Trabajo Integrador - UTN F.R.A

Sistemas Embebidos

UTN F.R.A

Autores:
Milovan Radakoff, Ramirez Selene
Escuela de Educación Técnica Secundaria N.º 7 (I.M.P.A)
milovanradakoff@impatrq.com
selenenahirramirez@impatrq.com

Resumen

Este informe presenta una explicación detallada sobre la implementación y el alcance de sistemas embebidos, haciendo uso del sistema operativo FreeRTOS y del lenguaje de programación C.

I. Introducción

A partir de los conocimientos adquiridos en el curso de Sistemas Embebidos dictado por el profesor **Fabrizio Carlassara** en la UTN, llevamos a cabo el desarrollo de un proyecto integrador. En él aplicamos los conceptos fundamentales de FreeRTOS y el uso del microcontrolador **LPC845 Breakout**, integrando hardware y software en un sistema operativo de tiempo real.

II. Desarrollo de contenidos

Para cumplir con los requerimientos del proyecto, iniciamos definiendo constantes en el archivo labels.h, lo cual permitió un acceso más claro y organizado a los pines del microcontrolador.

Los archivos wrappers.c y wrappers.h tienen como objetivo encapsular las funciones del SDK, facilitando su uso mediante una capa de abstracción.

En tareas.c y tareas.h se define la arquitectura multitarea del sistema: asignación de prioridades, creación de prototipos y pilas de tareas, además de inicializar semáforos y colas necesarias para la sincronización. Esta modularización mejora la legibilidad del código y permite escalar el sistema fácilmente.

En main.c se configura el sistema, se establece la frecuencia del reloj a 30 MHz y se inicializa la consola de depuración. Luego, se crean distintas tareas mediante xTaskCreate(), y finalmente se lanza el planificador de FreeRTOS con vTaskStartScheduler().

III. Tareas

A. Listado de tareas

1. **Display:**
    - tsk_display_write: escritura en pantalla
    - tsk_display_change: manejo de entradas de botón
    - tsk_control: control del sistema

2. **Inicialización:**
    - tsk_init: tareas iniciales del sistema
3. **Conversión analógica-digital:**
    - tsk_adc: lectura de ADC
4. **Sensor de luminosidad:**
    - tsk_BH1750: lectura del sensor BH1750
5. **Control de setpoint:**
    - tsk_setpoint: define el valor objetivo del sistema
6. **LEDs:**
    - tsk_led_azul: control del LED azul
    - tsk_leds_control: control de varios LEDs
7. **Buzzer:**
    - tsk_buzzer: activación del zumbador
8. **Consola:**
    - tsk_console_monitor: monitoreo y comandos por consola

B. Tabla de hardware utilizado

	Hardware utilizado en LPC 845 BREAKOUT	
Cantidad	**Componente**	**Denominación**
1	Buzzer	-
3	Botones	S1, S2 y USER
2	Potenciómetro	RV21
1	Led adicional	D1
1	Display 7 segmentos	-
1	BH1750	-
1	Sensor infrarrojo	IR
1	Led tricolor	-

C. Conceptos de FreeRTOS

¿Qué son las colas y para qué se usaron en este proyecto?

Las **colas** permiten la transferencia de datos entre tareas y/o interrupciones de manera segura y sincronizada. En nuestro proyecto se usaron para:

- Enviar datos desde sensores
- Transmitir configuraciones o inputs de usuario
- Sincronizar y desacoplar procesos entre tareas

¿Qué son los semáforos y para qué se usaron en este proyecto?

Los **semáforos** son mecanismos de sincronización que controlan el acceso a recursos compartidos y permiten coordinar eventos. En este trabajo los utilizamos para:

- Sincronizar eventos como pulsaciones de botones
- Evitar el acceso simultáneo a recursos
- Permitir que interrupciones despierten tareas específicas

¿Por qué asignar prioridades a las tareas?

Cada tarea en FreeRTOS puede tener una prioridad distinta. Esto permite

gestionar qué tareas deben ejecutarse antes en función de su importancia. En nuestro proyecto:

- Las tareas críticas (como botones o display) tienen prioridad alta
- Las tareas de lectura de sensores tienen prioridad media
- Las tareas de salida (ej. LEDs) tienen prioridad baja
- Las tareas de inicialización solo corren al principio y luego se eliminan

Una correcta gestión de prioridades asegura un sistema eficiente, rápido y predecible.

IV. Conclusiones

La realización de este trabajo nos permitió reforzar y aplicar nuestros conocimientos sobre sistemas embebidos, programación en lenguaje C y el uso de FreeRTOS. Además, fortalecimos nuestras habilidades de trabajo en equipo, organización de tareas y resolución de problemas técnicos.

Agradecimientos

Queremos destacar la guía y acompañamiento brindado por los profesores **Fabrizio Carlassara** y **Sergio Medina** durante todo el proceso.

Referencias

1. Proyecto integrador de ejemplo utilizando FreeRTOS
2. Apuntes y prácticas del curso de Sistemas Embebidos
