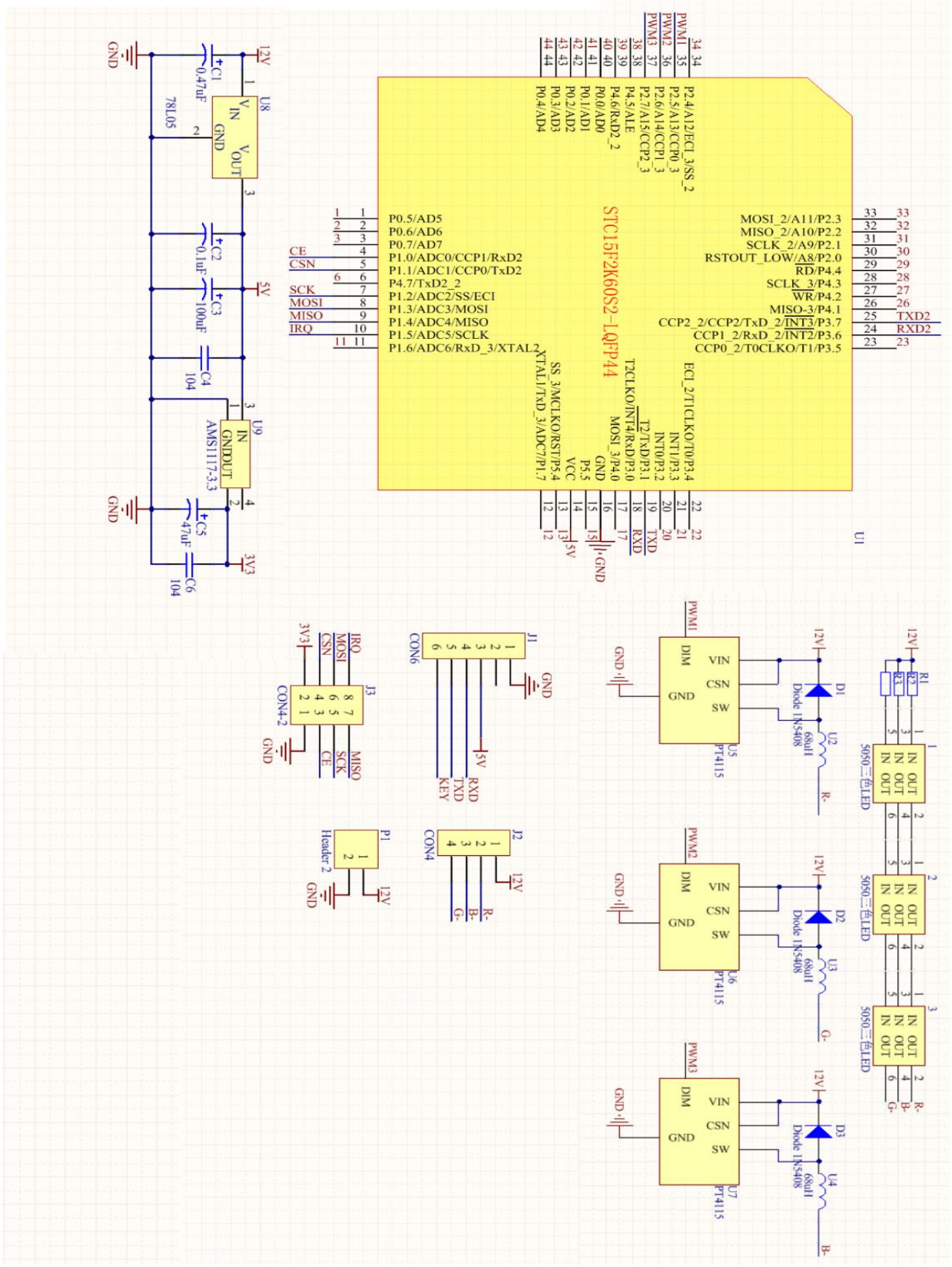
附录

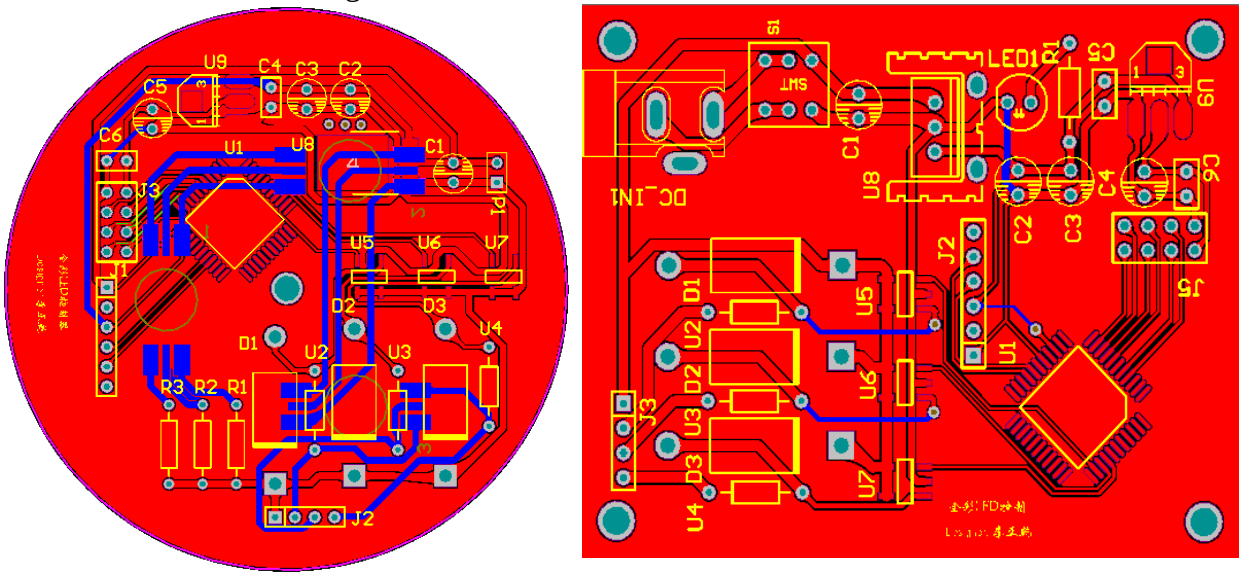
附录 1 系统原理图

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统原理图

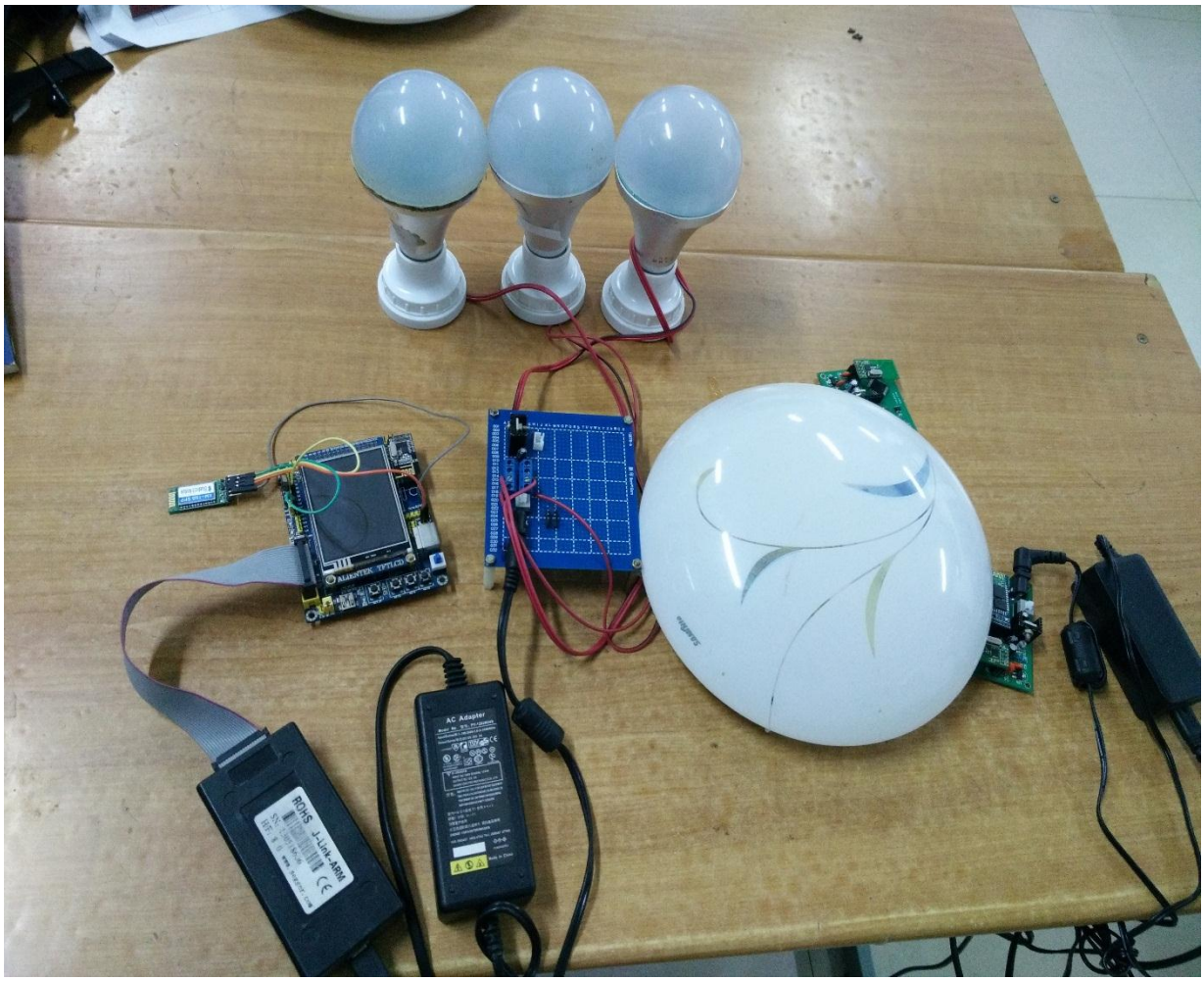


附录 2 系统 PCB 和实物

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统 PCB 图

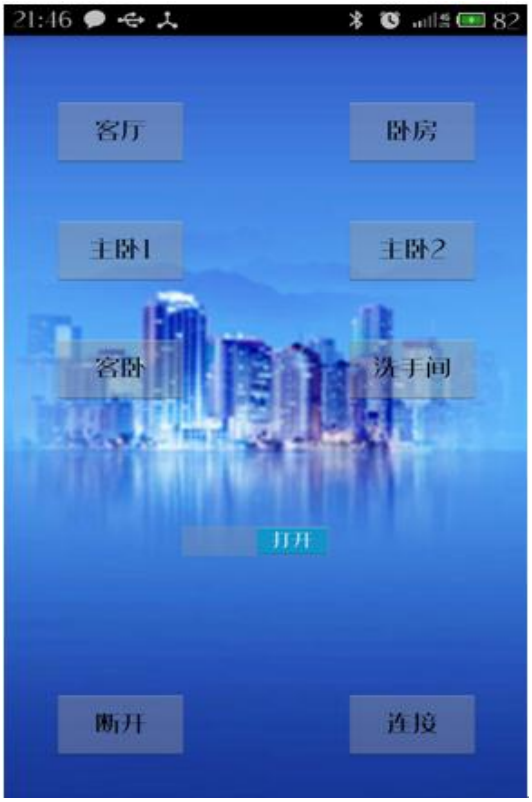


基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统实物图



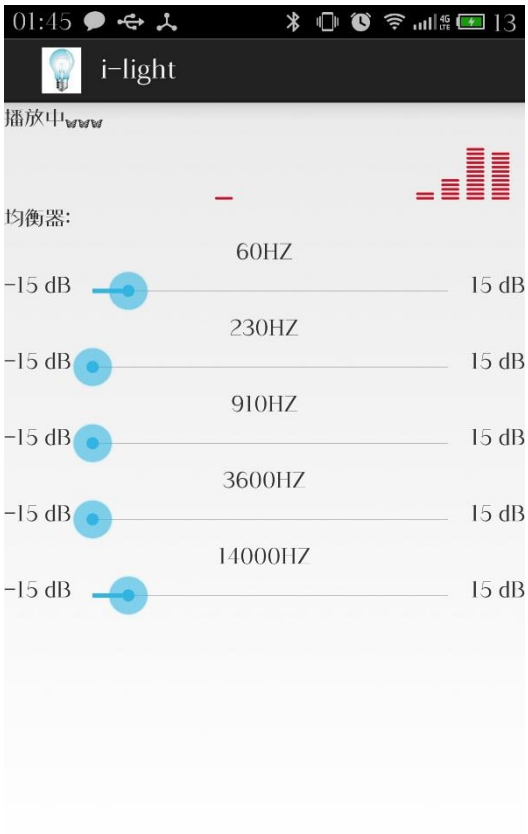
附录 3 手机端应用程序界面

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统手机应用程序界面

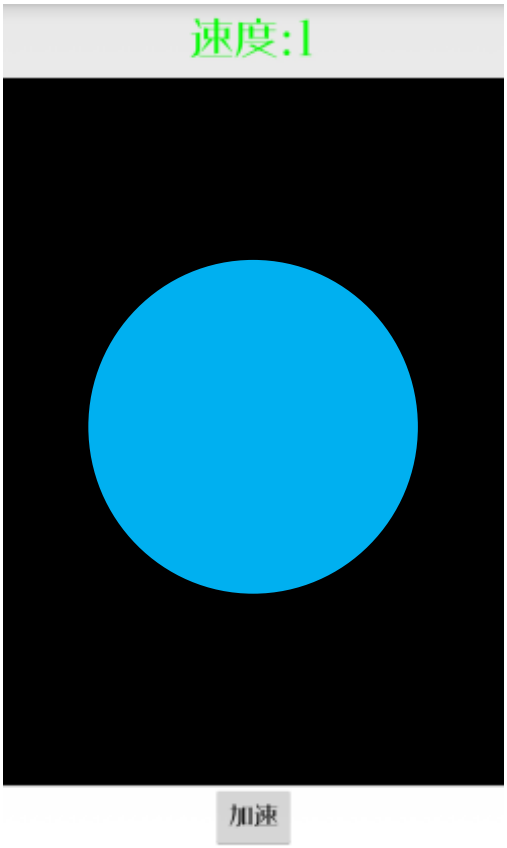


手机 APP 主界面

灯光状态调节界面



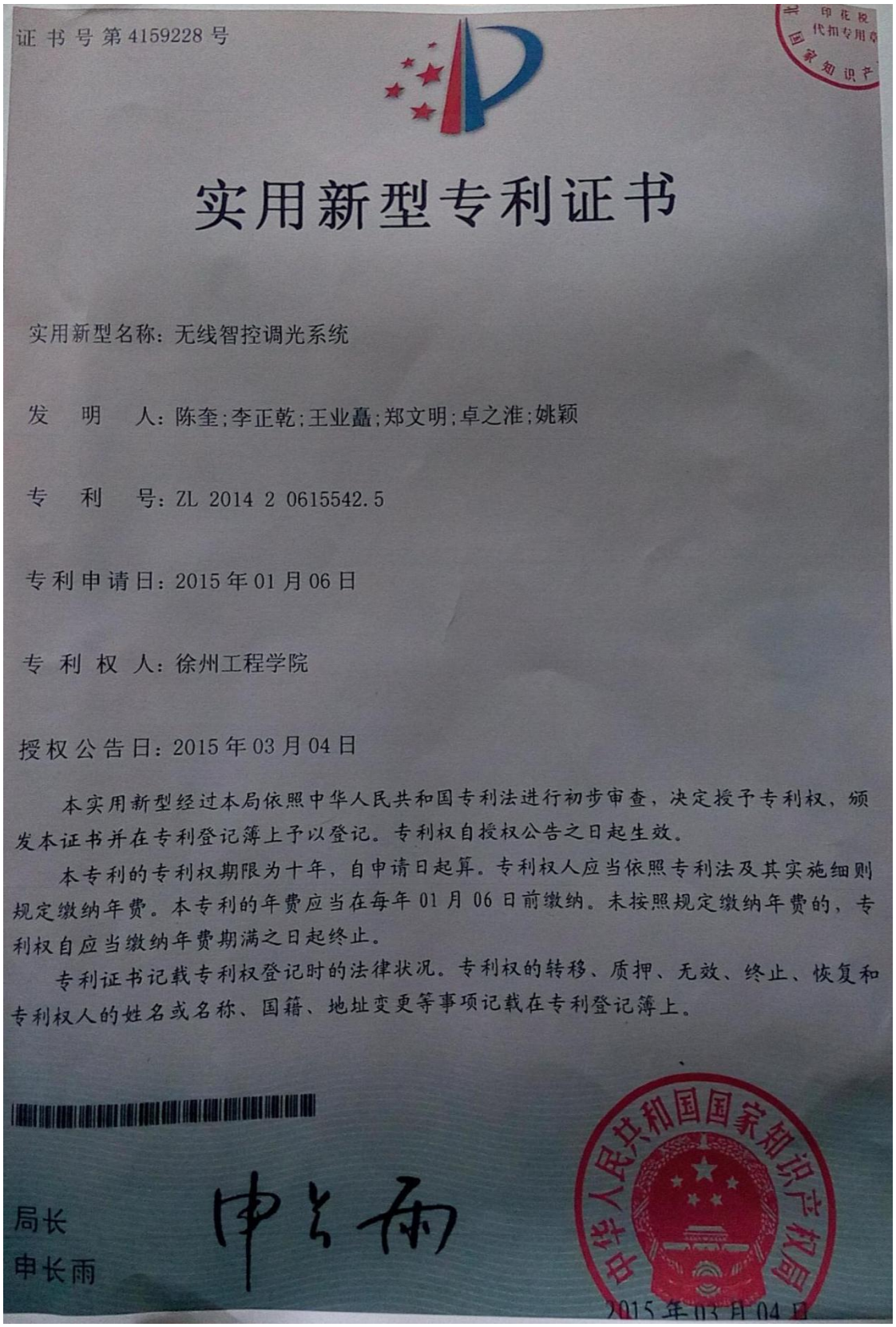
音乐频谱



七彩渐变

附录 4 专利证书

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统的实用新型专利



证书号第4159228号

实用新型专利证书

实用新型名称：无线智控调光系统

发　明　人：陈奎;李正乾;王业矗;郑文明;卓之淮;姚颖

专　利　号：ZL 2014 2 0615542.5

专利申请日：2015年01月06日

专利权人：徐州工程学院

授权公告日：2015年03月04日

本实用新型经过本局依照中华人民共和国专利法进行初步审查，决定授予专利权，颁发本证书并在专利登记簿上予以登记。专利权自授权公告之日起生效。

本专利的专利权期限为十年，自申请日起算。专利权人应当依照专利法及其实施细则规定缴纳年费。本专利的年费应当在每年01月06日前缴纳。未按照规定缴纳年费的，专利权自应当缴纳年费期满之日起终止。

专利证书记载专利权登记时的法律状况。专利权的转移、质押、无效、终止、恢复和专利权人的姓名或名称、国籍、地址变更等事项记载在专利登记簿上。

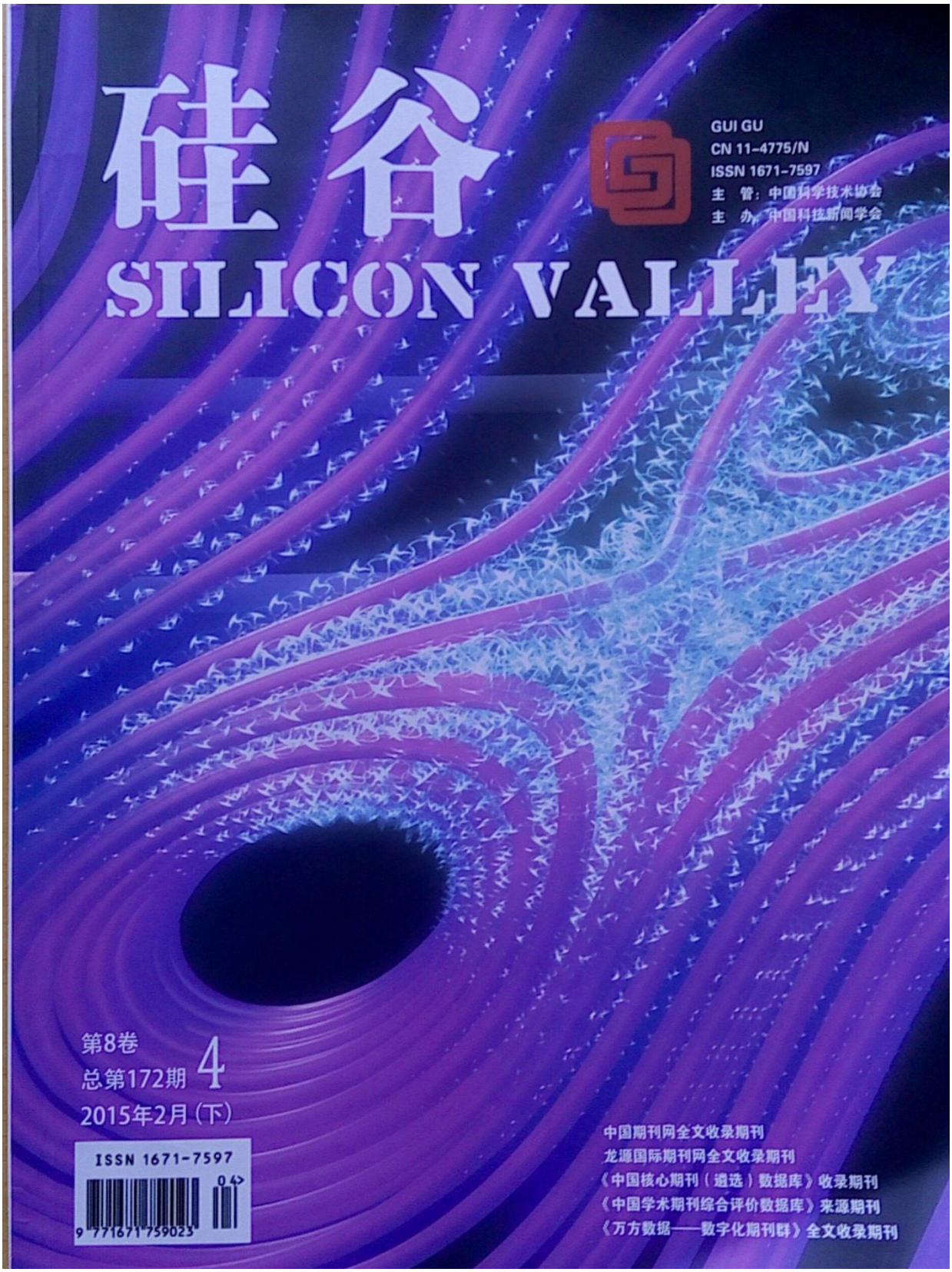
局长
申长雨

中华人民共和国国家知识产权局

2015年03月04日

附录 5 期刊论文

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统的期刊论文



基于Cortex™-M3的无线灯光智能开关的设计与实现

李正乾，王业矗，郑文明，陈 奎，潘晓博
（徐州工程学院信电工程学院，江苏徐州 221018）

摘 要 在家居智能系统中，开发出了一种基于Cortex™-M3嵌入式的无线灯光智能控制开关。它主要由人机界面和灯光控制两部分组成。它可以通过操作TFT-LCD液晶触摸屏终端，经由NRF2401无线模块发送操作命令至主控电路实现灯光的智能控制。通过无线智能控制方式，实现了对家庭照明系统的有效控制，解决了传统开关浪费资源，操作单一的问题，并且系统界面可以实时显示各照明设备的状态信息，便于用户直接操控，给用户带来了良好的智能化、人性化的体验。

关键词 嵌入式；智控；触摸屏；无线通信；PWM
中图分类号：TN929.5 文献标识码：A 文章编号：1671-7597（2015）04-0018-02

近年来，随着电子信息、物联网、LED智能照明等技术的发展，智能控制、绿色照明、个性化照明等产品、产业日益兴起。随着嵌入式技术的发展，将微处理芯片引入到电子墙壁开关中，使得具有各种不同功能的电子墙壁开关变得切实可行。目前白炽灯的时代已经过去，LED颠覆了传统，取代了白炽灯，结束了白炽灯一百多年的主导地位。然而对于LED灯而言，在拥有开关功能的同时，更为重要的是还需要调光、调色功能，引入更新的、更多的人性化功能，节能环保功能，将使电器使用更加简便，更加智能。LED的技术在照明行业的应用，形成了革命性的爆发，同时也对智能开关的发展提出了更高的要求。科技时代迅猛发展使得人们的生活方式和生活理念也随之改变，传统机械师墙壁开关已经不能满足人们的生活需求，而时下正在兴起的智能开关，功能特色多、使用安全，而且式样美观，打破了传统墙壁开关的开与关的单一作用，将对传统机械开关进行颠覆，取代传统机械开关是基于物联网智能家居发展的必然趋势[1]。

目前，市场上90%以上的开关都是手动操作元件，除一些少数要求高的、不方便控制的一般采用继电器控制、无线控制或与光电，声等传感器相结合进行控制。家装智能开关种类繁多，而且品牌仍不断增加，但绝大多数都功能单一。物联网近几年的飞速发展，使得具备个性化、智能化、便捷化的智能开关成为智能家居的必备品，其需求越来越高。

基于上述背景，本文将具有人性化、智能化的嵌入式技术和具有新颖、节能、低碳、可靠性高等特点的LED灯具相融合，设计了一种基于嵌入式的无线灯光智能开关，可以通过触摸屏交互界面根据环境需要来调节灯组的开、关，颜色和亮度，从而满足用户的不同需求。

1 系统总体设计

嵌入式无线灯光智控开关包括人机界面和灯光控制两部分，其中人机界面包括Cortex™-M3嵌入式最小系统、触摸屏驱动电路、NRF2401模块。灯光控制包括单片机最小系统、LED驱动电路、RBG三基色LED灯珠、NRF2401模块[2]。人机界面主要功能是通过触摸或点击触摸屏调色板及控件产生指令，经嵌入式处理器编码后由NRF2401发送。灯光控制主要功能是通过NRF2401接收人机界面编码后的指令，并经单片机处理器解码后控制产生相应的PWM，作为LED驱动电路的输入信号驱动LED开关及颜色、亮度变化。下图1为系统的整体框图。

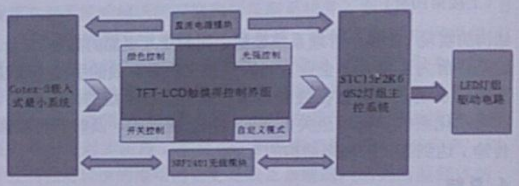


图1 无线灯光智控开关总体设计框图

整个系统的工作原理是：用户根据实际情况在触摸屏上选定所要实现的设备状态，经由Cortex™-M3所控制的无线模块发送相应的控制指令，灯光控制电路接收指令并译码，然后对驱动电路调用相应的处理程序，从而实现设备的开、关，颜色，亮度等的控制。

2 灯光控制电路设计

本系统是由单片机STC15F2K60S2解码指令后输出三路相应PWM作为LED驱动电路的输入信号来调节LED各路电流变化进而调节LED灯的RGB三基色，从而实现灯光颜色、亮度的调节[3]。灯光控制电路的系统框图如下图2所示。

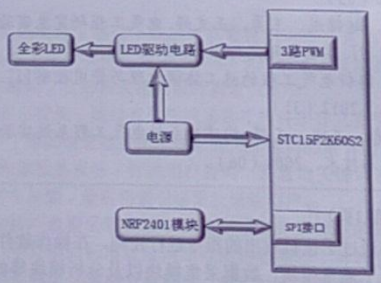


图2 灯光控制电路框图

2.1 单片机STC15F2K60S2

STC15F2K60S2是STC生产的一款高速、可靠、抗强干扰的新一代单片机，内置晶振及复位电路，减少最小系统的外围电路、PCB板面积及设计成本。另外此芯片资源丰富，功能强大，能稳定输出3路PWM，1组高速同步串行通信端口SPI，符合本设计要求。

本设计使用三路 PWM 为LED驱动电路提供PWM输入信号，通过SPI控制NRF2401无线通信模块的数据收发。

2.2 LED灯组驱动电路

目前LED驱动芯片按驱动方式可分为：恒压式驱动芯片、恒流式驱动芯片。恒压式驱动芯片通常应用在小功率的LED模组中。而恒流式驱动芯片通常应用在大功率的LED产品比如LED日光灯照明，替换卤钨灯的LED射灯等领域。根据LED特性以及控制方式，本文选用恒流式驱动芯片，通过改变输入信号来控制输入电流的改变进而控制LED颜色和亮度的变化。

本设计使用MT7201作为LED驱动芯片。MT7201是一款降压恒流源，其电压输入范围为7V-40V，输出电流可以通过控制sADJ引脚输入的PWM占空比而连续调节，并且最大最大输出功率可以达到三十多瓦[4]。其PWM调光的精度高，几乎不会出现模拟调光时LED颜色偏移的状态，调光范围宽泛从0～100%。

2.3 NRF2401无线通信模块

NRF2401是射频收发芯片，主要在2.4～2.5GHz ISM频段工作，芯片集成了频率合成、功率放大、晶体振荡以及调制等功能模块，可以通过程序配置输出功率以及通信信道。芯片能耗低，具有多种低功率工作模式，节能设计方便。NRF2401工作于多种无线通信系统，诸如无线键盘、智能玩具、遥控赛车等。NRF2401主要工作于以下四种模式：收发、配置、空闲以及关机模式，并且可以实现一对多及多对一通信。

3 人机界面设计

3.1 主控芯片：STM32F103ZET6

STM32F103ZET6是一款基于ARM的Cortex™-M3内核的微控制器是最新一代的嵌入式ARM处理器，外部晶振8M，内嵌锁相环（PLL），最高可提供72M的时钟频率。能够运行UCOSII实时操作系统以及uCGUI，实现人机交互界面。STM32F103ZET6功能丰富有优异的实时性能、杰出的功耗控制、良好的性价对比，易于开发，另外其外设及软件的高度兼容性，为开发者提供了全方位的灵活性。

3.2 触控电路模块

本设计采用TFT-LCD液晶触摸显示屏用来作为开关系统的主控元件。

TFT-LCD显示屏采用的是“背透式”的照射方式——即假使光源的路径自下而上。那么这样就是把特殊的光管嵌入到在液晶屏的后方，光源在照射时就可以通过下偏光板向上透出。通过把上下夹层的电极改为FET电极和共通电极，当FET电极导通时，液晶分子的状态必将会发生改变，那么可以利用遮光和透光来完成显示，又因为它对比度比TN-LCD更高，色彩也比它丰富得多，荧屏更新的频率也更快，因此通常把TFT叫做“真彩”[5]。

TFT-LCD触摸屏具有以下优点，性能好，可以应用于低压电路，使用安全性和可靠性高；可视化效果好，体积小，节约了大量资源和生产个；耗能低，它的耗能仅为CRT显示器的1/10，而反射式的TFT-LCD甚至不到CRT的百分之一；环保性能良好；零辐射、无闪烁，对用户的健康无损害。适用范围较宽，它可以在-20℃到+50℃的温度范围内正常使用，通过温度加固处理的TFT-LCD低温工作温度甚至可达到零下80℃。

在这里我们采用这款触摸屏显示及控制LED灯光颜色和亮度，触摸按键相对于传统的机械按键有寿命长、占用空间少、易于操作等诸多优点触摸屏、触摸按键大行其道，然而传统的机械按键，就要在人机交互界面中消失。

3.3 人机界面软件程序设计

无线灯光智控开关的程序设计流程图如下图3所示，其主要包括：

1）系统初始化：在系统初始化程序中，主要完成对各模块的启动处理，其中包括：Cortex™-M3系统的初始化、触摸屏进入欢迎界面、无线模块NRF2401启动。

2）检测系统状态：系统初始化以后，嵌入式系统开始检测无线模块，并且检测触摸屏的状态，一切正常后，等待进入系统启动状态。

3）启动任务：检测系统状态正常后，开始检测触摸屏是否有事件发生，即用户是否对触摸屏操作，如果有那么系统开始发送相应的指令到灯光控制模块，从而实现智能开关的功能，如果没有系统保持待机功能[6]。

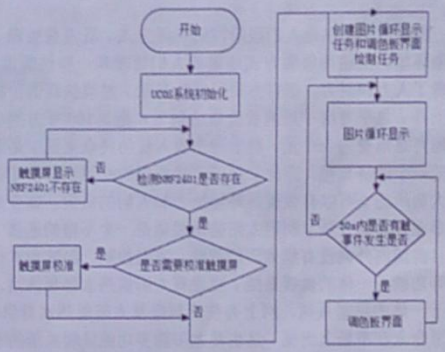


图3 人机界面程序设计流程图

4 结论

通过把物联网技术、嵌入式技术和节能、低碳、可靠性高等特点的LED照明相融合，应用与无线灯光智能开关，不仅可以实现灯光的智能化控制，能源的合理利用，而且在一定程度上改变了人们对传统照明控制方式理念的改变。本设计操作简便、体积小、可靠性高、便于实施，适用于广大家庭照明系统，具有较高的推广与应用价值。

项目基金

徐州工程学院大学生创业创新基金项目（190）；国家级创业实践项目（201411998085）。

参考文献

[1]韩伟，裴春梅，王艳秋，杨秀清.基于ARM技术的无线视频监控系统设计[J].现代电子技术，2013（20）：108-110.
[2]蒋峰.远程无线灯光智能控制开关的设计[J].中南林业科技大学学报，2011（11）：187-191.
[3]石建国，何惠龙，马云辉，张金锋.无线调控LED照明系统设计[J].现代计算机，2013（09）：58-60.
[4]杨军平，侯沛德，许存禄.基于嵌入式的无线手持控制系统的设计与实现[J].机械研究与应用，2014（2）：178-180.
[5]管来奇，吴闯.智能家居系统设计[J].电脑编程技巧与维护，2014（18）：77-79.
[6]杨军平.基于嵌入式的无线手持控制系统的设计与实现[D].甘肃，兰州大学，2014：23-30.

作者简介

李正乾（1991-），男，河南洛阳人，物联网助理工程师，学生，本科，研究方向：主要从事电子信息技术方面。


PRACTICAL ELECTRONICS
电子制作
总第264期 2014年8月(上)
中国期刊全文数据库(CJFD) 万方数据库-数字化期刊群 中国核心期刊(遴选)数据库 www.ele169.com
一款自卸车电子遥控升降系统的设计与制作
基于Atmega16的自主寻线避障智能车设计
基于LabVIEW的多普勒流量计的数据处理系统设计
一种宽频带单臂螺旋天线的设计
基于s3c6410的温室环境监控系统的设计
ISSN 1006-5059
国际刊号 ISSN1006-5059 国内刊号 CN11-3571/TN 邮发代号 82-541 定价:15元

基于蓝牙的无线智控全彩 LED 情景灯的设计

王业鑫 李正乾 徐州工程学院信电学院 徐州江苏 221018
赵鹏飞 徐州工程学院土木学院 徐州江苏 221018
刘 隽 徐州工程学院 经济学院 徐州江苏 221018

【文章摘要】
针对目前照明系统多以功能性为主,难以满足用户个性化、智能化的需求。本文设计一种通过手机界面无线智控全彩 LED 情景灯。采用 STC12C5A32S2 单片机、PT4115 驱动芯片、LED 灯组、蓝牙模块、手机 APP 界面实现一种情景照明系统,通过蓝牙发送指令到主控模块,由主控芯片来调节 PWM 电流,从而控制 RGB 实现 DIY 情景 LED 照明。经实验本设计可以灵活的实现 LED 亮度的无级调节,调色多达 100 万种,且系统操作方便,性价比高,具有较好的发展前景。

【关键词】
蓝牙;手机 APP;无线智控;LED 情景;PWM
中图分类号:TN929.5
文献标志码:A

1 引言

随着电子信息、物联网、嵌入式以及 LED 高效照明等技术的发展,智能照明、绿色照明的概念、应用和产品也随之蓬勃发展,业已成为新的经济增长点,不断满足着人们日益增长的需求。目前,人民生活水平得以提高,对家庭电器智能化、使用便捷化的需求越来越强。家庭照明是家庭电器组成中最重要的一环,在此领域,节能环保的 LED 照明因其低耗能及环保的特点,正在逐步取代传统照明,而且智能 LED 情景照明系统已成为了一个研究热点。传统的照明多采用固定式开关,颜色亮度大多不可调控,随着人们对方便的无线开关方式以及对室内灯色多彩多变的追求以及环保观念的增强,除了对光源的发光时间、发光亮度,灯光场景氛围等基本要求有所提高外,还对灯光控制与管理的智能化、操作简单化、灵活化等方面提出更高的要求。

基于上述背景,本文着眼于智能家居和物联网技术应用,设计了一种基于蓝牙的无线智控 LED 情景照明灯,可以通过手机 APP 根据环境需要来调节 LED 灯组的开关,颜色和光强,从而营造一种舒适、惬意的光照环境,是用户感觉到场景的效果。

2 系统设计方案

无线智控 LED 情景照明灯包括硬件软件两部分。其中硬件包括 LED 驱动电路、单片机主控电路、蓝牙模块和灯罩及灯座。软件包括单片机嵌入式软件和手机 APP 软件。主控芯片的主要功能是检测蓝牙连接状态和接收客户端发送的指令,进而控制电路实现指令要求。智能手机控制软件则是基于 Android 系统开发的一款应用 APP,手机 APP 可以与 LED 灯组建立蓝牙连接、发送开关和调色调光等指令、预设情景模式等。

2.1 硬件系统设计

本系统是由单片机 STC12C5A32S2 输出三路 PWM[4] 电流来调节 LED 灯组的 RGB,从而实现情景灯发出各种不同颜色、不同亮度的光。三路 PWM 的占空比可以通过无线蓝牙模块连接的手机 APP 界面。LED 驱动电路[5] 由独立的电源模块,即使没有单片 STC12C5A32S2 控制,LED 灯也可以点亮,只是不能变化。系统硬件总框图如图 1 所示。

2.1.1 单片机 STC12C5A32S2

STC12C5A32S2/AD/PWM 系列单片机是宏晶科技生产的单时钟 / 机器周期的单片机,是高速 / 低功耗 / 超强抗干扰的新一代 8051 单片机,指令代码完全兼容传统 8051,但速度快 8–12 倍。内部集成 MAX810 专用复位电路,3 路 PWM,8 路高速 10 位 A/D 转换 (250K/S,即 25 万次 / 秒)。本设计用到了三个 PWM 口输出 PWM 信号给驱动芯片的 ENA、ENB、ENC。串口用于连接蓝牙串口模块。

2.1.2LED 灯组驱动电路

目前发光二极管驱动芯片按类型可分为:恒压式驱动芯片、恒流式驱动芯片。其中恒压式驱动芯片一般为常见的 DC/DC 升压芯片。恒压式驱动芯片成本便宜没有复杂的外围电路,但只恒定电压驱动 LED,就会造成驱动输出时电路电流的不可控,无法保证 LED 亮度的一致性。恒流式驱动芯片则解决了之前恒压式驱动的电流不可控问题。

实现校企的有效合作。校企合作作为一项多方参与的公共事务,缺少有效的监督、约束制度是很难深入进行的。因此,在校企合作立法中,不仅要强调参与主体的义务,同时更要规定不履行义务所应承担的后果,即承担一定的法律责任(包括民事和刑事责任)。职业教育校企合作立法尽管不是刑法、行政处罚法、治安管理条例等强制性法律,但依然可以依据相关法律的要求,对未履行义务的行为追究相关法律责任。目前,关于校企合作的政策性文件虽然不少,但是,这些文件是没有法律效力的,没有国家强制力作为保障,而且这些文件也仅仅停留在政策导向层面上,可操作性不强。因此,现有的法律体系无法保障校企合作在实践教学中的深入开展。所以,制定具有法律性的校企合作办学法规,是促进校企合作制度化的根本举措。

校企合作法律、法规应在实践中具有主导和媒介作用。政府应采取措施,让企业在确立市场需求、人才规格、知识技能结构、课程设置、教学内容和成绩评定等方面发挥相应作用。学校在时刻关注企业需求变化的同时,政府应有相应的指导部门,指导学校进行专业方向调整,确定培养培训规模,在开发、设计、实施灵活的培养培训方案上起积极的先导作用,真正发挥政府在校企合作中的主导和媒介作用。

建立校企合作的新机制,特别是在社会主义市场经济环境下,要以校企两方面双赢为目标。加强校企合作,是经济社会的快速发展对我国职业院校和企业提出的要求,职业院校的传统教育模式要革新离不开企业的协同,企业的成长也离不开学校的人才培养,而企业的壮大亦会反哺学校的发展,所以两者有一定的依存性、促进性。校企合作,不仅提高了人才培养的质量,也有利于企业的发展壮大。事实上,在校企合作中,真正连接学校和企业的纽带就是"利益",只有重视建立校企合作中双方的利益机制,解决好校企合作中的利益问题,校企双方的合作才有动力、才有积极性、才能持久深入。所以,我们的校企合作法规应通过某种制度设计确保具有可操作性,要让校、企双方通过合作"有利可图",满足职业院校和企业互惠互利的合作需求,以充分发挥各自在职业教育方面的优势,使我国的职业教育走上共赢发展的良性轨道。

3 结束语

建立和完善有中国特色的职业教育校企合作制度是一个漫长的过程,不可能一蹴而就,

但我们不能因此而忽视它的作用和意义。从目前来看,这是职业教育走出困境的良策;从长远的角度看,是形成校企合作长效机制的必然选择。在我国校企合作缺乏传统根基的情况下,只有坚持不懈地推进校企合作的制度化,职业教育校企合作才能最终走上持续健康、良性循环的发展之路。

【参考文献】
[1] 国务院.关于大力发展职业教育的决定 [EB/OL]. http://news.163.com, 2005-11-09.
[2] 张健.校企合作制度建设若干问题的探讨 [J].职教通讯,2011,(7).
[3] 张海峰.高职教育校企合作制度化研究 [J].教育与职业,2010,(17).

【作者简介】
杨少昆(1957 ~),男,长江工程职业技术学院副教授;研究方向:电工、电子、电气自动化等。

附录 6 获奖证书

基于手机蓝牙控制的-Light 智能家居彩灯和控制系统获奖证书

附录 7 系统手机端程序

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统的手机端程序

```
package com.qian.bluetooth;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;
import com.qian.bluetooth.sysinfo.SysInfoUtil;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences.Editor;
import android.util.Log;
import android.widget.Toast;
public class BluetoothActivity extends Activity {
    private static final int OPEN_BLUETOOTH_REQUESTCODE = 2;
    private static final int DISCOVERY_BLUETOOTH_REQUESTCODE = 1;
    protected static boolean isConnected = false;
    protected String bluetoothMAC;
    private static String TAG = "BaseActivity.java--";
    protected static BluetoothSocket socket = null;
    private static InputStream bluetooth_inStream = null;
    private static OutputStream bluetooth_outStream = null;
    //打开蓝牙
    protected void openBluetooth() {
        Intent intent = new Intent();
        intent.setAction(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(intent, OPEN_BLUETOOTH_REQUESTCODE);
    }
    // 判断是否已连接蓝牙设备
    public boolean isConnected() {
        return isConnected;
    }
    protected void initMAC() {
        bluetoothMAC = getStrData("BluetoothMAC");
    }
    private String getStrData(String key) {
        return getSharedPreferences(new SysInfoUtil(this).getPackageName(),
                Context.MODE_PRIVATE).getString(key, null);
    }
    private void saveMAC(String key, String macAddress) {
        Editor editor = getSharedPreferences(
                new SysInfoUtil(this).getPackageName(), Context.MODE_PRIVATE)
                .edit();
        editor.putString(key, macAddress);
        editor.commit();
    }
```

```
//搜索蓝牙
    protected void discoveryBluetoothMAC() {
        Intent intent = new Intent();
        intent.setClass(this, ListActivityMAC.class);
        startActivityForResult(intent, DISCOVERY_BLUETOOTH_REQUESTCODE);
    }
     //蓝牙连接
    protected boolean createBluetoothConnection() {
        BluetoothDevice device;
        if (isConnected) {
            try {
                bluetooth_inStream.close();
                bluetooth_outStream.close();
                socket.close();
            } catch (IOException e) {
                bluetooth_inStream = null;
                bluetooth_outStream = null;
                socket = null;
                isConnected = false;
            }
        }
        try {
            device = BluetoothAdapter.getDefaultAdapter().getRemoteDevice(
                    bluetoothMAC);
            socket = device.createRfcommSocketToServiceRecord(UUID
                    .fromString("00001101-0000-1000-8000-00805F9B34FB"));
            socket.connect();
            // 保存 MAC
            saveMAC("BluetoothMAC", bluetoothMAC);
            bluetooth_inStream = socket.getInputStream();
            bluetooth_outStream = socket.getOutputStream();
            Log.e(TAG, "BluetoothSocket 获取输入、输出流成功!");
            isConnected = true;
            return true;
        } catch (Exception e) {
            bluetoothMAC = null;
            saveMAC("BluetoothMAC", bluetoothMAC);
            Log.e(TAG, "BluetoothSocket 获取输入、输出流失败!");
            isConnected = false;
            return false;
        }
    }
//字节流发送数据
    protected int sendDate(byte[] data) {
        if (isConnected) {
            try {
                bluetooth_outStream.write(data);
                return 1;
            } catch (IOException e) {
                terminateConnection();
                return -1;
```

```
            }
        } else
            return 0;
    }
// 中断与外部蓝牙设备的连接
    private void terminateConnection() {

        if (isConnected) {
            isConnected = false;
            try {
                bluetooth_inStream.close();
                bluetooth_outStream.close();
                socket.close();
            } catch (IOException e) {
                bluetooth_inStream = null;
                bluetooth_outStream = null;
                socket = null;
            }

        }
        Log.e(TAG, "发送数据异常时中断与外部蓝牙设备的连接");
    }
    protected void showInfo(String info) {
        Toast.makeText(this, info, Toast.LENGTH_SHORT).show();
    }
}
package com.qian.palette;
import java.io.IOException;
import com.qian.bluetooth.BluetoothActivity;
import com.qian.bluetooth.BluetoothCtrl;
import com.qian.palette.R;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.Switch;
import android.widget.CompoundButton.OnCheckedChangeListener;
public class MainActivity extends BluetoothActivity implements OnClickListener {
```

```
private Switch allLights;
private Button[] btns = new Button[8];
private int[] ids = new int[] { R.id.room1, R.id.room2, R.id.room3,
        R.id.room4, R.id.room5, R.id.room6, R.id.duanBtn, R.id.connBtn };
private BluetoothAdapter bTA;
private AlertDialog dialog = null;
private byte[] data;
private String TAG = "RoomActivity--.java";
/*
 * AsyncTask:创建 BluetoothSocket 对象,并使其初始化和管理与外部蓝牙设备连接.
 */
class MyTask extends AsyncTask<String[], String, Integer> {
    @Override
    protected Integer doInBackground(String[]... params) {
        Log.e(TAG + "AsyncTask", "doInBackground() ");
        if (createBluetoothConnection())
            return 1;
        else
            return -1;
    }
    @Override
    protected void onPostExecute(Integer result) {
        Log.e(TAG + "AsyncTask", " onPostExecute(Integer integer) ="
                + result.intValue());
        if (result.intValue() == 1)
            showInfo(getString(R.string.msg_connect_ok));
        else
            showInfo(getString(R.string.msg_connect_fail));
    }
}
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getWindow().requestFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.activity_main);
    init();
    initMAC();
}
private void init() {
    for (int i = 0; i < btns.length; i++) {
        btns[i] = (Button) findViewById(ids[i]);
        btns[i].setOnClickListener(this);
    }
    allLights = (Switch) findViewById(R.id.all_lights);
    allLights.setOnCheckedChangeListener(new SwitchListener());
    bTA = BluetoothAdapter.getDefaultAdapter();
    if (!bTA.isEnabled()) {
        bTA.enable();
    }
}
class SwitchListener implements OnCheckedChangeListener {
```

```
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
        // 控制灯的开关.
        if (isChecked) {
            sendRGB("255x255y255z*");
            // doSend(1);
        } else {
            sendRGB("0x0y0z*");
            // doSend(2);
        }
    }
}
void sendRGB(String dStr) {
    int flag;
    try {
        flag = sendDate(dStr.getBytes());
        if (flag == -1)
            showInfo(getString(R.string.msg__SendBytesErr));
        else if (flag == 0)
            showInfo(getString(R.string.msg_please_connect));
    } catch (Exception e) {
        showInfo("数据格式异常!");
    }
}
// 处理灯的开关,单命令控制.(接收方接收到的是:ASCII 值.)
public void doSend(int f) {
    Log.e(TAG + "126-----", "doSend():");
    data = new byte[1];
    // 将十进制数强制转化为 byte.
    if (f == 1) {
        data[0] = (byte) (0xff & Integer.decode("0x4f"));
    } else if (f == 2) {
        data[0] = (byte) (0xff & Integer.decode("0x43"));
    }
    int flag = sendDate(data);
    if (flag == -1)
        showInfo(getString(R.string.msg__SendBytesErr));
    else if (flag == 0)
        showInfo(getString(R.string.msg_please_connect));
}
@Override
public void onClick(View v) {
    Intent intent = new Intent();
    intent.setClass(this, GuestRoom.class);
    switch (v.getId()) {
    case R.id.room1:
        this.startActivity(new Intent(this, PaletteActivity1.class));
        break;
    case R.id.room2:
        this.startActivity(new Intent(this, PaletteActivity2.class));
```

```
            break;
        case R.id.room3:
            this.startActivity(new Intent(this, PaletteActivity3.class));
            break;
        case R.id.room4:
            this.startActivity(new Intent(this, PaletteActivity4.class));
            break;
        case R.id.room5:
            this.startActivity(new Intent(this, PaletteActivity5.class));
            break;
        case R.id.room6:
            intent.putExtra("room", "room4");
            this.startActivity(intent);
            break;
        case R.id.duanBtn:
            Log.e(TAG, "断开按钮");
            if (socket != null) {
                try {
                    socket.close();
                    isConnected = false;
                    showInfo("断开成功!");
                } catch (IOException e) {
                    showInfo("断开失败!");
                }
            }
            break;
        case R.id.connBtn:
            connectDevices();
            break;
        }
    }
    // 连接蓝牙设备.
    private void connectDevices() {
        // 连接设备前,判断蓝牙是否开启.
        if (!bTA.isEnabled()) {
            openBluetooth();
            return;
        }
        if (!isConnected()) {
            if (bluetoothMAC == null) {
                // 搜索蓝牙设备
                discoveryBluetoothMAC();
                return;
            }
            // bluetoothMAC != null,连接
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle(getString(R.string.menu_main_Connection));
            builder.setMessage(getString(R.string.msg_connect_history));
            builder.setPositiveButton(getString(R.string.btn_connect),
                    new DialogInterface.OnClickListener() {
                        @Override
```

```
                    public void onClick(DialogInterface dialog, int which) {
                        new MyTask().execute(new String[0]);
                    }
                });
        builder.setNegativeButton(getString(R.string.btn_reSearch),
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        discoveryBluetoothMAC();
                    }
                });
        builder.create().show();
    }
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Log.e(TAG + "", "onActivityResult(int,int,Intent)");
    Log.e("onActivityResult(int,int,Intent)", "Intent data=" + data);
    if (requestCode == 2 && resultCode == RESULT_CANCELED) {// 请求打开蓝牙
        finish();
    } else if (requestCode == 1 && resultCode == -1) {
        BluetoothDevice device = (BluetoothDevice) data
                .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
        if (device.getBondState() == BluetoothDevice.BOND_NONE) {
            try {
                showInfo(getString(R.string.msg_actDiscovery_Bluetooth_Bond_none));
                BluetoothCtrl.createBond(device);
            } catch (Exception e) {
                showInfo(getString(R.string.msg_actDiscovery_Bluetooth_Bond_fail));
            }
        } else {
            bluetoothMAC = device.getAddress();
            Log.e(TAG, "获取 MAC=" + bluetoothMAC);
            // 执行异步任务 asynchronous
            Log.e(TAG, "获取 MAC=" + new String[0]);
            new MyTask().execute(new String[0]);
        }
    }
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.about) {
        showDialogInfo(R.layout.about_text);
    } else if (id == R.id.help) {
        showDialogInfo(R.layout.help_text);
    }
    return true;
}
private void showDialogInfo(int id) {
```

```
        AlertDialog.Builder dialogBuilder = new Builder(this);
        LayoutInflater inflater = LayoutInflater.from(this);
        final View view = inflater.inflate(id, null);
        ImageButton imgBtn = (ImageButton) view.findViewById(R.id.backBtnId);
        imgBtn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                dialog.dismiss();
            }
        });
        dialogBuilder.setView(view);
        dialog = dialogBuilder.create();
        dialog.show();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.e("onDestroy()--", "关闭蓝牙");
        if (bTA.isEnabled()) {
            bTA.disable();
        }
    }
}
package com.qian.palette;
import com.qian.bluetooth.BluetoothActivity;
import com.qian.bluetooth.ConstantUtil;
import com.qian.palette.R;
import com.qian.palette.PaletteView.onColorChangedListener;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.ImageView;
import android.widget.ToggleButton;
public class PaletteActivity1 extends BluetoothActivity implements
        onColorChangedListener {
    private ToggleButton switchOff;
    private ImageView img_state;
    private byte[] data;
    private static String initColor = "#000000";
    // 控制是否可发送RGB.
    private boolean isOpened = false;
    // 接收并传递开关的状态值(开->true.)
    private boolean flag_light = false;
    private String TAG = "PaletteActivity.java----------------";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 设置布局
```

```
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.palette_layout);
    init();
}
private void init() {
    switchOff = (ToggleButton) findViewById(R.id.switchOnOff);
    switchOff.setOnCheckedChangeListener(new SwitchListener());
    img_state = (ImageView) findViewById(R.id.light_state);
    PaletteView palette = (PaletteView) this.findViewById(R.id.paletteView);
    palette.setOnColorChangedListener(this);
}
@Override
public void setOnColorChanged(int color) {
    // 提取 RGB 值.
    convertToRGBHEX(color);
    Log.e(TAG + "----------", "rgb:" + convertToRGBHEX(color));
    // 将 RGB 发送.(当点击和滑动事件同时发生时,只在滑动情况下发送数据.)
    if (isOpened && ConstantUtil.touchEventCtrl) {
        Log.e("发送一次数据", "发送一次数据");
        if (!initColor.equals(convertToRGBHEX(color)))
            sendRGB(convertToRGB(color));
    }
}
private void sendRGB(String dStr) {
    int flag;
    try {
        flag = sendDate(dStr.getBytes());
        if (flag == -1)
            showInfo(getString(R.string.msg__SendBytesErr));
        else if (flag == 0)
            showInfo(getString(R.string.msg_please_connect));
        else {
            isOpened = flag_light;
            if (isOpened)
                switchOff.setTextOn("开灯");
            else
                switchOff.setTextOff("关灯");
            img_state.setImageResource(isOpened ? R.drawable.on
                    : R.drawable.off);
        }
    } catch (Exception e) {
        showInfo("数据格式异常!");
    }
}
// 将颜色值 int 转换为 对应的 hex
public static String convertToRGB(int color) {
    String red = Integer.toHexString(Color.red(color));
    String green = Integer.toHexString(Color.green(color));
    String blue = Integer.toHexString(Color.blue(color));
    if (red.length() == 1) {
        red = "0" + red;
```

```
        }
        if (green.length() == 1) {
            green = "0" + green;
        }
        if (blue.length() == 1) {
            blue = "0" + blue;
        }
        String str_red = null;
        String str_green = null;
        String str_blue = null;
        try {
            str_red = String.valueOf(Integer.decode("#" + red));
            str_green = String.valueOf(Integer.decode("#" + green));
            str_blue = String.valueOf(Integer.decode("#" + blue));
        } catch (Exception e) {
        }
        return str_red + "x" + str_green + "y" + str_blue + "z"+"!";
    }
    public static String convertToRGBHEX(int color) {
        String red = Integer.toHexString(Color.red(color));
        String green = Integer.toHexString(Color.green(color));
        String blue = Integer.toHexString(Color.blue(color));
        if (red.length() == 1) {
            red = "0" + red;
        }
        if (green.length() == 1) {
            green = "0" + green;
        }
        if (blue.length() == 1) {
            blue = "0" + blue;
        }
        return "#" + red + green + blue;
    }
    class SwitchListener implements OnCheckedChangeListener {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked) {
            // 控制灯的开关.
            if (isChecked) {
                flag_light = isChecked;
                sendRGB("255x255y255z!");
                //doSend("0x42");//B
            } else {
                flag_light = isChecked;
                sendRGB("0x0y0z!");
                //doSend("0x62");//b
            }
        }
    }
    // 处理灯的开关,单命令控制.(接收方接收到的是:ASCII 值.)
    public void doSend(String hexStr) {
```

```
        Log.e(TAG + "126-----", "doSend():");
        data = new byte[1];
        // 将十进制数强制转化为 byte.
        data[0] = (byte) (0xff & Integer.decode(hexStr));
        int flag = sendDate(data);
        if (flag == -1)
            showInfo(getString(R.string.msg__SendBytesErr));
        else if (flag == 0)
            showInfo(getString(R.string.msg_please_connect));
        else {
            isOpened = flag_light;
            if (isOpened)
                switchOff.setTextOn("开灯");
            else
                switchOff.setTextOff("关灯");
            img_state.setImageResource(isOpened ? R.drawable.on
                    : R.drawable.off);
        }
        Log.e("是否灯已经打开...", "isOpened=" + isOpened);
    }
}
```

附录 8 系统 LED 控制器程序

基于手机蓝牙控制的 i-Light 智能家居彩灯和控制系统的 LED 控制器程序

```
/*************************主程序************************/
#include "reg52.h"
#include "Delay.h"
#include "2401.h"
#include "stdlib.h"
#include "string.h"
#include "uart.h"
unsigned char pwm0=0x40,pwm1=0x40,pwm2=0x40,flag0=1;
unsigned char charx,chary,charz ,str[20]="",strx[20]="";
uchar  test[]="";uchar RevTempDate0[20]="";
char * mid(char *dst,char *src, int n,int m);
void System_init(void);
unsigned char ReceiveDate();
void main()
{   System_init();//系统初始化
    while(1){};
}
/*********************************************************************/
/*函数原型: void System_init(void)
/*函数功能:系统初始化
/*********************************************************************/
void System_init(void)
{   CCON = 0;//初始化 PCA 控制寄存器//PCA 定时器停止//清除 CF 标志//清除模块中断标志
    CL = 0; CH = 0;//复位 PCA 寄存器
    CMOD = 0x0A; //设置 PCA 时钟源 //时钟源输入：sysclk/4
    PCA_PWM0 = 0x00; //PCA 模块 0 工作于 8 位 PWM
    CCAP0H = CCAP0L = 0x96;//PWM0 的占空比为 87.5% ((100H-20H)/100H)
    CCAPM0 = 0x02;//PCA 模块 0 为 8 位 PWM 模式  //脉宽调节模式
    PCA_PWM1 = 0x00; // PCA 模块 0 工作于 8 位 PWM
    CCAP1H = CCAP1L = 0x96;//PWM1 的占空比为 87.5% ((100H-20H)/100H)
    CCAPM1 = 0x02;//PCA 模块 0 为 8 位 PWM 模式  //脉宽调节模式
    PCA_PWM2 = 0x00;// PCA 模块 0 工作于 8 位 PWM
    CCAP2H = CCAP2L = 0x96;//PWM2 的占空比为 87.5% ((100H-20H)/100H)
    CCAPM2 = 0x02; //PCA 模块 0 为 8 位 PWM 模式  //脉宽调节模式
    CR = 1; //禁止 PCA 定时器溢出中断
    CCAP0H = CCAP0L = pwm0; //产生 PWM0
    CCAP1H = CCAP1L = pwm1; //产生 PWM1
    CCAP2H = CCAP2L = pwm2; //产生 PWM2
    Uart_Init();//串口初始化
    NRF24L01Int();//NRF2401 初始化
}
/*********************************************************************/
/*函数原型:unsigned char ReceiveDate(void)
/*函数功能:NRF2401 接收数据
/*********************************************************************/
unsigned char ReceiveDate(void)
{   uchar i;//接收通道号
```

```
    sta=NRFReadReg(R_REGISTER+STATUS);//发送数据后读取状态寄存器
    if(sta&0x40)                 // 判断是否接收到数据  MASK_RX_DR
    {   flag0=1;CE=0;//待机
        NRFReadRxDate(R_RX_PAYLOAD,RevTempDate0,RX_DATA_WITDH); // 从RXFIFO读取数据通道0
        for(i=0;i<strlen(RevTempDate0);i++)//清空RevTempDate0缓存
        RevTempDate0[i]='\0';}
    else flag0=0;return flag0;}
char * mid(char *dst,char *src, int n,int m) /*n为长度，m为位置*/
{   char *p = src; char *q = dst; int len = strlen(src);
    if(n>len) n = len-m;if(m<0) m=0;if(m>len) return NULL;
    p += m;while(n--) *(q++) = *(p++);*(q)='\0';
    return dst;}
/**************************串口中断服务子程序*********************/
/*函数原型: void Uart_Receive() interrupt 4
/*函数功能:执行串口中断程序
/*****************************************************************/
void Uart_Receive() interrupt 4
{   unsigned char temp,k,count,charx,chary,charz ,str[20]="",strx[20]="";//定义变量和缓存
    if(RI==1){
      RI = 0;//清空接收中断标志位
      temp = SBUF;test[count++]=temp;//接收数据test[count]='\0';//添加结束符
      switch(temp){
      case '!'://数据处理后直接产生3路PWM输入LED控制器驱动电路
          strcpy(strx,test);charx=strchr(strx,'x')-strx+1;//截取PWM0字符型数据
           chary=strchr(strx,'y')-strx+1;//截取PWM1字符型数据
          charz=strchr(strx,'z')-strx+1;//截取PWM2字符型0数据
          pwm0=atoi(mid(str,strx,charx-1,0));// PWM0字符串转为整数
          pwm1=atoi(mid(str,strx,chary-charx-1,charx)); // PWM1字符串转为整数
          pwm2=atoi(mid(str,strx,charz-chary-1,chary)); // PWM2字符串转为整数
          CCAP0H = CCAP0L = pwm0; //产生PWM0
          CCAP1H = CCAP1L = pwm1; //产生PWM1
          CCAP2H = CCAP2L = pwm2; //产生PWM2
          for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存
          for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存
          count=0;//接收字节数清零break;跳出switch语句
      case '#'://通过NRF2401模块发送接收的数据到第二个LED控制器
          NRFSetTxMode(test);//设置NRF2401为发送模式
          while(CheckACK());    //等待发送完毕
          for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存
          for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存
          count=0;//接收字节数清零break;跳出switch语句
      case '$'://通过NRF2401模块发送接收的数据到第三个LED控制器
          NRFSetTxMode1(test);//设置NRF2401为发送模式
          while(CheckACK());    //等待发送完毕
          for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存
          for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存
          count=0;//接收字节数清零break;跳出switch语句
      case '%'://通过NRF2401模块发送接收的数据到第四个LED控制器
          NRFSetTxMode2(test);//设置NRF2401为发送模式
          while(CheckACK());    //等待发送完毕
          for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存
```

```
            for(k=0;k<strlen(test);k++)test[k]=0;//清空 test 缓存
            count=0;//接收字节数清零 break;跳出 switch 语句
        case '^':通过 NRF2401 模块发送接收的数据到第五个 LED 控制器
            NRFSetTxMode3(test);//设置 NRF2401 为发送模式
            while(CheckACK());    //等待发送完毕
            for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空 strx 缓存
            for(k=0;k<strlen(test);k++)test[k]=0;//清空 test 缓存
            count=0;//接收字节数清零 break;跳出 switch 语句
        case '&':通过 NRF2401 模块发送接收的数据到第六个 LED 控制器
            NRFSetTxMode4(test);//设置 NRF2401 为发送模式
            while(CheckACK());    //等待发送完毕
            for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空 strx 缓存
            for(k=0;k<strlen(test);k++)test[k]=0;//清空 test 缓存
            count=0;//接收字节数清零 break;跳出 switch 语句
        case '*':通过 NRF2401 模块发送接收的数据到第七个 LED 控制器
            NRFSetTxMode5(test);//设置 NRF2401 为发送模式
            while(CheckACK());    //等待发送完毕
            for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空 strx 缓存
            for(k=0;k<strlen(test);k++)test[k]=0;//清空 test 缓存
            count=0;//接收字节数清零 break;跳出 switch 语句
          }
    }
}
/**********************NRF24L01 驱动程序**********************/
#include"2401.h"
#include"delay.h"
#define uint unsigned int
#define uchar unsigned char
uchar code RxAddr0[]={0x34,0x43,0x10,0x10,0x01};//编号 0 接收地址这个地址和发送方地址一样!
uchar code RxAddr1[]={0xc2,0xc2,0xc2,0xc2,0xc1};//编号 1
uchar code RxAddr2[]={0xc3};//编号 2
uchar code RxAddr3[]={0xc4};//编号 3
uchar code RxAddr4[]={0xc5};//编号 4
uchar code RxAddr5[]={0xc6};//编号 5
uchar code TxAddr[]={0xc1,0xb2,0xb3,0xb4,0x01};//发送地址
uchar code TxAddr1[]={0xb0,0xb2,0xb3,0xb4,0x01};//编号 1
uchar code TxAddr2[]={0xb1,0xb2,0xb3,0xb4,0x01};//编号 2
uchar code TxAddr3[]={0xb2,0xb2,0xb3,0xb4,0x01};//编号 3
uchar code TxAddr4[]={0xb3,0xb2,0xb3,0xb4,0x01};//编号 4
uchar code TxAddr5[]={0xb4,0xb2,0xb3,0xb4,0x01};//编号 5
/**********************状态标志**************************/
uchar bdata sta;   //状态标志
sbit RX_DR=sta^6;
sbit TX_DS=sta^5;
sbit MAX_RT=sta^4;
/**********************SPI 时序函数**************************/
uchar NRFSPI(uchar date)
{
    uchar i;
    for(i=0;i<8;i++)          // 循环 8 次
    {  if(date&0x80)MOSI=1;
```

```
        else MOSI=0;    // byte最高位输出到MOSI
       date<<=1;        // 低一位移位到最高位
       SCLK=1;
       if(MISO)         // 拉高SCK，nRF24L01从MOSI读入1位数据，同时从MISO输出1位数据
         date|=0x01;    // 读MISO到byte最低位
       SCLK=0;          // SCK置低
     }
     return(date);                 // 返回读出的一字节
}
/*******************NRF24L01初始化函数**********************/
void NRF24L01Int()
{    Delay(2);//让系统什么都不干
     CE=0; //待机模式1
     CSN=1;
     SCLK=0;
     IRQ=1;
}
/******************SPI读寄存器一字节函数********************/
uchar NRFReadReg(uchar RegAddr)
{  uchar BackDate;
   CSN=0;//启动时序
   NRFSPI(RegAddr);//写寄存器地址
   BackDate=NRFSPI(0x00);//写入读寄存器指令
   CSN=1;
   return(BackDate); //返回状态
}
/******************SPI写寄存器一字节函数********************/
uchar NRFWriteReg(uchar RegAddr,uchar date)
{
   uchar BackDate;
   CSN=0;//启动时序
   BackDate=NRFSPI(RegAddr);//写入地址
   NRFSPI(date);//写入值
   CSN=1;
   return(BackDate);
}
/*****************SPI读取RXFIFO寄存器的值*****************/
uchar NRFReadRxDate(uchar RegAddr,uchar *RxDate,uchar DateLen)
{  //寄存器地址//读取数据存放变量//读取数据长度//用于接收
    uchar BackDate,i;
    CSN=0;//启动时序
    BackDate=NRFSPI(RegAddr);//写入要读取的寄存器地址
    for(i=0;i<DateLen;i++)RxDate[i]=NRFSPI(0);//读取数据
    CSN=1;
    return(BackDate);
}
/******************SPI写入TXFIFO寄存器的值****************/
uchar NRFWriteTxDate(uchar RegAddr,uchar *TxDate,uchar DateLen)
{ //寄存器地址//写入数据存放变量//读取数据长度//用于发送
   uchar BackDate,i;
   CSN=0;
```

```
    BackDate=NRFSPI(RegAddr);//写入要写入寄存器的地址
    for(i=0;i<DateLen;i++)NRFSPI(*TxDate++);//写入数据
    CSN=1;
    return(BackDate);
}
/**************NRF 设置为发送模式并发送数据**************/
void NRFSetTxMode(uchar *TxDate)
{//发送模式
     CE=0;
     NRFWriteTxDate(W_REGISTER+TX_ADDR,TxAddr,TX_ADDR_WITDH);//写寄存器指令+接收地址使能指
令+接收地址+地址宽度
     NRFWriteTxDate(W_REGISTER+RX_ADDR_P0,TxAddr,TX_ADDR_WITDH);//为了应答接收设备，接收通
道 0 地址和发送地址相同
     NRFWriteTxDate(W_TX_PAYLOAD,TxDate,TX_DATA_WITDH);//写入数据
     /******下面有关寄存器配置**************/
     NRFWriteReg(W_REGISTER+EN_AA,0x03f);      // 使能接收通道 0 自动应答
     NRFWriteReg(W_REGISTER+EN_RXADDR,0x03f); // 使能接收通道 0
     NRFWriteReg(W_REGISTER+SETUP_RETR,0x0a); // 自动重发延时等待 250us+86us，自动重发 10 次
     NRFWriteReg(W_REGISTER+RF_CH,0x40);       // 选择射频通道 0x40
     NRFWriteReg(W_REGISTER+RF_SETUP,0x07);  // 数据传输率 1Mbps，发射功率 0dBm，低噪声放大
器增益
     NRFWriteReg(W_REGISTER+CONFIG,0x0e);     // CRC 使能，16 位 CRC 校验，上电
     CE=1;
     Delay(5);//保持 10us 秒以上
}
/**************NRF 设置为发送模式并发送数据**************/
void NRFSetTxMode1(uchar *TxDate)
{//发送模式
     CE=0;
     NRFWriteTxDate(W_REGISTER+TX_ADDR,TxAddr1,TX_ADDR_WITDH);//写寄存器指令+接收地址使能
指令+接收地址+地址宽度
     NRFWriteTxDate(W_REGISTER+RX_ADDR_P1,TxAddr1,TX_ADDR_WITDH);//为了应答接收设备,接收通
道 0 地址和发送地址相同
     NRFWriteTxDate(W_TX_PAYLOAD,TxDate,TX_DATA_WITDH);//写入数据
     /******下面有关寄存器配置**************/
     NRFWriteReg(W_REGISTER+EN_AA,0x3f);       // 使能接收通道 0 自动应答
     NRFWriteReg(W_REGISTER+EN_RXADDR,0x3f);  // 使能接收通道 0
     NRFWriteReg(W_REGISTER+SETUP_RETR,0x0a); // 自动重发延时等待 250us+86us，自动重发 10 次
     NRFWriteReg(W_REGISTER+RF_CH,0x40);       // 选择射频通道 0x40
     NRFWriteReg(W_REGISTER+RF_SETUP,0x07);   // 数据传输率 1Mbps，发射功率 0dBm，低噪声放大
器增益
     NRFWriteReg(W_REGISTER+CONFIG,0x0e);     // CRC 使能，16 位 CRC 校验，上电
     CE=1;
     Delay(5);//保持 10us 秒以上
}
void NRFSetTxMode2(uchar *TxDate)
{//发送模式
     CE=0;
     NRFWriteTxDate(W_REGISTER+TX_ADDR,TxAddr2,TX_ADDR_WITDH);//写寄存器指令+接收地址使能
指令+接收地址+地址宽度
     NRFWriteTxDate(W_REGISTER+RX_ADDR_P2,TxAddr2,TX_ADDR_WITDH);//为了应答接收设备,接收通
```

```
道 0 地址和发送地址相同
    NRFWriteTxDate(W_TX_PAYLOAD,TxDate,TX_DATA_WITDH);//写入数据
    /******下面有关寄存器配置**********/
    NRFWriteReg(W_REGISTER+EN_AA,0x3f);        // 使能接收通道 0 自动应答
    NRFWriteReg(W_REGISTER+EN_RXADDR,0x3f);    // 使能接收通道 0
    NRFWriteReg(W_REGISTER+SETUP_RETR,0x0a);  // 自动重发延时等待 250us+86us，自动重发 10
次
    NRFWriteReg(W_REGISTER+RF_CH,0x40);        // 选择射频通道 0x40
    NRFWriteReg(W_REGISTER+RF_SETUP,0x07);     // 数据传输率 1Mbps，发射功率 0dBm，低噪声放
大器增益
    NRFWriteReg(W_REGISTER+CONFIG,0x0e);       // CRC 使能，16 位 CRC 校验，上电
    CE=1;
    Delay(5);//保持 10us 秒以上
}
void NRFSetTxMode3(uchar *TxDate)
{  //发送模式
    CE=0;
    NRFWriteTxDate(W_REGISTER+TX_ADDR,TxAddr3,TX_ADDR_WITDH);//写寄存器指令+接收地址使能
指令+接收地址+地址宽度
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P3,TxAddr3,TX_ADDR_WITDH);//为了应答接收设备，接收通
道 0 地址和发送地址相同
    NRFWriteTxDate(W_TX_PAYLOAD,TxDate,TX_DATA_WITDH);//写入数据
    /******下面有关寄存器配置**********/
    NRFWriteReg(W_REGISTER+EN_AA,0x3f);       // 使能接收通道 0 自动应答
    NRFWriteReg(W_REGISTER+EN_RXADDR,0x3f);   // 使能接收通道 0
    NRFWriteReg(W_REGISTER+SETUP_RETR,0x0a); // 自动重发延时等待 250us+86us，自动重发 10 次
    NRFWriteReg(W_REGISTER+RF_CH,0x40);       // 选择射频通道 0x40
    NRFWriteReg(W_REGISTER+RF_SETUP,0x07);    // 数据传输率 1Mbps，发射功率 0dBm，低噪声放大
器增益
    NRFWriteReg(W_REGISTER+CONFIG,0x0e);      // CRC 使能，16 位 CRC 校验，上电
    CE=1;
    Delay(5);//保持 10us 秒以上
 }
void NRFSetTxMode4(uchar *TxDate)
{   //发送模式
    CE=0;
    NRFWriteTxDate(W_REGISTER+TX_ADDR,TxAddr4,TX_ADDR_WITDH);//写寄存器指令+接收地址使能
指令+接收地址+地址宽度
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P4,TxAddr4,TX_ADDR_WITDH);//为了应答接收设备，接收通
道 0 地址和发送地址相同
    NRFWriteTxDate(W_TX_PAYLOAD,TxDate,TX_DATA_WITDH);//写入数据
    /******下面有关寄存器配置**********/
    NRFWriteReg(W_REGISTER+EN_AA,0x3f);       // 使能接收通道 0 自动应答
    NRFWriteReg(W_REGISTER+EN_RXADDR,0x3f);   // 使能接收通道 0
    NRFWriteReg(W_REGISTER+SETUP_RETR,0x0a); // 自动重发延时等待 250us+86us，自动重发 10 次
    NRFWriteReg(W_REGISTER+RF_CH,0x40);      // 选择射频通道 0x40
    NRFWriteReg(W_REGISTER+RF_SETUP,0x07);  // 数据传输率 1Mbps，发射功率 0dBm，低噪声放大
器增益
    NRFWriteReg(W_REGISTER+CONFIG,0x0e);     // CRC 使能，16 位 CRC 校验，上电
    CE=1;
    Delay(5);//保持 10us 秒以上
```

```
}
void NRFSetTxMode5(uchar *TxDate)
{   //发送模式
    CE=0;
    NRFWriteTxDate(W_REGISTER+TX_ADDR,TxAddr5,TX_ADDR_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P5,TxAddr5,TX_ADDR_WITDH);//为了应答接收设备,接收通道 0 地址和发送地址相同
    NRFWriteTxDate(W_TX_PAYLOAD,TxDate,TX_DATA_WITDH);//写入数据
    /******下面有关寄存器配置**************/
    NRFWriteReg(W_REGISTER+EN_AA,0x3f);       // 使能接收通道 0 自动应答
    NRFWriteReg(W_REGISTER+EN_RXADDR,0x3f);  // 使能接收通道 0
    NRFWriteReg(W_REGISTER+SETUP_RETR,0x0a); // 自动重发延时等待 250us+86us，自动重发 10 次
    NRFWriteReg(W_REGISTER+RF_CH,0x40);       // 选择射频通道 0x40
    NRFWriteReg(W_REGISTER+RF_SETUP,0x07);   // 数据传输率 1Mbps，发射功率 0dBm，低噪声放大器增益
    NRFWriteReg(W_REGISTER+CONFIG,0x0e);      // CRC 使能，16 位 CRC 校验，上电
    CE=1;
    Delay(5);//保持 10us 秒以上
}
/**************NRF 设置为接收模式并接收数据*************/
//接收模式
void NRFSetRXMode()
{   CE=0;
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P0,RxAddr0,TX_ADDR_WITDH);  // 接收设备接收通道 0 使用和发送设备相同的发送地址
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P1,RxAddr1,TX_ADDR_WITDH);  // 接收设备接收通道 1 使用和发送设备相同的发送地址
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P2,RxAddr2,1);  // 接收设备接收通道 0 使用和发送设备相同的发送地址
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P3,RxAddr3,1);  // 接收设备接收通道 1 使用和发送设备相同的发送地址
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P4,RxAddr4,1);  // 接收设备接收通道 0 使用和发送设备相同的发送地址
    NRFWriteTxDate(W_REGISTER+RX_ADDR_P5,RxAddr5,1);  // 接收设备接收通道 1 使用和发送设备相同的发送地址
    NRFWriteReg(W_REGISTER+EN_AA,0x3f);                // 使能数据通道 5 自动应答
    NRFWriteReg(W_REGISTER+EN_RXADDR,0x3f);            // 使能接收通道 5
    NRFWriteReg(W_REGISTER+RX_PW_P0,TX_DATA_WITDH);  // 接收通道 0 选择和发送通道相同有效数据宽度
    NRFWriteReg(W_REGISTER+RX_PW_P1,TX_DATA_WITDH);  // 接收通道 1 选择和发送通道相同有效数据宽度
    NRFWriteReg(W_REGISTER+RX_PW_P2,TX_DATA_WITDH);  // 接收通道 0 选择和发送通道相同有效数据宽度
    NRFWriteReg(W_REGISTER+RX_PW_P3,TX_DATA_WITDH);  // 接收通道 1 选择和发送通道相同有效数据宽度    NRFWriteReg(W_REGISTER+RX_PW_P0,TX_DATA_WITDH);  // 接收通道 0 选择和发送通道相同有效数据宽度
    NRFWriteReg(W_REGISTER+RX_PW_P4,TX_DATA_WITDH);  // 接收通道 1 选择和发送通道相同有效数据宽度
    NRFWriteReg(W_REGISTER+RX_PW_P5,TX_DATA_WITDH);  // 接收通道 0 选择和发送通道相同有效数据宽度
```

```
    NRFWriteReg(W_REGISTER+RF_CH,0x40);// 选择射频通道 0x40
    NRFWriteReg(W_REGISTER+RF_SETUP,0x07);              // 数据传输率 1Mbps，发射功率 0dBm，
低噪声放大器增益
    NRFWriteReg(W_REGISTER+CONFIG,0x0f);               //CRC 使能，16 位 CRC 校验，上电，接收
模式
    CE = 1; Delay(5);
}
/**********************检测应答信号**********************/
uchar CheckACK()
{  //用于发射
    sta=NRFReadReg(R_REGISTER+STATUS);        // 返回状态寄存器
    if(TX_DS||MAX_RT)                         //发送完毕中断
    {  NRFWriteReg(W_REGISTER+STATUS,0xff);  // 清除 TX_DS 或 MAX_RT 中断标志
       CSN=0; NRFSPI(FLUSH_TX); CSN=1;
       return(0);
    }
    else return(1);
}
/**********判断是否接收收到数据，接到就从 RX 取出***********/
//用于接收模式
uchar NRFRevDate(uchar *RevDate)
{    uchar RevFlags=0;
     sta=NRFReadReg(R_REGISTER+STATUS);//发送数据后读取状态寄存器
     if(sta&0x4)          // 判断是否接收到数据
     {  CE=0;             //SPI 使能
        NRFReadRxDate(R_RX_PAYLOAD,RevDate,RX_DATA_WITDH);// 从 RXFIFO 读取数据
        RevFlags=1;     //读取数据完成标志
      }
     NRFWriteReg(W_REGISTER+STATUS,0xff); //接收到数据后 RX_DR,TX_DS,MAX_PT 都置高为 1，通
过写 1 来清楚中断标
     CSN=0; NRFSPI(FLUSH_RX); CSN=1;
     return(RevFlags);
}
```