Sistemas Operativos en Tiempo Real - Trabajo Integrador

Juan Montilla Escuela Técnica N° 7 de Quilmes montillajuanignacio@gmail.com Ezequiel Bourlot Escuela Técnica N° 7 de Quilmes ezebourlot@ejemplo.com

Abstract—Este trabajo presenta el desarrollo de un sistema de control de iluminación usando FreeRTOS sobre una placa con microcontrolador LPC. El sistema incluye lectura de luz ambiente, control mediante potenciómetro y botones, y visualización del estado en un display de 7 segmentos. Se detallan las tareas del RTOS, su comunicación mediante colas y semáforos, y se presenta el diagrama del código.

I. Introducción

El objetivo de este trabajo es implementar un sistema embebido que controle el nivel de iluminación ambiente utilizando FreeRTOS. El sistema emplea un sensor de luz (BH1750), un potenciómetro para ajustar el brillo de un LED, botones para definir un setpoint deseado y un display para visualizar información en tiempo real.

II. PROBLEMA PLANTEADO

Se requiere diseñar un sistema que mantenga la luminosidad deseada (setpoint) comparando la luz ambiente medida con un sensor. Además, el usuario debe poder modificar el setpoint, visualizar datos en un display y tener respuesta visual mediante un LED RGB y sonora mediante un buzzer. Todo el sistema debe operar de manera multitarea usando FreeRTOS.

III. ARQUITECTURA DEL SISTEMA

El sistema se compone de múltiples tareas concurrentes implementadas en FreeRTOS. La comunicación entre ellas se realiza mediante colas ('xQueue') y semáforos ('xSemaphore').



Fig. 1. Diagrama de tareas, colas y semáforos en FreeRTOS.

IV. TAREAS FREERTOS

- task_adc_trigger: Activa conversión ADC periódicamente.
- task_light_sensor: Lee lux de sensor BH1750.
- task_pwm_control: Ajusta brillo del LED principal según potenciómetro.

- task_rgb_controller: Controla LED RGB en función del error respecto al setpoint.
- task_display_manager: Muestra setpoint o lux en el display.
- task_display_toggle: Alterna el modo de visualización.
- task_user_button: Detecta pulsaciones en botón de usuario.
- task_setpoint_buttons: Incrementa/decrementa el setpoint con botones S1/S2.
- task_buzzer_control: Controla el buzzer ante flancos del sensor infrarrojo.
- task_system_monitor: Reporta estado del sistema por consola.

V. COLAS Y SEMÁFOROS

- Colas: queue_adc, queue_lux, queue_setpoint, queue_display, queue_display,
- Semáforos: semphr_usr, semphr_counter, semphr buzz.

VI. CONCLUSIONES

Se logró implementar un sistema multitarea funcional usando FreeRTOS que responde a estímulos del ambiente y del usuario. La arquitectura modular facilita el mantenimiento y escalabilidad del código.