北京航空航天大學
BEIHANG UNIVERSITY

《微机原理》实验报告

姓名：曹建钦

学号：20375177

实验内容

（一）PC 机与 Cortex-M7 处理器之间串行通信

- 定义全局变量（如图 1 所示）和串口初始化函数（如图 2 所示）：

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

UART_HandleTypeDef UartHandle;
uint8_t RxBuffer[1];
```

图 1：定义全局变量

```
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 configuaration*/
  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);
}
```

图 2：UART 初始化函数

- 主函数程序编写：

  初始化串口句柄，并调用串口初始化函数 HAL_UART_Init，对串口进行初始化，如图 3 所示：

```
//输入代码部分
UartHandle.Instance             = USART3;
UartHandle.Init.BaudRate        = 115200;
UartHandle.Init.WordLength      = UART_WORDLENGTH_8B;
UartHandle.Init.StopBits        = UART_STOPBITS_1;
UartHandle.Init.Parity          = UART_PARITY_NONE;
UartHandle.Init.HwFlowCtl       = UART_HWCONTROL_NONE;
UartHandle.Init.Mode            = UART_MODE_TX_RX;
UartHandle.Init.ClockPrescaler  = UART_PRESCALER_DIV1;
UartHandle.Init.OneBitSampling  = UART_ONE_BIT_SAMPLE_DISABLE;
UartHandle.Init.OverSampling    = UART_OVERSAMPLING_16;

HAL_UART_MspInit(&UartHandle);
USART3_IRQHandler();
if(HAL_UART_Init(&UartHandle) != HAL_OK){
  Error_Handler();
}
```

图 3：初始化串口句柄

编写程序（如图 4），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”。且 PC 机发送其他字符时，实验平台返回发送的字符。

```
HAL_StatusTypeDef statu = 1;
while(1){
  while(statu){
    statu = HAL_UART_Receive_IT(&UartHandle, (uint8_t*)&RxBuffer, 1);
  }
    if(RxBuffer[0] == 'Y')
{
  LED1_GPIO_CLK_ENABLE();
  GPIO_InitTypeDef GPIO_InitStructForLed1;
  GPIO_InitStructForLed1.Pin = LED1_PIN;
  GPIO_InitStructForLed1.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructForLed1.Pull = GPIO_NOPULL;
  GPIO_InitStructForLed1.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(LED1_GPIO_PORT, &GPIO_InitStructForLed1);
  HAL_GPIO_WritePin(LED1_GPIO_PORT, LED1_PIN, GPIO_PIN_SET);
  char strSend[] = "ON";
  HAL_UART_Transmit(&UartHandle, (uint8_t*)&strSend, countof(strSend), HAL_TIMEOUT_VALUE);
}

else if(RxBuffer[0] == 'N')
{
  LED1_GPIO_CLK_ENABLE();
  GPIO_InitTypeDef GPIO_InitStructForLed1;
  GPIO_InitStructForLed1.Pin = LED1_PIN;
  GPIO_InitStructForLed1.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructForLed1.Pull = GPIO_NOPULL;
  GPIO_InitStructForLed1.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(LED1_GPIO_PORT, &GPIO_InitStructForLed1);
  HAL_GPIO_WritePin(LED1_GPIO_PORT, LED1_PIN, GPIO_PIN_RESET);
  char strSend[] = "OFF";
  HAL_UART_Transmit(&UartHandle, (uint8_t*)&strSend, countof(strSend), HAL_TIMEOUT_VALUE);
}
else{
  HAL_UART_Transmit(&UartHandle, (uint8_t*)&RxBuffer, countof(RxBuffer), HAL_TIMEOUT_VALUE);
}
  statu = 1;
}
/*
```

图 4：控制程序

思路为：串口要想持续工作，首先应该让串口收发数据函数内嵌于循环结构中；而要想让 PC 机发送字符后实验平台才执行对应任务，且实验平台发送完后应该处于等待输入而非持续输出的状态，这里我发现 HAL_UART_Receive_IT 函数的返回值类型为 HAL_StatusTypeDef，如图 5 所示，而 HAL_StatusTypeDef 为枚举类型，如图 6 所示，所以调用串口接收数据函数后会返回整数 0，于是这里

用变量 statu 作为接收数据的标志，当未接收数据时一直进行执行 135-137 行的循环体，持续接收数据，一旦接收到了数据，则执行 138-167 行的判断语句，实现单次任务的执行。

```
1370 HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
1371 {
1372   /* Check that a Rx process is not already ongoing */
1373   if (huart->RxState == HAL_UART_STATE_READY)
1374   {
1375     if ((pData == NULL) || (Size == 0U))
1376     {
1377       return HAL_ERROR;
1378     }
1379
1380     __HAL_LOCK(huart);
1381
1382     /* Set Reception type to Standard reception */
1383     huart->ReceptionType = HAL_UART_RECEPTION_STANDARD;
1384
1385     if (!(IS_LPUART_INSTANCE(huart->Instance)))
1386     {
```

图 5：HAL_UART_Receive_IT 函数返回值类型

```
39 typedef enum
40 {
41   HAL_OK       = 0x00,
42   HAL_ERROR    = 0x01,
43   HAL_BUSY     = 0x02,
44   HAL_TIMEOUT  = 0x03
45 } HAL_StatusTypeDef;
46
```

图 6：枚举类型

- 验证：

  PC 机通过端口与实验平台连接，如图 7 所示，连接成功，且使用 COM7 虚拟端口：

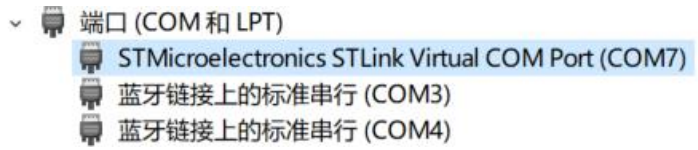

图 7：连接成功

打开串口调试工具 sscom，并打开串口，如图 8 所示：

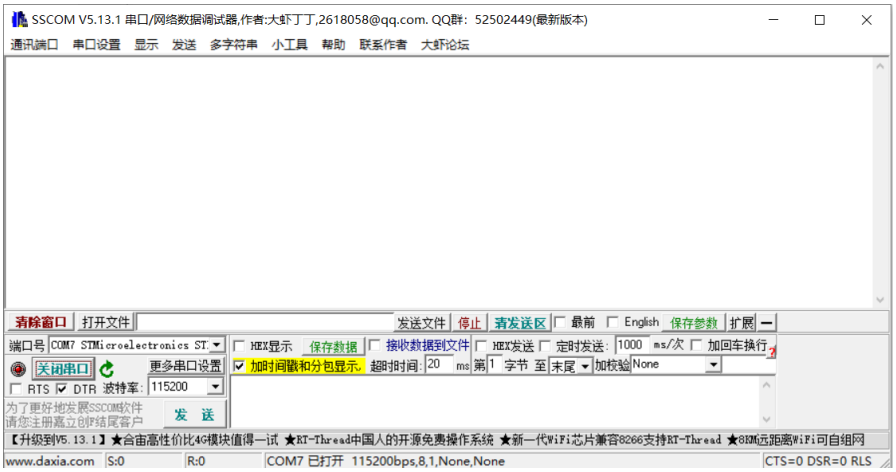

图 8：打开串口调试工具

发送字符‘Y’，实验平台 LED1 亮，如图 9 所示，返回“ON”，如图 10 所示：

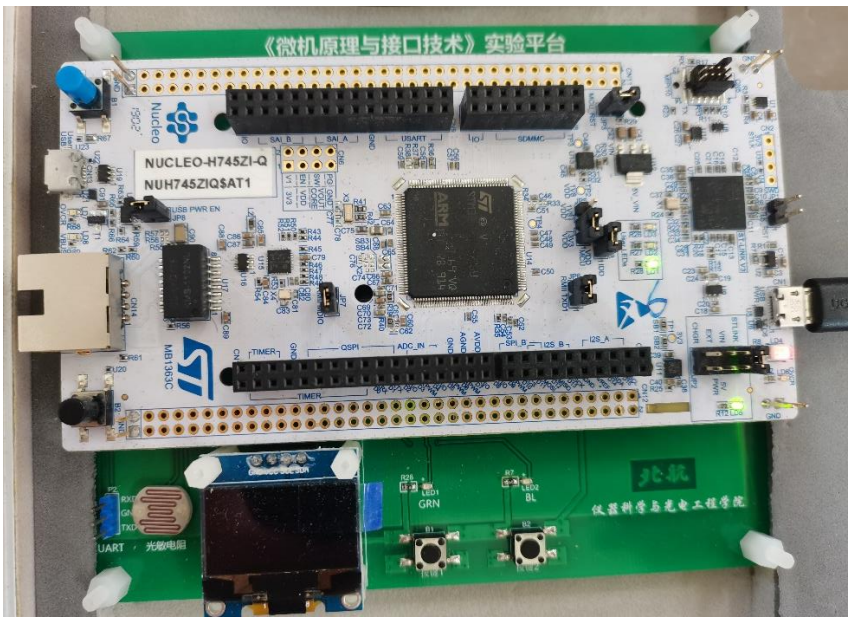

图 9：实验平台 LED1 亮

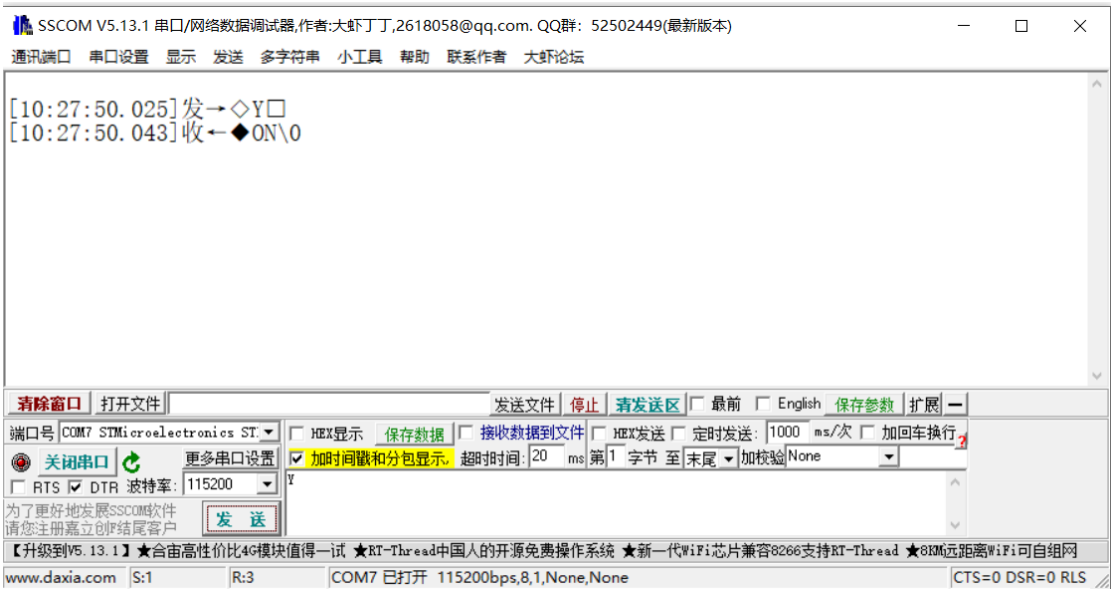

图 10：返回“ON”

发送字符‘N’，实验平台 LED1 熄灭，如图 11 所示，返回“OFF”，如图 12 所示：

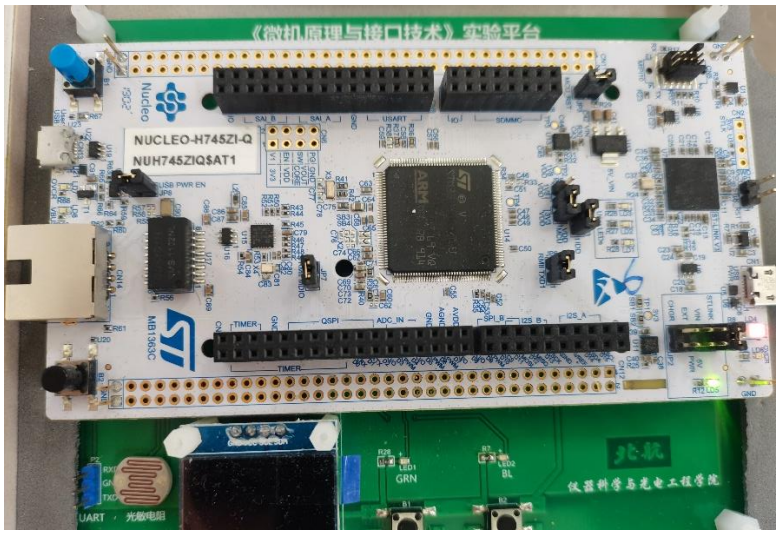

图 11：实验平台 LED1 熄灭

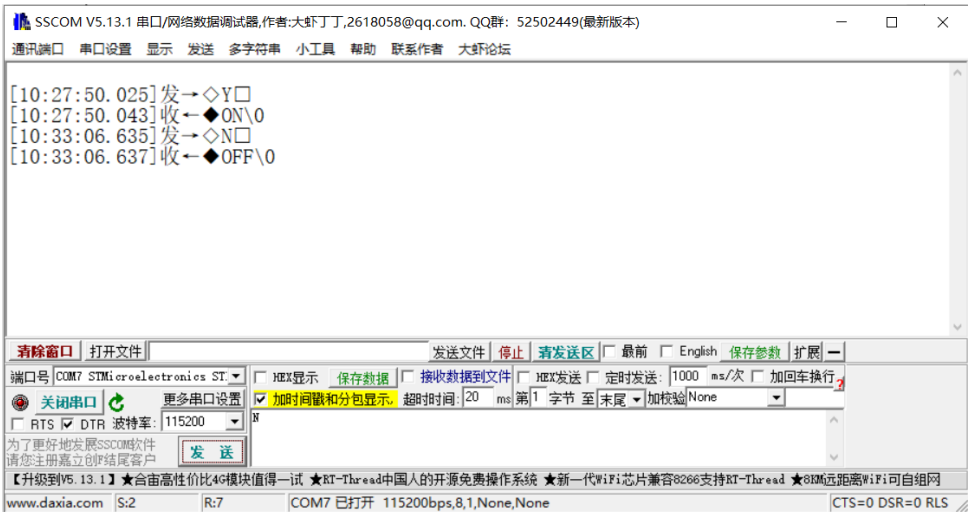

图 12：返回“OFF”

发送其他字符/字符串，实验平台无响应，返回发送的字符，如图 13，14 所示：

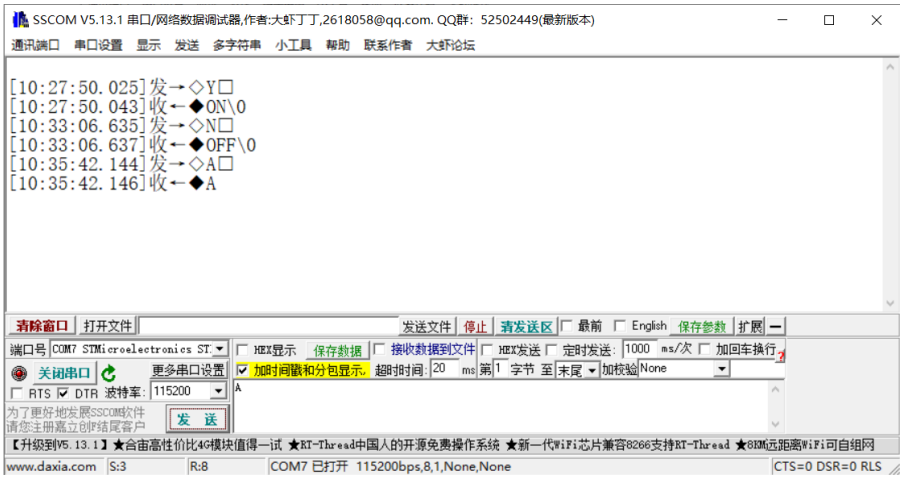

图 13：发送字符‘A’

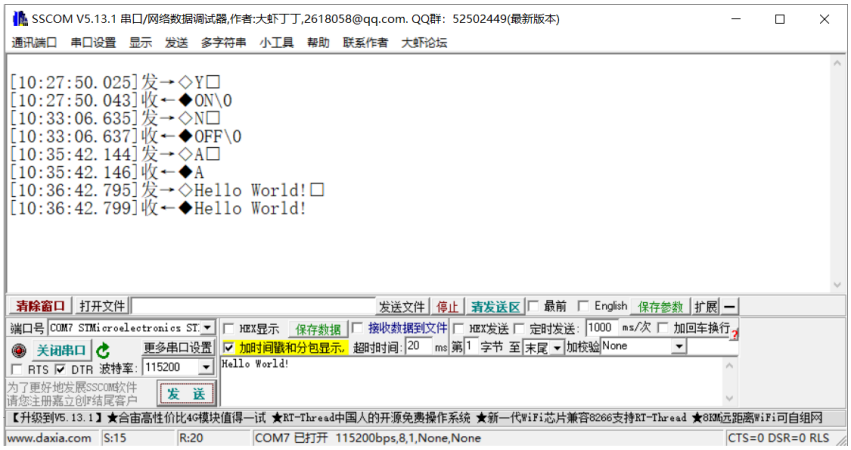

图 14：发送字符串“Hello World!”

综上，需求满足。

（二）定时器：利用定时器完成 LED 灯的闪烁、利用定时器的 PWM 输出模式，实现 LED 灯渐变（呼吸灯）

1. 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：

- 创建定时器句柄和定时器初始化函数，如图 15 所示：

```
11  TIM_HandleTypeDef TIM3_Handler;
12
13
14  void TIM3_Init(uint16_t arr,uint16_t psc)
15  {
16      TIM3_Handler.Instance = TIM3;
17      TIM3_Handler.Init.Prescaler = psc;
18      TIM3_Handler.Init.CounterMode = TIM_COUNTERMODE_UP;
19      TIM3_Handler.Init.Period = arr;
20      TIM3_Handler.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
21      HAL_TIM_Base_Init(&TIM3_Handler);
22      HAL_TIM_Base_Start_IT(&TIM3_Handler);
23  }
24
```

图 15：创建定时器句柄和初始化函数

- 初始化定时器中断相关函数，如图 16 所示：

```c
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
{
    if(htim->Instance==TIM3)
    {
      __HAL_RCC_TIM3_CLK_ENABLE();
      HAL_NVIC_SetPriority(TIM3_IRQn,1,3);
      HAL_NVIC_EnableIRQ(TIM3_IRQn);
    }
}

void TIM3_IRQHandler(void)
{
    HAL_TIM_IRQHandler(&TIM3_Handler);
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim==(&TIM3_Handler)){
      LED1_GPIO_CLK_ENABLE();
      GPIO_InitTypeDef GPIO_InitStructForLed1;
      GPIO_InitStructForLed1.Pin = LED1_PIN;
      GPIO_InitStructForLed1.Mode = GPIO_MODE_OUTPUT_PP;
      GPIO_InitStructForLed1.Pull = GPIO_NOPULL;
      GPIO_InitStructForLed1.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
      HAL_GPIO_Init(LED1_GPIO_PORT, &GPIO_InitStructForLed1);
      HAL_GPIO_WritePin(LED1_GPIO_PORT, LED1_PIN, 1-HAL_GPIO_ReadPin(LED1_GPIO_PORT,LED1_PIN));
    }
}
```

图 16：定时器中断相关函数

- 快闪主函数程序如图 17 所示：

  快闪时$T = 1s = \frac{((arr+1)*(psc+1))}{Ft}$，其中$Ft = 200MHz$，可取$arr = 9999, psc = 19999$。

```c
//------------------------------------------------------------
  //输入代码部分

  //T = 1s
  TIM3_Init(9999,19999);

//------------------------------------------------------------
```

图 17：快闪 T=1s

快闪视频见附件。

- 慢闪主函数程序如图 18 所示：

  慢闪时$T = 5s = \frac{((arr+1)*(psc+1))}{Ft}$，其中$Ft = 200MHz$，可取$arr = 49999, psc = 19999$。

```c
//------------------------------------------------------------
  //输入代码部分
/*
  //T = 1s
  TIM3_Init(9999,19999);
*/

  //T = 5s
  TIM3_Init(49999,19999);

```

图 18：慢闪 T=5s

慢闪视频附件。

2. 完成 Nucleo 板的底板 **LED1（PE11，TIM1_CH2）**呼吸灯实验：

- 创建定时器句柄和定时器 1 通道 2 句柄，如图 19：

```
12  TIM_HandleTypeDef TIM1_Handler;
13  TIM_OC_InitTypeDef TIM1_CH2Handler;
```

图 19：创建定时器句柄和通道

- 编写初始化 TIM1 的 PWM 函数，如图 20 所示：

```
15  void TIM1_PWM_Init(uint16_t arr,uint16_t psc)
16  {
17      TIM1_Handler.Instance = TIM1;
18      TIM1_Handler.Init.Prescaler = psc;
19      TIM1_Handler.Init.CounterMode = TIM_COUNTERMODE_UP;
20      TIM1_Handler.Init.Period = arr;
21      TIM1_Handler.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
22      HAL_TIM_Base_Init(&TIM1_Handler);
23
24      TIM1_CH2Handler.OCMode=TIM_OCMODE_PWM1;
25      TIM1_CH2Handler.Pulse=arr/2;
26      TIM1_CH2Handler.OCPolarity=TIM_OCPOLARITY_LOW;
27      HAL_TIM_PWM_ConfigChannel(&TIM1_Handler,&TIM1_CH2Handler,TIM_CHANNEL_2);
28      HAL_TIM_PWM_Start(&TIM1_Handler,TIM_CHANNEL_2);
29  }
```

图 20：初始化 TIM1 的 PWM 函数

- 编写定时器底层驱动，如图 21 所示：

```
31  void HAL_TIM_PWM_MspInit(TIM_HandleTypeDef *htim)
32  {
33      GPIO_InitTypeDef GPIO_Initure;
34      __HAL_RCC_TIM1_CLK_ENABLE();
35      __HAL_RCC_GPIOE_CLK_ENABLE(); //PE11
36
37      GPIO_Initure.Pin=GPIO_PIN_11;  //PE11
38      GPIO_Initure.Mode=GPIO_MODE_AF_PP;
39      GPIO_Initure.Pull=GPIO_PULLUP;
40      GPIO_Initure.Speed=GPIO_SPEED_FREQ_VERY_HIGH;
41      GPIO_Initure.Alternate=GPIO_AF1_TIM1;
42      HAL_GPIO_Init(GPIOE,&GPIO_Initure);
43
44  }
```

图 21：定时器底层驱动

- 设置 TIM1 通道 2 的占空比函数，如图 22 所示：

```
46  void TIM_SetTIM1Compare2(uint32_t compare)
47  {
48      TIM1->CCR2=compare;
49  }
```

图 22：设置占空比函数

- 主函数编写：

```
102  //输入代码部分
103  HAL_TIM_PWM_MspInit(&TIM1_Handler);
104  TIM1_PWM_Init(255,199);
105  int i = 0;
106  while(1){
107    if(i <= 255){
108      TIM_SetTIM1Compare2(i);
109    }
110    else if(i <= 510){
111      TIM_SetTIM1Compare2(510-i);
112    }
113    else
114    {
115      i = 0;
116    }
117    HAL_Delay(10);
118    i = i+1;
119  }
```

图 23：主函数

这里 103 行直接调用了 HAL_TIM_PWM_MspInit 且程序能正常运行，视频见附件，按理说应该使用 HAL_TIM_PWM_Init，但因为实验平台在队友手里，不想麻烦他再跑一遍，因此没有验证 HAL_TIM_PWM_Init 函数是否可行。

（三）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，从而控制底板 LED1 亮度

- 初始化 ADC 并创建 ADC 句柄：

```
53  ADC_HandleTypeDef ADC3_Handler;
54
55  void MY_ADC_Init(void){
56      ADC3_Handler.Instance=ADC3;
57      ADC3_Handler.Init.ClockPrescaler=ADC_CLOCK_SYNC_PCLK_DIV4;
58      ADC3_Handler.Init.Resolution=ADC_RESOLUTION_16B;
59      ADC3_Handler.Init.ScanConvMode=DISABLE;
60      ADC3_Handler.Init.EOCSelection=ADC_EOC_SINGLE_CONV;
61      ADC3_Handler.Init.LowPowerAutoWait=DISABLE;
62      ADC3_Handler.Init.ContinuousConvMode=DISABLE;
63      ADC3_Handler.Init.NbrOfConversion=1;
64      ADC3_Handler.Init.DiscontinuousConvMode=DISABLE;
65      ADC3_Handler.Init.NbrOfDiscConversion=0;
66      ADC3_Handler.Init.ExternalTrigConv=ADC_SOFTWARE_START;
67      ADC3_Handler.Init.ExternalTrigConvEdge=ADC_EXTERNALTRIGCONVEDGE_NONE;
68  //    ADC3_Handler.Init.BoostMode=ENABLE;
69      ADC3_Handler.Init.Overrun=ADC_OVR_DATA_OVERWRITTEN;
70      ADC3_Handler.Init.OversamplingMode=DISABLE;
71      ADC3_Handler.Init.ConversionDataManagement=ADC_CONVERSIONDATA_DR;
72      HAL_ADC_Init(&ADC3_Handler);
73
74      HAL_ADCEx_Calibration_Start(&ADC3_Handler,ADC_CALIB_OFFSET,ADC_SINGLE_ENDED);
75  }
```

图 24：初始化 ADC

- 编写 ADC 底层驱动：

```
77  void HAL_ADC_MspInit(ADC_HandleTypeDef *hadc){
78    GPIO_InitTypeDef GPIO_Initure;
79    __HAL_RCC_ADC3_CLK_ENABLE();
80    __HAL_RCC_GPIOF_CLK_ENABLE();
81    __HAL_RCC_ADC_CONFIG(RCC_ADCCLKSOURCE_CLKP);
82
83    GPIO_Initure.Pin=GPIO_PIN_6;
84    GPIO_Initure.Mode=GPIO_MODE_ANALOG;
85    GPIO_Initure.Pull=GPIO_NOPULL;
86    HAL_GPIO_Init(GPIOA,&GPIO_Initure);
87  }
88
```

图 25：编写 ADC 底层驱动

- 获取 ADC 值的函数：

```
int16_t Get_Adc(int32_t ch){
  ADC_ChannelConfTypeDef ADC3_ChanConf;
  ADC3_ChanConf.Channel=ch;
  ADC3_ChanConf.Rank=ADC_REGULAR_RANK_1;
  ADC3_ChanConf.SamplingTime=ADC_SAMPLETIME_64CYCLES_5;
  ADC3_ChanConf.SingleDiff=ADC_SINGLE_ENDED;
  ADC3_ChanConf.OffsetNumber=ADC_OFFSET_NONE;
  ADC3_ChanConf.Offset=0;
  HAL_ADC_ConfigChannel(&ADC3_Handler,&ADC3_ChanConf);

  HAL_ADC_Start(&ADC3_Handler);

  HAL_ADC_PollForConversion(&ADC3_Handler,10);
  return(int16_t)HAL_ADC_GetValue(&ADC3_Handler);
}
```

图 26：单次获取 ADC 值

- 获取指定通道的转换值，取 times 次，然后平均，后续主函数调用时 times 取 5：

```
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;
}
```

图 27：获得指定通道的转换值

- 主函数代码如图 28 所示，用手慢慢遮挡光敏电阻，此时光强减小，则底板 LED1 也越来越暗，视频见附件。

```
  //ADC
    int times=5;
    HAL_TIM_PWM_MspInit(&TIM1_Handler);
    HAL_ADC_Init(&ADC3_Handler);
    MY_ADC_Init();
    TIM1_PWM_Init(65535,19);
    int adcGet=0;
    while(1){
      adcGet=Get_Adc_Average(ADC_CHANNEL_8,times);
      TIM_SetTIM1Compare2(adcGet);
      HAL_Delay(500);
  }
```