Sistemas embebidos: Reloj led con STM32

Integrantes: Monica Julieth Paez,
Lizeth Camila Sánchez

***Resumen*—El sistema de reloj digital utiliza un microcontrolador STM32 y LEDs dispuestos como manecillas de reloj. Tiene dos modos: run", donde muestra horas y minutos al pulsar el botón, y "set", donde se ajusta la hora al mantener el botón. En run", el LED de horas se enciende y el de minutos parpadea cinco veces cada 300 ms, seguido de un apagado. En "set", el sistema emite dos pulsos de sonido con el buzzer para confirmar el cambio de modo. El sistema de reloj digital propuesto es una implementación práctica y funcional que utiliza los módulos GPIO y la función de retardo "delay"del microcontrolador STM32.**

***Keywords*—** Microcontrolador STM32, LEDs, Reloj Digital, Modos de Operación, GPIO, Función Delay, Botón de Control, Buzzer

I. Introducción

El sistema de reloj digital propuesto utiliza un microcontrolador STM32 con sus módulos GPIO y la función de retardo "delay". Los LEDs están dispuestos en un patrón que simula las manecillas de un reloj analógico. El sistema opera en dos modos: runz "set".

En el "modo run", el sistema cuenta internamente las horas, minutos y segundos. Para visualizar la hora actual, el usuario debe pulsar una vez el botón. El sistema enciende el LED correspondiente a las horas y hace parpadear cinco veces el LED de los minutos, con un intervalo de 300 ms entre cada parpadeo. Después de 1500 ms, los LEDs se apagan.

En el "modo set", el usuario puede fijar la hora y los minutos actuales. Para cambiar al "modo set", el usuario debe mantener presionado el botón durante 5000 ms. El sistema emite un sonido de dos pulsos con el buzzer, con un intervalo de 300 ms entre cada pulso, para confirmar el cambio de modo. Una vez en el modo de configuración (modo set), el LED correspondiente a las horas se iluminará. Al presionar el botón una vez, se aumentará el valor de las horas en incrementos de una hora. Para ingresar el valor de los minutos, se debe mantener presionado el botón durante 3000 ms, lo que emitirá un sonido de tres pulsos con intervalos de 300 ms. Luego, al presionar el botón una vez, se aumentará el valor de los minutos en incrementos de 5 minutos. Para volver al modo de ejecución normal (modo run), se debe mantener presionado el botón durante 5000 ms, lo que emitirá un sonido de un pulso de 2000 ms para confirmar.

El sistema de reloj digital propuesto es una implementación práctica y funcional que utiliza los módulos GPIO y la función de retardo "delay"del microcontrolador STM32.

II. Marco teorico

La placa STM32F401CEUx es una plataforma de desarrollo de microcontroladores de la serie STM32 de la empresa STMicroelectronics. Los microcontroladores de esta serie se basan en la arquitectura ARM Cortex-M y son conocidos por su alta eficiencia energetica y su alto rendimiento. El ´ microcontrolador STM32F401CEU6 que se encuentra en la placa cuenta con una arquitectura de procesador ARM CortexM4 de 32 bits y una velocidad de reloj de hasta 84 MHz. Esta arquitectura permite un alto rendimiento de procesamiento de datos y la posibilidad de realizar tareas complejas en tiempo real[1]

La placa ofrece una amplia gama de características, incluyendo interfaces de comunicacion como USB, UART, SPI y I2C. Estas interfaces permiten la comunicacion con otros dispositivos y la transmision de datos. Además, la placa cuenta con una gran cantidad de pines GPIO que permiten la conexión de sensores y actuadores, lo que permite la implementación de proyectos de electronica y robótica, también cuenta con una memoria Flash de 512 KB y SRAM de 128 KB, lo que permite el almacenamiento de programas y datos. Esto es especialmente importante en proyectos donde se requiere una gran cantidad de memoria para el procesamiento de datos complejos y la implementacion de algoritmos de control.

Esta posee una gran compatibilidad con una amplia gama de herramientas de desarrollo, incluyendo el entorno de desarrollo integrado STM32CubeIDE y otros entornos de desarrollo populares como Keil y IAR. Estas herramientas permiten a los desarrolladores crear, depurar y optimizar sus aplicaciones de manera eficiente [2].

III. Diagrama de flujo

El diagrama de flujo, que se encuentra en la Figura 2 y 3 de los anexos, es una representación visual del proceso general del programa. En él, se puede observar cómo todo el proceso se inicia desde el momento en que se pulsa el botón. A partir de esta acción, se desencadenan una serie de eventos que incluyen la visualización de la hora, el ajuste de los minutos y horas, y el retorno al modo de ejecución normal (modo run").

IV. Procedimiento

Comenzando con el entendimiento del laboratorio, se priorizó el diseño esquemático del circuito, el cual sería fundamental para lograr la visualización intuitiva de la hora mediante LEDs. Además, este diseño debía permitir la configuración de la hora con la ayuda de un pulsador en modo pull down y una bocina que indicaría el ingreso al modo de configuración. A continuación, se presenta el esquemático utilizado para implementar la solución al problema planteado. [Anexos: Figura 1]

El bucle principal del programa, que se ejecuta continuamente mientras el microcontrolador está encendido. En esta parte del código, se lee el valor del pulsador utilizando la función HAL_GPIO_ReadPin(), que devuelve 1 si el botón está presionado y 0 si no lo está. Este valor se almacena en la variable button_counter. A continuación, se ejecuta un bucle while que se repite mientras el botón esté presionado. Dentro de este bucle, se incrementa el contador counter en 1 cada milisegundo utilizando la función HAL_Delay(1). También se llama a la función real_time() para actualizar las variables de tiempo (horas, minutos y segundos).

Dentro del bucle while, se comprueba si la variable status es falsa. Si es así, se establece counter en 0 y se cambia el valor de status a verdadero. Esto se hace para asegurarse de que counter se reinicie cada vez que se presione el botón. Después de salir del bucle while, se comprueba si button_counter es mayor o igual a 1000. Si es así, se llama a la función mode_run(), que es la función que se encarga de mostrar la hora actual en los LEDs. Si button_counter es mayor que 5000, se llama a la función mode_set(), que es la función que se encarga de ajustar la hora actual.

Finalmente, se reinician las variables button_counter y status a 0 y falso, respectivamente, para que estén listas para la próxima vez que se presione el botón.

```
    /* USER CODE BEGIN 3 */
          HAL_Delay(1);
          counter++;
          real_time();
          while(!(HAL_GPIO_ReadPin(GPIOB,
   GPIO_PIN_5) == 0))
          {
              if (status == false) {
                  counter = 0;
                  status = true;
              }
              HAL_Delay(1);
              counter++;
              button_counter++;
              real_time();
          }

          if ((button_counter >= 400) && (button_counter< 2000))
        {
                  button_counter = 0;
                  mode_run();
                  button_counter = 0;

          }
          else if (button_counter > (2000))
          {
                  button_counter = 0;
              mode_set();
              button_counter = 0;
          }

          button_counter = 0;
          status = false;

  }
  /* USER CODE END 3 */
```

La función real_time() es responsable de actualizar las variables de tiempo (horas, minutos y segundos) en función del valor del contador counter. Este contador se incrementa cada milisegundo en el bucle principal del programa. La función real_time() se ejecuta cada vez que se presiona el botón y se mantiene presionado, lo que permite que las variables de tiempo se actualicen continuamente mientras el botón está presionado.

```
void real_time()
{
    if(counter >= segundo)
    {
        seconds++;
        if (seconds >= 60)
        {
            seconds = 0;
            minutes++;
            if (minutes >= 60)
            {
                minutes = 0;
                hours++;
                if (hours >= 11)
                {
                    hours = 0;
                }
            }
        }
        counter=0;
    }
}
```

La función show_hours() se encarga de encender el LED correspondiente a las horas en el reloj. Recibe como argumentos el puerto GPIO y el pin correspondiente al LED de las horas. La función HAL_GPIO_WritePin() se utiliza para encender el LED.

```
switch (h) {
 case 0:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 1);
break;
case 1:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, 1);
 break;
 case 2:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, 1);
break;
 case 3:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1);
    break;
 case 4:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 1);
 break;
 case 5:
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, 1);
break;
 case 6:
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, 1);
break;
case 7:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);
break;
 case 8:
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 1);
break;
case 9:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 1);
break;
case 10:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 1);
break;
case 11:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
break;
default:
break;
   }
}
```

La función show_minutes() se encarga de encender el LED correspondiente a los minutos en el reloj. Recibe como argumentos el puerto GPIO y el pin correspondiente al LED de los minutos. La función HAL_GPIO_WritePin() se utiliza para encender el LED. Además, la función show_minutes() incluye un bucle for que se ejecuta 10 veces. Dentro de este bucle, se enciende y apaga el LED de los minutos utilizando la función HAL_GPIO_WritePin() y se actualiza el contador counter cada milisegundo utilizando la función HAL_Delay(1) y la función real_time().

```
void show_minutes(GPIO_TypeDef* port,
uint16_t Pin)
{
    for (int i = 0; i < 10; i++)
    {
```

```
        HAL_GPIO_WritePin(port, Pin,
         !(HAL_GPIO_ReadPin(port, Pin)));
        button_counter = 0;
        while( button_counter <= 300)
                {
                        HAL_Delay(1);
                        button_counter++;
                        counter++;
                        real_time();
                }
    }
}
```

V. Conclusiones

- Al implementar el control de tiempo mediante el uso de Delay, es importante tener en cuenta el tiempo que el programa tarda en realizar las operaciones de comparación y conteo en relación con el tiempo real. Esto puede proocar errores de 1 segundo o más durante la ejecución del programa, el cual va en aumento, es por esto que decidimos buscar un valor para reemplazar en las condiciones que involucran los milisegundos.
- Se busco organización en el código al poner todo en diferentes funciones y llamarlas cuando se requeria.

VI. Anexos

VI-A. Esquematico

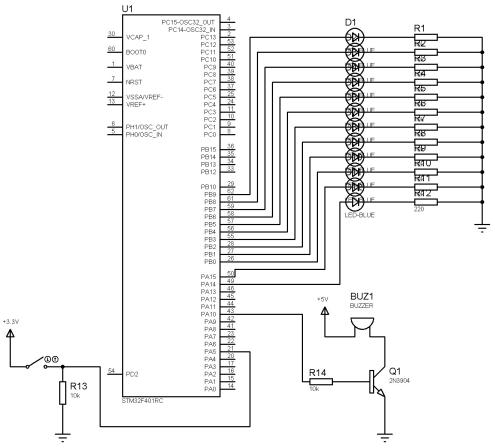

Figura 1. Esquemático del circuito en Proteus

VI-B. Diagrama de flujo

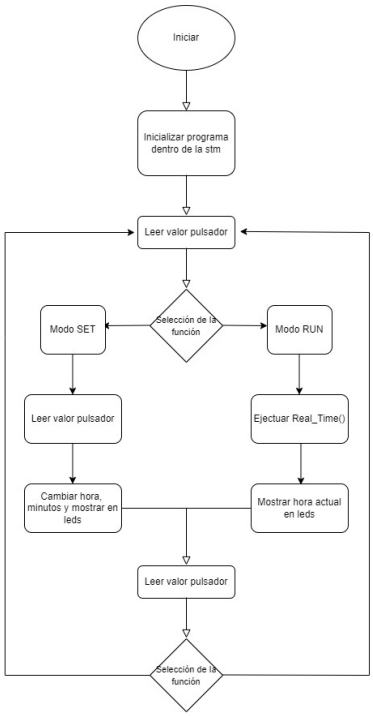

Figura 2. Diagrama de flujo general del código

VII. Referencias

[1] “Stm32f411ce.” [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f411ce.html [2] C. Guindon, “The community for open innovation and collaboration: The eclipse foundation.” [Online]. Available: https://www.eclipse.org/ [3] “Stm32cubeide.” [Online]. Available: https://www.st.com/en/development-tools/stm32cubeide.html

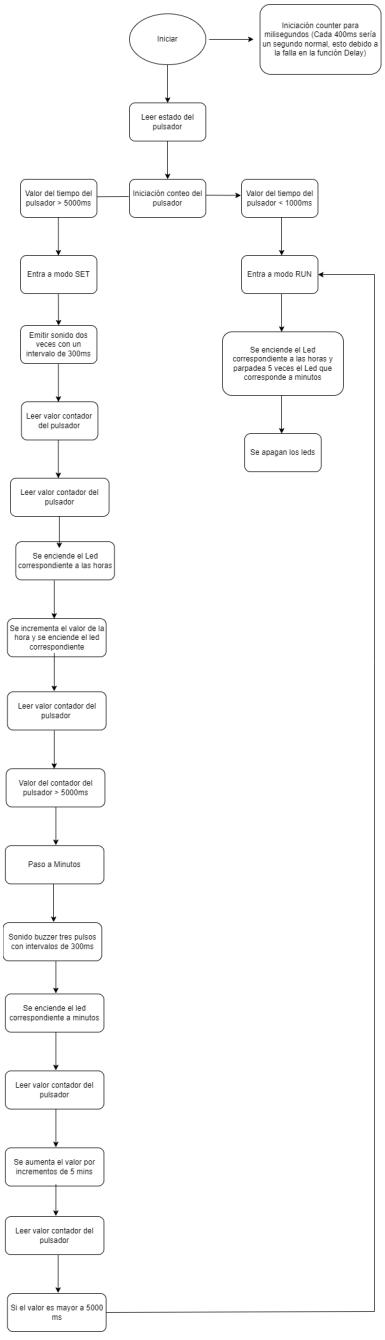

Figura 3. Diagrama de flujo por el contador del boton