ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ

BÁO CÁO BÀI TẬP LỚN MÔN HỌC MẠNG MÁY TÍNH
ĐỀ TÀI NHÀ THÔNG MINH

Thầy: Nguyễn Phan Hải Phú

Nhóm: 24

Họ Và Tên	MSSV
Trần Đình Khánh	2113716
Trần Minh Ý	1916069
Trần Đức Hảo	1734011

TP.HỒ CHÍ MINH 1/12/2024

1. GIỚI THIỆU

Yêu cầu và lựa chọn đề tài

Với yêu cầu về những kiến thức cần đạt được khi học môn học lập trình nhúng. Nắm vững các kiến thức cơ bản của ngôn ngữ C ứng dụng trong lập trình nhúng, nguyên lý lập trình nhúng, máy trạng thái, cách giao tiếp với phần cứng, hiểu về nguyên lý và lập trình thời gian thực (RTOS), tối ưu hoá hiệu năng sử dụng hiệu quả bộ nhớ và giao tiếp các thiết bị ngoại vi,Cả nhóm đã thảo luận thống nhất chọn đề tài về nhà thông minh sử dụng vi điều khiển ESP32 Serial Wi-Fi & Bluetooth Microcontroller SoC, để triển khai thực hiện nhằm ứng dụng các kiến thức đã được học trên lớp vào một bài toán nhỏ thực tế.

2. NỘI DỤNG THỰC HIỆN BÀI TẬP LỚN

2.1. ý tưởng và mục tiêu cho cho bài làm

2.1.1 mục tiêu

Xây dựng mô hình nhà thông minh đơn giản dung ESP32 để điều khiển các thiết bị trong nhà thông qua giao tiếp không dây

Tích hợp cảm biến để tăng tính tự động

2.1.2 Chức năng

Điều khiển đèn từ xa, tự động bật/tắt đèn dựa trên cảm biến chuyển động.

Điều khiển thiết bị gia dụng từ xa thông qua relay và giao tiếp không dây

Theo dõi nhiệt độ, cảm biến báo khói váo cháy khi phát hiện bất thường thông qua loa báo động và gửi thông báo cập nhật dữ liệu từ xa qua giao diện web hoặc ứng dụng di động.

2.1.3 Ý tưởng triển khai

Điều khiển đèn khi phát hiện chuyển động cảm biến PIR gửi tín hiệu ESP32 để bật đèn, giao diện web cho phép người dùng điều khiển đèn thủ công từ xa

Giám sát nhiệt độ, khói báo cháy, ESP32 đọc dữ liệu từ DHT22/BME280 và MQ-2/MQ-135, Gửi dữ liệu môi trường lên giao diện web hoặc hiển thị trên OLED, khi nồng độ khói từ cảm biến vượt ngưỡng hoặc nhiệt độ tăng bất thường sẽ kích hoạt cảnh báo âm thanh và gửi tin nhắn qua wifi

2.2. Tìm hiểu về các thành phần và Modul cần thiết

2.2.1 MCU ESP32

ESP32 là một vi điều khiển mạnh mẽ do Espressif Systems phát triển, được tích hợp Wi-Fi và Bluetooth, lý tưởng cho các ứng dụng IoT và hệ thống nhúng. Với CPU Xtensa LX6 (dual-core hoặc single-core) chạy ở xung nhịp tối đa 240 MHz, ESP32 cung cấp hiệu năng cao, hỗ trợ đa dạng cổng giao tiếp như GPIO, ADC, DAC, I2C, SPI, UART và PWM. Ngoài ra, ESP32 còn có RAM 520 KB, hỗ trợ PSRAM ngoài, và chế độ tiết kiệm năng lượng như deep sleep. Người

dùng có thể lập trình dễ dàng bằng Arduino IDE, ESP-IDF, hoặc PlatformIO. Với khả năng kết nối linh hoạt, tính năng bảo mật cao, và giá thành hợp lý, ESP32 phù hợp cho các ứng dụng nhà thông minh, giám sát môi trường, và hệ thống tự động hóa.

Sơ đồ pinout

Thông số kỹ thuật của ESP32

Loại: Wifi + Bluetooth Module

Cổng nạp: Type C || Micro (tùy chọn trong phần phân loại)

Mô hình: ESP32 38 chân

Điện áp nguồn (USB): 5V DC

Đầu vào/Đầu ra điện áp: 3.3V DC

Công suất tiêu thụ: 5µA trong hệ thống treo chế độ

Hiệu suất: Lên đến 600 DMIPS

Tần số: lên đến 240MHz

Wifi: 802.11 B/g/n/E/I (802.11N @ 2.4 GHz lên đến 150 Mbit/S)

Bluetooth: 4.2 BR/EDR BLE 2 chế độ điều khiển

Bộ nhớ: 448 Kbyte ROM, 520 Kbyte SRAM, 6 Kbyte SRAM trên RTC và QSPI Hỗ trợ đèn flash / SRAM chip

Chip USB-Serial: CP2102

Ăng ten: PCB

GPIO kỹ thuật số: 24 chân (một số chân chỉ làm đầu vào)

Kỹ thuật số Analog: 12bit SAR loại ADC, hỗ trợ các phép đo trên lên đến 18 kênh, một số chân hỗ trợ một bộ khuếch đại với lập trình tăng

Bảo mật: IEEE 802.11, bao gồm cả WFA, WPA/WPA2 và WAPI

Phần cứng tăng tốc mật mã học: AES, SHA-2, RSA, hình elip mật mã Đường Cong (ECC), số ngẫu nhiên Máy phát điện (RNG)

2.2.2 Cảm biến nhiệt độ và độ ẩm DHT11

Cảm biến DHT11

Thông số kỹ thuật của DHT11

Điện áp hoạt động: 5VDC

Chuẩn giao tiếp: TTL, 1 wire.

Khoảng đo độ ẩm: 20%-80%RH sai số ± 5%RH

Khoảng đo nhiệt độ: 0-50°C sai số ± 2°C

Tần số lấy mẫu tối đa 1Hz (1 giây / lần)

2.2.3 Cảm biến chuyển động PIR HC-SR501

PIR HC-SR501

Thông số kỹ thuật của PIR HC-SR501

Thời gian báo: 30 giây có thể tùy chỉnh bằng biến trở.

Độ nhạy có thể điều chỉnh bằng biến trở.

Kích thước: 1,27 x 0,96 x 1.0 (32,2 x 24,3 x 25,4 mm)

2.2.4 Cảm biến khí ga MQ2

Cảm biến khí ga MQ2

Thông số kỹ thuật

Phạm vi phát hiện: góc 360 độ hình nón, độ xa tối đa 6m.

Nhiệt độ hoạt động: 32-122 ° F (050 ° C)

Điện áp hoạt động: DC 3.8V – 5V

Mức tiêu thụ dòng: ≤ 50 uA

Điện áp hoạt động: 3.3V-5V

Kích thước PCB: 3cm * 1.6cm

Led đỏ báo nguồn vào, Led xanh báo gas

IC so sánh : LM393

VCC: 3.3V-5V

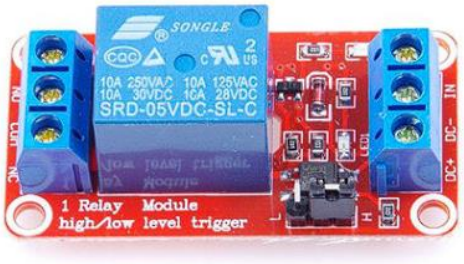
GND: 0V

DO: Đầu ra tín hiệu số (0 và 1)

AO: Đầu ra Analog (Tín hiệu tương tự)

Cấu tạo từ chất bản dẫn Sno2

2.2.5 Relay



Relay

Thông số kỹ thuật

Điện áp cuộn dây (coil): 5V DC

Dòng tiêu thụ cuộn dây: Khoảng 70mA

Điện áp điều khiển: 3.3V - 5V (tương thích ESP32)

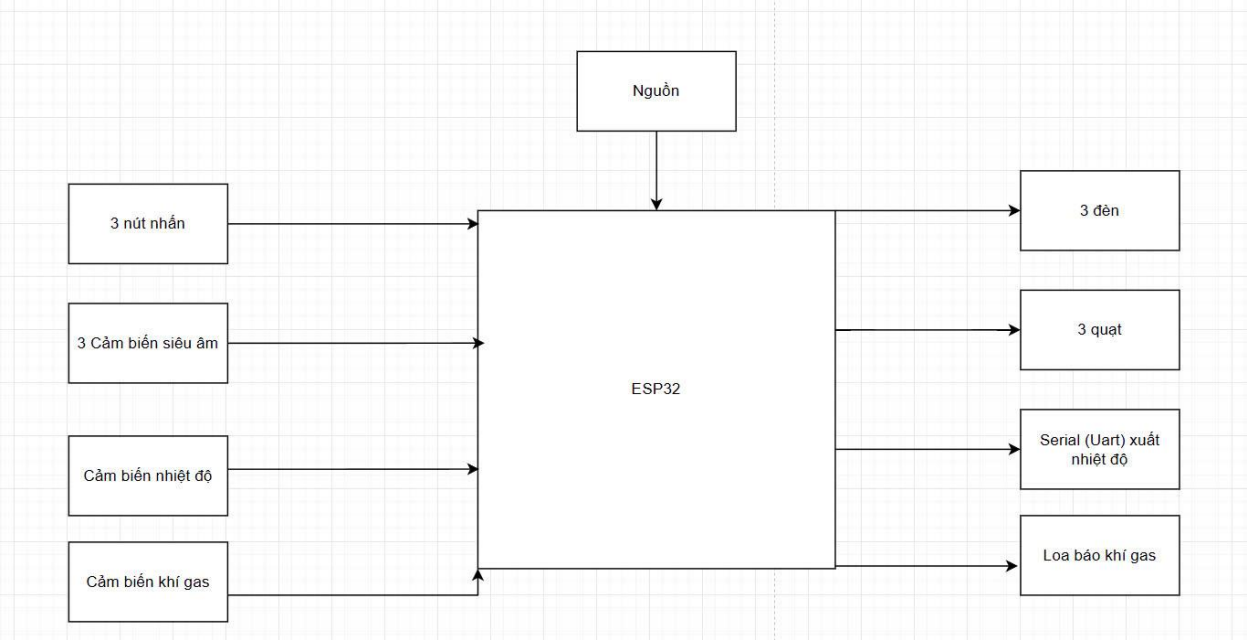
Loại tiếp điểm: SPDT (Single Pole Double Throw)

Điện áp tải tối đa: 250V AC hoặc 30V DC

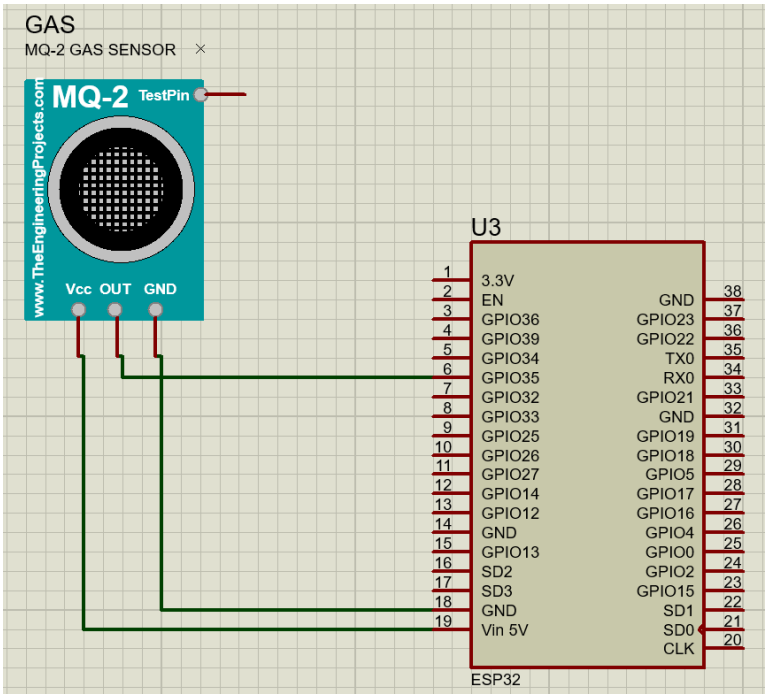
Dòng tải tối đa: 10A

Điện trở cách điện: > 100 MΩ (500V DC)

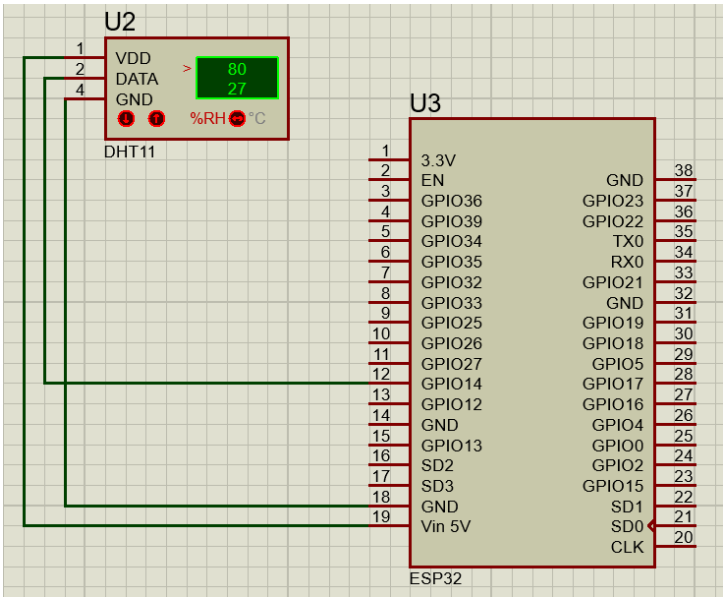
2.3. Sơ đồ kết nối phần cứng các thiết bị ngoại vi với ESP32.



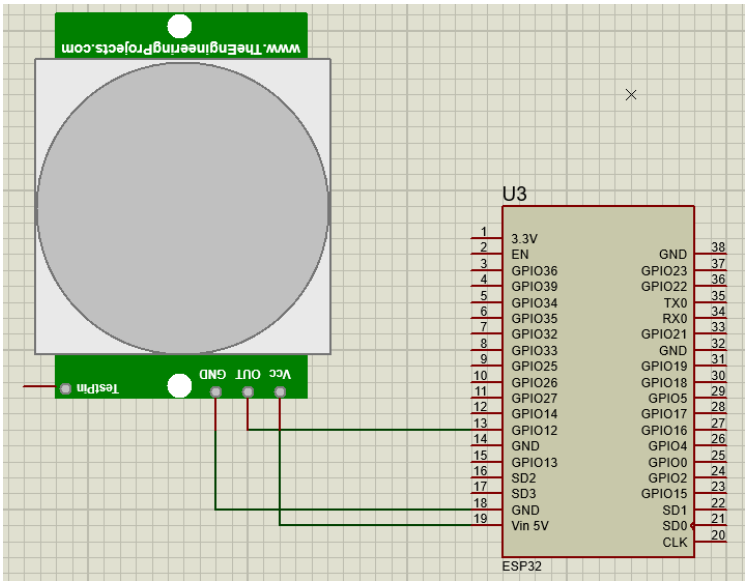
Sơ đồ kết nối các thiết bị ngoại vi với ESP32



Kết nối ESP32 với cảm biến khí ga

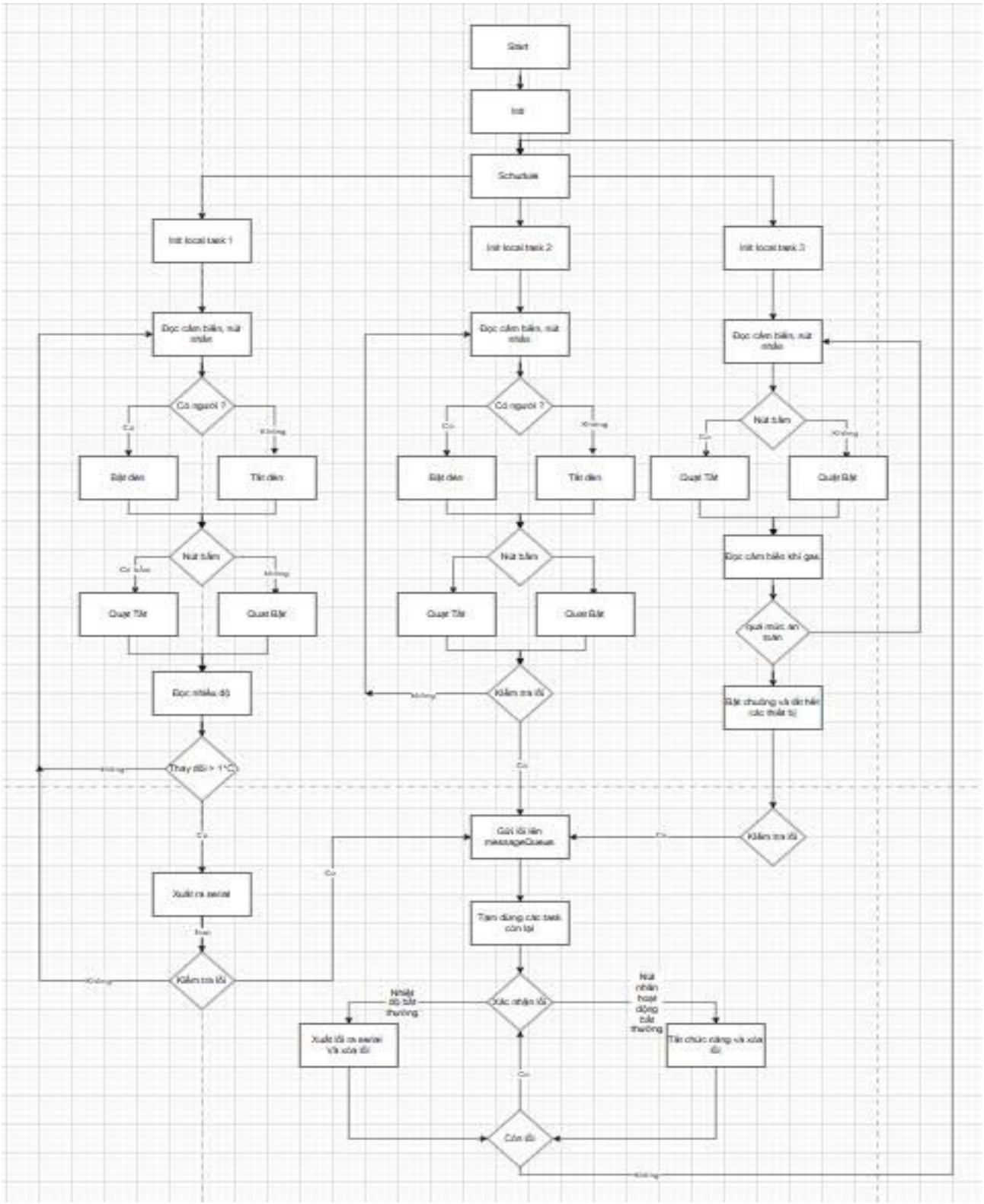


Kết nối ESP với cảm biến DHT11



Kết nối ESP32 với PIR sensor

2.4. Lưu đồ cho phần triển khai code



2.5. Code thực hiện trên ESP-IDF

Sau dây là chương trình nhúng sử dụng ESP-IDF, với mục đích điều khiển các thiết bị trong một ngôi nhà thông minh. Nó quản lý các thiết bị và cảm biến trong 3 khu vực: **phòng khách**, **phòng bếp**, và **phòng vệ sinh**. Các chức năng được xử lý thông qua các **task FreeRTOS** và giao tiếp bằng hàng đợi tin nhắn (**message queue**).

2.5.1 Khai báo thư viện.

```
#include "driver/gpio.h"
#include "esp_log.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"
#include "driver/uart.h"
```

Import các thư viện cần thiết để làm việc với ESP32:

gpio.h: Điều khiển các chân GPIO (Input/Output).

esp_log.h: Ghi log thông tin (debug).

FreeRTOS: Hỗ trợ quản lý đa nhiệm, task, queue.

adc.h: Đọc giá trị analog từ các cảm biến.

uart.h: Điều khiển giao tiếp UART.

2.5.2 Định nghĩa các macro

```
#define TAG "System"
#define BUF_SIZE 1024
#define MESSAGE_QUEUE_SIZE 5
#define UART_NUM UART_NUM_1
#define TXD_PIN 17
#define RXD_PIN 18
```

TAG “System” :Tên định danh cho log (hiển thị trong terminal).

BUF_SIZE: Kích thước bộ đệm UART.

MESSAGE_QUEUE_SIZE: Số lượng tin nhắn tối đa trong hàng đợi.

UART:

UART_NUM: Sử dụng UART1.

TXD_PIN, RXD_PIN: Chân truyền (TX) và nhận (RX) dữ liệu

2.5.3 Cấu hình GPIO các thiết bị

```
#define DOOR_SENSOR_PIN 15
#define BUTTON_PIN 16
#define FAN_PIN_LIVING 12
#define LIGHT_PIN_LIVING 14
#define TEMP_SENSOR_CHANNEL ADC1_CHANNEL_0 // GPIO 36 (ADC)
```

Chức năng:

GPIO:

Cảm biến cửa: Chân GPIO 15.

Nút nhấn: Chân GPIO 16.

Quạt và đèn phòng khách: GPIO 12 và 14.

Cảm biến nhiệt độ:

Kết nối qua kênh ADC1, chân GPIO 36.

2.5.4 Biến trạng thái của thiết bị

```
static int light_living_state = 0;
static int fan_living_state = 0;
```

Chức năng:

Lưu trạng thái **bật/tắt** của đèn và quạt trong các phòng.

0: Tắt, **1**: Bật.

Khai báo Task Handle và Message Queue

```
TaskHandle_t living_room_task_handle = NULL;
TaskHandle_t kitchen_task_handle = NULL;
TaskHandle_t bathroom_task_handle = NULL;
static QueueHandle_t message_queue;
static volatile int has_error = 0;
```

Chức năng:

Task Handle:

Quản lý các task của phòng khách, bếp, và vệ sinh.

Message Queue:

Hàng đợi tin nhắn dùng để giao tiếp giữa các task.

has_error:

Biến trạng thái báo lỗi.

2.5.5 Task quản lý phòng vệ sinh

```
void bathroom_task(void *pvParameters) {
    int prev_motion_state = 0;
    int prev_button_state = 0;
    int button_hold_time = 0;
```

Chức năng:

prev_motion_state: Lưu trạng thái trước đó của cảm biến chuyển động.

prev_button_state: Lưu trạng thái trước đó của nút nhấn.

button_hold_time: Đếm thời gian nút được nhấn giữ.

Xử lý cảm biến chuyển động

```
int motion_state = gpio_get_level(MOTION_SENSOR_BATHROOM_PIN);
if (motion_state == 1 && prev_motion_state == 0) {
    light_bathroom_state = !light_bathroom_state;
    gpio_set_level(LIGHT_PIN_BATHROOM, light_bathroom_state);
}
prev_motion_state = motion_state;
```

Chức năng:

Đọc trạng thái cảm biến chuyển động:

motion_state = 1: Phát hiện chuyển động.

Nếu có thay đổi trạng thái, chuyển đèn phòng vệ sinh sang bật/tắt.

2.5.6 Xử lý nút nhấn và quạt

```
int button_state = gpio_get_level(BUTTON_BATHROOM_PIN);
if (button_state == 1) {
    button_hold_time++;
    if (button_hold_time > 50) {
        char message[128];
        snprintf(message, sizeof(message), "Phòng vệ sinh: Nút nhấn bị giữ quá lâu (>5s)");
        xQueueSend(message_queue, &message, portMAX_DELAY);
    }
} else {
    if (button_hold_time > 0 && button_hold_time <= 50) {
        fan_bathroom_state = !fan_bathroom_state;
        gpio_set_level(FAN_PIN_BATHROOM, fan_bathroom_state);
    }
    button_hold_time = 0;
```

```
}
```

Chức năng:

Đếm thời gian nút được giữ.

Nếu nút giữ quá 5 giây, gửi thông báo lỗi qua **message queue**.

Nếu nhấn ngắn, chuyển trạng thái bật/tắt quạt

2.5.7 Task xử lý tin nhắn

```
void message_handler_task(void *pvParameters) {
    char received_message[128];
    while (1) {
        if (xQueueReceive(message_queue, &received_message, portMAX_DELAY)) {
            ESP_LOGW(TAG, "Thông báo lỗi nhận được: %s", received_message);

            has_error = 1;

            vTaskSuspend(living_room_task_handle);
            vTaskSuspend(kitchen_task_handle);
            vTaskSuspend(bathroom_task_handle);

            vTaskDelay(pdMS_TO_TICKS(5000));

            has_error = 0;

            vTaskResume(living_room_task_handle);
            vTaskResume(kitchen_task_handle);
            vTaskResume(bathroom_task_handle);
        }
    }
}
```

Chức năng:

Nhận tin nhắn từ hàng đợi.

Ghi log lỗi, tạm dừng các task đang chạy.

Sau khi xử lý lỗi, tiếp tục các task.

2.5.8 Hàm khởi tạo

```
void app_main() {
    gpio_config_t io_conf = {
        .pin_bit_mask = (1ULL << DOOR_SENSOR_PIN) | (1ULL << BUTTON_PIN) |
                        (1ULL << FAN_PIN_LIVING) | (1ULL << LIGHT_PIN_LIVING) |
                        (1ULL << BUTTON_KITCHEN_PIN) | (1ULL << FAN_PIN_KITCHEN) |
```

```
                (1ULL << LIGHT_PIN_KITCHEN) | (1ULL <<
MOTION_SENSOR_BATHROOM_PIN) |
                (1ULL << BUTTON_BATHROOM_PIN) | (1ULL << FAN_PIN_BATHROOM) |
                (1ULL << LIGHT_PIN_BATHROOM),
    .mode = GPIO_MODE_INPUT_OUTPUT,
    .pull_up_en = GPIO_PULLUP_ENABLE,
    .pull_down_en = GPIO_PULLDOWN_DISABLE,
    .intr_type = GPIO_INTR_DISABLE
};
gpio_config(&io_conf);
```

Chức năng:

Cấu hình các chân GPIO:

Input/Output.

Bật chế độ pull-up.

2.5.9 Tạo task

```
xTaskCreate(&living_room_task, "living_room_task", 4096, NULL, 5, &living_room_task_handle);
xTaskCreate(&kitchen_task, "kitchen_task", 4096, NULL, 5, &kitchen_task_handle);
xTaskCreate(&bathroom_task, "bathroom_task", 2048, NULL, 5, &bathroom_task_handle);
xTaskCreate(&message_handler_task, "message_handler_task", 2048, NULL, 5, NULL);
```

Chức năng:

Tạo các task để quản lý thiết bị trong từng phòng và xử lý thông báo lỗi.

3. Tổng kết và kinh nghiệm đạt được

3.1. Tổng kết

3.1.1 Mục tiêu hoàn thành:

Xây dựng hệ thống nhà thông minh quản lý nhiều thiết bị trong các khu vực khác nhau (phòng khách, bếp, phòng vệ sinh).

Sử dụng ESP-IDF để quản lý GPIO, giao tiếp UART, và xử lý ADC.

Tổ chức chương trình theo mô hình đa nhiệm với FreeRTOS, tạo các task riêng biệt cho từng phòng.

Xử lý lỗi bằng cơ chế message queue, ghi log để giám sát hệ thống và khôi phục hoạt động.

3.1.2 Các chức năng chính:

Điều khiển đèn, quạt và các cảm biến (chuyển động, nút nhấn) trong từng phòng.

Giao tiếp UART để truyền/nhận dữ liệu.

Xử lý thông báo lỗi và tạm dừng các task khi có sự cố.

Đọc dữ liệu từ cảm biến nhiệt độ thông qua ADC.

3.1.3 Ưu điểm của giải pháp:

Modular: Mỗi phòng là một module riêng, dễ bảo trì và mở rộng.

Xử lý lỗi tốt: Cơ chế hàng đợi giúp xử lý lỗi theo thứ tự, đồng thời đảm bảo hệ thống an toàn.

Tính hiệu quả: Task chạy độc lập, sử dụng tốt tài nguyên của ESP32.

3.2. Kiến thức đạt được và củng cố sau khi thực hiện

Sau khi thực hiện đề tài bài tập lớn nhóm đã biết cách xây dựng, thiết kế, cách hoạt động của một hệ thống nhúng. Được làm quen với kiến trúc ESP32 và cách giao tiếp, điều khiển ngoại vi với ESP32. Và làm quen với các kĩ năng khi thực hiện code một dự án nhỏ như sau

3.2.1 Quản lý GPIO:

Hiểu rõ cách cấu hình các chân GPIO ở chế độ input/output.

Tận dụng các chức năng pull-up/pull-down và interrupt để tăng độ tin cậy.

Tổ chức chương trình:

Việc sử dụng các Task cho từng phòng giúp chương trình rõ ràng hơn, tránh xung đột giữa các thành phần.

Message Queue là cách hiệu quả để giao tiếp giữa các task, đặc biệt khi xử lý thông báo lỗi.

3.2.2 Xử lý lỗi trong hệ thống:

Tạm dừng các task liên quan khi phát hiện lỗi và khôi phục khi lỗi được giải quyết giúp hệ thống hoạt động ổn định hơn.

Ghi log với ESP_LOG để dễ dàng debug và theo dõi trạng thái hoạt động.

3.2.3 Sử dụng ADC:

Học được cách đọc giá trị từ cảm biến thông qua kênh ADC.

Biết cách sử dụng ESP-IDF để hiệu chỉnh giá trị ADC đảm bảo độ chính xác

3.3. Kinh nghiệm nhóm rút ra từ việc thực hiện bài tập lớn

3.3.1 Lập kế hoạch kỹ càng trước khi code:

Xác định rõ chức năng của từng thành phần, mối quan hệ giữa chúng.

Tạo sơ đồ khối hoặc lưu đồ (flowchart) giúp hình dung cách tổ chức chương trình

3.3.2 Test từng module trước khi tích hợp:

Đảm bảo từng thành phần (ví dụ: GPIO, ADC, UART) hoạt động độc lập đúng cách trước khi tích hợp vào hệ thống lớn.

3.3.3 Quản lý lỗi là yếu tố quan trọng:

Thiết lập cơ chế phát hiện và xử lý lỗi là bước không thể thiếu để đảm bảo hệ thống hoạt động ổn định lâu dài.

3.3.4 Học cách debug hiệu quả:

Sử dụng log và UART để theo dõi trạng thái của hệ thống trong quá trình chạy.

Chạy thử với các trường hợp bất thường (nút nhấn giữ lâu, cảm biến không hoạt động, v.v.).

3.3.5 Tận dụng tài nguyên của ESP-IDF:

Sử dụng các API có sẵn của ESP-IDF giúp tiết kiệm thời gian và tăng tính ổn định của chương trình.

4. .Kết luận và cảm ơn

Bài tập này không chỉ giúp rèn luyện kỹ năng lập trình nhúng mà còn cung cấp kiến thức thực tiễn trong việc thiết kế hệ thống điều khiển nhúng phức tạp. Các kỹ năng quản lý tài nguyên và xử lý đồng bộ trong FreeRTOS sẽ rất hữu ích trong các dự án nhúng quy mô lớn sau này.

Bài tập này không chỉ giúp em cải thiện khả năng lập trình mà còn cho em một góc nhìn thực tế về việc triển khai và vận hành hệ thống điều khiển nhúng trong các dự án thực tế.

Cả nhóm xin gửi lời cảm ơn chân thành đến thầy đã tận tình hướng dẫn em trong suốt quá trình thực hiện bài tập này. Sự hỗ trợ và những góp ý quý báu của thầy đã giúp nhóm em hoàn thiện bài tập, đồng thời rút ra nhiều bài học kinh nghiệm quý giá.

5. Phụ lục

Các nguồn tham khảo

[1] Playlist Lập trình ESP 32-FREERTOS, https://s.net.vn/88Wr

[2] Playlist Lập trình ESP 32, https://s.net.vn/USYS

[3] Mô hình nhà thông minh - Smart Home K58 - ĐHBKHN, https://s.net.vn/9JXh

[4] Open ai chat GPT