



成 绩	
评 定	
教 师	
签 名	

# 四川大学电气信息学院

## 计算机应用设计（单片机）

### 实验报告

实验名称：\_\_\_\_\_ 实验一 反应速度测试 \_\_\_\_\_

实验地点：\_\_\_\_\_ 高压实验楼 \_\_\_\_\_

年 级：\_\_\_\_\_ 2018 级 \_\_\_\_\_

姓 名：\_\_\_\_\_ 汪雨甜 \_\_\_\_\_

学 号：\_\_\_\_\_ 2018141411218 \_\_\_\_\_

实验时间： 2020 年 12 月 3 日

## 实验一 反应速度测试

### 一、实验内容

1. 数码管倒计时进入准备状态
2. 亮 LED 灯
3. 测试人员看见亮灯后按键
4. MCU 记录反应时间
5. 在数码管上显示反应时间
6. 按另一键重新开始
7. 如果出现抢跑的情况，给出提示

### 二、设计思路

#### 2.1 整体设计思路

首先使用 PIT 模块计时，并且使用时间处理函数，对时间进行我们常规化的处理，使它变成我们常用的分秒计时。同时，在倒计时的时间内定义一个函数检查是否抢跑，如果抢跑则给出文字和灯光提示，还需要一个可以将中断记下的时间显示在数码管上的函数。

#### 2.2 分布设计思路

①设置 SW1 为测试反应时间的按键，SW2 为倒计时开始（即复位）键，设置 PIT 的定时中断时间间隔为 10ms，每十秒后  $seconde+1$

②SW2 被按下后，count 清零，倒计时开始

③倒计时为 0 瞬间，四个 LED 同时开启，小灯颜色可以通过 SW5-SW8 进行更改。测试开始，seconde 开始计数。

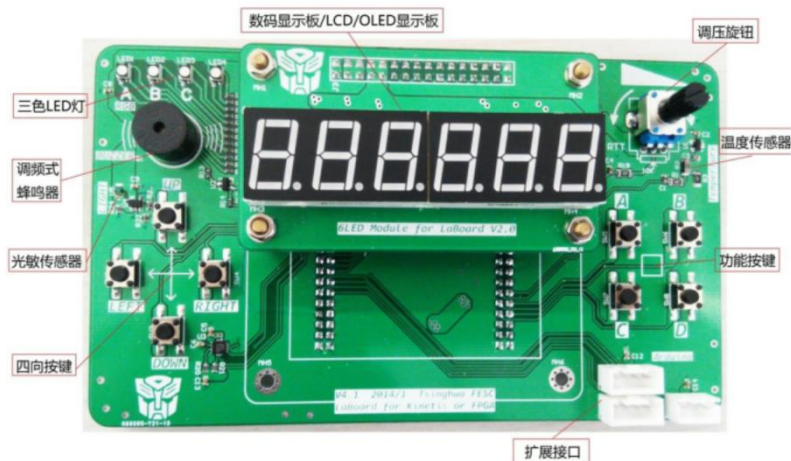
④如果在倒计时的时间内按下 SW1 即为抢跑，抢跑会有提示：LED 四个红色灯全部开启，数码管显示错误提示。

⑤在倒计时内没有抢跑，按下 SW1 键，中断程序启动，记录的时间显示在数码管上。

⑥任何时刻按下 SW2 键，可以重新启动倒计时，即重新开始测试。

### 三、硬件系统构成

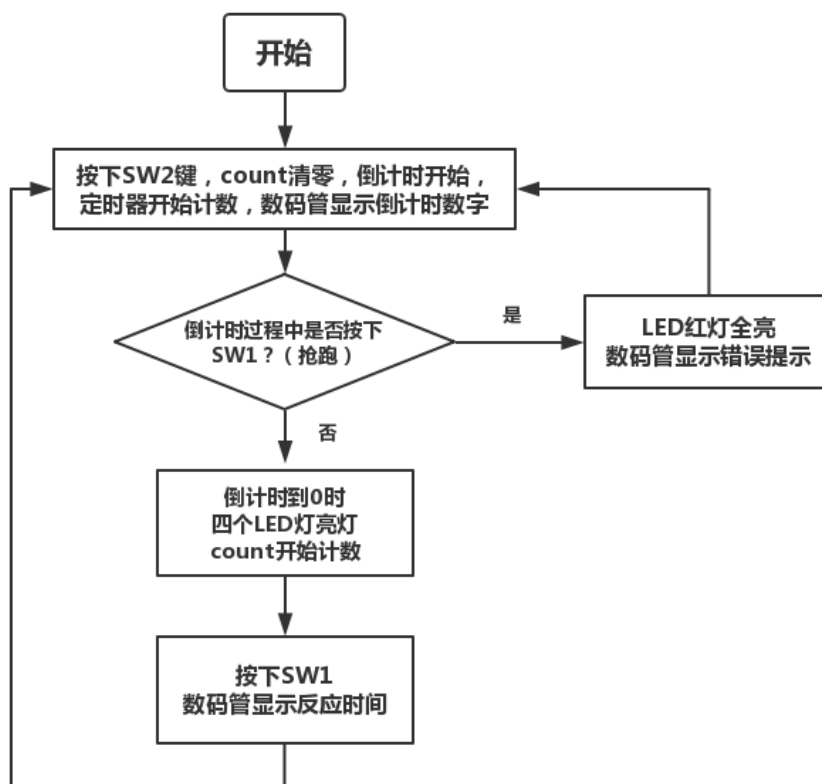
K20 芯片



K20 系列单片机有 144 引脚 LQFP、144 引脚 MAPBGA、80 引脚 LQFP 和 64 引脚 LQFP、32 引脚 QFN 等 12 种封装可供选择，片内集成了 SPI，I2C，UART，高达 16 位 ADC，CMP，RTC，PIT，LPTMR，16 位 FTM，PDB，DMA，电容触摸控制器等多种外围设备。

上图为本次实验所使用的 80 引脚封装的单片机，具有 512KB 的 FLASH 空间和 128KB 的 SRAM 空间。实验中使用了 LED\*4，数码管\*6，按键\*8，定时器\*1，计时器\*1 这几个硬件模块。

#### 四、 软件系统设计



#### 五、 调试过程

1. 在数码管上显示的时候六个灯无法显示应显示的数字

经检查发现是 LED 片选的时候左移循环逻辑有误, 后经过如下修改问题得到解决。

```

1.  if(flag_reset==1)//复位
2.  {
3.      flag=1;
4.      Select_LED=Select_LED5;
5.      GPIOC_PDOR=0xFFFF;
6.      for(i=3;i>0;i--)
7.      {
8.          GPIOA_PDOR |= 0x03F000;
9.          GPIOD_PDOR &= ~0xFF;
10.         GPIOD_PDOR |= num[i];
11.         GPIOA_PDOR &= Select_LED;
12.         delay();
    
```

```
13.     delay();
14.     delay();
15.     Select_LED=Select_LED5;
16. }
```

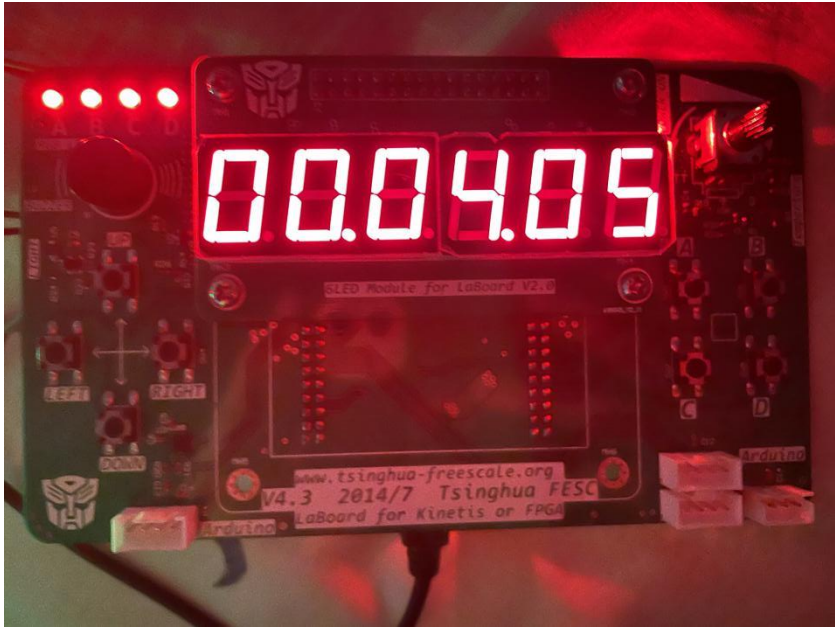
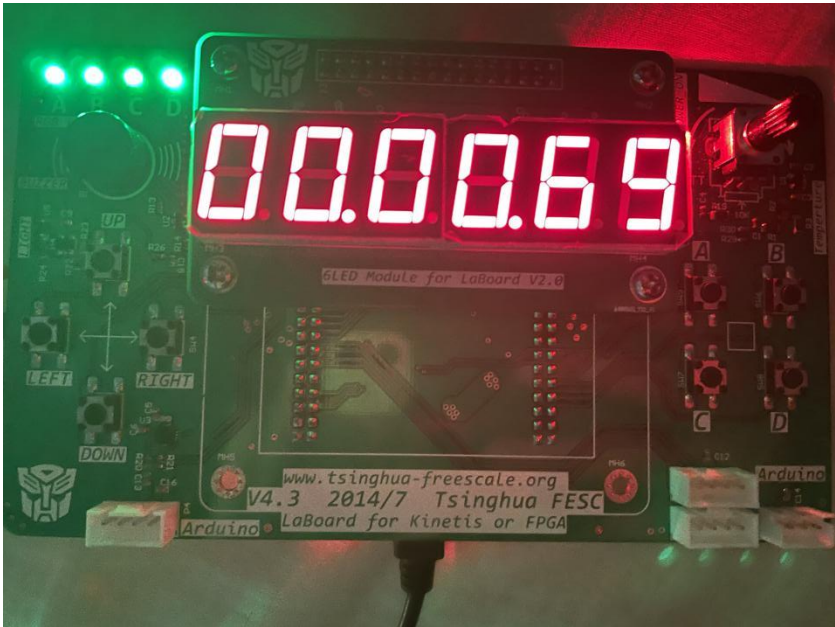
## 2. 无法改变 PIT 定时中断时长使得它按照需要的时长来中断

通过对 PIT 初始化函数的修改，再通过主函数输入合适的参数值，得到以下代码，问题得到解决。

```
1. PIT_init(0x0A);
2.
3. void PIT_init(unsigned int number_ms)
4. {
5.     /*Trun on PIT clocks*/
6.     SIM_SCGC6|=SIM_SCGC6_PIT_MASK;
7.
8.     /*Enable PIT Module*/
9.     PIT_MCR&=~(PIT_MCR_MDIS_MASK);
10.
11.     /*Setup the channel0 of PIT*/
12.     PIT_LDVAL0=20000*number_ms;//小灯闪烁时间
13.
14.     /*Enable the IRQ of channel0,PIT*/
15.     PIT_TCTRL0|=PIT_TCTRL_TIE_MASK;
16.
17.     /*Running channel0,PIT*/
18.     PIT_TCTRL0|=PIT_TCTRL_TEN_MASK;
19.
20. }
```

## 六、实验结果

实验结果展示如下：

项目	图片
红灯开启并按下 SW1 测试反应速度	
绿灯开启并按下 SW1 测试反应速度	

<p>蓝灯开启并按下 SW1 按键测试反应速度，数码管显示分、秒、毫秒计时</p>	
<p>抢跑错误提示 数码管显示 EROR, 四个红塞 LED 灯亮起</p>	

## 七、实验分析

创新点：

①通过 SW5~SW8 按键对 LED 颜色进行改变

②可以使用分秒毫秒对反应时间进行计时，可以达到一个简单秒表的功能

③在抢跑时可以在数码管上显示“ERROR”，达到简单的人机交互的目的，提高了游戏的可玩性。

讨论：

在展示过程中我们对 pit 定时中断以及如何实现准确的分秒计时进行了讨论，我是通过 time\_pro 函数对时间进行处理，达到 ms 千位进位，秒和分钟均为 60 个刻度进位，达到最终的时间控制。

## 八、实验心得

通过本次实验，我更加熟悉了 PIT 计时器的使用，加深了对数码管、LED 灯的显示的印象和理解。从设计函数再到编写每一个函数，到尝试运行结果，根据运行结果推测问题出现在哪里，从而对代码进行调整，因为对中断的理解不够清晰，中间耗费了大量时间尝试主函数中相应函数顺序的编写。在编写函数的过程中也遇到了很多问题，通过查找资料自主学习等方式完成了此次实验内容，完成之后发现逻辑线十分清晰，感觉代码能力也在这个过程中得到了提高，对微机的理解也更加深入，对之前微机原理所学习的知识和内容有了更深刻地理解。





成 绩	
评 定	
教 师	
签 名	

# 四川大学电气信息学院 计算机应用设计（单片机） 实验报告

实验名称： 实验二 路口红绿灯管理模拟

实验地点： 望江校区高压楼 501

年 级： 2018 级

姓 名： 赖梦婷

学 号： 2018141442020

实验时间： 2020 年 12 月 3 日

## Lab Project2 路口红绿灯管理模拟

### 九、实验内容

1. 左边 2 个 LED 为横向路口红绿灯
2. 右边 2 个 LED 为纵向路口红绿灯
3. 横向路口为主干道，纵向路口为一般道路，两个路口绿灯通行时间分别为 9 秒和 6 秒。
4. 在绿灯变为红灯之前插入 1 秒黄灯时间
5. 在数码管上左边和右边分别显示两个路口倒计时等待时间

### 二、设计思路

本实验要完成路口红绿灯管理模拟，需要实现用四个 LED 灯指示横向、纵向路口红绿灯状态、用数码管显示路口等待时间等基本功能。本实验以 MK20DN512 单片机为核心，设计的重点主要是如何定时？如何控制四个 LED 灯的颜色？如何将倒计时的数字显示在数码管上？由于软件延时将导致 cpu 的利用率不高，定时不够准确等问题，故本实验中使用 PIT 定时模块这一硬件定时功能。

整个程序从功能上可将其分为 PIT 计时，LED 灯显示、数码管显示两个部分进行设计，

PIT 定时功能的实现分为三步：SIM 使能相应的 PIT 时钟；初始化 PIT；编写中断服务子程序完成相应的功能；

控制 LED 灯颜色分为三步：将四个 LED 灯相应的端口设置为 GPIO 功能，并设置为输出；初始化；在中断服务子程序中编写控制的程序。黄灯的实现为控制一个 LED 灯同时亮红灯和绿灯。

数码管显示分为三步：将数码管相应的端口设置为 GPIO 功能，并设置为输出；初始化；在中断服务子程序中编写控制的程序。

### 三、硬件系统构成

#### K20 Play board

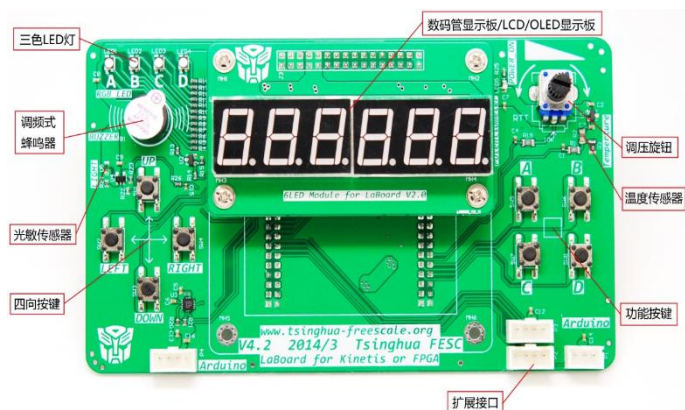


图 2-2 K20 Play board

## 1.LED 数码管显示

在单片机系统中，发光二极管（LED）常常作为重要的显示手段。由于 LED 显示器主要用于显示各种数字符号，故又称之为 LED 数码管，每个显示器还有一个圆点型发光二极管，由于显示小数点。

## 2.PIT 模块

PIT 是周期中断定时器的名称，实际上 PIT 模块就是一个 24 位递减计数器，每过一个总线周期，计数器进行减一操作，当计数器减为 0 之后，触发中断，并再次自动载入初值。

## 四、软件系统设计（包括程序流程图（加注释））

首先，初始化将使用到的 LED 灯、数码管、PIT 模块（设置定时周期为 1ms）；

然后，使用一个循环，循环进行 LED 灯左边两个亮红灯右边两个亮绿灯，两侧数码管倒计时 6 秒；四个灯亮黄灯，时间为 1 秒；LED 灯右边两个亮红灯右边两个亮绿灯，两侧数码管倒计时 9 秒；四个灯亮黄灯，时间为 1 秒；

程序流程图如图 2-1 所示：

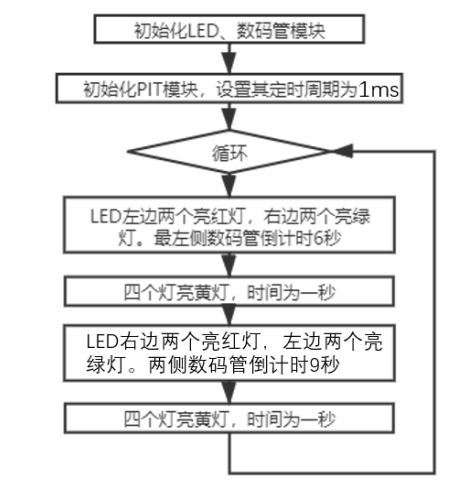


图 2-1 路口红绿灯管理模拟系统程序流程图

图 2-1 程序流程图

其中，循环过程内功能的实现见下面的子程序流程图 2-2：

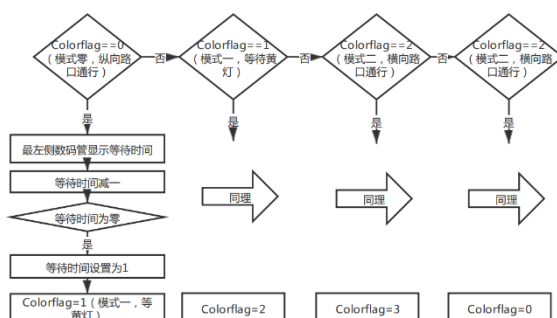


图 2-2 子程序流程图

## 五、调试过程

### 1. 硬件调试

经验证，本实验所使用的单片机功能均正常，故而不需特别调试。

### 2. 软件调试：

在调试阶段，出了几个错误，而后改正，一一将这些错误记录如下：

错误一：在中断服务子程序里，在完成一次中断后的操作之后，没有清除 PIT 标志。

改正：加上代码：PIT\_TFLG0|=PIT\_TFLG\_TIF\_MASK;

错误二：在设置转换 LED 灯颜色前，没有先将所有小灯关闭，再设置打开相应颜色的小灯，导致颜色显示错误。

改正：在设置转换 LED 灯颜色前，先将所有小灯关闭，再设置打开相应颜色的小灯。

完整实验代码见程序文件中。

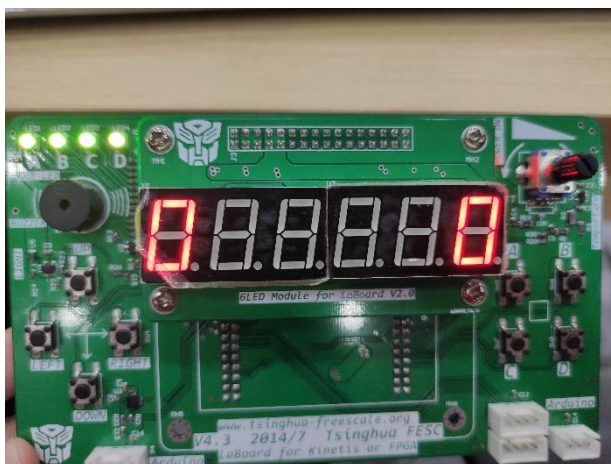
## 六、实验结果

左边两个灯指示主干道的情况，通行时间为 9s，右边两个灯指示一般道路的情况，通行时间为 6s：

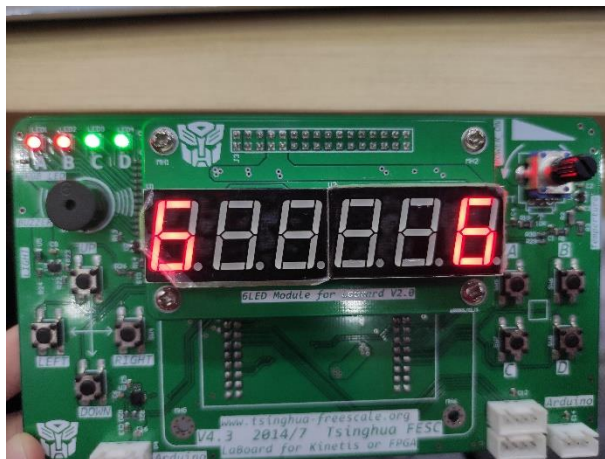
主干道通行：



亮黄灯：



纵向道路通行:



## 七、实验分析

总体来说, 本实验是一个比较简单的实验, 所用的模块和变量较少, 使用的函数也比较简单。所以完成实验所用的时间并不多, 期间出现的错误也不多。从实验结果来看, 可以正确实现功能。

## 八、实验心得

单片机是自动化专业的一门重要的专业核心课程, 该课程的应用性很强, 因此, 较多的实验对我们加深对理论知识的理解 and 应用是很有帮助的。

在从前学习 C 语言、数据结构等课程的时候, 往往一头雾水, 不知道学着有什么用。通过单片机这门课程, 我们有了把所学知识用到实践的机会。理论联系实际的过程, 也使得我们对所学知识的内容比如数电、微机原理等课程了解得更深入透彻了。

最后, 感谢课堂中不厌其烦悉心教导的老师 and 助教老师, 你们给我们学习单片机的道路上提供了很多帮助。感谢一起完成实验的组员, 是我们友好的互相协作, 共同努力, 才令这次实验变得如此有趣、有意义。今后我们也将继续认真学习本课程, 学习的路途还远, 我们的脚步也不会停下。



成 绩	
评 定	
教 师	
签 名	

# 四川大学电气信息学院 计算机应用设计（单片机） 实验报告

实验名称：\_\_\_\_\_实验三 关照监控报警\_\_\_\_\_

实验地点：\_\_\_\_\_高压实验楼\_\_\_\_\_

年 级：\_\_\_\_\_2018 级\_\_\_\_\_

姓 名：\_\_\_\_\_朱琪霖\_\_\_\_\_

学 号：\_\_\_\_\_2018141241025\_\_\_\_\_

实验时间： 2020 年 12 月 3 日

## 一、实验内容

1. 采样光照传感器的值，显示在数码管上；
2. 利用按键可以查看的光照上下限报警值；
3. 利用按键改变光照上下限报警值；
4. 光照正常时，LED1 绿灯亮；
5. 光照超上限时，LED1 红灯亮，同时蜂鸣器响（报警）；
6. 光照低于下限时，LED1 蓝灯亮，同时蜂鸣器响（报警）；
7. 光照恢复正常值，显示恢复正常状态。

## 二、设计思路

本实验要完成光强检测报警器，需要实现光强信号的采集与 A/D 转换、数据处理、数据输出等基本功能。从功能上可将其分为光强信号采集及 A/D 转换、数据处理、人机交互、执行四大部分进行设计，设计思路框图如下图 3-1 所示：

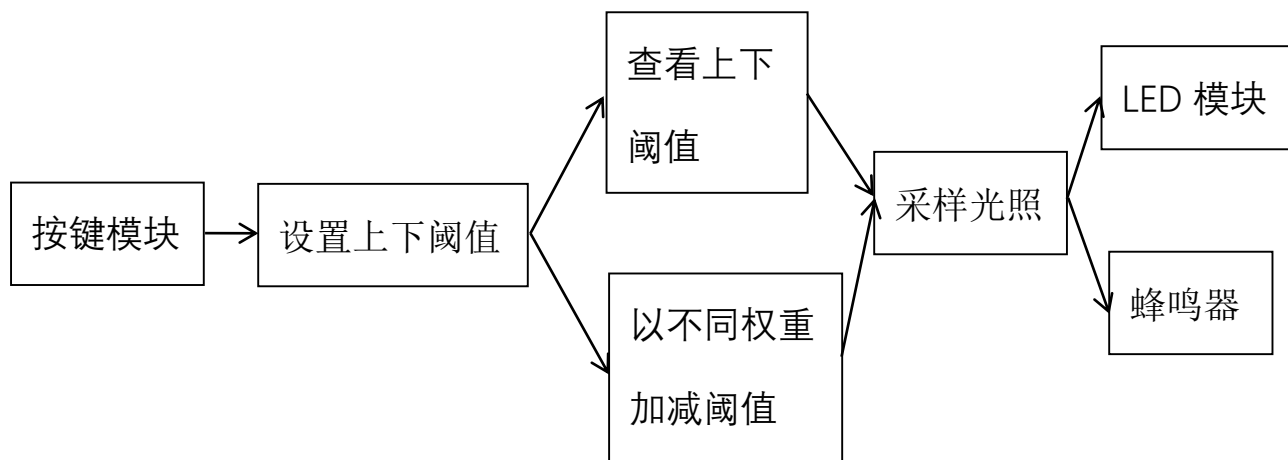


图 3-1 设计思路框图

本实验以 MK20DN512 单片机为核心，在单片机内部完成数据的存储及处理功能，通过数模转换芯片完成模拟信号到数字信号的转换及输入，再将数据存入存储芯片，在单片机进行数据处理后在对需要显示的数字信号进行译码显示在数码管显示器上。本实验完成了采集功能、存储功能、数据处理功能、测试数据显示功能，达到了设计的基本要求。



### 三、硬件系统构成

#### K20 Play board

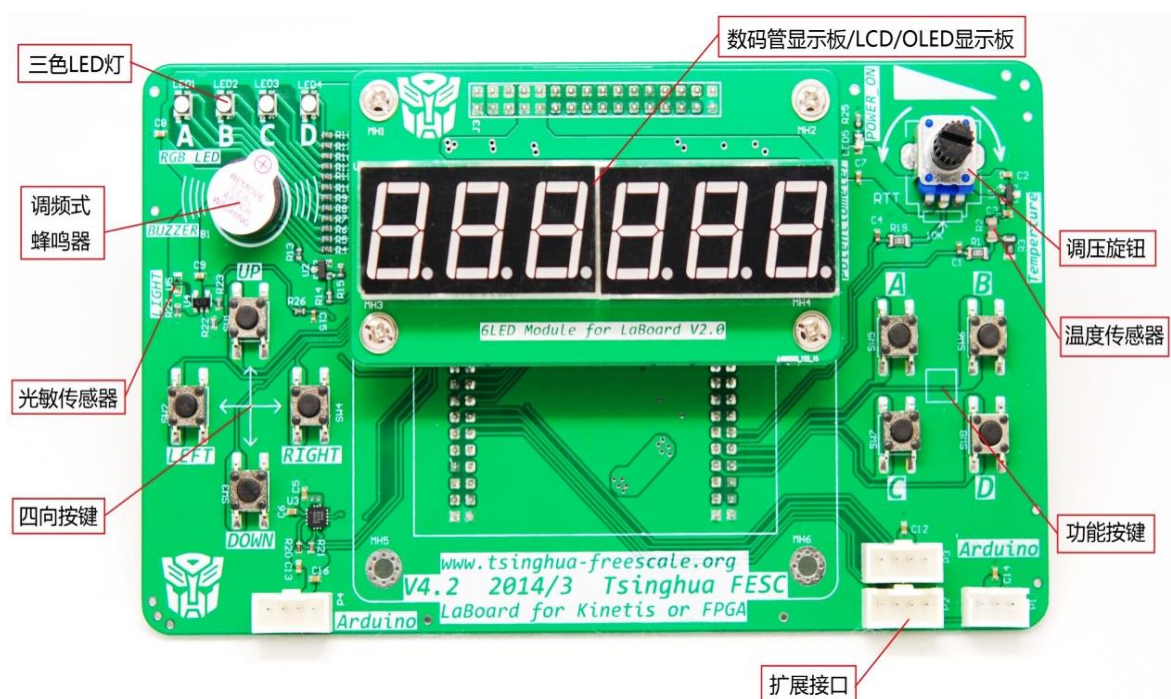


图 3-2 K20 Play board

##### 1.LED 数码管显示

在单片机系统中，发光二极管（LED）常常作为重要的显示手段。由于 LED 显示器主要用于显示各种数字符号，故又称之为 LED 数码管，每个显示器还有一个圆点型发光二极管，由于显示小数点。

##### 2.A/D 转换

AD 模块具有初始化、采样、中值滤波、均值滤波等操作。按照构件的思想，可将它们封装成独立的功能函数。AD 构件包括头文件 `adc.h` 和 `adc.c` 文件。AD 构件头文件中主要包括相关宏定义、AD 的功能函数原型说明等内容。AD 构件程序文件的内容是给出 AD 各功能函数的实现过程。

##### 3.按键

按键部分实现的主要原理是单片机读取与按键相连接的 I/O 口状态，来判定按键是否按下，达到系统参数设置的目的。键盘在单片机应用系统中的作用是实现数据输入。命令输入，



是人工干预的主要手段。

#### 四、软件系统设计（包括程序流程图（加注释））

光强信号采集子程序,主要完成光强信号采集与 A/D 功能,采集子程序主要包括写命令、写数据和读数据等部分。

数据处理子程序,当单片机收到光强信号后,数据处理子程序对该数据进行处理,主要是把采集到的二进制的光强数据转换成十进制光强数据。人机交互子程序包括按键子程序、LED 显示子程序。LED 显示子程序的功能是,实现将数据处理后的十进制光强数据,使用 LED 显示出来。执行子程序,该子程序所实现的功能,是把程序设置的系统光强限定值与数据处理子程序处理后的当前光强值进行比较,根据比较的结果,执行单片机的 I/O 口输出的状态。控制 LED 灯与蜂鸣器,主程序流图如下图 3-3 所示

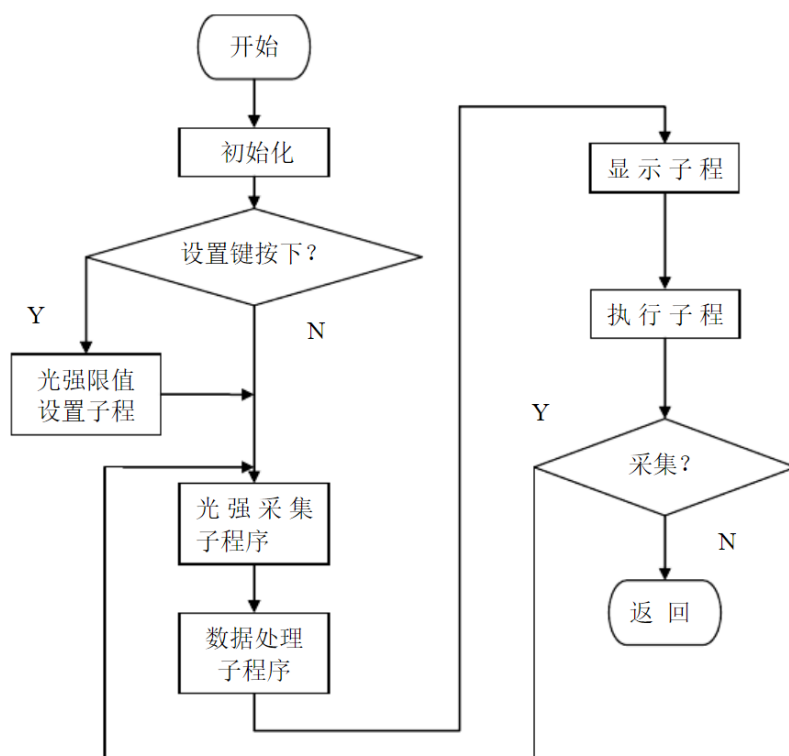


图 3-3 主程序流图

模/数（A/D）转换测量子函数用来控制对模拟输入电压进行 A/D 转换，并将对应的数值移入内存单元。

在本设计当中，当按键被按下时，I/O 口电平为低；松开时，I/O 口电平为高。按键扫描程序通过读取 I/O 口的电平即可知道对应按键的状态。

本部分通过单片机 I/O 口输出的高电平或者低电平，LED 和蜂鸣器报警。

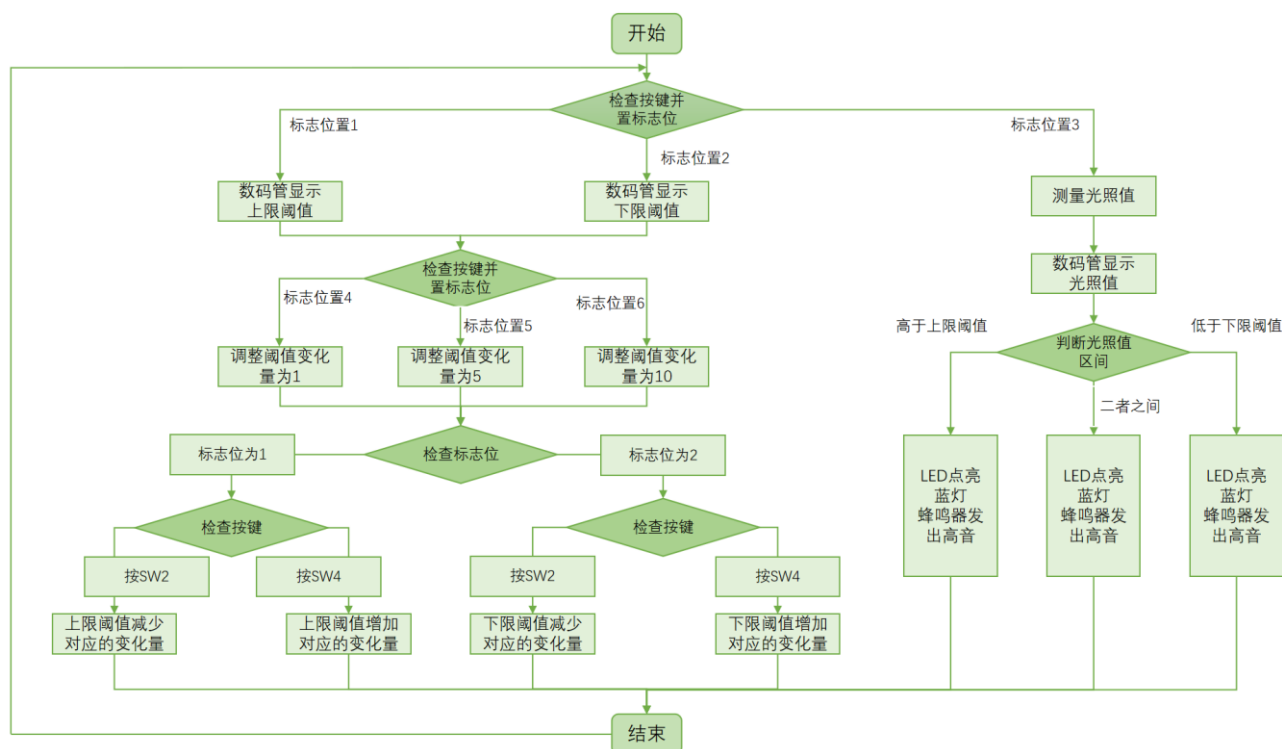


图 3-4 子程序流程图

## 九、调试过程

### 3. 硬件调试：

首先，我们应该排除元器件失效问题。造成这类错误的原因有两个：一是元器件买来时就已坏了；另一个是由于焊接错误,造成器件损坏。

其次，排除电源故障问题。在通电前，一定要检查电源电压的幅值和极性，否则很容易造成集成块损坏。若有高压，将会损坏单片机等，有时会使应用系统中的集成块发热损坏。

本次实验的硬件调试顺序为先显示器后键盘。在显示器调试通过后，键盘调试就比较简单，完全可以借助于显示器，利用程序进行调试。利用开发装置对程序进行设置断点，通过断点可以检查程序在断点前后的键值变化，这样可知键盘工作是否正常。

### 4. 软件调试：

在确认硬件设备无误后，我们就可以开始利用 Codewarrior 对该实验进行调试：

代码如下：

```

⊕ * main implementation: use this 'C' sample to create your own application

#include "derivative.h" /* include peripheral declarations */
#include "LED.h"
#include "LED_CODE.h"

⊕ int LED(void)
{
    SIM_SCGC5 |= (0x800 | 0x200 | 0x1000); // 使能PORTE D C A时钟
    PORTA_PCR12 = 0x100; // LED0灯对应引脚设置为GPIO
    PORTA_PCR13 = 0x100; // LED1灯对应引脚设置为GPIO
    PORTA_PCR14 = 0x100; // LED2灯对应引脚设置为GPIO
    PORTA_PCR15 = 0x100; // LED3灯对应引脚设置为GPIO
    PORTA_PCR16 = 0x100; // LED4灯对应引脚设置为GPIO
    PORTA_PCR17 = 0x100; // LED5灯对应引脚设置为GPIO
    PORTD_PCR0 = 0x0100; // 片选1设置为GPIO
    PORTD_PCR1 = 0x0100; // 片选2设置为GPIO
    PORTD_PCR2 = 0x0100; // 片选3设置为GPIO
    PORTD_PCR3 = 0x0100; // 片选4设置为GPIO
    PORTD_PCR4 = 0x0100; // 片选5设置为GPIO
    PORTD_PCR5 = 0x0100; // 片选6设置为GPIO
    PORTD_PCR6 = 0x0100; // 片选7设置为GPIO
    PORTD_PCR7 = 0x0100; // 片选8设置为GPIO
    GPIOA_PDDR |= 0x3F000; // 六个LED对应端口都设置为输出
    GPIOA_PDOR |= 0x3F000; // 六个LED设置为高电平
    GPIOD_PDDR |= 0xFF; // 片选设置为输出
    GPIOD_PDOR = 0xFF; // 片选全无效
}

⊕ void ADC0_Init (void) {
    /*enable ADC0 clock*/
    SIM_SCGC6 |= SIM_SCGC6_ADC0_MASK;
    SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK;
    /* Set pin0 of PORTE as analog function */
    PORTB_PCR3 = 0x0000;
    //long sample time single-end 12bit conversion
    ADC0_CFG1 = 0X00000014;
    //ADxxxa channel select
    ADC0_CFG2 = 0X00000000;
    //default voltage reference Vrefh and Vrefl, software trigger
    ADC0_SC2 = 0X00000000;
    //continuous conversions
    ADC0_SC3 = 0X00000008;
    //interrupt disable and select ADC0_SE13a channel as input
    ADC0_SC1A = 0X0000000d;
    //SIM_SCGC5 |= SIM_SCGC5_PORTE_MASK;
    //SIM_SCGC3 |= SIM_SCGC3_ADC1_MASK;

    //PTE0 and PTE1 configured as pin of ADC module
    //PORTE_PCR0 = PORT_PCR_MUX(0X0);
    //PORTE_PCR1 = PORT_PCR_MUX(0X0);

    //long sample time and single-end 8bit conversion
    //ADC1_CFG1 = ADC_CFG1_ADLSMP_MASK + ADC_CFG1_MODE(1);

    //continuous conversions
    //ADC1_SC3 = ADC_SC3_ADC0_MASK;

    //interrupt disable and select AD1_se4a channel as input
    //ADC1_SC1A = ADC_SC1_ADCH(4); //potentiometers sensor path acquire
}

```

```

void PIT_init(unsigned int number_ms)
{
    /*Trun on PIT clocks*/
    SIM_SCGC6|=SIM_SCGC6_PIT_MASK;

    /*Enable PIT Module*/
    PIT_MCR&=~(PIT_MCR_MDIS_MASK);

    /*Setup the channel0 of PIT*/
    PIT_LDVAL0=20000*number_ms;

    /*Enable the IRQ of channel0,PIT*/
    PIT_TCTRL0|=PIT_TCTRL_TIE_MASK;

    /*Running channel0,PIT*/
    PIT_TCTRL0|=PIT_TCTRL_TEN_MASK;
}

unsigned int ADC_Result;
unsigned char xianshi[3];
unsigned char xianshi1[3];
unsigned char xianshi2[3]; //数码管显示
unsigned int yuzhishang=0;
unsigned int yuzhixia=0; //光照上下限
extern int mode;
extern int shu1,shu2;
unsigned int flag1=0;
unsigned int flag2=0;
unsigned int flag11=1;
unsigned int flag22=1;
unsigned int j=0;
unsigned int a,jia_flag,jian_flag; //加减标志
unsigned long int led[3]={BIT12,BIT13,BIT14}; //用右边三个数码管显示光照转换值
unsigned short num xianshi[11]= {0xA0,0xBE,0x62,0x2A,0x3C,0x29,0x21,0xBA,0x20,0x28,0x7F};

int main(void)
{
    PIT_init(0x01); //定时周期
    LED_Init(); //LED灯
    ADC0_Init(); //数码管显示
    LED();
    EN_Init();
    KEY_Init();
    Beep_Init();
    FTM2_Init();
    for(;;)
    {
        if(mode==1) //高阈值显示
        {
            unsigned int jia;
            unsigned int shang;
            for(jia_flag=0;jia_flag<3&&flag11;) //数码管循环显示
            {
                if(flag1) //通过SW3 SW4计算一次新阈值
                {
                    flag1=0;
                    yuzhishang+=shu1;
                }
                shang=yuzhishang; //傀儡数，等着被分离用于显示
                GPIOA_PDOR |= 0x03F000;
                GPIOD_PDOR = num_xianshi[xianshi1[jia_flag]];
                GPIOA_PDOR &= ~(led[jia_flag]);
                jia_flag++; //三位循环显示
            }
        }
    }
}

```

```

        if(jia_flag==3)
        {
            jia_flag=0;
        }
        for(jia=0;jia<3;jia++)                //分离要显示的各位数字便于显示
        {
            xianshi1[jia]=shang%10;
            shang/=10;
        }
    }}
if(mode==2) //低阈值显示
{
    unsigned int jian;
    unsigned int xia;

    for(jian_flag=0;jian_flag<3&&flag22;) //数码管循环显示
    {
        if(flag2)
        {
            flag2=0;
            yuzhixia+=shu2;} //通过SW3 SW4计算一次新阈值
            xia=yuzhixia; //傀儡数，等着被分离用于显示
            GPIOA_PDOR |= 0x03F000;
            GPIOD_PDOR = num_xianshi[xianshi2[jian_flag]];
            GPIOA_PDOR &= ~(led[jian_flag]);
            jian_flag++;
            for(jian=0;jian<3;jian++)                //分离要显示的各位数字
            {
                xianshi2[jian]=xia%10;
                xia/=10;
            }
            if(jian_flag==3) //三位循环显示
            {
                jian_flag=0;
            }
        }
    }
}

```

以下结果显示代码无误，调试成功：

Problems 0 items						
Description	Resource	Path	Location	Type		

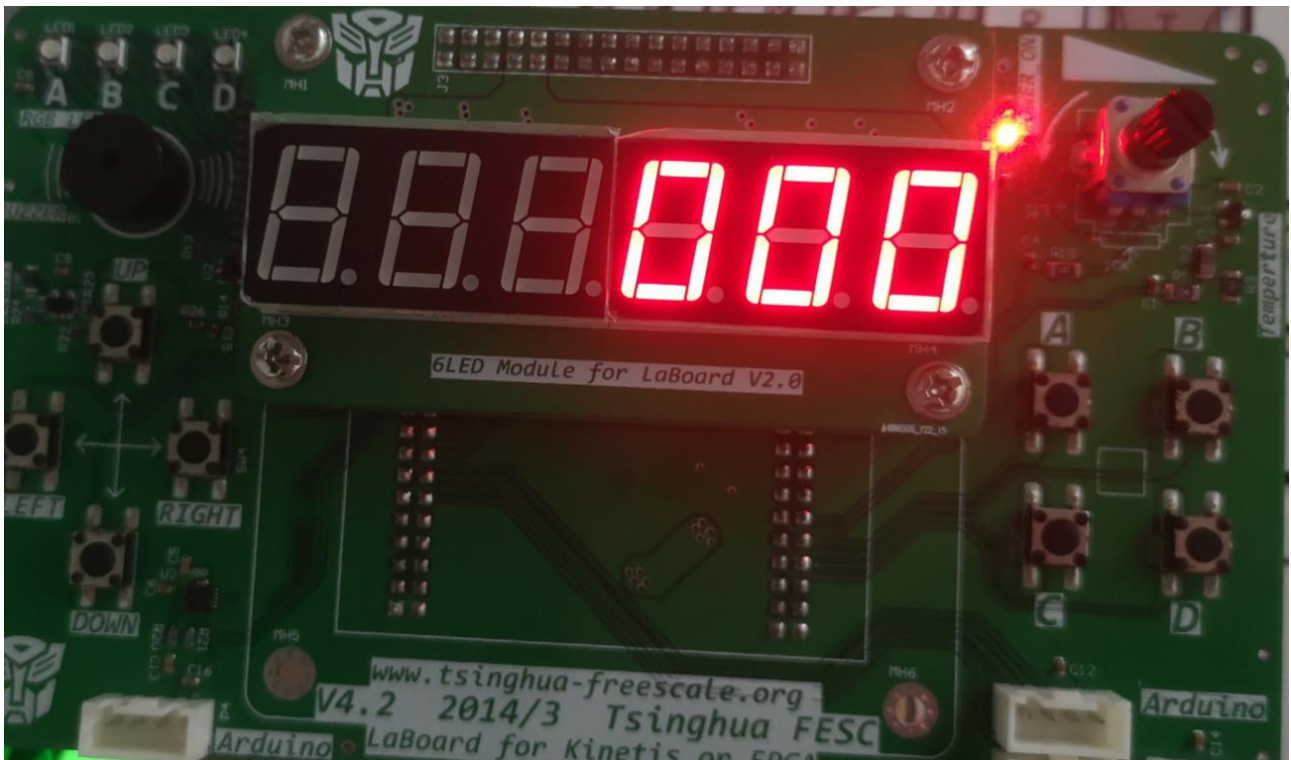
Debug

- guangzhao\_FLASH\_USBDM [CodeWarrior]
  - ARM Processors, guangzhao.elf (Suspended)
    - Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
      - 2 main() main.c:108 0x00000f30
      - 1 \_\_thumb\_startup() \_\_arm\_start.c:279 0x00000c7c

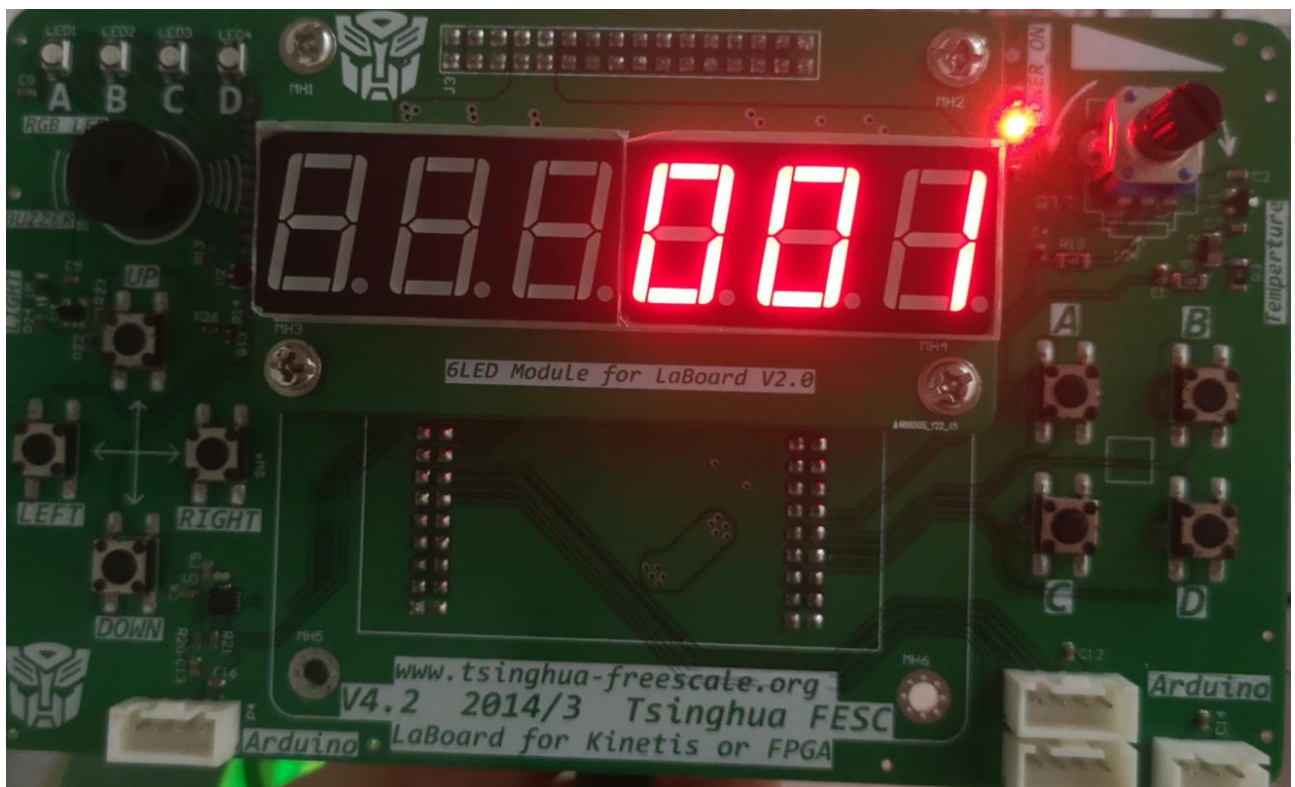
C:\Users\mac pro\Desktop\µ¥Æ→»ú\iÄ;3 1âÖÖ¼i²âÆ÷\guangzhao\FLASH\guangzhao.elf (12/8/20 9:28 PM)

## 十、实验结果

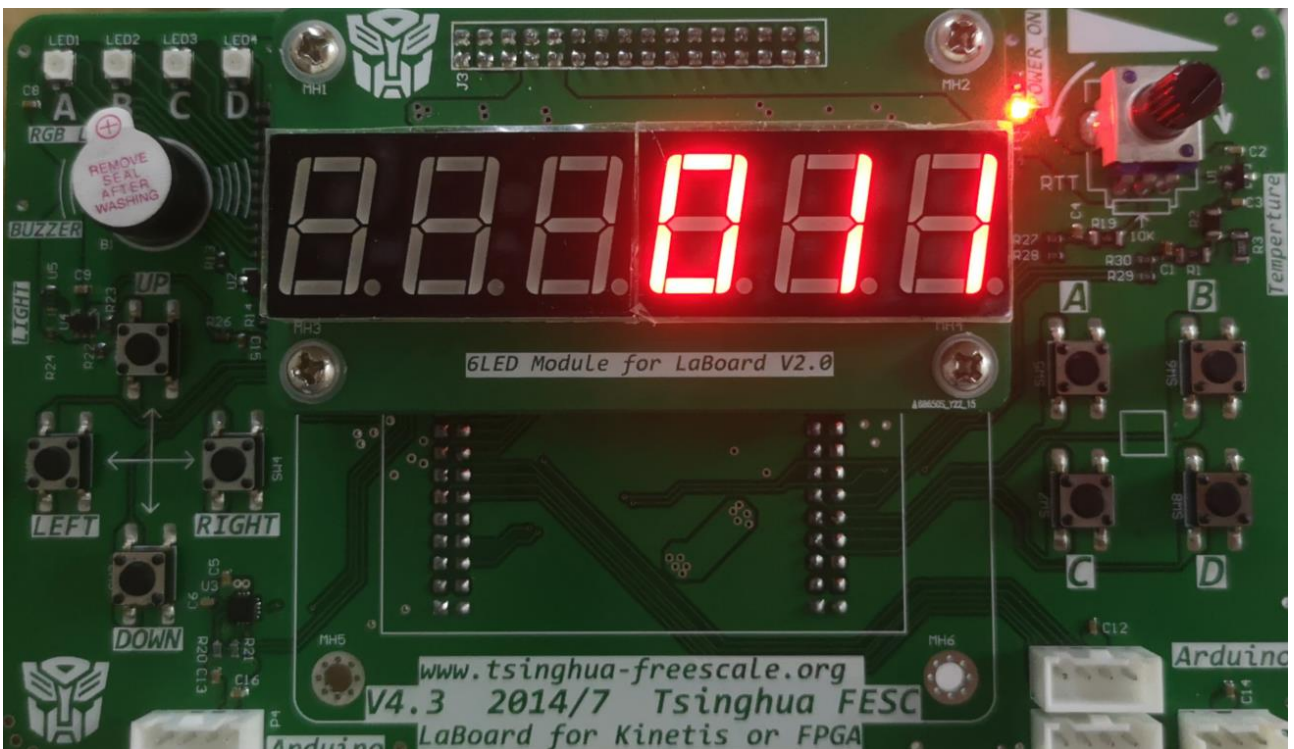
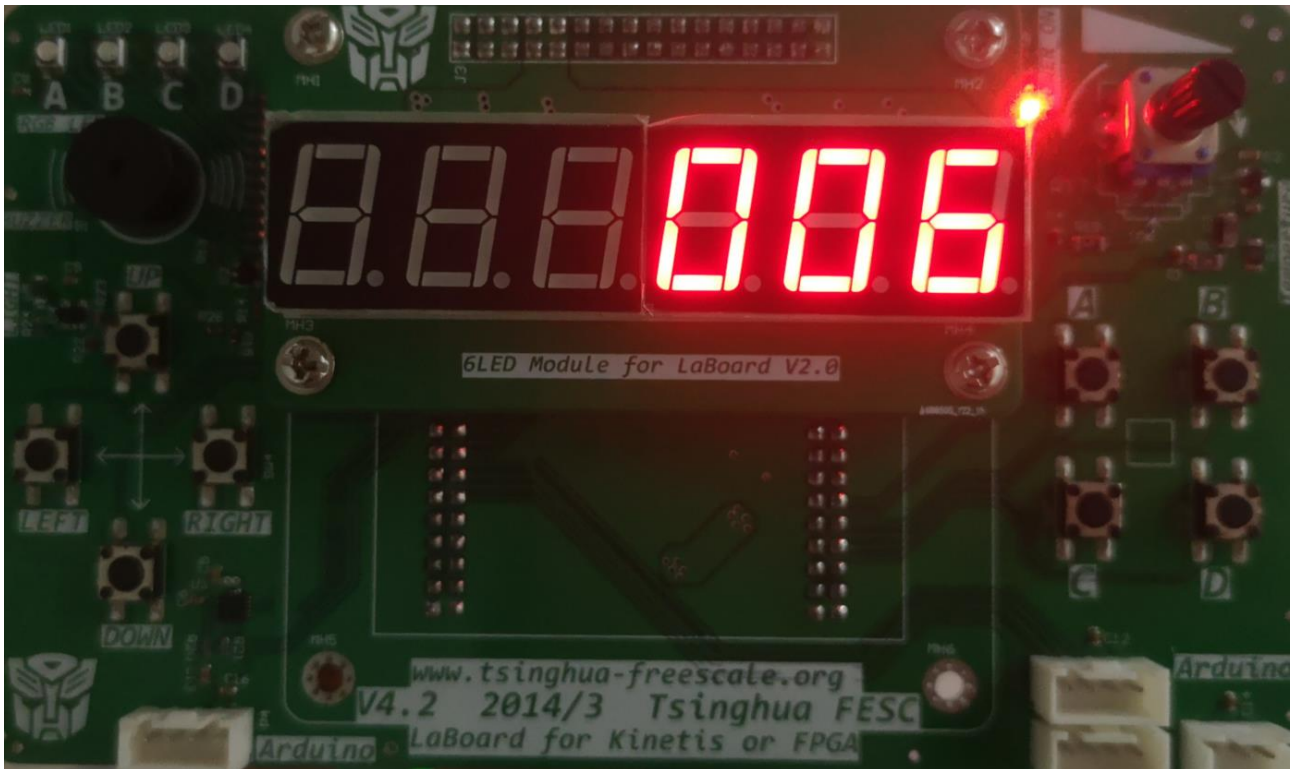
1.运行开始，按下 SW1，查看上阈值，最初未设置时为 000



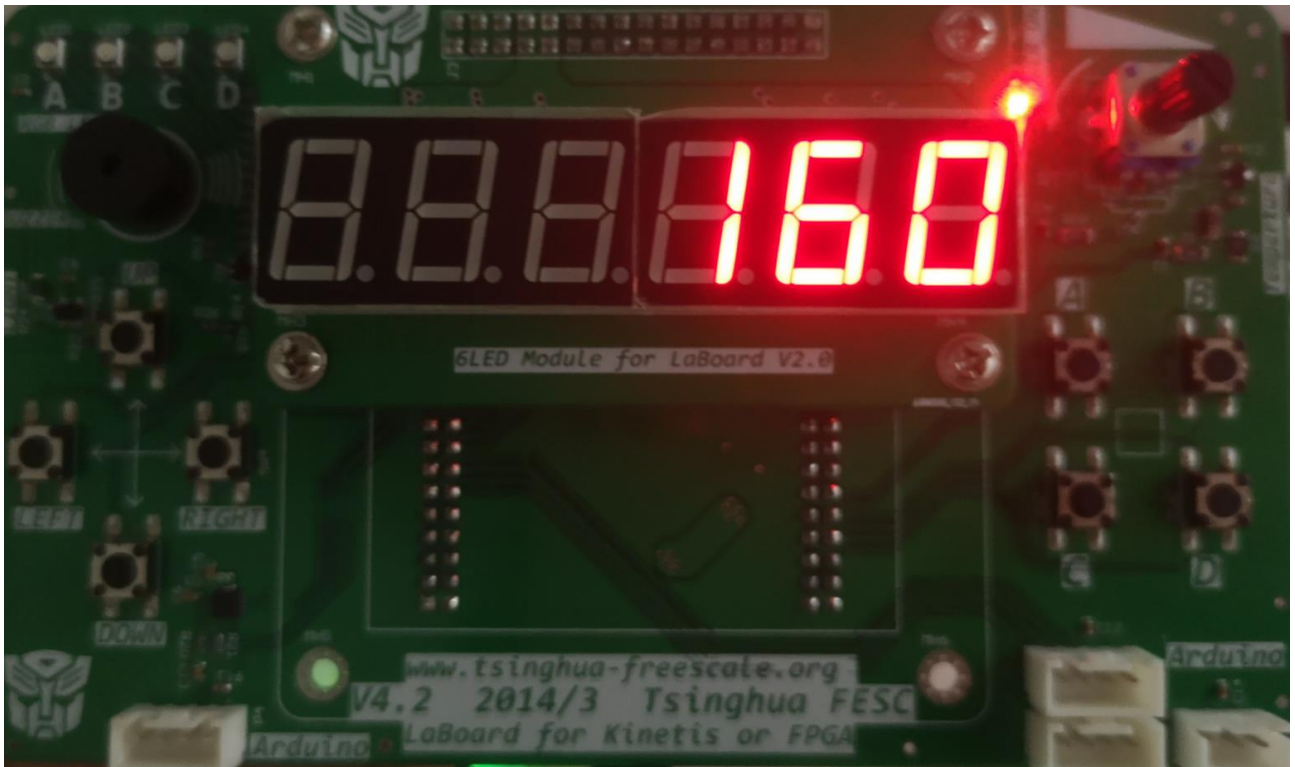
2.分别按下 SW6、SW7、SW8，再按下 SW4、SW2，每次分别加减 1、5、10，演示如下图：



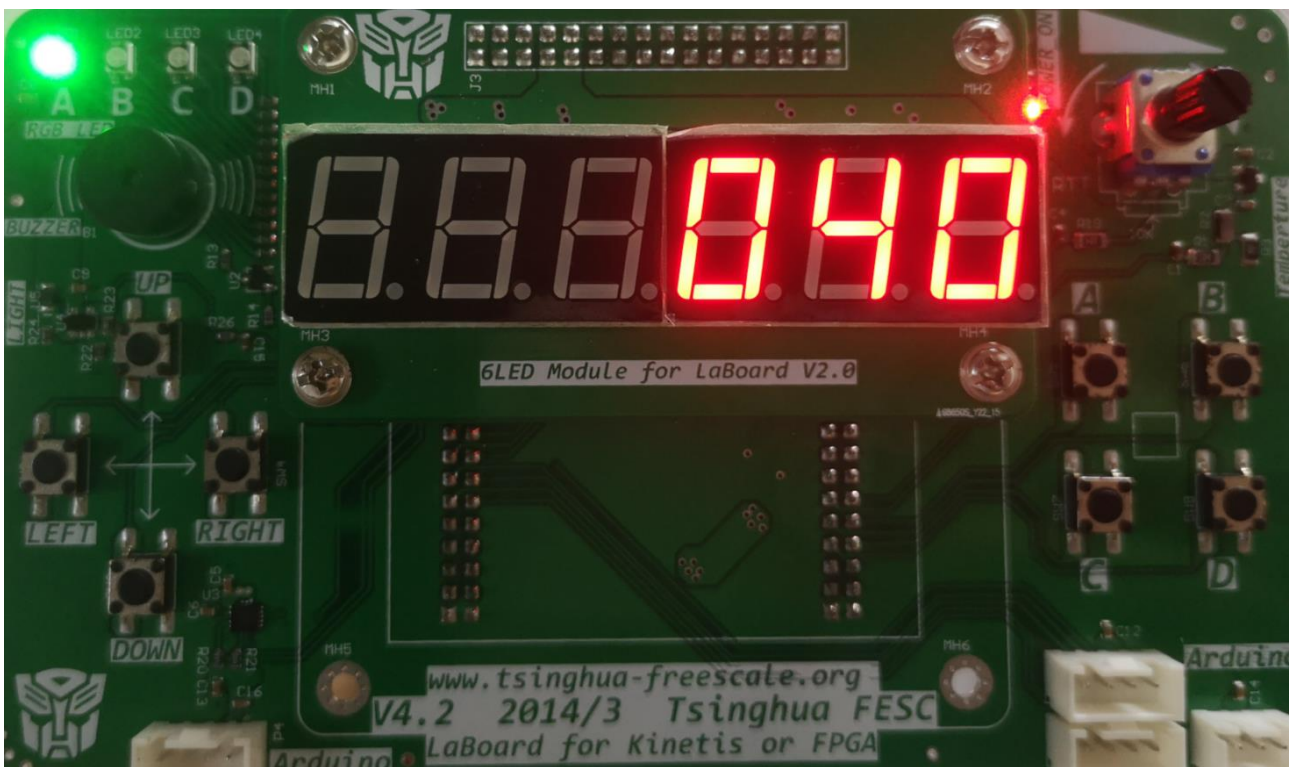




3.按照步骤 2 的方法设置上阈值为 160



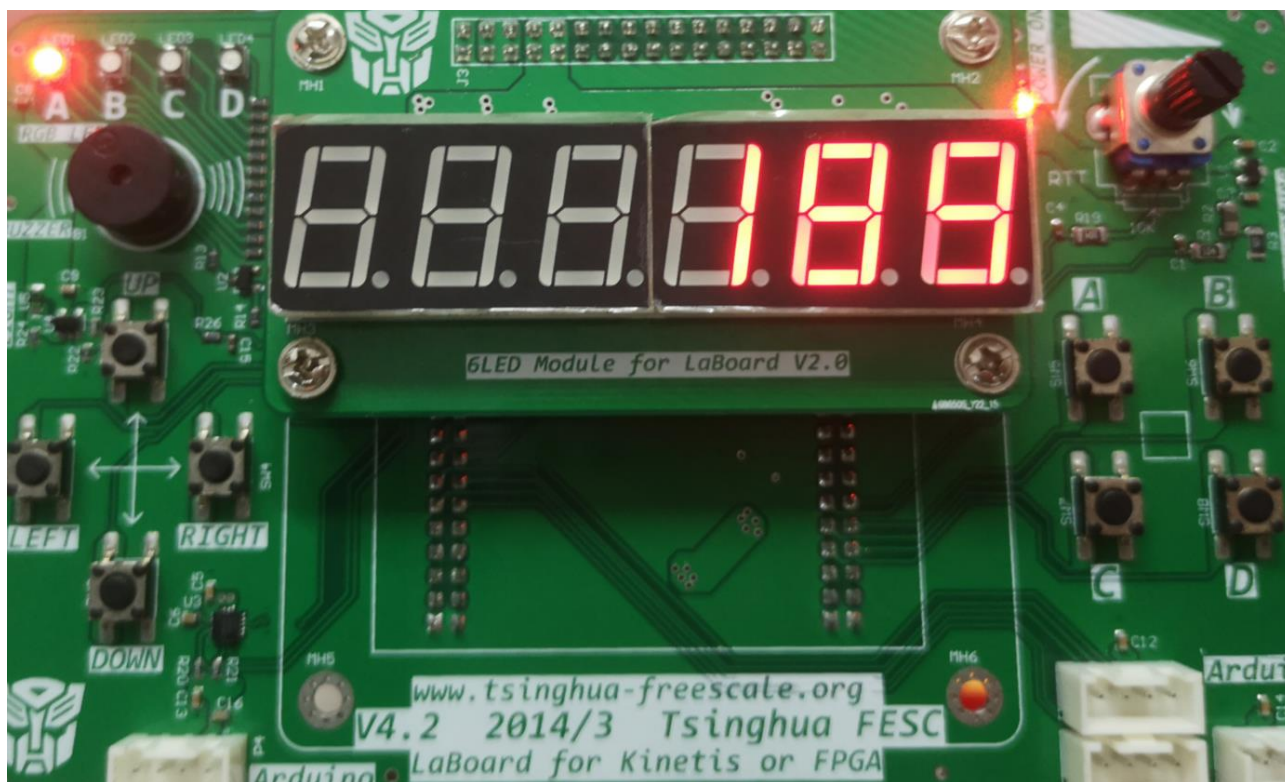
4.按照步骤 2 的方法设置下阈值为 40



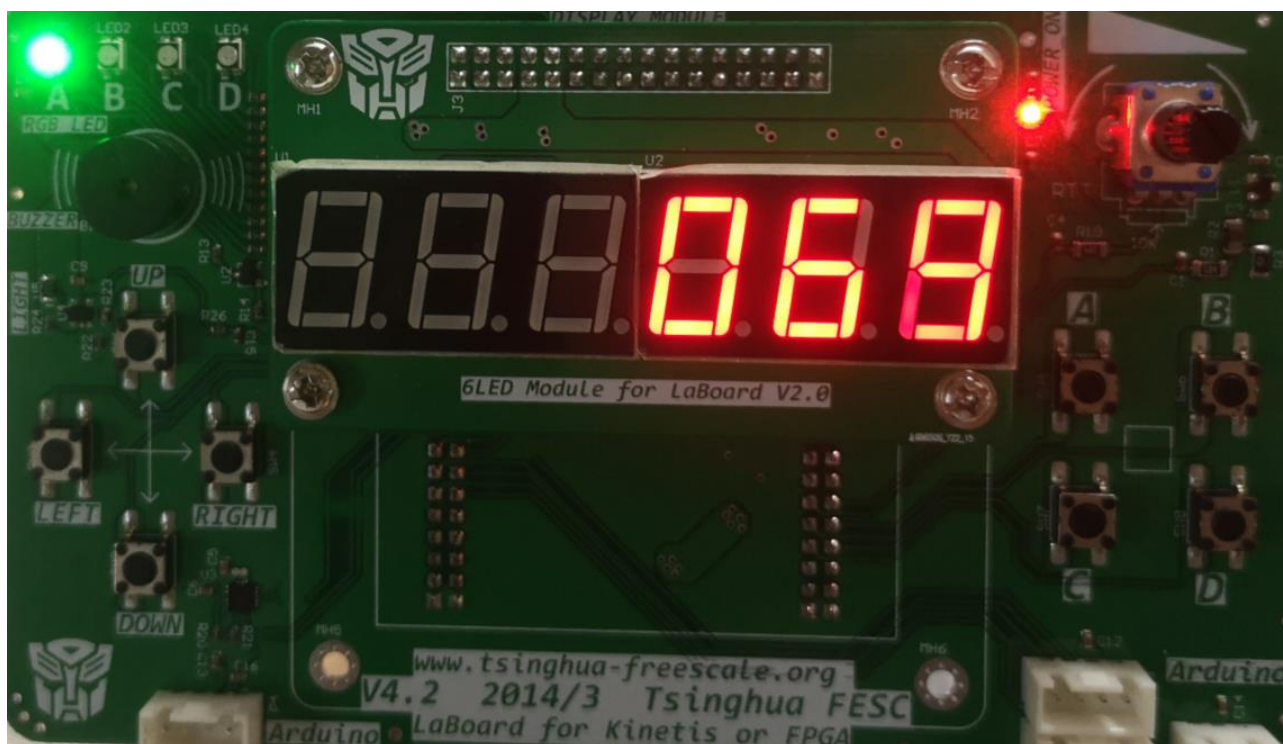


5.按下 SW5，测得此时光照值

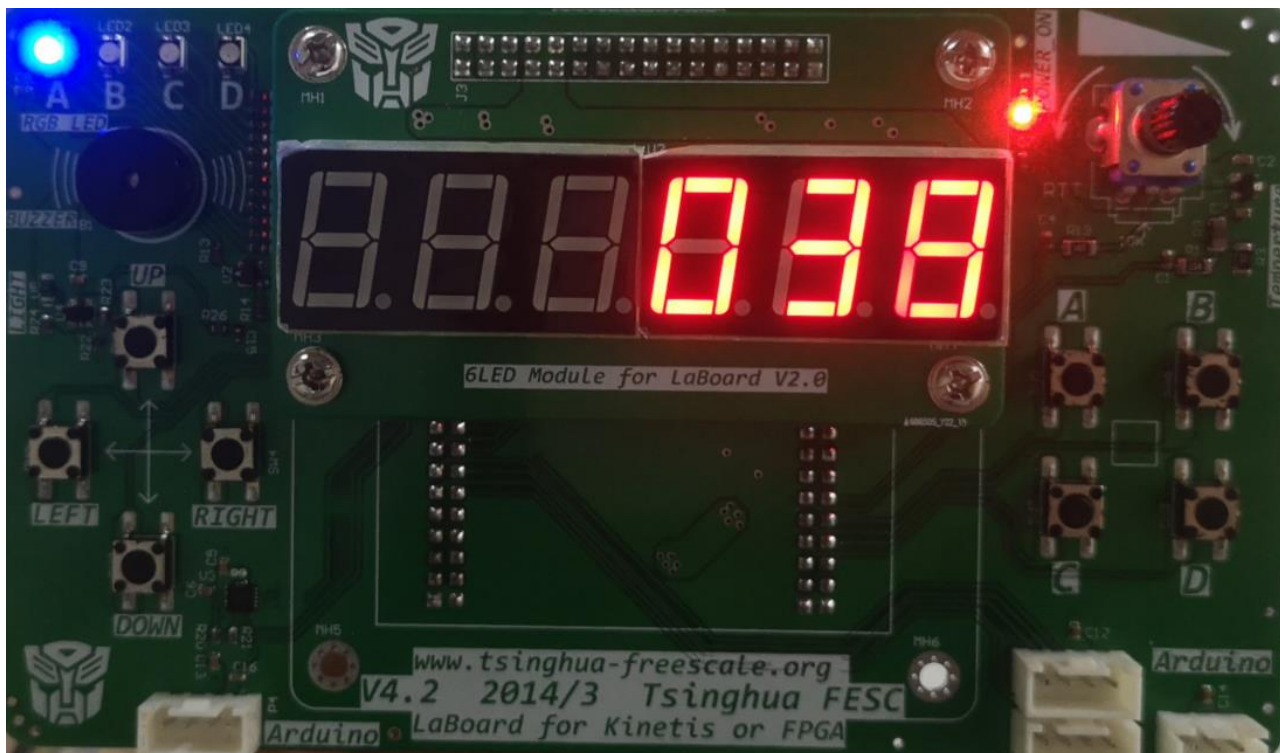
①当光照值大于 160 时，红灯亮，并且报警（此时光照值为为 188）



②当光照值在 40~160 时，绿灯亮（此时光照值为 69）



③当光照值小于 40 时，蓝灯亮，并且报警（此时光照值为 38）



## 十一、实验分析

当单片机收到光强信号后，数据处理子程序对该数据进行处理，主要是把采集到的二进制的光强数据转换成十进制光强数据。实现将数据处理后的十进制光强数据，使用 LED 显示出来。把程序设置的系统光强限定值与数据处理子程序处理后的当前光强值进行比较，根据比较的结果，执行单片机的 I/O 口输出的状态，控制 LED 灯与蜂鸣器。控制输出采用 3 个不同颜色的 LED 灯，进行显示报警，报警功能实现的是当环境光强值超过系统设置的上限值或小于系统设置的下限值时，都将通过 I/O 口驱动蜂鸣器，进行蜂鸣器报警。

## 十二、实验心得

本实验是关于光强检测报警器的设计，对环境光强信号进行采集，并将采集到的光强值转换成数字信号送到单片机进行处理，用 LED 对当前光强信号值进行显示，并且根据数据处理的结果，使用单片机控制 LED 灯报警和蜂鸣器报警。

关于本次实验，由于经验的不足和所掌握知识的限制，当初设想的一些其他功能无法完全实现。通过这次的实验让我想到许多之前犯的错误，没有把学习的目标摆正，学的很多知

识都只是为了应付考试而学，对于自己真正掌握了解的只是却显得十分不足，所以从一开始的设计就显得自己无从下手。从这次毕业设计中，让我更深刻的体会到理论联系实际的重要性，只有多听多看多实践，才能更好的运用所学知识。这对于以后的学习工作都同样的重要，让我更好的意识到：实践方能出真知。



成 绩	
评 定	
教 师	
签 名	

四川大学电气信息学院

# 计算机应用设计（单片机） 实验报告

实验名称：实验四 小游戏

实验地点：高压实验楼

年 级：2018

姓 名：许如玉

学 号：2018141442004

实验时间：2020 年 12 月 3 日

## 一、实验内容

编程以在单片机上实现以下功能，完成一个小游戏：设置一个启动键，按下后开始游戏，从最右边数码管开始，数码管上向左滚动一个随机数(1-8)。按下相应数值按键，按下时相应数码管上的显示数字停止滚动，定格显示 1s 本关游戏通过，进入下一轮游戏，否则按启动键后游戏重新开始随着通关数的增加，数码管的数字滚动频率不断加大。

## 二、设计思路

程序主要涉及到两个硬件模块，按键输入和数码管显示。两个按键部分都需要使用，故需要打开 PORTB 和 PORTE；数码管需要打开 PORTA 和 PORTD。

### 2.1 游戏运行状态

设置两个旗帜变量 Startflag 和 Endflag；Startflag=1 代表小游戏程序运行，Endflag=1 代表正在闯关。小游戏程序正在运行时主程序部分运行一个大循环，里面套一个小循环，用于正在闯关时数字从右向左滚动显示及判断是否按键正确对应。

### 2.2 游戏通关条件判断

设置两个变量 ChosenNum 和 CurrentNum。CurrentNum 代表当前数字，即在数码管上滚动的数字，其值为 1 致 8 之间的一个随机数；ChosenNum 初始值为 0，在小游戏程序运行时按键端口的八个按键按下后分别将 ChosenNum 的值修改为 1-8。

如果 ChosenNum 等于 CurrentNum，则本关通过。将当前数字显示一秒后，再次选择 CurrentNum，从最右端开始，向左滚动显示。

### 2.3 移动速度改变

程序中使用软件延时，利用 delay 函数，改变显示的时间，以改变数字滚动的速度。关卡越高，数字显示越靠左，停留显示的时间越短。

### 2.4 结束条件

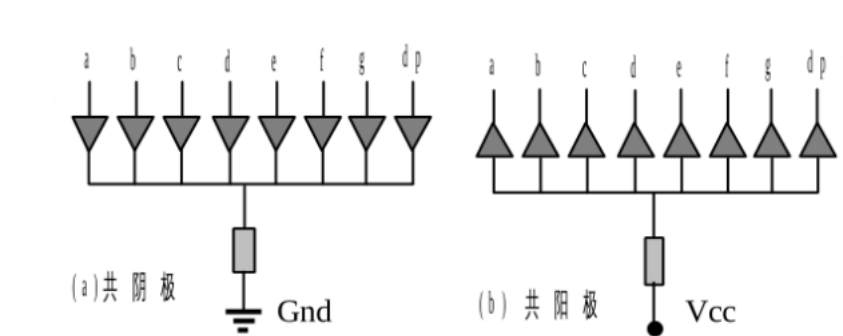
小游戏的结束条件有两种，其中一种是已经玩到了 20 关，游戏自动结束；另一种是在数字滚动结束之前都没有按下对应的按键。两种方式游戏结束后，都将在数码管上显示完成的关卡数。

### 三、硬件系统构成

#### 3.1. 按键

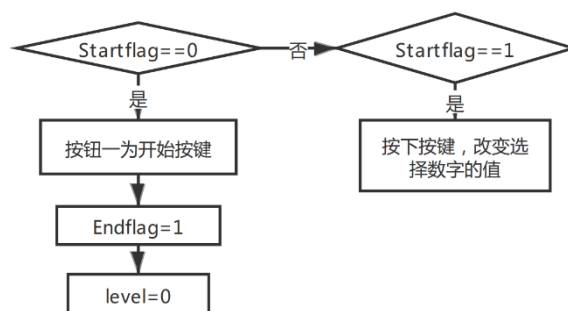


#### 2. 数码管



### 四、软件系统设计（包括程序流程图（加注释））

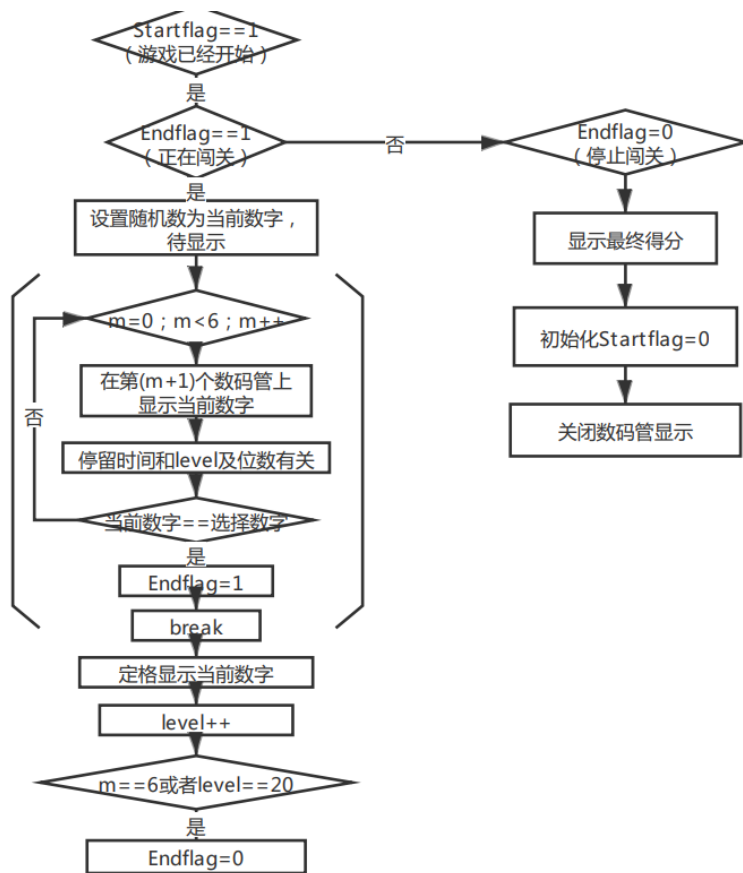
#### 4.1 中断模块：



当小游戏程序尚未开始运行时（即 Startflag=0），按下按键一，将 Startflag、Endflag 赋值为一，关卡计数变量 level 初始化为 0。当小游戏程序开始运行（即 Startflag=1）后，按下按键，修改选择数字变量（ChosenNum）的值。



## 4.2. 主程序主循环



先判断 Startflag 是否等于一，如果是则表示游戏已经开始；紧接着再判断 Endflag 是否等于一，如果是则代表正在闯关，设置 CurrentNum 为一随机值，此处设置随机数使用 rand()和 srand()函数。大循环里面有一个小循环，每一个 level 会执行一次。这个循环主要负责 CurrentNum 的滚动显示以及是否过关的判断。设置计数变量 m，循环内结尾处 m 自增一，自然条件下退出循环条件为 m 大于或等于六，即 CurrentNum 在六片数码管上都显示过了。在每一片数码管上停留的时间与 level 以及 m 的值有关，具体表达式型如 falshtime（停留时间）=T（某一固定数值）-level（或者 m）。如果 CurrentNum 和从按键模块输入的 ChosenNum 相等，则使用 break 语句退出显示部分循环，并令 Endflag 等于一。退出显示部分循环后定格显示 CurrentNum 一秒，然后令 level 自增一，再次为 CurrentNum 设置随机数，再次进入显示循环。如果显示循环结束后判断 m 等于六或者 level 等于二十结果为真，则复赋值 Endflag 为零。

如果第二步时判断 Endflag 为零，则停止闯关，在数码管上显示最终得分，并初始化 Startflag 为零，并关闭数码管模块。

## 五、调试过程

### 5.1. 随机数的生成方法

一开始采用自制随机数，即自己写了一个数组，数组内元素为自己随机写下的。关卡增加时自动选取下一个随机数。

但是这种随机数有很大的局限性，如果多玩几次游戏，其实玩家完全可以提前知晓下一个要出现的数字，从而影响游戏的公平性。

所以使用 rand() 函数，调用库 stdlib.h，并使用 rand()/7+1 语句将所有生成的值范围规定在 1~8 之间。但是这种情况下产生的随机数仍然是一个伪随机数，其出现的数依旧是固定顺序、可重复的。

故再引入一个函数 srand()。函数使用方法：void srand(unsigned int seed); 它需要一个 unsigned int 类型的参数。多数情况下可以用时间作为参数，只要每次播种的时间不同，那么生成的种子就不同，最终的随机数也就不同。但是在 Codewarrior 平台里 time(NULL) 语句所需的 time.h 头文件无法正常调用，故我们使用自制种子；定义一个 int 型变量 counter，每一次通过令 counter 自增一。

最终使用的用于生成随机数的程序中语句即：

```
srand(counter++); .....CurrentNum=(rand()%7)+1;
```

### 5.2. 逻辑错误

出错内容：如果连续两个关卡显示数字一样，当前一个关卡顺利通过之后，下一个关卡只在最右侧显示随机数后自动通过。

出错原因：ChosenNum 未及时清空。

改正方法：在成功通过判定后，增加语句 ChosenNum=0; 因为显示的数字的范围为 1~8，即 CurrentNum 取值为 1~8 的整数，所以 ChosenNum=0 时不会出现漏判。

## 六、实验结果

按下 SW5 (右侧板左上角按键) 游戏开始，数码管上显示英文大写字母：GO **【创新点①】**。



一随机数从右向左滚动显示，且越向左移动越快【创新点②】，按下对应按键后数字在原位置停留一秒。通关后一个新的数字从右向左滚动显示，且滚动速度比上一关更快。如果在数字滚动结束之前都没有按下对应按键，则闯关失败或者已经闯完所有二十关，数字停留显示一秒，然后数码管上显示总共通关得分【创新点③】。结束后如果想要再次玩游戏，则重复以上步骤。

结果符合预期，程序运行稳定。

#### 随机数测试：

测试数	每关数字	闯关结果	数码管显示
第一次	1 2 6 3 4 5 7 1 1 8 5 2 4 6 7 2 5 3 6 7	闯完所有二十关	20
第二次	8 1 4 2 5 6 4 7 (最后一个未通关)	未及时按下按键	07
第三次	2 2 3 1 6 3 8 4 2 6 5 3 4 7 1 (最后一个未通关)	未及时按下按键	14

### 七、实验分析（数据分析与讨论等）

实验结果符合预期，且三个创新点功能较为完备。总体运行良好。

实验中一个难点为随机数的生成，这里我尝试了多种方法，最终使用了 rand()配合 srand()函数的方法，所得的结果也比较好，生成的数较为随机，不会出现明显的重复。

实验中我设置了两个速度变化的点，玩游戏的时候感觉如果多玩几关会清楚地体会到速度在变快，游戏还是蛮刺激的。

设置的关卡为 20 关的选择是有理由的。20 关游戏人的疲惫感不会太高，且设置的所有设计 delay()函数的输入参数都不会小于零。

### 八、实验心得

本次实验较为全面地总结应用了本课程所学内容，包括中断、数码管显示等，通过本次实验我对之前所学有了一次大检测，察觉到了之前学习时一些不明了的知识点，并及时地查明。同时这次的实验也是对自己能力的一次综合考验，着其中对我而言最重要的就是不断精进、不断纠错并改正的过程。我在实验时遇见的一次次报错中，越来越了解查错、改错的方法，

比如设置断点、设置合适的全局变量以观测等。我感觉自己获益良多。

## 九、实验分工（每人具体负责的工作）

我们组共四人，每个人分别负责一个实验（project）的程序、ppt 及报告。

实验一 反应速度测试	汪雨甜 2018141411218
实验二 路口红绿灯管理模拟	赖梦婷 2018141442020
实验三 光照监控报警	朱琪霖 2018141241025
实验四 小游戏	许如玉 2018141442004