

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



**HỆ THỐNG NHÚNG (CO3053)**

---

**Lab 5 - FreeRTOS Software Timer**

**Lab 6 - ESP32: WiFi Subsystem**

---

Giảng viên hướng dẫn: Vũ Trọng Thiên  
Nhóm sinh viên thực hiện - Nhóm 33: Phạm Duy Quang - 2011899  
Dương Đức Nghĩa - 2011671

Thành phố Hồ Chí Minh, Tháng 12/2023



## Mục lục

<b>1</b>	<b>Github</b>	<b>2</b>
<b>2</b>	<b>Lab 5</b>	<b>2</b>
2.1	Exercise . . . . .	2
2.2	Define . . . . .	2
2.3	Implementation . . . . .	2
2.3.1	Tạo timer . . . . .	2
2.3.2	Callback function . . . . .	3
2.4	Result . . . . .	4
<b>3</b>	<b>Lab 6</b>	<b>5</b>
3.1	Giới thiệu . . . . .	5
3.1.1	Mục tiêu . . . . .	5
3.1.2	Nội dung yêu cầu . . . . .	5
3.1.3	WiFi trong ESP32 . . . . .	5
3.1.4	Khởi tạo và thiết lập chế độ hoạt động của WiFi . . . . .	5
3.1.5	Các hàm của WiFi . . . . .	6
3.2	Nội dung hiện thực . . . . .	8
3.2.1	Access Point . . . . .	8
3.2.2	Station . . . . .	10
3.3	Kết quả đạt được . . . . .	13

## 1. Github

Link Github các bài lab của nhóm 33: <https://github.com/ULTIMATE-Mystery/Embedded-System-HCMUT-Semester-231>.

## 2. Lab 5

### 2.1 Exercise

Tạo 2 software timer nhưng chỉ sử dụng chung 1 callback function:

- Timer đầu tiên in ra "ahihi" mỗi 2s và dừng lại sau 10 lần in.
- Timer đầu tiên in ra "ihaha" mỗi 3s và dừng lại sau 5 lần in.

### 2.2 Define

Để dễ cho việc hiện thực code, nhóm define các timer eTimeStamp, eTimer1 và eTimer2:

- eTimeStamp: Timer dùng để tính timestamp.
- eTimer1: Timer cho việc in "ihahi" mỗi 2s.
- eTimer1: Timer cho việc in "ihaha" mỗi 3s.

```
1 typedef enum { eTimeStamp, eTimer1, eTimer2 } TimerID_t;
```

### 2.3 Implementation

#### 2.3.1 Tạo timer

Ta tạo các timer tương ứng với việc define timer ở trên với các bước như sau:

1. Tạo timer bằng API **xTimerCreate()**.
2. Kiểm tra xem việc tạo được timer thành công hay không bằng điều kiện **if (xTimers[x] == NULL)**. Nếu timer không được tạo thành công thì in ra console log cho người dùng biết. Ngược lại nếu tạo thành công ta kiểm tra thêm việc timer có được chạy rồi hay chưa bằng điều kiện **if (xTimerStart(xTimers[x], 0) != pdPASS)**.

```
1 TimerID_t xTimerID[NUM_TIMERS] = {eTimeStamp, eTimer1, eTimer2};
2
3 void app_main(void) {
4     // Create the timer
5     for (int x = 0; x < NUM_TIMERS; x++) {
6         xTimers[x] = xTimerCreate("Timer", xTimerPeriodInTicks[x], pdTRUE,
7                                   (void*)xTimerID[x], vTimerCallback);
8     }
```

```
9     if (xTimers[x] == NULL) // check if the timer is created
10     {
11         printf("xTimerCreate() API failed !\n");
12     } else // check if the timer is started
13     {
14         if (xTimerStart(xTimers[x], 0) != pdPASS) {
15             printf("The Timer started failed !\n");
16         } else {
17             printf(" Initialized timer %d successfully \n", x);
18         }
19     }
20 }
21 }
```

### 2.3.2 Callback function

Ta hiện thực hàm callback function bằng cách overwrite lại hàm **void vTimerCallback(TimerHandle\_t xTimer)**.

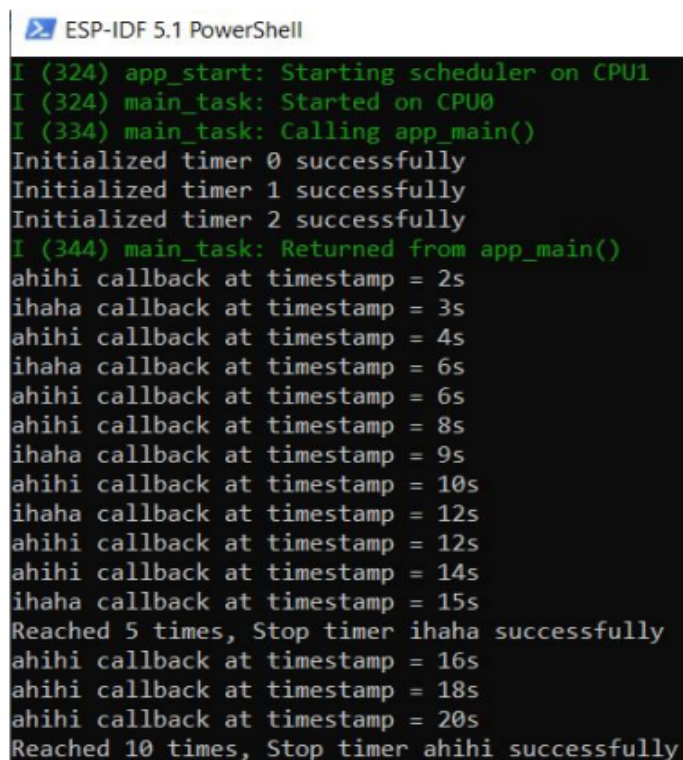
1. Việc sử dụng chung 1 hàm callback thì ta cần phải kiểm tra timer nào đang gọi hàm callback đó bằng API **pvTimerGetTimerID(xTimer)**.
2. Sau khi biết được timer đó là timer nào bằng việc kiểm tra timer ID (ID này được tạo trong phần tạo timer trên ở parameter thứ 4 của API **xTimerCreate()** thì ta sử dụng switch case để hiện thực timer đó cần làm gì. Ví dụ: Timer 1 cần in ra "ihihi" và kiểm tra nếu in 10 lần thì dừng, Timer 2 thì in ra "ihaha" và kiểm tra nếu in ra 5 lần thì dừng timer bằng API **xTimerStop()**.

```
1 void vTimerCallback(TimerHandle_t xTimer) {
2     BaseType_t timerID = (BaseType_t)pvTimerGetTimerID(xTimer);
3     static uint32_t timer1Count = 0;
4     static uint32_t timer2Count = 0;
5
6     switch (timerID) {
7         case eTimestamp:
8             g_timestamp++; // Increment global timestamp variable
9             break;
10
11         case eTimer1:
12             printf("ahihi callback at timestamp = %lds\n",
13                 g_timestamp / pdMS_TO_TICKS(1000));
14             timer1Count++;
15
16             if (timer1Count >= uMaxExpiryCountBeforeStopping[eTimer1]) {
17                 if (xTimerStop(xTimer, 0) == pdPASS) {
18                     printf("Reached %ld times, Stop timer ahihi successfully\n",
19                         timer1Count);
20                 } else {
21                     printf("Stop timer ahihi failed \n");
22                 }
23             }
24     }
```

```
24     break;
25
26 case eTimer2:
27     printf("ihaha callback at timestamp = %lds\n",
28           g_timestamp / pdMS_TO_TICKS(1000));
29     timer2Count++;
30
31     if (timer2Count >= uMaxExpiryCountBeforeStopping[eTimer2]) {
32         if (xTimerStop(xTimer, 0) == pdPASS) {
33             printf("Reached %ld times, Stop timer ihaha successfully\n",
34                   timer2Count);
35         } else {
36             printf("Stop timer ihaha failed\n");
37         }
38     }
39     break;
40
41 default:
42     printf("Invalid Timer ID\n");
43     break;
44 }
45 }
```

## 2.4 Result

Kết quả sau khi chạy mô phỏng:



```
ESP-IDF 5.1 PowerShell
I (324) app_start: Starting scheduler on CPU1
I (324) main_task: Started on CPU0
I (334) main_task: Calling app_main()
Initialized timer 0 successfully
Initialized timer 1 successfully
Initialized timer 2 successfully
I (344) main_task: Returned from app_main()
ahihi callback at timestamp = 2s
ihaha callback at timestamp = 3s
ahihi callback at timestamp = 4s
ihaha callback at timestamp = 6s
ahihi callback at timestamp = 6s
ahihi callback at timestamp = 8s
ihaha callback at timestamp = 9s
ahihi callback at timestamp = 10s
ihaha callback at timestamp = 12s
ahihi callback at timestamp = 12s
ahihi callback at timestamp = 14s
ihaha callback at timestamp = 15s
Reached 5 times, Stop timer ihaha successfully
ahihi callback at timestamp = 16s
ahihi callback at timestamp = 18s
ahihi callback at timestamp = 20s
Reached 10 times, Stop timer ahihi successfully
```

Hình 1: Kết quả chạy mô phỏng software timer

## 3. Lab 6

### 3.1 Giới thiệu

#### 3.1.1 Mục tiêu

Hiểu và có khả năng cấu hình hệ thống WiFi của ESP32 thành:

- Một điểm truy cập (Access Point).
- Một trạm (Station)

#### 3.1.2 Nội dung yêu cầu

Xây dựng và hiện thực hệ thống Wifi trên ESP32 với các chức năng:

- Xử lý sự kiện WiFi (Handling WiFi Events).
- Quét các điểm truy cập (Scanning for access points).
- Kết nối với một điểm truy cập (Connecting to an access point).
- Là một điểm truy cập (Being an access point).

#### 3.1.3 WiFi trong ESP32

Một trong những tính năng tuyệt vời nhất của ESP32 cung cấp là nó không những chỉ kết nối với mạng Wifi hiện có và hoạt động như một Web Server, mà còn có thể thiết lập một mạng riêng, cho phép các thiết bị khác kết nối trực tiếp với nó và truy cập các trang web.

Điều này có thể thực hiện được vì ESP32 có thể hoạt động ở 3 chế độ khác nhau: chế độ **Station**, chế độ **Access Point** và cả hai chế độ cùng một lúc. Điều này cung cấp khả năng xây dựng Network.

#### 3.1.4 Khởi tạo và thiết lập chế độ hoạt động của WiFi

##### a) Khởi tạo

WiFi chỉ là một phần khả năng của ESP32 nên chúng ta cần phải khởi tạo một hệ thống WiFi dành cho ESP32 để có thể sử dụng được.

```
1 wifi_init_config_t config = WIFI_INIT_CONFIG_DEFAULT();  
2 esp_wifi_init(&config);
```

##### b) Thiết lập chế độ WiFi

ESP32 có thể trở thành một trạm (Station) trong mạng, một điểm truy cập (Access Point) cho các thiết bị khác hoặc cả hai.

- **Station:** Khi là một trạm, ESP32 nhận IP từ router mà nó được kết nối. Với địa chỉ IP này, nó có thể thiết lập một Web Server và cung cấp các trang web đến tất cả các thiết bị được kết nối trong mạng WiFi hiện có.
- **Access Point:** ESP32 tạo mạng WiFi riêng và hoạt động như một trung tâm (Giống như Router WiFi) cho một hoặc nhiều trạm. Ở chế độ AP, ESP32 tạo một mạng WiFi mới và đặt SSID (tên mạng) và địa chỉ IP cho nó. Với địa chỉ IP này, nó có thể cung cấp các trang web đến tất cả các thiết bị được kết nối trong mạng riêng của nó.

### 3.1.5 Các hàm của WiFi

#### a) Xử lý các sự kiện WiFi

Trong quá trình hoạt động như một thiết bị WiFi, có thể xảy ra một số sự kiện mà ESP32 cần biết. Những sự kiện này có thể quan trọng hoặc đáng chú ý đối với các ứng dụng đang chạy trên nó. Vì chúng ta không biết khi nào, hoặc thậm chí có thể không xảy ra sự kiện nào, nên chúng ta không thể để ứng dụng của mình chờ đợi sự kiện xảy ra. Thay vào đó, điều chúng ta nên làm là định nghĩa một hàm gọi lại sẽ được kích hoạt nếu sự kiện thực sự xảy ra.

```
1 esp_err_t eventHandler(void* ctx, system_event_t* event) {
2     // Handle event here ...
3     return ESP_OK;
4 }
```

#### b) Quét điểm truy cập

Quét điểm truy cập (WiFi Access Points) là một chức năng quan trọng khi làm việc với ESP32, giúp xác định các mạng WiFi có sẵn trong phạm vi. Dưới đây là một ví dụ cơ bản về cách thực hiện quét điểm truy cập trên ESP32:

```
1 static void start_wifi_scan() {
2     ESP_ERROR_CHECK(esp_wifi_scan_start(NULL, true));
3     ESP_LOGI(TAG, "Scanning for access points...");
4
5     // Wait for scan to complete
6     uint16_t ap_count;
7     esp_wifi_scan_get_ap_num(&ap_count);
8
9     if (ap_count == 0) {
10         ESP_LOGI(TAG, "No access points found.");
11         return;
12     }
13
14     wifi_ap_record_t *ap_list =
15         (wifi_ap_record_t *)malloc(sizeof(wifi_ap_record_t) * ap_count);
16     ESP_ERROR_CHECK(esp_wifi_scan_get_ap_records(&ap_count, ap_list));
17
18     ESP_LOGI(TAG, "Found %d access points:", ap_count);
19     for (int i = 0; i < ap_count; i++) {
20         ESP_LOGI(TAG, "- SSID: %s, RSSI: %d dBm", (char *)ap_list[i].ssid,
21                 ap_list[i].rssi);
22     }
23     free(ap_list);
24 }
```

#### c) Kết nối đến một điểm truy cập

Sau khi ESP32 đã được cấu hình với chi tiết cấu hình của trạm, bao gồm tên SSID và mật khẩu, chúng ta đã sẵn sàng thực hiện kết nối đến điểm truy cập mục tiêu.

```
1 static void wifi_init_sta() {
2     wifi_event_group = xEventGroupCreate();
3 }
```

```
4 tcpip_adapter_init();
5 ESP_ERROR_CHECK(esp_event_loop_create_default());
6
7 wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
8 ESP_ERROR_CHECK(esp_wifi_init(&cfg));
9
10 ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID,
11                                             &event_handler, NULL));
12 ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP,
13                                             &event_handler, NULL));
14
15 wifi_config_t wifi_config = {
16     .sta =
17     {
18         .ssid = WIFI_SSID,
19         .password = WIFI_PASSWORD,
20     },
21 };
22
23 ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
24 ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config));
25 ESP_ERROR_CHECK(esp_wifi_start());
26
27 ESP_LOGI(TAG, "Connecting to WiFi...");
28 xEventGroupWaitBits(wifi_event_group, BIT0, false, true, portMAX_DELAY);
29
30 ESP_LOGI(TAG, "Connected to WiFi");
31 }
```

#### d) Trở thành điểm truy cập

Để trở thành một điểm truy cập, chúng ta cần định nghĩa SSID để cho phép các thiết bị khác phân biệt mạng của chúng ta. SSID này có thể được đặt là ẩn nếu chúng ta không muốn nó xuất hiện trong quá trình quét. Ngoài ra, chúng ta cũng cần cung cấp chế độ xác thực sẽ được sử dụng khi một trạm muốn kết nối với chúng ta. Điều này được sử dụng để chỉ cho phép các trạm được ủy quyền và ngăn chặn những trạm không được ủy quyền. Chỉ có các trạm biết mật khẩu của chúng ta mới được phép kết nối.

```
1 static void wifi_init_softap() {
2     wifi_event_group = xEventGroupCreate();
3
4     tcpip_adapter_init();
5     ESP_ERROR_CHECK(esp_event_loop_create_default());
6
7     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
8     ESP_ERROR_CHECK(esp_wifi_init(&cfg));
9
10    wifi_config_t wifi_config = {
11        .ap =
12        {
13            .ssid = AP_SSID,
14            .password = AP_PASSWORD,
```



```
15         .max_connection = 4,  
16         .authmode = WIFI_AUTH_WPA_WPA2_PSK,  
17     },  
18 };  
19  
20 ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));  
21 ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_AP, &wifi_config));  
22 ESP_ERROR_CHECK(esp_wifi_start());  
23  
24 ESP_LOGI(TAG, "WiFi SoftAP started");  
25 }
```

## 3.2 Nội dung hiện thực

### 3.2.1 Access Point

Nhóm hiện thực một Access Point với các thông số sau:

- SSID là: lab6.
- Mật khẩu là 12345678.
- Số channel là 1.
- Số station kết nối tối đa: 5.

```
1 #include <string.h>  
2  
3 #include "esp_event.h"  
4 #include "esp_log.h"  
5 #include "esp_mac.h"  
6 #include "esp_wifi.h"  
7 #include "freertos/FreeRTOS.h"  
8 #include "freertos/task.h"  
9 #include "lwip/err.h"  
10 #include "lwip/sys.h"  
11 #include "nvs_flash.h"  
12  
13 #define EXAMPLE_ESP_WIFI_SSID "lab6"  
14 #define EXAMPLE_ESP_WIFI_PASS "12345678"  
15 #define EXAMPLE_ESP_WIFI_CHANNEL 1  
16 #define EXAMPLE_MAX_STA_CONN 5  
17  
18 static const char *TAG = "wifi softAP";  
19  
20 static void wifi_event_handler(void *arg, esp_event_base_t event_base,  
21                               int32_t event_id, void *event_data) {  
22     if (event_id == WIFI_EVENT_AP_STACONNECTED) {  
23         wifi_event_ap_staconnected_t *event =  
24             (wifi_event_ap_staconnected_t *)event_data;
```

```
25     ESP_LOGI(TAG, "station" MACSTR "join , AID =%d", MAC2STR(event->mac),
26             event->aid);
27 } else if (event_id == WIFI_EVENT_AP_STADISCONNECTED) {
28     wifi_event_ap_stadisconnected_t *event =
29         (wifi_event_ap_stadisconnected_t *)event_data;
30     ESP_LOGI(TAG, "station" MACSTR "leave , AID =%d", MAC2STR(event->mac),
31             event->aid);
32 }
33 }
34
35 void wifi_init_softap(void) {
36     ESP_ERROR_CHECK(esp_netif_init());
37     ESP_ERROR_CHECK(esp_event_loop_create_default());
38     esp_netif_create_default_wifi_ap();
39
40     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
41     ESP_ERROR_CHECK(esp_wifi_init(&cfg));
42
43     ESP_ERROR_CHECK(esp_event_handler_instance_register(
44         WIFI_EVENT,
45
46         ESP_EVENT_ANY_ID, &wifi_event_handler, NULL, NULL));
47
48     wifi_config_t wifi_config = {
49         .ap =
50         {
51             .ssid = EXAMPLE_ESP_WIFI_SSID,
52             .ssid_len = strlen(EXAMPLE_ESP_WIFI_SSID),
53             .channel = EXAMPLE_ESP_WIFI_CHANNEL,
54             .password = EXAMPLE_ESP_WIFI_PASS,
55             .max_connection = EXAMPLE_MAX_STA_CONN,
56             .authmode = WIFI_AUTH_WPA_WPA2_PSK,
57             .pmf_cfg =
58             {
59                 .required = false,
60             },
61         },
62     };
63     if (strlen(EXAMPLE_ESP_WIFI_PASS) == 0) {
64         wifi_config.ap.authmode = WIFI_AUTH_OPEN;
65     }
66
67     ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
68     ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_AP, &wifi_config));
69     ESP_ERROR_CHECK(esp_wifi_start());
70
71     ESP_LOGI(TAG, "wifi_init_softap finished .SSID :%s
72             password
73             : % s channel
```

```
74         : % d ",
75         EXAMPLE_ESP_WIFI_SSID,
76         EXAMPLE_ESP_WIFI_PASS, EXAMPLE_ESP_WIFI_CHANNEL);
77 }
78
79 void app_main(void) {
80     // Initialize NVS
81     esp_err_t ret = nvs_flash_init();
82     if (ret == ESP_ERR_NVS_NO_FREE_PAGES ||
83         ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
84         ESP_ERROR_CHECK(nvs_flash_erase());
85         ret = nvs_flash_init();
86     }
87     ESP_ERROR_CHECK(ret);
88
89     ESP_LOGI(TAG, "ESP_WIFI_MODE_AP");
90     wifi_init_softap();
91 }
```

Program 1: Hiện thực ESP32 là một Access Point

### 3.2.2 Station

Nhóm hiện thực ESP32 như một Station kết nối tới 1 mạng WIFI có sẵn. Và số lần kết nối lại nếu không thành công là 5.

```
1 #include <string.h>
2
3 #include "esp_event.h"
4 #include "esp_log.h"
5 #include "esp_system.h"
6 #include "esp_wifi.h"
7 #include "freertos/FreeRTOS.h"
8 #include "freertos/event_groups.h"
9 #include "freertos/task.h"
10 #include "lwip/err.h"
11 #include "lwip/sys.h"
12 #include "nvs_flash.h"
13
14 #define EXAMPLE_ESP_WIFI_SSID "Pham Duy Quang"
15 #define EXAMPLE_ESP_WIFI_PASS "HCMUT"
16 #define EXAMPLE_ESP_MAXIMUM_RETRY 5
17
18 #if CONFIG_ESP_WIFI_AUTH_OPEN
19 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_OPEN
20 #elif CONFIG_ESP_WIFI_AUTH_WEP
21 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WEP
22 #elif CONFIG_ESP_WIFI_AUTH_WPA_PSK
23 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA_PSK
24 #elif CONFIG_ESP_WIFI_AUTH_WPA2_PSK
```

```
25 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA2_PSK
26 #elif CONFIG_ESP_WIFI_AUTH_WPA_WPA2_PSK
27 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD
28 WIFI_AUTH_WPA_WPA2_PSK
29 #elif CONFIG_ESP_WIFI_AUTH_WPA3_PSK
30 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA3_PSK
31 #elif CONFIG_ESP_WIFI_AUTH_WPA2_WPA3_PSK
32 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD
33 WIFI_AUTH_WPA2_WPA3_PSK
34 #elif CONFIG_ESP_WIFI_AUTH_WAPI_PSK
35 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WAPI_PSK
36 #endif
37
38 static EventGroupHandle_t s_wifi_event_group;
39
40 #define WIFI_CONNECTED_BIT BIT0
41 #define WIFI_FAIL_BIT BIT1
42
43 static const char *TAG = "wifi station";
44
45 static int s_retry_num = 0;
46
47 static void event_handler(void *arg, esp_event_base_t event_base,
48                          int32_t event_id, void *event_data) {
49     if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
50         esp_wifi_connect();
51     } else if (event_base == WIFI_EVENT &&
52               event_id == WIFI_EVENT_STA_DISCONNECTED) {
53         if (s_retry_num < EXAMPLE_ESP_MAXIMUM_RETRY) {
54             esp_wifi_connect();
55             s_retry_num++;
56             ESP_LOGI(TAG, "retry to connect to the AP");
57         } else {
58             xEventGroupSetBits(s_wifi_event_group, WIFI_FAIL_BIT);
59         }
60         ESP_LOGI(TAG, "connect to the AP fail");
61     } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
62         ip_event_got_ip_t *event = (ip_event_got_ip_t *)event_data;
63         ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->ip_info.ip));
64         s_retry_num = 0;
65         xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
66     }
67 }
68
69 void wifi_init_sta(void) {
70     s_wifi_event_group = xEventGroupCreate();
71
72     ESP_ERROR_CHECK(esp_netif_init());
73 }
```

```
74 ESP_ERROR_CHECK(esp_event_loop_create_default());
75 esp_netif_create_default_wifi_sta();
76
77 wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
78 ESP_ERROR_CHECK(esp_wifi_init(&cfg));
79
80 esp_event_handler_instance_t instance_any_id;
81 esp_event_handler_instance_t instance_got_ip;
82 ESP_ERROR_CHECK(esp_event_handler_instance_register(
83     WIFI_EVENT,
84
85     ESP_EVENT_ANY_ID, &event_handler, NULL, &instance_any_id));
86 ESP_ERROR_CHECK(esp_event_handler_instance_register(
87     IP_EVENT,
88
89     IP_EVENT_STA_GOT_IP, &event_handler, NULL, &instance_got_ip));
90
91 wifi_config_t wifi_config = {
92     .sta =
93     {
94         .ssid = EXAMPLE_ESP_WIFI_SSID,
95         .password = EXAMPLE_ESP_WIFI_PASS,
96         .threshold.authmode = ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD,
97         .sae_pwe_h2e = WPA3_SAE_PWE_BOTH,
98     },
99 };
100 ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
101 ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config));
102 ESP_ERROR_CHECK(esp_wifi_start());
103
104 ESP_LOGI(TAG, "wifi_init_sta finished.");
105
106 EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
107     WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
108     pdFALSE, pdFALSE, portMAX_DELAY);
109
110 if (bits & WIFI_CONNECTED_BIT) {
111     ESP_LOGI(TAG, "connected to ap SSID :%s password :%s",
112         EXAMPLE_ESP_WIFI_SSID, EXAMPLE_ESP_WIFI_PASS);
113 } else if (bits & WIFI_FAIL_BIT) {
114     ESP_LOGI(TAG, "Failed to connect to SSID :%s, password
115         : % s ",
116         EXAMPLE_ESP_WIFI_SSID,
117         EXAMPLE_ESP_WIFI_PASS);
118 } else {
119     ESP_LOGE(TAG, "UNEXPECTED EVENT");
120 }
121 }
```



The screenshot shows a Windows IDE with a C++ project for an ESP8266. The main window displays the 'station\_example\_main.c' file, which includes network configuration and ping tests. A terminal window is open, showing the output of the program, including connection details and ping statistics.

**Main Window (station\_example\_main.c):**

```

1  #include "esp_log.h"
2  #include "nvs_flash.h"
3  #include "wifi.h"
4  #include "esp_wifi.h"
5  #include "esp_system.h"
6  #include "esp_netif.h"
7  #include "esp8266.h"
8  #include "esp8266_pins.h"
9  #include "esp8266_gpio.h"
10 #include "esp8266_i2c.h"
11 #include "esp8266_uart.h"
12 #include "esp8266_pwm.h"
13 #include "esp8266_adc.h"
14 #include "esp8266_dac.h"
15 #include "esp8266_i2s.h"
16 #include "esp8266_i2c.h"
17 #include "esp8266_uart.h"
18 #include "esp8266_pwm.h"
19 #include "esp8266_adc.h"
20 #include "esp8266_dac.h"
21 #include "esp8266_i2s.h"
22 #include "esp8266_i2c.h"
23 #include "esp8266_uart.h"
24 #include "esp8266_pwm.h"
25 #include "esp8266_adc.h"
26 #include "esp8266_dac.h"
27 #include "esp8266_i2s.h"
28 #include "esp8266_i2c.h"
29 #include "esp8266_uart.h"
30 #include "esp8266_pwm.h"
31 #include "esp8266_adc.h"
32 #include "esp8266_dac.h"
33 #include "esp8266_i2s.h"
34 #include "esp8266_i2c.h"
35 #include "esp8266_uart.h"
36 #include "esp8266_pwm.h"
37 #include "esp8266_adc.h"
38 #include "esp8266_dac.h"
39 #include "esp8266_i2s.h"
40 #include "esp8266_i2c.h"
41 #include "esp8266_uart.h"
42 #include "esp8266_pwm.h"
43 #include "esp8266_adc.h"
44 #include "esp8266_dac.h"
45 #include "esp8266_i2s.h"
46 #include "esp8266_i2c.h"
47 #include "esp8266_uart.h"
48 #include "esp8266_pwm.h"
49 #include "esp8266_adc.h"
50 #include "esp8266_dac.h"
51 #include "esp8266_i2s.h"
52 #include "esp8266_i2c.h"
53 #include "esp8266_uart.h"
54 #include "esp8266_pwm.h"
55 #include "esp8266_adc.h"
56 #include "esp8266_dac.h"
57 #include "esp8266_i2s.h"
58 #include "esp8266_i2c.h"
59 #include "esp8266_uart.h"
60 #include "esp8266_pwm.h"
61 #include "esp8266_adc.h"
62 #include "esp8266_dac.h"
63 #include "esp8266_i2s.h"
64 #include "esp8266_i2c.h"
65 #include "esp8266_uart.h"
66 #include "esp8266_pwm.h"
67 #include "esp8266_adc.h"
68 #include "esp8266_dac.h"
69 #include "esp8266_i2s.h"
70 #include "esp8266_i2c.h"
71 #include "esp8266_uart.h"
72 #include "esp8266_pwm.h"
73 #include "esp8266_adc.h"
74 #include "esp8266_dac.h"
75 #include "esp8266_i2s.h"
76 #include "esp8266_i2c.h"
77 #include "esp8266_uart.h"
78 #include "esp8266_pwm.h"
79 #include "esp8266_adc.h"
80 #include "esp8266_dac.h"
81 #include "esp8266_i2s.h"
82 #include "esp8266_i2c.h"
83 #include "esp8266_uart.h"
84 #include "esp8266_pwm.h"
85 #include "esp8266_adc.h"
86 #include "esp8266_dac.h"
87 #include "esp8266_i2s.h"
88 #include "esp8266_i2c.h"
89 #include "esp8266_uart.h"
90 #include "esp8266_pwm.h"
91 #include "esp8266_adc.h"
92 #include "esp8266_dac.h"
93 #include "esp8266_i2s.h"
94 #include "esp8266_i2c.h"
95 #include "esp8266_uart.h"
96 #include "esp8266_pwm.h"
97 #include "esp8266_adc.h"
98 #include "esp8266_dac.h"
99 #include "esp8266_i2s.h"
100 #include "esp8266_i2c.h"
101 #include "esp8266_uart.h"
102 #include "esp8266_pwm.h"
103 #include "esp8266_adc.h"
104 #include "esp8266_dac.h"
105 #include "esp8266_i2s.h"
106 #include "esp8266_i2c.h"
107 #include "esp8266_uart.h"
108 #include "esp8266_pwm.h"
109 #include "esp8266_adc.h"
110 #include "esp8266_dac.h"
111 #include "esp8266_i2s.h"
112 #include "esp8266_i2c.h"
113 #include "esp8266_uart.h"
114 #include "esp8266_pwm.h"
115 #include "esp8266_adc.h"
116 #include "esp8266_dac.h"
117 #include "esp8266_i2s.h"
118 #include "esp8266_i2c.h"
119 #include "esp8266_uart.h"
120 #include "esp8266_pwm.h"
121 #include "esp8266_adc.h"
122 #include "esp8266_dac.h"
123 #include "esp8266_i2s.h"
124 #include "esp8266_i2c.h"
125 #include "esp8266_uart.h"
126 #include "esp8266_pwm.h"
127 #include "esp8266_adc.h"
128 #include "esp8266_dac.h"
129 #include "esp8266_i2s.h"
130 #include "esp8266_i2c.h"
131 #include "esp8266_uart.h"
132 #include "esp8266_pwm.h"
133 #include "esp8266_adc.h"
134 #include "esp8266_dac.h"
135 #include "esp8266_i2s.h"
136 #include "esp8266_i2c.h"
137 #include "esp8266_uart.h"
138 #include "esp8266_pwm.h"
139 #include "esp8266_adc.h"
140 #include "esp8266_dac.h"
141 #include "esp8266_i2s.h"
142 #include "esp8266_i2c.h"
143 #include "esp8266_uart.h"
144 #include "esp8266_pwm.h"
145 #include "esp8266_adc.h"
146 #include "esp8266_dac.h"
147 #include "esp8266_i2s.h"
148 #include "esp8266_i2c.h"
149 #include "esp8266_uart.h"
150 #include "esp8266_pwm.h"
151 #include "esp8266_adc.h"
152 #include "esp8266_dac.h"
153 #include "esp8266_i2s.h"
154 #include "esp8266_i2c.h"
155 #include "esp8266_uart.h"
156 #include "esp8266_pwm.h"
157 #include "esp8266_adc.h"
158 #include "esp8266_dac.h"
159 #include "esp8266_i2s.h"
160 #include "esp8266_i2c.h"
161 #include "esp8266_uart.h"
162 #include "esp8266_pwm.h"
163 #include "esp8266_adc.h"
164 #include "esp8266_dac.h"
165 #include "esp8266_i2s.h"
166 #include "esp8266_i2c.h"
167 #include "esp8266_uart.h"
168 #include "esp8266_pwm.h"
169 #include "esp8266_adc.h"
170 #include "esp8266_dac.h"
171 #include "esp8266_i2s.h"
172 #include "esp8266_i2c.h"
173 #include "esp8266_uart.h"
174 #include "esp8266_pwm.h"
175 #include "esp8266_adc.h"
176 #include "esp8266_dac.h"
177 #include "esp8266_i2s.h"
178 #include "esp8266_i2c.h"
179 #include "esp8266_uart.h"
180 #include "esp8266_pwm.h"
181 #include "esp8266_adc.h"
182 #include "esp8266_dac.h"
183 #include "esp8266_i2s.h"
184 #include "esp8266_i2c.h"
185 #include "esp8266_uart.h"
186 #include "esp8266_pwm.h"
187 #include "esp8266_adc.h"
188 #include "esp8266_dac.h"
189 #include "esp8266_i2s.h"
190 #include "esp8266_i2c.h"
191 #include "esp8266_uart.h"
192 #include "esp8266_pwm.h"
193 #include "esp8266_adc.h"
194 #include "esp8266_dac.h"
195 #include "esp8266_i2s.h"
196 #include "esp8266_i2c.h"
197 #include "esp8266_uart.h"
198 #include "esp8266_pwm.h"
199 #include "esp8266_adc.h"
200 #include "esp8266_dac.h"
201 #include "esp8266_i2s.h"
202 #include "esp8266_i2c.h"
203 #include "esp8266_uart.h"
204 #include "esp8266_pwm.h"
205 #include "esp8266_adc.h"
206 #include "esp8266_dac.h"
207 #include "esp8266_i2s.h"
208 #include "esp8266_i2c.h"
209 #include "esp8266_uart.h"
210 #include "esp8266_pwm.h"
211 #include "esp8266_adc.h"
212 #include "esp8266_dac.h"
213 #include "esp8266_i2s.h"
214 #include "esp8266_i2c.h"
215 #include "esp8266_uart.h"
216 #include "esp8266_pwm.h"
217 #include "esp8266_adc.h"
218 #include "esp8266_dac.h"
219 #include "esp8266_i2s.h"
220 #include "esp8266_i2c.h"
221 #include "esp8266_uart.h"
222 #include "esp8266_pwm.h"
223 #include "esp8266_adc.h"
224 #include "esp8266_dac.h"
225 #include "esp8266_i2s.h"
226 #include "esp8266_i2c.h"
227 #include "esp8266_uart.h"
228 #include "esp8266_pwm.h"
229 #include "esp8266_adc.h"
230 #include "esp8266_dac.h"
231 #include "esp8266_i2s.h"
232 #include "esp8266_i2c.h"
233 #include "esp8266_uart.h"
234 #include "esp8266_pwm.h"
235 #include "esp8266_adc.h"
236 #include "esp8266_dac.h"
237 #include "esp8266_i2s.h"
238 #include "esp8266_i2c.h"
239 #include "esp8266_uart.h"
240 #include "esp8266_pwm.h"
241 #include "esp8266_adc.h"
242 #include "esp8266_dac.h"
243 #include "esp8266_i2s.h"
244 #include "esp8266_i2c.h"
245 #include "esp8266_uart.h"
246 #include "esp8266_pwm.h"
247 #include "esp8266_adc.h"
248 #include "esp8266_dac.h"
249 #include "esp8266_i2s.h"
250 #include "esp8266_i2c.h"
251 #include "esp8266_uart.h"
252 #include "esp8266_pwm.h"
253 #include "esp8266_adc.h"
254 #include "esp8266_dac.h"
255 #include "esp8266_i2s.h"
256 #include "esp8266_i2c.h"
257 #include "esp8266_uart.h"
258 #include "esp8266_pwm.h"
```

Hình 3: Kết quả hiện thực ESP32 như là một Station

- Nhóm đã hiểu được cơ bản về WiFi trong ESP32.
- Hiện thực kết nối ESP32 tới WIFI khác và ESP32 như là Access Point.
- Hiện thực được các chế độ của WiFi trong ESP32.