2018 贵州省“欣源杯”大学生电子设计竞赛

无线联网刷卡系统（H 题）

2018 年 6 月 20 日

摘要

本文介绍了以无线路由器与 Wi-Fi 模块为基础构建的一个无线联网刷卡系统。用户刷卡信息通过 MFRC522 射频模块接收，然后由 ESP8266 Wi-Fi 模块发送给无线路由器，再由无线路由器上传到 PC 机。PC 机在接受到信息后存储在数据库中，通过 MVC 模式最终将数据同步在网站上，管理员可以通过手机或 PC 端登录网站查看持卡人信息及相应的刷卡数据，实现用户信息的管理和信息安全监控。

本系统中，集合了普通用户和管理员用户两种权限和设备号自定义的功能，并将不同的用户组加以区分，实现了普通用户数据上传、新建管理员、新建普通用户、删除普通用户、删除普通用户且删除用户数据等共计五种功能。在网站之中还可使用设备注册、用户注册以及数据查询等功能。

相较于传统的上位机程序设计，本系统采用的网站设计不仅可以在本机之中使用，也可以搭建在远端服务器之中，实现真正的跨平台访问。真正满足用户在实际生产生活中的多种需求。

关键词：ESP8266； MFRC522；MVC

目录

0. 引言

非接触式 IC 卡（又称射频卡）的刷卡管理系统是是以计算机管理为核心、以非接触式 IC 卡为信息载体、以刷卡器为终端的全新智能管理系统，现已广泛应用于企业、机关、学校、银行等公共场所。使用者只需持一张经过授权的 IC 卡进行感应读卡，即可完成各种刷卡过程。系统在后台强大的软环境和完善的硬件基础上完成信息加工处理工作，统一进行 IC 卡的发行、授权、取消、挂失、充值等工作，并可查询、统计、清算、报表、打印各类消费信息及其它相关业务信息。这些对于人们生活的便利都有了很大的帮助。基于这些要求，本文设计了并制作了基于无线联网的刷卡系统。

1. 预期实现目标

本文当中预期设计一套无线联网刷卡系统，以无线路由器与 WIFI 模块为基础构建一个无线联网刷卡系统。该系统装置由两台读卡器、多张射频卡（磁卡）、无线 AP、个人计算机和数据采集窗口等组成。持卡人可在读卡器设备上进行刷卡操作，读卡器设备收到卡片信息后通过无线 AP 将信息上传到后台之中，并将刷卡情况返回至刷卡设备端。持卡人信息及刷卡数据将统一存储在后端数据库中，管理员可实时访问网站查询任意持卡人刷卡数据信息。非数据库记录的持卡人不能刷卡进行任何操作，管理员可添加或删除指定持卡人且随时监控整个刷卡网络系统的信息安全。同时，在设备之中预置了设备号。此设备号将作为设备的唯一识别码，用于与主机进行通讯时实现校验功能，设备号一般不可见，用来防止信息泄露造成的信息丢失或数据库注入。

2. 方案论证与设计

2.1. 方案论证

2.1.1. 射频卡信息读取模块方案

方案一：采用 ACR122U 射频卡信息读取模块，其具有 Mifare、CPU、Felica 等卡的读取功能，但是成本较高。

方案二：采用 MFRC522 射频卡信息读取模块，其使用了先进的调制和解调概念，完全集成了在 13.56MHz 下所有类型的被动非接触式通信方式和协议。它与主机间的通信采用连线较少的串行通信，且可根据不同的用户需求，

选取 SPI、I2C 或串行 UART(类似 RS232)模式之一，有利于减少连线，缩小 PCB 板体积，降低成本。

综上所述，选择方案二。

2.1.2. 网络连接模块方案

方案一：采用 W5500 以太网模块，W5500 是一款全硬件 TCP/IP 以太网控制器，提供了 SPI（外设串行接口）从而能够更加容易与 MCU 整合。但是价格较贵，传输距离短，不能跨网络无线远距离传输，编程复杂。

方案二：采用 RM04 Wi-Fi 模块，使用 5V 直流电源，提供串口透传功能，优点是是价格便宜，但缺点也明显，功耗大，不稳定，时常有丢包的现象。

方案三：采用 ESP8266 Wi-Fi 模块，不仅可以实现数据传输功能，还可控制建立 Wi-Fi 热点，或者作为 Wi-Fi 客户端连接到某指定路由器，这款芯片使用了 3.3V 的直流电源，体积小，功耗低，支持透传，丢包率较低，而且价格低。

综上所述，采用方案三的 ESP8266 Wi-Fi 模块。

2.1.3. 软件设计方案

方案一：采用 PC 端上位机设计控制软件，上位机是可以直接发出操控命令的计算机，其可通过控制下位机（单片机）获取信息并发送操控作指令。但 PC 端上位机只能在电脑上进行操作。

方案二：采用建设网站的方法，在网站来发布和管理公布的信息。网站可通过手机和电脑实时访问，操作简单。

综上所述，选择方案二。

2.1.4. 数据库设计方案

方案一：采用 MongoDB 数据库，它是一个基于分布式文件存储的数据库。由 C++语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。它的特点是高性能、易部署、易使用，存储数据非常方便。

方案二：采用 MySQL 数据库，其是一个关系型数据库管理系统。体积小、速度快、总体拥有成本低。但其不支持热备份，安全系统复杂而非标准，只有到调用 mysqladmin 来重读用户权限时才发生改变，且没有一种存储过程语言。

方案三：采用 SQLServer 数据库，其是一个具备完全 Web 支持的数据库产品，提供了对可扩展标记语言(XML)的核心支持以及在 Internet 上和防火墙外进行查询的能力。但其难以处理日益增多用户数和数据，卷伸缩性有限，且安全性方面欠佳。

综上所述，选择方案一。

2.2. 总体设计概述

系统总体设计框架如图所示：

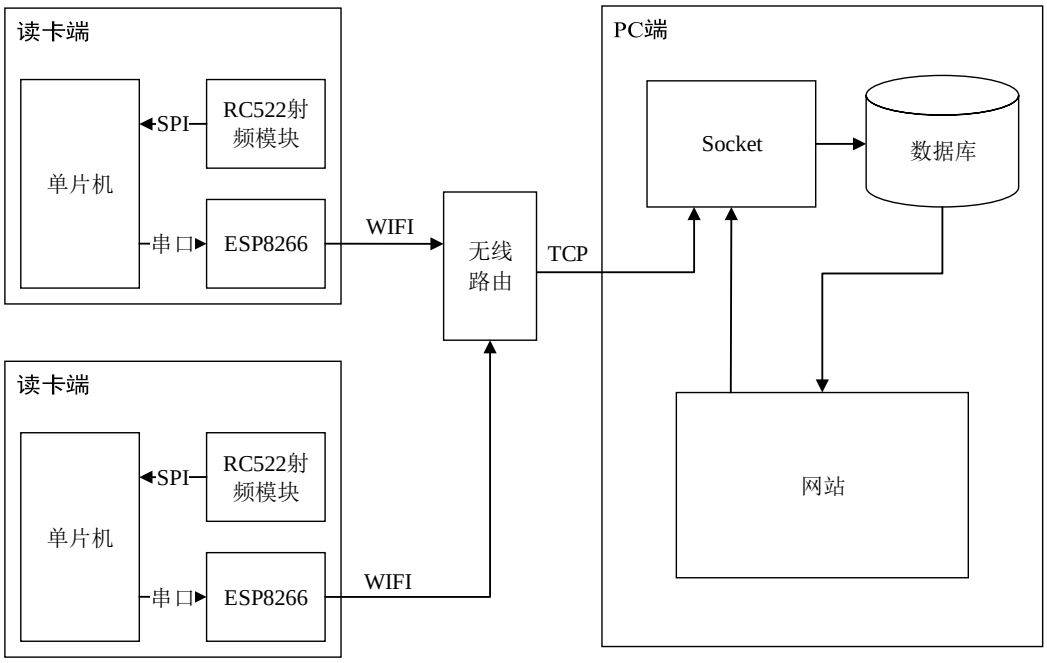

图 1 系统总体设计框架图

RC522 射频模块与 ESP8266 Wi-Fi 模块连接在单片机上，用户刷卡信息通过 RC522 射频模块接收，然后由 ESP8266 Wi-Fi 模块发送给无线路由器，由无线路由器上传到 PC 机后台，后端接受到信息后存储在数据库中，最后将数据同步在网站上，管理员可以通过手机或 PC 端登录网站看持卡人信息及相应的刷卡数据，实现用户信息的管理和信息安全监控。

3. 硬件设计与原理分析

3.1. 电路总体设计图

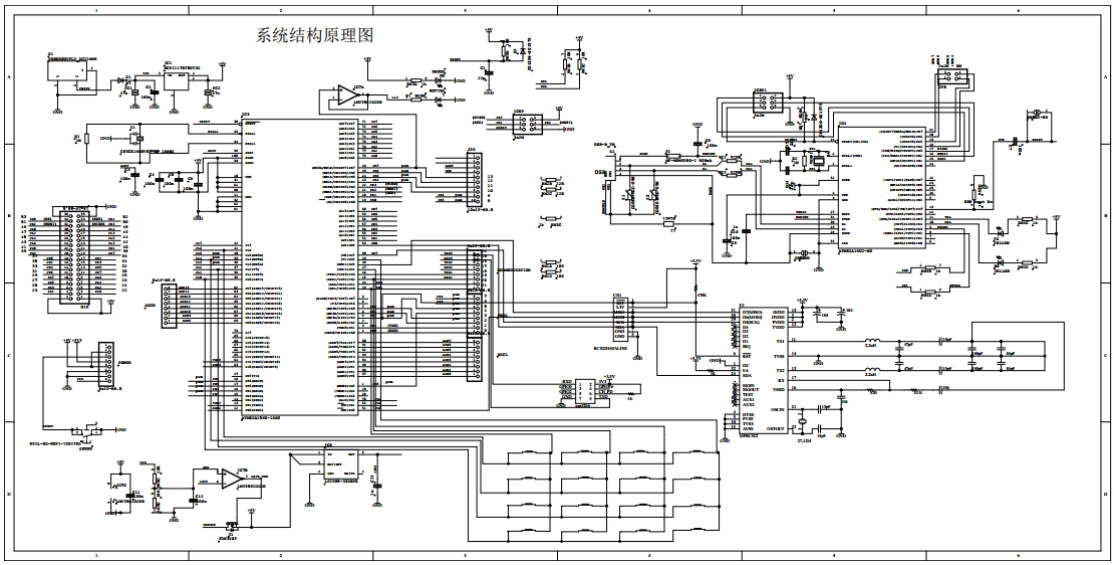

图 2 系统结构原理图

3.2. 射频卡信息读取模块

3.2.1. 原理分析

射频信息卡读取模块主要采用 MFRC522 实现，其结构框图如图 3 所示。检测器部分检测刷卡信息，模拟接口用来处理模拟信号的调制和解调，非接触式 UART 用来处理与主机通信时的协议要求。FIFO 缓冲区快速而方便地实现了主机和非接触式 UART 之间的数据传输，最后模块通过接口将卡片信息发送给单片机。

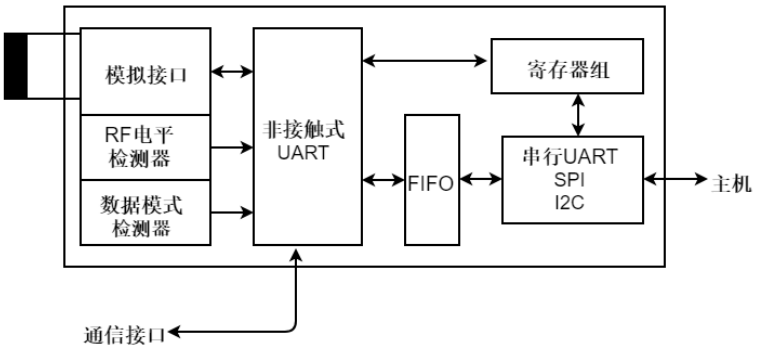

图 3 MFRC522 模块结构框图

3.2.2. 硬件电路设计

MFRC522 模块电路图如图 4 所示，其中 D7、D6、D5 连接 MISO、MOSI、SCK，串行接口类型为 SPI。在 SPI 通信中 MFRC522 模块用作从机。SPI 时钟 SCK 由主机产生。数据通过 MOSI 线从主机传输到从机；数据通过 MISO 线从 MFRC522 发回到主机。OSCIN 为晶振输入端；TX1、TX2 为发送器，用来传递调制的 13.56MHZ 的能量载波信号；EA 为外部地址，用来编码 I2C 地址；I2C 的作用是使 I2C 使能；RX 为接收器输入，是接收 RF 信号的管脚；VMID 为内部参考电压，提供内部参考电压；OUCOUT 为晶振输出，振荡器的反向放大器的输出；TVDD 为发送器电源，给 TX1 和 TX2 的输出级供电；DVDD 和 AVDD 分别为数字电源和模拟电源；PVDD 为管脚电源；而 TVSS 为发送器地，TX1 和 TX2 的输出级的地；DVSS 和 AVSS 分别为数字地和模拟地；PVSS 为管脚电源地。

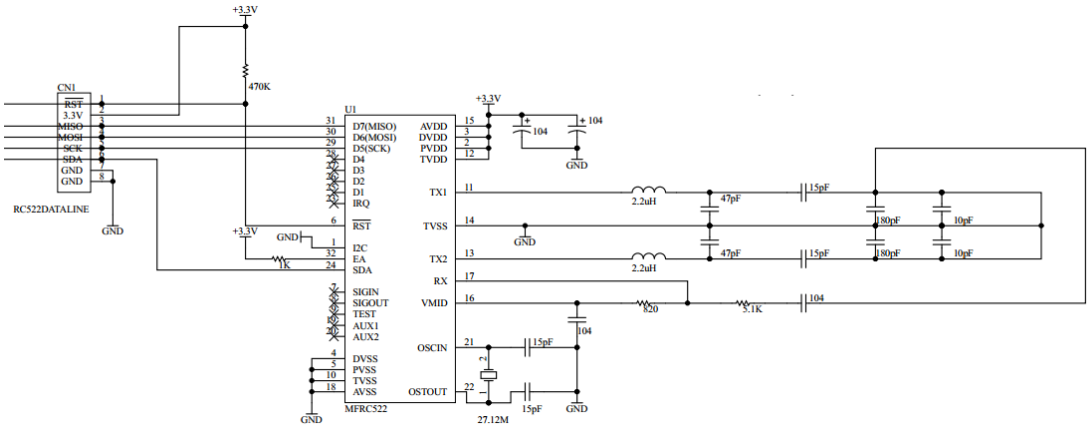

图 4 射频卡信息读取模块电路图

3.3. 网络连接模块

3.3.1. 原理分析

网络连接模块采用 ESP8266 模块，承担着将数据传送到路由器的任务，其结构如图 5 所示。ESP8266 是一个完整且自成体系的 Wi-Fi 网络解决方案，能够独立运行，也可以作为 slave 搭载于其他 Host 运行。ESP8266 强大的片上处理和存储能力，使其可通过 GPIO 口集成传感器及其他应用的特定设备，实现了最低前期的开发和运行中最少地占用系统资源。ESP8266 高度片内集成，包括天线开关 balun、电源管理转换器，因此仅需极少的外部电路，且包括前端模

块在内的整个解决方案在设计时将所占 PCB 空间降到最低。

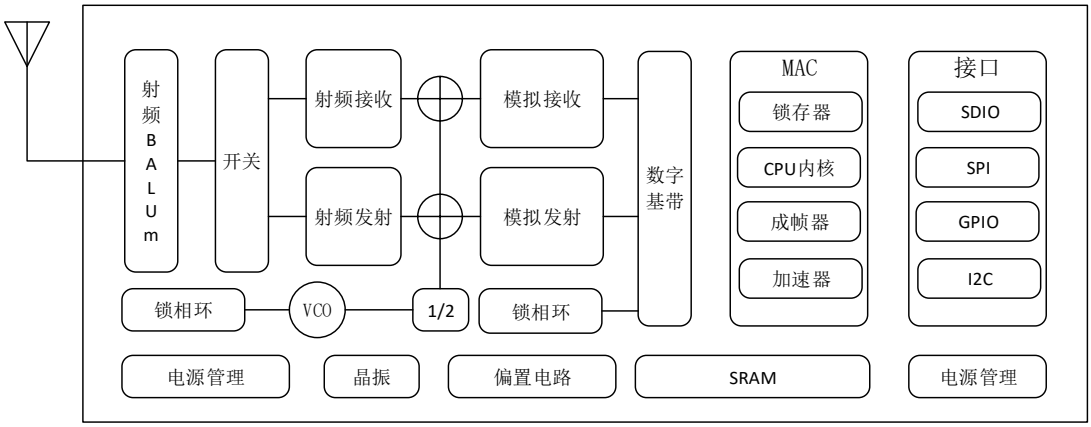

图 5 ESP8266 结构图

3.3.2. 硬件电路设计

ESP8266 模块电路图如图 6 所示，其中 VCC 接 3.3V 电源，GND 接地；TXD 为模块串口发送脚，接单片机 RXD3；RXD 为模块串口接收脚，接单片机 TXD3；CH_PD 接高电平工作，低电平模块供电必须关掉。

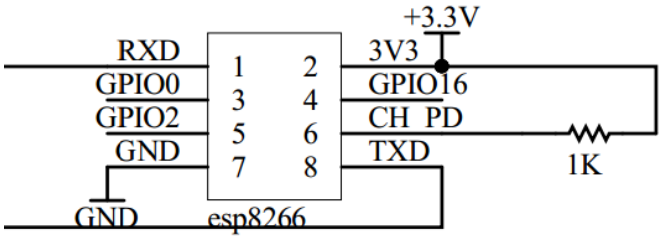

图 6 网络连接模块电路图

4. 软件设计与流程

4.1. 单片机程序设计

本次单片机硬件部分采用了 Atmel 公司出品的 MEGA2560，程序采用了 Arduino 开发环境，使用了 Arduino 类库和自己编写的 WirelessCard 库作为主要开发文件。

本系统中内设 5 种工作模式，分别为：普通用户刷卡模式、新建管理员模式、新建普通用户模式、删除用户但保留数据模式以及删除用户且删除相关数据模式。其中普通用户上传数据时需在后端验证其用户的存在性。本系统独创

了管理员模式的特色功能。管理员模式在应用时首先放置用户卡片，之后放置管理员卡片，并输入管理员密码。

本系统还采用了数据加密功能，保障数据通信的安全性。主要加密方式为在原有数据上对数据进行加 1 操作，例如设备号“27A9D773”在进行加 1 操作后即变为“38B:E884”。之后在后端再进行减 1 操作即可。

具体开发流程可参照附录中的相关流程图步骤。

4.2. 网站与后端设计

网站后端采用 Django 和 MVC(MTV)架构，前端采用 Semantic-UI 样式库。其中，MVC 指 Models、Views 和 Contents。Models 负责网站后端与数据库的连接，Views 负责处理请求，并返回相关的 Models 和 Contents 数据，Contents 主要负责网站的具体内容。整体网站和传感器数据上传功能皆由后端 Models 和 Views 协同操作。具体 MVC 模型的架构如下所示。

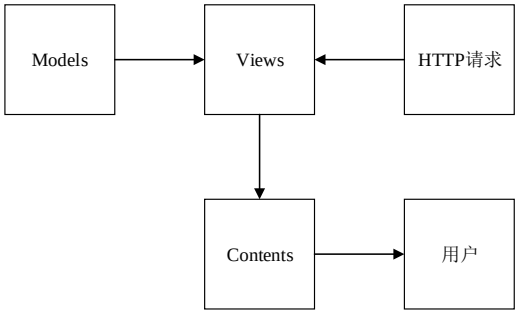

图 7 MVC 模型架构

后台会根据上传的请求数据进行检查并解析，并根据执行结果以 JSON 数据格式返回错误码（例如：{“error”: 0}），具体详见下表。

表 1 返回错误码

错误号	错误名	说明
0	success	无错误
1	data process error	是 POST 请求，但是数据处理失败
2	user does not exist	用户不存在
3	wrong uid	管理员 uid 格式错误或管理员 uid 前后不等或管理员设置的密码有误
4	admin account error	uid 错误或管理员账号或密码错误
5	mode error	模式选择错误

6	device_id error	设备号错误
7	data type error	数据结构不完整或错误
8	request error	不是 POST 请求
9	account already exists	管理员账户已存在
10	user already exists	用户已存在

具体操作逻辑可参考附录中的流程图步骤。

4.3. 数据库系统设计

数据库采用 NoSQL 技术具有代表性的 MongoDB 数据库，主要分为三个表格。表格及相关字段信息分别如下所示。

表 2 device 表

字段名	数据类型	说明
device_id	String	设备号
description	String	设备描述

表 3 user 表

字段名	数据类型	说明
uid	CHAR(8)	用户 UID
department	CHAR(6)	用户所属类别
password	INT32	用户密码

表 4 user_data 表

设备号	连接正确率	说明
uid	CHAR(8)	用户 UID
device_id	String	设备号
time	ctimestamp	数据上传时间
action_num	INT32	动作号
toUser	CHAR(8)	操作对象

5. 作品成效总结分析

5.1. 作品测试分析

5.1.1. 测试条件

测试一：220V 交流电压输入、两张不同 UID 卡号的复旦卡、一台 PC、一台已连接互联网的无线路由器、两套不同设备号的读卡端。

在以上仪器设备的条件下进行刷卡操作，分别进行 100 次测试后得出数据返回值测试等信息。

测试二：3.3V 直流电压输入、Wi-Fi 模块一块、ATMEL MEGA2560 单片机一块、一台 PC。

在以上仪器设备的条件下使用 Wi-Fi 模块与 PC 端进行 TCP 连接并发送 10000bit 的信息，测试 Wi-Fi 模块的丢包率。

测试三：3.3V 直流电压输入、MFRC522 模块一块、ATMEL MEGA2560 单片机一块、两张 UID 卡号不同的复旦卡、一台 PC。

在以上仪器设备的条件下进行读卡操作，分别对每张卡进行 100 次读取操作，测试读卡模块的错误率。

测试四：220V 交流电压输入、一台 PC、一台已连接互联网的无线路由器、两套不同设备号的读卡端。

在以上仪器设备的条件下测试 Wi-Fi 模块与路由器连接以及与后端连接的成功率。分别测试 20 次，并将测试结果通过串口打印在 PC 机上。

5.1.2. 测试结果

表 5 测试一测试结果

卡号	返回值正确率
C53C18AB	99.00%
27A9D773	99.00%

表 6 测试二测试结果

WIFI 模块丢包率
0.06%

表 7 测试三测试结果

设备号	读卡正确率
123XX123	99.00%
321HH321	99.00%

表 8 测试四测试结果

设备号	连接正确率
123XX123	85.00%
321HH321	80.00%

5.2. 创新特色总结展望

在本次测试过程之中，我们发现：在网络情况良好的情况下，只要 Wi-Fi 模块与路由器、与 PC 机后台之间能够正确地建立 TCP 连接，模块数据上传的正确率就可以接近 100%。但是 ESP-01 模块与路由器连接时不稳定，可能会出现连接失败或掉线的情况出现，而当其与无线路由器建立正确连接之后，它与网站后台之间连接的正确率就基本可以达到 100%。对比得知，在整个读卡端部分，读卡模块工作较为稳定，而 Wi-Fi 模块的稳定性还有待提高。

由于本套系统的连接部分使用了 ESP8266 系列的 ESP-01 模块，故若想能够提高稳定性，则应更换成同系列的 ESP-12 型号模块或稳定性更高的模块，或使用 GSM 模块并加装物联卡以满足不同场景下的多种需求。

参考文献

[1] 胡仁杰等. 全国大学生电子设计竞赛作品设计报告选编[M]. 东南大学出版社. 2016

[2] 尹寒, 陈峰. 近耦合射频识别系统的工作原理及天线设计[J]. 单片机与嵌入式系统. 2002

[3] NXP. MFRC522 Datasheet[EB/OL]. 2007

附录

1. 单片机程序设计流程图

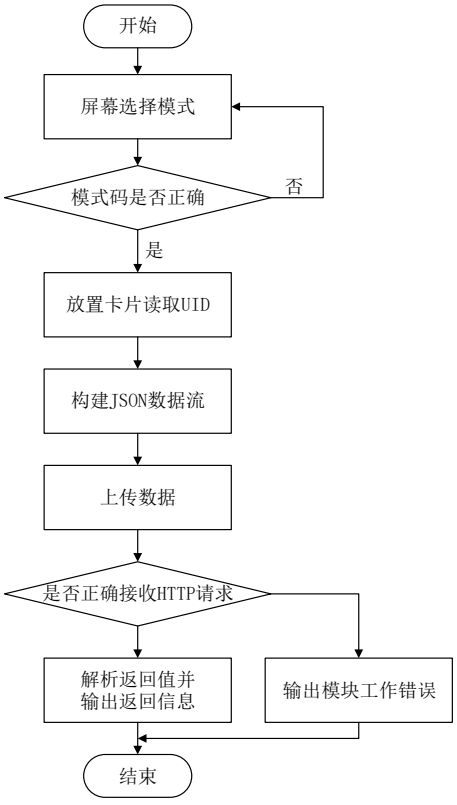

图 8 模式 0：普通用户上传数据

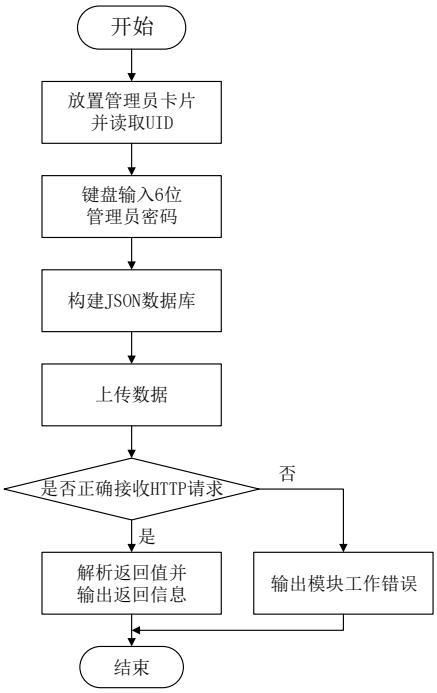

图 9 模式 1：新建管理员

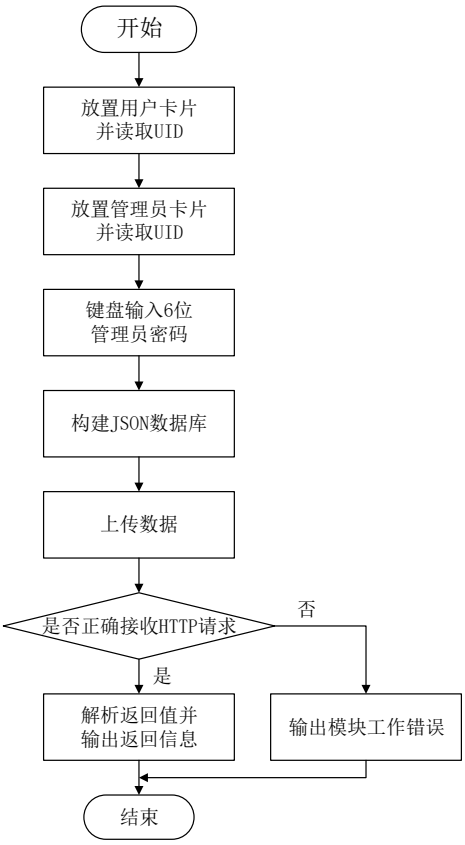

图 10 模式 2-4：新建用户、删除用户、删除用户及数据信息

2. 网站后台解析处理流程图

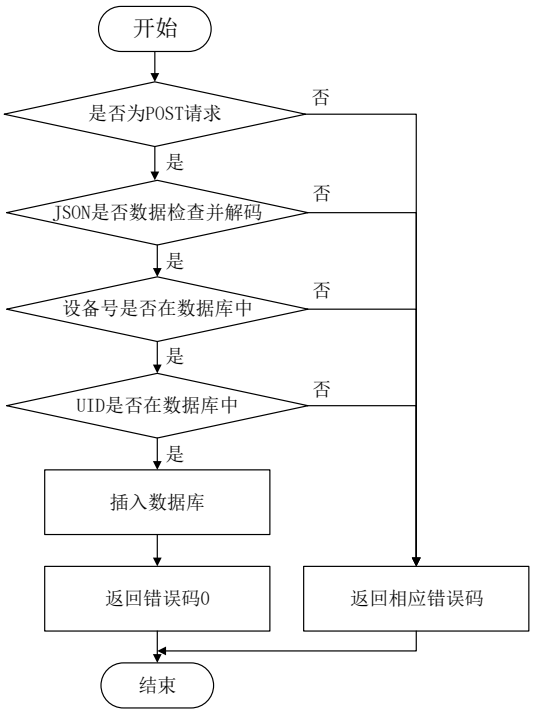

图 11 模式 0：普通用户上传数据

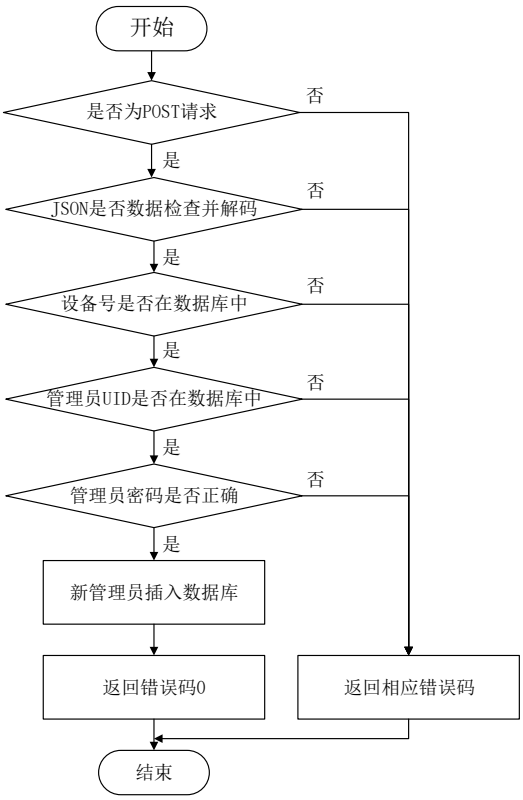

图 12 模式 1：新建管理员

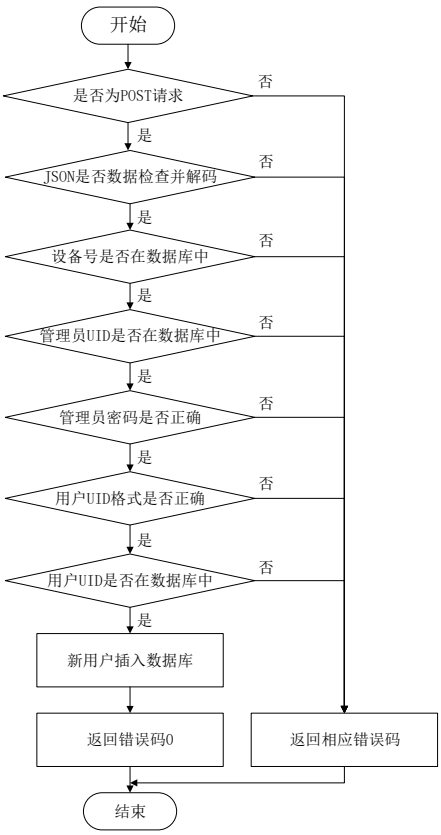

图 13 模式 1：后台新建普通用户、删除用户、删除用户及信息

3. 网站后台关键代码

```
def data_post(request):
    response = {
        'error': 0,
        'uid': None,
    }
    if request.method == 'POST':
        try:
            try:
                post_data = post_data_process(request.body) # convert request data to dict
(json)
                if post_data['uid'] != "":
                    post_data['uid'] = post_data['uid'].upper()
                    post_data['uid'] = decodeString(post_data['uid'])
                if post_data['admin']['admin_uid'] != "":
                    post_data['admin']['admin_uid'] =
post_data['admin']['admin_uid'].upper()
                    post_data['admin']['admin_uid'] =
decodeString(post_data['admin']['admin_uid'])
                if post_data['admin']['admin_pswd'] != "":
```

```
                post_data['admin']['admin_pswd'] =
post_data['admin']['admin_pswd'].upper()
                post_data['admin']['admin_pswd'] =
decodeString(post_data['admin']['admin_pswd'])
            print(post_data)
            print(decodeString(post_data['uid']))
        except:
            # 数据处理失败
            response['error'] = 1
            raise Exception
        if post_data_check(post_data) is True:
            response['uid'] = post_data["uid"]
            if post_data['action'] == 0:
                # 普通数据插入
                user_data_table.insert_one({
                    "uid": post_data['uid'],
                    "department": "common",
                    "time": current_time(),
                    "device_id": post_data['device_id'],
                    "action_num": 0,
                    "description": "user_data",
                    "toUser": None
                })
                response['error'] = 0

            elif post_data['action'] == 1:
                # 新建管理员
                user_table.insert_one({
                    "uid": post_data['admin']['admin_uid'],
                    "department": "admin",
                    "password": post_data['admin']['admin_pswd']
                })
                user_data_table.insert_one({
                    "uid": post_data['admin']['admin_uid'],
                    "department": "admin",
                    "time": current_time(),
                    "device_id": post_data['device_id'],
                    "action_num": 1,
                    "description": "new_admin",
                    "toUser": None
                })
                response['error'] = 0

            elif post_data['action'] == 2:
```

```
        # 新建普通用户
        user_table.insert_one({
            "uid": post_data['uid'],
            "department": "common",
            "password": None
        })
        user_data_table.insert_one({
            "uid": post_data['admin']['admin_uid'],
            "department": "admin",
            "time": current_time(),
            "device_id": post_data['device_id'],
            "action_num": 2,
            "description": "new_user",
            "toUser": post_data["uid"]
        })
        response['error'] = 0

    elif post_data['action'] == 3:
        # 删除用户但保留数据
        user_table.delete_many({
            "uid": post_data["uid"],
            "department": "common"
        })
        user_data_table.insert_one({
            "uid": post_data['admin']['admin_uid'],
            "department": "admin",
            "time": current_time(),
            "device_id": post_data['device_id'],
            "action_num": 3,
            "description": "delete_user_remain_data",
            "toUser": post_data["uid"]
        })
        response['error'] = 0

    elif post_data['action'] == 4:
        # 删除用户及其数据
        user_table.delete_many({
            "uid": post_data["uid"],
            "department": "common"
        })
        user_data_table.delete_many({
            "uid": post_data["uid"],
            "department": "common"
        })
```

```
                        user_data_table.insert_one({
                            "uid": post_data['admin']['admin_uid'],
                            "department": "admin",
                            "time": current_time(),
                            "device_id": post_data['device_id'],
                            "action_num": 4,
                            "description": "delete_user_without_data",
                            "toUser": post_data["uid"]
                        })
                        response['error'] = 0
                    else:
                        response['error'] = 7
                        raise Exception
                else:
                    response['error'] = post_data_check(post_data)
                    raise Exception
            except:
                pass
        else:
            # 不是 POST 请求
            response['error'] = 8

        del response['uid']
        res = JsonResponse(response)
        del res['Date']
        del res['Server']
        del res['X-Frame-Options']
        print("error: ", response['error'])
        # return HttpResponse(simplejson.dumps(response), content_type='application/json')
        return res
```

4. 单片机关键代码

```
/*
 * @intro: 构建并上传 json 数据
 * @param <String> deviceID: 设备号
 * @param <String> uid: 卡号 UID
 * @param <int> action: 动作号，即模式号
 * @param <String> adminUID: 管理员模式下的管理员卡号
 * @param <String> adminPSWD: 管理员模式下管理员账号的密码
 */
void Card::jsonDataUpdate(String deviceID, String uid, int action, String adminUID, String
adminPSWD)
{
```

```
    uid.toUpperCase();
    jsonConstruct(deviceID, uid, action, adminUID, adminPSWD);
    WIFISerial.print("POST /data_post HTTP/1.1\r\n");
    WIFISerial.print("Host: 39.108.7.66\r\n");
    WIFISerial.print("Content-Type: application/json\r\n");
    WIFISerial.print("Content-Length: " + String(jsonData.measureLength()) + "\r\n");
    WIFISerial.print("\r\n");
    jsonData.printTo(WIFISerial);
    WIFISerial.print("\r\n");
}
/*
 * @intro: 查看返回的 json 数据
 * @return <int>: 返回错误码，若返回-1，则返回值解析失败
 */
int Card::jsonDataReturn()
{
    char temp;
    int pos;
    int commaPosition, commaPosition1, commaPosition2;
    String inputString, jsonString, wantedString1, wantedString2;
    memset(esp8266buffer, 0, 256);
    WIFISerial.readBytes(esp8266buffer, 256);
    for (unsigned int ii = 0; ii < 256; ii++)
    {
        temp=esp8266buffer[ii];
        inputString += temp;
    }
    // DebugSerial.println(inputString);
    if (inputString.indexOf("HTTP/1.1 200 OK") != -1)
    {
        commaPosition=inputString.indexOf('{');
        wantedString1=inputString.substring(commaPosition,inputString.length());
        DebugSerial.println(wantedString1);
        commaPosition1=wantedString1.indexOf('}');
        commaPosition2=wantedString1.indexOf(':');
        wantedString2=wantedString1.substring(commaPosition2+2, commaPosition1);
        // DebugSerial.println(wantedString2);
        return wantedString2.toInt();
    }
    return -1;
}
```