UTN Universidad Tecnológica Nacional

Facultad Regional Tucumán

Departamento Electrónica

Técnicas Digitales II

Actividad de Formación Práctica 4

Tema: Funciones no bloqueantes, aplicación con SysTick en STM32CubeIDE, programación de microcontroladores.

Profesor:
Ing. Rubén Darío Mansilla

ATTP:
Ing. Lucas Abdala

vencimiento:
05 de setiembre de 2025

Año: 2025

1. Objetivos

- Que el alumno logre implementar un módulo de software para trabajar con retardos no bloqueantes.
- Que el alumno aplique el concepto de funciones no bloqueantes en aplicaciones prácticas concretas utilizando el *IDE* STM32CubeIDE.
- Que el alumno desarrolle un driver de funciones no bloqueantes.
- Que el alumno aplique este driver en la implementación de aplicaciones con el IDE que estamos utilizando.

2. Generalidades

Esta actividad de formación práctica es del tipo **práctica de laboratorio**.

Cómo desarrollar esta actividad:

- Esta actividad debe desarrollarse siguiendo las consignas del punto 3.
- Cada integrante del grupo debe tomar al menos una de las Apps desarrolladas en la actividad de formación práctica 2, utilizar el driver desarrollado para funciones no bloqueantes y realizar las modificaciones correspondientes en el **main.c** de la App para que utilice las funciones de este driver.
- Los integrantes del grupo deben ponerse de acuerdo para usar el mismo nombre para el driver de funciones no bloqueantes y para las funciones que lo integran.
- Una vez modificadas todas las aplicaciones, se las ubicará en el repositorio grupal para su presentación. Todas las Apps deben presentar una coherencia en cuanto al formato del nombre del mismo y de las funciones desarrolladas para este.
- Se debe utilizar el repositorio grupal creado en GitHub para la presentación de las actividades de formación práctica:
  - Se debe crear una carpeta para cada actividad de formación práctica.
  - Se deben ubicar en la carpeta correspondiente, en este repositorio, las aplicaciones desarrolladas para esta actividad de formación práctica respetando el siguiente formato de nombre: **App_ 1_ Y_ Grupo_ X_ 2025**.

  Nota sobre el formato de nombre de la carpeta: X es el número de su grupo y **Y** es el número de aplicación como figura en el enunciado.
- Se debe insertar el link al repositorio de GitHub al final del informe que se presenta con esta actividad de formación práctica.
- El archivo que contiene el desarrollo del informe de la actividad de formación práctica debe ser de tipo pdf y su nombre debe respetar el siguiente formato:

  AFP_ 4_ Grupo_ X_ TDII_ 2025.pdf

- Se debe realizar la presentación de las aplicaciones, en el laboratorio de Sistemas Digitales, funcionando según lo que pide la consigna, con su correspondiente defensa de lo desarrollado. Fecha a acordar. Cada alumno debe presentar y defender la aplicación que ha desarrollado para esta actividad.

Nota 1: Para el desarrollo de las aplicaciones de esta actividad de formación práctica se utilizará el entorno de desarrollo integrado STM32CubeIDE para aplicar sobre una placa de desarrollo STM32-NUCLEO-F4XX-YY.

Nota 2: Las placas de desarrollo sugeridas para las prácticas son: STM32-NUCLEO-F401-RE, STM32-NUCLEO-F412-ZG, STM32-NUCLEO-F413-ZH o STM32-NUCLEO-F429-ZI.

Nota 3: El desarrollo de las actividades de formación práctica se realizará y presentará de manera grupal.

3. Consignas para desarrollar

1. Tome como base un proyecto nuevo vacío para implementar las funciones auxiliares necesarias para usar retardos no bloqueantes en un archivo fuente **main.c** con su correspondiente archivo de cabecera **main.h**.

   1.1. En **main.h** se deben ubicar los prototipos de las siguientes funciones y las declaraciones:

   - Definición de tipos de variables personalizadas del driver:
     - **typedef** *uint32_t tick_t*; Ver qué biblioteca se debe incluir para que esto compile.
     - **typedef** *bool bool_t*; Ver qué biblioteca se debe incluir para que esto compile.
     - **typedef** *struct {*

```
        tick_t startTime;
        tick_t duration;
        bool_t running;
    } delay_t;
```

   - Declaración de funciones para el driver de funciones no bloqueantes:
     - *void* **delayInit**(delay_t * delay, tick_t duration);
     - *bool_t* **delayRead**(delay_t * delay);
     - *void* **delayWrite**(delay_t * delay, tick_t duration);

   1.2. En **main.c** se debe ubicar la implementación de todas las funciones. Consideraciones para la implementación:

   - **delayInit** debe:
     - cargar el valor de duración del retardo en la estructura, en el campo correspondiente.
     - No debe iniciar el conteo del retardo.
     - Debe inicializar el flag running en '*false*'.
   - **delayRead** debe verificar el estado del flag running:
     - si es *false*, tomar marca de tiempo y cambiar *running* a '*true*'.

- si es *true*, hacer la cuenta para saber si el tiempo del retardo se cumplió o no: '**marca de tiempo actual** - **marca de tiempo inicial** es mayor o igual a **duración del retardo**'? y devolver un valor booleano que indique si el tiempo se cumplió o no.
- Cuando el tiempo se cumple se debe cambiar el flag *running* a *false*.

▪ **delayWrite** debe permitir cambiar el tiempo de duración de un *delay* existente.

Nota 4: para obtener una marca de tiempo se puede usar la función **HAL GetTick()** que devuelve un valor que se incrementa cada 1 ms y que se puede usar como base de tiempo.

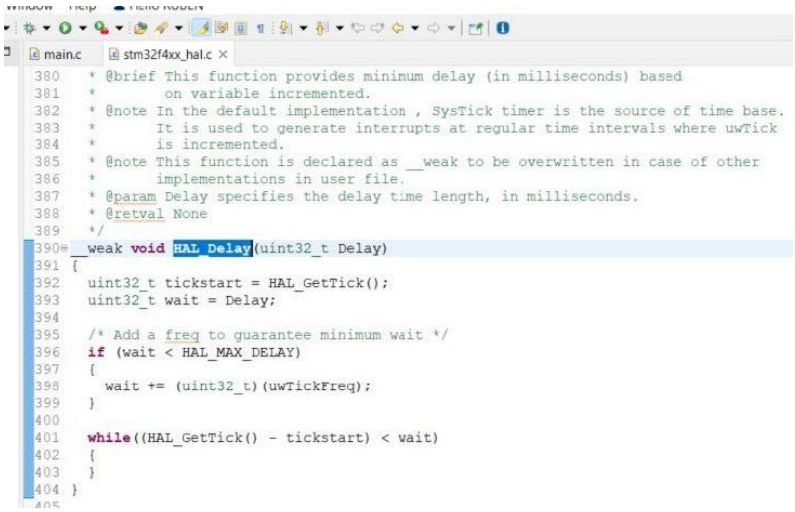

Figura 1. Funciones del SysTick.

2. Una vez creadas estas funciones, haga las pruebas básicas necesarias en **main.c** para verificar que estas se comportan como se pide en el punto anterior. Compile el proyecto y verifique que funciona como corresponde.

3. Cree los archivos **fuente.c** y **header.h** para armar el driver de funciones no bloqueantes. Mueva las definiciones de tipos de variables y declaraciones de funciones al archivo **header.h** y las implementaciones al archivo **fuente.c** y haga los ajustes necesarios para que compile sin errores. Ubique estos archivos del nuevo driver en la carpeta API, donde corresponde en la estructura de archivos del proyecto.

4. Compile, y realice el *degub* y la programación de la placa de desarrollo para verificar que la aplicación corre y ejecuta las operaciones correctamente como lo hac´ıa originalmente antes de la modificación. En este punto deber´ıa notar ciertas mejoras en la performance de las aplicaciones.

5. Cada alumno, integrante del grupo, tomará una App de la practica anterior y realizará las modificaciones necesarias para integrar el nuevo driver de funciones no bloqueantes y reemplazar el uso de la función bloqueante **HAL_delay()** por las nuevas funciones no bloqueantes desarrolladas en esta practica. Compilará y correrá la App en la placa de desarrollo.

6. Desarrolle un breve informe que contenga los siguientes ´ıtems:

   6.1. **Introducción al tema:** Todo lo que vio acerca de las funciones de retardo no bloqueantes, en que se basan, qué ventajas tiene el uso de este tipo de funciones en una aplicación comercial.

6.2. **Aplicaciones desarrolladas:**

- **Aplicación:** Nombre de la aplicación modificada.
- **Autor de la modificación:** Apellido y nombres del alumno que realizó las modificaciones en cada aplicación.
- **Observaciones:** sobre lo realizado en esa aplicación. En este campo puede compartir su experiencia en el desarrollo, los problemas que pudo haber enfrentado y como los solucionó. Recomendaciones si las hubiera.

6.3. **Link al repositorio grupal:** en el que se encuentran las aplicaciones desarrolladas para esta actividad de formación práctica.

4. Bibliografía y documentación

Documentación accesible desde el IDE en:

Help → Information Center → STM32CubeIDE Manuals

Documentación desarrollada por la cátedra: Funciones no bloqueantes.pdf

ESTUDIANTE: Roqué Luciano J..... Luciano J. Roqué

ESTUDIANTE: Cruz Rodolfo Martín.....

FECHA DE INICIO: __/__/__ FECHA DE PRESENTACIÓN: 5 /9 /25

CONFORMIDAD DEL DOCENTE:.....

Informe: Uso de retardos no bloqueantes

En el desarrollo de sistemas embebidos es frecuente la necesidad de implementar retardos de tiempo para coordinar acciones, como encender o apagar un LED, leer sensores o controlar periféricos. Tradicionalmente, esto se resuelve con funciones bloqueantes como HAL_Delay(), que detienen por completo la ejecución del programa durante el lapso especificado.
Sin embargo, este enfoque no resulta eficiente en aplicaciones comerciales, donde el microcontrolador debe realizar múltiples tareas en paralelo y no puede "quedarse esperando". Para resolver esta limitación se utilizan retardos no bloqueantes, los cuales se basan en la medición del tiempo transcurrido a partir de una marca inicial, usualmente obtenida con la función HAL_GetTick() que incrementa cada milisegundo.
La ventaja principal de este método es que el procesador sigue ejecutando el resto de las instrucciones mientras verifica periódicamente si el tiempo requerido ya pasó. De este modo, se logra una mayor concurrencia, mejor aprovechamiento de recursos y una arquitectura de software más escalable y confiable, especialmente en sistemas que requieren respuesta en tiempo real.

Aplicación desarrollada: App_1_1_Grupo_4_2025_Driver_Delay

Autor de la modificación: Cruz Rodolfo Martin.

Observaciones:

En esta primera aplicación el objetivo fue implementar una secuencia básica de encendido y apagado de los LEDs de la placa, respetando un tiempo fijo de alternancia. La adaptación al uso de retardos no bloqueantes representó un cambio importante en la forma de pensar la solución, ya que no se trataba simplemente de “esperar” un tiempo, sino de estructurar el programa de manera que pudiera seguir ejecutándose en paralelo. La experiencia permitió comprender las diferencias fundamentales entre los retardos bloqueantes y los no bloqueantes, y cómo estos últimos resultan más convenientes en proyectos donde se requiere escalabilidad o capacidad de respuesta ante distintos eventos.
Recomendaciones:
Al desarrollar aplicaciones con retardos no bloqueantes, conviene planificar la lógica en términos de fases o estados. Esto hace que el programa sea más claro y evita errores comunes al migrar desde un retardo tradicional.

Aplicación desarrollada: App_1_2_Grupo_4_2025_Driver_Delay

Autor de la modificación: Roqué Luciano J.

Observaciones:
La segunda aplicación planteó un escenario más dinámico, ya que además de la secuencia de LEDs, se debía tener en cuenta la interacción del usuario mediante el pulsador. Esto implicó trabajar con condiciones externas y garantizar que el sistema respondiera adecuadamente cada vez que se producía una acción sobre el botón. El mayor aprendizaje fue entender que al combinar retardos no bloqueantes con entradas externas se obtiene un programa más flexible, capaz de adaptarse a cambios de estado

sin detenerse ni perder funcionalidad. Este tipo de diseño es esencial en aplicaciones reales, donde se deben coordinar múltiples tareas en simultáneo.

Recomendaciones:

Es recomendable mantener el código modular y bien organizado, separando la lógica de control de LEDs de la lectura del pulsador. Esto no solo facilita la comprensión, sino que también ayuda a detectar y corregir problemas en etapas tempranas.

Aplicación desarrollada: App_1_3_Grupo_4_2025_Driver_Delay
Autor de la modificación: Cancino Alan Maximiliano.

Observaciones:

En esta implementación se desarrolló una librería de retardos no bloqueantes basada en el uso de la función HAL_GetTick(). El diseño de la estructura delay_t resultó muy adecuado, ya que permitió organizar de manera clara los parámetros principales del retardo (duración, estado y tiempo de inicio). De esta forma, se logró reemplazar el uso de retardos bloqueantes como HAL_Delay(), favoreciendo una lógica de ejecución continua en la cual el programa puede seguir respondiendo a otros eventos mientras se cumple el intervalo configurado. Este enfoque representa un avance importante respecto a la programación tradicional, ya que sienta las bases para desarrollar aplicaciones más escalables y reactivas.

Recomendaciones:

- Mantener y reforzar el uso de retardos no bloqueantes como enfoque principal, ya que ofrecen mayor flexibilidad y capacidad de respuesta frente a eventos concurrentes.
- Incorporar validaciones en la inicialización y escritura de la duración del retardo para evitar valores inválidos que puedan generar resultados no deseados.

En conclusión, esta práctica permitió no solo implementar un retardo no bloqueante, sino también comprender sus ventajas frente a los enfoques bloqueantes tradicionales. Al mismo tiempo, abrió la posibilidad de plantear mejoras que harían la librería más robusta, portable y adaptable a proyectos de mayor complejidad.

Aplicación desarrollada: App_1_4_Grupo_4_2025_Driver_Delay
Autor de la modificación: Cusi Alejandro Daniel.

Observaciones:

En esta app seguimos avanzando con los drivers para 4 secuencias de parpadeo, donde los 3 leds onboard parpadearan simultáneamente con ciclos de 100, 250, 500 y 1000 ms respectivamente, el cambio de secuencia vendrá dado por la pulsación del botón de usuario también onboard, luego de la última secuencia deberá volver al ciclo de la primera secuencia.

En esta ocasión, se desarrollo los drivers para quitar los retardos bloqueantes que nos ocasionaba el HAL_Delay, lo cual hacia nuestro código muy propenso a perder información del exterior, esto se logro mediante el conteo de ticks del sistema (mediante la captura y comparacion del valor de Systick).

Siguiendo con la línea previa, seguimos la idea de un código legible y facilmente modificable en cuanto a las variables que contiene (como ser numero de leds y/o tiempos de retardo)

Comentarios y recomendaciones:

En mi caso no seguí el consejo del profesor Mansilla e intenté hacer el código directamente sobre la app y al no funcionar no lograba encontrar la falla. Recomiendo trabajar de forma modular, es decir, desarrollar y verificar el correcto funcionamiento del driver antes de intentar incorporarlo en la app correspondiente

Grupo 4 : Cancino Alan Maximiliano
Cruz Martín Rodolfo
Cusi Alejandro Daniel
Roqué Luciano Javier

Actividad de Formación Practica 4
Repositorio Grupal: Grupo-4-TDII-2025