UNIVERSIDAD SERGIO ARBOLEDA

SISTEMAS EMBEBIDOS

INGENIERÍA ELECTRÓNICA

WENDY JOHANNA CHAPARRO – CRISTOPHER RAMÍREZ RAMÍREZ

INFORME DE LABORATORIO #2

Para este laboratorio se hizo uso de la placa de desarrollo STM32F411 con el programa STM32CUBE IDE con el objetivo de registrar el número de veces (entre 1 y 255) que el usuario oprime un pulsador; una vez que se deje de oprimir, deben pasar 5 segundos para mostrar el número de veces que fue oprimido el pulsador en formato binario en 8 LEDs. Este número se debe visualizar durante 10 segundos y el sistema debe apagar los LEDs y quedar listo para una nueva pulsación.

LIBRERÍAS IMPLEMENTADAS:

- **HAL_GPIO:**

La librería HAL_GPIO se encarga de la configuración y control de los pines GPIO (General Purpose Input/Output) del microcontrolador. Los GPIO son pines que pueden ser configurados como entradas o salidas para interactuar con otros dispositivos o periféricos externos, como sensores, LEDs, botones, etc.

Funcionalidades principales:

- **Configuración de pines:** Puedes configurar un pin como entrada, salida, analógico, o con funciones alternativas.
- **Lectura de pines:** Permite leer el estado de un pin configurado como entrada (alto o bajo).
- **Escritura en pines:** Permite establecer el estado de un pin configurado como salida (alto o bajo).
- **Interrupciones:** Puedes configurar interrupciones en los pines GPIO para responder a eventos específicos, como un cambio de estado en un pin.

- **HAL_Delay:**

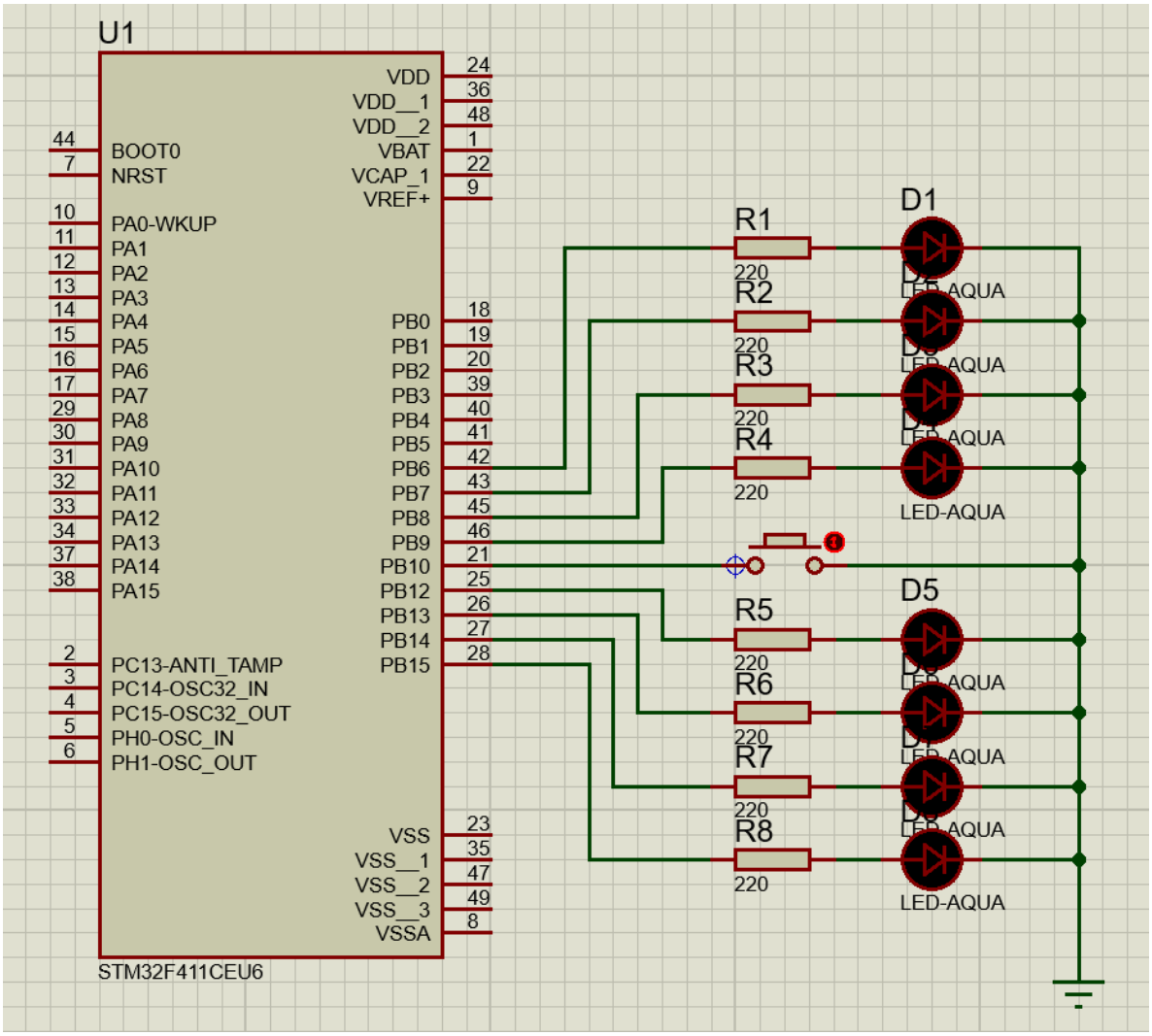
La librería HAL_Delay proporciona funciones para generar retardos en el tiempo de ejecución de tu programa. Es comúnmente utilizada para esperar una cantidad específica de tiempo antes de continuar con la siguiente instrucción, por ejemplo, para controlar el parpadeo de un LED o para sincronizar procesos en tiempo real.

Funcionalidades principales:

- **Retardo en milisegundos:** Proporciona una función para generar un retardo de una cantidad específica de milisegundos.
- **Retardo en microsegundos:** En algunos casos, también puedes tener funciones para retardos más cortos, en microsegundos.

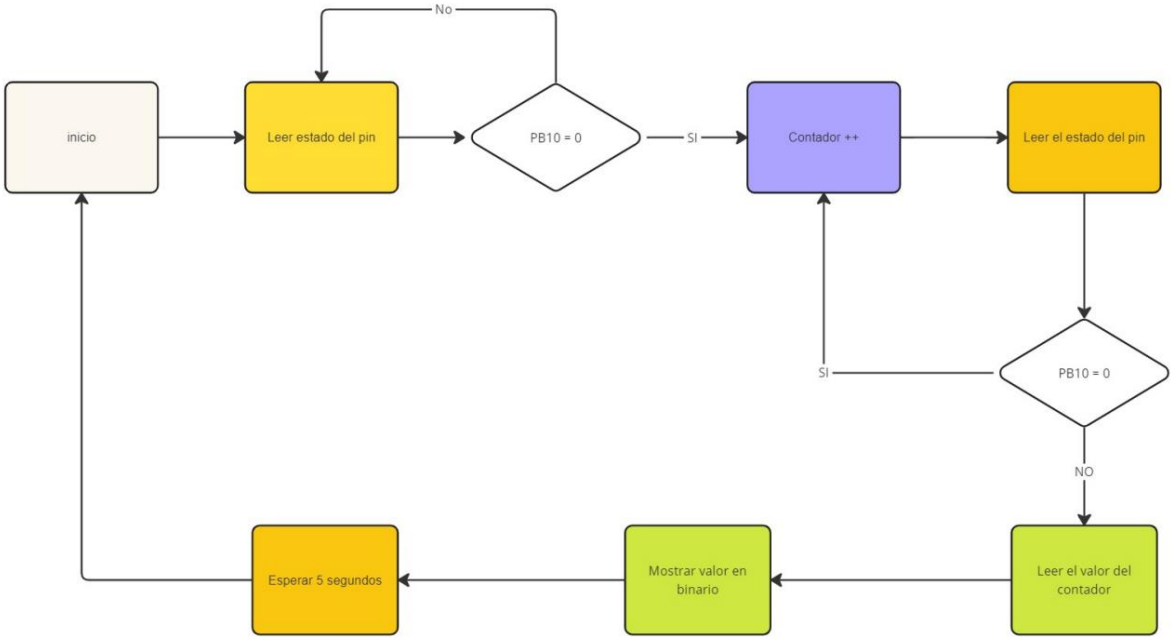
ANÁLISIS Y DESARROLLO:

Para la parte física y como se menciono anteriormente se utilizó la placa de desarrollo STM32F411, 8 leds (cada uno con una resistencia de 220 Ω) los cuales serían los encargados de mostrar la cantidad de pulsaciones en formato binario, además se uso un pulsador en configuración pull-up que se encarga de enviar una señal (la configuración pull-up se hizo de forma interna en el microcontrolador).



En un inicio se interpretó el problema teniendo en cuenta el estado de los pines del microcontrolador. Para el pull-up se utilizó el pin PB10 y para la salida de los pines de los leds se eligieron los pines PB6 – 7 - 8 – 9 – 12 - 13 – 14- 15. Se pensó tener presente el estado del pin de entrada (entre 1 y 0), cuando este estuviera en 0, se entraba en un bucle en el cual se detectaba cuando había un cambio de estado y así ir acumulando este valor en una variable (Contador). Si bien este análisis permitió obtener un buen funcionamiento en la depuración del código, cuando se ejecutaba el código en tiempo real, era muy complicado pulsar el botón en un tiempo específico para que fuera detectado por el algoritmo que se elaboró.

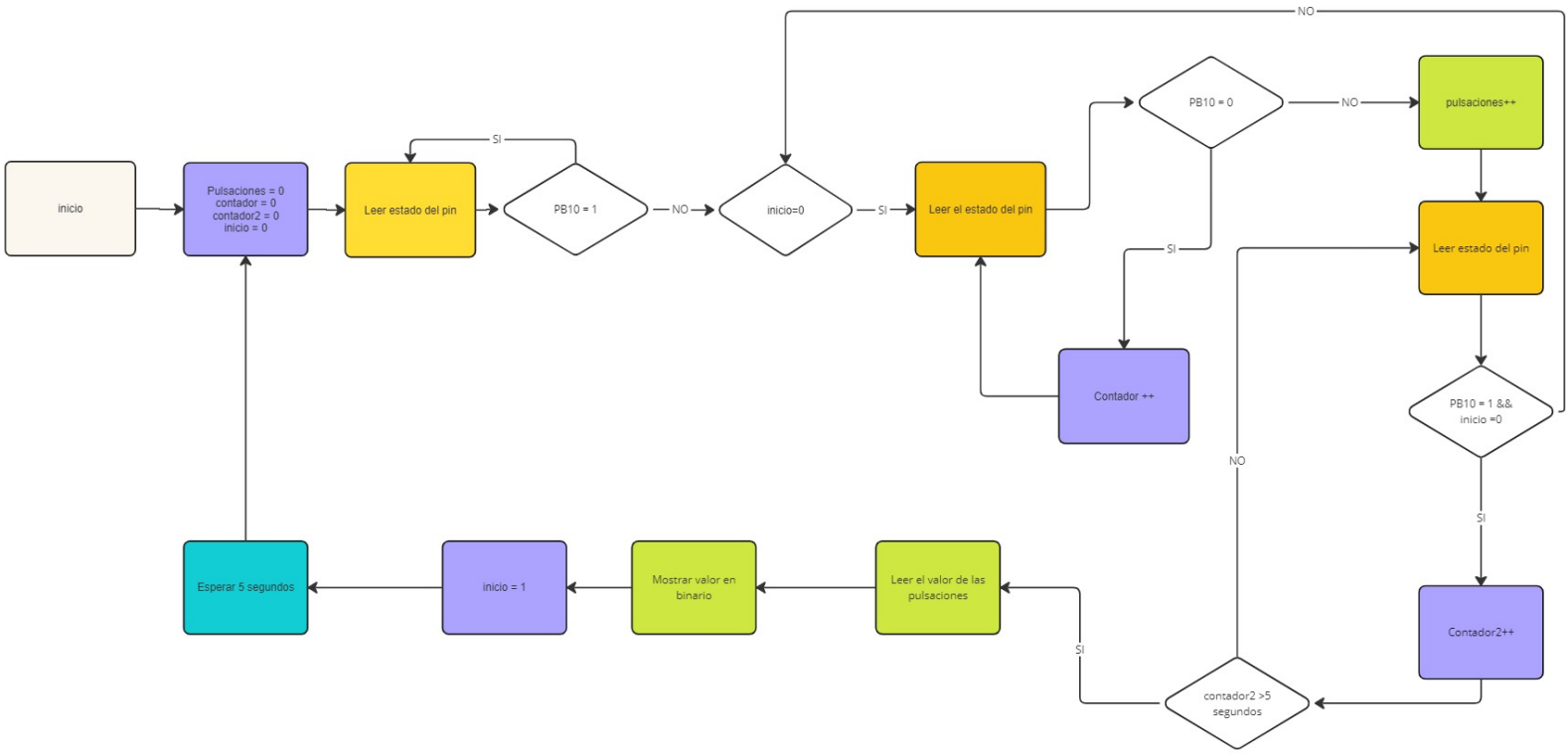
Para encender los leds, se extrajo la dirección de memoria de la variable contador almacenándola en el puntero binario, para así por medio de un bucle for recorrer cada uno de los 8 bits almacenados en esa dirección de memoria. De esta manera se logró guardar cada bit en un arreglo, el cual se repartió en cada puerto de salida del microcontrolador. Con esta parte del código no hubo problema, ya que la lógica implementada fue la idónea.



Al ver que en la primera parte del problema no se estaban obteniendo los resultados requeridos, se cambió el enfoque de la solución. Ahora el enfoque estuvo en la duración del botón pulsado y el botón sin pulsar. Para esto se hizo uso de las variables contador, contador2, inicio y pulsaciones las cuales fueron variando a medida que se cambiaba el estado del pin de entrada. Cuando el pin de entrada cambiaba su estado a cero, se iniciaba el primer contador, el cual midió el tiempo que estuvo presionado el botón. Al volver a su estado en 1, se terminaba este contador, aumentando la variable pulsaciones en 1. Ahora se hace el

conteo del tiempo que el botón estuvo sin presionar, almacenando este valor en la variable contador2. Si contador2 supera un tiempo de cinco segundos, lee el valor que se obtuvo en la variable pulsaciones para mostrarla en los pines de salida del microcontrolador. Al finalizar el código se cambia el estado de la variable inicio para que quede igual a 1 y pulsaciones igual a 0 y así el código vuelva a iniciar todo su proceso.

Como no se tuvo problema en mostrar el valor binario en los leds en el primer análisis, se utilizó el mismo algoritmo anterior.



CONCLUSIONES:

- Es importante analizar las distintas posibilidades que puede haber para una sola solución, ya que el enfoque donde se oriente dicha solución juega un papel fundamental a la hora de hacer un diagrama de flujo y a su posterior codificación.
- Obtener buena documentación es fundamental, debido a que es la fuente en la que se puede conocer el uso de las herramientas (sea software o hardware) que se van a utilizar. En el desarrollo de este laboratorio se evidencio que la stm32f411 tiene fallos en la configuración pull-up en su pin PB2, lo cual tuvo que ser corroborado en foros oficiales.
- El uso del depurador permitió identificar y corregir errores en el código, facilitando la visualización de registros, la inspección de variables, y el seguimiento de la ejecución del programa paso a paso.