Estudio comparativo entre los microcontroladores PIC16F887 y ATmega328P - pila, entradas y salidas, temporizador, interrupciones y memoria EEPROM

Julio Javier Schwendener Morales

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Estudio comparativo entre los microcontroladores PIC16F887 y ATmega328P - pila, entradas y salidas, temporizador, interrupciones y memoria EEPROM

Trabajo de graduación presentado por Julio Javier Schwendener Morales
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

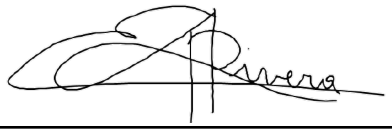
Estudio comparativo entre los microcontroladores PIC16F887 y ATmega328P - pila, entradas y salidas, temporizador, interrupciones y memoria EEPROM

Trabajo de graduación presentado por Julio Javier Schwendener Morales
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

(f) 

Dr. Luis Alberto Rivera Estrada

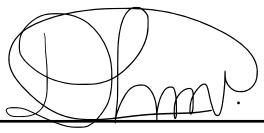
Tribunal Examinador:

(f) 

Ing. Kurt Emmanuel Kellner Juárez

(f) 

M. Sc. José Eduardo Morales Espinoza

(f) 

Ing. Diego Alberto Morales Ibáñez

Fecha de aprobación: Guatemala, 3 de enero de 2023.

Prefacio

Un camino comenzado hace 5 años está llegando a su final y quisiera tomar este espacio para agradecer a las instituciones y personas que han significado tanto para mí en todo este tiempo.

A mi familia, por su incondicional apoyo durante toda mi vida y, sobre todo, en tiempos difíciles en estos últimos años.

A la Fundación Juan Bautista Gutiérrez por la oportunidad de permitirme haber emprendido este camino y su continuo apoyo en estos 5 años.

Al Centro Universitario Ciudad Vieja por darme, más que un lugar para poder vivir en este tiempo, un hogar y una segunda casa.

A mis amigos, por todos los momentos que compartimos y compartiremos, esto solo comienza.

A mi asesor Luis Rivera, por su tiempo, consejos y constante retroalimentación y apoyo en este trabajo.

Y a todas las personas que, sin duda, han sumado para que esta experiencia haya sido aún mejor.

Índice

Lista de figuras

Lista de cuadros

Resumen

En la presente investigación, se comparó el desempeño de los microcontroladores PIC16F-887 y ATmega328P, haciendo énfasis en los módulos de entradas y salidas, pila, temporizador, interrupciones y memoria EEPROM. Para esto, se desarrollaron programas enfocados en el funcionamiento de cada uno de los módulos mencionados, basados en los programas propuestos por los laboratorios del curso de Programación de Microcontroladores de la Universidad del Valle de Guatemala. Los programas fueron implementados en ambos microcontroladores con los lenguajes C y ensamblador.

Para evaluar el desempeño de los microcontroladores, se hizo uso de los Analizadores Lógicos, los cuales fueron conectados a los microcontroladores mientras ejecutaban los programas.

Para el módulo de temporizadores, se obtuvo que el PIC16F887 tiene un rendimiento más estable, es decir, que los tiempos medidos varían en un rango menor de datos, y que su porcentaje de error entre el promedio y el valor teórico esperado, es menor.

Abstract

In the present investigation, the performance of the PIC16F887 and ATmega328P microcontrollers was compared, emphasizing the input and output, stack, timers, interrupts and EEPROM memory modules. For this, programs focused on the functionality of every sigle module were developed, based on the programs proposed by the laboratories of the Microcontroller Programming course of Universidad del Valle de Guatemala. The programs were implemented on both microcontrollers with the C and assembler languages.

To evaluate the performance of the microcontrollers, Logic Analyzers were used, wich were connected to the microcontrollers while they executed the programs.

For the timer module, it was obtained that the PIC16F887 has a more stable performance, that is, that the measured times vary in a smaller range of data, and that the percentage of error between the average time of the PIC and the theoretical value is lower.

CAPÍTULO 1

Introducción

La programación de microcontroladores es un área de sumo interés en la electrónica, debido a las diversas aplicaciones que tienen. Esto la hace un conocimiento esencial en las ingenierías Electrónica, Mecatrónica y Biomédica. Un método común de enseñanza de esta rama de la electrónica se da a través de los microcontroladores de 8 bits.

Habiendo una gran variedad de microcontroladores de 8 bits, es necesario, para las universidades y centros de estudio, seleccionar uno para impartir los conocimientos de esta área. Por este motivo, en esta investigación se comparan dos microcontroladores involucrados en la enseñanza de la electrónica en la Universidad del Valle de Guatemala.

La comparación realizada abarca los módulos de pila, entradas y salidas, temporizador, interrupciones y memoria EEPROM de los microcontroladores PIC16F887 y ATmega328P utilizando un Analizador Lógico. Los programas utilizados se elaboraron con base en los propuestos en el curso de Programación de Microcontroladores de la Universidad del Valle de Guatemala, enfocándolos al uso individual de los módulos antes mencionados.

En los siguientes capítulos, se da una breve descripción de los módulos, se describen y muestran pseudocódigos de los programas empleados para realizar las mediciones, se describe la forma en que los resultados serán evaluados y qué se compara en cada caso, se muestran los circuitos empleados para cada microcontrolador en cada módulo, y se discute sobre los resultados obtenidos para, posteriormente, determinar el microcontrolador con mejor rendimiento en cada módulo.

CAPÍTULO 2

Antecedentes

El PIC16F887 es el microcontrolador empleado en la enseñanza de la programación básica de microcontroladores, basado en el lenguaje de Assembler y C, en la Universidad del Valle de Guatemala [1]. El ATMega328P es el microcontrolador empleado por la placa de desarrollo Arduino UNO, facilitando la programación del microcontrolador con el software de Arduino. En algunos modelos de Arduino UNO, este microcontrolador puede ser desmontado y programado de forma independiente, tanto en Assembler como en C, funcionando como alternativa a la programación en el software de Arduino [2].

A continuación, se describen trabajos de comparación de microcontroladores similares.

2.1. Comparación de microcontroladores de 8 bits

Microchip es una empresa estadounidense dedicada a la manufactura de microcontroladores. Es la encargada del desarrollo y fabricación, entre otros, de los microcontroladores de la línea PIC16.

En 2002, Microchip realizó un trabajo de investigación, comparando distintos microcontroladores de 8 bits, tales como Intel 8051 y Motorola MC68HC05, con el fin de probar la superioridad de los microcontroladores de la línea PIC16C5. En la investigación realizada, se especifican las comparaciones con un enfoque técnico. Entre los aspectos comparados, Microchip realiza pruebas sobre el control de *loop*, el cual consiste en el tiempo de respuesta de los microcontroladores estudiados frente a un *loop* de un contador decreciente que, cuando llega a cero, se reinicia el contador. Para este experimento, el PIC de la línea 16C5 obtuvo el menor tiempo de todos los microcontroladores, entre 0.4 y 0.6 µs [3].

Como parte de las pruebas realizadas, se registró el tiempo de ejecución del módulo de temporizadores con un *delay* de 10 µs en cada uno de los microcontroladores estudiados. Para este experimento, se obtuvo que los PIC16C5 terminan la ejecución en un aproximado de 10.011 ms. Sin ser este el tiempo más bajo, Microchip indica que esto se debe a la precisión

con la que los PIC16C5 pueden trabajar al momento de implementar los temporizadores en otras aplicaciones, a diferencia del resto de microcontroladores.

2.2. Comparación entre las familias PIC y AVR

Fried [4], realizó un trabajo de investigación, publicado en 2012, comparando distintos aspectos sobre los microcontroladores de Microchip y Atmel, específicamente las familias PIC de Microchip y AVR de Atmel. Los modelos comparados en esta investigación fueron PIC12F629, PIC16F628, PIC18F452, ATtiny13, ATtiny2313 y ATmega32.

El trabajo presentado por Fried intenta emparejar los tres microcontroladores de la familia PIC con los de Atmel tomando en cuenta los pines que estos poseen. La investigación se enfoca en beneficio del usuario, por lo que los aspectos a comparar fueron, mayoritariamente, en torno a la facilidad de adquisición y programación de los microcontroladores

Fried comienza exponiendo los precios de compra de los distintos microcontroladores y, tomando en cuenta el emparejamiento inicial, indica que los costos son similares y no declara una ventaja en este aspecto para ninguna de las familias. El siguiente aspecto comparado son los lenguajes de programación. Fried, compara la programación tanto en lenguaje C como en *Assembler*. En el entorno de *Assembler*, Fried indica que los microcontroladores de Atmel, al no tener bancos de memoria y utilizar un formato más amigable con los usuarios, tienen una clara ventaja en este aspecto, mientras que en los entornos de lenguaje C, atribuye la ventaja a Atmel debido a las optimizaciones que permite el compilador, además de la excelente compatibilidad entre compiladores, a diferencia de los microcontroladores PIC, los cuales no son 100 % compatibles con otros compiladores.

2.3. Microcontroladores en otras universidades

En otras universidades con cursos de electrónica y programación de microcontroladores, se pueden indentificar los distintos microcontroladores utilizados en estas tareas. En la India, en Muzaffarpur Institute of Technology [5], para el curso de microcontroladores, utilizan microcontroladores de la familia PIC y el microcontrolador Intel 8051. Este curso abarca la arquitectura de los microcontroladores, programación en entorno *Assembler*, manejo de temporizadores e iterrupciones, y comunicaciones seriales.

En Europa, universidades como la Universidad de Génova (UniGe) [6], utilizan los microcontroladores ARM Cortex-M4. En el curso de programación de microcontroladores se enfocan en el aprendizaje en el lenguaje C, manejo de procesos en paralelo, tipos de datos, arquitectura y las instrucciones del microcontrolador, comunicaciones seriales y temporizadores.

En Estados Unidos, la universidad Northwestern [7], en Illinois, Chicago utilizan microcontroladores proporcionados por Texas Instruments, específicamente el MSP430FR5994. Su aprendizaje de programación de microcontroladores comienza con el estudio de la arquitectura del microcontrolador. La programación abarca el manejo de temporizadores, comu-

nicaciones seriales, conversiones analógico a digital e implementación de PMWs.

2.4. Microcontroladores en la Universidad del Valle de Guatemala

En la Universidad del Valle de Guatemala, la programación de microcontroladores para las carreras de Ingeniería Electrónica, Mecatrónica y Biomédica, comienza en el curso Introducción a la Ingeniería Electrónica y Mecatrónica [8], donde se tiene un primer acercamiento a la programación básica de microcontroladores, por medio del software de Arduino.

El segundo acercamiento que se tiene a los microcontroladores se da en el curso Programación de Microcontroladores. El alcance del curso abarca el aprendizaje en entorno ensamblador y el manejo de los bancos de memoria, temporizadores, lectura de entradas digitales y analógicas, comunicaciones seriales, y la implementación de procesos en paralelo en el PIC16F887 [1].

Finalmente, en el curso Electrónica Digital 2, se complementa el aprendizaje del microcontrolador PIC16F887 empleando el lenguaje C, que abarca las comunicaciones seriales (*UART*, *SPI* e *I2C*) y el paso de información a otros sistemas, tales como la ESP32 y TivaC [9].

CAPÍTULO 3

Justificación

El uso de microcontroladores es una parte fundamental de la electrónica, ya que su uso está implícito en una gran variedad de dispositivos electrónicos. La programación de los microcontroladores es, entonces, un campo de gran interés para las ingenierías Electrónica, Mecatrónica y Biomédica.

Actualmente, en la Universidad del Valle de Guatemala [1], la programación de microcontroladores se enseña con el PIC16F887, de Microchip, comenzando con en el entorno de ensamblador para, posteriormente, pasar al lenguaje C en cursos más avanzados. Esta tarea abarca el aprendizaje del funcionamiento básico del microcontrolador: configuración de los pines, tanto entradas como salidas; entradas y salidas analógicas; temporizadores; interrupciones; y comunicaciones seriales.

Por este trabajo, se pretende evaluar la eficiencia del microcontrolador utilizado en la Universidad del Valle de Guatemala (PIC16F887) y el ATmega328P con el fin de determinar, cuál de los dos es una mejor opción para el aprendizaje de programación de microcontroladores.

Es importante mencionar que, a futuro, los resultados de este trabajo podrían influenciar en la forma de enseñanza de los cursos de programación de microcontroladores en la Universidad del Valle de Guatemala, lo cual significaría mantener una continuidad respecto a los lenguajes de programación y plataformas empleadas durante el transcurso de las carreras de Ingeniería Electrónica, Ingeniería Mecatrónica e Ingeniería Biomédica.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Realizar un estudio comparativo entre los microcontroladores PIC16F887 y ATmega-328P, de los módulos de pila, entradas y salidas, temporizador, interrupciones y memoria EEPROM, y evaluar los parámetros necesarios para determinar la mejor opción para la enseñanza de microcontroladores en la Universidad del Valle de Guatemala.

4.2. Objetivos específicos

- Comparar el desempeño de cada microcontrolador con el uso de los analizadores lógicos disponibles en la Universidad del Valle de Guatemala, evaluando los módulos de pila, entradas y salidas, temporizador, interrupciones y memoria EEPROM.
- Realizar los laboratorios y proyectos propuestos en el curso de Programación de Microcontroladores en ambas plataformas relacionados a los módulos a evaluar.

CAPÍTULO 5

Alcance

Este trabajo se enfoca en comparar cinco de los módulos disponibles en los microcontroladores PIC16F887 y ATmega328P para definir cuál es más eficiente y adecuado para la enseñanza de programación de microcontroladores en la Universidad del Valle de Guatemala. Los módulos estudiados son pila, entradas y salidas, temporizadores, interrupciones y memoria EEPROM, en los lenguajes C y ensamblador.

Es necesario mencionar que los módulos evaluados en este trabajo no son todos los módulos disponibles en los microcontroladores estudiados y tampoco son todos los módulos utilizados en el curso de Programación de Microcontroladores de la Universidad del Valle de Guatemala [1]. El resto de módulos usados en el curso se abarcan en otro estudio comparativo, por lo que están fuera del alcance del presente trabajo.

El método de evaluación de los módulos abarcó el uso de analizadores lógicos para lograr una medición precisa del tiempo de ejecución de los programas planteados en ambos microcontroladores y en ambos lenguajes de programación.

CAPÍTULO 6

Marco teórico

6.1. Generalidades de un microcontrolador

Según Bettina [10], se puede resumir un microcontrolador como un procesador reducido, que posee memoria, temporizadores, pines de entrada y salida (*I/O pins*), y otros periféricos; todas estas características integradas en un espacio pequeño, optimizando costos de manufactura y tiempos de desarrollo.

Los microcontroladores son empleados en diversas aplicaciones de control, monitoreo y procesamiento de sistemas. También son empleados en dispositivos de uso común, como teléfonos celulares, máquinas de lavado, microondas y vehículos [11].

Generalmente, los microcontroladores incluyen un procesador central, puertos de entrada y salida, memoria para programar y almacenamiento de datos, un reloj interno (en inglés *internal clock*) y una gran variedad de periféricos [11]. Entre los periféricos más comunes se encuentran los temporizadores (*timers*), contadores, conversiones analógico a digital (por sus siglas en inglés *ADC*) y comunicaciones seriales [10].

6.2. Pila

La pila (más conocida como *stack*), es un área de la memoria RAM de los microcontroladores que permite el almacenamiento de los parámetros de variables, también es la responsable de recordar el orden en que las funciones son llamadas para poder regresar de forma correcta [11].

6.3. Entradas y salidas

Estas son todos los pines con capacidad de recibir o emitir un dato digital, es decir un voltaje *HIGH* o *LOW*, entre el rango definido. Estos voltajes son traducidos por el microcontrolador como un valor binario independientemente del valor del voltaje que se esté recibiendo, tomando en cuenta las especificaciones del fabricante de mínimos y máximos admitidos y las zonas en las que se interpretan como 1 o 0 lógicos [11].

6.4. Interrupciones

Una interrupción es una señal asíncrona, que funciona como mecanismo para evitar pérdidas del tiempo de procesador de los microcontroladores [11]. Las interrupciones pueden ser activadas tanto por banderas internas del microcontrolador como por señales externas de algún origen previamente especificado. Se pueden resumir como instrucciones que serán realizadas en el momento exacto en que alguna señal es recibida por el microcontrolador.

6.5. Temporizadores

Los temporizadores (conocidos en inglés como *timers*), son contadores, los cuales, al alcanzar un número previamente determinado, activan interrupciones en el programa. Pueden tener funciones extras como la captura y comparación de datos en tiempo, así como pre-escalados [11].

6.6. Memoria EEPROM

Por sus siglas en inglés *Electrically-Erasable Programmable Read-Only Memory*, es utilizada como un almacenamiento no volátil, diferenciándose de la memoria RAM y permitiendo el almacenamiento de datos aún cuando la energía es removida [11].

6.7. PIC16F887

El PIC16F887 [12], es un microcontrolador de 8 bits y 40 pines desarrollado por Microchip. Posee 14 canales de conversiones analógico a digital de 10 bits, tiene una memoria EEPROM programable de 256 bytes, 2 salidas PWM, comunicaciones SPI, UART e I2C, 3 temporizadores, comparadores, 35 pines de entrada y salida de propósito general y opera a una frecuencia máxima de 20MHz. Los microcontroladores PIC son generalmente utilizados en una gran variedad de sistemas embebidos. La asignación de pines del PIC16F887 se muestra en la Figura 1:

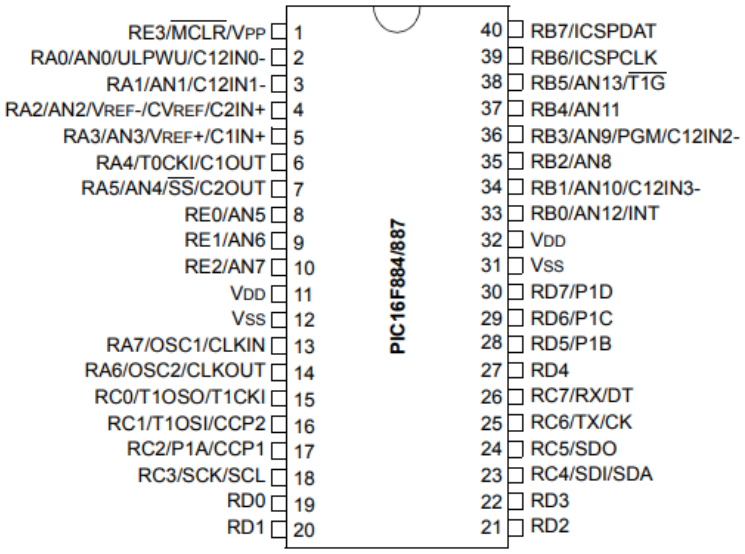


Figura 1: Asignación de pines del PIC16F887 [13].

6.8. ATmega328P

El ATmega328P [14], es un microcontrolador de 8 bits y 28 pines, diseñado originalmente por la extinta Atmel y fabricado actualmente por Microchip. Posee de 6 a 8 canales de conversiones analógico a digital de 10 bits, pines de entrada y salida de propósito general, comunicaciones I2C, SPI y UART, y 3 temporizadores con modos de comparación. Los ATmega328P son conocidos por ser los microcontroladores empleados en la plataforma Arduino debido a su alto rendimiento y bajo consumo [15]. La asignación de pines del ATmega328P se muestra en la Figura 2:

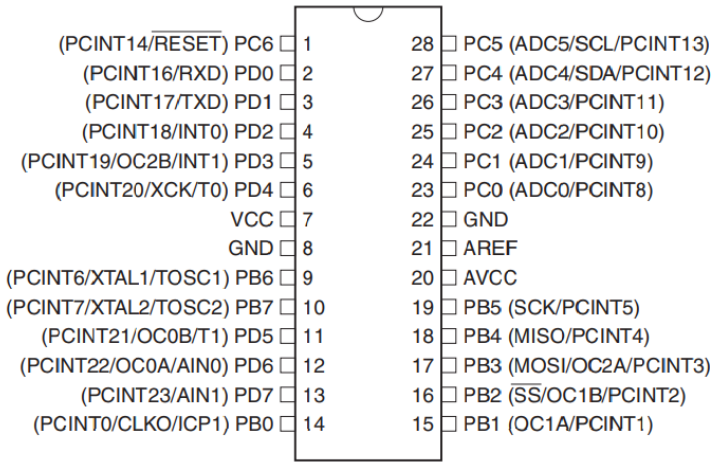


Figura 2: Asignación de pines del ATMega328P [2].

6.9. Lenguajes de programación

La programación es el método utilizado para hacer llegar las instrucciones deseadas al microcontrolador. Generalmente, los microcontroladores son programados en lenguajes de alto nivel, como C++ [16] o Java [17]. Sin embargo, los microcontroladores entienden únicamente el código de máquinas, por lo que es necesario traducir el código creado en un lenguaje de alto nivel, al lenguaje de máquinas que el microcontrolador puede entender y ejecutar [10].

6.10. Ensamblador

Escribir un código directamente en lenguaje binario puede ser complicado e ineficiente. Para solucionar este problema, se utiliza el lenguaje ensamblador, el cual asigna nombres a cada uno de los comandos que posee el microcontrolador, permitiendo a los usuarios memorizar palabras en lugar de secuencias binarias [10]. Sin embargo, estos comandos no son comprendidos por los microcontroladores, por lo que es necesario traducir los comandos a binario por medio de un compilador. El proceso de compilación para los códigos de ensamblador se muestra en la Figura 3:

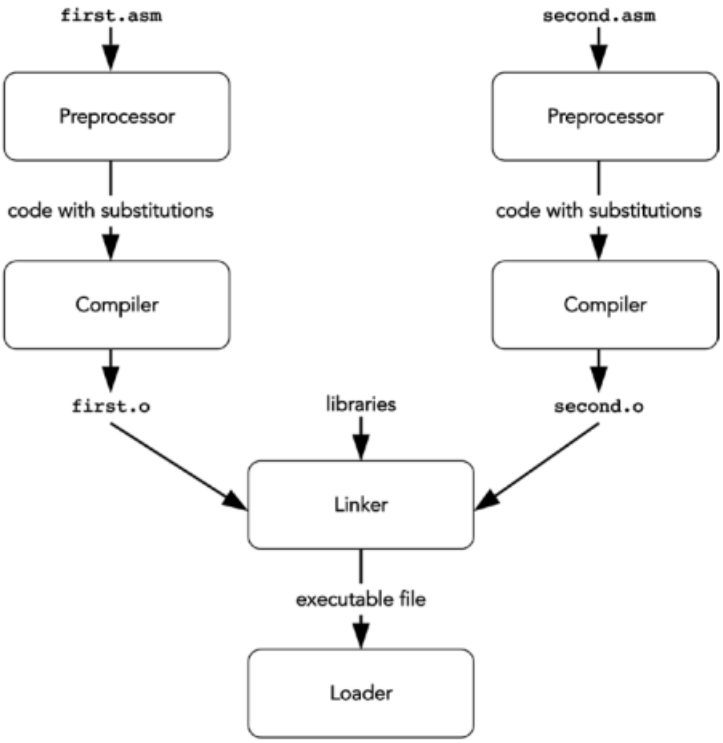


Figura 3: Compilación para códigos en ensamblador [18].

6.11. Compiladores

Una vez el código está completo, este debe ser cargado en el microcontrolador. La función de un compilador es lograr esta tarea, traduciendo el lenguaje de programación de alto nivel a lenguaje de máquinas para poder cargarlo. Un compilador es, pues, una herramienta de software que toma un código de un nivel más alto y lo optimiza para ensamblador (conocido también como *Assembler*) [17].

6.12. Analizador lógico

Los analizadores lógicos (*Logic Analyzers*), son dispositivos de visualización, capaces de obtener datos analógicos y digitales, con conexión USB a las computadoras [19]. Los analizadores lógicos son utilizados comúnmente para verificar el funcionamiento y encontrar errores en los sistemas embebidos o en programas ejecutados por microcontroladores. El dispositivo es capaz de grabar la información recopilada y mostrarla al usuario mediante una interfaz gráfica en la que se pueden realizar mediciones para, posteriormente, ser analizada y comparada con los datos esperados por el usuario.

6.12.1. Capturas

Los resultados obtenidos por los analizadores lógicos se entregan como capturas en el software *Logic*. Las capturas proporcionan los datos de las señales de entrada en las puntas que posee el analizador lógico [19].

CAPÍTULO 7

Módulo de pila

Como se explicó anteriormente, la Pila es el módulo de los microcontroladores encargado de mantener el orden en las funciones programadas que se emplean en los programas.

7.1. Programa

Para la evaluación de este módulo se implementó un programa que consiste en el encendido y apagado de dos luces LED dentro de funciones definidas. Se realizaron dos variantes de este programa: la primera variante hace uso de una única función para cambiar el estado de las luces LED, mientras que la segunda variante hace uso de funciones anidadas:. El número de funciones anidadas utilizadas está dado en las especificaciones de la hoja de datos del PIC16F887 [12], donde se indica que tiene una capacidad de almacenamiento de pila de hasta 8 niveles. El ATmega328P, por otro lado, tiene una capacidad de almacenamiento de pila tan grande como sea la memoria RAM disponible [14], por lo que, debido a estas características, se optó por utilizar el máximo del PIC, es decir, 8 niveles.

A continuación, en la Figura 4, se muestra el pseudocódigo del programa implementado:

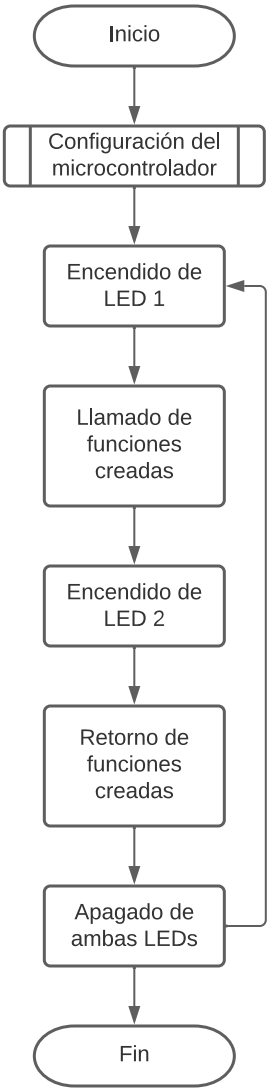


Figura 4: Pseudocódigo del programa implementado para el módulo de pila.

Para medir los resultados, se utilizó el Analizador Lógico (*Logic Analyzer*) con 4 millones de mediciones por segundo, durante un segundo, con puntas en las dos LEDs colocadas como indicadores en cada microcontrolador.

7.2. Comandos del módulo

En ensamblador, se utilizan funciones de llamado y retorno para ejecutar correctamente las funciones; en ambos microcontroladores, el comando referente al llamado de funciones es *CALL*, mientras que el comando de retorno de funciones es distinto, siendo *RETURN* en el PIC y *RET* en el ATmega. En C, las funciones son definidas con el tipo vacío (*void*)

7.3. Esquemáticos

Los circuitos implementados se muestran en las figuras 5 y 6:

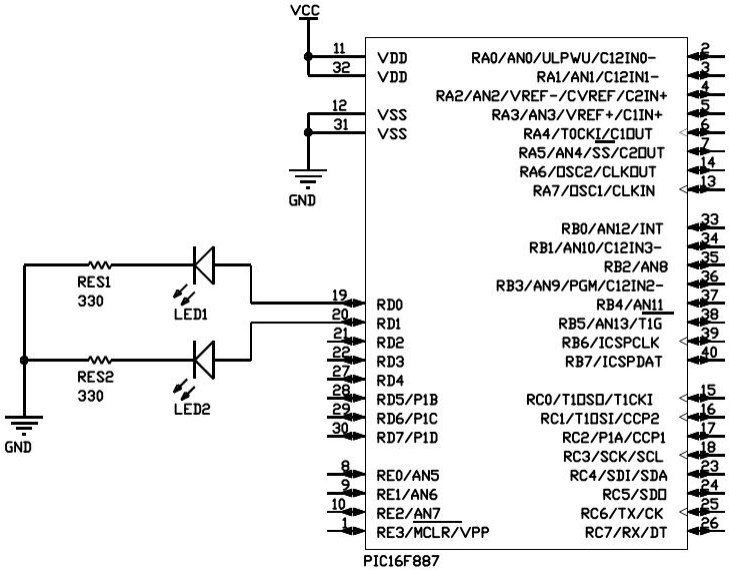


Figura 5: Circuito para la evaluación del módulo de pila en el PIC.

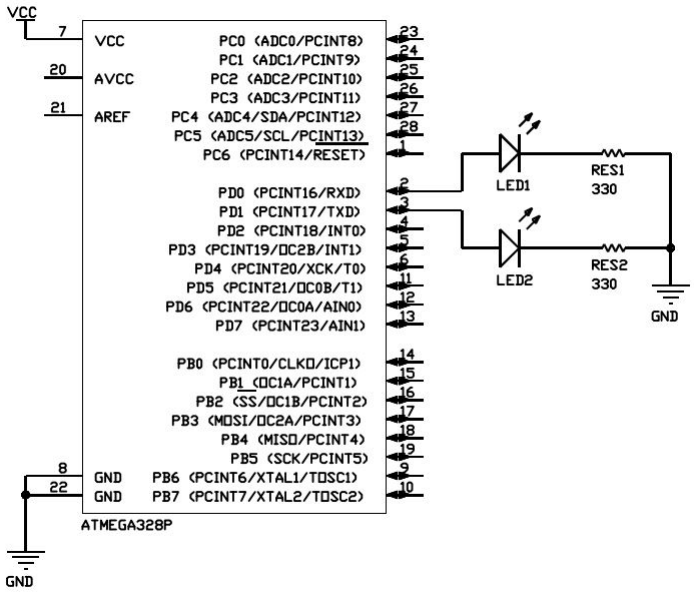


Figura 6: Circuito para la evaluación del módulo de pila en el ATmega.

7.4. Resultados

Anteriormente se mencionó que se realizaron dos variantes del programa de evaluación del módulo, basados en la capacidad máxima de Pila del PIC. Como se muestra en las figuras 4, 5 y 6, se colocaron dos LEDs en pines específicos de ambos microcontroladores, estas LEDs fueron conectadas al Analizador Lógico para obtener capturas similares a la mostrada en la Figura 7 (para más datos, referirse al anexo 16.1):

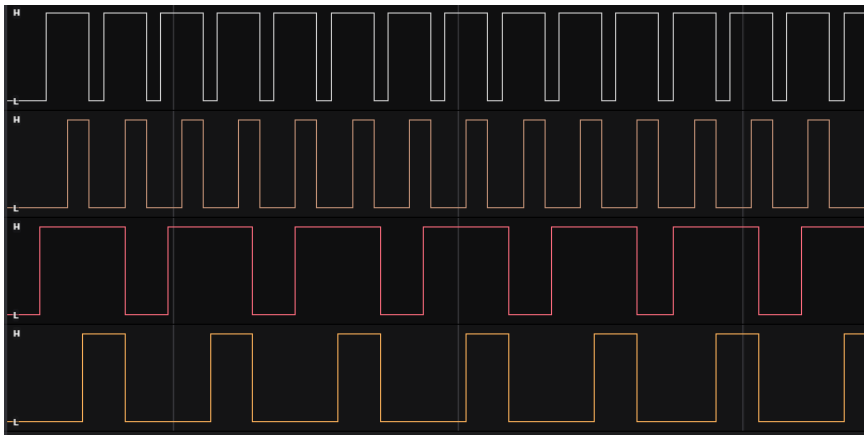


Figura 7: Segmento de captura de medición de la pila.

De las capturas, se obtuvo la moda en las mediciones para los siguientes casos:

1. Pila simple en ensamblador.
2. Pila simple en C.
3. Pila anidada en ensamblador.
4. Pila anidada en C.

A continuación, se muestran los resultados obtenidos en cada caso:

7.4.1. Pila simple en ensamblador

En el Cuadro 1, se muestran los resultados obtenidos para la modalidad simple en ensamblador, donde la LED 1 representa la LED que se enciende al inicio de la secuencia mostrada en la Figura 4, y la LED 2 representa la LED que se enciende al final de la secuencia:

Microcontrolador	Período (μs)	LED 1 encendida (μs)	LED 2 encendida (μs)
PIC16F887	4.5	3	1.5
ATMega328P	2	1.5	0.75

Cuadro 1: Resultados obtenidos del módulo de pila en modalidad simple en ensamblador.

De los resultados obtenidos en el Cuadro 1, se puede observar que, en el PIC, la ejecución total del programa duró un total de 4.5 microsegundos, con la LED 1 encendida por 3 microsegundos y la LED 2 por 1.5 microsegundos. Por otro lado, el ATMega obtuvo una duración total de 2 microsegundos, con la LED 1 encendida por 1.5 microsegundos y la LED 2 por 0.75 microsegundos. De esto, es posible obtener los datos mostrados en el Cuadro 2:

Microcontrolador	Llamado de funciones (μs)	Retorno de funciones (μs)
PIC16F887	1.5	1.5
ATMega328P	0.75	0.75

Cuadro 2: Tiempo empleado para el llamado y retorno de funciones en ambos microcontroladores en modalidad simple en ensamblador.

Los datos mostrados en el Cuadro 2 representan el tiempo que requieren los microcontroladores para llamar y retornar de las funciones empleadas en el programa. Se puede observar que el PIC emplea 1.5 microsegundos tanto para el llamado de funciones como para el retorno, mientras que el ATMega únicamente emplea 0.75 microsegundos para ambas acciones.

7.4.2. Pila simple en C

En el Cuadro 3, se muestran los resultados obtenidos para la modalidad simple en C, donde la LED 1 representa la LED que se enciende al inicio de la secuencia mostrada en la Figura 4, y la LED 2 representa la LED que se enciende al final de la secuencia:

Microcontrolador	Período (μs)	LED 1 encendida (μs)	LED 2 encendida (μs)
PIC16F887	10.5	7.5	3.5
ATMega328P	2.25	1.5	0.75

Cuadro 3: Resultados obtenidos del módulo de pila en modalidad simple en C.

De los resultados obtenidos en el Cuadro 3, se puede observar que, en el PIC, la ejecución total del programa duró un total de 10.5 microsegundos, con la LED 1 encendida por 7.5 microsegundos y la LED 2 por 3.5 microsegundos. Por otro lado, el ATMega obtuvo una duración total de 2.25 microsegundos, con la LED 1 encendida por 1.5 microsegundos y la LED 2 por 0.75 microsegundos. De esto, es posible obtener los datos mostrados en el Cuadro 4:

Microcontrolador	Llamado de funciones (μs)	Retorno de funciones (μs)
PIC16F887	4	3.5
ATMega328P	0.75	0.75

Cuadro 4: Tiempo empleado para el llamado y retorno de funciones en ambos microcontroladores en modalidad simple en C.

Se puede observar, en el Cuadro 4, que el PIC emplea 4 microsegundos para el llamado

de funciones y 3.5 microsegundos para el retorno, mientras que el ATMega emplea 0.75 microsegundos para ambas acciones.

7.4.3. Pila anidada en ensamblador

En el Cuadro 5, se muestran los resultados obtenidos para la modalidad anidada en ensamblador, donde la LED 1 representa la LED que se enciende al inicio de la secuencia mostrada en la Figura 4, y la LED 2 representa la LED que se enciende al final de la secuencia:

Microcontrolador	Período (μs)	LED 1 encendida (μs)	LED 2 encendida (μs)
PIC16F887	22.5	21	8.5
ATMega328P	10	9.5	4.25

Cuadro 5: Resultados obtenidos del módulo de pila en modalidad anidada en ensamblador.

De los resultados obtenidos en el Cuadro 5, se puede observar que, en el PIC, la ejecución total del programa duró un total de 22.5 microsegundos, con la LED 1 encendida por 21 microsegundos y la LED 2 por 8.5 microsegundos. Por otro lado, el ATMega obtuvo una duración total de 10 microsegundos, con la LED 1 encendida por 9.5 microsegundos y la LED 2 por 4.25 microsegundos. De esto, es posible obtener los datos mostrados en el Cuadro 6:

Microcontrolador	Llamado de funciones (μs)	Retorno de funciones (μs)
PIC16F887	12.5	8.5
ATMega328P	5.25	4.25

Cuadro 6: Tiempo empleado para el llamado y retorno de funciones en ambos microcontroladores en modalidad anidada en ensamblador.

En el Cuadro 6, se puede observar que el PIC emplea 12.5 microsegundos para el llamado de funciones y 8.5 microsegundos para el retorno, mientras que el ATMega emplea 5.25 microsegundos para el llamado de funciones y 4.25 para el retorno.

7.4.4. Pila anidada en C

En el Cuadro 7, se muestran los resultados obtenidos para la modalidad anidada en C, donde la LED 1 representa la LED que se enciende al inicio de la secuencia mostrada en la Figura 4, y la LED 2 representa la LED que se enciende al final de la secuencia:

Microcontrolador	Período (μs)	LED 1 encendida (μs)	LED 2 encendida (μs)
PIC16F887	41	38.5	17.5
ATMega328P	10	9.5	4

Cuadro 7: Resultados obtenidos del módulo de pila en modalidad anidada en C.

De los resultados obtenidos en el Cuadro 7, se puede observar que, en el PIC, la ejecución total del programa duró un total de 41 microsegundos, con la LED 1 encendida por 38.5 microsegundos y la LED 2 por 17.5 microsegundos. Por otro lado, el ATMega obtuvo una duración total de 10 microsegundos, con la LED 1 encendida por 9.5 microsegundos y la LED 2 por 4 microsegundos. De esto, es posible obtener los datos mostrados en el Cuadro 4:

Microcontrolador	Llamado de funciones (μs)	Retorno de funciones (μs)
PIC16F887	21	17.5
ATMega328P	5.5	4

Cuadro 8: Tiempo empleado para el llamado y retorno de funciones en ambos microcontroladores en modalidad anidada en C.

Se puede observar, en el Cuadro 8, que el PIC emplea 21 microsegundos para el llamado de funciones y 17.5 microsegundos para el retorno, mientras que el ATMega emplea 5.5 microsegundos para el llamado de funciones y 4 microsegundos para el retorno.

7.5. Discusión

Como se puede observar en los cuadros del 1 al 8, en todos los casos, el ATMega requiere de una menor cantidad de tiempo para ejecutar el programa de la Figura 4 que el PIC, se observa, también, que el tiempo en que las LEDs permanecen encendidas y, por ende, el tiempo requerido para ejecutar el llamado y retorno de funciones, es menor en el ATMega que en el PIC. Finalmente, se observa que el aumento de tiempo de ejecución del programa al pasar de ensamblador a C es considerablemente superior en el PIC, aumentando un 133 % en la modalidad simple y un 82 % en modalidad anidada, mientras que los aumentos en el ATMega son del 13 % y 0 % respectivamente.

CAPÍTULO 8

Módulo de entradas y salidas

Se ha explicado, anteriormente, que el módulo de entradas y salidas de los microcontroladores consiste en todos los pines que posee un microcontrolador, por medio de los cuales puede entrar o salir información específica.

En el PIC [13], existe un total de 36 pines distribuidos en 5 puertos. Los puertos del A al D tienen 8 pines cada uno y el puerto E tiene solamente 4 pines. Por otro lado, el ATMega [2], tiene únicamente 23 pines distribuidos en 3 puertos. Los puertos B y D tienen 8 pines y el puerto C tiene 7.

8.1. Programa

El programa utilizado para la evaluación de este módulo consiste en la implementación de 2 contadores binarios de 4 bits, capaces de incrementar y decrementar su valor con la pulsación de botones para, finalmente, sumar ambos contadores en un tercer contador de 4 bits con un quinto bit como acarreador, por lo que también es implementado un botón que permite realizar esta operación, tal como se puede observar en las figuras 9 y 10.

Debido a que el ATMega es físicamente más pequeño que el PIC y, como se mencionó anteriormente, tiene menos pines de entrada y salida, se optó por implementar los dos contadores binarios de 4 bits en un único puerto de 8 bits en ambos microcontroladores, debido a las limitantes físicas del ATMega [2].

A continuación, en la Figura 8, se muestra el pseudocódigo del programa utilizado en este módulo:

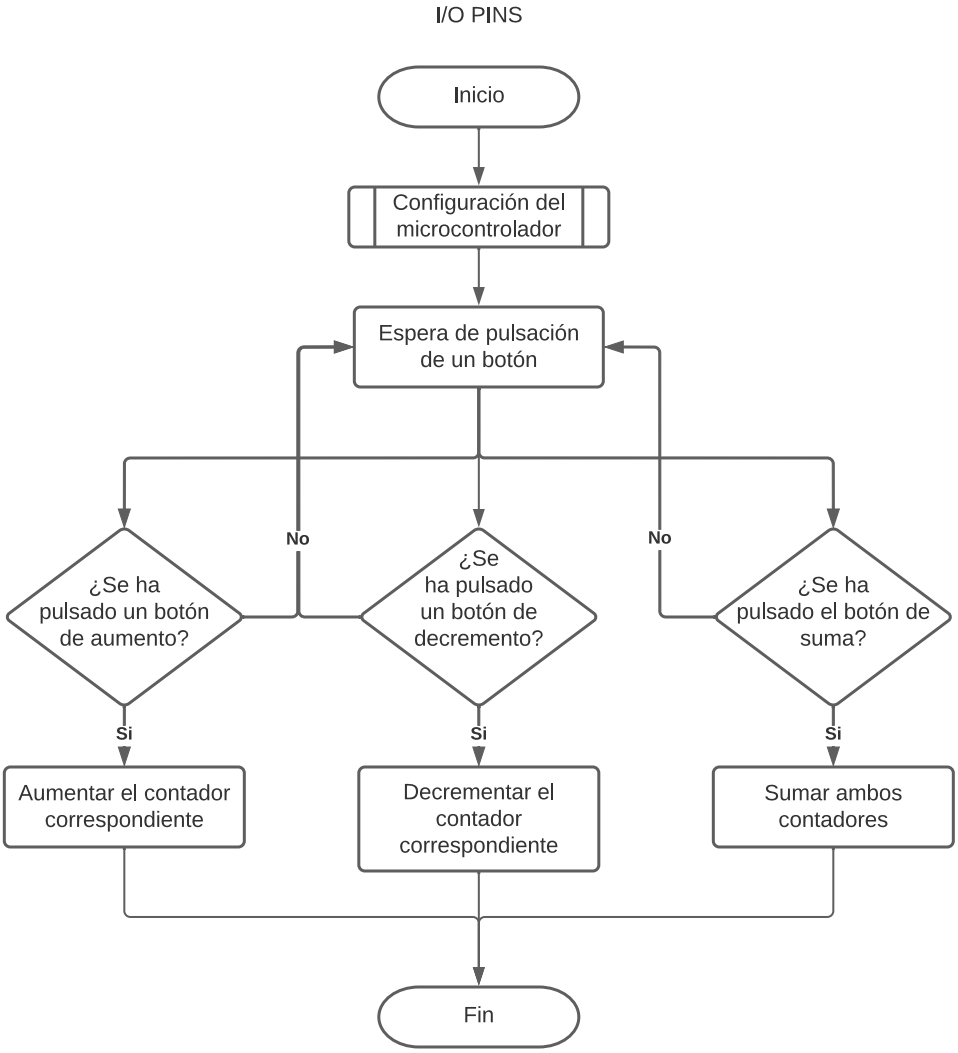


Figura 8: Pseudocódigo del programa implementado para el módulo de entradas y salidas.

8.2. Registros del módulo

A continuación, en el Cuadro 9, se muestran los registros requeridos para configurar los pines de entrada y salida en ambos microcontroladores:

Microcontrolador	Comando
PIC16F887	TRISx PORTx
ATmega328P	DDRx PINx PORTx

Cuadro 9: Registros de los pines de entrada y salida.

En el Cuadro 9, la x hace referencia al puerto específico que se está configurando, es decir, en el PIC, es reemplazada por A, B, C, D o E; mientras que en el ATMega es reemplazada por B, C y D. También se puede observar que el PIC requiere de 2 tipos de registros para configurar la totalidad de los pines de entrada y salida: el registro *TRIS* permite indicar si los pines de un puerto en específico son de entrada o salida, mientras que *PORT* permite leer o escribir valores lógicos (1 o 0) en pines específicos. Por otro lado, el ATMega requiere de 3 registros para configurar sus pines: el registro *DDR* permite indicar si los pines de un puerto son de entrada o salida; el registro *PIN* permite la lectura de pines en caso hayan sido configurados como entradas; y el registro *PORT* permite la escritura de valores lógicos en los pines, en caso hayan sido configurados como salidas.

8.3. Esquemáticos

Los circuitos implementados se muestran en las figuras 9 y 10

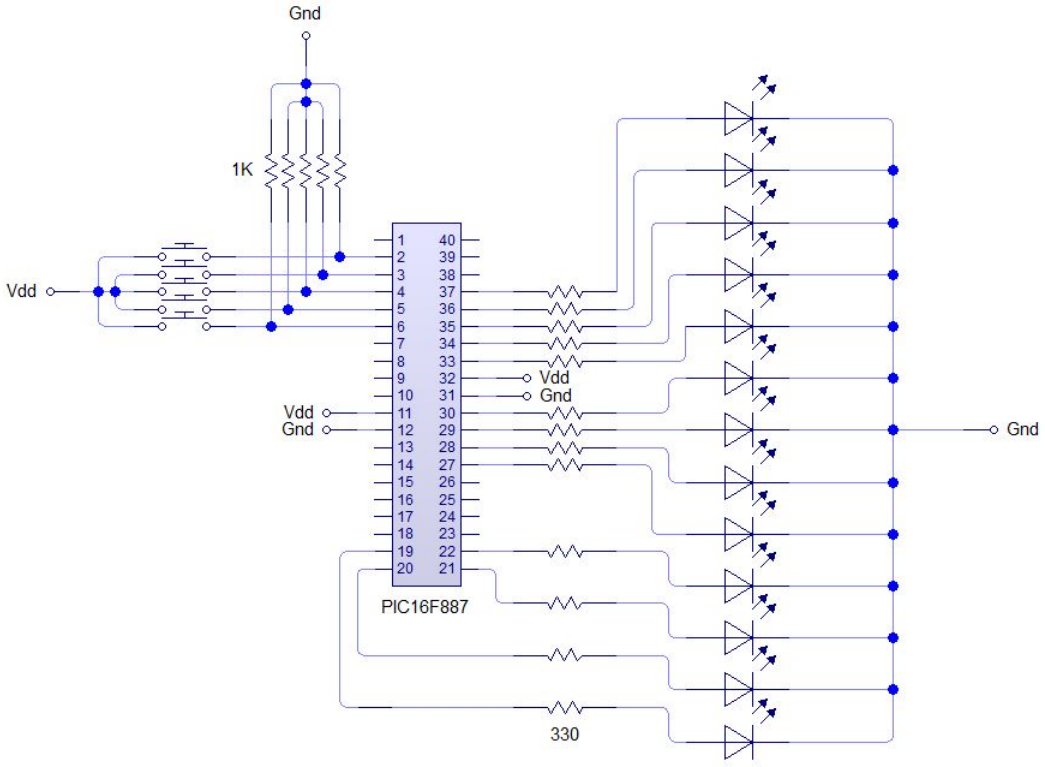


Figura 9: Circuito para la evaluación del módulo de entradas y salidas del PIC.

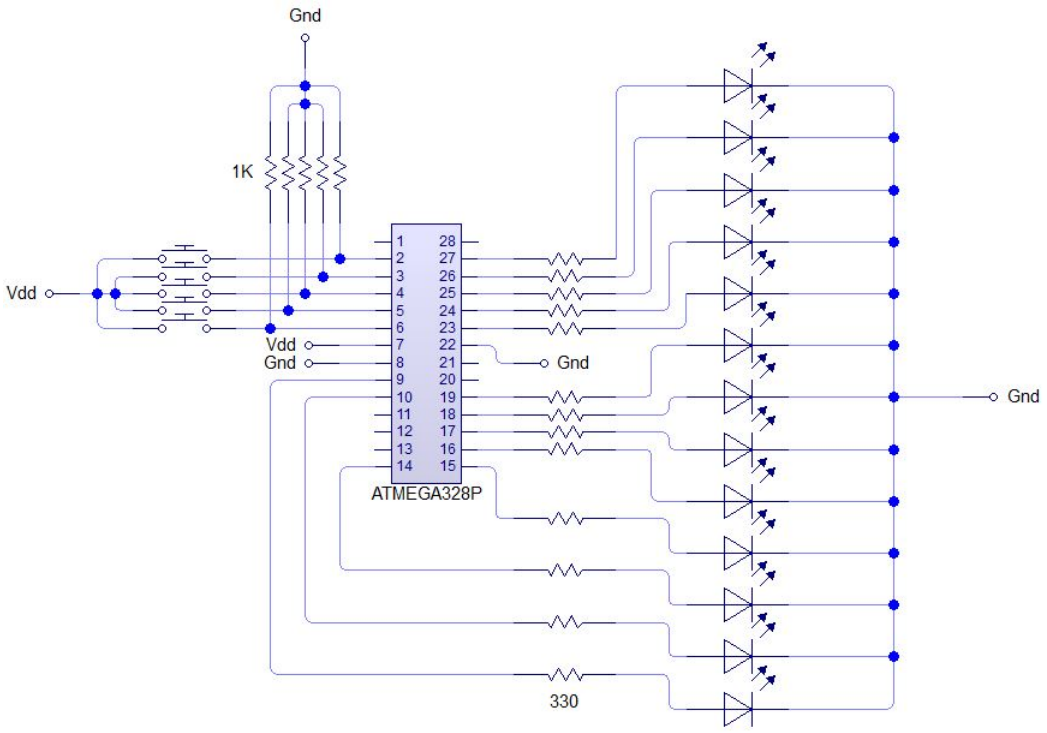


Figura 10: Circuito para la evaluación del módulo de entradas y salidas del ATMega.

8.4. Resultados

Las mediciones se realizaron al ejecutar el programa representado por la secuencia de la Figura 8 con el analizador lógico conectado a dos de los botones implementados: el botón de incremento del primer contador de 4 bits y al botón de suma de los contadores. Además, se conectaron las LEDs correspondientes al primer bit del primer contador de 4 bits de ambos microcontroladores y a las LEDs del primer bit del contador de suma. Dadas las conexiones, se midió el tiempo que requirieron ambos microcontroladores para ejecutar las operaciones de incremento de contadores y suma de contadores.

A continuación, en la Figura 11, se muestra un segmento de captura obtenido de las mediciones realizadas con el analizador lógico (para más datos, referirse al anexo 16.1):

Figura 11: Segmento de captura de medición de entradas y salidas.

8.4.1. Resultados en ensamblador

A continuación, en el Cuadro 10, se muestran los tiempos obtenidos para la ejecución del programa mostrado en la Figura 8 en ensamblador en ambos microcontroladores, donde LED 1 representa el tiempo que le toma al microcontrolador encender la LED del primer bit del contador 1 después de haber presionado el botón de incremento y LED 2 representa el tiempo requerido para encender la LED del primer bit del contador de suma en ensamblador:

Microcontrolador	LED 1 (μs)	LED 2 (μs)
PIC16F887	15.25	11
ATMega328P	4	3.5

Cuadro 10: Resultados obtenidos de entradas y salidas en ensamblador.

8.4.2. Resultados en C

A continuación, en el Cuadro 11, se muestran los tiempos obtenidos para la ejecución del programa mostrado en la Figura 8 en ensamblador en ambos microcontroladores, donde LED 1 representa el tiempo que le toma al microcontrolador encender la LED del primer

bit del contador 1 después de haber presionado el botón de incremento y LED 2 representa el tiempo requerido para encender la LED del primer bit del contador de suma en C:

Microcontrolador	LED 1 (μs)	LED 2 (μs)
PIC16F887	56.25	30.25
ATMega328P	5.25	4.5

Cuadro 11: Resultados obtenidos de entradas y salidas en C.

8.4.3. Comparación de entradas y salidas entre lenguajes de programación

A continuación, en los cuadros 12 y 13, se muestran las diferencias porcentuales del programa implementado en el cambio de lenguaje de ensamblador a C, tomando como referencia el resultado obtenido en ensamblador:

Microcontrolador	Ensamblador (μs)	C (μs)	Diferencia porcentual (%)
PIC16F887	15.25	56.25	269
ATMega328P	4	5.25	31

Cuadro 12: Comparación de los resultados obtenidos en los distintos lenguajes para el primer bit en el primer contador.

Microcontrolador	Ensamblador (μs)	C (μs)	Diferencia porcentual (%)
PIC16F887	11	30.25	175
ATMega328P	3.5	4.5	29

Cuadro 13: Comparación de los resultados obtenidos en los distintos lenguajes para el primer bit en el contador de suma.

8.5. Discusión

Como se puede observar en los cuadros 10 y 11, los tiempos que requiere el PIC para ejecutar las funciones implementadas en el programa de la Figura 8 son significativamente mayores a los requeridos por el ATMega, siendo esto más evidente en el Cuadro 11, donde el tiempo requerido por el PIC para realizar un incremento en el primer contador asciende hasta $56.25\mu s$, mientras que el ATMega requiere solamente de $5.25\mu s$.

Al comparar el cambio de lenguaje para la ejecución del programa, se observa, en los cuadros 12 y 13, que el PIC se ve más afectado por el cambio de lenguaje de ensamblador a C, teniendo un incremento de 269 % para encender la primer LED del primer contador y un incremento de 175 % para encender la primer LED del contador de suma, mientras que el ATMega tiene incrementos de 31 % y 29 % respectivamente. Esto indica que el cambio de lenguaje es más eficiente en el ATMega, al tener un impacto menor en el tiempo de ejecución del programa.

Finalmente, es necesario mencionar que la ejecución del programa en el PIC no es eficiente, debido a que no se están aprovechando las características físicas que posee, ya que, como se mencionó antes, se ajustó el programa para utilizar un único puerto para ambos contadores requeridos, cuando lo más eficiente en el PIC, sería implementar un contador en cada puerto.

CAPÍTULO 9

Módulo de temporizadores

Los temporizadores, en los microcontroladores evaluados, funcionan con registros específicos que son contadores de 8 bits e incrementan en cada ciclo de reloj, el cual depende de la frecuencia de oscilación del microcontrolador. Esto implica que los temporizadores tienen un número máximo al cual pueden contar (256), por lo que se hace uso de los preescalados del temporizador.

El preescalado en el PIC16F887 llega a un máximo de 256, mientras que, en el ATmega328P, el preescalado llega hasta 1024. La diferencia se debe a que el PIC16F887 utiliza un cuarto de la frecuencia de oscilación, es decir, el temporizador del PIC aumenta cada 4 ciclos de reloj, contrario al ATmega, que aumenta cada ciclo de reloj.

9.1. Programa

El programa realizado para la evaluación del módulo consiste en el cambio de estado de una LED cada, aproximadamente, 0.25 segundos (250 milisegundos). Se decidió implementar dos variantes de este programa para comparar las capacidades que poseen los temporizadores con la utilización de interrupciones y con la ausencia de ellas. En el capítulo 10, se profundizará en el estudio de las interrupciones de los microcontroladores.

A continuación, en la Figura 12, se muestra el pseudocódigo del programa utilizado en este módulo:

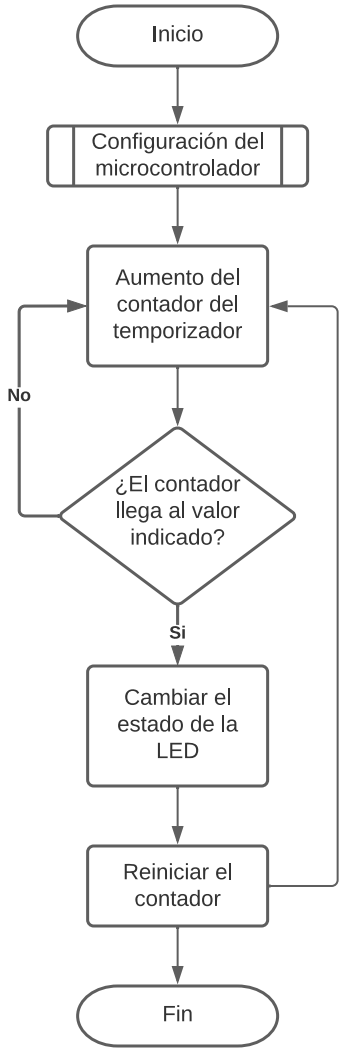


Figura 12: Pseudocódigo del programa implementado para el módulo de temporizadores.

Para medir los resultados, se utilizó el Analizador Lógico con 4 millones de mediciones por segundos, durante 60 segundos, con puntas en las LEDs colocadas como indicadores en cada microcontrolador. Para obtener una estadística fiable, se realizaron 5 capturas en cada caso.

9.2. Registros del módulo

A continuación, en el Cuadro 14, se muestran los registros necesarios para configurar los temporizadores en ambos microcontroladores:

Microcontrolador	Registro
PIC16F887	OPTION_REG TMR0 INTCON
ATmega328P	TCNT0 TCCR0A TCCR0B TIMSK0

Cuadro 14: Registros de los temporizadores.

9.3. Esquemáticos

Los circuitos implementados se muestran en las figuras 13 y 14:

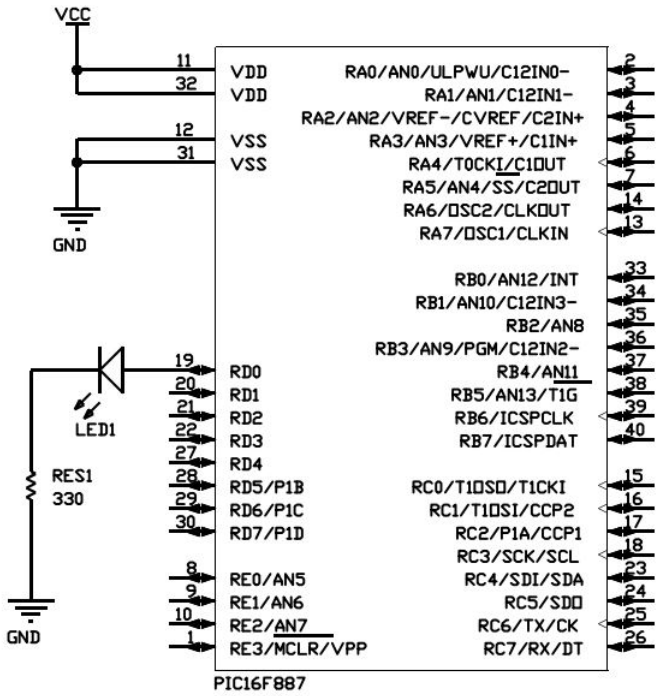


Figura 13: Circuito para la evaluación del módulo de temporizador en el PIC.

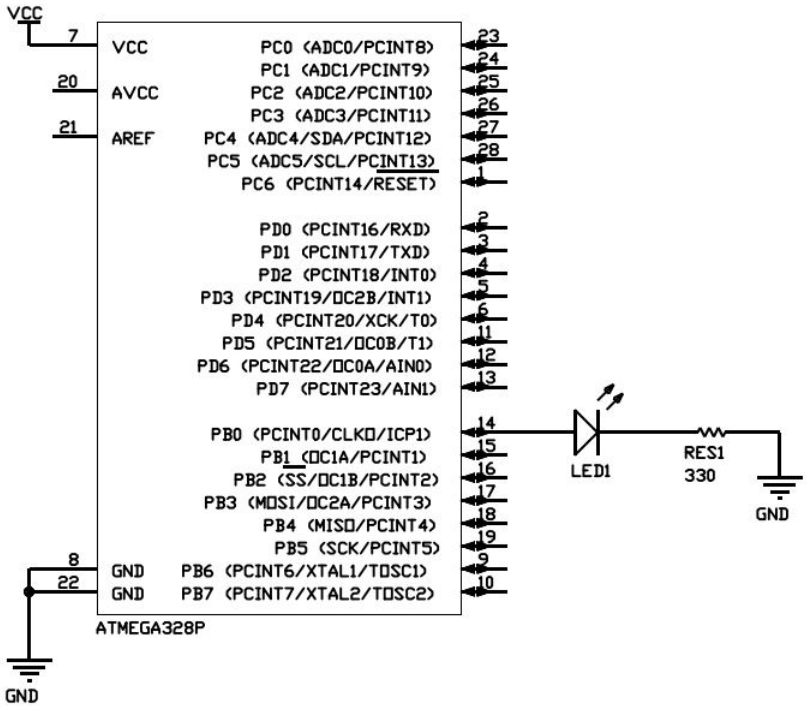


Figura 14: Circuito para la evaluación del módulo de temporizador en el ATmega.

9.4. Resultados

Como se mencionó anteriormente, se realizaron dos variantes del programa para cambiar el estado de una LED en ambos microcontroladores con y sin interrupciones. A continuación, en la Figura 15, se muestra un segmento de captura obtenido del Analizador Lógico empleado con la configuración antes mencionada y, en la Figura 16, un segmento de las tablas formadas en Excel a partir de los datos obtenidos (para más datos, referirse al anexo 16.1):

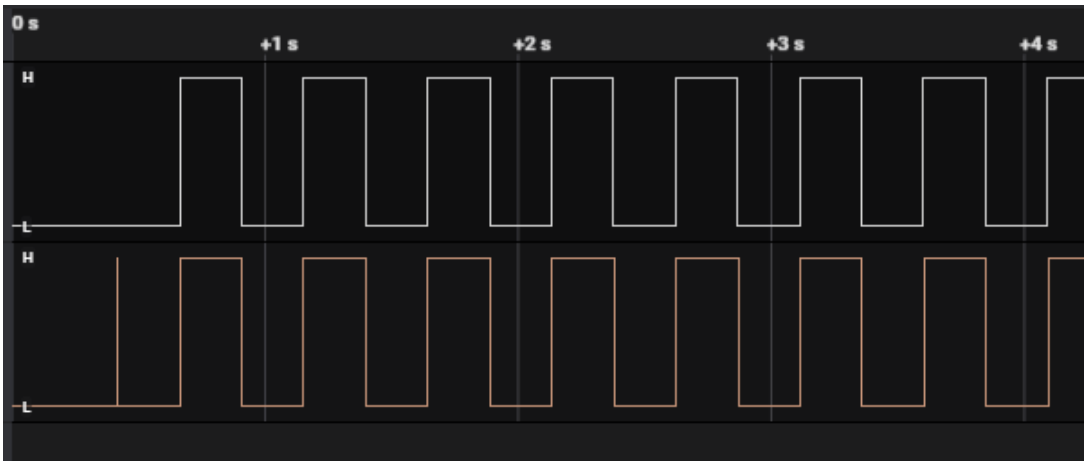


Figura 15: Segmento de captura de medición del temporizador.

PIC										ATMEGA										Status
1.000	DIFF1	2.000	DIFF2	3.000	DIFF3	4.000	DIFF4	5.000	DIFF5	1.000	DIFF1	2.000	DIFF2	3.000	DIFF3	4.000	DIFF4	5.000	DIFF5	
0.484	0.000	0.424	0.000	0.571	0.000	0.641	0.000	0.574	0.000	0.484	0.000	0.424	0.000	0.571	0.000	0.640	0.000	0.573	0.000	1
0.729	245.121	0.670	245.132	0.816	245.098	0.886	245.131	0.819	245.139	0.728	244.609	0.668	244.471	0.815	244.674	0.885	244.560	0.818	244.419	0
0.974	245.104	0.915	245.128	1.062	245.091	1.131	245.139	1.064	245.117	0.973	244.559	0.913	244.385	1.060	244.589	1.129	244.342	1.062	244.397	1
1.219	245.108	1.160	245.119	1.307	245.115	1.376	245.135	1.309	245.108	1.217	244.674	1.157	244.602	1.304	244.616	1.373	244.483	1.307	244.674	0
1.465	245.111	1.405	245.108	1.552	245.095	1.621	245.121	1.554	245.088	1.462	244.523	1.402	244.524	1.549	244.566	1.618	244.393	1.551	244.536	1
1.710	245.119	1.650	245.111	1.797	245.114	1.866	245.128	1.799	245.119	1.707	244.602	1.647	244.725	1.794	244.563	1.862	244.467	1.796	244.456	0
1.955	245.089	1.895	245.085	2.042	245.108	2.112	245.113	2.044	245.089	1.951	244.639	1.891	244.675	2.038	244.470	2.107	244.391	2.040	244.499	1
2.200	245.113	2.140	245.118	2.287	245.140	2.357	245.114	2.290	245.100	2.196	244.639	2.136	244.635	2.282	244.386	2.351	244.553	2.285	244.566	0
2.445	245.108	2.385	245.138	2.532	245.125	2.602	245.101	2.535	245.117	2.440	244.526	2.380	244.412	2.527	244.340	2.596	244.411	2.529	244.321	1
2.690	245.133	2.631	245.153	2.777	245.132	2.847	245.120	2.780	245.100	2.685	244.630	2.625	244.458	2.771	244.459	2.840	244.516	2.774	244.598	0
2.935	245.144	2.876	245.111	3.022	245.118	3.092	245.117	3.025	245.071	2.929	244.434	2.869	244.549	3.016	244.384	3.085	244.415	3.018	244.668	1
3.180	245.136	3.121	245.156	3.268	245.155	3.337	245.114	3.270	245.094	3.174	244.677	3.114	244.416	3.260	244.207	3.329	244.576	3.263	244.652	0
3.426	245.119	3.366	245.135	3.513	245.143	3.582	245.089	3.515	245.099	3.419	244.659	3.358	244.367	3.504	244.082	3.574	244.573	3.507	244.474	1
3.671	245.169	3.611	245.143	3.758	245.158	3.827	245.129	3.760	245.120	3.663	244.469	3.603	244.469	3.748	244.235	3.818	244.447	3.752	244.432	0
3.916	245.134	3.856	245.106	4.003	245.149	4.072	245.143	4.005	245.101	3.908	244.504	3.847	244.538	3.992	243.993	4.062	244.154	3.996	244.445	1
4.161	245.162	4.101	245.128	4.248	245.160	4.318	245.142	4.250	245.104	4.152	244.494	4.092	244.577	4.236	244.116	4.307	244.376	4.241	244.540	0
4.406	245.149	4.346	245.138	4.493	245.142	4.563	245.133	4.495	245.083	4.397	244.342	4.336	244.345	4.480	244.137	4.551	244.322	4.485	244.524	1
4.651	245.127	4.592	245.140	4.739	245.158	4.808	245.140	4.741	245.111	4.641	244.708	4.581	244.446	4.725	244.178	4.795	244.365	4.730	244.493	0

Figura 16: Segmento de tablas de datos en Excel.

Para el cálculo del tiempo exacto al que se debe encender la LED, se utiliza la siguiente fórmula:

$$T = P \times (256 - C_0) \times \frac{1}{F_{osc}} \tag{1}$$

Donde T es el Tiempo estimado, P es el valor asignado al Preescalador, C_0 es el valor asignado inicialmente al Contador (en caso se estén utilizando interrupciones) o el valor al que el Contador debe llegar (en caso no se estén utilizando interrupciones), y F_{osc} es la Frecuencia de oscilación a la que se está manejando el microcontrolador.

Dado a que el PIC aumenta su contador cada 4 ciclos de reloj, el valor obtenido en la Ecuación 1 debe multiplicarse por 4 en caso se esté calculando el tiempo estimado para el temporizador del PIC, es decir:

$$T_{PIC} = 4 \times T \tag{2}$$

Donde T_{PIC} es el Tiempo estimado para el PIC.

Para lograr un aproximado a 250 milisegundos, es necesario utilizar un contador auxiliar que permita retrasar el cambio de estado en la LED, por lo que la Ecuación final utilizada se ve de la siguiente forma:

$$T = P \times (256 - C_0) \times \frac{1}{F_{osc}} \times C_A \tag{3}$$

Donde C_A es el Contador auxiliar empleado para retrasar el cambio de estado de la LED.

Finalmente, el cálculo del tiempo estimado es el siguiente:

$$T = 1024 \times (256 - 128) \times \frac{1}{8MHz} \times 15 = 0.24576s = 245.76ms \tag{4}$$

Con el Analizador Lógico, se obtuvieron 241 mediciones por captura, dando un total de 1205 mediciones por microcontrolador para cada uno de los siguientes casos:

1. Temporizador sin interrupciones en ensamblador.
2. Temporizador sin interrupciones en C.
3. Temporizador con interrupciones en ensamblador.
4. Temporizador con interrupciones en C.

A continuación se muestran gráficos de dispersión, en los que se pretende mostrar la distribución de los datos obtenidos para cada microcontrolador, diagramas de caja y bigote, en los que se muestra la concentración de datos obtenidos, y cuadros estadísticos, en los que se recopila la estadística descriptiva y porcentajes de error obtenidos respecto al dato teórico calculado en la Ecuación 3:

9.4.1. Temporizador sin interrupciones en ensamblador

De las 1205 muestras de este programa, se obtuvo la estadística mostrada en el Cuadro 19:

Microcontrolador	Promedio(ms)	Máximo(ms)	Mínimo(ms)	Varianza(ms)	Desviación(ms)
PIC16F887	245.160	245.256	245.104	0.153	0.026
ATMega328P	244.928	245.234	244.545	0.689	0.095

Cuadro 15: Estadística descriptiva del temporizador sin interrupciones en ensamblador.

Con el tiempo estimado calculado en la Ecuación 3 y el promedio obtenido por microcontrolador se obtienen los porcentajes de error mostrados en el Cuadro 16:

Microcontrolador	Promedio (ms)	Porcentaje de error (%)
PIC16F887	245.160	0.24
ATMega328P	244.928	0.34

Cuadro 16: Porcentaje de error para cada microcontrolador del temporizador sin interrupciones en ensamblador.

Los resultados se muestran de forma gráfica en las figuras de la 17 a la 20:

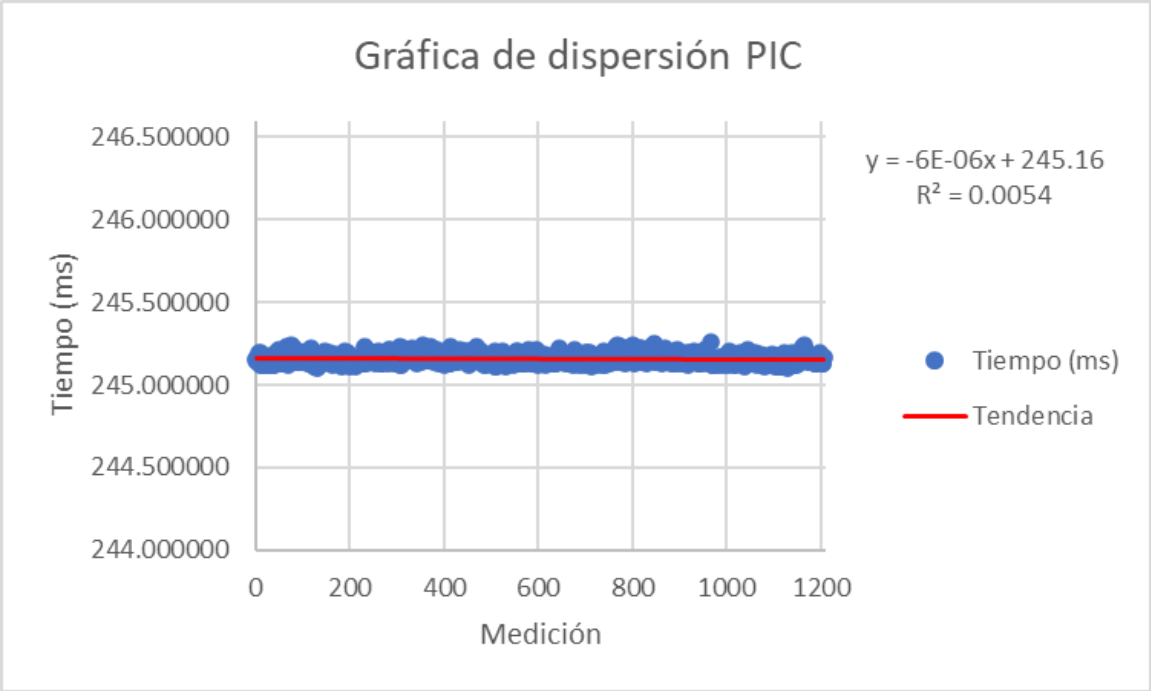


Figura 17: Gráfica de dispersión del temporizador sin interrupciones en ensamblador del PIC16F887.

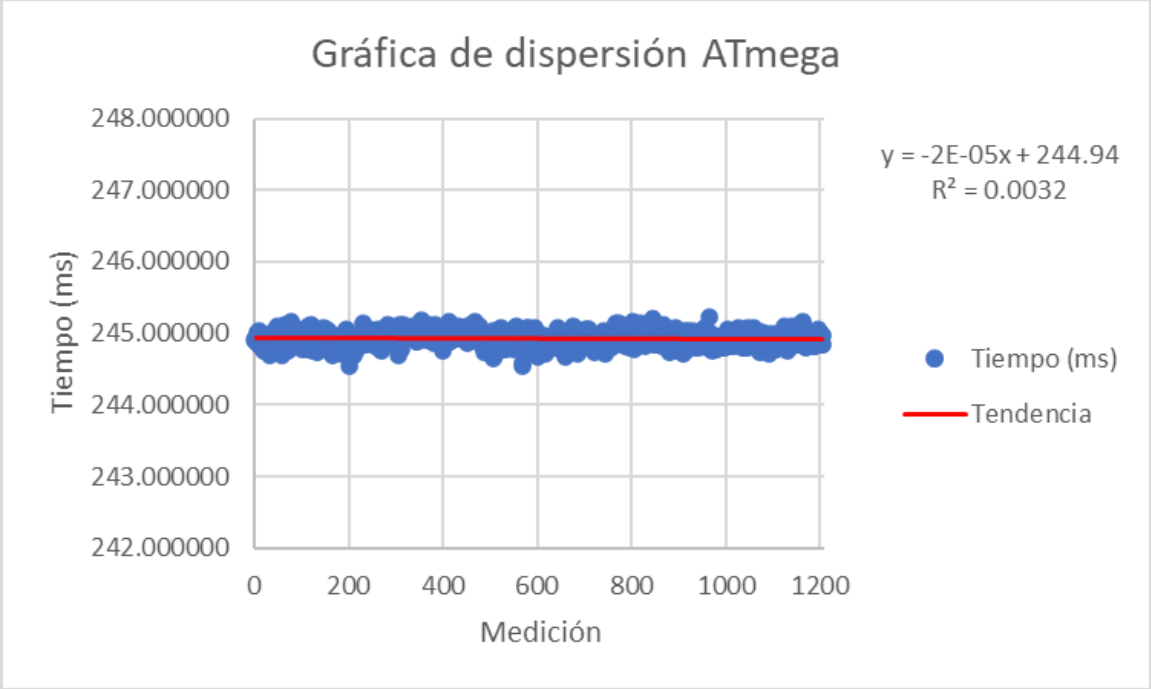


Figura 18: Gráfica de dispersión del temporizador sin interrupciones en ensamblador del ATmega328P.

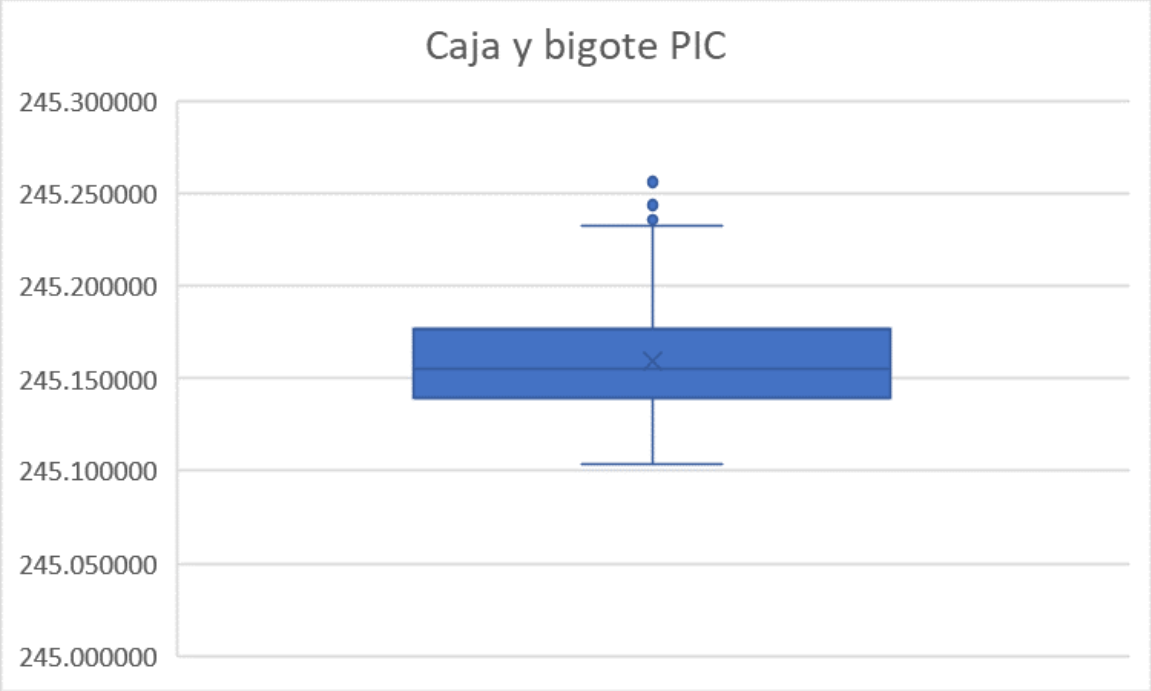


Figura 19: Diagrama de caja y bigote del temporizador sin interrupciones en ensamblador del PIC16F887.

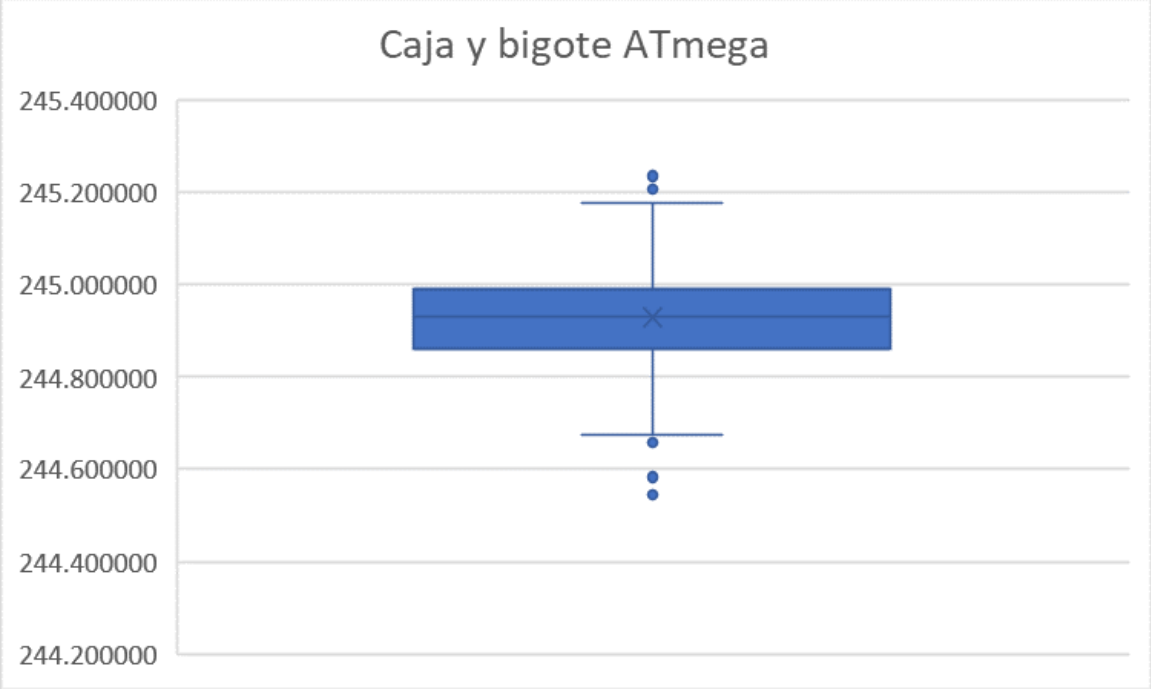


Figura 20: Diagrama de caja y bigote del temporizador sin interrupciones en ensamblador del ATMega328P.

9.4.2. Temporizador sin interrupciones en C

De las 1205 muestras de este programa, se obtuvo la estadística mostrada en el Cuadro 21:

Microcontrolador	Promedio(ms)	Máximo(ms)	Mínimo(ms)	Varianza(ms)	Desviación(ms)
PIC16F887	245.119	245.169	245.051	0.118	0.020
ATMega328P	244.420	244.777	243.911	0.866	0.132

Cuadro 17: Estadística descriptiva del temporizador sin interrupciones en C.

Con el tiempo estimado calculado en la Ecuación 3 y el promedio obtenido por microcontrolador se obtienen los porcentajes de error mostrados en el Cuadro 18:

Microcontrolador	Promedio (ms)	Porcentaje de error (%)
PIC16F887	245.119	0.26 %
ATMega328P	244.420	0.55 %

Cuadro 18: Porcentaje de error para cada microcontrolador del temporizador sin interrupciones en C.

Los resultados se muestran de forma gráfica en las figuras de la 21 a la 24:

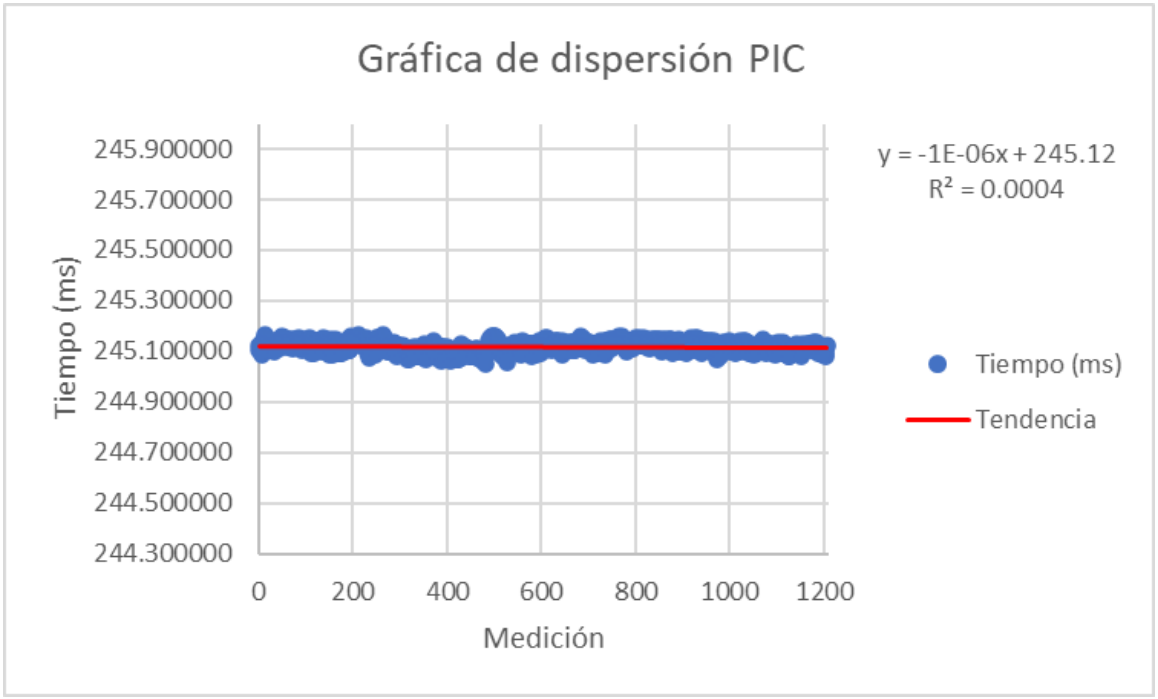


Figura 21: Gráfica de dispersión del temporizador sin interrupciones en C del PIC16F887.

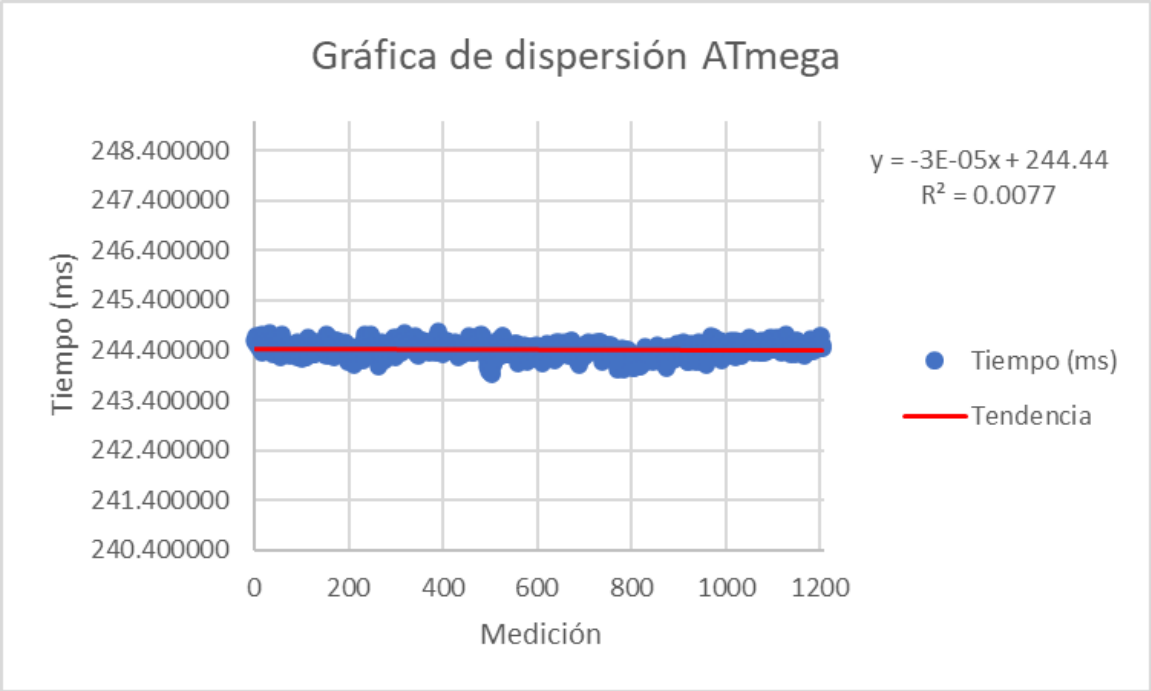


Figura 22: Gráfica de dispersión del temporizador sin interrupciones en C del ATmega328P.

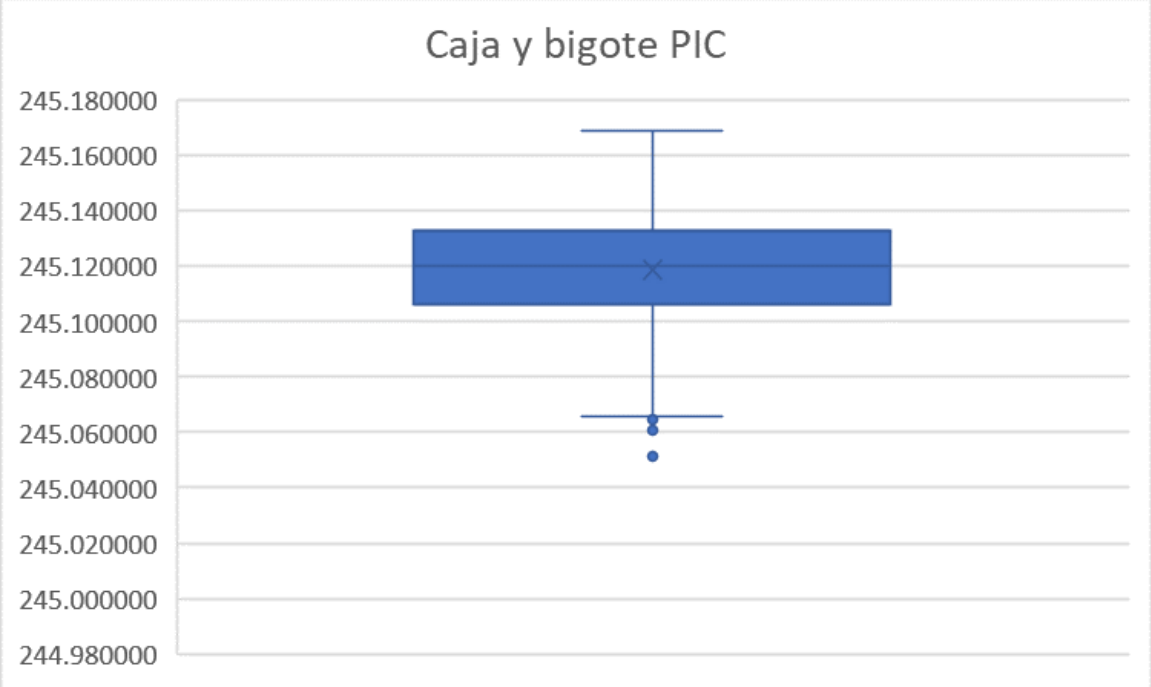


Figura 23: Diagrama de caja y bigote del temporizador sin interrupciones en C del PIC16F887.

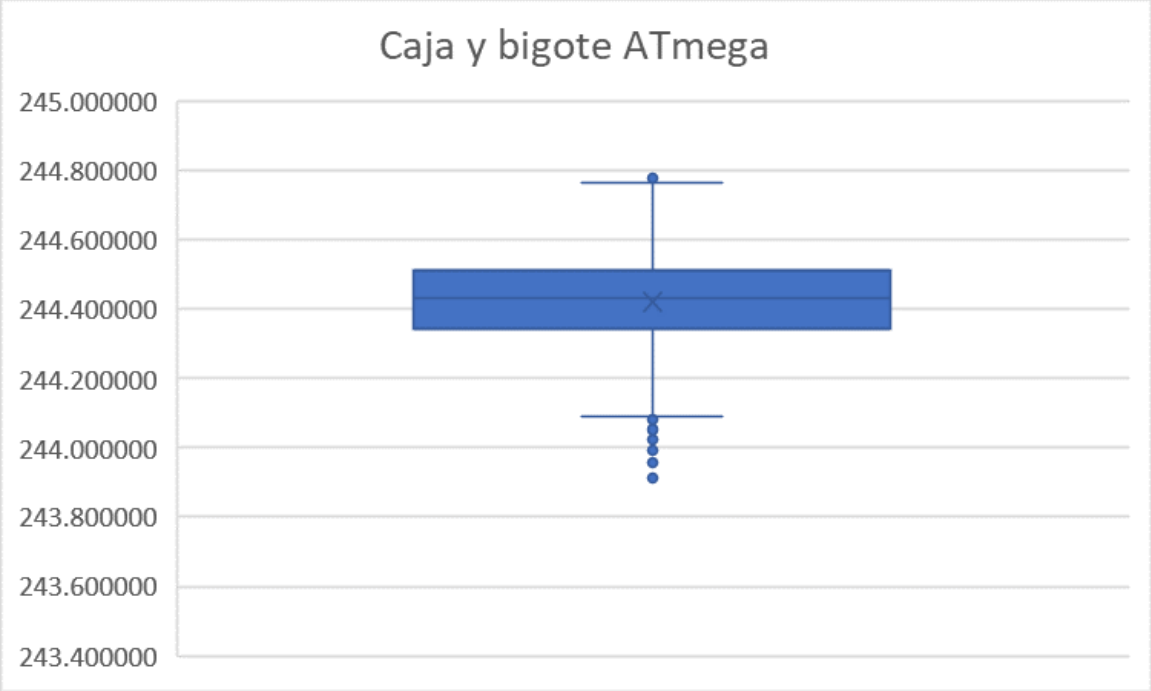


Figura 24: Diagrama de caja y bigote del temporizador sin interrupciones en C del ATMega328P.

9.4.3. Temporizador con interrupciones en ensamblador

De las 1205 muestras de este programa, se obtuvo la estadística mostrada en el Cuadro 19:

Microcontrolador	Promedio(ms)	Máximo(ms)	Mínimo(ms)	Varianza(ms)	Desviación(ms)
PIC16F887	245.215	245.287	245.157	0.130	0.026
ATMega328P	245.032	245.318	244.694	0.625	0.102

Cuadro 19: Estadística descriptiva del temporizador con interrupciones en ensamblador.

Con el tiempo estimado calculado en la Ecuación 3 y el promedio obtenido por microcontrolador se obtienen los porcentajes de error mostrados en el Cuadro 20:

Microcontrolador	Promedio (ms)	Porcentaje de error (%)
PIC16F887	245.215	0.22 %
ATMega328P	245.032	0.30 %

Cuadro 20: Porcentaje de error para cada microcontrolador del temporizador con interrupciones en ensamblador.

Los resultados se muestran de forma gráfica en las figuras de la 25 a la 28:

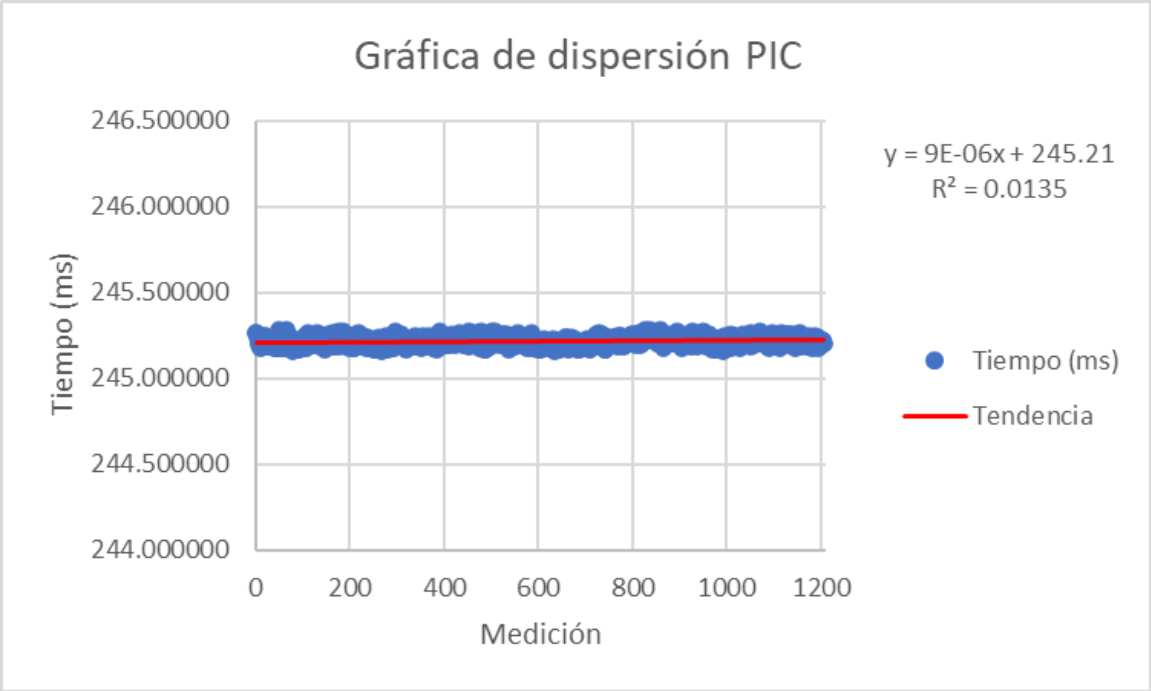


Figura 25: Gráfica de dispersión del temporizador con interrupciones en ensamblador del PIC16F887.

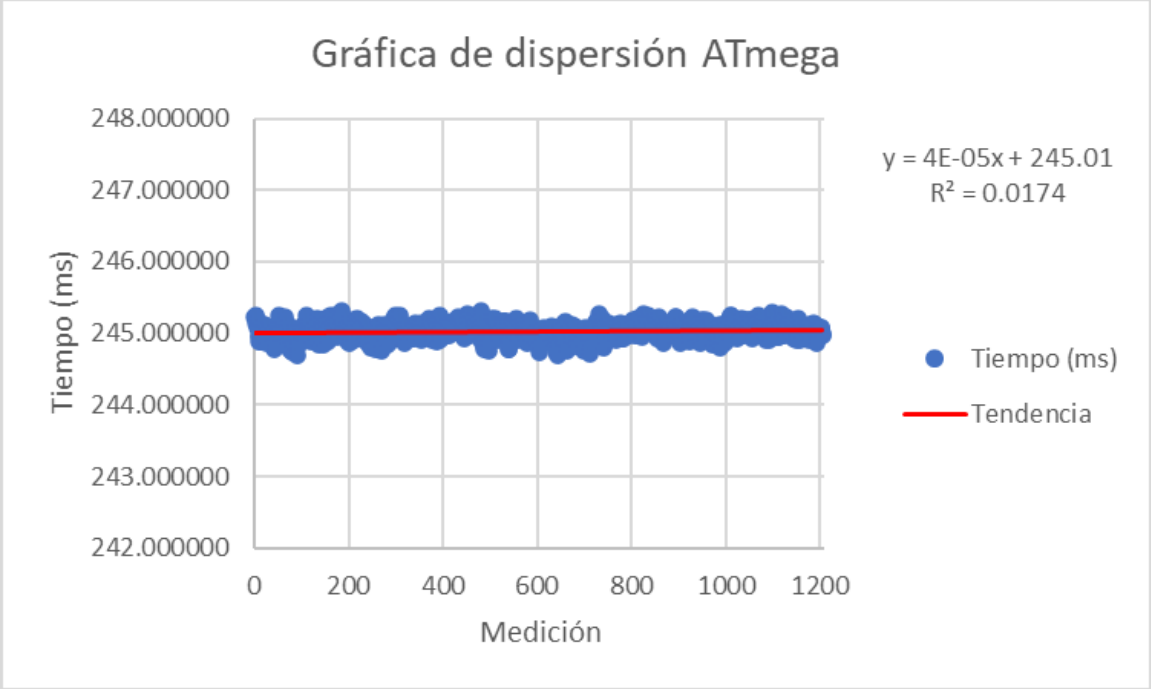


Figura 26: Gráfica de dispersión del temporizador con interrupciones en ensamblador del ATmega328P.

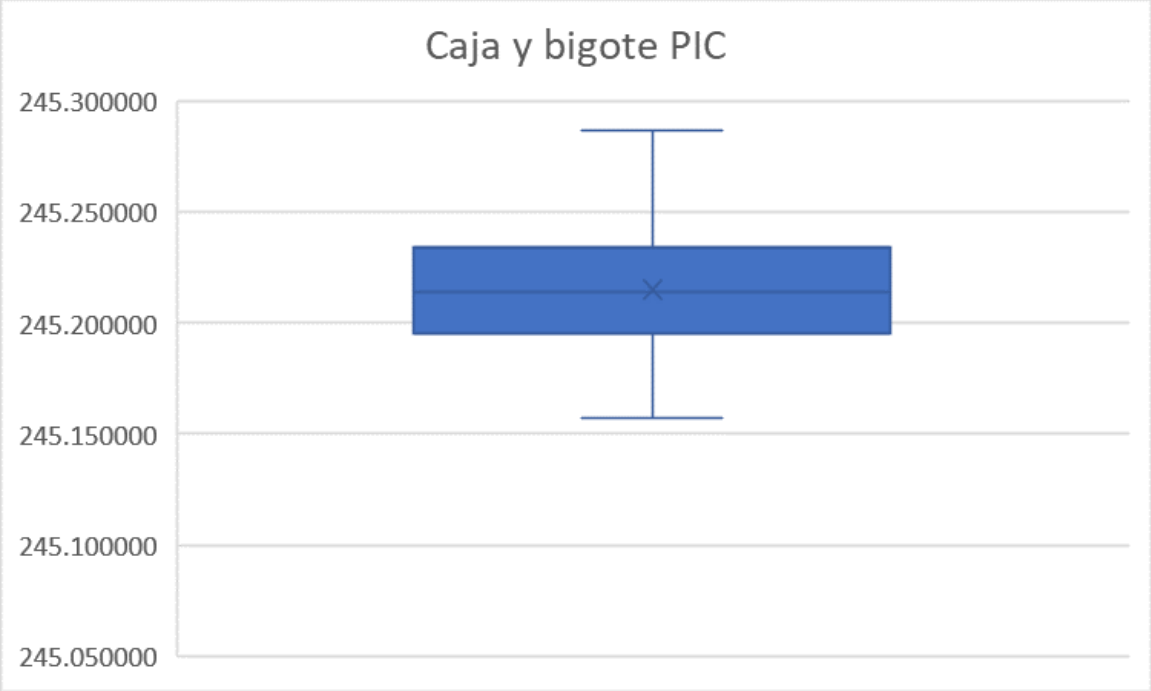


Figura 27: Diagrama de caja y bigote del temporizador con interrupciones en ensamblador del PIC16F887.

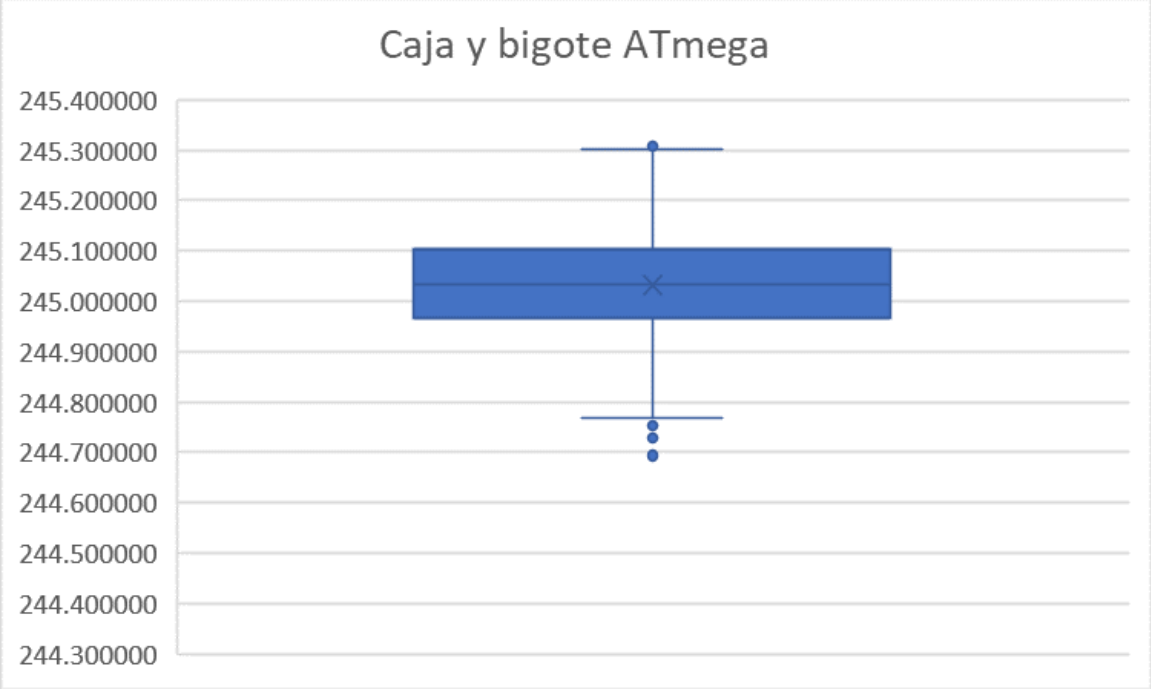


Figura 28: Diagrama de caja y bigote del temporizador con interrupciones en ensamblador del ATMega328P.

9.4.4. Temporizador con interrupciones en C

De las 1205 muestras de este programa, se obtuvo la estadística mostrada en el Cuadro 21:

Microcontrolador	Promedio(ms)	Máximo(ms)	Mínimo(ms)	Varianza(ms)	Desviación(ms)
PIC16F887	245.335	245.396	245.258	0.138	0.026
ATMega328P	245.075	245.487	244.701	0.786	0.105

Cuadro 21: Estadística descriptiva del temporizador con interrupciones en C.

Con el tiempo estimado calculado en la Ecuación 3 y el promedio obtenido por microcontrolador se obtienen los porcentajes de error mostrados en el Cuadro 22:

Microcontrolador	Promedio (ms)	Porcentaje de error ($\%$)
PIC16F887	245.335	0.17 $\%$
ATMega328P	245.075	0.28 $\%$

Cuadro 22: Porcentaje de error para cada microcontrolador del temporizador con interrupciones en C.

Los resultados se muestran de forma gráfica en las figuras de la 29 a la 32:

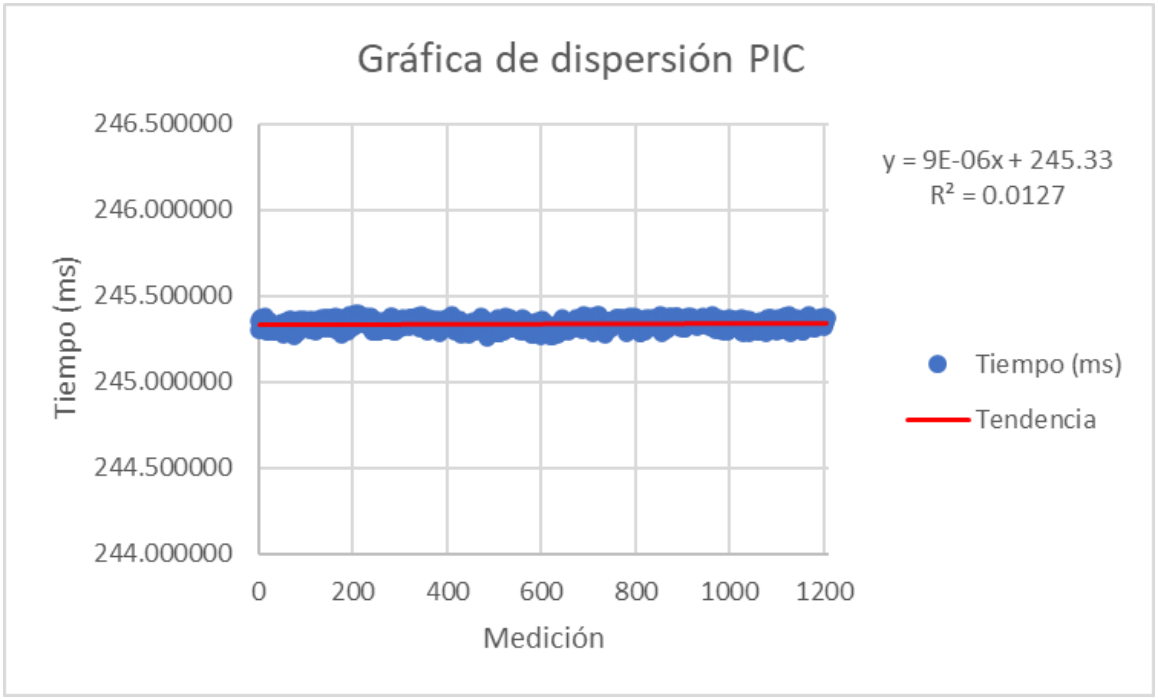


Figura 29: Gráfica de dispersión del temporizador con interrupciones en C del PIC16F887.

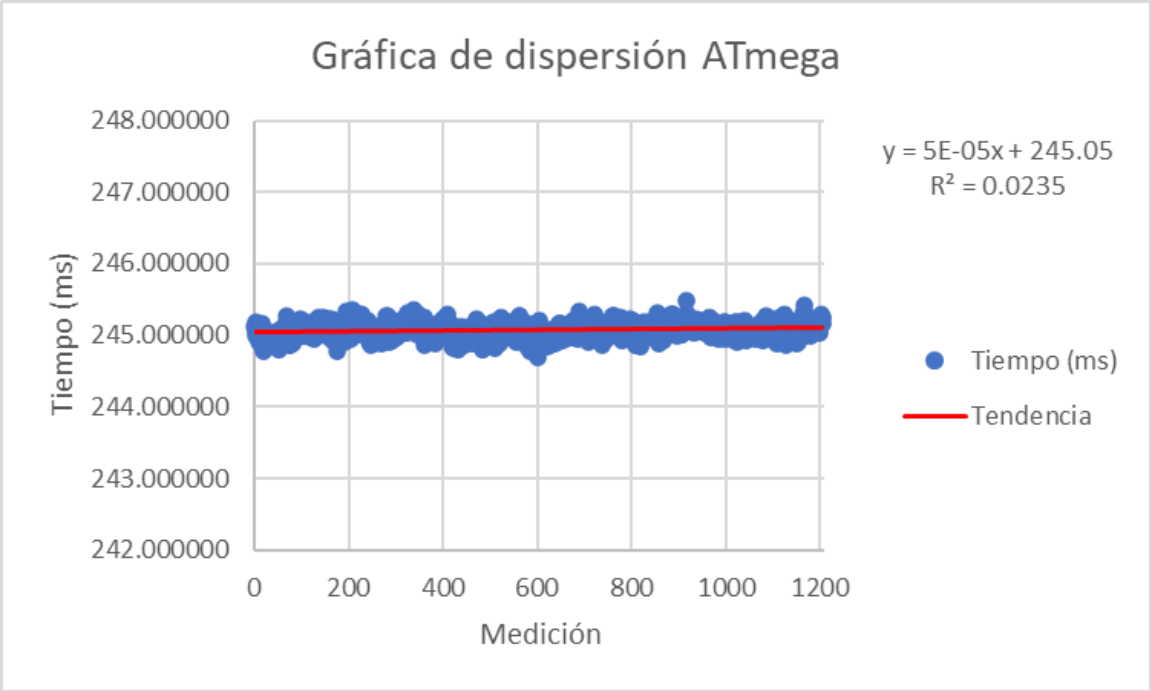


Figura 30: Gráfica de dispersión del temporizador con interrupciones en C del ATmega328P.

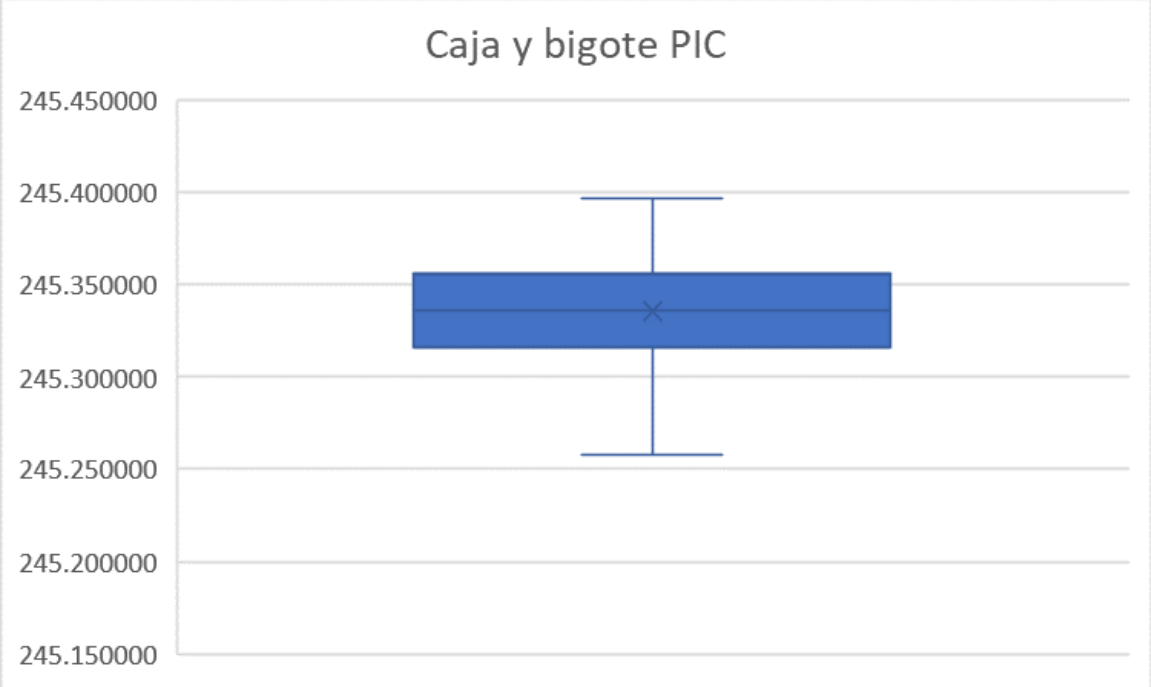


Figura 31: Diagrama de caja y bigote del temporizador con interrupciones en C del PIC16F887.

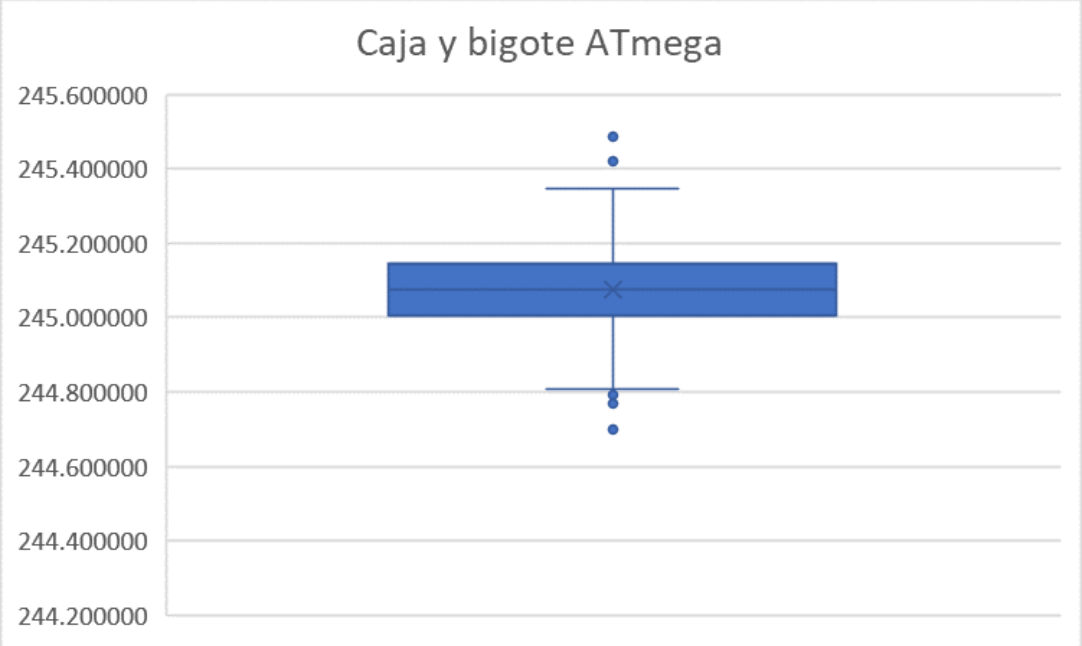


Figura 32: Diagrama de caja y bigote del temporizador con interrupciones en C del ATMega328P.

9.5. Discusión

En el caso de temporizador sin interrupciones en ensamblador, se puede observar en el Cuadro 15 que el promedio de valores presentado por el PIC es más aproximado al tiempo estimado que el presentado por el ATMega, como se puede observar en el Cuadro 16. También es necesario notar que la varianza de los datos del PIC es considerablemente menor que la varianza del ATMega, esto implica que la distribución de los datos del PIC es más uniforme que los datos del ATMega, esto se puede observar en las figuras 17 y 18, donde la pendiente de la línea de tendencia del PIC es de -6×10^{-6}, mientras que la del ATMega es de -2×10^{-5}, además de presentar un valor R^2 mayor, lo cual indica una mejor adaptación de la línea de tendencia respecto a los datos recopilados. Finalmente, es necesario observar los diagramas de caja y bigote en las figuras 19 y 20, donde el rango en que hay mayor concentración de datos del PIC es menor que el rango de concentración de datos del ATMega.

Al comparar los demás casos de temporizador, se puede observar que la tendencia mencionada anteriormente se mantiene, como se puede observar en los cuadros 17, 19 y 19, donde las varianzas de datos del PIC son considerablemente menores a las del ATMega, en los cuadros 18, 20 y 22, donde los porcentajes de error presentados por el PIC son menores a los del ATMega, en las figuras 21, 22, 25, 26, 29 y 30, donde las pendientes de las líneas de tendencia del PIC son menores a las del ATMega, y en las figuras 23, 24, 27, 28, 31 y 32, donde las concentraciones de datos del PIC y la cantidad de datos atípicos son menores a las del ATMega.

Finalmente, es necesario mencionar que, al emplear interrupciones para los temporizadores, en el caso de evaluación utilizado en este capítulo, el PIC tiene ventaja sobre el ATMega debido a la forma en que ejecuta las interrupciones, como se verá en el capítulo 10.

CAPÍTULO 10

Módulo de interrupciones

Las interrupciones, como se ha mencionado, son mecanismos que utilizan los microcontroladores para ejecutar un conjunto de instrucciones especificado por el usuario al momento en que una señal, ya sea externa o interna, sea detectada en el microcontrolador. En los microcontroladores evaluados, las interrupciones se separan por vectores, que hacen referencia a una localidad de memoria específica, a partir de la cuál se ejecutará el programa de interrupción.

Según la hoja de datos del PIC [13], el PIC16F887 posee un único vector de interrupción, en la localidad de memoria 0004h, en el cuál se ejecutan todas las interrupciones que posee el microcontrolador. Por este motivo, es necesario verificar qué interrupción fue activada al momento de llegar al vector de interrupción con el fin de ejecutar correctamente los programas que hagan uso de este módulo.

Por otro lado, según la hoja de datos del ATMega, [2], el ATMega328P posee 25 vectores distintos de interrupción, que abarcan las localidades de memoria desde 00002h hasta 0032h, en los cuales se ejecutaran interrupciones específicas según el tipo de interrupción que ha sido activada dentro del microcontrolador. En este caso, no es necesario verificar qué interrupción ha sido activada, ya que, al tener diversos vectores de interrupción, el código ejecutado puede ser programado específicamente para la interrupción correspondiente a un único vector de interrupción.

10.1. Programa

Para la evaluación de este módulo, se optó por implementar un programa capaz de cambiar el estado de una LED según el estado de un botón colocado en un pin de interrupción, como se puede observar en las figuras 34 y 35. Según lo indicado en la secuencia mostrada

en la Figura 33, la LED indicadora cambia de estado de la misma forma que el botón de interrupción, es decir, que si el botón de interrupción es activado, la LED también es activada; lo mismo para el caso contrario, si el botón de interrupción es desactivado, la LED también es desactivada.

A continuación, en la Figura 33, se muestra el pseudocódigo del programa utilizado en este módulo:

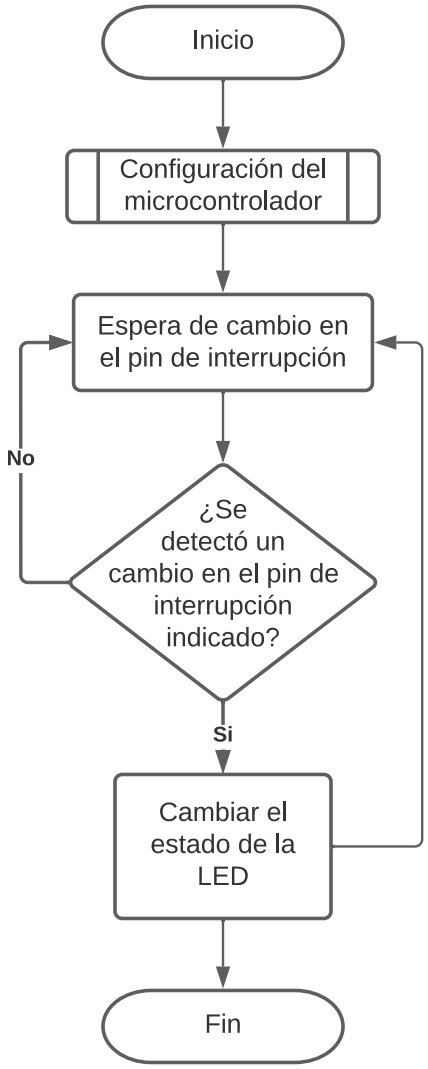


Figura 33: Pseudocódigo del programa implementado para el módulo de interrupciones.

10.2. Registros del módulo

A continuación, en el Cuadro 23, se muestran los registros requeridos para configurar las interrupciones en cada microcontrolador:

Microcontrolador	Registro
PIC16F887	INTCON IOCB RETFIE
ATmega328P	PCICR PCMSK0 PCMSK1 PCMSK2 RETI

Cuadro 23: Registros y comandos de las interrupciones evaluadas.

Es necesario mencionar que los registros mencionados aplican únicamente para las interrupciones en los pines de interrupción de los microcontroladores, es decir, interrupción por entradas. Esto, en el PIC, aplica únicamente para los pines del Puerto B (8 pines en total), mientras que en el ATMega aplica en todos los pines de los puertos disponibles (23 pines en total).

Adicionalmente, es necesario configurar, también, el vector de interrupción: en el PIC es, como se mencionó anteriormente, la localidad de memoria 0004h, mientras que en el ATMega es necesario referirse a la hoja de datos para identificar el vector de interrupción específico de cada tipo de interrupción [2], en este caso, se utiliza el vector *PCINT2*, ubicado en la localidad de memoria 000Ah.

Finalmente, en el Cuadro 24, se muestran las funciones utilizadas específicamente en C para la definición de los vectores de interrupción:

Microcontrolador	Función
PIC16F887	__interrupt()
ATMega328P	ISR()

Cuadro 24: Funciones para la definición de vectores de interrupción en C en ambos microcontroladores.

Como se muestra en el Cuadro 24, la definición de interrupciones en C para el PIC se realiza con la función *__interrupt()*, la cual es de tipo vacío. Por otro lado, en el ATMega se utiliza la función *ISR()*, en la que se debe especificar, según lo indicado en su hoja de datos [2], el vector de interrupción específico que se utiliza.

10.3. Esquemáticos

Los circuitos implementados se muestran en las figuras 34 y 35:

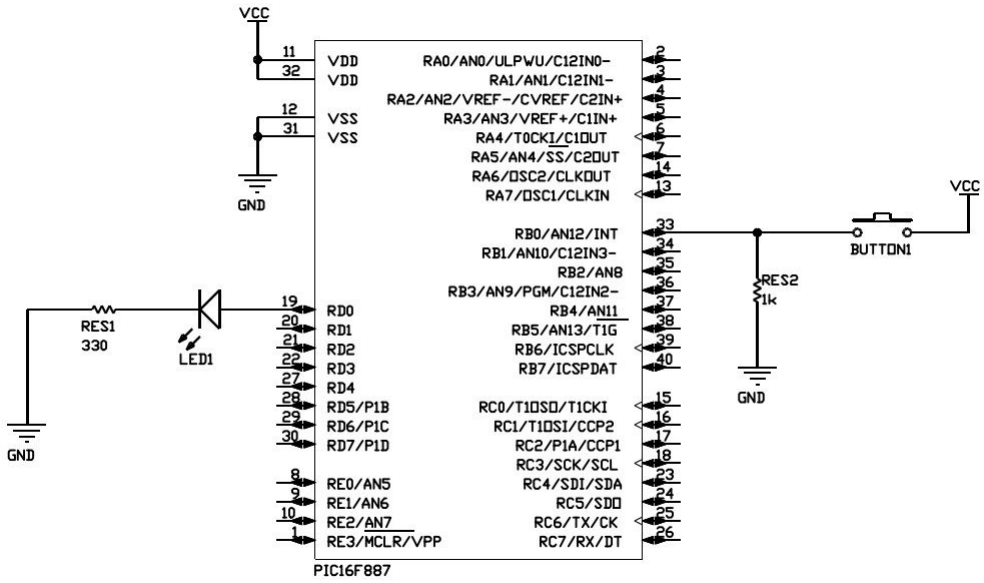


Figura 34: Circuito para la evaluación del módulo de interrupciones en el PIC.

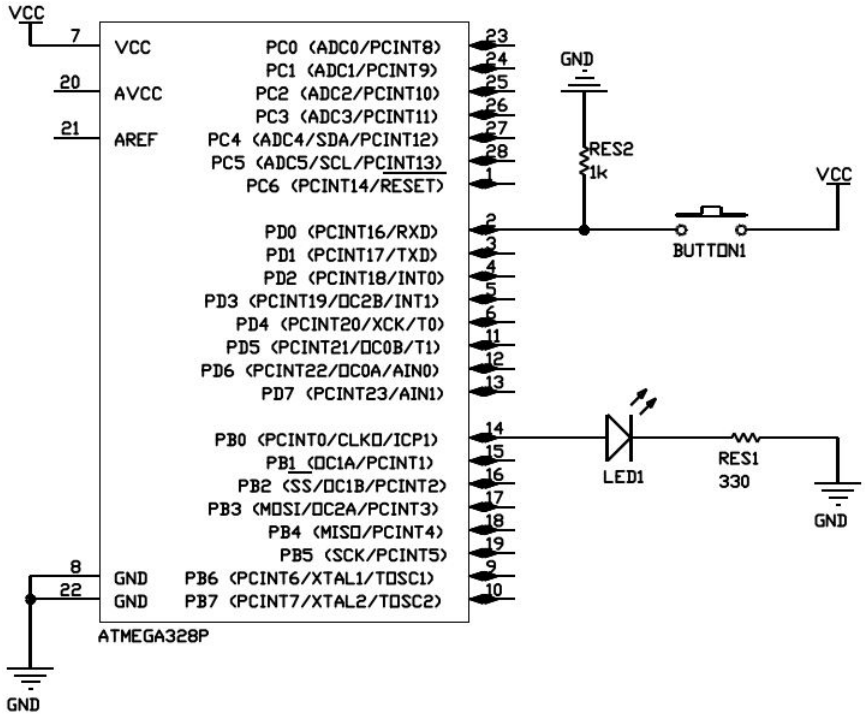


Figura 35: Circuito para la evaluación del módulo de interrupciones en el ATmega.

10.4. Resultados

Como se mencionó anteriormente, la evaluación del módulo se realizó con el programa descrito por la secuencia mostrada en la Figura 33, encendiendo y apagando una LED según el estado del botón en el pin de interrupción establecido en los programas. En la Figura 36 se muestra un segmento de captura obtenido al ejecutar los programas (para más datos, referirse al anexo 16.1):

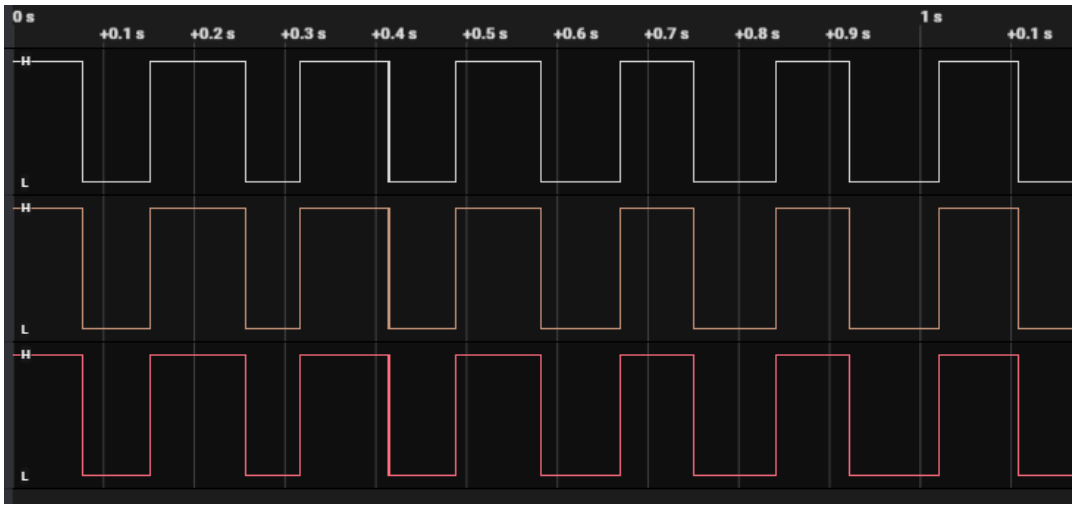


Figura 36: Segmento de captura de medición de interrupciones.

De las capturas obtenidas, se mide el tiempo que requieren ambos microcontroladores para entrar y salir de la interrupción y se obtiene la moda en todas las mediciones realizadas, tomando como referencia el momento en que se pulsa y se suelta el botón en los pines de interrupción.

10.4.1. Resultados en ensamblador

A continuación, en los cuadros 25 y 26, se muestran los resultados obtenidos para el ingreso a las interrupciones en ambos microcontroladores en ensamblador:

Microcontrolador	Tiempo requerido (μs)
PIC16F887	4.25
ATMega328P	1.75

Cuadro 25: Resultados obtenidos para el ingreso a interrupciones en ensamblador.

Microcontrolador	Tiempo requerido (μs)
PIC16F887	4
ATMega328P	0.75

Cuadro 26: Resultados obtenidos para la salida de interrupciones en ensamblador.

Como se puede observar en los cuadros 25 y 26, en ensamblador, el PIC requiere de $4.25\mu s$ para entrar a interrupciones y de $4\mu s$ para la salida de interrupciones, mientras que el ATMega requiere de $1.75\mu s$ y $0.75\mu s$ respectivamente.

10.4.2. Resultados en C

A continuación, en los cuadros 27 y 28, se muestran los resultados obtenidos para el ingreso a las interrupciones en ambos microcontroladores en C:

Microcontrolador	Tiempo requerido (μs)
PIC16F887	9.75
ATMega328P	1.75

Cuadro 27: Resultados obtenidos para el ingreso a interrupciones en C.

Microcontrolador	Tiempo requerido (μs)
PIC16F887	9.25
ATMega328P	0.75

Cuadro 28: Resultados obtenidos para la salida de interrupciones en C.

Como se puede observar en los cuadros 27 y 28, en C, el PIC requiere de $9.75\mu s$ para entrar a interrupciones y de $9.25\mu s$ para la salida de interrupciones, mientras que el ATMega requiere de $1.75\mu s$ y $0.75\mu s$ respectivamente.

10.4.3. Comparación de interrupciones entre lenguajes de programación

A continuación, en los cuadros 29 y 30, se muestra la diferencia porcentual para la entrada y salida de interrupciones en ambos lenguajes de programación empleados, tomando como referencia el resultado obtenido en ensamblador:

Microcontrolador	Ensamblador (μs)	C (μs)	Diferencia porcentual (%)
PIC16F887	4.25	9.75	129.41
ATMega328P	1.75	1.75	0

Cuadro 29: Comparación entre ensamblador y C para la entrada a interrupciones de ambos microcontroladores.

Microcontrolador	Ensamblador (μs)	C (μs)	Diferencia porcentual (%)
PIC16F887	4	9.25	131.25
ATMega328P	0.75	0.75	0

Cuadro 30: Comparación entre ensamblador y C para la salida de interrupciones de ambos microcontroladores.

10.5. Discusión

Como se puede observar en los cuadros 25, 26, 27 y 28, en todos los casos, el PIC requiere de una mayor cantidad de tiempo tanto para entrar como para salir de las interrupciones que el ATMega. Sin embargo, los tiempos mayores del PIC se ven aumentados debido a las funciones *POP* y *PUSH* que debe implementar al comienzo y al final de su vector de interrupción.

Al comparar la diferencia entre lenguajes, se puede observar que, en el Cuadro 29, el PIC tiene un aumento de, aproximadamente, 129.41 % al pasar de ensamblador a C en la entrada de interrupciones, mientras que en el ATMega, el cambio de lenguaje de programación no afecta en los tiempos empleados, teniendo una diferencia porcentual de 0 %. De la misma forma ocurre en la salida de las interrupciones, como se observa en el Cuadro 30, donde se mantiene la misma tendencia y el PIC tiene un aumento de 131.25 % al pasar de ensamblador a C, y el ATMega mantiene la diferencia porcentual de 0 %. Por lo que se puede decir que el cambio de lenguaje de ensamblador a C le afecta considerablemente al PIC en los tiempos de entrada y salida de interrupciones, mientras que no causa ninguna diferencia en el ATMega.

CAPÍTULO 11

Módulo de memoria EEPROM

Como ya se ha mencionado, la memoria EEPROM: se utiliza como almacenamiento no volátil en los microcontroladores, es decir, que es capaz de guardar la información aún cuando se deja de suministrar energía al microcontrolador, por lo que, es un método de almacenamiento resistente a reinicios.

11.1. Programa

El programa empleado para la evaluación del módulo consiste en implementar, en los microcontroladores, contadores hexadecimales, cuyos valores fueron mostrados en visualizadores de siete segmentos, como se puede observar en las figuras 38 y 39. Estos contadores fueron modificados gracias a 2 botones para incrementar y disminuir su valor, tal como se aprecia en la secuencia mostrada en la Figura 37. Finalmente, se utilizaron dos botones adicionales, el primero para guardar el valor del contador en la EEPROM y el segundo para leerlo una vez guardado. Tanto para la escritura como para la lectura de datos en la memoria EEPROM se hizo uso de las secuencias recomendadas en las hojas de datos de ambos microcontroladores ([13] y [2]).

A continuación, en la Figura 37, se muestra el pseudocódigo del programa implementado:

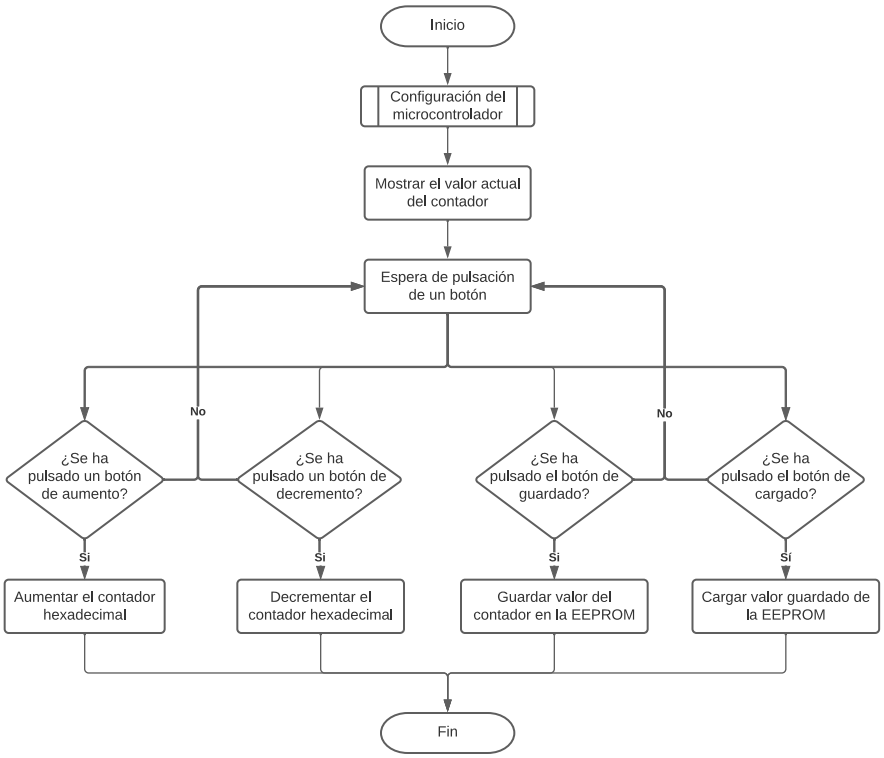


Figura 37: Pseudocódigo del programa implementado para el módulo de memoria EEPROM.

11.2. Registros del módulo

A continuación, en el Cuadro 31, se muestran los registros necesarios para realizar lecturas y escrituras en la memoria EEPROM en ambos microcontroladores:

Microcontrolador	Registro
PIC16F887	EEDATA EEDATH EEADR EEADRH EECON1 EECON2
ATmega328P	EEARL EEARH EEDR EECR

Cuadro 31: Registros de la memoria EEPROM.

11.3. Esquemáticos

Los circuitos implementados se muestran en las figuras 38 y 39

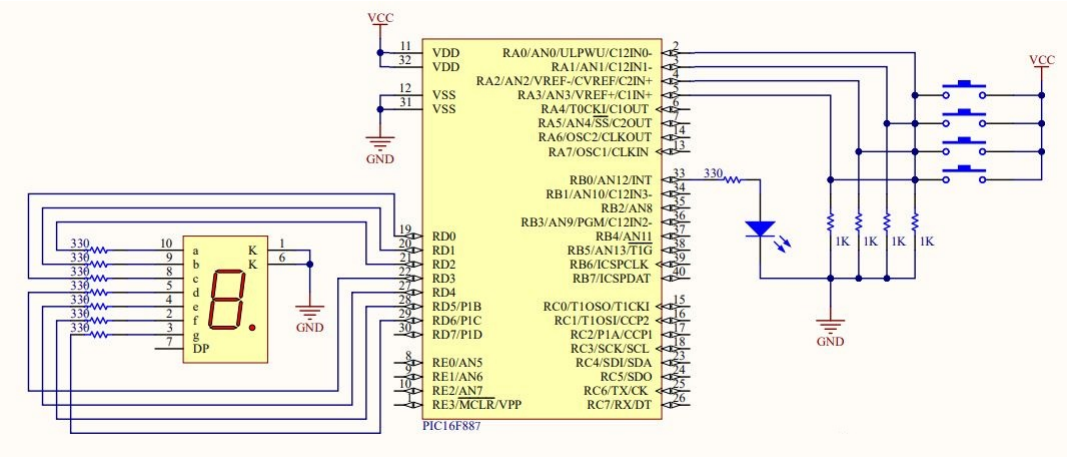


Figura 38: Circuito para la evaluación del módulo de memoria EEPROM en el PIC.

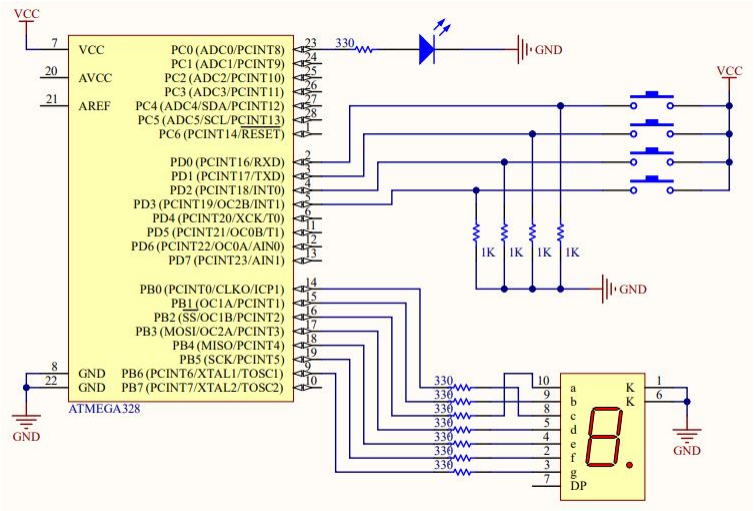


Figura 39: Circuito para la evaluación del módulo de memoria EEPROM en el ATmega.

11.4. Resultados

Dado que, en la memoria EEPROM, se puede tanto escribir como leer datos guardados, se realizaron las mediciones aprovechando los dos botones implementados en los circuitos que ejecutaron estas funciones en ambos microcontroladores, según la secuencia mostrada en la Figura 37. A continuación, en la Figura 40, se muestra un segmento de una de las capturas obtenidas al realizar las mediciones con el analizador lógico (para más datos, referirse al anexo 16.1):

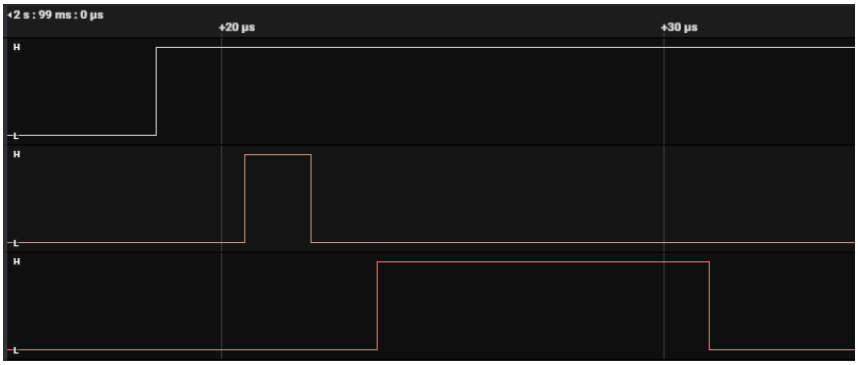


Figura 40: Segmento de captura de medición de la memoria EEPROM.

De las capturas obtenidas, se obtienen las modas en los tiempos que emplea cada microcontrolador en cada uno de los siguientes casos:

1. Escritura de la memoria EEPROM en ensamblador.
2. Lectura de la memoria EEPROM en ensamblador.
3. Escritura de la memoria EEPROM en C.
4. Lectura de la memoria EEPROM en C.

Los datos obtenidos se muestran a continuación:

11.4.1. Escritura de la memoria EEPROM en ensamblador

En el Cuadro 32 se muestran los resultados obtenidos para la escritura de la memoria EEPROM en ensamblador:

Microcontrolador	Tiempo de escritura (μs)
PIC16F887	3578
ATMega328P	1.5

Cuadro 32: Resultados obtenidos de la memoria EEPROM para escritura en ensamblador.

Como se puede observar en el Cuadro 32, el tiempo de escritura empleado por el PIC es de $3578\mu s$, mientras que el del ATMega es de $1.5\mu s$. Esto indica que el PIC emplea un tiempo considerablemente mayor que el ATMega para guardar datos en la memoria EEPROM.

11.4.2. Lectura de la memoria EEPROM en ensamblador

En el Cuadro 33 se muestran los resultados obtenidos para la lectura de la memoria EEPROM en ensamblador:

Microcontrolador	Tiempo de lectura (μs)
PIC16F887	7.5
ATMega328P	1.5

Cuadro 33: Resultados obtenidos de la memoria EEPROM para lectura en ensamblador.

En el Cuadro 33, se observan que el tiempo requerido en el PIC para la lectura de datos en la memoria EEPROM es de $7.5\mu s$, mientras que en el ATMega, el tiempo requerido es de $1.5\mu s$. Esto demuestra que el PIC requiere de una mayor cantidad de tiempo para completar la lectura de datos guardados en la memoria EEPROM.

11.4.3. Escritura de la memoria EEPROM en C

En el Cuadro 34 se muestran los resultados obtenidos para la escritura de la memoria EEPROM en C:

Microcontrolador	Tiempo de escritura (μs)
PIC16F887	3578
ATMega328P	2

Cuadro 34: Resultados obtenidos de la memoria EEPROM para escritura en C.

Se observa, en el Cuadro 34, que, nuevamente, el tiempo que requiere el PIC para la escritura de datos en la memoria EEPROM es considerablemente superior al requerido por el ATMega; el PIC requiriendo de $3578\mu s$, mientras que el ATMega únicamente requiere de $2\mu s$.

11.4.4. Lectura de la memoria EEPROM en C

En el Cuadro 35 se muestran los resultados obtenidos para la lectura de la memoria EEPROM en C:

Microcontrolador	Tiempo de lectura (μs)
PIC16F887	9.5
ATMega328P	2.25

Cuadro 35: Resultados obtenidos de la memoria EEPROM para lectura en C.

Como se muestra en el Cuadro 35, nuevamente, se observa que el PIC requiere de más tiempo para completar la lectura de datos de la memoria EEPROM que el ATMega, empleando $9.5\mu s$, mientras que el ATMega utiliza únicamente $2.25\mu s$. Esto significa que el PIC requiere de, aproximadamente, 4 veces más tiempo que el ATMega para la lectura de datos en la memoria EEPROM.

11.4.5. Escritura de la memoria EEPROM del ATMega utilizando la secuencia del PIC

A continuación, se muestran los resultados obtenidos de las mediciones de la escritura de la memoria EEPROM del ATMega, ajustando la secuencia de escritura para que sea equivalente a la indicada en la hoja de datos del PIC.

En el Cuadro 36, se muestran los resultados obtenidos:

Lenguaje	Tiempo de escritura (μs)
ASM	3500
C	3500

Cuadro 36: Resultados obtenidos de la memoria EEPROM para escritura del ATMega usando la secuencia del PIC.

11.4.6. Comparación de escritura y lectura de datos entre lenguajes de programación

A continuación, en los cuadros 37 y 38, se muestra la diferencia porcentual para la escritura y lectura de datos por microcontrolador en ambos lenguajes empleados para la programación de la memoria EEPROM, tomando como referencia el resultado obtenido en el lenguaje de ensamblador:

Microcontrolador	Ensamblador (μs)	C (μs)	Diferencia porcentual (%)
PIC16F887	3578	3578	0
ATMega328P	1.5	2	33.33
ATMega328P ajustado	3500	3500	0

Cuadro 37: Comparación entre ensamblador y C para la escritura de datos en la memoria EEPROM de ambos microcontroladores.

Microcontrolador	Ensamblador (μs)	C (μs)	Diferencia porcentual (%)
PIC16F887	7.5	9.5	26.66
ATMega328P	1.5	2.25	50

Cuadro 38: Comparación entre ensamblador y C para la lectura de datos en la memoria EEPROM de ambos microcontroladores.

11.5. Discusión

Como se puede observar en los cuadros 32, 33, 34 y 35, en todos los casos, el PIC presenta un mayor empleo de tiempo que el ATMega, siendo más notable en los casos de escritura de datos a la memoria EEPROM, como se muestra en los cuadros 32 y 34, obteniendo un tiempo de $3578\mu s$ tanto en ensamblador como en C, mientras que el ATMega emplea $1.5\mu s$ y 2.25 respectivamente.

Al comparar la escritura de datos en la memoria EEPROM, como se muestra en el Cuadro 37, el PIC tiene una diferencia porcentual de 0 % entre ensamblador y C, mientras que el ATMega tiene un 33.33 %. Por otro lado, al comparar la lectura de datos de la memoria EEPROM, como se muestra en el Cuadro 38, el PIC tiene una diferencia porcentual de 26.66 % entre ensamblador y C, mientras que el ATMega tiene un 50 %. Esto demuestra que el PIC, en porcentaje, varía menos que el ATMega al pasar de ensamblador a C tanto en la escritura como en la lectura de datos de la memoria EEPROM, sin embargo, en el caso de lectura de datos, el PIC aumenta su tiempo de lectura por $2\mu s$, mientras que el ATMega aumenta únicamente $0.75\mu s$. En el caso de escritura, el PIC no aumenta el tiempo y el ATMega varía únicamente en $0.5\mu s$.

Es necesario mencionar que los altos tiempos de escritura en el PIC se dan, principalmente, debido a la secuencia indicada en la hoja de datos en ambos microcontroladores, donde, en el PIC [13], la secuencia de escritura recomendada, requiere que se realice la comprobación de que la escritura se ha completado satisfactoriamente inmediatamente después de haber comenzado la escritura a la memoria EEPROM. Por otro lado, la comprobación de escritura a la memoria EEPROM en el ATMega [2] se realiza previo a comenzar una nueva escritura, es decir, el ATMega permite realizar una escritura y seguir con la programación indicada inmediatamente después, sin embargo, si se desea realizar una nueva escritura, entonces se comprueba si la escritura anterior se completó para proceder con la nueva escritura. Por este motivo, en el Cuadro 36, se muestran los datos para la escritura de la memoria EEPROM en el ATMega con la secuencia del PIC, obteniendo, así, una mejor comparación. Se observa que los tiempos de escritura del ATMega con la secuencia del PIC ascienden hasta $3500\mu s$, siendo inferior a los tiempos de escritura registrados en el PIC por $78\mu s$, lo cual indica que, con la secuencia del PIC, el ATMega sigue siendo más rápido que el PIC para la escritura de datos en la memoria EEPROM en ambos lenguajes.

CAPÍTULO 12

Módulos en conjunto - Reloj

A modo de integración de todos los módulos estudiados, se realizó una versión modificada del primer proyecto del curso de Programación de Microcontroladores de la Universidad del Valle de Guatemala [1].

La versión modificada consiste en la implementación de un reloj capaz de mostrar, en visualizadores de siete segmentos, los minutos y la hora del día en formato de 24 horas. Los contadores de minutos y horas deben ser modificables con botones de entrada en los microcontroladores. El reloj debe mostrar la configuración de una alarma y encenderla cuando la hora del reloj sea igual a la de la alarma. Finalmente, el reloj debe almacenar, en la memoria EEPROM, una configuración de alarma guardada por el usuario.

12.1. Programa

A continuación, en las figuras 41, 42, 43, 44 y 45, se muestra el pseudocódigo del programa implementado:

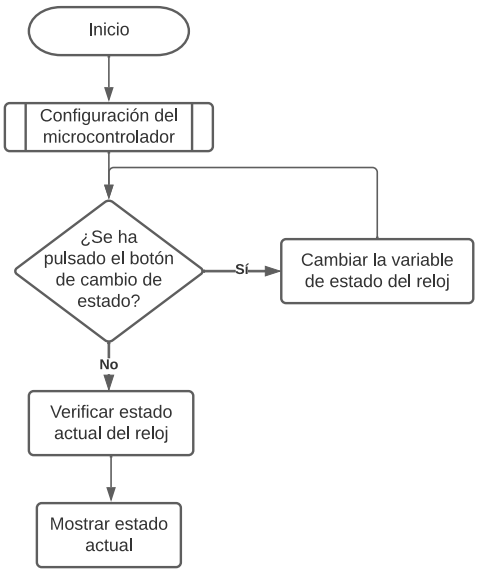


Figura 41: Inicio del programa del reloj.

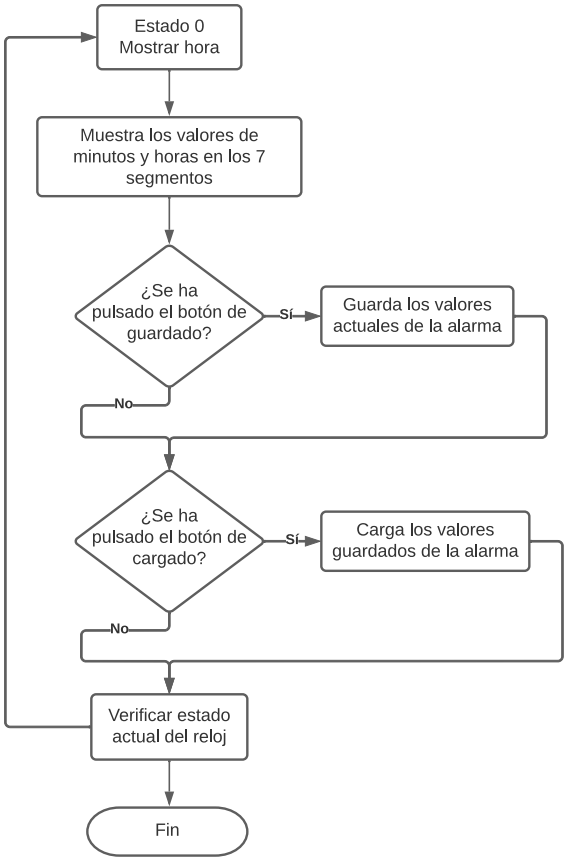


Figura 42: Estado 0 del programa del reloj.

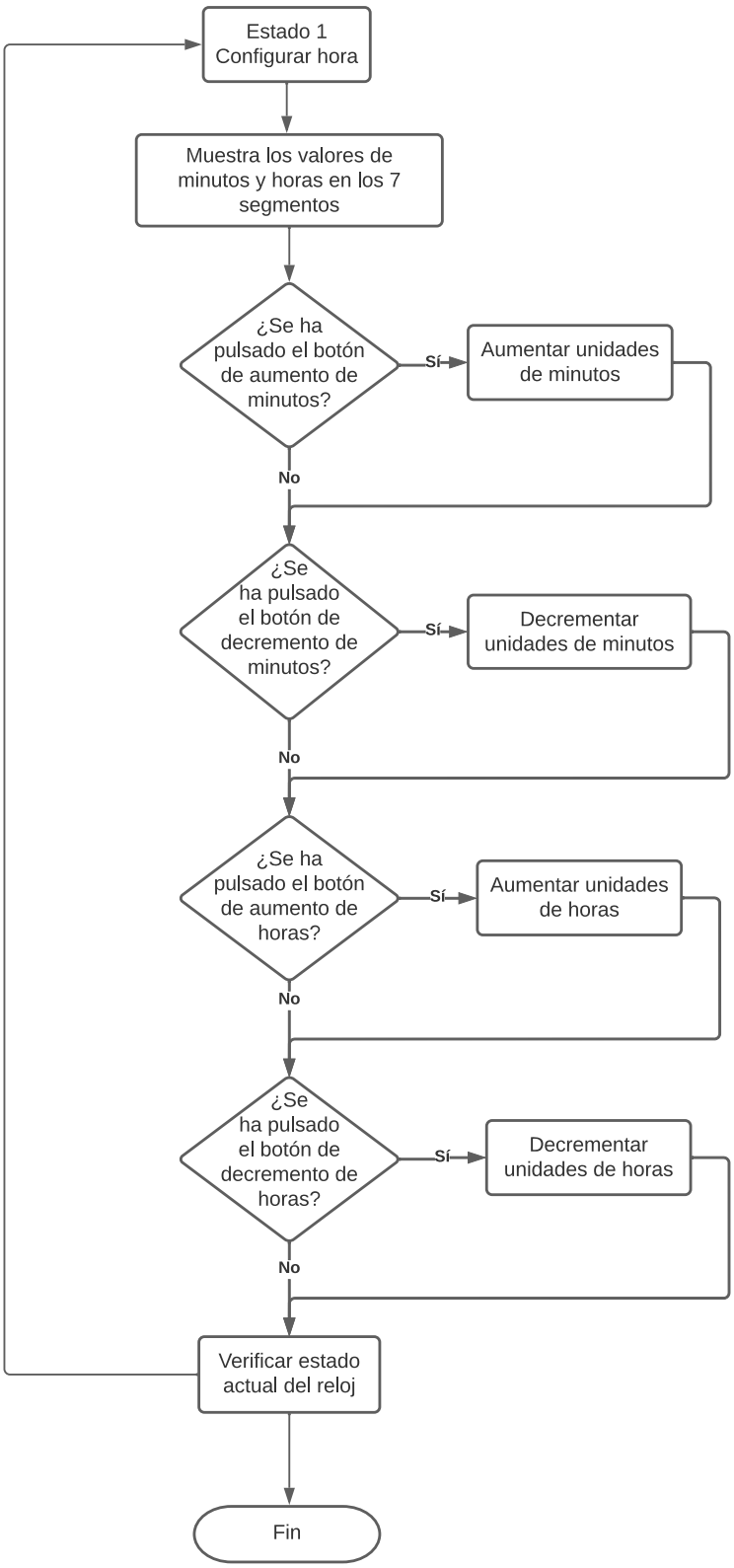


Figura 43: Estado 1 del programa del reloj.

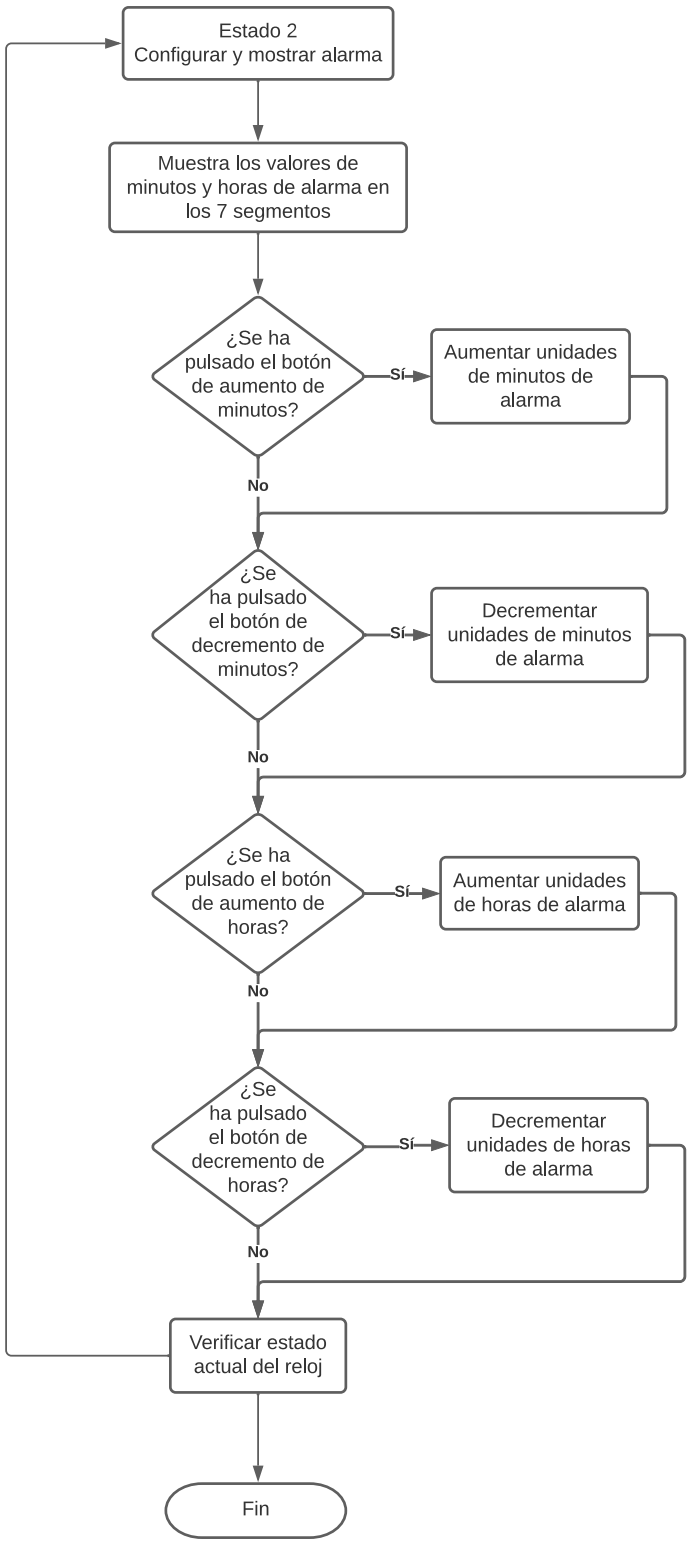


Figura 44: Estado 2 del programa del reloj.

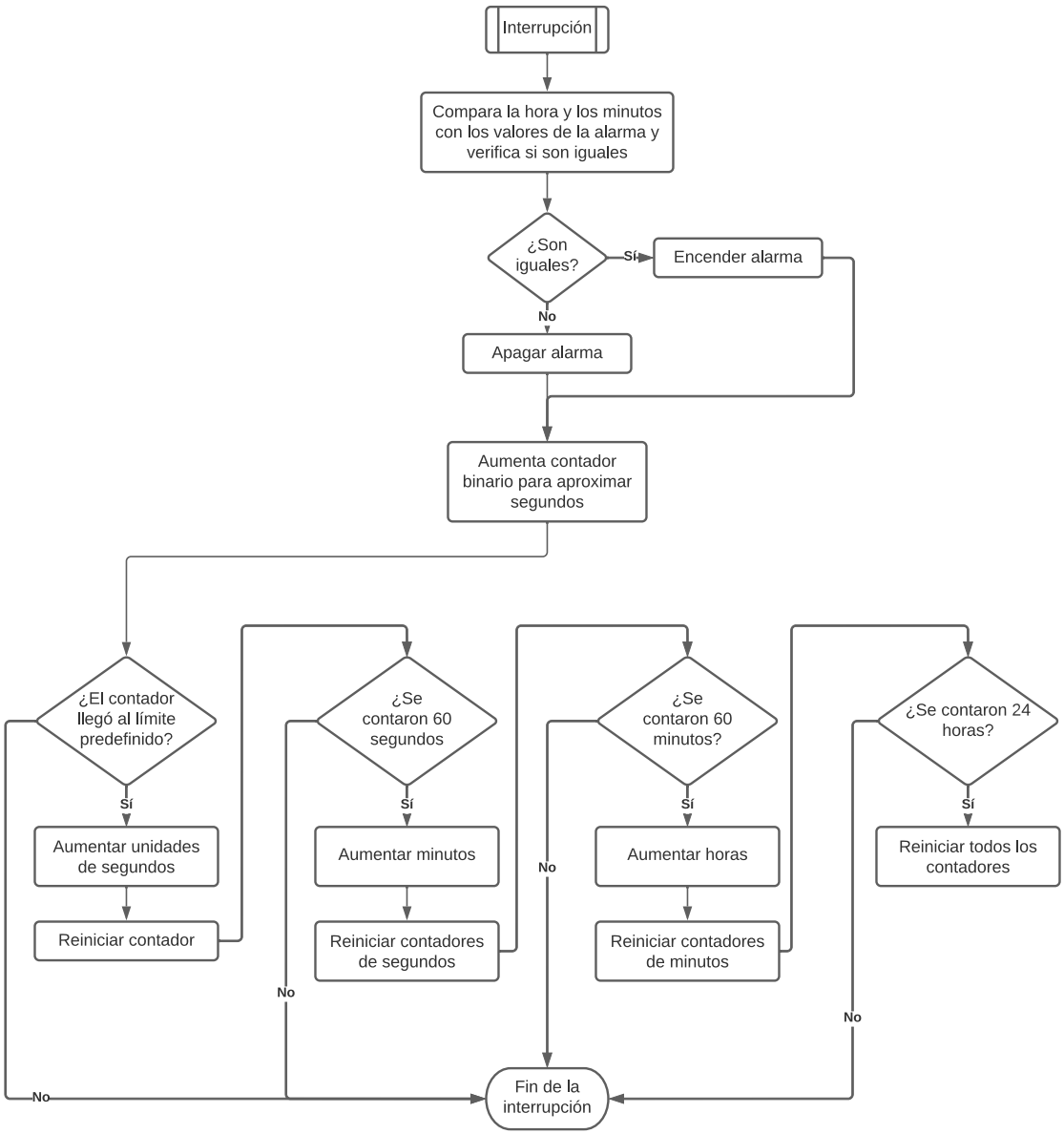


Figura 45: Interrupción del programa del reloj.

12.2. Esquemáticos

Los circuitos implementados se muestran en las figuras 46 y 47:

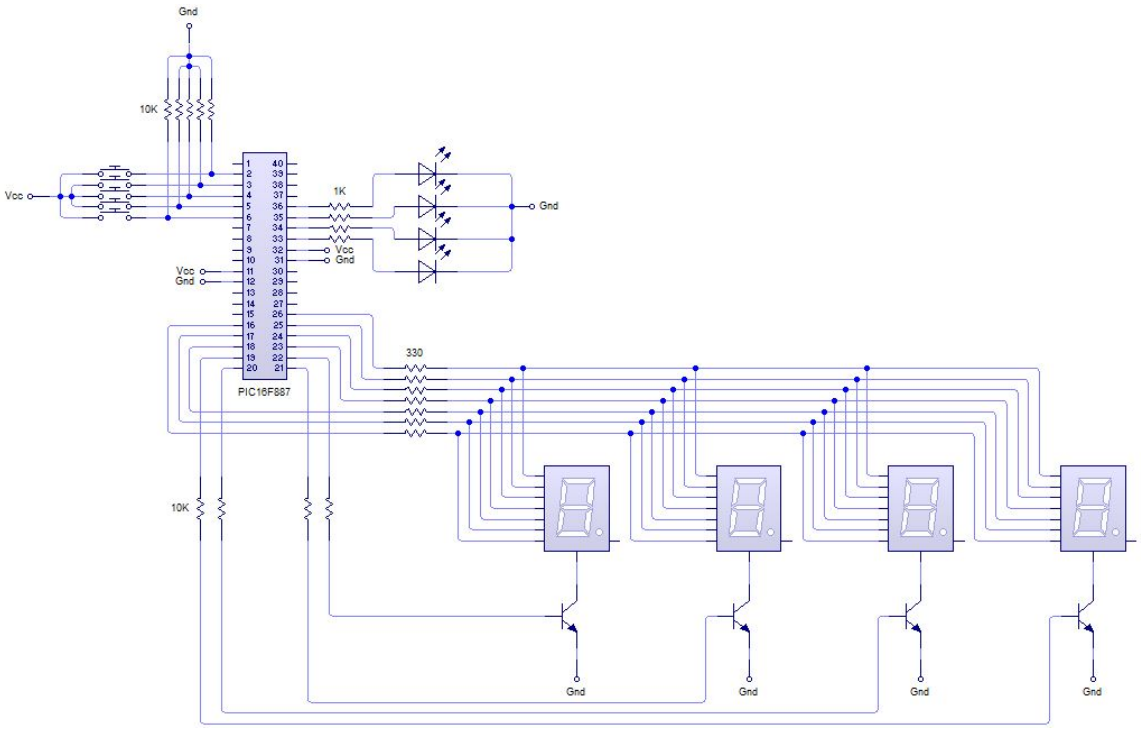


Figura 46: Circuito de la implementación del reloj en el PIC.

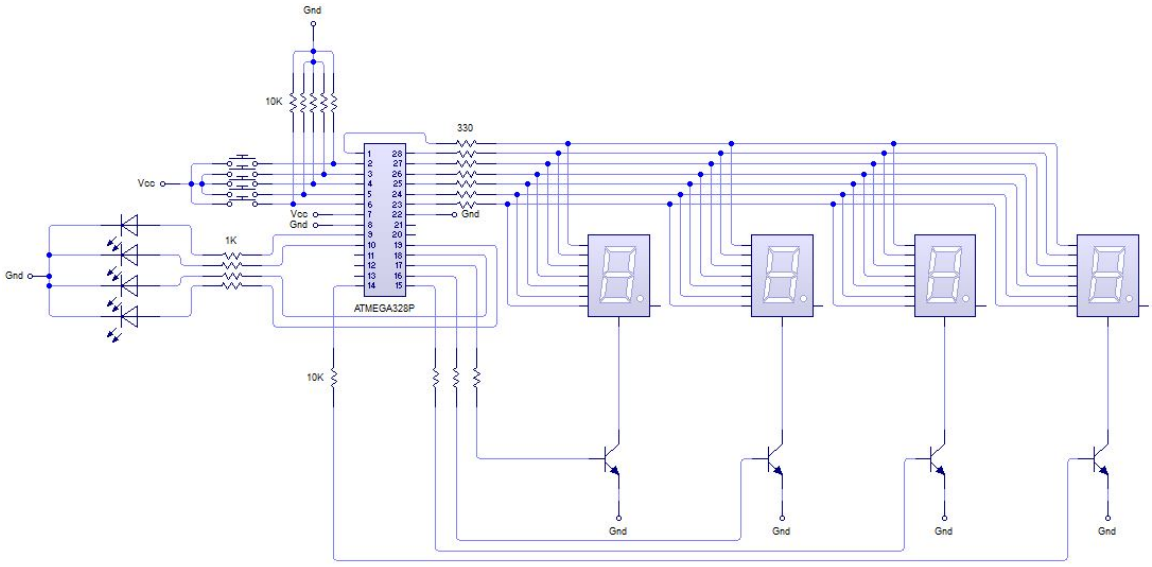


Figura 47: Circuito de la implementación del reloj en el ATMega.

12.3. Comparación de ventajas y desventajas entre los microcontroladores

A continuación, en el Cuadro 39, se muestran las ventajas y desventajas encontradas sobre la implementación del reloj en ambos microcontroladores:

	PIC16F887	ATMega328P
Ventajas	- Posee mayor cantidad de pines. Facilita la construcción de circuitos. - Los temporizadores que posee son más exactos y estables.	- Maneja múltiples vectores de interrupción. - Posee múltiples registros de trabajo. Vuelve más eficiente el manejo de variables.
Desventajas	- Posee un único vector de interrupción.	- Posee menor cantidad de pines y puertos, agravado por el pin C6, que es de uso exclusivo para el reinicio.

Cuadro 39: Ventajas y desventajas de la implementación del reloj en ambos microcontroladores.

Como se muestra en el Cuadro 39, para la implementación del reloj en ambos microcontroladores se observa que la cantidad de pines disponibles en ambos microcontroladores es un punto importante a tomar en cuenta. La cantidad de pines disponibles es una de las ventajas que posee el PIC respecto al ATMega debido a que facilita la construcción de circuitos, esto permite que la programación se facilite debido a que la distribución de componentes es más ligera en cada puerto disponible. Por otro lado, en el ATMega es necesario construir los circuitos y realizar la programación a modo de aprovechar al máximo cada uno de los pines disponibles debido a que son limitados, lo cual, inevitablemente, termina en un caso como el que se muestra en la Figura 47, donde se tienen puertos que cumplen con múltiples funciones.

Otro punto importante es el vector de interrupción. El PIC posee un único vector de interrupción, como se mencionó en el Capítulo 10, esto complica el uso del PIC debido a que cada vez que se ingresa al vector de interrupción, se debe verificar qué interrupción ha sido activada. Caso contrario es el ATMega, donde cada interrupción apunta a un único vector de interrupción y eso permite ahorrar tiempo y líneas de código en la verificación de las interrupciones.

Otra ventaja del PIC, es que posee temporizadores más exactos y estables, tal como se discute en el Capítulo 9, donde se demuestra esta eficiencia, lo cuál permite la implementación de un reloj con menor error respecto al tiempo real. Otra ventaja del ATMega es el manejo de múltiples registros de trabajo, mientras que en el PIC es necesario reservar espacios de memoria para guardar valores como variables y moverlos por el registro de trabajo W, en el ATMega, las variables son implementadas a través de sus registros de trabajo y pueden manipularse a conveniencia y de forma directa, sin necesidad de pasar por un registro general, lo cuál facilita la programación y el manejo de las variables requeridas en el reloj.

CAPÍTULO 13

Conclusiones

1. En el Capítulo 7, se mencionó que, para todos los casos, el ATMega328P resultó ser más eficiente al realizar llamados y retornos de funciones, por lo que, el ATMega328P es más rápido y eficiente en la manipulación del modulo de pila.

2. En el Capítulo 8, se demostró que, para el programa evaluado de entradas y salidas, el ATMega328P mostró resultados con menor tiempo de ejecución, por lo que, según las condiciones planteadas para la evaluación del módulo, el ATMega328P es más eficiente en la ejecución de operaciones en los pines de entrada y salida que el PIC16F887.

3. En el Capítulo 9, se discutió sobre el rendimiento de los temporizadores ambos microcontroladores. Se determinó que el rendimiento del PIC16F887 es más estable y aproximado al tiempo teórico esperado mostrado en la Ecuación 3, por lo que, el PIC16F887 tiene un mejor rendimiento bajo los parámetros de evaluación y condiciones de medición establecidos para el módulo de temporizadores.

4. En el Capítulo 10, se evaluó que el PIC16F887 requiere de una cantidad mayor de tiempo para la ejecución de entrada y salida de las interrupciones que el ATMega328P. También se comparó el aumento de tiempo dado por el cambio de lenguaje de ensamblador a C, cambio que al PIC16F887 le supone un aumento de más del 100 % tanto en entradas como en salidas de interrupciones, mientras que al ATMega328P no supone ninguna diferencia, por lo que, el ATMega328P, es más eficiente al entrar y salir de las interrupciones, requiriendo menos tiempo que el PIC16F887.

5. En el Capítulo 11, se comprobó la eficiencia de los microcontroladores evaluados al momento de escribir y leer datos de la memoria EEPROM. Se mostró que el tiempo requerido para la escritura y lectura de datos es considerablemente menor en el ATMega328P que en el PIC16F887, al utilizar las secuencias recomendadas por las hojas de datos de ambos microcontroladores. También se observó que, con la secuencia del PIC16F887, el ATMega328P aumenta considerablemente su tiempo de escritura a la memoria EEPROM, acercándose a los tiempos medidos en el PIC16F887. Sin embargo, aún bajo estas condiciones, el ATMega328P resulta ser más rápido en la escritura

de datos, por lo que, bajo estos parámetros de evaluación, el ATMega328P es más eficiente para leer y escribir datos a la memoria EEPROM.

CAPÍTULO 14

Recomendaciones

1. El estudio comparativo se realizó empleando los osciladores internos de ambos microcontroladores. Se recomienda emplear osciladores externos para añadir otra capa de profundidad al estudio y observar posibles diferencias con los resultados mostrados en el presente trabajo.

2. El estudio abarcó únicamente la comparación del tiempo de ejecución para diversos programas planteados en ambos microcontroladores. Se recomienda analizar otras capacidades de los microcontroladores estudiados, tales como manejo interno de memoria o rendimiento a distintas temperaturas.

3. En el Capítulo 7 se optó por utilizar la capacidad máxima de almacenamiento de pila del PIC16F887 tanto en ensamblador como en C. Se recomienda realizar las pruebas necesarias para comprobar los posibles efectos de usar más niveles de almacenamiento que los disponibles en el PIC16F887 y verificar, con programas similares, cuál puede llegar a ser un aproximado del máximo almacenamiento de pila en el ATMega328P.

4. En el Capítulo 8, se programaron los microcontroladores para hacer uso de un único puerto para ambos contadores binarios de 4 bits. Se recomienda aprovechar las capacidades físicas del PIC16F887, evaluando su rendimiento al usar más puertos para distribuir de mejor forma los contadores binarios y realizar un programa más óptimo en este microcontrolador.

5. En el Capítulo 9 se analizó únicamente uno de los temporizadores disponibles en ambos microcontroladores. Se recomienda realizar un estudio similar haciendo uso del resto de temporizadores disponibles en los microcontroladores con el fin de complementar la recopilación de datos realizada.

6. En el Capítulo 11, se utilizaron, únicamente, las secuencias recomendadas por las hojas de datos de ambos microcontroladores [13] y [2], tanto para la escritura como para la lectura de datos de la memoria EEPROM. Se recomienda evaluar el rendimiento de ambos microcontroladores utilizando interrupciones para el manejo de la memoria EEPROM.

CAPÍTULO 15

Bibliografía

[1] U. del Valle de Guatemala, *Programación de microcontroladores*, 2019.

[2] Microchip, "mega AVR Data Sheet," 2020.

[3] ——, "A comparison of 8-Bit Microcontrollers," 1997.

[4] L. Fried. "PIC vs. AVR: Ultimate fight!" (2012), dirección: http://www.ladyada.net/library/picvsavr.html.

[5] S. Kumar, *Course File Microcontrollers*, 2016. dirección: https://www.mitmuzaffarpur.org/wp-content/uploads/2018/08/MICROCONTROLLER.pdf.

[6] F. Ansovini, *Microcontroller Programming - Syllabus*. dirección: http://www.phd-stiet.diten.unige.it/images/Documents/MicrocontrollerProgramming.pdf.

[7] N. University. "Microprocessor System Design." (2019), dirección: https://canvas.northwestern.edu/courses/92393.

[8] U. del Valle de Guatemala, *Introducción a la Ingeniería Electrónica y Mecatrónica*, 2018.

[9] ——, *Electrónica Digital 2*, 2021.

[10] G. Bettina. "Introduction to Microcontrollers." (2007), dirección: https://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf.

[11] J. Sanchez y M. Canton, *Microcontroller Programming - The Microchip PIC*. CRC Press, 2007.

[12] Microchip. "PIC16F887." (2022), dirección: https://www.microchip.com/en-us/product/PIC16F887.

[13] ——, "28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers," 2012.

[14] ——, "ATmega328P." (2022), dirección: https://www.microchip.com/en-us/product/ATmega328P.

[15] Arduino, *UNO R3*. dirección: https://docs.arduino.cc/hardware/uno-rev3 (visitado 28-04-2022).

[16] M. C. Romero, *Lenguaje de programación C*. 2015.

[17] J. Foxworth, *How to program a microcontroller*. dirección: https://www.egr.msu.edu/classes/ece480/capstone/spring15/group13/assets/app_note_john_foxworth.docx.pdf (visitado 05-05-2022).

[18] I. Zhirkov, *Low-Level Programming*. Apress, 2017.

[19] Saleae, "Saleae Logic Pro 16 USB Logic Analyzer," 2022.

[20] L. J. Aguilar, *Programación en C++*. McGrawHill, 2006.

CAPÍTULO 16

Anexos

16.1. Repositorio

A continuación se presenta el enlace al repositorio manejado en el estudio realizado:

https://github.com/javierschwendener/AVR_vs_PIC

CAPÍTULO 17

Glosario

anidadas: Se dice que una función (o estructura) es anidada cuando contiene otra función. Permiten ahorrar tiempo en la escritura de programas [20]. 13

EEPROM: Por sus siglas en inglés *Electrically-Erasable Programmable Read-Only Memory*, es un almacenamiento no volátil en los microcontroladores. 50