期末项目设计报告

1. 实验目的

1. 进一步掌握本学期以来所做实验用到的各种元器件的使用方法和编程；
2. 加深GPIO、中断、定时器、LED等的使用；
3. 加强综合问题解决能力和拓展思维创新能力。

2. 实现功能

1. 使用LED灯实现一个速度变的呼吸灯。
2. 通过不同按钮的组合可以调整呼吸灯速度和颜色。
3. 通过按钮进行不同的模式可以控制灯的颜色初定为为1000色，通过3色LED的每色10级亮度组合来实现颜色调整。
4. 通过串口可以手动输入上述灯的颜色参数。

3. 设计思路

- **PWM**：通过改变PWM波的占空比可以实现一个呼吸灯，通过改变占空比改电压从而变灯的亮度，进而实现呼吸灯效果。同时以PWM波改变占空比也可以改变灯的亮度，最终会反映到颜色组合上形成1000色RGB。
- **定时器**：使用定时器组合分频来模拟PWM波，虽然开发板上有PWM波的功能，为了符合实验要求使用中断分频，然后通过计数器模拟PWM波，当计数器到达设定值之后输出0，否则为1，计数器达到上限值时置0。如此我们可以获得一个可变占空比的PWM波。
- **中断**：
  - 定时器中引发的中断用于分频设置PWM波。
  - GPIO的中断设置为两个按钮的中断，之后通过GPIO的查询在中断中查询两个按钮中是哪个按下，这样就避免了按下按钮之后会一致循环判断按钮被按下。
- **串口**：在按钮选择到特定模式时会要求输入RGB三个灯的亮度参数从而实现亮度调节，同时在我们进行模式切换，或者改变当天模式的状态，比如呼吸灯速度时会打印出提示，此处使用串口助手进行连接，使用已经给出的串口代码实现输入输出。

4. 功能实现

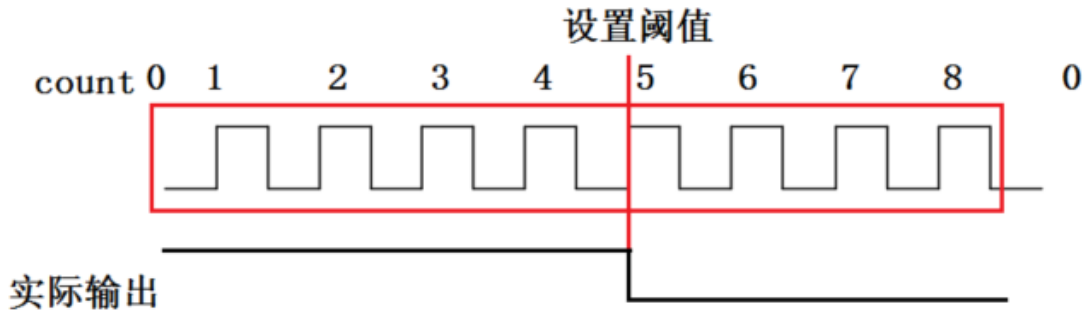
- **串口**

  串口的实现直接使用了UART_4C123中的串口模板，主要用于输入RGB的色阶，输出用于我们的模式切换的时候输出提示，此处的代码都是调用模板中的函数，不在展示。

- **呼吸灯以及1000色RGB的实现**

这两种功能都需要通过PWM波实现，此处PWM波的产生设计到中断，定时器和按钮中断，通过按钮调整阈值，然后结合计数器输出0，1。实现的关机在于PWM波的频率要高，这样肉眼不会看出明显明显变化，整个呼吸灯的呼吸过程也会流程。

- **设计原理**



原理如上图，通过定时器产生的中断进行计数，当计数到一定程度时(设置最大值)清零，当计数器小于阈值时我们使输出为1，否则输出为零。此时我们红框框起来的长度就是对应一个PWM波的周期，此时的实际输出通过阈值进行调整，我们可以得到变化占空比的PWM波。这样我们就可以进行呼吸灯和1000RGB的具体实现。

- **代码实现**

首先我们使用定时器进行分频操作，得到上图中的count计数器，此处设置其频率为200Khz,作为计数器其频率越高呼吸灯可变得级数越多越流畅，同时RGB颜色也会越多。

```
unsigned short Counts;
void SysTick_Init(uint32_t period){
  Counts = 0;
  NVIC_ST_CTRL_R = 0;            // disable SysTick during setup
  NVIC_ST_RELOAD_R = period - 1; // reload value
  NVIC_ST_CURRENT_R = 0;         // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x60000000; //priority 3

  NVIC_ST_CTRL_R = 0x00000007;   // enable with core clock and interrupts
}
void SysTick_Handler(void){         // Div F to Breathe base frequency
  Counts++;
    if(Counts>128){
        Counts=0;
    }
}

SysTick_Init(F200KHZ);      //initialize SysTick (200K Hz)
```

然后我们用timer0A实现呼吸灯调速和改变阈值，在Timer0A每一次触发中断后阈值都会上升，当阈值上升到一定程度后开始每一次中断阈值减少。中断函数如下：

```
void UserTask(void){    //Div F to Breathe speed (change Duty cycle)
  if(flag)Count_speed+=speed; //speed change
    else Count_speed-=speed;
    if(Count_speed > 127)flag=0;//change on->off
    if(Count_speed==0)flag=1;   // off->on
}

Timer0A_Init(&UserTask, F50HZ);  // initialize timer0A (50 Hz)
```

为了使得呼吸灯变化一次的速度不会太快，设置初始化频率为50hz。代码中的speed通过按键控制。

之后实现10级1000色RGB，实现方式也是相同，类似呼吸灯，通过改变占空比实现颜色变化，这里为了防止和原呼吸灯冲突，使用了Timer3A，作为RGB功能的COUNT，然后通过按钮改变阈值进行设置占空比。

```
unsigned short Count_RGB;
void UserTask1(void){                       //Div F to RGB
  Count_RGB++;
    if(Count_RGB>9){
        Count_RGB=0;
    }
}
Timer3A_Init(&UserTask1, F1KHZ);
```

其中Timer3A和0A的初始化函数不同的就是在SYSCTL_RCGCTIMER_R、NVIC_EN1_R、和优先级上，其他的都是一些常规的改变

```
SYSCTL_RCGCTIMER_R |= 0x08;   // 0) activate TIMER3
……
NVIC_PRI8_R = (NVIC_PRI8_R&0x00FFFFFF)|0x80000000; // 8) priority 4
NVIC_EN1_R = 1<<(35-32);      // 9) enable IRQ 35 in NVIC
```

- **按钮中断及查询**

  按钮的中断和查询项目中把他结合在一起了，因为这样可以省去一个做防抖的计时器，通过按钮改变显示的LED显示的方式，和进行模式测切换。

  功能表如下

模式	SW1	SW2
0	进入模式1	控制呼吸灯的速度
1	如果颜色为无色（即全灭）进入模式2用户自定义颜色，否则返回模式1	切换呼吸灯的颜色
2	进入模式3	控制红灯的亮度
3	进入模式4	控制绿灯的亮度
4	进入模式5	控制蓝灯的亮度
5	无法触发	无法触发

此处为了减少使用防抖我们为两个按钮全部加上中断，中断之后查询是哪个按钮按下，根据不同的按钮进行功能的变化，此处模式5是输入模式，进入输入模式之后等待输入完成会自动跳回模式0。

初始化代码如下，此处需要解锁同时注册两个按钮的中断,这里要保证按钮按下时会优先处理所以设置优先级为1。

```
volatile uint32_t speed = 1;
void EdgeCounter_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x00000020; // (a) activate clock for port F
  speed = 1;             // (b) initialize counter
    model=0;
    GPIO_PORTF_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port F
  GPIO_PORTF_CR_R = 0x1F;           // allow changes to PF4-0
  GPIO_PORTF_DIR_R = 0x0E;    // (c) make PF4 in (built-in button)
  GPIO_PORTF_AFSEL_R &= ~0x11;  //     disable alt funct on PF4 PF1
  GPIO_PORTF_DEN_R |= 0x1F;     //     enable digital I/O on PF4-1
  GPIO_PORTF_PCTL_R = 0x00000000; // configure PF4-1 as GPIO
  GPIO_PORTF_AMSEL_R = 0;       //     disable analog functionality on PF
  GPIO_PORTF_PUR_R |= 0x11;     //     enable weak pull-up on PF4 PF1
  GPIO_PORTF_IS_R &= ~0x11;     // (d) PF4 PF1 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x11;    //     PF4 PF1 is not both edges
  GPIO_PORTF_IEV_R &= ~0x11;    //     PF4 PF1 falling edge event
  GPIO_PORTF_ICR_R = 0x11;      // (e) clear flag4 1
  GPIO_PORTF_IM_R |= 0x11;      // (f) arm interrupt on PF4 *** No IME bit as mentioned in
Book ***
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF00FFFF)|0x00200000; // (g) priority 1
  NVIC_EN0_R = 0x40000000;      // (h) enable interrupt 30 in NVIC
  EnableInterrupts();           // (i) Clears the I bit
}
```

中断函数如下，我们在函数开头就要使用查询来获取哪个按钮被按下，根据不同按钮和当前的模式进行不同的处理，同时调用串口输出提示。此时SW2按下的主要功能都是在修改阈值。

```
void GPIOPortF_Handler(void){       //btn Interrupts
  GPIO_PORTF_ICR_R = 0x11;      // acknowledge flag4
    int btn=GPIO_PORTF_DATA_R & 0x11;  //get btn
    if(btn == 0x01){                            //PF4
        if(model!=5)model++;
```

```
        if(color!=7&&model>1)model=0;
        if(model==0){UART_OutString("model = 0 SW2 to  Change breathe speed");OutCRLF();}
        else if(model==1){UART_OutString("model = 1 SW2 to Change color");OutCRLF();}
        else if(model==2){UART_OutString("model = 2 SW2 to Change R");OutCRLF();}
        else if(model==3){UART_OutString("model = 3 SW2 to Change G");OutCRLF();}
        else if(model==4){UART_OutString("model = 4 SW2 to Change B");OutCRLF();}
        else if(model==5){UART_OutString("model = 5 Input R G B !");OutCRLF();}
    }
    else if(btn == 0x10){                   //PF1
        if(model == 0){                         //model 1  add speed
            speed = speed * 2;              //add speed
            if(speed>127)speed=1;           //limit
            Count_speed=0;                                  //clear count
            flag=1;                                     //clear status
            UART_OutString("Speed="); UART_OutUDec(speed); OutCRLF();
        }
        else if(model==1){                  //change color
            color++;
            if(color>7)color=0;
            if(color==7){UART_OutString("customize RGB! use SW1 to change R-G-B SW2 to
change bright");OutCRLF();}
        }else if (model==2){                //RGB  R 0-10
            colorR++;
            if(colorR>10)colorR=0;
            UART_OutString("set R="); UART_OutUDec(colorR); OutCRLF();
        }else if(model==3){                 //RGB  G 0-10
            colorG++;
            if(colorG>10)colorG=0;
            UART_OutString("set G="); UART_OutUDec(colorG); OutCRLF();
        }else if(model==4){                 //RGB  B 0-10
            colorB++;
            if(colorB>10)colorB=0;
            UART_OutString("set B="); UART_OutUDec(colorB); OutCRLF();
        }
    }
}
```

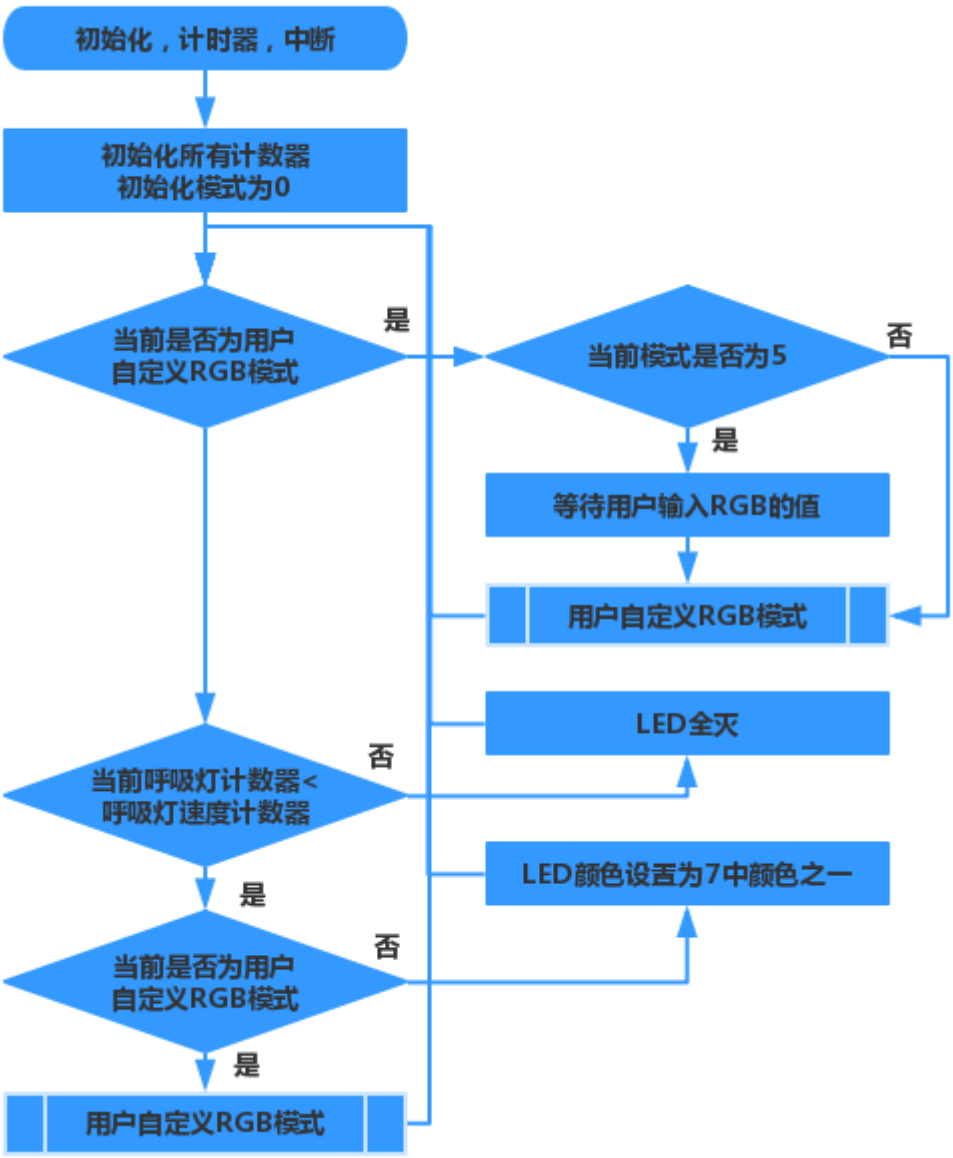
- 主函数部分，主函数主要功能为通过计数器和阈值判断当前的LED该亮或者灭，同时在输入模式时负责处理输入。主函数中初始化不在展示。在模式为5的时候需要读取输入，等待输入完毕之后自动返回模式0。在我们设置自定义RGB的时候呼吸灯的功能是关闭的，当我们设置完成后返回模式0呼吸灯也会变成我们设定的颜色。

```
UART_OutString("model = 0 SW1 to Model SW2 to speed");OutCRLF();
  while(1){
    WaitForInterrupt();
            if(color==7 && model > 1 ){// User set RGB
                    if(Count_RGB<colorR)PF1|=0x02; //set R light
                    else PF1 &= ~0x02;
                    if(Count_RGB<colorB)PF2|=0x04;//set B light
                    else PF2 &= ~0x04;
                    if(Count_RGB<colorG)PF3|=0x08;//set G light
                    else PF3 &= ~0x08;
```

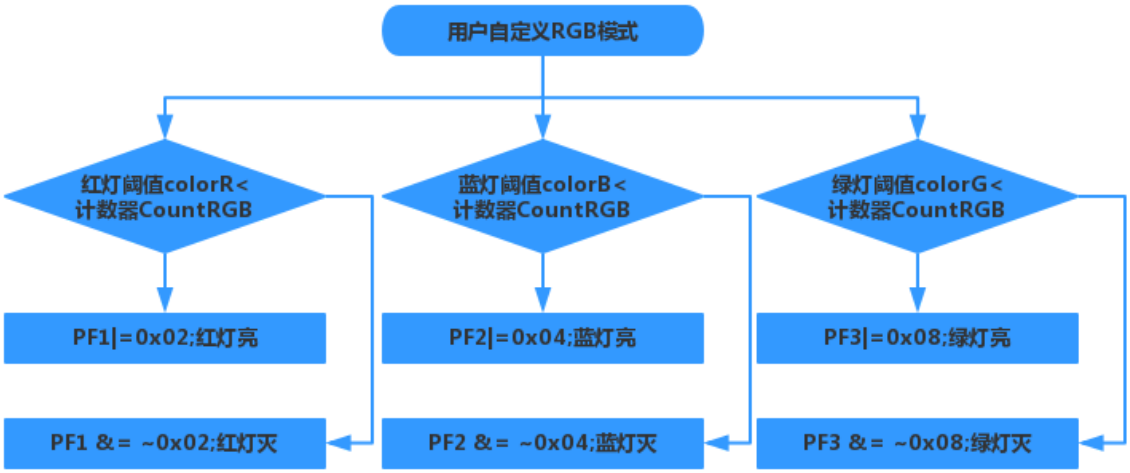
```
                    if(model == 5){ //input RGB
                        UART_OutString("Input R(0-10): ");
                        colorR=UART_InUDec(); OutCRLF();
                        if(colorR>10)colorR=10;
                        UART_OutString("R = ");
                        UART_OutUDec(colorR); OutCRLF();
                        UART_OutString("Input G(0-10): ");
                        colorG=UART_InUDec(); OutCRLF();
                        if(colorG>10)colorG=10;
                        UART_OutString("G = ");
                        UART_OutUDec(colorG); OutCRLF();
                        UART_OutString("Input B(0-10): ");
                        colorB=UART_InUDec(); OutCRLF();
                        if(colorB>10)colorB=10;
                        UART_OutString("B = ");
                        UART_OutUDec(colorB); OutCRLF();
                        UART_OutString("RGB set complete model = 0 ");
                        OutCRLF();
                        model=0;
                    }
            }
            else{
                if(Counts<Count_speed){ //Breathe
                    if(color==7){// user RGB
                        if(Count_RGB<colorR)PF1|=0x02;
                        else PF1 &= ~0x02;
                        if(Count_RGB<colorB)PF2|=0x04;
                        else PF2 &= ~0x04;
                        if(Count_RGB<colorG)PF3|=0x08;
                        else PF3 &= ~0x08;
                    }
                    else LEDS=COLORWHEEL[color]; //change color
                }else{
                    LEDS=0x00;
                }
        }
    }
```

5. 程序流程

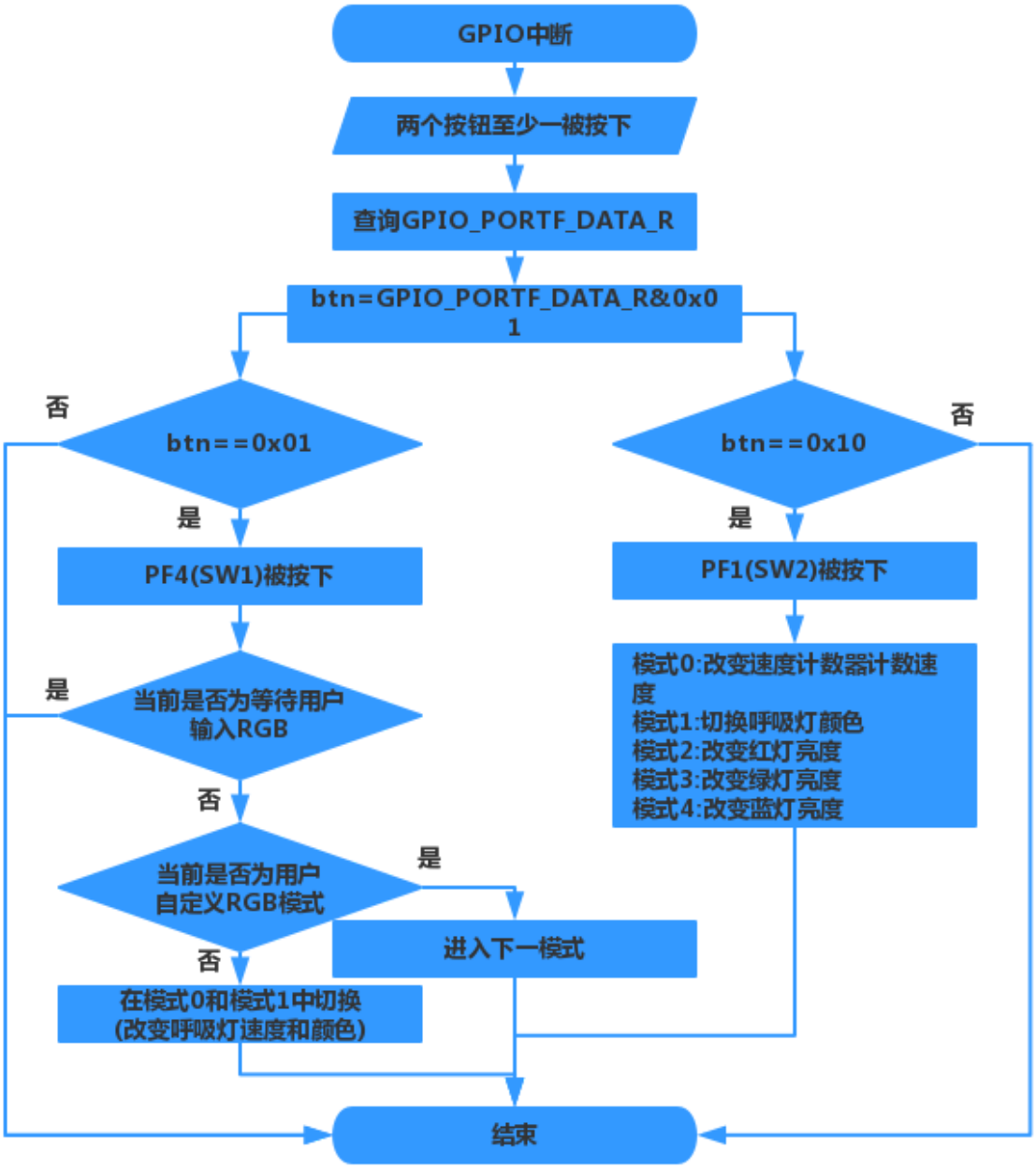
- **总流程**


初始化，计时器，中断
初始化所有计数器
初始化模式为0
当前是否为用户
自定义RGB模式
是
当前模式是否为5
否
是
等待用户输入RGB的值
用户自定义RGB模式
LED全灭
当前呼吸灯计数器<
呼吸灯速度计数器
否
LED颜色设置为7中颜色之一
是
当前是否为用户
自定义RGB模式
否
是
用户自定义RGB模式

- **自定义RGB模式流程**

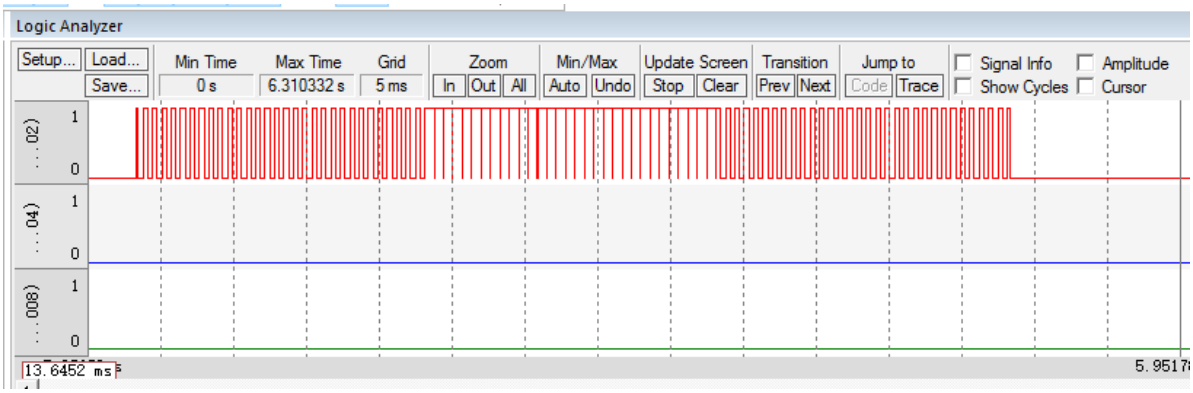

用户自定义RGB模式
红灯阈值colorR<
计数器CountRGB
蓝灯阈值colorB<
计数器CountRGB
绿灯阈值colorG<
计数器CountRGB
PF1|=0x02;红灯亮
PF2|=0x04;蓝灯亮
PF3|=0x08;绿灯亮
PF1 &= ~0x02;红灯灭
PF2 &= ~0x04;蓝灯灭
PF3 &= ~0x08;绿灯灭

- **GPIO中断流程**



6. 运行效果

- 呼吸灯及颜色切换效果
  - 仿真

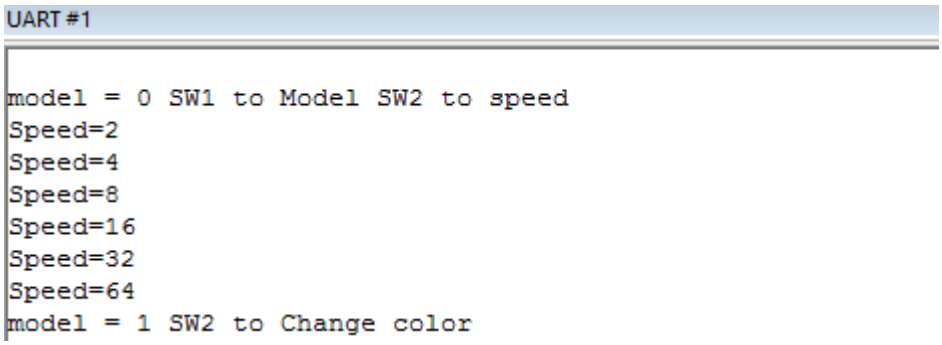


此处我们可以看到此时的PF2的波形为变占空比的方波PWM。

我们更设置改变速度并切换颜色之后

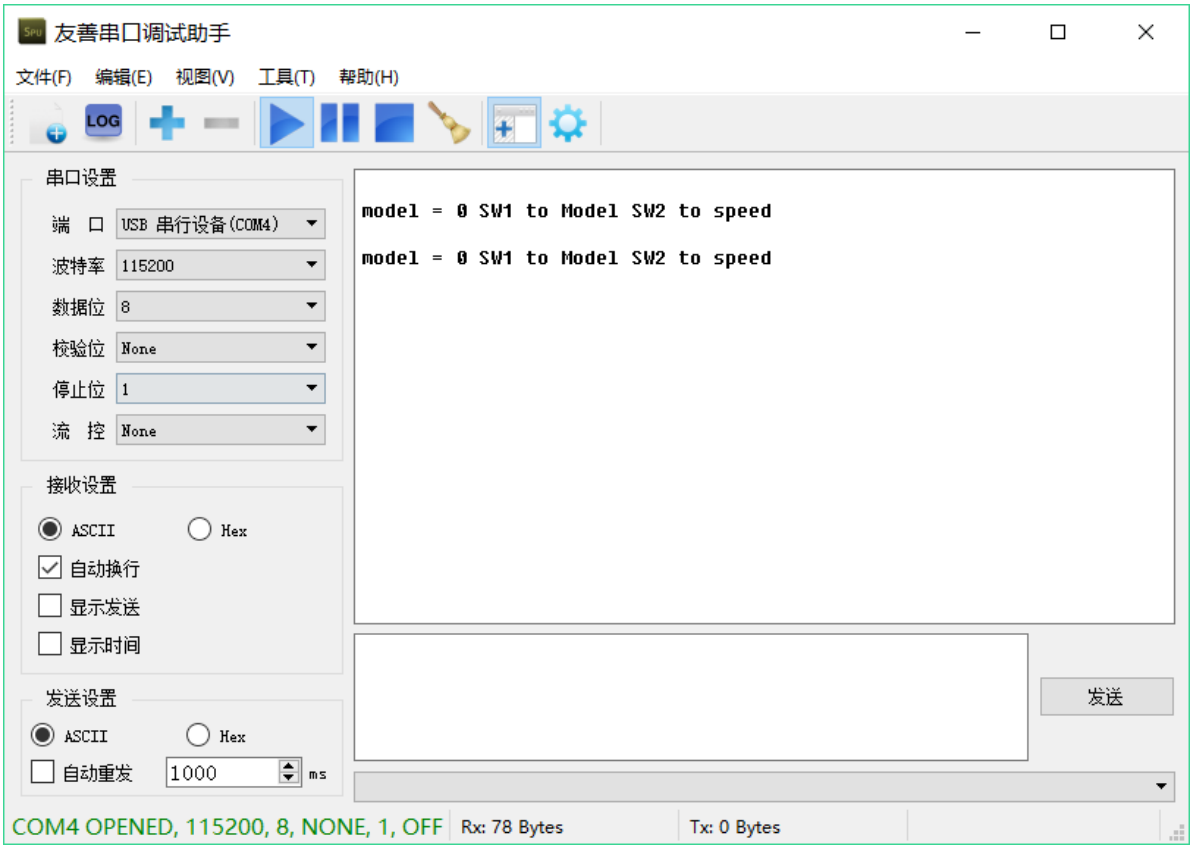
此时PWM变化速度减慢，此时绿灯也同步亮起，形成黄灯

我们看到串口提示如下：



- **板级运行**

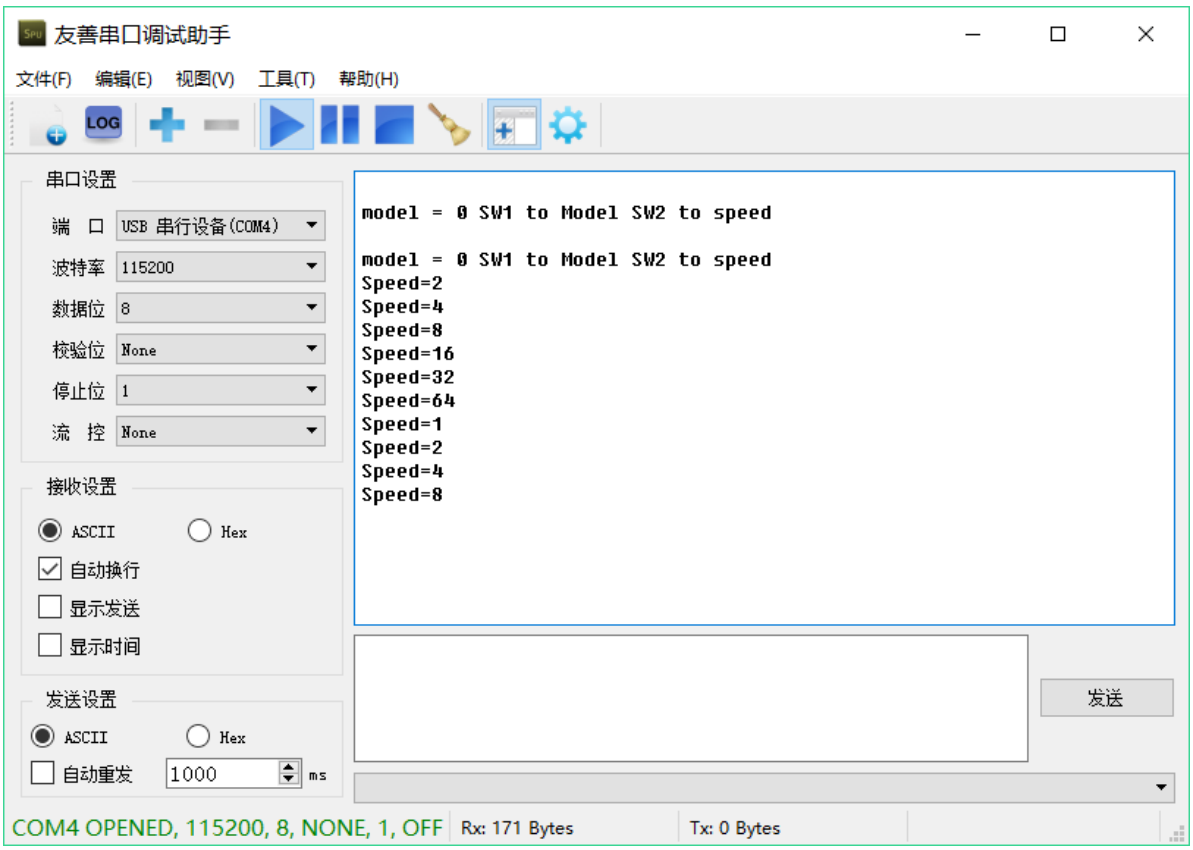
  运行时我们连接上串口调试助手，将波特率修改为115200



开始运行之后我们获得以下代码输出的信息

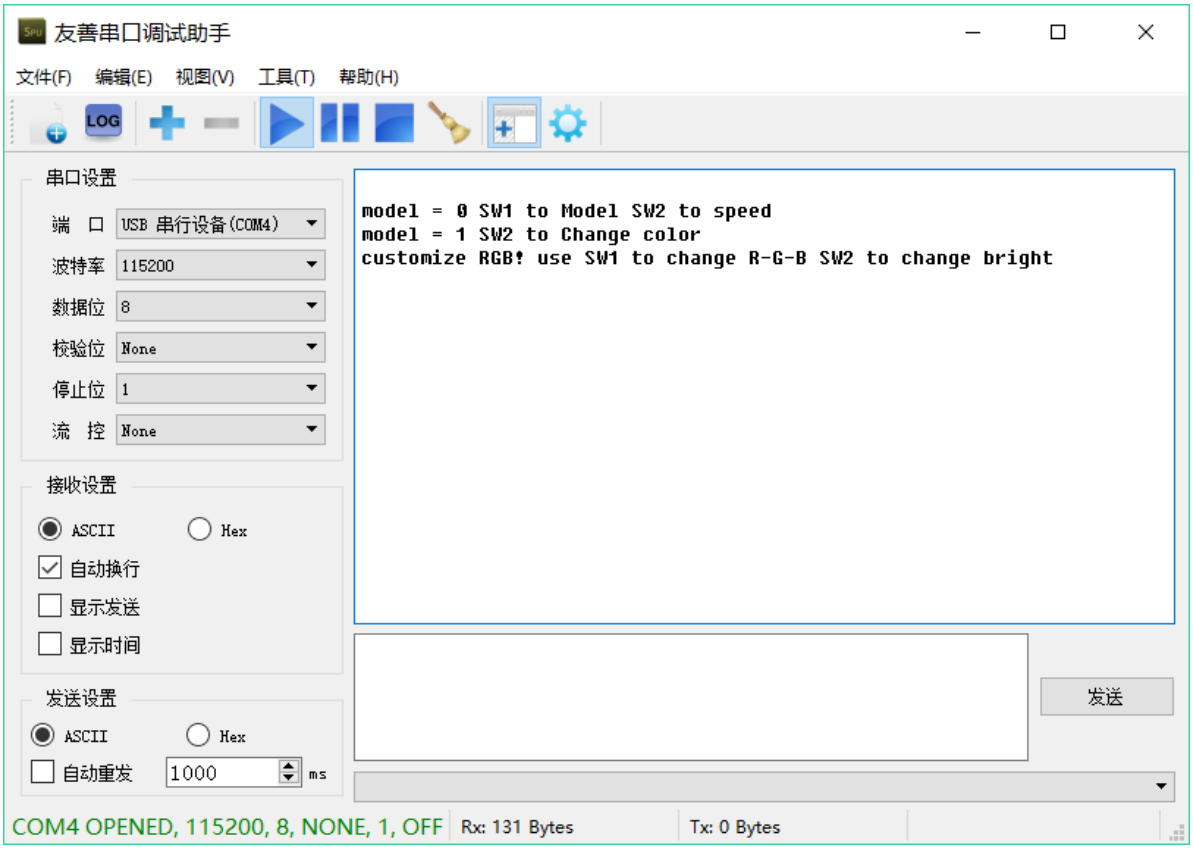
```
UART_OutString("model = 0 SW1 to Model SW2 to speed");OutCRLF();
```

此时我们调整呼吸灯速度



串口会输出当前的速度。

此时我们进入调色模式调整颜色

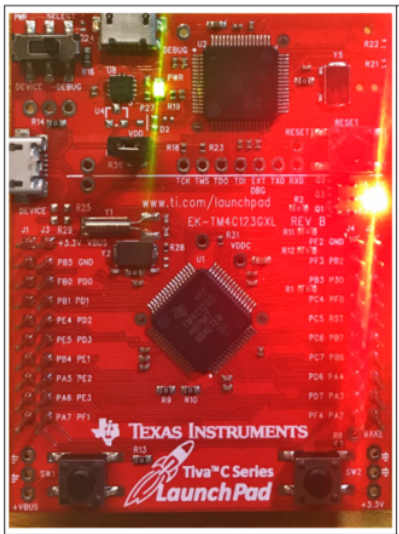
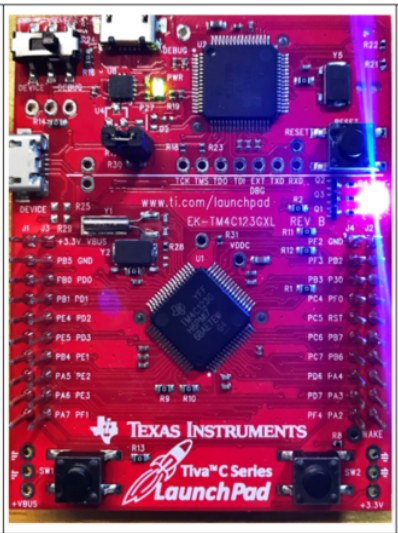
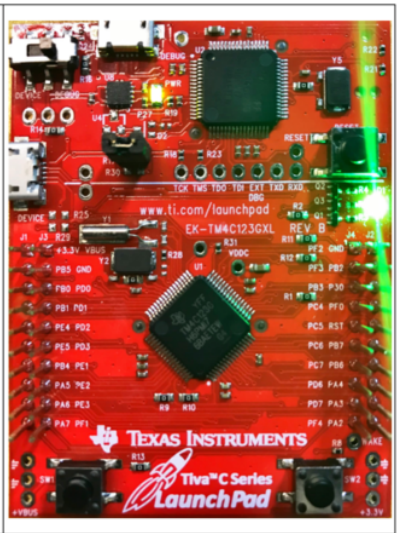


此处当我们循环完设定的7中颜色到无色的时候回提示要进入设置自定义RGB。

此处为所有固定颜色。

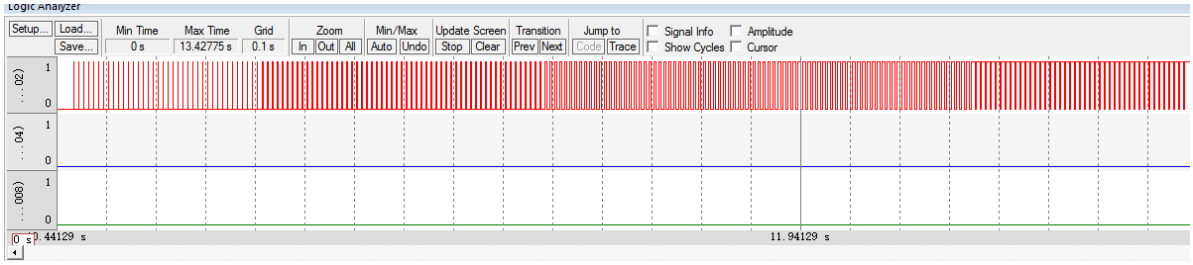
```
const long COLORWHEEL[WHEELSIZE] = {RED, RED+GREEN, GREEN, GREEN+BLUE, BLUE, BLUE+RED,
RED+GREEN+BLUE, 0};
```

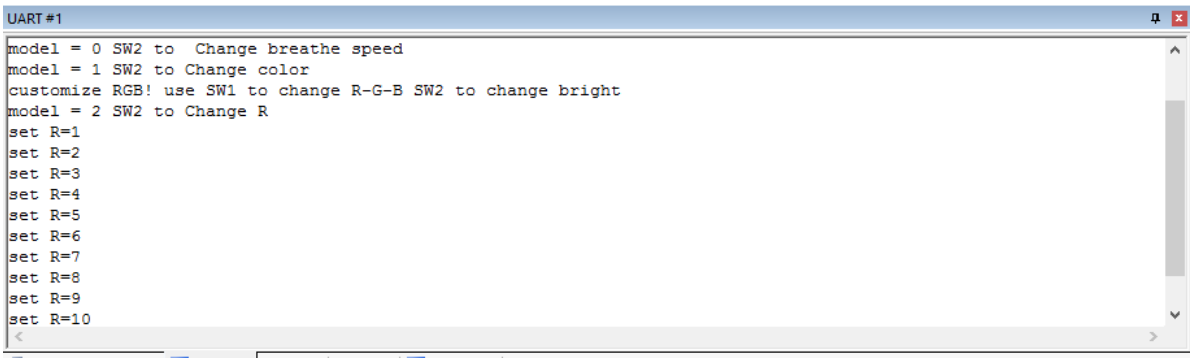
此处就先只展示3种颜色，其余的就不展示了，呼吸灯的呼吸效果这照片无法展示，自行脑补，或者见上步仿真波形。

		
红	蓝	绿

- **用户自定义1000色RGB**
  - **仿真**

    首先是模式2-4对应的按钮控制亮度。

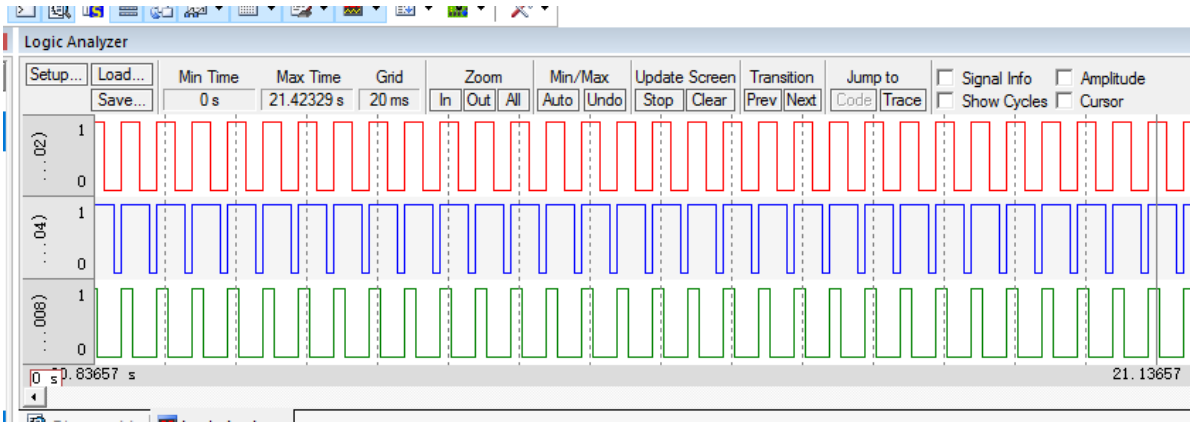




此时我们可以看到随着我们的按钮此时红灯的占空比不断变化，熄灭面到全亮状态。

其他两种颜色的灯变化情况也是类似。

然后仿真三种灯的颜色组合

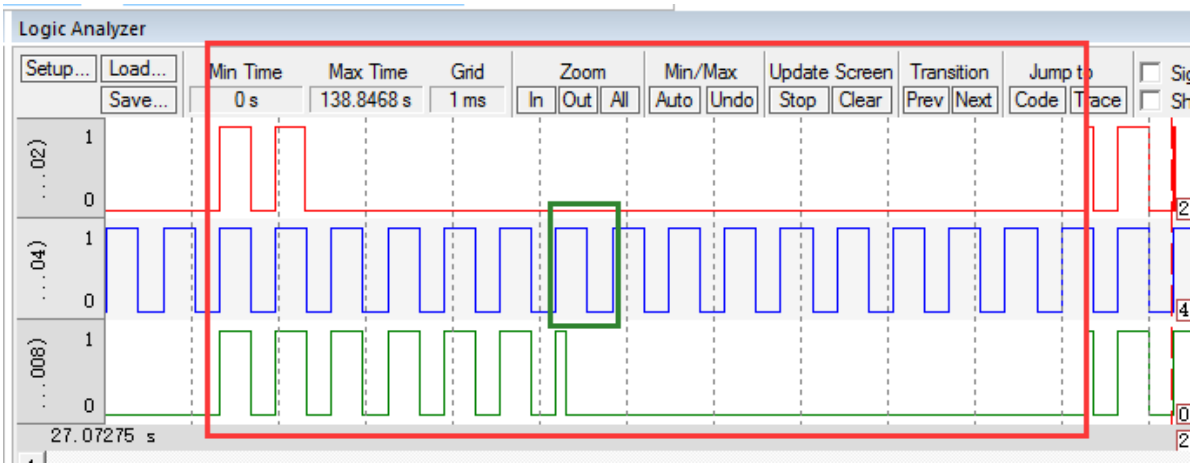


```
UART #1
set R=8
set R=9
set R=10
set R=0
set R=1
set R=2
set R=3
set R=4
set R=5
model = 3 SW2 to Change G
set G=1
set G=2
set G=3
model = 4 SW2 to Change B
set B=1
set B=2
set B=3
set B=4
set B=5
set B=6
set B=7
set B=8
```

此时我们三个LED灯有不同的占空比，所以亮度不同，组合出一种新的颜色。

之后仿真模式5的输入自定义RGB

```
UART #1
set B=8
model = 5 Input R G B !
Input R(0-10): 1
R = 1
Input G(0-10): 4
G = 4
Input B(0-10): 10
B = 10
RGB set complete model = 0
Speed=2
```

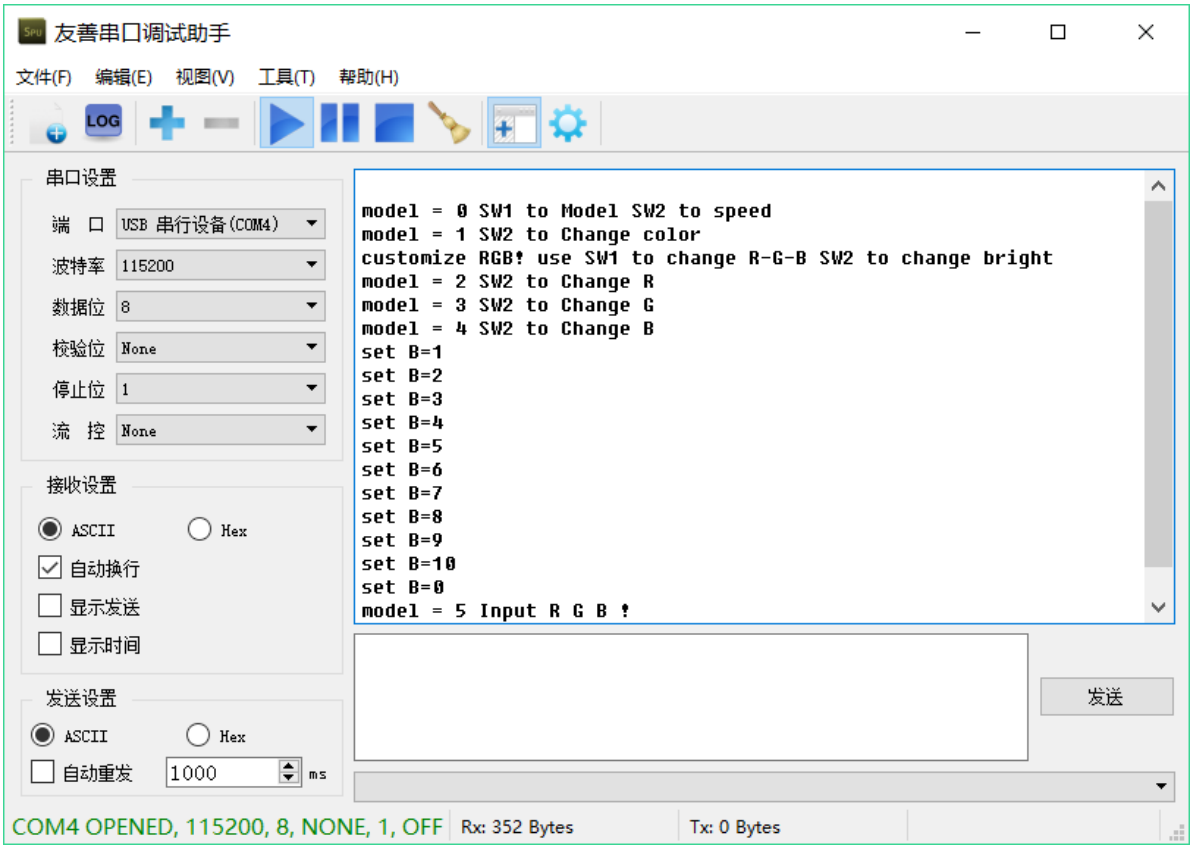


此时设定完颜色之后进入呼吸灯模式，此时rgb的PWM和呼吸灯的PWM就出现了重叠，当且他们都为1才会输出1，因为呼吸灯的计数器频率比rgb计数器频率更高，此时绿圈中为一个呼吸灯的计数器周期，红圈为rgb计数器周期。

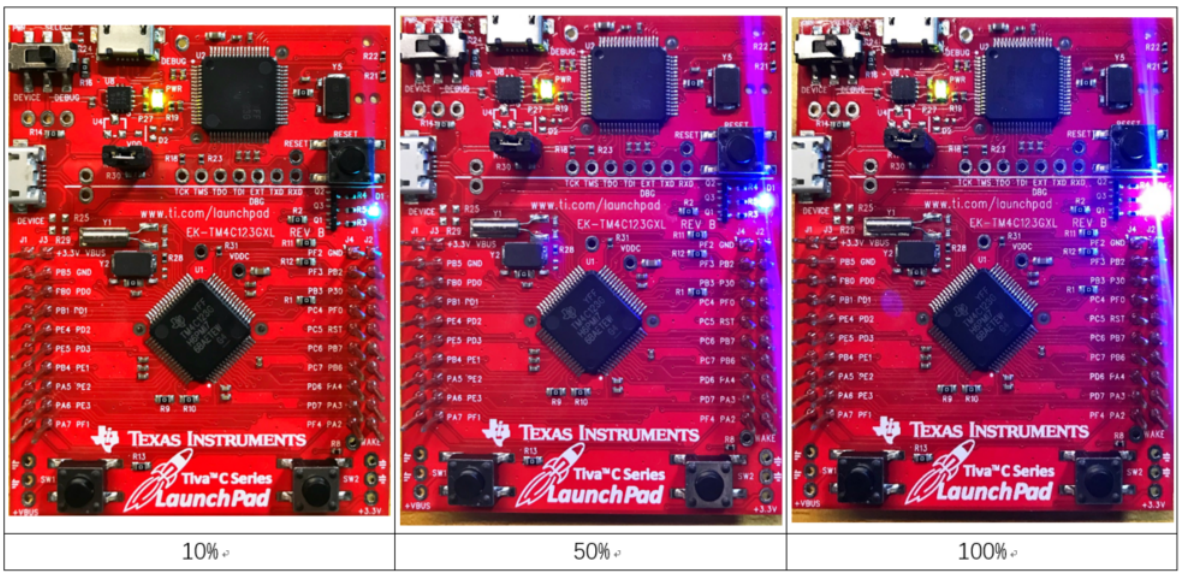
- **板级运行**

  此处运行效果展示一下10级灯的不同亮度，和一些照片能显示出来的颜色。

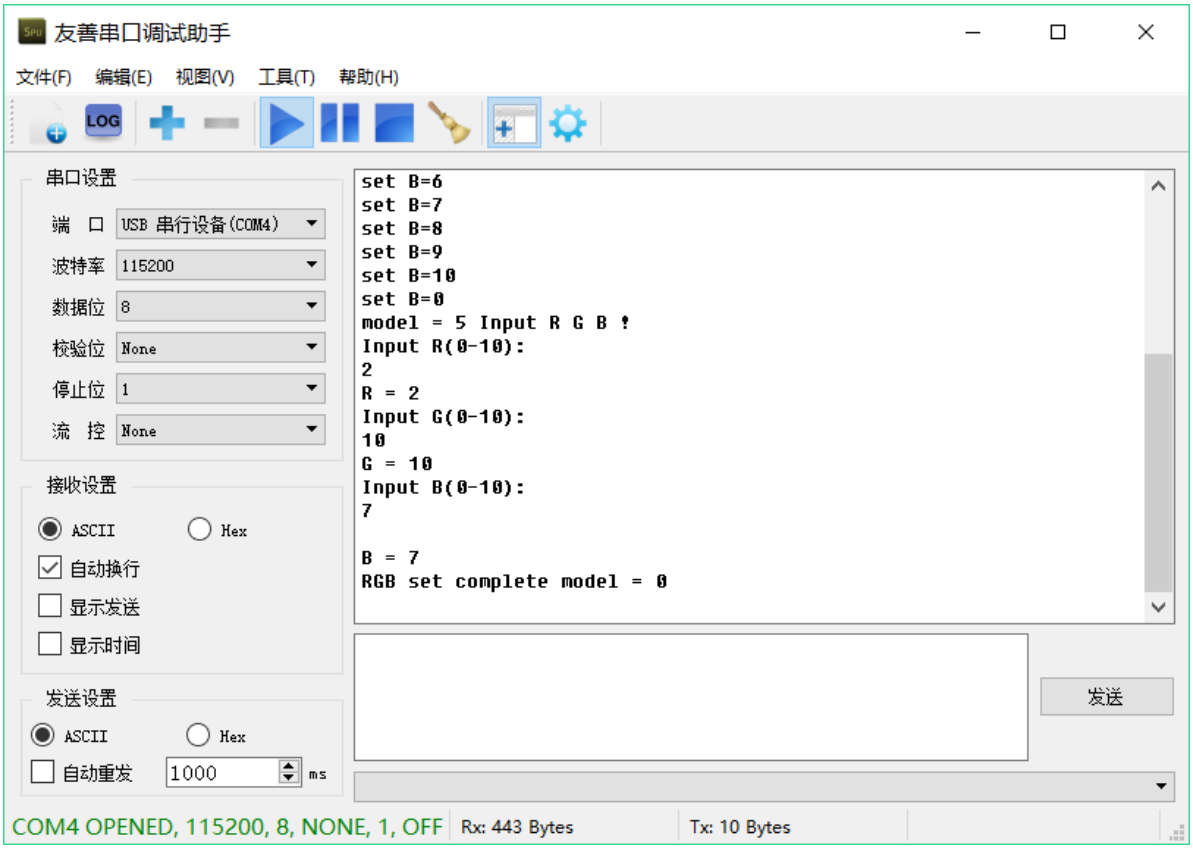
  此处我们设置蓝灯亮度，从0-10都会有提示。

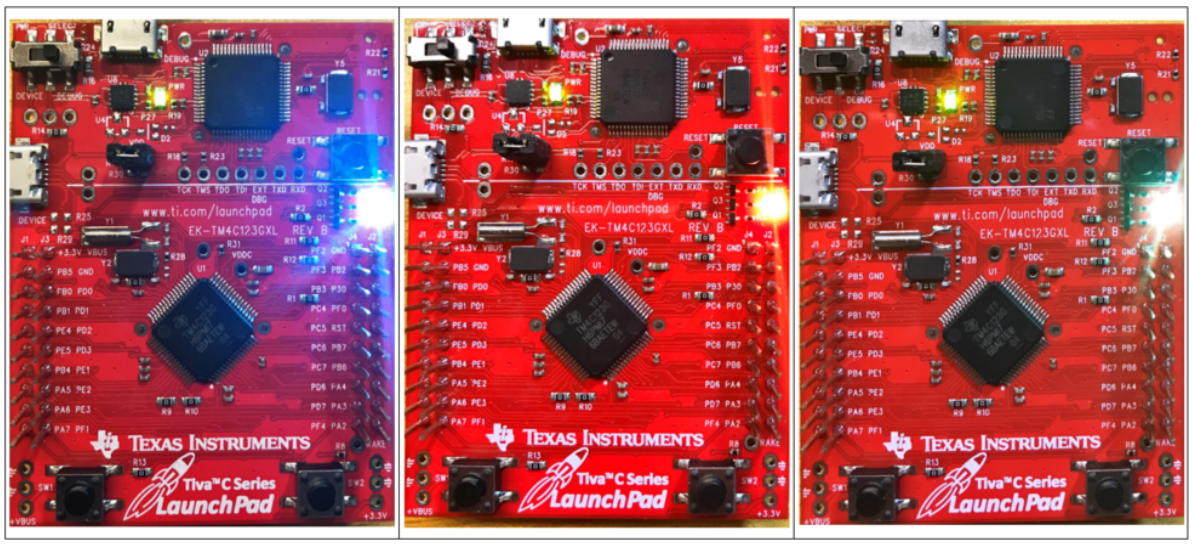


展示蓝灯亮度为1,5,10的效果



展示输入效果





这个功能使用照片不是很好的能够展示。

7. 实验体会

本次实验为开放式性题目，主要就是检验我们半学期所学到的内容，实验中一开始想实现呼吸灯功能的时候，看了一下板子的说明书，发现其是PWM驱动的，可以直接使用方法也很简单，也有给出对应的例程，但是后来考虑到需要用到这学期所学的内容，于是放弃了使用PWM驱动的方法，转而使用中断模拟的方法进行设计，通过阈值加计数器的方式实现PWM，这个其实也不难，主要就是使用计数器和控制阈值，同时因为PWM波调节占空比就是在调节电压，这是以前学过的内容。至于中断这个，首先是使用中断分频做计数器，之后是GPIO中断，这里结合了查询，两个按钮都可以触发中断，然后在中断中查询，主要是一开始想着这样可以省去做防抖的时间，但是实际效果并不是很好，依然会出现抖动的问题。串口的使用用于输出提示已经输入数字设定灯的亮度，这个是直接仿照例程，直接调用里面的函数进行输入输出，使用没什么困难。

附录

主代码

```
#include "..//inc//tm4c123gh6pm.h"
#include <stdint.h>
#include "PLL.h"
#include "Timer0A.h"
#include "Timer3A.h"
#include "UART.h"

#define PF0             (*((volatile uint32_t *)0x40025004))
#define PF1       (*((volatile uint32_t *)0x40025008))
#define PF2       (*((volatile uint32_t *)0x40025010))
#define PF3       (*((volatile uint32_t *)0x40025020))
#define PF4       (*((volatile uint32_t *)0x40025040))
#define LEDS      (*((volatile uint32_t *)0x40025038))
#define RED       0x02
#define BLUE      0x04
#define GREEN     0x08
#define WHEELSIZE 8           // must be an integer multiple of 2
                              //    red, yellow,    green, light blue, blue, purple,   white,
       dark
const long COLORWHEEL[WHEELSIZE] = {RED, RED+GREEN, GREEN, GREEN+BLUE, BLUE, BLUE+RED,
RED+GREEN+BLUE, 0};


void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode
unsigned short Count_speed=0;
unsigned short flag=1;
unsigned short model=0;
unsigned short color=0;
unsigned short colorR=0;
unsigned short colorG=0;
unsigned short colorB=0;

//---------------------OutCRLF---------------------
// Output a CR,LF to UART to go to a new line
// Input: none
// Output: none
void OutCRLF(void){
  UART_OutChar(CR);
  UART_OutChar(LF);
```

```
}

volatile uint32_t speed = 1;
void EdgeCounter_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x00000020; // (a) activate clock for port F
  speed = 1;              // (b) initialize counter
    model=0;
    GPIO_PORTF_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port F
  GPIO_PORTF_CR_R = 0x1F;           // allow changes to PF4-0
  GPIO_PORTF_DIR_R = 0x0E;    // (c) make PF4 in (built-in button)
  GPIO_PORTF_AFSEL_R &= ~0x11;  //     disable alt funct on PF4 PF1
  GPIO_PORTF_DEN_R |= 0x1F;     //     enable digital I/O on PF4-1
  GPIO_PORTF_PCTL_R = 0x00000000; // configure PF4-1 as GPIO
  GPIO_PORTF_AMSEL_R = 0;       //     disable analog functionality on PF
  GPIO_PORTF_PUR_R |= 0x11;     //     enable weak pull-up on PF4 PF1
  GPIO_PORTF_IS_R &= ~0x11;     // (d) PF4 PF1 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x11;    //     PF4 PF1 is not both edges
  GPIO_PORTF_IEV_R &= ~0x11;    //     PF4 PF1 falling edge event
  GPIO_PORTF_ICR_R = 0x11;      // (e) clear flag4 1
  GPIO_PORTF_IM_R |= 0x11;      // (f) arm interrupt on PF4 *** No IME bit as mentioned in Book
***
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF00FFFF)|0x00200000; // (g) priority 1
  NVIC_EN0_R = 0x40000000;      // (h) enable interrupt 30 in NVIC
  EnableInterrupts();           // (i) Clears the I bit
}
void GPIOPortF_Handler(void){       //btn Interrupts
  GPIO_PORTF_ICR_R = 0x11;      // acknowledge flag4
    int btn=GPIO_PORTF_DATA_R & 0x11;  //get btn
    if(btn == 0x01){                            //PF4
        if(model!=5)model++;
        if(color!=7&&model>1)model=0;
        if(model==0){UART_OutString("model = 0 SW2 to  Change breathe speed");OutCRLF();}
        else if(model==1){UART_OutString("model = 1 SW2 to Change color");OutCRLF();}
        else if(model==2){UART_OutString("model = 2 SW2 to Change R");OutCRLF();}
        else if(model==3){UART_OutString("model = 3 SW2 to Change G");OutCRLF();}
        else if(model==4){UART_OutString("model = 4 SW2 to Change B");OutCRLF();}
        else if(model==5){UART_OutString("model = 5 Input R G B !");OutCRLF();}
    }
    else if(btn == 0x10){                   //PF1
        if(model == 0){                         //model 1  add speed
            speed = speed * 2;              //add speed
            if(speed>127)speed=1;           //limit
            Count_speed=0;                                  //clear count
            flag=1;                                     //clear status
            UART_OutString("Speed="); UART_OutUDec(speed); OutCRLF();
        }
        else if(model==1){                  //change color
            color++;
            if(color>7)color=0;
            if(color==7){UART_OutString("customize RGB! use SW1 to change R-G-B SW2 to change
bright");OutCRLF();}
        }else if (model==2){                //RGB  R 0-10
            colorR++;
```

```
            if(colorR>10)colorR=0;
            UART_OutString("set R="); UART_OutUDec(colorR); OutCRLF();
        }else if(model==3){                 //RGB  G 0-10
            colorG++;
            if(colorG>10)colorG=0;
            UART_OutString("set G="); UART_OutUDec(colorG); OutCRLF();
        }else if(model==4){                 //RGB  B 0-10
            colorB++;
            if(colorB>10)colorB=0;
            UART_OutString("set B="); UART_OutUDec(colorB); OutCRLF();
        }
    }
}
void UserTask(void){                        //Div F to Breathe speed (change Duty cycle)
  if(flag)Count_speed+=speed;       //speed change
    else Count_speed-=speed;
    if(Count_speed > 127)flag=0;    //change on->off
    if(Count_speed==0)flag=1;           // off->on
}

unsigned short Count_RGB;
void UserTask1(void){                       //Div F to RGB
  Count_RGB++;
    if(Count_RGB>9){
        Count_RGB=0;
    }
}

unsigned short Counts;
void SysTick_Init(uint32_t period){
  Counts = 0;
  NVIC_ST_CTRL_R = 0;            // disable SysTick during setup
  NVIC_ST_RELOAD_R = period - 1; // reload value
  NVIC_ST_CURRENT_R = 0;         // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x60000000; //priority 3
  NVIC_ST_CTRL_R = 0x00000007;   // enable with core clock and interrupts
}
void SysTick_Handler(void){         // Div F to Breathe base frequency
  Counts++;
    if(Counts>128){
        Counts=0;
    }
}

// if desired interrupt frequency is f, Timer0A_Init parameter is busfrequency/f
#define F50HZ (50000000/50)
#define F1KHZ (50000000/1000)
#define F200KHZ (50000000/200000)


int main(void){
  PLL_Init();                       // bus clock at 50 MHz
    EdgeCounter_Init();
```

```
  Timer0A_Init(&UserTask, F50HZ);  // initialize timer0A (50 Hz)
    Timer3A_Init(&UserTask1, F1KHZ);    //initialize timer3A (1K Hz)
    SysTick_Init(F200KHZ);                    //initialize SysTick (200K Hz)
    UART_Init();                              // initialize UART
  OutCRLF();
  EnableInterrupts();
    UART_OutString("model = 0 SW1 to Model SW2 to speed");OutCRLF();
  while(1){
    WaitForInterrupt();
            if(color==7 && model > 1 ){// User set RGB
                    if(Count_RGB<colorR)PF1|=0x02; //set R light
                    else PF1 &= ~0x02;
                    if(Count_RGB<colorB)PF2|=0x04;//set B light
                    else PF2 &= ~0x04;
                    if(Count_RGB<colorG)PF3|=0x08;//set G light
                    else PF3 &= ~0x08;
                    if(model == 5){ //input RGB
                        UART_OutString("Input R(0-10): ");
                        colorR=UART_InUDec(); OutCRLF();
                        if(colorR>10)colorR=10;
                        UART_OutString("R = ");
                        UART_OutUDec(colorR); OutCRLF();
                        UART_OutString("Input G(0-10): ");
                        colorG=UART_InUDec(); OutCRLF();
                        if(colorG>10)colorG=10;
                        UART_OutString("G = ");
                        UART_OutUDec(colorG); OutCRLF();
                        UART_OutString("Input B(0-10): ");
                        colorB=UART_InUDec(); OutCRLF();
                        if(colorB>10)colorB=10;
                        UART_OutString("B = ");
                        UART_OutUDec(colorB); OutCRLF();
                        UART_OutString("RGB set complete model = 0 ");
                        OutCRLF();
                        model=0;
                    }
            }
            else{
                if(Counts<Count_speed){ //Breathe
                    if(color==7){// user RGB
                        if(Count_RGB<colorR)PF1|=0x02;
                        else PF1 &= ~0x02;
                        if(Count_RGB<colorB)PF2|=0x04;
                        else PF2 &= ~0x04;
                        if(Count_RGB<colorG)PF3|=0x08;
                        else PF3 &= ~0x08;
                    }
                    else LEDS=COLORWHEEL[color]; //change color
                }else{
                    LEDS=0x00;
                }
        }
  }
```

```
}
```

新增的Timer3A

Timer3A.h

```
#ifndef __TIMER3AINTS_H__ // do not include more than once
#define __TIMER3AINTS_H__
void Timer3A_Init(void(*task)(void), uint32_t period);
void Timer3A_Set_Period(uint32_t period);
```

Timer3A.c

```
#include <stdint.h>
#include "..//inc//tm4c123gh6pm.h"

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode
void (*PeriodicTask1)(void);   // user function

void Timer3A_Init(void(*task)(void), unsigned long period){long sr;
  sr = StartCritical();
  SYSCTL_RCGCTIMER_R |= 0x08;   // 0) activate TIMER3
  PeriodicTask1 = task;          // user function
  TIMER3_CTL_R = 0x00000000;    // 1) disable TIMER3A during setup
  TIMER3_CFG_R = 0x00000000;    // 2) configure for 32-bit mode
  TIMER3_TAMR_R = 0x00000002;   // 3) configure for periodic mode, default down-count settings
  TIMER3_TAILR_R = period-1;    // 4) reload value
  TIMER3_TAPR_R = 0;            // 5) bus clock resolution
  TIMER3_ICR_R = 0x00000001;    // 6) clear TIMER3A timeout flag
  TIMER3_IMR_R = 0x00000001;    // 7) arm timeout interrupt
  NVIC_PRI8_R = (NVIC_PRI8_R&0x00FFFFFF)|0x80000000; // 8) priority 4
// interrupts enabled in the main program after all devices initialized
// vector number 51, interrupt number 35
  NVIC_EN1_R = 1<<(35-32);      // 9) enable IRQ 35 in NVIC
  TIMER3_CTL_R = 0x00000001;    // 10) enable TIMER3A
    EndCritical(sr);
}

void Timer3A_Handler(void){
  TIMER3_ICR_R = TIMER_ICR_TATOCINT;// acknowledge TIMER3A timeout
  (*PeriodicTask1)();                // execute user task
}
void Timer3A_Set_Period(uint32_t period){
  TIMER3_TAILR_R = period-1;    // 4) reload value
  TIMER3_TAPR_R = 0;            // 5) bus clock resolution
  TIMER3_ICR_R = 0x00000001;    // 6) clear TIMER3A timeout flag
```

```
  TIMER3_IMR_R = 0x00000001;    // 7) arm timeout interrupt
}
```

> 代码基于PeriodicTimer0AInts_4C123修改，除了以上代码外，其余需要将UART_4C123中的UART.h和UART.c拷贝到工程目录下。