哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY,SHENZHEN

实验与创新实践教育中心

实验报告

课程名称：电子工艺实习-嵌入式系统软件设计

实验名称：基于单片机 MSP430F5529 的展馆灯光控制

姓　　名：宋子洋

学　　号：190210305

专业-班级：19级电信3班

实验日期：2021 年 7 月 22 日

教师评语：

评分：

教师签字：

日　　期：

源代码要求：关键代码要求注释
电子版文件名格式：专业--姓名--学号—实验 X
实验报告提交：纸质版实验报告由班长统一收取；电子版报告按班级打包，发到主讲老师 QQ 邮箱。

1. 实验内容

任务：该系统主要用于展览馆等需要解说员解说，且需要调节光线以达到最佳演示效果的场合。系统检测到外界声音后打开灯光，系统根据周边环境光的情况自动调整 LED 灯的亮度，以达到最佳展示效果。如果一定时间内都没有检测到声音信号，则自动关闭 LED 灯，以达到节能目的。

从中提取关键信息：

1.能感知声音信号

2.能感知光强信号

3.能调节 LED 灯亮度

4.能自动关闭 LED 灯

演示效果： 以小功率 LED 灯 L1~L6 亮灯的数量指示环境光强值。有声音时，打开大功率 LED 灯并进行光线调节，使小功率 LED 灯在各种光线条件下维持在恒定的亮灯数量。一定时间没检测到声音，则关闭大功率 LED 灯。要求大功率 LED 光强平滑改变，不可突变，开环或闭环控制均可。

2. 实验原理（用文字或图表阐述要实现的实验现象、相关单片机模块的理论知识、寄存器使用方法等）

单片机理论知识：

1. H 桥驱动电路驱动大功率 LED 灯的工作：

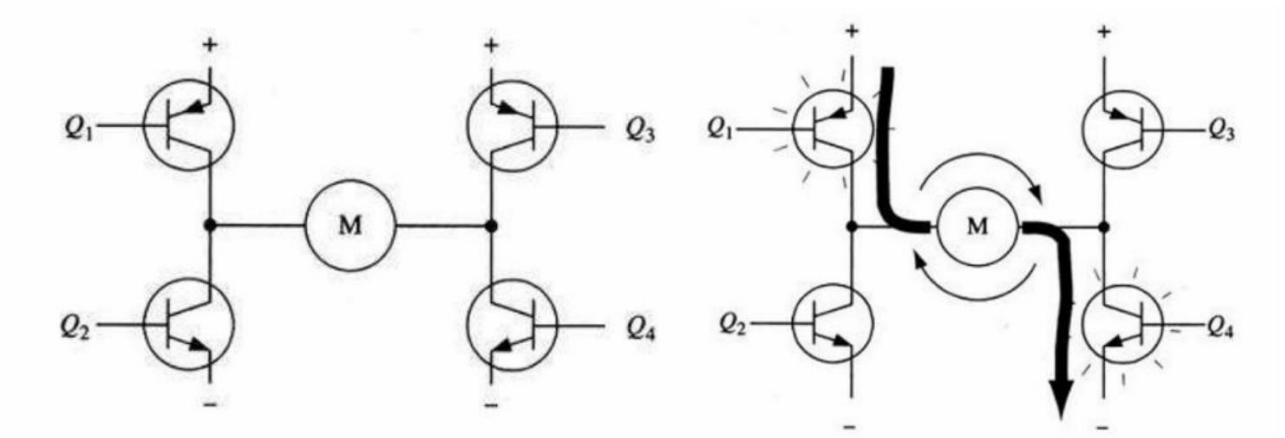


Table 1. DRV8837 Device Logic

nSLEEP	IN1	IN2	OUT1	OUT2	FUNCTION (DC MOTOR)
0	X	X	Z	Z	Coast
1	0	0	Z	Z	Coast
1	0	1	L	H	Reverse
1	1	0	H	L	Forward
1	1	1	L	L	Brake

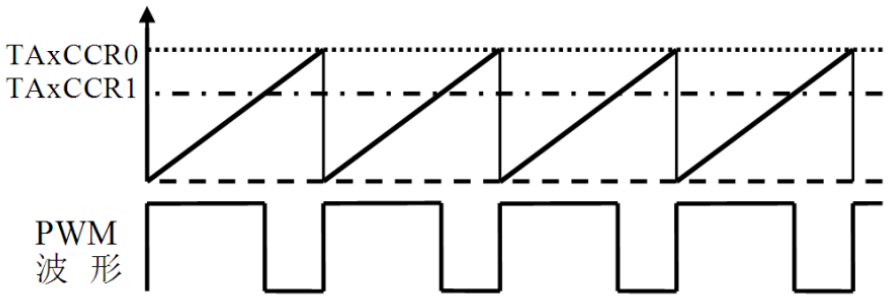
三个引脚分别为 nSLEEP、IN1 和 IN2，对应单片机的引脚分别是 P1.5、P2.4 和 P2.5，用三个引脚实现控制通断的方法如上图所示。

2. LED 灯亮度调节与控制的方法：

使用外设功能，用 PWM 波形的占空比控制 LED 灯的亮度，占空比越大，LED 灯越亮；反之越暗。

Timer_A 定时器的计数器工作在增计数方式或者增减计数方式，输出采用输出模式 7（复位/置位模式）或者输出模式 6（翻转/置位模式），则可利用寄存器 TAxCCR0 控制 PWM 波形的周期，用某个寄存器 TAxCCRx 控制占空比。这样 Timer_A 就可以产生出任意占空比的 PWM 波形。

通过逐渐缓慢调整占空比，可以使得 LED 灯的亮度实现平滑变化，而不产生突变。当亮度足够低的时候，再用引脚控制电机休眠即可使其完全关闭。



3. ADC 模数转换器使用方法：

包括的模块有：

转换控制寄存器 0（ADC12CTL0 Register）
转换控制寄存器 1（ADC12CTL1 Register）
转换控制寄存器 2（ADC12CTL2 Register）
存储寄存器（ADC12MEMx Register）
存储控制寄存器（ADC12MCTLx Register）
中断使能寄存器（ADC12IE Register）
中断标志寄存器（ADC12IFG Register）

中断向量寄存器（ADC12IV Register）

初始化时，重点处理以下功能的通断：

（1）在 ADC12ENC==0 时（默认）,初始化各寄存器，后打开 ADC 转换使能（ADC12ENC==1）。

（2） 选择多路采样转换（首次需要 SHI 信号上升沿触发采样定时器）自动循环采样转换。

（3） 启动（打开）ADC12 模块。

（4） 选择作为起始的寄存器地址。

（5） 选择采样转换模式。

（6） 选择采样保持模式，代码中选择采样信号（SAMPCON）的来源是采样定时器。

（7） 配置存储寄存器的选择通道，代码中寄存器 0 选择通道 P6.2 麦克风，寄存器 1 选择通道 P6.0 连接光敏电阻（GPIO 默认方向是输入方向）。

（8） ADC 转换使能。

（9） 采样开始前，要设置开始采样转换。

4. 根据光强值调节 LED 亮度的方法：

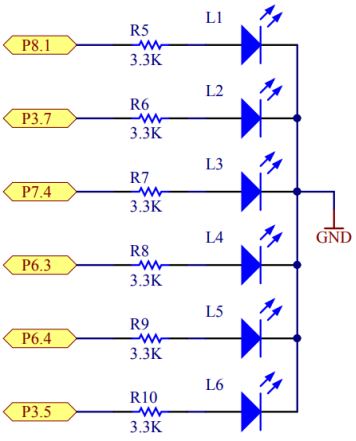
$$\frac{采集到的光强电压值}{3300} = \frac{一个周期点亮LED的时间}{LED点亮周期}$$

在实际实验过程中，由于光强电压总是处于一定的范围，因此实际采用的是分段函数特殊判断法（见“**实验中遇到过哪些问题？**”部分）

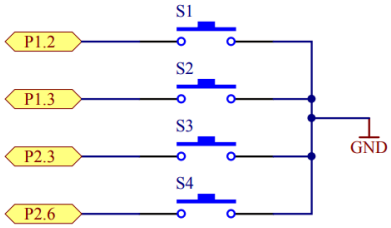
5. 定时关闭的方法：

采用 Timer_A 定时进入中断的特性，设置合适的计数周期，达到计数周期的限定值即可关断，同时将计数器清零。

3. 硬件原理图（原理图、接线图等，要阐明具体引脚并与程序对应）



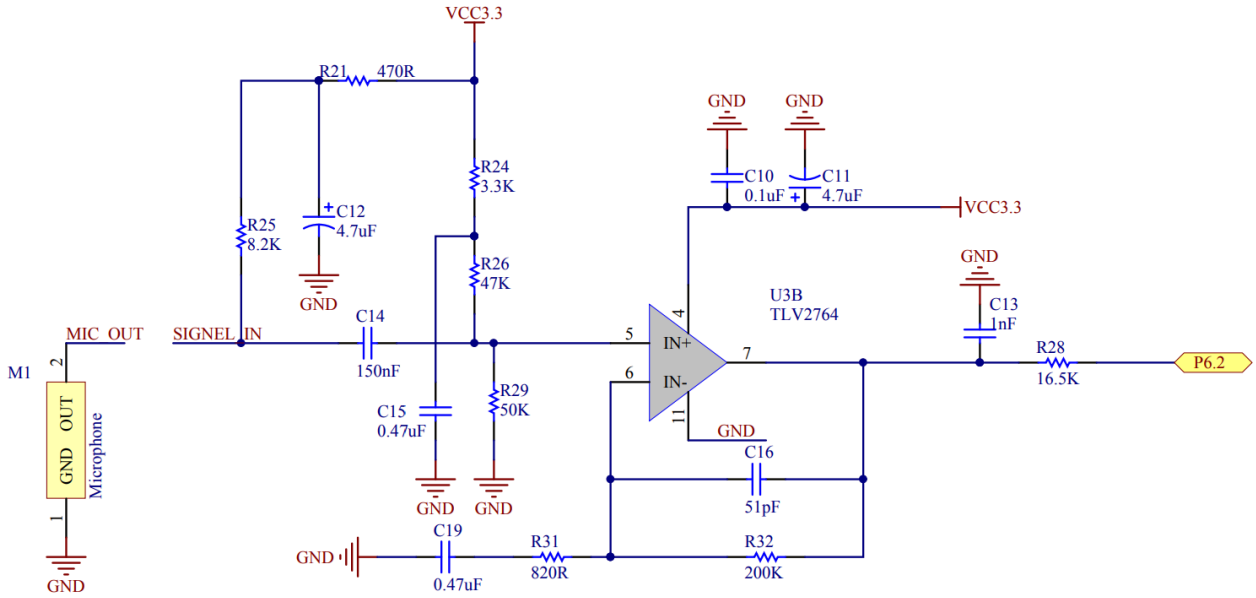
LED



USER Key

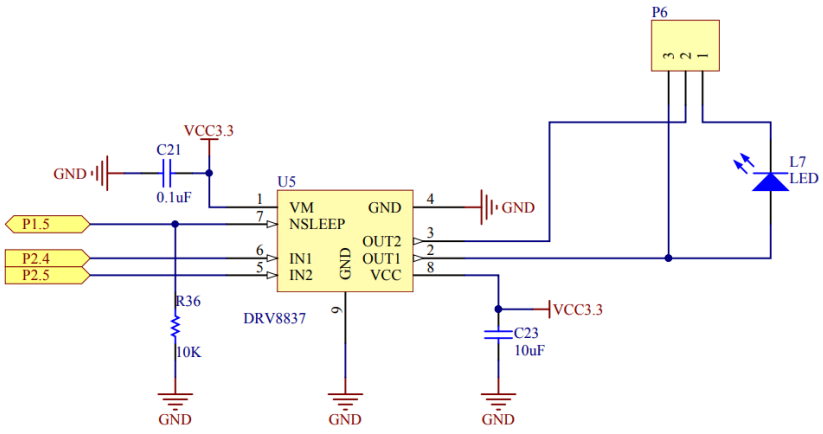
LED 灯端口输出，依次是 L1 对应 P8.1；L2 对应 P3.7；L3 对应 P7.4；L4 对应 P6.3；L5 对应 P6.4；L6 对应 P3.5，功能设为 GPIO。

用户按键 S1 对应 P1.2 端口，功能设为 GPIO，使能上下拉电阻功能且置为上拉电阻方式，使能中断。



音频输入调理

P6.2 端口对应麦克风输入，功能设为外设。



直流电机模块

P1.5、P2.4、P2.5 端口对应大功率 LED 灯的直流电机，P1.5、P2.4 功能设为 GPIO，P2.5 功能设为外设，对应 Timer_A 的 TA2CCR2。

接线为口袋板 MIC 和 OPA_IN 端口用跳线帽连接，同时光敏电阻传感器模块用杜邦线连接到单片机开发板上，VCC 对应口袋版 3.3V，GND 对应口袋版 GND ，AO 对应开发板 P6.0。

4. 程序流程图（画出详细设计的程序流程图，具体到每个步骤）

流程图的绘制详见实验报告末尾附页。

5. 源代码

源代码（main.c）详见实验报告末尾附页。

6. 实验结果（文字阐述实验现象，实验波形可手画）

（1）开关 S1 按下，系统开始工作。开关 S1 再次按下，系统进入休眠。

（2）在大功率 LED 灯熄灭的状态下，以小功率 LED 灯 L1~L6 亮灯的数量指示环境光强值。

（3）有声音时，大功率 LED 灯打开并进行光线调节，小功率 LED 灯在各种光线条件下维持在恒定的亮灯数量。

（4）一定时间没检测到声音（本例中关闭时间约为 48 秒），则关闭大功率 LED 灯。大功率 LED 光强平滑改变，不突变。

（5）当光敏电阻由环境光调节时，大功率 LED 灯亮度稳定，环境光强时 LED 灯亮，环境光弱时 LED 灯暗，起到负反馈调节的作用。

（6）当光敏电阻由 LED 灯调节时，大功率 LED 灯亮度呈现亮-暗周期波动变化，体现了亮度调节的负反馈作用。

具体实验现象以及音量柱表现情况请参考附件中的测试视频。

7. 实验中遇到过哪些问题？如何解决的？

实验过程中，由于光敏电阻传感器的接收的环境光强值总是在一定范围内，因此为了使得 LED 亮度调节更加合理，将光强值与 LED 点亮周期的线性关系改为由反复实验得出的如下分段函数关系。(TA0CCR0=65535)

$$LEDPeriod = \begin{cases} 255 & lightValue \le 200 \\ (lightValue - 196) \times 80 & 200 < lightValue < 1000 \\ 65535 & lightValue \ge 1000 \end{cases}$$

同时由于 LED 灯对于采集光强的变化较为敏感，因此不仅在光强采集模块中增加滤波部

分，而且在 LED 亮度调整的部分采用较小变化不调整的特殊判断以减少 LED 亮度反复变化、不稳定的现象。而且关闭 LED 灯时，如果仅仅采用降低 PWM 的办法，LED 不能完全熄灭，因此在设置中还增加了使用端口 P1.5 控制 LED 灯工作状态的操作使其能完全被关闭。在实际测试的过程中发现 unsigned int 变量发生在 65536 时常常溢出使得工作效果不稳定，因此在程序中进行了一些修改调整，以避免出现这样的情况。

由于程序内部存在一定运行时间上的时延，因此定时中断关闭 LED 灯的周期数目与理论计算有一定偏差，通过大量反复实验总结，对于这个周期计数数目值修改了一个合理的大小，使得时延时间控制得更稳定。

同时，由于开关 S1 的中断亦有一定的敏感性，通过仿照老师课上所给的 Timer_A 示例代码中设置时延和二重判断的办法，使得开关 S1 更稳定地工作。

另外，由于两路采集同时进行，需要设置两个寄存器，通过查阅数据手册，再仿照之前的实验，通过对 ADC 的初始化解决了此问题。

8. 实验体会与建议

由于之前的课上实验中已经做好一部分的内容，接下来关键为整合知识，编写出一个功能完整的程序。本实验难度有所提升，而且模块功能较多，需要考虑功能之间、中断之间互相配合的问题，本次实验作为上手实践是一个很好的锻炼机会，为接下来的大作业做充足的准备。

附页：源代码/*main.c*/

```
#include <msp430.h>

static volatile unsigned int systemOn = 1,  //系统开关标记
                          LEDcycle = 0,  //计数周期变量
                          LEDon = 0,     //LED开关
                          LEDPeriod = 0, //LED点亮周期
                          volume = 0,    //声音采样值
                          lightValue = 4095,     //光强采样值
                          lightNum = 0,          //排灯点亮个数
                          lightCmp[7]={0,700,600,500,400,300,200},//光强比较值
                          target = 0xFFFE;       //目标周期变量

#define XT2_FREQ   4000000
#define MCLK_FREQ 16000000
#define SMCLK_FREQ 4000000
void ClockInit()
{
     UCSCTL6 &= ~XT1OFF;    //启动XT1
     P5SEL |= BIT2 + BIT3;  //XT2引脚功能选择
     UCSCTL6 &= ~XT2OFF;    //打开XT2
     __bis_SR_register(SCG0);
     UCSCTL0 = DCO0+DCO1+DCO2+DCO3+DCO4;
     UCSCTL1 = DCORSEL_4;   //DCO频率范围在28.2MHZ以下
     UCSCTL2 = FLLD_5 + 1;  //D=16，N=1
     UCSCTL3 = SELREF_5 + FLLREFDIV_3;      //n=8,FLLREFCLK时钟源为XT2CLK;
DCOCLK=D*(N+1)*(FLLREFCLK/n);DCOCLKDIV=(N+1)*(FLLREFCLK/n);
     UCSCTL4 = SELA_4 + SELS_3 +SELM_3;     //ACLK的时钟源为DCOCLKDIV,MCLK\SMCLK的时钟源为DCOCLK
     UCSCTL5 = DIVA_5 +DIVS_1;             //ACLK由DCOCLKDIV的32分频得到，SMCLK由DCOCLK的2分频得到
     //最终MCLK:16MHZ,SMCLK:8MHZ,ACLK:32KHZ
//   __bic_SR_register(SCG0);                 //Enable the FLL control loop
}
int Filter(int Reg)//中值滤波法
{
    int i, j, t;
    int value_buf[N];
    int sum = 0;
    if (Reg == 0) {
        for (i = 0; i < N; i++)//Reg=0，声音滤波
        {
            value_buf[i] = ADC12MEM0;
            _delay_cycles(1000);
        }
    }
```

```
    else {//Reg=1，光强滤波
        for (i = 0; i < N; i++)
        {
            value_buf[i] = ADC12MEM1;
            _delay_cycles(50);
        }
    }

    for (j=0; j<N-1; j++)
    {
        for (i=0; i<N-j-1; i++)
        {
            if (value_buf[i] > value_buf[i+1])
            {
                int temp = value_buf[i];
                value_buf[i] = value_buf[i+1];
                value_buf[i+1] = temp;
            }
        }
    }
    for(i=1; i<N-1; i++) sum += value_buf[i];
    t = sum/(N-2);
    return t;
}
void light_on()//排灯全亮
{
    P8OUT |= BIT1;
    P3OUT |= BIT7;
    P7OUT |= BIT4;
    P6OUT |= BIT3;
    P6OUT |= BIT4;
    P3OUT |= BIT5;
}
void light_off()//排灯全灭
{
    P8OUT &= ~BIT1;
    P3OUT &= ~BIT7;
    P7OUT &= ~BIT4;
    P6OUT &= ~BIT3;
    P6OUT &= ~BIT4;
    P3OUT &= ~BIT5;
}
void light_ignite()//排灯按照光强点亮
{
```

```
    //  LED1~LED6灯的亮灭判断
    if (lightValue <= lightCmp[1] && systemOn) {P8OUT |= BIT1; lightNum = 1;} else {P8OUT &=
~BIT1;}
    if (lightValue <= lightCmp[2] && systemOn) {P3OUT |= BIT7; lightNum = 2;} else {P3OUT &=
~BIT7;}
    if (lightValue <= lightCmp[3] && systemOn) {P7OUT |= BIT4; lightNum = 3;} else {P7OUT &=
~BIT4;}
    if (lightValue <= lightCmp[4] && systemOn) {P6OUT |= BIT3; lightNum = 4;} else {P6OUT &=
~BIT3;}
    if (lightValue <= lightCmp[5] && systemOn) {P6OUT |= BIT4; lightNum = 5;} else {P6OUT &=
~BIT4;}
    if (lightValue <= lightCmp[6] && systemOn) {P3OUT |= BIT5; lightNum = 6;} else {P3OUT &=
~BIT5;}
}
void LEDstart(void)//LED开灯，渐亮
{
    P1OUT |= BIT5; //打开P1.5
    for (; (LEDPeriod < 0xFF00) && systemOn; LEDPeriod += 0x00FF) {
        TA2CCR2 = LEDPeriod;
        __delay_cycles(MCLK_FREQ*0.005);
    }
}
void LEDend(void)//LED熄灯，渐暗
{
    for (; LEDPeriod >= 0x01FF; LEDPeriod -= 0x00FF) {
        TA2CCR2 = LEDPeriod;
        __delay_cycles(MCLK_FREQ*0.01);
    }
    P1OUT &= ~BIT5; //要把休眠开关P1.5关掉，否则LED仍有光
    LEDon = LEDcycle = 0;
}
void LEDadjust()//LED亮度调节
{
    if (lightValue <= 200) target = 0x00FF;
    else if (lightValue >= 1000) target = 0xFFFE;
    else target = (lightValue-196)*80;
    if (LEDPeriod < target) {
        if (target - LEDPeriod <= 0x1FFF) return;
        for (; (LEDPeriod <= target && LEDPeriod < 0xFF00) && systemOn; LEDPeriod += 0x00FF) {
                TA2CCR2 = LEDPeriod;
                __delay_cycles(MCLK_FREQ*0.005);
         }
    }
    else {
```

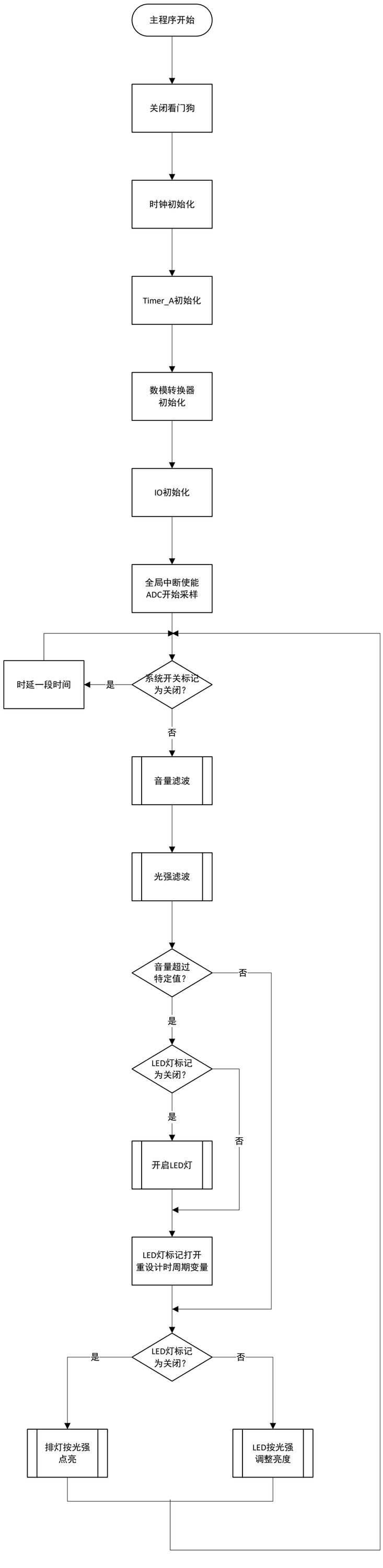
```
        if (LEDPeriod - target <= 0x1FFF) return;
        for (; (LEDPeriod >= target && LEDPeriod >= 0x00FF) && systemOn; LEDPeriod -= 0x007F) {
                TA2CCR2 = LEDPeriod;
                __delay_cycles(MCLK_FREQ*0.005);
         }
    }
}
/*******************************************************************************
   IO端口初始化
说明： 口袋板LED灯端口输出，依次是L1-->P8.1; L2-->P3.7; L3--> P7.4; L4-->P6.3; L5-->P6.4; L6-->P3.5
机械按钮输入，依次是S1-->P1.2; S2-->P1.3;
*******************************************************************************/
void InitIO()
{
                    //LED1~LED6的初始化
    P8DIR |= BIT1;  //p8.1 output
    P3DIR |= BIT7;  //p3.7 output
    P7DIR |= BIT4;  //p7.4 output
    P6DIR |= BIT3;  //p6.3 output
    P6DIR |= BIT4;  //p6.4 output
    P3DIR |= BIT5;  //p3.5 output

                    //麦克风对应IO初始化
    P6DIR |= BIT2;
    P6SEL |= BIT2;
                    //光敏电阻对应IO初始化
    P6DIR |= BIT1;
    P6SEL |= BIT1;

    P1DIR &= ~BIT2;
    P1REN |= BIT2;  // 使能P1.2上下拉电阻功能
    P1OUT |= BIT2;  // 置P1.2为上拉电阻方式
    P1IE  |= BIT2;  //enable P1.2 interrupt
    P1IES |= BIT2;  //high-low transition

                    //LED灯控制
     P1DIR |= BIT5;
     P1OUT |= BIT5;

     P2SEL |= BIT5; //设置P2.5由TA2.2控制
     P2DIR |= BIT5; //设置P1.5和P2.5为输出模式

                     //点亮熄灭测试LED
     systemOn = 1;
```

```
    light_on();
    LEDstart();
    light_off();
    LEDend();
    systemOn = 0;
}
void AD_Init()
{// 在ADC12ENC==0时（默认）,初始化各寄存器，后打开ADC转换使能（ADC12ENC==1）
    ADC12CTL0 |= ADC12MSC;//多路采样转换（首次需要SHI信号上升沿触发采样定时器）自动循环采样转换
    ADC12CTL0 |= ADC12ON; //启动（打开）ADC12模块

    ADC12CTL1 |= ADC12CONSEQ_3;//选择序列通道多次循环采样转换
    ADC12CTL1 |= ADC12CSTARTADD_0; //开始地址使用寄存器0
    ADC12CTL1 |= ADC12SHP;//采样保持模式选择，选择采样信号（SAMPCON）的来源是采样定时器

    //存储寄存器0选择通道P6.2麦克风
    ADC12MCTL0 |= ADC12INCH_2;
    //存储寄存器1选择通道P6.1光敏电阻
    ADC12MCTL1 |= ADC12INCH_0 + ADC12EOS;//End of sequence回到寄存器0

    //ADC转换使能
    ADC12CTL0 |= ADC12ENC;
}
void TimerInit(void)//TimerA初始化
{
    TA2CTL = TASSEL__SMCLK + TACLR + MC_1;
    TA2CCR0 = 0xFFFF;
    TA2CCTL2 = OUTMOD_7;                          //复位-翻转

    TA0CTL |= TASSEL__ACLK + TACLR + MC_1 + TAIE;//时钟为ACLK(32768Hz),增计数模式，开始时清零计数器
    TA0CCTL0 =  CCIE;                             //比较器中断使能
    TA0CCR0  = 32768;                             //周期1秒进一次中断

    __enable_interrupt();
}

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer
                          //初始化
    ClockInit();
    TimerInit();
    AD_Init();
    InitIO();
```

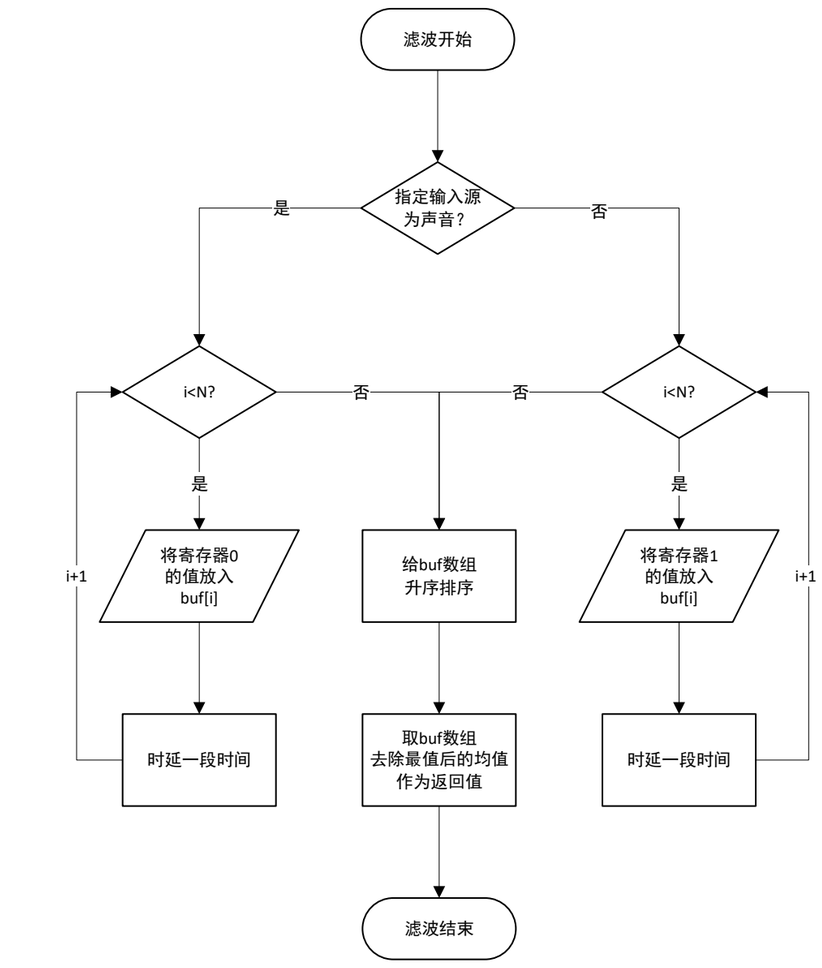
```
    __no_operation();       // For debugger
    __delay_cycles(MCLK_FREQ*2);

    ADC12CTL0 |= ADC12SC;   //开始采样

    for(;;)
    {
       if (systemOn == 0) {//系统休眠，不工作
           __delay_cycles(10000);
           continue;
       }

       volume = Filter(0);      //音量，经过滤波
       lightValue = Filter(1);  //光强，经过滤波
       if (volume >= 3000 && systemOn) {
           if (!LEDon) LEDstart();
           LEDon = 1;
           LEDcycle = 0;        //重新开始计时
       }
       if (!LEDon) light_ignite();  //LED未触发时排灯根据光强点亮
       else LEDadjust();            //根据光强调整LED亮度
    }
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    if (P1IFG & BIT2) {
        __delay_cycles(100); //时延防止敏感
        if (P1IFG & BIT2)
        {
            if (systemOn) {
                systemOn = 0;
                light_off();
                LEDend();
            }
            else {
                systemOn = 1;
                LEDon = LEDcycle = 0;
            }
            P1IFG &= ~BIT2; //清除中断标记
        }
    }
}
```

```
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
      if (LEDon != 0) {
          LEDcycle++;
          if(LEDcycle >= 30) LEDend();//调整关灯延时的计数值
      }
      TA0CTL &= ~TAIFG; //清除中断标记
      TA0CTL |= TACLR;  //清除计数器
}
```

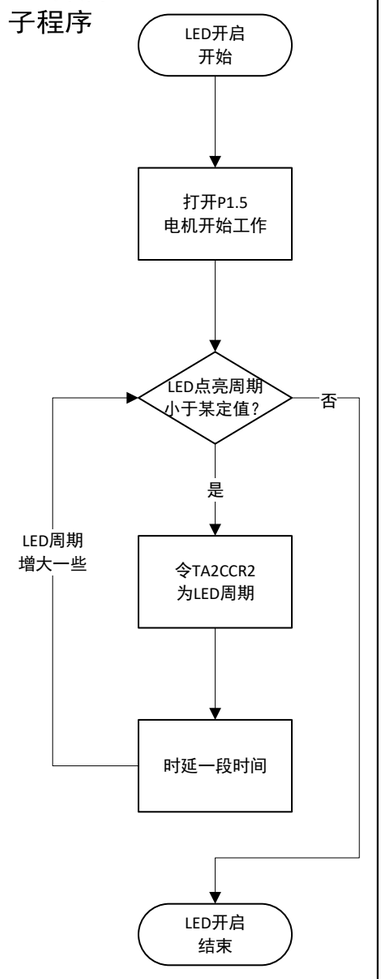
主程序



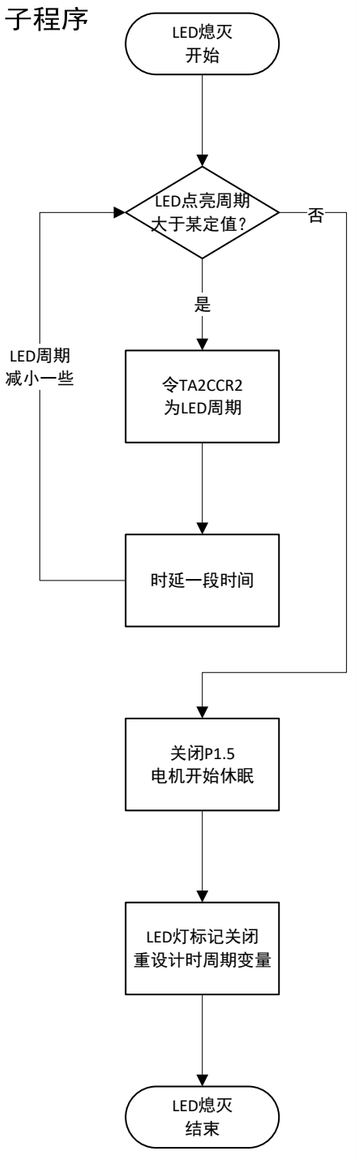
滤波子程序



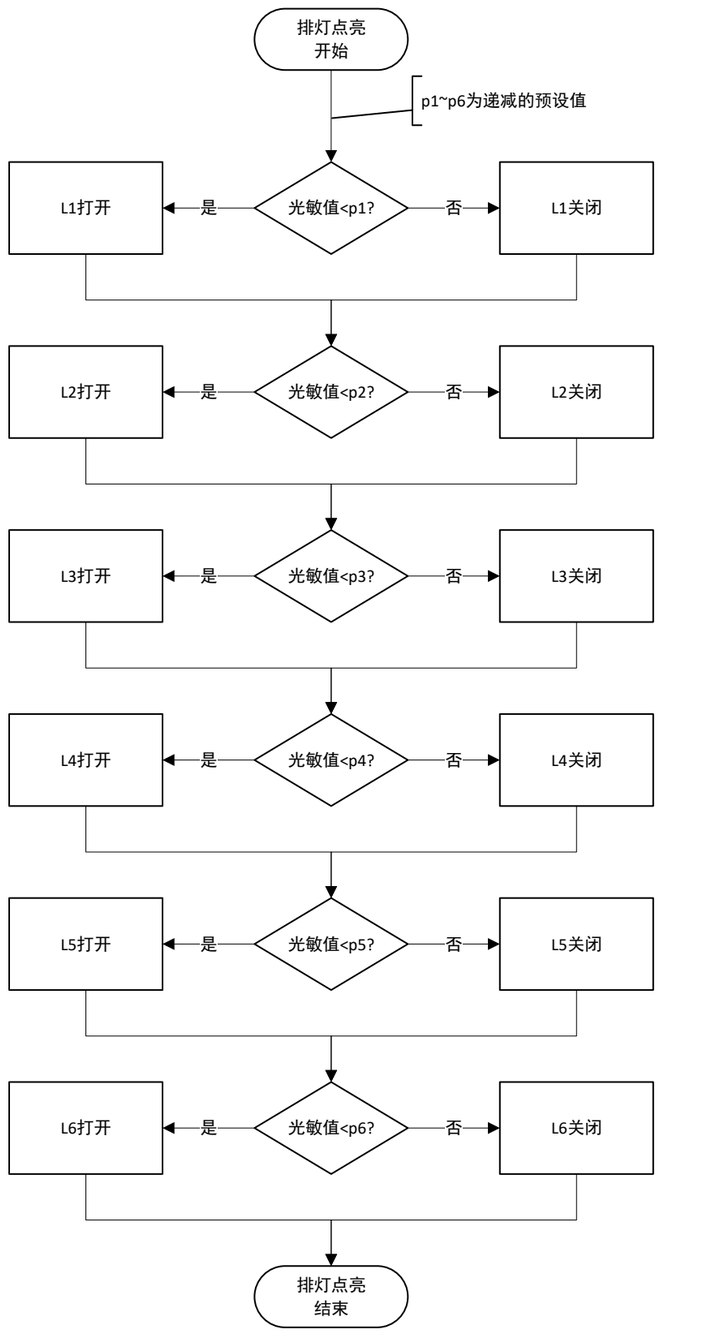
LED开启子程序



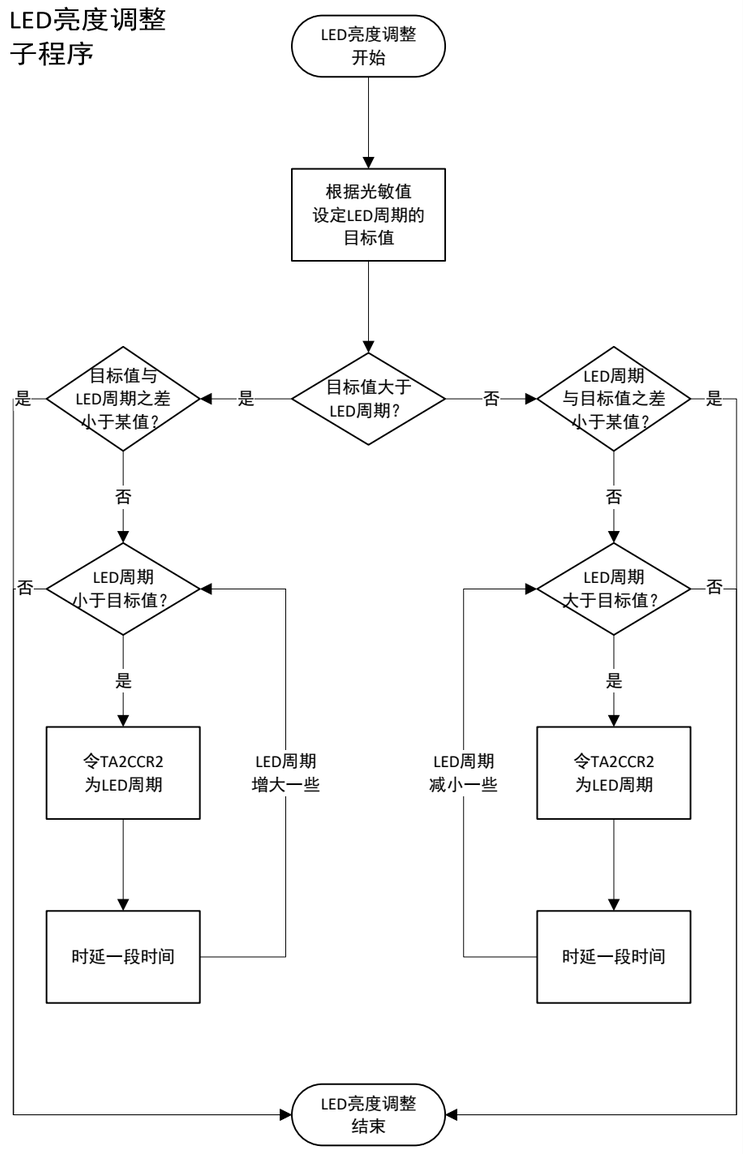
LED熄灭子程序



排灯点亮子程序

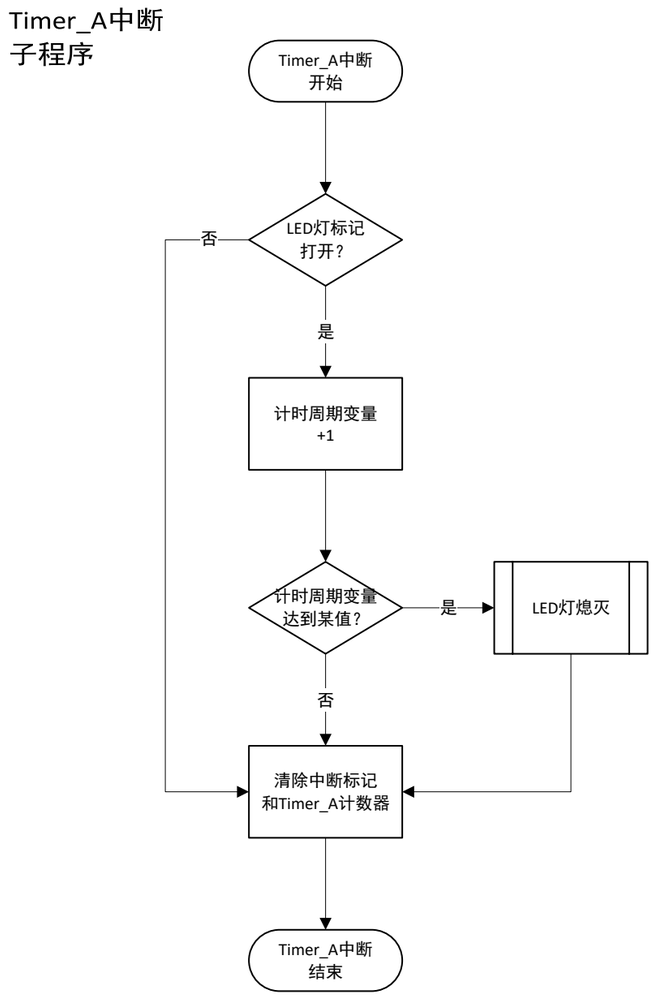


LED亮度调整子程序



P1中断子程序

Timer_A中断子程序



基于单片机MSP430F5529的展馆灯光控制流程图绘制

*采用Microsoft Visio 2021软件制图

学校： 哈尔滨工业大学（深圳）
班级： 19级电信3班
姓名： 宋子洋
学号： 190210305
日期： 2021/7/22