Tuần 1 - Hệ điều hành FreeRTOS

Giới thiệu về FreeRTOS

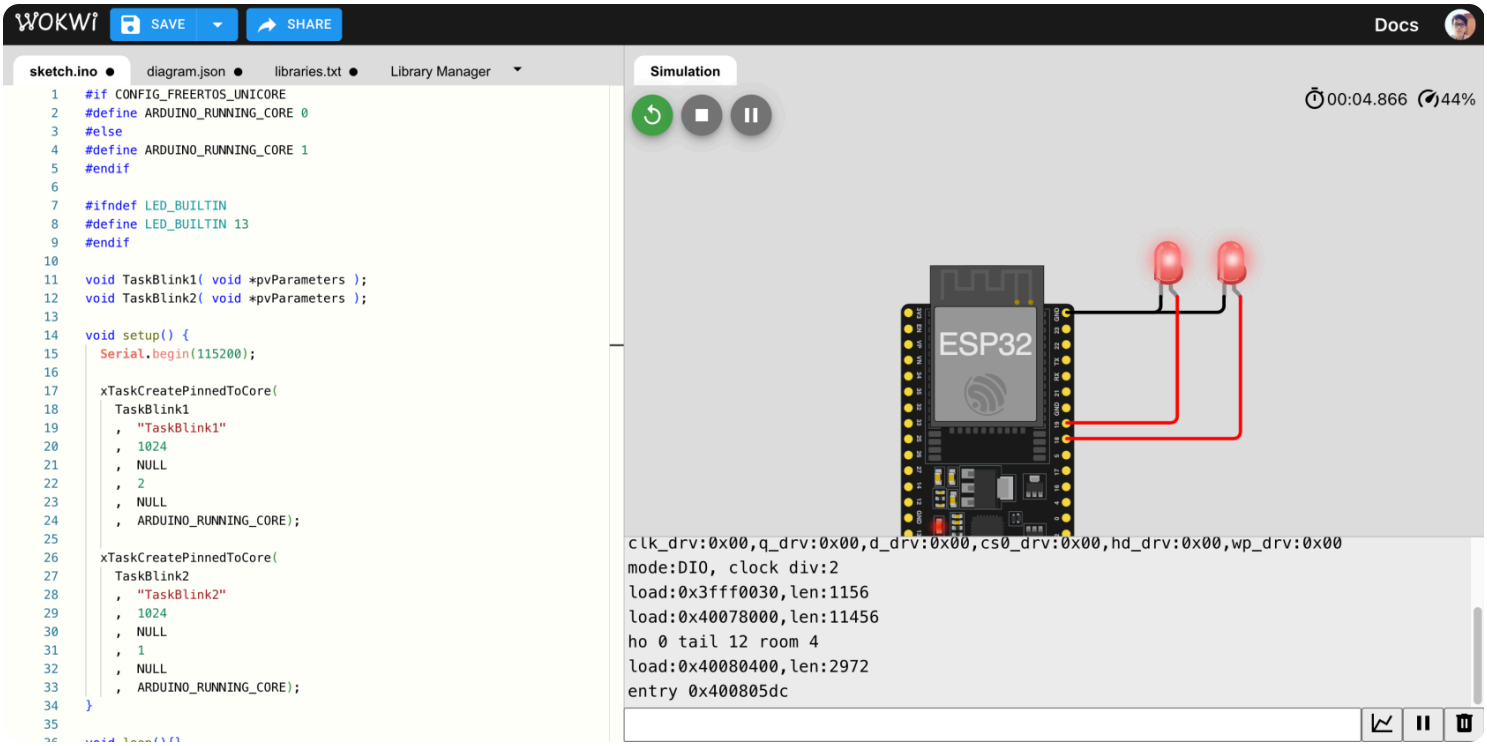
FreeRTOS là một hệ điều hành thời gian thực (Real-Time Operating System - RTOS) miễn phí và mã nguồn mở được thiết kế cho các hệ thống nhúng. Nó cung cấp các tính năng đa nhiệm, đồng bộ hóa, quản lý bộ nhớ và giao tiếp giữa các tác vụ. FreeRTOS Arduino là sự tích hợp của FreeRTOS vào nền tảng Arduino. Nó cho phép các dự án Arduino sử dụng các tính năng của FreeRTOS để xử lý đa nhiệm và các yêu cầu thời gian thực

Cài đặt

Thư viện FreeRTOS tại (Dành riêng cho bo mạch Arduino): https://github.com/feilipu/Arduino_FreeRTOS_Library

Code tham khảo trên dịch vụ https://wokwi.com/:

Thực hiện đấu nối 2 LED và điều khiển theo mô hình song song trên RTOS.



Code tham khảo:

```
#if CONFIG_FREERTOS_UNICORE
#define ARDUINO_RUNNING_CORE 0
#else
#define ARDUINO_RUNNING_CORE 1
#endif

```

```
#ifndef LED_BUILTIN
#define LED_BUILTIN 13
#endif

void TaskBlink1( void *pvParameters );
void TaskBlink2( void *pvParameters );

void setup() {
  Serial.begin(115200);

  xTaskCreatePinnedToCore(
    TaskBlink1
    ,  "TaskBlink1"
    ,  1024
    ,  NULL
    ,  2
    ,  NULL
    ,  ARDUINO_RUNNING_CORE);

  xTaskCreatePinnedToCore(
    TaskBlink2
    ,  "TaskBlink2"
    ,  1024
    ,  NULL
    ,  1
    ,  NULL
    ,  ARDUINO_RUNNING_CORE);
}

void loop(){}

void TaskBlink1(void *pvParameters)
{
  (void) pvParameters;
  pinMode(19, OUTPUT);

  for (;;)
  {
    digitalWrite(19, HIGH);
    vTaskDelay(100);
    digitalWrite(19, LOW);
    vTaskDelay(100);
  }
}

void TaskBlink2(void *pvParameters)
{
  (void) pvParameters;
  pinMode(18, OUTPUT);

  for (;;)
```

```
  {
    digitalWrite(18, HIGH);
    vTaskDelay(100);
    digitalWrite(18, LOW);
    vTaskDelay(100);
  }
}
```

1. Delay function

Hàm `delay()` thường được sử dụng trong Arduino. Nhưng trong trường hợp ESP32 với RTOS, chúng ta sử dụng hàm `vTaskDelay()`. Điểm khác biệt chính là khi chúng ta sử dụng `delay()`, toàn bộ quá trình vẫn được giữ trong một khoảng thời gian cụ thể. Nhưng nếu chúng ta tính đến `vTaskDelay()`, nó chỉ ảnh hưởng đến việc thực thi công việc cụ thể mà nó được gọi bên trong.

```
void vTaskFunction( void * pvParameters )
{
    // Block for 500ms.
    const TickType_t xDelay = 500 / portTICK_PERIOD_MS;
    for( ;; )
    {
        // Simply toggle the LED every 500ms, blocking between each toggle.
        vToggleLED();
        vTaskDelay( xDelay );
    }
}
```

2. Queue

Trong FreeRTOS, Queue (hàng đợi) là một cơ chế giúp các task có thể giao tiếp và đồng bộ hóa với nhau bằng cách trao đổi dữ liệu thông qua một hàng đợi. Queue hoạt động theo nguyên tắc FIFO (First-In-First-Out), tức là dữ liệu được đưa vào queue trước sẽ được lấy ra trước.

```
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/queue.h>

// Task handlers
TaskHandle_t task1Handle;
TaskHandle_t task2Handle;

// Queue handler
QueueHandle_t sendingQueue;

// Task function for Task 1
void task1(void *parameter) {
  while (true) {
    int randomNumber = random(100);
    Serial.print("Random number in Task 1: ");
    Serial.println(randomNumber);

```

```
    if (xQueueSend(sendingQueue, &randomNumber, 0) != pdTRUE) {
21    Serial.println("Failed to send data to queue");
22    }
23
24    vTaskDelay(500 / portTICK_PERIOD_MS); // Delay for 500 milliseconds
25  }
26 }
27
28 // Task function for Task 2
29 void task2(void *parameter) {
30   int receivedNumber;
31   while (true) {
32     if (xQueueReceive(sendingQueue, &receivedNumber, portMAX_DELAY) == pdTRUE) {
33       Serial.print("Received number in Task 2: ");
34       Serial.println(receivedNumber);
35     }
36   }
37 }
38
39 void setup() {
40   Serial.begin(115200);
41
42   // Create the queue
43   sendingQueue = xQueueCreate(1, sizeof(int));
44
45   // Create tasks
46   xTaskCreatePinnedToCore(task1, "Task 1", 2048, NULL, 1, &task1Handle, 0);
47   xTaskCreatePinnedToCore(task2, "Task 2", 2048, NULL, 1, &task2Handle, 1);
48 }
49
50 void loop() {
51   // Empty loop
52 }
```

3. Semaphore

Semaphore được dùng để động bộ giữa các

- Task
- Ngắt và task

Có 2 dạng là

- Binary semaphore: Binary semaphore có duy nhất 1 token và sử dụng để đồng bộ một action
- Counting semaphore: Counting semaphore sẽ có nhiều token và đồng bộ nhiều action khác nhau

Bài tập

Câu 1: Thực hiện sử dụng thư viện FreeRTOS để thực hiện các yêu cầu sau:

a. Lập trình điều khiển song song 2 LED:LED 1 bật 2s và tắt sau mỗi 3s; LED 2 bật trong 2.5s và sau mỗi 0.5s? Hiển thị trạng thái của các đèn LED lên màn hình LCD?

b. Đấu nối thêm nút nhấn D vào bo mạch, khi nhấn nút D thực hiện bật/tắt LED 1 và LED 2 xen kẽ nhau (LED 1 bật thì LED 2 tắt và ngược lại)?

Câu 2: Thực hiện sử dụng thư viện FreeRTOS với Queue để thực hiện các yêu cầu sau:

a. Tạo Task1 với nhiệm vụ đọc dữ liệu nhiệt độ từ cảm biến DHT11 và đưa dữ liệu vào Queue?

b. Tạo Task2 với nhiệm vụ hiển thị dữ liệu lên màn hình LCD từ dữ liệu lấy được từ Queue?

c. Tạo Task3 đọc dữ liệu từ cảm biến ánh sáng thấp hơn mức ánh sáng môi trường tự nhiên (< 1000 Lux) thì bật đèn?

Câu 3: Sử dụng Semaphore để đồng bộ hóa các tác vụ với thư viện FreeRTOS thực hiện các yêu cầu sau:

a. Lập trình điều khiển 3 đèn LED với 3 hiệu ứng nhấp nháy khác nhau. Sử dụng FreeRTOS với Semaphore để đồng bộ hóa giữa các tác vụ.

b. Tạo cơ chế để ưu tiên tác vụ cao nhất có quyền điều khiển LED trước, ưu tiên ngay cả khi các tác vụ khác đang thực thi.

c. Thêm 2 nút nhấn để điều khiển việc bật hoặc tắt các tác vụ nhấp nháy LED. Khi nhấn nút 1, các tác vụ liên quan đến LED 1 và LED 2 sẽ bị tạm dừng, chỉ còn tác vụ của LED 3 hoạt động. Nhấn nút 2 sẽ kích hoạt lại tất cả các tác vụ.