

# THIẾT KẾ HỆ THỐNG NHÚNG KHÔNG DÂY

## BÁO CÁO LAB04

NHÓM 5	
Họ và tên	MSSV
Trần Lê Minh Đăng	21520684
Lê Hữu Đạt	21520697
Trần Văn Dương	21520763

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


#### Broker


Trả lời:


- Đầu tiên, cấu hình kết nối wifi


```
ESP_ERROR_CHECK(esp_wifi_set_storage(WIFI_STORAGE_RAM));
wifi_config_t wifi_config = {
    .sta = {
        .ssid = "MyWifi",
        .password = "101112131415",
        .scan_method = EXAMPLE_WIFI_SCAN_METHOD,
        .sort_method = EXAMPLE_WIFI_CONNECT_AP_SORT_METHOD,
        .threshold.rssi = CONFIG_EXAMPLE_WIFI_SCAN_RSSI_THRESHOLD,
        .threshold.authmode = EXAMPLE_WIFI_SCAN_AUTH_MODE_THRESHOLD,
    },
};
```


- Tiếp đến tạo mới token trên flespi

 Token

 TEMPLATES

 ?

 X

 Edit

Token

Info

ESP32

☒ ?

Expire

?

TTL

1

▼ years

Max: 1 years

☒ ?

Access

STANDARD

MASTER ACL

☒ ?

IPs whitelist





?

Metadata

?

Enabled

?

   SAVE + OPEN  SAVE

- Tiếp đến cấu hình để ESP32 kết nối với flespi MQTT Broker

```
static void mqtt_app_start(void)
{
    esp_mqtt_client_config_t mqtt_cfg = {
        // .uri = CONFIG_BROKER_URL,
        .host = "mqtt.flespi.io",
        .port = 1883,
        .username = "lw02TSD5X3uIR5C4bmfs6xKWQglTZq2IH5dgy9RXZA88ycPBX2NcMNinYVLjUdAb",
        .client_id = "test1234"
    };
};
```

Trong đó:

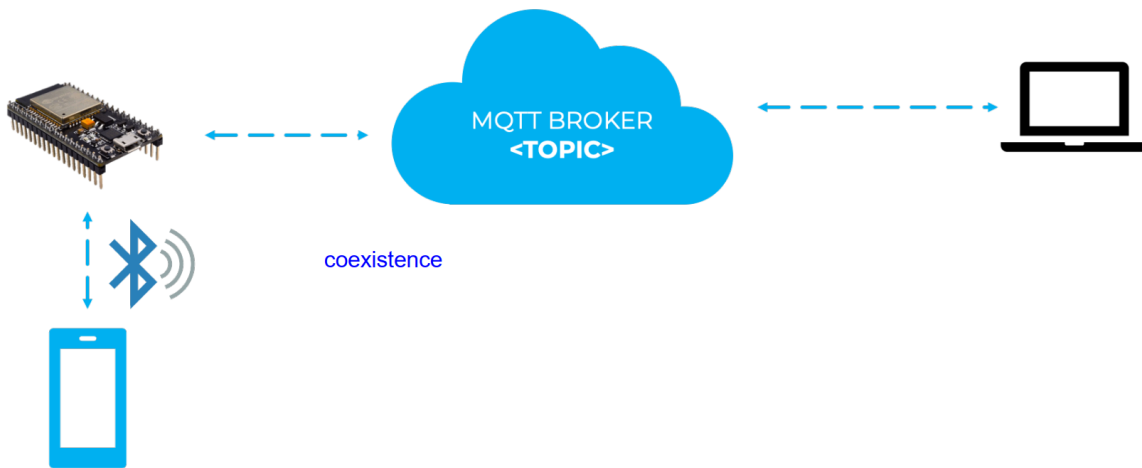
- Host là tên miền máy chủ MQTT( IPv4 dưới dạng chuỗi)
- Port là 1883 cho giao thức TCP không bảo mật
- Username là token vừa mới tạo
- Client\_id là id của ESP32

-

```
esp_mqtt_client_handle_t client = esp_mqtt_client_init(&mqtt_cfg);
/* The last argument may be used to pass data to the event handler, in this example mqtt_event_handler */
esp_mqtt_client_register_event(client, ESP_EVENT_ANY_ID, mqtt_event_handler, NULL);
esp_mqtt_client_start(client);
```

- Tạo client với các thông số đã cấu hình ở trên.
- Hàm esp\_mqtt\_client\_register\_event() được sử dụng để đăng ký một hàm xử lý sự kiện mqtt\_event\_handler cho client MQTT.
- Hàm esp\_mqtt\_client\_start() được gọi để khởi động client MQTT và bắt đầu kết nối đến máy chủ MQTT với các cấu hình đã được chỉ định. Khi client MQTT được khởi động, nó sẽ bắt đầu gửi và nhận các thông điệp MQTT với máy chủ MQTT đã được cấu hình.

## Bài tập 2. Hiện thực mô hình



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

- Source code: <https://github.com/DangUIT/CE232.O21.git>
- Video demo: <https://youtu.be/4KVciXOP8pk>
- Cấu hình trong menuconfig
  - Cấu hình wifi

Example Connection Configuration

☒ connect using WiFi interface ⓘ

WiFi SSID ⓘ

MyWifi

WiFi Password ⓘ

101112131415

WiFi Scan Method ⓘ

All Channel ▼

- Trong phần wifi chọn software controls WiFi/Bluetooth coexistence

**Wi-Fi**

☒ Software controls WiFi/Bluetooth coexistence ⓘ

If enabled, WiFi & Bluetooth coexistence is controlled by software rather than hardware. Recommended for heavy traffic scenarios. Both coexistence configuration options are automatically managed, no user intervention is required. If only Bluetooth is used, it is recommended to disable this option to reduce binary file size.

Max number of WiFi static RX buffers ⓘ

10

Max number of WiFi dynamic RX buffers ⓘ

32

- Enable Bluetooth

**Bluetooth**

☒ Bluetooth ⓘ

**Bluetooth controller**

Bluetooth controller mode (BR/EDR/BLE/DUALMODE) ⓘ

BLE Only ▼

BLE Max Connections ⓘ

3

The cpu core which bluetooth controller run ⓘ

Core 0 (PRO CPU) ▼

HCI mode ⓘ

VHCI ▼

- Cấu hình Partition Table: Partition Table là một cấu trúc dữ liệu xác định cách bộ nhớ flash được tổ chức và chia thành các phần khác nhau. Các phần này bao gồm khu vực cho firmware, hệ thống tệp, NVS (Non-Volatile Storage), cập nhật OTA (Over-The-Air), và nhiều hơn nữa. Bảng phân vùng

rất quan trọng đối với ESP32 để biết nơi ứng dụng, dữ liệu và các tài nguyên khác được mong đợi được tìm thấy.

### Partition Table

Partition Table ⓘ

Custom partition table CSV ▾

The partition table to flash to the ESP32. The partition table determines where apps, data and other resources are expected to be found.

The predefined partition table CSV descriptions can be found in the components/partition\_table directory. These are mostly intended for example and development use, it's expect that for production use you will copy one of these CSV files and create a custom partition CSV for your application.

Custom partition CSV file ⓘ

partitions.csv

Offset of partition table ⓘ

0x8000

☒ Generate an MD5 checksum for the partition table ⓘ

partitions.csv

1	#	ESP-IDF	Partition	Table		
2	#	Name,	Type,	SubType,	Offset,	Size, Flags
3	nvs,	data,	nvs,	0x9000,	0x6000,	
4	phy_init,	data,	phy,	0xf000,	0x1000,	
5	factory,	app,	factory,	0x10000,	2M,	

- Sau khi cấu hình xong ấn Save
- Kết hợp 2 project đã làm ở bài tập 1 và bài tập BLE ở lab 3 và chỉnh sửa một số phần sau

```

case ESP_GATTS_WRITE_EVT:
    if (!param->write.is_prep){
        // the data length of gatts write must be less than GATTS_EXAMPLE_CHAR_VAL_LEN_MAX.
        if (gatt_db_handle_table[IDX_CHAR_CFG_C_2] == param->write.handle && param->write.len == 2){
            uint16_t descr_value = param->write.value[1]<<8 | param->write.value[0];
            uint8_t notify_data[2];
            notify_data[0] = 0xAA;
            notify_data[1] = 0xBB;

            if (descr_value == 0x0001){
                //the size of notify_data[] need less than MTU size
                esp_ble_gatts_send_indicate(gatts_if, param->write.conn_id, gatt_db_handle_table[IDX_CHAR_VAL_C],
                    sizeof(notify_data), notify_data, false);
                ESP_LOGI(EXAMPLE_TAG, "(6) ***** send notify AA BB ***** \n");
            }else if (descr_value == 0x0002){
                //the size of indicate_data[] need less than MTU size
                esp_ble_gatts_send_indicate(gatts_if, param->write.conn_id, gatt_db_handle_table[IDX_CHAR_VAL_C],
                    sizeof(notify_data), notify_data, true);
            }
            else if (descr_value == 0x0000){
                ESP_LOGI(EXAMPLE_TAG, "notify/indicate disable ");
            }else{
                ESP_LOGE(EXAMPLE_TAG, "unknown descr value");
                esp_log_buffer_hex(EXAMPLE_TAG, param->write.value, param->write.len);
            }
        }
    }

    /*Publish data from ble to broker*/
    if(gatt_db_handle_table[IDX_CHAR_VAL_B] == param->write.handle) {
        if(param->write.len>0){
            int msg_id;

            msg_id = esp_mqtt_client_publish(client, "esp32", (char*)param->write.value, 0, 0, 0);
            ESP_LOGI(TAG, "sent publish successful, msg_id=%d", msg_id);
        }
    }
    /*-----*/

```

- Ở trong hàm gatts\_profile\_event\_handler, trường hợp ESP\_GATTS\_WRITE\_EVT (ghi dữ liệu) ta thêm điều kiện nếu dữ liệu ghi vào IDX\_CHAR\_VAL\_B thì sẽ publish dữ liệu đó lên topic “esp32”

```

case MQTT_EVENT_DATA:
    ESP_LOGI(TAG, "MQTT_EVENT_DATA");
    printf("TOPIC=.%s\r\n", event->topic_len, event->topic);
    printf("DATA=.%s\r\n", event->data_len, event->data);

    /*Read data publish in ble*/
    esp_ble_gatts_set_attr_value(gatt_db_handle_table[IDX_CHAR_VAL_B], event->data_len, (uint8_t*)event->data);
    /*-----*/

    break;

```

- Ở trong hàm mqtt\_event\_handler, trường hợp nhận dữ liệu từ MQTT Broker.
  - Dữ liệu được nhận sẽ in ra màn hình cùng với thông tin về chủ đề (topic) tương ứng.

- Sau đó, dữ liệu nhận được từ broker MQTT được gán cho characteristic BLE có chỉ số `IDX_CHAR_VAL_B` trong bảng handle của cơ sở dữ liệu GATT. Điều này cho phép dữ liệu nhận được từ broker MQTT được chuyển đến thiết bị BLE.