

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



- Embedded System Lab 05 -

---

# FreeRTOS Software Timer

---

Giảng viên hướng dẫn: Vũ Trọng Thiên  
Sinh viên thực hiện: Nguyễn Hữu Hiếu - 2013149.  
Lê Bá Dũng - 2012863.

Tp. Hồ Chí Minh, 05/12/2023



## Mục lục

<b>Thành viên</b>	<b>ii</b>
<b>Danh sách hình ảnh</b>	<b>iii</b>
<b>Danh sách chương trình</b>	<b>iii</b>
<b>1 Giới thiệu</b>	<b>1</b>
1.1 Nội dung hiện thực . . . . .	1
1.2 Thiết bị và môi trường . . . . .	1
1.2.1 Thiết bị . . . . .	1
1.2.2 Môi trường . . . . .	2
<b>2 Hiện thực phần bắt buộc</b>	<b>3</b>
2.1 Cấu hình chung . . . . .	3
2.2 Cấu hình Timer . . . . .	4
2.3 Mô tả hàm . . . . .	4
2.4 Kết quả hiện thực . . . . .	5



## Thành viên

No.	Họ & Tên	MSSV	Mức độ hoàn thành
1	Nguyễn Hữu Hiếu	2013149	100%
2	Lê Bá Dũng	2012863	100%



## Danh sách hình vẽ

1	Kit ESP32-DevKitC . . . . .	1
2	Cấu hình bộ nhớ flash . . . . .	3
3	Menu cấu hình cho ESP32 . . . . .	3
4	Output của chương trình . . . . .	5

## Danh sách chương trình

### Listings

1	Call back function . . . . .	4
2	Khởi tạo timer . . . . .	5

# 1 Giới thiệu

## 1.1 Nội dung hiện thực

Thiết kế và triển khai các ví dụ cho từng lịch trình sau:

- Prioritized Pre-emptive Scheduling with Time Slicing.
- Prioritized Pre-emptive Scheduling (without Time Slicing)
- Co-operative Scheduling

**Extra exercise:** Sử dụng idle task hook để điều khiển CPU utilization. ESP32 dùng dual-core

## 1.2 Thiết bị và môi trường

### 1.2.1 Thiết bị

#### 1. ESP32-DevKitC V4 Module WiFi Bluetooth 2.4GHz

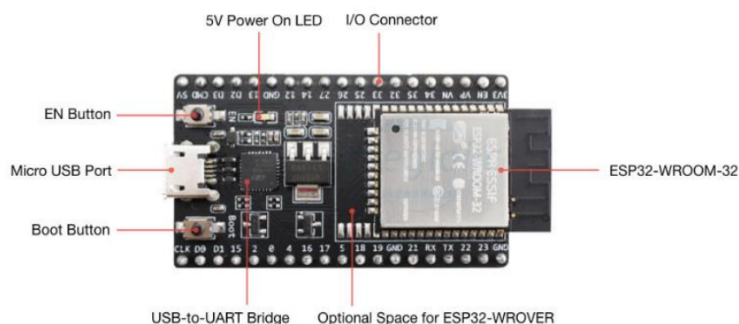
ESP32-DevKitC V4 là bo mạch phát triển dựa trên ESP32 cỡ nhỏ do Espressif sản xuất. Hầu hết các chân I/O được chia thành các đầu chân cắm ở cả hai bên để dễ dàng giao tiếp. Các nhà phát triển có thể kết nối các thiết bị ngoại vi bằng dây nhảy hoặc gắn ESP32-DevKitC V4 trên bảng mạch khung.

#### Tùy chọn cung cấp điện

- Đầu nối micro USB cho giao tiếp và nguồn PC.
- Cấp nguồn qua đầu nối: 5 V hoặc 3,3 V.

**Lưu ý:** Nguồn điện phải được cung cấp bằng một và chỉ một trong các tùy chọn ở trên, nếu không bo mạch và/hoặc nguồn điện có thể bị hỏng.

#### 2. Micro-B Cable Sử dụng để cấp nguồn điện và nạp firmware cho thiết bị.



Hình 1: Kit ESP32-DevKitC



### 1.2.2 Môi trường

- ESP-IDF Version 5.11
- Python Version 3.9.13
- Hệ điều hành máy tính: Window 11
- IDE: VSCode
- Extension: Espressif IDF

## 2 Hiện thực phần bắt buộc

### 2.1 Cấu hình chung

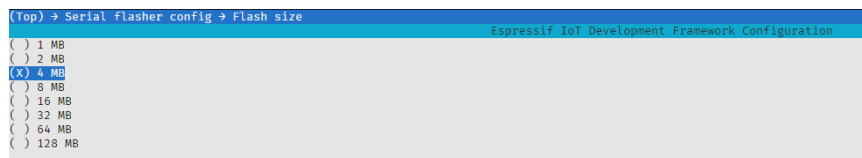
Để cấu hình cơ bản cho ESP32, sử dụng các lệnh sau:

```
1 // target config
2 idf.py set-target esp32
3 // Open menu config
4 idf.py menuconfig
```

**Cấu hình flash memory cho ESP32.**

Với module ESP32-Devkit, bộ nhớ flash của thiết bị là 4MB. *Hình 2*

Serial Flash → Flash config → Flash size

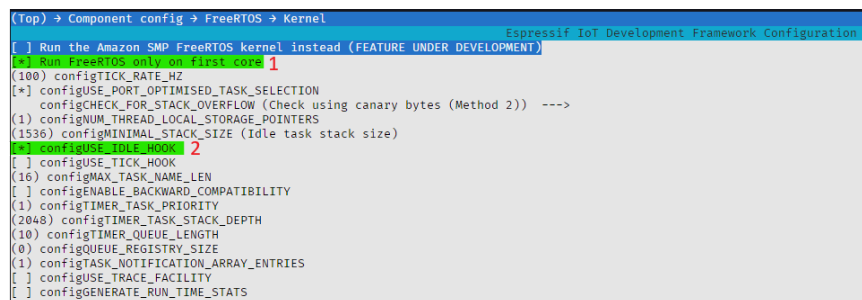


Hình 2: Cấu hình bộ nhớ flash

**Cấu hình FreeRTOS service cho ESP32:**

Component config → FreeRTOS → Kernel

1. Cấu hình FreeRTOS chạy trên một core duy nhất. (1) trong *Hình 3*
2. Cấu hình idle hook cho ESP32. (2) trong *Hình 3*



Hình 3: Menu cấu hình cho ESP32

## 2.2 Cấu hình Timer

Để sử dụng timer service, chúng ta cần config những thông số nâng cao liên quan tới dịch vụ FreeRTOS. Hằng số này nằm trong file `timers.h`, theo đường dẫn sau:

`$your_dir\esp-idf\components\freertos\esp_additions\include\freertos\FreeRTOSConfig.h`

- `configUSE_TIMERS` : Hằng số timers. gán cho hằng số này giá trị 1.

## 2.3 Mô tả hàm

**a, Call back function** Trong bài lab này, nhóm hiện thực 1 "call back function" để sử dụng cho chung 2 timer:

Nhóm sử dụng `pvTimerGetTimerID(xTimer)` để xác định timer.

```
1 // Call back function
2 void ATimerCallback(TimerHandle_t xTimer)
3 {
4     uint32_t taskID = (uint32_t)pvTimerGetTimerID(xTimer);
5
6     if (taskID == ahihi_ID)
7     {
8         ulCounter_0 += 1;
9         printf("At %ld ms, count \"ahihi\" %ld times\n", pdTICKS_TO_MS
10             (xTaskGetTickCount()), ulCounter_0);
11         if (ulCounter_0 >= MAX_COUNT_AHIHI)
12         {
13             ESP_LOGI(TAG, "At %ld ms, STOP count \"ahihi\"",
14                 pdTICKS_TO_MS(xTaskGetTickCount()));
15             xTimerStop(xTimer, 0);
16         }
17     }
18     else if (taskID == ihaha_ID)
19     {
20         ulCounter_1 += 1;
21         printf("At %ld ms, count \"ihaha\" %ld times\n", pdTICKS_TO_MS
22             (xTaskGetTickCount()), ulCounter_1);
23         if (ulCounter_1 >= MAX_COUNT_IHAHA)
24         {
25             ESP_LOGI(TAG, "At %ld ms, STOP count \"ihaha\"",
26                 pdTICKS_TO_MS(xTaskGetTickCount()));
27             xTimerStop(xTimer, 0);
28         }
29     }
30 }
```

Function 1: Call back function



### b, Khởi tạo timer

```

1 xTimer_ahihi = xTimerCreate("vTimer_0", pdMS_TO_TICKS(2000),
  pdTRUE, (void *)ahihi_ID, ATimerCallback);
2 xTimer_ihaha = xTimerCreate("vTimer_1", pdMS_TO_TICKS(3000),
  pdTRUE, (void *)ihaha_ID, ATimerCallback);
3 // Check if timer is available
4 if (xTimerStart(xTimer_ahihi, 0) != pdPASS || xTimerStart(
  xTimer_ihaha, 0) != pdPASS)
5 {
6     ESP_LOGI(TAG, "At %ld, timer start failed", pdTICKS_TO_MS(
  xTaskGetTickCount()));
7 }

```

Function 2: Khởi tạo timer

## 2.4 Kết quả hiện thực

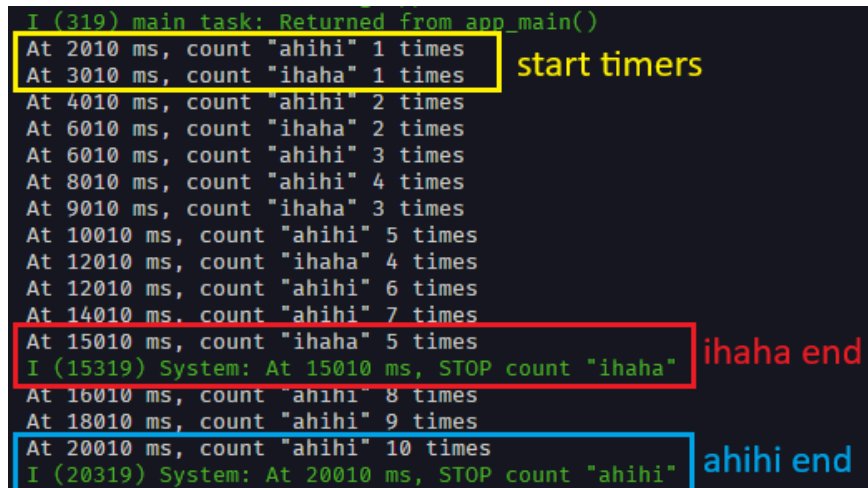
### Đầu ra kỳ vọng

xTimer\_ahihi in ra console "ahihi" theo chu kỳ 2.000ms. Vì vậy, Kỳ vọng xTimer\_ahihi sẽ kết thúc sau 10 chu kỳ (time\_end: 20.000 ms).

xTimer\_ihaha in ra console "ihaha" theo chu kỳ 3.000ms. Vì vậy, Kỳ vọng xTimer\_ihaha sẽ kết thúc sau 5 chu kỳ (time\_end: 15.000 ms).

### Kết quả Hình 4

- . xTimer\_ahihi hiện thực chu kỳ đầu tiên tại 2.010 ms.  
Sau 20 chu kỳ (20.000 ms), kết thúc vào 20.010 ms (ahihi end)
- . xTimer\_ihaha hiện thực chu kỳ đầu tiên tại 3.010 ms.  
Sau 15 chu kỳ (15.000 ms), kết thúc vào 15.010 ms (ihaha end)



The screenshot shows the serial monitor output of an ESP8266. It displays the execution of two timers: 'vTimer\_0' (ahihi) and 'vTimer\_1' (ihaha). The output is as follows:

```

I (319) main task: Returned from app_main()
At 2010 ms, count "ahihi" 1 times
At 3010 ms, count "ihaha" 1 times
At 4010 ms, count "ahihi" 2 times
At 6010 ms, count "ihaha" 2 times
At 6010 ms, count "ahihi" 3 times
At 8010 ms, count "ahihi" 4 times
At 9010 ms, count "ihaha" 3 times
At 10010 ms, count "ahihi" 5 times
At 12010 ms, count "ihaha" 4 times
At 12010 ms, count "ahihi" 6 times
At 14010 ms, count "ahihi" 7 times
At 15010 ms, count "ihaha" 5 times
I (15319) System: At 15010 ms, STOP count "ihaha"
At 16010 ms, count "ahihi" 8 times
At 18010 ms, count "ahihi" 9 times
At 20010 ms, count "ahihi" 10 times
I (20319) System: At 20010 ms, STOP count "ahihi"

```

Annotations on the image:

- A yellow box highlights the first two lines of output: "At 2010 ms, count 'ahihi' 1 times" and "At 3010 ms, count 'ihaha' 1 times".
- A red box highlights the line "At 15010 ms, count 'ihaha' 5 times" and the subsequent system message "I (15319) System: At 15010 ms, STOP count 'ihaha'".
- A blue box highlights the line "At 20010 ms, count 'ahihi' 10 times" and the subsequent system message "I (20319) System: At 20010 ms, STOP count 'ahihi'".

Hình 4: Output của chương trình