PRÁCTICA N°1: TIMERS

Presentado por:
SEBASTIÁN TOBÓN ECHAVARRÍA
CRISTIAN LOPERA TRUJILLO
JOSE MANUEL MACHADO LOAIZA

PROYECTO INSTRUMENTAL II

Presentado a:
AUGUSTO CARMONA VALENCIA

INGENIERÍA FÍSICA
DEPARTAMENTO DE CIENCIAS FÍSICAS
ESCUELA DE CIENCIAS
UNIVERSIDAD EAFIT
MEDELLÍN
2020

CONTENIDO

LISTA DE FIGURAS

INTRODUCCIÓN

En la práctica, es común programar sistemas embebidos haciendo uso de funciones bloqueantes o que consumen toda la capacidad del sistema para llevar a cabo tareas sencillas. El ejemplo más común es el uso de la función *delay()* en Arduino para generar interrupciones o esperas dentro del programa. Naturalmente, el objetivo al programar un sistema de este tipo es aprovechar todas sus capacidades para llevar a cabo una tarea con la mayor eficiencia posible, por lo que se hace necesario buscar alternativas a estas funciones bloqueantes.

En el caso particular del manejo del tiempo, los microcontroladores cuentan con dispositivos denominados temporizadores o *timers*, los cuales permiten llevar un conteo de tiempo sin consumir toda la capacidad del sistema. Así, es indudable que los temporizadores son elementos fundamentales dentro del microcontrolador, por lo cual aprender a manejarlos es indispensable para poder programar procesos lógicos de la forma más eficiente.

Una aplicación básica de los temporizadores es dentro de las máquinas de estado finito (MEF), abstracciones computacionales que describen el comportamiento de un sistema reactivo mediante un número determinado de estados y un número determinado de transiciones entre dichos estados. Las MEF permiten realizar procesos bien definidos en un tiempo discreto, por lo cual requieren de un conteo de tiempo preciso. Aquí es donde entran en juego los temporizadores.

El objetivo de la siguiente práctica es desarrollar una máquina de estado finito para controlar un dispositivo con ingreso de información, más específicamente un horno microondas. Para ello, se hará uso de los temporizadores de un microcontrolador AVR ATmega2560 para manejar las transiciones propias del dispositivo.

1. OBJETIVOS

- Desarrollar una Máquina de Estados Finitos para el control de un dispositivo con ingreso de información.
- Comprender el funcionamiento de los timers de un microcontrolador y la forma en la que pueden optimizar un proceso lógico.
- Programar un sistema embebido tipo microcontrolador para simular el funcionamiento del dispositivo mediante el uso de los timers.
- Programar el funcionamiento del dispositivo empleando el software *Proteus* como sistema de prueba.

2. MARCO TEÓRICO

A la hora de programar los temporizadores de un microcontrolador, lo que se suele hacer es acceder a los registros del dispositivo y cambiarlos manualmente. Alternativamente, se pueden usar librerías que facilitan el control de los timers, como por ejemplo la librería *TimerOne* [1]. En esta práctica lo haremos modificando los registros del Arduino ATmega2560, más específicamente para el Timer 1. Los registros [2] para este temporizador se muestran en la **Figura 1**.

Figura 1. Registros para el timer 1 de un microcontrolador AVR.

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Para programar el timer como contador se debe colocar todos los bits del registro TCCR1A en cero. Esto en realidad no es necesario ya que el registro se inicializa automáticamente a 0, sin embargo es una buena práctica pues este registro será útil cuando se utilice el timer en modo comparación y para la modulación de ancho de pulso PWM.

Además, para programar el timer como contador se deben manipular los bits CS12, CS11 y CS10 del registro TCCR1B, la forma en que se configuren determina una reducción en la frecuencia de conteo, lo que se conoce como configuración de prescaler. En la **Figura 2** se muestran las distintas configuraciones con sus reducciones correspondientes.

Figura 2. Configuración de prescaler para timer 1.

CS12	CS11	CS10	Description
0	0	0	No clock source (timer/counter stopped)
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

En la práctica se busca que la interrupción (el cambio de estado) se genere por un desborde del temporizador, por lo tanto, se debe tener en cuenta el valor máximo que éste puede alcanzar. Para determinar la frecuencia de conteo, se utiliza la **Eq. 1**:

$$conteo = \frac{f_{CPU}}{P(V_{max} - V_i)} \tag{1}$$

donde f_{CPU} es la frecuencia de reloj del microcontrolador, P es el valor del prescaler, V_{max} es el valor máximo que alcanza el temporizador y V_i es su valor inicial.

De la **Eq. 1** se puede inferir que, para modificar el valor del conteo, existen dos posibilidades: cambiar el prescaler o cambiar el valor inicial del temporizador. La primera opción tiene la ventaja de que consume menos recursos, aunque está limitada por las configuraciones de prescaler disponibles.

De esta forma, se puede determinar cuántos conteos equivalen al paso de un segundo y configurar el código con base en este resultado.

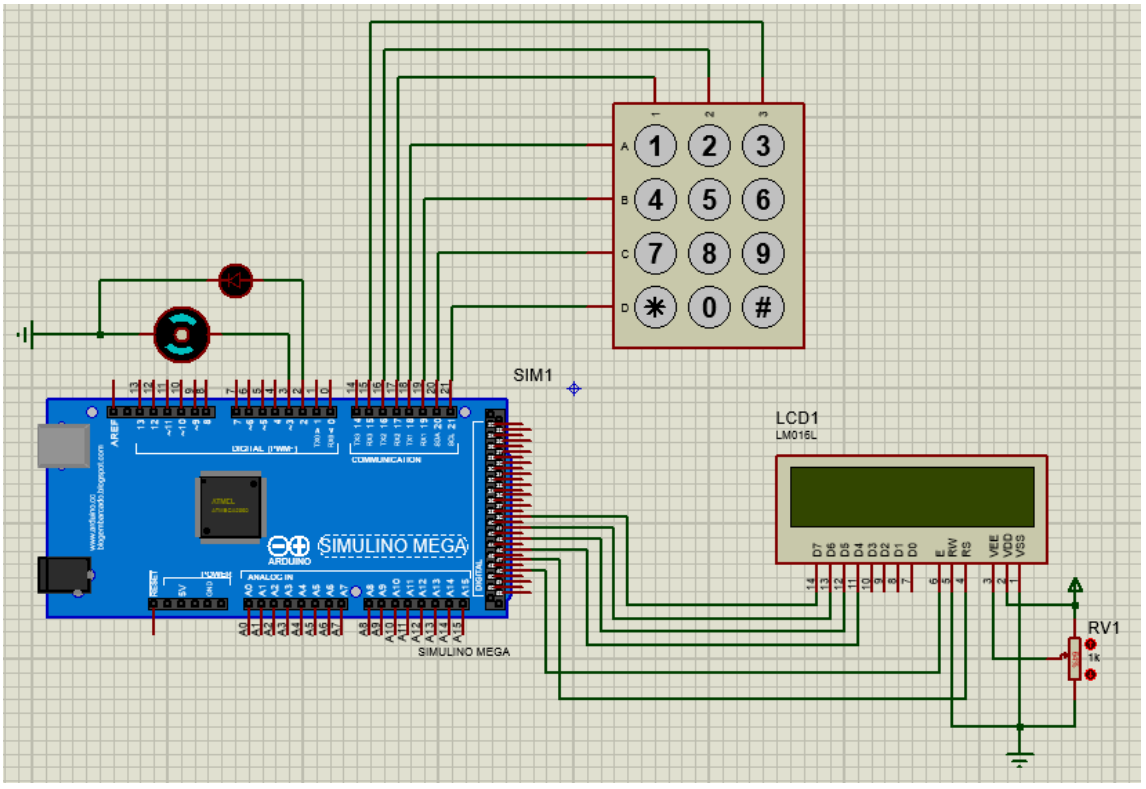
3. PROCEDIMIENTO

Los códigos del horno microondas a realizar e implementar usando el sistema Arduino deben cumplir con los siguientes requerimientos [3]:

- El tiempo deberá ingresarse vía teclado, mostrando dicho valor sobre el display LCD.
- El temporizador deberá usar alguno de los 5 timers disponibles en el ATMEGA2560 como elemento de conteo y el estado del temporizador siempre debe poderse visualizar en el display LCD.
- En el teclado deberán existir dos funciones adicionales: * para Aceptar, # para cancelar.
- Con el paso del temporizador, el motor deberá estar encendido, simulando el proceso de cocción. En el momento de terminar, el sistema deberá "titilar" el LED mostrando que ya se finalizó el proceso.

Para el desarrollo del laboratorio se realiza el montaje en *Proteus* como se muestra en la **Figura 3**.

Figura 3. Esquema de conexión.



Antes de comenzar con la programación, se debe realizar el diagrama de estados del autómata. Esto facilitará posteriormente la implementación del código, pues desarrolla claridad acerca de los estados del sistema y las transiciones entre ellos. A su vez, se debe determinar la frecuencia de conteo mediante la **Eq. 1** para poder configurar el timer dentro del programa.

Una vez terminada la programación, se obtiene el archivo *.hex* y se carga al Arduino de la **Figura 3**, para así poder simular el funcionamiento del programa. En vista de los resultados de la simulación, se pueden hacer las correcciones y retoques necesarios al código, permitiendo así la optimización y buen funcionamiento del dispositivo.

4. RESULTADOS Y ANÁLISIS

4.1. Programación y simulación Adjunto a este informe se encuentran los códigos para la realización del laboratorio en un repositorio en GitHub. Dichos códigos están debidamente comentados y resuelven las tareas del autómata descrito anteriormente. El repositorio cuenta con un vídeo de la simulación cuando el programa es ejecutado, evidenciando así la funcionalidad del dispositivo.

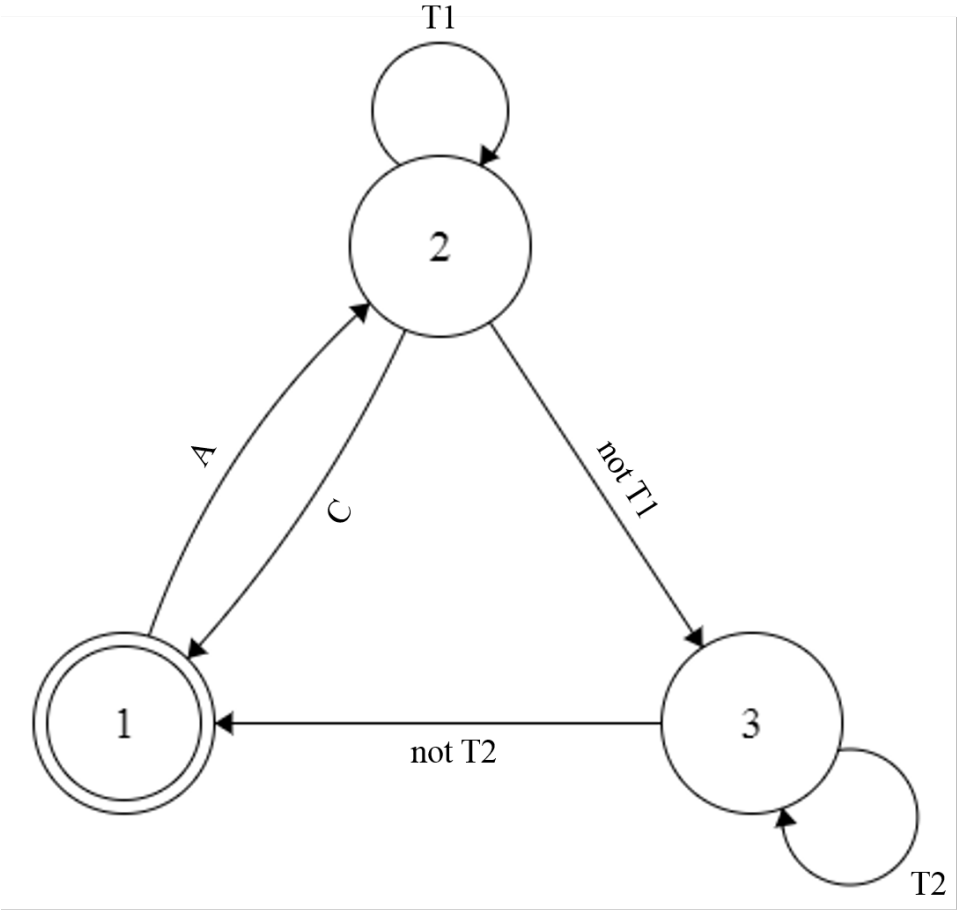
4.2. Diagrama de estados Para la realización del diagrama de estados, se utiliza la herramienta *Finite State Machine Designer* disponible en $https://www.cs.unc.edu/\ otternes/comp455/fsm_designer/$. El diagrama de estados resultante se muestra en la **Figura 4**.

Para lograr una mayor comprensión del diagrama mostrado con anterioridad, se propone una tabla explicativa (**Tabla 1**) para cada uno de los estados y transiciones presentes en el diagrama.

Tabla 1. Definición de estados del autómata.

Estado	**Transición**
1 : Estado inicial	A : Presiona la tecla *
2 : Proceso de cocción	T1 : conteo $<$ tiempo de cocción C : Presiona la tecla #
3 : Fin de la cocción	T2 : conteo $<$ 5 segundos

Figura 4. Diagrama de estados del autómata.



4.3. Configuración del temporizador La frecuencia de reloj del ATmega2560 es de 16 MHz, pero para efectos prácticos se toma la frecuencia de reloj como la mitad de este valor. Teniendo en cuenta que el timer 1 tiene 16 bits (valor máximo 65535), se tienen los datos necesarios para hacer el cálculo.

Con el objetivo de hacer cada ciclo de conteo equivalente a un segundo y tomando un prescaler de 1024, es posible hallar el valor inicial del temporizador por medio de la **Eq. 1**:

$$1 = \frac{8E6}{1024(65535 - V_i)}$$

De esta forma, $V_i \approx 57723$. Para configurar el prescaler de modo que equivalga 1024, según la **Figura 2**, los bits CS12 y CS10 se deben igualar a 1.

5. CONCLUSIONES

- Se logró desarrollar una MEF para controlar un dispositivo con ingreso de información, en este caso un horno microondas.
- Se logró implementar un código para simular el funcionamiento de una MEF haciendo uso de los temporizadores de un sistema embebido tipo microcontrolador.
- Se consiguió simular y evaluar el funcionamiento de una MEF mediante el software de simulación electrónica *Proteus*.

Referencias

[1] "Librería TimerOne En Arduino." https://creatividadcodificada.com/arduino/libreria-timerone-en-arduino/, 2019.

[2] "AVR Timer programming." https://exploreembedded.com/wiki/AVR-Timer-programming.

[3] D. d. C. F. Universidad EAFIT, Escuela de Ciencias, "Práctica de Laboratorio No.1: Temporizadores," 2020.