分析。

4）开发对应的小程序

（1） 打开小程序后，会申请访问地址，如果没有打开手机定位功能则会提示开启手机定位，开启定位后。直接是智能家居的主界面，顶部模块会自动调用天气接口更新当地天气情况，温湿度和风向，风力等级。

（2）导航栏右侧可以登陆用户，用户登陆这个功能调用的微信小程序的接口，请求登陆的时候弹出确认框，点击确认即可登陆用户。

（3）点击门锁，即可进入门锁界面，进入后自动连上阿里云物联网平台，中间圆形按钮显示门锁的状态。下方三个解锁方式分别是，密码解锁，人脸解锁，指纹解锁。点击密码解锁会唤起手机安全键盘，输入密码点击确认即可开门。点击指纹解锁即可唤起手机指纹识别，只需要识别手机已录入的指纹即可开门。人脸识别由于接口更新，所以未实现。

2.2.2 性能需求

1）基于校园 WI-FI 或者 4G 搭建数据传输网络中，数据丢包率不高于 1%，响应时间应控制在 1s

内

2）微信小程序刷新状态响应时间应控制在 1s 内

2.2.3 运行需求

1）门禁系统需要具备 RFID 信息读取、人脸识别、指纹识别、闸机开关等功能，可以有效监控出入人员的身份信息。

2）开发了智慧门禁系统的小程序，可以实时查看门禁系统的开启状态

3）基于 WI-FI 信号搭建数据传输网络，实现边缘采集数据、小程序监测信息和数据管理中心只见的数据交互。

4）基于云平台或远端服务器搭建数据库，对用户进出智慧门禁系统的数据进行管理。同时，数据管理平台具有统计和分析功能。

2.2.4 其他需求

1）在易用性和兼容性方面：该软件的设计应注重简洁明了的按键名称和使用方法，以便用户通过按键图标就能理解对应的功能。此外，界面切换也应该简单直观。保证软件能够在 iOS、安卓等多个操作系统上正常安装和运行，将有助于扩大用户群体并提供一致的用户体验。因此，在设计和开发过程中需要考虑不同操作系统的特点和要求，以确保软件在不同设备上都能够顺利运行。

2）低内存占用和低流量消耗:该软件主要用于查看最近一次的温度和健康打卡，这两项功能都不需要软件长时间运行，因此需要考虑后台的内存占用情况，在后台运行时尽量保持低内存占用，节省流量消耗。

3）安全性和稳定性：该软件采用阿里云物联网平台作为数据的转发和存储，这为数据传输提供了一定程度的保障。通过填写阿里云的三元组来调用接口，相当于使用密钥进行认证，有效地确保了数据传输的安全性。此外，阿里云物联网平台处理数据的响应时间非常迅速，而且具有良好的稳定性，这为软件的正常运行提供了坚实的基础。同时，阿里云作为业内领先的云计算服务提供商，拥有完善的安全防护机制和稳定的基础设施，可以有效应对各类网络安全威胁和突发状况，从而保障软件系统的安全性和稳定性。因此，通过整合阿里云物联网平台，该软件能够获得可靠的安全保障和稳定的数据存储与传输服务，为用户带来更加可靠和安全的使用体验。

2.3 系统结构

STM32 对采集到的信息进行处理，通过 ESP8266 上传到阿里云物联网平台，将数据流转写入 RDS 数据库，并下发到微信小程序，实现对用户的信息提示。详细结构如图 2.1。

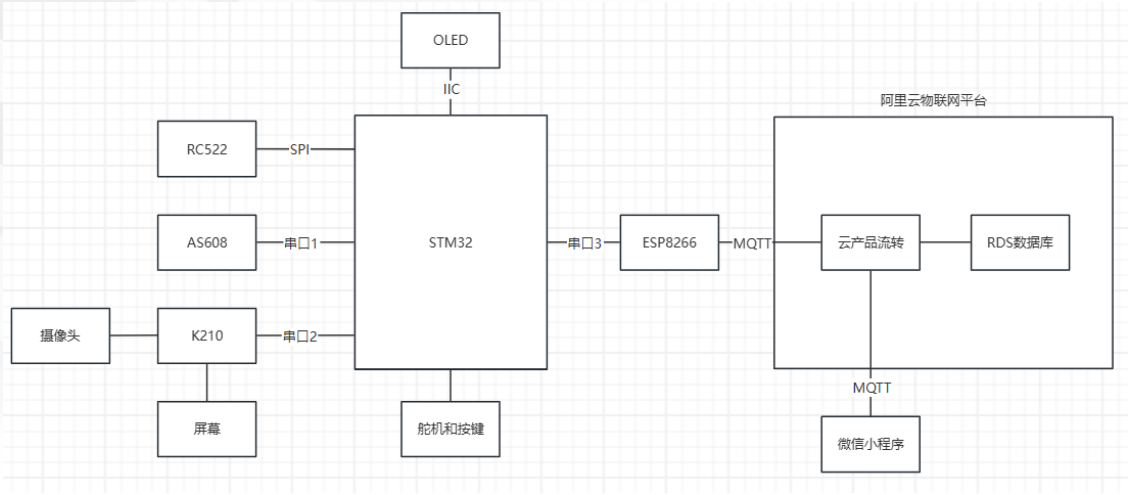

图 2.1 系统结构

2.4 功能框架

功能设计包括：

微信小程序显示位置、天气、门禁状态和开门方式，当有人开门时会提示开门的方式和更新门状态，门禁关闭时会提示门闭。

STM32 采集 RFID 刷卡记录、K210 人脸识别信息、AS608 指纹信息、门禁的状态信息，通过 ESP8266-01SWiFi 模块上传门禁状态和开门方式到云平台。

云平台能够对数据进行流转和分发，将 STM32 上传的信息发送给微信小程序进行状态的更新，并写入 RDS 数据库中。

功能框架如图 2.2。

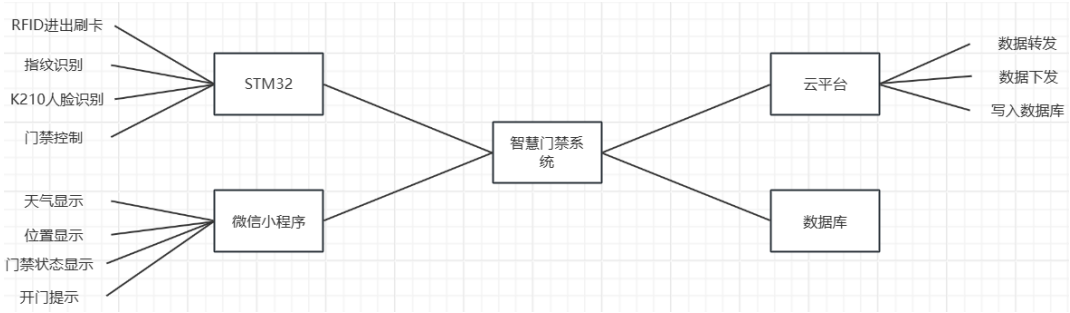

图 2.2 系统功能框架图

第三章 详细设计

3.1 概述

RFID 采用 RFID-RC522 模块，采用 SPI 通信协议，摄像头采用 K210 模块配套的摄像头，指纹识别采用 AS608 模块，采用串口通信协议，模块提供了很多接口，且自带存储可存储指纹库，便于用户调用开发。都可以采集数据上传云服务器，通过与数据库内录入的信息相互匹配，实现对智慧门禁系统的控制。

3.2 整体设计

我们使用 STM32 作为主控芯片，通过串口与 K210、AS608、ESP8266 通信，以此获取人脸、指纹数据和上传数据，通过 SPI 接口与 RC522 通信获取 RFID 门禁卡的信息，整体设计如图 3.1 所示。

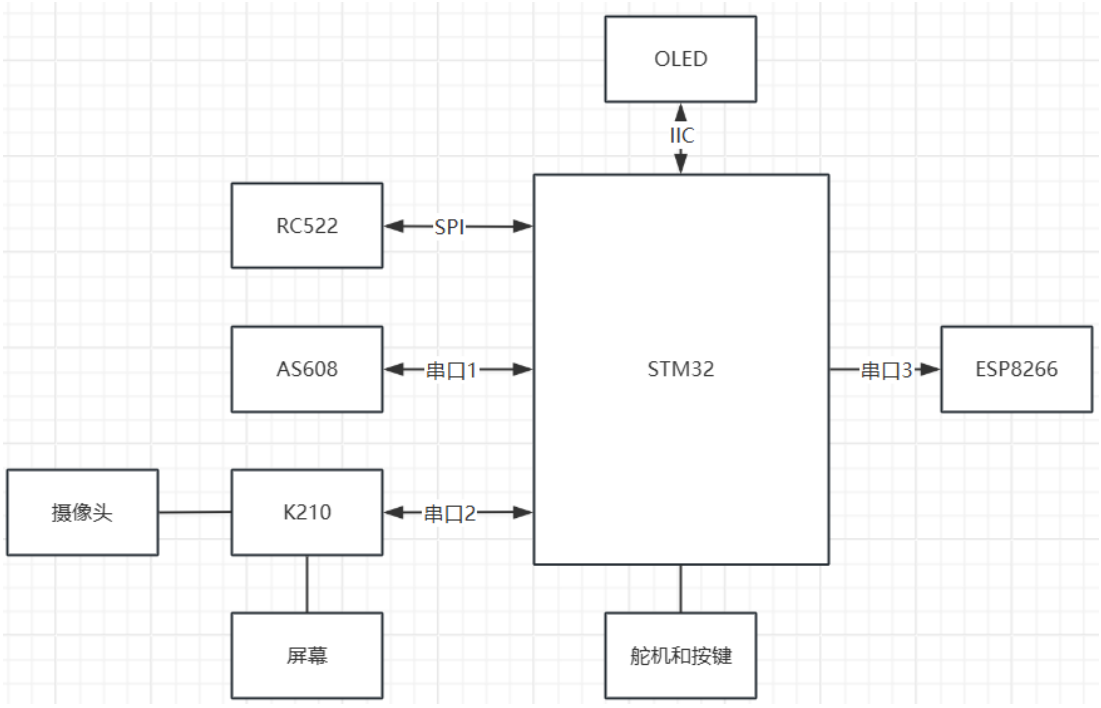

图 3.1 硬件整体设计

图中，STM32 作为主控芯片，通过 SPI 协议与 RC522 模块进行通信，在添加 id 卡时，识别到 id 卡信息后，将 id 卡信息存储到内存中，识别到已经添加的 id 卡后，打开成功；通过串口 1 与 AS608 通信，实现指纹识别，指纹信息存储在 AS608 内置 flash 中，实现掉电保存；通过串口 3 与 K210 通信，K210 读取 SD 卡内的人脸识别模型，使用摄像头采集人脸数据，对比与 SD 卡内存储的人脸特征点，特征符合，识别通过，识别通过后发送通过标志给 STM32，实现了人脸识别；RFID、指纹、人脸方式识别成功后通过 pwm 控制舵机转动，打开门禁，同时通过串口 3 与 ESP8266 通信，将数据发送到阿里云平台。

3.3 门禁模块

本模块为智能门锁系统的硬件模块，UML 用例图如图 3.2。

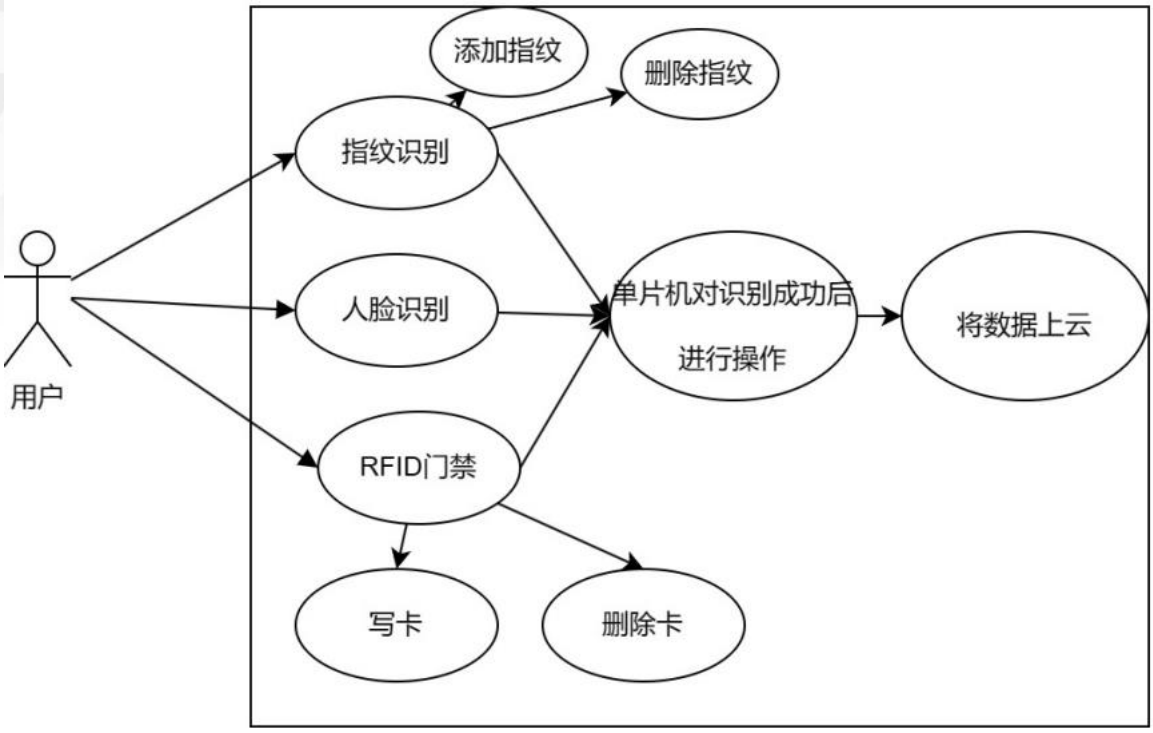

图 3.2 硬件模块 UML 用例图

3.3.1 RC522 模块基本原理

本次设计中利用 RC522 模块作为刷卡解锁功能的控件。RC522 是一种非接触式读写卡芯片，底层采用 SPI 模拟时序，可以应用于校园一卡通、水卡充值消费、公交卡充值消费设计、门禁卡等。

有两个部分，射频读写器和 IC 卡。射频读写器向 IC 卡发一组固定频率的电磁波，卡片内有一个 LC 串联谐振电路，其频率与读写器发射的频率相同，这样在电磁波激励下，LC 谐振电路产生共振，从而使电容内有了电荷；在这个电荷的另一端，接有一个单向导通的电子泵，将电容内的电荷送到另一个电容内存储，当所积累的电荷达到 2V 时，此电容可作为电源为其它电路提供工作电压，将卡内数据发射出去或接受读写器的数据。

非接触性 IC 卡与读卡器之间通过无线电波来完成读写操作。二者之间的通讯频率为 13.56MHZ。非接触性 IC 卡本身是无源卡，当读写器对卡进行读写操作时，读写器发出的信号由两部分叠加组成: 一部分是电源信号，该信号由卡接收后，与本身的 L/C 产生一个瞬间能量来供给芯片工作。另一部分则是指令和数据信号，指挥芯片完成数据的读取、修改、储存等，并返回信号给读写器,完成一次读写操作。具体的 RFID 模块识别流程如图 3.3 所示。

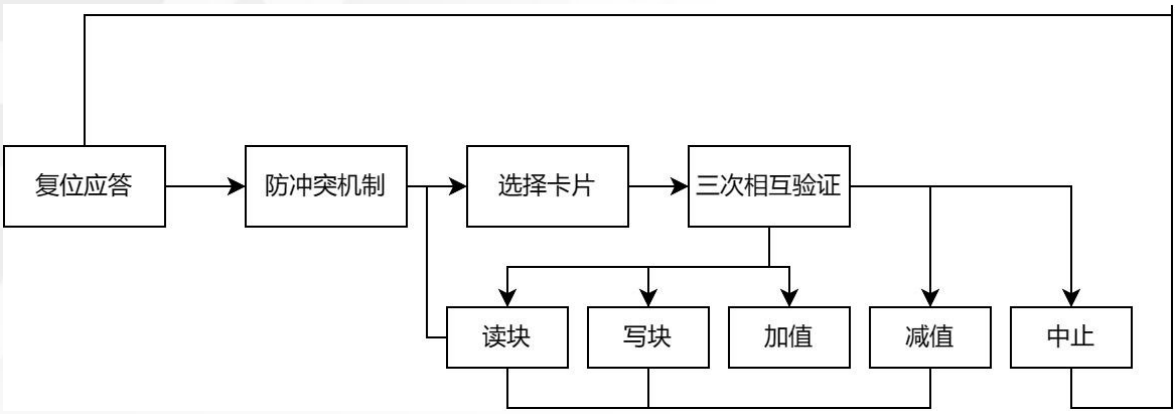

图 3.3 RFID 模块识别流程

复位应答：M1 射频卡的通讯协议和通讯波特率是定义好的，当有卡片进入读写器的操作范围时，读写器以特定的协议与它通讯，从而确定该卡是否为 M1 射频卡，即验证卡片的卡型。

防冲突机制：当有多张卡进入读写器操作范围时，防冲突机制会从其中选择一张进行操作，未选中的则处于空闲模式等待下一次选卡，该过程会返回被选卡的序列号。

选择卡片：选择被选中的卡的序列号，并同时返回卡的容量代码。

三次互相确认：选定要处理的卡片之后，读写器就确定要访问的扇区号，并对该扇区密码进行密码校验，在三次相互认证之后就可以通过加密流进行通讯。（在选择另一扇区时，则必须进行另一扇区密码校验）

如果概括来说的话，主要也就四部分：开关连接、寻卡、验证密码、读取。

3.3.2 源码解析

1）数据块数据读取和写入

利用上述通信流程，利用代码实现寻卡、放冲突和选卡。利用指定扇区的区尾存储 密码对卡片进行认证，再通过发送读写命令，对指定的数据块进行读取或写入数据的操作，最后停止数据的加密过程。卡数据块读取和写入的主要代码如图 3.4 和图 3.5。

```
char PcdRead(unsigned char addr, unsigned char *pData)
{
    char status;
    unsigned int unLen;
    unsigned char i, ucComMF522Buf[MAXRLEN];

    ucComMF522Buf[0] = PICC_READ;
    ucComMF522Buf[1] = addr;
    CalulateCRC(ucComMF522Buf, 2, &ucComMF522Buf[2]);

    status = PcdComMF522(PCD_TRANSCEIVE, ucComMF522Buf, 4, ucComMF522Buf, &unLen);
    if ((status == MI_OK) && (unLen == 0x90))
    //   {   memcpy(pData, ucComMF522Buf, 16);   }
    {
        for (i = 0; i < 16; i++)
        {
            *(pData + i) = ucComMF522Buf[i];
        }
    }
    else
    {
        status = MI_ERR;
    }

    return status;
}
```

图 3.4 读取 M1 卡数据块代码

```
char PcdWrite(unsigned char addr, unsigned char *pData)
{
    char status;
    unsigned int unLen;
    unsigned char i, ucComMF522Buf[MAXRLEN];

    ucComMF522Buf[0] = PICC_WRITE;
    ucComMF522Buf[1] = addr;
    CalulateCRC(ucComMF522Buf, 2, &ucComMF522Buf[2]);

    status = PcdComMF522(PCD_TRANSCEIVE, ucComMF522Buf, 4, ucComMF522Buf, &unLen);

    if ((status != MI_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))
    {
        status = MI_ERR;
    }

    if (status == MI_OK)
    {
        // memcpy(ucComMF522Buf, pData, 16);

        for (i = 0; i < 16; i++)
        {
            ucComMF522Buf[i] = *(pData + i);
        }
        CalulateCRC(ucComMF522Buf, 16, &ucComMF522Buf[16]);

        status = PcdComMF522(PCD_TRANSCEIVE, ucComMF522Buf, 18, ucComMF522Buf, &unLen);
        if ((status != MI_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))
        {
            status = MI_ERR;
        }
    }

    return status;
}
```

图 3.5 MIFARE1 卡数据写入函数代码

写入数据后即可进行后续的卡片和读卡器的通信，当进行读卡时，首先会进行寻卡操作，清空 RC522 寄存器位，写入当前卡片信息，再将其置顶寄存器位，进行防冲撞的检测后，返回状态信息，如果卡片是已录入的卡片则会返回 OK，如果不是则会 ERROR。代码如图 3.6。

```
char PcdRequest(unsigned char req_code, unsigned char *pTagType)
{
    char status;
    unsigned int unLen;
    unsigned char ucComMF522Buf[MAXRLEN];
    //  unsigned char xTest ;
    ClearBitMask(Status2Reg, 0x08);
    WriteRawRC(BitFramingReg, 0x07);

    //  xTest = ReadRawRC(BitFramingReg);
    //  if(xTest == 0x07 )
    //    { LED_GREEN  =0 ;}
    // else {LED_GREEN =1 ;while(1){}}
    SetBitMask(TxControlReg, 0x03);

    ucComMF522Buf[0] = req_code;

    status = PcdComMF522(PCD_TRANSCEIVE, ucComMF522Buf, 1, ucComMF522Buf, &unLen);
    //      if(status  == MI_OK )
    //    { LED_GREEN  =0 ;}
    //    else {LED_GREEN =1 ;}
    if ((status == MI_OK) && (unLen == 0x10))
    {
        *pTagType = ucComMF522Buf[0];
        *(pTagType + 1) = ucComMF522Buf[1];
    }
    else
    {
        status = MI_ERR;
    }

    return status;
}
```

图 3.6 RC522 寻卡函数

3.4 AS608 模块

3.4.1 AS608 模块基本原理

系统内设有一个 72K 字节的图像缓冲区与二个 512bytes 大小的特征文件缓冲区，名字分别称为：ImageBuffer，CharBuffer1 和 CharBuffer2。用户可以通过指令读写任意一个缓冲区。

CharBuffer1 或 CharBuffer2 既可以用于存放普通特征文件也可以用于存放模板特征文件。通过 UART 口上传或下载图像时为了加快速度，只用到像素字节的高 4 位，即将两个像素合成一个字节传送。通过 USB 口则是整 8 位像素。

指纹库容量根据挂接的 FLASH 容量不同而改变，系统会自动判别。指纹模板按照序号存放，序号定义为：0—（N-1）（N 为指纹库容量）。用户只能根据序号访问指纹库内容。

用户记事本：系统在 FLASH 中开辟了一个 512 字节的存储区域作为用户记事本，该记事本逻辑上被分成 16 页，每页 32 字节。上位机可以通过 PS_WriteNotepad 指令和 PS_ReadNotepad 指令访问任意一页。注意写记事本某一页的时候，该页 32 字节的内容被整体写入，原来的内容被覆盖。

随机数产生器：系统内部集成了硬件 32 位随机数生成器（不需要随机数种子），用户可以通过指令让模块产生一个随机数并上传给上位机。

模块地址 （大小：4bytes ，属性：读写） ）

模块的默认地址为 0xFFFFFFFF，可通过指令修改，数据包的地址域必须与该地址相配，命令包/数据包才被系统接收。 注：与上位机通讯必须是默认地址 0xFFFFFFFF ！

模块口令 （大小：4bytes ，属性：写）系统默认口令为 0，可通过指令修改。若默认口令未被修改，则系统不要求验证口令，上位机和 MCU 与芯片通讯；若口令被修改，则上位机与芯片通讯的第一个指令必须是验证口令，只有口令验证通过后，芯片才接收其它指令。 数据包大小设置（大小：1bytes ，属性：读写）发送数据包和接收数据包的长度根据该值设定。波特率数 系数 N 设置 （大小：1bytes ，属性：读写）USART 波特率=N×9600，N=1~12。安全等级 level 设置（大小：1bytes ，属性：读写）

系统根据安全等级设定比对阀值，level=1~5。安全等级为 1 时认假率最高，拒认率最低。安全等级为 5 时认假率最低，拒认率最高。

3.4.2 源码解析

大致流程是这样的：

1） 用 GZ_HandShake 函数与 AS608 握手，这个握手类似于单片机向 AS608 发送一些信息，如果 AS608 能正常工作的话它会返回一些对应东西来回答你。这个函数用于确认 AS608 能正常运行，开始运行时先执行这个函数。代码如图 3.7。

```c
uint8_t GZ_HandShake(uint32_t *GZ_Addr)
{
    SendHead();
    SendAddr();
    Com_SendData(0X01);
    Com_SendData(0X00);
    Com_SendData(0X00);
    HAL_Delay(200);
    if (RX_len)
    {
        RX_len = 0;
        if ( // 判断是不是模块返回的应答包
            aRxBuffer[0] == 0XEF && aRxBuffer[1] == 0X01 && aRxBuffer[6] == 0X07)
        {
            *GZ_Addr = (aRxBuffer[2] << 24) + (aRxBuffer[3] << 16) + (aRxBuffer[4] << 8) + (aRxBuffer[5]);
            return 0;
        }
    }
    return 1;
}
```

图 3.7 GZ_HandShake 函数代码

2）用 GZ_GetImage 来获取按在模块上手指的指纹图像，然后执行 GZ_GenChar 来获取这个图像里面的特征，将这个特征存在 CharBuffer1 或 CharBuffer2，是 AS08 中存储的两个区域。这边的两个函数要执行两次，也就是说录入一个指纹的时候需要按两次，然后 AS608 执行 GZ_Match( )去对比刚刚采集到的两个特征，如果符合条件，则执行生成模板 GZ_RegModel( )函数，然后再存储 GZ_StoreChar( )。 代码如图 3.8 和图 3.9。

```c
uint8_t GZ_GetImage(void)
{
    uint16_t temp;
    uint8_t ensure;
    uint8_t *data;
    SendHead();
    SendAddr();
    SendFlag(0x01); // 命令包标识
    SendLength(0x03);
    Sendcmd(0x01);
    temp = 0x01 + 0x03 + 0x01;
    SendCheck(temp);
    data = JudgeStr(2000);
    if (data)
        ensure = data[9];
    else
        ensure = 0xff;
    return ensure;
}
```

图 3.8 录入指纹函数代码

```
uint8_t GZ_GenChar(uint8_t BufferID)
{
    uint16_t temp;
    uint8_t ensure;
    uint8_t *data;
    SendHead();
    SendAddr();
    SendFlag(0x01); // 命令包标识
    SendLength(0x04);
    Sendcmd(0x02);
    Com_SendData(BufferID);
    temp = 0x01 + 0x04 + 0x02 + BufferID;
    SendCheck(temp);
    data = JudgeStr(2000);
    if (data)
        ensure = data[9];
    else
        ensure = 0xff;
    return ensure;
}
```

图 3.9 生成特征值函数代码

2） GZ_DeletChar(u16 PageID,u16 N)这个是用来删除指定位置的指纹，GZ_Empty(void);这个函数是清空指纹库。

```
uint8_t GZ_DeletChar(uint16_t PageID, uint16_t N)
{
    uint16_t temp;
    uint8_t ensure;
    uint8_t *data;
    SendHead();
    SendAddr();
    SendFlag(0x01); // 命令包标识
    SendLength(0x07);
    Sendcmd(0x0C);
    Com_SendData(PageID >> 8);
    Com_SendData(PageID);
    Com_SendData(N >> 8);
    Com_SendData(N);
    temp = 0x01 + 0x07 + 0x0C + (PageID >> 8) + (uint8_t)PageID + (N >> 8) + (uint8_t)N;
    SendCheck(temp);
    data = JudgeStr(2000);
    if (data)
        ensure = data[9];
    else
        ensure = 0xff;
    return ensure;
}
```

图 3.10 GZ_DeletChar 函数代码

4）读取内部指纹个数：GZ_ValidTempleteNum(u16 *ValidN)，可以令一个数等于 ValidN,再打印出来即可看见。

5）GZ_Search(uint8_t BufferID,uint16_t StartPage,uint16_t PageNum,SearchResult *p);搜索函数，也就是搜索现在刷的指纹在指纹库库中有没有对应的指纹。

```
uint8_t GZ_Search(uint8_t BufferID, uint16_t StartPage, uint16_t PageNum, SearchResult *p)
{
    uint16_t temp;
    uint8_t ensure;
    uint8_t *data;
    SendHead();
    SendAddr();
    SendFlag(0x01); // 命令包标识
    SendLength(0x08);
    Sendcmd(0x04);
    Com_SendData(BufferID);
    Com_SendData(StartPage >> 8);
    Com_SendData(StartPage);
    Com_SendData(PageNum >> 8);
    Com_SendData(PageNum);
    temp = 0x01 + 0x08 + 0x04 + BufferID + (StartPage >> 8) + (uint8_t)StartPage + (PageNum >> 8) + (uint8_t)PageNum;
    SendCheck(temp);
    data = JudgeStr(2000);
    if (data)
    {
        ensure = data[9];
        p->pageID = (data[10] << 8) + data[11];
        p->mathscore = (data[12] << 8) + data[13];
    }
    else
        ensure = 0xff;
```

图 3.11 GZ_Search 函数部分代码

3.5 联网模块

3.5.1 ESP8266 模块原理

ESP8266 支持 STA、AP、AP+STA 三种工作模式。

1）STA 模式（Station）

一般用于远距离传输。ESP8266 通过路由器连接互联网，终端设备通过互联网实现对设备的远程控制。简单来说，此时的 ESP8266 可以当作是一个客户端，可以向服务端进行数据的下载与传输。这就类似于，手机/平板/笔记本（客户端）可以通过 WIFI 连接到路由器进行上网。

2）AP 模式（Access Point）

一般用于近距离传输。ESP8266 作为热点，提供无线接入服务、数据访问，一般的无线路由/网桥工作在 AP 模式下，最多支持 4 台设备接入。简单来说，此时的 ESP8266 可以当作是一个服务端。这就类似于，ESP8266 变身为一个路由器，然后手机/平板/笔记本可以通过 WIFI 连接到 ESP8266 进行上网。

3）AP+STA 模式

两种模式的共存模式，可以通过互联网控制可实现无缝切换，方便操作。简单来说，此时的 ESP8266 可以当作是一个路由器既可以做服务端接收也可以当客户端连接路由器，进行联网传输和控制。

而 ESP8266 又是通过 AT 指令控制通信的，AT 指令（AT Commands）最早是由发明拨号调制解调器的贺氏公司为了控制拨号调制解调器而发明的控制协议。后来随着网络带宽的升级，速度很低的拨号调制解调器基本退出市场，但是 AT 指令被保留了下来。在嵌入式开发中，经常是使用 AT 命令去控制各种通讯模块，比如 ESP8266 模块、4G 模块、GPRS 模块等等。一般就是主芯片通过硬件接口（比

如串口、SPI）发送 AT 指令给通讯模块，模块接收到数据之后回应响应的数据。

如图 4.16，AT 指令分为四种类型：

类型	格式	功能
测试指令	AT + < X > = ?	查询设置命令或内部程序设置的参数及其取值范围
查询指令	AT + < X > ?	返回参数的当前值
设置指令	AT + < X > = < ... >	设置用户自定义的参数值
执行指令	AT + < X >	执行受模块内部程序控制的变参数不可变

图 3.12 AT 指令类型示意图

3.5.2 源码解析

首先初始化 ESP8266 后需要设置它的模式，然后连接 WIFI 和阿里云，代码如图 3.13。

```
void wifi_init(void){
    uint8_t ret = 0;
    ret  = ESP8266_Reset();
    HAL_Delay(200);
////    printf("1: %d\r\n",ret);
    ret = ESP8266_ATE(0);
    HAL_Delay(200);
////    printf("2: %d\r\n",ret);
//  ret = ESP8066_Mode(1);
//  printf("3: %d\r\n",ret);
    ret = ESP8266_WiFi(WIFI_User, WIFI_Pass);
    HAL_Delay(2000);
//  printf("4: %d\r\n",ret);
    ret = ESP8266_MQTTUSERCFG(ESP8266_UserName, ESP8266_PassWord);
    HAL_Delay(2000);
//  printf("5: %d\r\n",ret);
    ret = ESP8266_MQTTCLIENTID(ESP8266_ClientID);
    HAL_Delay(2000);
//  printf("6: %d\r\n",ret);
    ret =  ESP8266_MQTTCONN(ESP8266_Domain_Name,ESP8266_Port,ESP8266_Reconnect);
    HAL_Delay(5000);
//  printf("7: %d\r\n",ret);
    ret =  ESP8266_MQTTSUB(ESP8266_Post_re);
//  printf("8: %d\r\n",ret);
```

图 3.13 ESP8266 初始化函数

ESP8266 处理和发送数据的流程如下：使用 cJSON 库创建一个空的 JSON 对象和一个空的子 JSON 对象。将门状态等数据添加到子 JSON 对象中。使用 cJSON_PrintUnformatted() 将 JSON 对象转换为字符串，不包含格式化字符。在转换的过程中，对特殊字符进行处理，如添加反斜杠。使用 sprintf() 将 MQTT 的发布命令字符串格式化，其中包含 JSON 数据字符串。ESP8266_Post 是一个宏定义，用于指定 MQTT 主题。调用 ESP8266_Sent_AT() 函数发送命令给 ESP8266 模块，并等待返回 "OK" 响应。删除 JSON 对象以释放内存。如果 JSON 字符串内存被分配，释放该内存。流程和代码如图 3.14 和 3.15。

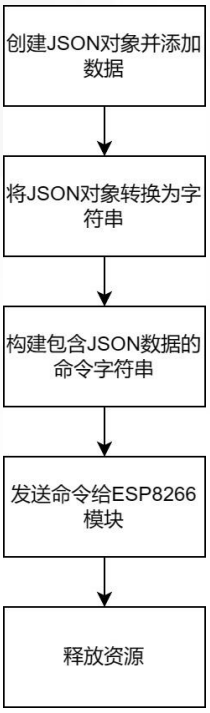

图 3.14 处理待发送的数据流程

```
void create_Json(uint8_t temperature, uint8_t Door_state){
    uint8_t cmd[1024]; // 用于存储构建的AT命令
    char *str = NULL; // 指向JSON字符串的指针
    int i = 0; // 循环迭代变量
    uint8_t params_buf[1024]; // 用于存储处理过的JSON字符串
    uint16_t move_num = 0; // 用于记录字符串处理过程中的移动次数

    cJSON *json = cJSON_CreateObject(); // 创建一个空的JSON对象
    cJSON *params_cjson = cJSON_CreateObject(); // 创建一个空的子JSON对象

    // 向子JSON对象添加数据
    cJSON_AddNumberToObject(params_cjson, "door", Door_state);
    cJSON_AddItemToObject(json, "params", params_cjson);
    // 将JSON对象转换为无格式的字符串
    str = cJSON_PrintUnformatted(json);
    // 为MQTT发布添加转义字符
    for(i = 0; *str != '\0'; i++){
        params_buf[i] = *str;
        // 如果下一个字符是引号或逗号，添加转义字符
        if(*(str + 1) == '"' || *(str + 1) == ','){
            params_buf[++i] = '\\';
        }
        str++;
        move_num++;
    }
    str = str - move_num; // 回退指针到JSON字符串的开始

    // 构建AT命令
    sprintf((char *)cmd,"AT+MQTTPUB=0,\""ESP8266_Post"\",\"%s\",0,0\r\n",params_buf);
```

图 3.15 Creat_Json 函数部分代码示例

3.6 人脸识别模块

本次设计中人脸识别模块的识别以及识别结果采用 MicroPython 实现，是 Python 3 语言的精简高效实现 ，包括 Python 标准库的一小部分，并针对嵌入式微控制器（单片机）和受限制的环境进行了优化，它是 Python 延伸出来的一个落地产物。MicroPython 是运行在微控制器硬件之上的完全的 Python 编译器和运行时系统，它提供给用户一个交互式提示符（REPL）来立即执行所支持的命令。除了包括选定的核心 Python 库，MicroPython 还包括了给予编程者访问低层硬件的模块。

使用官方库人脸检测算法，算法的流程如图 3.16。

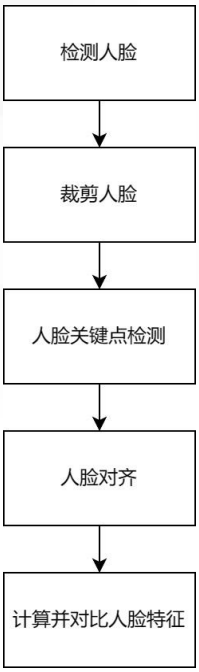

图 3.16 官方库人脸检测流程

```
if code: # 如果检测到人脸
    for i in code: # 迭代坐标框
        # Cut face and resize to 128x128
        a = img.draw_rectangle(i.rect()) # 在屏幕显示人脸方框
        face_cut=img.cut(i.x(),i.y(),i.w(),i.h()) # 裁剪人脸部分图片到 face_cut
        face_cut_128=face_cut.resize(128,128) # 将裁出的人脸图片 缩放到128 * 128像素
        a=face_cut_128.pix_to_ai() # 将裁出图片转换为kpu接受的格式
        #a = img.draw_image(face_cut_128, (0,0))
        # Landmark for face 5 points
        fmap = kpu.forward(task_ld, face_cut_128) # 运行人脸5点关键点检测模型
        plist=fmap[:] # 获取关键点预测结果
        le=(i.x()+int(plist[0]*i.w() - 10), i.y()+int(plist[1]*i.h())) # 计算左眼位置， 这里在w方向-10 用来补偿模型转换带
        re=(i.x()+int(plist[2]*i.w()), i.y()+int(plist[3]*i.h())) # 计算右眼位置
        nose=(i.x()+int(plist[4]*i.w()), i.y()+int(plist[5]*i.h())) #计算鼻子位置
        lm=(i.x()+int(plist[6]*i.w()), i.y()+int(plist[7]*i.h())) #计算左嘴角位置
        rm=(i.x()+int(plist[8]*i.w()), i.y()+int(plist[9]*i.h())) #右嘴角位置
        a = img.draw_circle(le[0], le[1], 4)
        a = img.draw_circle(re[0], re[1], 4)
        a = img.draw_circle(nose[0], nose[1], 4)
        a = img.draw_circle(lm[0], lm[1], 4)
        a = img.draw_circle(rm[0], rm[1], 4) # 在相应位置处画小圆圈
        # align face to standard position
        src_point = [le, re, nose, lm, rm] # 图片中 5 坐标的位置
        T=image.get_affine_transform(src_point, dst_point) # 根据获得的5点坐标与标准正脸坐标获取仿射变换矩阵
        a=image.warp_affine_ai(img, img_face, T) #对原始图片人脸图片进行仿射变换，变换为正脸图像
        a=img_face.ai_to_pix() # 将正脸图像转为kpu格式
        #a = img.draw_image(img_face, (128,0))
        del(face_cut_128) # 释放裁剪人脸部分图片
        # calculate face feature vector
        fmap = kpu.forward(task_fe, img_face) # 计算正脸图片的196维特征值
        feature=kpu.face_encode(fmap[:]) #获取计算结果
        reg_flag = False
        scores = [] # 存储特征比对分数
        for j in range(len(record_ftrs)): #迭代已存特征值
            score = kpu.face_compare(record_ftrs[j], feature) #计算当前人脸特征值与已存特征值的分数
            scores.append(score) #添加分数总表
```

图 3.17 人脸识别代码主要部分

如图 3.18，将人脸的特征值数据通过 with open 方法操作文件写入 SD 卡中的 data.txt 中，即可实现掉电存储人脸数据。

```
#写入特征点到SD卡（转换为二进制）
def save_feature(feat):
    with open('/sd/data.txt','a') as f:
        record =ubinascii.b2a_base64(feat)
        f.write(record)
```

图 3.18 将人脸数据写入 SD 卡的函数

那么在 SD 卡中存储好人脸的数据后，由于存储时将其转换为了二进制，在读取时也就可以通过 with open 同样进行读取再解析，由于 k210 模块设置好上电自动识别 SD 卡的内容，那么即可做到上电就重新将 SD 卡中 data.txt 的数据存储到数组中，则又可以识别已录入的人脸了，代码如图 3.19 所示。

```
#打开文件进行读取，如果有特征点信息，将其导入存储数组中
with open('/sd/data.txt','rb') as f:
    s = f.readlines()

    for line in s:
            #print(ubinascii.a2b_base64(line))
        record_ftrs.append(bytearray(ubinascii.a2b_base64(line)))
# check = 0
# save = 0
```

图 3.19 读取 SD 卡人脸数据的函数

3.7 k210 模块

K210 开发板我也是第一次使用，下面说明使用过程，首先是 MaixPy IDE 开发 K210：

1） 用 type-c 接口的数据线连接板子和电脑

2） 连接 MaixPy(烧录固件)

3） 运行实例代码

那么接下来实现人脸识别，过程如下：

进入官网（首次登陆需要注册）获取人脸识别源码，如图 3.20

图 3.20 MaixHub 官网人脸识别模型

然后获取 key_gen 机器码，即可下载文件，下载得到一个以机器码为名字的 kfpkg 文件，直接将该文件使用 klash_gui 软件烧录到开发板。

用 Maipy 连接开发板运行代码，至此就完成了最基本的人脸识别功能

根据代码来看，主要分为以下几个过程：

1）加载各种模型

2）运行人脸检测模型，在图片中找到人脸位置并框出人脸

3）将裁出的人脸图片转换成 kpu 接收的格式

4）运行人脸 5 点关键点模型，获取到左眼、右眼、鼻子、左嘴角、右嘴角的位置

5）对原始图片人脸图片进行仿射变换，变换为正脸图像，将正脸图像转为 kpu 格式

6）使用人脸 196 维特征值模型计算正脸图片的 196 维特征值，计算得到最终的人脸特征 feature

再将得到的人脸特征与之前保存过的人脸特征进行对比得到一组分数，选择其中最大的一个分数，且该分数超过 85 分（可以自己设置）就认为识别出该人，并根据对应下标从 names 列表中得到该人的姓名。

至此已实现人脸识别功能

3.8 微信小程序部分

微信小程序的设计由我负责，首先对于整体设计我将其设计为蓝白主题，将各个模块功能分开，如图 3.21 为首页也就是主界面

图 3.21 微信小程序首页

对于设计要求中的显示天气信息，我将天气信息显示写在这个主页面的第一个板块，天气信息是调用了和风天气的 API，在获取位置信息后即可获得天气信息，如图 3.22

```
* 根据坐标获取城市信息
*/
getCityByLoaction() {
  var that = this
  wx.request({
    url: 'https://geoapi.qweather.com/v2/city/lookup?key=' + APIKEY + "&location=" + that.data.
    location,
    success(result) {
      var res = result.data
      if (res.code == "200") {
        var data = res.location[0]
        that.setData({
          City: data.adm2,
          County: data.name
        })
```

图 3.22 获取天气的部分代码示意

门锁界面的设计我将一个圆形设计为门的状态显示，下方三个功能分别为密码解锁，人脸解锁，指纹解锁，人脸解锁由于官方接口关闭，实际上未能实现。在进入门锁界面时，会根据 js 中的阿里云三元组以及 topic，进行连接阿里云，如图 3.23 所示

```
    'options.username': clientOpt.username,
  })
  console.log("this.data.options host:" + host);
  console.log("this.data.options data:" + JSON.stringify(this.data.options));

  //访问服务器
  this.data.client = mqtt.connect(host, this.data.options);

  this.data.client.on('connect', function (connack) {
    wx.showToast({
      title: '连接成功'
    })
    console.log("连接成功");
  })
```

图 3.23 微信小程序连接阿里云物联网平台

对于密码解锁的设计，由于我不想多设计一个界面，所以我设计了一个隐藏的输入框，当点击密码解锁时候才会出现输入框并唤起手机的安全键盘，输入密码确认后又会消失，如图 2.24 为部分代码

```
checkPassword: function() {
  // 点击确认按钮时触发的函数，用于验证密码是否正确
  if (this.data.password === this.data.correctPassword) {
    wx.showToast({
      title: '密码解锁成功',
      icon: 'success',
      duration: 2000
    });
    // 在这里可以执行解锁成功后的操作
    that.onClickChange()
    setTimeout(() => {
      that.onClickChange();
    }, 3000);
  } else {
    wx.showToast({
      title: '密码错误',
```

图 2.24 微信小程序密码解锁部分代码

指纹解锁调用官方的接口，再根据官方文档给出的例程代码，稍加修改即可使用，部分代码如图 2.25

```
wx.checkIsSupportSoterAuthentication({
  success(res) {
    if (res.supportMode.includes('fingerPrint')) {
      // 支持指纹识别，调用指纹识别接口
      wx.startSoterAuthentication({
        requestAuthModes: ['fingerPrint'],
        challenge: 'challenge', // 这里可以自定义一个挑战值
        success(res) {
          if (res.errCode === 0) {
            // 指纹验证成功，执行解锁操作
            console.log('指纹验证成功');
            // 在这里可以执行解锁操作
            that.onClickChange()
            setTimeout(() => {
              that.onClickChange();
            }, 3000);
```

图 2.25 微信小程序指纹解锁部分代码

至此微信小程序设计部分结束。

第四章 总结

4.1 遇到的问题

1）上传数据到 RDS 数据库时显示错误，原因是数据类型不匹配。

2）ESP8266 端和阿里云通信失败，原因是订阅和发布的主题设置不正确

3）微信小程序端接收消息后并未正确处理，原因是对 json 格式解析有误

4）从 Maixhub 官网下载的人脸识别模型在 k210 开发板上运行失败，原因是下载模型需要自己 k210 开发板的机器码，每个 k210 板子的机器码不一样，需要自己烧录固件查看机器码

4.2 问题的解决过程

1）在 RDS 数据库和云产品流转中将数据类型设置为 int。

2）进入阿里云物联网平台，设置消息云流转，将两端 topic 对应，通信成功

3）重新修改代码，并让发送消息的格式固定，在微信小程序端则可以正常地解析并进行后续操作

4）重新烧录固件查看自己机器码后，在下载模型时填写自己的机器码后，模型即可用

4.3 个人体会

见个人报告。

4.4 不足之处与展望

本次设计不足的地方就是，STM32 对 ESP8266 从阿里云云流转过来的消息没有正确地接收，用串口助手看 ESP8266 收到的消息是正确的，但是 ESP8266 将消息发送给单片机似乎有问题，猜测是对 json 格式消息的解析出了问题，但是这个问题没有解决也就意味着微信小程序端无法控制单片机端，是一个遗憾之处。而微信小程序端的人脸识别功能，由于微信小程序官方关闭了人脸识别的接口，导致想要识别人脸这个功能变得比较困难，在本次设计中也没有实现。

未来希望能实现单片机对云流转来的消息正确的接收并处理，以及实现门锁可以根据一些特殊情况进行报警或者通知这样的功能。

致谢

我们想要向所有在这个项目中给予支持和帮助的人表达最诚挚的感谢和深深的敬意。在这个项目的每个阶段，您们的支持和协助都是不可或缺的。在本次课程设计中，感谢老师为我们提供模块设备和指导，感谢小组成员们的团结写作和不懈解决问题。也很感谢 Maixhub 官网提供的人脸识别模块，使我们顺利完成人脸识别这个功能，在过程中我们也有遇到许多问题，例如：k210 的使用需要烧录固件，而不同的固件运行效果不同，在上电后需要将 k210 复位，并将 STM32 复位才可继续进行后续操作。在遇到困难的过程中，小组成员们积极解决问题，也感谢网络上各个作者为我们提供的解决问题的思路，再次感谢每一位为这个项目付出过努力和支持过我们的人，没有你们，本次设计将无法取得如此成功的成果。

参考文献

[1] 周奕. 运用人脸识别技术的宿舍门禁系统设计[J]. 信息系统工程, 2020(2): 21-22.

[2] 董理. 企业级人脸识别门禁系统的应用研究[J]. 中国管理信息化：综合版, 2020(10): 178-179.

[3] 朱宁. 基于人脸识别技术的高校图书馆门禁系统设计[J]. 无线互联科技, 2020, 17(13): 95-97.

[4] 王灿田. 基于人脸识别的嵌入式在线网页门禁控制系统[J]. 电视技术, 2020, 44(05): 32-38.

[5] 郑定超. 智慧指纹门禁系统设计[J]. 电脑知识与技术, 2020, 16(33): 88-90.

[6] 宿静宜, 刘久付, 杨明海. 采用人脸身份识别的智慧门禁系统[J]. 软件导刊, 2019, 18(4): 32-35.

[7] 杨仁山, 陈玉龍, 伍银波. 一种开放式智慧门禁控制系统设计与应用[J]. 电子技术与软件工程, 2021(18): 124-126.

[8] Dinculeană D, Cheng X. Vulnerabilities and limitations of MQTT protocol used between IoT devices[J]. Applied Sciences, 2019, 9(5): 848.

[9] 陈亚聪, 冯兴乐, 陈书鹏. 基于 MQTT 协议的智能家居模块化拓展系统[J]. 单片机与嵌入式系统应用, 2024, 24(02): 86-90.

[10] 王亚东. 基于 MQTT 协议的智能开关系统的设计[J]. 科技创新导报, 2022, 19(4): 53-55.

[11] 申志强, 赵天翔. 基于阿里云的实时天气状况监测装置设计[J]. 电脑知识与技术, 2021, 17(36): 160-161+164.

[12] Dewanto S A, Munir M, Wulandari B, et al. Mfrc522 rfid technology implementation for conventional merchant with cashless payment system[C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 1737(1): 012012.

[13] 曹伟. MySQL 云数据库服务的实现[J]. 程序员, 2012(12): 97-101.

[14] 吕瑞妮, 程雨鑫, 富晓乾, et al. 基于 AS608 光学指纹识别模块研制防逃课识别装置[J]. 电子测试, 2020(16): 60-61+115.

[15] 张乐, 王悦. 基于 STM32 的指纹识别系统设计[J]. 沈阳大学学报(自然科学版), 2019, 31(02): 113-117.

附件 A 设计图纸

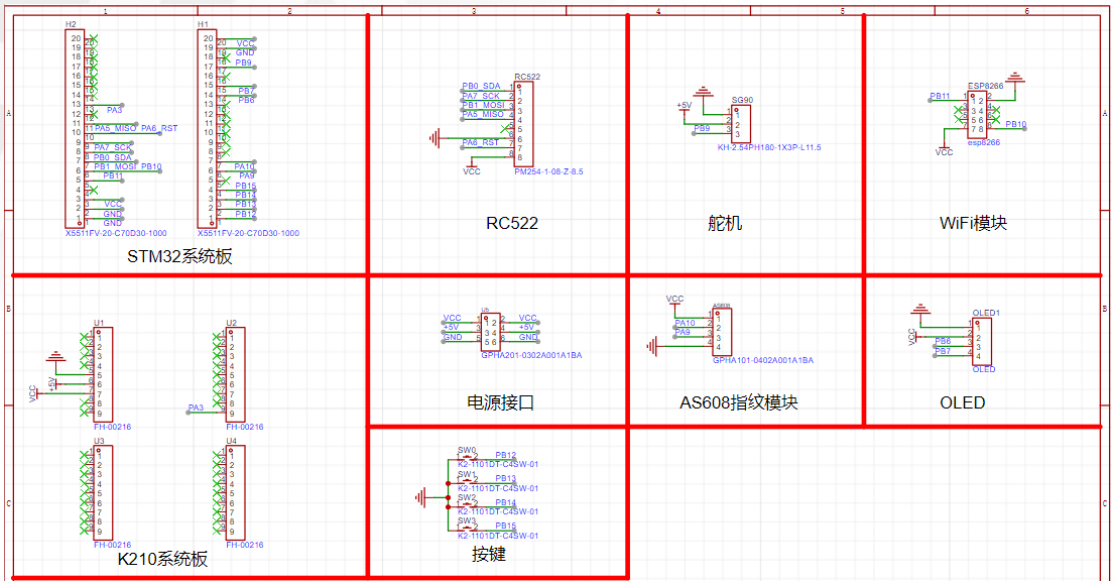

图 1 电路原理图

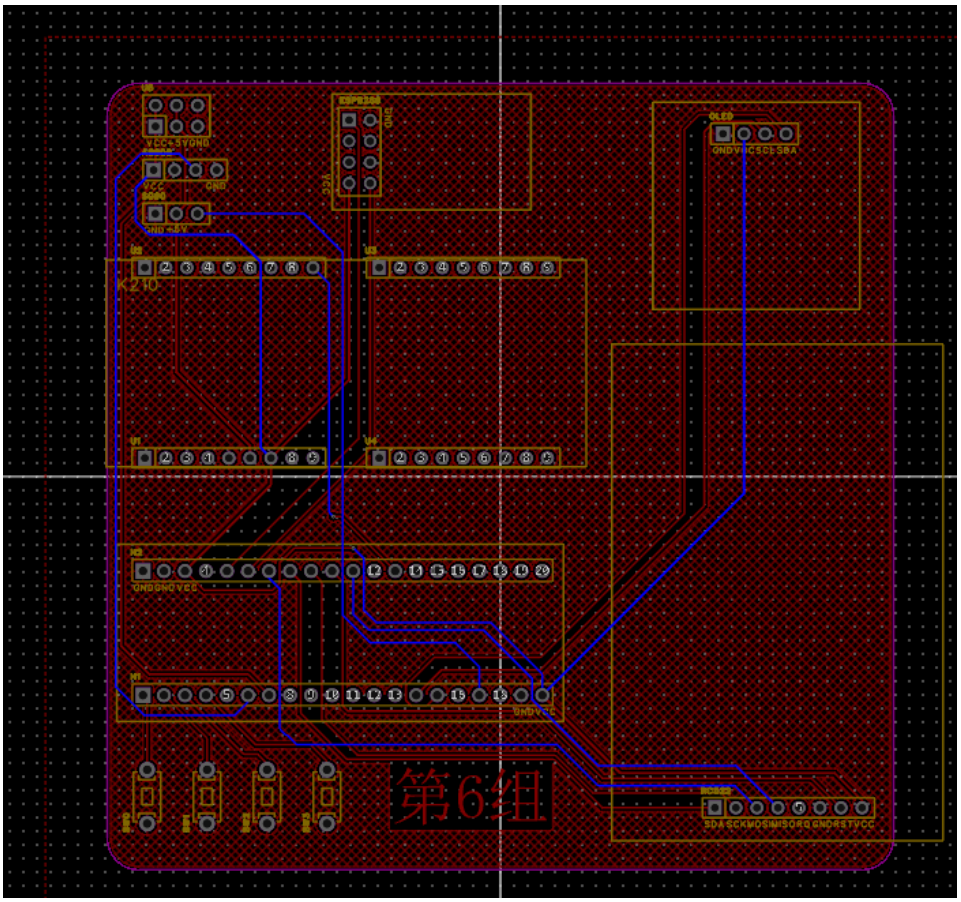

图 2 PCB 图