Blinky With Mutex

Shinu Shaji C0761203
Stebin Yohannan C0770947
Hima Sijin C0769744
Sani Thomas C0761838

Objective: Use semaphore to run led task without overlapping

Working :

Declare a structure which store the values for led number and its corresponding initial delay

```
struct data{
        void * key; // key holds the semphore
        int led;    // led number
}dd[3];
```

Create semaphore of type xSemaphoreHandle

```
xSemaphoreHandle xSemaphore = NULL; /* make xsemaphore variable of type
xSemaphoreHandle */
xSemaphore = xSemaphoreCreateMutex();
```

Make 3 led task create function and pass dd[] of type data struct to the tasks

```
xTaskCreate(vLEDTask2, (signed char *) "vTaskLed1",
configMINIMAL_STACK_SIZE, &dd[0], (tskIDLE_PRIORITY + 1UL),
(xTaskHandle *) NULL);
```

Create led tasks which can take the void pointer and cast to the data struct type and retrive the led number and the semaphore .

```
static void vLEDTask1(void *pvParameters) {
        struct data *d = pvParameters;
        int led_n = (*d).led;
        //int led_n = 0;
        bool LedState = false;
        while (1) {
                if (xSemaphoreTake((d->key),50)){ // take the semaphore
                        Board_LED_Set(led_n, false);/*  */
                        vTaskDelay(1000);
                        Board_LED_Set(led_n, true);
                        xSemaphoreGive( (d->key) ); //give the semaphore
                        vTaskDelay(1000);
                        }
                }
        }
```

RESULT

All LED's glow separate without mixing

VIDEO

https://drive.google.com/file/d/1NRBaYrMXS3tXmAxP1FQ-RVgggBtO-90r/view?usp=sharing

Code

```
/*
* @brief FreeRTOS Blinky example
*
* @note
* Copyright(C) NXP Semiconductors, 2014
* All rights reserved.
*/
```

```
#include "board.h"
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"

/* Sets up system hardware */
struct data{
	void * key;
	int led;
}dd[3];


static void prvSetupHardware(void)
{
	SystemCoreClockUpdate();
	Board_Init();
	/* Initial LED0 state is off */
	Board_LED_Set(0, true);
	Board_LED_Set(1, true);
	Board_LED_Set(2, true);
}


/* LED1 toggle thread */
static void vLEDTask1(void *pvParameters) {
	struct data *d = pvParameters;
	int led_n = (*d).led;
	//int led_n = 0;
	bool LedState = false;
	while (1) {
		if (xSemaphoreTake((d->key),50)){
			Board_LED_Set(led_n, false);
			LedState = (bool) !LedState;
			/* About a 3Hz on/off toggle rate */
			vTaskDelay(1000);
			Board_LED_Set(led_n, true);
			//vTaskDelay(3500);
			xSemaphoreGive( (d->key) );
			vTaskDelay(1000);
```

```
                        }
                }
        }
int main(void)
{
        xSemaphoreHandle xSemaphore = NULL;
        xSemaphore = xSemaphoreCreateMutex();
        prvSetupHardware();
        /* LED1 toggle thread */
        if(xSemaphore != NULL){
                dd[0].key = xSemaphore;
                dd[0].led = 0;
                dd[1].key = xSemaphore;
                dd[1].led = 1;
                dd[2].key = xSemaphore;
                dd[2].led = 2;

                xTaskCreate(vLEDTask1, (signed char *) "vTaskLed1",
                configMINIMAL_STACK_SIZE,&dd[0], (tskIDLE_PRIORITY+1UL),
                (xTaskHandle *) NULL);


                xTaskCreate(vLEDTask1, (signed char *) "vTaskLed2",
                configMINIMAL_STACK_SIZE, &dd[1], (tskIDLE_PRIORITY + 1UL),
                (xTaskHandle *) NULL);

                xTaskCreate(vLEDTask1, (signed char *) "vTaskLed3",
                configMINIMAL_STACK_SIZE, &dd[2], (tskIDLE_PRIORITY + 1UL),
                (xTaskHandle *) NULL);
                /* Start the scheduler */
                vTaskStartScheduler();
                }
        /* Should never arrive here */
        return 1;
        }


/**
```

```
* @}
*/
```