Wesley Kerr
Hayden Earl

Lab 1 Report - Blinky LED

Procedure

For this lab, a microcontroller was programmed to toggle an LED when a button is pressed. First, Keil needed to be set up. We installed Keil on two separate laptops, and when there were issues, we moved to the lab briefly. C code was written to implement tasks using FreeRTOS.

Results

The trickiest part of this lab was definitely getting Keil to work properly with the microcontroller and FreeRTOS. At first, Keil was not recognizing the STM header file, and (while writing this very sentence) we realized we had typed “1” instead of “L” in the file, since they look very similar in the default text editor. The file was able to compile after the switch. It also seemed that we needed to debounce it, since the LED was behaving weirdly but after switching off the debugger it functioned properly.

Figures

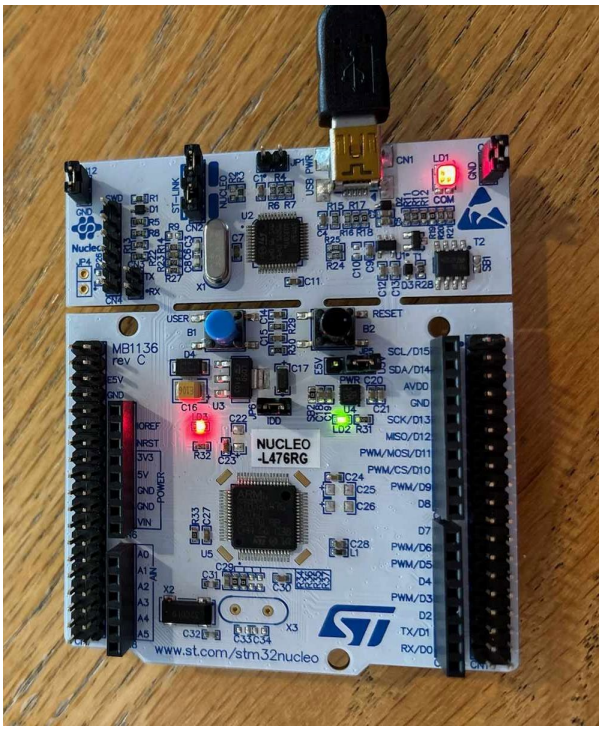

Figure 1. Microcontroller with LED lit up

Wesley Kerr
Hayden Earl

Conclusion

Using a button and LED was a great way to introduce FreeRTOS in a lab setting, including the concept of tasks. It was much easier to perform this task than writing it in Assembly code. Using FreeRTOS be useful going forward with further labs.

```c
#include <stdio.h>
#include "FreeRTOS.h"
#include "task.h"
#include <stdbool.h>
#include "stm32l476xx.h"


bool BTN_ST = false; // global button state


void vLED_Control(void *pvParameters){ // led is PA5
    while (true) {
            if(BTN_ST){
                //turn led on
                GPIOA->ODR |=0x20;
            }else{
                //turn led off
                GPIOA->ODR &=0x00;
                //GPIOA->MODER|=GPIO_MODER_MODE5_0;
            }

        // Wait for button press signal
        //ulTaskNotifyTake(pdTRUE, portMAX_DELAY);

        // Toggle LED state
        //gpio_write(LED_PIN, !gpio_read(LED_PIN));
    }
}

void vButton_Control(void *pvParameters){ // button is PC13, active low
    while (true) {
            if((GPIOC->IDR&0x2000)  ==0 && !BTN_ST){ // Poll button state
                BTN_ST=true;
            }
            else if((GPIOC->IDR&0x2000)==0){
                BTN_ST=false;
            }


        //vTaskDelay(pdMS_TO_TICKS(10)); // Polling interval
    }
}
int main(void){
    BaseType_t xLED_return;
    BaseType_t xBTN_return;

    //setup hardware
    //output for LED
    RCC->AHB2ENR|=RCC_AHB2ENR_GPIOAEN;
    GPIOA->MODER&=~GPIO_MODER_MODE5;
    GPIOA->MODER|=GPIO_MODER_MODE5_0;

    //input for BTN
    RCC->AHB2ENR|=RCC_AHB2ENR_GPIOCEN;
    GPIOC->MODER&=~GPIO_MODER_MODE13;


    //setup tasks
    xLED_return=xTaskCreate(vLED_Control, "LED_Control", configMINIMAL_STACK_SIZE + 10,
    NULL, tskIDLE_PRIORITY + 1, NULL); // need to set value to stack size
    if(xLED_return==pdPASS){
        xBTN_return=xTaskCreate(vButton_Control, "Button_Control",
        configMINIMAL_STACK_SIZE + 10,NULL, tskIDLE_PRIORITY + 1, NULL);
        if(xBTN_return==pdPASS){
            //start task scheduler
            vTaskStartScheduler();
```

```
65              }
66          }
67
68
69          for(;;);
70          return 0;
71      }
```