UTN Universidad Tecnológica Nacional
Facultad Regional Tucumán

Departamento de Electrónica

Técnicas Digitales II

Informe

Actividad de Formación Practica N°3

Tema: Funciones No bloqueantes, aplicaciones con SysTick en STM32CubeIDE y programación de microcontroladores.

Profesor:
Ing. Rubén Darío Mansilla

ATTP:
Ing. Lucas Abdala

Grupo 8:
Gao Luciano, Mitre Emilio, Jorrat Tomas.

Introducción:

Las funciones de retardo no bloqueantes son herramientas en la programación de microcontroladores que garantizan la ejecución eficiente de múltiples tareas. A diferencia de las funciones bloqueantes como **HAL_Delay**, que suspenden completamente la ejecución del programa durante un tiempo específico, los retardos no bloqueantes permiten realizar otras tareas mientras se supervisa el tiempo que va pasando.

Estas funciones se basan en el uso de temporizadores internos del microcontrolador, como el **SysTick**, que genera interrupciones periódicas que por defecto son de 1ms, pero se puede reprogramar a criterio del usuario, teniendo en cuenta las precauciones necesarias. El SysTick permite contar el tiempo de manera precisa y eficiente sin detener la CPU, actualizando variables o indicadores en cada interrupción.

Ventajas de las funciones de retardo no bloqueantes:

1. **Ejecución multitarea**: Permiten realizar varias operaciones simultáneamente, mejorando el rendimiento del sistema y evitando bloqueos innecesarios.
2. **Uso eficiente del procesador**: Mientras se espera que el tiempo transcurra, la CPU puede atender otras tareas prioritarias o manejar interrupciones críticas.
3. **Escalabilidad**: Facilitan el desarrollo de sistemas complejos, donde múltiples procesos o eventos dependen de temporizaciones independientes.
4. **Consumo energético optimizado**: En sistemas de bajo consumo, pueden combinarse con estados de bajo consumo de la CPU, mejorando la eficiencia energética.
5. **Aplicación en sistemas comerciales**: En productos como electrodomésticos, sistemas de automatización o dispositivos IoT, estas funciones son importantes para manejar tareas simultáneas, como comunicación, monitoreo de sensores y control de periféricos.

Estas ventajas hacen de los retardos no bloqueantes una práctica estándar en el diseño de software embebido, especialmente en aplicaciones comerciales donde la eficiencia y la multitarea son prioritarias.

La función de retardo no bloqueante se realizó tomando como base la función HAL_Delay() y modificándola como se indica en el siguiente archivo:

Permalink: Funciones No Bloqueantes

Función No bloqueante API_Delay.c modificada en base a la HAL_Delay() :

Permalink: Implementacion de la funcion de retardo no bloqueante

Elementos principales del código:

- **Estructura delay_t:** Guarda tres datos principales:
  - Cuando empezó el retardo (startTime).
  - Cuánto tiempo tiene que durar (duration).
  - Si el retardo está en marcha (running).

- **Funciones:**
  - **delayInit:** Configura o reinicia un retardo, indicando cuánto tiempo debe durar.
  - **delayRead:** Verifica si el tiempo ya pasó y devuelve true cuando el retardo ha terminado.
  - **delayWrite:** Permite cambiar cuánto tiempo debe durar el retardo mientras está en uso.

6.2 Aplicaciones desarrolladas:

En esta práctica se implementó el driver API_Delay (".c" y ".h") de función no bloqueante en las 4 aplicaciones que vinimos desarrollando a lo largo del año. Se recomienda leer la descripción e ir comparando con el código para lograr entender mejor.

App 3.1: Encendido (200ms) y apagado (200ms) de manera secuencial en el orden de los leds Verde (LED1), Azul (LED2), Rojo (LED3).

- Permalink : Archivo main.c de la App 3.1
- Observaciones: Los cambios que se hicieron son respecto a la AFP_2_App_2.4. Se modifico la estructura de datos "typedef enum", donde se definió una enumeración (estado_t) encendido "ENCENDIDO_LED#" y apagado de los leds "ESPERAR_LED#_OFF".
  La variable (estado) del tipo estado_t se inicializa en "ENCENDER_LED0" y luego indica en que punto de la secuencia se encuentra el programa.
  Dentro del bucle infinito "while(1)", el programa utiliza un switch case para ejecutar la accion que corresponda al estado que tiene actualmente, es decir, siempre va comenzar en ENCENDI DO_LED0 y a partir de allí irá tomando otros valores la variable estado, pero siempre estando controlado por la función delayRead(). Cuando se alcanza el ultimo estado (ESPERAR_LED2_OFF), la secuencia vuelve al estado inicial y así repetitivamente.

App 3.2: Alterna entre 2 secuencias. Inicia con la secuencia de la app 3.1 y al presionar el pulsador de la placa se invierte.

- Permalink : Archivo main.c App 3.2
- Observaciones: En esta app se crearon 2 vectores que recorren los elementos cuyos valores hacen referencia a los leds, una secuencia normal que es de la app 3.1 y una secuencia invertida. Dentro del bucle infinito hay un chequeo del botón para evitar qué, si el usuario lo mantiene presionado, el micro lo interpretará como si tuviese que alternar de secuencia una y otra vez hasta que lo deje de presionar. Luego hay un switch-case, el cual asigna a la función "control_leds" inicialmente la secuencia normal debido a la variable estado_secuencia que comienza en "1". La funcióncontrol_leds recorre el vector "secuencia_normal" o "secuencia_invertida" dependiendo del parámetro puntero que reciba.

App 3.3: Alterna entre 4 secuencias. Inicia con la secuencia de la app 3.1 y al presionar el pulsador, pasa a la secuencia 2 que hace parpadear los 3 leds simultáneamente, luego a la 3 que hace parpadear los 3 leds con distinta frecuencia cada uno y por último la secuencia 4 que hace parpadear simultáneamente LED1 y LED3, mientras que LED2 lo hará de manera inversa, con una alternancia de 150 ms.

- Permalink : Archivo main.c App 3.3
- Observaciones: En esta aplicación se debieron implementar diferentes secuencias para el encendido y apagado de los leds, se debió tener particular consideración utilizando la función de delay no bloqueante para poder alternar entre las secuencias y que cada una se ejecute de manera correcta, ya que nos encontramos con problemas como que se alteraba el orden de encendido en ciertas secuencias o que directamente algunos leds permanecían encendidos. Por lo que se agregaron líneas de código para establecer el correcto inicio de cada secuencia al presionar el botón y producirse el cambio.

App 3.4: Parpadea simultáneamente los 3 leds y al presionar el pulsador cambia consecutivamente entre secuencias. Cada secuencia tiene una frecuencia de parpadeo de mayor a menor. 100ms, 250ms, 500ms y 1000ms

- Permalink : Archivo main.c App 3.4
- Observaciones: Se implemento la función delay sin tener que alterar mucho el código respecto a las practicas anteriores ya que al encender y apagar todos los leds en simultaneo no se generaron los problemas que mencionamos para la app 3.3.

Enlace al repositorio Grupo 8: AFP_3_TDII_2024_GRUPO_8