Práctica No. 2 Ensamblador, Interrupciones & Temporizadores

Departamento Académico de Sistemas Digitales
Instituto Tecnológico Autónomo de México
Primavera 2019

JEAN PAUL VIRUEÑA ITURRIAGA 155265
FABIÁN ORDUÑA FERREIRA 159001

Abstract. En esta práctica empleamos una placa de Arduino, para poder aprender a programar tareas básicas en un microcontrolador (AVR). Empleamos dos distintos lenguajes para programar el microcontrolador y comparamos las diferencias entre estos. Finalmente, empleamos interrupciones y temporizadores en el microcontrolador.

I. INTRODUCCIÓN

En la actualidad, la tecnología es un recurso que tenemos para optimizar nuestras actividades diarias. Las computadoras son un gran ejemplo de esto, mismas que pueden realizar diversas y muy complejas tareas. Sin embargo, hay ocasiones en las que se desean realizar tareas sencillas y concretas.

Para realizar tareas específicas y de una complejidad no tan alta, se emplean los sistemas embebidos. Estos sistemas pueden ser considerados como computadoras sencillas ya que cuentan con componentes básicas de las mismas. Los taxímetros, los sistemas de control de acceso, los sistemas de control de máquinas expendedoras, los sistemas GPS, son algunos ejemplos.

Por tal motivo, es que en esta práctica tenemos como objetivo aprender a programar microcontroladores ya que son la parte medular de los sistemas embebidos. Asimismo, buscamos conocer las diferencias entre los distintos tipos de programación que se pueden emplear en el microcontrolador, aprender a utilizar interrupciones y temporizadores en estos sistemas.

Este reporte está organizado en distintas secciones, comenzando con un marco teórico, en el que se presenta la información relevante y de importancia para una mejor comprensión de lo que aquí se aborda; seguido del desarrollo, en el que se presenta la manera en que se trabajó para la obtención de los resultados; la sección de resultados, en la que se presenta un análisis de lo obtenido y la sección de conclusiones.

II. MARCO TEÓRICO

Los sistemas embebidos son sistemas basados en microcontroladores o microprocesadores los cuales son diseñados para realizar una tarea en específico; es decir, para realizar operaciones específicas, comúnmente simples y en tiempo real. Los microcontroladores son circuitos integrados, programables, compuestos por diversos grupos funcionales. Son “computadoras pequeñas” ya que cuentan con las unidades funcionales básicas de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida [1] y [2].

Para programar los microcontroladores existen dos alternativas: los lenguajes de alto nivel y los de bajo nivel. Los lenguajes de alto nivel son aquellos que no contemplan, al crear los algoritmos, la capacidad que tienen las máquinas para ejecutar órdenes, sino las capacidades cognitivas de los seres humanos. Por el contrario, los lenguajes de bajo nivel son aquellos que ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan [3], [4] y [5].

Algunos de los recursos más importantes, durante la programación de los microcontroladores, debido a sus aplicaciones son las interrupciones y los temporizadores. Las interrupciones son señales que recibe para indicar que se debe interrumpir la ejecución en curso con la finalidad de ejecutar algo en específico y después reanudar el proceso interrumpido. Los temporizadores son registros que aumentan su valor en una unidad con cada

ciclo de reloj; son empleados para obtener medidas de tiempos muy precisas y generalmente se trabaja con ellos para contar, tanto con señales internas, como señales externas [6], [7] y [8].

III. DESARROLLO

Para cumplir con los objetivos de la práctica se realizaron se realizaron tres ejercicios que consistían en desarrollar un código e implementar un circuito con un Arduino Mega 2560. Los ejercicios se describen a continuación:

El primer ejercicio con fue escribir tres códigos en C en un primer momento para después escribirlos en ensamblador y poder compararlos. Los códigos que se programaron fueron: un código que al tener los coeficientes de una ecuación de segundo grado sea capaz de calcular su determinante, un código que al tener cinco valores sea capaz de calcular si promedio y un código que sea capaz de encender un led.

El segundo ejercicio consistía en hacer parpadear un LED a 2 Hz con ayuda de la función *delay* y seguido de esto se le conectó un botón que incrementara un contador si se presiona el botón. Después se reemplazó el botón por por un LED y un fotodiodo para finalmente asociar una interrupción al código.

El tercer ejercicio consistía de configurar temporizadores en modo normal para que corra a 2 Hz y en modo CTC a 4 Hz para hacer parpadear un LED. Para lograr eso se tuvo que calcular el pre escalamiento. Después de esto, se implementó un semáforo.

IV. RESULTADOS

V.

A pesar de tener que emplear tres sesiones de laboratoria y horas fuera del laboratorio, se obtuvieron resultados positivos ya que pudimos programar en ensamblador, en C y compararlos. Descubrimos que al pasar el código de C mediante un compilador a ensamblador, se genera mucha basura y lo que se programamos en ensamblador está compuesto de mucho menos líneas.

Además se pudo implementar con éxitos los circuitos y programar la interrupción y el timer con ayuda de las siguientes instrucciones donde se configuran los registros mediante el siguiente código:

```
void setup()
{
  Serial.begin(9600);
  cli();
  DDRD &= ~(1 << DDD1);
  PORTD |= (1 << PORTD1);
  EICRA |= (1 << ISC10);
  EIMSK |= (1 << INT1);
  sei();
  pinMode(pin10, INPUT);
  pinMode(led, OUTPUT);
}
```

Figura 1. Configuración de interrupciones, timer.

```
void setup() {
  cli();
  TCCR1B = 0; TCCR1A = 0;
  TCCR1B |= (1 << CS12);
  TCNT1 = 34286;
  //TCNT1 = 3036;
  TIMSK1 |= (1 << TOIE1);
  sei();

  pinMode(led, OUTPUT);

  }
```

Figura 2. Configuración de interrupciones, timer.

```
void setup(){
    cli();
    TCCR1B = 0; TCCR1A = 0;
    TCCR1B |= (1 << CS12);//habilitar temporizadores
    TCCR1B |= (1 << WGM12);
    OCR1AH = 0x3D; OCR1AL = 0x09;
    TIMSK1 |= (1 << OCIE1A);
    sei();
    pinMode(led, OUTPUT);
  }
```

Figura 3. Configuración de temporizador con preescalamiento.

Cabe destacar que al durante el segundo ejercicio al conectar el botón se tenía que esperar a que el LED termine de parpadear para que al presionar el botón el contador tenga que ser incrementado.

VI. CONCLUSIONES

Fabián Orduña Ferreira

Gracias a esta práctica aprendí la manera en que los microcontroladores funcionan; aprendí a programar funciones básicas; aprendí cuáles son

las diferencias y ventajas que existen entre los distintos lenguajes de programación que se pueden emplear para programarlo. También comprendí, de manera pragmática, la diferencia que existe en emplear distintos tipos de interrupciones y su importancia para desarrollo de soluciones, así como en el uso de los temporizadores.

Jean Paul Virueña Iturriaga
Fue muy gratificante poder aterrizar varios conocimientos mencionados en la teoría mediante esta práctica, como son el manejo de registros y el funcionamiento de los timers. Además es interesante ver la cantidad de código que fue se genera al compilar ensamblador que al programar directamente en este último, aunque sigo pensando en que la simplicidad de programar en un lenguaje de alto nivel es una gran ventaja. Es importante resaltar que a pesar de haber hecho esta práctica, sigo teniendo lagunas respecto al funcionamiento de los interruptores.

VII. ROLES

Para el desarrollo de las actividades, de las que mostramos previamente los resultados, ambos miembros del equipo colaboramos de forma conjunta. En lo que a este documento concierne, Fabián se enfocó en mayor proporción al abstract, a la introducción y al marco teórico, mientras que Paul se enfocó más al desarrollo y a los resultados.

VIII. FUENTES DE CONSULTA

[1] Tutorialspoint. Embedded Systems - Overview. Consultado el 20 de Febrero de 2019. https://www.tutorialspoint.com/embedded_systems/es_overview.htm

[2] IoT Agenda. Microcontroller. Consultado el 20 de Febrero de 2019. https://internetofthingsagenda.techtarget.com/definition/microcontroller

[3] Wikipedia. AVR. Consultado el 20 de Febrero de 2019. https://es.wikipedia.org/wiki/AVR

[4] DevCode. Arduino: Programación de microcontroladores. Consultado el 20 de Febrero de 2019. https://devcode.la/articulos/programacion-de-microcontroladores/

[5] Universidad Internacional de Valencia. Lenguaje de alto nivel, los más utilizados. Consultado el 20 de Febrero de 2019. https://www.universidadviu.com/lenguaje-alto-nivel-los-mas-utilizados/

[6] Wikipedia. Arduino. Consultado el 20 de Febrero de 2019. https://es.wikipedia.org/wiki/Arduino

[7] Wikipedia. Interrupciones. Consultado el 20 de Febrero de 2019. https://es.wikipedia.org/wiki/Interrupci%C3%B3n

[8] Ingeniería Mecafenix. Temporizador. Consultado el 20 de Febrero de 2019. https://www.ingmecafenix.com/electricidad-industrial/temporizador-tipos-temporizador/