西北工業大學

NORTHWESTERN POLYTECHNICAL UNIVERSITY

本科毕业设计论文

题 目 __基于物联网的智能宿舍系统__

专业名称：自动化

学生姓名：贺林

指导教师：李彬

毕业时间：2017 年 7 月

毕业 设计 论文 任务书

一、题目

基于物联网的智能宿舍系统

二、研究主要内容

本设计基于物联网的三层架构：感知层、网络层和应用层，设计各层软硬件架构，并制作具有基本功能的实物。

1. 感知层：利用单片机及各类传感器模块，实现智能门锁、智能插座、温度及烟雾报警器、红外遥控器等；
2. 网络层实现的内容为：通过一个广播式 433MHz 通信模块实现类 ZigBee 的自组网功能；
3. 应用层实现的内容为：使用树莓派，处理、存储及显示来自感知层及用户的数据消息。

三、主要技术指标

4. 设计一种自组网数据传输协议；
5. 基于该协议对单片机进行编程，制作具有自组网功能的终端、中继及中控设备；
6. 设计及制作具有各种功能的终端设备，包括门锁、插座、红外遥控、温度及烟雾检测装置；
7. 中控能够将接收到的各类数据分析、处理并存储在数据库中；
8. 在中控处搭建服务器，使用户能够通过网页控制、查询各终端设备。

四、进度和要求

01-02 周　　查阅相关资料，明确设计内容、设计流程，撰写开题报告；

03-04 周　　参阅 ZigBee 协议，设计一种自组网数据格式；

05-06 周　　进行终端设备设计，实现各终端设备中单片机功能编程；

07-08 周　　基于设计的数据格式，初步完成自组网系统编程，包括终端设备编程及中继编程；

09-10 周　　调试已完成的自组网系统，排除一些存在的问题，自组网系统调试完成后，开始进行中控（树莓派）设计；

11-12 周　　完成中控编程，将整套设备调试至能够正常运行状态；

13-14 周　　设计各终端、中继、中控设备所需电路板及外形，将各类设备制作完成；

15-16 周　　完成毕业论文的撰写。

五、主要参考书及参考资料

[1] 吴小芳, 物联网与大数据的新思考[J], 通讯世界, 卷 01, pp. 1-2, 2017.

[2] 郭天祥, 51 单片机 C 语言教程[M], 北京: 电子工业出版社, 2013, pp. 2-11.

[3] STC 公司, STC15F2K60S2 系列单片机器件手册[M], 2014.pp.63-83

[4] 宁炳武, Zigbee 网络组网研究与实现[D], 大连理工大学, 2007.

[5] 陈慕雷.基于多模传输协议的智能家居网关开发[D]. 西北工业大学.2014

[6] 李晓卉. 智能家居无线传感器网络路由[M]. 北京: 电子工业出版社，2014.3

[7] 吕雪峰. 嵌入式 Linux 软件开发从入门到精通[M]. 北京: 清华大学出版社，2014.9

[8] 邓自军.基于 433M 模组局域网协议的设计与实现[D].北京邮电大学,2012

学生学号 2013301812　　学生姓名 _____

指导教师 _____　　系 主 任 _____

摘 要

随着物联网的迅速发展，智能家居开始进入人们的视野。为了让学生宿舍更加智能化，提高学生宿舍的安全性与可管理性，本文基于物联网技术研究了智能宿舍的实现方案。

本文先介绍了开发智能宿舍系统的目的，随后，以物联网技术的三层架构：感知层，网3络层和应用层为顺序依次展示了技术实现过程。在感知层的实现中介绍了用STC15W408AS 、433M 模块及各类传感器改造的设备，比如宿舍门锁，单独设计的遥控器，温度烟雾传感器，还有遥控插线板等。网络层中为了保证服务器与各个设备之间的畅通通信，开发设计了一套在宿舍楼内的通信网络协议。该协议基于ZigBee协议进行开发，使用C语言在单片机中实现了简单的MAC接入控制、随机退避、动态路由功能。在应用层中，使用Python语言对树莓派进行编程，设计了应用层分析、处理及存储信息的后台程序，同时设计了用于显示和控制的前端页面，这是人与整个系统进行交互的直接接口。随后对智能宿舍系统的发明效果进行了展示。最后对本系统的未来设计进行了展望。

智能宿舍系统的创新点在于，基于物联网思想，将宿舍系统智能化，然后将其组网统一监控，使学生宿舍里的学生、管理人员享受到现代智能家居带来的方便与便捷它满足了高校学生对简易，时尚，高效的家居型电子产品的需求，具备良好的市场前景。

关键词：物联网，智能宿舍，自组网，树莓派，单片机

ABSTRACT

With the rapid development of Internet of things, smart home has begun to enter people's field of vision. In order to make the dormitory more intelligent and improve the safety and manageability of the dormitory, this paper studies the implementation scheme of the intelligent dormitory based on the Internet of things.

This paper first introduces the purpose of developing the smart dormitory system, and then, display the technology implementation process based on the three layers of the Internet of things technology: the perceptual layer, the network layer and the application layer. In the perceptual layer, I use STC15W408AS MCU, 433MHz wireless communication module and other sensors to make some device, such as dormitory door lock, infrared remote controller, temperature and smoke sensors, remote control socket etc. In order to ensure the smooth communication between the server and each device, a communication network protocol is designed and developed in the network layer. The protocol is developed based on ZigBee protocol. I use C language to implement simple MAC access control, random back-off and dynamic routing in single chip microcomputer. In the application layer, I use Python language program in raspberry pie, designing the background program of analyzation, processing and storage of information and front page using for display and control, which is the only interface between man and the system. Subsequently, the invention of the intelligent dormitory system was demonstrated. Finally, the future design of the system is prospected.

The innovation point of this smart dormitory is I make the dormitory system more intelligent based on IOT. I used the ad hoc network to communicate from device to device. The students and managers can enjoy the convenient of modern intelligent device. It meets the needs of college students for simple, fashionable and efficient home type electronic products, and has a good market prospect.

KEY WORDS: IOT, smart dormitory, Ad hoc networks, raspberry pi, MCU

目 录

第一章　绪论

1.1 研究背景

1.1.1 物联网

物联网的概念最早于 1999 年被提出，其最初是指物物相连的互联网形式，而在后期的不断总结与发展中，将其分为两层不同的含义：①物联网主要是指在互联网的基础上，实现对互联网的延伸以及拓展；②物联网主要是将用户延伸到了物品与物品之间，从而实现对信息的传递、交换和通信。目前，人们所接触到的物联网主要包括感知层、应用层以及网络层[1]。

感知层的主要功能是感知物联网中的“物”中的各类信息，其主要技术有：传感器技术、RFID 技术和传感器网络。

网络层的功能是通过各类型的网络（有线、无线、互联网、自组网络等）将信息进行传输，该信息即可能是来自于感知层的各种的“物”的信息，也可能是来自于应用层的信息。网络层是物联网的第二层，它起着连接感知层和应用层纽带的作用。

应用层负责将来自感知层的信息进行处理、存储、分析和应用，同时应用层也是物联网与人进行直接交互的通道。

1.1.2 智能宿舍

近年来智能家居概念深入人心并发展迅速，当前我国高等院校的学生公寓用电系统仍然以一种比较传统的粗犷方式运行并管理，这不仅造成了学生日常生活上的不方便，而且造成了大量不必要的能源浪费。

学生公寓是每个学校的用电大户，当前公寓用电虽然各个高校管理方式不尽相同，但受硬件和节能理念的制约，现有管理方式普遍存在过于粗犷的不足，造成了大量能源浪费。比如，对于管理严格的公寓，一般会采取定时拉总闸限电的措施，但是对于没有断电复位的宿舍照明系统来说，在第二天恢复供电的话会造成大量的日光灯和台灯空亮，出现不少宿舍因为粗心导致人走后日光灯空亮，造成大量电能浪费。特别是对于有空调的宿舍普遍存在炎热夏天盖着被子吹空调，后半夜不关而出现吹感冒的现象。

针对上述状况，本项目基于物联网技术提出并设计了一套开放的宿舍楼用电智能化管理系统方案，具体为:通过对宿舍内用电器接入线路加装 433M 无线遥控开关的简单改造，然后使用网页通过网络方式实现对宿舍空调、电灯、电脑、甚至门锁等用电器的开关控制。

本系统容纳了每个宿舍电器甚至非电器（门锁适当改造等），通过组网方式使得每个电器能够被精确控制，并且具有用户（学生）和管理人员（楼管）的不同优先级的多方控制功能。在这个构架下，使得科学、人性并且节能的管理方式变成了可能。因此是未来宿舍用电智能化、无线化、人性化、节能化管理的发展方向。

1.2 总体设计方案概述

1.2.1 研究内容

本设计主要研究内容为基于物联网的三层架构：感知层、网络层和应用层，设计各层软硬件架构，并制作具有基本功能的实物。

1、感知层：利用单片机及各类传感器模块，实现智能门锁、智能插座、温度及烟雾报警器等；

2、网络层实现的内容为：通过一个广播式 433MHz 通信模块实现类 ZigBee 的自组网功能；

3、应用层实现的内容为：使用树莓派，处理、存储及显示来自感知层及用户的数据消息。

1.2.2 系统架构

本系统是以 ARM 嵌入式系统、51 单片机为基础，利用无线传感器技术、信号处理与通信技术实现，系统的整体构架如图 1-1 所示。该系统按照功能结构可以分为应用层、网络层和感知层。系统应用层包含学生用户端的 Android 应用程序和楼管管理端的网页。网络层中，用树莓派开发板搭建的服务器实现宿舍内部网络和外部网络的连接和数据交换，用 STC15W408AS 和 433M 通信模块搭建的中继网络节点连接了各个感知层部分，用蓝牙和和 433M 通信模块构成的通信节点使手机控制端嵌入楼内通信网络。感知层包含各种用 STC15W408AS 和 433M 模块改造的设备，比如遥控改装锁，遥控开关（插线板），温湿度显示器等设备。

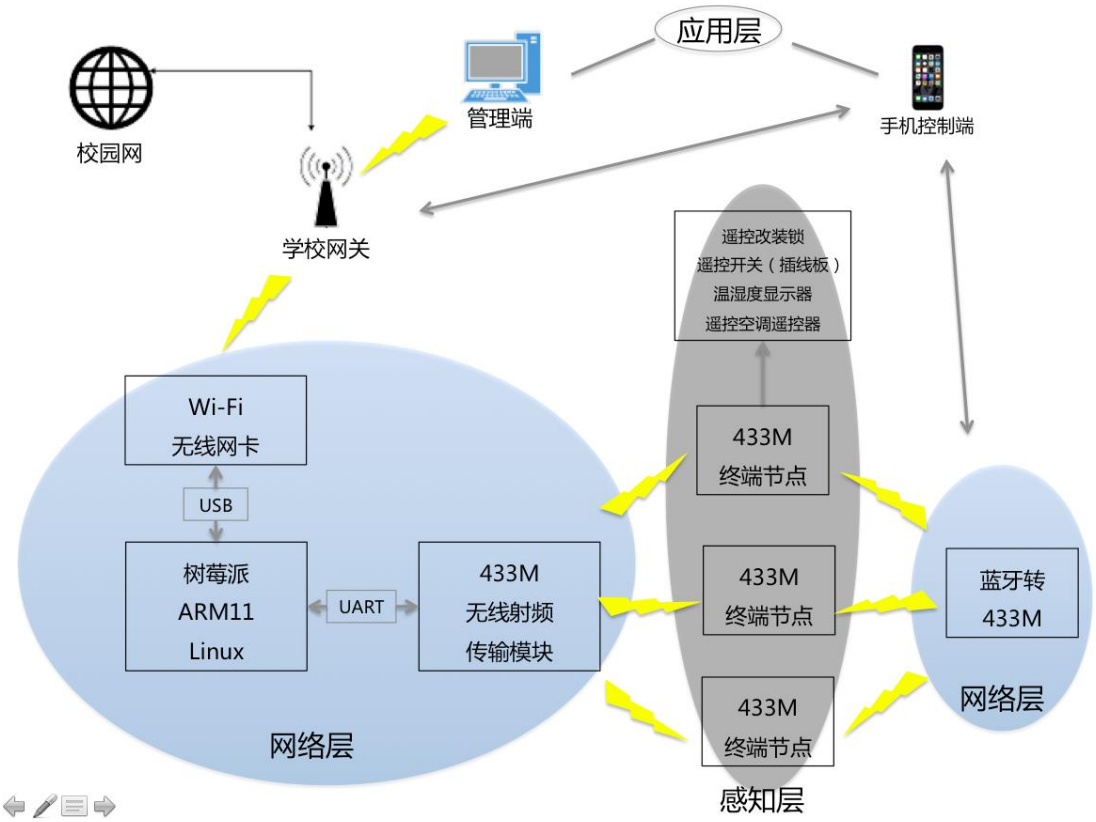

图 1-1 系统架构图

1.2.3 系统实现过程

如图 1-2 所示，系统的实现过程主要有以下部分：

首先，实现感知层各终端设备的功能，主要为单片机程序的编写。只有本部分实现了各终端的功能，在后续组网设计的时候才有系统所需信息参考，并且能够测试组网后信息传递的正确性；

其次，实现系统组网，其包括组网通信协议的设计和单片机程序设计两个方面。该部分的设计为本设计的核心部分；

再次，实现应用层功能，应用层分为后台与前端两个部分。后台负责接收来自感知层的数据并对其分析、处理和存储，前端是人与整个系统进行交互的直接接口。

最后制作功能完备的测试实物。

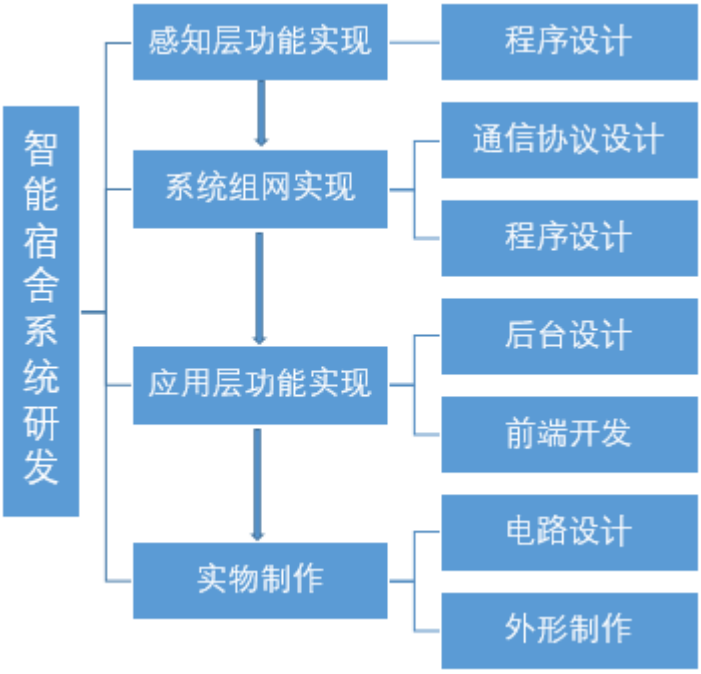
智能宿舍系统研发
感知层功能实现
程序设计
系统组网实现
通信协议设计
程序设计
应用层功能实现
后台设计
前端开发
实物制作
电路设计
外形制作

图 1-2 系统实现过程

第二章　系统组件与开发环境介绍

2.1 树莓派

2.1.1 树莓派简介

树莓派是一款基于 ARM 的微型电脑主板，以 SD/MicroSD 卡为内存硬盘，卡片主板周围有 1/2/4 个 USB 接口和一个 10/100 以太网接口（A 型没有网口），可连接键盘、鼠标和网线，同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口，以上部件全部整合在一张仅比信用卡稍大的主板上，具备所有 PC 的基本功能只需接通电视机和键盘，就能执行如电子表格、文字处理、玩游戏、播放高清视频等诸多功能。

2.1.2 树莓派开发环境搭建

1、系统安装

使用 Win32DiskImager 软件将树莓派系统镜像 rasbian 写入一张 SD 卡即制作完成树莓派系统

改 root 密码：sudo passwd root

换 root 环境：su - root

换源：sudo vi /etc/apt/sources.list 把内容换成下面两行

deb http://mirrors.cqu.edu.cn/Raspbian/raspbian jessie main contrib non-free rpi

deb-src http://mirrors.cqu.edu.cn/Raspbian/raspbian/ jessie main contrib non-free rpi

更新软件目录：sudo apt-get update

安装 vim：sudo apt-get install vim

2：树莓派系统自带 Python，但开发环境需要安装 setuptools 和 Python serial 模块

先用 FileZilla 连接到树莓派，然后解压后传文件夹 setuptools 进去，进入机器上的对应 setuptools 文件夹

python setup.py install

easy_install pyserial

查看目前插入 USB 的串口设备　　python -m serial.tools.list_ports

安装　sudo apt-get install　python-dev

sudo apt-get install libmysqlclient-dev

3：mySQL+Aapache+PHP+phpmyadmin 安装

sudo apt-get install apache2

sudo apt-get install mysql-server

sudo apt-get install php5

sudo apt-get install php5-mysql

sudo apt-get install phpmyadmin

4：nginx+php

安装 Nginx：

sudo apt install nginx

php 在上面已安装

配置 nginx

Setup nginx. nginx 的配置文件存放在/etc/nginx/sites-available 目录下。

只需要修改该目录下的 default 配置即可

cd /etc/nginx/sites-available

sudo vim default

根据提示添加 index.php,取消 php 配置前的注释。

```
# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;
# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;

        # With php5-cgi alone:
#       fastcgi_pass 127.0.0.1:9000;
```

```
        # With php5-fpm:
        fastcgi_pass unix:/var/run/php5-fpm.sock;
    }
```

5：Django 安装

需要先安装 pip

sudo apt install python-pip

安装 python3 版本的 pip

sudo apt install python3-pip

安装 Django

pip 安装方法

pip install Django

python3 安装方法

pip3 install Django

或者用 apt 安装

sudo apt install python-django

检查是否安装成功，进入 python shell 模式

```
>>> import django
>>> django.VERSION
(1, 8, 7, 'final', 0)
```

显示版本号安装成功。

2.2 51 单片机

2.2.1 51 单片机简介

单片机（Microcontrollers）是一种集成电路芯片，是采用超大规模集成电路技术把具有数据处理能力的中央处理器 CPU、随机存储器 RAM、只读存储器 ROM、多种 I/O 口和中断系统、定时器/计数器等功能（可能还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D 转换器等电路）集成到一块硅片上构成的一个小而完善的微型计算机系统，在工业控制领域广泛应用。从上

世纪 80 年代，由当时的 4 位、8 位单片机，发展到现在的 300M 的高速单片机[2]。

51 单片机是对所有兼容 Intel 8031 指令系统的单片机的统称。该系列单片机的始祖是 Intel 的 8004 单片机，后来随着 Flash rom 技术的发展，8004 单片机取得了长足的进展，成为应用最广泛的 8 位单片机之一，其代表型号是 ATMEL 公司的 AT89 系列，它广泛应用于工业测控系统之中。很多公司都有 51 系列的兼容机型推出，今后很长的一段时间内将占有大量市场。51 单片机是基础入门的一个单片机，还是应用最广泛的一种。

2.2.2 51 单片机选型

本设计中需要使用单片机作为中央处理芯片的部分有中继和终端设备。对于中继来说，处理芯片需要具备得能力为串口及一定的数据处理能力，同时需要芯片具有掉电存储空间

对于终端设备来说，因设备上有多种类型的传感器（温度传感器、电流传感器、烟雾传感器等）及外设（舵机等），需要单片机具有 A/D 转换，SPI 接口、串口、电压比较等功能。

由于上述原因，传统 8051 单片机较为吃力，所以我选用 STC 公司的增强型 8051 单片机。具体型号为：STC15W408AS-35I-SOP16。

2.2.3 STC15W408AS-35I-SOP16 介绍

STC15W408AS 系列单片机是 STC 生产的单时钟/机器周期（lT）的单片机，是宽电压/高速/高可靠/低功耗/超强抗干扰的新一代 8051 单片机，采用 STC 第九代加密技术，无法解密，指令代码完全兼容传统 8051，但速度快 8-12 倍，其主要功能有[3]：

内部集成高精度 R/C 时钟（±0.3%），±0.1%温飘（-40 ℃～＋85 ℃），时钟频率范围为：5MHz~35MHz，相当于普通 8051 的 60MHz~420MHz；

8K 字节片内 Flash 程序存储器，擦写次数 10 万次以上；

片内集成 S l2 字节的 S RAM ， 包括常规的 2S6 字节 RAM <idata＞和内部扩展的 2S6 字节 XRAM <xdata>；

有片内 EEPROM 功能，5K 空间大小，擦写次数 10 万次以上；

共 8 通道 10 位高速 ADC，速度可达 30 万次 / 秒；

共 3 通道捕获 / 比较单元（CCP/PWM/PCA)，也可用来再实现 3 个定时器或 3 个外部中断（支持上升沿 / 下降沿中断）；

一组高速同步串行通信端口 SPI；

图 2-1 显示了 STC15W408AS 单片机每个引脚的详细功能及其定义。

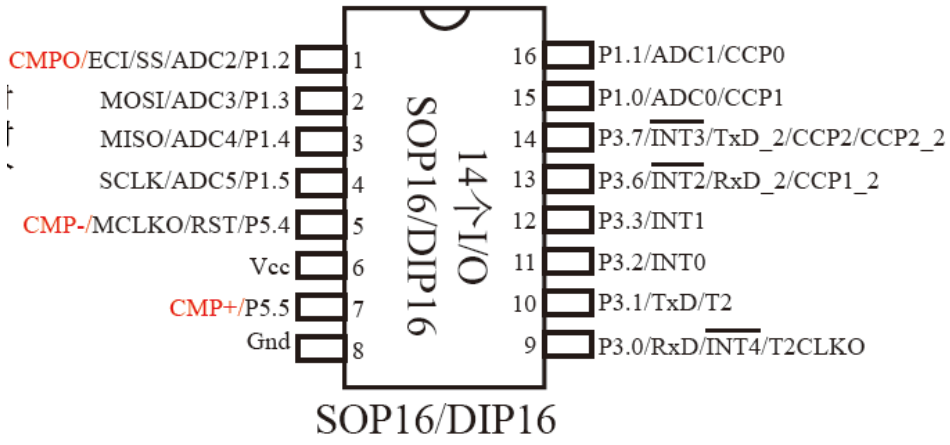

图 2-1 STC15W408AS-35I-SOP16 引脚图

2.2.4 STC15W408AS 单片机开发环境及烧录

STC15W408AS 单片机与传统 51 单片机的开发环境相同，使用 keil C51 程序进行开发。开发语言为 C 语言或者汇编语言。

51 单片机使用的 C 语言（C51）在基本语法上与 C 语言基本相同，但仍有一些不同，如：

1、C51 需要定义寄存器地址，使用 sfr 语句，如定义电源控制寄存器 PCON，地址为 0x87：sfr PCON = 0x87；

2、sbit 位类型符用于定义在可位寻址字节或特殊功能寄存器中的位，如定义 P4 口 0 位，sbit P1_0=0X91（按地址）或 sbit P1_0=P1^0（按寄存器）；

3、C51 编程中主函数形式固定为 void main()，其不应有返回值和参数，且主函数一般有一死循环 while(1)，使单片机处于一直运行状态。

因 STC15W408AS 单片机为 STC 公司研发生产的增强型 8051 单片机，其兼容传统 8051 单片机的寄存器与 I/O 口设置，但其具有许多传统 8051 不具有的接口、功能及特殊功能寄存器（如 P4、P5 口，定时器/计数器 2、3、4、5，A/D 等），所以在开发过程中需要定义新的寄存器，本设计中使用到的特殊功能寄存器及 I/O 口设置如下：

```
//I/O 口特殊功能寄存器
sfr P5          =   0xC8;   //端口 5 地址
//ADC 特殊功能寄存器
sfr ADC_CONTR   =   0xBC;   //A/D 转换控制寄存器
sfr ADC_RES     =   0xBD;   //A/D 转换结果高 8 位
sfr ADC_RESL    =   0xBE;   //A/D 转换结果低 2 位
sfr P1ASF       =   0x9D;   //端口 1 模拟功能配置寄存器
//CCP/PAC 寄存器
```

```
sfr CCON        =   0xD8;   //PCA 控制寄存器
sbit CF         =   CCON^7;
sbit CR         =   CCON^6;
sbit CCF2       =   CCON^2;
sbit CCF1       =   CCON^1;
sbit CCF0       =   CCON^0;
sfr CMOD        =   0xD9;   //PCA 工作模式寄存器
sfr CL          =   0xE9;   //PCA 计数器低字节
sfr CH          =   0xF9;   //PCA 计数器高字节
sfr CCAPM0      =   0xDA;   //PCA 模块 0 的 PWM 寄存器
sfr CCAPM1      =   0xDB;   //PCA 模块 1 的 PWM 寄存器
sfr CCAPM2      =   0xDC;   //PCA 模块 2 的 PWM 寄存器
sfr CCAP0L      =   0xEA;   //PCA 模块 0 的捕捉/比较寄存器低字节
sfr CCAP1L      =   0xEB;   //PCA 模块 1 的捕捉/比较寄存器低字节
sfr CCAP2L      =   0xEC;   //PCA 模块 2 的捕捉/比较寄存器低字节
sfr CCAP0H      =   0xFA;   //PCA 模块 0 的捕捉/比较寄存器高字节
sfr CCAP1H      =   0xFB;   //PCA 模块 1 的捕捉/比较寄存器高字节
sfr CCAP2H      =   0xFC;   //PCA 模块 2 的捕捉/比较寄存器高字节
sfr AUXR        =   0x8E;   //辅助寄存器
//定时器 2 相关寄存器
sfr AUXR        =   0x8E;   //辅助寄存器，用于定时器 2 控制
sfr T2H         =   0xD6;   //T2 高字节
sfr T2L         =   0xD7;   //T2 低字节
sfr IE2         =   0xAF;   //中断控制寄存器 2，用于定时器 2 中断
//串口 1 特殊功能寄存器
sfr AUXR1       =   0xA2;   //辅助寄存器 1，用于串口切换
//IAP/EEPROM 存储空间操作寄存器
sfr IAP_DATA    =   0xC2;   //EEPROM 数据寄存器
sfr IAP_ADDRH   =   0xC3;   //EEPROM 地址高字节
sfr IAP_ADDRL   =   0xC4;   //EEPROM 地址第字节
sfr IAP_CMD     =   0xC5;   //EEPROM 命令寄存器
sfr IAP_TRIG    =   0xC6;   //EEPRPM 命令触发寄存器
sfr IAP_CONTR   =   0xC7;   //EEPROM 控制寄存器
```

这些寄存器定义需写在程序之中才能正常使用单片机。或者使用 STC 公司提供的适合该单片机的头文件。

单片机程序写好后，需要将其编译为 HEX 文件，才能将其烧录进单片机，烧录软件为 STC 公司提供的 STCISP。STC 公司生产的单片机在烧录时仅需使用四个引脚：VCC、GND、TXD、RXD，若使用 USPISP 模块，仅需将 VCC、GND 接在对应引脚，TXD 与 RXD 跟 USPISP 的 TXD 与 RXD 反接即可。

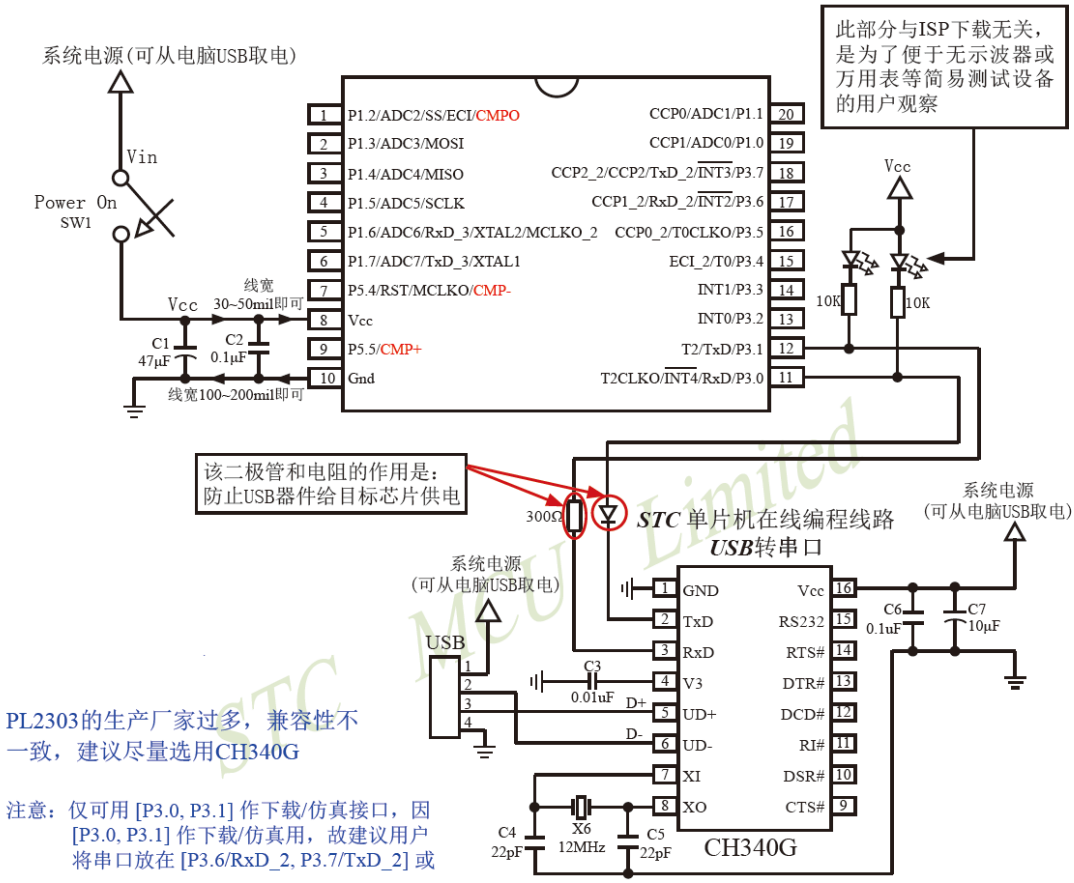

图 2-2 STC15W408AS 单片机烧录电路示意图

2.3 单片机主要外设

2.3.1 HC-11 433MHz 通信模块

HC-11 433MHz 无线通信模块使用 CC1101 无线数传芯片，其工作频段为 434MHz~437MHz。该通信模块为广播式模块，一个模块发出的消息，所有相同模式的该模块均能接受到该消息。该模块使用串口作为数据接口，单片机等上位机可以使用串口简单的接收及发送数据。

模块有四种串口透传模式，FU1（默认）向下兼容，FU2 空闲电流仅 80uA，FU5 是 FU1 的优化，FU6 传送距离优化。

用户无须对模块编程，四种模式都是只需收发串口数据即可，使用方便。

不限一次往模块串口发送的字节个数。

各种功能和参数通过指令更改，掉电保存。

2.3.2 MG90S 舵机

MG90S 舵机为 41g 金属齿舵机，其尺寸与 MG90 9g 舵机一致。其详细参数有：

产品净重: 13.4g

产品尺寸: 22.8*12.2*28.5mm

产品扭矩: 1.8kg/cm(4.8v) 2.2kg/cm(6.0v)

反应速度: 0.10sec/60degree(4.8v) 0.08sec/60degree(6.0v)

工作电压: 4.8V-6V

使用温度: 0-55 度

动作死区: 5us

齿轮介质: 金属

2.3.3 DS18B20 温度传感器

DS18B20 是常用的温度传感器，具有体积小，硬件开销低，抗干扰能力强，精度高的特点。DS18B20 采用 IIC 通信协议，但其为单线通信。其主要特性有：

适应电压范围更宽，电压范围：3.0～5.5V，在寄生电源方式下可由数 据线供电

独特的单线接口方式，DS18B20 在与微处理器连接时仅需要一条口线即可实现微处理器与 DS18B20 的双向通讯

DS18B20 支持多点组网功能，多个 DS18B20 可以并联在唯一的三线上，实现组网多点测温

DS18B20 在使用中不需要任何外围元件，全部 传感元件及转换电路集成在形如一只三极管的集成电路内

温范围－55℃～+125℃，在-10～+85℃时精度为±0.5℃

可编程 的分辨率为 9～12 位，对应的可分辨温度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃，可实现高精度测温

在 9 位分辨率时最多在 93.75ms 内把温度转换为数字，12 位分辨率时最多在 750ms 内把温度值转换为数字，速度更快

测量结果直接输出数字温度信号，以"一 线总线"串行传送给 CPU，同时可传送 CRC 校验码，具有极强的抗干扰纠错能力

负压特性：电源极性接反时，芯片不会因发热而烧毁， 但不能正常工作。

2.3.4 MQ-2 烟雾传感器模块

MQ-2 烟雾传感器液化气、丁烷、丙烷、甲烷、烟雾等的探测，其将环境中的可燃气体浓度或者烟雾浓度转换为电压信号并输出，通过单片机 A/D 转换后可得出烟雾浓度。

2.3.5 ACS712 电流检测模块

ACS712 是基于霍尔检测原理的电流检测芯片，该芯完全基于霍尔感应的原理设计，由一个精确的低偏移线性霍尔传感器电路与位于接近 IC 表面的铜箔组成（如下图所示），电流流过铜箔时，产生一个磁场, 霍尔元件根据磁场感应出一个线性的电压信号，经过内部的放大、滤波、斩波与修正电路，输出一个电压信号，该信号从芯片的第七脚输出，直接反应出流经铜箔电流的大小。ACS712 根据尾缀的不一样，量程分为三个规格：±5A、±20A、±30A 。输入与输出在量程范围内为良好的线性关系，其系数 Sensitivity 分别为，185 mV/A、100 mV/A、66mV/A。因为斩波电路的原因，其输出将加载于 0.5*Vcc 上。ACS712 的 Vcc 电源 一般建议采用 5V。输出与输入的关系为 Vout=0.5Vcc+Ip*Sensitivity。一般输出的电压信号介于 0.5V~4.5V 之间。

本系统使用的是 20A 量程检测模块。

2.3.6 LCD12864 显示器

LCD12864 显示器是一种具有 4 位/8 位并行、2 线或 3 线串行多种接口方式，内部含有国标一级、二级简体中文字库的点阵图形液晶显示模块。其显示分辨率为 128×64，内置 8192 个 16×16 点汉字，和 128 个 16×8 点 ASCII 字符集。利用该模块灵活的接口方式和简单、方便的操作指令，可构成全中文人机交互图形界面。可以显示 8×4 行 16×16 点阵的汉字。也可完成图形显示。低电压低功耗是其又一显著特点。由该模块构成的液晶显示方案与同类型的图形点阵液晶显示模块相比，不论硬件电路结构或显示程序都要简洁得多，且该模块的价格也略低于相同点阵的图形液晶模块。

基本特性：

低电源电压（VDD：+3 .0~＋5. 5V）

显示分辨率：128×64 点

内置汉字字库，提供 8192 个 16×16 点阵汉字（简繁体可选）

内置 128 个 16×8 点阵字符

2MHz 时钟频率

显示方式：STN、半透、正显

驱动方式： l/32DUTY，l/5BIAS

视角方向：6 点

背光方式：侧部高亮白色 LED，功耗仅为普通 LED 的 1/5 一 1/10

通讯方式：串行、并口可选

内置 DC-DC 转换电路，无需外加负压

无需片边信号，简化程序设计

工作温度：0℃ 一 +55℃

2.3.7 抗浪涌继电器

继电器是使用低电压控制高电压的典型器件，在智能插座中常有使用。抗浪涌继电器具有抗浪涌的功能，使用在插座中后能使插座拥有放雷击的作用。

第三章　系统实现

3.1 网络层

网络层的设计为本系统的核心部分，

本系统参考 ZigBee 自组网协议设计一种新型的无线自组网通信协议，该通信协议简单明了、配置灵活、方便测试，可以在各种环境下传输信息。该协议创建了一种蜂窝状网络，实现点对点的信息传输，经过实践操作，测试得其运行稳定，模块化好，达到了设计的目的[4]。

3.1.1 设计方案

系统整体框架图如图 3-1 所示，

图 3-1 自组网系统

本系统基于物联网，使用类 ZigBee 的方式自建组网实现上下行的数据传输。整个系统自下而上结构为设备，中继，中控三层结构。

3.1.2 通信协议设计

如表 3-1 所示，传输数据由十三位字节组成，其中第一位和第十三位作为起始和终止的标志位，定义起始位为 0xF1 表示正常上行，0xF2 表示正常下行，0xF3 表示组网请求上行，0xF4 表示组网请求下行，0xF5 表示测试指令，对应的终止位为其起始位的最后一位取反，分别为 0xFE、0XFD、0xFC、0xFB、0xFA；第二位第三位共同表示信号发出的设备标识号，第四位第五位共同表示数据最初来源的设备表示号；第六位表示操作数；第七位是一个计数位；第八、九、十、十一、十二位表示中继 ID，这五位初始值为 F6，自组网成功后其从左至右的顺序表示其在系统中传输数据的路径。3

表 3-1 数据格式

信息类型	中继号	设备号	数据位	指针位	路由表	停止位
1 Byte (data[0])	2 Byte (data[1:2])	2 Byte (data[3:4])	1 Byte (data[5])	1 Byte (data[6])	5 Byte (data[7:11])	1 Byte (data[12])

每次正常上行工作可由设备发出十三位十六进制的数据，一级中继接收到这个数据后首先会判断这是否是上行消息且是否为自己处理，如果是则进行修改再重新发出，否则自动忽略这条信息，二级中继接收到消息后所做出的事和一级中继是一样的，判断改编后再发出，最后是树莓派，接收到数据判断后将信息写入数据库中，然后根据数据库内容显示在专门的网页界面上，使用者便可看到设备进行了什么操作变化。

正常下行数据传输时，首先使用者可通过网页中的按键选项对设备进行一种操作，此时这种携带着操作指令的九位十六进制的消息将从树莓派这一级传至二级中继。二级中继判断其是否为下行消息且是否为自己处理，如果是，则对这段数据进行改编再传至下一级，否则自动忽略这条消息，一级中继接收到这个消息后也和二级中继做一样的判断并处理数据，传输到设备这一级时，设备根据发送来的指令做出相应的动作并根据需求返回数据。

为了保障数据传输的稳定性，我们采取了退避算法，当数据传向下一级后，若下一级收到这个信息，自动向上一级发出确认收到的回复，上一级在设定时间内收到特定的回复，则停止发送消息，若超出设定时间未收到特定回复，此时表示下一级未收到消息，上一级也将停止发送消息，在一段时间内选择任意时间退避等待，再重新发送消息，等待回复，若仍未收到回复，继续退避等待，以此类推。

3.1.3 具体实施方案

3.1.3.1 自组网实例：

当一个中继（假设 ID 为‘0’‘1’）要自组网时，此时操作数应为 0xF3 即发送组网请求，计数位为初始值 0x01，他将发送消息：

0xF3 0x30 0x31 0x30 0x31 0xF3 0x01 0xF6 0xF6 0xF6 0xF6 0xF6 0xFC

当中控收到这个组网消息时，初始位与终止位满足对应关系是有效数据，同时返回给中继收到消息的通知：0x30 0x31 0x46 0x46，中继收到此消息时发现是自己的 ID 并以 0x46 0x46 结尾，则停止发送组网消息，中控通过 0xF3 发现是申请加入组网的请求，判别计数位为 0x01 初始值则此中继未加入组网，中控将其中继 ID 的第二位 0x31 写入数据第八位即路由表的第二位，将计数位改为 0x06，再把这个传输路径写入数据库，同时中控将给中继返回组网数据，操作数 0xF6 表示组网成功，下行数据计数位加 1，则消息为：

0xF4 0x46 0x46 0x30 0x31 0xF6 0x07 0x31 0xF6 0xF6 0xF6 0xF6 0xFB

当中继收到这个组网消息时，初始位与终止位满足对应关系是有效数据，同时返回给中继收到消息的通知：0xFF 0xFF 0x46 0x46，中控收到此消息时发现是自己的 ID 并以 0x46 0x46 结尾，则停止发送组网消息，中继通过 0xF6 发

现是组网成功的请求，判断计数位的值与其指向数据数组的值相等，则将路由表保留到自己的数据里，以后发送消息的七到十二位为：

0x07 0x31 0xF6 0xF6 0xF6 0xF6

以上即为中继与中控上行自组网的全部数据传输过程。相似的可以得到设备，中继，中控三者的组网过程，如图 3-2 所示。

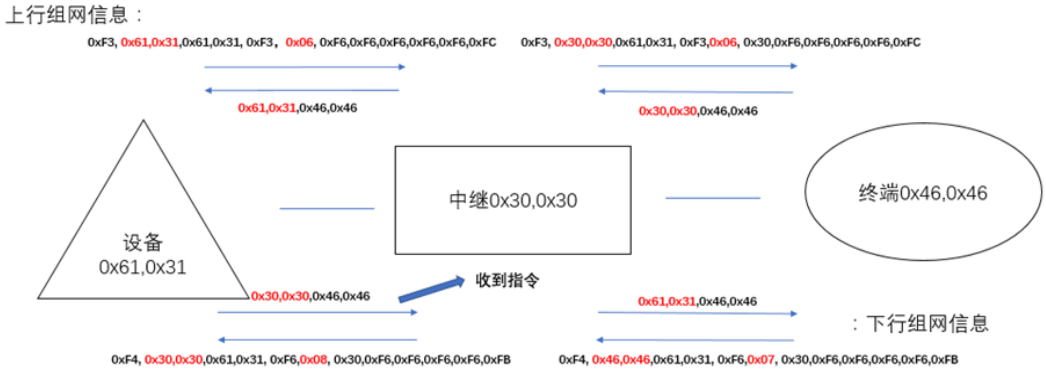

图 3-2 自组网实例

3.1.3.2 数据传输实例

现假设已经存在了一条数据传输网络中控（id 为’F’ ‘F’）、中继（ID 为‘0’ ‘0’）和设备（ID 为‘a’ ‘1’）。

首先具体描述一般信息下行，即信息由设备产生发出经由中继到中控的过程。

首先是设备 0x61,0x31 向中继发送指令：当起始位为 0xF1 时，设备请求上行，请求由‘a’ ‘1’发出，则第二、三位按其编号顺序分别为 0x61,0x31；原始请求由设备提出，则第四、五位分别对应设备编号的两位：0x61,0x31；第六位为指令位，这里假设为 0x01；第七位指针位的 P 本为 8，上行左移一位变为 7；第 8、9、10、11、12 位分别为已组网成功的目标路由表，即 0x30,0xF6,0xF6,0xF6,0xF6；终止位为 0xFE。

则该设备发出来的整条数据串即为：

0xF1,0x61,0x31，0x61,0x31,0x01,0x07,0x30,0xF6,0xF6,0xF6,0xF6,0xFE

数据发送出去后，设备会在一个时间间隔内等待直到收到 0x61,0x31,0x46,0x46 的下级（即中继 0x30,0x30）回复，否则重发，或者重新申请组网。

中继 0x30,0x30 收到上条数据串时，通过识别起始位为 0xF1，和指针位所指向的 a[7]路由表第一位 0x30,为自己，然后会进向相关处理，即向上级回复收到指令和向下级发送指令。具体相关处理为：（1）将第二、三位 0x61,0x31 添加在 0x46,0x46 前，将其发出，即为回复收到指令。（2）将第二、三位

0x61,0x31 替换为自己的设备号 0x30,0x30，再将指针位减 1 变为 6，然后将其发送出去，即为向下级发送指令。

则该设备发出来的整条数据串即为：

0xF1,0x30,0x30,0x61,0x31,0x01,0x06,0x30,0xF6,0xF6,0xF6,0xF6,0xFE。

数据发送出去后，设备会在一个时间间隔内等待直到收到 0x30,0x30,0x46,0x46 的下级（即中控 0x46,0x46）回复，否则重发，或者重新申请组网。

中控 0x46,0x46 收到上条数据串时，通过识别起始位为 0xF1，和指针位 P 为 0x06,为自己要处理的信息，然后会进向相关处理，和向上级回复收到指令 0x30,0x30,x46,0x46。

以上即为一般上行全部数据传输过程。相似的可以得到一般的信息下行过程，如图 3-3 所示。

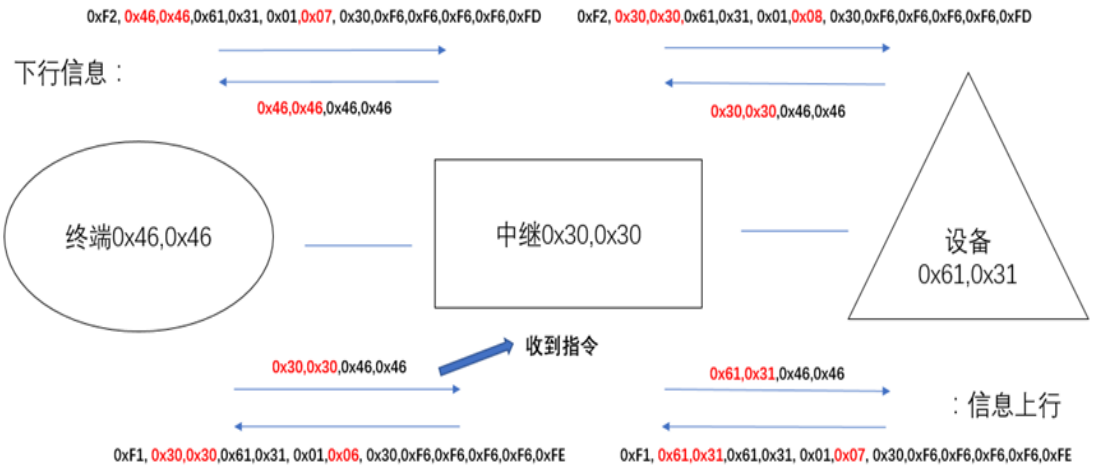

图 3-3 数据传输实例

3.1.4 程序设计

如图 3-4 所示为网络层中继程序流程图。中继上电时及程序开始时，第一步先判断本中继是否已经在路由表存在，若没有路由表存在则发送一串组网指令进行组网。若存在路由表，则程序进入主循环。程序在主循环中不断判断是否有行为标志，若没有则继续判断，若有行为标志，则根据行为标志的不同进行相应操作（上行、下行、其他设备的组网请求）。在进行上下行操作时，中继有可能会遇到数据冲突，程序中采取随机退避的方法减小冲突概率。若本次操作的 16 次重传均失败，则系统进行重新组网。

程序使用串口接收或发送 433MHz 无线通信模块的数据。在接收数据时会产生中断，保证中继能够及时接受到来自他人的数据。中继首先判断该数据是否为属于本中继的数据，若不是则将该数据包直接扔掉。若是则进行下一步操作。中

断函数中的主要操作为判断接收到的数据是那种操作行为（上行、下行、其他设备的组网请求），并将行为标志位置相应值。

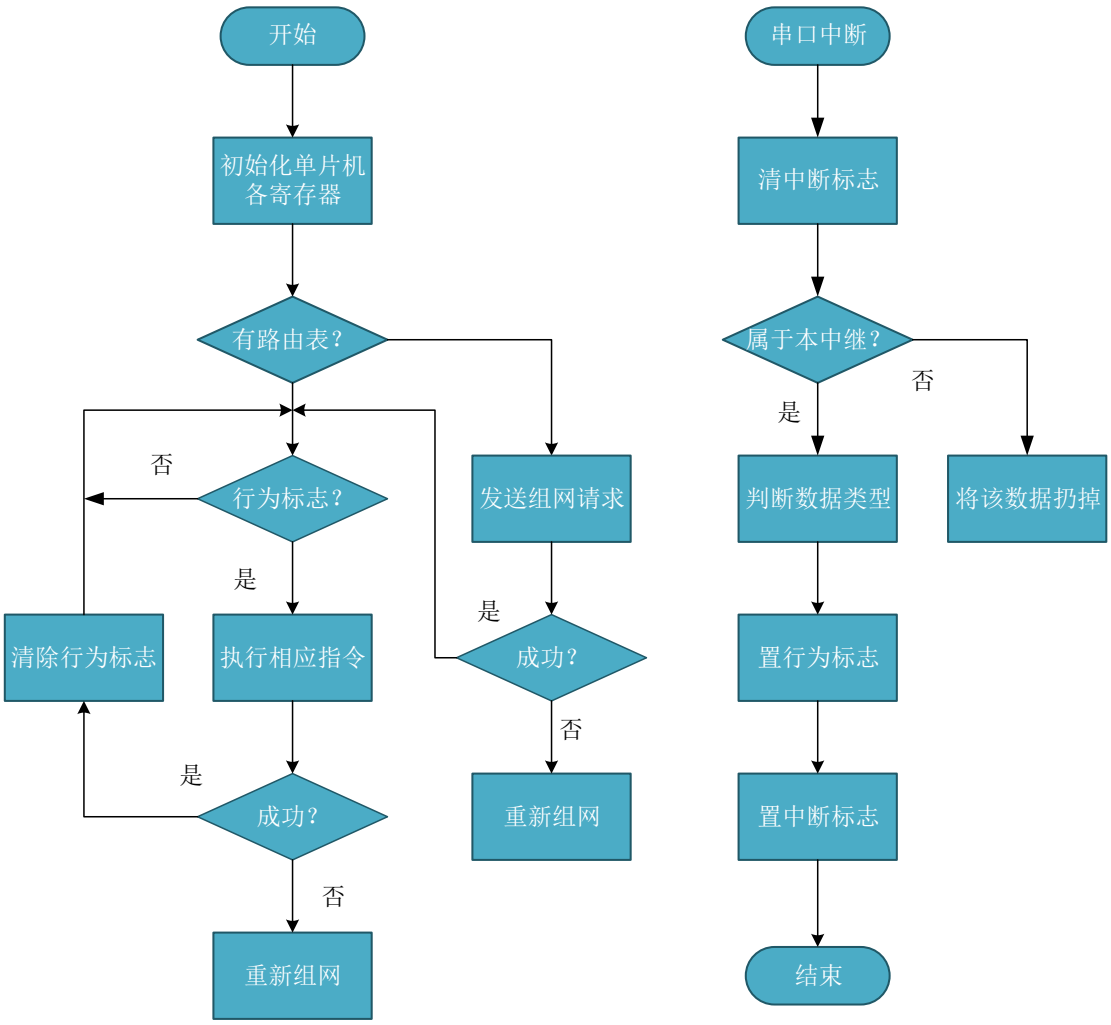

图 3-4 网络层中继程序流程图

如图 3-5 所示为终端设备程序流程，该流程中的状态传感器指温度传感器、烟雾传感器和电流传感器，定时器中断的作用为控制传感器检测时间间隔，如每 1s 检测一次状态，若状态达到报警值则发送报警信息。所以，在四个终端中仅有温度及烟雾报警器和插座有该中断。门锁和红外遥控仅有接收控制信号后才进行动作并反馈自身状态。

图 3-5 中包含网络层在终端设备中的程序流程，其同样使用串口中断进行数据的接收和前期处理，串口中断流程同图 3-4 中的串口中断流程。网络层在终端设备中的行为与中继中的行为有不同之处。网络成中继的主要任务为上下行组网的连接与数据的中间传递，而终端设备不中转数据。

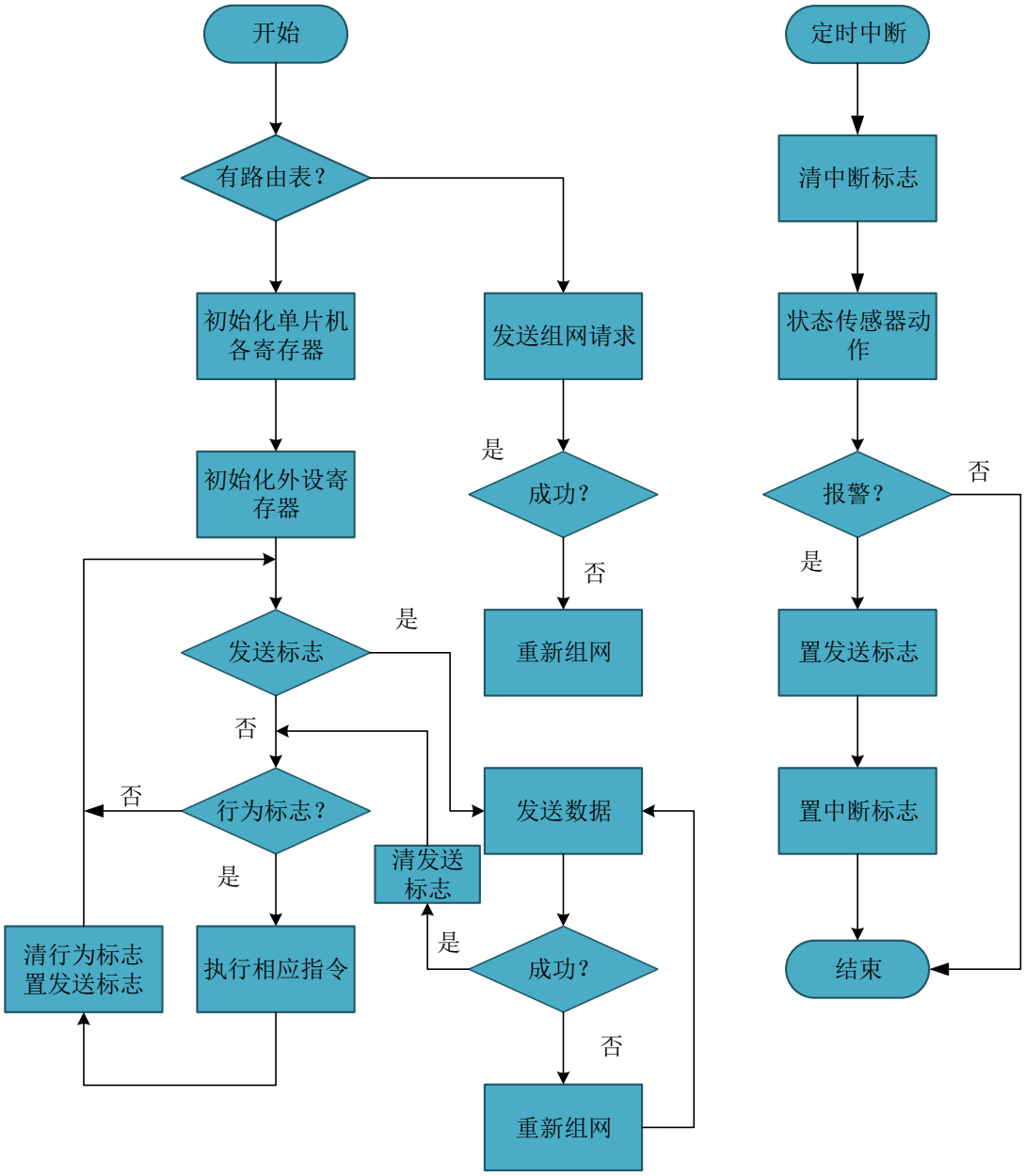

图 3-5 设备程序流程

对树莓派而言，其在网络层中的主要工作为接收来自终端的所有数据、发送对终端的控制指令和记录到达终端设备的所有路由。

3.2 感知层

在本系统中，感知层即是各终端设备，包括智能门锁、智能插座、温度及烟雾报警器和红外遥控器。

3.2.1 硬件设计

感知层中的歌终端设备使用单片机加各类传感器、电子模块搭建。所有终端设备使用的主控芯片均为 STC15W408AS，综合考虑个电子模块所需电路设计单片机最小系统。所需电路及接口有：

电源模块（包括供电接口电源指示灯）

一路组网状态 led 指示灯

一路 PWM 输出（通过 I/O 口输出，舵机控制）

一路 ISP 程序烧录接口

一路异步串行通讯接口（433MHz 通信模块）

一路红外发射接口电路（需有放大电路）

全引脚引出共其余各模块连接。

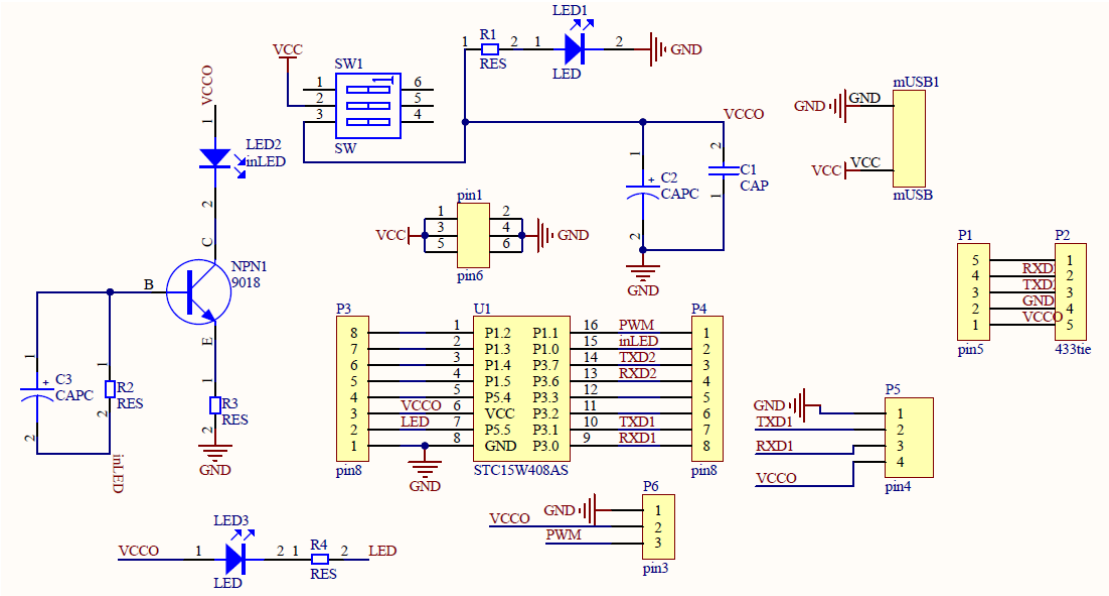

图 3-6 STC15W408AS 最小系统板原理图

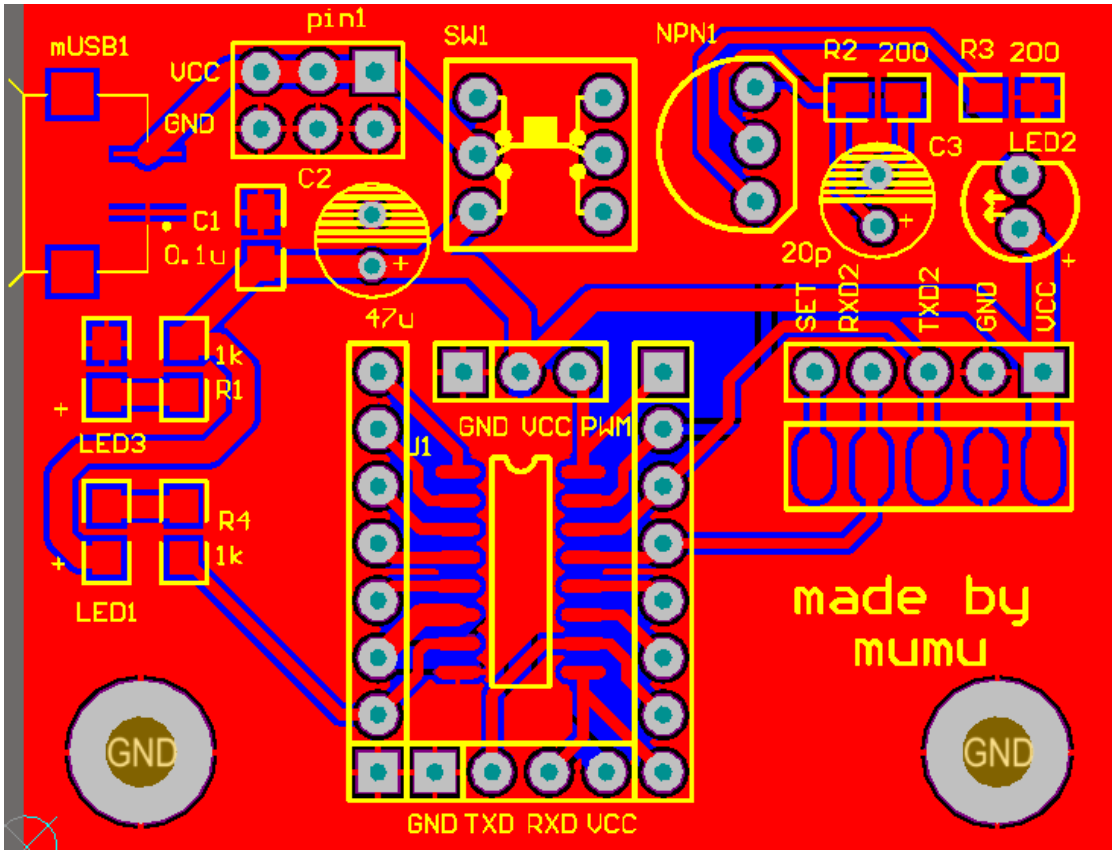

图 3-7 STC15W408AS 最小系统板 PCB

制板后测试板子各模块功能正常。

在智能门所得制作过程中，还需要设计舵机的传动结构，我利用 SolidWorks 程序进行了三维设计，并使用 3D 打印机将其制作了出来。

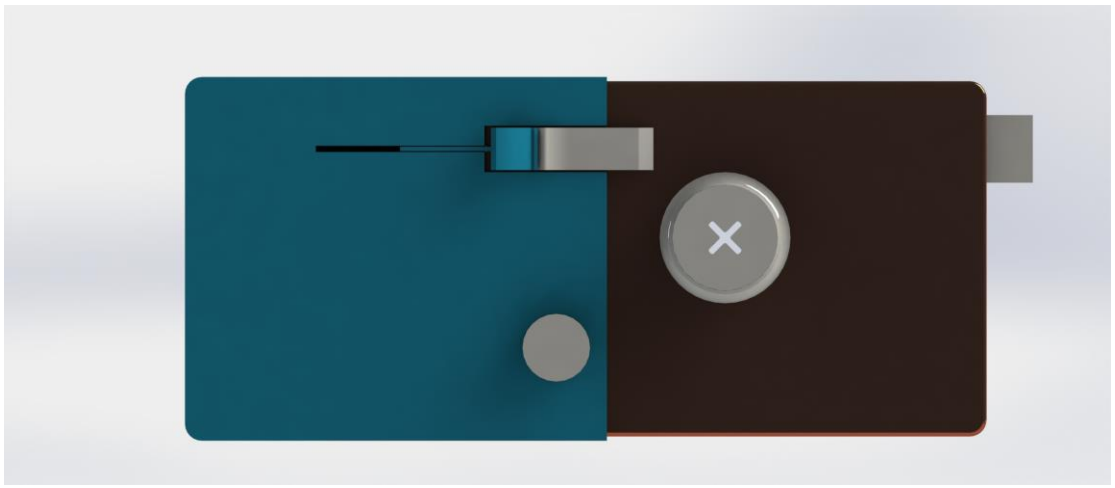

图 3-8 智能门锁渲染图

3.2.2 程序设计

本系统每个设备（终端，中继，中控）均包括 433MHz 无线通信模块，关于 433MHz 无线组网的程序设计在 3.1.4 做介绍，本节程序设计均不涉及组网程序设计。

本系统设计过程中单片机时钟选用 18.432MHz。定时器使用 12T 模式，串口波特率均为 115200。

单片机程序设计需要协调各外设之间的工作，有些外设响应不需要很严格的时序控制，如 LED，仅需控制高低电平的输出便能控制其亮灭，而有些外设需要精确的时序控制和及时响应，如 SPI 接口和串口数据的发送与接收。这使得单片机中的定时/计数器和中断系统尤为重要。

在本系统的设计过程中大量应用了定时/计数器和中断系统。

3.2.2.1 智能门锁程序设计

智能门锁主要是利用单片机对舵机进行控制。

舵机控制是使用 PWM（脉冲宽度调制）波进行控制的，其控制周期为 20ms，高电平时间 0.5ms-2.5ms 之间。当高电平时间为 0.5ms 时，舵机转动 0 度，当高电平时间为 1.5ms 时，舵机转动 90 度，当高电平时间为 2.5ms 时，舵机转动 180 度，一般舵机最大转动角度为 180 度，其余角度所需高电平时间可以整除得到。

使用单片机产生精确周期的及占空比需要使用单片机定时/计数器及其中断。51 单片机的定时/计数器为加式溢出中断定时器，即定时器初始化后每过一个机器周期（12T 定时器为 12 个时钟周期）定时器加 1，当定时器计满（根据定时器位数 n，计满数时为 2^n）时会将标志位置 1，若开启中断则会产生中断。51 单片机的 16 位定时器最多可计数 2^{16} 即 65536 个数，对于 18.432MHz 时钟 12T 定时器，则最多可定时：

$$\frac{1}{18.432\times10^6}\times 12\times 65536 = 0.0042667s = 42667us = 42.667ms$$

通过设置定时器初值，使计数个数不同产生不同时间的中断。

由上述可知，定时器一次计数可满足 20ms 周期要求，但为使单一定时器能够复用，一般不采用定时 18.5ms 使引脚输出低电平，再定时 1.5ms 使引脚输出高电平（以舵机转动 90 度为例）方案，而采用较小定时间隔（如本系统采用 0.125ms 定时），然后计数中断次数，使用定时器的不同设备使用不同技术变量，达到单一定时器复用的目的。

在本系统中，STC15W408AS 仅有定时器 0 和定时器 2，没有定时器 1，定时器 0 用作组网计时，定时器 2 用作波特率发生器，则使用 PCA 作为定时器。

```
//初始化 PCA
void PCA_init()
{
```

```
    CCON=0;

    CL=0;
    CH=0;
    CMOD=0x00;

    value=THz;
    CCAP0L=value;
    CCAP0H=value>>8;
    value+=THz;
    CCAPM0=0x49;

    CR=1;
    EA=1;
}
//PCA 用作定时器，PCA 中断函数
void PCA_isr() interrupt 7 using 2
{
    CCF0=0;
    CCAP0L=0x40;
    CCAP0H=0xff;
    value += THz;
    if(count< jd)
        pwm=1;
    else
        pwm=0;
    count++;
    if(count==160) count=0;
}
```

其中，count 为定时器中断次数计数变量，每产生一次中断，count 加 1，count 为 160 时，即定时 160*0.125=20ms，count 清零。jd 变量计数高电平时间，若舵机转动 90 度，则 jd=12。

3.2.2.2 智能插座程序设计。

智能插座需要对继电器与电流检测模块进行操作。

继电器仅需通过 I/O 口输出高电平（有的继电器为低电平）即可使继电器接通，输出低电平即可使继电器断开。

ACS712 电流检测模块输出为模拟电压，需通过 A/D 转换读出模拟电压值。并根据公式$V_{out} = 0.5 \times V_{cc} + I_p * \text{Sensitivity}$得$I_p = \frac{V_{out} - 0.5 \times V_{cc}}{\text{Sensitivity}}$测得电流。

3.2.2.3 温度及烟雾报警器

温度及烟雾报警器主要三个模块：18B20、MQ-2 烟雾传感器、LCD12864 显示模块。

MQ-2 烟雾传感器输出为模拟电压，使用 A/D 转化并通过公式进一步得到烟雾浓度。关于 A/D 的程序设计见 3.2.2.2 介绍。

18B20 为单总线式器件，其程序设计需按照一定的时序关系。18B20 的主要时序有初始化时序，写时序和读时序。

如图 3-9 所示，主机首先发出一个 480－960 微秒的低电平脉冲，然后释放总线变为高电平，并在随后的 480 微秒时间内对总线进行检测，如果有低电平出现说明总线上有器件已做出应答。若无低电平出现一直都是高电平说明总线上无器件应答。

作为从器件的 DS18B20 在一上电后就一直在检测总线上是否有 480－960 微秒的低电平出现，如果有，在总线转为高电平后等待 15－60 微秒后将总线电平拉低 60－240 微秒做出响应存在脉冲，告诉主机本器件已做好准备。若没有检测到就一直在检测等待。

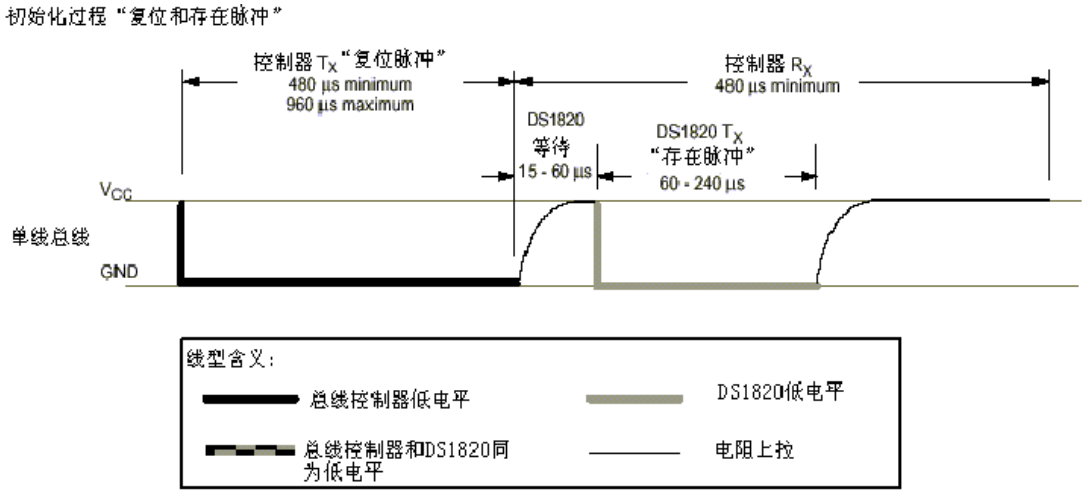

图 3-9 18B20 初始化时序

如图 3-10 所示，写周期最少为 60 微秒，最长不超过 120 微秒。写周期一开始作为主机先把总线拉低 1 微秒表示写周期开始。随后若主机想写 0，则继续拉低电平最少 60 微秒直至写周期结束，然后释放总线为高电平。若主机想写 1，在一开始拉低总线电平 1 微秒后就释放总线为高电平，一直到写周期结束。而作为从机的 DS18B20 则在检测到总线被拉底后等待 15 微秒然后从 15us 到 45us 开始对总线采样，在采样期内总线为高电平则为 1，若采样期内总线为低电平则为 0。

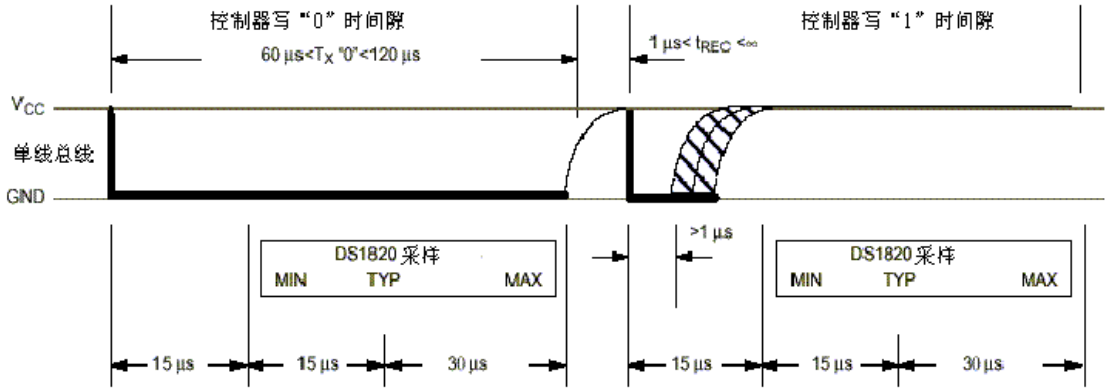

图 3-10 18B20 写时序

对于读数据操作时序也分为读 0 时序和读 1 时序两个过程。如图 3-11 所示读时隙是从主机把单总线拉低之后，在 1 微秒之后就得释放单总线为高电平，以让 DS18B20 把数据传输到单总线上。DS18B20 在检测到总线被拉低 1 微秒后，便开始送出数据，若是要送出 0 就把总线拉为低电平直到读周期结束。若要送出 1 则释放总线为高电平。主机在一开始拉低总线 1 微秒后释放总线，然后在包括前面的拉低总线电平 1 微秒在内的 15 微秒时间内完成对总线进行采样检测，采样期内总线为低电平则确认为 0。采样期内总线为高电平则确认为 1。完成一个读时序过程，至少需要 60us 才能完成

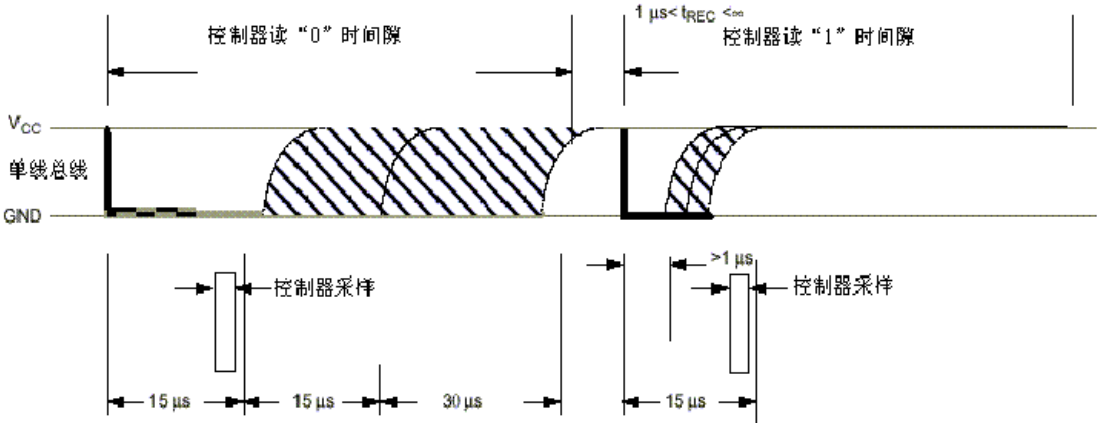

图 3-11 18B20 读时序

完成一次温度转化需按一下步骤：

1. 对 18B20 进行复位初始化
2. 写一条跳过 ROM（0xCC）指令
3. 写一条温度转换指令

LCD12864 显示屏使用串行接口进行数据交互，单片机与其通讯也需遵照一定时序。

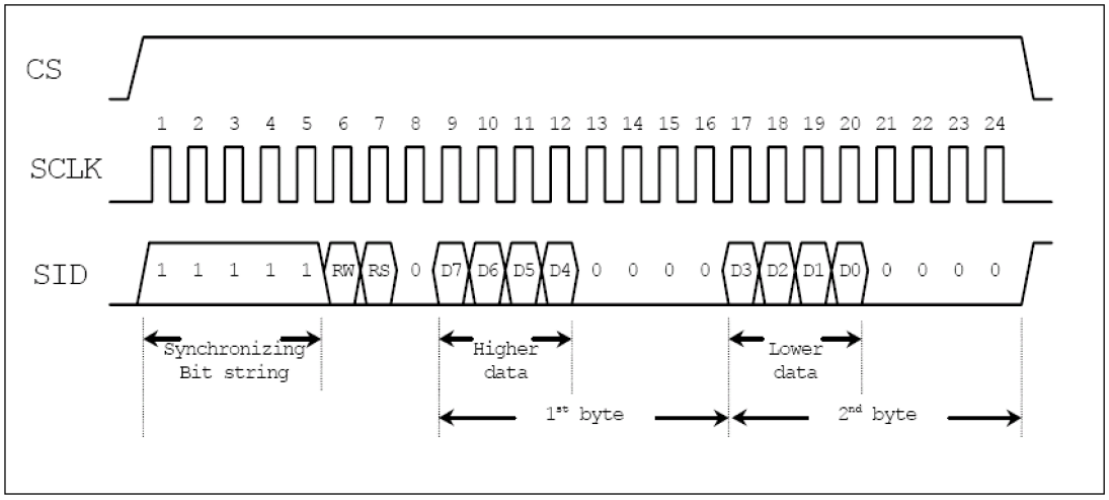

图 3-12 串口数据传输过程

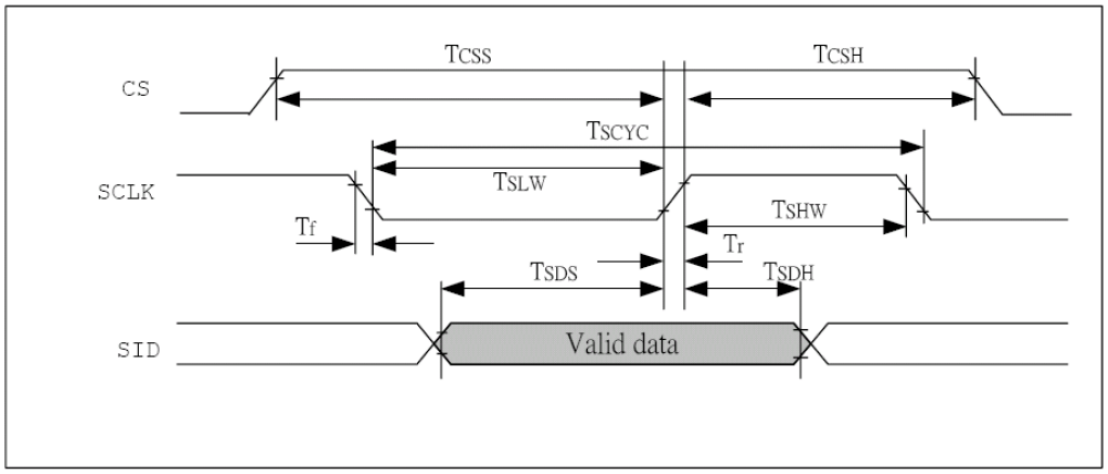

图 3-13 串口读写时序

LCD12864 程序设计分三步：

1. 初始化 LCD12864
2. 写指令
3. 写数据

3.2.2.4 红外遥控器

红外发射头仅需上电后对其信号输入脚发送对应波形即可。本模块实现的红外遥控功能所产生的波形为 PWM 波形。其波形如图 3-14 所示，当给信号输入脚输入 0.56ms 高电平后输入 0.565ms 低电平（共 1.125ms）则代表 bit 位 0，发送 0.56ms 高电平后发送 1.69ms 低电平（共 2.25ms）则代表 bit 位 1。连续发送 bit 位 0,1 的不同便能产生不同控制信号。

红外遥控一般具有引导码，本遥控器的引导码组成为 9ms 高电平后 4.5ms 低电平。遥控信号的发送需先发送引导码，然后再发送带有带有 0、1 数据的数据码。

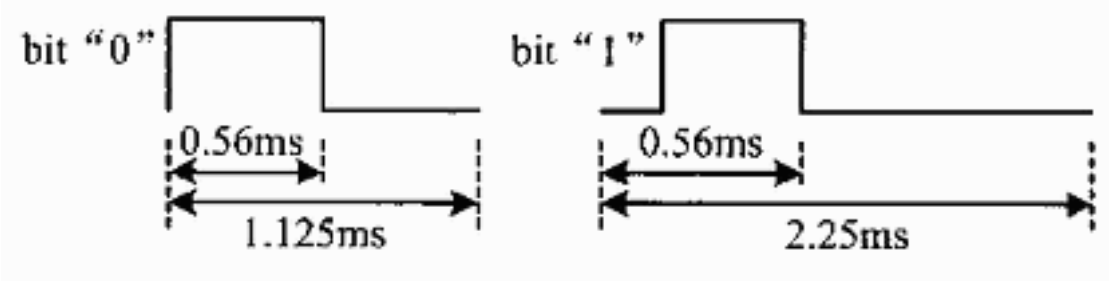

图 3-14 红外遥控编码波形

3.3 应用层

应用层使用树莓派作为开发环境， Python 语言和 Django 框架搭建。应用层架构如图 3-15 所示。Django 是时 Python 程序语言驱动的 web 应用框架，它包含对 HTML 页面的操作和对数据库的操作，这样就可以使用 python 语言通过 Django 架构对前端和后台进行控制。同时，树莓派通过 usb 转串口接收和发送 433MHz 无线通信模块的数据也是使用 python 语言进行的，这样通过 Python 语言对应用层的所有模块均能进行操作，增加了应用层程序开发的整体性，也便于程序开发的维护性。

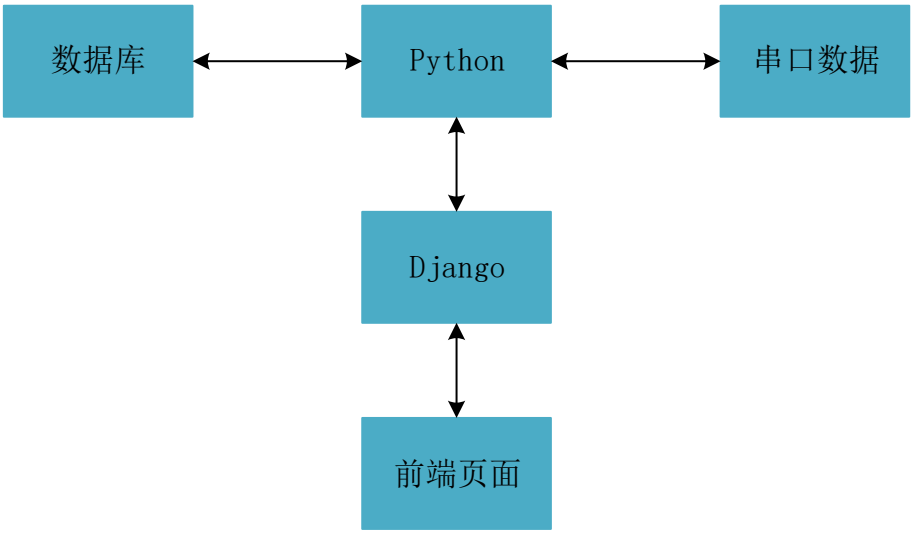

图 3-15 应用层架构

3.3.1 Python 接收及发送数据

433MHz 无线通信模块使用串口作为数据接口，我们通过 USBISP 模块将串口转换为 USB 再接在树莓派上。

Python 语言提供了丰富的库，树莓派使用 Python 语言能够简单地进行串口操作。Python 提供了串口操作库 serial。读取串口数据语句如下

```
import serial
SER = serial.Serial('/dev/ttyUSB0', 115200, timeout=0,
interCharTimeout=0.001)
r_data = SER.readline()
```

仅使用三条语句便能将串口数据读出，其中'/dev/ttyUSB0'为 USB 设备号，115200 为波特率，timeout 为超时设置，其为 0 表示无超时设置，interCharTimeout 为字符间隔超时。r_data 即为接收到的数据

发送串口数据语句如下：

```
import serial
SER = serial.Serial('/dev/ttyUSB0', 115200, timeout=0,
interCharTimeout=0.001)
SER.write('test')
```

其中，test 为发送出去的数据。

3.3.2 python 对数据库进行操作

Python 通过 MySQLdb 库对 MySQL 数据库进行操作。对数据库操作语句：

```
db = MySQLdb.connect("localhost", "root", "sql123", "sushe")
cursor = db.cursor()
# SQL 插入语句
sql = "INSERT INTO upload(roomname,status,time,dev)
values(%s,%s,%s,%s)"
times = time.strftime(
    '%Y-%m-%d %H:%M:%S', time.localtime(time.time()))
param = (name, status, times, dev)
try:
    # 执行 sql 语句
    cursor.execute(sql, param)
# 提交到数据库执行
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()
# 关闭数据库连接
db.close()

db = MySQLdb.connect("localhost", "root",
                     "sql123", "jiang", charset='utf8')
cursor = db.cursor()
```

```
# SQL 插入语句
sql = "INSERT INTO dormdb_dorm(roomname,status,time,dev)
values(%s,%s,%s,%s)"
times = time.strftime(
    '%Y-%m-%d %H:%M:%S', time.localtime(time.time()))
param = (name, status, times, dev)
try:
    # 执行 sql 语句
    cursor.execute(sql, param)
# 提交到数据库执行
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()
# 关闭数据库连接
db.close()
```

3.3.3 数据库结构

如图 3-16 所示为数据库结构，其中 upload 表存放来自感知层的数据，其表结构如图 3-17 所示，图 3-18 展示了 upload 表中的部分数据，在宿舍各数据信息表内 roomname 为宿舍号，time 为检测时刻，其中当 dev=02，status<100 时，status-100 为当前宿舍温度；当 dev=02，status>100 时，status 为当前宿舍一氧化碳浓度；当 dev=03，status=01 时，表示开门；当 dev=03，status=02 时，表示房门出现故障；当 dev=04，status=01 时，表示插座关闭；当 dev=04，status=02 时，表示插座打开。

user 表存放用户数据，用户数据以宿舍为单位，每个宿舍一个用户，通过宿舍号与密码进行登录，登陆后可以对应宿舍中的终端设备进行操作。同时，设置一具有较高级权限的管理员账号，该账号可供管理员使用，其可对所有宿舍的中断设备进行操作。

图 3-19 所示为 user 表结构，图 3-20 展示了部分表数据，用户数据表内 ID 为用户名，password 为用户密码，其中 201 用户为管理员用户，拥有所有房间管理权限，其余为普通用户，只对自己房间拥有管理权限。

表	操作
auth_group	浏览 结构 搜索 插入 清空 删除
auth_group_permissions	浏览 结构 搜索 插入 清空 删除
auth_permission	浏览 结构 搜索 插入 清空 删除
auth_user	浏览 结构 搜索 插入 清空 删除
auth_user_groups	浏览 结构 搜索 插入 清空 删除
auth_user_user_permissions	浏览 结构 搜索 插入 清空 删除
django_admin_log	浏览 结构 搜索 插入 清空 删除
django_content_type	浏览 结构 搜索 插入 清空 删除
django_migrations	浏览 结构 搜索 插入 清空 删除
django_session	浏览 结构 搜索 插入 清空 删除
downsignal	浏览 结构 搜索 插入 清空 删除
frontend_module	浏览 结构 搜索 插入 清空 删除
upload	浏览 结构 搜索 插入 清空 删除
user	浏览 结构 搜索 插入 清空 删除

图 3-16 数据库结构

#	名字	类型	排序规则	属性	空	默认	额外
1	ID	int(8)			否	无	AUTO_INCREMENT
2	roomname	varchar(10)	utf8_general_ci		是	NULL	
3	status	varchar(10)	utf8_general_ci		是	NULL	
4	time	datetime			是	NULL	
5	dev	varchar(10)	utf8_general_ci		是	NULL	

图 3-17 upload 表结构

	ID	roomname	status	time	dev
编辑 复制 删除	288	4848	1	2017-03-09 20:57:27	1
编辑 复制 删除	287	4848	2	2017-03-09 20:57:18	1
编辑 复制 删除	286	4848	1	2017-03-09 20:56:43	1
编辑 复制 删除	283	4848	2	2017-03-09 20:31:50	4
编辑 复制 删除	285	4848	2	2017-03-09 20:31:50	4
编辑 复制 删除	284	4848	2	2017-03-09 20:31:50	4
编辑 复制 删除	282	4848	1	2017-03-09 20:31:47	4
编辑 复制 删除	281	4848	2	2017-03-09 20:31:32	4
编辑 复制 删除	280	4848	1	2017-03-09 20:28:56	4
编辑 复制 删除	279	4848	2	2017-03-09 20:28:18	4
编辑 复制 删除	278	4848	48	2017-03-09 20:08:41	48
编辑 复制 删除	277	4848	48	2017-03-09 20:08:36	48
编辑 复制 删除	276	4848	48	2017-03-09 20:08:31	48
编辑 复制 删除	275	4848	2	2017-03-09 19:02:12	4
编辑 复制 删除	274	4848	1	2017-03-09 19:02:11	4

图 3-18 upload 部分数据

#	名字	类型	排序规则	属性	空	默认	额外
1	ID	int(11)			否	无	
2	password	varchar(30)	utf8_general_ci		否	无	

图 3-19 user 表结构

	ID	password
编辑 复制 删除	210	songhan
编辑 复制 删除	4848	hello
编辑 复制 删除	4849	000

图 3-20 user 表部分数据

3.3.4 Django 环境搭建

1、创建 django 工程

django-admin startproject hello

会创建 hello 目录，进入 hello 目录可以看到结构如下.

```
.
├── hello
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py
```

hello 为工程名，manage.py 为管理工具，settings.py 是工程的相关设置，包括数据库与安装的 app 等设置,urls.py 为网站页面的相关映射，将 python 相关函数映射为网址。

3.新建 app

python manage.py startapp index

此时目录结构如下

```
.
├── hello
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-35.pyc
│   │   └── settings.cpython-35.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── index
│   ├── admin.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── manage.py
```

可以看到该目录下多了 index 的文件夹。相比之下该目录下少了 settings.py 与 urls.py，主要用的有 views.py 与 models.py，还有需要自己创建的 forms.py，tamplates 文件夹。

4.编辑视图

编辑 views.py

```
from django.shortcuts import render

# Create your views here.

def index(request):
    string='hello'
    return render('index.html',{'string':string})
```

定义 index 函数，返回 index.html，同时传入 string 变量。

5.注册 index app

编辑 settings.py，添加 index 到已安装 app

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'index',
)
```

6.将 url 与函数连接

编辑 urls.py，将网址与函数连接

```
from django.conf.urls import include, url
from django.contrib import admin
#添加这一行
from index import views as index_views

urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    #添加这一行
    url(r'^index/$', index_views.index,name='index'),
]
```

访问 index 时将执行 index 函数。

7.编辑 index.html

在 Django 中，网页以模板的形式展现，Django 会自动寻找目录下的 tamplates 文件夹，所以先创建该文件夹

mkdir templates

cd templates

touch index.html

此时目录结构如下

```
.
├── hello
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-35.pyc
│   │   └── settings.cpython-35.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── index
│   ├── admin.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── templates
│   │   └── index.html
│   ├── tests.py
│   └── views.py
└── manage.py
```

编辑 index.html

```
<html>
<head>
    <meta charset="UTF-8">
    <title>Hello</title>
</head>
<body>
    {{string}}
</body>
</html>
```

8.启动服务器

需要先运行一下命令进行迁移，不是很懂 migrate 的翻译

python manage.py migrate

然后执行一下命令运行服务器

python manage.py runserver

通过访问 http://127.0.0.1:8000 就可以看到页面了

9.监听 ip,测试用服务器

修改 settings.py

ALLOWED_HOSTS = ['192.168.199.144']

执行以下命令

python manage.py runserver 0.0.0.0:8000

此时变完成了 Django 一个项目的生成，同理，使用该方式创立如图 3-21 所示结构的项目。

图 3-21 Django 框架

其中，serialrecv.py 为树莓派接收与写入数据库程序，db.sqlite3 为数据库文件，index 文件夹中包含首页网页页面等文件，nginx_conf 为 Nginx 服务器的配置文件，dormdb 中有登录界面等文件。

第四章 系统开发成果

4.1 实物展示

本系统中所涉及的功能模块均制作了具有完整功能的实物设备，如下列各图所示：

图 4-1 温度既烟雾报警器

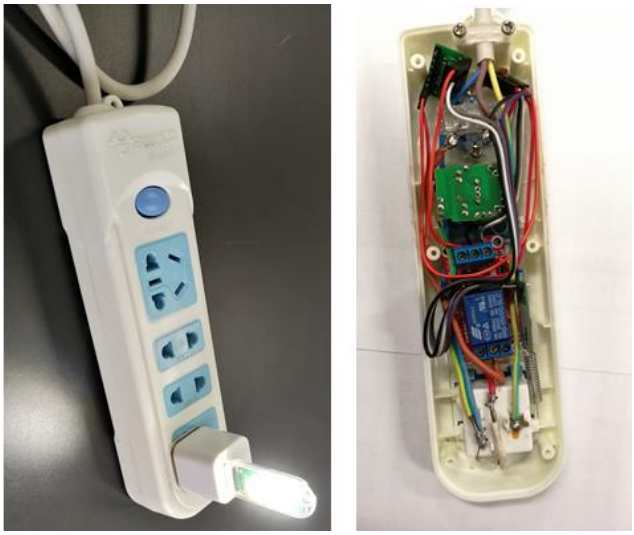

图 4-2 智能插座

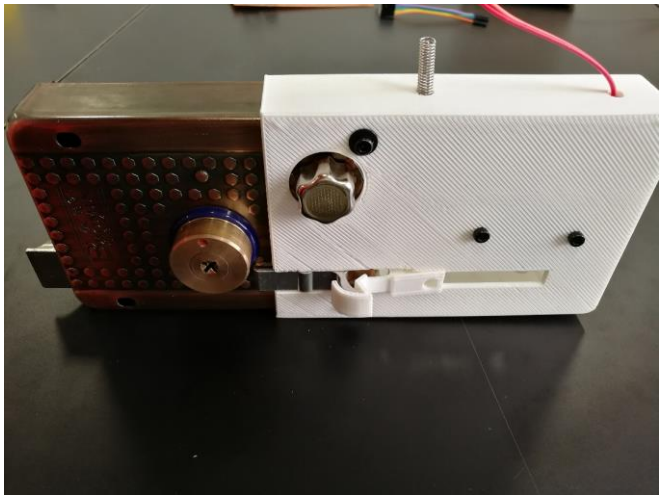

图 4-3 智能门锁

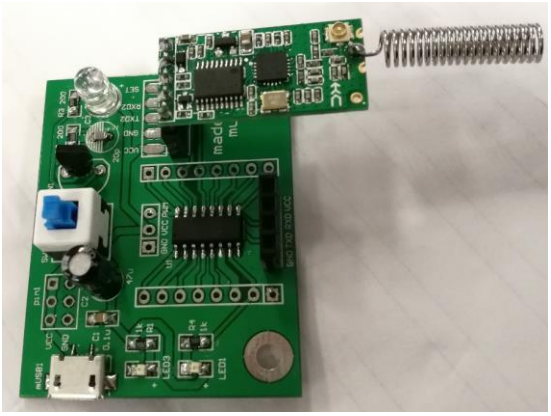

图 4-4 中继

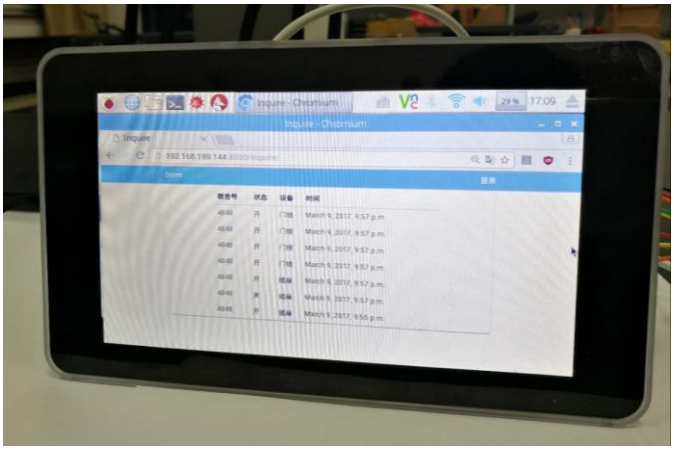

图 4-5 中控（树莓派）

智能宿舍系统的功能通过搭建的智能宿舍平台来实现。使学校生宿舍里的学生管理人员享受到现代智能家居带来方便与捷。用户主要分为两类：一是生活在宿舍的学群体，其可通过该平台上注册为一般用户来对宿舍内的电器开关

等进行远程监测与控制；另外一类是楼管理者，其可通过在该平台上注 册为管理员 ，来对整栋楼所有宿舍进行监控与管理。

使用网页端时，先进行身份确认，如图 4-6 所示为登录界面，分为普通用户和管理员两个通道。管理员登录后，可对数据库内所有宿舍进行查询和更改设置，既可以看到整栋楼的用户数据和宿舍内情况汇总，也可对电器进行操作，达到对宿舍监督的作用，保证宿舍的安全。普通用户登陆后，只能看到自己宿舍内的基本情况，可以对用电器进行操作，如图 4-7 所示。每次登陆，网页 form 表单通过对数据库内容进行访问后，格式化在网页内显示数据结果。管理员登陆界面内，roomname 是宿舍号，time 为检测时刻，status 表示状态，dev 表示设备。呈现结果如图 4-8 所示。

图 4-6 宿舍管理系统登录界面

图 4-7 用户管理界面

图 4-8 查询结果

4.2 系统测试

我对所开发的系统进行了测试，将系统中所有实物上电后进行数据传输测试。下图展示了测试场景

图 4-9 测试场景一

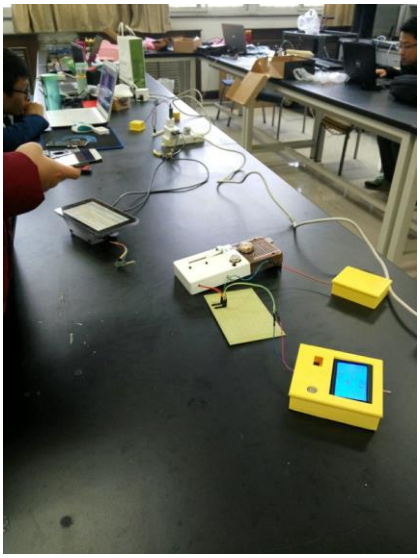

图 4-10 测试场景二

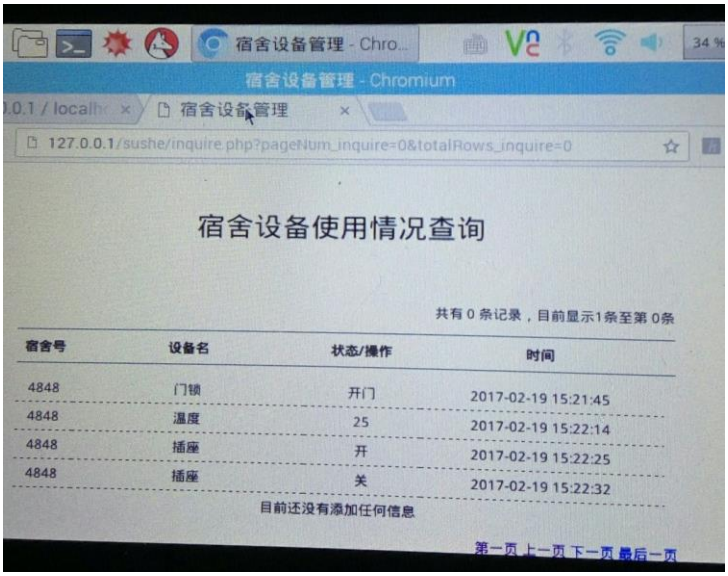

图 4-11 测试场景三

经过测试，本系统自组网功能正常，数据传输正常，但在多跳后系统具有较高延时（三个中继以上时有 1s 左右延时），系统显示及控制正常。

第五章　总结与展望

5.1 总结

本系统设计制作完成了一套具有自组网功能的智能宿舍系统。

本文首先介绍了物联网与智能宿舍系统的基本概念以及基于这些概念提出的一种智能宿舍解决方案的总体设计思想；随后，本文介绍了本智能宿舍系统所需要的前期准备及必要软硬件条件；然后，介绍了智能宿舍系统详细的的软硬件实现方式；最后，介绍了该系统的开发成果。

5.2 展望

本文所介绍的基于物联网的智能宿舍系统实现方案是对物联网、智能宿舍系统实现的初步探索，本系统仍然有较多的问题有待解决。在后续的研究中，需解决的问题有：如何提高组网效率；如何降低系统延时；提高数据库反应速度；增加设备容量等问题。

参考文献

[1] 吴小芳, 物联网与大数据的新思考[J], 通讯世界, 卷 01, pp. 1-2, 2017.

[2] 郭天祥, 51 单片机 C 语言教程[M], 北京: 电子工业出版社, 2013, pp. 2-11.

[3] STC 公司, STC15F2K60S2 系列单片机器件手册[M], 2014.pp.63-83

[4] 宁炳武, Zigbee 网络组网研究与实现[D], 大连理工大学, 2007.

[5] 陈慕雷.基于多模传输协议的智能家居网关开发[D]. 西北工业大学.2014

[6] 李晓卉. 智能家居无线传感器网络路由[M]. 北京: 电子工业出版社，2014.3

[7] 吕雪峰. 嵌入式 Linux 软件开发从入门到精通[M]. 北京: 清华大学出版社，2014.9

[8] 邓自军.基于 433M 模组局域网协议的设计与实现[D].北京邮电大学,2012

致谢

大学四年的光阴转瞬即逝，转眼间就到了毕业的时候，回想起大学四年的时光，最让我难忘的还是和同学们在一起的欢乐和一起讨论学习时乐趣。在这四年里，我不论在思想、生活、学习上都得到了很大的成长，这一切都离不开同学和老师在学习和生活中对我的帮助，对我来说，这四年是充实的四年，这四年是难忘的四年，这四年也将成为一段美好并值得怀念的记忆　　。

通过这一个学期的努力，我终于完成了毕业论文的撰写，在这里我要非常感谢在我的论文写作过程中给我提供支持和帮助的老师、朋友和同学们。特别要感谢的是李彬老师，在我的毕业设计过程中，不论是项目中遇到的技术上的问题，还是方向上的疑惑，李老师都会耐心地指导我，从毕设刚开始的方向的制定，技术方案的建议，对成果的检测等方面，李老师都倾注了大量的心血，提出了众多宝贵意见和建议，给予了精心的指导，让我的论文得以最终完稿，李老师对学生平易近人，对学术的严谨的态度都让我获益匪浅，在此谨表示我深深的谢意!

再者我要感谢我的同学们，在我毕设过程中遇到的很多问题是在请教了我的同学们后得到了解决，感谢他们对我不厌其烦地指导，感谢他们的宝贵的意见和建议，也要感谢他们在四年大学生活中对我的照顾。

然后我还要感谢我的父母，感谢他们在我求学道路上对我的支持。

最后，还要郑重地感谢西北工业大学，感谢学校提供给我的优秀的学习环境，感谢学校对学生无私的栽培，感谢学校对我的敦敦教诲。

毕业设计小结

本文第一章介绍了物联网和智能宿舍的基本背景与其实现所需的基本技术，然后基于这些技术背景提出了一套智能宿舍系统并概述了本系统的基本架构和实现过程。

第二章介绍了为实现本设计所需的软硬件条件以及所需各类传感器、MCU及嵌入系统的选择。

在第三章中，详细介绍了本设计的实现细节。本系统基于物联网的三个层次：感知层、网络层和应用层进行开发。感知层为系统中感知及进行各种具体操作的直接设备，在感知层中，本系统利用单片机及各类传感器实现了具有一定功能的中断设备。网络层用于传输系统中产生的各类信息数据，网络层的实现基础是设计了一套适合 433MHz 广播式无线通信模块的具有自组网能力的数据格式，其编程实现主要为中继设备的编程，同时在终端及中控中也包含有网络层的编程设计。应用层分析、处理和存储各类数据，该层也是系统与人进行直接交互的接口，应用层使用树莓派作为中控机，树莓派运行基于 Linux 的 rasbian 操作系统，利用 Python、Django 框架作为工具，实现应用层的设计。

完成本系统的所有设计任务后，在第四章展示了设计成果——一套具有完整功能的智能宿舍系统。

第五章的内容是对本次毕业设计的内容做简单的总结，并且对该平台以后的发展做出了评价与展望。