Sistema de implementación de periféricos con freeRTOS en LPC-845

Curso: Sistemas Embebidos UTN-FRA Avellaneda

Integrantes y correos

- Vicente Seta – vicenteseta@impatrq.com
- Lautaro Cabeza – lautarovalentincabeza@impatrq.com

Resumen–Abstract

Este proyecto implementa un sistema de control de iluminación inteligente utilizando el microcontrolador LPC-845 con FreeRTOS. El sistema mide la luz ambiente mediante el sensor BH1750, ajusta la intensidad de LEDs RGB mediante PWM, y permite visualizar y configurar un setpoint deseado a través de un display de 7 segmentos y botones. Además, se incorpora un sensor infrarrojo para detección de movimiento, que activa una alarma acústica. Todo el funcionamiento se organiza en tareas concurrentes gestionadas por un sistema operativo en tiempo real.

I. Introducción y problema planteado

El objetivo del presente trabajo es diseñar e implementar un sistema de control de iluminación que se adapte dinámicamente a las condiciones lumínicas del entorno.
La solución debe ser capaz de:

- Medir la luz ambiente mediante un sensor digital (BH1750).
- Ajustar la intensidad de LEDs RGB en función del nivel de luz respecto de un setpoint configurable.
- Permitir al usuario observar y modificar el setpoint a través de botones y display.
- Detectar movimiento con un sensor infrarrojo (IR) y activar un buzzer como alarma.
- Ejecutar todas las tareas de forma eficiente y en tiempo real utilizando FreeRTOS.

El desarrollo contempla la inicialización de periféricos, la creación de tareas independientes para cada función, y el uso de semáforos y colas para sincronización y comunicación entre tareas.

II. Diseño e implementación

En la aplicación desarrollada se utilizan los componentes de los sistemas operativos a nuestro favor para así garantizar el correcto funcionamiento de la misma.
En efecto, estamos hablando de colas, empleadas para establecer la

comunicación de datos entre las tareas; semáforos mutex esencialmente para que las tareas accedan a un determinado recurso si y sólo si este no está siendo utilizado por otra tarea, evitando que se “peleen” por un mismo recurso; y el scheduler que finalmente levanta el sistema operativo y decide en función a las prioridades asignadas qué tarea se debe ejecutar en cada momento.

El sistema está compuesto por múltiples tareas en FreeRTOS, cada una encargada de una función específica. Se trabajó sobre un microcontrolador **LPC845**, con inicialización de reloj a 30 MHz. A continuación se describen las principales tareas y sus responsabilidades:

A. Sensor de luz BH1750

Se implementa una tarea (`Task_Lux`) encargada de obtener el valor de luminosidad ambiente desde el sensor BH1750 mediante el protocolo I2C. El dato es convertido a un porcentaje teniendo en cuenta que 30000 lux equivale al 100%.

B. Ajuste del setpoint

Mediante botones conectados a entradas digitales, se puede incrementar o disminuir el setpoint de iluminación en pasos de 1%. La tarea `Task_Botones` gestiona estas entradas y se asegura de mantener el setpoint dentro del rango 25%–75%.

C. Display 7 segmentos

La tarea `Task_Display` muestra en el display el valor del setpoint o la luminosidad actual. Se alterna entre ambos al presionar el botón USER.

D. Control de LED RGB por PWM

En la tarea `Task_ControlLED_Blue y Task_ControlLED_Red`, se realiza el control proporcional de los LEDs del tricolor. Si la luz ambiente está por debajo del setpoint, se enciende el LED azul con una intensidad proporcional a la diferencia; si está por encima, se enciende el LED rojo de igual forma.

E. Control del brillo de LED azul con RV22

Un potenciómetro conectado a una entrada ADC ajusta el brillo del LED azul (independiente del control RGB anterior), entre 0% y 100%. La tarea `Task_ADC` convierte el valor analógico del RV22 y actualiza el duty cycle de un PWM.

F. Detección de obstáculo (sensor IR) y buzzer

Cuando el sensor IR detecta un obstáculo, se activa un buzzer mediante la tarea `Task_IR`. Se implementa antirrebote y se evita la activación continua del sonido.

G. Inicialización

`Task_Init` tarea se encarga de inicializar todos los periféricos, incluyendo pines, ADC, I2C, PWM, y el display de 7 segmentos.

H. Organización del sistema (main.c)

Todas las tareas se crean en el `main` con sus prioridades respectivas. Luego se inicia el planificador.

III. Resultados

Se realizaron pruebas unitarias para cada tarea, así como pruebas integradas para verificar la interacción entre tareas. Los resultados son los siguientes:

El sensor BH1750 proporciona lecturas estables y confiables. El control de LEDs responde correctamente ante variaciones de luz ambiente. El display alterna correctamente entre el setpoint y la lectura de luz. Los botones responden sin rebotes visibles. El buzzer se activa correctamente al tapar el sensor IR. La consola muestra información clara y actualizada cada segundo. El sistema se mantiene estable sin reinicios ni cuelgues.

IV. Conclusiones

El proyecto cumplió satisfactoriamente con todos los requerimientos propuestos. Se logró una implementación modular y eficiente, con uso intensivo de FreeRTOS para la separación de responsabilidades. La utilización del sensor BH1750, el control PWM y la lectura de entradas ADC y digitales fue correctamente integrado.