**REPORT LAB**

**EMBEDDED SYSTEM - CO3054**

*Group:* **CC02**

*Student:* **Dương Gia An – 1952163**

Contents

[**I.** **INTRODUCTION TO ESP32 AND ESP-IDF** 2](#_Toc117868766)

[1. After install ESP-IDF extension on VS code, I create a project with HelloWorld Template. 2](#_Toc117868767)

[2. Build the Project and connect ESP32 (COM3). 2](#_Toc117868768)

[3. Flash (UART) project code to ESP32 by Press BOOT button on ESP32 while Flashing. 3](#_Toc117868769)

[4. Press Monitor Device to see ESP execute flashed code. 5](#_Toc117868770)

[**II.** **ESP32 GPIO AND FREERTOS TASK** 6](#_Toc117868771)

[Code in file **main.c** 6](#_Toc117868772)

[Link Github: CO3054\_ES\_LAB/LAB1 at main · kinggiaan/CO3054\_ES\_LAB (github.com) 7](#_Toc117868773)

[***Explain:*** 7](#_Toc117868774)

[***Result:*** 8](#_Toc117868775)

[***Does the ESP-IDF need the vTaskStartScheduler() routine?*** 9](#_Toc117868776)

1. **FREERTOS SOFTWARE TIMER**
2. After create new Project with Template. I config “configUSE\_TIMERS” in FreeRTOSConfig.h by new file override.h

A screenshot of a computer

Description automatically generated with medium confidence

1. Initialize Global variables: 2 timer by NULL and counter by 0.

Graphical user interface, text

Description automatically generated

1. In app\_main(), I use **xTimerCreate**  for 2 timer **auto\_loader\_timer1** and **auto\_loader\_timer2.**   
   Text

   Description automatically generated
2. After creae 2 timer. I set condition to make sure them ok ( mean not **NULL**) to Start timer by **xTimerStart(timer, delay).**  
   Text

   Description automatically generated
3. In call back function for Timer. I check Timer ID by p**vTimerGetTimerID**. When a timer in 2 timer expires, I check if timer’s ID is 1 or 2 then print as requirment.   
   After they done their task, I use **xTimerStop** to stop timer.

Text

Description automatically generated

1. After 2 timer done there task, about 25 second, I check if true that 2 timer done there task.

Text

Description automatically generated

1. Result and github link for project: [HK221\_CO3054\_ES\_LAB/LAB3/LAB3\_SoftwareTimer at lab3 · kinggiaan/HK221\_CO3054\_ES\_LAB (github.com)](https://github.com/kinggiaan/HK221_CO3054_ES_LAB/tree/lab3/LAB3/LAB3_SoftwareTimer)   
    Text

   Description automatically generated
2. **WIFI**

## Code in file **main.c**

1. #include <stdio.h>
2. #include "sdkconfig.h"
3. #include "freertos/FreeRTOS.h"
4. #include "freertos/task.h"
5. #include "esp\_system.h"
6. #include "esp\_spi\_flash.h"

9. **void** print\_id(**void** \*pvParameter){
10. **while**(1){
11. **printf**("DUONG GIA AN : %d \n",1952163);
12. vTaskDelay(1000/portTICK\_PERIOD\_MS);
14. }
15. vTaskDelete(NULL);
17. }
19. **void** blinky(**void** \*pvParameter){
20. **while**(1){
21. **printf**("Press Button\n");
22. **int**  rd = **rand**() % (5000 + 1 - 0) + 0;
23. vTaskDelay(rd /portTICK\_PERIOD\_MS);
24. }
25. vTaskDelete(NULL);
26. }

29. **void** app\_main(){
30. xTaskCreate(&print\_id, "print\_id", 2048, NULL, 0, NULL);
31. xTaskCreate(&blinky, "blinky", 2048,NULL,0,NULL );
33. **for** (**int** i = 20; i >= 0; i--) {
34. **printf**("Remaing %d seconds...\n", i);
35. vTaskDelay(1000 / portTICK\_PERIOD\_MS);
36. }
37. **printf**("Restarting now.\n");
38. vTaskDelay(5000 / portTICK\_PERIOD\_MS);
39. **fflush**(stdout);
40. esp\_restart();
41. }

## Link Github: [CO3054\_ES\_LAB/LAB1 at main · kinggiaan/CO3054\_ES\_LAB (github.com)](https://github.com/kinggiaan/CO3054_ES_LAB/tree/main/LAB1)

## ***Explain:***

* + Cyclic task: void **print\_id()** is task that print my student ID every 2 seconds.
  + Acylic task: void **Blinky()** is alternated for button in GPIO in ESP32. I change to a random time to press button from 0 – 5000ms.
  + **app\_main()** will print time stamp every 1 second and restart ESP after 20 seconds.

The *priority* and *usStackDepth* of task cyclic/acylic is the same as 0 and 2048 (mean 2048\*4 bytes will be allocated for these tasks).

## ***Result:***

Graphical user interface, text

Description automatically generated

## ***Does the ESP-IDF need the vTaskStartScheduler() routine?***

No, because ESP-IDF will call vTaskStartScheduler() automatically.

|  |
| --- |
| Unlike Vanilla FreeRTOS, users must not call [**vTaskStartScheduler()**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html#_CPPv419vTaskStartSchedulerv). Instead, ESP-IDF FreeRTOS is started automatically. The entry point is a user defined void app\_main(void) function.   * Typically, users would spawn the rest of their applications task from app\_main. * The app\_main function is allowed to return at any point (i.e., before the application terminates). * The app\_main function is called from the main task.   The main task is one of multiple tasks that are automatically spawned by ESP-IDF during startup.[[1]](#footnote-1) |

1. [FreeRTOS - ESP32 - — ESP-IDF Programming Guide latest documentation (espressif.com)](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html) [↑](#footnote-ref-1)