Aplicación de FreeRTOS con Sensado de Luminosidad y Control de Periféricos en LPC845

Bianco Tomás
tobianco2007@gmail.com

Lin Rodrigo
alexisrodrigolin@gmail.com

***Resumen*—Se desarrolló una aplicación en FreeRTOS para el microcontrolador LPC845 que integra sensado de luz, control de LEDs, display, buzzer y botones, utilizando colas y semáforos para la comunicación entre tareas. La intensidad luminosa se mide con un sensor BH1750 y se compara con un setpoint ajustable mediante botones. También se ajusta la iluminación con un potenciómetro y se da feedback sonoro usando un buzzer activado por un sensor infrarrojo.**

***Index Terms*—FreeRTOS, BH1750, PWM, buzzer, setpoint, semáforos, colas, LPC845, tareas concurrentes, MCUXpresso**

I. Introducción

En este trabajo se explora el desarrollo de una aplicación embebida utilizando FreeRTOS sobre un microcontrolador LPC845. El objetivo es gestionar múltiples tareas en paralelo que interactúan con diversos periféricos: sensor de luz, LEDs RGB, display de 7 segmentos, botones de usuario, un potenciómetro, sensor infrarrojo y buzzer. Se utilizan mecanismos de comunicación como colas y semáforos para coordinar las tareas en un entorno de tiempo real.

II. Desarrollo del Sistema

II-A. Descripción general

El sistema mide la luminosidad ambiental mediante un sensor BH1750. Esta lectura se compara con un setpoint definido por el usuario mediante botones. Según esta comparación, se controla la intensidad de un LED rojo o azul. Además, se utiliza un potenciómetro para controlar el brillo del LED azul. La información se muestra alternadamente en un display de 7 segmentos. Cuando el sensor infrarrojo detecta una obstrucción, se activa un buzzer. También se imprime información por consola cada segundo.

II-B. Tareas y funcionalidad

A continuación, se describen las tareas utilizadas y su comunicación:

II-C. Comunicación entre tareas

Las tareas se comunican utilizando mecanismos de FreeRTOS: colas para pasar valores de luminosidad, ADC y setpoint y control de display y semáforos binarios o counting para activar tareas como el buzzer, la selección de setpoint y la lectura del botón USER. El diseño permite una sincronización eficiente y un uso óptimo de los recursos del sistema embebido

Cuadro I
Tareas e ISRs del sistema

Tarea/ISR	Función	Comunicación
`task_init`	Inicializa periféricos	-
`task_bh1750`	Mide intensidad lumínica	Cola `lux`
`task_adc`	Dispara ADC del potenciómetro	-
`task_pwm`	Ajusta PWM LED D1	Semáforo y cola
`task_counter_btns`	Maneja S1/S2 (setpoint)	Cola setpoint/counter
`task_display`	Muestra valores en display	Cola `lux`
`task_buzzer`	Activa buzzer por IR	Semáforo `buzz`
`task_ShowValues`	Muestra por consola datos	Lectura de colas
`task_tricolour`	Ajusta LED tricolor según colas	Colas `lux, setpoin`
`task_usr`	Detecta botón USER por polling	Semáforo `usr`
`task_display_change`	Detecta botón USER por polling	Semáforo `usr`
`isr_cn70`	Entrega semáforo por IR	`xSemaphoreGiveFr`
`isr_adc_seqa`	Lee la conversion del ADC	`xQueueOverwriteF`

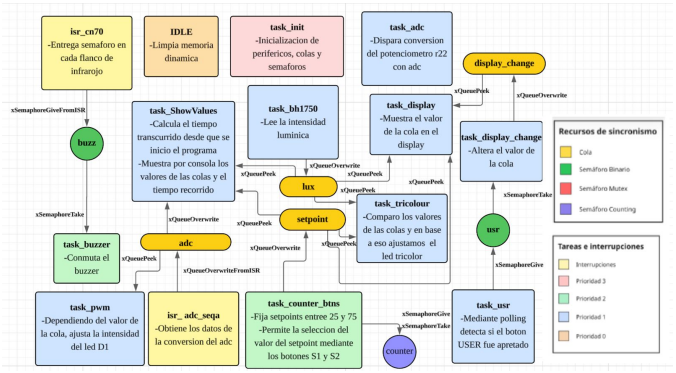

Figura 1. Diagrama de tareas y comunicación del sistema.

III. Diagrama de tareas

El siguiente diagrama muestra la relación entre tareas, ISR, colas y semáforos del sistema desarrollado.

IV. Conclusiones

El sistema propuesto demuestra el uso eficaz de FreeRTOS para manejar múltiples tareas concurrentes en un entorno de tiempo real. El uso de colas y semáforos permitió una comunicación eficiente entre tareas y la correcta interacción con periféricos. El trabajo representa un ejemplo sólido de integración de software embebido con hardware en sistemas basados en microcontroladores.