Aplicación de FreeRTOS con control de periféricos en LPC845

Octavio Lautaro Pino
email: octaviolautaropino@impatrq.com

Misael Castillo
email: misaelcastillo@impatrq.com

***Abstract*—**

Este trabajo documenta el desarrollo de una aplicación embebida utilizando FreeRTOS sobre el microcontrolador LPC845. El objetivo es gestionar múltiples tareas concurrentes que interactúan con diversos periféricos, incluyendo un sensor de luz BH1750, LEDs RGB, un display de 7 segmentos, botones de usuario, un potenciómetro, un sensor infrarrojo y un buzzer. Se emplean mecanismos de comunicación como colas y semáforos para coordinar las tareas en un entorno de tiempo real, asegurando una sincronización eficiente y un uso óptimo de los recursos del sistema.

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

I. INTRODUCCIÓN Y PROBLEMA PLANTEADO

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.

El problema planteado consiste en desarrollar una aplicación con FreeRTOS que integre sensado de luz, control de LEDs, display, buzzer y botones. La aplicación debe medir la intensidad lumínica con un sensor BH1750, compararla con un setpoint ajustable por botones, y controlar la iluminación de LEDs tricolor. Adicionalmente, se ajusta la iluminación con un potenciómetro, se activa un buzzer por un sensor infrarrojo y se muestra información relevante por consola. El sistema debe asegurar una respuesta en tiempo real a las interacciones del usuario y una gestión eficiente de los recursos.

II. DIAGRAMA DE CÓDIGO

A continuación, se presenta el diagrama de tareas y comunicación del sistema, mostrando las relaciones entre las tareas de FreeRTOS, las interrupciones (ISR), las colas y los semáforos utilizados para la comunicación. Este diagrama visualiza la estructura concurrente de la aplicación y los mecanismos de sincronización.

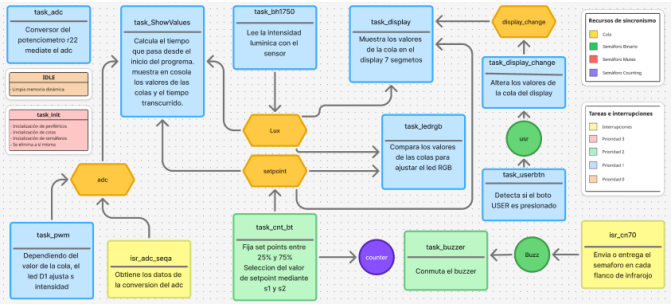

Fig. 1. diagrama de tareas y comunicación del sistema.

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. Este diseño permite una sincronización eficiente y un uso óptimo de los recursos del sistema embebido.

A. prioridades y comunicación

En FreeRTOS, cada tarea tiene una prioridad que determina el orden de ejecución cuando varias tareas están listas para ejecutarse. Una buena elección de prioridades es crucial para asegurar una respuesta rápida a eventos críticos y un funcionamiento fluido del sistema.

Las prioridades se asignan para que el sistema sea responsivo, eficiente y predecible. Tareas críticas y de respuesta rápida (como la lectura de botones o la gestión del display) tienen prioridad más alta. Las tareas de adquisición de datos (como la lectura de sensores) tienen prioridad media. Las tareas de actualización de salidas (como el control PWM de LEDs) pueden tener prioridad baja. Las tareas de inicialización tienen la prioridad más baja, ya que solo se ejecutan al principio.

Las colas (*Queues*) son estructuras de datos que permiten la comunicación segura entre tareas y/o interrupciones. Funcionan como un "puente" para transferir datos atómicamente. En este proyecto, las colas se utilizan para transferir datos de sensores, enviar información de usuario o configuración, y sincronizar y desacoplar tareas.

Los semáforos son mecanismos de sincronización que permiten a las tareas y a las interrupciones coordinar el acceso a recursos compartidos o señalar la ocurrencia de eventos. Son utilizados para sincronizar eventos de usuario, proteger recursos compartidos y coordinar la ejecución entre ISR y tareas.

III. RESULTADOS Y CONCLUSIONES

El sistema propuesto demuestra el uso eficaz de FreeRTOS para manejar múltiples tareas concurrentes en un entorno de tiempo real. La implementación exitosa cumple con los requisitos de la consigna, integrando el sensado de luz, el control de LEDs, el display, el buzzer y los botones, utilizando colas y semáforos para la comunicación entre tareas.

El uso de colas y semáforos permitió una comunicación eficiente entre tareas y la correcta interacción con los periféricos. Esto se evidencia en la capacidad del sistema para medir la intensidad lumínica, ajustar un setpoint, alternar la visualización en el display, controlar los LEDs tricolor proporcionalmente a la diferencia de luz, ajustar la intensidad del LED azul con un potenciómetro, activar un buzzer por detección infrarroja y mostrar datos en consola cada segundo.

Este trabajo representa un ejemplo sólido de integración de software embebido con hardware en sistemas basados en microcontroladores, consolidando los conocimientos adquiridos en FreeRTOS y manejo de periféricos en lenguaje C. Además, reforzó la capacidad de trabajar en equipo, dividiendo las tareas y desarrollando las consignas de manera eficiente. La modularización del código y la adecuada asignación de prioridades han resultado en un sistema responsivo y predecible.