Departamento Electrónica

Técnicas Digitales II

Actividad de Formación Práctica 4

Tema: Anti-rebote desarrollado por MEF para garantizar detección positiva de pulsadores mecánicos, aplicación en STM32CubeIDE, programación de microcontroladores.

Profesor:
Ing. Rubén Darío Mansilla

ATTP:
Ing. Lucas Abdala

vencimiento:
15 de noviembre de 2024

Año: 2024

Índice

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	23 de octubre de 2024
1	Se agrega bibliografía	25 de octubre de 2024
2	Se agrega bibliografía sobre la HAL de STM32	1 de noviembre de 2024

1. Objetivos

- Que el alumno logre aplicar el concepto de MEF y de utilización de *drivers* para implementar funciones anti-rebotes por software utilizando el *IDE* STM32CubeIDE.
- Que el alumno desarrolle un *driver* de funciones *debounce* (anti-rebotes).
- 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 3, 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 *debounce* 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_2024**.

  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_2024.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 practicas son: STM32-NUCLEO-F401-RE, STM32-NUCLEO-F412-ZG, STM32-NUCLEO-F413-ZH ó 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, para lo cual parta de una copia del ultimo proyecto desarrollado para la practica 3, que ya tiene la estructura del *driver* **API_Delay**, para implementar las funciones auxiliares necesarias para detectar el estado de pulsadores con sistema anti-rebotes (*debounce*) por software en el 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** *enum* {

```
BUTTON_UP,
BUTTON_FALLING,
BUTTON_DOWN,
BUTTON_RISING
```

       } debounceState_t;
   - Declaración de funciones para el driver de funciones *debounce*:
     - *void* **debounceFSM_init()**;
     - *void* **debounceFSM_update()**;
     - *void* **buttonPressed()**; Debe
     - *void* **buttonReleased()**; Debe invertir el estado del *LED3*.
       Las declaraciones de estas funciones también podrían ubicarse en el sector correspondiente en el **main.c**

   1.2. El tiempo de anti-rebote debe ser de 40 ms con un retardo no bloqueante como los implementados en la práctica 3.

   La función **debounceFSM_update()** debe llamarse periódicamente.

   En **main.c** se debe ubicar la implementación de todas las funciones. Consideraciones para la implementación:
   - **debounceFSM_init** debe:
     - cargar el estado inicial:
       - carga el estado inicial de la MEF.
       - carga el delay que usara la MEF: 40 ms.
       - Estado de los LEDS apagados.
       - Carga el estado de la variable de lectura del pulsador en no activo.
   - **debounceFSM_update** debe:
     - leer las entradas.

- resolver la lógica de transición de estados.
- actualizar las salidas.

- **buttonPressed** debe invertir el estado del *LED1*.
- **buttonReleased** debe invertir el estado del *LED3*.

Nota 4: El comportamiento del la MEF que resuelve el sistema anti-rebotes (*debounce*) por software se puede ver en el diagrama de estados de la figura 1. Para mas detalles sobre la implementación de la MEF ver el documento:

Implementación de Antirrebote con Máquina de Estado.pdf

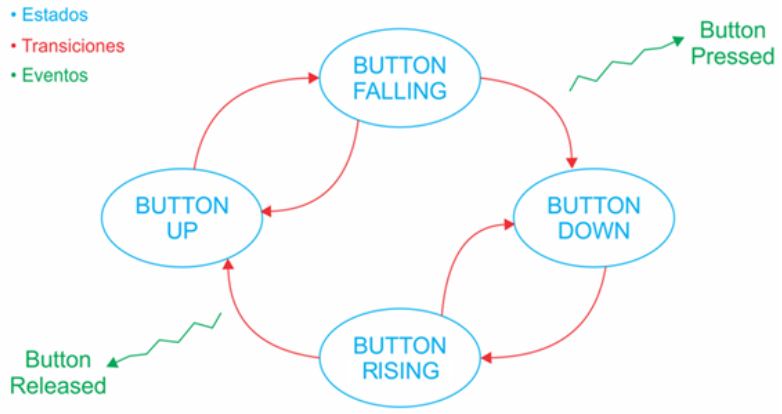

Figura 1. Diagrama de estados de la MEF para debounce.

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 un nuevo proyecto como copia del proyecto con la implementación del punto 1 de esta práctica (Práctica 4).

4. Implemente un módulo de software en un archivo fuente **API_debounce.c** con su correspondiente archivo de cabecera **API_debounce.h** y ubiquelos en el proyecto dentro de las carpetas ***/drivers/API/src*** y ***/drivers/API/inc***, respectivamente.

5. En **API_debounce.h** se deben ubicar los prototipos de las funciones públicas y las declaraciones:

   - void **debounceFSM_init()**;
   - void **debounceFSM_update()**;

6. Implemente una función *bool_t* **readKey()** que debe realizar lo siguiente:

   - leer una variable interna del módulo y devolver *true* o *false* si la tecla fue presionada.
   - si devuelve true, debe resetear (poner en *false*) el estado de la variable.

7. En **API_debounce.c** se deben ubicar las declaraciones privadas, los prototipos de las funciones privadas y la implementación de todas las funciones del módulo, privadas y públicas:

   - La declaración de *debounceState_t* debe ser privada en el archivo **.c** y la variable de estado de tipo *debounceState_t* debe ser global privada (con *static*).

- Declare una variable tipo *bool_t* global privada que se ponga en *true* cuando ocurre un flanco descendente y se ponga en *false* cuando se llame a la función **readKey()**;

8. Modifique las Apps desarrolladas en la práctica anterior (Práctica 3) para que utilicen las funciones anti-rebote del módulo **API_debounce** y los retardos no bloqueantes del módulo **API_delay**, cada vez que se presione la tecla.

9. 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 hacía originalmente antes de la modificación. En este punto debería notar ciertas mejoras en la performance de las aplicaciones.

10. Cada alumno, integrante del grupo, tomará una App de la practica anterior y realizará las modificaciones necesarias para integrar el nuevo driver de funciones *debounce* y el de funciones no bloqueantes. Compilará y correrá la App en la placa de desarrollo.

11. Desarrolle un breve informe que contenga los siguientes ítems:

    11.1. **Introducción al tema:** Todo lo que vio acerca de la implementación de funciones *debounce* mediante el uso de *MEF* y qué ventajas tiene el uso de este tipo de funciones en una aplicación comercial.

    11.2. **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.

    11.3. **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 de STM sobre las placas de desarrollo Núcleo F4XX:
  - MB1137-EsquematicoPlacaSTM-F4XX.pdf
  - UM1974 - STM32 Nucleo-144 board User Manual.pdf
- Documentación de STM sobre las funciones y organización de la HAL:
  - UM1725 - Descripcion de la HAL STM32
- Bibliografia sobre programacion en STM32
  - Donald Norris - Programming With STM32_ Getting Started With the Nucleo Board and C_C++-McGraw-Hill Education (2018)
  - Mastering STM
- Documentación desarrollada por la cátedra:
  - Implementación de Antirrebote con Máquina de Estado.pdf