计算机科学与技术学院

嵌入式系统实验报告

（八）

姓　　名：banban

专　　业：计算机科学与技术

班　　级：

学　　号：

指导教师：

2023 年 4 月 25 日

一、任务要求

1、 跑通并理解 pwmdemo 项目（可以将四路占空比不同的 pwm 输出接至四个 led，观察其亮度差别）

2、 自行设计程序产生一路 pwm 输出，通过两个独立按键动态改变占空比实现调光（接 led）或直流电机转速调整（接直流电机）

二、实验报告要求

1、 任务中自编程序的源代码（加上注释）

2、 能说明软件仿真结果的截图、反映硬件电路连接和硬件验证结果的图片或视频

三、实验过程

一. 任务一：跑通并理解 pwmdemo 项目（可以将四路占空比不同的 pwm 输出接至四个 led，观察其亮度差别）

1. 代码：

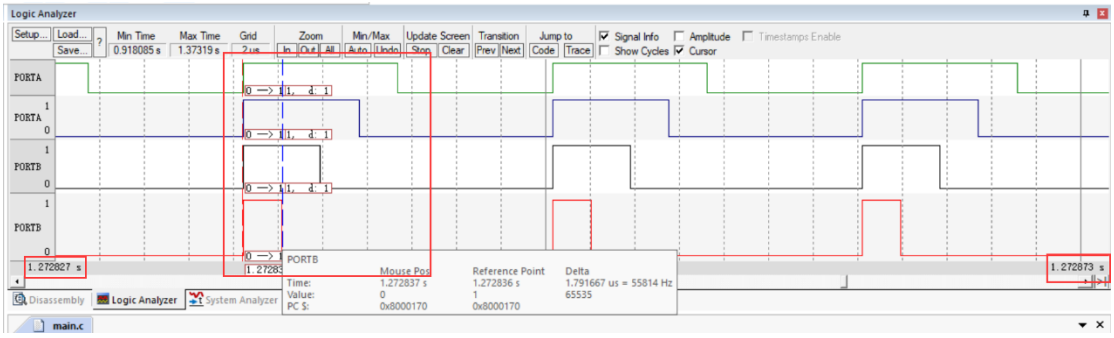
```
// main.c
// -----------------------------------------------------------
// 配置 TIM3 复用输出 PWM 时用到的 I/O
static void TIM3_GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    // PCLK1 经过 2 倍频后作为 TIM3 的时钟源等于 72MHz
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    // GPIOA 和 GPIOB 时钟使能
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB,
ENABLE);
    // GPIOA 配置：TIM3 通道 1 和 2 作为复用功能推挽
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  // 复用推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
```

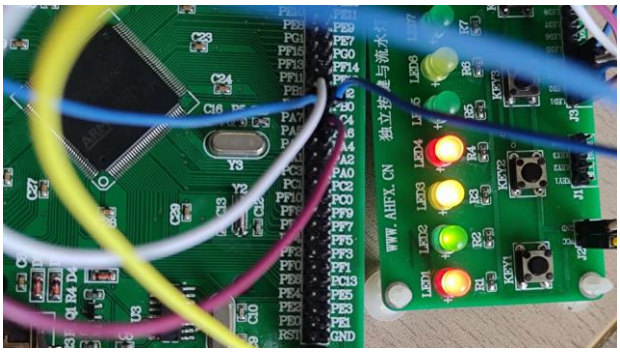
```
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    // GPIOB 配置：TIM3 通道 3 和 4 作为复用功能推挽
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
}
// 配置 TIM3 输出的 PWM 信号的模式，如周期、极性、占空比
static void TIM3_Mode_Config(void) {
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;

    /* PWM 信号电平跳变值 */
    u16 CCR1_Val = 500;
    u16 CCR2_Val = 375;
    u16 CCR3_Val = 250;
    u16 CCR4_Val = 125;

    /* Time base configuration */
    TIM_TimeBaseStructure.TIM_Period = 999;  // 当定时器从 0 计数到 999，即为 1000 次，为一个定时周期
    TIM_TimeBaseStructure.TIM_Prescaler = 0;  // 设置预分频：不预分频，即为 72MHz
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;  // 设置时钟分频系数：不分频
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; // 向上计数模式
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

    /* PWM1 Mode configuration: Channel1 */
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;  // 配置为 PWM 模式 1
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = CCR1_Val;  // 设置跳变值，当计数器计数到这个值时，电平发生跳变
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;  // 当定时器计数值小于 CCR1_Val 时为高电平
    TIM_OC1Init(TIM3, &TIM_OCInitStructure);  // 使能通道 1
```

```
    TIM_OC1PreloadConfig(TIM3, TIM_OCPreload_Enable);

    /* PWM1 Mode configuration: Channel2 */
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = CCR2_Val;  // 设置通道 2 的电平跳变值，输出另外一个占空比的 PWM
    TIM_OC2Init(TIM3, &TIM_OCInitStructure);  // 使能通道 2
    TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Enable);

    /* PWM1 Mode configuration: Channel3 */
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = CCR3_Val;  // 设置通道 3 的电平跳变值，输出另外一个占空比的 PWM
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);  // 使能通道 3
    TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Enable);

    /* PWM1 Mode configuration: Channel4 */
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = CCR4_Val;  // 设置通道 4 的电平跳变值，输出另外一个占空比的 PWM
    TIM_OC4Init(TIM3, &TIM_OCInitStructure);  // 使能通道 4
    TIM_OC4PreloadConfig(TIM3, TIM_OCPreload_Enable);
    TIM_ARRPreloadConfig(TIM3, ENABLE);  // 使能 TIM3 重载寄存器 ARR
    TIM_Cmd(TIM3, ENABLE);  // 使能定时器 3
}

// TIM3 输出 PWM 信号初始化，只要调用这个函数，TIM3 的四个通道就会有 PWM 信号输出
void TIM3_PWM_Init(void) {
    TIM3_GPIO_Config();
    TIM3_Mode_Config();
}

int main(void) {
    TIM3_PWM_Init();    /* TIM3 PWM 波输出初始化，并使能 TIM3 PWM 输出 */
    while (1) {}
```

}

2. 图片效果





二. 任务二：自行设计程序产生一路 pwm 输出，通过两个独立按键动态改变占空比实现调光或呼吸灯

1. 代码：

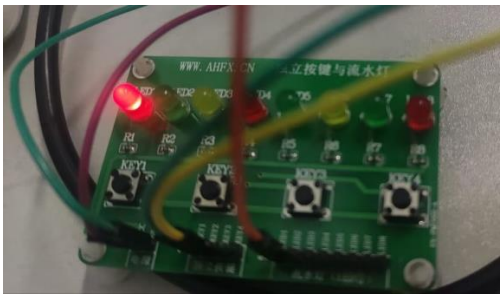
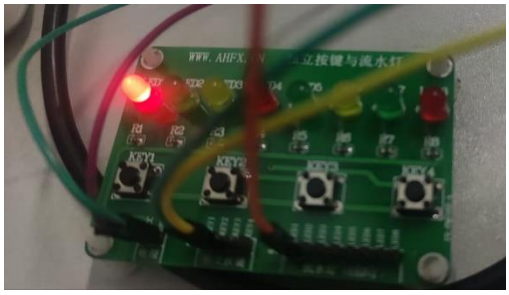
```
// main.c
// ------------------------------------------------------------
```

```c
#include "stm32f10x.h"
#include "SysTick.h"
#include "tim.h"

void myTIM_Tim3BaseInit(u16 _arr, u16 _psc) {
    TIM_TimeBaseInitTypeDef TIM_TimBaseStructure;   // 时基初始化结构体
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);    // 时钟使能
    TIM_DeInit(TIM3);                 // 复位定时器 3
    TIM_InternalClockConfig(TIM3);    // 选择内部时钟

    /* 定时器 TIM3 初始化 */
    TIM_TimBaseStructure.TIM_Period = _arr;  // 设置在下一个更新事件装入活动的自动重装载寄存器
    TIM_TimBaseStructure.TIM_Prescaler = _psc;  // 设置用来作为 TIMx 时钟频率除数的预分频值
    TIM_TimBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;  // TIM 向上计数模式
    TIM_TimeBaseInit(TIM3, &TIM_TimBaseStructure);  // 根据指定的参数初始化 TIMx 的时间基数单位

    TIM_Cmd(TIM3, DISABLE);  // 失能 TIM3
}

void myTIM_TimingItInit(void) {
    NVIC_InitTypeDef NVIC_InitStructure; // NVIC 初始化结构体

    /* 使能更新中断，并清除更新中断标志位 */
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE); // 使能指定的 TIM3 中断
    TIM_ClearITPendingBit(TIM3, TIM_IT_Update); // 清除 TIMx 更新中断标志

    /* 中断优先级 NVIC 设置 */
    NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

void myTIM_Tim3PWM2Init(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;//输出比较结构体变量

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO,
```

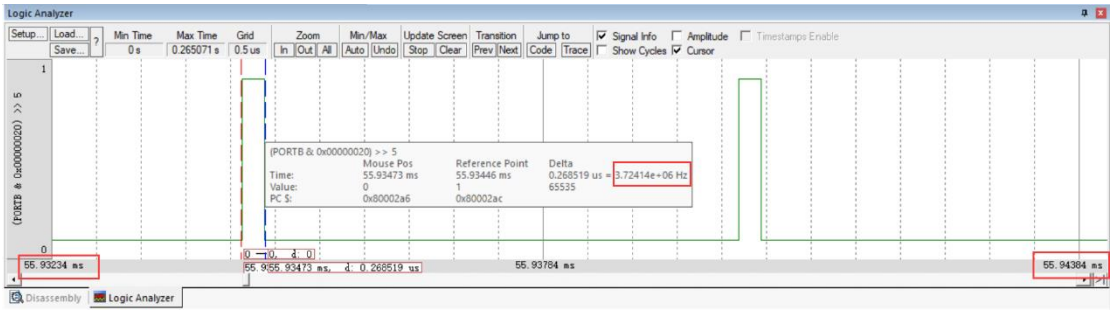
```
ENABLE);

    GPIO_PinRemapConfig(GPIO_PartialRemap_TIM3, ENABLE); // Timer3 定时器 3
部分引脚重映射
    //PB5 初始化 设置该引脚为复用输出功能，输出 TIM3 CH2 的 PWM 脉冲波形
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;   // TIM_CH2
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  // 复用推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);  // 初始化 GPIO

    /* 初始化 TIM3 Channel2 PWM 模式 */
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_OCNPolarity = TIM_OCPolarity_High;
    TIM_OC2Init(TIM3, &TIM_OCInitStructure);

    TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Enable);  // 使能 Timer3 在 CCR 上
的定时器预装载功能
}

/* 使能定时器 x 的计数器 */
void myTIM_TimxStart(TIM_TypeDef *TIMx) {
    TIM_Cmd(TIMx, ENABLE);//使能定时器 3
}

void LED_Init(void){
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOB,&GPIO_InitStructure);
    LED2(1);
}

int main(void) {
   SysTick_Init(); // 系统延时函数初始化
   LED_Init();
   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  // 设置 NVIC 中断分组 2:2
抢占优先级
   myTIM_Tim3BaseInit(600,0);  // 时基单元初始化，arr=899,psc=0
   myTIM_Tim3PWM2Init();       // 定时器 3 通道 2PWM 初始化
   myTIM_TimxStart(TIM3);      // 开启定时器 3
```

```
    while (1){
        for(int i=1; i<=500; i++){
            Delay_ms(4);
            TIM_SetCompare2(TIM3, i);//将 PWM 的值不断赋给 CCR，实现调整占空比的目的
        }
        for(int i=500; i>=1; i--){
            Delay_ms(4);
            TIM_SetCompare2(TIM3, i);//将 PWM 的值不断赋给 CCR，实现调整占空比的目的
        }
    }
}
```

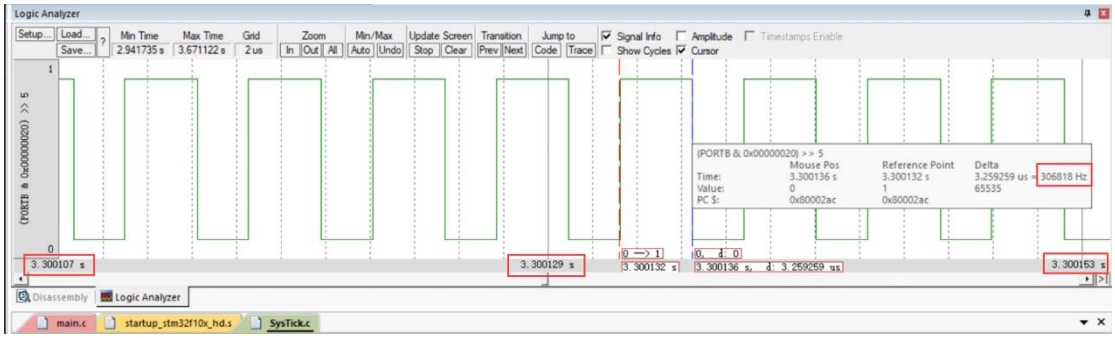
2. 图片效果

55ms 左右



3.3s 左右



四、总结与分析

本次实验目的在于熟悉 PWM 技术，并通过实验来理解和掌握 PWM 输出的基本原理和应用。

实验中我们使用了 pwm demo 项目，并将四路占空比不同的 PWM 输出接至四个 LED，观察其亮度差别。同时，我们还自行设计程序产生一路 PWM 输出，通过两个独立按键动态改变占空比实现调光（接 LED）或直流电机转速调整（接直流电机）。

经过此实验，我已经初步掌握了 PWM 技术的基本原理和应用方法。我们通过实验掌握了如何使用按键来动态地改变 PWM 输出的占空比，从而实现对 LED 亮度和直流电机转速的动态调节。这种实验方式既增加了我们对 PWM 技术的理解，也提高了我们的动手实验能力。