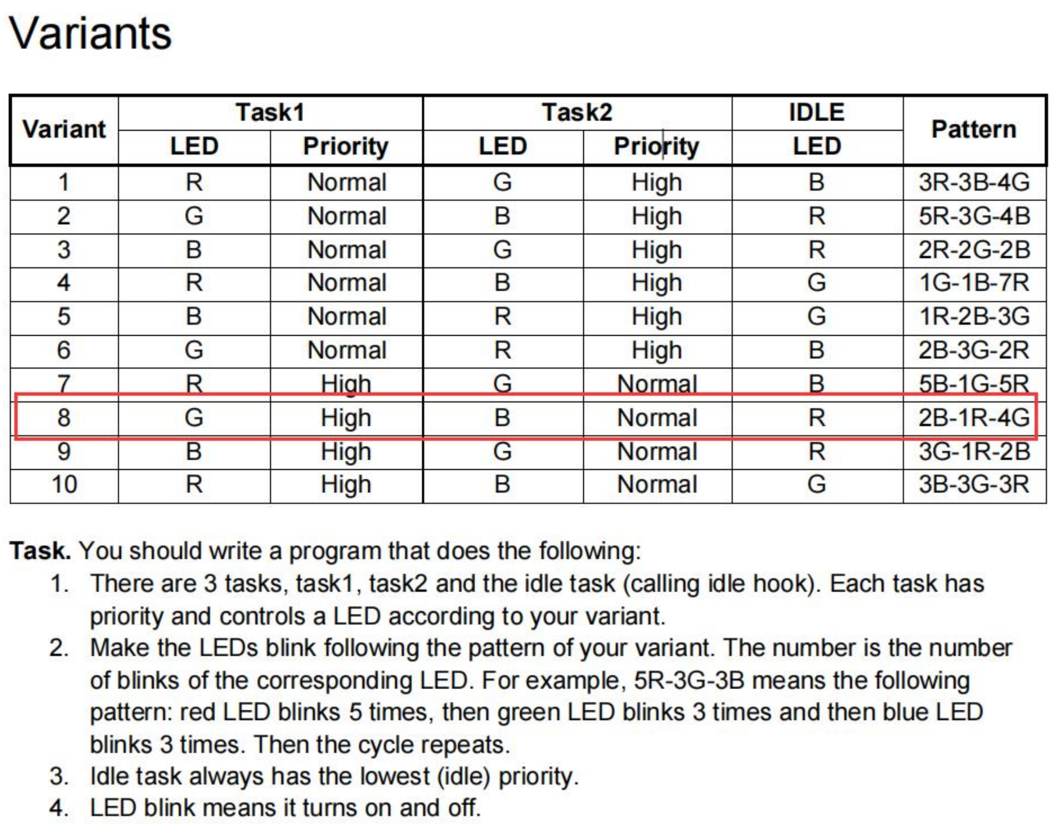
# Embedded Systems Design (2022)

# Report on LAB 2

- Student: JUNYAN, YANG（杨钧彦）
- Student Number: 212320028
- Date: 23/09/2022

# 1. Task Statement



## Variants

| Variant | Task1 | | Task2 | | IDLE | Pattern |
|---|---|---|---|---|---|---|
| | LED | Priority | LED | Priority | LED | |
| 1 | R | Normal | G | High | B | 3R-3B-4G |
| 2 | G | Normal | B | High | R | 5R-3G-4B |
| 3 | B | Normal | G | High | R | 2R-2G-2B |
| 4 | R | Normal | B | High | G | 1G-1B-7R |
| 5 | B | Normal | R | High | G | 1R-2B-3G |
| 6 | G | Normal | R | High | B | 2B-3G-2R |
| 7 | R | High | G | Normal | B | 5B-1G-5R |
| 8 | G | High | B | Normal | R | 2B-1R-4G |
| 9 | B | High | G | Normal | R | 3G-1R-2B |
| 10 | R | High | B | Normal | G | 3B-3G-3R |

**Task.** You should write a program that does the following:

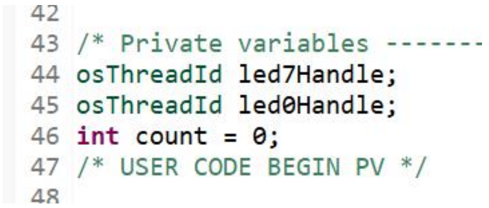
1. There are 3 tasks, task1, task2 and the idle task (calling idle hook). Each task has priority and controls a LED according to your variant.
2. Make the LEDs blink following the pattern of your variant. The number is the number of blinks of the corresponding LED. For example, 5R-3G-3B means the following pattern: red LED blinks 5 times, then green LED blinks 3 times and then blue LED blinks 3 times. Then the cycle repeats.
3. Idle task always has the lowest (idle) priority.
4. LED blink means it turns on and off.

# 2. Environment：

Win10, STM32CubeIDE

# 3. Screenshot for lab1：

### main.c

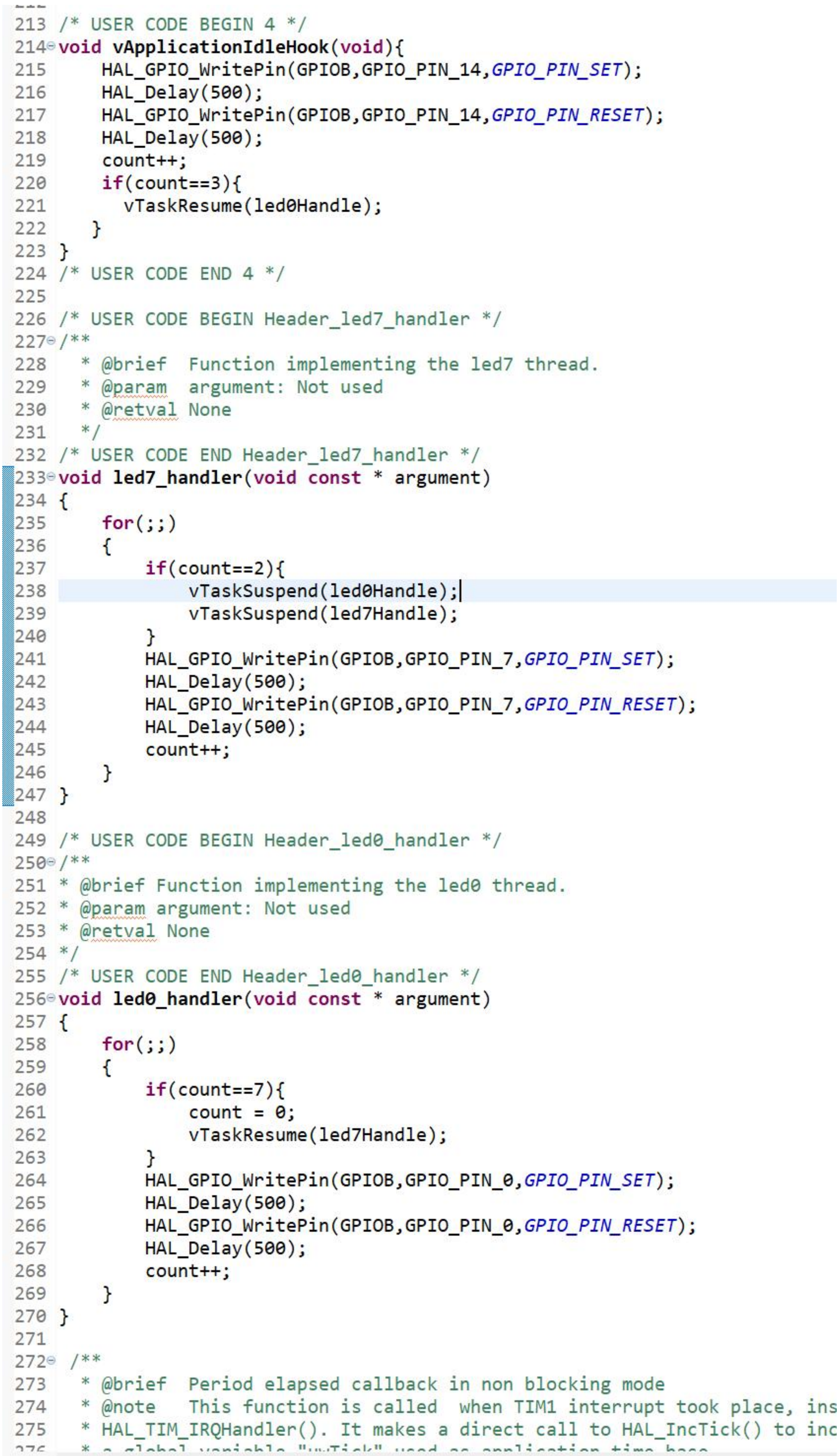


```

/* Private variables -------
osThreadId led7Handle;
osThreadId led0Handle;
int count = 0;
/* USER CODE BEGIN PV */

```

I use variant ‘count’ to record time.



```c
/* USER CODE BEGIN 4 */
void vApplicationIdleHook(void){
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,GPIO_PIN_SET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,GPIO_PIN_RESET);
    HAL_Delay(500);
    count++;
    if(count==3){
      vTaskResume(led0Handle);
   }
}
/* USER CODE END 4 */

/* USER CODE BEGIN Header_led7_handler */
/**
  * @brief  Function implementing the led7 thread.
  * @param  argument: Not used
  * @retval None
  */
/* USER CODE END Header_led7_handler */
void led7_handler(void const * argument)
{
    for(;;)
    {
        if(count==2){
            vTaskSuspend(led0Handle);
            vTaskSuspend(led7Handle);
        }
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,GPIO_PIN_SET);
        HAL_Delay(500);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,GPIO_PIN_RESET);
        HAL_Delay(500);
        count++;
    }
}

/* USER CODE BEGIN Header_led0_handler */
/**
* @brief Function implementing the led0 thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_led0_handler */
void led0_handler(void const * argument)
{
    for(;;)
    {
        if(count==7){
            count = 0;
            vTaskResume(led7Handle);
        }
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);
        HAL_Delay(500);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
        HAL_Delay(500);
        count++;
    }
}

 /**
  * @brief  Period elapsed callback in non blocking mode
  * @note   This function is called  when TIM1 interrupt took place, ins
  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to inc
  * a global variable "uwTick" used as application time base
```

And vTaskResume and vTaskSuspend to control state of three functions.