基于 Ruff 操作系统的室内灯控系统的研究与设计

摘　要

现代社会，随着人们对舒适性生活要求的不断提高，智能家居的需求与日俱增。而随着 JavaScript 语言的迅速发展，本文使用 JavaScript 语言来操控各型硬件设备，构建了一种实用的物联网家庭服务系统。

该灯控系统基于 Ruff 操作系统，一个支持 JavaScript 语言开发应用的物联网操作系统。该系统由自适应灯光模块、睡眠模块、监控模块和心情颜色模块这 4 个功能模块组成。并使用到了 LED 灯、光照传感器、声音传感器以及人体红外传感器等。此外，本文还设计了一个 HTML5 移动端页面，可以方便地通过手机端来控制 Ruff 操作系统，从而控制这些硬件模块。页面包括模块控制、IP 修改以及颜色修改模块。

本文设计的基于 Ruff 操作系统的灯控系统，模拟和实现了一个智能家居的情景。例如，自适应灯光模块可以根据周围环境亮度改变灯光亮暗度，从而节省电能；睡眠模块可以监听人们睡眠时的状态，免去黑暗中摸索开灯的烦恼；而监控模块能在主人不在家时，监控家里的情况，警告入侵者并远程通知主人。这为智能家居以及 JavaScript 语言在物联网领域的发展奠定了重要基础。

关键词　Ruff 操作系统；灯光控制；智能家居；手机远程控制

Ruff-based research and design of indoor light control system of the operating system

Abstract

With the rapid increasement of requirements for comfortable life in modern society, the demand for smart home growing. With the rapid development of JavaScript language, this article uses JavaScript language to manipulate various types of hardware devices, and constructs a practical Internet of family service system.

The lamp-control system is based on Ruff operating system, a IoT operating system that supports JavaScript language development applications. The system consists of 4 functional modules: responsive light module, sleep module, monitoring module and mood color module. And the use of LED lights, light sensors, sound sensors and human infrared sensors. In addition, this article also designed a HTML5 mobile page, which can conveniently control the Ruff operating system through the handset side, thereby controlling the hardware modules. The page includes module control, IP modification, and color modification modules.

This article designs a lamp control system based on Ruff operating system, simulates and realizes a scene of intelligent home. For example, the responsive light module can change the brightness of the light automatically according to the ambient brightness to save the energy; sleep module can listen to people's sleep state, eliminating the trouble to explore the lights in the dark; and monitoring module can monitor the situation when owner is not at home. It can warn the intruder and remotely notify the owner. This has laid an important foundation for the development of smart home and JavaScript language in the field of IoT.

Keywords Ruff operating system; lighting control; smart home; remote control of mobile phone

目　录

第 1 章 绪论 1
1.1 研究背景 1
1.2 研究意义 1
1.3 国内外研究现状 2
1.4 本文研究内容 2
第 2 章 智能家居硬件设计 4
2.1 Ruff 开发板 4
2.2 人体红外传感器 4
2.3 LED 灯 5
2.4 光照传感器 6
2.4.1 功能特点 6
2.5 声音传感器 6
2.6 蜂鸣器 7
2.7 LCD 液晶显示器 7
第 3 章 Ruff 操作系统软件设计 9
3.1 JavaScript 语言 9
3.2 Ruff 操作系统 9
3.3 自适应灯光模块 10
3.3.1 HSL 模型 10
3.3.2 RGB 转换到 HSL 10
3.3.3 HSL 转换到 RGB 11
3.4 睡眠模块 13
3.5 监控模块 15
3.6 心情颜色模块 16
3.7 服务器监听模块 17
3.7.1 同源跨域 18
第 4 章 远程控制界面设计 20
4.1 HTML 20
4.2 CSS 20
4.3 工具 20
4.4 页面 21
4.5 Scss 设计 21
4.6 JavaScript 设计 23

4.6.1 正则 23
第 5 章 系统功能实验与分析 25
5.1 自适应灯光模式实验 25
5.2 睡眠模式实验 26
5.3 监控模式实验 26
5.4 心情颜色模式实验 26
结 论 28
致 谢 ..... **错误!未定义书签。**
参 考 文 献 ..... **错误!未定义书签。**
附录 A Ruff 代码 ..... **错误!未定义书签。**
附录 B APP 代码 ..... **错误!未定义书签。**
附录 C 译文 ..... **错误!未定义书签。**
智能家居网络：结合无线和电力线网络的经验 ..... **错误!未定义书签。**
附录 D 外文原文 ..... **错误!未定义书签。**

第1章 绪论

1.1 研究背景

随着生活水平的提高，人们迫切渴望拥有一套先进、友好、智能的家居管理系统，以便免除一些繁杂的工作，因此智能家居领域逐渐成为科研学者关注的焦点。

在20世纪80年代末，随着电子技术、通信与信息技术的发展，人们可以利用总线技术对家中通信、家电设备进行管理，这便是智能家居的原型，在当时的美国叫Smart Home。21世纪后，随着技术的发展，各种各样的实现方式直接推动了智能家居的多样化。智能家居的发展主要经历了4个阶段。第一个阶段主要利用同轴线、两芯线组成家庭组网，从而控制灯光、窗帘和安防。第二个阶段使用了RS-485线和部分IP技术组网，实现了可视对讲、安防等新功能。第三个阶段集中化了家庭智能控制系统，由主机控制安防、计量。第四个阶段基于全IP技术，末端设备利用zigbee等技术，并且为了让用户的需求得到定制化和个性化的服务，业务采用“云”技术[1]。图1.1为一个理想化的智能家居概念图。

图1.1 智能家居概念图

1.2 研究意义

智能家居对提升人们生活品质有着重要的价值。比如，自动灯控系统可以节省能源、保护视力甚至改善心情；自动监控系统可以大幅度提升居家安全，一定程度上避免经济损失和维护社会安定；自动温度调节系统可以提升居住体验，对抗外界的恶劣环境，在一些极端环境下起到重要作用。所以说智能家居系统的市场充满了机遇，但是在功能实现和系统推广的道路上也布满了荆棘。智能家居涉及到的业务影响了生活的各方面及众多其他传

统产业，带动了建筑业、制造业、信息技术行业等非常多的领域发展。

1.3 国内外研究现状

美国的智能家居追求舒适与享受，在家居构造和硬件设计上比较奢侈豪华，因而能源消耗大，与当前低碳环保的观念不符。日本的智能家居则以人为本，受到土地资源的大小和人口密集程度，在设计上大量使用新材料和新技术，在保证功能的同时，还兼顾未来的可维护性及环保性。并且充分地利用了网络、控制以及人工智能技术，为现代化住宅提供了许多非常新奇有趣的思路。德国的智能家居则更专注和严谨，注重基本和专项功能。

而中国的智能住宅化目前还处在初级发展阶段，因为智能家居在我国起步比较迟。以前我国主要是有线部署的方式，这样可以简化各个功能的设计，但是布线繁杂，难以维护。而随着物联网的出现，科学家们利用无线技术结合传感器，在智能家居的发展上走了很大一步。比如马菁菁等人研究了zigbee无线通信技术在智能化家居中的应用[2]，申斌等人研究了物联网在智能家居领域的设计实现[3]，而邵鹏飞等人更是将智能家居延伸到了移动互联网[4]。而目前智能家居还存在着各种不同又令人头疼的问题，比如缺乏规范、统一的标准，系统操作比较复杂（相对于使用者而言），功能实用性不够强而且维护成本比较高，还有在连入因特网以后可能会面对黑客的网络攻击，造成个人隐私泄露或者财产损失。

20世纪末21世纪初时代的人们所提的物联网和智能家居，大部分都是使用C语言或者汇编语言等偏底层的语言去实现的，因为当时人们对硬件的理解不够深，而且在软件思想方面受到限制，没有做高层次的抽象。在平常的大学专业课上，学生接触到的都是没有联网功能的单片机，比如51单片机或者是ARM开发板，只能实现一些比较简单，没有实际意义的功能。大部分智能家居终端传输技术是通过蓝牙技术或者Zigbee技术来实现的，可是，如果一个智能家居不能联上因特网，那么必然会失去很多实用又有趣的功能，尤其是在现在的大数据时代，完善的数据统计分析可以为用户提供定制化个性化的服务。单片机上的传感器测量的数据以及控制功能，只有放到互联网上或者说人们可以远程得到数据才有更好的利用价值。

1.4 本文研究内容

本文主要的研究工作即利用JavaScript语言开发一个实用的智能家居系统，并且可以在移动端通过联网控制这个智能家居系统，系统设计如图1.1所示。

第1章，绪论。首先介绍了智能家居的研究背景及研究意义，然后分析了智能家居在国内外研究的现状，并根据其优缺点，提出一种新的解决方案。

第2章，智能家居硬件设计。介绍这个简单的智能家居系统包含的各个硬件模块，并阐

述每个硬件模块的参数、特点、使用方法。

第3章，Ruff操作系统软件设计。主要介绍了Ruff操作系统，Ruff开发板上所应该搭载的模块功能，以及功能的设计和实现。

第4章，远程控制界面设计。首先介绍了HTML语言，CSS语言和JavaScript语言，接着对网页设计和逻辑控制进行阐述。

第5章，实验部分。对各个功能模块进行测试，根据测试结果，调整程序。

结论。总结本文主要研究内容，思考文中设计的缺点，并畅想其在未来的发展。

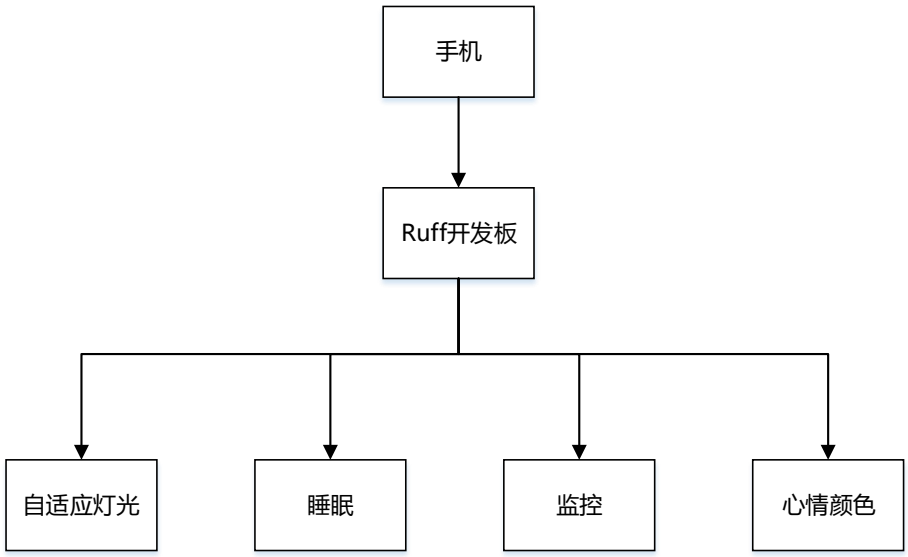

图1.2 灯控系统设计图

第2章 智能家居硬件设计

本章将对智能家居系统所用到的硬件模块进行详细介绍。所有的硬件模块包含 Ruff 开发板、人体红外传感器、LED 灯、光照传感器、声音传感器、蜂鸣器以及 LCD 液晶显示器。

2.1 Ruff 开发板

Ruff 开发板是内置 Ruff 操作系统的硬件板卡，它的所有信息保存在一个 json 文件中。开发板的资源包括硬件接口、电源管脚、接地管脚、板载设备。Ruff 支持的硬件接口类型有 GPIO、I2C、UART、PWM、ADC，并且为这些接口都提供了驱动，在官网写了详细的文档。Ruff 开发板的驱动是 RuffOS，VCC 有 3.3 V 和 5 V 接口，GND 是通用的。在 Ruff 开发板上有 3 对 I2C 读写接口，但是在虚拟硬件连接图上只显示一对，因为软件设计上这 3 对接口是共用的。图 2.2 是 Ruff 开发板实物图，黄色的是 ADC 引脚，粉色的是 gpio 和 io 引脚，红色是 i2c.SDA 引脚和 i2c.SCL 引脚，淡紫色的是 gnd 引脚，绿色的是 vdd 引脚，紫色的是 pwm 引脚。这块板子还自带 3 个按钮和一个 reset 按钮。

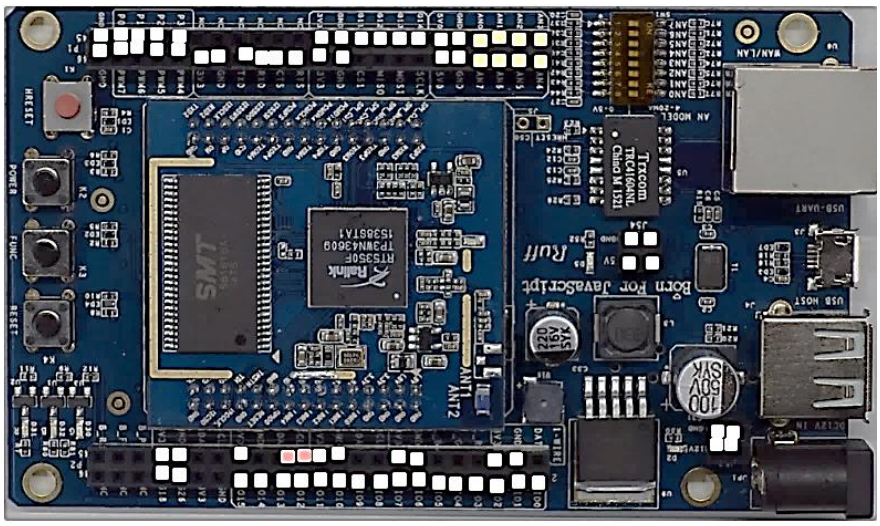

图 2.1 Ruff 开发板实物及引脚图

2.2 人体红外传感器

本文使用的是自动控制的人体红外传感器，型号是 HC-SR501。在设计中使用到了 LHI778，来自德国制造。这个传感器可以在超低电压下工作，所需要的电流比较少，所以可以用在干电池供电的适应各种场景的设备下。图 2.2 是 HC-SR01 的实物图，有两个电平可以调节触发不同模式。

HC-SR501 在检测到有人在其检测范围内活动时，就输出高电平。这时候根据由使用者所设置的触发方式来判断，如果是可重复触发，且一直检测到有人在活动，则一直输出

高电平；如果不可重复触发，那么就等待一段时间由高电平变为低电平。这个等待的时间叫感应封锁时间，就是这段时间内不接受人体活动的信号。此外，本文所用的人体传感器还有温度补偿机制，高温时会补偿性能。而且在白天可以设置为不接收任何信号。如图 2.3 为 HC-SR501 电路原理图。

图 2.2 HC-SR501 实物图

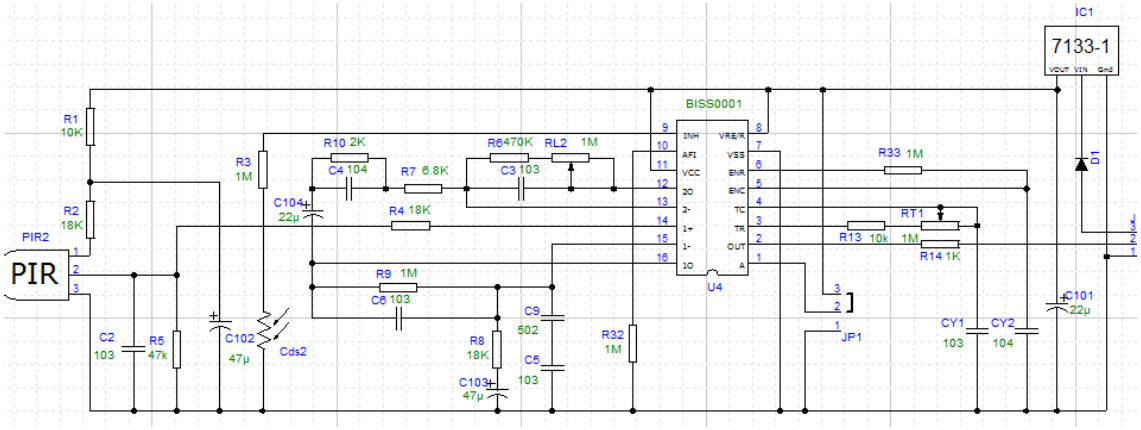

图 2.3 HC-SR501 原理图

2.3 LED 灯

本文使用的 LED 灯型号是 ky-016，一个包含 R、G、B 三个 PWM 接口的全彩 LED 模块。本文通过控制红绿蓝三原色的值，也就是控制三个 PWM 波的值，来达到任意的颜色，因为三原色可以混合成任意颜色，任意一个 PWM 的值的范围是 0~255。全彩 LED 灯的主要工作原理如下，通过红绿蓝这三种光的三原色以各种比例混合，也就是调节 LED 灯的占空比，来实现各种颜色。因为 LED 灯的扫描速度很快，再加上人眼自带的视觉惰性，人会感觉到渐变的效果。如图 2.4 是三色灯实物图，图 2.5 是三色灯的电路结构图。

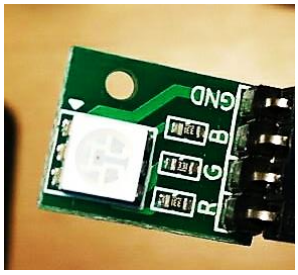

图 2.4　三色灯实物图

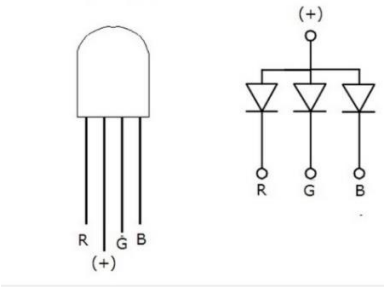

图 2.5　三色灯电路结构图

2.4 光照传感器

本文使用的光照传感器是数字光模块 GY-30，它是一种通用的光照度检测模块，内置模数转换电路，可以直接数字输出。这个数字光模块所用芯片为 BH1750FVI，该芯片是用于 I2 Cbus 接口的数字环境光传感器 IC。它使用了 I2C 总线接口，其中包含了 VCC、SCL、SDA、ADDR、GND 等接口，工作电压范围为 3 – 5 V，光照范围为 0-65535。图 2.5 是数字光模块 GY-30 的实物图，图 2.7 是 GY-30 的电路说明图。

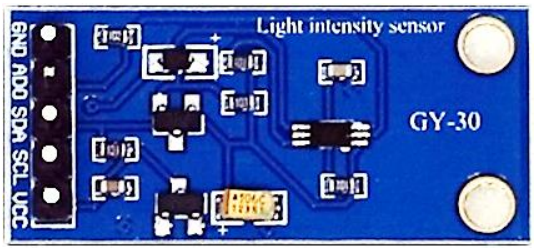

图 2.6　GY-30 实物图

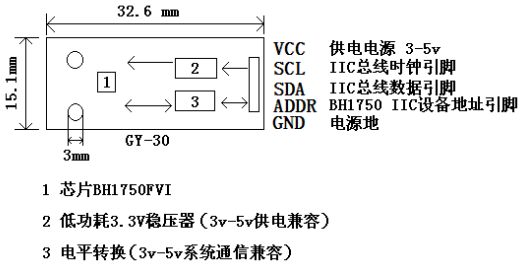

图 2.7　GY-30 电路说明图

2.4.1 功能特点

①直接数字输出，使使用者跳过复杂的计算，跳过繁杂的校准；

②不区分环境光，室内和室外都可以使用；

③比较接近视力的光谱特征；

④广泛应用于 1 勒克斯高精度测量的情景；

⑤遵循标准的恩智浦 IIC 通信协议。

2.5 声音传感器

本文使用的声音传感器只能识别有没有声音，不能用来识别声音的频率或者大小。因为此声音传感器是通过震动原理来判断，因此不能用来做语音识别等内容。此声音传感器包含 VCC、GND、D0 接口，工作电压范围是 3.3 V - 5 V。D0 口在检测不到声音时，保持高电平状态，当检测到周围的声音强度大于或者等于传感器内设置的阈值时，此时算检测到声音，D0 接口由高电平变为低电平。图 2.8 是声音传感器实物图。

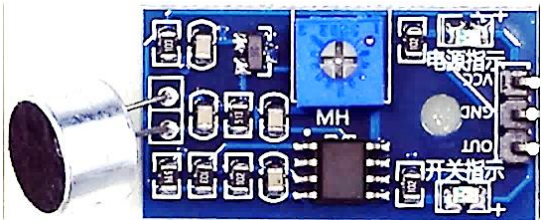

图 2.8　声音传感器实物图

2.6 蜂鸣器

本文使用的蜂鸣器是 FC-49 模块，是一个无源蜂鸣器。工作电压范围是 3.3-5V，包含 VCC、GND、I/O 口。蜂鸣器驱动电路由电源滤波电容、三极管、续流二极管和蜂鸣器组成。有源蜂鸣器自身带有震荡电路，所以直流信号可以直接让其鸣叫，也就是插上电源就会叫，而且比较贵，而无源蜂鸣器则必须利用加音频驱动信号。图 2.9 为蜂鸣器实物图，图 2.10 为蜂鸣器电路结构图。

图 2.9　蜂鸣器实物图

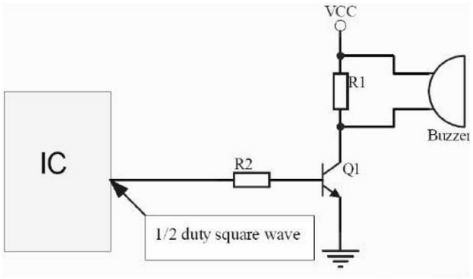

图 2.10　蜂鸣器电路结构图

2.7 LCD 液晶显示器

本文使用的 LCD 液晶显示屏型号是 LCD1602，是能够显示 32 个字符的工业的点阵型字符型液晶，不过缺点是智能显示字母、数字、符号等字符，工作电压为 5V。也就是说，该液晶显示器无法显示 ASCII 码表以外的字符，比如 unicode 字符类型的中文字符。图 2.11 为 LCD1602 实物图，图 2.12 为 LCD1602 引脚图。

图 2.11　LCD1602 实物图

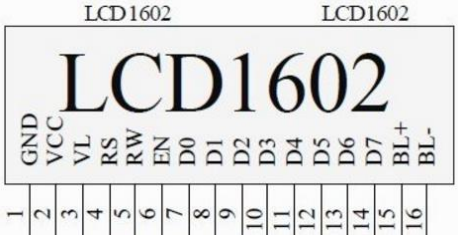

图 2.12　LCD1602 引脚图

LCD1602 共 16 个管脚，编程用到的主要管脚有：RS（数据命令选择端），R/W（读写选择端），E（使能信号）。一般是通过这 3 个管脚来进行初始化、写命令和写数据。LCD1602 内部控制器总共有 11 条指令，1 为高电平，0 为低电平。如果是输入数据，则根

据 D0~D7 管脚的高低电平查找字符集的字符代码，并映射出字符。表 2.1 为 LCD1602 位控制表。

序号	指令	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回	0	0	0	0	0	0	0	0	1	*
3	置输入模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	置功能	0	0	0	0	1	DL	N	F	*	*
7	置字符发生存储器地址	0	0	0	1	字符发生存储器地址					
8	置数据存储器地址	0	0	1	显示数据存储器地址						
9	读忙标志或地址	0	1	BF	计数器地址						
10	写数到 CGRAM 或 DDRAM	1	0	要写的数据内容							
11	从 CGRAM 或 DDRAM 读数	1	1	读出的数据内容							

表 2.1　LCD1602 位控制表

第3章 Ruff 操作系统软件设计

本章首先会介绍 JavaScript 语言和 Ruff 操作系统，然后将会对 Ruff 各个功能模块作详细介绍，包括自适应灯光模块、睡眠模块、监控模块、心情颜色模块和服务器监听模块。

3.1 JavaScript 语言

JavaScript，1995 年由 Netscape 公司的 Brendan Eich 创造而成，是一种轻量的，解释性（无须预编译），弱类型的脚本语言，初期主要用来向 HTML 页面添加交互行为[5]。名字上蹭了 Java 当时的火热，这是因为当时 Netscape 公司的高层管理希望这门语言像 Java 一样火。然而实际上 JavaScript 更像 C 语言和 Self 语言的后代。JavaScript 语言过去的运行速度比较糟糕，而且一些设计比如所有变量都运行在全局模式下等设计有重大缺陷[6]。但即使如此，因为它是所有浏览者支持的唯一语言，可以说是集万千宠爱于一身，不断地有大牛来给它填坑，所以火热程度不亚于 Java。

随着 Web 的演进以及互联网的火热发展，JavaScript 也变得更为强大和成熟。直到 2008 年，Google 开发了 V8 引擎，它直接将 JavaScript 编译成机器码，不经过字节码，将 JavaScript 的性能提升到了极致，使得 JavaScript 在浏览器端大放异彩。于是在 2009 年，Ryan Dahl，基于 V8 引擎创造了 node.js，使得 JavaScript 甚至可以在服务器端使用，而且运行的很好。至此，JavaScript 在客户端和服务器端都能应用，越来越多的 JavaScript 开发者向全栈工程师进发。

3.2 Ruff 操作系统

随着物联网的发展，许多 JavaScript 开发者都在想，JavaScript 语言是否可以应用于嵌入式端呢？类似于 nodemcu 可以运行 Lua 脚本一样，JavaScript 语言同样也可以跑在嵌入式硬件上。然后就出现了 Espruino，Tessel，Ruff 等利用 JavaScript 语言来操控硬件的系统，但是 Espruino 解释运行 JavaScript 语言速度很慢，只能用于测试环境或者低实时控制的场景，比如简单的开关控制器；而 Tessel 呢，是能跑 node.js 的最低配置硬件。

Ruff 开发者 Roy Li 认为嵌入式开发门槛太高，很大的原因在于抽象层次做的不够。就算用 JavaScript 语言去开发嵌入式端，如果还是对 pin 脚，对 GPIO 操作一个变频来达到目标，那么就还是用原始的方法在做开发。如今的应用开发，要么就是不需要写代码，要么就是能凭着自己的想法下意识地构思出代码。所以说，语言并不是最重要的，重要的是对硬件的抽象。

Ruff是一个JavaScript运行时[7]，专为硬件开发而设计。Ruff对硬件进行了抽象，使用了

基于事件驱动、异步I/O的模式，使硬件开发变得轻量而且高效。他借鉴了JQuery的语法糖，JQuery是一个快速简洁且热门的JavaScript框架[8]。且提供了一些封装好的硬件模块，支持各种传感器、外设模块等，用想象力连接物理世界，激发真正的创造力。此外，它告别了传统的交叉编译、烧写板卡的低效循环，应用逻辑全部在PC上完成，可简洁地一键部署。Ruff的简洁高效以及缜密舒适的文档吸引着越来越多的JavaScript开发者。

3.3 自适应灯光模块

自适应灯光模块，顾名思义，就是根据当前的环境，使 LED 灯的亮度自动变化。从原理上讲，就是隔一段时间获取当前环境的光照度，然后“赋值”给 LED 灯。这样不仅可以节省电能，而且能够给用户传递一种时间和氛围感。

3.3.1 HSL 模型

因为全彩 LED 只能设置 RGB 值，不能直接设置亮度，所以本文需要一种能够将亮度和 RGB 值互相转换的方法。本文采用了 HSL 色彩模型，这是一种将 RGB 色彩模型中的点在圆柱坐标系中的表示法[9]。HSL 色彩模型包含色相(Hue)、饱和度(Saturation)、亮度(Lightness)。色相是色彩的基本属性，每个颜色都有不同的色相，角度从 0 到 360°；饱和度是色彩的纯度，范围是 0~100%；亮度控制色彩的明暗，取值范围是 0~100%。如图 3.1 所示，a 代表 HSL 的 3d 模型截图，b 表示当设置饱和度固定为 1 的时候，亮度和色相的关系图，c 表示亮度固定为 1/2 时，色相和饱和度的关系图，d 表示色相固定为 0°/180°时，亮度和饱和度的关系图。图 3.1 为 HSL 模型图以及 H、S、L 关系图。

3.3.2 RGB 转换到 HSL

设(r,g,b)分别代表红、绿和蓝，$r,g,b \in [0,1]$。设max为(r,g,b)中最大值，min为(r,g,b)中最小值。转化为(h,s,l)，$h \in [0,360)°$，$s,l \in [0,1]$。计算公式如下：

$$h=\begin{cases} 0°, & if\ \ max=min \\ 60°\times\frac{g-b}{max-min}+0°, & if\ \ max=r\ \ and\ \ g\geq b \\ 60°\times\frac{g-b}{max-min}+360°, & if\ \ max=r\ \ and\ \ g<b \\ 60°\times\frac{b-r}{max-min}+120°, & if\ \ max=g \\ 60°\times\frac{r-g}{max-min}+240°, & if\ \ max=b \end{cases} \tag{3-1}$$

$$s=\begin{cases} 0, & if\ \ l=0\ \ or\ \ max=min \\ \frac{max-min}{max+min}=\frac{max-min}{2l}, & if\ \ 0<l\leq\frac{1}{2} \\ \frac{max-min}{2-(max+\min)}=\frac{max-min}{2-2l}, & if\ \ l>\frac{1}{2} \end{cases} \tag{3-2}$$

$$l=\frac{1}{2}(max+min) \tag{3-3}$$

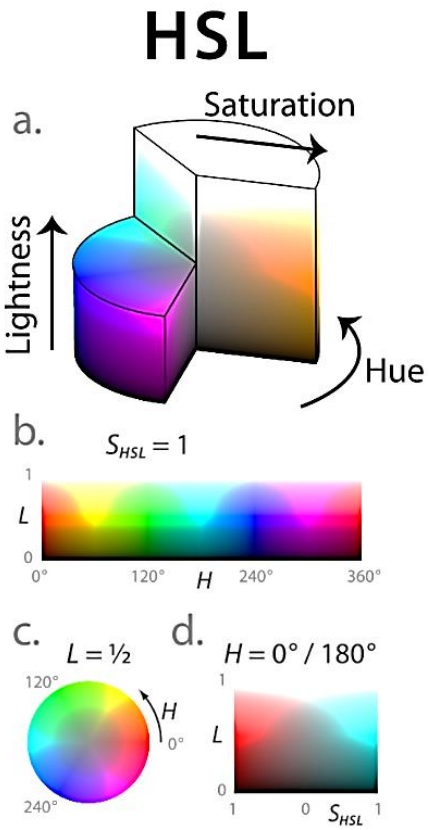

图 3.1　HSL 模型图

3.3.3 HSL 转换到 RGB

设(h,s,l)分别代表颜色的色相、饱和度和亮度，$h \in [0,360)°$，$s,l \in [0,1]$。转化为(r,g,b)，$r,g,b \in [0,1]$计算公式如下：

$$r = g = b = l,\ if\ \ s = 0 \tag{3-4}$$

$$\begin{cases} q = \begin{cases} l \times (1+s), & if\ \ l < \frac{1}{2} \\ l + s - (l \times s), & if\ \ l \geq \frac{1}{2} \end{cases} \\ p = 2 \times l - q \\ h_k = \frac{h}{360}\ （将\ h\ 单位换算[0,1) \\ t_R = h_k + \frac{1}{3} \\ t_G = h_k \\ t_B = h_k - \frac{1}{3} \end{cases} \tag{3-5}$$

下面以p,q,t_R为例，计算 R 的值，G 和 B 的值计算同理。

$$\begin{cases} if\ \ t_R > 1\ \ t_R = t_R - 1 \\ if\ \ t_R < 0\ \ t_R = t_R + 1 \end{cases} \tag{3-6}$$

$$R=\begin{cases} p+\big((q-p)\times 6\times t_R\big), & if\ \ t_R<\frac{1}{6} \\ q, & if\ \frac{1}{6}\le t_R<\frac{1}{2} \\ p+\Big((q-p)\times 6\times\big(\frac{2}{3}-t_R\big)\Big), & if\ \frac{1}{2}\le t_R<\frac{2}{3} \\ p, & otherwise \end{cases} \tag{3-7}$$

如果代入情景的话，要考虑用户的需求，比如当光照度小于某个程度的时候，不能让灯再暗下去了，所以本文设置一个最小灯光阈值 MIN_ALLOW_L；同理，当光照度大于某个值的时候，可能就不需要灯光了，再设置一个最大灯光阈值 MAX_ALLOW_L。

自适应灯光模块设计思路，图 3.2 是自适应模式流程图：

步骤1. 当 Ruff 开发板初始化时，进行自检并监听命令。

步骤2. 如果自适应模式已打开或者接收到远程命令，则进入自适应程序。

步骤3. 首先获取当前 LED 的 RGB 值，并将其转化为 HSL，然后保存 H 和 S 两个值。因为本文只是改变亮度而已，这样可以让不同颜色的灯光自适应。

步骤4. 接着获取当前光照度，再根据设定的最小和最大阈值对光照度进行过滤，得到 L。

步骤5. 根据计算得到的 L，还有先前保留的 H 和 S，将这 3 个参数转化为 RGB 值并赋值给 LED。

步骤6. 当这一段程序都完成以后，判断自适应灯光模块是否依然打开，如果打开的话，则以 1s 的间隔重复执行步骤 3-6，如果关闭的话，则关闭 LED。

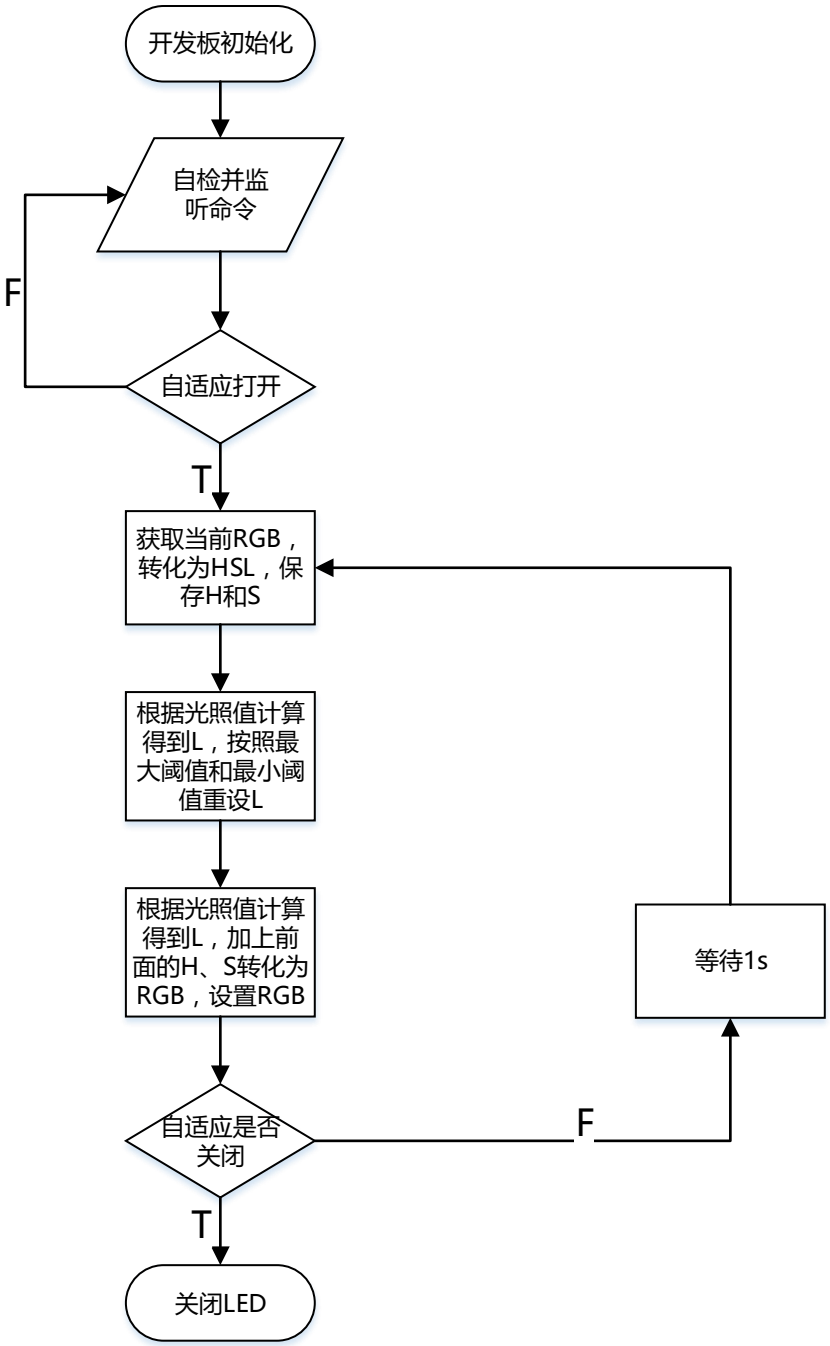

图 3.2　自适应模式流程图

3.4 睡眠模块

大家可能都经历过晚上想要起来上厕所，然后在黑暗中搜寻灯光开关的痛苦经历吧。纵然现在家居设计都设计了床头灯，亦或者是将开关放在床头便于搜索的位置，甚至还有设计于晚上一直开着的走廊灯或者睡眠灯。但是这些设计或多或少还是有些缺陷，比如床头灯和开关，还是要在黑暗中或者迷糊中摸索开关，一不小心可能就把手机什么的碰到地上。还有晚上一直开着的走廊灯或者睡眠灯，对于浅睡眠、入睡难和对灯光敏感的人又是一种折磨。所以本文设计了睡眠模式，当进入晚上的时候，如果检测到

声音，则厕所灯打开，为用户的夜间行为提供方便。

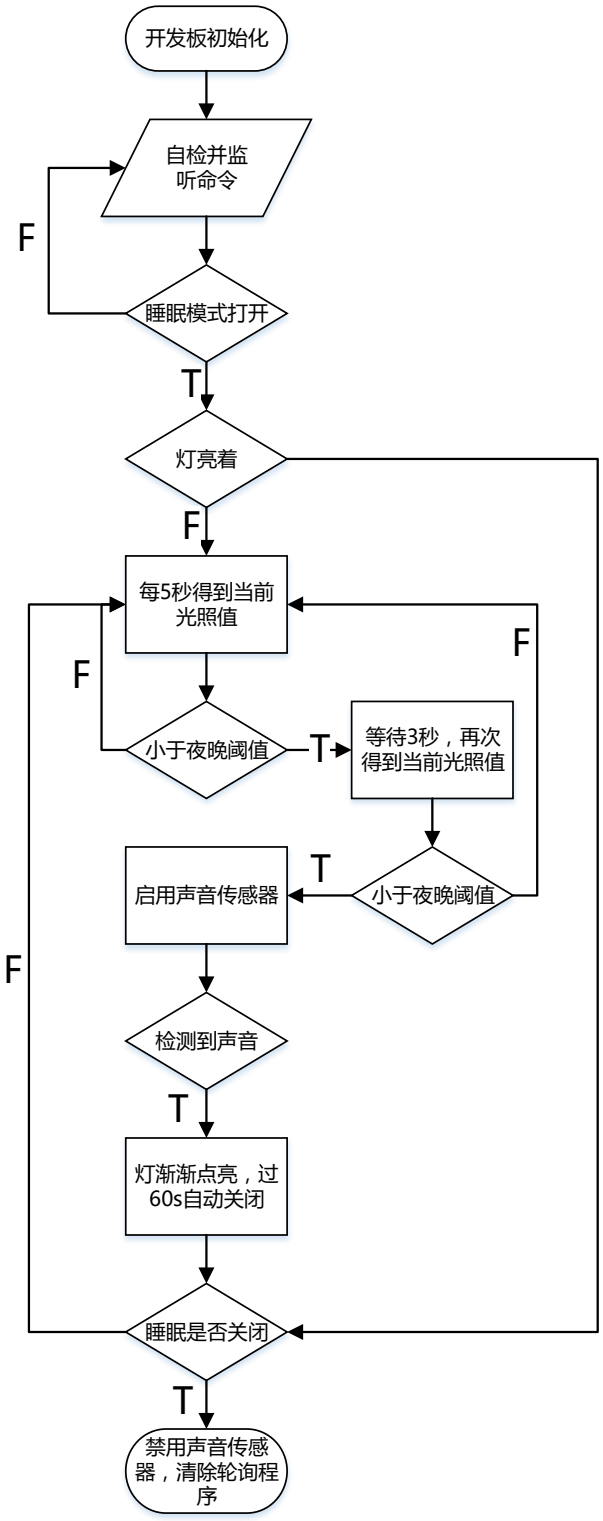

图 3.3　睡眠模式流程图

睡眠模式设计思路，图 3.3 是睡眠模式流程图：

步骤1. 当 Ruff 开发板初始化时，进行自检并监听命令。

步骤2. 如果睡眠模式已打开或者接收到远程命令，则进入睡眠程序。

步骤3. 判断灯是否亮着，如果灯亮着，则原地等待。

步骤4. 如果灯是关着的，每 5 秒获取一次当前光照值，如果当前光照值小于我们设定的夜间阈值的话，则等待 3 秒以后，再次判断当前光照值是否小于夜间阈值。这是一种类似按键防抖的思想，很可能一次阴影或者遮挡，导致了这次的误判。

步骤5. 等确认到了晚上，打开声音传感器，这里绑定一次侦听事件。如果听到声音，则利用循环逐渐点亮灯，过了 60 秒自动关闭。智能家居的主要目的就是为了提高用户的舒适性，所以要时时刻刻为用户考虑。

步骤6. 再判断睡眠模式是否依然打开，如果打开，则重复步骤 3~5，否则禁用声音传感器，清除轮询程序。

这个模块还有一些 Bug，比如在检测声音的时候，也应该作类似与按键防抖类似的措施，但是因为声音传感器事件绑定的特殊性以及 JavaScript 语言的回调特性，导致暂时没有想到好的解决方案。还有因为每 5 秒的轮询事件是一直在进行的，可能会重复多次给声音传感器绑定侦听事件，因此我们要设置一个 nightFlag，用来判断是否已经进入夜晚。即使不给声音传感器绑定侦听事件的时候，它还是在不断获取外部声音信息，本文利用 enable 和 disable 方法来开启关闭此传感器。

3.5 监控模块

当用户不在家的时候，希望能远程监控家里的情况，类似于一种安防报警系统。如果家里进了小偷或者侵犯者，可以发送消息到手机上。其实，如果只是检测人的话，只需要一个人体红外传感器就够了，但是如何让Ruff开发板发送信息到手机上呢？毫无疑问，Ruff开发板要先连接到公网。然后要发送信息到手机上，这又是一个难点，这里本文借助一个服务——Server酱，这是一个从服务器推报警和日志到手机的工具。本文利用node.js的http库向所绑定的微信号发送信息，但是我们在微信上要先绑定这个公众号，然后会得到一个SCKEY，我们利用这个SCKEY来辨识服务器发的信息。

监控模式设计思路，图 3.4 是监控模式流程图：

步骤1. 当 Ruff 开发板初始化时，进行自检并监听命令。

步骤2. 如果监控模式已打开或者接收到远程命令，则打开人体传感器。

步骤3. 当有人在传感范围内活动时，推送消息到微信，无源蜂鸣器发出刺耳鸣叫，LED 灯在红色和蓝色之间不间断地闪烁。

步骤4. 然后判断监控是否关闭，如果关闭的话，那么清除人体传感器上的侦听事件，关闭 LED 灯，否则等待 60 秒再关闭 LED 灯，防止用户忘记关灯。

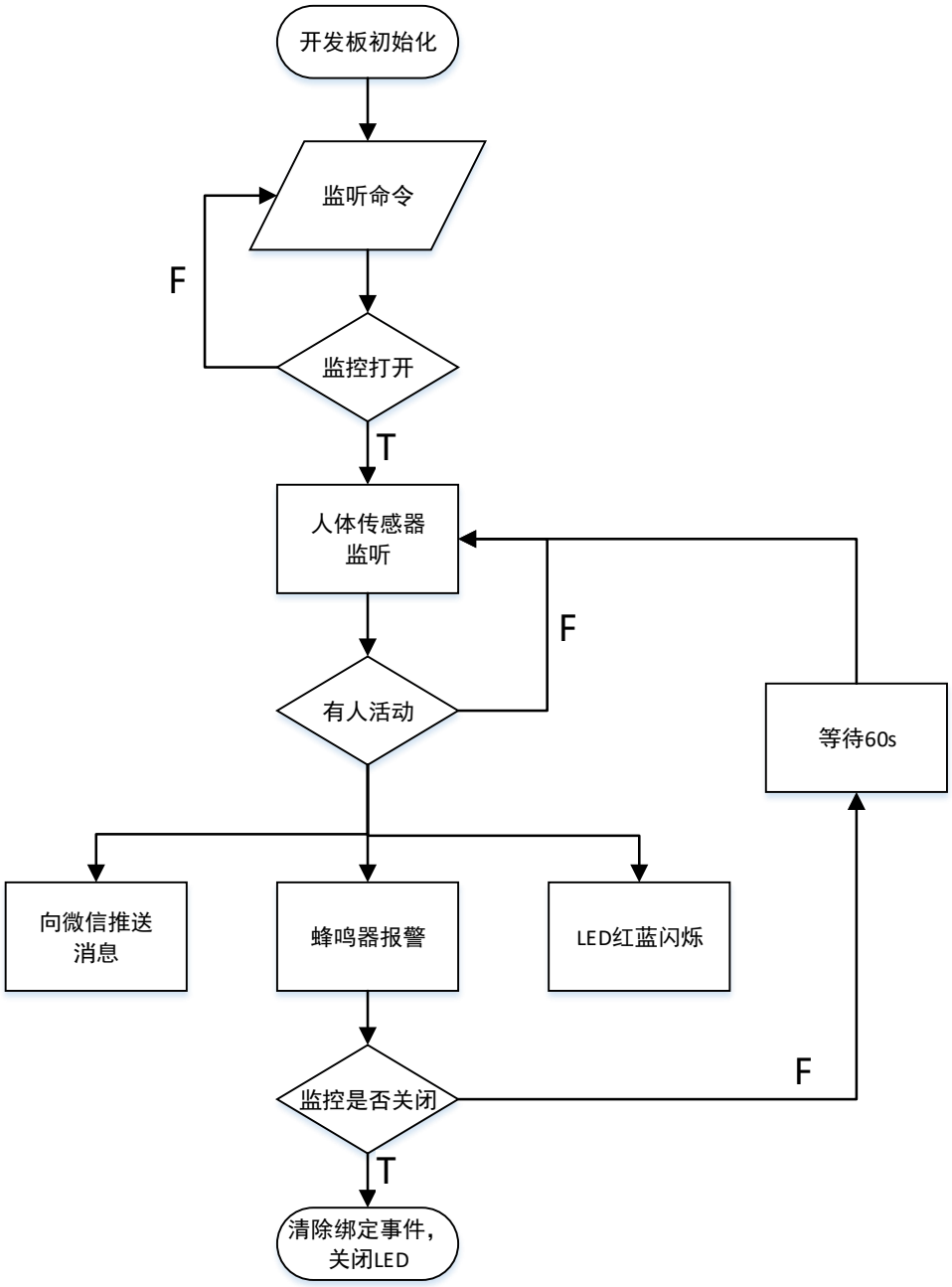

图 3.4　监控模式流程图

3.6 心情颜色模块

众所周知，颜色对人的心情有着隐式的影响，而且有色彩心理学这门学科。理所应当地，智能家居也应该有可以改变灯光颜色的功能，根据不同的场景和心情[10]，用户可以选择不同的灯光或者自定义颜色。色彩为生活带来的感官刺激，逐渐被视为理所当然。

网上有一款吸色灯叫做 ColorUp[11]，它充分利用色彩对感官的刺激所带来的互动体验，让原本单纯照明用的灯光有了意义，也为生活增添更多火花。比如浅蓝色代表 SAFE，红色代表 EXCITED，浅绿色代表 FRESH，粉色代表 LOVE。一盏桌灯运用它的神奇力量，将隐藏于对象中的颜色逐一释放，成为妆点环境的空间魔法师。

3.7 服务器监听模块

本文使用手机远程控制 Ruff 开发板，控制的方式有两种。一种方式是手机连入 Ruff 开发板的无线局域网下，这种方式称为 AP 模式；还有一种方式是手机和 Ruff 开发板都连在同一个局域网下，这种方式称为局域网模式。AP 模式的好处是，相对手机而言，Ruff 开发板的 ip 地址是固定的，但是 Ruff 开发板的无线网可能会不稳定。以前使用 AP 模式的时候，手机是上不了公网的，现在 Ruff 官方添加了网络渗透功能，只要 Ruff 开发板连入了公网，其他连入 Ruff 开发板无线网的设备都可以连入公网。而局域网模式就相对常见一点，只是 Ruff 开发板的 ip 相对会变，不过可以通过绑定 MAC 地址或者一直让 Ruff 开发板连接在局域网上不下线来解决，正式环境中局域网模式更加稳定些。图 3.5 是 AP 模式和局域网模式，图中服务器为 Ruff 板。

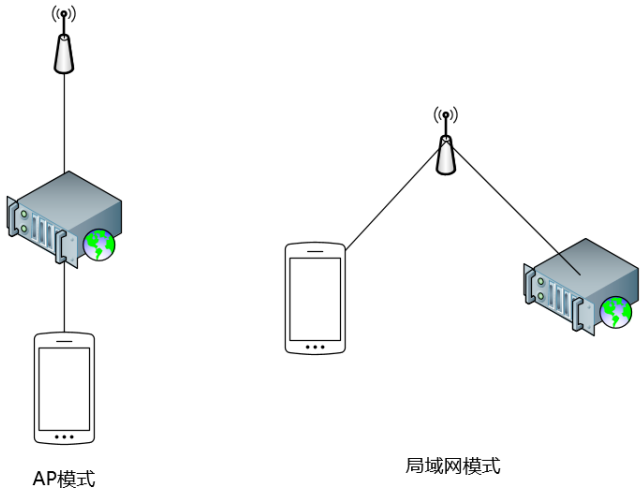

图 3.5　AP 模式和局域网模式

本文为方便调试，采用 AP 模式，即手机连接到 Ruff 的无线网，Ruff 连接到路由器上，此时 Ruff 开发板的 ip 固定为 192.168.78.1。此时，控制页面在手机端，那么手机就相当于一个客户端，Ruff 开发板相当于一个服务器端。整个控制过程可以分为以下几步，如图 3.6 所示：

步骤1. 客户端要访问 http://192.168.78.1(Ruff 开发板 ip):6318/modeStatus 这个地址，首先要解析这个 url，url 包括协议、域名、端口、路径。

步骤2. 将 url 中非 ASCII 的 Unicode 字符转换，此处并没有 Unicode 字符。

步骤3. 进行 DNS 解析，将域名转化为 ip 地址，由于这边本身就是用的 ip 地址，不需要再解析了。

步骤4. 进行 ARP 广播，在缓存和路由表中找到 ip 为 192.168.78.1 的设备。

步骤5. 请求在运输层被封装成 TGP 段，目标端口加入头部。

步骤6. TCP 段被送往网络层，加入一个 ip 头部，包含目标服务器以及本机 ip 地址，封装成一个 TCP 数据包。

步骤7. TCP 数据包进入链路层，加入 frame 头部，包含本地内置网卡的 MAC 地址和网关的 MAC 地址，然后通过 WiFi 传输。

步骤8. 服务器接收到这个请求，解析包得到路径，接着服务器会使用指定的处理程序分析处理所请求的文件，并把输出结果返回给请求者。传输过程与前几个步骤类似。

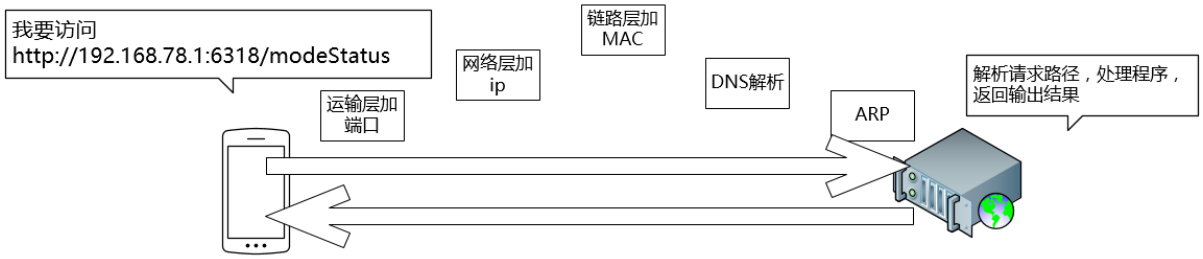

图 3.6　远程控制流程图

那么需要在 Ruff 开发板上处理客户端的请求，针对不同的请求写特定的处理程序和特定的返回结果。浏览器以及 node.js 已经帮助做了很多封装工作，本文使用 node.js 的 http 库来处理收发信息，node.js 提供了一套完整高效的 API。本文所使用到的请求没有携带大量数据，所以仅仅是 GET 请求就可以满足使用。GET 请求的意思是将数据或者参数编码放入 url 中，然后服务器用某种方法解析出这些参数。Ruff 官方封装了一个模块，更是高度集成了 node.js 的 API，让使用者可以更加专注于逻辑控制。

3.7.1 同源跨域

客户端和服务器端发送接收的数据格式一般是 json 格式的，json 是一种非常轻量级的数据交换格式。客户端和服务器端通过 json 收发数据的这种模式叫 Restful API，在前后端分离的项目中比较常见。这种方式要解决的第一个问题就是同源跨域问题。

浏览器有同源策略，它限制两个不同源的文档和脚本资源进行交互，这是为了防止 CSRF(Cross-site request forgery)攻击，中文名是跨站请求伪造。CSRF 攻击的思想是盗用访问者的身份，比如登陆某些网站以后得到的 cookie，利用访问者的名义发送恶意请求。比如可以以访问者的名义发邮件，盗取账号，购买商品等。所以浏览器利用同源策略来避免此类问题，但是这在一定程度上造成了页面获取资源的困难。官方定义只有当协议，域名，端口相同的时候，才算同一个源，否则就是跨域。

关于同源策略有一个误区，它不是禁止浏览器发送请求，服务器是可以正常接收到请求的，只是服务器返回的数据被浏览器阻止了，并在控制台输出一系列错误信息。

常见的跨域方案有 document.domain、window.name、jsonp、HTML5 的

postMessage、CORS。document.domain 只适用于父子 window 之间的通信，xhr 无法使用；window.name 最大数据量只有 2M 左右，并且只能传递字符串；jsonp 通过动态 script 标签来做跨域请求，但是只能以 GET 方式，并且需要前后端联调；HTML5 的 postMessage 方法在 PC 端有浏览器兼容问题，需要作降级处理，但是不需要后端介入，有一定的安全问题，且无法做到一对一的传递；CORS 是通过前后端配置 http 的 header 来跨域的，在 PC 端的 IE 和移动端的 Opera 有一定的兼容性问题，但使用方便，服务器端只要在返回的 response 上添加一条 header 就行。比如 Access-Control-Allow-Origin: origin，这边的 origin 的意思是允许来自哪些网站的请求，虽然设置为*，也就是所有来源比较方便，但存在一定的安全问题，建议在开发环境中使用。

本文使用最后一种方法，即 CORS 方式进行跨域。

第4章 远程控制界面设计

本章会首先介绍 HTML 和 CSS 语言，然后简单介绍一些用到的在前端领域比较热门的工具，最后介绍页面布局、Scss 设计和 JavaScript 设计。

4.1 HTML

HTML，HyperText Markup Language，超文本标记语言。它是标准通用标记语言下的一个应用，也是一种规范[12]。平常我们所见到的网页，本质上就是 HTML。HTML 的基本结构包含头部内容和主体内容，头部内容一般包含页面的标题、序言、说明等内容，主体内容包含了许多标签元素。其主要特点是：简易、通用、平台无关。

4.2 CSS

CSS, Cascading Style Sheets，层叠样式表，是一种用来表现 HTML 等文件样式的计算机语言[13]。CSS 能对网页中的元素的排版进行像素级精确控制，并且可以修改元素的字体、颜色、样式等。它主要是用来修饰静态网页，解决 HTML 杂乱和臃肿的排版问题。一般 HTML、CSS 和 JavaScript 分别对应一个网页的结构、样式和行为。

4.3 工具

本文使用到了前端资源模块化管理和打包工具 Webpack[14]，它可以根据特定的依赖或者规则，将不同的松散的模块打包，这样在生产环境中可以自动化处理依赖。不仅如此，webpack 还可以使代码按需加载，只要写法符合特定规则，则可以在需要的时候，使代码异步加载。在 Webpack 中，一切皆模块。

本文使用的 JavaScript 版本为 ECMAScript 6，通过 Webpack 的插件 babel-loader 来转换成各个浏览器都支持的 ECMAScript 5 版本。ECMAScript 是 JavaScript 的一种规范，2011 年，ECMAScript 5.1 版本发布，现在几乎所有浏览器都支持 ECMAScript 5.1 标准。标准委员会规定，标准在每年的 6 月份正式发布一次，不再有什么版本序号什么的，类似于 HTML5 的概念。也就是说 ES6 是一个泛指，含义是 5.1 版本以后的 JavaScript 的下一代标准，它的第一个版本，在 2015 年 6 月发布。在 ES6 中，对 JavaScript 的基础类型做了扩展，并新增了许多语法糖[15]。

本文使用了 CSS 预编译工具 Sass，Syntatically Awesome Style Sheets[16]。因为 CSS 简单的语句结构以及局部变量的需求，这个成熟、稳定、强大的专业 CSS 扩展语言诞生了。Sass 支持变量，嵌套，计算，代码的复用，甚至可以使用条件语句和自定义函数，是非常强大的语言。Sass 是由 Ruby 语言写的，在 windows 系统上安装可能会出现问题。

以前网页设计中，所有的 icon 都是图片格式的，这导致了许多问题。比如，每一个

小 icon 都是一次 http 请求，如果一个页面上有 5，6 个 icon 的话，页面速度会明显下降。而且 icon 的图片也不好管理，要更换 icon，只能更换图片地址，图片的命名等是相当繁琐的。由于图片元素的 inline-block 特殊性，导致 icon 在页面中的排版布局比较困难，即使在移动端可以使用 flexbox 布局，但是在 pc 端兼容性问题就比较大了。

一项革命性的技术问世了，如果能够用字体代替这些 icon 的话，就只需要加载一次就够了。关键问题在于，是否有某种字体，经过某种样式修改，能够画出 icon 的样子，答案显然是可以的，而且 iconfont 是矢量化图形，跟 svg 一样不受像素影响，可以不失真地方法缩小。比较有名的 iconfont 有 fontawesome 和阿里的 iconfont 库，本文所使用的 icon 都来自于阿里的 iconfont。

4.4 页面

本次移动端页面包含 3 个页面，分别是控制台主页、关于页面和颜色修改页面。

主页面有模式控制，包含自适应模式、睡眠模式、监控模式，分别对应 Ruff 设计的 3 个模式，在开关方面借鉴了系统的 switch 开关样式。Switch 开关主要用到了 CSS3 的 transform 属性和 transition 属性，移动中间的白色滑块。然后还有 IP 修改，包含了一个输入框和确认按钮。最下面是颜色修改，点击跳转到颜色修改页面。

颜色修改页面包含一些常用的颜色以及可以自定义的输入框。

关于页面是一些工具介绍，以及一些参考资料。

图 4.1 是控制台主页面图，图 4.2 是颜色修改页面图，图 4.3 是关于页面图。

4.5 Scss 设计

因为浏览器会默认设置一些元素的样式，而各个浏览器或者移动端的默认样式又不一样，所以本文需要首先重置统一每个元素的样式，使得这些元素在所有端看到的样式是一样的。由于 CSS 是能够复用的，本文尽可能将每行 CSS 语句的利用率提升到最大，并且因为 CSS 的全局变量属性，防止各模块冲突也需要考虑。

比如按钮的样式是通用的，除了每个按钮的颜色不一样，那么本文定义一个通用的 class 为 btn，然后再定义一个 class 为 btn-red，只有一条属性即背景颜色为红色。那么一个红色按钮元素的 class 就是 btn btn-red。还有全局通用的就是每个页面的 header 部分、底部的固定的 nav 部分、checkbox 部分、icon 部分，这些部分每个都是一个单独的模块。而比如 switch 模块可以直接写在 HTML 页面中，如果只有一个页面有的话。

图 4.1　控制台主页面图

图 4.2　颜色修改页面图

图 4.3　关于页面图

因为本文所写页面是移动端页面，所以要保证布局和字体大小要自适应设备的大小。本文使用了比较先进的 flexbox，在移动端没有兼容性问题，这个可以在 can I use 网站上查到。flexbox 是一种为了处理不同屏幕尺寸和不同显示方式而专门设计的布局模式，它并没有使用到 float，所以不用清除浮动，而且盒子的 margin 没有被内容的 margin 所干涉到。flexbox 可以很好地处理垂直居中以及在一个盒子中兄弟元素之间的间距，而 float 和 position 这些方式都很丑陋。

要让字体大小自适应的意思就是，无论使用者的手持设备屏幕或者分辨率是多少，使用者看到的字体大小应该一致，也就是说，屏幕越大时字体应该相应的变大。在前端行业中，比较通用的方法是这样的。设置 html 根元素的 font-size 比如为 62.5%，然后其他单位一律用 rem，这里的 rem 单位就是根据根元素的 font-size 大小变化。html 默认 font-size 为 16px，62.5%即 10px，在 body 设置 font-size 为 1.6rem，则又让字体默认大小回到了 16px。CSS3 中又推出了新的大小单位 vw、vh，也就是设备的宽度和高度，但是在实际使用中和%单位一样存在问题。

4.6 JavaScript 设计

本文使用前端热门框架 JQuery，可以有效的避免大量重复性的 DOM 操作并且简化 ajax 操作。JQuery 库的思想是，Write Less, Do More。

因为局域网模式下 ip 是会改变的，所以本文预留了一个修改 ip 的接口。ip 是每个页面都要用的，所以将 ip 这个变量放到 LocalStorage 中，LocalStorage 是存储在本地的，而且不像 sessionStorage 里的数据会在浏览器会话结束时被清除，当然必须要在相同的协议、主机名和端口下才能读取到里面的数据。

4.6.1 正则

有时候用户会输入格式错误的 ip 地址，或者利用某些漏洞恶意输入某些脚本，这虽然可以通过后端来过滤控制，但前端同样可以也应该保证没有安全问题。必须要让用户输入的字符串符合 xx.xx.xx.xx 的模式，才修改 ip 地址，否则不予通过。显然，匹配字符串模式要用到正则。

正则隶属于计算机科学的概念，是用来匹配字符串特定规则的表达式，常用来检索或者替换某个模式的文本，一般叫 re，regular expression。正则表达式里通过元字符来表达某种规则。比如“\d+”，\d 匹配数字，+表示前面这个元字符重复一次或更多次。那么就可以匹配到 1、12、122 这种模式的字符串。当然正则表达式不会这么简单，还包括比如字符转义、分支条件、反义、断言、贪婪懒惰等功能。

本文所需使用的 ip 模式匹配比较简单，只有数字和.，正则表达式为 /^([0 − 9]{1,3}\.){3}[0 − 9]{1,3}$/ 。^表示匹配字符串的开始，$表示匹配字符串的结束，[0-9]的意思与\d一致，{1,3}表示前面的元字符可能会出现 1 到 3 次。

第5章 系统功能实验与分析

本章会对自适应灯光、睡眠、监控和心情颜色 4 个功能模块进行实验，并总结心得。首先，根据 Ruff 操作系统生成的硬件连接图，上面明确标明了每个引脚的连线，当然用户也可以手动修改对应的连线，然后用杜邦线连接引脚。接着我们在 raff 根目录打开 cmd 窗口，输入命令，rap deploy –s，将编写好的程序烧入，它会自动将 index.js 编译成字节码烧入 Ruff 开发板中。然后连入 Ruff 开发板的 wifi 中，访问网页控制台，将 Ruff 开发板连接到其他网络热点上。这时 Ruff 会先关闭自己的 wifi，在连上热点并稳定后，会重新开启 wifi，整个过程有 20~30 秒左右。图 5.1 是硬件连线图，图 5.2 是硬件实物图。

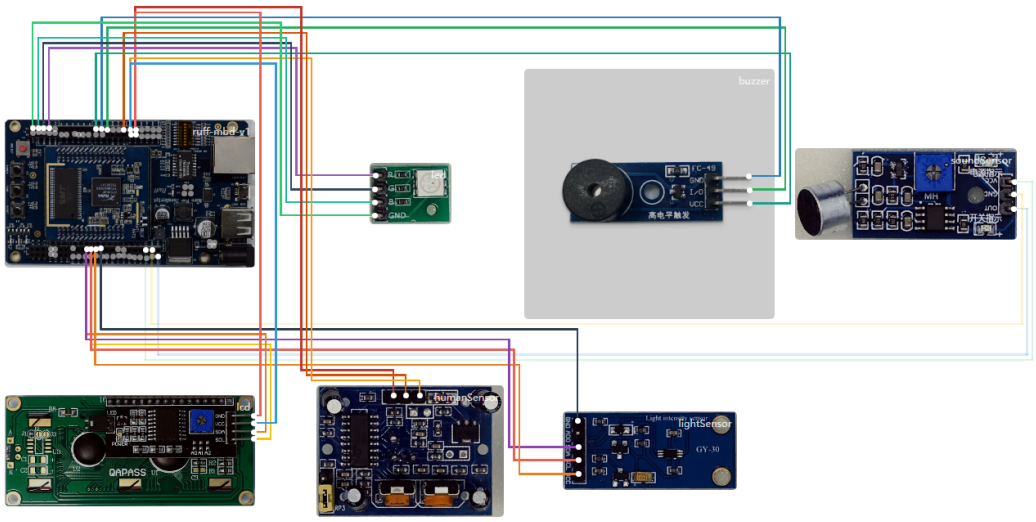

图 5.1　硬件连线图

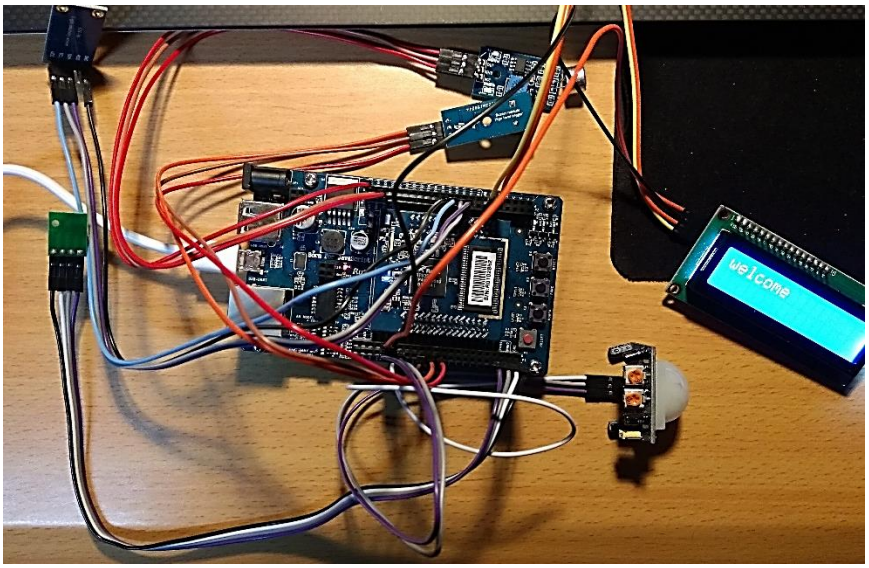

图 5.2　硬件实物图

5.1 自适应灯光模式实验

首先输入命令 rap log，记录调试过程。然后手机端开启自适应灯光模式，LCD 液晶屏

上显示 responsive on，调试台输出提示信息，随后不断输出亮度， LED 亮度随着当前照度变化。当关闭自适应灯光模式时，控制台和 LCD 屏显示 responsive off。图 5.3 为自适应灯光模式调试图。

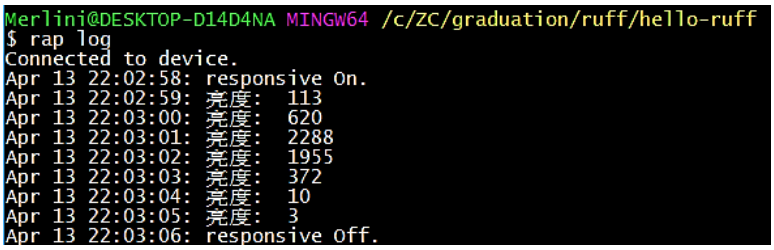

图 5.3　自适应灯光模式调试图

5.2 睡眠模式实验

首先输入命令 rap log，记录调试过程。然后开启睡眠模式，调试台输出提示信息，LCD 也显示 sleep on，随后检测光照度，判断是否到夜晚，如果是夜晚等待 3 秒再次确认，然后我们发出声音，被声音传感器接收到，LED 逐渐点亮。当关闭睡眠模式时，控制台和 LCD 屏显示 sleep off。图 5.4 是睡眠模式调试图。

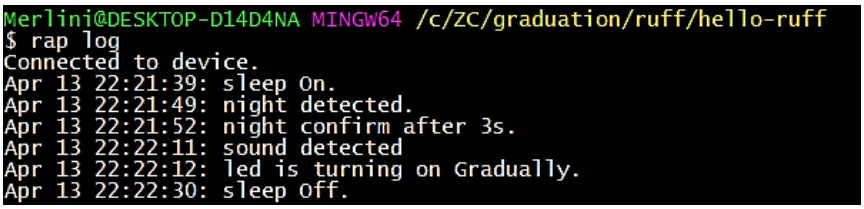

图 5.4　睡眠模式调试图

5.3 监控模式实验

首先输入命令 rap log，记录调试过程。然后开启监控模式，调试台输出提示信息，LCD 也显示 monitor on，在人体红外传感器上晃动手，LED 红蓝闪烁，蜂鸣器鸣叫，并且推送消息到手机端。消息标题为“家中有人请注意”，内容为当前日期。当关闭睡眠模式时，控制台和 LCD 屏显示 monitor off。图 5.5 为监控模式调试图，图 5.7 为微信接收图。

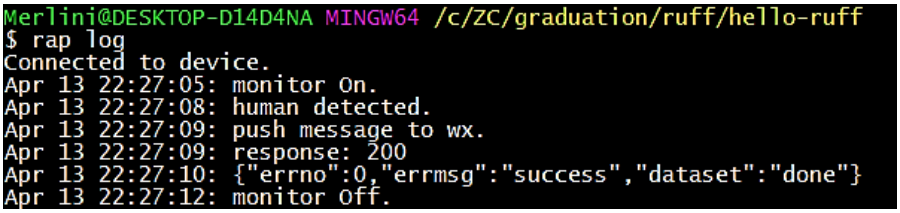

图 5.5　监控模式调试图

5.4 心情颜色模式实验

首先输入命令 rap log，记录调试过程。然后选择天蓝色，调试台显示颜色修改的代码，LCD 现在 color Change。图 5.6 为心情颜色调试图。

```
Merlini@DESKTOP-D14D4NA MINGW64 /c/ZC/graduation/ruff/hello-ruff
$ rap log
Connected to device.
Apr 13 22:42:57: shut down.
Apr 13 22:43:09: Console server bound.
Apr 13 22:43:15: color Change: #87cefa
```

图 5.6　心情颜色调试图

图 5.7　微信接收消息图

这 4 个功能模块通过了正常的模拟测试，完成了基本的功能。不过途中也有些小意外，比如网络连接不稳定，偶尔会断掉。还有就是人体红外传感器检测不是很准确，因为传感模块在通电后需要 1 分钟左右的初始化时间，并且风和灯光也会对其造成影响。

结　论

本次室内灯控系统，是 JavaScript 在物联网中的一次小试牛刀，也是我第一次使用我熟悉和喜欢的语言来开发软硬件结合的工程。随着这个项目从设计到完成，我的角色由产品经理转向开发又转向测试。虽然说独立开发很爽，可以自由地使用喜欢的工具，但是可能在用户需求上的考虑欠妥，因为有些功能可能只是自我臆测而不是用户所需要的。

在制作手机控制页面的时候，在初期选型上耗费的时间太多，一直想用新技术来做，比如 React Native，没有考虑到沉重的学习成本。虽然最后还是尝试了很多新技术，毕竟得到提升才是关键，但是这提醒了我在选型和时间上要找到一个平衡点。而不同的框架要适应不同的场景，比如这次 App 设计，其实只有 3 个页面，没有大量数据交互，基本不用上框架的。最后使用了打包工具 webpack、ES6、Scss 也很好地完成了任务。

在编写 Ruff 程序的时候，也得到了许多提升，比如说对 node.js 和 http 的使用更加熟悉，对 Restful API 和服务器端有了较深的了解。然后就是对全局的把控，全局变量和局部变量的设计，IIFE 的理解。同时还一度遇到了 JavaScript 的“回调深渊”，还好以前就了解 Promise 的设计理念，重构了代码以后，结构很清晰。同时在注释方面也多加注意，方便以后熟悉代码。

在遇到问题的时候，要多加利用搜索引擎，要学会使用英文关键词，要学会科学上网，同时还要善于利用 QQ 技术群，里面可能会有技术大神解答你的问题，但是你要首先能描述好你的问题。作为一个大学生，尤其是在技术型的专业，我们要有流畅阅读英文文档和文献的能力，不然技术是很难提升的，同时应该多在开源社区活跃。

本次毕业设计，可以说将我所学的都杂糅到一起，完整地运用出来了。即使它在功能或者是需求上有些小问题，依旧不失为 JavaScript 在物联网的一次成功应用，且探索试用了一些前端前沿技术。