**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

# BÁO CÁO BÀI THỰC HÀNH LAB04
# LỚP CE232.N21.1


**Giảng viên hướng dẫn:** TRẦN HOÀNG LỘC

**Sinh viên thực hiện nhóm 7:**

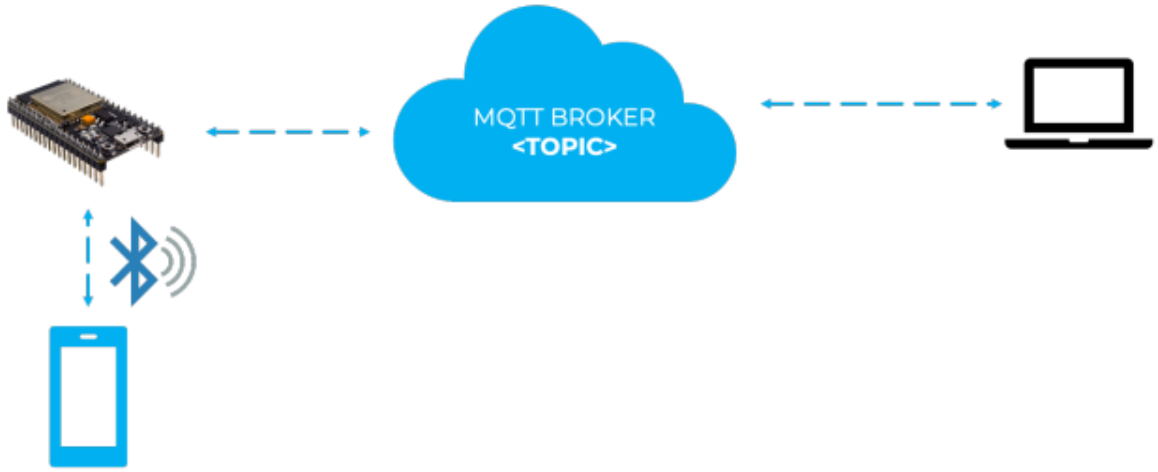
| | |
|---|---|
| Trương Hữu Khang | 20520211 |
| Nguyễn Linh Anh Khoa | 20520219 |
| Hà Vĩnh Kiện | 20520597 |
| Phan Duy Thông | 20520789 |


**TP. Hồ Chí Minh, 2023**

**PHẦN ĐỀ:**

**Bài tập 1. Trình bày các bước cài đặt để kết nối ESP32 đến flespi MQTT Broker**

**Bài tập 2. Hiện thực mô hình trong Hình 16**



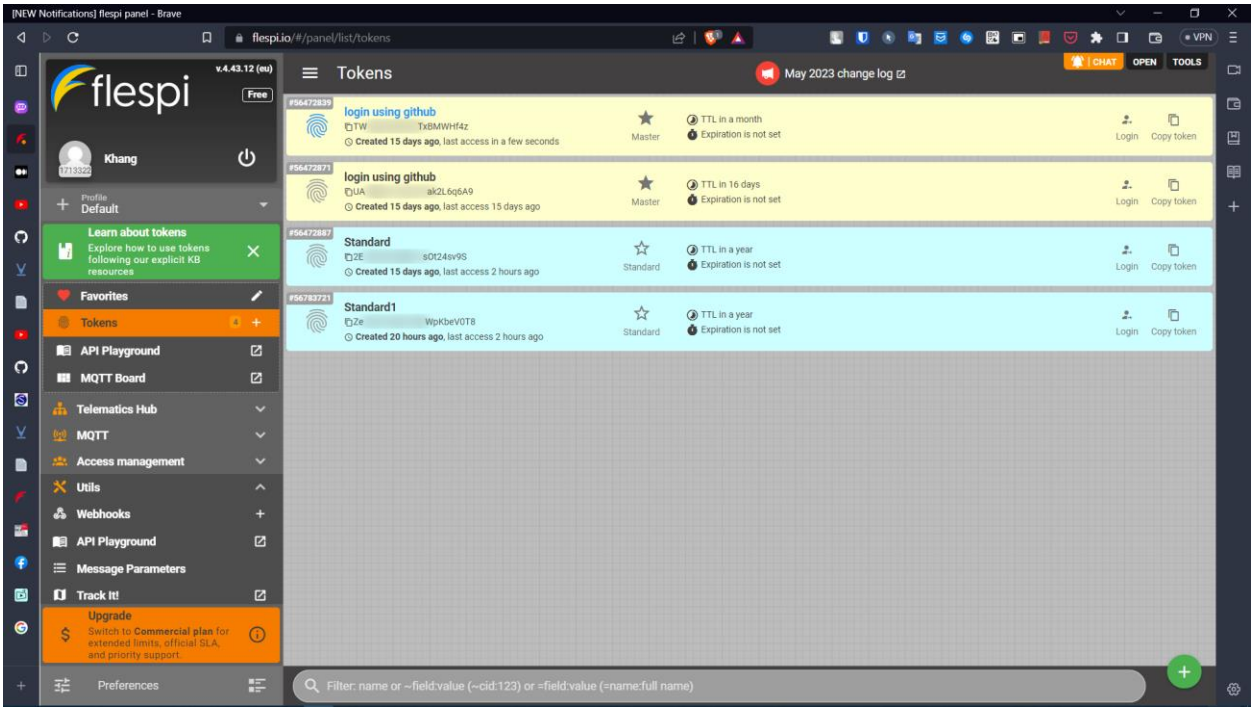

*Hình 16. Mô hình kết nối điện thoại, ESP32, end-user*

**PHẦN BÁO CÁO**

**Bài tập 1. Trình bày các bước cài đặt để kết nối ESP32 đến flespi MQTT Broker**

Để kết nối ESP32 đến flespi MQTT Broker, cần thực hiện các bước sau:

1. Tạo một tài khoản flespi và tạo một MQTT Broker.




2. Cài đặt ESP-IDF và cấu hình các biến môi trường cho project.

Dựa trên project mẫu, sử dụng example của ESP-IDF để thực hiện kết nối với MQTT

3. Điều chỉnh các thông số trong đoạn code như tên mạng WiFi, mật khẩu, địa chỉ broker MQTT và thông tin xác thực cho client từ project mẫu.

Ta cần khai báo các biến chưa thông tin tên WiFi, mật khẩu đăng nhập, số lần truy cập và địa chỉ broker

```
#define EXAMPLE_ESP_WIFI_SSID "DESKTOP-OI97DKV 6851"
#define EXAMPLE_ESP_WIFI_PASS "40n9X+82"
#define EXAMPLE_ESP_MAXIMUM_RETRY 5
#define CONFIG_BROKER_URL "mqtt://mqtt.flespi.io"
```

Để cấu hình các thông số kết nối MQTT, cần chỉnh sửa cấu trúc esp_mqtt_client_config_t trong hàm mqtt_app_start() trong đoạn code.

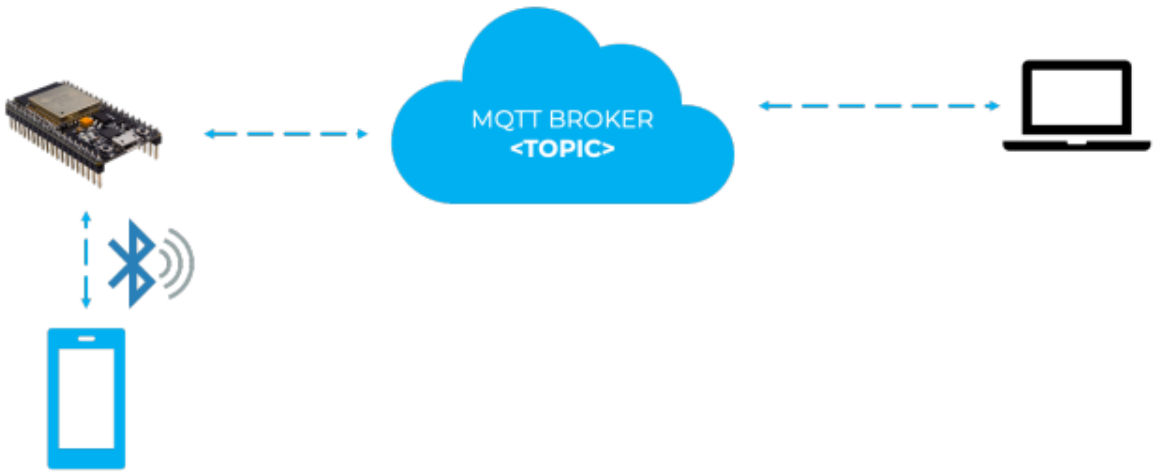
```
static void mqtt_app_start(void)
{
    esp_mqtt_client_config_t mqtt_cfg = {
        .broker.address.uri = CONFIG_BROKER_URL,
        .broker.address.port = 1883,
        .credentials.username = "2ETpFZptbMzxpFD8J5G7pgDhzKoMp9CfBlu7bJocjNxwi0uPWYMb5ehsOt24sv9S",
        .credentials.client_id = "esp32",
    };
```

Các thông số cần chỉnh sửa bao gồm

địa chỉ broker MQTT (chỉ định bằng URL), port (mặc định là 1883), tên đăng nhập và client ID.

4. Biên dịch và nạp chương trình vào ESP32.
5. Sau khi ESP32 được nạp chương trình thành công và kết nối đến mạng WiFi, nó sẽ tự động kết nối đến broker MQTT và gửi các thông điệp tương ứng.

**Bài tập 2. Hiện thực mô hình**




Trong mô hình trên ta sẽ sử dụng lại cách kết nối Bluetooth của lab3 và kết nối với MQTT của bài 1. Các bước thực hiện để khởi tạo các thành phần khác nhau, bao gồm Bluetooth và Wi-Fi, và sau đó bắt đầu một MQTT client:

1. Tạo hai nhóm sự kiện: s_mqtt_event_group và s_wifi_event_group. Nhóm sự kiện là một cơ chế đồng bộ hóa để đợi và xử lý các sự kiện kích hoạt bởi các phần khác trong ứng dụng.

2. Khởi tạo các mức độ nhật ký khác nhau bằng hàm esp_log_level_set(). Điều này cho phép chúng ta chỉ định các loại thông tin nhật ký mà chúng ta muốn in ra màn hình. Hàm ESP_LOGI() được sử dụng để ghi lại thông tin nhật ký.
3. Khởi tạo NVS bằng hàm nvs_flash_init() để lưu trữ các thông tin cấu hình của ứng dụng. Nếu NVS đã được khởi tạo trước đó, hàm này sẽ trả về lỗi ESP_ERR_NVS_NO_FREE_PAGES hoặc ESP_ERR_NVS_NEW_VERSION_FOUND. Trong trường hợp này, hàm nvs_flash_erase() được sử dụng để xóa các trang NVS không còn được sử dụng.
4. Giải phóng bộ nhớ được sử dụng bởi Classic Bluetooth Mode bằng hàm esp_bt_controller_mem_release().
5. Khởi tạo Bluetooth Controller bằng cách đọc các giá trị mặc định bằng hàm BT_CONTROLLER_INIT_CONFIG_DEFAULT() và truyền vào hàm esp_bt_controller_init().
6. Kích hoạt chế độ BLE bằng hàm esp_bt_controller_enable().
7. Khởi tạo Bluetooth stack bằng hàm esp_bluedroid_init() và kích hoạt nó bằng hàm esp_bluedroid_enable().
8. Đăng ký các lời gọi lại cho các sự kiện GATT và GAP bằng hàm esp_ble_gatts_register_callback() và esp_ble_gap_register_callback().
9. Đăng ký ứng dụng GATT với một ID cụ thể bằng hàm esp_ble_gatts_app_register().
10. Đặt MTU cục bộ thành 33 byte bằng hàm esp_ble_gatt_set_local_mtu().
11. Đặt các tham số bảo mật khác nhau cho giao tiếp BLE bằng hàm esp_ble_gap_set_security_param().
12. Khởi tạo Wi-Fi ở chế độ station và đợi nó kết nối đến một điểm truy cập bằng hàm wifi_init_sta() và hàm xEventGroupWaitBits().
13. Bắt đầu MQTT client bằng hàm mqtt_app_start().

**Cơ chế hoạt động:**

Trường hợp: Điện thoại muốn gửi một tin nhắn đến máy tính thông qua ESP32 và MQTT broker và ngược lại. Quá trình này có thể được thực hiện như sau:

1. Điện thoại gửi tin nhắn đến ESP32 thông qua giao thức BLE. Tin nhắn được truyền dưới dạng các gói tin dữ liệu BLE và được nhận bởi ESP32 thông qua các callback đã được đăng ký cho các sự kiện BLE.
2. ESP32 lấy dữ liệu từ tin nhắn BLE và gửi nó lên MQTT broker thông qua giao thức MQTT. Để làm điều này, ESP32 sử dụng thư viện MQTT để kết nối đến broker và gửi tin nhắn dưới dạng một thông điệp MQTT.
3. MQTT broker nhận thông điệp từ ESP32 và xác định rằng tin nhắn đó cần được gửi đến máy tính. Broker sau đó gửi thông điệp đó đến máy tính thông qua kết nối mạng Wi-Fi và giao thức MQTT.
4. Máy tính kết nối đến MQTT broker và nhận thông điệp từ broker. Để làm điều này, máy tính sử dụng một thư viện MQTT và đăng ký một callback để xử lý thông điệp đến khi nó được nhận.
5. Callback trên máy tính xử lý thông điệp và hiển thị nó lên màn hình.

Trường hợp: Muốn thông điệp từ MQTT broker về điện thoại

```
    break;
case MQTT_EVENT_DATA:
    ESP_LOGI(TAG_MQTT, "MQTT_EVENT_DATA");
    printf("TOPIC=%.*s\r\n", event->topic_len, event->topic);
    printf("DATA=%.*s\r\n", event->data_len, event->data);
    esp_ble_gatts_set_attr_value(gatt_db_handle_table[IDX_CHAR_VAL_A], event->data_len, (uint8_t *)event->data);
    break;
```

Hàm MQTT_EVENT_DATA được sử dụng để xử lý sự kiện nhận dữ liệu từ MQTT broker trên ESP32. Khi một thông điệp MQTT được nhận, hàm này sẽ được gọi và thực hiện các bước sau:

1. In ra thông tin về chủ đề và dữ liệu trong thông điệp MQTT bằng cách sử dụng hàm printf.
2. Sử dụng hàm esp_ble_gatts_set_attr_value để cập nhật characteristic value của BLE tương ứng với thông điệp MQTT nhận được.
3. Để cập nhật characteristic value của BLE, hàm này sử dụng bảng định danh characteristic BLE (gatt_db_handle_table) để xác định đặc trưng cần cập nhật. Sau

đó, hàm này sử dụng định danh characteristic để cập nhật characteristic value bằng dữ liệu trong thông điệp MQTT.

Khi một thông điệp MQTT được nhận bởi ESP32, hàm MQTT_EVENT_DATA được gọi để xử lý sự kiện này. Hàm này in ra thông tin về chủ đề và dữ liệu trong thông điệp MQTT và sau đó cập nhật characteristic value của BLE tương ứng với thông điệp MQTT. Điều này cho phép dữ liệu được truyền từ MQTT broker đến ESP32 và được cập nhật trên characteristic value của BLE tương ứng để có thể được đọc hoặc ghi từ các thiết bị khác thông qua giao thức BLE.

Video demo:

https://youtu.be/hNeT0lex6iA

Source code:

https://github.com/otaros/CE232_Lab/tree/main/Lab4