#include <Arduino.h>

Esta línea incluye la librería principal de Arduino, que proporciona todas las funciones y definiciones necesarias para programar el ESP32.

const int ledPin = 2; // Usualmente el pin 2 en los ESP32 está conectado a un LED integrado

Define una constante ledPin para especificar el pin donde está conectado el LED. En muchos módulos ESP32, el pin 2 está conectado a un LED integrado.

volatile int randomNumber = 0; // Variable global para almacenar el número aleatorio

Declara una variable global randomNumber para almacenar el número aleatorio generado por Tarea1. El modificador volatile indica que la variable puede ser modificada por diferentes contextos (como interrupciones o tareas de RTOS).

TaskHandle\_t Task2Handle = NULL; // Manejador de la tarea 2

Declara un manejador Task2Handle para la tarea 2, que será utilizado para enviar notificaciones entre tareas.

void setup() {

// Inicializa la comunicación serial

Serial.begin(115200);

delay(1000);

// Configura el pin del LED como salida

pinMode(ledPin, OUTPUT);

// Crea las tareas

xTaskCreate(Tarea1, "Tarea1", 10000, NULL, 1, NULL);

xTaskCreate(Tarea2, "Tarea2", 10000, NULL, 1, &Task2Handle);

}

Inicia la comunicación serial a una velocidad de 115200 baudios y espera 1 segundo para asegurarse de que la inicialización esté completa.

Configura el pin ledPin (pin 2) como una salida digital. Esto es necesario para poder encender y apagar el LED.

Tarea1: Genera números aleatorios y los envía por UART cada segundo.

Tarea2: Enciende el LED durante 200 ms cuando recibe una notificación de Tarea1.

void loop() {

// El loop principal no hace nada, las tareas se ejecutan independientemente

delay(1000);

}

El loop principal no hace nada aparte de una espera de 1 segundo. En este programa, todas las acciones importantes son manejadas por las tareas creadas.

void Tarea1(void \* parameter) {

// Tarea 1: Generar números aleatorios y enviarlos vía UART cada segundo

while (true) {

randomNumber = random(0, 100); // Genera un número aleatorio entre 0 y 99

Serial.println(randomNumber); // Envía el número por UART

// Notifica a la tarea 2 que se ha enviado un número

xTaskNotifyGive(Task2Handle);

vTaskDelay(1000 / portTICK\_PERIOD\_MS); // Espera 1 segundo

}

vTaskDelete(NULL);

}

En cada iteración del bucle, Tarea1 genera un número aleatorio entre 0 y 99, lo imprime en el monitor serial y notifica a Tarea2 que se ha enviado un número. Luego, espera 1 segundo antes de repetir el proceso.

Esta línea elimina la tarea actual. No se alcanza debido al bucle infinito, pero se incluye por buenas prácticas.

void Tarea2(void \* parameter) {

// Tarea 2: Encender el LED durante 200 ms cuando se envíe un número

while (true) {

// Espera una notificación de la tarea 1

ulTaskNotifyTake(pdTRUE, portMAX\_DELAY);

digitalWrite(ledPin, HIGH); // Enciende el LED

vTaskDelay(200 / portTICK\_PERIOD\_MS); // Espera 200 ms

digitalWrite(ledPin, LOW); // Apaga el LED

}

vTaskDelete(NULL);

}

En cada iteración del bucle, Tarea2 espera una notificación de Tarea1 utilizando ulTaskNotifyTake. Cuando recibe una notificación, enciende el LED, espera 200 ms y luego lo apaga. Luego vuelve a esperar una nueva notificación.

Al igual que en Tarea1, esta línea elimina la tarea actual. No se alcanza debido al bucle infinito, pero se incluye por buenas prácticas.