THIẾT KẾ HỆ THỐNG NHÚNG KHÔNG DÂY

BÁO CÁO LAB01

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

1. Tìm hiểu về KIT ESP32 Lolin 32 (OLED 0.96)?

1.1. Bộ điều khiển trung tâm là gì? Các thông số kỹ thuật?

Bộ điều khiển trung tâm KIT ESP32 Lolin 32 (OLED 0.96") là một board phát triển phần cứng (hardware development board) dựa trên module WiFi + Bluetooth ESP32 của Espressif. ESP32 là một trong những module phổ biến nhất được sử dụng trong các ứng dụng IoT nhờ khả năng kết nối WiFi và Bluetooth, hiệu suất cao và tiêu thụ điện năng thấp.Các thông số kỹ thuật chính của KIT ESP32 Lolin 32 (OLED 0.96"):

- Module chính: ESP32-WROOM-32D
- Vi xử lý: Tổng hợp 2 nhân CPU Xtensa 32-bit LX6, tốc độ xung nhịp lên đến 240 MHz
- Bộ nhớ: 4MB Flash Memory và 520KB SRAM
- Giao tiếp: Wi-Fi 802.11 b/g/n và Bluetooth 4.2 BLE
- Cổng giao tiếp: 1 cổng Micro-USB, hỗ trợ chế độ OTG
- Màn hình hiển thị: màn hình OLED 0.96 inch tích hợp
- Đầu ra âm thanh: cổng jack 3.5mm, đầu ra mono
- Khe cắm thẻ nhớ: khe cắm thẻ nhớ microSD
- Pin: pin Lithium-Ion 3.7V, dung lượng 300mAh

• Kích thước: 51mm x 26mm x 7mm

• Trọng lượng: 8g

Ngoài các thông số kỹ thuật trên, KIT ESP32 Lolin 32 (OLED 0.96") còn có 2 nút nhấn để thực hiện các chức năng như nạp chương trình và khởi động lại board, và 26 chân GPIO để kết nối với các thiết bị ngoại vi khác.

1.2. Các chuẩn giao tiếp kèm theo chân giao tiếp tương ứng?

Các chuẩn giao tiếp của KIT ESP32 Lolin 32 (OLED 0.96") và các chân GPIO tương ứng:

• Wi-Fi: GPIO0, GPIO2, GPIO4, GPIO5, GPIO12, GPIO13, GPIO14, GPIO15, GPIO18, GPIO19, GPIO21, GPIO22, GPIO23, GPIO25, GPIO26, GPIO27

• Bluetooth: GPIO0, GPIO2, GPIO4, GPIO5, GPIO12, GPIO13, GPIO14, GPIO15, GPIO18, GPIO19, GPIO21, GPIO22, GPIO23, GPIO25, GPIO26, GPIO27

• UART: GPIO1 (TX), GPIO3 (RX)

• SPI: GPIO5 (CLK), GPIO18 (MISO), GPIO19 (MOSI), GPIO23 (CS)

• I2C: GPIO21 (SDA), GPIO22 (SCL)

• GPIO: GPIO0 - GPIO36

• Analog: GPIO32 - GPIO39

• USB: GPIO34 (input), GPIO12 (output)

1.3. Các chức năng của KIT? Có thể được sử dụng trong các ứng dụng như thế nào?

Các chức năng, bao gồm:

• Kết nối Wi-Fi và Bluetooth: KIT ESP32 Lolin 32 (OLED 0.96") được tích hợp với chuẩn Wi-Fi và Bluetooth, cho phép nó kết nối và tương tác với các thiết bị khác trong mạng không dây và giao tiếp không dây.

• Giao diện đồ họa OLED: KIT ESP32 Lolin 32 (OLED 0.96") được tích hợp với màn hình OLED 0,96 inch, cho phép hiển thị thông tin và hình ảnh với độ phân giải cao.

• Đa dạng các chuẩn giao tiếp: KIT ESP32 Lolin 32 (OLED 0.96") cung cấp các chuẩn giao tiếp như UART, SPI, I2C, GPIO và ADC, cho phép kết nối với các thiết bị ngoại vi khác nhau.

• Công nghệ Bluetooth Low Energy (BLE): KIT ESP32 Lolin 32 (OLED 0.96") hỗ trợ Bluetooth Low Energy (BLE), cho phép nó tương tác với các thiết bị BLE và thu thập thông tin từ các cảm biến BLE.

• Tích hợp bộ xử lý mạnh mẽ: KIT ESP32 Lolin 32 (OLED 0.96") được trang bị với bộ xử lý hai nhân, tốc độ xung nhịp cao và bộ nhớ lớn, cho phép xử lý dữ liệu và tính toán nhanh chóng và hiệu quả. Thiết bị Internet of Things (IoT): KIT ESP32 Lolin 32 (OLED 0.96") là một giải pháp lý tưởng cho các ứng dụng IoT như cảm biến thông minh, hệ thống điều khiển, hệ thống giám sát và hệ thống đo lường.

• Ứng dụng điều khiển từ xa: KIT ESP32 Lolin 32 (OLED 0.96") có thể được sử dụng để phát triển các ứng dụng điều khiển từ xa như điều khiển các thiết bị gia đình, điều khiển đèn chiếu sáng, hệ thống đóng mở cửa tự động và hệ thống an ninh.

• Các ứng dụng y tế và sức khỏe: KIT ESP32 Lolin 32 (OLED 0.96") có thể được sử dụng để phát triển các thiết bị y tế và sức khỏe như theo dõi sức khỏe, cảnh báo y tế và các thiết bị hỗ trợ y tế.

• Các ứng dụng đo lường và kiểm tra: KIT ESP32 Lolin 32 (OLED 0.96") có thể được sử dụng để phát triển các thiết bị đo lường và kiểm tra như máy đo nhiệt độ, độ ẩm, áp suất, độ rung và độ bền.

• Các ứng dụng giải trí: KIT ESP32 Lolin 32 (OLED 0.96") có thể được sử dụng để phát triển các ứng dụng giải trí như đồ chơi điện tử, máy trò chơi và hệ thống âm thanh.

• Các ứng dụng mạng xã hội: KIT ESP32 Lolin 32 (OLED 0.96") có thể được sử dụng để phát triển các ứng dụng mạng xã hội như chatbot, hệ thống nhắn tin và kết nối mạng xã hội.

• Các ứng dụng tương tác người-máy: KIT ESP32 Lolin 32 (OLED 0.96") có thể được sử dụng để phát triển các ứng dụng tương tác ngườimáy như robot, hệ thống nhận diện giọng nói và hệ thống điều khiển bằng giọng nói.

2. Viết chương trình chớp tắt LED trên ESP32, chu kỳ là 1 giây. Mỗi lần đèn đổi trạng thái, in trạng thái ra terminal?

GITHUB:
https://github.com/DangUIT/CE232.O21?fbclid=IwAR2TMuaQB1rIMXFKfTGygXl8CXP0EDp32k4yREk58yLu85rsA__zRpj5Mrk

VIDEO DEMO: https://www.youtube.com/shorts/1P5svVXpLWo

Giới thiệu về KIT ESP32 được nhóm sử dụng: ESP32 ESP-WROOM-32

Giải thích code:

- Đầu tiên, các thư viện và file header cần thiết được import. Các thư viện freertos/task.h, driver/gpio.h, esp_log.h, led_strip.h được sử dụng để tương tác với các thành phần của ESP32 như GPIO và LED, và file sdkconfig.h được sử dụng để lấy các cấu hình từ menuconfig trong quá trình biên dịch.

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include "led_strip.h"
#include "sdkconfig.h"
```

- BLINK_GPIO được định nghĩa để xác định GPIO mà đèn LED sẽ được kết nối đến. Giá trị của nó được đặt trong tệp cấu hình của SDK. Ở đây là chân số 5.

```c
#define BLINK_GPIO CONFIG_BLINK_GPIO
```

```c
#define CONFIG_BLINK_GPIO 5
```

- Khai báo một biến static uint8_t s_led_state với giá trị là 0 ể lưu trữ trạng thái hiện tại của LED.

```c
static uint8_t s_led_state = 0;
```

- Hàm blink_led() được sử dụng để điều khiển đèn LED. Trong hàm này:
  - Hàm gpio_set_level() được gọi để đặt mức logic (LOW hoặc HIGH) cho GPIO được chỉ định bởi BLINK_GPIO dựa trên trạng thái hiện tại của biến s_led_state.
  - Khi s_led_state có giá trị 0, thì GPIO sẽ được đặt ở mức thấp (LOW),đèn tắt. Khi s_led_state có giá trị 1, GPIO sẽ được đặt ở mức cao (HIGH), đèn bật.

```c
static void blink_led(void)
{
    /* Set the GPIO level according to the state (LOW or HIGH)*/
    gpio_set_level(BLINK_GPIO, s_led_state);
}
```

- Hàm configure_led() được sử dụng để cấu hình GPIO để điều khiển đèn LED.
    - ESP_LOGI(TAG, "Example configured to blink GPIO LED!"); :Dòng này ghi một thông điệp log với mức độ INFO sử dụng hàm ESP_LOGI() từ ESP-IDF Logging API. Thông điệp này sẽ xuất hiện trong log của hệ thống với nhãn được xác định bởi biến TAG.
    - gpio_reset_pin(BLINK_GPIO);: Hàm này được sử dụng để đặt trạng thái ban đầu của GPIO được chỉ định bởi BLINK_GPIO.
    - gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT); : Hàm này được sử dụng để đặt hướng của GPIO. Trong trường hợp này, GPIO được đặt để hoạt động như một đầu ra (output). Điều này cho phép chương trình ghi dữ liệu vào GPIO để điều khiển đèn LED.

```c
static void configure_led(void)
{
    ESP_LOGI(TAG, "Example configured to blink GPIO LED!");
    gpio_reset_pin(BLINK_GPIO);
    /* Set the GPIO as a push/pull output */
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);
}
```

- Hàm app_main() là hàm chính của chương trình và là nơi mà quá trình chạy của chương trình bắt đầu.
    - configure_led();: Hàm này được gọi để cấu hình GPIO để điều khiển đèn LED, như đã mô tả trước đó.
    - while (1) { ... }: Đây là một vòng lặp vô hạn, nghĩa là sau khi các cài đặt ban đầu được thực hiện, chương trình sẽ tiếp tục chạy trong vòng lặp này mãi mãi.
    - blink_led(); : Trong mỗi lần lặp, hàm blink_led() được gọi để chuyển đổi trạng thái của đèn LED.
    - s_led_state = !s_led_state;: Dòng này đảo ngược trạng thái của biến s_led_state, từ 0(OFF) sang 1(ON) hoặc ngược lại.

- ESP_LOGI(TAG, "LED %s!", s_led_state == true ? "ON" : "OFF");: Dòng này ghi một thông điệp log với mức độ INFO, cho biết trạng thái hiện tại của đèn LED là ON hoặc OFF, dựa vào giá trị của biến s_led_state.
- vTaskDelay(CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);: được sử dụng để tạm ngừng thực hiện của một task trong FreeRTOS trong một khoảng thời gian nhất định. Ở đây được sử dụng để tạo chu kì chớp tắt của LED.
  - CONFIG_BLINK_PERIOD: Đây là một hằng số được định nghĩa trong tệp sdkconfig.h. Nó xác định thời gian chờ giữa các lần nhấp nháy của đèn LED.

```
#define CONFIG_BLINK_PERIOD 1000
```

  - portTICK_PERIOD_MS: Đây là một hằng số trong FreeRTOS xác định độ dài của mỗi tick (thời gian nhỏ nhất có thể đo lường được trong hệ thống) dưới dạng miligiây. Ở đây được đặt là 10ms. Có nghĩa là có 100 ticks mỗi giây.

Tóm lại, dù portTICK_PERIOD_MS được đặt là bao nhiêu, nhưng CONFIG_BLINK_PERIOD được đặt là 1000 thì mỗi lần vòng lặp chờ n ticks, tương đương với 1 giây (với mỗi tick có thời gian là portTICK_PERIOD_MS ms), điều này giúp tạo ra chu kỳ nhấp nháy đèn LED mỗi giây.

```
void app_main(void)
{

    /* Configure the peripheral according to the LED type */
    configure_led();

    while (1) {
        //ESP_LOGI(TAG, "Turning the LED %s!", s_led_state == true ? "ON" : "OFF");
        blink_led();
        /* Toggle the LED state */
        s_led_state = !s_led_state;
        ESP_LOGI(TAG, "LED %s!", s_led_state == true ? "ON" : "OFF");
        vTaskDelay(CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
}
```