

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	Task2	Task3	Pattern
	LED	LED	LED	
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

Binary Semaphores		
Semaphore Name	Allocation	Control Block Name
myBinarySem01	Dynamic	NULL
myBinarySem02	Dynamic	NULL
myBinarySem03	Dynamic	NULL
		Add
		Delete

Tasks and Queues

Timers and Semaphores

Mutexes

Events

FreeRTOS Heap Usage

Config parameters

Include parameters

Advanced settings

User Constants

Tasks

Task Name	Priority	Stack Size (...)	Entry Function	Code Gene...	Parameter	Allocation	Buffer Name	Control Block...
task1	osPriorityNor...	128	StartTask1	Default	NULL	Dynamic	NULL	NULL
task2	osPriorityNor...	128	StartTask2	Default	NULL	Dynamic	NULL	NULL
task3	osPriorityIdle	128	StartTask3	Default	NULL	Dynamic	NULL	NULL

Add

Delete

Queues

main.c

```

43 /* Private variables -----
44 osThreadId task1Handle;
45 osThreadId task2Handle;
46 osThreadId task3Handle;
47 osSemaphoreId myBinarySem01Handle;
48 osSemaphoreId myBinarySem02Handle;
49 osSemaphoreId myBinarySem03Handle;
50 /* USER CODE BEGIN PV */

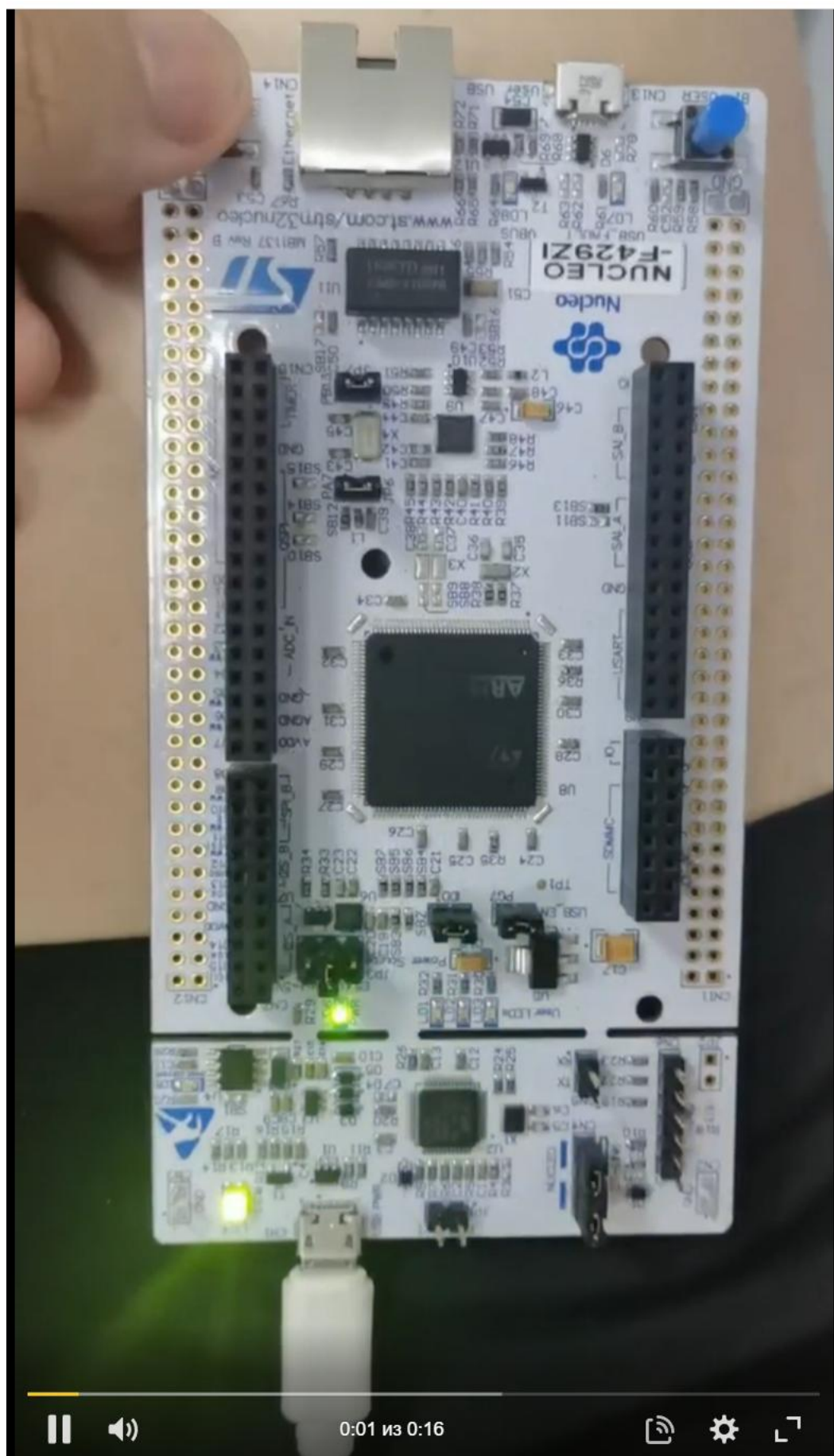
/* USER CODE BEGIN RTOS_SEMAPHORES */
/* add semaphores, ... */
osSemaphoreWait(myBinarySem01Handle,0);
osSemaphoreWait(myBinarySem02Handle,0);
osSemaphoreWait(myBinarySem03Handle,0);
/* USER CODE END RTOS_SEMAPHORES */

```

```

247 /* USER CODE END Header_StartTask1 */
248 void StartTask1(void const * argument)
249 {
250     /* USER CODE BEGIN 5 */
251     /* Infinite loop */
252     for(;;)
253     {
254         osSemaphoreWait(myBinarySem01Handle, osWaitForever);
255         HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
256         HAL_Delay(1000);
257         HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
258         HAL_Delay(1000);
259         osSemaphoreRelease(myBinarySem03Handle);
260     }
261     /* USER CODE END 5 */
262 }
263
264 /* USER CODE BEGIN Header_StartTask2 */
265 /**
266  * @brief Function implementing the task2 thread.
267  * @param argument: Not used
268  * @retval None
269  */
270 /* USER CODE END Header_StartTask2 */
271 void StartTask2(void const * argument)
272 {
273     /* USER CODE BEGIN StartTask2 */
274     /* Infinite loop */
275     for(;;)
276     {
277         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
278         HAL_Delay(500);
279         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
280         HAL_Delay(500);
281         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
282         HAL_Delay(500);
283         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
284         HAL_Delay(500);
285         osSemaphoreRelease(myBinarySem01Handle);
286         osSemaphoreWait(myBinarySem02Handle, osWaitForever);
287     }
288     /* USER CODE END StartTask2 */
289 }
290
291 /* USER CODE BEGIN Header_StartTask3 */
292 /**
293  * @brief Function implementing the task3 thread.
294  * @param argument: Not used
295  * @retval None
296  */
297 /* USER CODE END Header_StartTask3 */
298 void StartTask3(void const * argument)
299 {
300     /* USER CODE BEGIN StartTask3 */
301     /* Infinite loop */
302     for(;;)
303     {
304         osSemaphoreWait(myBinarySem03Handle, osWaitForever);
305         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
306         HAL_Delay(500);
307         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
308         HAL_Delay(500);
309         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
310         HAL_Delay(500);
311         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
312         HAL_Delay(500);
313         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
314         HAL_Delay(500);
315         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
316         HAL_Delay(500);
317         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
318         HAL_Delay(500);
319         HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
320         HAL_Delay(500);
321         osSemaphoreRelease(myBinarySem02Handle);
322     }
323     /* USER CODE END StartTask3 */
324 }
325
326 /**
327  * @brief Period elapsed callback in non blocking mode
328  * @note This function is called when TIM1 interrupt took p
329  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick

```



0:01 из 0:16

