Lab(PROJ_1-->RGB blink without mixing colours)

ESE 3025: EmbeddedReal Time Operating Systems

Lambton College in Toronto

Instructor: Takis Zourntos

STUDENT NAME & ID:

Vishal Hasrajani(C0761544)

Parth Patel(C0764929)

Goutham Reddy Alugubelly(C0747981)

Ratnajahnavi rebbapragada(C0762196)

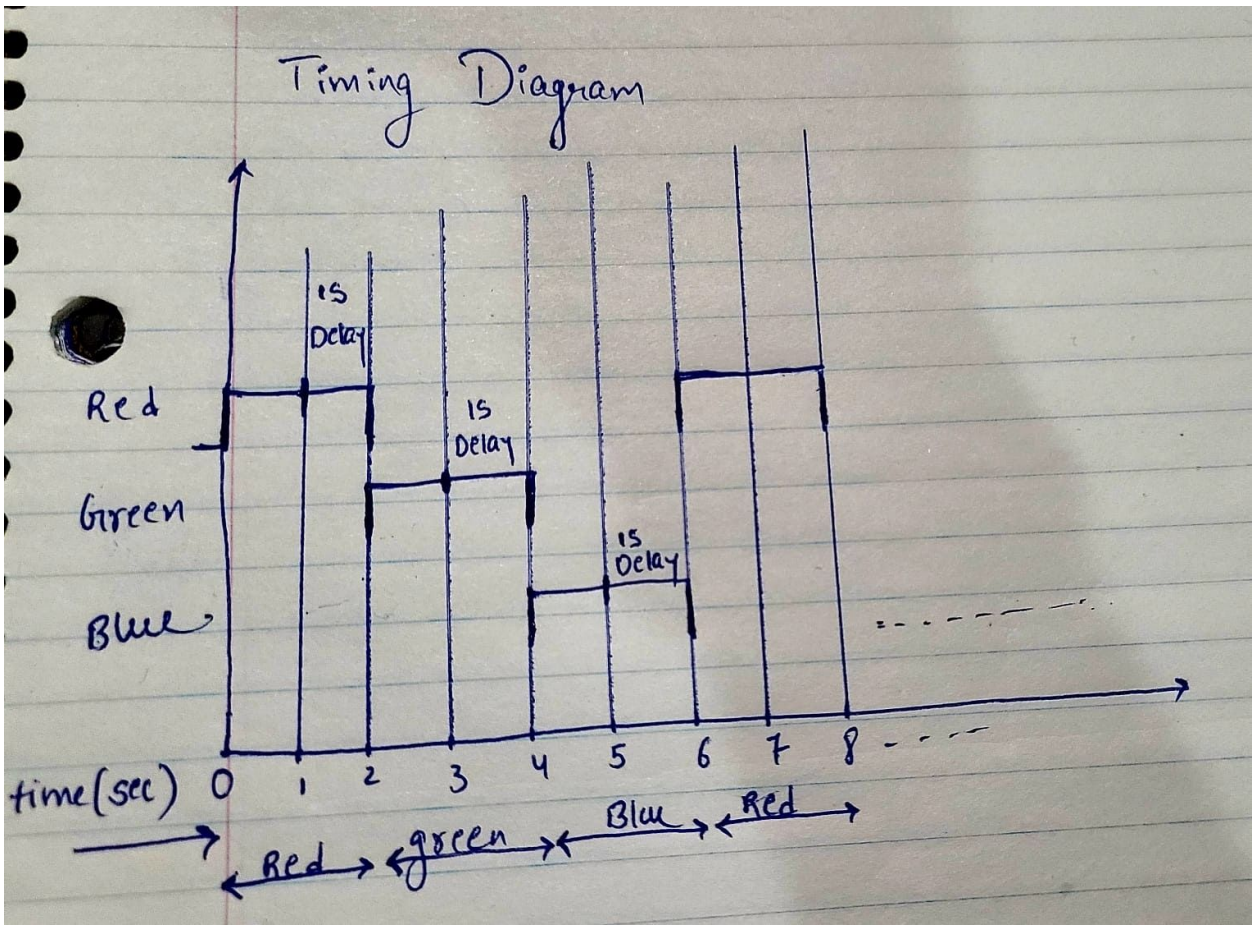
INTRODUCTION :

In this lab,we will add three tasks each for red,green and blue LEDs.we will further describe the functioning of RGB LEDs with delays using a Timing diagram.When any task runs,it turns the LED on,delays,then turns the LED off.When the task is blocked,it should be on delay only(we can use vTaskDelay() for the delay).Finally we will show the working project through the video provided on youtube.

DESCRIPTION :

Let's start with the timing diagram to show the active state and blocked state of LEDs.

Timing diagram is given below :



As you can see in the diagram,the RED LED is ON from 0-2sec with a delay of 1s while other LEDs are blocked (here green led is blocked having delay od 2sec and blue led id blocked having only delay of 4sec)

Similarly,when Green LED or Blue LED is in ON state ,other 2 LEDs are in blocked state with only delay.
Green LEd is in ON-state from 2-4sec
Blue LEd is in ON-state from 4-6sec

The main thing to notice here is that there will be no mixing of colours as each LED is having separate defined slots with a delay .

Now **freeRtosblinky-RGB.c** code will clear your idea regarding delays ,ON-state and Blocked state of LEDs.

Explanation of code :

1)

```
/* Sets up system hardware */
static void prvSetupHardware(void)
{
	SystemCoreClockUpdate();
	Board_Init();

	/* Initial red LED,green LED and blue LED states are
off */
	Board_LED_Set(red, false);
	Board_LED_Set(green, false);
	Board_LED_Set(blue, false);
}
```

Here this function **prvSetupHardware(void)** sets up the system hardware initially.
Then **SystemCoreUpdate()** is used to update the cpu core clock rate.

Board_Init() is used to set up and initialize all required blocks and functions related to board hardware.

Then we set LEDS(that are red,green and blue) to OFF state initially.

2) LED1(task-1) ,LED2(task-2) and LED3(task-3) threads.

```
/* LED1 toggle thread */
static void vLEDTask1(void *pvParameters) {



    while (1) {
        Board_LED_Set(red, true); //red led ON
        vTaskDelay(configTICK_RATE_HZ );//1sec delay
        Board_LED_Set(red, false);//red led OFF
        vTaskDelay(5*configTICK_RATE_HZ );//5sec delay


    }
}

/* LED2 toggle thread */
static void vLEDTask2(void *pvParameters) {

    vTaskDelay(2*configTICK_RATE_HZ );//2sec delay
    while (1) {

        Board_LED_Set(green, true); //green led ON
        vTaskDelay(configTICK_RATE_HZ );//1sec delay
        Board_LED_Set(green, false); //green led OFF
        vTaskDelay(5*configTICK_RATE_HZ );//5sec delay


    }
}
```

```
static void vLEDTask3(void *pvParameters) {

    vTaskDelay(4*configTICK_RATE_HZ );//4sec delay
    while (1) {

        Board_LED_Set(blue,true); //blue led ON
        vTaskDelay(configTICK_RATE_HZ ); //1sec delay
        Board_LED_Set(blue,false);//blue led OFF
        vTaskDelay(5*configTICK_RATE_HZ );//5sec delay



    }
}
```

Here , in this code you can see that the red LED has a total delay of 6 sec and it will turn off after 1sec delay.So we can include green LED only after 1 sec delay .So we set the Green LED to ON-state after 2sec delay as mentioned in timing diagram.Now It also has delay of 1sec in between the ON and off state,so we can set the blue LED to ON state only after 3 sec delay.Finally we set the blue LED to ON state after 4sec of delay so that it doesn't interfere other LEDs.

Now you can see that the red led will start again at 6sec delay.So there will be no mismatch and all other LEDs will follow the same Blocked and active state as explained above.

3)

```
int main(void)
{
     prvSetupHardware();

     /* LED1 toggle thread */
     xTaskCreate(vLEDTask1, (signed char *) "vTaskLed1",
                    configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 3UL),
                    (xTaskHandle *) NULL);

     /* LED2 toggle thread */
     xTaskCreate(vLEDTask2, (signed char *) "vTaskLed2",
                    configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 2UL),
                    (xTaskHandle *) NULL);
     /* LED3 toggle thread */
     xTaskCreate(vLEDTask3, (signed char *) "vTaskLed3",
                         configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 1UL),
                         (xTaskHandle *) NULL);

     /* Start the scheduler */
     vTaskStartScheduler();

     /* Should never arrive here */
     return 1;
}
```

So,we are setting up the hardware and then we created 3 tasks in real time with the priority of 3,2 and 1.Highest priority is for red LED and lowest priority is for Blue LED.These threads can be controlled as per our convenience.

Further moving on ,to schedule these tasks we started the system scheduler by using the function **vTaskStartScheduler()**.

Output :

Youtube Link -----)

https://youtu.be/JfJX0SB5bx8

CONCLUSION :

Overall,we learned how can we keep the one LED in ON state while keeping other LEDs to blocked state using vTaskDelay () function .We also understood the operation using the timing diagram.And finally we executed it successfully.

APPENDIX:

freertos_blinky_RGB.c

```
/*
 * @brief FreeRTOS Blinky example
 *
 * @note
 * Copyright(C) NXP Semiconductors, 2014
 * All rights reserved.
 *
 * @par
 * Software that is described herein is for
illustrative purposes only
 * which provides customers with programming
information regarding the
 * LPC products.  This software is supplied "AS IS"
without any warranties of
 * any kind, and NXP Semiconductors and its licensor
disclaim any and
 * all warranties, express or implied, including all
implied warranties of
 * merchantability, fitness for a particular purpose
and non-infringement of
 * intellectual property rights.  NXP Semiconductors
assumes no responsibility
 * or liability for the use of the software, conveys
no license or rights under any
 * patent, copyright, mask work right, or any other
intellectual property rights in
```

```
 * or to any products. NXP Semiconductors reserves
the right to make changes
 * in the software without notification. NXP
Semiconductors also makes no
 * representation or warranty that such application
will be suitable for the
 * specified use without further testing or
modification.
 *
 * @par
 * Permission to use, copy, modify, and distribute
this software and its
 * documentation is hereby granted, under NXP
Semiconductors' and its
 * licensor's relevant copyrights in the software,
without fee, provided that it
 * is used in conjunction with NXP Semiconductors
microcontrollers.  This
 * copyright, permission, and disclaimer notice must
appear in all copies of
 * this code.
 */

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

/*******************************************************
************************
```

```
 * Private types/enumerations/variables

**********************************************************
************************/

/**********************************************************
*************************
 * Public types/enumerations/variables

**********************************************************
************************/

/**********************************************************
*************************
 * Private functions

**********************************************************
************************/

/* Sets up system hardware */
static void prvSetupHardware(void)
{
    SystemCoreClockUpdate();
    Board_Init();

    /* Initial red LED,green LED and blue LED
states are off */
    Board_LED_Set(red, false);
    Board_LED_Set(green, false);
```

```
    Board_LED_Set(blue, false);
}

/* LED1 toggle thread */
static void vLEDTask1(void *pvParameters) {



    while (1) {
        Board_LED_Set(red, true); //red led ON
        vTaskDelay(configTICK_RATE_HZ );//1sec
delay
        Board_LED_Set(red, false);//red led OFF
        vTaskDelay(5*configTICK_RATE_HZ );//5sec
delay



    }
}

/* LED2 toggle thread */
static void vLEDTask2(void *pvParameters) {

    vTaskDelay(2*configTICK_RATE_HZ );//2sec delay
    while (1) {

        Board_LED_Set(green, true); //green led ON
        vTaskDelay(configTICK_RATE_HZ );//1sec
delay
```

```
        Board_LED_Set(green, false); //green led
OFF
        vTaskDelay(5*configTICK_RATE_HZ );//5sec
delay


    }
}

static void vLEDTask3(void *pvParameters) {

    vTaskDelay(4*configTICK_RATE_HZ );//4sec delay
    while (1) {

        Board_LED_Set(blue,true); //blue led ON
        vTaskDelay(configTICK_RATE_HZ ); //1sec
delay
        Board_LED_Set(blue,false);//blue led OFF
        vTaskDelay(5*configTICK_RATE_HZ );//5sec
delay



    }
}
/* UART (or output) thread */


/*************************************************
```

```
**************************
 * Public functions

*************************************************************
************************/

/**
 * @brief   main routine for FreeRTOS blinky
example
 * @return  Nothing, function should not exit
 */
int main(void)
{
    prvSetupHardware();

    /* LED1 toggle thread */
    xTaskCreate(vLEDTask1, (signed char *)
"vTaskLed1",
                configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 3UL),
                (xTaskHandle *) NULL);

    /* LED2 toggle thread */
    xTaskCreate(vLEDTask2, (signed char *)
"vTaskLed2",
                configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 2UL),
                (xTaskHandle *) NULL);
    /* LED3 toggle thread */
```

```
    xTaskCreate(vLEDTask3, (signed char *)
"vTaskLed2",
                    configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 1UL),
                    (xTaskHandle *) NULL);

    /* Start the scheduler */
    vTaskStartScheduler();

    /* Should never arrive here */
    return 1;
}

/**
 * @}
 */
```