



Actividad de Formación Práctica 3

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

Integrantes:

Castro Oscar Martín

Décima Enrique Emanuel

Ortiz Nicolás Agustín

Introducción

Las funciones de retardo no bloqueantes permiten pausar una tarea sin detener todo el flujo de ejecución de un programa. A diferencia de las funciones bloqueantes, que detienen el hilo principal hasta que se completa el retardo, las no bloqueantes permiten continuar ejecutando otras tareas en paralelo. Trabajar de esta forma tiene varias ventajas:

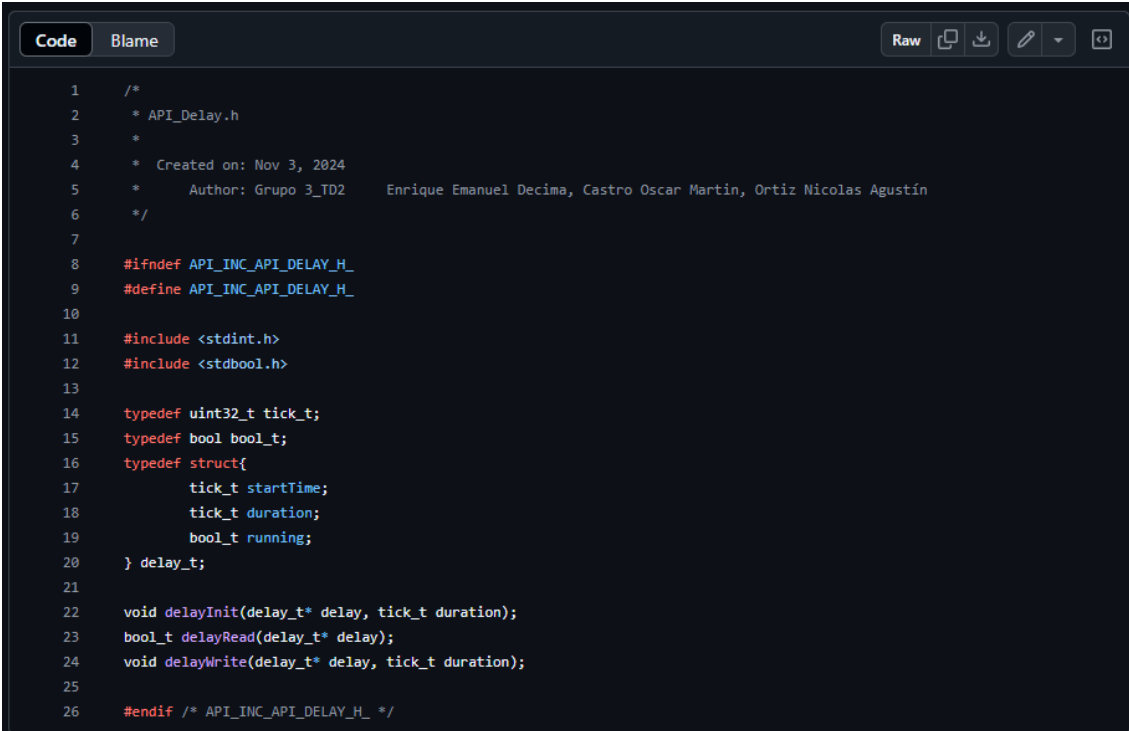
- 1- Mejor rendimiento: ya que permite la multitarea simultáneamente, podemos atender varias solicitudes sin que el sistema se detenga.
- 2- Escalabilidad: al no bloquear el flujo de ejecución, es posible agregar tareas sin necesidad de aumentar los recursos.
- 3- Experiencia fluida en la ejecución de las tareas: evitamos que la lectura de datos, o de pines se vea bloqueada en algún instante.

Creación de la librería API_Delay

Header

En el encabezado de la librería seguimos los siguientes pasos:

- 1- Incluimos las librerías `<stdint.h>` y `<stdbool.h>`.
- 2- Creamos el typedef `"tick_t"` (enteros de 32 bits que guardan el tiempo transcurrido en milisegundos)
- 3- Creamos el typedef `"bool_t"` (variables booleanas, que guardan true o false).
- 4- Creamos la estructura `delay_t`
- 5- Declaramos las funciones: `delayInit`, `delayRead`, `delayWrite`.



```
1  /*
2   * API_Delay.h
3   *
4   * Created on: Nov 3, 2024
5   * Author: Grupo 3_TD2 Enrique Emanuel Decima, Castro Oscar Martin, Ortiz Nicolas Agustín
6   */
7
8  #ifndef API_INC_API_DELAY_H_
9  #define API_INC_API_DELAY_H_
10
11  #include <stdint.h>
12  #include <stdbool.h>
13
14  typedef uint32_t tick_t;
15  typedef bool bool_t;
16  typedef struct{
17      tick_t startTime;
18      tick_t duration;
19      bool_t running;
20  } delay_t;
21
22  void delayInit(delay_t* delay, tick_t duration);
23  bool_t delayRead(delay_t* delay);
24  void delayWrite(delay_t* delay, tick_t duration);
25
26  #endif /* API_INC_API_DELAY_H_ */
```

Source

En el archivo de extensión “.c”, implementamos las funciones que declaramos en la cabecera.

delayInit

En esta función inicializamos el delay, recibimos de parámetro un tiempo de duración el cual le asignamos mediante el atributo *duration* de la estructura *delay_t*. Luego iniciamos *running* en *false*, esto se hace por que el inicio del conteo se hace al llamar a otra función llamada “*delayRead*”.

delayRead

Esta función revisa si ya paso el tiempo correspondiente al delay previamente definido, esto lo hace a través del SysTick, con la función “*HAL_GetTick()*” que devuelve el tiempo transcurrido en milisegundos desde que inicio la aplicación en el microcontrolador.

Si *running* es *false*, lo que hace es iniciar el conteo, almacenando en la variable *starTime* el tiempo en el que comienza el conteo, luego establece *running* en *true*, ya que el contador está corriendo. Si *running* es *true*, esto significa que el delay está en ejecución por lo tanto a cada llamado de la función corrobora si el tiempo de duración se ha cumplido, de ser así retorna *true* a la función informando que el delay ha finalizado y establece *running* en *false* ya que el delay dejo de contar. Si el tiempo de duración todavía no se cumplió retorna *false* a la función.

delayWrite

Lo único que hace esta función es permitir sobrescribir el tiempo de duración del delay.

```
4  * Created on: Nov 3, 2024
5  * Author: Grupo 3_TD2 Enrique Emanuel Decima, Castro Oscar Martin, Ortiz Nicolas Agustin
6  */
7
8  #include "main.h"
9  #include "API_Delay.h"
10
11  /*
12   * @brief Inicializacion del delay, establece la duracion
13   * @param delay_t delay, tick_t duration
14   * @retval None
15   */
16  void delayInit(delay_t * delay, tick_t duration){
17      delay->duration = duration;
18      delay->running = false;
19  }
20
21  /*
22   * @brief Revisa si ya paso el tiempo correspondiente al delay
23   * @param delay_t delay
24   * @retval false -> No transcurrio el tiempo del delay
25   *          true -> Ya transcurrio el tiempo del delay
26   */
27  bool_t delayRead(delay_t* delay){
28      if(delay->running){
29          if ((HAL_GetTick() - delay->startTime) >= delay->duration){
30              delay->running = false;
31              return true;
32          }else{
33              return false;
34          }
35      }else{
36          delay->startTime = HAL_GetTick();
37          delay->running = true;
38          return false;
39      }
40  }
41
42  /*
43   * @brief Reescribe el tiempo de duracion
44   * @param delay_t delay, tick_t duration
45   * @retval None
46   */
47  void delayWrite(delay_t* delay, tick_t duration){
48      delay->duration = duration;
49  }
```

Enlaces

Repositorio grupal:

[EmanuelDecima/Grupo 3 TDII 2024](#)

AFP 3:

[Grupo 3 TDII 2024/AFP 3 TDII 2024 at main · EmanuelDecima/Grupo 3 TDII 2024](#)

App 3.1:

[Grupo 3 TDII 2024/AFP 3 TDII 2024/AFP 3 Grupo 3 App 3.1 at main · EmanuelDecima/Grupo 3 TDII 2024](#)

App 3.2

[Grupo 3 TDII 2024/AFP 3 TDII 2024/AFP 3 Grupo 3 App 3.2 at main · EmanuelDecima/Grupo 3 TDII 2024](#)

App 3.3

[Grupo 3 TDII 2024/AFP 3 TDII 2024/AFP 3 Grupo 3 App 3.3 at main · EmanuelDecima/Grupo 3 TDII 2024](#)

App 3.4

[Grupo 3 TDII 2024/AFP 3 TDII 2024/AFP 3 Grupo 3 App 3.4 at main · EmanuelDecima/Grupo 3 TDII 2024](#)