《微机原理与接口技术》

实验指导书

实验三 串行通信和定时器

“微机原理与接口技术”课程教学团队

北京航空航天大学

仪器科学与光电工程学院

2022 年 11 月

一、实验目的

1.掌握串行通信的原理，学习 PC 机和 arm 硬件平台上的软硬件综合调试方法并实现串口通信。

2.了解定时器的几种工作模式以及几种模式各自的应用场景。

二、实验设备

1.PC 计算机；
2.Keil for ARM(MDK)开发环境 V5.36 及以上；
3.NUCLEO 开发板，或 Disco 开发板，主要查看相应的开发板电路图。
4.串口调试工具软件 sscom5.13.1

三、实验内容

1. PC 机与 Cortex-M7 处理器之间串行通信。

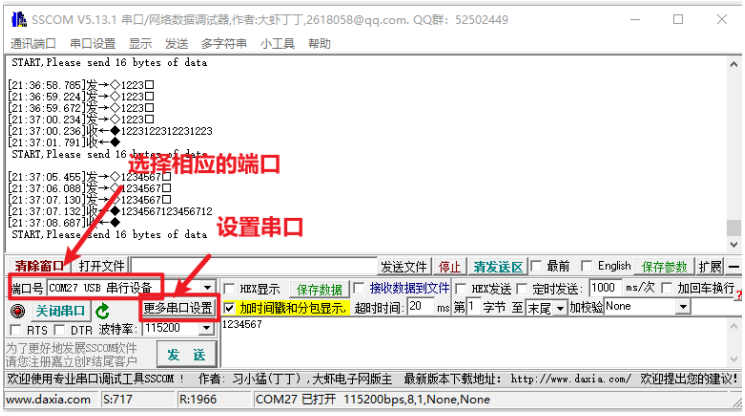
2. 定时器；利用定时器完成 LED 灯的 1s 的闪烁、利用定时器的 PWM 输出模式，实现 LED 灯渐变（呼吸灯），

3. 拓展实验。ADC 实验，将光敏电阻两端电压通过 ADC 转换为数字信号，通过数字信号的大小控制 PWM 输出，从而控制 LED 亮度。

4、拓展实验。自定义协议，通过 PC 机串行通信实现实验平台 LED 的控制。

四、实验步骤

4.1 PC 串口工具软件



ST-Link 仿真器通过 USB 虚拟出 1 个 UART 端口，该 UART 端口已经与实验的处理器的 UART 端口连接。PC 机可以直接使用 USB 虚拟出的 UART 与实验的 CPU 通信。

4.2 编写 UART 相关程序

1. 初始化串口

定义全局变量

```
#define HAL_TIMEOUT_VALUE 0xFFFFFFFF
#define countof(a) (sizeof(a) / sizeof(*(a)))

UART_HandleTypeDef UartHandle;
uint8_t RxBuffer[1];  //接收串口数据buffer
```

HAL_UART_MspDeInit 函数作用是初始化串口的 GPIO，不用在主函数调用，注意 Nucleo 开发板串口 3 的 TX 是 PD8、RX 是 PD9，Disco 开发板串口 3 的 TX 是 PB10、RX 是 PB11。

```
void HAL_UART_MspInit(UART_HandleTypeDef *huart)
{
  GPIO_InitTypeDef  GPIO_InitStruct;
  RCC_PeriphCLKInitTypeDef RCC_PeriphClkInit;

  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /* Select HSI as source of USARTx clocks */
  RCC_PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART3;
  RCC_PeriphClkInit.Usart234578ClockSelection = RCC_USART3CLKSOURCE_HSI;
  HAL_RCCEx_PeriphCLKConfig(&RCC_PeriphClkInit);

  /* Enable USARTx clock */
  __HAL_RCC_USART3_CLK_ENABLE();

  /*##-2- Configure peripheral GPIO ##########################################*/
  /* UART TX GPIO pin configuration  */
  GPIO_InitStruct.Pin       = GPIO_PIN_8;
  GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull      = GPIO_PULLUP;
  GPIO_InitStruct.Speed     = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /* UART RX GPIO pin configuration  */
  GPIO_InitStruct.Pin = GPIO_PIN_9;
  GPIO_InitStruct.Alternate = GPIO_AF7_USART3;

  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /* NVIC for USART */
  HAL_NVIC_SetPriority(USART3_IRQn, 0, 1);
  HAL_NVIC_EnableIRQ(USART3_IRQn);
}
```

```
void HAL_UART_MspDeInit(UART_HandleTypeDef *huart)
{
  /*##-1- Reset peripherals ##################################################*/
  __HAL_RCC_USART3_FORCE_RESET();
  __HAL_RCC_USART3_RELEASE_RESET();

  /*##-2- Disable peripherals and GPIO Clocks #################################*/
  /* Configure UART Tx as alternate function  */
  HAL_GPIO_DeInit(GPIOD, GPIO_PIN_8);
  /* Configure UART Rx as alternate function  */
  HAL_GPIO_DeInit(GPIOD, GPIO_PIN_9);
}
```

```
void USART3_IRQHandler(void)
{
  HAL_UART_IRQHandler(&UartHandle);
}
```

2. 主函数程序编写

初始化串口句柄

```
UartHandle.Instance             = USART3;
UartHandle.Init.BaudRate        = 115200;   //波特率
UartHandle.Init.WordLength      = UART_WORDLENGTH_8B;//字长为 8 位
UartHandle.Init.StopBits        = UART_STOPBITS_1;//一个停止位
UartHandle.Init.Parity          = UART_PARITY_NONE;//无奇偶校验位
UartHandle.Init.HwFlowCtl       = UART_HWCONTROL_NONE;//无硬件流控
UartHandle.Init.Mode            = UART_MODE_TX_RX;//收发模式
UartHandle.Init.ClockPrescaler  = UART_PRESCALER_DIV1;
UartHandle.Init.OneBitSampling  = UART_ONE_BIT_SAMPLE_DISABLE;
UartHandle.Init.OverSampling    = UART_OVERSAMPLING_16;
```

调用串口初始化函数 HAL_UART_Init，对串口进行初始化。

```
if(HAL_UART_Init(&UartHandle) != HAL_OK)
{
  Error_Handler();
}
```

串口收发数据函数如下，对于 receive 函数，三个参数分别为串口句柄、接收数据所用数组、接收数据长度，对于 transmit 函数四个参数分别为串口句柄、发送数据所用数组、发生数据长度、超时时间。

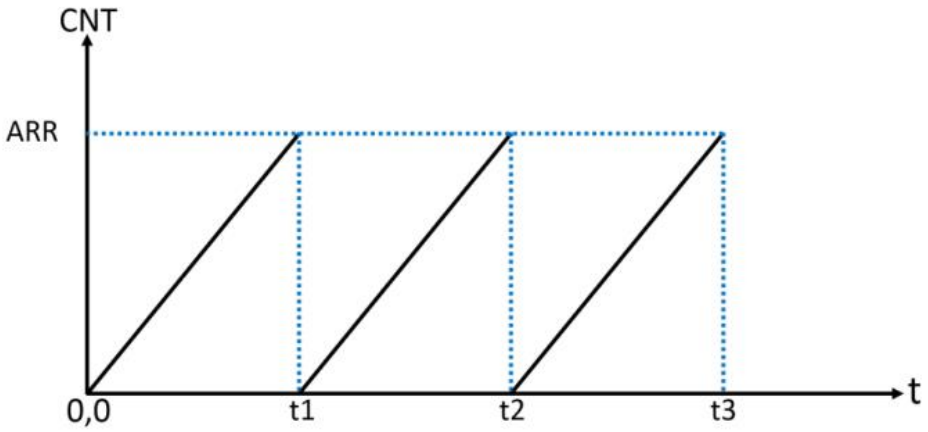
```
HAL_UART_Receive_IT(&UartHandle, (uint8_t*)&RxBuffer, 1);
HAL_UART_Transmit(&UartHandle, (uint8_t*)&RxBuffer, countof(RxBuffer), HAL_TIMEOUT_VALUE);
```

任务：编写程序，实现下述功能：通过 PC 机发送字符‘Y’，控制实验平台点亮 LED，并回传 PC 机“ON”；PC 机发送字符‘N’，实验平台熄灭 LED，并回传 PC 机“OFF”。

4.3 定时器

4.3.1 利用定时器完成 LED 周期闪烁

定时器设置为向上计数时，定时器中断示意如图：



定时器中断示意图

CNT 计数器从 0 开始计数，当 CNT 的值和 ARR 相等时（ t1），产生一个溢出中断，然后 CNT 复位（清零），然后继续向上计数，依次循环。图中的 t1、t2、 t3 就是定时器溢出中断产生的时刻。通过修改 ARR 的值，可以改变定时时间。另外，通过修改 PSC 的值，使用不同的计数频率（改变图中 CNT 的斜率），也可以改变定时时间。

1. 创建定时器句柄和定时器初始化函数

```
11  TIM_HandleTypeDef TIM3_Handler;      //定时器句柄
12
13
14  void TIM3_Init(uint16_t arr,uint16_t psc)
15  {
16      TIM3_Handler.Instance=TIM3;                          //通用定时器3
17      TIM3_Handler.Init.Prescaler=psc;                     //分频
18      TIM3_Handler.Init.CounterMode=TIM_COUNTERMODE_UP;    //向上计数器
19      TIM3_Handler.Init.Period=arr;                        //自动装载值
20      TIM3_Handler.Init.ClockDivision=TIM_CLOCKDIVISION_DIV1;//时钟分频因子
21      HAL_TIM_Base_Init(&TIM3_Handler);
22      HAL_TIM_Base_Start_IT(&TIM3_Handler); //使能定时器3和定时器3更新中断：TIM_IT_UPDATE
23  }
24
```

其中初始化函数中两个参数含义如下：arr：自动重装值。psc：时钟预分频数。

定时器溢出时间计算方法:Tout=((arr+1)*(psc+1))/Ft s。其中 Ft= 200MHz

2. 初始化定时器中断相关函数，代码如下

```c
//定时器底册驱动，开启时钟，设置中断优先级
//此函数会被HAL_TIM_Base_Init()函数调用
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
{
    if(htim->Instance==TIM3)
  {
    __HAL_RCC_TIM3_CLK_ENABLE();            //使能TIM3时钟
    HAL_NVIC_SetPriority(TIM3_IRQn,1,3);    //设置中断优先级，抢占优先级1，子优先级3
    HAL_NVIC_EnableIRQ(TIM3_IRQn);          //开启ITM3中断
  }
}

//定时器3中断服务函数
void TIM3_IRQHandler(void)
{
    HAL_TIM_IRQHandler(&TIM3_Handler);
}

//定时器3中断服务函数调用
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim==(&TIM3_Handler))
    {
        //定时器3每一次中断溢出都会调用这个函数，在这编写控制LED灯的代码
    }
}
```

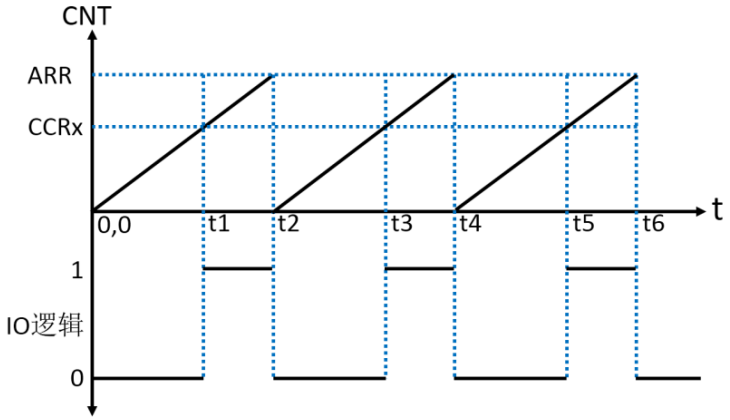
3. 主函数程序

主函数中调用 TIM3_Init（）函数，就能完成定时器控制 LED 闪烁实验了。

任务：完成 LED 灯闪烁，通过修改 arr 和 psc，达到快闪和慢闪的效果，其中快闪周期 1s，慢闪周期 5s。

4.3.2 利用 PWM 完成呼吸灯实验

PWM 原理如图所示：

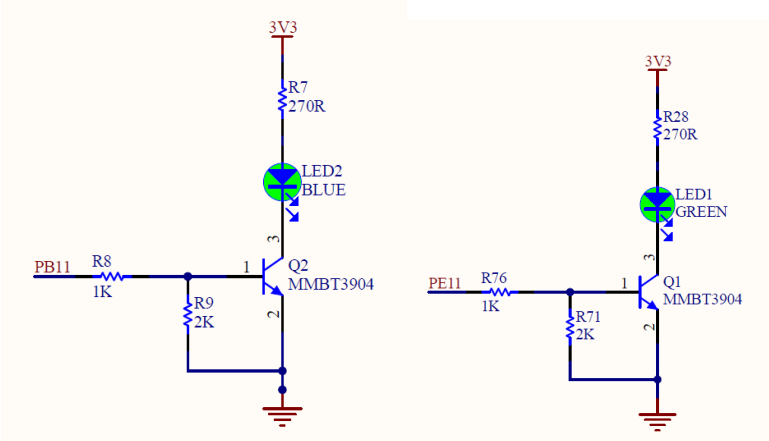


PWM 原理图

当定时器计算器增加大于 CCRx 时，管脚输出高电平，反之输出低电平，改变 CCRx 的值，就可以改变 PWM 输出的占空比，改变 ARR 的值，就可以改变 PWM 输出的频率，这就是 PWM 输出的原理。输出占空比不同，连在该 PWM 输出管脚的 LED 亮度也不同。

对于 Nucleo 板：PB11 对应 TIM2_CH4,PE11 对应 TIM1_CH2，下面原理图

对应的 LED 为底板的两个 LED。



下面拿底板的 LED2 呼吸灯代码举例，对于 Nucleo 板需要完成底板 LED1 的呼吸灯实验

1. **创建定时器句柄和定时器 2 通道 4 句柄，注意他们都是全局的结构体。**

```
TIM_HandleTypeDef TIM2_Handler;         //定时器句柄
TIM_OC_InitTypeDef TIM2_CH4Handler;     //定时器2通道4句柄
```

2. 编写初始化 TIM2 的 PWM 函数

此处代码和上个实验 TIM3_Init 函数类似，区别在于少了 HAL_TIM_Base_Start_IT 函数，再加上如下代码：

```
    TIM2_CH4Handler.OCMode=TIM_OCMODE_PWM1; //模式选择PWM1
    TIM2_CH4Handler.Pulse=arr/2;            //设置比较值,此值用来确定占空比,
                                            //默认比较值为自动重装载值的一半,即占空比为50%
    TIM2_CH4Handler.OCPolarity=TIM_OCPOLARITY_LOW; //输出比较极性为低
    HAL_TIM_PWM_ConfigChannel(&TIM2_Handler,&TIM2_CH4Handler,TIM_CHANNEL_4);//配置TIM2通道4
    HAL_TIM_PWM_Start(&TIM2_Handler,TIM_CHANNEL_4);//开启PWM通道4
```

3. 编写定时器底层驱动，代码如下：

```
//定时器底层驱动，时钟使能，引脚配置
//此函数会被HAL_TIM_PWM_Init()调用
//htim:定时器句柄
void HAL_TIM_PWM_MspInit(TIM_HandleTypeDef *htim)
{
    GPIO_InitTypeDef GPIO_Initure;
    __HAL_RCC_TIM2_CLK_ENABLE();        //使能定时器2
    __HAL_RCC_GPIOB_CLK_ENABLE();       //开启GPIOB时钟

    GPIO_Initure.Pin=GPIO_PIN_11;              //PB11
    GPIO_Initure.Mode=GPIO_MODE_AF_PP;         //复用推挽输出
    GPIO_Initure.Pull=GPIO_PULLUP;             //上拉
    GPIO_Initure.Speed=GPIO_SPEED_FREQ_VERY_HIGH;      //高速
  GPIO_Initure.Alternate=GPIO_AF1_TIM2; //PB11复用为TIM2_CH4
    HAL_GPIO_Init(GPIOB,&GPIO_Initure);
}
```

4. 设置 TIM 通道 4 的占空比函数

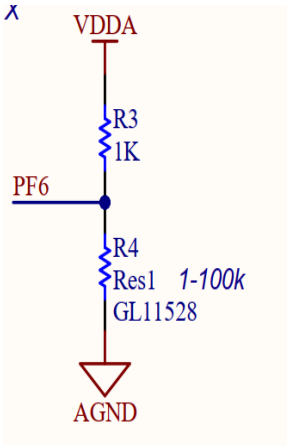
```
//设置TIM通道4的占空比
//compare:比较值
void TIM_SetTIM2Compare4(uint32_t compare)
{
   TIM2->CCR4=compare;
}
```

5. 主函数编写

在 while(1)中通过设置 TIM_SetTIM2Compare4( );函数来设置 PWM 输出的占空比（设置范围为 0- arr，arr 就是自动重装值），占空比不同 LED 灯亮度也会有变化，请自行编写 LED 从暗到亮，再由亮变暗往复循环的呼吸灯代码。

4.4 ADC 实验

Nucleo 板：连接光敏电阻的管脚如下，**PF6 连接了 ADC3 的通道 8（ADC_CHANNEL_8）**



实验步骤：

1. 初始化 ADC

```
void MY_ADC_Init(void)
{
    ADC3_Handler.Instance=ADC3;
    ADC3_Handler.Init.ClockPrescaler=ADC_CLOCK_SYNC_PCLK_DIV4;  //4分频，ADCCLK=PER_CK/4=64/4=16MHZ
    ADC3_Handler.Init.Resolution=ADC_RESOLUTION_16B;            //16位分辨率
    ADC3_Handler.Init.ScanConvMode=DISABLE;                     //非扫描模式
    ADC3_Handler.Init.EOCSelection=ADC_EOC_SINGLE_CONV;         //关闭EOC中断
    ADC3_Handler.Init.LowPowerAutoWait=DISABLE;                 //自动低功耗关闭
    ADC3_Handler.Init.ContinuousConvMode=DISABLE;               //开启单次转换
    ADC3_Handler.Init.NbrOfConversion=1;                        //1个转换在规则序列中 也就是只转换规则序列1
    ADC3_Handler.Init.DiscontinuousConvMode=DISABLE;            //禁止不连续采样模式
    ADC3_Handler.Init.NbrOfDiscConversion=0;                    //不连续采样通道数为0
    ADC3_Handler.Init.ExternalTrigConv=ADC_SOFTWARE_START;      //软件触发
    ADC3_Handler.Init.ExternalTrigConvEdge=ADC_EXTERNALTRIGCONVEDGE_NONE;//使用软件触发
//    ADC3_Handler.Init.BoostMode=ENABLE;                       //BOOST模式关闭
    ADC3_Handler.Init.Overrun=ADC_OVR_DATA_OVERWRITTEN;       //有新的数据的死后直接覆盖掉旧数据
    ADC3_Handler.Init.OversamplingMode=DISABLE;               //过采样关闭
    ADC3_Handler.Init.ConversionDataManagement=ADC_CONVERSIONDATA_DR;  //规则通道的数据仅仅保存在DR寄存器里面
    HAL_ADC_Init(&ADC3_Handler);                                //初始化

    HAL_ADCEx_Calibration_Start(&ADC3_Handler,ADC_CALIB_OFFSET,ADC_SINGLE_ENDED); //ADC校准
}
```

2. 编写 ADC 底层驱动

```
//ADC底层驱动，引脚配置，时钟使能
//此函数会被HAL_ADC_Init()调用
//hadc:ADC句柄
void HAL_ADC_MspInit(ADC_HandleTypeDef* hadc)
{
    GPIO_InitTypeDef GPIO_Initure;
    __HAL_RCC_ADC3_CLK_ENABLE();            //使能ADC3时钟
    __HAL_RCC_GPIOF_CLK_ENABLE();           //开启GPIOF时钟
    __HAL_RCC_ADC_CONFIG(RCC_ADCCLKSOURCE_CLKP); //ADC外设时钟选择

    GPIO_Initure.Pin=GPIO_PIN_6;            //PF6
    GPIO_Initure.Mode=GPIO_MODE_ANALOG;     //模拟
    GPIO_Initure.Pull=GPIO_NOPULL;          //不带上下拉
    HAL_GPIO_Init(GPIOA,&GPIO_Initure);
}
```

3. 获取 ADC 值

```
int16_t Get_Adc(int32_t ch)
{
    ADC_ChannelConfTypeDef ADC3_ChanConf;
    ADC3_ChanConf.Channel=ch;                                   //通道
    ADC3_ChanConf.Rank=ADC_REGULAR_RANK_1;                      //1个序列
    ADC3_ChanConf.SamplingTime=ADC_SAMPLETIME_64CYCLES_5;       //采样时间
    ADC3_ChanConf.SingleDiff=ADC_SINGLE_ENDED;                  //单边采集
    ADC3_ChanConf.OffsetNumber=ADC_OFFSET_NONE;
    ADC3_ChanConf.Offset=0;
    HAL_ADC_ConfigChannel(&ADC3_Handler,&ADC3_ChanConf);        //通道配置

    HAL_ADC_Start(&ADC3_Handler);                               //开启ADC

    HAL_ADC_PollForConversion(&ADC3_Handler,10);                //轮询转换
    return (int16_t)HAL_ADC_GetValue(&ADC3_Handler);            //返回最近一次ADC3规则组的转换结果
}
```

因为 ADC 为 16 位，所以此函数返回值大小在 0-65535 之间。

4. 获取指定通道的转换值，取 times 次,然后平均

```
int16_t Get_Adc_Average(int32_t ch,int8_t times)
{
  int32_t temp_val=0;
  int8_t t;
  for(t=0;t<times;t++)
  {
    temp_val+=Get_Adc(ch);
    HAL_Delay(5);
  }
  return temp_val/times;
}
```

5. 主函数代码

任务：主程序中实现如下功能，比如用手慢慢遮挡光敏电阻，此时光强减小，则某个 LED 也越来越暗（利用上节的 PWM）。

4.5 调试程序完成实验报告

调试程序，并完成实验报告。

实验报告每人提交一份，需要包含调试实验的具体步骤并进行相应的分析。

若出现极为明显的抄袭现象，所涉及的同学本次实验成绩为 0，提交报告时间另行通知。

提交内容：

1. 实验工程源代码；
2. 实验报告；

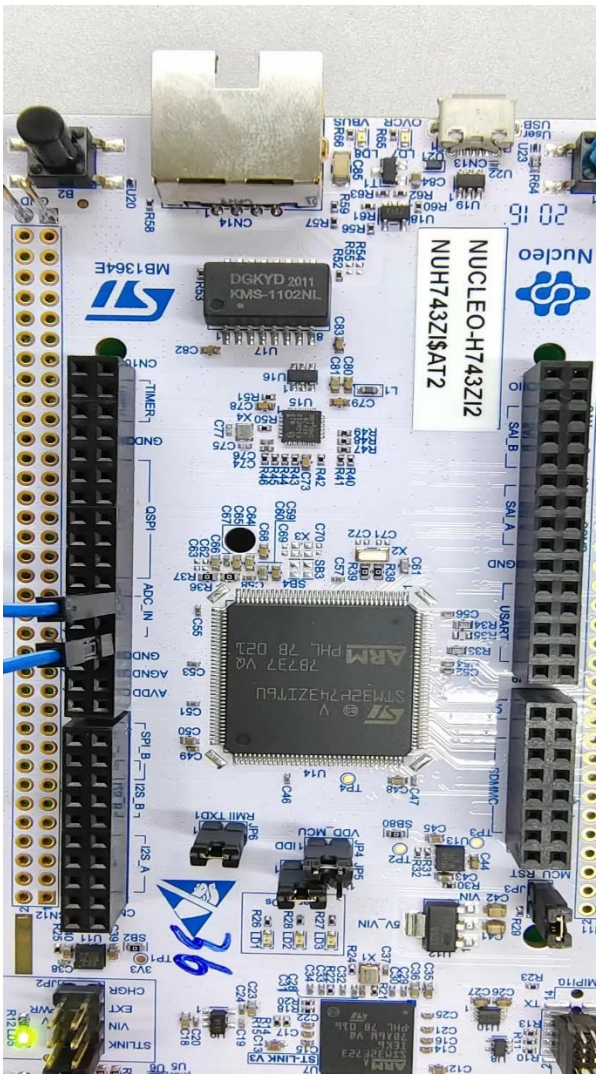
压缩打包，报告命名格式：“学号+姓名+实验 x+冯（于，吴）”

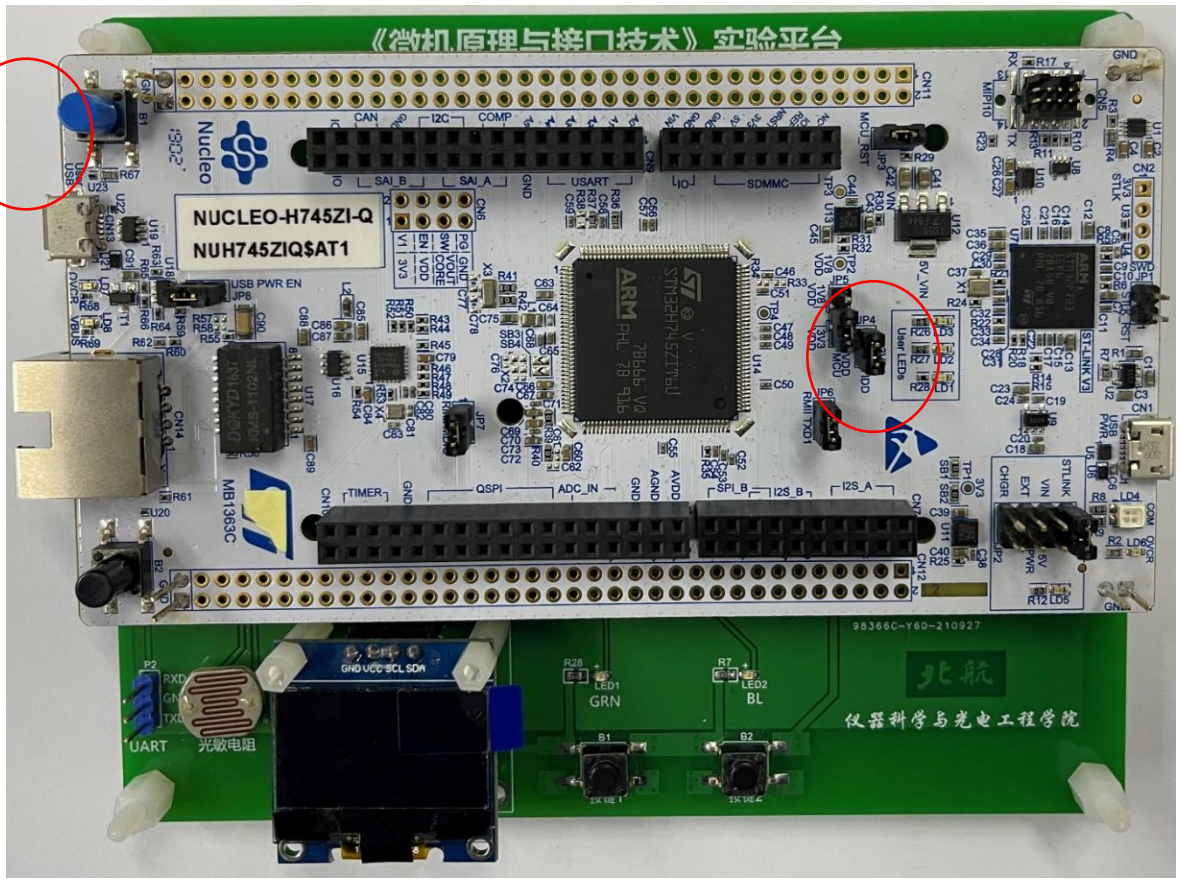
五、参考程序

见附件。

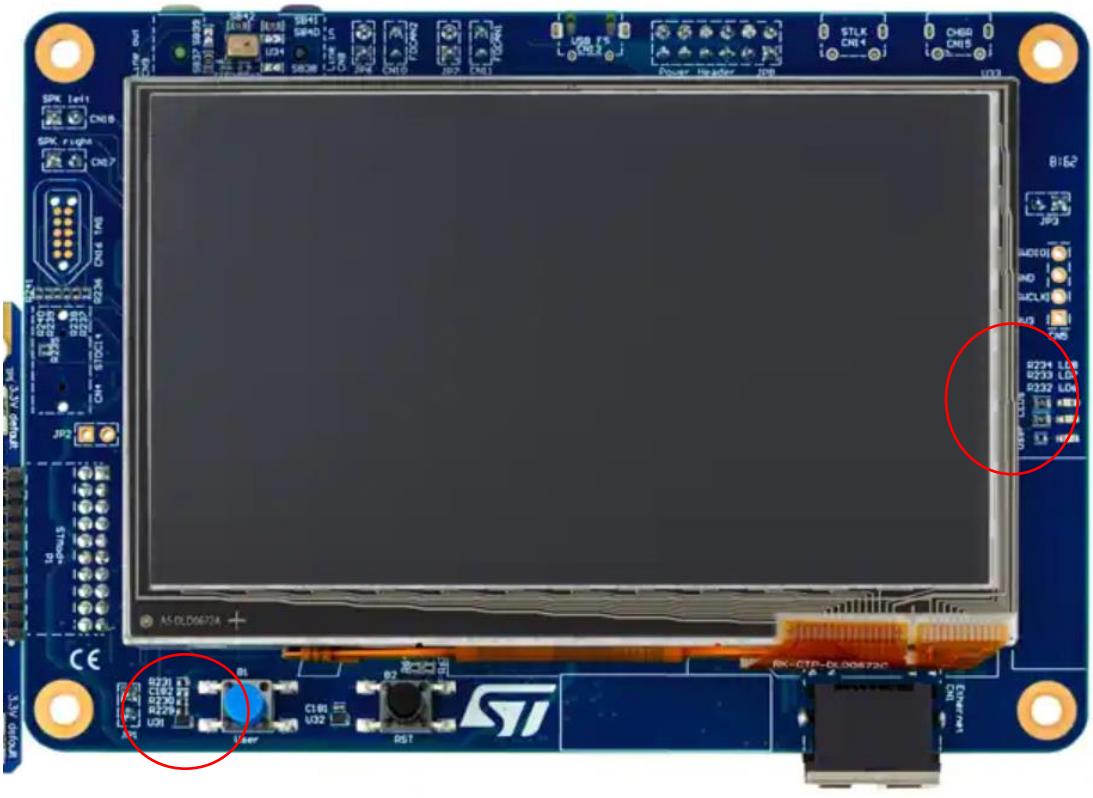
六、注意事项

（1）使用光敏电阻时注意，对于 nucleo-h743zi-q 开发板（此开发板只有两套），需要用杜邦线把 con10 的管脚 7 和管脚 11 短接，如图：





实验平台实物照片（Nucleo）



实验平台实物照片（Disco 版）