摘　要

在新时代智能家居与智能生活中，对灯光的控制是必不可少的，本文基于单片机 C8051F310，搭载光控和声控电路，对灯光进行控制，可调节不同的模式，对灯光进行不同的控制，在考虑其他噪声的情况下，对控制器进行软件除噪与防抖，使产品更加人性与智能

关键词： 灯光控制　　智能生活　　C8051F310　　软件除噪

目录

一、 技术指标

1.1 系统参数

系统数据	详细信息
CPU	80C51F310
电源	5VUSB 供电
工作电流	300mA-1A
显示	四位一体数码管
亮度等级	100 个等级
声控阈值	可调
有效声控时间	以拍手时间为有效时间，0.1s 以内
光控阈值	可调
按键	5 个

1.2 输出参数

输出参数	详细信息
灯光	LED 高亮白光
模式	3 种工作模式（智能灯，台灯，过道灯）
亮度等级显示	0-99 分别代表 100 个亮度等级

‘

二、基本原理

2.1 总体原理

对于一个控制器，则可以看成由三个部分组成，一是信号的采集，二是控制的输出，三是控制的逻辑。

对于本方案来说，我们用光敏电阻来构成光信号的采集电路，驻极体话筒来构成声音信号的采集电路，利用按键按作为控制信号，通过按键来调节不同的模式和亮度。利用 C8051F310 做信号的处理与控制，经过 CPU 之后，利用 I/O 口的电压控制三极管的开关特性来驱动 LED 发光，再利用 CD4511 译码器来驱动四位一体数码管。对于灯光亮度的控制，我们利用 C8051F310 可以输出 PWM 波，将 LED 受控制的 I/O 口，配置上 PWM 波，通过 CPU 改变不同的占空比来进行灯光亮度的控制

2.2 光敏电阻

光敏电阻器是利用半导体的光电导效应制成的一种电阻值随入射光的强弱而改变的电阻器，又称为光电导探测器；入射光强，电阻减小，入射光弱，电阻增大。

2.3 驻极体话筒

驻极体话筒体积小，结构简单，电声性能好，价格低廉，应用非常广泛。

高分子极化膜上生产时就注入了一定的永久电荷(Q)，由于没有放电回路，这个电荷量是不变的，在声波的作用下，极化膜随着声音震动，因此和背极的距离也跟着变化，也就是说极化膜和背极间的电容是随声波变化。

我们知道电容上电荷的公式是 Q=C·U，反之 U=Q/C 也是成立的。驻极体总的电荷量是不变，当极板在声波压力下后退时，电容量减小，电容两极间的电压就会成反比的升高，反之电容量增加时电容两极间的电压就会成反比的降低。最后再通过阻抗非常高的场效应将电容两端的电压取出来，同时进行放大，我们就可以得到和声音对应的电压了。由于场效应管是有源器件，需要一定的偏置和电流才可以工作在放大状态，因此，驻极体话筒都要加一

个直流偏置才能工作。

2.4 PWM 波

PWM 控制技术就是对半导体开关器件的导通和关断进行控制，使输出端得到一系列幅值相等而宽度不相等的脉冲，用这些脉冲来代替正弦波或其他所需要的波形。按一定的规则对各脉冲的宽度进行调制，既可改变逆变电路输出电压的大小，也可改变输出频率。

三、方案论证

3.1 系统结构

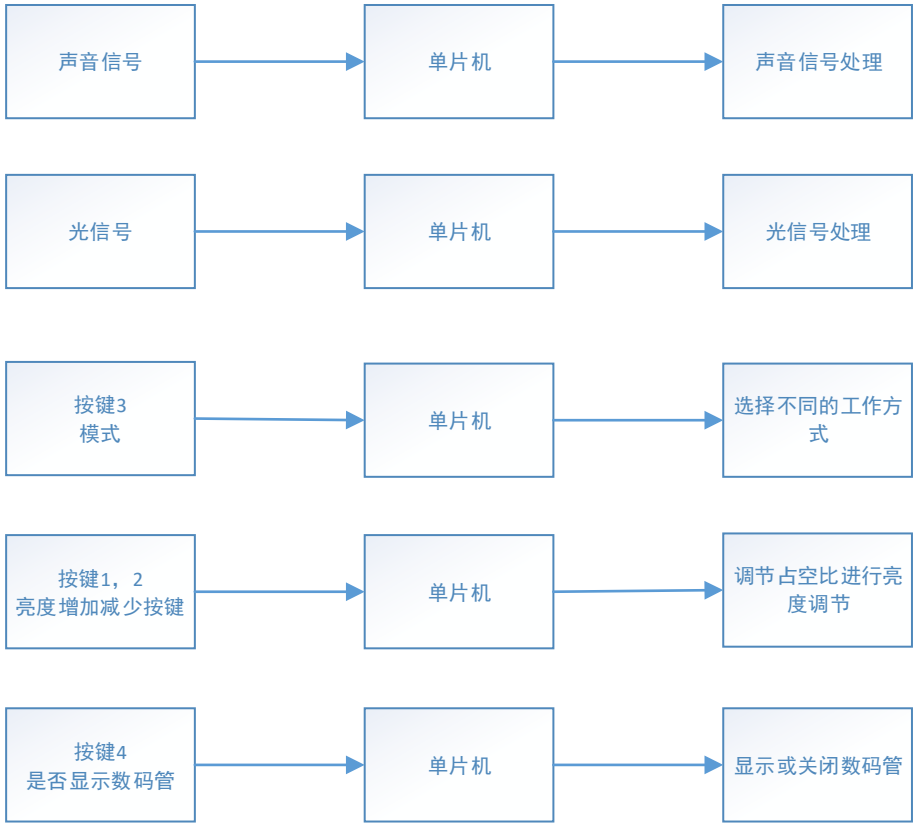


图1：系统总体框图

如图 1 所示，系统总体上是利用单片机采集声音信号和光信号，再通过单片机在不同的模式下对声音信号和光信号进行不同的算法处理。采用四个按键，分别对系统所控制的灯光进行控制，两个按键用作于灯光的亮度控制，1 个按键用作于灯光的模式选择，1 个按键用于开启关闭显示数码管。

3.2 光敏电阻

光敏电阻在不同的光照下有不同的电阻值，经测量，无光时 41K，全光时 3.8K。因此可以和一个电阻串联，利用光敏电阻在不同光照情况下的电阻不同，进而分压不同，再通过电位器的分压进行比较，从而判断光信号的有无。

3.3 声音信号的检测

采用驻极体话筒，经过单级放大之后会有一个电压，当有声音时，会产生一个扰动，进而通过比较器，可以用来检测声音的有无。

3.4 电源选择

由于方案要求只提供 5V 电源，而我们采用的单片机是 3.3 供电，所以我们采用 LM1117 芯片进行电源电压的转换，而 5V 电源我们采用 USB 供电，这是因为 USB 供电比较常见与方便。

3.5 驱动电路的选择

在本次设计中，主要考虑的是 LED 的驱动和数码管的驱动，由于 C8051F310 的输出电流仅有几 mA，不足以驱动这些电路，因此需要加驱动电路进行驱动，对于 LED 灯，我们可以通过 NPN 三极管进行驱动，对于数码管，我们用 CD4511 译码器进行段译码与段驱动。利用 LS1392-4 译码器，可以对四位一体数码管进行位选通与位驱动。

3.6 调试电路与复位电路

调试电路与复位电路采用老师提供的电路图，这一部分没有什么问题。方案可行。

四、硬件电路设计

4.1 微处理器

在本次设计中，我们选用是 C8051F310，因为一方面在上学期我们学习使用过这一款单片机，对这一款单片机有所了解与掌握，上学期在单片机实验中，我也做过二阶段，对这一款单片机的编程思路也掌握的比较好。另一方面，这一款单片机有丰富的中断资源，能尽可能多地满足我们的中断要求，此外，C8051F310 还能单独为某一端口配置 PWM 波，这也能满足我们的需求，综合以上原因，我们最终选定这一款单片机作为我们的 CPU。

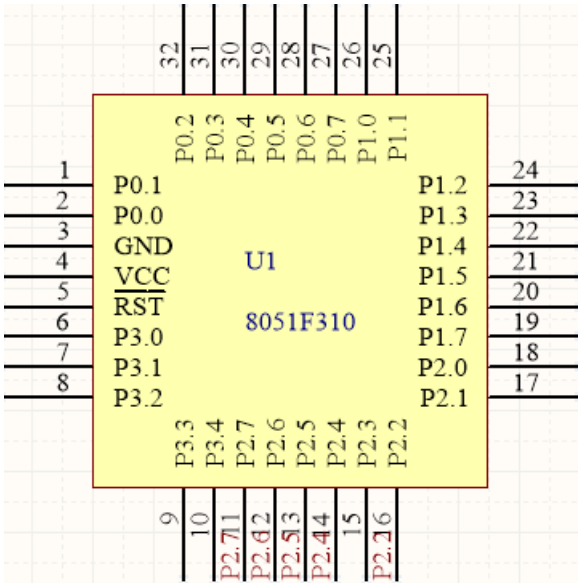


图 2　C8051F310 原理图封装

4.2 调试电路和复位电路

本设计采用老师所给的调试电路与复位电路，为手动复位，该电路操作简单，且可靠性高

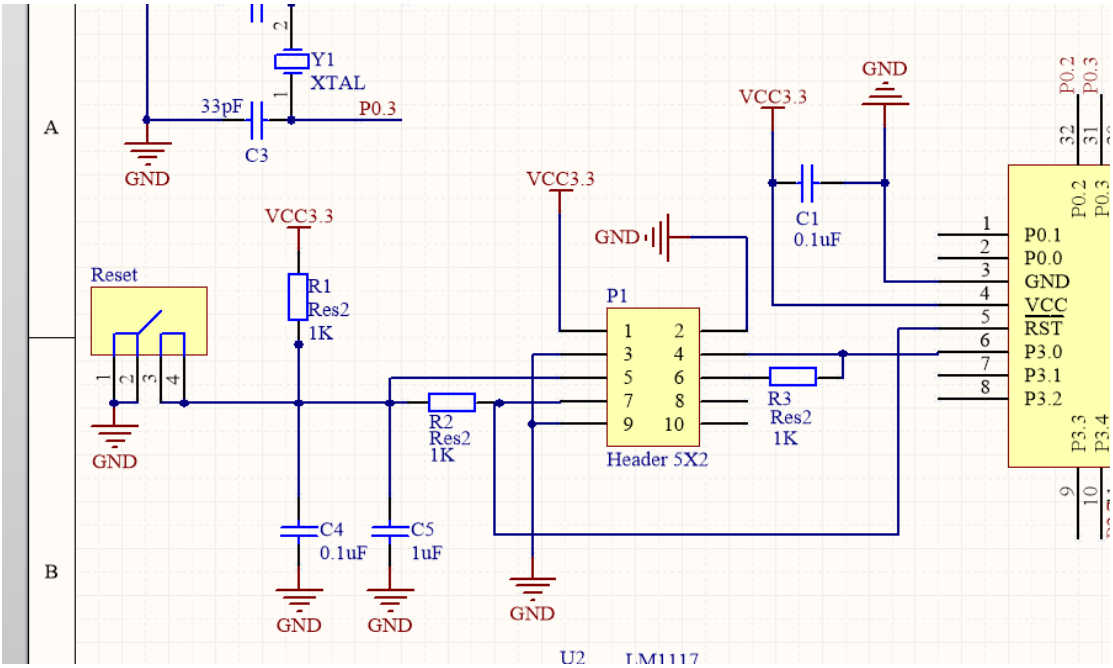


图 3　调试电路和复位电路

4.3 电源转换电路

本设计采用的是 USB 供电，因为 USB 接口在日常生活中使用广泛，比较方便，USB 接口一般输入电流有 500mA 到 1A 不等，在这个范围的输入电流均能满足系统要求，且都能使系统正常工作，另外一方面，由于本设计采用的 CPU 是 C8051F310，这款单片机的正常工作电压是 3.3.V，因此需要一个线性稳压源进行电压转换，构成一个线性电源。

本方案中采用的线性稳压芯片是 LM1117，搭配一些电容，便能构成一个 5V 转 3.3.V 的电源转换电路

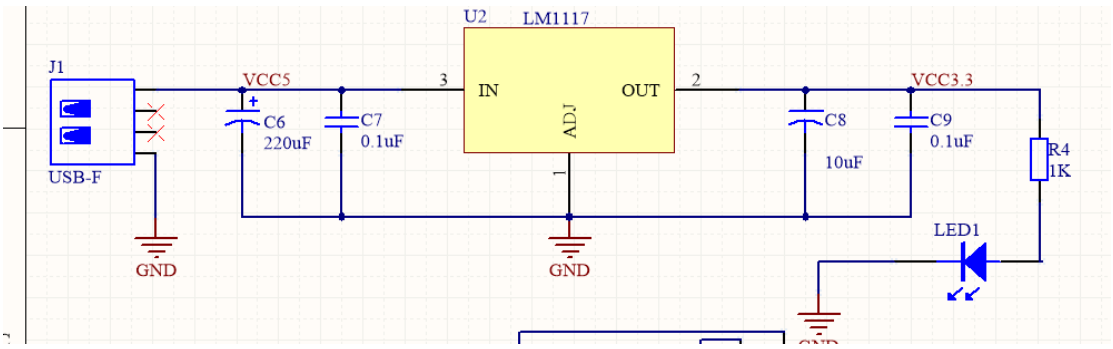


图 4　电源电路

如上图所示为 LM1117 的工作电路。其中，C6 为滤波铝电解电容，用以滤除输入纹波；

C7 为贴片电容，消除 C6 高频感性，高频滤波。C8 为输出滤波电解电容，用于滤除输出纹波，C9 则滤除高频，3.3V 输出端则搭载一个 LED 用于电源指示。

4.4 按键电路

由于我们所需要的按键少，所以并不采用矩阵键盘，二是采用四个独立按键。并用 4 个 I/O 口检测按键的状态。当按键按下时，相应的 I/O 口为低，当键未按下时，相应的 I/O 口为高。

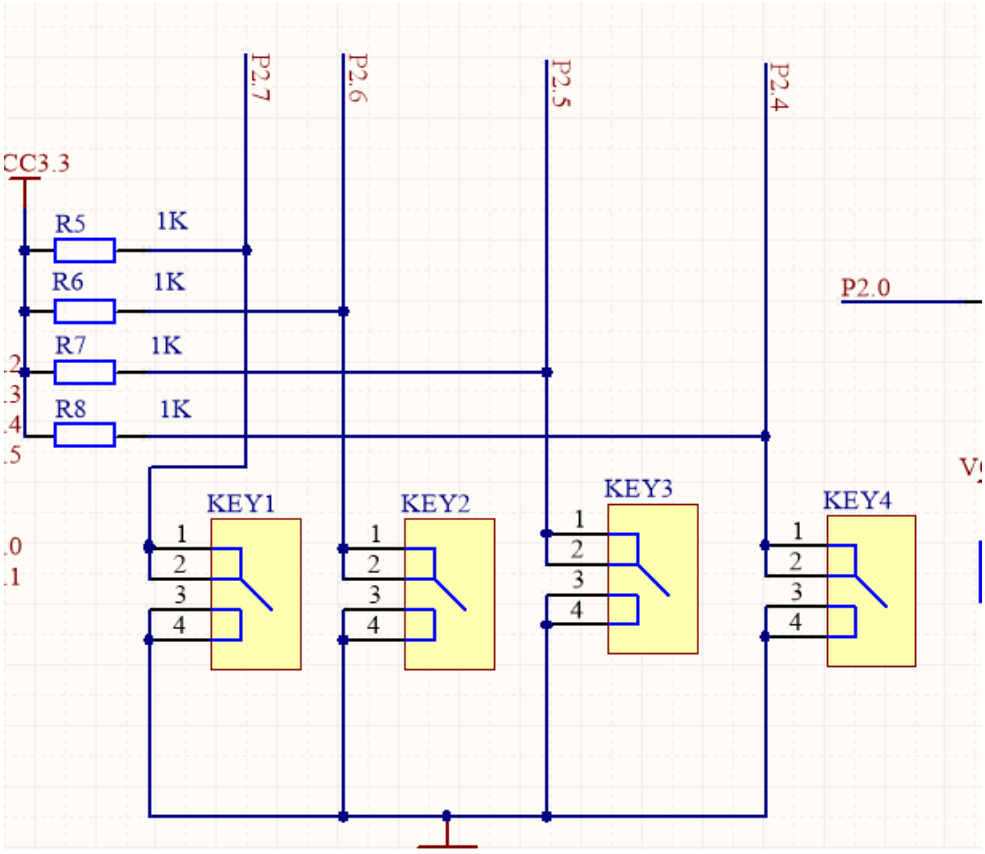


图 5　按键电路

4.5 光控电路

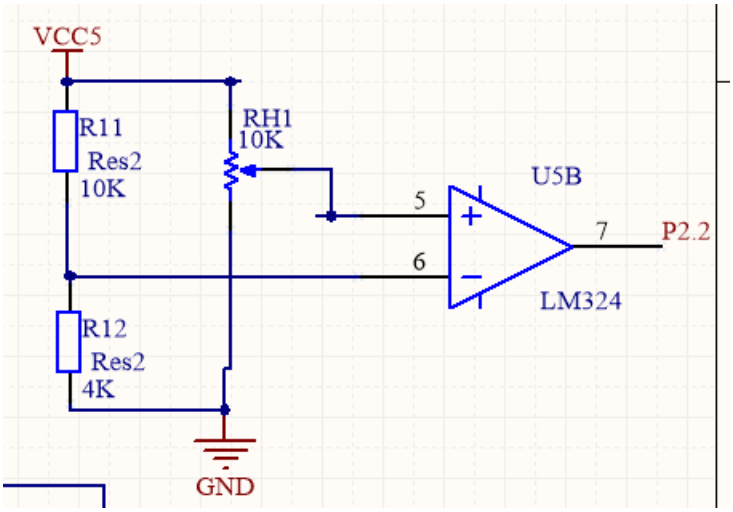


图 6　光控电路

如图 5 所示，R11 是一个 10K 的电阻，和 R12 光敏电阻串联。RH1 是一个滑动变阻器，提供一个可调的基准电压。LM324 在无反馈电路时，看作一个天然的比较器，比较光敏电阻上的电压和基准电压。当无光时，电阻大，分压大，将大于基准电压，比较器输出为 0，当有光时，电阻小，分压小，小于基准电压，输出为高电平，由于比较器并不理想，输出的高电平在 5V 的情况下输出为 3.7V，但能被单片机检测到。通过调节电位器，可以调节基准电压，进而调节光照阈值。

4.6 驱动电路

4.6.1 LED 驱动电路

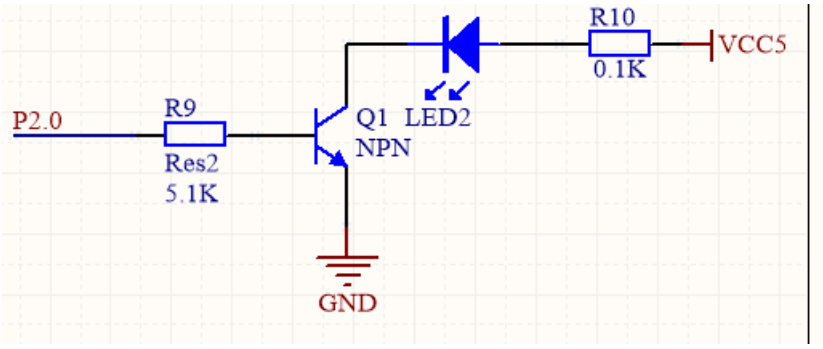


图 7　LED 驱动电路

LED 采用单个 NPN 三极管进行驱动，当单片机的 I/O 口输出为高电平时，将大于三极

管的开启电压，使得 LED 导通，R10 为限流电阻。当单片机的 I/O 口为低电平时，三极管则为关闭状态，三极管不导通。LED 灭。

4.6.2 四位一体数码管驱动电路

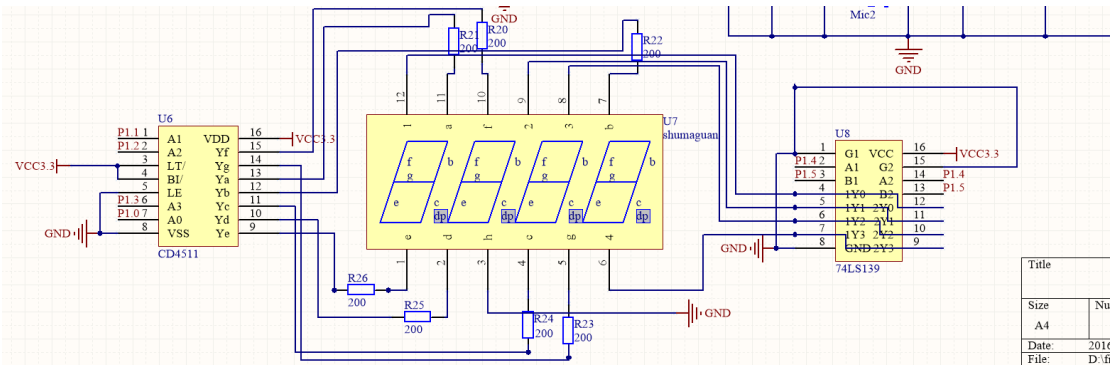


图 8　四位一体数码管驱动电路

如图 8 所示：对于数码管，我们用 CD4511 译码器进行段译码与段驱动。利用 LS139 2-4 译码器，可以对四位一体数码管进行位选通与位驱动。为了保护数码管，应该加限流电阻，为了使每段亮度一致，我们在每段都加上 200 欧姆的电阻作为限流。由于，我们在处理四位一体数码管的显示时，每个数码管的占空比实际上只有 1/4，所以为了保证亮度，采用 LS139 并联输出，提高亮度。两个芯片均采用 3.3V 供电。

4.7 声控电路

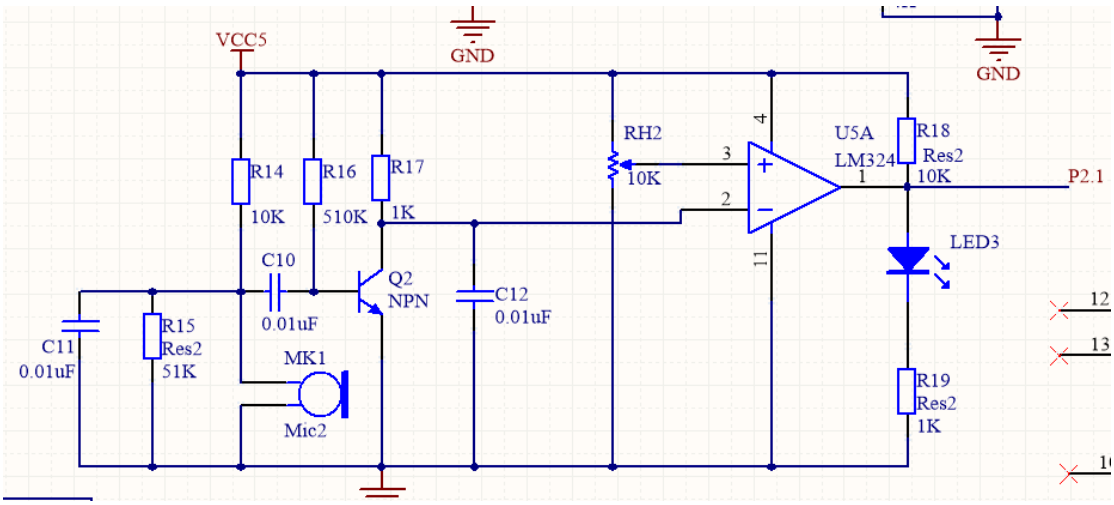


图 9　声控电路

本声控电路采用网上一种声音传感器的数据手册的电路，R14,R15 为驻极体话筒提供偏置电压，C11 为滤波电容，RH2 为电位器提供基准电压，调节 RH2，使基准电压略低于无声

音时 C12 的电压，这样没有声音时，比较器输出为低电平，LED 声控指示灯不亮，当有声音时，声音会上下抖动，经单极放大之后会引起 C12 两端的电压上下变化。声音越大，变化越大，所以 C12 变化引起 C12 的电压低于基准电压时，比较器则会输出高电平，即输出声音信号，此时，LED 指示灯也会亮。同时调节 RH2 电位器，可以调节声音阈值，为了保证有声音时输出为高电平，基准电压不能高于 C12 两端的静态电压，两者电压差越小，检测声音越敏感，这部分做了实验，实验如图 10 所示

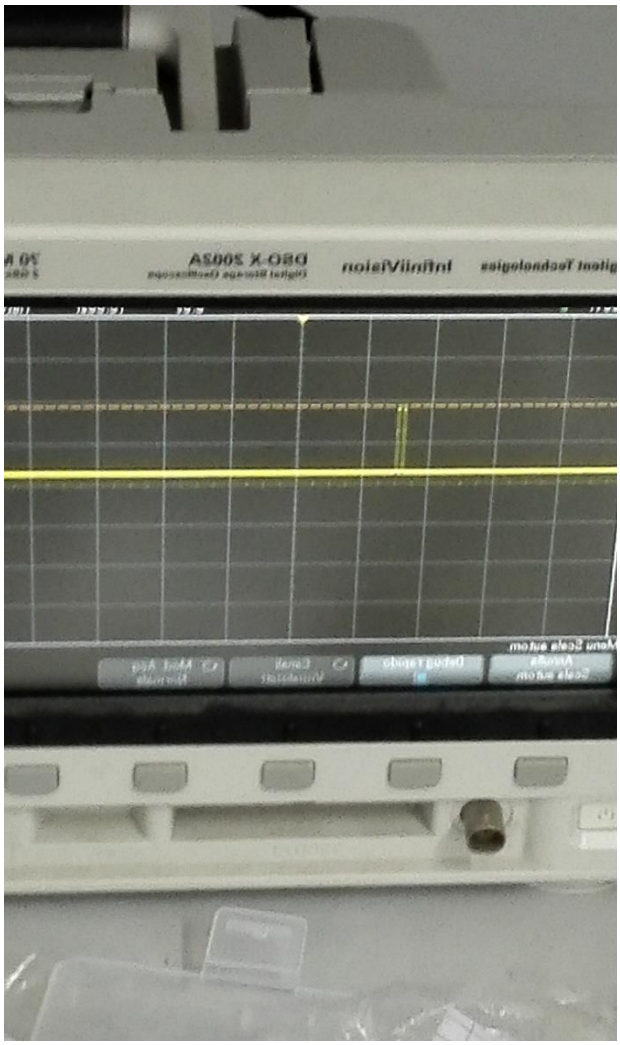


图 10　声控电路声音信号波形图

图 10，当拍手时，声音的输出波形图，可以看出有个脉冲，但实际上，拍一次手游多个脉冲，这点，可以在软件上进行除噪。

五、软件设计

5.1 总述

由于在上学期我在学习单片机中并在单片机设计二阶段中用汇编语言开发一个数字秒表，对汇编语言有比较深的理解与掌握，所以在这次设计中，我仍采用汇编语言进行软件设计，并对系统进行优化时，能用软件就不使用硬件，所以用软件去抖，软件去噪，以及用软件做波形检测，判断不同的声音，以提高系统的稳定性。在此次设计中，用了 4 个定时器中断，并对一些定时器中断进行复用，用了三组工作寄存器，以及多个位地址。

5.2 程序框图

5.2.1 主程序框图

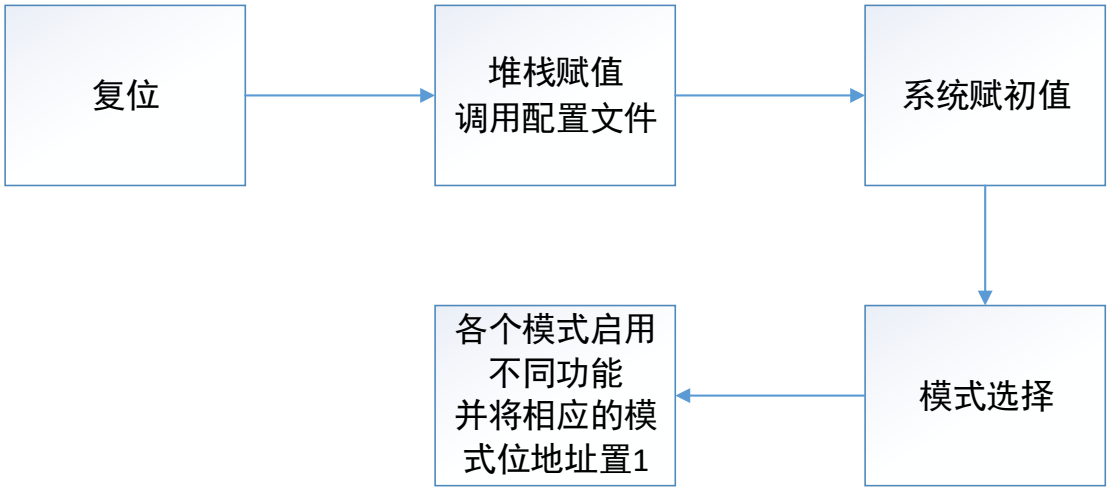


图 11 主程序框图

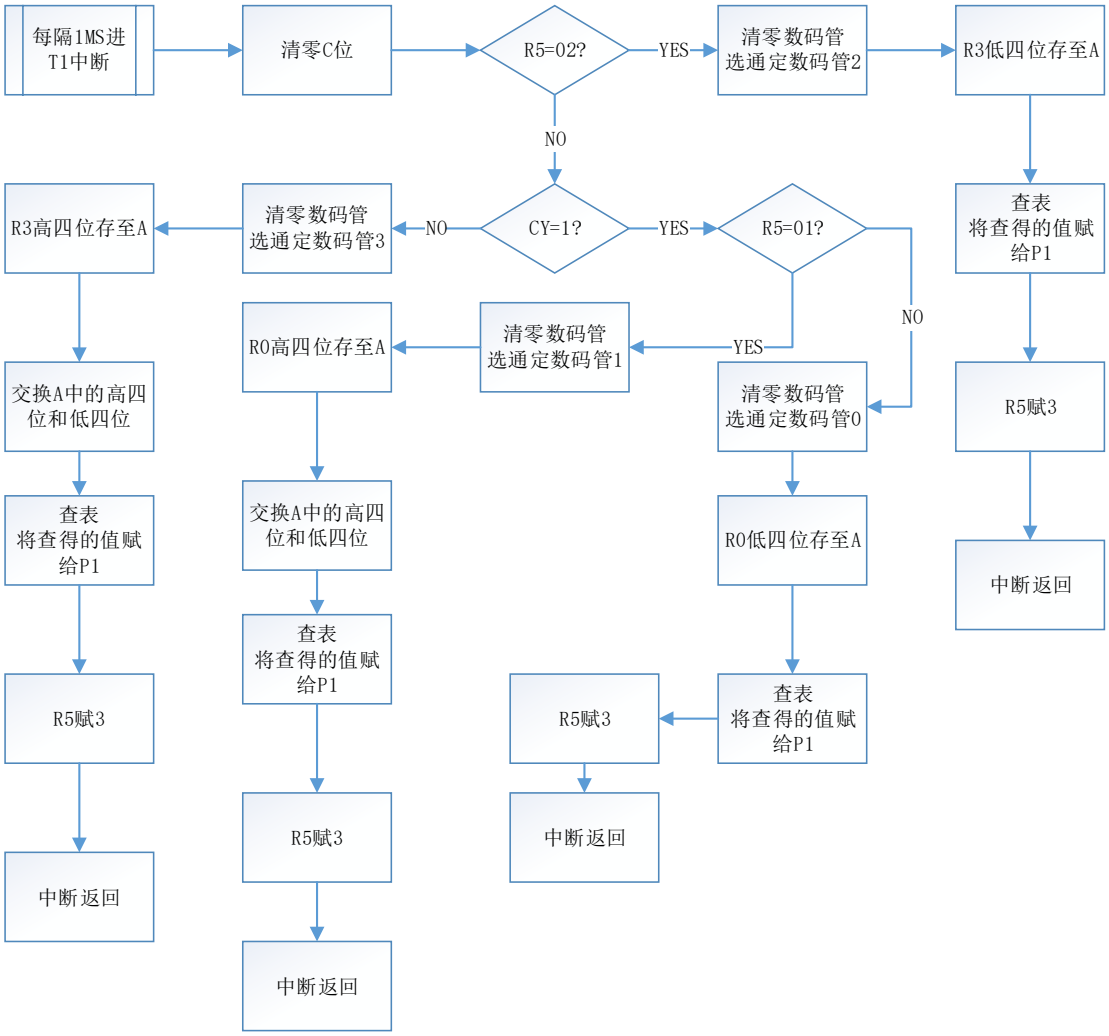


图 12　数码管框图

系统复位之后，对一些地址和寄存器及中断赋值初值，之后进入模式扫描选择，在不同模式下启用不同的功能，并将相应的模式位地址置 1，之所以设立了模式位地址，是为了便于之后的模式判断以及在不同的模式下，对按键和中断进行复用。系统默认的的模式是模式 0，即智能灯模式。要改变模式，只能通过按键 4 进行模式的改变。通过定时中断 0 每隔 1ms 扫描一个数码管，对数码管进行赋值，另外由于我们采用的是 4511 译码器，所以我们并不需要查表，可以直接对 P1 口低四位输出

5.2.2 模式 0 程序框图

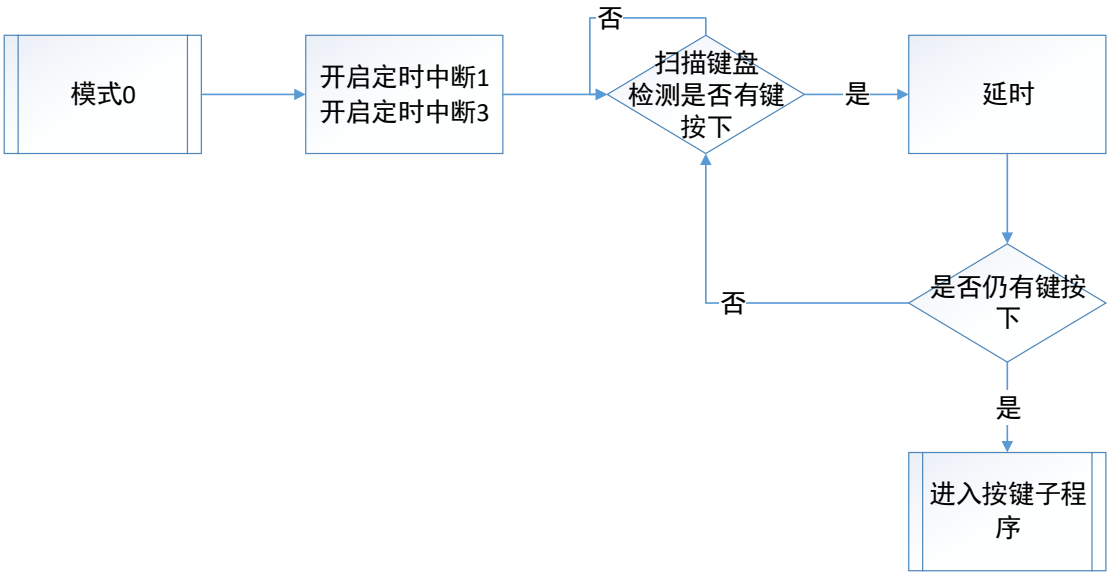


图13 模式0程序框图

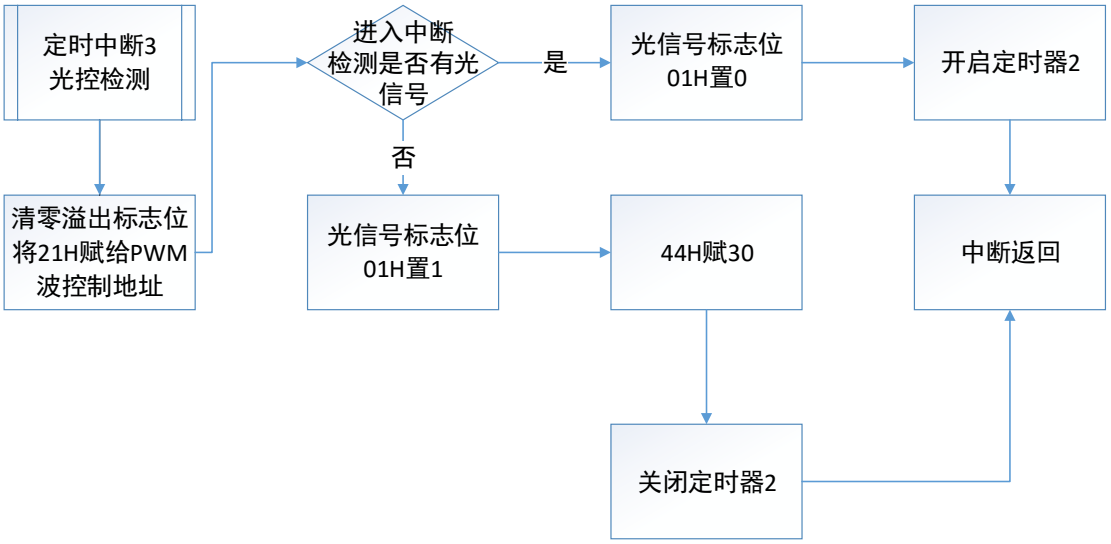


图14 光控检测流程图

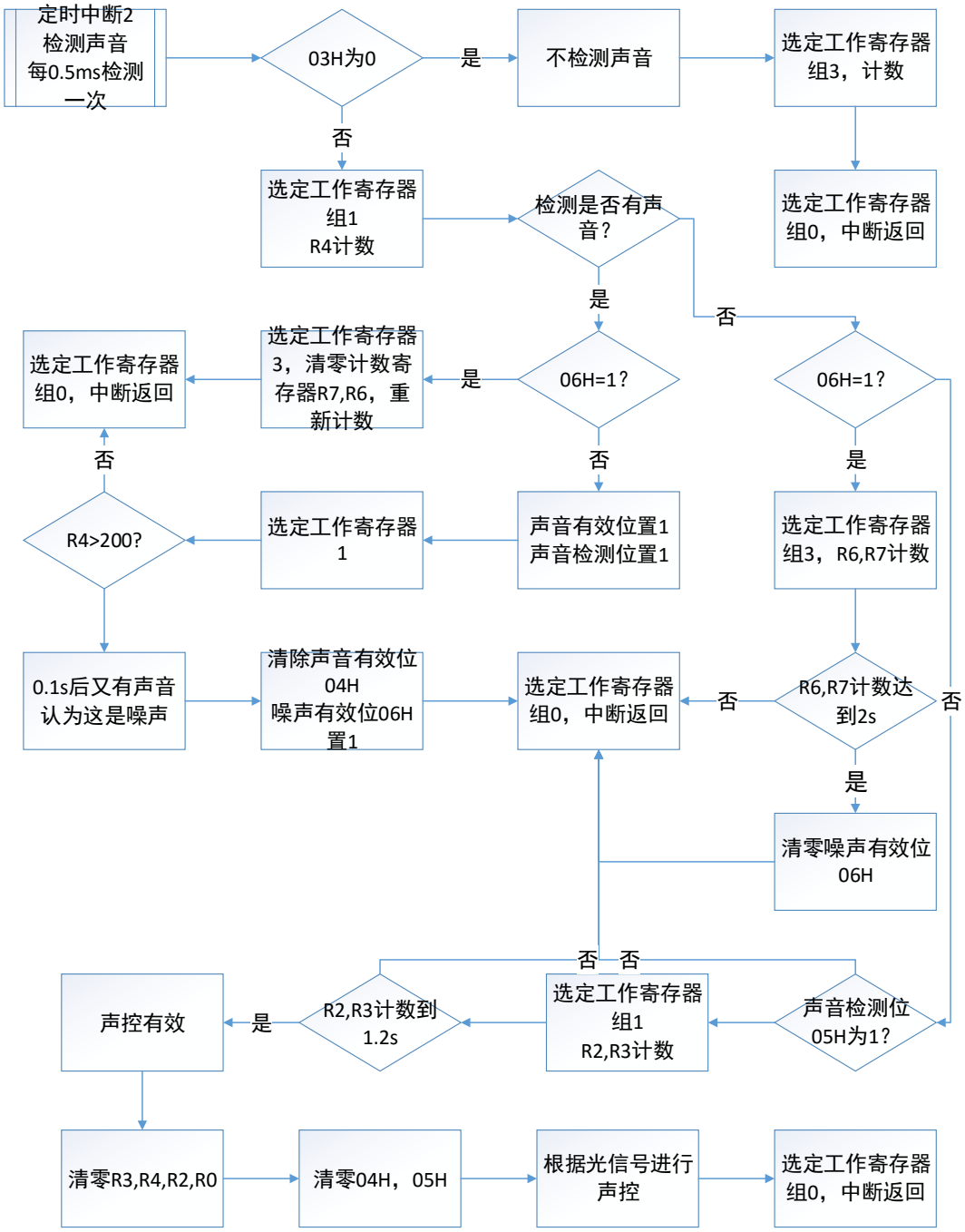


图15 声控流程图

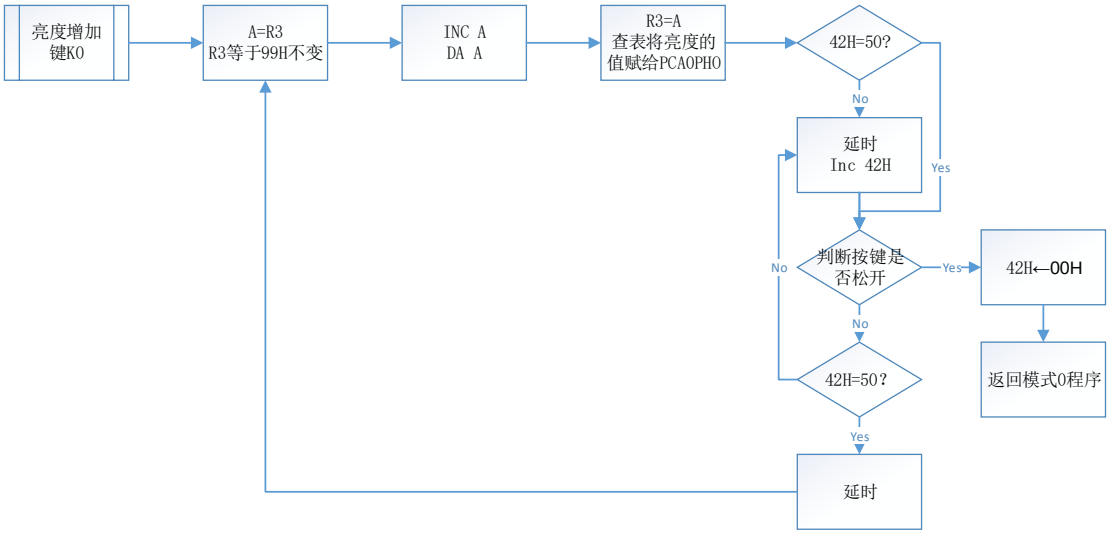
模式 0 可以同时检测声控与光控，当无光时，才可以对声控进行控制，声控光控均用中断实现，定时器三可以 16 位定时，由于本设计采用的时钟频率是 3.0625MHz，定时器均采用时钟频率的 12 分频，因此最多可以定时 65536×(1/3.0625)×12us=256794us=0.25s，因此我采用最长的定时时间，即大概 0.25s 检测一下光信号，将 21H 这个地址作为 PCA0CPM0 的等效地址，因为我们在控制 PWM 波的关闭和开启时，只需要改变

PWM 波的控制地址的某一位，而 PCA0CPM0 这个地址不能进行位操作，所以我便选择一个可以位寻址的地址作为它的等效地址。当有光和无光是选择一个位地址作为标志位，便于声控利用。

当有光时，则应关闭灯光，可能会是局部手电筒或者其他原因照射一下引起短时间有光，为了消除这种影响，我采用只有连续 3s 内都有光才会关闭，而这 3s 的延时显然不能直接用延时程序，这样就会不稳定，则耗 CPU 的资源，所以利用 T2 定时器去计时，每隔 T2 定时器每隔 100ms 进入一次，当 T2 定时器进入 30 次有光的状态还没有改变时，则认为一直有光，那么就关闭灯光。

对于声控检测，由于我们主要是以拍手为信号控制灯的亮与灭 ，但事实上,我们的环境还存在着其他声音，比如说我们人之间肯定是会有交谈的，不能说我们人之间的交谈会引起灯光的反复开关，以及可能会有其他的声音影响，为了消除这一些声音的影响，我们必须将这些声音区分开来，但是我们使用的是驻极体话筒，又是数字输入，所以我们没办法对声音做频谱分析和强度分析，后来我经过各种检测，发现拍掌声的时间一般不超过 0.1s，而其他的声音将大于这个时间长度，因此我通过检测这些声音长度判断声音是否准确，是否是我们需要的拍掌声，当大于 0.1s 到 1.2s 内没有声音，我们才认为这个声音是有效的，当在这个范围内再次有声音时，系统将认为是噪声，最常见的噪声即是我们的说话声，所以一旦捕捉到这个噪声，系统则认为人在说话，要 2s 内连续没有声音才会消除这一状态，当这一钻个消失，才会再次检测。

之后为了消除按键声带来的影响，每次按键后的 1s 内都不检测声音。



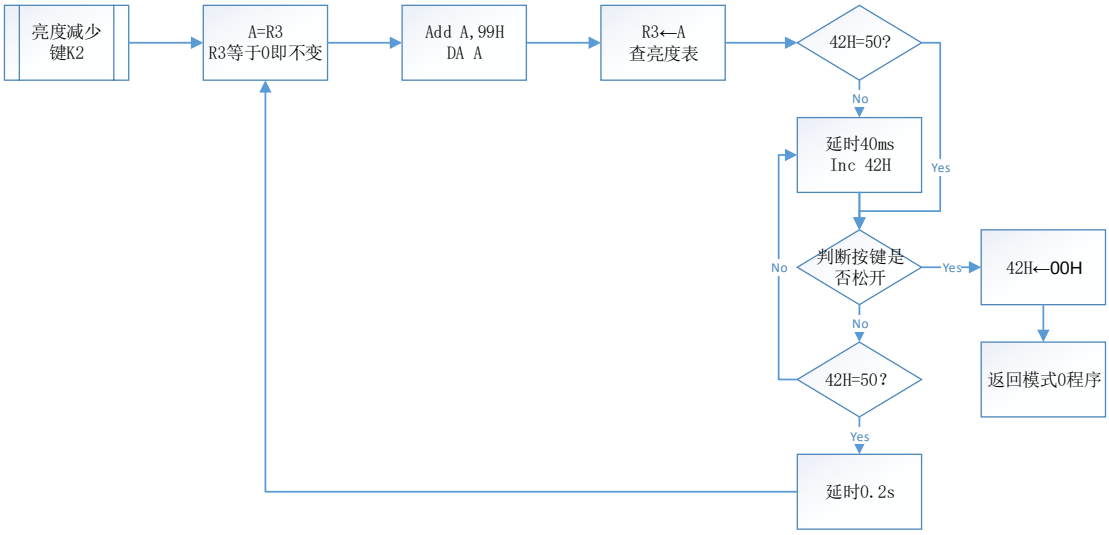


图 16 亮度增加减少键框图

如图 16 所示，当按键 K1,K2 可以分别增加灯的亮度，由于 8 位 PWM 波输出一共有 256 个不同的占空比，但是实验发现，占空比高时的占空比变化对亮度影响变化不及占空比低时的大，也就是人眼对灯光的感觉并不是一个线性的过程，所以我采用查表的方式，先是占空比等级+1，然后再+2，+3，+5。由于有 100 个亮度等级，为了快速调节，长按将会快速调节。

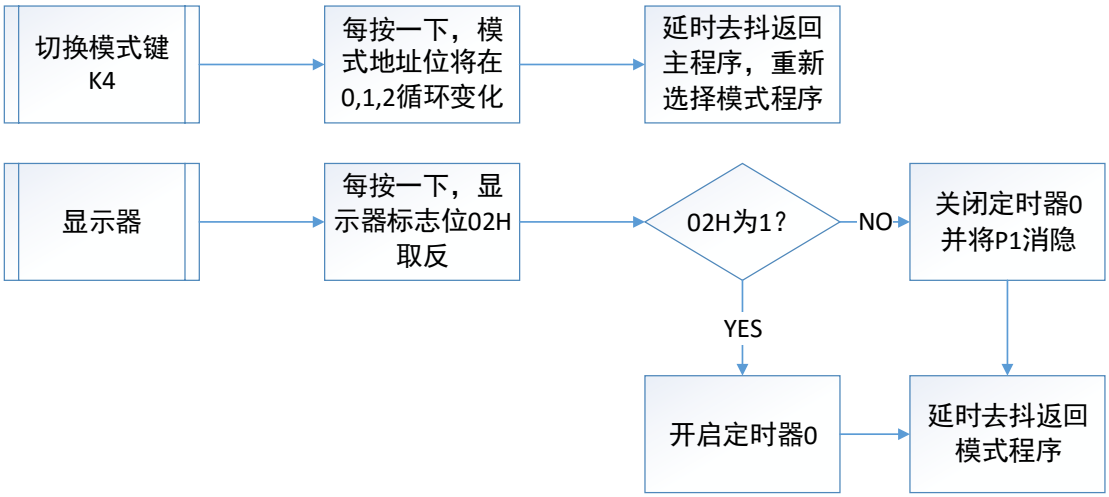


图 17 切换模式和显示按键

K3 为显示器按键，用于开启或关闭数码管

K4 为模式调节键，用于改变不同的模式。

以上按键在模式 0 时进入和结束时均设置了 03H 置 1，为了消除按键噪声

5.2.3 模式 1，台灯模式

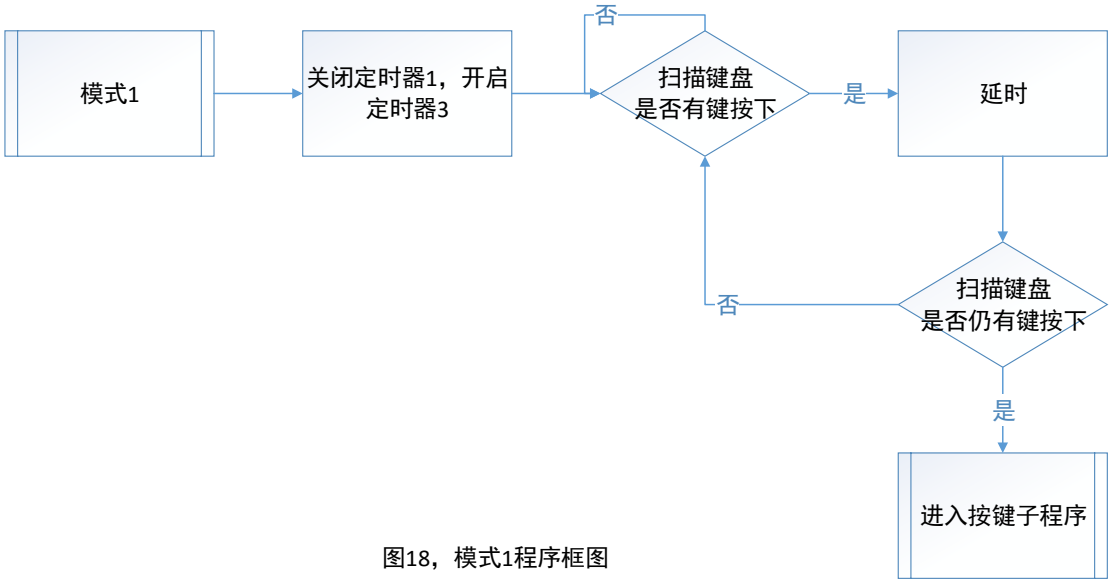


图18，模式1程序框图

在台灯模式下，把声控部分关掉了，其他的键仍和模式 0 进行复用，所以在每个键的的程序加了一些判断模式的操作，若是模式 1 则不进行对 T1 定时器的操作，在模式 1 下，亮度减少到 0 时，再按一下则关闭灯，另外在模式 1 的情况下，光控仍然有效，即台灯在天亮时，它就会自动关闭。

5.2.4 模式 2 过道灯模式

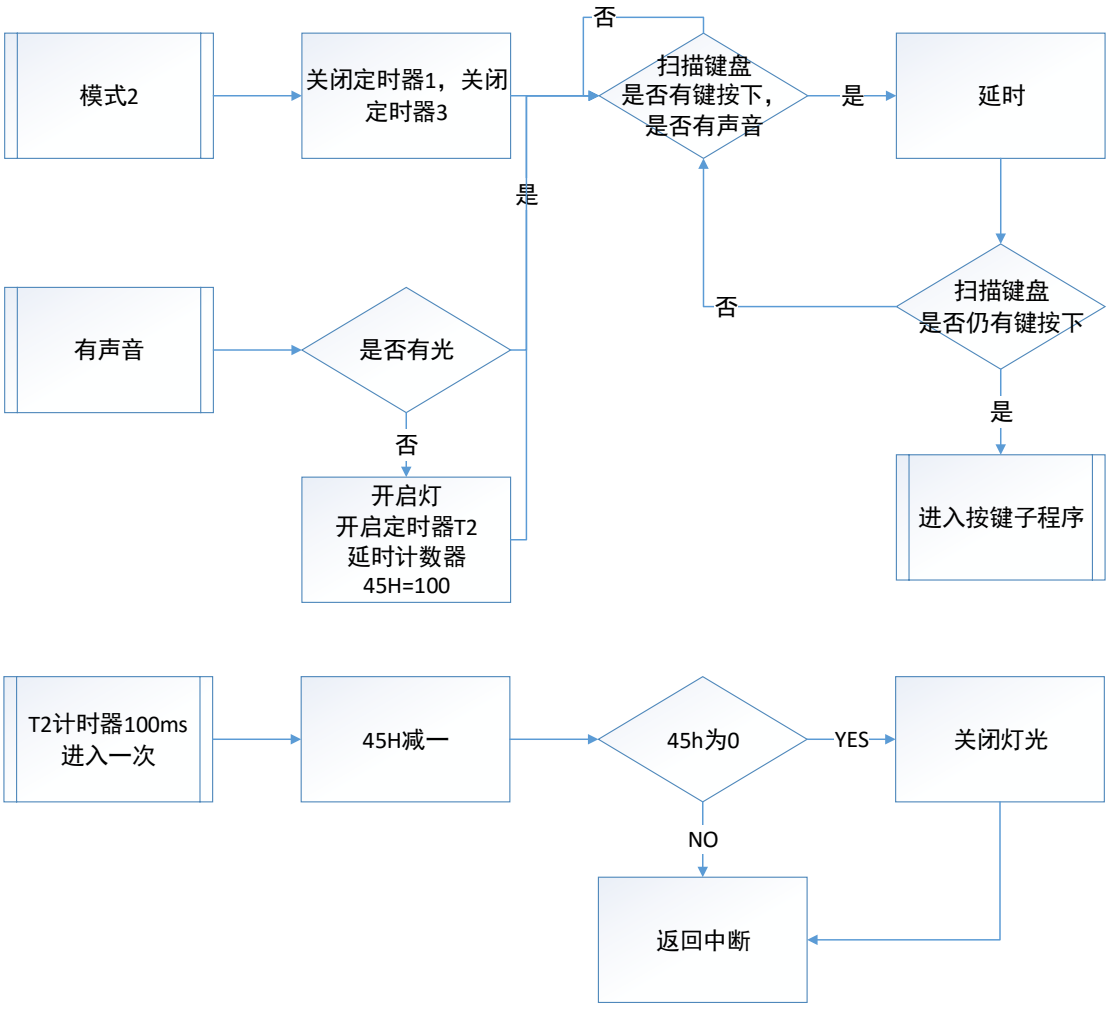


图19，模式2程序框图

在过道灯模式下，只要有响声就会使灯光亮 10s，这是基于家庭中，有时夜晚要上厕所方便开灯的背景下。

5.3 单片机硬件及配置

5.3.1 I/O 口配置

P2.1 做输入的声音信号端口

P2.2 做输入的光信号端口

P2.4-P2.7 做输入的按键端口

P1.0-P1.3 数字输出，四位一体数码管的段控制端口

P1.4-P1.5 数字输出，四位一体数码管位选端口

P2.0 数字输出，控制灯亮端口，PWM 端口输出

5.3.2 配置设置

1、时钟设置，晶振采用 3.0625MHz

2、T0、T1 定时器采用工作方式 2

T2、T3 定时器采用 16 位自动重装

定时器均采用 12 分频

3、PWM 端口输出分配到 P2.0

六、测试报告

首先测试点与电之间是否有短路断路，其实这一部分商家已经做了飞针测试，但是我们为了保险起见，还是将这一部分再次测试了一下。之后确认完好之后进行焊接

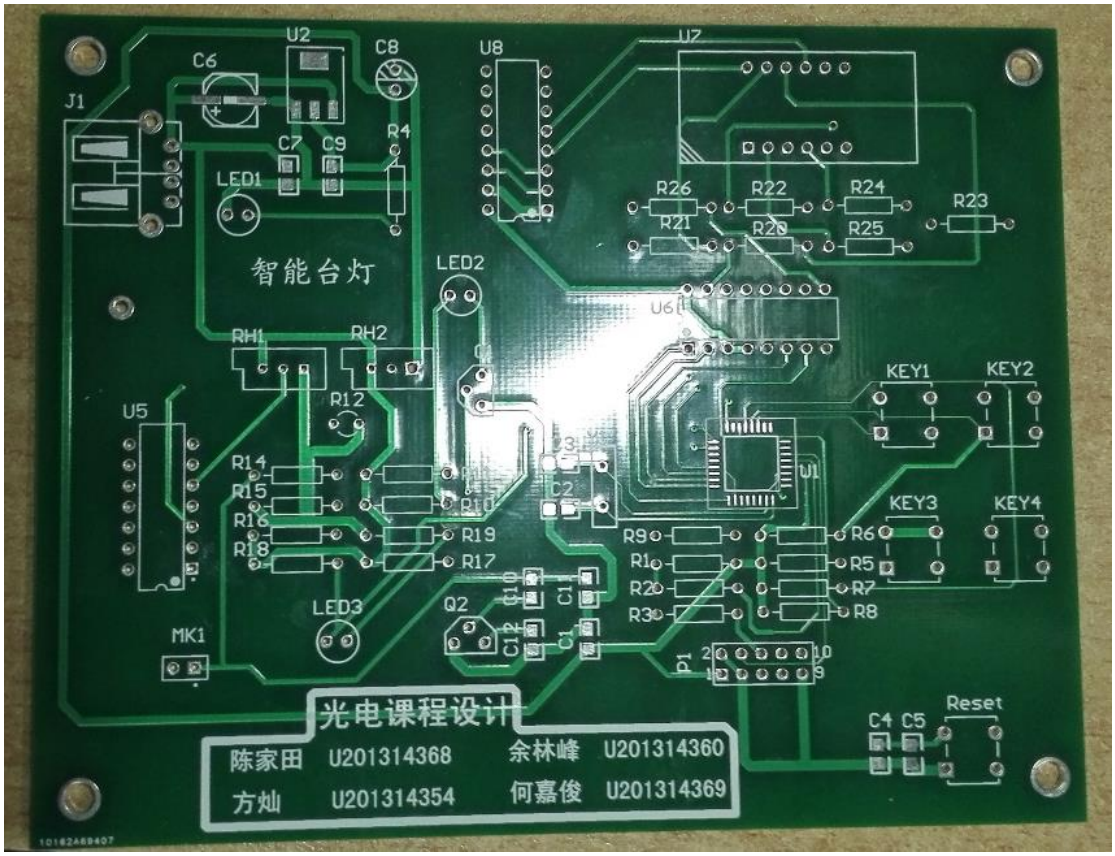


图 20 PCB 板的正面

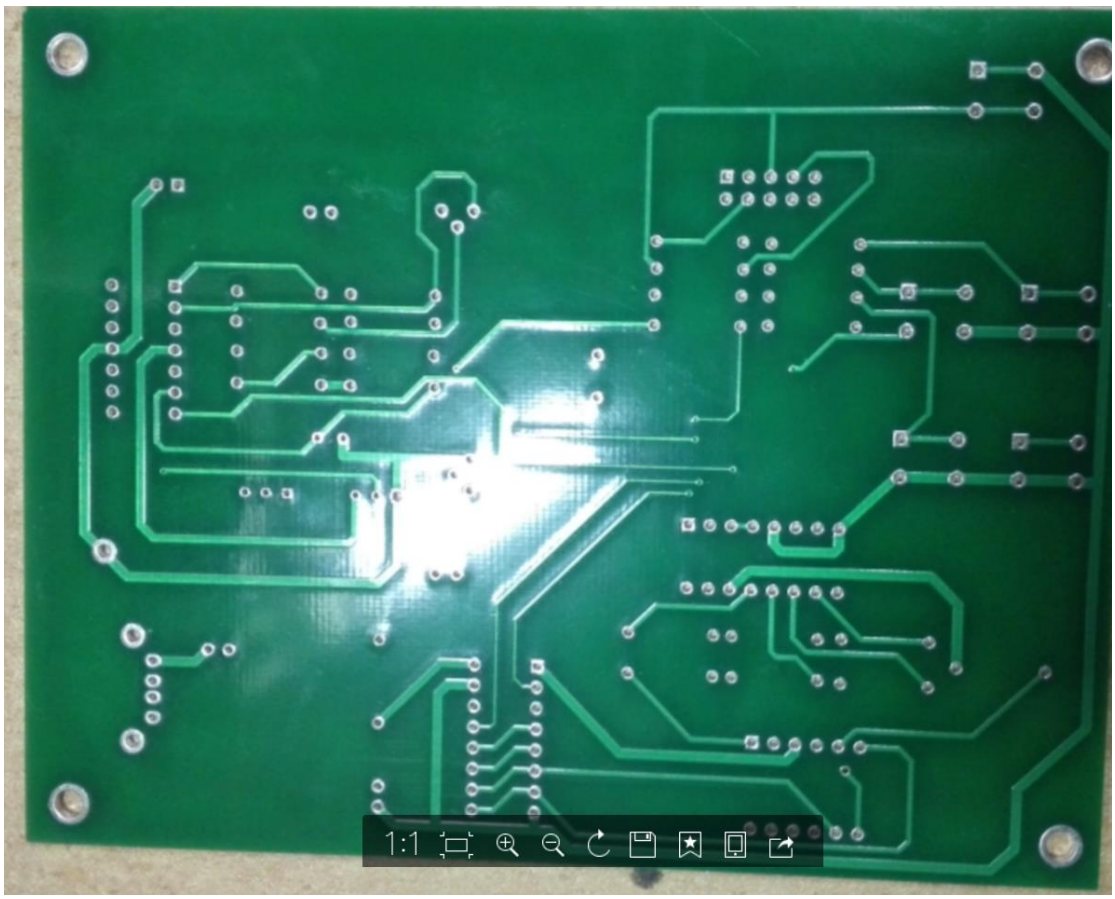


图 21　PCB 板的反面

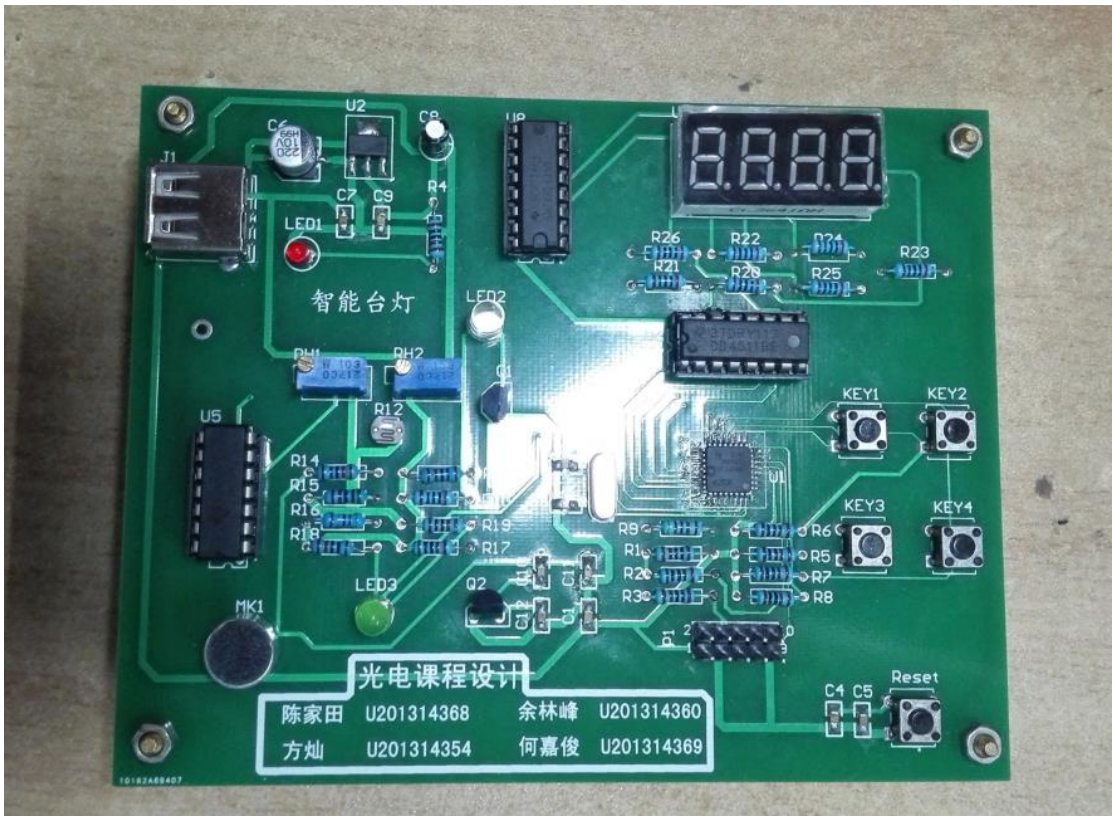
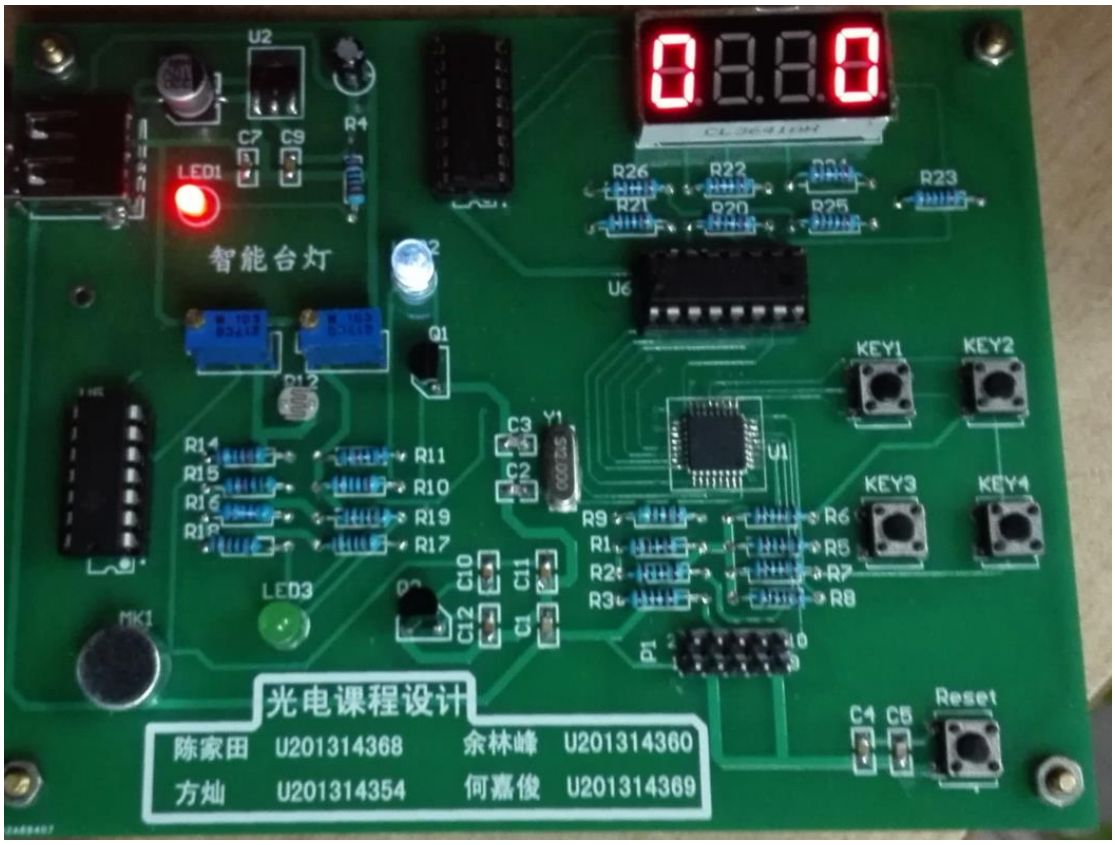


图 21　焊好的 PCB 板正面

焊好之后，我便进行程序烧写，准备测试一下这些驱动是否可用，但是意外来了，板子一直烧不进去，老是报错，说地址被写保护，以为是电脑问题，换了几台电脑试，又以为是调试器问题，也试了其他两个调试器，以为是焊接技术不行，后来叫老师给我们焊了个 cpu，而且换了另外买的 cpu，最后还是不能烧，以为设计问题，找了好久，烧录电路硬是没发现什么问题。又以为是板子问题，又拿了其他组可以烧的板子拿来焊，还是不能烧，就这样搞了将近一个星期，之后我们拿另外一块 CPU 来焊时，发现还是单片机的问题，然后写了下代码做了下输出测试，发现数码管有点不稳，之后发现是电阻没有焊稳，经过调节，最后焊好了。之后测试，输出特性全部正常



七、结论

在这个设计中，灯光控制器能输出 3 种模式，在智能模式下，能比较智能地区分人的说话声和拍掌声，但为了实现波形检测所付出的代价便是，每一次拍掌都会延迟一段时间才会有效。但总的来说，还是实现了比较完美的功能。

在本设计的核心便是对拍掌声和说话声进行检测，包括消除按键的噪声影响。

其他两个模式也均可行，且有一定的应用背景。

当然也有一些部分是可以改进与提高，比如说，可以减少板子的大小以减少成本，不应该采用 4511 做显示管的驱动芯片，因为 4511 经过译码之后能够显示的仅仅只有 9 种情况，这限制一些其他的应用。

八、心得体会

光电课程设计将是本科阶段最后一个设计，同时也是最难的一个设计，它全面考察了我们对所学的知识的综合应用，从数电模电到单片机到光电探测，以及各种软件的使用，在这个课程中都让我们得到很好的锻炼与前期课程的巩固。在确定要设计的一个产品的基础上，从原理图的设计，原理的实验验证，原理图的绘制，封装的绘制，到 PCB 的制作，PCB 的投板，再到焊接器件，软件设计与编程调试，只有整个流程都不能有任何错误，才能做出一个成功的产品，在这个过程中也体会到了产品的开发艰辛，但同时感觉到了完成出一个产品所带来的一种成就感和满足感。

在本次设计中，从做实验开始到 PCB 的绘制再到代码的优化和调试，前前后后花了不少时间，跑了不少实验室，在光控和声控的电路敲定之后，开始叫组员绘制原理图，然后我再去加封装，在加封装的时候，三极管和电位器的封装的引脚是反的，辛亏仔细核对之后才去投板，发现这一重大错误。在设计驱动数码管电路的时候，我们忽略了单片机的输出电流很小，必须外加驱动电路才能使数码管正常工作，辛亏给老师检查了一遍，老师指出了这一问题。

对于画 PCB，由于之前除了在电工实习时做过一次 PCB，其他时间就再也没有接触过了，而且当时还是单面板，又是自动布线，所以基本这一块是从零开始学。为此，我反复学习，反复请教老师，然后再回去修改，前前后后修改了好几次，最后终于布出一个较为理想的 PCB 板，但实际上还有修改的空间，我的板子布的太大，这如果开发产品，这一部分的费用也会随之提高。

最后便是写代码，因为焊好 PCB 器件之后的代码工作就全部交给我来做了，当时借来调试器准备烧录时，一直是写保护的状态，以为是电脑问题，换了几台电脑试，又以为是调试器问题，也试了其他两个调试器，以为是焊接技术不行，后来叫老师给我们焊了个 cpu，而且换了另外买的 cpu，最后还是不能烧，以为设计问题，找了好久，烧录电路硬是没发现什么问题。又以为是板子问题，又拿了其他组可以烧的板子拿来焊，还是不能烧，就这样搞了将近一个星期，之后我们拿另外一块 CPU 来焊时，发现还是单片机的问题。经过整整一个星期到 14 周周日才把这个问题解决。

解决了烧录，但是 15 周就要验收了，于是我周日晚上就熬夜到 4 点，把基本功能写好，周一便拿去给老师看一看，当时也没想着要验收，只是想寻求老师一点建议，老师看了我的之后建议我能否用软件实现人的说话声和拍手声。在之后几天里，我便一直熬夜到凌晨 3、4 点，研究了几种算法来实现这一功能，最后选取一个比较成功，比较实用的算法，另外又加了 2 个模式，在 16 周周四的上午拿去验收了，验收的时候，由于忽略了光控，一直亮不亮灯，搞得我当时很紧张，之后发现这一问题之后再答辩也忘记一些想说的细节，不过值得庆幸的是，还是得到老师肯定，产品作为样机收了上去，当时感觉太开心了，这么久的付出终于得到了回报。

这个产品的顺利开发，离不开组内各成员的通力协作以及吴老师的帮助，在此表示衷心的感谢。

九、参考文献

【1】康华光 模拟电路 数字电路 第六版

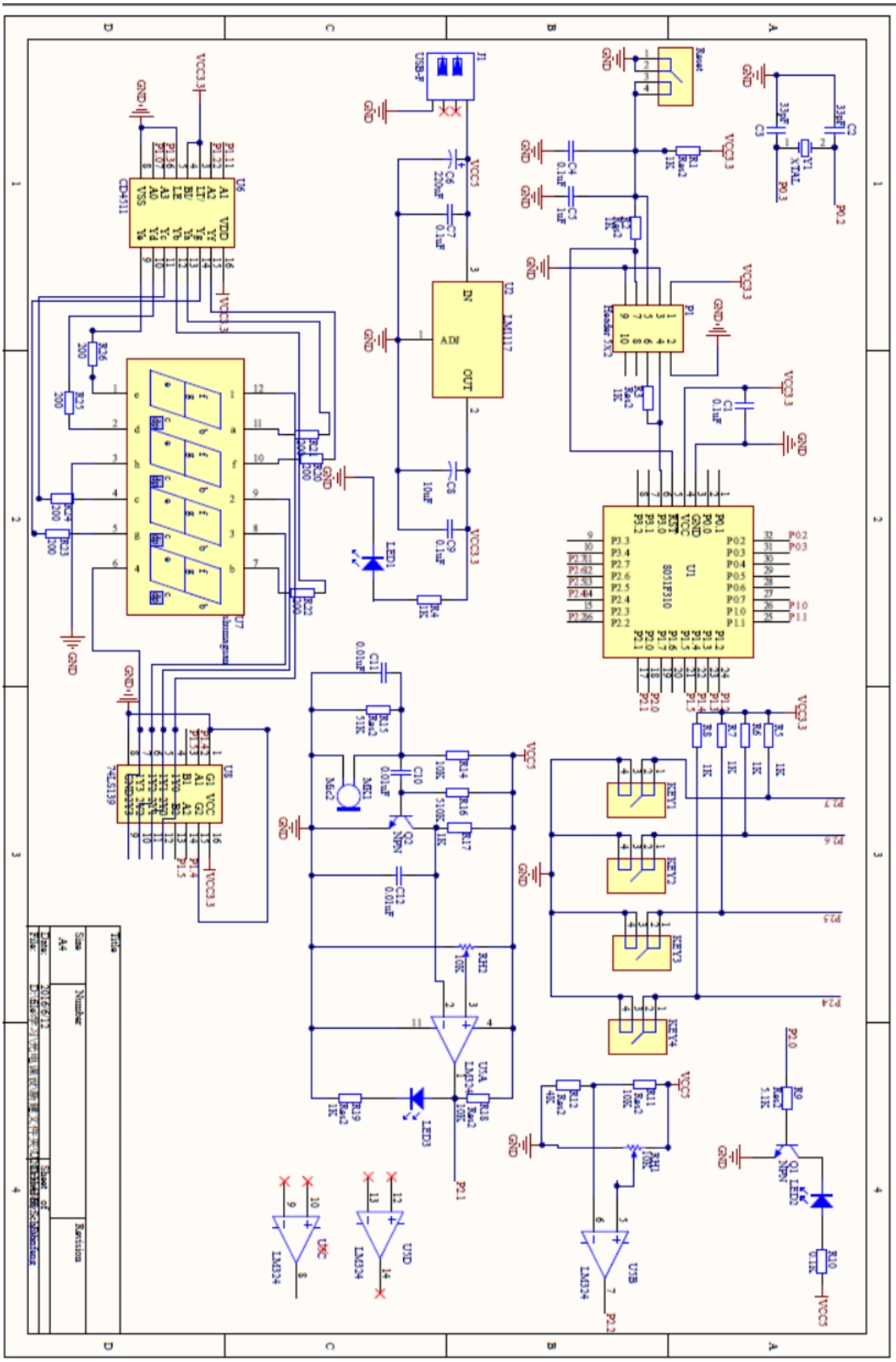
【2】徐汉斌等 单片机原理及应用

【3】高歌 Altium Designer 电子设计应用教程

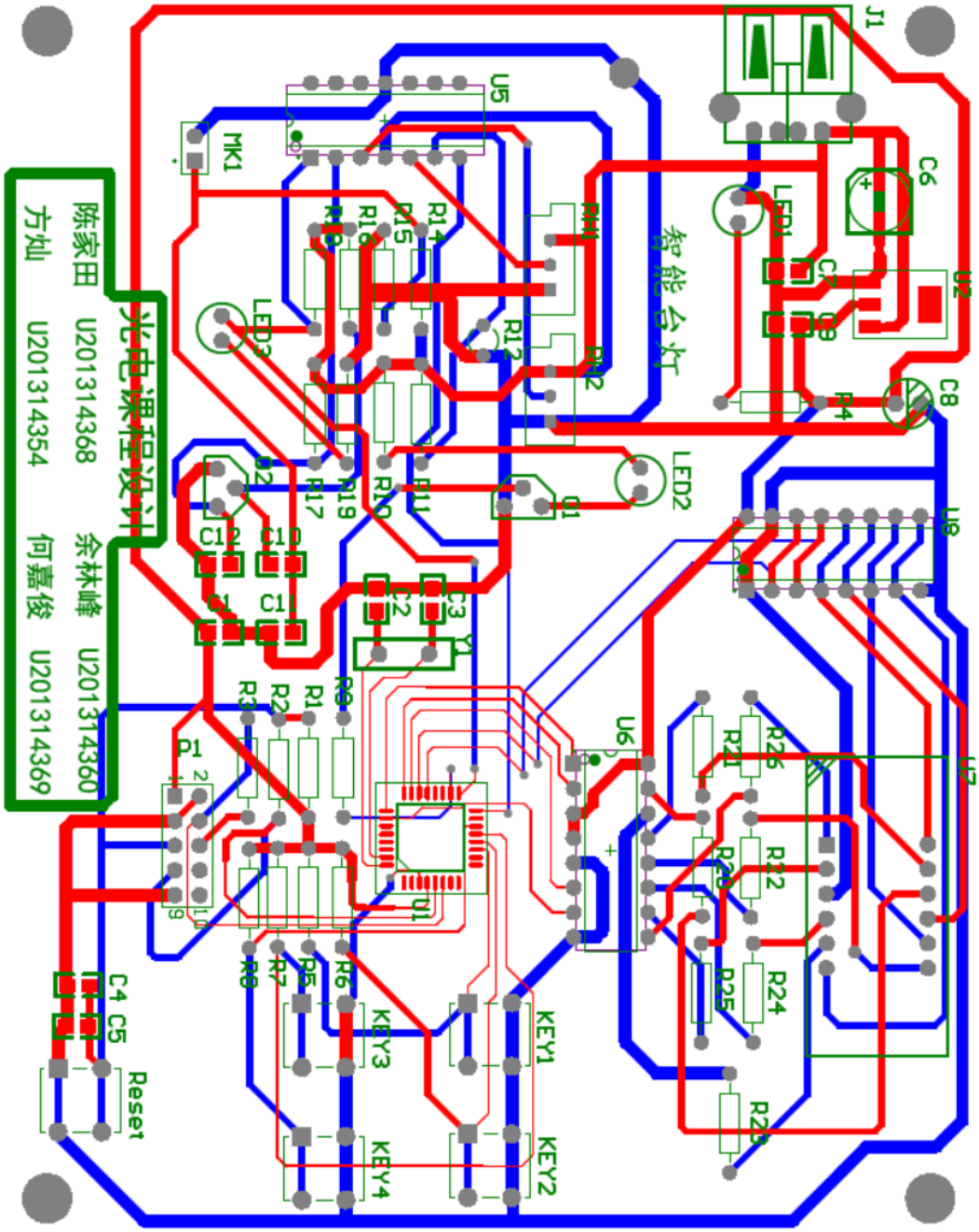
【4】电子技术基础实验（第三版）

【5】各芯片数据手册若干

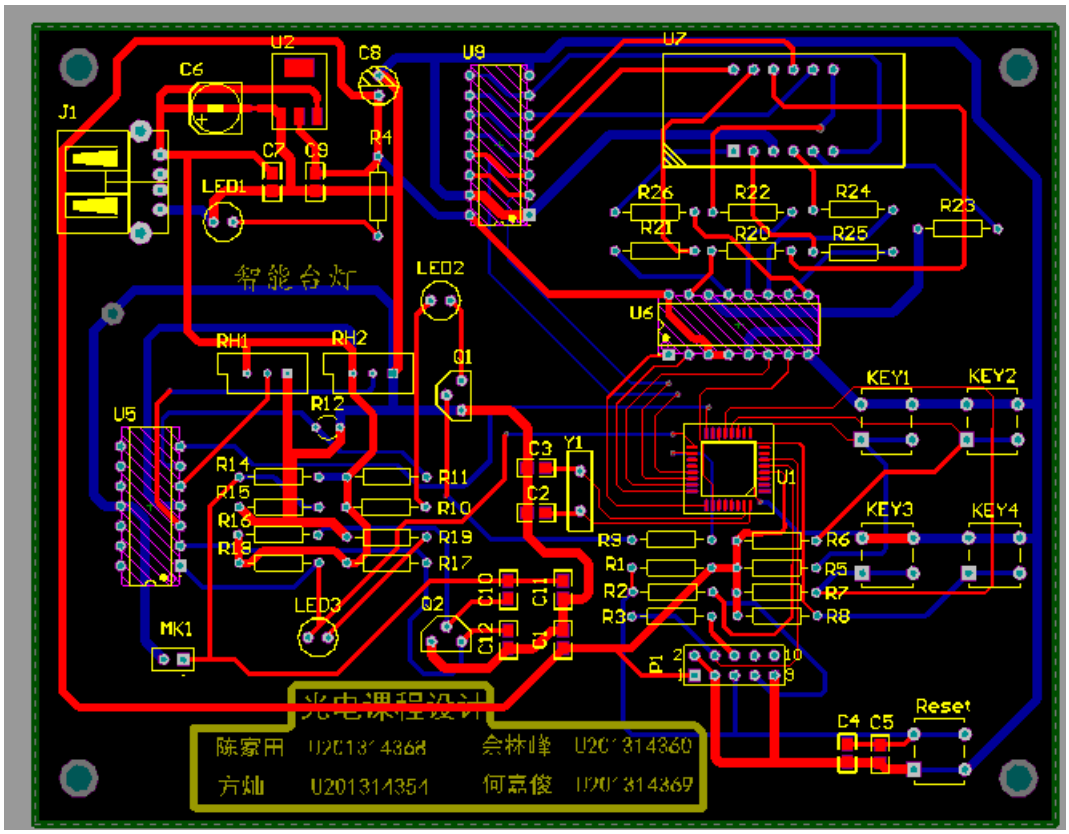
附录一　　原理图



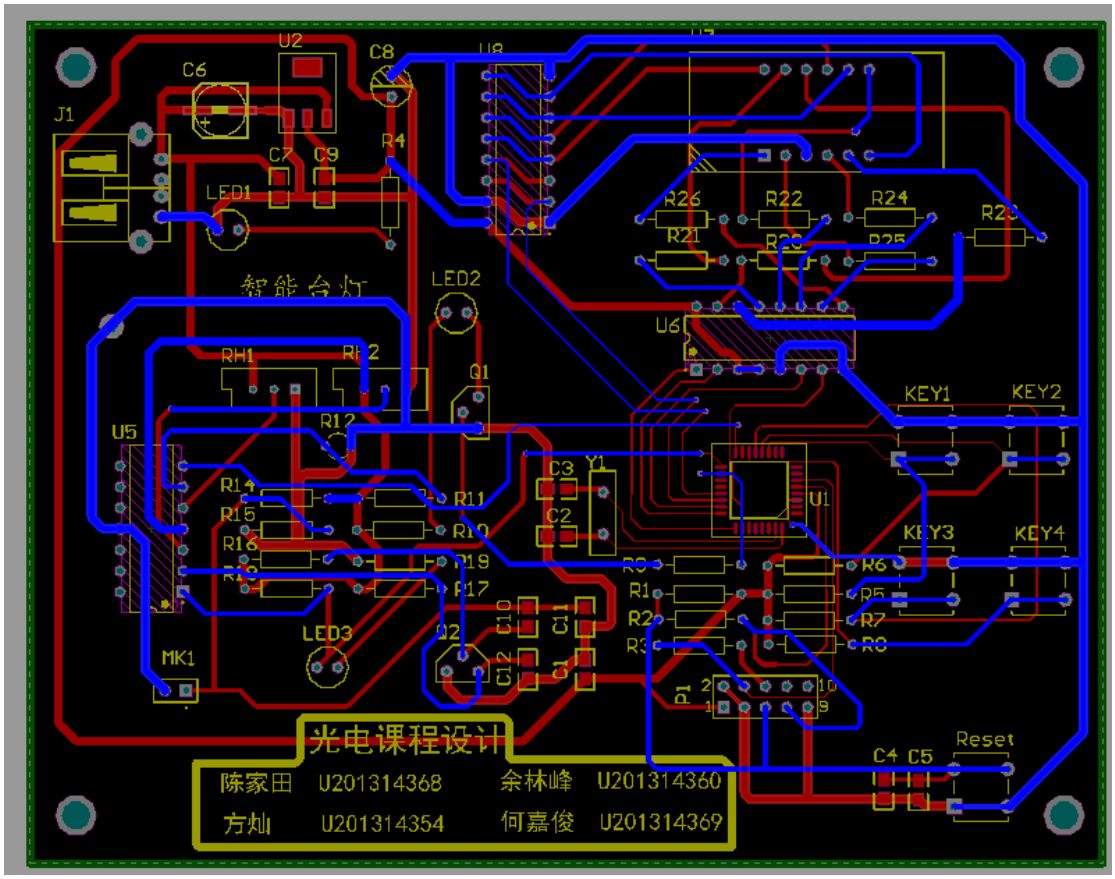
附录二、PCB 板


光电课程设计
陈家田 U201314368
余林峰 U201314360
方灿 U201314354
何嘉俊 U201314369
智能台灯
J1
U5
MK1
LED1
LED2
LED3
C6
C8
U2
U8
U7
U6
Y1
P1
KEY1
KEY2
KEY3
KEY4
Reset

Top layer



Bottom layer



附录三、元器件清单

Comment	Description	Designator	Footprint	LibRef	Quantity
Cap Semi	Capacitor (Semiconductor SIM Model)	C1, C2, C3, C4, C5, C7, C9, C10, C11, C12	0805	Cap Semi	10
	Polarized Capacitor (Surface Mount)	C6	SMD-EC-F6.3	Cap Pol3	1
Cap2	Capacitor	C8	RB1	Cap2	1
USB-F		J1	USB-F	USB-F	1
KEY		KEY1, KEY2, KEY3, KEY4, Reset	KEY1	KEY	5
LED		LED1, LED2, LED3	LEDP	LED	3
Mic2	Microphone	MK1	PIN2	Mic2	1
Header 5X2	Header, 5-Pin, Dual row	P1	HDR2X5	Header 5X2	1
NPN	NPN Bipolar Transistor	Q1, Q2	TOWYB	NPN	2
Res2	Resistor	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26	AXIAL-0.4	Res2	24
Res2	Resistor	R12	R38	Res2	1
RPot	Potentiometer	RH1, RH2	VR5	RPot	2
8051F310		U1	LQFP-32	8051F310	1
LM1117		U2	SOT223	LM1117	1
LM324		U5	DIP-14	LM124	1
CD4511		U6	DIP-16	CD4511	1
shumaguan		U7	LG3461AH	shumaguan	1
74LS139		U8	DIP-16	74LS139	1
XTAL	Crystal Oscillator	Y1	XTAL1	XTAL	1

附录四、代码

```
$include (C8051F310.inc)

        ORG     0000H

        LJMP    MAIN



;--------T0 定时器的中断入口---------
;每隔 1ms 进入中断，对数码管循环选通
        ORG     000BH                   ;--

        LJMP    display                 ;--
;-----------------------------------

;--------T1 定时器的中断入口---------
;进入中断，检测声音
;每 0.5ms 进入一次中断
        ORG     001BH                   ;--

        ljmp T1J                        ;--跳到定时中断 1 处理程序

;-----------------------------------


;--------T2 定时器的中断入口---------
```

```
;计时中断，根据不同的模式，提供不同的延时
            ORG     002BH               ;--
            CLR    TF2H                 ;清除中断标志位

            LJMP    T2INTU              ;--跳到定时中断 2 处理程序
;---------------------------------

;--------T3 定时器的中断入口---------
;每隔 0.25s 进入一次中断
            ORG     0073H               ;--
            anl    91H,#7FH             ;清除中断标志位
            MOV PCA0CPM0,21H            ;将 21H 的值赋给 PCAOCPM0
            LJMP    T3CHULI             ;--跳到定时中断 3 处理程序
;---------------------------------


;----------------T3 定时器中断程序-----
T3CHULI: JB P2.2,light                  ;判断是否有光
            setb 01H                    ;光标志位置 1

            mov 44H,#30                 ;延时计数器置 30
            CLR TR2                     ;关闭计数器 2
            RETI
LIGHT:      CLR 01H                     ;清零光标志位
            SETB TR2                    ;开启计数器 2
            RETI
;-------------------------------------
```

```
;----------------T2 定时器中断程序(100ms)-----
T2INTU:
        JB 10H,T2L0                ;模式 0 和 1 对应的计数延时程序
        JB 11H,T2L0
        JB 12H,T2L2                ;模式 2 对应的计数延时程序
T2L0:   ljmp t2l0p
T2L2:   ljmp t2l2p

;--------;模式 0 和 1 对应的计数延时程序-----------
;44H30 次进入 T2，即 3 秒内都有光，则关闭灯光------
t2l0p:  DJNZ 44h,rt2l0             ;======
        clr 0EH                    ;======
        mov 44H,#30                ;======
        reti                       ;======
rt2l0:                             ;======
        RETI                       ;======
;----------------------------------------======

;--------;模式 2 对应的计数延时程序-----------
;45H100 次进入 T2，即延时 10s 后关闭灯光------
t2l2p:                             ;======
        DJNZ 45h,rt2l2             ;======
        clr 0EH                    ;======
        MOV PCA0CPM0,21H           ;======
        mov 44H,#30                ;======
        reti                       ;======
rt2l2:                             ;======
```

```
        RETI                                    ;======
;---------------------------------------=======
;---------------------------------------------- ----------
NOJIANCE:setb psw.3             ;不检测声音程序,1s 后去除这个状态
        setb psw.4
        INC R3
        CJNE R3,#0,T11
        INC R2
T11:  cjne r2,#007H,T1R
        cjne r3,#0D0H,T1R
        MOV R3,#00H
        MOV R2,#0
        SETB 03H
T1R:    CLR PSW.3
        clr psw.4
        CLR 04H
        CLR 05H
        RETI
;-------------------------------------------------
```

```
;------消除说话声程序----------=
;-----两秒内均没有声音，才消除有噪声状态-----
XIAOCHUNV:setb psw.3
          setb psw.4
          INC R6
          CJNE R6,#00,TR7
          INC R7
TR7:      CJNE R7,#00FH,T1R
          CJNE R6,#0A0H,T1R
          MOV R7,#0
          CLR 06H
          MOV R0,#0
          MOV R4,#0
          CLR 05H
          sjmp t1r
;----------------------------------------==


;-----------------------------------------------------
;--------------T1 定时器中断程序-------------------------
;---------------声音检测程序----------------------------
;-----------------------------------------------------

t1j:      JNB 03H,NOJIANCE      ;若有按键，1 秒内不检测声音
          SETB PSW.3            ;否则开始检测，变换工作寄存器组
          CLR PSW.4
          INC R4                ;R4+1
JIANCE:   JB P2.1,havevoice     ;检测是否有声音
          JB P2.1,havevoice
          JB P2.1,havevoice
```

```
        JB P2.1,havevoice
        JB P2.1,havevoice
        JB P2.1,havevoice
        JB P2.1,havevoice
        JB P2.1,havevoice
        JB P2.1,havevoice

NOVOICE: JB 06H,XIAOCHUNV        ;若 06H 为 1，则认为噪声标志位未消除，进入噪声程序
        JnB 05h,nov3             ;若一直没声音，即声音位 05H 为 0，则直接跳出
        CLR PSW.4                ;选定工作寄存器 1
        SETB PSW.3
        JNB 04H,T3R2             ;若声音有效，R2，R3 则计数，当 1.2 秒内都没有声音则声音有效输出
        INC R2
        CJNE R2,#0,T2R
        INC R3
T2R:    CJNE R3,#09H,NOV1
        cjne R2,#060H,NOV1
        SJMP OUTEN
T3R2:   CJNE R4,#200,nov2
        JC NOV1
nov2:   mov r0,#0
        mov r4,#0
        CLR 04H
        CLR 05H
        mov R2,#0
        mov r3,#0
NOV1:   CLR PSW.3                ;还原工作寄存器 0
```

```
        CLR PSW.4
        RETI
NOV3:   mov r4,#0
        CLR PSW.3                ;还原工作寄存器 0
        CLR PSW.4
        RETI

;-----------声控有效，对 LED 进行控制--------
OUTEN:    MOV R3,#0
        MOV R2,#0
        mov R4,#0
        MOV R0,#0
        CLR 05H
        CLR 04H
        JB 0EH,off
        JNB 01H,T3R0
        SETB 0EH
        SETB 00H
T3R0:   CLR PSW.3
        CLR PSW.4
        RETI
off:    MOV PCA0CPM0,#02H        ;关闭
        CLR 0EH
        CLR 00H
        CLR PSW.3
        RETI
;----------------------------------------

havevoice:JB 06H,XIAOCHUV
```

```
        setb 05H
        SETB 04H
        CLR PSW.4                ;选定工作寄存器 1
        SETB PSW.3
        CJNE R4,#200,T3C1   ; 若 R4 大于 200 时还有声音，则认为声音无效
T3C1:   JNC NOV                  ;R4 大于 200 时，跳到 NOV
        CLR PSW.3                ;返回中段
        CLR PSW.4
        RETI
NOV:    SETB 06H          ;有声音大于 0.1s 又有声音，声音无效 ,需要两秒内无声音方可消除 06H
        CLR 04H           ;有声音的 0.1s 内无声音，声音无效,清零声音有效位
        MOV R0,#0
        MOV R1,#0
        MOV R4,#0
        CLR PSW.3
        RETI

;------------有说话声音后两秒内若有声音，则认为说话仍然进行---------
XIAOCHUV: SETB PSW.3
        SETB PSW.4
        MOV R7,#00H       ;消除声音
        mov r6,#0
        CLR PSW.3
        CLR PSW.4
        RETI
;--------------------------------------------------------------------------
```

```
;-----------------------------------
        ORG 0500H
;---------------模式 0 键盘长跳转中转地址--------------
KEY01:LJMP KEY1
KEY02:LJMP KEY2
KEY03:LJMP KEY3
KEY04:LJMP KEY4
;----------模式 0 键盘长跳转中转地址---------
JUDGE:    LCALL DL1                ;延时去抖
        JNB P2.4,KEY04          ;若 P2.4 为 0，则说明 KI0 列有键按入，跳至列 1 判断
的中转程序
        JNB P2.5,KEY03          ;若 P2.5 为 0，则说明 KI1 列有键按入，跳至列 2 判断
的中转程序
        JNB P2.6,KEY02          ;若 P2.6 为 0，则说明 KI2 列有键按入，跳至列 3 判断
的中转程序
        JNB P2.7,KEY01          ;若 P2.6 为 0，则说明 KI2 列有键按入，跳至列 3 判断
的中转程序
        JnB 10h,jnol1
        setb tr1
jnol1:    LJMP LOOP                ;跳回主程序
;-----------------------------------

;----------------------主程序-------------------------
MAIN:     MOV SP,#60H              ;堆栈指针赋值

        LCALL   Init_Device     ;长调用配置文件

    ;--------------赋初值--------------------
        MOV R2,#24               ;R2 置数
```

```
        MOV R1,#08H             ;存储空间首地址 24H
CLEAR:  MOV @R1,#00H                ;清零
        INC R1                  ;R1=R1+1
        DJNZ R2,CLEAR               ;循环清零工作寄存器
        MOV R0,#00H
        MOV R3,#00H
        MOV 40H,#00H
        MOV 41H,#00H
        MOV 42H,#00H
        MOV 43H,#00H
        MOV R2,#00H

        MOV TH0,#00H
        MOV TH0,#00H
        MOV R5,#00H
        SETB TR0                ;开启定时器 0
        ORL  91H,#04H               ;开启定时器 3
        MOV 20H,#0FFH
        SETB 02H                ;显示器标志位置 1
        CLR 04H                 ;清零声音有效位
        clr 06H                 ;清零噪声标志位
        CLR 05H                 ;清零声音标志位
        MOV 21H,#42H            ;PCA0 的等效地址赋值
        MOV 22H,#00H
        MOV 23H,#00H
        MOV PCA0CPH0,#0FFH      ;默认占空比最小

;------模式选择程序
LOOP:   ;-----模式 0----
```

```
        mov R7,43h
        CJNE R7,#00H,L1
        MOV 22H,#01H        ;模式 0 标志位置 1
        MOV 23H,#00H
        ORL   91H,#04H          ;开启光控声控
        SETB TR1
        LJMP LOOP0

L1:     CJNE R7,#01H,L2
        MOV 22H,#02H        ;模式 1 标志位置 1
        MOV 23H,#00H
        CLR TR1
        ORL   91H,#04H          ;开启光控关闭声控
        SETB 0EH
        LJMP LOOP0

L2:     CJNE R7,#02H,L3
        MOV 22H,#04H          ;模式 2 标志位置 1
        MOV 23H,#00H
        clr tr1               ;关闭光控声控
        MOV PCA0CPM0,#02H
        clr tr2
        ANL   91H,#0FBH
        LJMP LOOP2
 ;-------模式扩充------
L3:     CJNE R7,#03H,L4
        MOV 22H,#00H
        MOV 23H,#00H
        clr TR1
        LJMP LOOP2
```

```
L4:     CJNE R7,#04H,L5
        MOV 22H,#00H
        MOV 23H,#00H
        LJMP LOOP1
L5:     CJNE R7,#05H,L6
        MOV 22H,#00H
        MOV 23H,#00H
        LJMP LOOP1
L6:     CJNE R7,#06H,L7
        MOV 22H,#00H
        MOV 23H,#00H
        LJMP LOOP1
L7:     CJNE R7,#07H,LOOP
        MOV 22H,#00H
        MOV 23H,#00H
        LJMP LOOP1

;----------------模式 0--------
LOOP0:
        JNB P2.7,JUDGEL0    ;若 P2.7 为 0，则说明 K4 键按入，跳至 4 判断的中转程序
        JNB P2.4,JUDGEL0    ;若 P2.4 为 0，则说明 K1 键按入，跳至 1 判断的中转程序
        JNB P2.5,JUDGEL0    ;若 P2.5 为 0，则说明 K2 键按入，跳至 2 判断的中转程序
        JNB P2.6,JUDGEL0    ;若 P2.6 为 0，则说明 K3 键按入，跳至 3 判断的中转程序
        LJMP LOOP0
JUDGEL0:  clr tr1
        LJMP JUDGE
```

```
JUDGEL1:
        anl   91H,#7FH
        LJMP JUDGE
LOOP1:  JNB P2.7,JUDGEL1   ;若 P2.7 为 0，则说明 K4 键按入，跳至列 4 判断的中转程序
        JNB P2.4,JUDGEL1   ;若 P2.4 为 0，则说明 K1 键按入，跳至列 1 判断的中转程序
        JNB P2.5,JUDGEL1   ;若 P2.5 为 0，则说明 K2 键按入，跳至列 2 判断的中转程序
        JNB P2.6,JUDGEL1   ;若 P2.6 为 0，则说明 K3 键按入，跳至列 3 判断的中转程序
        LJMP LOOP

;----------------按键----------------

;-------------亮度增加键---------------------------
L1G1:    SETB 0EH
         SJMP KEY1L1
KEY1:    CLR 03H                ;关闭检测声音标志位
         JB 11H,L1G1            ;为了保持灯亮一致
KEY1L1:CJNE R3,#99H,k1
         SJMP W1
k1:      MOV A, R3
         NOP
         ADD A,#01H      ;A=A+1
         NOP
         DA A                   ;调整成 BCD 码
```

```
        NOP
        MOV R3,A            ;A 赋给 R3，用于显示亮度等级
        NOP
W1:     MOV A,R3
        NOP
        MOV DPTR,#TABLE     ;查表，改变亮度等级
        NOP
        MOVC A,@A+DPTR
        NOP
        MOV PCA0CPH0,A
        NOP
        MOV PCA0CPM0,21H
        CJNE R2,#5FH,RE1
W3:     JB P2.7,W2
        CJNE R2,#5FH,RE1    ;若长按，则再次进入按键
        MOV R4,#01H
        LCALL DL0
        SJMP KEY1

RE1:    LCALL DL1           ;延时去抖，检测按键是否松开
        INC R2
        SJMP W3             ;若 P2.7 扔为 0，说明按键未松开，重新检测

W2:     MOV R2,#00H         ;若非模式 1，则不对 T1 定时器进行处理
        CLR 03H
        jB 12H,K1L2
       JnB 10h,K1nol1
       setb tr1
K1nol1: LJMP LOOP0          ;返回主程序
```

```
K1L2:    LJMP LOOP2

 ;-------亮度减少键-------
L1G:     CLR 0EH
         SJMP w12
KEY2:    CLR 03H              ;关闭检测声音标志位
          CJNE R3,#00H,k2     ;为了保持灯亮一致
          JB 11H,L1G
         SJMP w12
k2:      MOV A, R3
         NOP
         ADD A,#99H          ;A=A+1
         NOP
         DA A                ;调整成 BCD 码
         NOP
         MOV R3,A            ;A 赋给 R3，用于显示亮度等级
         NOP
W12:     MOV A,R3
         NOP
         MOV DPTR,#TABLE     ;查表，改变亮度等级
         NOP
         MOVC A,@A+DPTR
         NOP
         MOV PCA0CPH0,A
         MOV PCA0CPM0,21H
         NOP
```

```
        CJNE R2,#5FH,RE12      ;若长按，则再次进入按键
W32:    JB P2.6,W22

        CJNE R2,#5FH,RE12      ;若长按，则再次进入按键
        MOV R4,#01H
        LCALL DL0
        SJMP KEY2

RE12:   LCALL DL1              ;延时去抖，检测按键是否松开
        INC R2
        SJMP W32               ;若 P2.7 扔为 0，说明按键未松开，重新检测
W22:    MOV R2,#00H
        LCALL DL1
      jB 12H,K2L2
       JnB 10h,K2nol1     ;若非模式 1，则不对 T1 定时器进行处理
       setb tr1
        CLR 03H
K2nol1: LJMP LOOP0             ;返回主程序
K2L2:   LJMP LOOP2

;-------------开启或关闭数码管键----------------------
kEY3:     CLR 03H
          JNB 02H,xianshi
        CLR TR0
        CLR 02H
        MOV P1,#0FH
        SJMP RE3
```

```
XIANSHI:    SETB TR0
            SETB 02H

RE3:        LCALL DL1           ;延时去抖，检测按键是否松开

            JNB P2.5,RE3        ;若 P2.5 扔为 0，说明按键未松开，重新检测
            JB 12H,K3L2
           JnB 10h,K3nol1       ;若非模式 1，则不对 T1 定时器进行处理
           setb tr1
            CLR 03H
K3nol1:    LJMP LOOP0           ;返回主程序
 K3L2:     LJMP LOOP2
 ;-------------模式切换  键-----

KEY4:       CLR 03H
            INC 43H
            MOV R0,43h
            MOV R7,43H
            CJNE R7,#3,re4
            MOV 43h,#0
            MOV R0,43h
RE4:        LCALL DL1           ;延时去抖，检测按键是否松开
            JNB P2.4,RE4        ;若 P2.5 扔为 0，说明按键未松开，重新检测
           JnB 10h,K4nol1       ;若非模式 1，则不对 T1 定时器进行处理
           setb tr1
            CLR 03H
K4nol1:    LJMP LOOP            ;返回主程序
```

```
;---------------模式 2----------------------------
 ;跳至相应的按键程序
KEY01L2:LJMP KEY1
KEY02L2:LJMP KEY2
KEY03L2:LJMP KEY3
KEY04L2:LJMP KEY4
;----------模式 2 键盘长跳转中转地址---------
JUDGEl2:      LCALL DL1                ;延时去抖
              JNB P2.4,KEY04l2         ;若 P2.4 为 0，则说明 K1 键按入，跳至 1 判断的中转
程序
              JNB P2.5,KEY03L2         ;若 P2.5 为 0，则说明 K2 键按入，跳至 2 判断的中转
程序
              JNB P2.6,KEY02L2         ;若 P2.6 为 0，则说明 K3 键按入，跳至 3 判断的中转
程序
              JNB P2.7,KEY01L2         ;若 P2.6 为 0，则说明 K4 键按入，跳至 4 判断的中转
程序

              LJMP LOOP2               ;跳回主程序


LOOP2:        JB P2.1,KONGZHI
              JNB P2.7,JUDGEL2    ;若 P2.7 为 0，则说明 K4 键按入，跳至 4 判断的中转程序
              JNB P2.4,JUDGEL2    ;若 P2.4 为 0，则说明 K1 键按入，跳至 1 判断的中转程序
              JNB P2.5,JUDGEL2    ;若 P2.5 为 0，则说明 K2 键按入，跳至 2 判断的中转程序
              JNB P2.6,JUDGEL2    ;若 P2.6 为 0，则说明 K3 键按入，跳至 3 判断的中转程序
              sjmp loop2
KONGZHI:   JB    P2.2,LOOP2
              MOV PCA0CPM0,#42H
              NOP
              SETB TR2
```

```
            LCALL DL1
            mov   45h,#100
            sjmp loop2



;---------------延时程序----------------
            ORG     2550H
DL1:        MOV     R7,   #10H
DLA:        MOV     R6,   #0FFH
DLB:        DJNZ    R6,   DLB
            DJNZ    R7,   DLA
            RET

DL4:        MOV     R4,#05H
DL0:        MOV     R7,   #0FFH
DL:         MOV     R6,   #0FFH
DL6:        DJNZ    R6,   DL6
            DJNZ    R7,   DL
            DJNZ    R4,DL0

            RET

;-------------------------------------------------
;-------------显示屏动态显示程序-------------------
;-------------------------------------------------
display:    CLR     C           ;清零 C 位
            CJNE R5,#02H,X1   ;判断 R5 与 02 的大小
            SJMP display2  ;R5 等于 2，选通数码管 2
X1:         JNC   display3      ;cy=0，表明 R5=3，选通数码管 3
```

```
        CJNE R5,#01H,X0   ;R5 小于 2，再与 1 进行比较
        SJMP display1  ;R5 等于 1，选通数码管 1
X0:     SJMP display0  ;R5 小于 1，选通数码管 0

    ;----------数码管 0 显示模式低位-----
display0: MOV P1,#0FFH      ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH        ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH        ;清零数码管,消除换数码管带来的毛刺
        NOP
        NOP
        NOP
        MOV A,#0FH          ;屏蔽 R0 高四位
        ANL A,R0            ;将 R0 与 A 进行与操作，将 R0 低四位存至 A 中
        CLR ACC.4           ;P0.7,P0.6 置零，选通数码管 0
        CLR ACC.5
        MOV P1,A            ;将查得的值赋给 P1，数码管 0 得到显示
        MOV R5,#1           ;R5 赋 1，下一次中断到来，选通数码管 1
        RETI                ;定时器 1 中断返回
    ;-------------------------------------

    ;---------数码管 1 显示模式高位------
display1: MOV P1,#0FFH      ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH        ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH        ;清零数码管,消除换数码管带来的毛刺
        NOP
        NOP
        MOV A,#0F0H         ;屏蔽 R0 低四位
        ANL A,R0            ;将 R0 与 A 进行与操作，将 R0 高四位存至 A 中
        SWAP A              ;交换 A 中的高四位和低四位
```

```
        CJNE A,#00H,light_l
        MOV P1,#0fh
        MOV R5,#2          ;R5 赋 0，下一次中断到来，选通数码管 0
        RETI               ;定时器 1 中断返回
light_l:  CLR ACC.5        ;P0.7 置 0,P0.6 置 1，选通定数码管 1
        SETB ACC.4
        MOV P1,A
        MOV R5,#2          ;R5 赋 2，下一次中断到来，选通数码管 2
        RETI               ;定时器 1 中断返回
    ;-------------------------------------

    ;-----------数码管 3 显示 R3 低四位----------
display2:  MOV P1,#0FFH    ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH       ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH       ;清零数码管,消除换数码管带来的毛刺
        nop
        NOP
        NOP
        MOV A,#0FH         ;屏蔽 R3 高四位
        ANL A,R3           ;将 R3 与 A 进行与操作，将 R3 低四位存至 A 中

        SETB ACC.5             ;P0.7 置 1,P0.6 置 0，选通定数码管 2
        SETB ACC.4
        MOV P1,A           ;将查得的值赋给 P1，数码管 2 得到显示
        MOV R5,#3          ;R5 赋 3，下一次中断到来，选通数码管 3
        RETI               ;定时器 1 中断返回
    ;-------------------------------------

    ;-----------数码管 2 显示亮度高四位--------
display3: MOV P1,#0FFH     ;清零数码管,消除换数码管带来的毛刺
```

```
        MOV P1,#0FFH        ;清零数码管,消除换数码管带来的毛刺
        MOV P1,#0FFH        ;清零数码管,消除换数码管带来的毛刺
        NOP
        NOP
        NOP
        MOV A,#0F0H         ;屏蔽 R3 低四位
        ANL A,R3            ;将 R3 与 A 进行与操作，将 R3 高四位存至 A 中
        SWAP A              ;交换 A 中的高四位和低四位
        CJNE A,#00H,light_h
        MOV P1,#3fh
        MOV R5,#0           ;R5 赋 0，下一次中断到来，选通数码管 0
        RETI                ;定时器 1 中断返回
light_h:  CLR ACC.4         ;P0.7 置 1,P0.6 置 1，选通定数码管 3
        SETB ACC.5
        MOV P1,A            ;将查得的值赋给 P1，数码管 3 得到显示
        MOV R5,#0           ;R5 赋 0，下一次中断到来，选通数码管 0
        RETI                ;定时器 1 中断返回
    ;-----------------------------------

;-----------------------------------------------
;----------亮度查找表-------------------------------
      ORG   2800H

TABLE: DB   0FFH,0FEH,0FDH,0FCH,0FBH,0FAH,0F9H,0F8H,0F7H,0F6H
       ORG 2810H
       DB   0F5H,0F4H,0F3H,0F2H,0F1H,0F0H,0EFH,0EEH,0EDH,0ECH
       ORG 2820H
       DB   0EBH,0EAH,0E9H,0E8H,0E7H,0E6H,0E5H,0E4H,0E3H,0E2H
       ORG 2830H
```

```
        DB   0E0H,0DEH,0DCH,0DAH,0D8H,0D6H,0D4H,0D2H,0D0H,0CEH
        ORG 2840H
        DB   0CCH,0CAH,0C8H,0C6H,0C4H,0C2H,0C0H,0BEH,0BCH,0BAH
        ORG 2850H
        DB   0B8H,0B6H,0B4H,0B2H,0B0H,0AEH,0ACH,0AAH,0A8H,0A6H
        ORG 2860H
        DB   0A3H,0A0H,9DH,9AH,97H,94H,91H,8EH,8BH,88H
        ORG 2870H
        DB   85H,82H,7FH,7CH,79H,76H,73H,70H,6DH,6AH
        ORG 2880H
        DB   67H,64H,61H,5CH,5AH,57H,54H,51H,4DH,40H
        ORG 2890H
        DB   3aH,34H,2fH,28H,23H,1ch,17h,10h,08h,00H
public  Init_Device

INIT SEGMENT CODE
    rseg INIT

; Peripheral specific initialization functions,
; Called from the Init_Device label
PCA_Init:
    mov   PCA0CN,     #040h
    anl   PCA0MD,     #0BFh
    mov   PCA0MD,     #000h
    mov   PCA0CPM0,   #042h
    ret

Timer_Init:
    mov   TMOD,       #022h
    mov   TL1,        #080h
```

```
MOV TH1,#80h
mov    TMR2RLL,     #04Fh
mov    TMR2RLH,     #09Ch
mov    TMR2L,       #04Fh
mov    TMR2H,       #09Ch
mov    TMR3RLL,     #008h
mov    TMR3RLH,     #0f6h
mov    TMR3L,       #008h
mov    TMR3H,       #0f6h
ret

Port_IO_Init:
; P0.0  -  Skipped,        Open-Drain, Digital
; P0.1  -  Skipped,        Open-Drain, Digital
; P0.2  -  Skipped,        Open-Drain, Digital
; P0.3  -  Skipped,        Open-Drain, Digital
; P0.4  -  Skipped,        Open-Drain, Digital
; P0.5  -  Skipped,        Open-Drain, Digital
; P0.6  -  Skipped,        Open-Drain, Digital
; P0.7  -  Skipped,        Open-Drain, Digital

; P1.0  -  Skipped,        Open-Drain, Digital
; P1.1  -  Skipped,        Open-Drain, Digital
; P1.2  -  Skipped,        Open-Drain, Digital
; P1.3  -  Skipped,        Open-Drain, Digital
; P1.4  -  Skipped,        Open-Drain, Digital
; P1.5  -  Skipped,        Open-Drain, Digital
; P1.6  -  Skipped,        Open-Drain, Digital
; P1.7  -  Skipped,        Open-Drain, Digital
; P2.0  -  CEX0 (PCA),    Open-Drain, Digital
```

```
    ; P2.1   -   Unassigned,   Open-Drain, Digital
    ; P2.2   -   Unassigned,   Open-Drain, Digital
    ; P2.3   -   Unassigned,   Open-Drain, Digital

    mov   P0SKIP,     #0FFh
    mov   P1SKIP,     #0FFh
    mov   XBR1,       #041h
    ret

Interrupts_Init:
    mov   IT01CF,     #020h
    mov   EIE1,       #080h
    mov   IE,         #0AAh
    ret

; Initialization function for device,
; Call Init_Device from your main program
Init_Device:
    lcall PCA_Init
    lcall Timer_Init
    lcall Port_IO_Init
    lcall Interrupts_Init
    ret

end
```