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,

2022

Vo.Bo.:

(f) _____

Dr. Luis Alberto Rivera Estrada

Tribunal Examinador:

(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

MSc. Carlos Esquit

(f) _____

Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, de de 2022.

Índice

Lista de figuras

Lista de cuadros

Resumen

En la presente investigación se compara el desempeño de los microcontroladores PIC16F887 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.

CAPÍTULO 1

Introducción

Introducción

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.

Finalmente, el alcance del curso de programación de microcontroladores de la Universidad del Valle de Guatemala abarca el aprendizaje en entorno ensamblador y abarca el manejo de los bancos de memoria, temporizadores, lectura de entradas digitales y analógicas, comunicaciones seriales, y la implementación de procesos en paralelo.

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, bajo el contexto de Guatemala, 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 en Biomédica.

CAPÍTULO 4

Objetivos

4.1. Objetivo General

Realizar un estudio comparativo entre los microcontroladores PIC16F887 y ATmega328P, 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

Marco teórico

5.1. Generalidades de un microcontrolador

Según Bettina [8], 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 [9].

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 [9]. Entre los periféricos más comunes se encuentran los temporizadores (*timers*), contadores, conversiones analógico a digital (por sus siglas en inglés ADC) y comunicaciones seriales [8].

5.2. 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 [9].

5.3. 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 [9].

5.4. Interrupciones

Una interrupción es una señal asíncrona, que funciona como mecanismo para evitar pérdidas del tiempo de procesador de los microcontroladores [9]. Las interrupciones pueden ser activadas por tanto por banderas internas del microcontrolador o 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.

5.5. 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 [9].

5.6. Memoria EEPROM

Por sus siglas en inglés *Electrically-Erasable Programmable Read-Only Memory*, es utilizada como un almacenamiento no volatil, diferenciándose de la memoria RAM y permitiendo el almacenamiento de datos aún cuando la energía es removida [9].

5.7. PIC16F887

El PIC16F887 [10], 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.

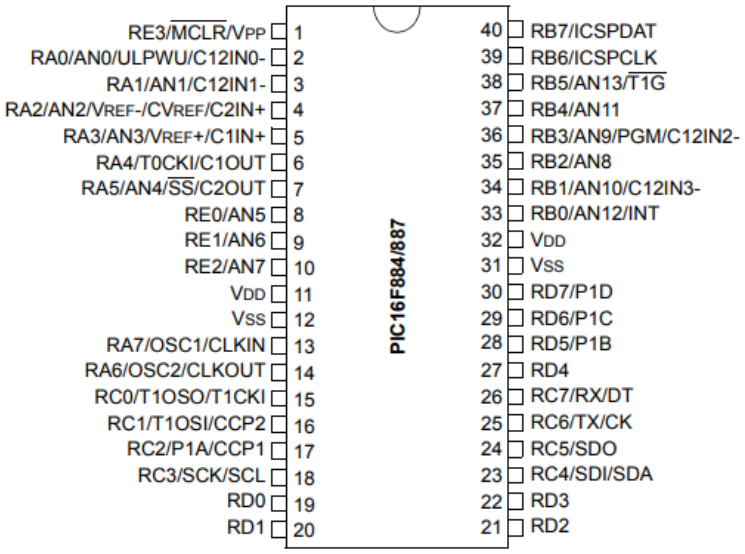

Figura 1: Asignación de pines del PIC16F887

5.8. ATmega328P

El ATmega328P [11], 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 [12].

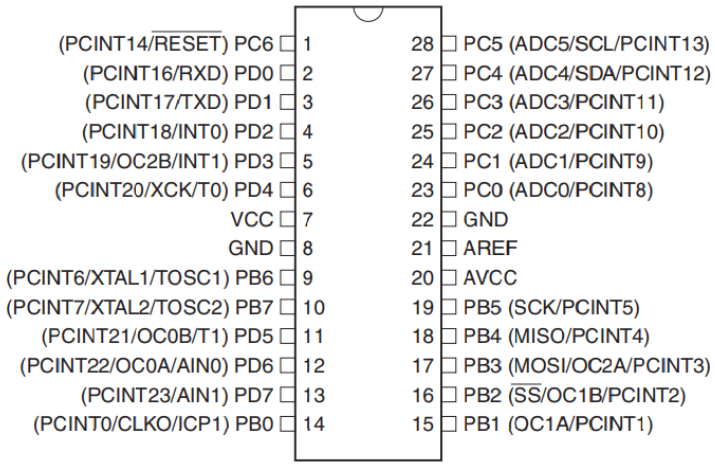

Figura 2: Asignación de pines del ATMega328P

5.9. 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++ o Java [13]. 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 entener y ejecutar [8].

5.10. 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*) [13].

5.11. 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 [8]. 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.

CAPÍTULO 6

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.

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 siguiente capítulo, se profundizará en el estudio de las interrupciones de los microcontroladores.

Para medir los resultados, se utilizó el Analizador Lógico con 4 millones de mediciones por segundo, 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.

6.1. Registros del módulo

Microcontrolador	Registro
PIC16F887	OPTION_REG TMR0 INTCON
ATmega328P	TCNT0 TCCR0A TCCR0B TIMSK0

Cuadro 1: Registros de los Temporizadores

6.2. Esquemáticos

Los circuitos implementados fueron los siguientes:

6.3. 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 se muestra un segmento de captura obtenido del Analizador Lógico empleado con la configuración antes mencionada y un segmento de las tablas formadas en Excel a partir de los datos obtenidos (para más datos, referirse a los anexos):

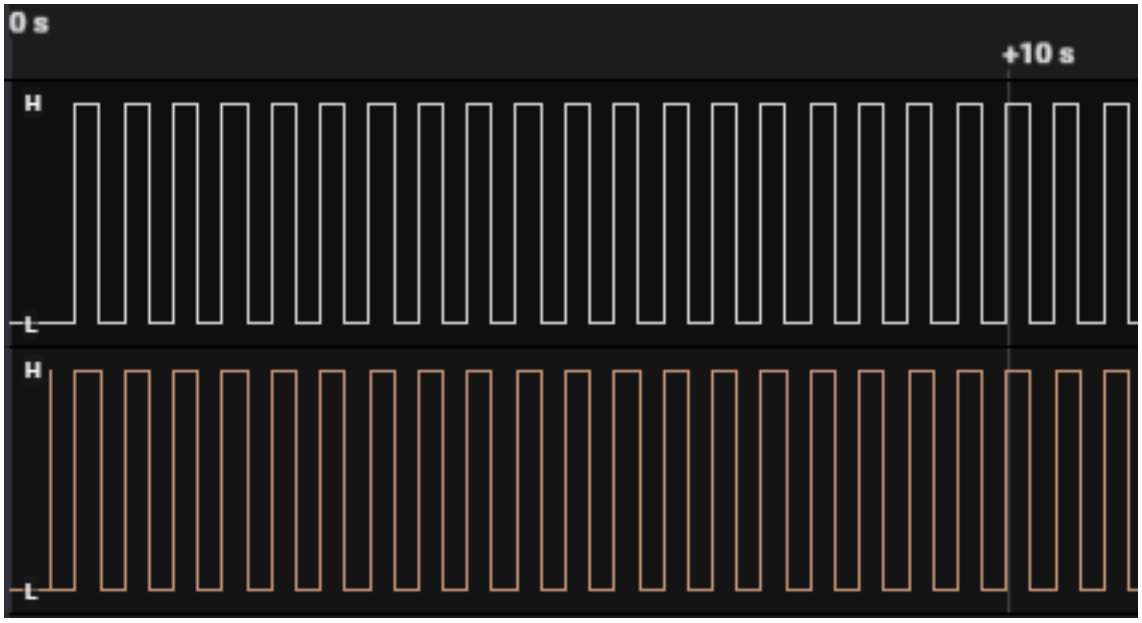

Figura 3: Segmento de captura del temporizador

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 4: 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:

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

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:

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

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:

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

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:

$$T = 1024 * (256 - 128) * \frac{1}{8MHz} * 15 = 0.24576seg = 245.76mseg \tag{4}$$

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

6.3.1. Temporizador sin interrupciones en ensamblador

De las 1205 muestras de este programa, se obtuvo la siguiente estadística:

Estadística	PIC16F887	ATMega328P
Promedio	245.160	244.928
Valor máximo	245.256	245.234
Valor mínimo	245.104	244.545
Varianza	0.153	0.689
Desviación Estándar	0.026	0.095

Cuadro 2: 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 siguientes porcentajes de error:

Microcontrolador	Promedio	%E
PIC16F887	245.160	0.24 %
ATMega328P	244.928	0.34 %

Cuadro 3: Porcentaje de error para cada microcontrolador del temporizador sin interrupciones en ensamblador

Dadas las muestras, se obtienen las siguientes gráficas:

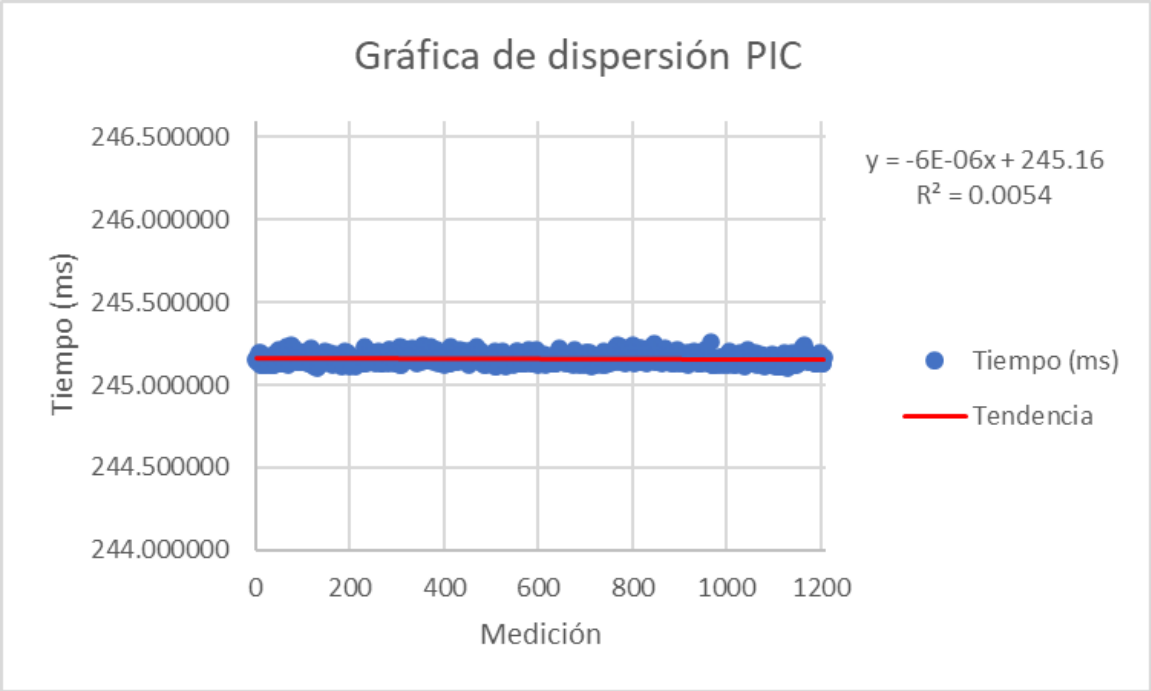

Figura 5: Gráfica de dispersión del temporizador sin interrupciones en ensamblador del PIC16F887

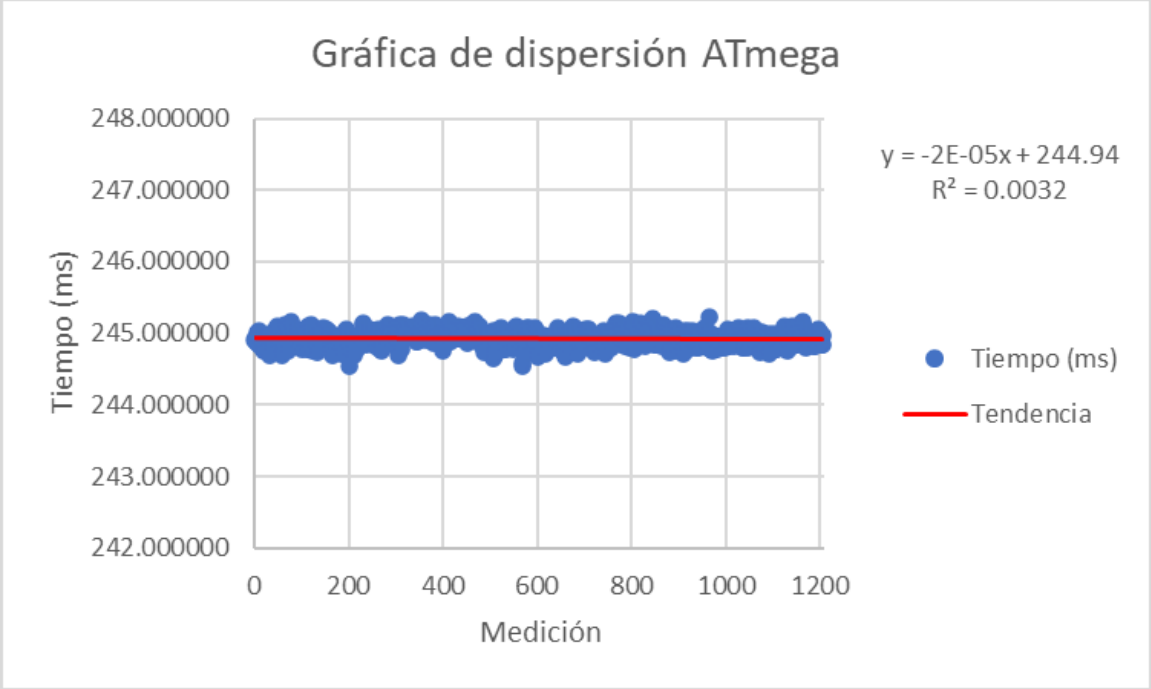

Figura 6: Gráfica de dispersión del temporizador sin interrupciones en ensamblador del ATmega328P

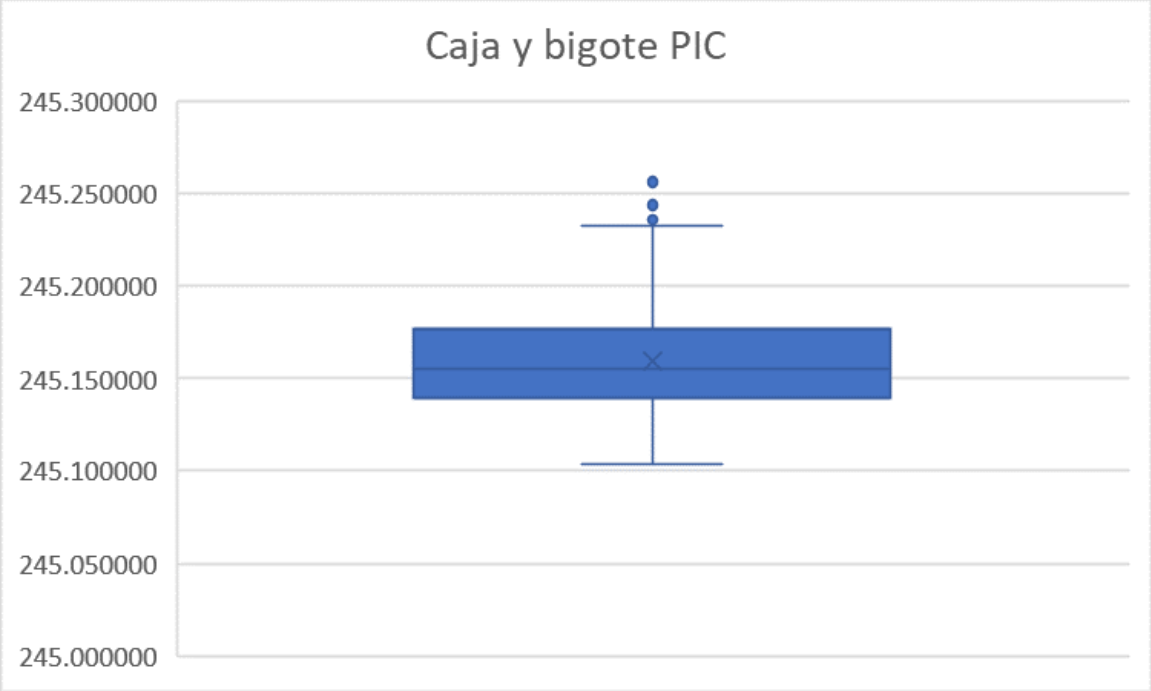

Figura 7: Diagrama de caja y bigote del temporizador sin interrupciones en ensamblador del PIC16F887

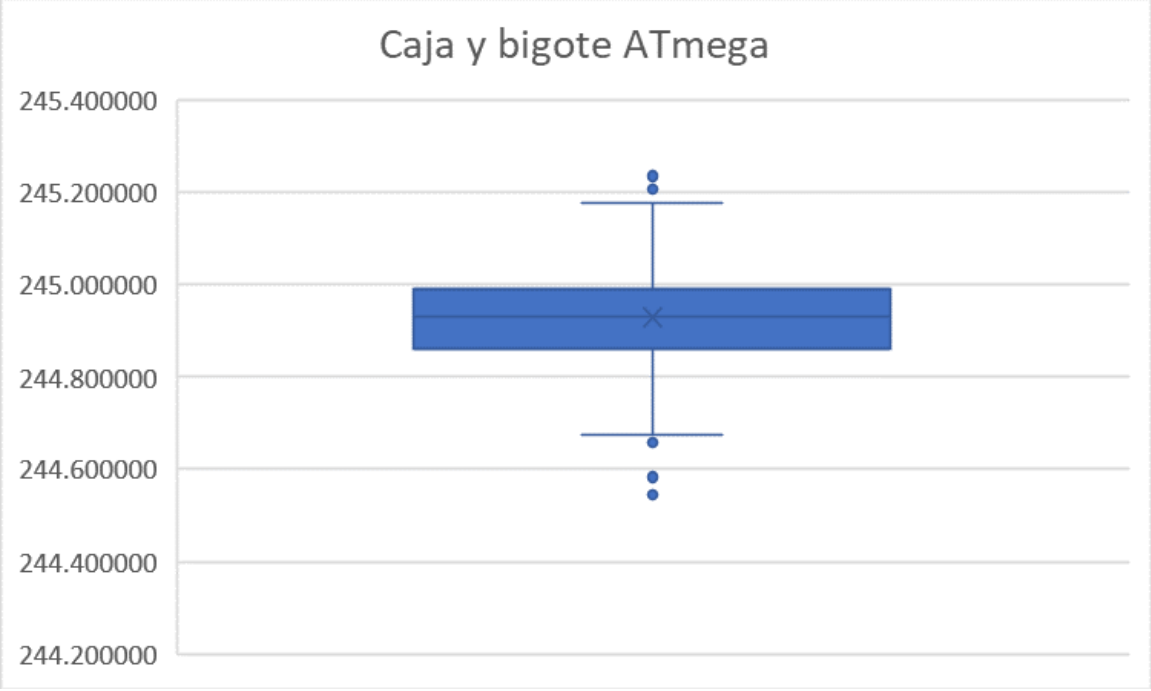

Figura 8: Diagrama de caja y bigote del temporizador sin interrupciones en ensamblador del ATMega328P

De los diagramas de caja y bigote, se obtiene la cantidad de datos atípicos para cada

microcontrolador:

6.3.2. Temporizador sin interrupciones en C

De las 1205 muestras de este programa, se obtuvo la siguiente estadística:

Estadística	PIC16F887	ATMega328P
Promedio	245.119	244.420
Valor máximo	245.169	244.777
Valor mínimo	245.051	243.911
Varianza	0.118	0.866
Desviación Estándar	0.020	0.132

Cuadro 4: 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 siguientes porcentajes de error:

Microcontrolador	Promedio	%E
PIC16F887	245.119	0.26 %
ATMega328P	244.420	0.55 %

Cuadro 5: Porcentaje de error para cada microcontrolador del temporizador sin interrupciones en C

Dadas las muestras, se obtienen las siguientes gráficas:

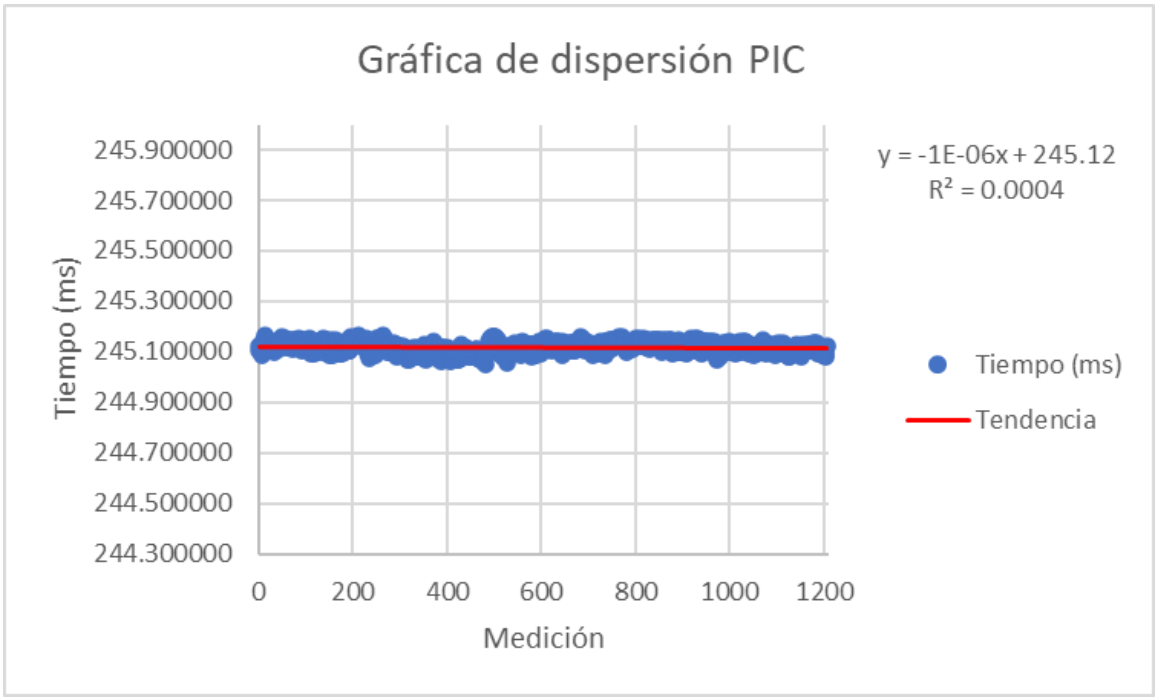

Figura 9: Gráfica de dispersión del temporizador sin interrupciones en C del PIC16F887

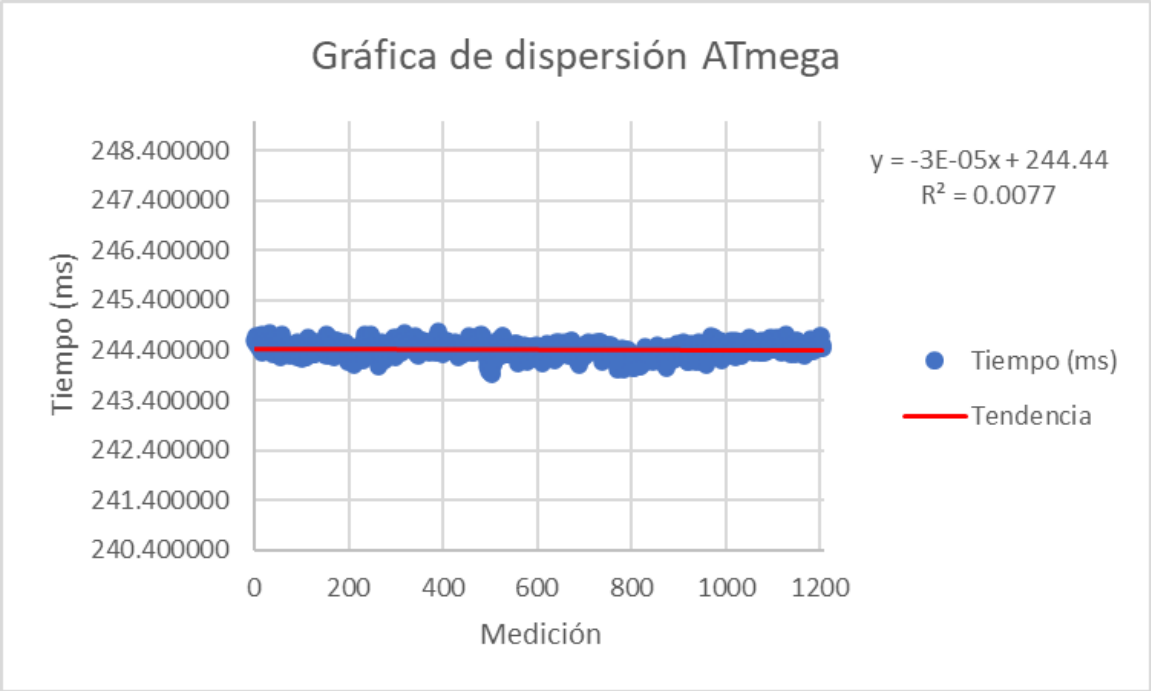

Figura 10: Gráfica de dispersión del temporizador sin interrupciones en C del ATmega328P

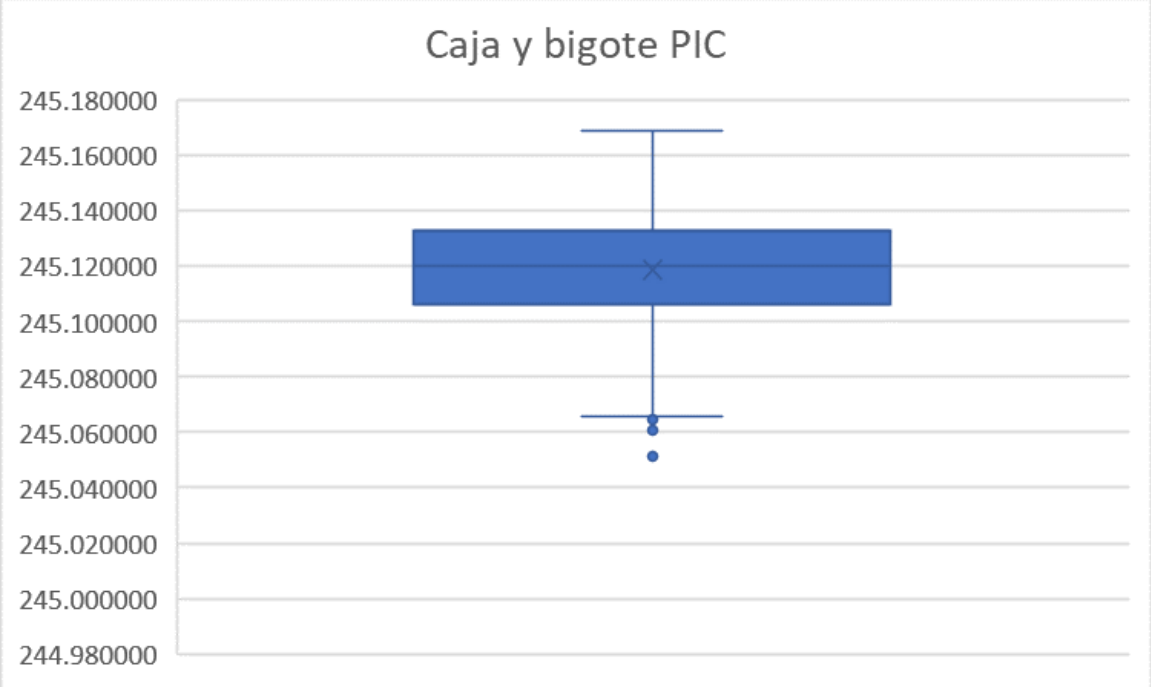

Figura 11: Diagrama de caja y bigote del temporizador sin interrupciones en C del PIC16F887

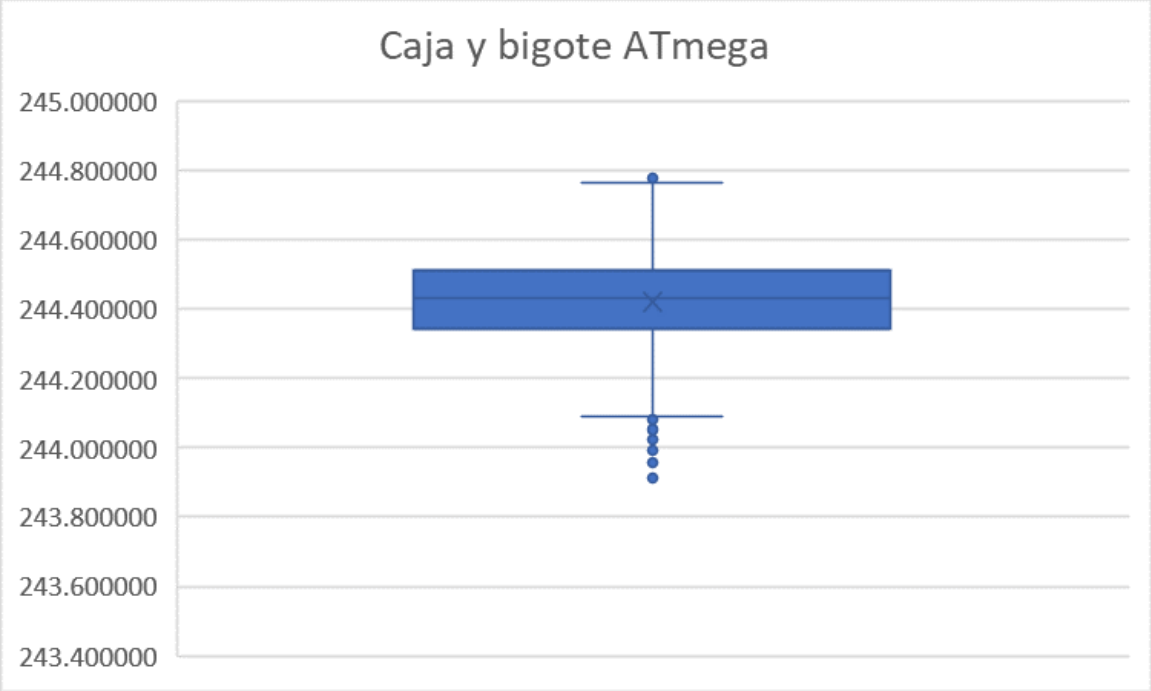

Figura 12: Diagrama de caja y bigote del temporizador sin interrupciones en C del ATMega328P

6.3.3. Temporizador con interrupciones en ensamblador

De las 1205 muestras de este programa, se obtuvo la siguiente estadística:

Estadística	PIC16F887	ATMega328P
Promedio	245.215	245.032
Valor máximo	245.287	245.318
Valor mínimo	245.157	244.694
Varianza	0.130	0.625
Desviación Estándar	0.026	0.102

Cuadro 6: 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 siguientes porcentajes de error:

Microcontrolador	Promedio	%E
PIC16F887	245.215	0.22 %
ATMega328P	245.032	0.30 %

Cuadro 7: Porcentaje de error para cada microcontrolador del temporizador con interrupciones en ensamblador

Dadas las muestras, se obtienen las siguientes gráficas:

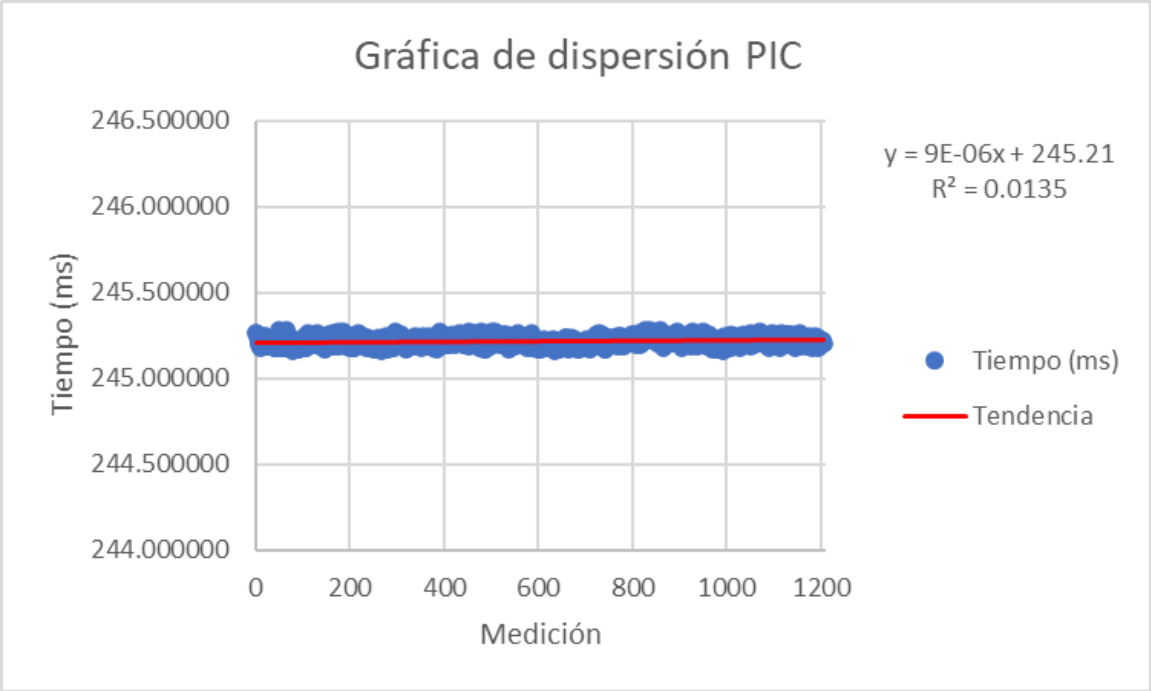

Figura 13: Gráfica de dispersión del temporizador con interrupciones en ensamblador del PIC16F887

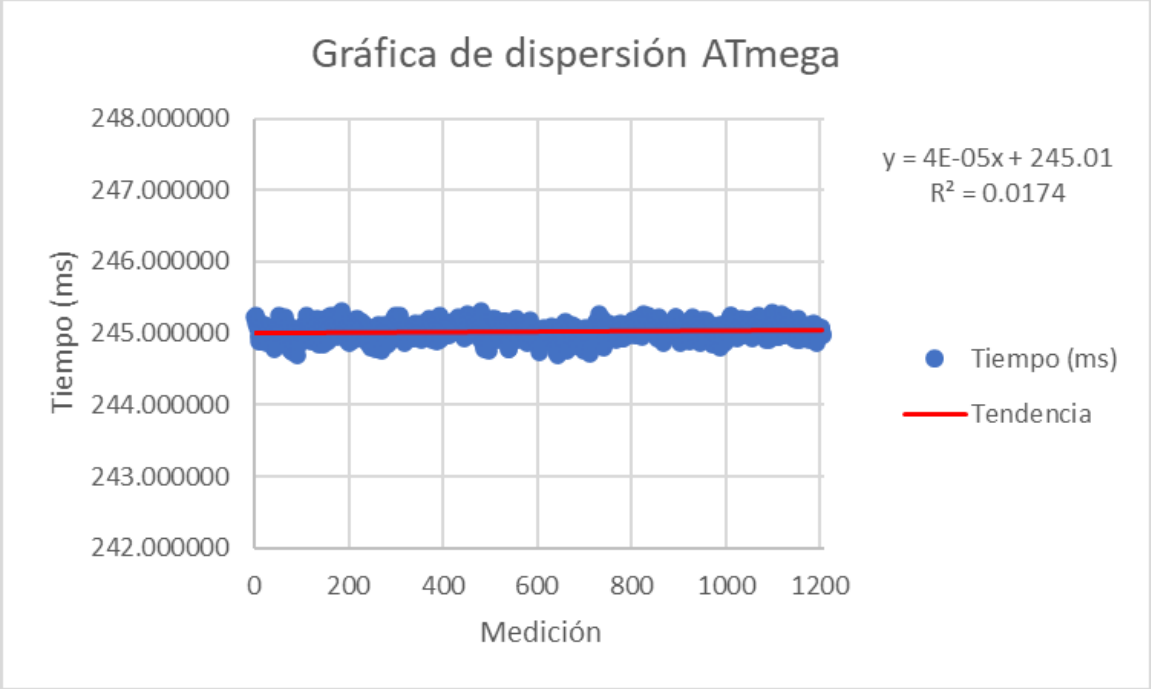

Figura 14: Gráfica de dispersión del temporizador con interrupciones en ensamblador del ATmega328P

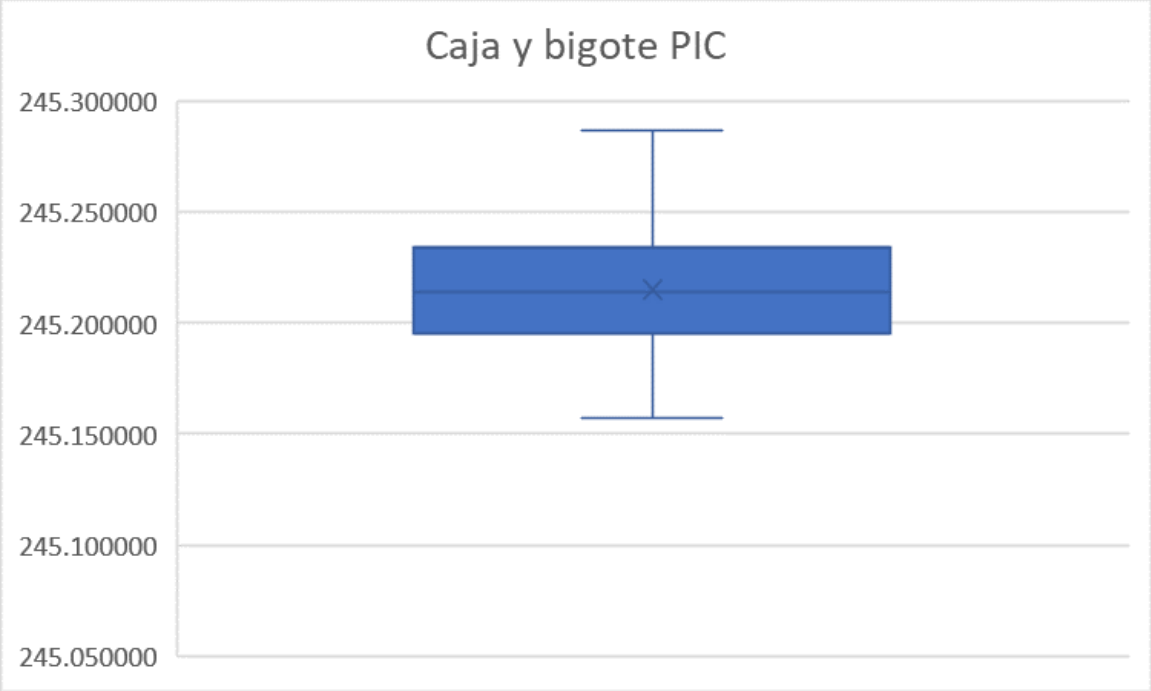

Figura 15: Diagrama de caja y bigote del temporizador con interrupciones en ensamblador del PIC16F887

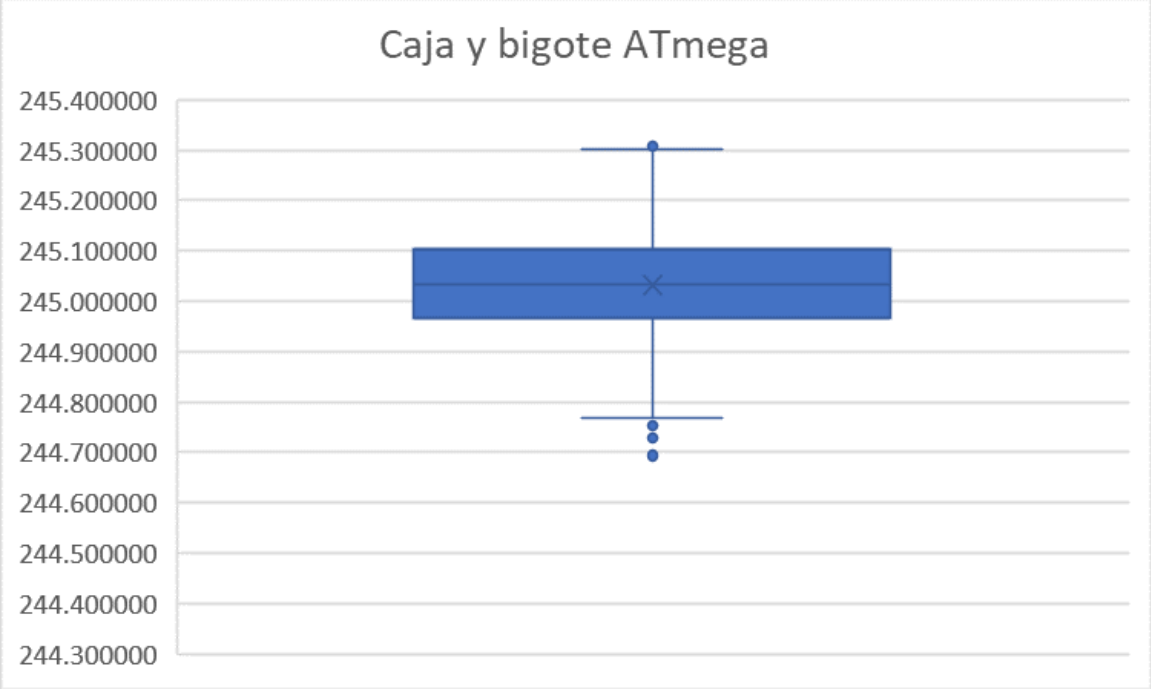

Figura 16: Diagrama de caja y bigote del temporizador con interrupciones en ensamblador del ATMega328P

6.3.4. Temporizador con interrupciones en C

De las 1205 muestras de este programa, se obtuvo la siguiente estadística:

Estadística	PIC16F887	ATMega328P
Promedio	245.335	245.075
Valor máximo	245.396	245.487
Valor mínimo	245.258	244.701
Varianza	0.138	0.786
Desviación Estándar	0.026	0.105

Cuadro 8: 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 siguientes porcentajes de error:

Microcontrolador	Promedio	%E
PIC16F887	245.335	0.17 %
ATMega328P	245.075	0.28 %

Cuadro 9: Porcentaje de error para cada microcontrolador del temporizador con interrupciones en C

Dadas las muestras, se obtienen las siguientes gráficas:

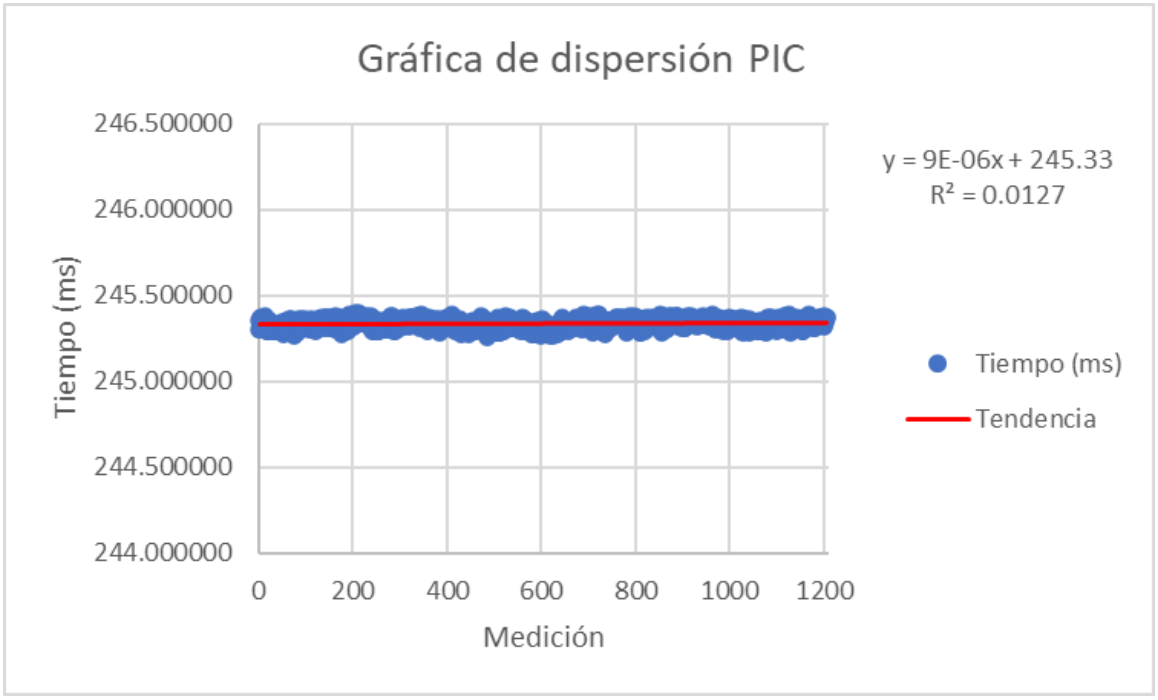

Figura 17: Gráfica de dispersión del temporizador con interrupciones en C del PIC16F887

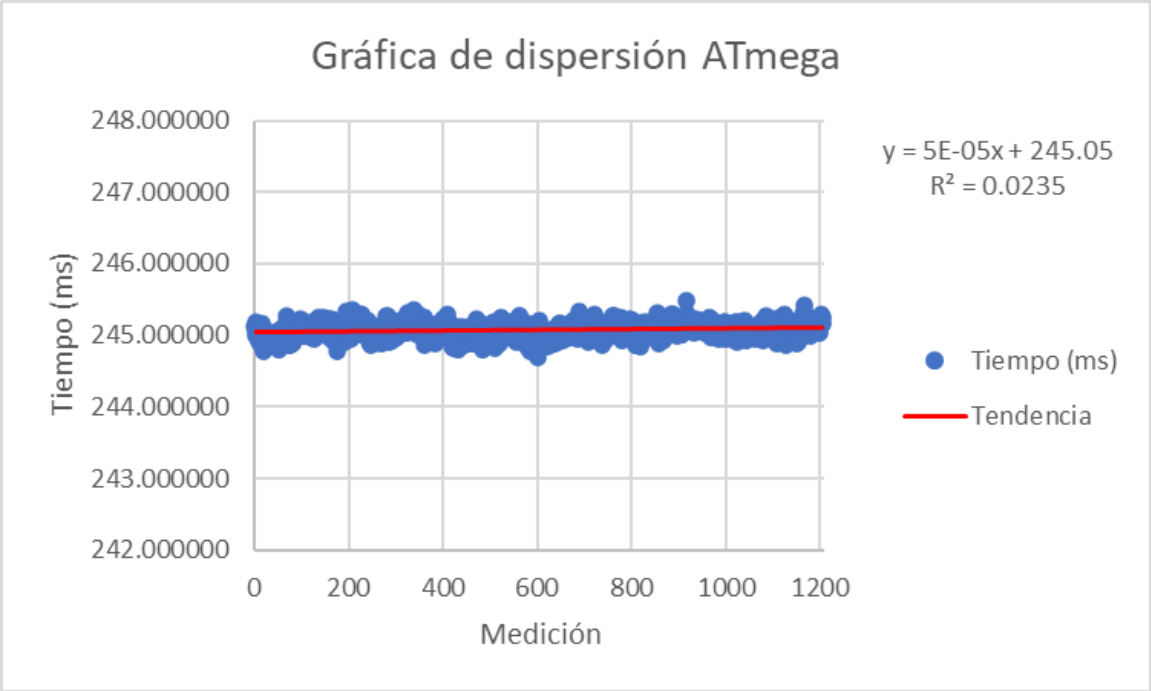

Figura 18: Gráfica de dispersión del temporizador con interrupciones en C del ATmega328P

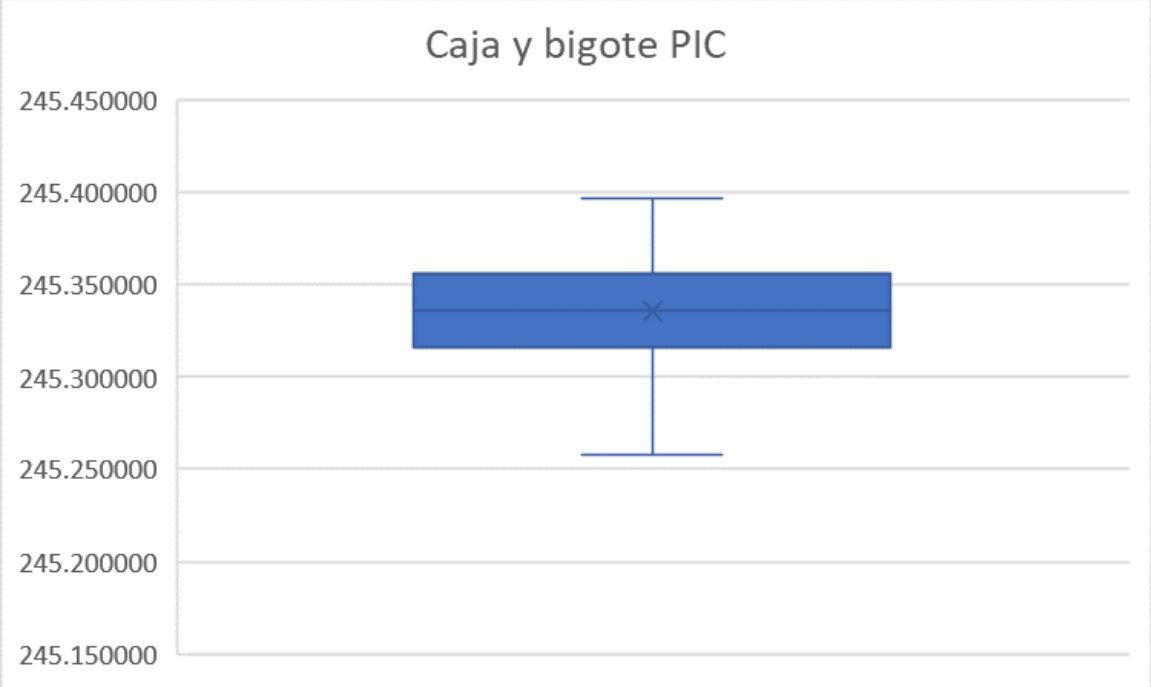

Figura 19: Diagrama de caja y bigote del temporizador con interrupciones en C del PIC16F887

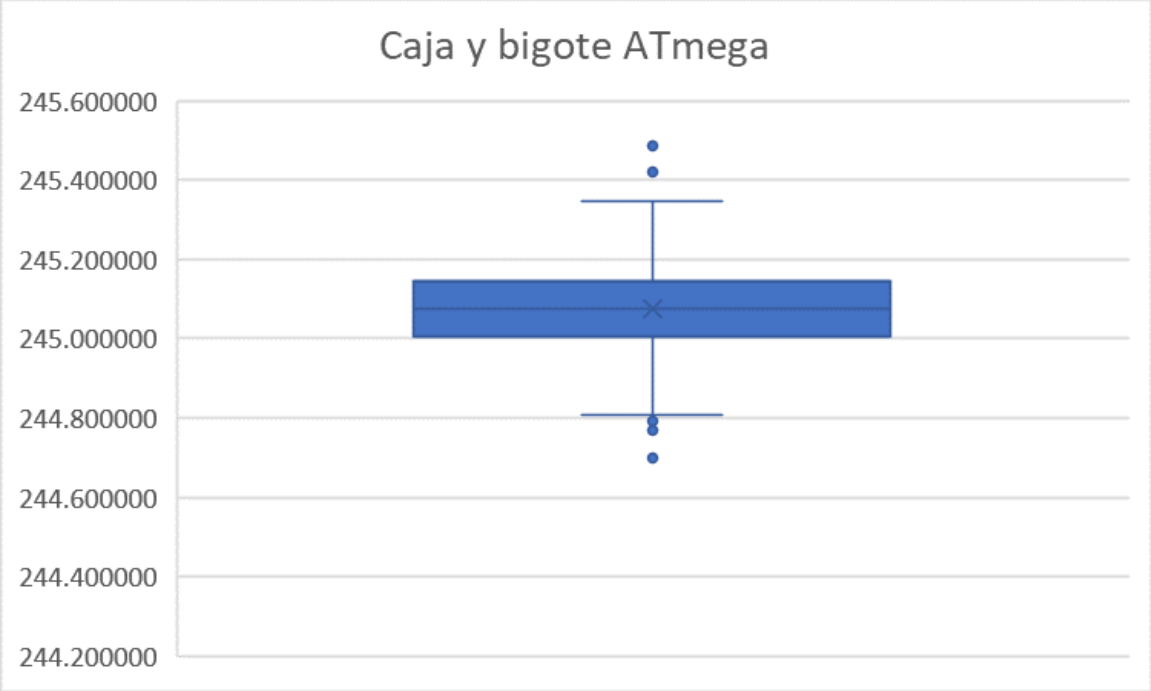

Figura 20: Diagrama de caja y bigote del temporizador con interrupciones en C del ATMega328P

6.4. Discusión

En el caso de temporizador sin interrupciones en ensamblador, se puede observar en el cuadro 2 que el promedio de valores presentado por el PIC16F887 es más aproximado al tiempo estimado que el presentado por el ATMega328P, como se puede observar en el cuadro 3. 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 5 y 6, donde la pendiente de la línea de tendencia del PIC es de $-6E-06$, mientras que la del ATMega es de $-2E-05$, 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 7 y 8, 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 4, 6 y 6, donde las varianzas de datos del PIC son considerablemente menores a las del ATMega, en los cuadros 5, 7 y 9, donde los porcentajes de error presentados por el PIC son menores a los del ATMega, en las figuras 9, 10, 13, 14, 17 y 18, donde las pendientes de las líneas de tendencia del PIC son menores a las del ATMega, y en las figuras 11, 12, 15, 16, 19 y 20, donde las concentraciones de datos del PIC y la cantidad de datos atípicos son menores a las del ATMega.

CAPÍTULO 7

Conclusiones

En el capítulo de Temporizadores, se discutió sobre el rendimiento de ambos microcontroladores al ejecutar programas referentes a este módulo. Se determinó que el rendimiento que el PIC16F887 no solo es más estable, sino que también es más aproximado al tiempo esperado, por lo que se puede concluir que, en este módulo, el PIC16F887 tiene un mejor rendimiento bajo los parámetros de evaluación y condiciones de medición establecidos.

CAPÍTULO 8

Recomendaciones

En el capítulo de Temporizadores 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.

CAPÍTULO 9

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

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

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

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

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

[13] 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).