Reporte: Práctica de Laboratorio 2

| Victoria Rodríguez de León  *Ingeniería en Robótica y Sistemas Digitales*  *Instituto Tecnológico y de Estudios Superiores de Monterrey*  CDMX, México  [A01656328@tec.mx](mailto:A01661890@tec.mx) | Israel Macías Santana  *Ingeniería en Robótica y Sistemas Digitales*  *Instituto Tecnológico y de Estudios Superiores de Monterrey*  CDMX, México  [A01027029@tec.mx](mailto:A01661649@tec.mx) | Rodríguez Alanis Lisa Valeria  *Ingeniería en Robótica y Sistemas Digitales*  *Instituto Tecnológico y de Estudios Superiores de Monterrey*  CDMX, México  [A01656306@tec.mx](mailto:A01656306@tec.mx) |
| --- | --- | --- |

***Abstract —***

***El adecuado uso de mecanismos de control de flujo, como timers, interrupciones y manejo de comunicación serial por USART, es fundamental en los sistemas embebidos para la eficiencia al gestionar diferentes tareas en tiempo real. El objetivo del presente estudio es conocer el equipamiento de la tarjeta Núcleo H745zi-q respecto a estas funciones con la finalidad de hacer uso de ellas para diferentes programas de aplicación. Esto incluye el uso de timers por polling e interrupción a diferentes frecuencias y así como la transmisión y recepción de datos por USART para la interacción con los programas desarrollados. Los métodos utilizados demuestran una ejecución exitosa validada a través del funcionamiento del hardware empleado así como el monitoreo vía serial para las comunicaciones. A través de este estudio se ponen en práctica competencias que serán clave para futuros desafíos de gestión en el presente medio.***

*Keywords* — NUCLEO H745ZI-Q, STM32CubeIDE, UART, GPIO, *Timers*, Prescalers, Counter Period, *Polling*, Interrupción, Contador.

1. INTRODUCCIÓN

El uso de microcontroladores se ha caracterizado por operar distintos sistemas sean simples o complejos. Su rentabilidad y ahorro de energía ha hecho que se usen en la operación de programas que no requieran controlar procesos tan complejos, por lo que se han vuelto bastante asequibles para esta industria. Operaciones cómo llevar el control de flujo en un sistema de riego, o mantener comunicación con otro microcontrolador para compartir tareas, son funciones que se pueden realizar por medio de la implementación de *timers* internos y puertos de comunicación por protocolos simples como UART o USART. A lo largo de esta práctica se va a revisar el funcionamiento de ambas funciones del microcontrolador STM32 H745IZ-Q con 4 ejercicios que permitan ver la configuración de cada función, así como su implementación práctica en programas sencillos que muestren su uso práctico.

1. MARCO TEÓRICO
2. **Timers en STM32H745**

El STM32H745 cuenta con un timer de alta resolución, denominado HTM1, que opera a 16 bits. Se puede utilizar como un generador de PWM trifásico y ofrece multiplexación en seis canales distintos. Adicionalmente, ofrece dos timers de control de motor avanzado: TIM1 y TIM8. Ambos son de 16 bits y pueden funcionar a frecuencias de hasta 240 MHz. Finalmente, incluye 10 timers de uso general de 16 bits que también pueden operar a esta frecuencia y pueden ser empleados en una amplia gama de aplicaciones que requieren temporización precisa (STMicroelectronics, 2021).

Es importante mencionar que los timers 2, 3, 4 y 5 tienen generación de solicitudes DMA independiente y pueden trabajar juntos o con otros timers de uso general y con timers de control avanzado. Además, su tamaño de palabra es diferente, Los timers 2 y 5 son de 32 bits y los timers 3 y 4 son de 16 bits.

Estos timers se basan en un contador ascendente de recarga automática de 16 bits y un preescalador de 16 bits. Los timers 13, 14, 16 y 17 cuentan con un canal independiente, mientras que los timers 12 y 15 tienen dos canales independientes para captura de entrada/comparación de salida, PWM o salida en modo de un pulso.

Otras características se mencionan a continuación:

* *5 timers de bajo consumo de 16 bits:* Designados como LPTIM1, LPTIM2, LPTIM3, LPTIM4 y LPTIM5, tienen la capacidad de funcionar con un reloj independiente y pueden seguir operativos incluso cuando el dispositivo está en modo Stop. Esto es posible si los timers están sincronizados con un reloj de baja velocidad externo (LSE), un reloj interno (LSI) o un reloj externo.
* *2 timers básicos:* Conocidos como TIM6 y TIM7, con un tamaño de palabra de 16 bits como base de tiempo genérica, se utilizan principalmente para la activación del Convertidor Digital-Analógico (DAC) y la generación de formas de onda.

**B. Configuración del reloj**

El Reset y controlador de reloj (RCC) gestiona la generación de todos los relojes, así como la activación del reloj y el control del sistema y periféricos se reinician. Proporciona una gran flexibilidad en la elección de fuentes de reloj y permite aplicar ratios para mejorar el consumo de energía. El RCC recibe las siguientes entradas de fuente de reloj:

* Oscilador Interno
* Oscilador Externo
  + 64 MHz HSI reloj
  + 48 MHz RC oscilador
  + 4 MHz CSI reloj
  + 32 kHz LSI reloj
* Reloj HSE
  + 4-50 MHz (generado de una fuente externa)
  + 4-48 MHz (generado de un resonador cristal/cerámica)
* Relojes LSE: 32.768 kHz

El RCC brinda 3 PLLs (Phase-Locked Loop), uno para el sistema del reloj y 2 para relojes kernel. El PLL se utiliza para generar una frecuencia de reloj más alta y estable a partir del oscilador principal.

1. **Prescalers**

Los preescaladores, también conocidos como prescalers, son componentes de hardware en sistemas digitales que dividen entre un valor entero (valor del prescaler) la frecuencia de un reloj de entrada antes de que esta señal llegue a ciertos periféricos o subsistemas. La razón por la que se utilizan es que permiten que un contador de tamaño fijo pueda mantener rangos de tiempo más largos al reducir la frecuencia del reloj, haciendo que el contador se incremente más lentamente; lo que añade flexibilidad en el sistema. También se consideran para ahorro de energía al reducir la frecuencia para ciertos periféricos (Leferman et al., 2010). Los tipos de preescaladores que existen son:

* Fijos: Tienen un valor de división que no se puede cambiar. Son más simples pero menos flexibles.
* Configurables: Permiten que el usuario seleccione entre varios valores de división. Son más flexibles pero tienen un mayor nivel de complejidad de configuración.
* Dinámicos: Permiten cambios en tiempo real.

Para el caso de la NUCLEO H745ZI-Q se tienen prescalers configurables que se pueden definir en el workspace de STM32Cube IDE.

1. **Transmisión y recepción de datos por UART**

Universal Asynchronous Receiver/Transmitter (UART), es un protocolo de reglas para el intercambio de datos en serie entre 2 dispositivos (STMicroelectronics, 2021). Este protocolo es muy simple y solo utiliza dos hilos entre el transmisor y el receptor para recibir y transmitir en ambas direcciones. La comunicación en UART puede ser de 3 maneras:

* Simplex ( los datos se envían en una sola dirección)
* Semiduplex ( cada lado transmite, pero solo uno a la vez)
* Dúplex completo (ambos lados pueden transmitir en simultáneo)

Que el protocolo sea asíncrono significa que no hay reloj compartido, por lo que lo que se necesita para que el UART funcione, es configurar el mimo bir o la velocidad de baudios en ambos lados de la conexión. El STM32 cuenta con 4 puertos UART para recibir y transmitir datos.

1. **Contador USART**

USART (Universal Synchronous Asynchronous Receiver Transmitter) permite que un dispositivo se comunique mediante protocolo serial, puede configurarse en modo asíncrono, como transmisor/receptor o en modo síncrono. El uso de Timers en combinación con la comunicación USART es común para controlar la temporización de la comunicación serial. Un contador con USART se basa en configurar un timer para contar y una UART para enviar el valor del contador a través de la comunicación serial. Se pueden transmitir/recibir datos de 2 maneras, por interrupciones o por polling.  
 Para transmitir se verifica que el estado de transmisión esté vacía se puede saber de su estado por interrupción o preguntando en un bucle si el registro está vacío. Cuando el registro esté vacío, carga el byte de datos (valor del contador) en el registro de transmisión de la USART y espera a que se complete la transmisión.

III. METODOLOGÍA

Los materiales requeridos para el desarrollo de la presente práctica fueron:

* Una computadora
* Una tarjeta de desarrollo Núcleo H745SZI-Q
* Un cable micro-USB
* Un monitor serial

En esta sección se explica el procedimiento de configuración de las funciones usadas en cada uno de los ejercicios de la práctica. En ellos se hace uso de los *timers* internos de la tarjeta, de la transmisión y recepción serial por medio del protocolo *USART*, así como la implementación de interrupciones para cambiar el método de uso de ellas mismas.   
 La configuración de los *Timers* de la tarjeta empieza desde la selección del o de los *Timers* que se requieran usar. En la pantalla de *Pinout & Configuration* del entorno gráfico se encuentra la lista de *Timers* disponibles para ambos núcleos de la tarjeta. En esta lista se encuentran Timers de 16 y de 32 bits. Las especificaciones de cada *timer* vienen dentro de ellos una vez que se seleccionan, y desde ahí se pueden hacer las modificaciones para personalizarlos*.* Para los ejercicios, los únicos valores modificados fueron el *Prescaler* y el valor del *Counter Period* para obtener los valores de conteo específicos de cada uno de ellos.   
 El cálculo del valor del *prescaler* está dado por las ecuaciones 1 y 2

… (1)

… (2)

donde la frecuencia de conteo del timer (*ftimx*)está dada por la relación entre el prescaler general que tiene el canal APBx (*fAPBx*) y el valor del *prescaler* personalizado (*PSC*).  
Estos valores se encuentran relacionados con el valor del *counter period*, el cual establece el valor de corte cada vez que el tiempo de conteo se estableció.

**Funciones para Timers**  
 La función necesaria dentro del código para inicializar el conteo de un *Timer* es *HAL\_TIM\_Base\_Start,* la cual inicializa los parámetros base que se generaron cuando se selecciona la configuración del *timer* seleccionado*.* La función recibe como parámetro el apuntador al handler de inicio del *timer*, dónde se encuentran las variables configuradas previamente.  
 La función *HAL\_TIM\_Get\_Counter,* obtiene el valor del registro de conteo del *timer* en el tiempo de ejecución y lo retorna. Esto significa que va a devolver el valor que se encuentre en el registro del *timer* en el momento exacto en que se manda a llamar. El parámetro que recibe es el apuntador al handler del *timer* que se seleccionó.  
 La función*HAL\_TIM\_PeriodElapsed\_Callback* junto con la función *HAL\_TIM\_Base\_Start\_IT***,** permiten el funcionamiento de los *Timers* por interrupción. Al igual que la función de *HAL\_TIM\_Base\_Start,* la función HAL\_TIM\_Base\_Start\_IT sirve para inicializar el conteo del timer y además habilitar su funcionamiento por interrupción. De forma que una vez que su conteo llegó al final, la función *HAL\_TIM\_PeriodElapsed\_Callback* ejecuta el código en su interior cuando se activa la interrupción por *Timer*.  
 La configuración de la comunicación por medio de USART empieza en la selección del protocolo que se quiere usar. En la pantalla de *Pinout & Configuration* del entorno gráfico se encuentra la lista de protocolos de comunicación disponibles para ambos núcleos. Las características de cada protocolo están dentro de ellas una vez que se seleccionan. En el caso de USART se pueden modificar las opciones de baud rate, la paridad, el tamaño de la palabra, el bit de stop, entre otras. Además dentro de la misma ventana, dentro de la pestaña de *NVIC,* se puede habilitar la interrupción del *timer*.

**Funciones para protocolo USART** La función que permite transmitir datos por medio del protocolo USART es*HAL\_UART\_Transmit,*la cual recibe 4 parámetros: un puntero al *handler* con las variables de inicialización del protocolo, (como el valor del *baud rate*, la paridad, si tiene *oversampling*, el tamaño del *buffer* de transmisión y recepción, entre otras), un puntero al buffer de transmisión, la cantidad de elementos que se van a enviar, y el tiempo de duración o *timeout.* La función para recibir por interrupción, al igual que en el caso de los *timers* se hace por medio de la función correspondiente con terminación IT, que, además de recibir, lo hace por medio de interrupción. La función *HAL\_UART\_Receive\_IT* permite la recepción de datos cada vez que la interrupción por recepción se active. Una vez que se detecta este proceso, se pueden llevar a cabo acciones para tratar la información que se recibió. Para eso está la función *HAL\_UART\_RxCpltCallback,*la cual permite la ejecución y proceso de la información una vez que la trama de información que se recibió está completa.

**Ejercicio I. Uso de timers por *polling* y transmisión con USART**

Para la primera parte de esta práctica, se realizó la configuración del *timer\_13* de 16 bits para su posterior uso en el envío de datos por UART. Para esto fue necesario buscar el Timer en el diagrama de bloques en la hoja de datos del microcontrolador con el fin de conocer el prescaler al que está conectado. El que corresponde al Timer seleccionado es el APB1. Después, en la sección de *“Clock Configuration”*, se buscó la frecuencia de trabajo del timer correspondiente a 75 MHz, para realizar el cálculo del prescaler. Para este se necesita la frecuencia deseada, que es de 1 MHz dado que se está calculando un valor para que el *timer* incremente en 1 cada 1 . La obtención de esta frecuencia se puede explicar de mejor manera con la siguiente fórmula:

… (3)

Con esto, ya se tienen todos los elementos para obtener el valor deseado como se muestra a continuación:

… (4)

… (5)

… (6)

El counter period está establecido en la configuración como 65535, que es el valor más grande representable en 16 bits. Sabiendo esto, el tiempo más grande que se podría medir con esta configuración de timer es de 65353 , lo que equivale a 65.535 ms o aproximadamente 0.0655 s.  
 Para el envío de datos por UART, se realizaron algunos cambios en el código inicial como incluir la biblioteca ‘<stdio.h>’, para imprimir texto en la salida estándar. Después se declaró un buffer de 50 caracteres para UART, un entero de 16 bits para el tamaño del mensaje y un entero de 16 bits sin signo para contar el tiempo transcurrido, además, se especificaron los handlers para el timer y UART. Después, en el main se envió un timer test y en el bucle principal se realizó la transmisión del valor del timer.  
 **Ejercicio II. Timer por Interrupción** Para este ejercicio se configuraron los tres LEDs de la tarjeta en GPIO y se habilitó el timer del ejercicio I, además de habilitar las interrupciones. Para la configuración, se calculó un prescaler para que el timer cuente cada 0.2 ms. A su vez, se calculó el counter period para que el contador se reinicie cada segundo. Considerando esto, la frecuencia del timer es de 0.005 MHz y aplicando la fórmula (1), el valor del prescaler es de 14999 y el del *counter period,* de 5000. Esto es porque se quiere que el contador se reinicie cada segundo y cada tick dura 0.2 ms, por lo tanto, al hacer la división, se obtiene el valor final, como se muestra a continuación:

… (7)

Además, se implementó una función de callback para capturar la bandera de interrupción por finalización de conteo utilizando el Timer seleccionado (timer 2). Con el handler se tiene un puntero a la estructura que contiene la configuración del timer. Dentro de la misma función se realizó un Toggle al LED verde de la tarjeta, que cambia cada segundo.  
 **Ejercicio III. Contador Binario por interrupción de Timer** Para el tercer ejercicio se configuró un *timer* de 16 bits con el fin de mostrar en el panel de LEDs integrados un contador binario de 3 bits cada 500 ms. En la pantalla de *Pinout & Configuration* se seleccionó el *timer\_13* con conexión al APB1. Este *timer* es de 16 bits y el prescaler del APB1 es de 75 MHz, por lo que para hacer que el reinicio del conteo del timer se hiciera cada 500 ms, se establecieron los valores del counter period en 2500 y el valor del *prescaler* en 14999. El valor del *prescaler* está dado por las ecuaciones 1 y 2, dónde la frecuencia de conteo del *timer* es de 5 KHz. El valor del *counter period* está dado por la relación que hay entre la frecuencia de conteo y el tiempo de refrescamiento de la secuencia, de modo que, si el periodo del *timer* es de 0.2 ms, y se quiere que los LEDs parpadeen cada 500ms, el counter period va a estar dado por la relación entre la frecuencia de refrescamiento del sistema y la frecuencia de conteo del *timer*.   
 Además, en la implementación del progama se configuraron los pines del LED verde (Puerto B, pin 0), LED amarillo (Puerto E, pin 1), y LED rojo (Puerto B, pin 14) para desplegar el contador binario.   
 La rutina del ejercicio se encuentra en el diagrama de flujo del Anexo 3, donde se especifica el procedimiento en la salida de los pines para llevar a cabo el contador. Esta rutina está dentro del ciclo infinito de la función *main*, pues se requiere que el contador se quede ahí de forma indeterminada, sin embargo, para que el programa funcione por interrupción, dentro de la función HAL\_TIM\_PeriodElapsedCallback se encuentra un contador para que una vez que el timer llegue a su tiempo de reinicio cada 500 ms, se mande llamar la función y el contador se sume.   
 **Ejercicio IV. Contador condicional mediante envío y recepción por USART** Para el cuarto ejercicio se configuró un *timer* de 16 bits y se habilitó la función de comunicación serial UART por interrupción. El objetivo de este ejercicio es usar la transmisión y recepción serial para comunicar la tarjeta a un monitor serial y enviar un contador de 8 bits que cambie su valor cada 500 ms con respecto a la entrada que se dé desde la terminal del monitor. La suma del contador va a variar dependiendo el dato recibido en base a las siguientes condiciones:

* Si se envía una “A” minúscula o mayúscula, el contador debe de sumar su valor de uno en uno
* Si se envía una “B” minúscula o mayúscula, el contador debe de sumar su valor de dos en dos.
* Si se envía una “C” minúscula o mayúscula el contador debe de disminuir su valor.
* Si se envía una “D” minúscula o mayúscula, el contador se debe detener.
* Si se envía una “E” minúscula o mayúscula, el contador se debe reiniciar.

En la pantalla de *Pinout & Configuration* se seleccionó el *timer\_13* con conexión al APB1. Este *timer* es de 16 bits y el prescaler del APB1 es de 75 MHz, por lo que para hacer que el reinicio del conteo del *timer* se haga cada 500 ms, se debe de establecer el valor del counter period en 2500 y el valor del *prescaler* en 14999. El valor del *prescaler* está dado por las ecuaciones 1 y 2, dónde la frecuencia de conteo del *timer* es de 5 KHz.  
 El valor del *counter period* está dado por la relación que hay entre la frecuencia de conteo y el tiempo de refrescamiento de la secuencia, de modo que si el periodo de conteo es de 0.2 ms, y se necesita que el valor se envíe por serial cada 500 ms, el *counter period* va a estar dado por la relación entre la frecuencia de refrescamiento del sistema y la frecuencia de conteo del *timer.*Para transmitir los datos por serial una vez que el *timer* terminó de realizar el conteo, se agrega la función de transmisión por serial sin interrupción en la función de *callback* del timer, pues esta función se manda llamar una vez que el timer llega al valor de corte definido por el *counter period.* Además, dentro de esta función se define la suma del contador y el número representativo de la variable que se recibe desde el monitor serial. Esto quiere decir que la suma del contador y un número representativo se tiene que hacer dentro de la función de *callback* del timer.  
 Por otro lado, para poder recibir el dato que se envía desde el monitor serial se tiene que inicializar la función de recepción por interrupción, de otra forma, la recepción del dato se tendría que hacer por método de *polling* en el ciclo infinito del *while.* Con la implementación de recepción por interrupción se tiene que agregar la función que permite operar la información después de realizar la interrupción. Dentro de esta función se implementa el código que discrimina la información recibida y opera las condiciones para asignar un valor a la variable representativa que se suma al contador en la función de *callback* del *timer.* Un detalle importante de mencionar es que la inicialización de la función de recepción por interrupción se tiene que volver a hacer una vez que se recibe, pues la bandera de esta función se apaga después de vaciar el *buffer* de recepción.El diagrama de flujo que representa el funcionamiento del programa está en el Anexo 4.

IV. RESULTADOS

**Resultados ejercicio 1: Uso de timers por *polling* y transmisión con USART** Para el primer caso fue con un delay de 50 ms, que se reflejó como 51000 en el monitor serial debido a que el programa acomoda su lógica a unidades de microsegundos. Se contempla un margen de error de 1 ms. Fuera de eso, el tiempo reportado va de acuerdo con el tiempo solicitado. Después se realizaron pruebas con diferentes retardos: 5 ms, 60 ms, 70 ms y 150 ms; correspondientemente Al cambiar cada retardo, se puede ver reflejado en el monitor serial. El comportamiento es funcional hasta 70 ms, en donde se produce un desbordamiento. Esto es a causa de que se está utilizando un timer de 16 bits y 70 ms rebasa su capacidad. La solución es implementar el mismo programa con un timer de 32 bits. Se seleccionó el Timer número 2 y se modificó el tipo de dato del buffer de UART así como de la variable del valor del timer. Con esto el comportamiento vuelve a ser normal para 70 ms y 150 ms. Véase Tabla 1.

**Tabla 1**: Pruebas de Timer.

| Tiempo esperado (ms) | Tiempo obtenido | Bits del Timer |
| --- | --- | --- |
| 50 ms | 51000 | 16 bits |
| 5 ms | 5999 | 16 bits |
| 60 ms | 61000 | 16 bits |
| 70 ms | 71000 | 32 bits |
| 150 ms | 151000 | 32 bits |

**Resultados ejercicio II. Timer por Interrupción** Los resultados obtenidos para este ejercicio muestran la implementación del *timer* 2 por interrupción con el uso de la función HAL\_TIM\_PeriodElapsedCallback. Como se puede observar en el video en el anexo, se muestra dicha implementación para la rutina del LED verde en el que realiza un toggle por interrupción del timer. Asimismo, se muestra que se calculó correctamente el prescaler y el counter period para que el contador cuente cada 0.2 segundos y se reinicie cada segundo.  
 **Resultados III. Contador Binario por interrupción de Timer.** Los resultados obtenidos para este ejercicio muestran la implementación de un contador binario visualizando el panel integrado de 3 LEDs disponibles en la tarjeta. El número máximo al que puede contar es 7 y después se reinicia el contador a 0. Con la función HAL\_TIM\_PeriodElapsedCallback se realizó correctamente el contador en un periodo de 500 ms y este cambio es visible en el video del anexo donde se observa como los LEDs cambian conforme el periodo del *timer*. El resultado del contador se lee de la siguiente manera: El número más significativo está representado por el LED rojo y el bit menos significativo por el LED verde.  
 **Resultados IV. Contador condicional mediante envío y recepción por USART** Los resultados obtenidos para este ejercicio muestran el envío y recepción correcto de los datos del programa por medio de comunicación serial. El funcionamiento del programa se muestra en el video del anexo 1 donde se observan los siguientes resultados. El programa inicia con la visualización de un contador en ceros que se refresca cada 500ms. Después de que se introduce una de las letras asignadas por las condicionales en el monitor serial se observa el cambio en el contador. El programa puede inicializarse con la letra A o B, pues son las que modifican el valor de la variable representativa del cambio. Una vez que da inicio el contador, las demás letras pueden enviarse para realizar los cambios necesarios. Sin embargo, cuando cualquier otra letra fuera de las condiciones se presiona, se manda un mensaje que muestra N/A, indicando que no esa letra no altera el valor del contador.

V. CONCLUSIONES INDIVIDUALES

**Victoria Rodríguez de León A01656328** Esta práctica fue una excelente oportunidad para poner en práctica el adecuado uso de mecanismos de control de flujo, como timers, interrupciones y manejo de comunicación serial por USART. Hacer el cálculo manual de los prescalers y del program counter para después configurar el timer en el IDE y ver reflejado el comportamiento en el monitor serial fue ideal para entender bien cómo operan los timers en los microcontroladores. También considero que esta práctica en especial fue útil para mejorar desarrollar habilidades de búsqueda en hojas de datos extensas, buscando palabras clave y secciones asociadas, como sucedió para el caso de la función de callback para que el timer operara por interrupción. Otro punto crítico en la elaboración de esta práctica fue realizar el programa para la recepción de datos, especialmente en la forma en que se declararon los buffers de datos y la ubicación de las funciones en el programa. A grandes rasgos puedo decir que fueron bastante relevantes los contenidos; especialmente porque después utilizaremos todo esto para nuestro reto.

**Lisa Valeria Rodríguez Alanís A01656306**

En esta práctica pusimos a práctica el uso de Timers e Interrupciones por timers , temas que serán muy importantes para el desarrollo del proyecto final. Gracias a este laboratorio, nos familiarizamos e identificamos nuevas funciones encargadas de inicializar y realizar las interrupciones por timer. Abordamos el cálculo del valor del prescaler y el *counter period,* para configurar el timer en el tiempo deseado y podemos concluir que el procedimiento y resultados fueron correctos ya que en todos los ejercicios trabajaron de manera correcta. Por otro lado implementamos la comunicación serial por USART, un protocolo fundamental para entender las interfaces alámbricas. Considero que esta práctica no solo nos ayudó a entender mejor estos temas si no a identificar y a entender cómo realizar estas acciones en el IDE de stm32h745 de una manera más práctica y sencilla. En conclusión, los contenidos de la práctica lograron juntar e implementar temas esenciales en el mundo de los microcontroladores y sistemas embebidos que serán de mucha utilidad para nosotros en el futuro.

**Israel Macías Santana A01027029**Con el desarrollo de esta práctica se empleó el uso de Timers y comunicación serial por protocolo UART para entender el funcionamiento y aplicación de estas funciones. Ambas son fundamentales para su uso en sistemas embebidos por su amplio rango de usos y funcionamientos. Saber cómo se usan los ciclos útiles de un timer por medio del prescaler y el counter period es de mucha importancia para poder contral bien el flujo en tiempo real de la información. Igualmente saber configurar las funciones que permiten la transmisión y recepción de datos por protocolo UART ayuda a saber los parámetros que se necesitan para enviar y recibir datos.   
Otra de las funciones que son necesarias para comprender el correcto uso del proceso de la información dentro de la implementación en un microcontrolador, es conocer cómo funciona la obtención de datos por interrupciones por medio de la inicialización de su función, así como la función de callback que permite procesar esos datos

VI. REFERENCIAS

Wright G. (2022). *USART (universal synchronous/asynchronous receiver/transmitter)* de Tech Target. Recuperado de:

<https://www.techtarget.com/whatis/definition/USART-Universal-Synchronous-Asynchronous-Receiver-Transmitter>

R & S®Essentials (s.f). *Entendiendo la UART* de Rohde-Schwarz. Recuperado de:

<https://www.rohde-schwarz.com/lat/productos/prueba-y-medicion/essentials-test-equipment/digital-oscilloscopes/entendiendo-el-uart_254524.html>

STMicroelectronics. (2023). STM32H745XI - STMicroelectronics. STMicroelectronics. Recuperado de:

<https://www.st.com/en/microcontrollers-microprocessors/stm32h745xi.html>

Leferman, M. J., Pu, D., & Wyglinski, A. M. (2010). GNU radio for cognitive radio experimentation. Elsevier EBooks, 507–538. https://doi.org/10.1016/b978-0-12-374715-0.00018-6

STMicroelectronics. (2021). STM32H745x1/G Datasheet (Rev. 3.0). Recuperado de <https://www.st.com/resource/en/datasheet/stm32h745zg.pdf>

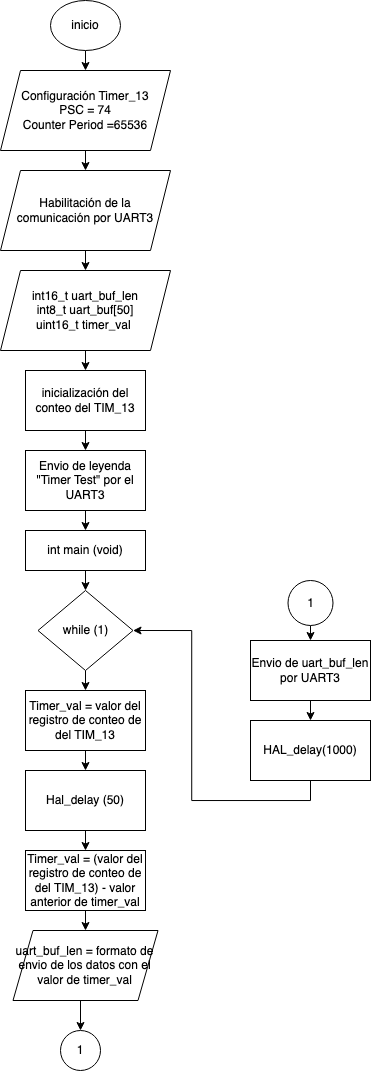
Intel Corporation. (2019). Intel Core i9 Processor [Datasheet]. https://www.intel.com/content/www/us/en/datasheets/i9-processor.html

VII. ANEXOS

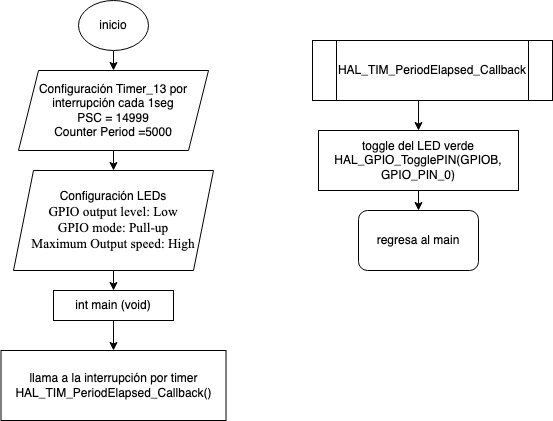
Video de evidencia (link YouTube)  
:<https://youtu.be/66nw8ScIvcE>

Link a presentación de Canva: <https://www.canva.com/design/DAFwCq7JD4Y/LewIz79OwhiP9hKqKeyN8A/edit?utm_content=DAFwCq7JD4Y&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>

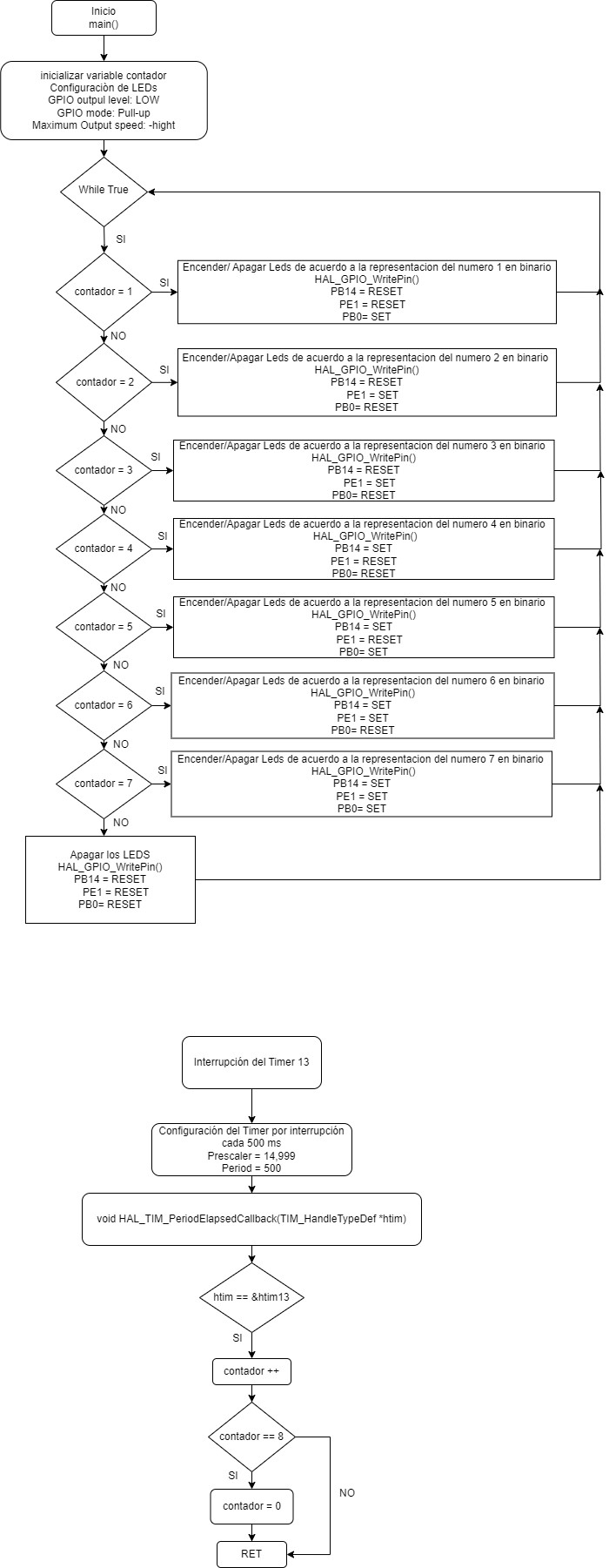
**Anexo 1.** Diagrama de flujo ejercicio 1



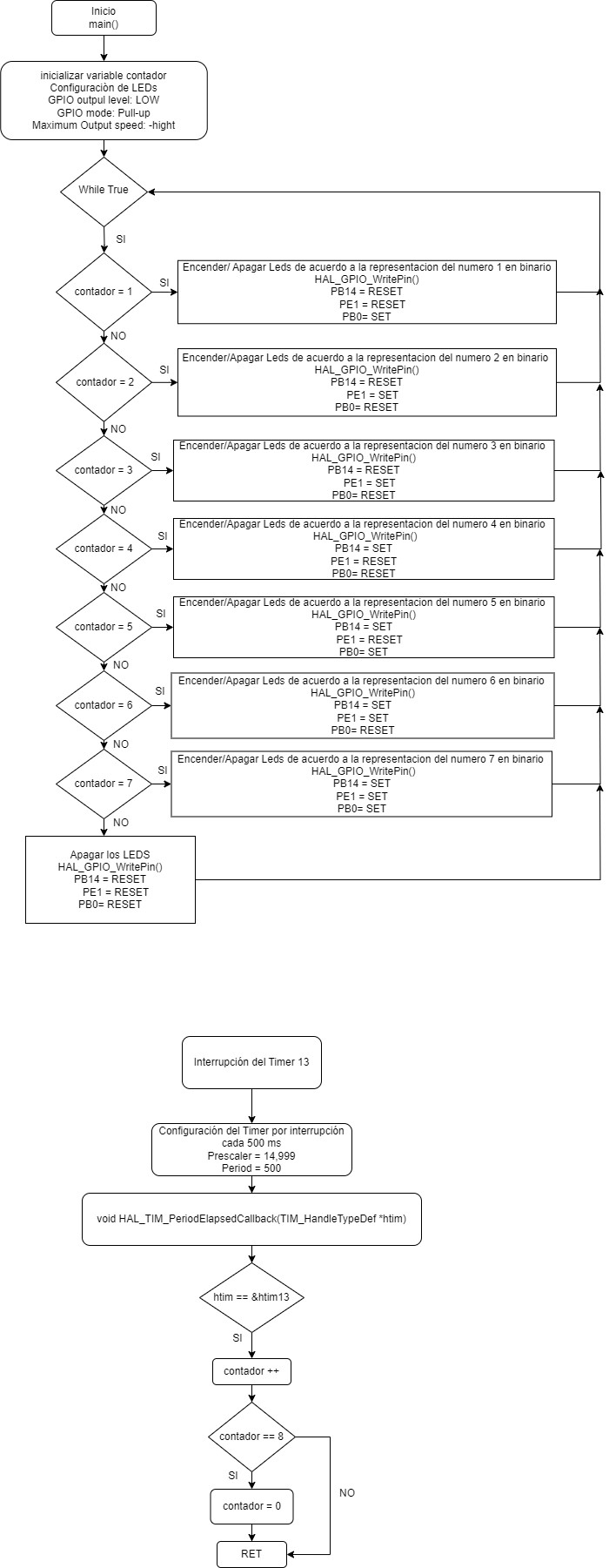
**Anexo 2.** Diagrama de flujo ejercicio 2



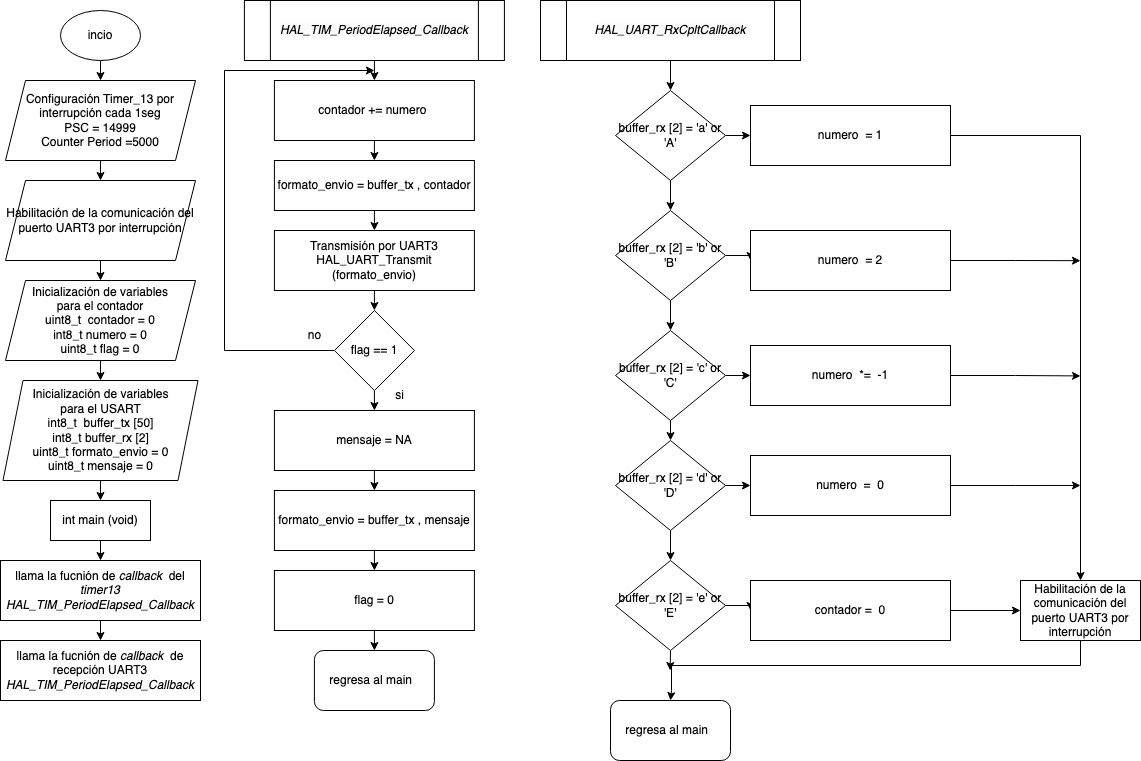
**Anexo 3.** Diagrama de Flujo ejercicio 3



**Anexo 3.1** Interrupciòn de Timer



**Anexo 4.** Diagrama de flujo ejercicio 4



: