Embedded Systems Design (2022)

Report on LAB 4

- Student: JUNYAN, YANG（杨钧彦）
- Student Number: 212320028

- Date: 30/09/2022
- Yandex Link: https://disk.yandex.ru/i/O7YzbpGi9WcweQ

1. Task Statement

Variants

Variant	Task1 LED	Task2 LED	Task3 LED	Pattern
1	R	G	B	3R-3B-4G
2	G	B	R	5R-3G-4B
3	B	G	R	2R-2G-2B
4	R	B	G	1G-1B-7R
5	B	R	G	1R-2B-3G
6	G	R	B	2B-3G-2R
7	R	G	B	5B-1G-5R
8	G	B	R	2B-1R-4G
9	B	G	R	3G-1R-2B
10	R	B	G	3B-3G-3R

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

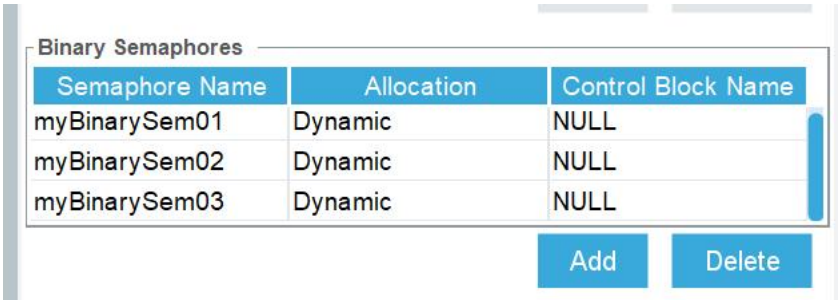
1. There are 3 tasks, task1, task2 and task3. Each task has NORMAL priority and controls a LED according to your variant.
2. Make the LEDs blink following the pattern of your variant. The number is the number of blinks of the corresponding LED. For example, 5R-3G-3B means the following pattern: red LED blinks 5 times, then green LED blinks 3 times and then blue LED blinks 3 times. Then the cycle repeats.
3. LED blink means it turns on and off.
4. Synchronization must be implemented using semaphores.

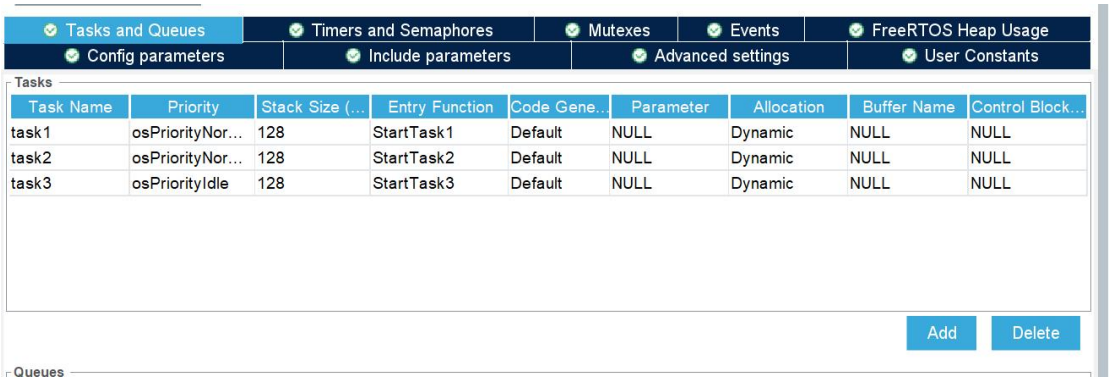
2. Environment：

Win10, STM32CubeIDE

3. Screenshot for lab4：

FreeRTOS





main.c

```
/* Private variables ---------------------------
osThreadId task1Handle;
osThreadId task2Handle;
osThreadId task3Handle;
osSemaphoreId myBinarySem01Handle;
osSemaphoreId myBinarySem02Handle;
osSemaphoreId myBinarySem03Handle;
/* USER CODE BEGIN PV */
```

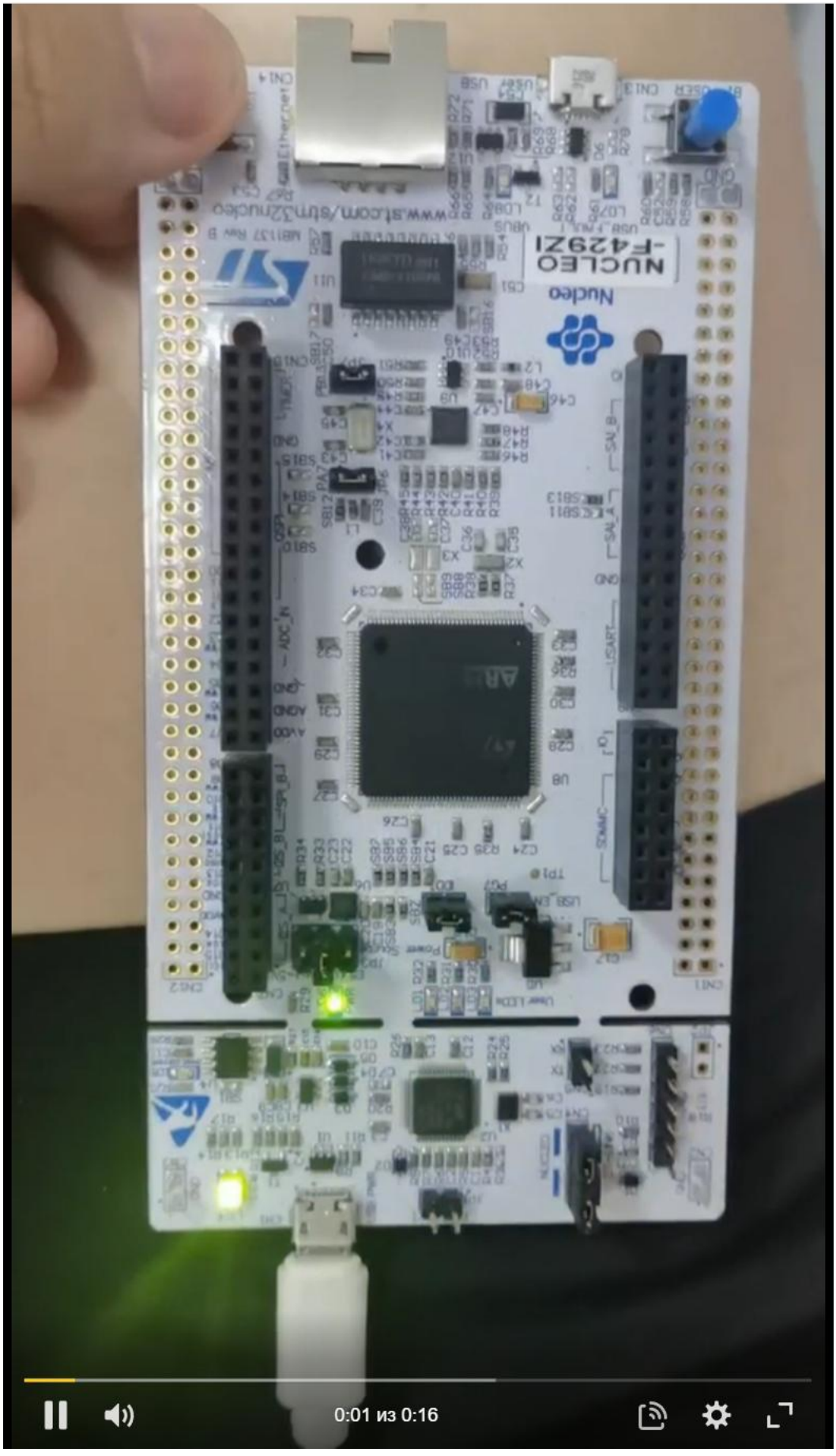
```
/* USER CODE BEGIN RTOS_SEMAPHORES */
/* add semaphores, ... */
osSemaphoreWait(myBinarySem01Handle,0);
osSemaphoreWait(myBinarySem02Handle,0);
osSemaphoreWait(myBinarySem03Handle,0);
/* USER CODE END RTOS_SEMAPHORES */
```

```c
/* USER CODE END Header_StartTask1 */
void StartTask1(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      osSemaphoreWait(myBinarySem01Handle, osWaitForever);
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
      HAL_Delay(1000);
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
      HAL_Delay(1000);
      osSemaphoreRelease(myBinarySem03Handle);
  }
  /* USER CODE END 5 */
}

/* USER CODE BEGIN Header_StartTask2 */
/**
* @brief Function implementing the task2 thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartTask2 */
void StartTask2(void const * argument)
{
  /* USER CODE BEGIN StartTask2 */
  /* Infinite loop */
  for(;;)
  {
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
      HAL_Delay(500);
      osSemaphoreRelease(myBinarySem01Handle);
      osSemaphoreWait(myBinarySem02Handle, osWaitForever);
  }
  /* USER CODE END StartTask2 */
}

/* USER CODE BEGIN Header_StartTask3 */
/**
* @brief Function implementing the task3 thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartTask3 */
void StartTask3(void const * argument)
{
  /* USER CODE BEGIN StartTask3 */
  /* Infinite loop */
  for(;;)
  {
      osSemaphoreWait(myBinarySem03Handle, osWaitForever);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
      HAL_Delay(500);
      osSemaphoreRelease(myBinarySem02Handle);
  }
  /* USER CODE END StartTask3 */
}

 /**
  * @brief  Period elapsed callback in non blocking mode
  * @note   This function is called  when TIM1 interrupt took p
  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick
```


NUCLEO
-F429ZI
Nucleo
0:01 из 0:16