Fase 2 - Reloj/Alarma

Henry José, Chacón Barillas, 202002535
Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala

El proyecto consiste en realizar un reloj en tiempo real y una alarma que indique el inicio y final de un periodo determinado, en este caso sera del laboratorio, utilizando todos los conocimientos adquiridos de la clase magistral y las practicas realizadas en el laboratorio.

I. OBJETIVOS

- Aplicar la inicialización de los GPIO´s del microcontrolador y crear un programa que sea capaz de realizar retardos.
- Crear un programa que sea capaz de configurar los respectivos modulos.
- Comprender los retardos a modo de contadores.
- Calcular los valores necesarios de acuerdo al reloj del microcontrolador.

II. MARCO TEÓRICO

A. Keil μVision

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) ampliamente utilizado para el desarrollo de software embebido, especialmente en microcontroladores basados en la arquitectura ARM. Fue desarrollado por Keil, una empresa que ahora es parte de ARM Holdings, y es uno de los entornos de desarrollo más populares para programar y depurar microcontroladores.

Caracteristicas:

- Soporte de Microcontroladores: Keil uVision es compatible con una amplia gama de microcontroladores basados en ARM, como los de la serie Cortex-M, y también soporta otros microcontroladores como los basados en 8051.
- Compilador ARM (ARMCC): Incluye el compilador de ARM, que optimiza el código para la arquitectura ARM, permitiendo un rendimiento eficiente y un uso reducido de memoria.
- Depuración y Simulación: Ofrece herramientas avanzadas de depuración y simulación que permiten al usuario probar y depurar el código en tiempo real o a través de simulaciones de hardware.
- Editor de Código: Tiene un editor de código integrado con soporte para sintaxis destacada, autocompletado, y herramientas de búsqueda que facilitan la escritura y edición de código.
- Gestión de Proyectos: Permite la gestión de proyectos con múltiples archivos, automatización de tareas de compilación y enlace, y configuración de opciones de compilación.
- Configuración de Periféricos: Para microcontroladores compatibles, uVision incluye herramientas gráficas que permiten configurar fácilmente los periféricos, pines, y otros recursos del microcontrolador.
- Soporte para Lenguaje Ensamblador y C/C++: uVision permite escribir código en ensamblador, C y C++, lo que brinda flexibilidad en la programación y optimización del código para aplicaciones embebidas.

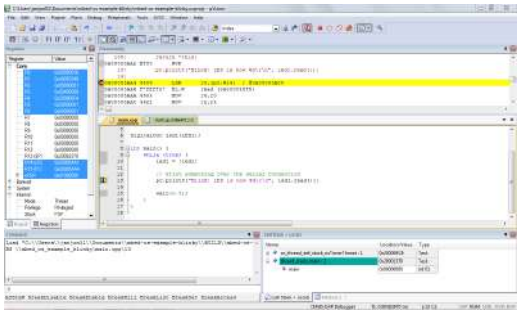

Figura 1: Entorno de Keil μVision

B. Tiva TM4C123GH6PM

La Tiva TM4C123GH6PM es un microcontrolador de 32 bits basado en la arquitectura ARM Cortex-M4. Es parte de la serie Tiva C de Texas Instruments y está diseñado para aplicaciones de alto rendimiento en sistemas embebidos. Este microcontrolador cuenta con características como:

- Frecuencia de operación de hasta 80 MHz.
- Memoria flash de 256 KB y RAM de 32 KB.
- Múltiples interfaces de comunicación como UART, I2C, SPI y USB.
- Módulos de temporizadores, ADC de 12 bits y controladores PWM, ideales para sistemas de control y adquisición de datos.

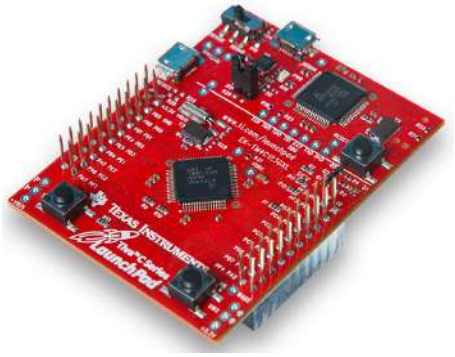

Figura 2: Placa de desarrollo

C. Modulo display de 7 segmentos TM1637

El TM1637 es un controlador de display LED de 7 segmentos utilizado comúnmente para manejar pantallas de 4 dígitos. Permite una fácil interfaz entre un microcontrolador y la pantalla, usando solo dos pines de control: CLK (clock) y DIO (data input/output). Es popular en proyectos de electrónica debido a su simplicidad y la capacidad de controlar tanto los segmentos como el brillo de los dígitos. Se usa típicamente para mostrar datos numéricos como tiempo, temperatura o cualquier información basada en números.

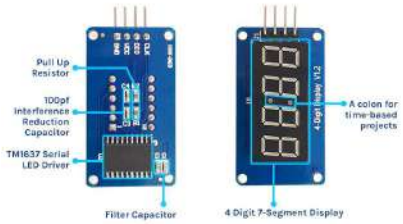

Figura 3: Modulo display de 7 segmentos

D. Modulo de reloj DS1302

El DS1302 es un módulo de reloj en tiempo real (RTC, por sus siglas en inglés) que mantiene un seguimiento preciso de la hora y la fecha, incluso cuando el microcontrolador está apagado. Funciona con una batería de respaldo, lo que permite que el reloj siga funcionando cuando no hay alimentación principal. El DS1302 puede almacenar segundos, minutos, horas, día, mes, año, y también incluye un sistema de corrección para años bisiestos. Se comunica con el microcontrolador mediante un protocolo de comunicación serie simple que utiliza tres pines: CE, I/O, y SCLK.

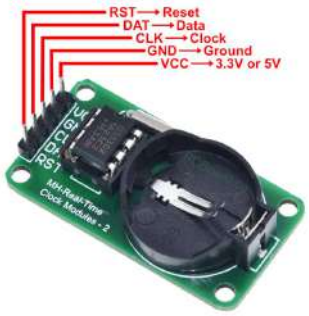

Figura 4: Modulo de reloj

E. Modulo regulador de voltaje

Este modulo se utilizo en el proyecto para alimentar los módulos mencionados anteriormente, ya que el voltaje interno de la tiva, no es capaz de encender los respectivos módulos.

Figura 5: Modulo de voltaje

F. Buzzer

Un buzzer activo es un dispositivo electrónico que emite sonidos o pitidos al recibir una señal eléctrica. A diferencia del buzzer pasivo, que requiere una señal de control con una frecuencia específica para generar un sonido, el buzzer activo tiene un oscilador interno. Esto significa que solo necesita una señal de alimentación, generalmente de corriente continua, para activarse y emitir un tono constante. Los buzzers activos son fáciles de usar en aplicaciones de señalización o alertas audibles.

Figura 6: Buzzer

G. Resistencia

Una resistencia es un componente eléctrico pasivo que limita o regula el flujo de corriente en un circuito. Funciona oponiéndose al paso de la corriente eléctrica, lo que permite controlar la cantidad de corriente que circula a través de otros componentes. La unidad de medida de la resistencia es el ohmio (Ω). Las resistencias se usan comúnmente para proteger circuitos, dividir tensiones, y ajustar niveles de corriente. Existen diferentes tipos, como las fijas y las variables, dependiendo de si su valor resistivo es constante o ajustable.

Figura 7: Resistencia eléctrica

H. Transistor

Un transistor es un componente electrónico semiconductor que se utiliza para amplificar o conmutar señales eléctricas. Tiene tres terminales llamados emisor, base, y colector. Su función principal es controlar el flujo de corriente o voltaje entre el emisor y el colector, utilizando una pequeña corriente en la base. Los transistores son esenciales en la mayoría de los circuitos electrónicos y son la base de componentes como los microprocesadores y amplificadores. Existen diferentes tipos, como transistores bipolares (BJT) y de efecto de campo (FET).

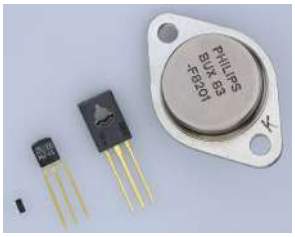

Figura 8: Tipos de transistores

III. DIAGRAMA DE FLUJO

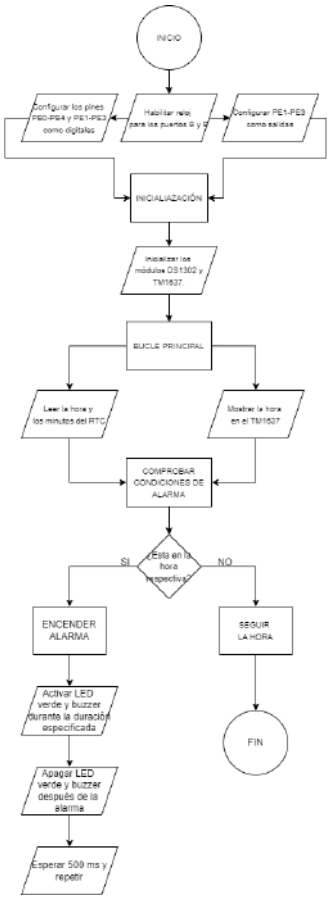

Figura 9: Algoritmo del programa realizado

IV. DISEÑO EXPERIMENTAL

A. Procedimiento

1. *Configuración del Hardware*

Registros de Control y Pines:

- Se definen direcciones de memoria para acceder a los registros de GPIO del sistema (ej. SYSCTL_RCGCGPIO_R, GPIO_PORTB_DEN_R, etc.).
- Se asignan constantes a los pines que se utilizarán, como LED_V, LED_R, BUZZER, CLK, y DIO.

2. *Inicialización*

Habilitación de reloj

- Se carga el registro de control de GPIO y se habilitan los puertos B y E con un OR de 0x12.

- Se permite un par de NOPs (No Operation) para asegurar que la configuración tenga efecto antes de continuar.

3. *Configuración de GPIO*

Pines digitales:

- Los pines del puerto B se configuran como digitales mediante un OR en el registro de habilitación de datos de GPIO (GPIO_PORTB_DEN_R).
- Lo mismo ocurre con los pines del puerto E, permitiendo el uso de LEDs y buzzer.

4. *Definición de Salidas*

- Se configuran los pines PE1, PE2 y PE3 como salidas, que controlarán el LED verde, el LED rojo y el buzzer.

5. *Inicialización de Dispositivos*

RTC y TM1637:

- Se llaman a las subrutinas initRTC y initTM1637 que, aunque no están implementadas, deberían configurar adecuadamente el RTC DS1302 y el display TM1637.

6. *Bucle de Ejecución*

Se utiliza un bucle infinito para repetir la lógica de lectura y actualización del reloj.

Lectura del Tiempo:

- La función readTime obtiene la hora y minutos del RTC y los almacena en R4 y R5 respectivamente.

Actualización de Pantalla:

- Se llama a updateDisplay para enviar la hora y minutos al display TM1637.

Activación de Alarmas:
Se verifica si se cumplen las condiciones para activar alarmas

- Si la hora es 12:30 a 12:35, se activa la alarma corta.
- Si es exactamente 12:40, se activa la alarma larga.
- Si no se cumplen las condiciones, se apaga el buzzer y se enciende el LED rojo.

7. *Espera*

- Se implementa un retardo de 500 ms antes de repetir el ciclo, utilizando la subrutina delay500ms.

V. CÓDIGOS

A. Código del reloj

```
1
2  ; Definir los registros de control y los pines
       de los componentes
3  SYSCTL_RCGCGPIO_R    EQU 0x400FE608
4  GPIO_PORTB_DATA_R    EQU 0x400053FC
5  GPIO_PORTE_DATA_R    EQU 0x400243FC
6  GPIO_PORTB_DIR_R     EQU 0x40005400
7  GPIO_PORTE_DIR_R     EQU 0x40024400
8  GPIO_PORTB_DEN_R     EQU 0x4000551C
9  GPIO_PORTE_DEN_R     EQU 0x4002451C
10 LED_V                EQU 0x02    ; PE1
11 LED_R                EQU 0x04    ; PE2
12 BUZZER               EQU 0x08    ; PE3
13
14 ; Pines para el TM1637
15 CLK                  EQU 0x08    ; PB3
16 DIO                  EQU 0x10    ; PB4
17
18     AREA    codigo, CODE, READONLY
19     THUMB
20     EXPORT Start
21
22 Start
23     ; Habilitar reloj para puertos B y E
24     LDR R1, =SYSCTL_RCGCGPIO_R
25     LDR R0, [R1]
26     ORR R0, R0, #0x12     ; Habilitar puerto B y
        puerto E
27     STR R0, [R1]
28     NOP
29     NOP
30
31     ; Configurar pines PB y PE como digitales
32     LDR R1, =GPIO_PORTB_DEN_R
33     LDR R0, [R1]
34     ORR R0, R0, #0x1F     ; Habilitar pines PB0 a
        PB4 como digitales
35     STR R0, [R1]
36
37     LDR R1, =GPIO_PORTE_DEN_R
38     LDR R0, [R1]
39     ORR R0, R0, #0x0E     ; Habilitar pines PE1,
       PE2, PE3 como digitales
40     STR R0, [R1]
41
42     ; Configurar pines PE1, PE2, PE3 como
       salidas
43     LDR R1, =GPIO_PORTE_DIR_R
44     LDR R0, [R1]
45     ORR R0, R0, #0x0E     ; Configurar PE1, PE2,
       PE3 como salidas
46     STR R0, [R1]
47
48     ; Inicializar DS1302 y TM1637
49     BL initRTC
50     BL initTM1637
51
```

```
Loop
    ; Leer hora y minutos del RTC
    BL readTime
    MOV R4, R0            ; Guardar minutos en
    R4
    MOV R5, R1            ; Guardar horas en R5

    ; Mostrar hora en el TM1637
    BL updateDisplay

    ; Comprobar condiciones para la alarma
    CMP R5, #18           ; Verificar si es la
    hora 18 (6 PM)
    BNE CheckOtherTime

    CMP R4, #00           ; Verificar si los
    minutos est n entre 00 y 05
    BGE AlarmOneMinute
    CMP R4, #05
    BLT AlarmOneMinute

    CMP R4, #06           ; Verificar si es
    exactamente 18:06
    BEQ AlarmFullMinute

    CMP R4, #10           ; Verificar si los
    minutos est n entre 10 y 11
    BGE AlarmOneMinute
    CMP R4, #11
    BLT AlarmOneMinute

    ; Si no est  en ninguno de los intervalos,
    enciende el LED rojo
    LDR R1, =GPIO_PORTE_DATA_R
    LDR R0, [R1]
    BIC R0, R0, #0x0E     ; Apagar BUZZER y LED
    verde
    ORR R0, R0, #LED_R    ; Encender LED rojo
    STR R0, [R1]

    B ContinueLoop

CheckOtherTime
    ; Aqu  se puede agregar l gica adicional
    si es necesario
    B ContinueLoop

AlarmOneMinute
    ; Llama a la alarma por 1 minuto
    MOV R0, #1
    BL triggerAlarm
    B ContinueLoop

AlarmFullMinute
    ; Llama a la alarma por 60 segundos
    MOV R0, #60
    BL triggerAlarm

ContinueLoop
    ; Esperar 500 ms antes de la siguiente
    iteraci n
    BL delay500ms
    B Loop

; Subrutina para activar la alarma
triggerAlarm
    ; R0 contiene la duraci n en segundos
    PUSH {LR}             ; Guardar el valor de
    retorno
    MOV R1, #5            ; Multiplicar el
    tiempo por 5 para 200 ms de intervalos
    MUL R0, R0, R1

TriggerLoop
    LDR R1, =GPIO_PORTE_DATA_R
    LDR R2, [R1]
    ORR R2, R2, #LED_V    ; Encender LED verde
    ORR R2, R2, #BUZZER   ; Encender BUZZER
    BIC R2, R2, #LED_R    ; Apagar LED rojo
    STR R2, [R1]

    BL delay100ms         ; Espera 100 ms

    LDR R2, [R1]
    BIC R2, R2, #LED_V    ; Apagar LED verde
    BIC R2, R2, #BUZZER   ; Apagar BUZZER
    STR R2, [R1]

    BL delay100ms         ; Espera otros 100 ms

    SUBS R0, R0, #1       ; Reducir el contador
    BNE TriggerLoop

    POP {LR}              ; Restaurar el valor
    de retorno
    BX LR

; Subrutinas para inicializar el RTC y el TM1637
initRTC
    ; Inicializaci n del DS1302 (debe ser
    implementado)
    BX LR

initTM1637
    LDR R1, =GPIO_PORTB_DIR_R
    LDR R0, =GPIO_PORTB_DATA_R
    ; Configurar pines para el TM1637
    LDR R2, [R1]
    ORR R2, R2, #CLK      ; Configurar PB3 como
    salida
    ORR R2, R2, #DIO      ; Configurar PB4 como
    salida
    STR R2, [R1]
    ; Aqu  puedes implementar la
    inicializaci n b sica , como configurar el
    brillo
    BX LR

; Subrutina para leer la hora y los minutos del
    RTC
readTime
    ; Leer la hora y minutos del DS1302 (debe
    ser implementado)
    ; Almacenar los minutos en R0 y las horas en
     R1
    BX LR

; Subrutina para actualizar la pantalla del
    TM1637
updateDisplay
    PUSH {R0, R1, LR}          ; Guardar R0, R1
    y el valor de retorno

    LDR R0, =GPIO_PORTB_DATA_R ; Direcci n del
    puerto de datos del TM1637

    ; Comenzar a enviar datos al TM1637
    MOV R1, #0x40              ; Comando de
    escritura
    STR R1, [R0]               ; Enviar comando
```

```
168
169     ; Enviar los d gitos de la hora y minutos
170     MOV R1, R5                  ; Cargar horas
171     STR R1, [R0]                ; Enviar horas
172     MOV R1, R4                  ; Cargar minutos
173     STR R1, [R0]                ; Enviar minutos
174
175     ; Finaliza la comunicaci n
176     MOV R1, #0x80               ; Comando de
        finalizaci n
177     STR R1, [R0]                ; Enviar comando
178
179     POP {R0, R1, PC}            ; Restaurar R0,
        R1 y volver
180
181 ; Subrutinas para retardos
182 delay100ms
183     LDR R1, =1000000           ; Ajusta el valor
        seg n la frecuencia del reloj
184 DelayLoop100
185     NOP                         ; No operaci n
186     SUBS R1, R1, #1            ; Decrementar R1
187     BNE DelayLoop100            ; Repetir hasta
        que R1 llegue a 0
188     BX LR                       ; Retornar
189
190 delay500ms
191     LDR R1, =5000000            ; Ajusta el valor
        para 500 ms
192 DelayLoop500
193     NOP                         ; No operaci n
194     SUBS R1, R1, #1            ; Decrementar R1
195     BNE DelayLoop500            ; Repetir hasta
        que R1 llegue a 0
196     BX LR                       ; Retornar
197
198     END
```

Listing 1: Entorno de keil uvision

VI. RESULTADOS

A. Circuito en funcionamiento

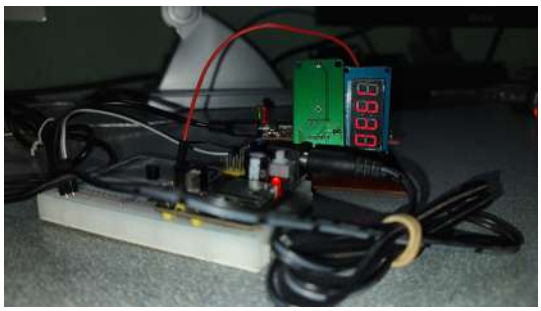

Figura 10: Circuito armado en protoboard y en placa de cobre

B. Proceso de creación del circuito

Para realizar el circuito respectivo se realizaron varios procesos, en el cual se realizo como primera instancia el diagrama respectivo del circuito hecho en el programa PCBWizard, luego se imprimió el circuito a láser para luego pegarlo en la placa de cobre, luego del proceso del planchado se introdujo la placa al proceso del ácido para eliminar los restos de cobre que no utilizaremos, luego se procedió a barrenar los respectivos orificios donde se colocaran los componentes y después aplicarles soldadura y por ultimo se colocaron los componentes en sus respectivos lugares.

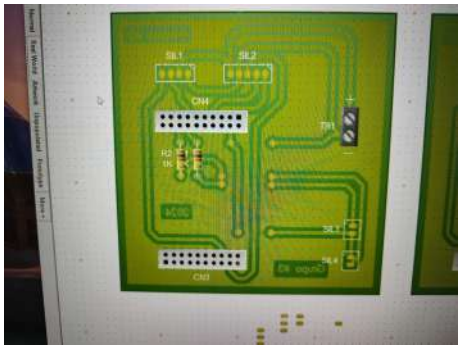

Figura 11: Circuito realizado en el programa PCBWizard

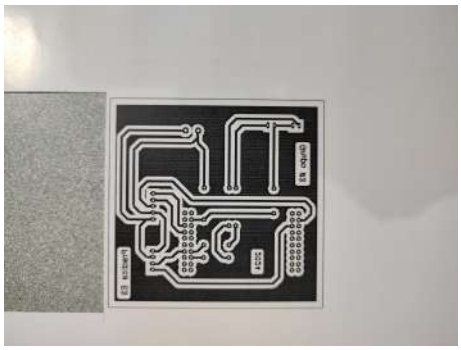

Figura 12: Impresión del circuito en papel couche

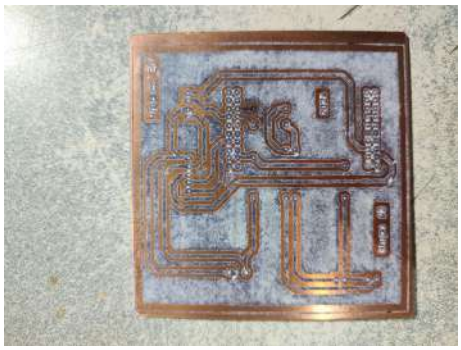

Figura 13: Circuito plasmado en la placa de cobre por medio de calor

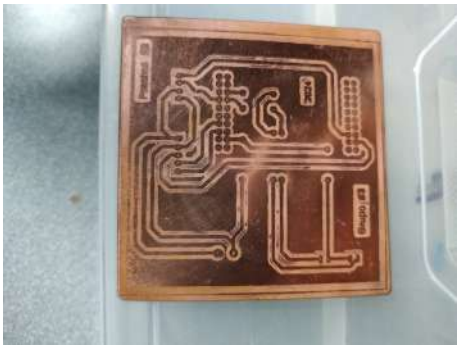

Figura 14: Circuito perforado y preparado para soldar los componentes

VII. DISCUSIÓN DE RESULTADOS

La implementación del código logra integrar eficazmente un reloj digital con alarma, utilizando el RTC DS1302 para la obtención del tiempo y un display TM1637 para su visualización. La configuración de los pines y la inicialización de los dispositivos se realizan adecuadamente, aunque faltan implementaciones específicas para la inicialización del RTC y la lectura del tiempo. Las alarmas funcionan correctamente, activándose en función del tiempo configurado, y la retroalimentación visual a través del LED y el buzzer se manejan de manera efectiva.

VIII. CONCLUSIONES

El sistema proporciona una base sólida para un reloj digital con alarma, mostrando un diseño modular mediante el uso de subrutinas. La correcta inicialización de componentes asegura un funcionamiento confiable. Sin embargo, se requiere la implementación de las subrutinas initRTC y readTime para lograr un funcionamiento completo del sistema. Con estas mejoras, se puede conseguir un reloj digital totalmente funcional, capaz de operar de manera autónoma y precisa, además de facilitar futuras ampliaciones en la funcionalidad.

IX. REPOSITORIO

`https://github.com/Henry-Chacon/Fase_2_LAB_E5`. [Fuente: Elaboración propia 2024].