CONTROLADOR CON LÒGICAFUZZY BASADO EN MICROCONTROLADOR Y LABVIEW

JAIME ELIÉCER VILLALOBOS BARRERA

JORGE ELIÉCER RODRIGUEZ GOMEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2004

CONTROLADOR CON LÒGICAFUZZY BASADO EN MICROCONTROLADOR Y LABVIEW

Proyecto de grado para optar al título de Ingeniero Electrónico

JAIME ELIÉCER VILLALOBOS BARRERA

Proyecto de grado para optar al título de Ingeniero Eléctrico

JORGE ELIÉCER RODRIGUEZ GOMEZ

Director

MPE. JAIME GUILLERMO BARRERO PEREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2004

AGRADECIMIENTOS

A Dios, por la vida y la oportunidad de ser quien soy.

A mis padres Roque y Betsabé,
Mis hermanos José Luis, Gloria y Olga Lucía,
y a mi tía María del Carmen.

A Claudia Patricia, mi amor
y a mi hija, Anggie Valeria, mi ilusión más grande.

A mis amigos.

Jorge Eliécer Rodríguez Gómez

A mi padre Eliécer Villalobos, a mi madre
Mariela por tenerme gran fe y paciencia en todo este tiempo a pesar de los momentos difíciles que pase muchas gracias

A mis hermanos Edith, Diana y Carlos por su apoyo y a mis tíos especialmente Sara

A Dios.

Jaime Eliécer

TITULO: CONTROLADOR CON LÒGICAFUZZY BASADO EN MICROCONTROLADOR Y LABVIEW[*]

AUTOR(ES): VILLALOBOS BARRERA, Jaime Eliécer y RODRÍGUEZ GÓMEZ, Jorge Eliécer[**]

PALABRAS CLAVES: Fuzzy, Antecedentes, Consecuencias, Control, Controlador, Funciones de membresía, Base de Reglas, Fuzificación, Defuzificación, Algoritmos, LabVIEW, Generación de código, Pic16f873, 68hc908jl3.

RESUMEN: Se presenta una herramienta básica en software y hardware para la implementación eficiente, rápida, y económica de un tipo de controlador con Lógica Fuzzy basado en un microcontrolador de gama media o baja. Se trata esencialmente de un algoritmo en lenguaje ensamblador para microcontroladores PIC y MOTOROLA, y que puede extenderse fácilmente para su uso en sistemas basados en Microprocesador o Procesador digital de señales (DSP).

Además se construye una herramienta software en LabVIEW® para MS-Windows® para la caracterización del sistema Fuzzy de dos entradas una salida y generación del código en lenguaje ensamblador que realiza control con Lógica Fuzzy en los microcontroladores PIC o Motorola CPU08.

Adicionalmente se describe el diseño del hardware construido para la demostración del control Fuzzy (tanto desde el microcontrolador como desde el computador) así como las principales características de sus componentes.

Ejemplo de aplicación del formato de control Fuzzy y del software FUZZYE3T. Se controla la temperatura de un sistema realimentado simple: Un elemento resistivo de calefacción. Se plantean las características del sistema y se implementa un control Fuzzy; utilizando como dispositivo de procesamiento, primero el computador IBM PC-Compatible y luego un microcontrolador.

---

[*] Proyecto de Grado.

[**] Facultad de Ingenierías Físico mecánicas, Programa de Ingeniería Electrónica. Director: BARRERO PÉREZ, Jaime Guillermo (UIS).

TITLE: CONTROLLER FUZZY LOGIC BASED ON MICRO CONTROLLER AND LABVIEW [*]

AUTHOR(S): VILLALOBOS BARRERA, Jaime Eliécer and RODRÍGUEZ GÓMEZ, Jorge Eliécer [**]

KEYWORDS: Fuzzy, Antecedents, Consequences, Control, Controller, Functions from membership, Base from Rules, Fuzificación, Defuzificación, Algorithms, LabVIEW, Generation of code, Pic16f873, 68hc908jl3.

ABSTRACT: Itself show a basic tool in software and hardware for the efficient, quick, and economic implementation of a type of controller based on Fuzzy Logic with a micro controller of half or low range. It is essentially of an algorithm in assembly language for PIC micro controller and MOTOROLA, that it could extend easily for their use in systems based on Microprocessor or digital Processor of signs (DSP).

A software tool is also built in LabVIEW® for MS-Windows® for the characterization of the Fuzzy system of two entrances one exit and generation of the code in assembly language that it carry out control with Fuzzy Logic in the PIC micro controller or Motorola CPU08.

Additionally the design of the built hardware is described for the demonstration of the Fuzzy control (so much from the micro controller like from the counter) as well as the main characteristic of their components.

Example of application the format of Fuzzy control and the software FUZZYE3T. The temperature of a system simple feedback is controlled: An element resistive of heating. The characteristics of the system are proposed and implemented a Fuzzy control is implemented by using like device of prosecution, first the counter IBM PC-Compatible and then a micro controller.

---

[*] Grade Project.

[**] Physic-Mechanical Engineering Faculty, Electronic Engineering. Direct by: BARRERO PÉREZ, Jaime Guillermo (UIS),

CONTENIDO

Lista de Tablas

PAG

Lista de Ecuaciones

PAG

Lista de Figuras

PAG

INTRODUCCIÓN

En el ámbito de la automatización y el control de procesos industriales, los controladores con lógica Fuzzy se encuentran actualmente en una etapa de desarrollo acelerado de sus aplicaciones prácticas. El empleo de controladores con Lógica *Fuzzy* ha cobrado especial importancia en sistemas con características de funcionamiento desfavorables tales como: No-linealidad, interferencia, tiempo muerto y perturbaciones externas, etc.

La primera aplicación de la Lógica *Fuzzy* a la solución de problemas reales consistió en el control de una máquina de vapor, trabajo realizado por el ingeniero británico Ebrahim Mandani[1], a comienzos de los años 70's. Sin embargo, la idea fue introducida en el campo de la ingeniería actual hacia el año 1965 por un matemático iraní; Lotfi Zadeh[2] profesor de la universidad de California en Berkeley (Estados Unidos), quien propuso la Lógica *Fuzzy* como un método de razonamiento abstracto similar al patrón de pensamiento humano para representar los problemas de control del mundo real.

Con el avance de la electrónica y en especial de los sistemas VLSI (o ULSI), es posible no solo implementar eficientemente las aplicaciones de control desde el computador, sino también desde pequeños sistemas electrónicos basados en microcontroladores,

---

[1] Mamdani, E. H. "Applications of Fuzzy Algorithms for Simple Dynamic Plant", Proc. IEEE, vol. 121, pp. 1585-1588,1974.

[2] Zadeh, L. A., "Fuzzy sets, Information and Control", vol. 8, pp. 338-353, 1965.

microprocesadores o DSPs. Lo anterior, con la reducción del costo y el tiempo de implementación de dichos sistemas han constituido factores decisivos en la consolidación de Lógica *Fuzzy* en el medio tecnológico comercial.

En este proyecto se desarrolla una herramienta básica en software y hardware para la implementación eficiente, rápida, y económica de un tipo de controlador con Lógica *Fuzzy* basado en un microcontrolador de gama media o baja. Se trata esencialmente de un algoritmo en lenguaje ensamblador para microcontroladores PIC y MOTOROLA, y que puede extenderse fácilmente para su uso en sistemas basados en Microprocesador o Procesador digital de señales (DSP).

Se evalúa entonces el control con Lógica *Fuzzy* tanto desde el computador como desde el microcontrolador, para luego comparar su desempeño. En la primera alternativa, haciendo el control desde el computador, se obtienen las señales de entrada a través de una tarjeta de adquisición de datos externa, luego se procesan en el computador mediante lógica Fuzzy y por ultimo se entrega una señal de salida a un sistema realimentado para probar la eficiencia de dicho control. Ahora bien la segunda alternativa, se diferencia de la anterior en cuánto el algoritmo de control *Fuzzy* es ejecutado exclusivamente y de manera autónoma desde un microcontrolador PIC16F873 o Motorola MC68HC908JL3.

La consecución de este aporte tecnológico es de gran significado para la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones por cuánto implica el paso concreto de la conceptualización a la puesta en práctica de sistemas de control lógicos *Fuzzy* de este tipo en nuestra universidad.

Motivaron a la realización de este proyecto, las grandes expectativas de nuestra creciente industria y sus necesidades de sistemas de control altamente eficientes a costos reducidos. Particularmente, la idea surge al dar soporte tecnológico a la automatización de la Planta Piloto para la Formación e Inhibición de Hidratos, del Centro de

Investigación del Gas de la Universidad Industrial de Santander. El detalle de este trabajo puede ser consultado en la tesis de grado titulada: "DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS PARA LA AUTOMATIZACIÓN DE UNA PLANTA DE FORMACIÓN E INHIBICIÓN DE HIDRATOS BAJO CONDICIONES DINÁMICAS"[3].

Este proyecto se enmarca en la misión de impulsar el desarrollo empresarial, social y científico de nuestro país, y a su vez fortalece el vínculo universidad, sociedad y empresa.

La organización del presente texto es la siguiente:

El Capítulo uno (1) consta de un Marco Teórico, en el cual se hace una breve inducción a la Lógica *Fuzzy*, al software (y el lenguaje de programación) con el cual se desarrolla el proyecto y a la perspectiva del diseño de hardware. Conceptos claves para la comprensión de las temáticas tratadas más adelante.

Los dos capítulos siguientes son esenciales en el desarrollo de este proyecto de grado:

En el Capítulo dos (2) se presenta el diseño del formato de software de control *Fuzzy* en lenguaje ensamblador, con el cual se logra implementar el control *Fuzzy* en microcontroladores de propósito general.

---

[3] ALVERNIA GÁLVIZ, Luis Emiro. RODRÍGUEZ GÓMEZ, Jorge Eliécer. "Diseño e Implementación de Algoritmos para la Automatización de una Planta de Formación e Inhibición de Hidratos bajo condiciones dinámicas ". Bucaramanga, Colombia, 2003. Proyecto de grado (Ingenieros Electrónicos). Universidad Industrial de Santander. Facultad de Ingenierías Físico – mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones.

El Capítulo tres (3) se describe el programa FUZZYE3T, elaborado en LabVIEW para implementar el formato de software de control *Fuzzy* en aplicaciones particulares definidas por el usuario.

En el capítulo cuatro (4) se desarrolla un ejemplo de aplicación del formato de control *Fuzzy* y del software FUZZYE3T. Se controla la temperatura de un sistema realimentado simple: Un elemento resistivo de calefacción. Se plantean las características del sistema y se implementa un control *Fuzzy;* utilizando como dispositivo de procesamiento, primero el computador y luego un microcontrolador.

El diseño de la tarjeta electrónica para la demostración del control *Fuzzy* (tanto desde el microcontrolador como desde el computador) así como las principales características de sus componentes se analizan en el Capítulo cinco (5) de esta tesis.

El Capítulo seis (6) recoge los resultados de las pruebas hechas al control lógico *Fuzzy* implementado.

Finalmente, en el Capítulo siete (7) se dan las conclusiones obtenidas en el presente trabajo y las tendencias que se sugiere tener en cuenta para la implementación de controladores *Fuzzy* de carácter comercial para la industria.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar una herramienta software y hardware para la implementación de un controlador con Lógica *Fuzzy* en el computador y el microcontrolador.

OBJETIVOS ESPECÍFICOS

- Recopilar y analizar el material bibliográfico referente a los conceptos básicos de Lógica *Fuzzy* y al sistema electrónico de la planta donde será implementado el controlador.

- Diseñar y construir el hardware necesario para implementar un control *Fuzzy* de temperatura, ya sea desde el computador o desde un microcontrolador (PIC 16F873 o Motorola MC68HC908JL3).

- Implementar un sistema de control *Fuzzy* de temperatura desde el computador utilizando el paquete *Fuzzy Logic Toolkit* de LabVIEW.

- Elaborar en LabVIEW un algoritmo que permita generar un archivo en código ensamblador para ejecutar en el microcontrolador (PIC 16F873 o Motorola

MC68HC908JL3) un control con Lógica *Fuzzy*, a partir de la caracterización del controlador que se haga en el entorno de programación de *Fuzzy Logic Toolkit*.

- Realizar pruebas de desempeño al controlador lógico *Fuzzy*, mediante la utilización de variables reales.

- Elaborar el documento de soporte y el manual técnico de operación para el usuario del controlador lógico *Fuzzy*.

1. MARCO TEÓRICO

En el contexto de ingeniería en el cual se concibe la herramienta en software y hardware para el desarrollo de controladores con lógica *Fuzzy*, tratada en esta tesis, se destacan principalmente los siguientes aspectos:

La lógica *Fuzzy*, como fundamentación teoría para la realización de estrategias de control no convencionales.

LabVIEW, software especializado de alto nivel utilizado en este proyecto para la caracterización e implementación de un tipo de controladores *Fuzzy*.

Assembler, lenguaje de programación en bajo nivel al cual se traduce un programa de control *Fuzzy* para ejecutar desde un microcontrolador, en este proyecto.

Hardware, pautas generales para tener en cuenta al particularizar el hardware del controlador *Fuzzy*.

1.1. LÓGICA *FUZZY*

La **Lógica** (del griego *logos* = razón) es la ciencia que trata los principios formales y normativos (leyes, modos y formas) del razonamiento. La **lógica clásica** hace referencia al **razonamiento lógico**, con el cual se deduce una conclusión a partir de **proposiciones precisas** relacionadas por medio de implicaciones; y en donde los **valores de certeza**

son normalmente **Uno (1)** para **Verdadero** y **Cero (0)** para **Falso**. La **lógica *Fuzzy*** por su parte, hace referencia al **razonamiento aproximado** con el cual se deduce una conclusión a partir de **proposiciones imprecisas** relacionadas por medio de implicaciones *Fuzzy*; y en donde el **grado de certidumbre** normalmente esta asociado al **intervalo [0;1]**, aceptándose de esta forma valores intermedios entre falso (0) y verdadero (1).

La característica principal de la lógica *Fuzzy* es la asignación de un grado de verdad a las fórmulas lógicas y está basada en la teoría matemática de los conjuntos *Fuzzy*, ésta se describe brevemente a continuación.

1.1.1. Conjuntos *Fuzzy*

Un conjunto *Fuzzy* es una clase de objetos (x's) cuyo grado de pertenencia al conjunto se representa mediante una función característica continua entre 0 (no pertenencia) y 1 (pertenencia completa). Dicha función es llamada **función de membresía** (o pertenencia) y se representa con la letra **μ;** puede interpretarse como una medida de la compatibilidad entre un valor del dominio (D) y la idea fundamental encerrada en el conjunto *Fuzzy*.

Como una extensión de la teoría de conjuntos clásica se tiene que: Un conjunto *Fuzzy* se considera **vacío** cuando su función de pertenencia es igual a cero (0) para todos los elementos del conjunto. Dos conjunto *Fuzzy* se consideran **iguales** cuando el valor de sus funciones de membresía es igual para todos los elementos del dominio. Un conjunto *Fuzzy* A es **subconjunto** de un conjunto *Fuzzy* B, si para cada elemento el grado de pertenencia al conjunto A es menor o igual que el grado de pertenencia al conjunto B. A diferencia de la teoría clásica de conjuntos, en teoría de conjuntos ***Fuzzy***, la unión de un conjunto *Fuzzy* y su complemento no produce el Universo; y la intersección de un conjunto *Fuzzy* y su complemento no es vacía. Las operaciones básicas de conjuntos se verán más adelante.

Un conjunto *Fuzzy* A puede representarse como pares elemento – grado de pertenencia, de la siguiente forma:

$$A = \{(x, \mu(x)) / x \in D\} \quad \text{Ec. 1}$$

1.1.1.1. Funciones de Membresía

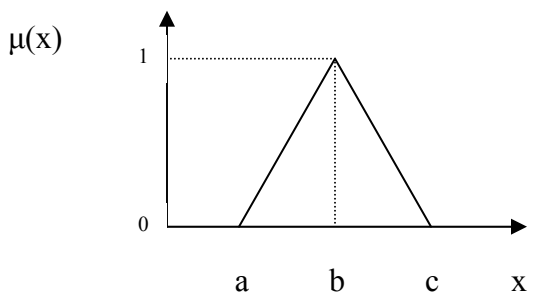
Aunque en principio la función de membresía puede tomar cualquier forma, en general es preferible usar funciones simples por cuanto simplifican los cálculos; a continuación se presentan las funciones de membresía usadas en el presente texto:

1.1.1.1.1. Triangular

Definida por sus límites inferior **a**, superior **c**, y el valor modal **b**; tal que:

$$\mu_{\Delta}(x) = \begin{cases} 0 & si \quad x \le a \\ \dfrac{(x-a)}{b-a} & si \quad x \in (a,b] \\ \dfrac{(c-x)}{(c-b)} & si \quad x \in (b,c) \\ 0 & si \quad x \ge c \end{cases} \quad \text{Ec. 2.}$$

Figura 1. Función de membresía triangular.



También puede representarse como:

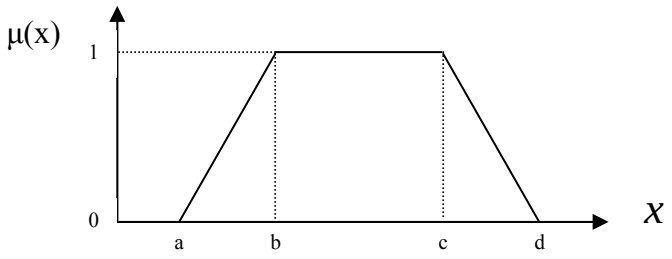
$$\mu_{\Delta}(x;a,b,c) = \text{máx}\left\{\text{mín}\left\{\frac{(x-a)}{b-a}, \frac{(c-x)}{(c-b)}\right\}, 0\right\} \quad \text{Ec. 3.}$$

1.1.1.1.2. Función Trapezoidal

Definida por sus límites inferior **a** y superior **d,** y los valores intermedios inferior **b** y superior **c**.

$$\mu_{\Pi}(x) = \begin{cases} 0 & si\ x \le a, \vee, x \ge d \\ \dfrac{(x-a)}{b-a} & si\ x \in (a,b) \\ 1 & si\ x \in [b,c] \\ \dfrac{(d-x)}{(d-c)} & si\ x \in (c,d) \end{cases} \qquad \text{Ec. 4}$$

Figura 2. Función de membresía trapezoidal.



Opcionalmente, puede representarse como:

$$\mu_{\Pi}(x;a,b,c,) = \text{máx}\left\{\text{mín}\left\{\frac{(x-a)}{(b-a)}, 1, \frac{(d-x)}{(d-c)}\right\}, 0\right\} \quad \text{Ec. 4.}$$

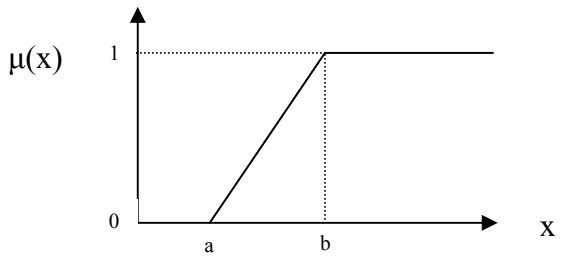
1.1.1.1.3. Función S

Las aproximaciones por segmentos de recta de las funciones S y Γ (Gamma) son similares, y corresponden básicamente a:

$$\mu_s(x) = \begin{cases} 0 & si\ x \le a \\ \dfrac{(x-a)}{b-a} & si\ x \in (a,b) \\ 1 & si\ x \ge c \end{cases} \quad \text{Ec. 5.}$$

Gráficamente como se ve en la siguiente figura:

Figura 3. Función de membresía S.



Esta función puede representarse también como:

$$\mu_s(x;a,b) = \text{máx}\left\{\text{mín}\left\{\frac{(x-a)}{b-a},1\right\},0\right\} \quad \text{Ec. 6.}$$

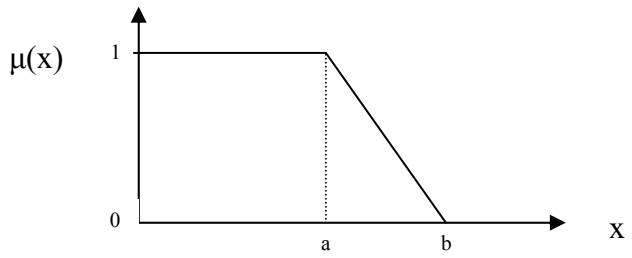
1.1.1.1.4. Función Z

En su forma lineal coincide con la de función S y se puede expresar como:

$$\mu_z(x) = \begin{cases} 1 & si\ x \le a \\ \dfrac{(b-x)}{(b-a)}\ si\ x \in (a,b) \\ 0 & si\ x \ge c \end{cases} \quad \text{Ec. 7.}$$

Gráficamente:

Figura 4. Función de membresía Z.



Además, puede expresarse de la siguiente forma:

$$\mu_z(x;a,b) = \text{máx}\left\{\text{mín}\left\{1, \frac{(b-x)}{(b-a)}\right\}, 0\right\} \quad \text{Ec. 8.}$$

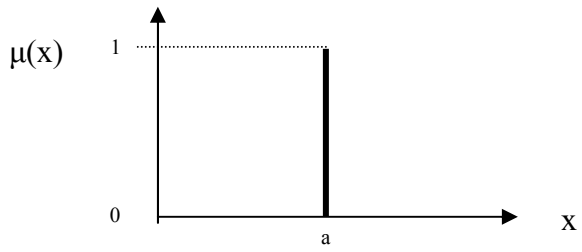
1.1.1.1.5. S*ingleton*

Esta es una función de membresía normalizada muy especial, debido a que se considera como un pulso de ancho infinitesimal establecido en un punto dado de las abscisas. El *singleton* es útil para modelar un valor *crisp* con un conjunto *Fuzzy*. Puede expresarse como:

$$\mu_\delta(x) = \begin{cases} 1\ si\ x = a \\ 0\ si\ x \ne a \end{cases} \quad \text{Ec. 9.}$$

Gráficamente, tiene la siguiente forma:

Figura 5. Función de membresía de tipo *singleton.*

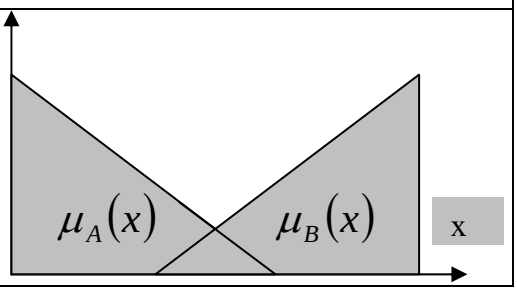


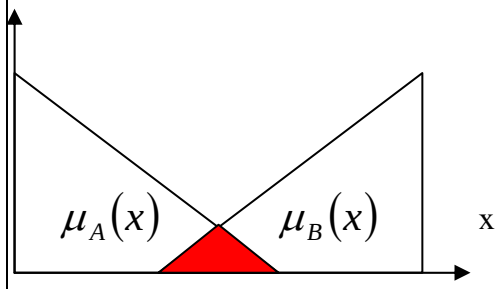
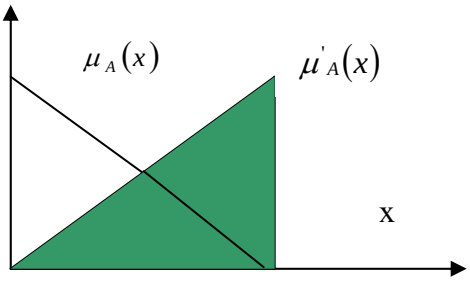
1.1.1.2. Operaciones Básicas sobre Conjuntos *Fuzzy*

Las operaciones básicas sobre conjuntos *Fuzzy* son: Unión, intersección y complemento. Cada una de estas operaciones sobre conjuntos Fuzzy da como resultado otro conjunto Fuzzy. Según las definiciones de Zadeh, la función de membresía para la **unión** e **intersección** de conjuntos *Fuzzy* representa respectivamente el **máximo** y el **mínimo** grado de pertenencia de cada elemento al conjunto; mientras que la función de membresía del **complemento** de un conjunto *Fuzzy* representa la **diferencia** entre la pertenencia total (1) y el grado de pertenencia de cada elemento al conjunto.

A continuación se presenta una descripción simbólica y una gráfica para cada operación, en el universo de discurso X tal que: $x \in X$.

Tabla 1. Operaciones básicas sobre conjuntos *Fuzzy*.

OPERACIÓN	**DESCRIPCIÓN**	
	SIMBÓLICA	**GRÁFICA**
Unión	$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$	

Intersección	$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$	
Complemento	$\mu'(x) = 1 - \mu(x)$	

1.1.2. Relaciones *Fuzzy*

Una relación *Fuzzy* establece cierto grado de interacción o asociación entre los elementos de los conjuntos *Fuzzy*. Las relaciones entre conjuntos *Fuzzy* dan como resultado otros conjuntos *Fuzzy* y pueden establecerse tanto sobre dominios iguales como diferentes.

Las relaciones *Fuzzy* se forman a partir de modelos de inferencia y permiten establecer propiedades de conexión entre conjuntos *Fuzzy*.

Otras operaciones importantes son el producto cartesiano de conjuntos *Fuzzy* y la composición de relaciones *Fuzzy.*

1.1.3. Operadores Lógicos

En el resumen de las definiciones de operadores lógicos que se presenta a continuación se utiliza la nomenclatura **v(A)** que representa el grado de verdad de la fórmula lógica.

Tabla 2. Operadores lógicos.

OPERADOR LÓGICO	FÓRMULA LÓGICA
Negación	$v(\neg A) = 1 - v(A)$
Conjunción	$v(A \wedge B) = min\{v(A), v(B)\}$
Disyunción	$v(A \vee B) = max\{v(A), v(B)\}$
Implicación[3]	$v(A \Rightarrow B) = max\{1 - v(A) + v(B), 1\}$
Universal	$v(\forall x P(x)) = \inf\{v(P(x_i))\}$
Existencial	$v(\exists x P(x)) = \sup\{v(P(x_i))\}$

1.1.4. Inferencia *Fuzzy*

Es el método con el cual se realizan los razonamientos dentro de la lógica *Fuzzy*. Es el proceso de aplicar una relación en particular para la evaluación de una implicación difusa. Según la relación utilizada la inferencia recibe un calificativo, normalmente el nombre de la relación utilizada.

1.1.5. Variable y términos lingüísticos

En Lógica *Fuzzy*, se puede decir que, una variable lingüística es un esquema lingüístico que tiene un significado susceptible de ser completado o ampliado por términos lingüísticos. Define el estado de un proceso variable por el grado de membresía de los parámetros de cada término lingüístico que la componen.

[3] La implicación lógica *fuzzy* puede definirse de varias formas, aquí se presenta la de Lukasiewicz.

Cada Variable Lingüística está conformada por un conjunto de propiedades entre las cuales podemos destacar:

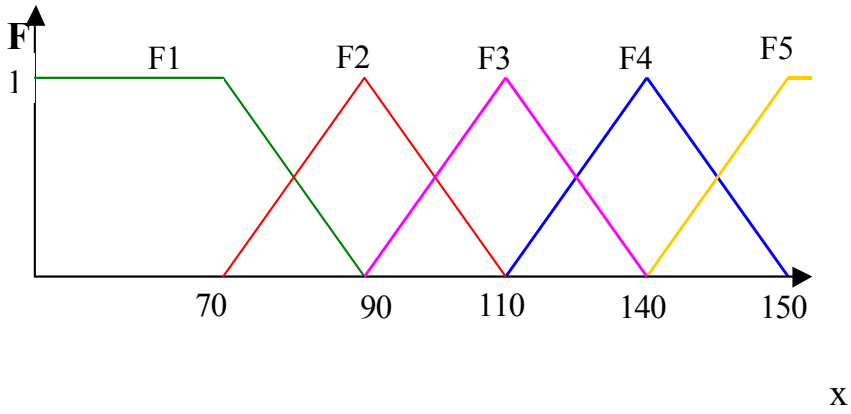
- **Nombre de la Variable**. Es una etiqueta con la que se distingue la naturaleza de la variable (por ejemplo, Error de Temperatura, Cambio de Error, etc).

- **Universo de Discurso**. Es el intervalo real dentro del cual la variable concreta puede tomar un valor (por ejemplo, [-1,1], [0,100], etc).

- **Términos Lingüísticos.** Son los posibles calificativos que puede tomar la variable (por ejemplo negativo, bajo, medio, máximo, etc.).

Los términos lingüísticos son atributos (categorías, etiquetas) semánticos para describir la variable lingüística en todo su espacio *Fuzzy*. Un término es definido cuantitativamente por la correspondiente función de membresía del conjunto *Fuzzy* que lo representa. En la siguiente figura se muestra una variable lingüística determinada (F) con la definición de sus términos lingüísticos (F1, F2, F3, F4, F5).

Figura 6. Variable lingüística.



1.1.6. Conceptos del Diseño *Fuzzy*

En esencia, un controlador lógico *Fuzzy* contiene un algoritmo capaz de convertir una estrategia lingüística de control en una estrategia de control automático. Dicho algoritmo se basa en reglas obtenidas a partir del conocimiento empírico de cómo controlar el fenómeno, sin necesidad de conocer el modelo matemático del sistema a controlar. Este conocimiento puede expresarse de una manera muy natural, empleando las variables lingüísticas que son descritas mediante conjuntos *Fuzzy*.

Diseñar un controlador *Fuzzy* significa esencialmente escribir las reglas de control, determinando los antecedentes y los consecuentes. Dicho diseño plantea como mínimo la selección de los siguientes parámetros:

a) Número de Variables Concretas de Entrada y de Salida.

b) Estructura de las Variables Lingüísticas asociadas a cada Variable Concreta de Entrada y de Salida.

c) Definición de los Conjuntos *Fuzzy* asociados a cada Valor Lingüístico de cada Variable Lingüística.

d) Definición del tipo de Fuzificador empleado para cada Variable de Entrada.

e) Definición del número de reglas presentes en la Base de reglas.

f) Valores Lingüísticos de cada una de las variables del Antecedente y del Consecuente para cada regla.

g) Tipo de relación de Implicación, operador AND, OR es la composición a utilizar en el Motor de Inferencia.

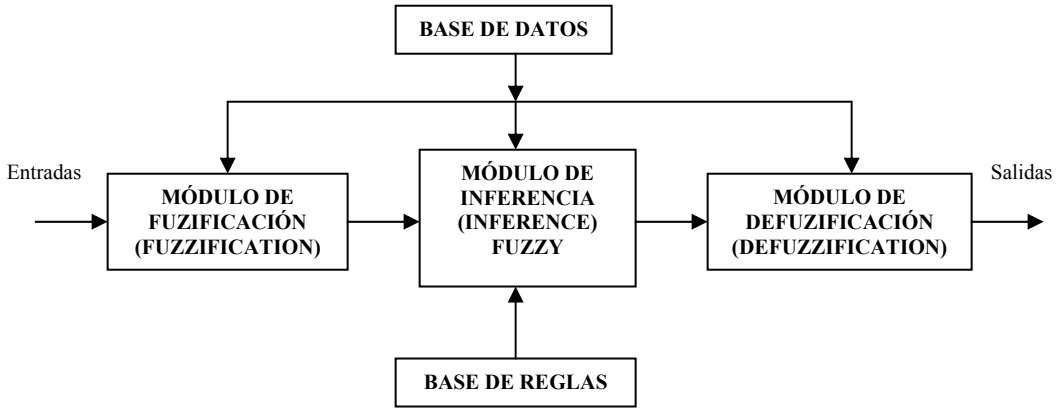
h) Tipo de Defuzificador empleado para cada Variable de Salida.

i) Tipo de Unión o Intersección a utilizar en el Defuzificador.

El objetivo de un sistema de control *Fuzzy* es encontrar una respuesta satisfactoria, es decir que la salida de la planta siga a la entrada de referencia a pesar de la presencia de disturbios en la planta y de los errores en el sensor.

El proceso de mantener la salida del controlador cerca de la referencia o *setpoint* se conoce como el proceso de regulación. Cuando un sistema tiene buena regulación en presencia de disturbios se dice que tiene buen rechazo al disturbio. A un sistema que tiene buena regulación ante cambios en los parámetros de la planta se le llama de baja sensitividad a estos parámetros. Un sistema que tiene tanto buen rechazo al disturbio como baja sensibilidad se le conoce como robusto.

Figura 7. Esquema general de un controlador *Fuzzy.*



1.1.6.1. Estructura General de un Controlador *Fuzzy*

Contempla básicamente las siguientes etapas: Módulo de Fuzificación, Base de conocimiento, Módulo de Inferencia, y Módulo de Defuzificación. En la figura siete (7) se presenta el esquema de esta estructura general.

Un Sistema de Lógica *Fuzzy* puede entenderse como un sistema no lineal de múltiples entradas y múltiples salidas, cuya estructura interna es del mismo tipo que la anterior.

1.1.6.1.1. Módulo de Fuzificación

Es la etapa inicial, en ella se trasladan las entradas precisas (por ejemplo la lectura tomada por un sensor de temperatura) dentro de una representación lingüística. Es decir, transforma las variables de entrada en sus respectivos valores difusos definidos por los términos lingüísticos de los conjuntos difusos a los que pertenecen y su correspondiente grado de pertenencia.

1.1.6.1.2. Base de conocimientos

Esta etapa esta formada por dos (2) componentes básicos:

- **Base de Datos**: Son los conceptos asociados para caracterizar las reglas del control *Fuzzy*. Estos conceptos, generalmente se definen subjetivamente con base en el conocimiento de los expertos.

- **Base de Reglas de Control *Fuzzy***: Permite la toma de las decisiones de control. Las reglas de control tratadas en este caso son del tipo: **SI** {antecedente} **ENTONCES** {consecuente}.

1.1.6.1.3. Módulo de Inferencia *Fuzzy*

Son los procedimientos con los que se evalúa la estrategia de control contenida en las reglas lógicas *Fuzzy*, por medio de la implicación *Fuzzy*. En esta etapa se selecciona entre las reglas definidas para el controlador las que tienen un grado de certeza mayor que cero (soporte) para los valores *Fuzzy* calculados en la etapa de Fuzificación y se calcula el grado en que éstas influyen a la decisión final.

1.1.6.1.4. Módulo de Defuzificación

Traslada el resultado de la etapa anterior a un valor preciso. Es decir, las acciones de control de tipo *Fuzzy* se transforman, en esta etapa, en acciones de control de tipo cuantitativo, que permiten un adecuado funcionamiento de uno o más actuadores del sistema bajo control. Esto se logra aplicando las conclusiones de control de tipo *Fuzzy* a los conjuntos *Fuzzy* de las variables de salida.

1.2. LABVIEW

LabVIEW es un software que ofrece un ambiente de desarrollo gráfico con una metodología fácil de dominar por ingenieros y científicos. Con esta herramienta se pueden crear fácilmente interfases de usuario para la instrumentación virtual sin necesidad de elaborar código de programación. Para especificar las funciones sólo se requiere construir diagramas de bloque. Se tiene acceso a una paleta de controles de la cual se pueden escoger desplegados diferentes tipos de controles e indicadores (numéricos, medidores, termómetros, tanques, gráficas, etc.) e incluirlos como iconos en la programación de la aplicación.

Se basa en un modelo de programación de flujo de datos denominado G, que libera a los programadores de la rigidez de las arquitecturas basadas en texto, aunque también las permite a través de los CIN's (Code Interface Node). El Paquete de software *NI LabVIEW Profesional development System*, además, es un sistema de programación gráfica que tiene un compilador que genera código optimizado, cuya velocidad de

ejecución es comparable al lenguaje C. Los desarrollos construidos son plenamente compatibles con las normas VISA, GPIB, VXI y la alianza de sistemas VXI Plug & Play.

El paquete software adicional *Fuzzy Logic for G Toolkit* de LabVIEW provee un entorno para la caracterización de los controladores *Fuzzy*. Este se describe a continuación.

1.2.1. Fuzzy Logic Toolkit

Con *Fuzzy Logic for G Toolkit* es posible diseñar controladores con lógica *Fuzzy* e implementarlos en aplicaciones desarrolladas en LabVIEW. Combinando los VI's de éste *Toolkit* con las funciones lógicas y matemáticas de LabVIEW, es posible desarrollar estrategias de control real mediante diagramas de bloques.

Este *Toolkit* posee cuatro VI's principales: *Fuzzy Logic Controller Design VI, Load Fuzzy Controller VI, Fuzzy Controller VI,* y *Test Fuzzy Control VI.*

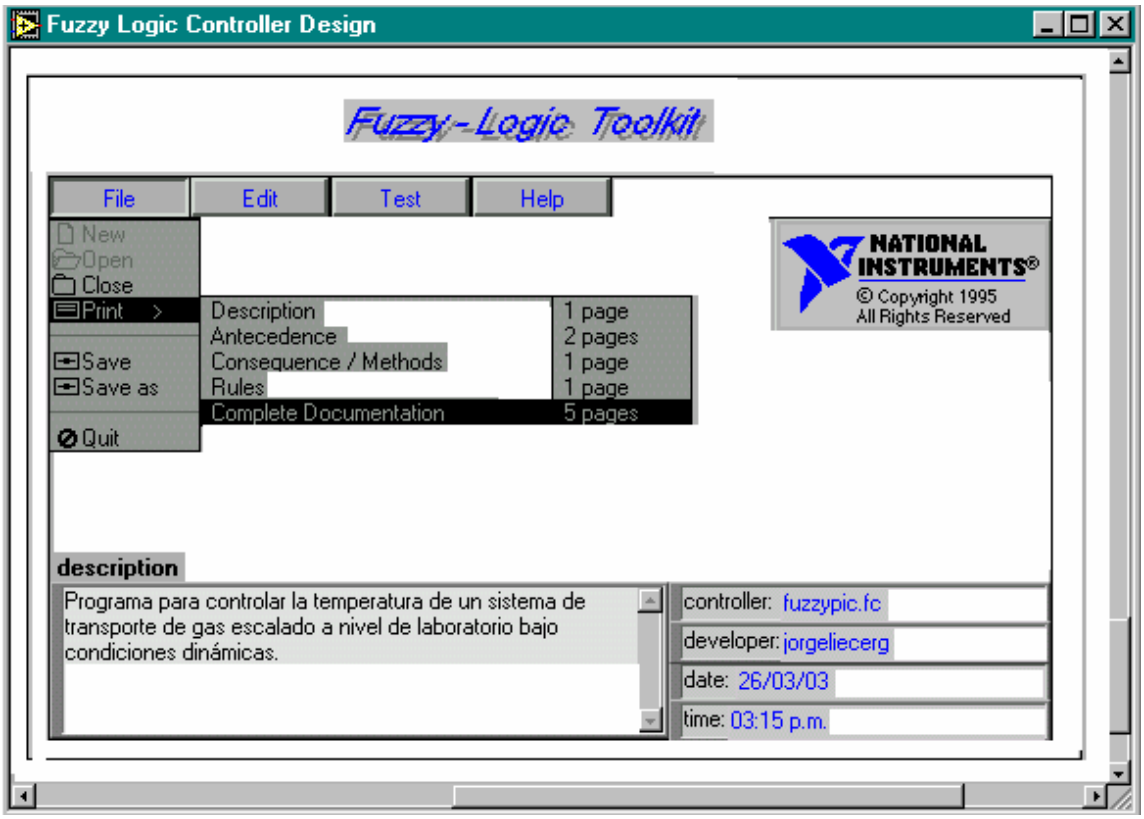
1.2.1.1 Fuzzy Logic Controller Design VI

Este VI provee una interfase gráfica de usuario para definir las funciones de membresía, la base de reglas de control y los parámetros del controlador *fuzzy.* Los SubVIs que estructuran este VI se organizan en cuatro capas de abstracción: *Project Manager*, *Fuzzy Set Editor, Rule Base Editor* y *Test*.

1.2.1.1.1. Project Manager

Este SubVI contiene el proyecto. Su panel frontal puede verse en la siguiente figura.

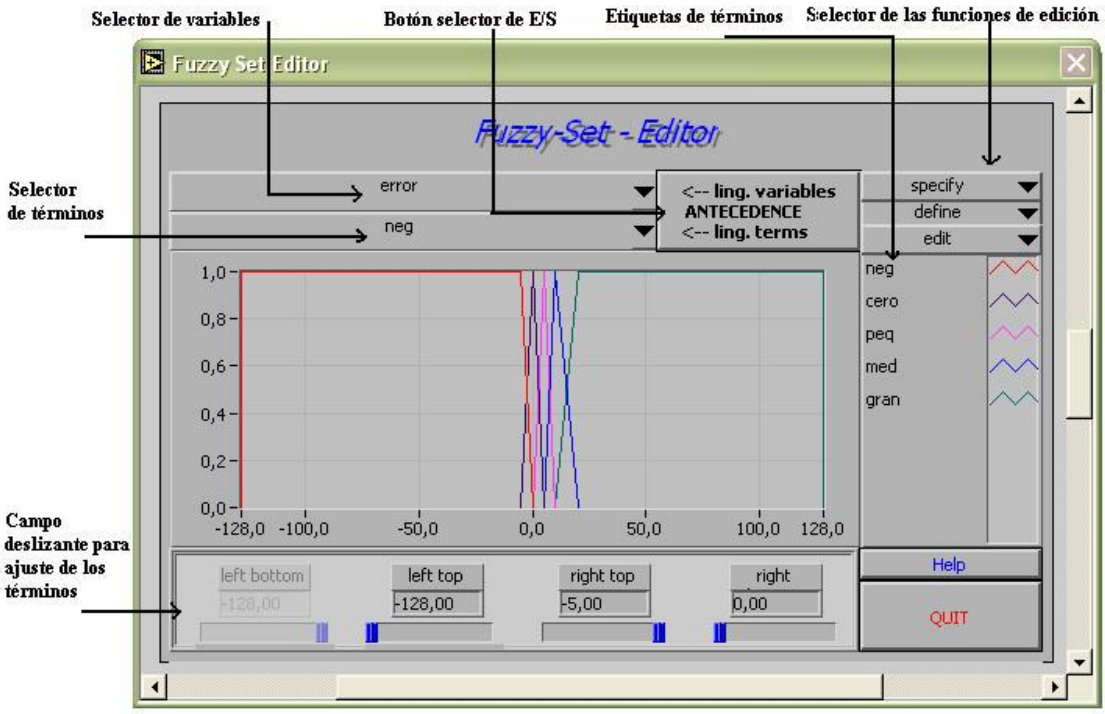
Figura 8. *Project Manager* .



1.2.1.1.2. Fuzzy Set Editor

Con este SubVI se definen y modifican las variables y sus términos linguisticos. En la siguiente figura se ve su panel frontal.

Figura 9. Panel frontal de *Fuzzy Set Editor.*

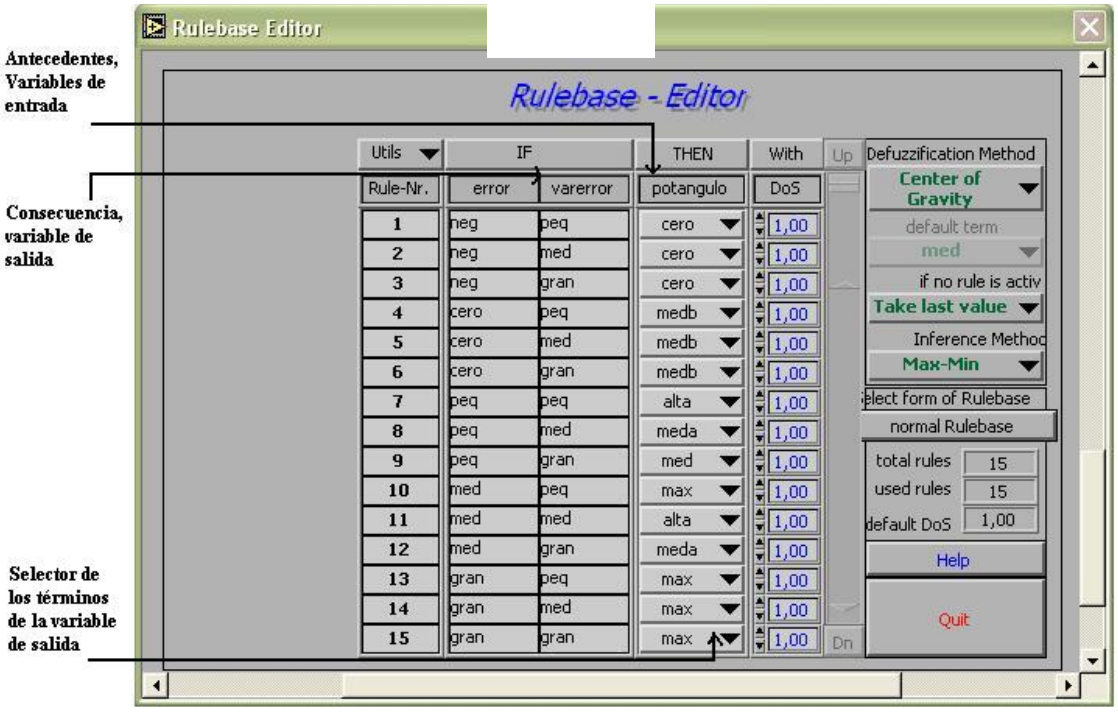


El editor presenta unos valores por defecto que pueden ser modificados mediante los campos de selección y ajuste de variables y términos lingüísticos. Además, mediante los atributos de edición se accede a opciones como renombrar, cambiar rangos, adicionar y remover términos y variables, etc.

1.2.1.1.3. R*ule Base Editor*

Con este SubVI se define y modifica la base de reglas del sistema *fuzzy* a desarrollar. El panel frontal de este SubVI puede verse en la siguiente figura.

Figura 10. Panel frontal de *Rule Base Editor*.



El *Rule Base Editor* presenta unos valores por defecto que pueden ser modificados mediante la manipulación de los selectores de campo provistos en el.

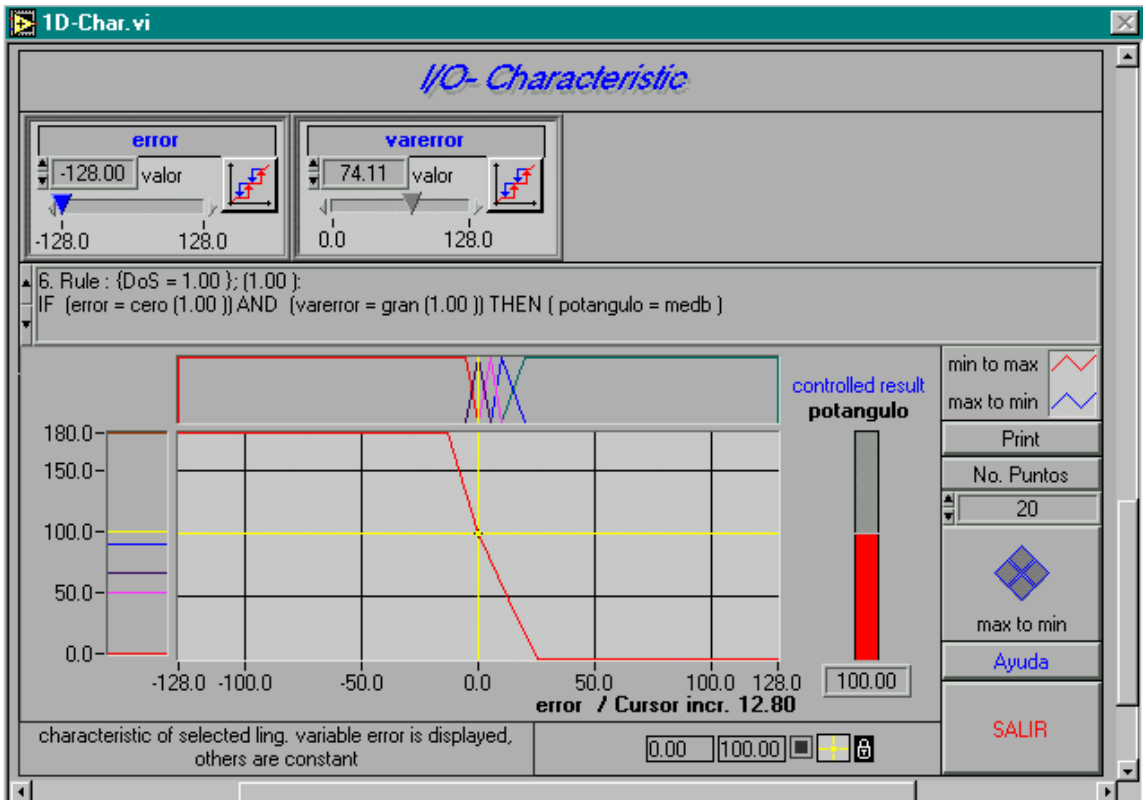
Cada regla esta asociada con un factor de peso o grado de soporte (DoS) para acentuar o reducir la influencia de la regla en la caracterización del controlador. El factor de peso junto con técnicas tales como los algoritmos genéticos puede ser una excelente alternativa para la optimización del controlador.

En este panel frontal también es posible modificar el método de Defuzificación (*Center of Maximum, Center of Gravity, Mean of Maximum*) entre otras opciones.

1.2.1.1.4. *Test*

Con este SubVI se prueban las características de entrada/salida del controlador *fuzzy* desarrollado mediante los SubVIs anteriores. El panel frontal de este SubVI se ve a continuación.

Figura 11. Panel frontal de *Test.*



1.2.1.2 *Load Fuzzy Controller VI*

Este VI carga el conjunto completo de parámetros e información del controlador definidos en el VI anteriormente descrito. La extensión del archivo usada para el archivo de datos es **".*fc*"**.

1.2.1.3. F*uzzy Controller VI*

Este VI es utilizado para implementar la aplicación de control con lógica *Fuzzy* en LabVIEW. Para la utilización de este VI se requiere que el archivo de datos del controlador (desarrollado con el *Fuzzy Logic Controller Design VI*) sea cargado antes (mediante el *Load Fuzzy Controller VI*) y cableado al terminal de entrada de este VI.

1.2.1.4. T*est Fuzzy Control VI*

Este VI es un ejemplo de aplicación para probar el controlador *Fuzzy* desarrollado con *Fuzzy Logic Toolkit*. Al iniciar este VI requiere del archivo de datos del controlador *Fuzzy*.

Para mayor información al respecto de *Fuzzy Logic Toolkit for LabVIEW* remítase al Anexo 1 y a las referencias bibliografías del presente texto.

1.2.1.5. Restricciones

Para el desarrollo de un controlador en *Fuzzy Logic Toolkit for LabVIEW*, se deben tener en cuenta las siguientes restricciones:

- El máximo número de variables lingüísticas de entrada permitidas al controlador es cuatro (4).

- El máximo número de términos lingüísticos para cada variable lingüística es nueve (9).

- Las funciones de membresía normalizadas útiles en este entorno gráfico son las de tipo: Triangular, trapezoidal, Z, S y *singleton*.

- El máximo número de variables lingüísticas de salida permitidas al controlador es una (1).

1.3. HARDWARE

Actualmente, existen técnicas novedosas de implementación de sistemas con lógica *Fuzzy* dirigidas hacia el diseño de hardware *Fuzzy*. Estas técnicas presentan entre sus herramientas básicas el hardware configurable y los procesadores especializados para este fin. Los sistemas así desarrollados ejecutan tareas en tiempo real aún para sistemas de alta complejidad.

El tipo de controlador Fuzzy desarrollado en este proyecto esta orientado a satisfacer las necesidades de la industria a un bajo costo, sus aplicaciones pueden revestir de una significativa complejidad pero que puede ser convenientemente cubierta con la implementación de las técnicas *Fuzzy* en software aquí descritas y soportadas sobre un hardware basado en un microcontrolador de propósito general (aunque puede extenderse a sistemas basados en microprocesador o DSP).
Desde esta perspectiva, es claro que para particularizar el controlador *Fuzzy* en un diseño concreto hay que tener en cuenta una multitud de factores propios del sistema y de la aplicación específica. Al seleccionar el microcontrolador y demás componentes del controlador se debe prever que cumpla con las normas de seguridad y confiabilidad a las que debe ajustarse un equipo electrónico, manteniendo una relación costo/beneficio bajo.

En el caso especifico de la utilización de los microcontroladores de ocho bits se presentan en el anexo tres (3) las características de los modelos PIC16F873 y MC68HC908JL3 de las empresas Microchip y Motorola respectivamente, considerando

que estos son actualmente en nuestro medio de los más difundidos y fácil adquisición en el mercado; sin embargo, no se pretende hacer una comparación pues como ya se dijo anteriormente esto debe evaluarse para cada aplicación en particular.

2. DISEÑO DEL SOFTWARE DE CONTROL FUZZY EN CÓDIGO ENSAMBLADOR

En esencia, se diseñó en este proyecto un algoritmo para realizar control con lógica *Fuzzy* de manera relativamente eficiente tanto en tiempo de ejecución como en tamaño de código. El algoritmo en principio puede traducirse a cualquier lenguaje de programación y ejecutarse en un dispositivo tal como un microprocesador, microcontrolador o DSP. Sin embargo, se hace la programación sólo en lenguaje ensamblador particularizado para microcontroladores de ocho bits de la familia PIC (gamas básica y media) de MICROCHIP y de la familia CPU08 de MOTOROLA; Considerando que con estos Circuitos Integrados se alcanza el propósito intrínseco, de este trabajo, de proporcionar una herramienta de desarrollo de controladores con lógica Fuzzy construidos en hardware de bajo costo. Y con lenguaje ensamblador se obtiene un control más eficiente sobre las operaciones de estos dispositivos, por tener una representación directa de las instrucciones en código máquina de sus procesadores.

2.1. CARACTERÍSTICAS

El algoritmo de control *Fuzzy* desarrollado tiene las siguientes características generales:

- Tipo de inferencia Fuzzy: Condicional de reglas del tipo SI... ENTONCES.
- Tipo de funciones de membresía: Triangular, Z, S, y trapezoidal.
- Método de Defuzificación: Centroide.

- ✂ Método de implicación Fuzzy: Mínimo – máximo.
- ✂ Reglas por defecto: La regla anteriormente usada y la de salida cero.

- ✂ Variables de entrada: Admite dos (2) variables de entrada para la Fuzificación.

- ✂ Variables de salida: Entrega el valor de una (1) variable de salida luego de la defuzificación.

- ✂ Términos lingüísticos de las variables de entrada: El algoritmo no esta limitado a un número estricto de términos lingüísticos, pero se recomienda que la suma de los términos de todas las variables de entrada sea menor o igual a ocho (8).

- ✂ Términos lingüísticos de la variable de salida: Máximo (9) términos lingüísticos.

2.2. METODOLOGÍA DEL DISEÑO

La metodología que se utilizada para el diseño del formato de software de control *Fuzzy* puede resumirse de la siguiente forma:

- ✂ Análisis del procedimiento de control con lógica *Fuzzy*. En esta etapa, se hizo un seguimiento de un ejemplo didáctico de control *Fuzzy* y se detectó que era posible desarrollar un control de este tipo mediante operaciones matemáticas simples con base en la representación lineal de las funciones de membresía Triangular, Trapezoidal, Z, S y tipo *Singleton*.

- ✂ Organización de las tareas especificas del control *Fuzzy* en estructuras algorítmicas independientes. En esta etapa se crearon y ordenaron segmentos de programa mediante instrucciones básicas de programación (de control lineal, de selección y

repetición definidas) para cumplir independientemente con las tareas de Fuzificación, Inferencia *Fuzzy* y Defuzificación.

✂ Ensamble del formato de programa general y Prueba del mismo. Finalmente, se agruparon los segmentos de programa de tareas especificas creados en la etapa anterior para luego de un procedimiento de depuración de código, simulación y prueba del mismo (mediante variables reales), presentar la solución lógica al problema inicialmente planteado, de una manera fácil de programar o implementar en cualquier lenguaje o herramienta software disponible en el mercado.

2.3. PROCEDIMIENTO DE CONTROL FUZZY

A continuación se presenta el procedimiento de control con lógica *Fuzzy* utilizado en este proyecto, mediante un ejemplo didáctico.

2.3.1. Ejemplo Práctico

Suponga que el control de un sistema determinado es factible de hacerse mediante un controlador *Fuzzy* con la siguiente caracterización:

✂ Dos variables lingüísticas de entrada: F y G.

✂ Cinco términos lingüísticos cada variable, definidos con los conjuntos *Fuzzy* como se muestra en las siguientes figuras.

Figura 12. Términos lingüísticos de la variable lingüística F.

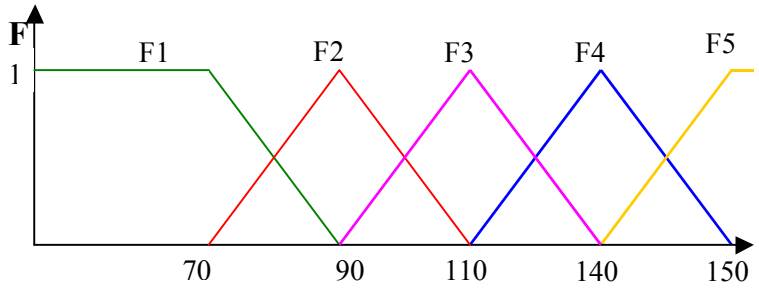
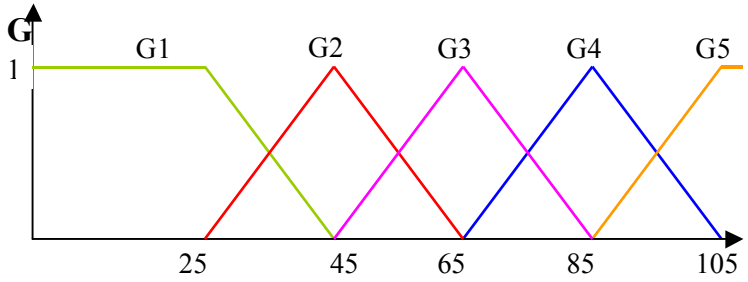
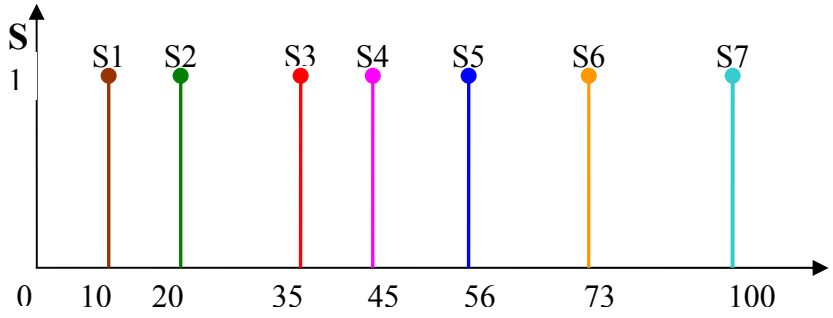


Figura 13. Términos lingüísticos de la variable lingüística G.



- Una variable de salida: S. Con siete términos lingüísticos definidos por las funciones de membresía como se muestra a continuación:

Figura 14. Términos lingüísticos de la variable de salida.



Una definición de reglas como sigue:

Si F es F1 y G es G1 es entonces S es S7

2. Si F es F1 y G es G2 es entonces S es S7
3. Si F es F1 y G es G3 es entonces S es S6
4. Si F es F1 y G es G4 es entonces S es S6
5. Si F es F1 y G es G5 es entonces S es S5
6. Si F es F2 y G es G1 es entonces S es S6
7. Si F es F2 y G es G2 es entonces S es S6
8. Si F es F2 y G es G3 es entonces S es S5
9. Si F es F2 y G es G4 es entonces S es S5
10. Si F es F2 y G es G5 es entonces S es S5
11. Si F es F3 y G es G1 es entonces S es S5
12. Si F es F3 y G es G2 es entonces S es S4
13. Si F es F3 y G es G3 es entonces S es S4
14. Si F es F3 y G es G4 es entonces S es S3
15. Si F es F3 y G es G5 es entonces S es S2
16. Si F es F4 y G es G1 es entonces S es S3
17. Si F es F4 y G es G2 es entonces S es S2
18. Si F es F4 y G es G3 es entonces S es S3
19. Si F es F4 y G es G4 es entonces S es S2
20. Si F es F4 y G es G5 es entonces S es S1
21. Si F es F5 y G es G1 es entonces S es S2
22. Si F es F5 y G es G2 es entonces S es S2
23. Si F es F5 y G es G3 es entonces S es S2
24. Si F es F5 y G es G4 es entonces S es S1
25. Si F es F5 y G es G5 es entonces S es S1

Suponiendo las siguientes entradas se tiene:

para Xf= 85 y Xg =50

✂ El proceso de Fuzificación es el siguiente.

$$F1(85) = \frac{90-85}{90-70}$$

$$F1(85) = 0.25$$

de manera similar se tiene que

$$F2(85) = 0.25$$

ahora para la otra variable

$$G2(50) = \frac{65-50}{65-45}$$

$$G2(50) = 0.75$$

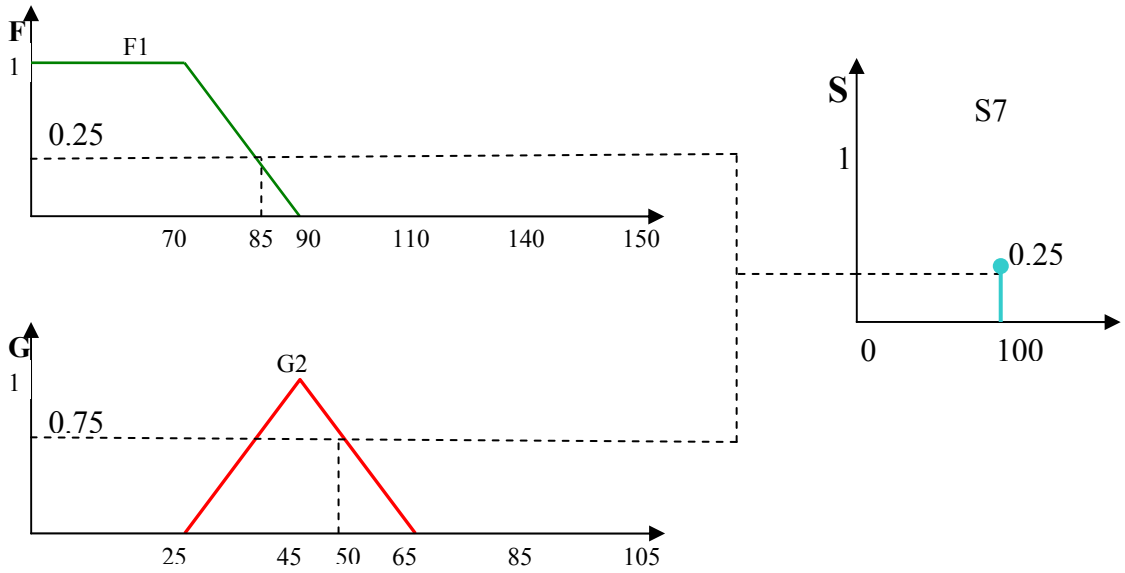
De manera similar se tiene que

$G3(50) = 0.75$

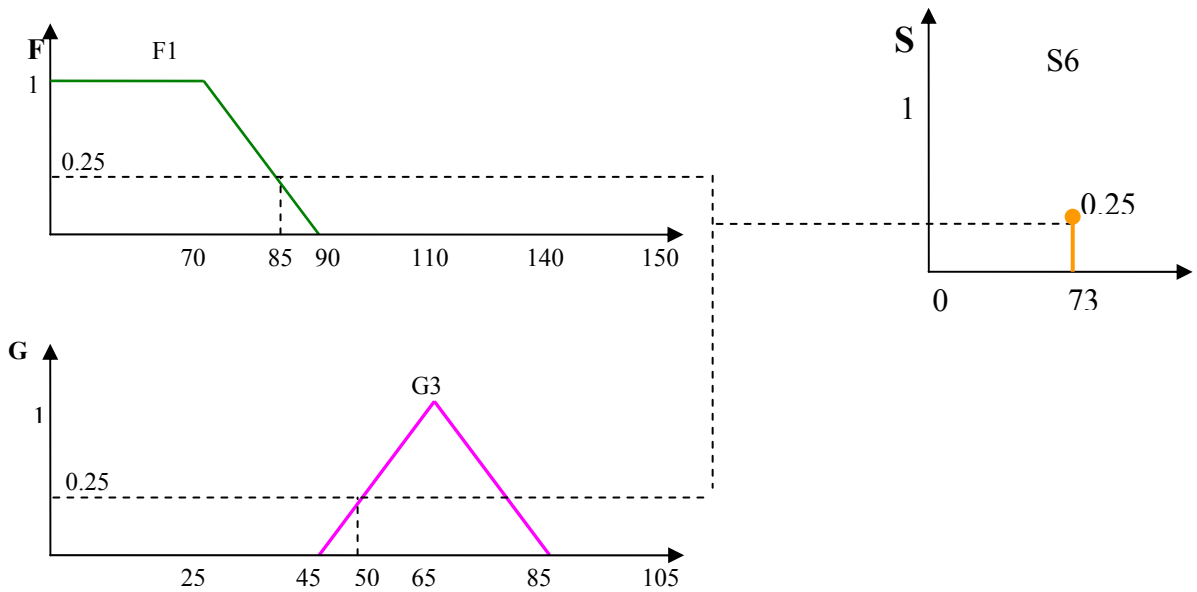
✂ El proceso de inferencia *Fuzzy* es el siguiente.

Si F es F1 y G es G2 entonces S es S7

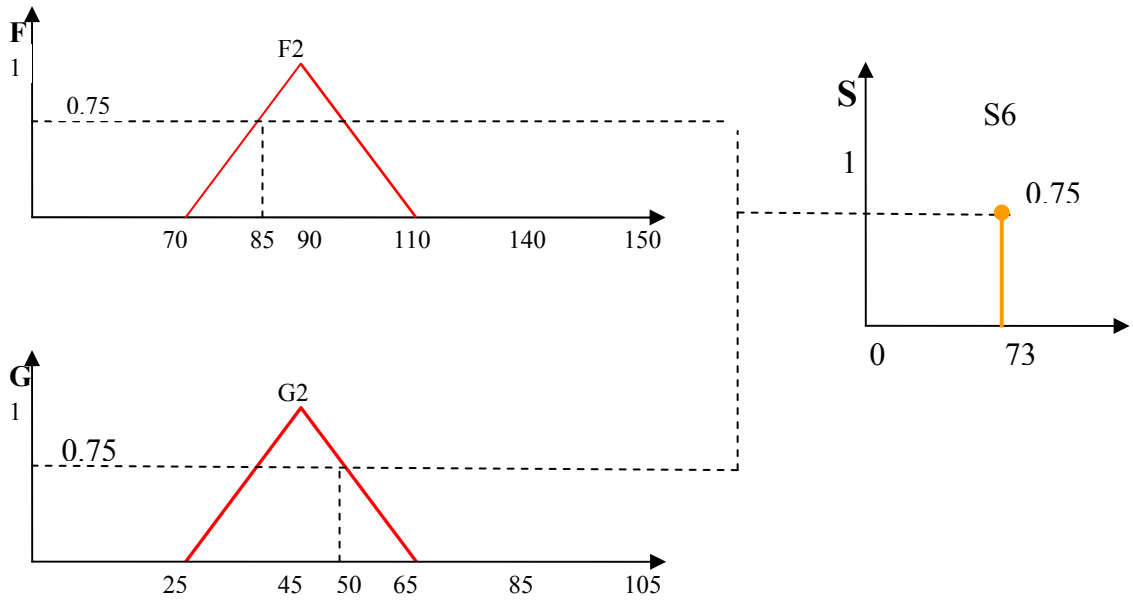
Figura 15. Proceso de inferencia *Fuzzy*.



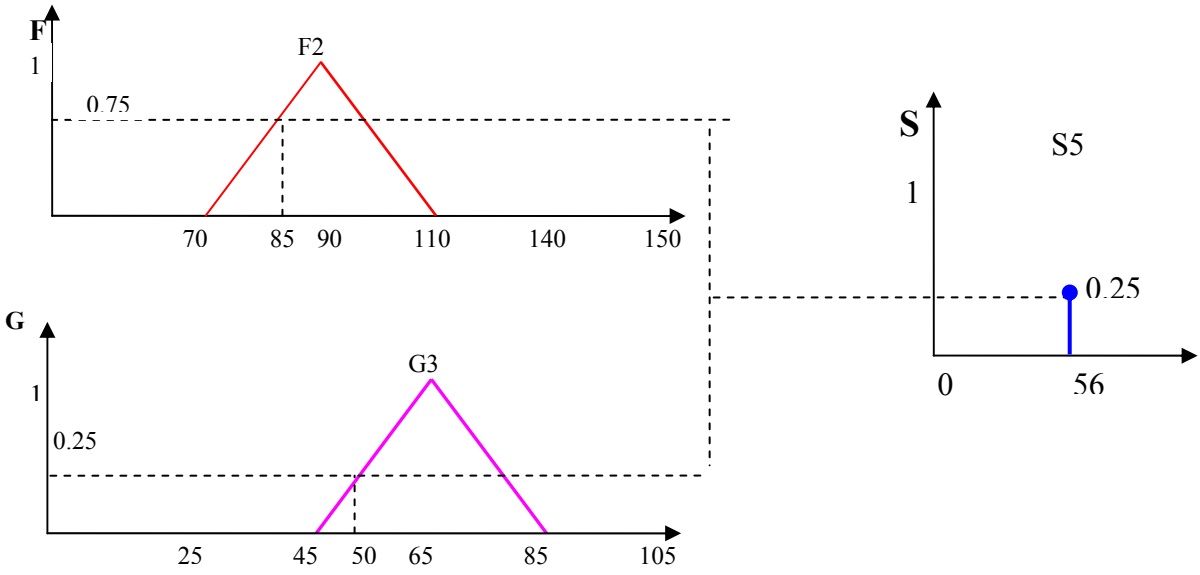
Si F es F1 y G es G3 entonces S es S6



Si F es F2 y G es G2 entonces S es S6



Si F es F2 y G es G3 entonces S es S5



✂ La defuzificación por medio de centroides es:

$$SALIDA = \sum_i \frac{Si \times Xsi}{Si}$$

$$SALIDA = \frac{100 \times 0.25 + 73 \times 0.25 + 73 \times 0.75 + 56 \times 0.25}{0.25 + 0.25 + 0.75 + 0.25}$$

$$SALIDA = \frac{112}{1.5} = 74.6666$$

$$SALIDA = 74.6666$$

2.4. ESTRUCTURAS ALGORÍTMICAS DE CONTROL FUZZY

Las estructuras algorítmicas diseñadas tienen correspondencia directa con las etapas de la estructura interna del controlador *Fuzzy* descritas en el apartado 1.1.6.1 del Marco Teórico. Éstas etapas son: Fuzificación, Base de conocimiento, Inferencia *Fuzzy* y Defuzificación.

Los algoritmos y código fuente serán explicados inicialmente para el microcontrolador PIC y luego se presentarán para el microcontrolador MOTOROLA.

2.4.1. Etapa de Fuzificación

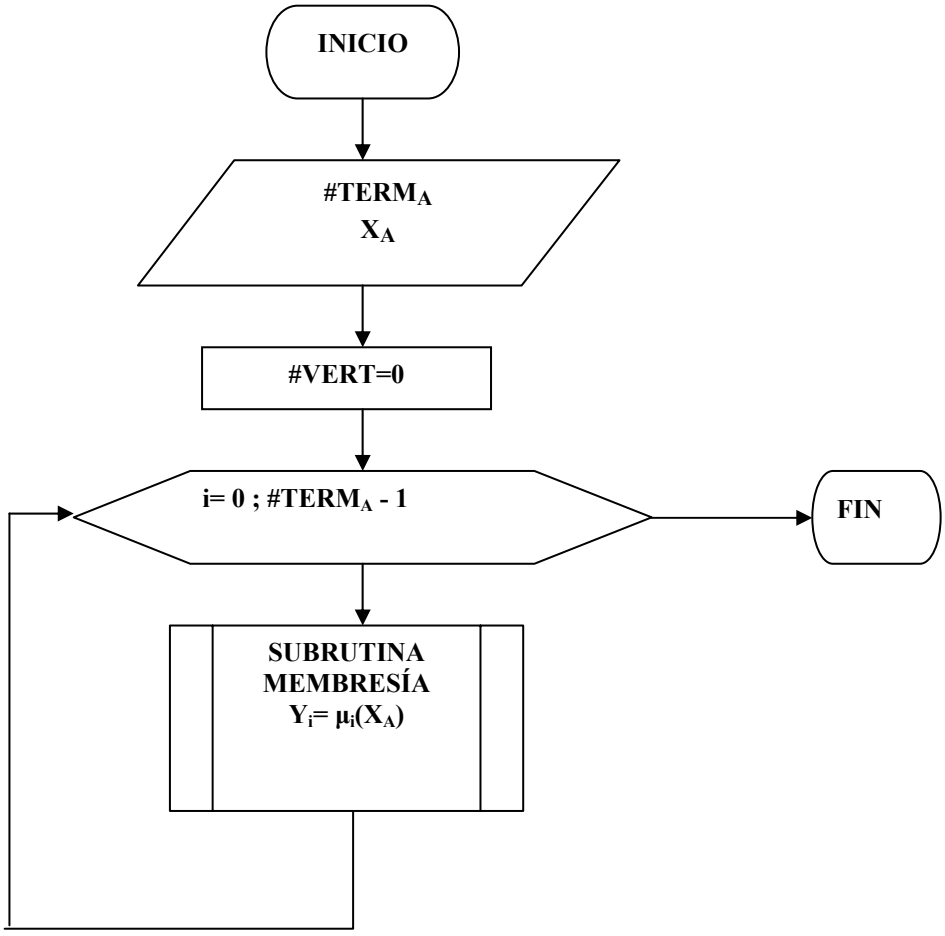
Para trasladar una determinada entrada de valor preciso (*crisp*) dentro de una representación lingüística de acuerdo a una caracterización *Fuzzy* dada, se deben seguir tres pasos fundamentalmente:

1. Comprobar que el valor de entrada coincida en el universo de discurso de la definición de las variables lingüísticas del controlador *Fuzzy.*

2. Evaluar el grado de pertenencia que corresponda de acuerdo a cada uno de los conjuntos *Fuzzy* en donde tenga cabida en sus dominios el valor de entrada.

3. Almacenar el valor *Fuzzy* que resulta de la transformación.

Figura 16. Diagrama de flujo simplificado de la etapa de fuzificación se muestra en la siguiente



Este programa toma dos datos de entrada: El número de términos lingüísticos "**#TERM$_A$**" de una variable lingüística determinada y el valor de entrada " **X_A"** que se desea convertir en un valor *Fuzzy.*

La variable contador "**#VERT"** se emplea para ejecutar la subrutina **MEMBRESÍA** conjuntamente con la etapa de base de datos, como se verá más adelante.

El algoritmo de fuzificación evalúa el grado de pertenencia del valor *crisp* de entrada a cada uno de los conjuntos que representan los términos lingüísticos de una variable *Fuzzy*. Un valor de entrada puede tener más de una representación *Fuzzy*. Esto tiene que ver con el solapamiento que se permite entre los conjuntos *Fuzzy* de una variable lingüística.

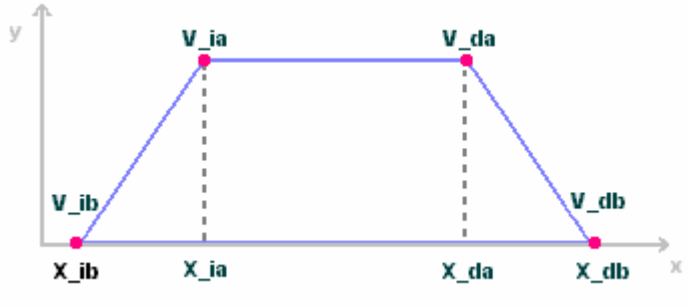
2.4.1.1. Evaluación del Grado de Membresía a un Conjunto Fuzzy

Los procesos técnicos para los cuales es factible aplicar el controlador con lógica *Fuzzy* diseñado en esta tesis, deben ser modelados con funciones de membresía normalizadas estándar de formas lineales como las de tipo Z, Triangular, Trapezoidal y Tipo S.

Estas funciones representan los conjuntos *Fuzzy* de manera tal que pueden dividirse por segmentos de recta para evaluarlos con interpolación lineal de primer orden. Esto permite, además, menor espacio de memoria para almacenar las funciones de membresía; ya que se tienen cuatro puntos como máximo para definir cada función.

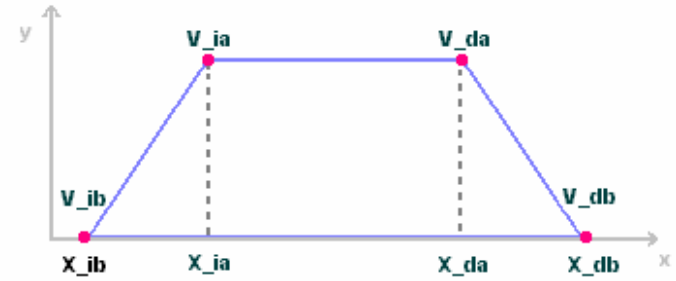
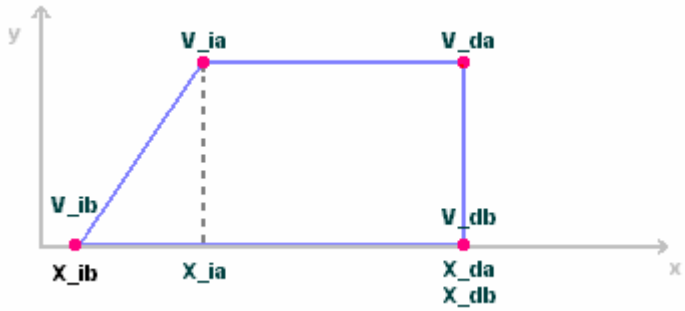
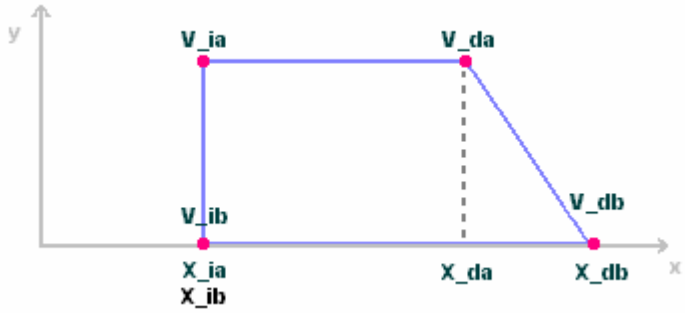
Sobre la figura geométrica de un trapecio se especifican los cuatros extremos que pueden representar las funciones de membresía como se muestra en la siguiente figura:

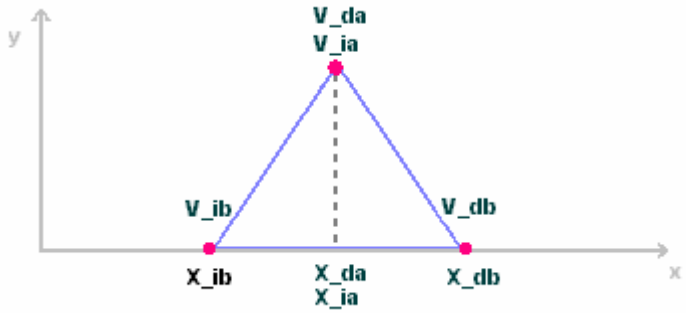
Figura 17:Trapecio en el que se representa funciones de membresía



De esta forma cualquiera de las funciones de membresía (Z, S, Triangular, Trapezoidal) puede ser representada por cuatro puntos, esto se ve en la siguiente tabla:

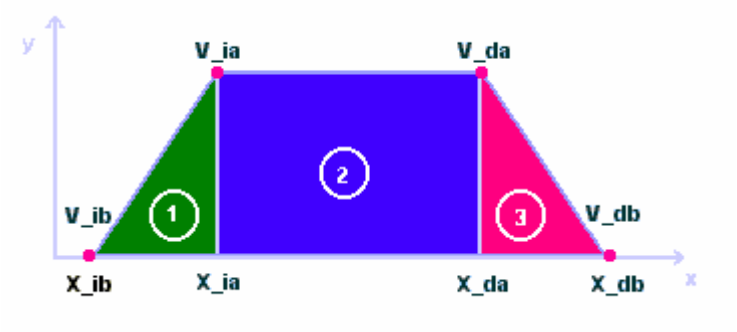
Tabla 3. Funciones de membresía Trapezoidal, S, Z, Triangular.

TIPO DE FUNCIÓN DE MEMBRESÍA	FIGURA	DESCRIPCIÓN POR ABSCISAS
Trapezoidal		**X_ib** **X_ia** **X_da** **X_db**
S		**X_ib** **X_ia** **X_da = X_db**
Z		**X_ib = X_ia** **X_da** **X_db**

Triangular		**X_ib** **X_ia = X_da** **X_db**

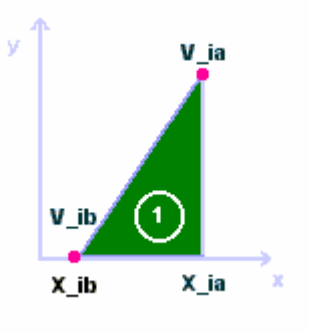
Cada una de las funciones de membresía se segmenta de tal manera que puede evaluarse el grado de pertenencia de un valor de entrada a un conjunto Fuzzy dado, resolviendo la ecuación de una recta o dándole un valor constante según el segmento que corresponda. Esto es; para la función de membresía trapezoidal se tienen los siguientes segmentos de recta y sus respectivas ecuaciones:

Figura 18: Segmentos de recta para la función de membresía trapezoidal



Recta A. En este segmento el grado de membresía se evalúa mediante la ecuación de la recta ascendente en función de los puntos X_{ib} y X_{ia} como se describe a continuación:

Figura 19. Recta A.



$$\mu(x) = \frac{(x - x_{ib})}{x_{ia} - x_{ib}} \qquad x \in (x_{ib}, x_{ia})$$

El código de programa que realiza esta operación para los microcontrolador PIC16F873 y MC68HC908JL3 es de la siguiente forma:

**********PIC16F873**********

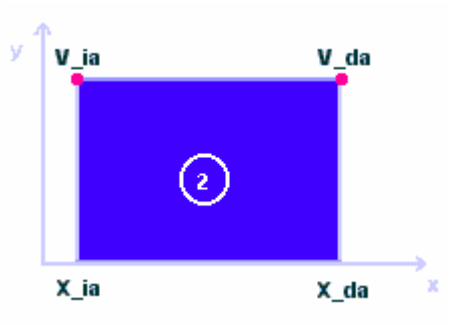
```
RECT_A
        movf    Xib_lsb,0
        movwf   Q2_lsb
        movf    Xib_msb,0
        movwf   Q2_msb
        call    REST@
        movlw   0x03
        movwf   Q2_msb
        movlw   0xFF
        movwf   Q2_lsb
        call    M_MSNUM
        movf    Xia_lsb,0
        movwf   Q1_lsb
        movf    Xia_msb,0
        movwf   Q1_msb
        movf    Xib_lsb,0
        movwf   Q2_lsb
        movf    Xib_msb,0
        movwf   Q2_msb
        call    REST@
        movf    Q1_lsb,0
        movwf   SDEN_1
        movf    Q1_msb,0
        movwf   SDEN_2
        call    DIV@
        bsf     DNUMDEN_2,7
        return
IF_3
```

```
                movf    Xda_lsb,0
                movwf   Q2_lsb
                movf    Xda_msb,0
                movwf   Q2_msb
                call    COMP@
                xorlw   0x01
                btfss   STATUS,2
                goto    CTE

********************************* MC68HC908JL3*********************
RECT_A
  lda    Xia_lsb
  sub    Xib_lsb
  sta    Q3_1
  lda    Q1_lsb
  sub    Xib_lsb
  ldx    #$80
  mul
  pshx
  pulh
  ldx    Q3_1
  div
  sta    DNUMDEN_1
  jmp    SALE_0
IF_3
  cmp    Xda_lsb
  bhi    RECT_D
  jmp    CTE
```

Constante. El valor constante que se da en este segmento para una función de membresía normalizada es la unidad (1), sin embargo, puede escalarse a otro valor.

Figura 20. Constante



$$\mu(x) = 1 \quad x \in [x_{ia}, x_{da}]$$

El código de programa que realiza esta □ontinuac en los □ontinuación□dores PIC y Motorola es el siguiente:

CONSTANTE

********** PIC16F873**********

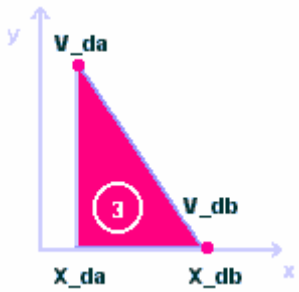
```
        CTE
                movlw   0x03
                movwf   DNUMDEN_2
                movlw   0xFF
                movwf   DNUMDEN_1
                bsf     DNUMDEN_2,7
        BRE2B   return
```

********** MC68HC908JL3**********

```
CTE
      mov   #$7F,DNUMDEN_1
SALE_0   bset 7, DNUMDEN_2
      jmp   FUERA
SALE_1
      bclr  7,DNUMDEN_2
FUERA
  rts
```

Recta D. La □ontinua de la recta que sirve para evaluar este segmento y la gráfica que lo representa se muestran a □ontinuación:

Figura 21. Recta D



$$\mu(x) = \frac{(x_{db} - x)}{(x_{db} - x_{da})} \quad x \in (x_{da}, x_{db})$$

El código fuente que realiza esta operación es el siguiente:

RECTA_D

**********PIC16F873**********

```
RECT_D
        movf    Q1_lsb,0
        movwf   Q2_lsb
        movf    Q1_msb,0
        movwf   Q2_msb
        movf    Xdb_lsb,0
        movwf   Q1_lsb
        movf    Xdb_msb,0
        movwf   Q1_msb
        call    REST@
        movlw   0x03
        movwf   Q2_msb
        movlw   0xFF
        movwf   Q2_lsb
        call    M_MSNUM
        movf    Xdb_lsb,0
        movwf   Q1_lsb
        movf    Xdb_msb,0
        movwf   Q1_msb
        movf    Xda_lsb,0
        movwf   Q2_lsb
        movf    Xda_msb,0
        movwf   Q2_msb
        call    REST@
        movf    Q1_lsb,0
        movwf   SDEN_1
        movf    Q1_msb,0
        movwf   SDEN_2
        call    DIV@
        bsf     DNUMDEN_2,7
        return
```

********** MC68HC908JL3**********

```
RECT_D
 lda    Xdb_lsb
 sub    Xda_lsb
 sta    Q3_1
 lda    Xdb_lsb
 sub    Q1_lsb
 ldx    #$80
 mul
 pshx
 pulh
 ldx    Q3_1
```

```
 div
 sta   DNUMDEN_1
 jmp   SALE_0
```

2.4.1.2. Algoritmo de Membresía

Este algoritmo toma un dato de entrada "X" y un vector "VÉRTICES" con los cuatro (4) valores que describen una función de membresía en la caracterización del controlador. Se selecciona el segmento del dominio en el cual coincide el valor de entrada y conforme a este evalúa la ecuación correspondiente para dar el grado de pertenencia.

El código fuente de la rutina de membresía es de la siguiente forma:

MEMBRESÍA

```
**************************PIC16F873**********************************
        MEMBR    movf    N_VERT,0
                 call    VERTICES
                 movwf   Xib_lsb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xib_msb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xia_lsb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xia_msb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xda_lsb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xda_msb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xdb_lsb
                 incf    N_VERT,1
                 movf    N_VERT,0
                 call    VERTICES
                 movwf   Xdb_msb
                 incf    N_VERT,1
        IF_0     movf    Xib_lsb,0
                 movwf   Q2_lsb
                 movf    Xib_msb,0
```

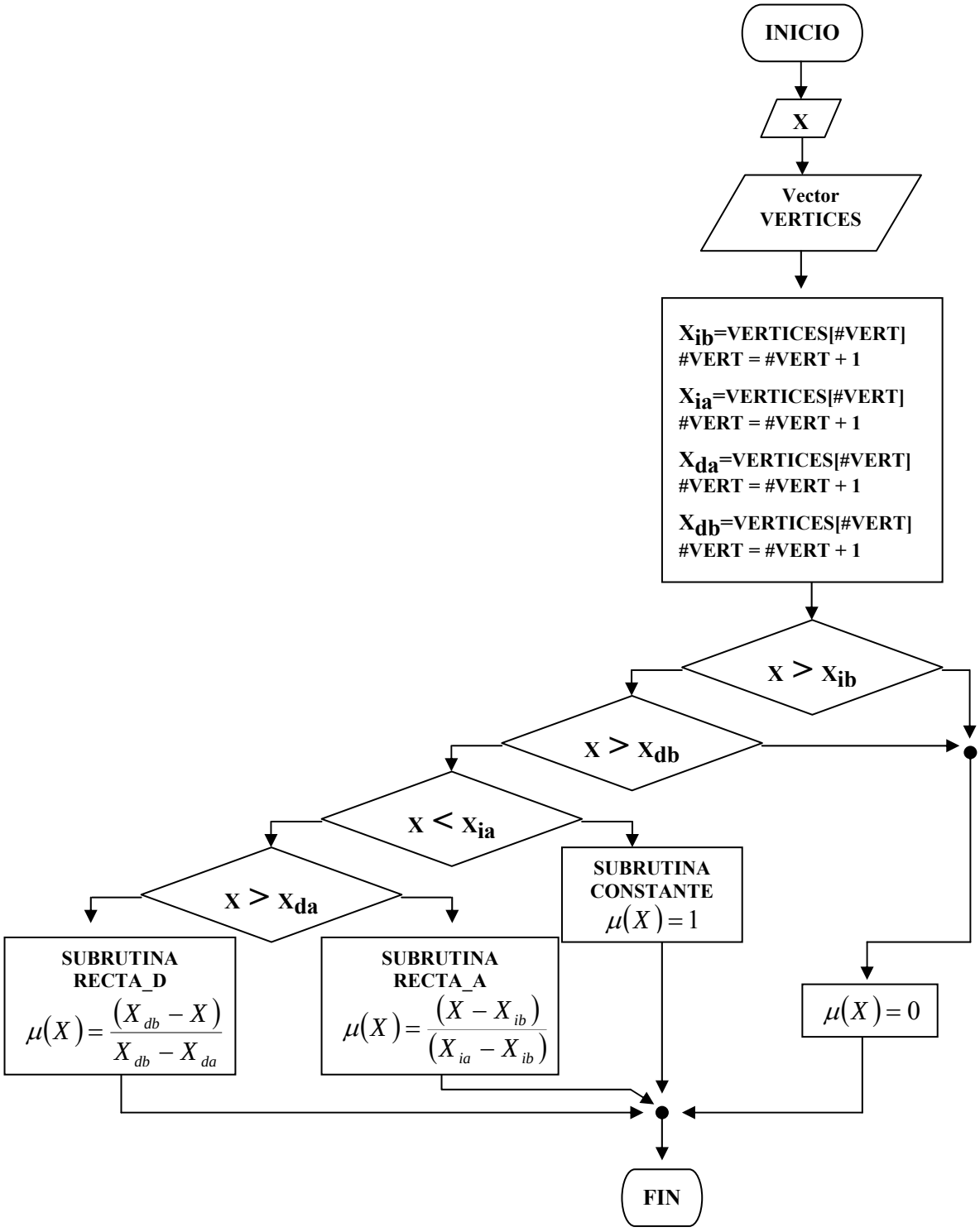
```
                movwf   Q2_msb
                call    COMP@
                xorlw   0x01
                btfsc   STATUS,2
                goto    IF_1
                bcf     DNUMDEN_2,7
                return
        IF_1    movf    Xdb_lsb,0
                movwf   Q2_lsb
                movf    Xdb_msb,0
                movwf   Q2_msb
                call    COMP@
                xorlw   0x01
                btfss   STATUS,2
                goto    IF_2
                bcf     DNUMDEN_2,7
                return
        IF_2    movf    Xia_lsb,0
                movwf   Q2_lsb
                movf    Xia_msb,0
                movwf   Q2_msb
                call    COMP@
                xorlw   0x02
                btfss   STATUS,2
                goto    IF_3
```

**********MC68HC908JL3**********

```
MEMBR
      clrh
      ldx    N_VERT
 lda    VERTICES,x
 sta    Xib_lsb
 inc    N_VERT
 ldx    N_VERT
 lda    VERTICES,x
 sta    lefttop_lsb
 inc    N_VERT
 ldx    N_VERT
 lda    VERTICES,x
 sta    Xda_lsb
 inc    N_VERT
 ldx    N_VERT
 lda    VERTICES,x
 sta    Xdb_lsb
      inc    N_VERT
IF_0  lda    Q1_lsb
      cmp    Xib_lsb
      blo    SALE_1
IF_1  cmp    Xdb_lsb
          bhi    SALE_1
IF_2      cmp    Xia_lsb
          blo    RECT_A
          jmp    IF_3
```

El diagrama de flujo resumido de la subrutina Membresía se muestra a continuación:

Figura 22: Diagrama de flujo subrutina de membresía



2.4.2. Base de Conocimiento

Esta formada por dos componentes básicos: Base de Datos y Base de Reglas.

2.4.2.1. Base de Datos

Uno de los conceptos importantes, antes de caracterizar las reglas de control *Fuzzy*, es la definición de los límites de los conjuntos Fuzzy con los cuales se representa el conocimiento. Por lo general, estos límites se definen subjetivamente con base en el conocimiento de los expertos.

Para el controlador *Fuzzy,* realizado en esta tesis, se relaciona un vector de datos que especifican los contornos de los conjuntos *Fuzzy* mediante cuatro datos cada uno. Los datos que definen las funciones de membresía se almacenan secuencial mente en el siguiente orden:

Para las variables lingüísticas de entrada:

X_ib_{11} ; Abscisa del vértice **izquierdo bajo** del primer término de la primer variable lingüística.
X_ia_{11} ; Abscisa del vértice **izquierdo alto** del primer término de la primer variable lingüística.
X_da_{11}; Abscisa del vértice **derecho alto** del primer término de la primer variable lingüística.
X_db_{11}; Abscisa del vértice **derecho bajo** del primer término de la primer variable lingüística.
X_ib_{21} ; Abscisa del vértice **izquierdo bajo** del segundo término de la primer variable lingüística.
X_ia_{21} ; Abscisa del vértice **izquierdo alto** del segundo término de la primer variable lingüística.
X_da_{21}; Abscisa del vértice **derecho alto** del segundo término de la primer variable lingüística.
X_db_{21}; Abscisa del vértice **derecho bajo** del segundo término de la primer variable lingüística.
. . .
X_ib_{mn}; Abscisa del vértice **izquierdo bajo** del enésimo término de la enésima variable lingüística.
X_ia_{mn} ; Abscisa del vértice **izquierdo alto** del enésimo término de la enésima variable lingüística.
X_da_{mn}; Abscisa del vértice **derecho alto** del enésimo término de la enésima variable lingüística.
X_db_{mn}; Abscisa del vértice **derecho bajo** del enésimo término de la enésima variable lingüística.

El código en lenguaje ensamblador para el PIC es de la siguiente forma:

```
VERTICES
        addwf   PCL,1
        retlw   0x00
        retlw   0x00
```

```
retlw   0x00
retlw   0x00
retlw   0xEC
retlw   0x01
retlw   0x00
retlw   0x02 ...
```

El código en lenguaje ensamblador para el microcontrolador Motorola es de la siguiente forma:

```
VERTICES
cbeqa   #!0,cero
 cbeqa   #!1,cero
 cbeqa   #!2,cero
 cbeqa   #!3,medb
 cbeqa   #!4,medb
cbeqa   #!5,medb
 cbeqa   #!6,alta
 cbeqa   #!7,meda
```

Para la variable linguistica de salida:

$Xsing_1$; Abscisa del primer *singleton.*
...
$Xsing_n$; Abscisa del enésimo *singleton.*

Dado que se utiliza una sola variable de salida en el controlador *Fuzzy* diseñado en este proyecto, las abscisas de los singletones de dicha variable se introducen al programa como constantes, en el segmento de programa de Defuzificación.

2.4.2.2. Base de Reglas

La base de reglas es un conjunto de *m* reglas, cada una de las cuales es de la forma:

IF *(entrada 1 es conjunto* i_1 ***AND*** *entrada 2 es conjunto* i_2 ***THEN*** *(salida 1 es conjunto* i_1*)*

En donde *conjunto* i_j es uno de los Valores Lingüísticos que puede tomar la variable (de entrada o de salida) j .

En cada regla pueden distinguirse dos partes: el Antecedente y el Consecuente; de tal forma que cada regla puede escribirse en forma abreviada como:

***IF** (Antecedente) **THEN** (Consecuente).*

Si las *m* reglas cubren todas las posibles combinaciones de Valores Lingüísticos de los Antecedentes, se dice que la Base de Reglas es Completa. En ningún caso puede permitirse que existan dos reglas con el mismo Antecedente.

Las reglas se incorporan al algoritmo del controlador *fuzzy* de manera implícita, dentro del algoritmo de inferencia. Esto se verá a continuación.

2.4.3. Etapa de Inferencia Fuzzy

Esta etapa es usualmente conocida como el motor de inferencia en los sistemas *fuzzy*, y hacen parte de ella los procedimientos destinados a la toma de las acciones de control *fuzzy* por medio de la implicación y las reglas *fuzzy*. Se utilizan las reglas contenidas la etapa anterior para realizar el control.

2.4.3.1. Algoritmo de Inferencia Fuzzy

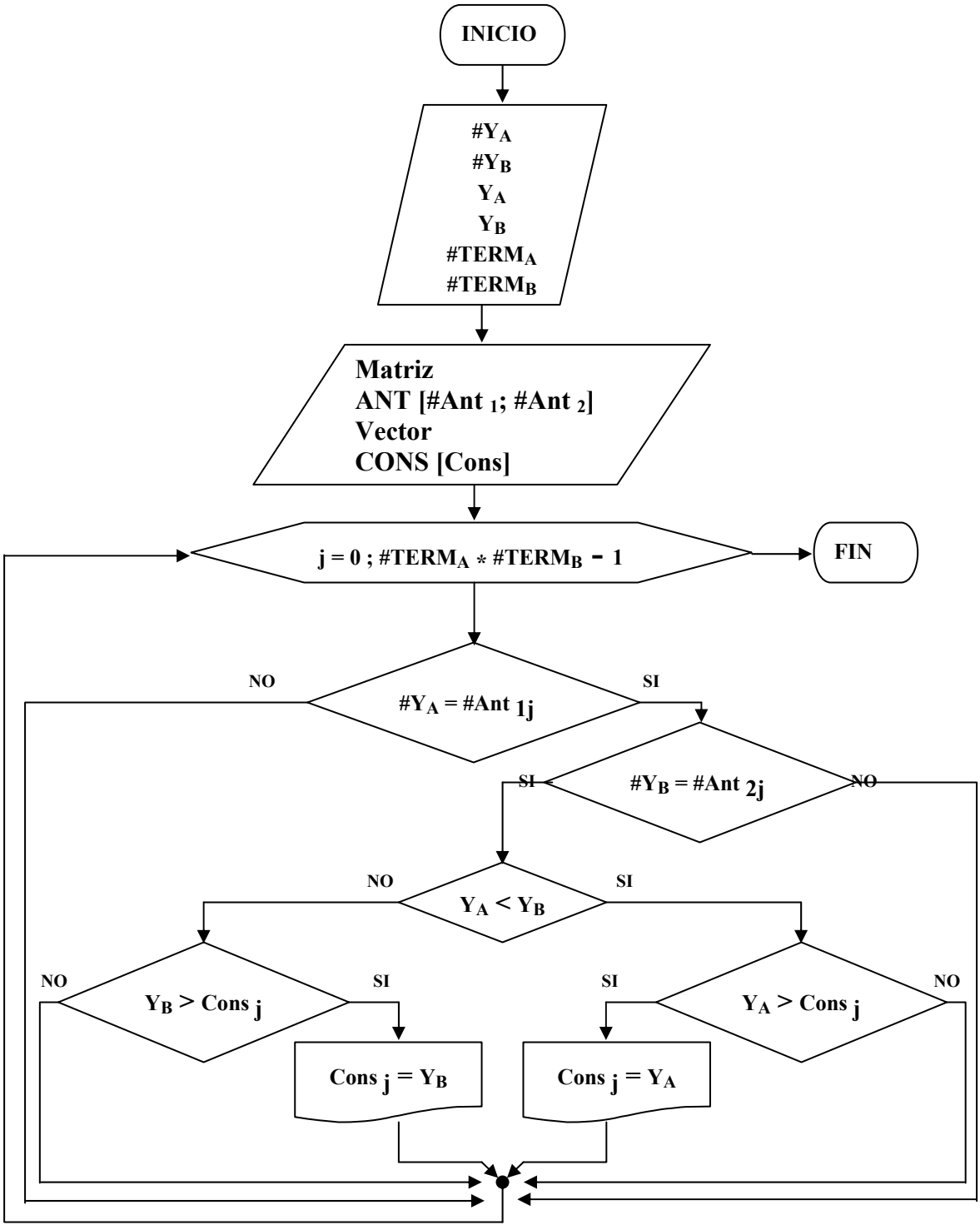
A este algoritmo toma como entrada:

- Los valores Y_A, Y_B,... Y_N, que representan el grado de pertenencia a los conjuntos *fuzzy* de los valores de entrada X_A, X_B,... X_N, respectivamente.

- Los números $\#Y_A$, $\#Y_B$,... $\#Y_N$ con los cuales se distinguen los conjuntos *fuzzy*, estos números son tomados como los términos lingüísticos dentro del algoritmo *fuzzy* desarrollado en esta tesis.

- $\#TERM_A$, $\#TERM_B$, ... $\#TERM_N$ representan los números totales de términos lingüísticos (o conjuntos *fuzzy*) de las variables lingüísticas A, B, ..., N.

Figura 23: Diagrama de flujo algoritmo de inferencia



El código fuente de la rutina que realiza la etapa de inferencia *fuzzy* es el siguiente:

INFERENCIA (REGLAS)

**********PIC**********

```
REGLAS
                call    MINMAX
                movf    N_TERM_0,0
                addwf   N_TERM_0,0
                addwf   N_TERM_0,0
                addwf   N_TERM_1,0
                addwf   PCL,1
                goto    cero
                goto    cero
                goto    cero
                goto    medb
                goto    medb
                goto    medb
                goto    alta
                goto    meda
                goto    med
                goto    max
                goto    alta
                goto    meda
                goto    max
                goto    max
                goto    max
        max
                movf    singleton1_lsb,0
                movwf   Q1_lsb
                movf    singleton1_msb,0
                movwf   Q1_msb
                call    MINMAX
                movf    Q1_lsb,0
                movwf   singleton1_lsb
                movf    Q1_msb,0
                movwf   singleton1_msb
                return
        alta
                movf    singleton2_lsb,0
                movwf   Q1_lsb
                movf    singleton2_msb,0
                movwf   Q1_msb
                call    MINMAX
                movf    Q1_lsb,0
                movwf   singleton2_lsb
                movf    Q1_msb,0
                movwf   singleton2_msb
                return
        meda
                movf    singleton3_lsb,0
                movwf   Q1_lsb
                movf    singleton3_msb,0
```

```
                movwf   Q1_msb
                call    MINMAX
                movf    Q1_lsb,0
                movwf   singleton3_lsb
                movf    Q1_msb,0
                movwf   singleton3_msb
                return
        med
                movf    singleton4_lsb,0
                movwf   Q1_lsb
                movf    singleton4_msb,0
                movwf   Q1_msb
                call    MINMAX
                movf    Q1_lsb,0
                movwf   singleton4_lsb
                movf    Q1_msb,0
                movwf   singleton4_msb
                return
        medb
                movf    singleton5_lsb,0
                movwf   Q1_lsb
                movf    singleton5_msb,0
                movwf   Q1_msb
                call    MINMAX
                movf    Q1_lsb,0
                movwf   singleton5_lsb
                movf    Q1_msb,0
                movwf   singleton5_msb
                return
        cero
                movf    singleton6_lsb,0
                movwf   Q1_lsb
                movf    singleton6_msb,0
                movwf   Q1_msb
                call    MINMAX
                movf    Q1_lsb,0
                movwf   singleton6_lsb
                movf    Q1_msb,0
                movwf   singleton6_msb
        BRE3B   return
```

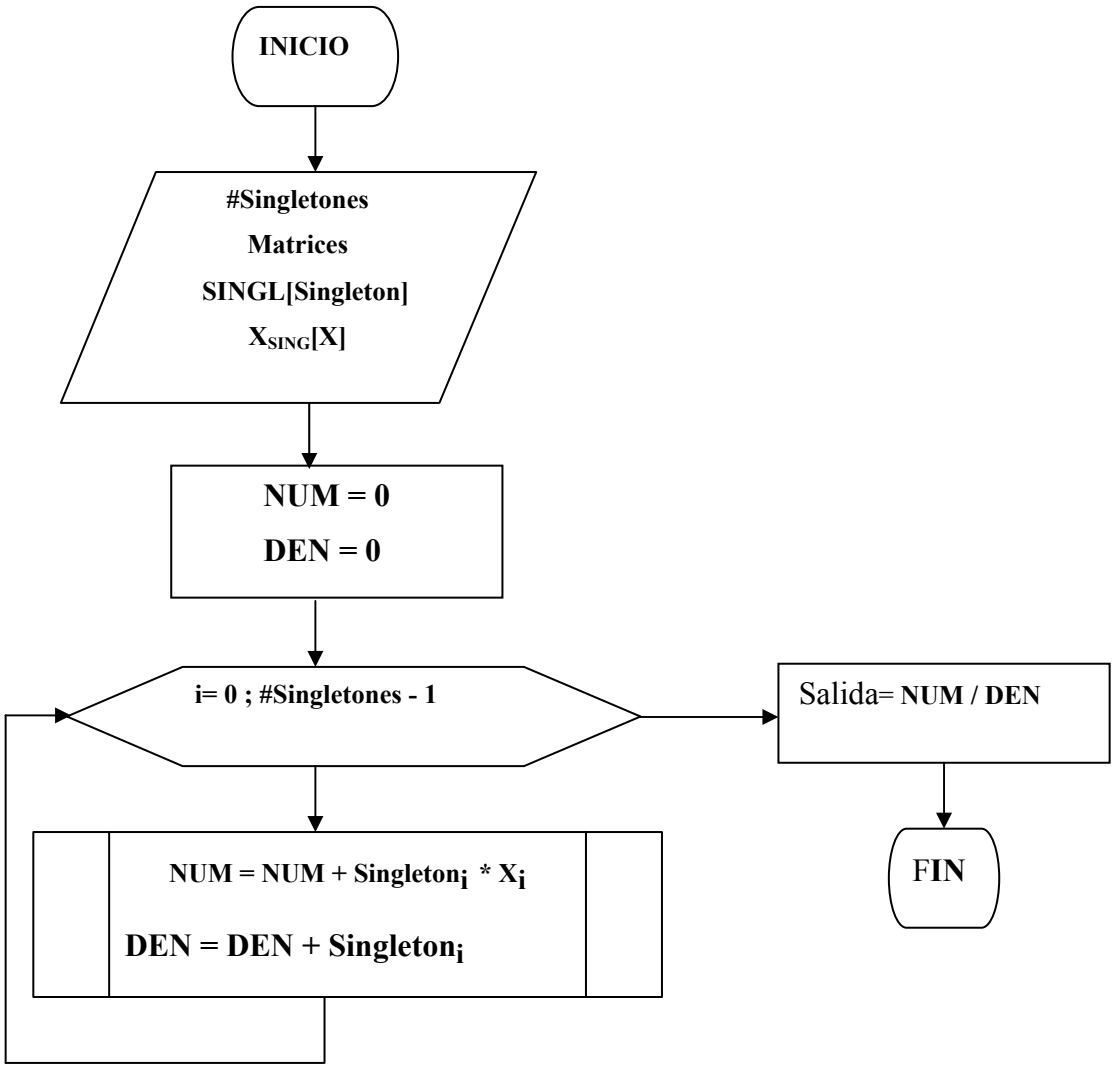
********** MC68HC908JL3**********

```
REGLAS
  jsr   MINMAX
  lda   N_TERM_0
 ldx   #!3
  mul
 add   N_TERM_1
cbeqa   #!0,cero
cbeqa   #!1,cero
cbeqa   #!2,cero
cbeqa   #!3,medb
cbeqa   #!4,medb
cbeqa   #!5,medb
cbeqa   #!6,alta
cbeqa   #!7,meda
cbeqa   #!8,med
cbeqa   #!9,max
cbeqa   #!10,alta
```

```
 cbeqa   #!11,meda
 cbeqa   #!12,max
 cbeqa   #!13,max
 cbeqa   #!14,max
max
   mov     singleton1,Q1_lsb
   jsr     MINMAX
   mov     Q1_lsb,singleton1
   jmp     RETORNA
alta
   mov     singleton2,Q1_lsb
   jsr     MINMAX
   mov     Q1_lsb,singleton2
   jmp     RETORNA
meda
   mov     singleton3,Q1_lsb
   jsr     MINMAX
   mov     Q1_lsb,singleton3
   jmp     RETORNA
med
   mov     singleton4,Q1_lsb
   jsr     MINMAX
   mov     Q1_lsb,singleton4
   jmp     RETORNA
medb
   mov     singleton5,Q1_lsb
   jsr     MINMAX
   mov     Q1_lsb,singleton5
   jmp     RETORNA
cero
   mov     singleton6,Q1_lsb
   jsr     MINMAX
   mov     Q1_lsb,singleton6
RETORNA
 rts
```

2.4.4. Etapa de Defuzificación

Figura 24: Diagrama de flujo algoritmo de Defuzificación



2.4.4.1. Algoritmo de Centroide

El código ensamblador para este algoritmo es el siguiente:

DEFUZIFICACION(CENTROIDE)

**********PIC**********

```
            CENTROID
            clrf    MSNUM_1
            clrf    MSNUM_2
            clrf    MSNUM_3
            clrf    SDEN_1
            clrf    SDEN_2
            clrf    SDEN_3
            clrf    DNUMDEN_1
            clrf    DNUMDEN_2
c_max       movf    singleton1_lsb,0
            movwf   Q1_lsb
            movf    singleton1_msb,0
            movwf   Q1_msb
            iorwf   singleton1_lsb,0
            btfsc   STATUS,2
            goto    c_alta
            call    SUM@I
            movlw    0x00
            movwf   Q2_lsb
            movlw    0x00
            movwf   Q2_msb
            call    MULSUM@S
c_alta      movf    singleton2_lsb,0
            movwf   Q1_lsb
            movf    singleton2_msb,0
            movwf   Q1_msb
            iorwf   singleton2_lsb,0
            btfsc   STATUS,2
            goto    c_meda
            call    SUM@I
            movlw    0x1C
            movwf   Q2_lsb
            movlw    0x01
            movwf   Q2_msb
            call    MULSUM@S
c_meda      movf    singleton3_lsb,0
            movwf   Q1_lsb
            movf    singleton3_msb,0
            movwf   Q1_msb
            iorwf   singleton3_lsb,0
            btfsc   STATUS,2
            goto    c_med
            call    SUM@I
            movlw    0x77
            movwf   Q2_lsb
            movlw    0x01
            movwf   Q2_msb
            call    MULSUM@S
```

```
        c_med   movf    singleton4_lsb,0
                movwf   Q1_lsb
                movf    singleton4_msb,0
                movwf   Q1_msb
                iorwf   singleton4_lsb,0
                btfsc   STATUS,2
                goto    c_medb
                call    SUM@I
                movlw    0x00
                movwf   Q2_lsb
                movlw    0x02
                movwf   Q2_msb
                call    MULSUM@S
        c_medb  movf    singleton5_lsb,0
                movwf   Q1_lsb
                movf    singleton5_msb,0
                movwf   Q1_msb
                iorwf   singleton5_lsb,0
                btfsc   STATUS,2
                goto    c_cero
                call    SUM@I
                movlw    0x38
                movwf   Q2_lsb
                movlw    0x02
                movwf   Q2_msb
                call    MULSUM@S
        c_cero  movf    singleton6_lsb,0
                movwf   Q1_lsb
                movf    singleton6_msb,0
                movwf   Q1_msb
                iorwf   singleton6_lsb,0
                btfsc   STATUS,2
                goto    SALIDA
                call    SUM@I
                movlw    0xFF
                movwf   Q2_lsb
                movlw    0x03
                movwf   Q2_msb
                call    MULSUM@S
        SALIDA  call    DIV@
                movf    DNUMDEN_1,0
                movwf   SALIDA_1
                movf    DNUMDEN_2,0
                movwf   SALIDA_2

********************** MC68HC908JL3**********************
;       Defuzzyficación
CENTROID
  clra
  clr    MSNUM_1
  clr    MSNUM_2
  clr    SDEN_1
  clr    DNUMDEN_1

  lda    SDEN_1
  add    singleton1      ;*h1
  sta    SDEN_1
  lda    singleton1      ;*h1
  ldx    #$00       ;*a1
  mul
```

```
 jsr    suma16     ;
 lda    SDEN_1
 add    singleton2     ;*h2
 sta    SDEN_1
 lda    singleton2     ;*h2
 ldx    #$47       ;*a2
 mul
 jsr    suma16
 lda    SDEN_1
 add    singleton3     ;*h3
 sta    SDEN_1
 lda    singleton3     ;*h3
 ldx    #$5E       ;*a3
 mul
 jsr    suma16
 lda    SDEN_1
 add    singleton4     ;*h4
 sta    SDEN_1
 lda    singleton4     ;*h4
 ldx    #$80       ;*a4
 mul
 jsr    suma16
 lda    SDEN_1
 add    singleton5     ;*h5
 sta    SDEN_1
 lda    singleton5     ;*h5
 ldx    #$8E       ;*a5
 mul
 jsr    suma16
 lda    SDEN_1
 add    singleton6     ;*h6
 sta    SDEN_1
 lda    singleton6     ;*h6
 ldx    #$FF       ;*a6
 mul
 jsr    suma16     ;sumar MSNUM = (X:A)+MSNUM
*** DNUMDEN = (MSNUM/SDEN_1) **************
       ldhx    MSNUM_2
       lda     MSNUM_1
       ldx     SDEN_1
       div
       sta     SALIDA
  rts
```

2.5. FORMATO DE PROGRAMA DE CONTROL FUZZY

El programa fue implementado inicialmente en un microcontrolador PIC16F873 de MICROCHIP y luego se tradujo su código de programación a un microcontrolador MC68HC908 de MOTOROLA. Los resultados de las pruebas se presentan en el Anexo 2.

3. PROGRAMA PARA LA GENERACIÓN DEL CÓDIGO ENSAMBLADOR DEL CONTROLADOR FUZZY

El programa desarrollado para el diseño de controladores con lógica *Fuzzy* y la generación de su código en lenguaje ensamblador se denominó **FUZZYE3T.**

Este es un software para elaborar la programación básica de controladores con lógica *Fuzzy* para ser implementada en microcontroladores de ocho (8) bits de la familia PIC (gamas básica y media) de MICROCHIP y de la familia CPU08 de MOTOROLA, a través de una interfase gráfica de diseño.

El formato de programa en código ensamblador para implementar controladores *Fuzzy* desarrollado en este proyecto y descrito en el capítulo anterior es particularizado para cada aplicación mediante el programa **FUZZYE3T**. En este software se realiza la caracterización del controlador *Fuzzy* y automáticamente se genera el código ensamblador para el tipo de microcontrolador seleccionado (PIC o MOTOROLA).

El archivo generado es de extensión ".asm" y puede ser compilado de dos (2) formas: Como un archivo *INCLUDE* para ejecutarse como una subrutina donde sea llamado dentro de un programa principal. O como el formato de programa principal donde se le anexan las demás subrutinas que el diseñador estime necesarias para el desarrollo de su aplicación en particular.

El programa finalmente implementado por el diseñador (usuario de FUZZYE3T) debe estar en capacidad de controlar el proceso para el cual se particularizó, de forma autónoma desde el microcontrolador. Un ejemplo de esto se muestra en el capítulo

cuatro (4) de esta tesis, en el cual se presenta el control *Fuzzy* de la temperatura de una resistencia de calefacción.

El lenguaje de programación utilizado es LabVIEW de la empresa National Instruments. Para lo cual, el Centro de Investigación del Gas y la Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones patrocinaron este proyecto colocando a su disposición los siguientes paquetes de software:

- NI LabVIEW *Profesional development System*
- NI *Fuzzy Logic Toolkit* for LabVIEW

El entorno de desarrollo y ejecución es bajo Microsoft Windows 98 Segunda Edición, pero podría ser trasladado a Microsoft Windows 9x, 2000, XP sin ningún problema según especificaciones de LabVIEW.

Además, se han utilizado paquetes gráficos como *Paint* para la creación y retoque de imágenes.

A continuación se presentan la descripción funcional y la estructura de programación de los módulos implementados en el programa **FUZZYE3T**.

3.1. SOFTWARE FUZZYE3T

Este software presenta las siguientes características principales:

✂ Posee una interfase gráfica para la caracterización de un tipo de controlador *fuzzy*, de fácil comprensión y manejo para ingenieros.

- Permite hacer chequeo de las características de entrada / salida de los controladores *fuzzy* diseñados, para el afinamiento de sus parámetros antes de ser implementados.

- Ayuda en línea para que el usuario resuelva dudas acerca del funcionamiento del programa.

- Generación automática de un reporte en formato html de la caracterización del controlador *fuzzy* diseñado y con opción de impresión de dicho reporte.

- Programa ejecutable e instalable, compilado mediante la herramienta de software denominada *Application Builder* y que no requiere que el entorno de programación de LabVIEW se encuentre instalada para funcionar; para esto ultimo sólo necesita el sistema operativo y el software *Runtime* (de LabVIEW).

Dado que este programa tiene como base el algoritmo de control *Fuzzy* descrito en el capítulo anterior, las características de éste son válidas en cuanto al tipo de control y se citan a continuación:

- Tipo de inferencia *fuzzy*: Condicional de reglas del tipo **SI** A es F **Y** B es G **ENTONCES** S es H.

- Tipo de funciones de membresía: Triangular, Z, S, y Trapezoidal.

- Método de Defuzificación: Centroide.

- Método de implicación *fuzzy*: Mínimo – máximo.

- Reglas por defecto: La regla anteriormente usada y la de salida definida por el usuario.

- ✂ Variables de entrada: Admite dos (2) variables de entrada para la fuzificación.

- ✂ Variables de salida: Entrega el valor de una (1) variable de salida luego de la defuzificación.

- ✂ Términos lingüísticos de las variables de entrada: El algoritmo no esta limitado a un número estricto de términos lingüísticos, pero se recomienda que la suma de los términos de todas las variables de entrada sea menor o igual a ocho (8).

- ✂ Términos lingüísticos de la variable de salida: Máximo (9) términos lingüísticos.

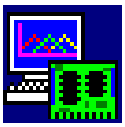
Además, se recomienda utilizar nombres de pocos caracteres (máximo 6 caracteres) para las variables de entrada y salida del controlador *Fuzzy* que se vaya a diseñar con el programa FUZZYE3T, debido a que son los mismos con los cuales se implementa el programa en código ensamblador. Esto último, hace el trabajo más pedagógico en cuánto que puede seguirse el flujo de instrucciones para su comprensión.

En los siguientes apartados se describe la funcionalidad de los elementos que componen el software **FUZZYE3T**.

3.1.1. Presentación del Programa

El programa posee el siguiente icono particular que lo distingue:

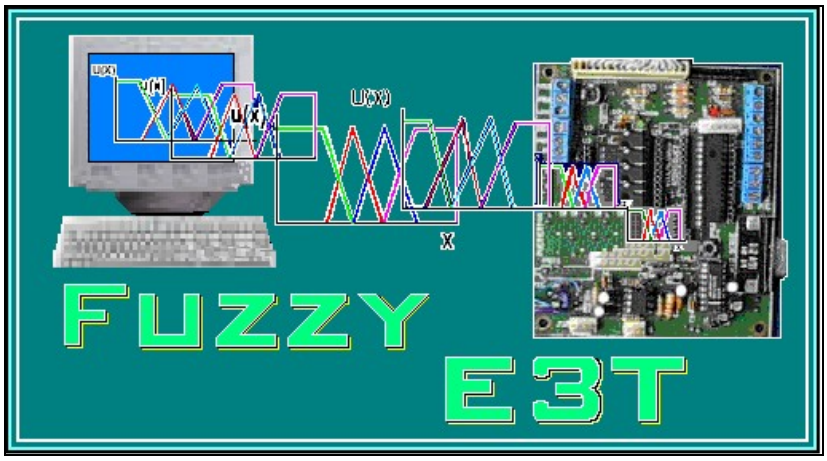
Figura 25: Icono del programa FUZZYE3T



Al iniciar el programa FuzzyE3T V1.0 aparece una animación del logo del programa y luego se abre la ventana principal y la ventana de Inicio. En esta última se tienen tres opciones Abrir, Nuevo y Ayuda.

El logo de presentación, figura 35 muestra la representación de un computador con las gráficas características de las principales funciones de membresía que se utilizan en la caracterización del controlador *fuzzy*. Al lado, la representación de una tarjeta electrónica con dos microcontroladores; haciendo alusión a los microcontroladores PIC y MOTOROLA a los cuales se dirige la implementación del controlador.

Figura 26. Logo del programa FUZZYE3T.



3.1.2. Menú Principal

Figura 27. Menú principal

Este menú consta de seis ítems principales: Archivo, Código Fuente, Conjuntos Fuzzy, Base de Reglas, Características E/S y Ayuda. Cada uno de estos tiene asociado un submenú como se ve a continuación:

Figura 28 . Menú Archivo.

El menú **Archivo** contiene las opciones de:

- Nuevo. Para crear un nuevo archivo de caracterización de controlador *fuzzy*. Asequible también por medio de las teclas Ctrl+N.

- Abrir. Para abrir un archivo de extensión *.fc que contenga la caracterización de un controlador *fuzzy*. Puede usarse las teclas rápidas Ctrl+A.

- ✂ Cerrar. Para cerrar un archivo *.fc. Se puede accesar a esta opción mediante Ctrl+C. Si el archivo ha sufrido cambios el programa pregunta en una caja de diálogo si desea guardar los cambios.

- ✂ Actualizar. A cada cambio hecho en las ventanas de variables y de reglas hay que utilizar esta opcion par refrescar la ventana de variable de salida.

- ✂ Guardar. Para guardar un archivo, también puede utilizarse Ctrl+G.

- ✂ Guardar como. Similar al anterior, pero con la opción de cambiar de nombre al archivo. Sus teclas rápidas son Ctrl+Shift+G.

- ✂ Imprimir. Se accede a un cuadro que permite seleccionar la documentación que se desea imprimir. Ctrl+I también puede ser utilizado para tal fin.

- ✂ Salir. Con esta opción se sale del programa FUZZYE3T, no sin antes preguntar si desea aplicar los cambios que se hayan hecho al archivo trabajado.

El menú **Herramientas** tiene dos (2) opciones para generar el código del controlador *fuzzy* en lenguaje ensamblador, uno para la familia de microcontroladores PIC (gamas básica y media) de la Empresa MICROCHIP y otra para la familia CPU08 de microcontroladores MOTOROLA. Los códigos de programa generados con estas opciones son compatibles con los software MPLAB e ICS08JLZ respectivamente.

Figura 29. Menú Código Fuente.

El menú **Conjuntos Fuzzy** permite dos opciones principales: Especificar y Definir. Con la primera se puede Nombrar variables, nombrar términos y editar los rangos de las variables lingüísticas, Adicionar términos después y antes del término activo (seleccionado desde la interfaz gráfica), remover términos.

Figura 30. Menú Editor de Conjuntos Fuzzy.

El submenú Conjuntos Fuzzy aplica tanto para las dos variables de entrada como para la variable de salida (a excepción de la ultima opción de este submenú, Salida en Singletones, que solo aplica a la variable de salida), la variable a la cual se aplicara cada opción de este submenú depende de la ventana que se encuentre activa; la activación consiste en hacer doble clic sobre la ventana (en BASE DE REGLAS se debe hacer un clic en la parte superior de dicha ventana). Cada una de las cuatro subventanas PRIMERA VARIABLE DE ENTRADA, SEGUNDA VARIABLE DE ENTRADA, VARIABLE DE SALIDA Y REGLAS al ser activa se resalta su apariencia, de lo contrario será opaca.

Luego de hacer un cambio en una de las ventanas anteriormente mencionadas, si se quiere que dicho cambio sea aceptado, es necesario seleccionar la opción de Actualizar en el submenu Archivo.

El menú **Base de Reglas**. Con este menú se accede a otro: Reglas sin activar, con el cual se puede configurar que valor tomar cuando una regla no se haya definido.

Figura 31. Menú Base de Reglas.

El menú **Características E/S**. En este ítem se activa la ventana Visor de características además permite ingresar el número de puntos que se desea graficar en la característica de entrada / salida del controlador *Fuzzy*.

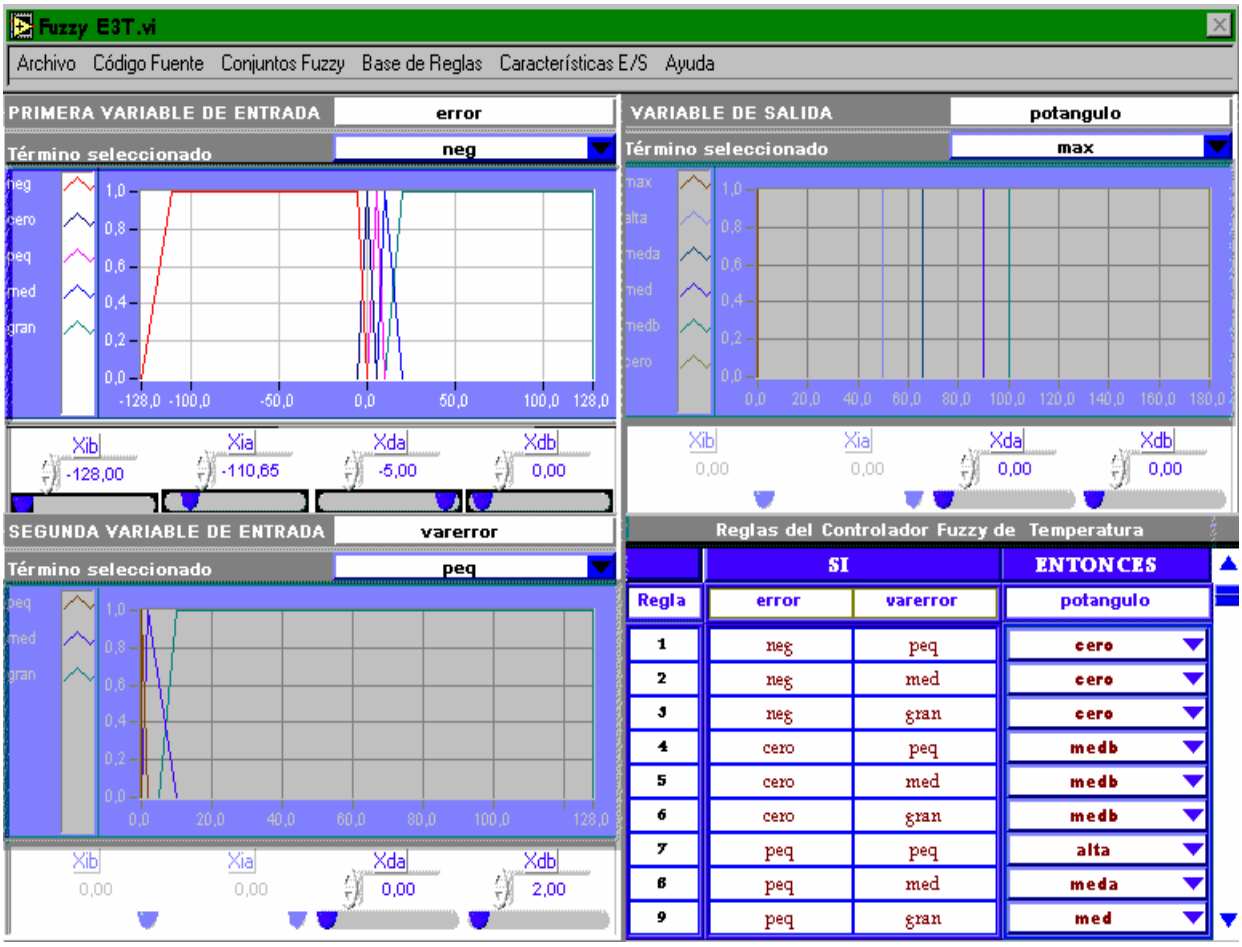
. Figura 32. Menú Características E/S.

El menú **Ayuda** presenta las siguientes opciones: Ayuda general del Programa FUZZYE3T y Acerca de. Como se ve en la siguiente figura.

Figura 33. Menú Ayuda

3.1.3. Interfase Gráfica de Caracterización del Controlador Fuzzy

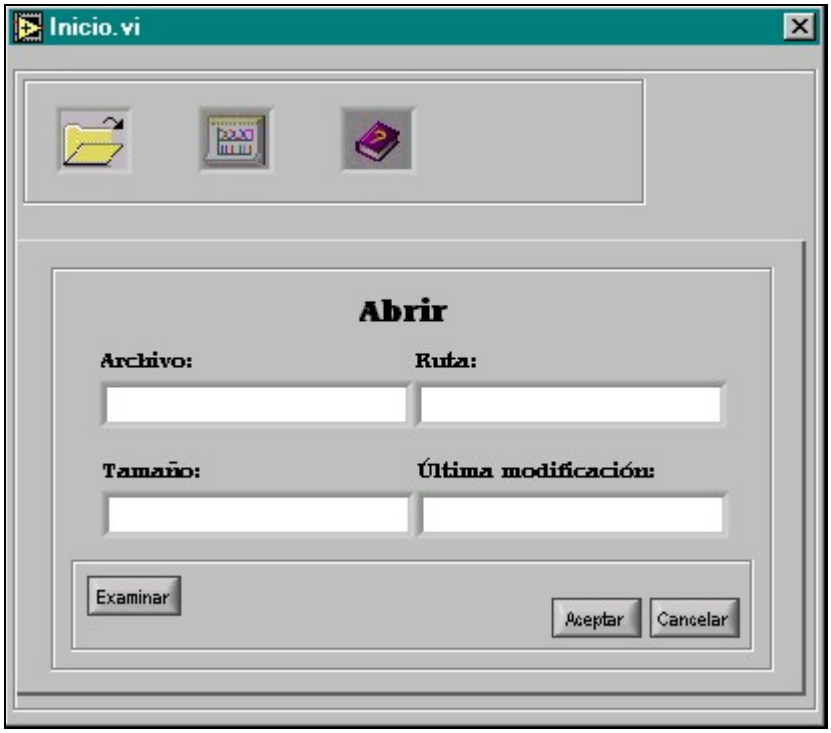
Figura 34. Interfaz gráfica de diseño del programa FUZZYE3T.



Esta interfase se basa en el paquete de software *Fuzzy Logic for G Toolkit*. Al iniciar el programa aparece una ventana de inicio sobre el entorno de diseño de los controladores *Fuzzy*, esta ventana tiene tres despliegues *setups* diferentes concernientes a las tres

funciones que puede desempeñar; estas son: Abrir, Nuevo y Ayuda. Estos pueden activarse mediante los tres íconos que se encuentran en su parte superior y que representan cada una de esas acciones. Por defecto se inicia con la ventana de Abrir para acceder un archivo de extensión *.fc que contenga la caracterización de un controlador *Fuzzy* que se halla diseñado anteriormente con **FUZZYE3T** o con ***Fuzzy Logic Toolkit***. Su apariencia puede observarse en la siguiente figura.

Figura 35. Ventana de Inicio del programa FUZZYE3T, con la opción de abrir un archivo *.fc.



Para buscar un archivo *.fc existen dos (2) opciones: La primera es llenar los dos primeros espacios en blanco del esquema con el título del archivo y la Ruta del mismo. La segunda opción es haciendo clic en el ítem **Examinar,** con el cual aparece una caja de diálogo para la búsqueda del archivo.

En cualquiera de las opciones anteriores, al encontrar el archivo aparecerán los cuatro espacios en blanco llenos con los datos del archivo: Título del archivo, ruta, Tamaño y última modificación.

Luego, para continuar se procede con el ítem **Aceptar** o si se desea salir de esta ventana con el ítem **Cancelar**.

La ventana de inicio para crear un nuevo archivo toma la siguiente forma:

Figura 36. Ventana de inicio para crear un nuevo archivo de caracterización de controlador *Fuzzy*.

En esta ventana se introduce el Título del Controlador *Fuzzy* que se va a diseñar, el nombre del diseñador, la fecha y la hora de creación del archivo que llevara la caracterización propia del controlador.

Las opciones de Aceptar y cancelar tienen el mismo significado que en la ventana anterior.

La opción de Ayuda que presenta ventana de Inicio es la siguiente:

Figura 37. Ventana de Inicio para obtener la Ayuda del programa FUZZYE3T.

En esta ventana se selecciona un ítem en particular para obtener ayuda y automáticamente aparece la explicación asociada con dicho ítem.

La opción **Acerca de FUZZYE3T** hace alusión a las refencias del programa como tal y a sus autores.
También se provee en esta ventana de las opciones de **Aceptar** y **Cancelar.**

La interfaz gráfica de FUZZYE3T presenta tres (3) ventanas asociadas que permiten editar la información contenida en un archivo de caracterización de un controlador *Fuzzy,* o crear dicha información; estas ventanas son: Conjuntos Fuzzy de las variables linguisticas de entrada, Conjuntos Fuzzy de la variable linguistica de salida y Base de Reglas. Además, se provee de una cuarta ventana que permite observar las Características de E/S del controlador *Fuzzy* que se este diseñando. Estas ventanas son los paneles frontales de los VI's que se [illegible]eguida.
Luego de hacer un cambio en una de las ventanas anteriormente mencionadas, si se quiere que dicho cambio sea aceptado, es necesario seleccionar la opción de Actualizar en el submenu Archivo.

Para que el código se genere correctamente es necesario que las funciones de membresía de los términos de la variable linguistica de salida sean del tipo singleton, esto se logra utilizando los desplazadores de la parte inferior de la ventana VARIABLE DE SALILDA o mediante la opción Salida en Singletones del submenu Conjuntos Fuzzy. Además, se deben tener definidas totalmente las reglas en la BASE DE REGLAS, el programa por defecto coloca el primer termino linguistico de la variable de salida si se pide generar el código en assembler sin definir totalmente las reglas. Dicho termino por defecto puede ser cambiado en el submenu Base de Reglas.

El código generado en assembler para el PIC16F873 contiene además del formato de control fuzzy, unas subrutinas que permiten la comunicación serial entre el microcontrolador y el computador (mediante la interfase diseñada en este proyecto). Esto lo hace un programa completo que sirve como ejemplo para desarrollo de aplicaciones particulares y además permite que se realicen pruebas del funcionamiento del programa Fuzzy en código ensamblador debido a que se puede comparar con el

mismo algoritmo de control fuzzy pero ejecutado en el computador con las herramientas de Fuzzy Logic Toolkit for LabVIEW. Para estas pruebas se creo otra aplicación de LabVIEW denominada Probar Código FuzzyE3T.

El código en ensamblador para el JL3 es únicamente el formato de control fuzzy (subrutinas) que requiere la adición del resto de cuerpo de programa según la aplicación que se desee.

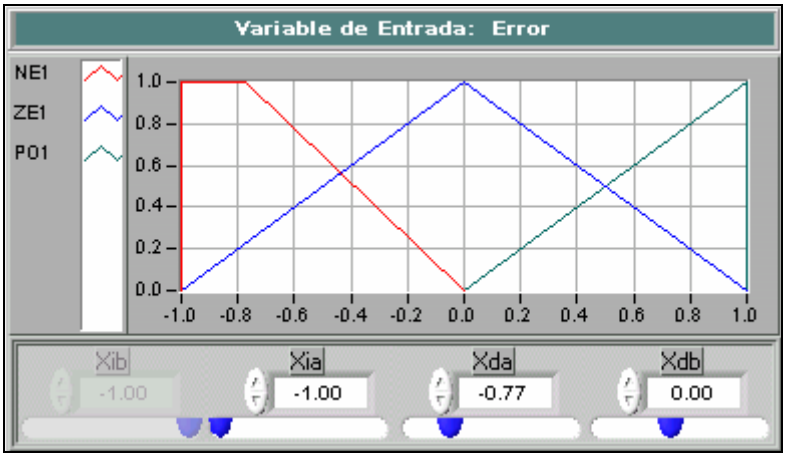
El código generado para cualquiera de los dos tipos de microcontroladores, toma los nombres los nombres de las variables y términos linguisticos dados por el usuario en la caracterización de Controlador. Esto permite hacer un seguimiento del algoritmo de manera mas didáctica.

Nota:
Los archivos de texto generados con extensión *.asm deben ser copiados al editor propia de cada uno de los fabricantes de microcontroladores (Microchip y Motorola) para realizar la compilación correcta. Se recomienda abrirlos con dichos editores de texto, seleccionar todo el texto y pegarlo como un nuevo archivo con un nombre diferente. Esto evitara cualquier tipo de errores de compilación.

3.1.3.1. Editor de Conjuntos Fuzzy.VI

Este VI sirve para editar los conjuntos *fuzzy* de las variables de entrada y salida. Al lado izquierdo del panel frontal se encuentra el selector de los términos linguisticos de la variable linguistica que se este tratando, junto con la convención de colores utilizada dentro de la gráfica.

Figura 38. Panel Frontal del Editor de Conjuntos Fuzzy

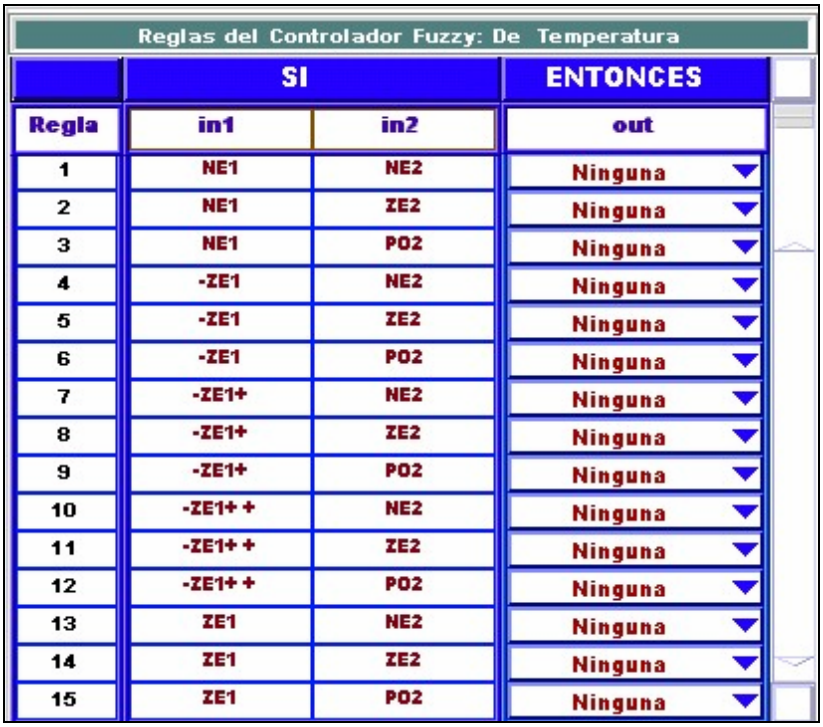


Al hacer *clic* en la etiqueta de cada término lingüístico se activan los controles de entrada numérica y deslizable de la parte inferior, con los cuales es posible modificar de manera interactiva las funciones de membresía de los conjuntos *fuzzy*. Las modificaciones de las figuras que representan a los términos linguisticos están sujetas a las restricciones propias de las funciones de membresía utilizadas para definirlos.

Otras opciones de modificación de una variable linguistica y de sus términos son asequibles desde el menú principal del programa y para tal fin se debe seleccionar el objeto a modificar haciendo *clic* en su etiqueta. Estas opciones para una variable linguistica son: Nombrar (o cambiar de nombre a) una variable, editar sus rangos (el universo de discurso), adicionar y remover variables. De manera similar, para los términos linguisticos las opciones son: Nombrar términos, adicionar términos antes o después y remover términos.

3.1.3.2. Editor de Base de Reglas.VI

Figura 39. Panel Frontal del Editor de Base de Reglas



La base de reglas se fundamenta principalmente en el conocimiento de un experto acerca del control del proceso, existen otras técnicas para establecer dichas reglas, pero este tópico no esta dentro de los alcances de este proyecto.

El Editor de Base Reglas es básicamente una tabla en la cual se resumen las reglas de la siguiente forma:

Tabla 4. Formato de la Base de Reglas

Número de Regla	**SI**		**ENTONCES**
	Variable de Entrada 1	**Variable de Entrada 2**	**Salida**

El significado explícito de este formato para una regla es el siguiente:

SI la variable de entrada 1 es el término A **Y** la variable de entrada 2 es el término B **ENTONCES** la variable de salida es S.

Este editor inicia con un conjunto de reglas por defecto o definidas a partir de los términos de las variables de entrada, estas reglas se particularizan seleccionando en la columna **ENTONCES** una de las etiquetas concernientes a un *Singleton* de la variable de salida mediante un selector que se encuentra en cada fila donde se define una regla.

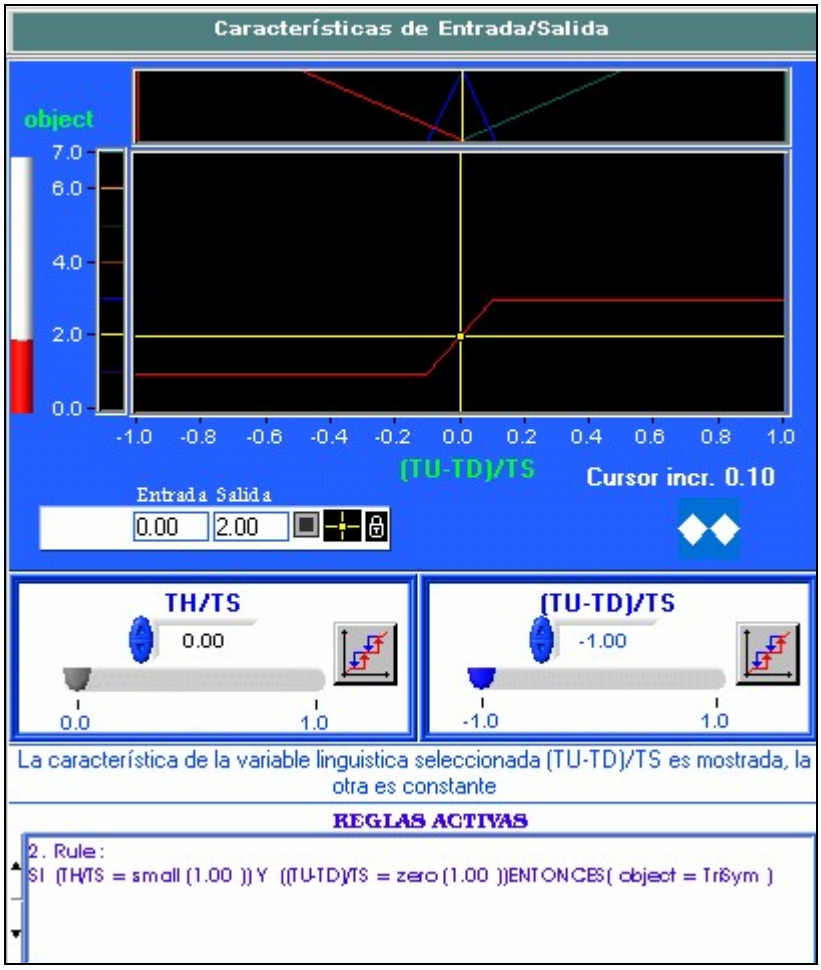
Existen dos (2) opciones para definir la regla que debe tomar el editor cuando quede sin especificar en una regla su consecuente (singleton de la variable de salida) o en otras palabras que quede inactiva, estas opciones son: **Ultimo valor** y **Valor por defecto**. Y pueden en el menú escoger en el ítem **Herramientas<<Reglas sin activar**.

3.1.3.3. Visor de Características de E/S.VI

Esta ventana permite observar las características de Entrada/Salida del controlador *Fuzzy* que se este diseñando, para esto el programa coloca una de las variables de entrada como un valor constante; la otra variable se varía en todo su universo de discurso. Se gráfica la característica tomando en el eje x la variable de entrada y en el eje y la variable de salida.

Existen dos (2) recuadros en la parte inferior de la gráfica de la característica de Entrada / Salida; cada uno de los cuales contiene un icono que se activa para graficar en la característica de E/S la variable □escripción de entrada que representa su etiqueta, contra la variable □escripción de salida. Además, contiene un control numérico y un control deslizable para asignar un valor constante a la variable □escripción de entrada que no sea graficada en la característica de E/S.

Figura 40. Panel Frontal del Visor de Características de E/S



En la parte inferior de la gráfica se encuentra el cursor que permite desplazarse a través de la curva, y unos indicadores de la entrada y salida en cada punto seleccionado. Cada punto dentro de la curva de E/S es el resultado que se ejerce a través de una o más reglas de control *Fuzzy* definidas en la Base de reglas. Las reglas que son activas para cada punto (x;y) seleccionado en la curva pueden observarse en el recuadro inferior de la ventana.

3.2. ESTRUCTURA DE PROGRAMACIÓN DEL GENERADOR DE CÓDIGO

Dentro del programa **FUZZYE3T** se encuentran los programas denominados **FUZZYASM_P** y **FUZZYASM_M**, los cuales convierten la caracterización del controlador *Fuzzy* (hecha mediante las herramientas de *Fuzzy Logic Toolkit*) a un formato de Descripcripcion del control *Fuzzy* en lenguaje ensamblador para los microcontroladores PIC y MOTOROLA respectivamente. En general, cada uno de estos programas puede dividirse en cuatro (4) partes para su descripción; estas son:

a)Módulo de Conversión de Antecedentes.

b) Módulo de Reglas.

c) Módulo de Centroide.

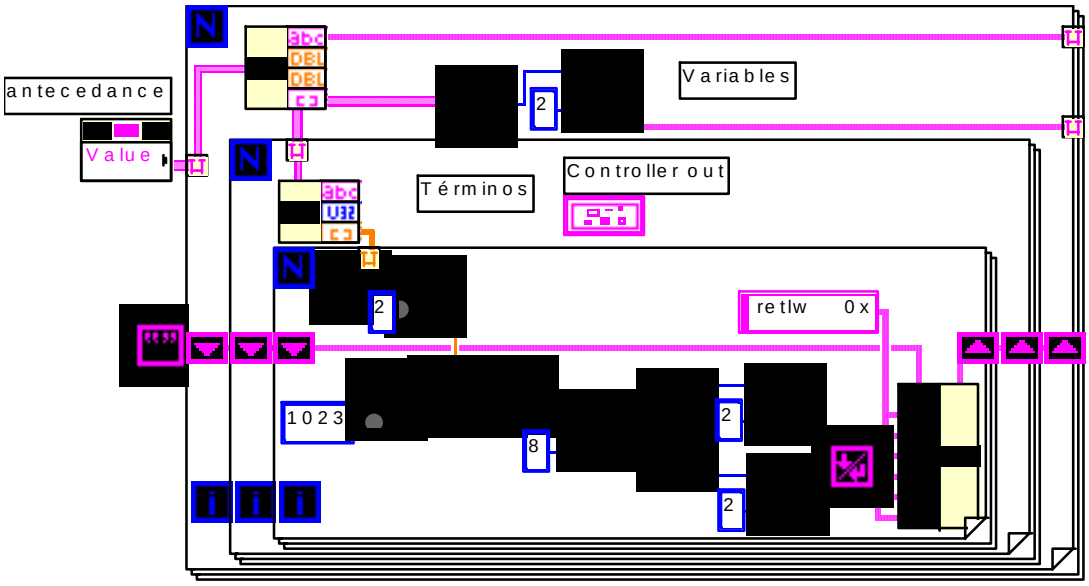
d) Acople.

3.2.1. Módulo de Conversión de Antecendentes

La programación gráfica de este módulo consta principalmente de tres estructuras de repetición anidadas del tipo *FOR*..

Como resultado de esta programación se obtienen dos (2) vectores de cadenas de caracteres y una cadena de caracteres adicional. El primer vector contiene las etiquetas (nombres) de las variables linguisticas de entrada definidas por el usuario. El segundo vector contiene los número de términos linguisticos definidos para cada una de las variables linguisticas de entrada. La cadena de caracteres adicional contiene el código en lenguaje ensamblador que sirve como tabla, que almacena las abscisas que definen las funciones de membresía de cada término linguistico (4 bytes para cada término). Esta programación puede verse en la siguiente figura:

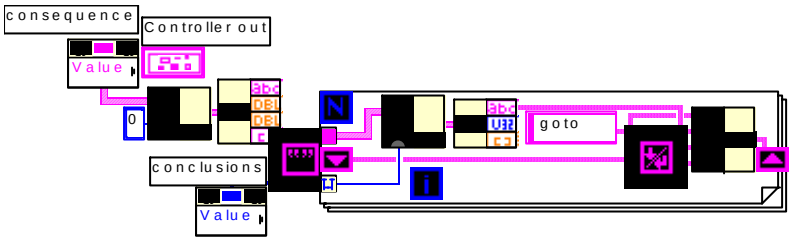
Figura 41. Programación gráfica del módulo de conversión de antecedentes.



3.2.2. Módulo de Reglas

Este módulo consta principalemte de una estructura del tipo *FOR*. El resultado de este módulo de programación es una cadena de caracteres que contiene el segmento de código en lenguaje ensamblador para direccionar las reglas de control *Fuzzy*.

Figura 42. Programación gráfica del módulo de reglas.

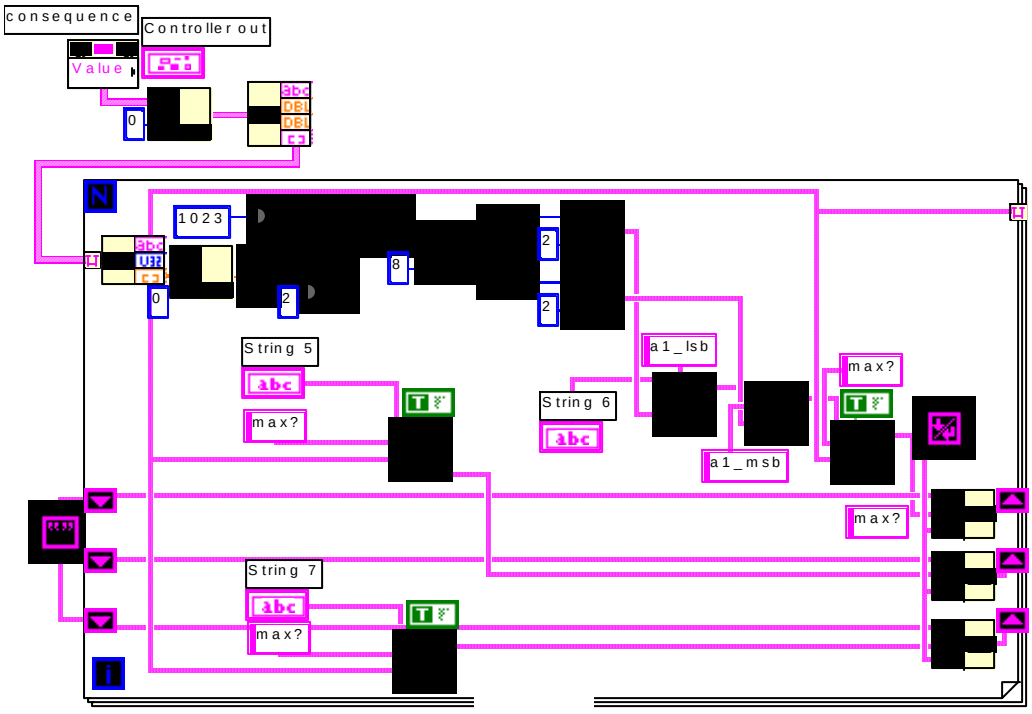


En esencia, en este segmento se utilizan estructuras de salto condicional para acceder a una regla *fuzzy* determinada dentro del formato de programa de control *fuzzy*.

3.2.3. Módulo de Centroide

Este módulo utiliza un lazo *FOR* para obtener como resultado tres cadenas de caracteres. La primera cadena representa el código en lenguaje ensamblador para ejecutar las operaciones de multiplicación, suma y división que involucra el cálculo del centroide de los *singletones* de la variable linguistica de salida. La segunda cadena, contiene el código de programa que sirve para ejecutar la implicación mínimo – máximo y almacenar el resultado en la variable (de programa) que representa el término de salida, según una regla determinada. La tercer cadena de caracteres relaciona los nombres de las variables de programa que representan todos los términos linguisticos de la variable linguistica de salida.

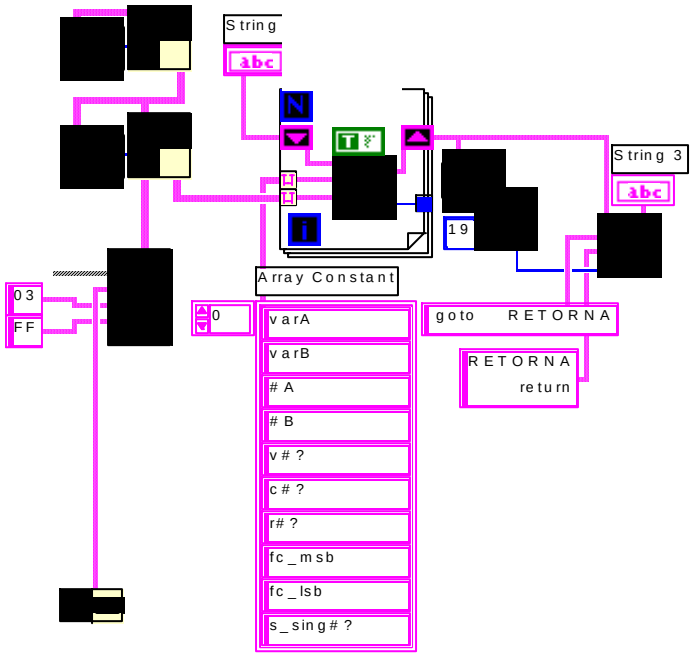
Figura 43. Programación gráfica del módulo de Centroide.



3.2.4. Acople

Esta programación es la encargada de tomar los resultados de los módulos anteriormente descritos y sintetizar una cadena de caracteres que representa el código del programa del control *Fuzzy* en lenguaje ensamblador. Esto se hace, incrustando los segmentos de código en el formato de programa de control *Fuzzy* (que se expuso en el capítulo 3). El formato es particularizado según la caracterización presente en el archivo *.fc que se formo en la interfaz gráfica de diseño de controladores *Fuzzy* y presentado finalmente como una subrutina, para esto último se cierra el código con las instrucciones de retorno de subrutinas.

Figura 44. Programación gráfica que acopla los módulos del Programa FUZZYASM.



4. APLICACIÓN: CONTROL FUZZY DE LA TEMPERATURA DE UNA RESISTENCIA DE CALEFACCIÓN

El control de temperatura es ampliamente utilizado en varios procesos industriales. A continuación se desarrolla un Controlador Lógico *Fuzzy* capaz de regular la energía entregada a un elemento resistivo para mantener su temperatura en rangos preestablecidos.

Primero, se muestra el control *Fuzzy* realizado por medio del computador y luego, el mismo control hecho mediante un microcontrolador de ocho bits de propósito general; en este caso se hace la prueba para cada uno de los microcontroladores PIC16F873 y MC68HC908JL3.

4.1. CARACTERÍSTICAS DEL SISTEMA

Este sistema físicamente consta de un bombillo incandescente de 120 Voltios – 60 Vatios como elemento resistivo objeto del control de temperatura, conectado en serie con una fuente de voltaje de 115 Vrms a través de un tiristor. El lazo de realimentación lo conforma básicamente un sensor de temperatura. El lazo directo se cierra por medio del controlador propiamente dicho, este puede ser implementado en el computador o en el microcontrolador.

El *setpoint* de temperatura puede introducirse al sistema a través de un potenciómetro (señal analógica) o por medio de un teclado alfanumérico (señal digital). La temperatura medida en la superficie de la ampolla de vidrio del bombillo es convertida a una señal eléctrica por medio del sensor (transductor) de estado sólido C.I. LM35.

La salida del controlador *Fuzzy* es una señal de voltaje modulada en ancho de pulso, que luego de pasar por una etapa de acondicionamiento de señal sirve para dar el disparo al tiristor actuador (TRIAC). Esto hace que el ángulo de conducción del triristor sea variable y se produzca una señal de voltaje sinusoidal recortada en fase en los terminales del elemento resistivo. Se utiliza el control A.C. de línea para evitar la componente de D.C. presente en el control unidireccional, y por tanto mejorar la eficiencia del sistema de calentamiento de la resistencia (bombillo incandescente). Esto es posible aprovechando la característica de funcionamiento del tiristor (TRIAC) para regular la tensión eficaz aplicada a la carga resistiva.

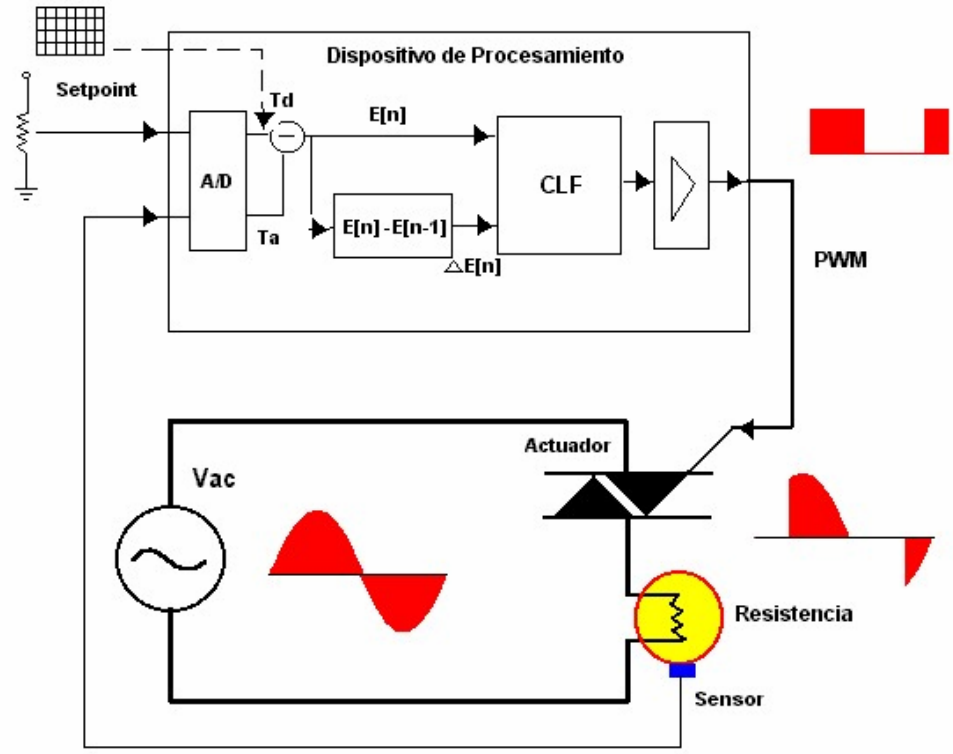
Finalmente, si las reglas del controlador *Fuzzy* son adecuadas, la energía entregada a la resistencia eléctrica se regula de tal forma que la temperatura alcanza el *setpoint* establecido y se mantiene aproximadamente constante, aún en presencia de cambios externos. El esquema general del sistema puede verse en la siguiente figura:

En el sistema implementado, el sensor de temperatura determina la entrada de datos al controlador. Esta entrada es procesada para obtener las dos variables linguisticas de entrada (o antecedentes) del controlador, error (***E***) y variación del error (**Δ*E*)**, de la siguiente forma:

$$E = Td - Ta \qquad \text{(Ec. 15)}$$

$$\Delta E = E_2 - E_1 \qquad \text{(Ec. 16)}$$

Figura 45. Esquema general del sistema.



Donde ***Td*** es la Temperatura deseada, ***Ta*** es la temperatura actual leída por el sensor, ***E2*** es el error anterior y ***E1*** es el error actual del sistema de control.

Considerando constante el *setpoint* de un muestreo de señal a otro, se tiene que:

$$Td2 = Td1 \qquad \text{(Ec. 17)}$$

Entonces,

$$\Delta E = Ta1 - Ta2 \qquad \text{(Ec. 18)}$$

Donde ***Td2*** es la temperatura deseada anterior, ***Td1*** es la temperatura deseada actual, ***Ta1*** es la temperatura leída actual y ***Ta2*** es la temperatura leída anterior.

La variable de salida (o consecuencia) es determinada según el método de funcionamiento del actuador, en este caso corresponde a una señal de PWM que se suministra a un dispositivo optoacoplador para dar el disparo a un TRIAC.

Las variables lingüísticas seleccionadas se estructuran con un total de ocho términos lingüísticos. Cinco términos lingüísticos la primer variable (Error Negativo, Error Cero, Error Pequeño, Error Medio y Error Grande) y tres la segunda (Variación del Error Cero, Variación del Error Medio y Variación del Error Grande), tomados éstos del lenguaje natural utilizado para la descripción del estado de la temperatura del elemento de calentamiento.

Las funciones de membresía utilizadas para cada término lingüístico dentro de cada variable se esbozan en las siguientes figuras:

Figura 46. Funciones de membresía utilizadas en los términos lingüísticos de la variable Error.

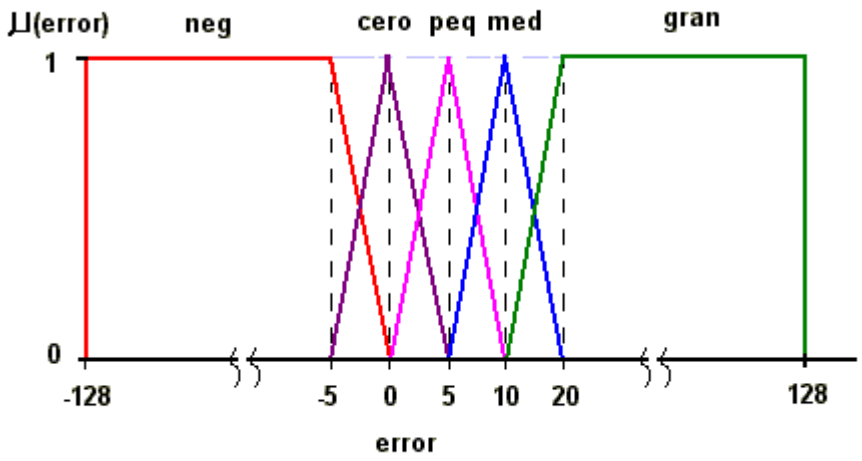
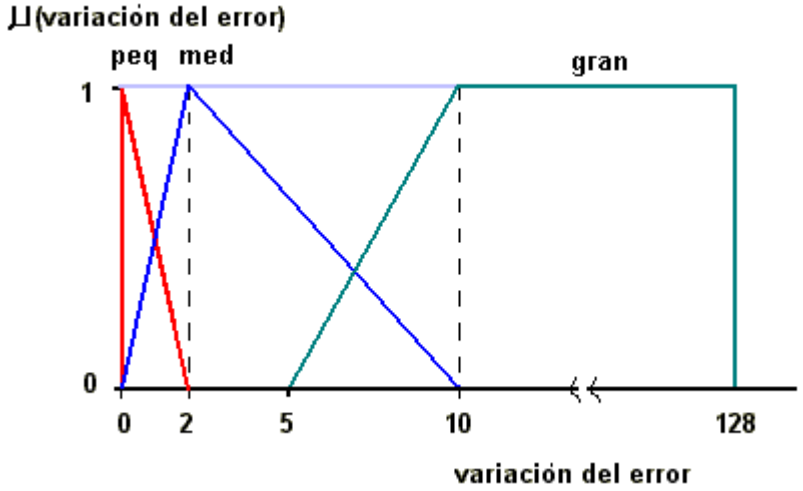


Figura 47. Funciones de membresía utilizadas en los términos lingüísticos de la variable Variación del Error



El ajuste de estas funciones es un factor determinante para el buen funcionamiento del controlador, y depende principalmente de pruebas empíricas.

La variable lingüística de salida o consecuencia se conforma de seis (6) términos lingüísticos definidos como *singletones*, estos son: Potencia Máxima, Potencia Alta, Potencia Media Alta, Potencia Media, Potencia Media Baja y Potencia Cero. Ver Figura, 4.11. El ajuste de los *singletones* (para los términos lingüísticos) en el universo de discurso de la variable de salida actual, se hace con base en la distribución de potencia al elemento resistivo por parte de la fuente de tensión, esto es:

Si ***Vs*** =√2 Vs* sen (wt) es el voltaje de entrada, y los ángulos de retraso del TRIAC en ambos sentidos son iguales (α1= α2 = α) el voltaje RMS de salida de determina como sigue:

$$V_o = \left[\frac{2}{2\pi} \int_{\alpha}^{\pi} 2V_S^2 \operatorname{sen}^2(wt) d(wt) \right]^{1/2} \qquad \text{(Ec. 19)}$$

$$V_o = \left[\frac{4Vs^2}{4\pi} \int_{\alpha}^{\pi} (1 - \cos(2wt)) d(wt) \right]^{1/2} \qquad \text{(Ec. 20)}$$

$$V_o = Vs \left[\frac{1}{\pi} \left(\pi - \alpha + \frac{\operatorname{sen} 2\alpha}{2} \right) \right]^{1/2} \qquad \text{(Ec. 21)}$$

La potencia en la carga seria de la forma:

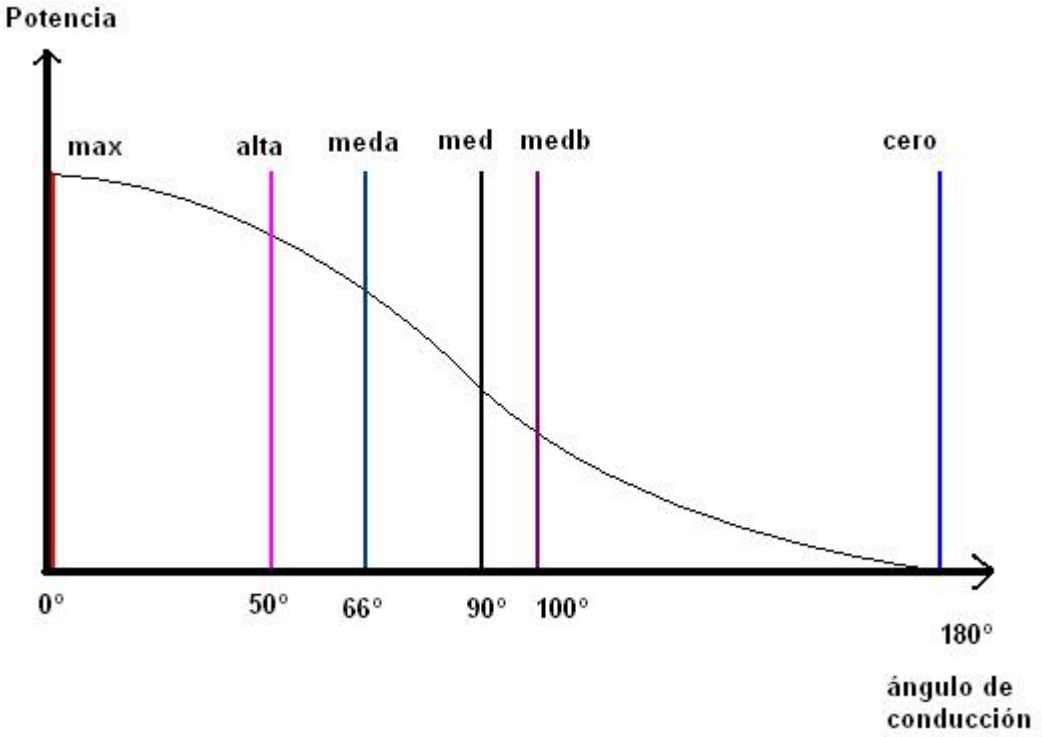
$$P_o = Vo^2 * R \qquad \text{(Ec. 22)}$$

$$P_o = \frac{\frac{Vs^2}{\pi} \left(\pi - \alpha + \frac{sen2\alpha}{2} \right)}{R} = \frac{Vs^2 [2(\pi - \alpha) + sen2\alpha]}{2\pi * R} \qquad \text{(Ec. 23)}$$

$$P_o = \frac{Vs^2 [2(\pi - \alpha) + \operatorname{sen} 2\alpha]}{2\pi * R} \qquad \text{(Ec. 24)}$$

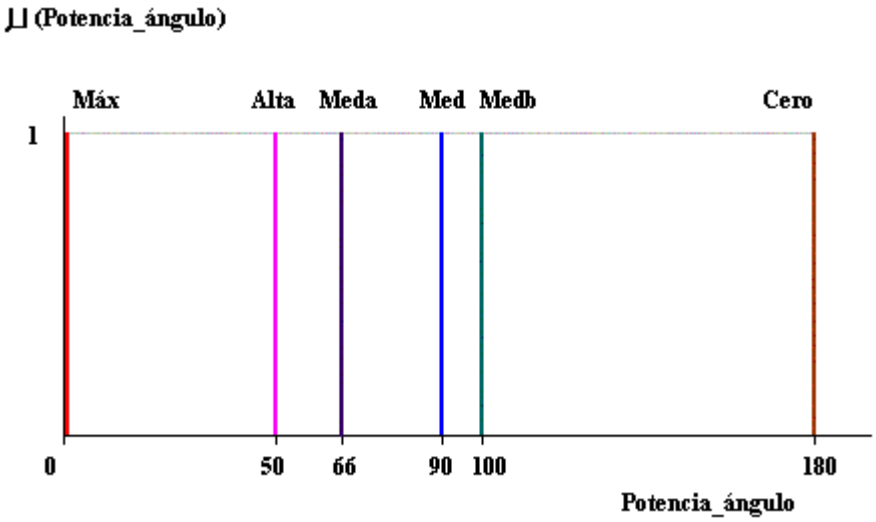
La curva que representa dicha potencia en función del ángulo de conducción, junto con la ubicación de los singletones es mostrada en la figura, 48.

Figura 48. Ajuste de los términos lingüísticos de la variable de salida sobre su universo de discurso.



El hecho de establecer las funciones de membresía de la variable de salida del tipo *singleton*, nos permite aplicar el método de defuzzyficación por centroide de forma simplificada. Para el controlador difuso se tiene, entonces, la variable de salida toma la siguiente forma (ver figura, 49):

Figura, 49. Términos lingüísticos de la variable de salida Potencia_ángulo.



El número de reglas que se utilizan en el control *Fuzzy* de este sistema esta dado por:

(Ec. 25)

(Ec. 26)

Donde Pi es el número de términos para la variable de entrada i, y m es el número de la variable de entrada.

Estas quince (15) reglas se forman a partir del ajuste de una consecuencia para cada posible combinación de los términos de las variables antecedentes del controlador difuso. La experiencia e intuición de un operador en el control de un sistema de calentamiento sencillo como este puede ser esbozada de forma resumida por medio de los siguientes enunciados:

1. Si el error de temperatura (diferencia entre la temperatura deseada y la temperatura leída) es negativo; es decir que la temperatura del elemento de calentamiento es superior a la temperatura deseada, no queda otra opción más que quitar la alimentación al sistema (pues no se cuenta con sistemas de enfriamiento).

2. Si el error de temperatura es positivo grande; es decir que el sistema esta lejos de alcanzar la temperatura deseada, luego la solución debe ser proporcionar la máxima potencia a la carga para que se produzca el calentamiento lo más rápido posible.

3. Si el proceso de alcanzar una temperatura deseada (*setpoint*) avanza muy lentamente (variación del error pequeño) y no se esta entregando la máxima potencia, entonces se debe aumentar dicha potencia.

4. Si la temperatura leída por el sensor se aproxima rápidamente a la temperatura deseada (cuando, temperatura deseada > temperatura leída), entonces puede ser ventajoso disminuir la gradualmente la potencia entregada a la carga resistiva para evitar sobrepasar el *setpoint*.

El conjunto completo de reglas puede resumirse en la siguiente tabla:

Tabla 5. Reglas utilizadas para la caracterización del control Fuzzy de temperatura

Regla Nº.	**SI**		**ENTONCES**
	ERROR	**VARIACIÓN DEL ERROR**	**POTENCIA**
1	negativo	Pequeño	Cero
2	negativo	Medio	Cero
3	negativo	Grande	Cero
4	Cero	Pequeño	Media baja
5	Cero	Medio	Media baja

6	Cero	Grande	Media baja
7	Pequeño	Pequeño	Alta
8	Pequeño	Medio	Media alta
9	Pequeño	grande	Media
10	Medio	Pequeño	Máxima
11	Medio	Medio	Alta
12	Medio	grande	Media alto
13	Grande	Pequeño	Máxima
14	Grande	Medio	Máxima
15	Grande	grande	Máxima

En aplicaciones de control *Fuzzy* de lazo cerrado, como el de este ejemplo, son de uso común los operadores de AND (Mínimo) y OR (Máximo), dando como mecanismo de inferencia estándar el método de Mínimo-Máximo que se adopta en este proyecto.

El método de defuzzyficación de Centroides da como resultado de manera directa un valor preciso para la variable real de salida de control, una señal de tipo pulso modulado en amplitud (para este ejemplo).

La documentación completa de la caracterización del controlador es mostrada el anexo 3.

4.2. CONTROLADOR FUZZY DESDE EL COMPUTADOR

El control de sistemas físicos por computador se hace cada vez más común en la industria debido al incremento en la flexibilidad de los programas de control y la capacidad lógica de los sistemas digitales. El esquema general del sistema basado en computador es el mismo mostrado en la figura 45, donde el dispositivo de procesamiento hace alusión al PC.

Los principales requerimientos para el desarrollo de esta aplicación son:

- Hardware de adquisición de datos.

- Hardware de control.

- Dispositivo de procesamiento central, en este caso un computador personal para procesar los datos adquiridos y la generan la salida correspondiente para el control con Lógica *Fuzzy* del sistema.

- Software, para la aplicación de control. Para esto se desarrolló un programa en LabVIEW, basándose en la herramienta *Fuzzy Logic for G Toolkit.*

Con estos requerimientos básicos, el control *Fuzzy* es aplicable a casi todo sistema susceptible de ser controlado; basta con tener alguna experiencia en el funcionamiento del sistema a controlar (o utilizar alguna otra técnica de extracción de conocimiento) para que mediante un lenguaje natural se generen las reglas que determinarán el comportamiento del controlador *Fuzzy*.

Como hardware de adquisición de datos y acondicionamiento de la señal de salida se utiliza una tarjeta electrónica externa. Los detalles de la tarjeta pueden verse en el capítulo 5 de esta tesis de grado.

A continuación se describe el programa principal para esta aplicación, ejecutada desde el computador.

4.3. PROGRAMA PRINCIPAL

El programa de aplicación para el control *Fuzzy* desde el computador reúne las siguientes características:

- Interfaz gráfica intuitiva del proceso.

- Visualización de las señales de entrada y de las acciones de control generadas mediante gráficos en pantalla.

- Registro continuo de las señales de entrada y salida en disco, para tener un seguimiento histórico del comportamiento del sistema, mediante consultas de base de datos, y elaboración de gráficos directamente de los datos obtenidos.

- Permite seleccionar un archivo de extensión *.fc con la caracterización del controlador *Fuzzy*, esto permite probar diferentes caracterizaciones para escoger la más apropiada.

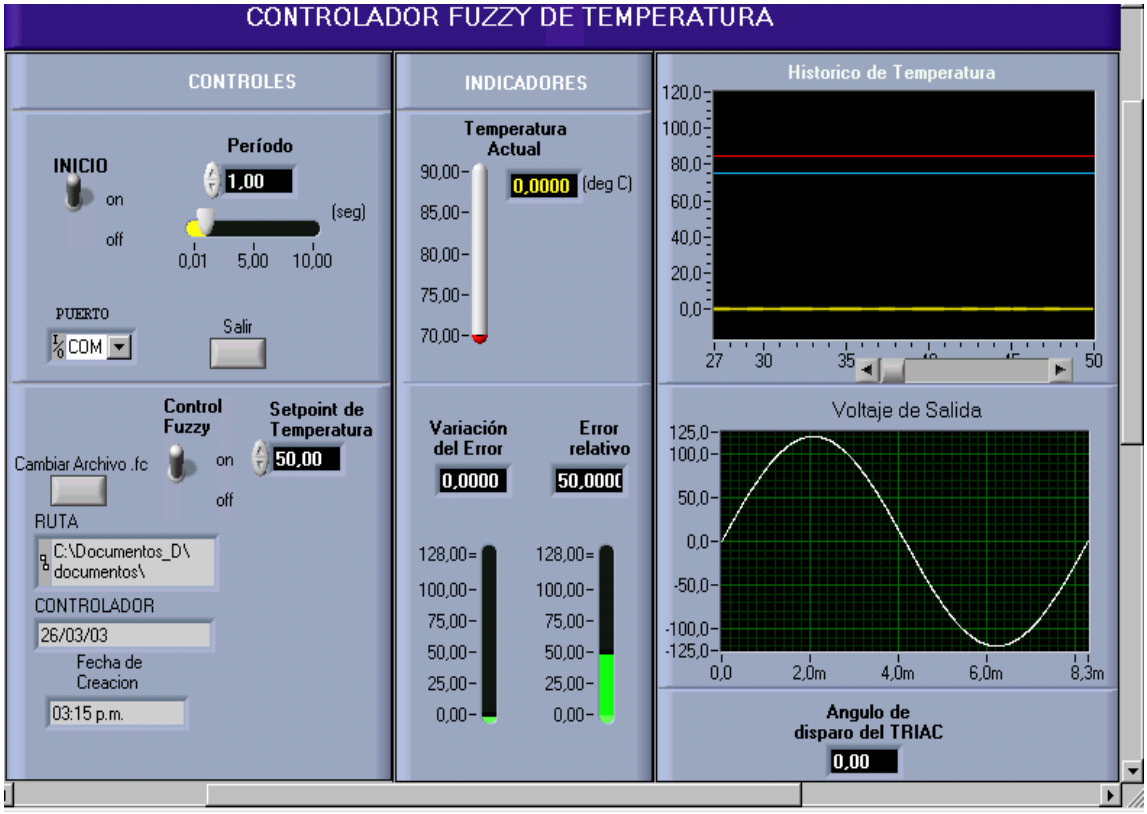
4.1.1.1. Interfase con el Usuario

El panel frontal del Controlador con Lógica *Fuzzy* basado en el computador se muestra en la figura 50.

En este Panel Frontal se puede observar el progreso de la variable controlada temperatura, mediante gráficas e indicadores numéricos; así como también se ve un análisis estadístico de dicha variable. Mediante los controles del campo de Rango de Temperatura es posible fijar los límites alto y bajo en la gráfica de temperatura, de manera tal que se produzca una alarma visual si se pasa uno de estos límites. El inicio de conversión de datos A/D y el del análisis de los datos adquiridos se dan en el campo denominado Controles del Sistema. Un indicador muestra el ángulo de disparo del

TRIAC y en una gráfica adicional se visualiza el voltaje de salida que se envía a la carga.

Figura 50. Interfase de usuario del programa de control de temperatura.



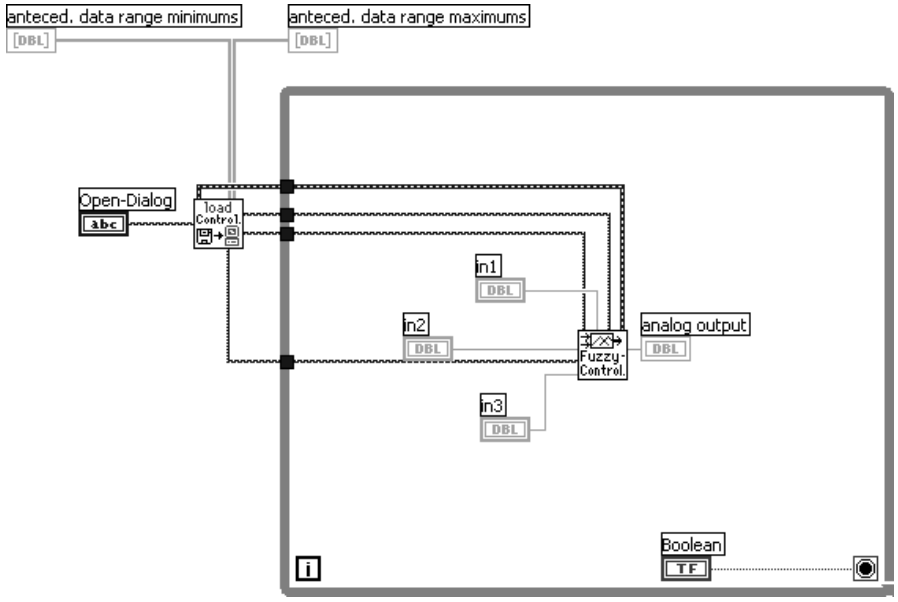
Al inicio del programa se abre una ventana de abrir, con el cual se selecciona el archivo de caracterización *. fc que se desea utilizar para el control del sistema.

4.1.1.2.Subrutina de Control

En esta subrutina se carga el archivo de control *Fuzzy* mediante el *Load Fuzzy Controller VI* y se procede a ejecutar el control con el *Fuzzy Controller VI*. Para esto, se toma el *setpoint* de temperatura y la temperatura leída por la tarjeta de adquisición de datos, y se procesa esta última para obtener las dos (2) variables linguisticas entradas del

controlador. La salida de esta subrutina es el ángulo de disparo para el actuador (TRIAC) que es enviado a la tarjeta externa de control via puerto serial.

Figura 51. Subrutina de control.



4.4. CONTROLADOR FUZZY BASADO EN MICROCONTROLADOR

El formato de software de control *Fuzzy* desarrollado en esta tesis, esta dirigido a aplicaciones con microcontroladores de la familia PIC (gamas básica y media) de MICROCHIP y de la familia CPU08 de MOTOROLA. Para la aplicación particular de control *Fuzzy* de temperatura fueron escogidos dos (2) microcontroladores de 8 bits de bajo costo:

- PIC16F873 de la empresa *MICROCHIP*
- HC908JL3 de la empresa *MOTOROLA.*

Los pasos que se deben seguir para implementar un controlador *Fuzzy* en un sistema basado en microcontrolador son los siguientes:

1. Caracterizar el sistema mediante una herramienta de diseño tal como FUZZYE3T o Fuzzy Logic Toolkit.

2. Afinar los parámetros del controlador, esto puede hacerse en el programa FUZZYE3T meditante la observación de las características de E/S o por medio de simulaciones en otros programas como MATLAB.

3. Generar el código en lenguaje ensamblador del programa del controlador *Fuzzy* por medio de un software como FUZZYE3T, o por cuenta propia del diseñador.

4. Ensamblar el código de control *Fuzzy* junto con las demás rutinas necesarias para la aplicación en particular.

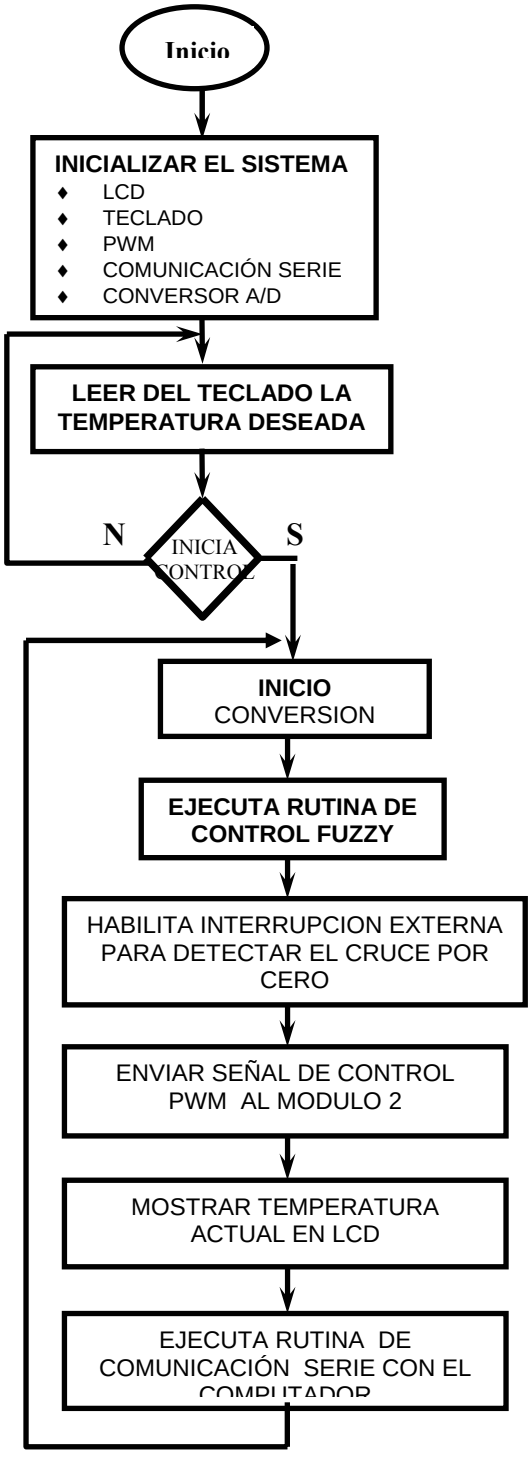
5. Grabar el código de programa del controlador en el microcontrolador. Generalmente, los fabricantes de microcontroladores proveen su propio software para tal fin, por ejemplo: MPLAB para los microcontroladores de MICROCHIP y el ICS08JLZ para los de MOTOROLA.

Para la aplicación de control de temperatura de una resistencia de calefacción se complementó el código de control *fuzzy* (generado por FUZZYE3T) con rutinas de adquisición y visualización de datos, de generación de señales moduladas en ancho de pulso y manejo de interrupciones, entre otras. Esto se verá a continuación.

4.1.1.3. Diagrama de Flujo del Control de Temperatura

El diagrama de flujo del programa de control *fuzzy* del proceso de calentamiento de una resistencia de calefacción puede verse en la siguiente figura.

Figura 52. Diagrama de flujo programa de control *fuzzy* solo en el microcontrolador



4.1.1.4. Programa de Control *Fuzzy* en Código Ensamblador para los Microcontroladores PIC16F873 y HC08JL3

Los códigos de programa para la ejecución del control *fuzzy* de temperatura de una resistencia de calefacción son mostrados en el anexo 4.

5. DISEÑO DEL HARDWARE PARA EL CONTROLADOR FUZZY

En este capítulo se describe el hardware construido para el control Fuzzy en particular se presenta las características del mismo para implementar el control de temperatura de un sistema realimentado simple.

Para la implementación de este proyecto se construyo un sistema electrónico modular, en primera instancia dos módulos; Figura 53.

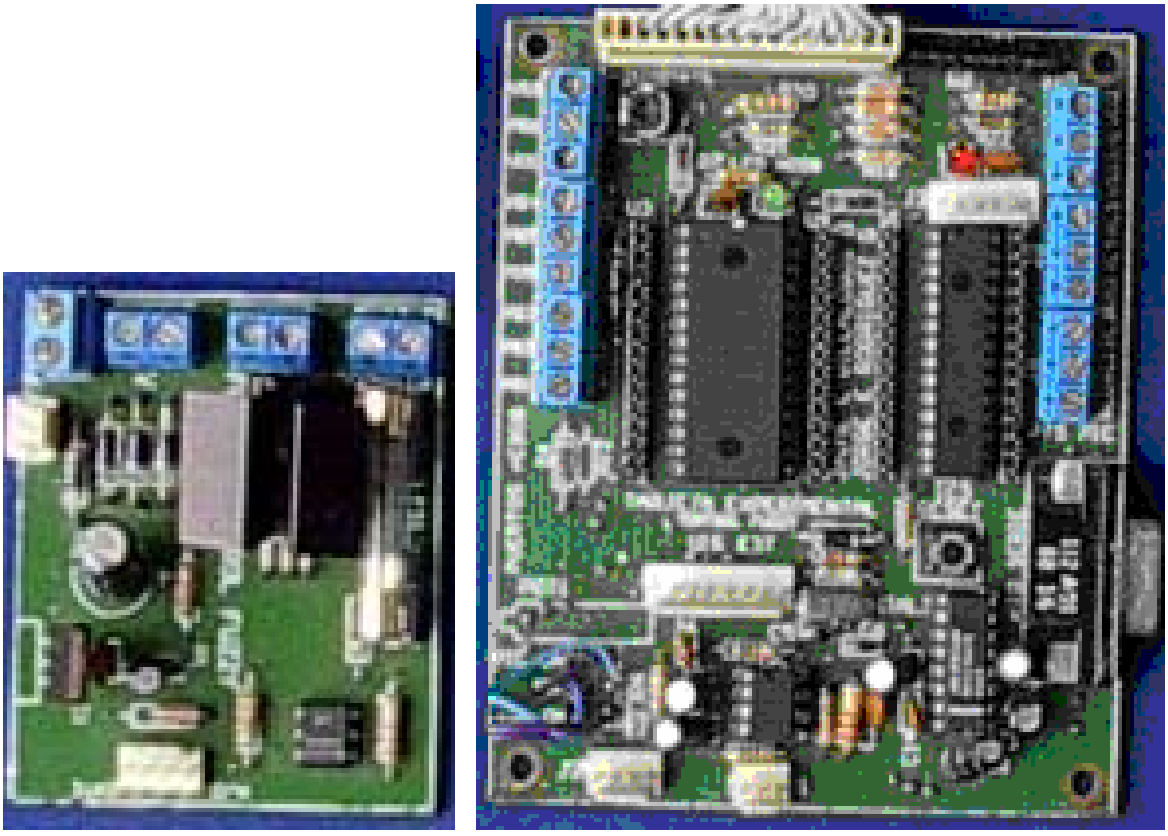


Figura 53, Vista superior de los módulos construidos (diseñados en demo Protel 99 SE)

El primer módulo es la etapa analógica, este agrupa la fuente, el detector de cruce por cero y el relé de estado sólido para controlar la carga. El segundo módulo es la *tarjeta experimental para el control Fuzzy*, que agrupa dispositivos digitales y analógicos.

Este sistema se puede utilizar para aplicaciones de control de procesos, soportando estrategias de control lógico Fuzzy, en una aplicación específica de control de temperatura de un sistema, control realizado desde el computador o exclusivamente desde alguno de los dos microcontroladores.

5.1. DESCRIPCIÓN

En la figura 54 se muestra el diagrama de bloques del sistema, cada uno de ellos esta formado por un grupo específico de componentes agrupados en los dos módulos.

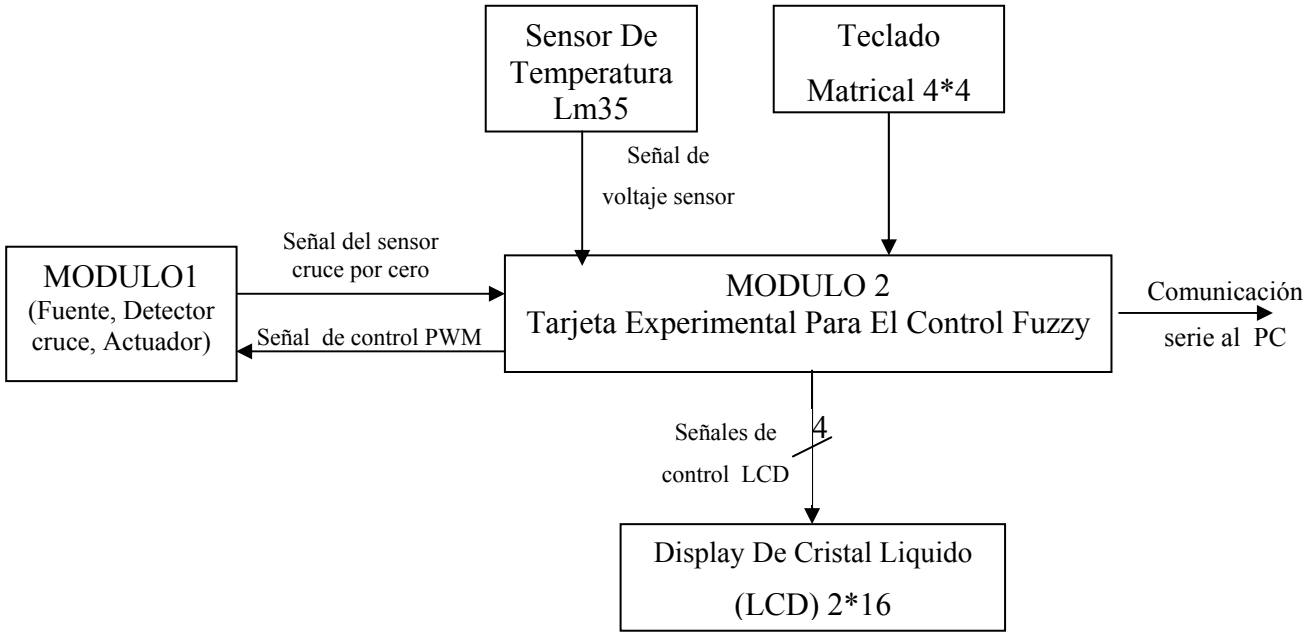


Figura 54, Diagrama de bloques del sistema

5.2. MODULO 1

Este modulo (figura 55) esta compuesto por:

- Fuente de alimentación +5v (cinco voltios positivos) 800mA
- Detector de cruce por cero de onda sinusoidal 60 Hz.
- Relé de estado sólido (control de potencia).

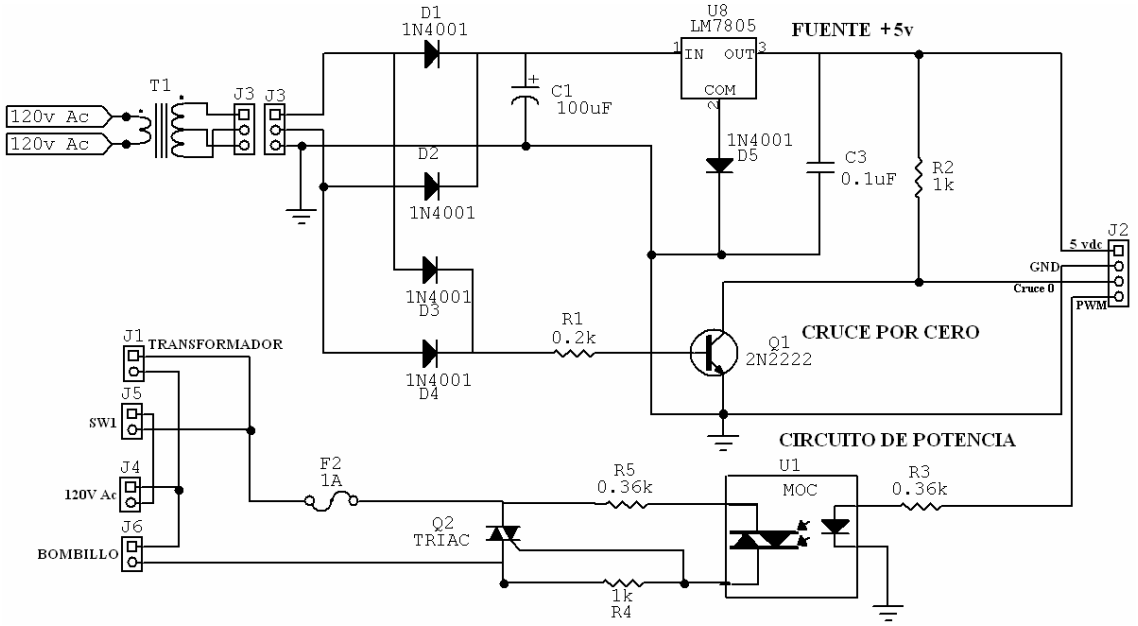


Figura 55, Modulo 1 (Fuente, Actuador y Detector de cruce por cero)

A continuación de describe cada uno componentes de este modulo.

5.2.1.1. Fuente de Alimentación

El sistema se alimenta de la red publica de 110 VAC, posee un transformador con derivación central que reduce dicha tensión a 18VA en los extremos o 9VA en la derivación central, un rectificador de onda completa formado por dos diodos (1 Amperio

de corriente máxima cada uno) y un condensador polarizado permite obtener un voltaje aproximado de 11VDC para alimentar el regulador +5v (7805).

Esta fuente alimenta el transistor del detector de cruce cero y los otros módulos (*Tarjeta Experimental Para El Control Fuzzy*).

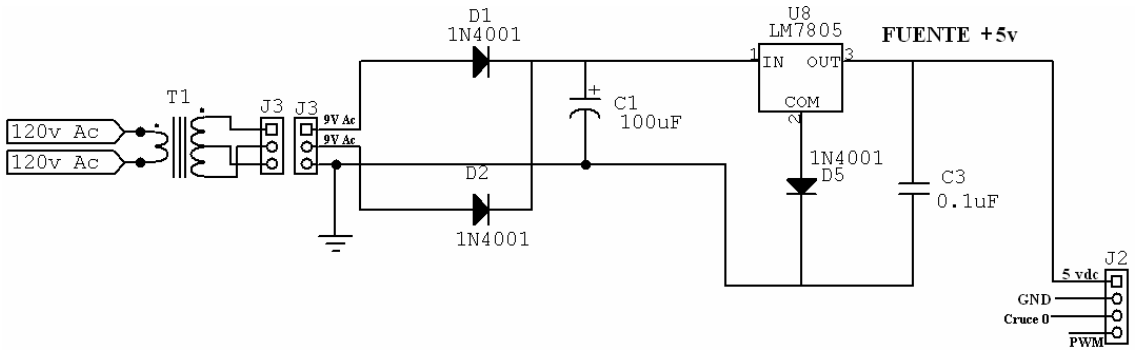


Figura 56, Fuente de Alimentación

5.2.1.2. Detector de Cruce por Cero

Es el circuito encargado de detectar los cruces por cero de una señal sinusoidal esta formado un rectificador de onda completa con dos diodos, conectados a un transistor inversor, se encarga de dar la señal de conducción o no conducción.

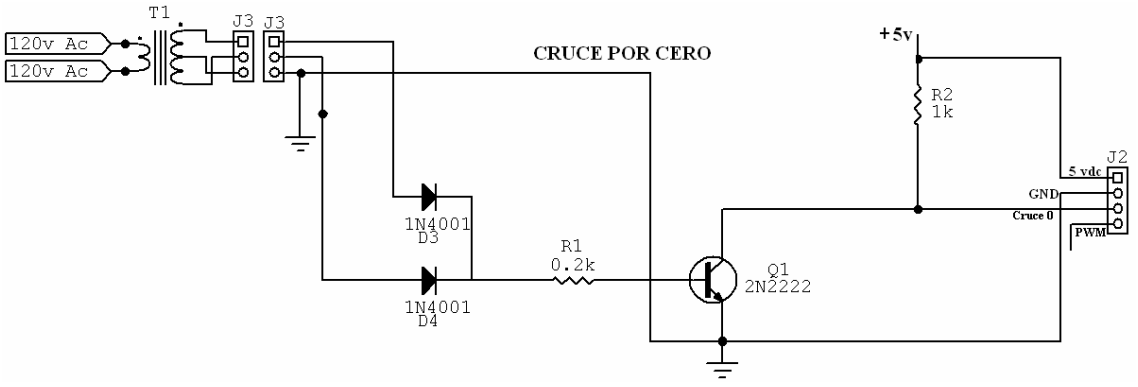


Figura 57, Detector de Cruce por Cero

5.2.1.3. Elemento de Control de Potencia

Es básicamente un interruptor de estado sólido encargado de conmutar cargas de potencia a partir de señales de control de bajo nivel. Estas ultimas provenientes de los microcontroladores y dirigidas a motores, lámparas, solenoides, calefactores, etc. El circuito emplea un optó acoplador (Moc3010) para aislar las señales de control de las de potencia, además emplea un triac (Bta06) como elemento de conmutación para el control de fase de la onda sinusoidal de voltaje, especificado para 115 voltios y 6 amperios de corriente.

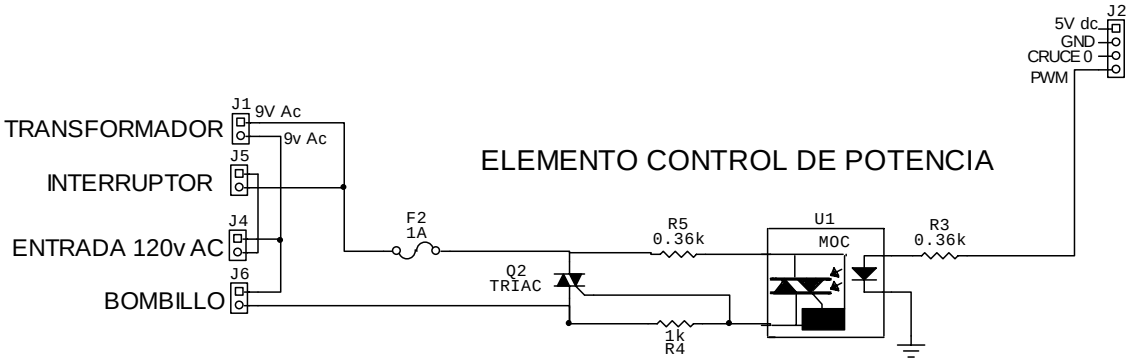


Figura 58, Elemento de Control de Potencia

5.3. MODULO 2

Este modulo esta compuesto por 9 etapas o circuitos, agrupados en la *Tarjeta Experimental Para El Control Fuzzy*.

- Sensor De Temperatura
- Etapa De Acondicionamiento De La Señal Analógica
- Etapa de entrada de datos al sistema (Ajuste De Set Point mediante un teclado)
- Interfaz grafica con el usuario (modulo LCD)
- Etapa Selección Del Microcontrolador
- Etapa Oscilador Del Sistema
- Etapa Reset Del Sistema

- Etapa Microcontrolador PIC 16F873
- Etapa Microcontrolador Motorola MC6HC908JL3
- Etapa De Interfase Para Comunicación Serial Asíncrona Con El PC
- Terminales varios de entrada-salida.

Se presenta a continuación una descripción de cada una de ellas.

5.3.1.1. Sensor de Temperatura

Formado por un sensor de temperatura activo de estado sólido LM35 el cual entrega una respuesta lineal de temperatura con resolución de 10mV/ºC entre 0 y 150ºC Ecuación 4.1, este sensor se conecta con el modulo 2 a través del conector J1.

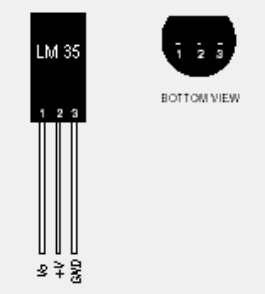


Figura 59, Sensor de Temperatura

Ecuación 27: [Vo = f (T)= 10mV/ºC]; para (0º<T<150ºC)

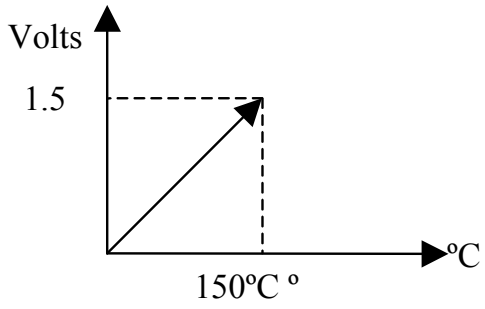


Figura 60. Curva característica del sensor de temperatura

5.3.1.2. Etapa de Acondicionamiento de la Señal Analógica

Encargada de recibir y acoplar adecuadamente la señal proveniente del sensor de temperatura (10mV/ºC entre 0 y 150ºC), amplificando lo suficiente para adecuar esta señal a los niveles de entrada de la siguiente etapa. Esta etapa de acondicionamiento de la señal analógica esta compuesta principalmente por el amplificador operacional LM358 el primero en configuración "buffer" y el segundo como amplificador no inversor de la señal de entrada la ganancia total es:

Ecuación 28: $$G = \frac{R15 + R8}{R8}$$

$$G = \frac{790\Omega + 330\Omega}{330\Omega} = 3.4$$

Cuando se presenta la máxima salida de voltaje del sensor (LM35) que corresponde a 1.5voltios (1.5*3.4 = 5.1V) ocurre el máximo voltaje de salida de esta etapa que es 5.1voltios DC.

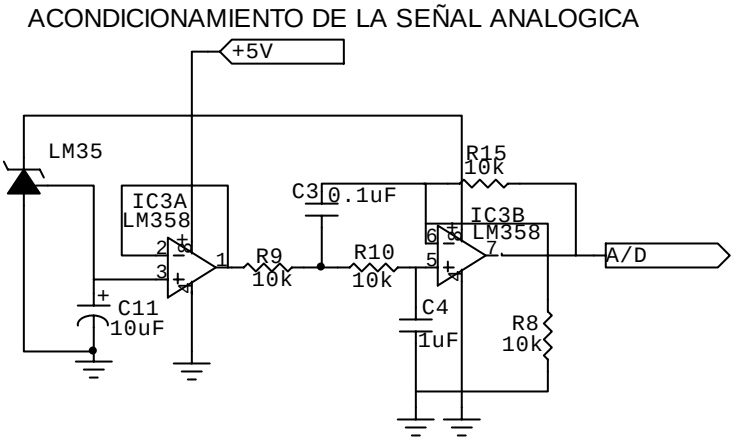


Figura 61, Etapa de Acondicionamiento de la Señal Analógica

5.3.1.3. Etapa De Entrada De Datos Al Sistema

Haciendo uso de un teclado matricial conectado al puerto b del microcontrolador pic16f873 y a los puertos a y d del microcontrolador hc908jl3 y utilizando la interrupción de teclado que estos poseen permite al usuario ajustar el punto de control o set point del controlador fuzzy. Esta etapa esta compuesta por un teclado matricial 4x4 y cuatro resistencias de 2kΩ conectadas a las filas del teclado estas evitan poner en corto circuito a los microcontroladores si accidentalmente se oprime una tecla cuando el modulo lcd esta funcionando. El conector denominado teclado se utiliza para conectar el teclado al modulo 2.

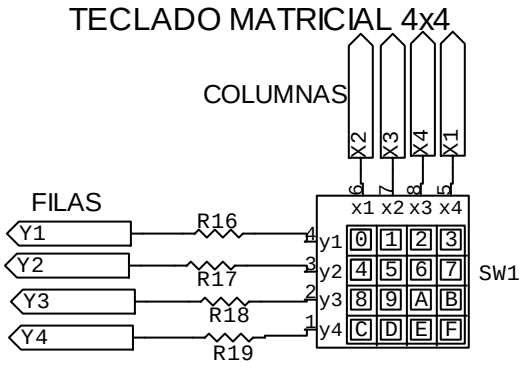


Figura 62, Etapa Ajuste de *Set Point*

Un teclado de este tipo consta de 16 teclas (matriz 4 X 4). Por cada fila y cada columna de la matriz hay un "cable" que pasa por detrás de las teclas, las cuales están colocadas en las intersecciones entre filas y columnas.

Así pues, la columna 1 (COL1) es un "cable" que pasa por debajo del 1, del 4, del 7 y del * ; la fila 1 (FIL1) pasa debajo de la A, 3, 2 y 1.Filas y columnas no están conectadas entre si.

Cuando se pulsa una tecla se conecta la columna y la fila que pasa por debajo de dicha tecla. Por ejemplo, si se pulsa el 1 se conecta la COL1 con FIL1; si se pulsa el 8 se conecta la COL2 con FIL3, y así sucesivamente con el resto de las teclas.

5.3.1.4. Interfaz Gráfica con el Usuario (modulo LCD)

Para visualizar información, parámetros de seto, etc. Se utilizo un *display* de cristal liquido o LCD 2x16 (2 filas por 16 caracteres) la interfase entre el microcontrolador y el *display* de cristal liquido es de un bus de datos de 4 bits, mas dos señales de control (habitación E, selección RS). El pin que indica si se va a escribir o leer (R/W) el modulo LCD siempre esta en cero lógico (escritura). Este modulo comparte el puerto B del microcontrolador pic16f873 y a los puertos A y D del microcontrolador HC908JL3. El conector denominado LCD se utiliza para conectar el *display* de cristal liquido con el modulo 2.

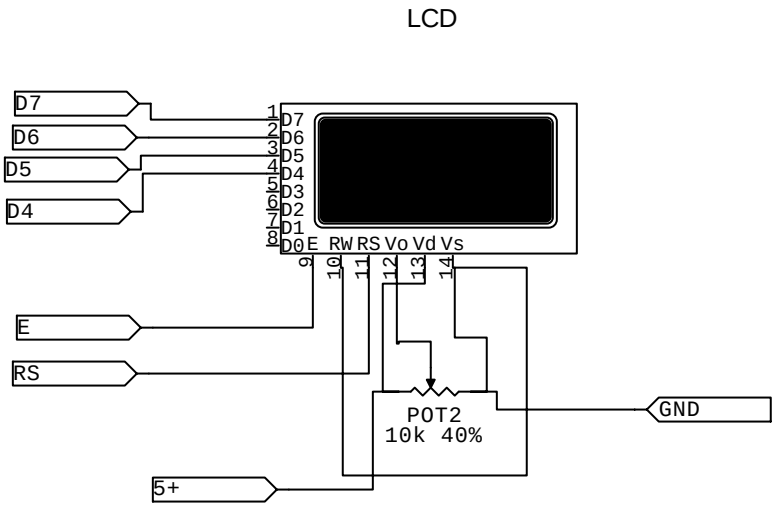


Figura 63, Interfaz Grafica con el Usuario (modulo LCD)

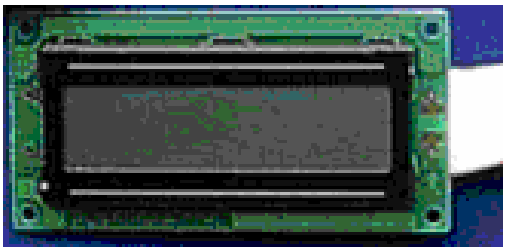


Figura 64,Vista real del modulo LCD

5.3.1.5. Etapa Selección del Microcontrolador

Básicamente es un interruptor de doble polo y permite seleccionar el paso de la señal de alimentación (+5v y tierra) y energizar al microcontrolador PIC 16F873 o el Motorola MC68HC908JL3.

Figura 65. Vista real del Teclado

Figura 66, Etapa Selección del Microcontrolador

5.3.1.6. Etapa Oscilador del Sistema

Encargada de generar la señal de reloj para el funcionamiento de los microcontroladores. Formada por un cristal 4Mhz dos condensadores y una resistencia de carga.

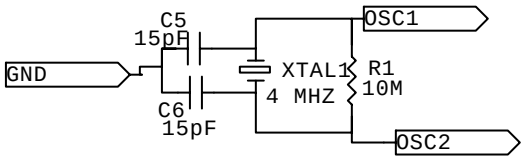


Figura 67, Etapa Oscilador del Sistema

5.3.1.7. Etapa Reset del Sistema

Produce la señal da reset a los microcontroladores, cuando el usuario oprime el pulsador. Esta formada por un pulsador, un diodo y varias resistencias.

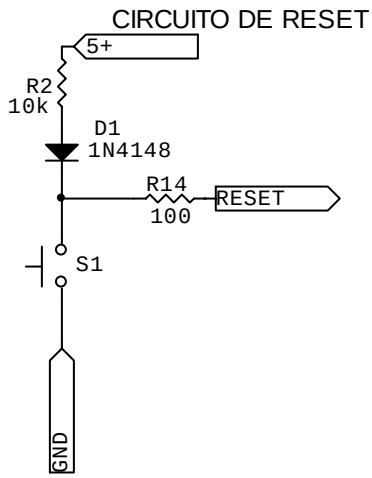


Figura 68, Etapa Reset del Sistema

5.3.1.8. Etapa Microcontrolador PIC 16F873

Realiza dos funciones diferentes.

- Se encarga de realizar el control Fuzzy exclusivamente desde el microcontrolador PIC, al dispositivo le llegan las señales de temperatura y cruce por cero, provenientes de las anteriores etapas. Realiza la conversión A/D de las señales de temperatura y *set Point*, ejecuta el algoritmo de control Fuzzy y genera la señal de control o disparo que activa el elemento de potencia.
- Cuando se realiza el control Fuzzy desde el computador esta opción es seleccionado por software desde el mismo(computador), el microcontrolador realiza las siguientes funciones:
  - Conversión A/D de las señales de temperatura y set Point,
  - Ejecución de la comunicación serie entre el microcontrolador y el PC
  - Proporciona la señal de disparo para el TRIAC de acuerdo al ángulo de conducción transferido desde el computador por el puerto serial.

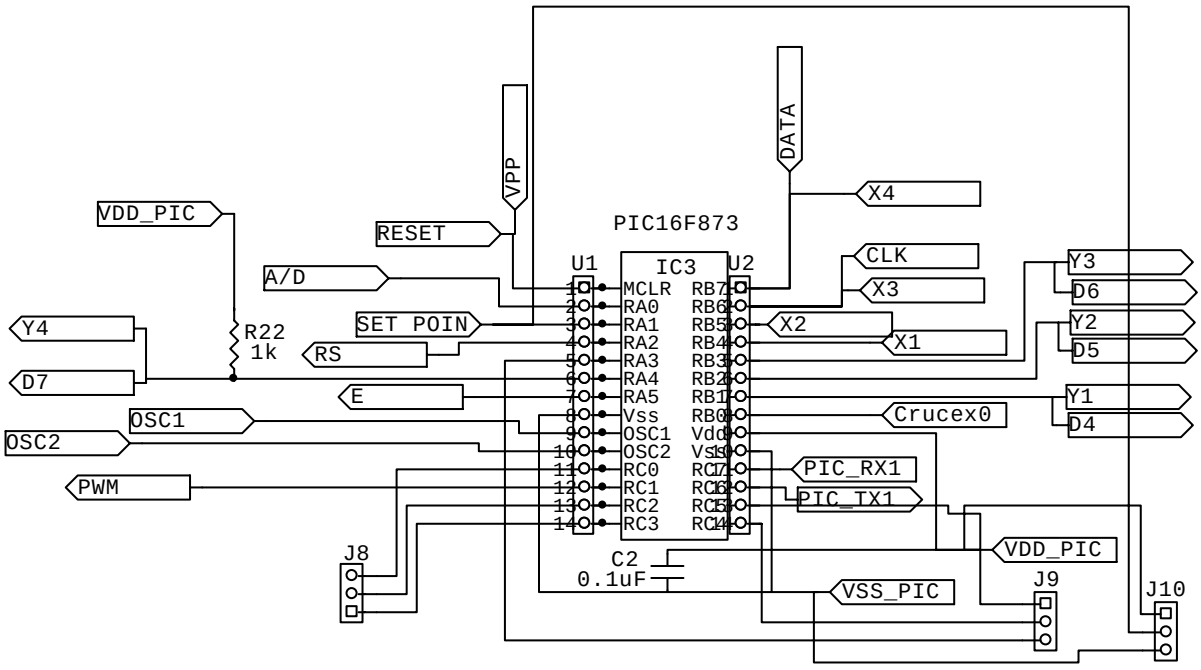


Figura 69. Etapa Microcontrolador PIC 16F873

5.3.1.9. Etapa Microcontrolador Motorola MC6HC908JL3

Al igual que la anterior etapa realiza las mismas dos funciones (control desde el microcontrolador ó control desde el computador).

- Control Fuzzy desde el microcontrolador MC6HC908JL3, recibe las señales de las seis primeras etapas. Realiza la conversión A/D de las señales de temperatura y set point, ejecuta el algoritmo de control Fuzzy y genera la señal de control o disparo que activa el elemento de potencia.
- Control Fuzzy desde el computador, esta opción es seleccionada por software desde el mismo (computador), el microcontrolador ejecuta las mismas acciones que la etapa anterior para este caso, realiza la conversión A/D de las señales de temperatura, ejecuta la comunicación serie entre el microcontrolador y el PC, proporcionar la señal de disparo para el TRIAC de acuerdo al ángulo de conducción transferido desde el computador por el puerto serial.

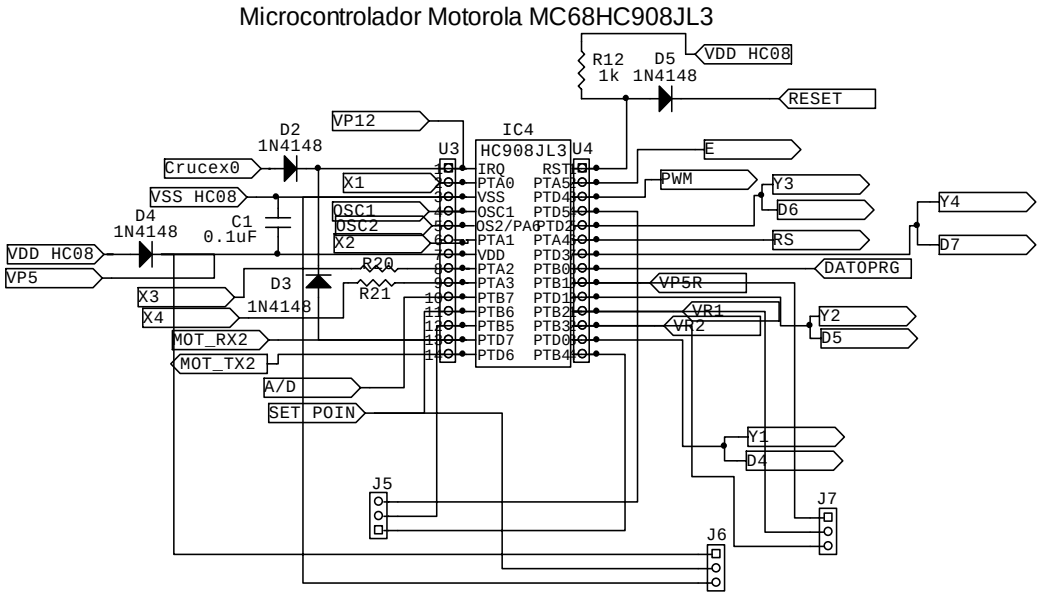


Figura 70, Etapa Microcontrolador Motorola MC6HC908JL3

5.3.1.10. Etapa de Interfase para Comunicación Serial Asíncrona con el PC

Es la encargada de realizar la conversión de los niveles de voltaje TTL a los niveles del protocolo serial RS232. La etapa consiste del circuito integrado MAX232 que realiza la conversión y cinco condensadores para configuración del MAX232. Para conexión física con el computador se utiliza un cable blindado serie.

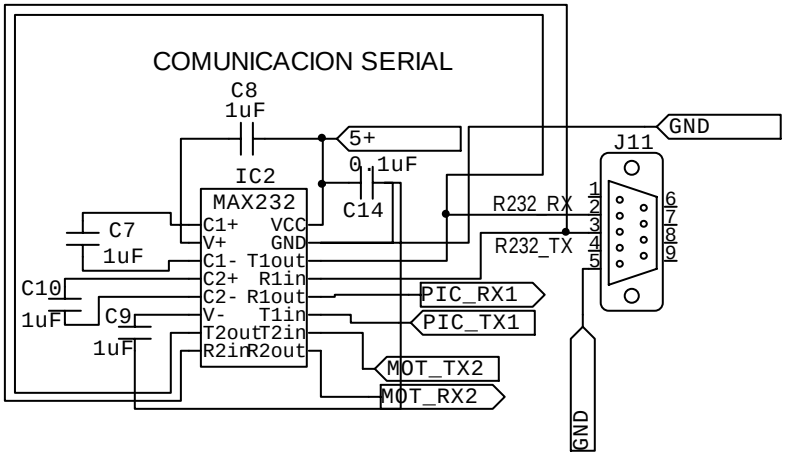


Figura 71, Interfase para Comunicación Serial Asíncrona con el PC

5.3.1.11. Terminales Varios de Entrada-Salida.

El modulo 2 posee conectores de acceso a todos los pines de los dos microcontroladores

- PIC: U1,U2.
- Motorola: U3,U4

Adicionalmente se cuenta con borneras de acceso rápido a los pines sin utilizar de ambos microcontroladores

- PIC: J8,J9,J10.
- Motorola: J5,J6,J7

La alimentación de la tarjeta se realiza mediante el conector J2. A este conector tan bien le llega la señal de cruce por cero proveniente del modulo 1, y la señal PWM que se dirige hacia el TRIAC.

Los conectores J4 y J3 se utilizan para verificar los pines de grabación e cada dispositivo.

6. RESULTADOS

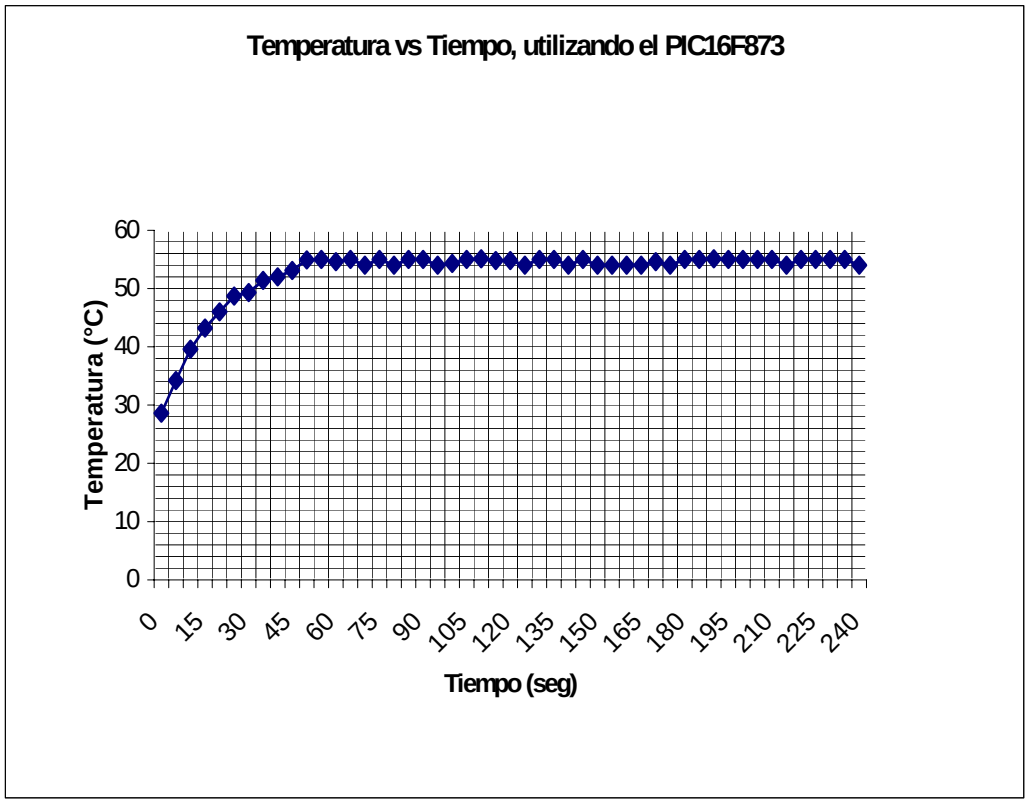
Se comprobó el control de temperatura, mediante diferentes archivos de caracterización Fuzzy diseñados a partir del Programa FuzzyE3T, probados tanto para el controlador desde el Computador como desde los Microcontroladores.

Los experimentos representativos de los controladores Fuzzy implementados se presentan enseguida.

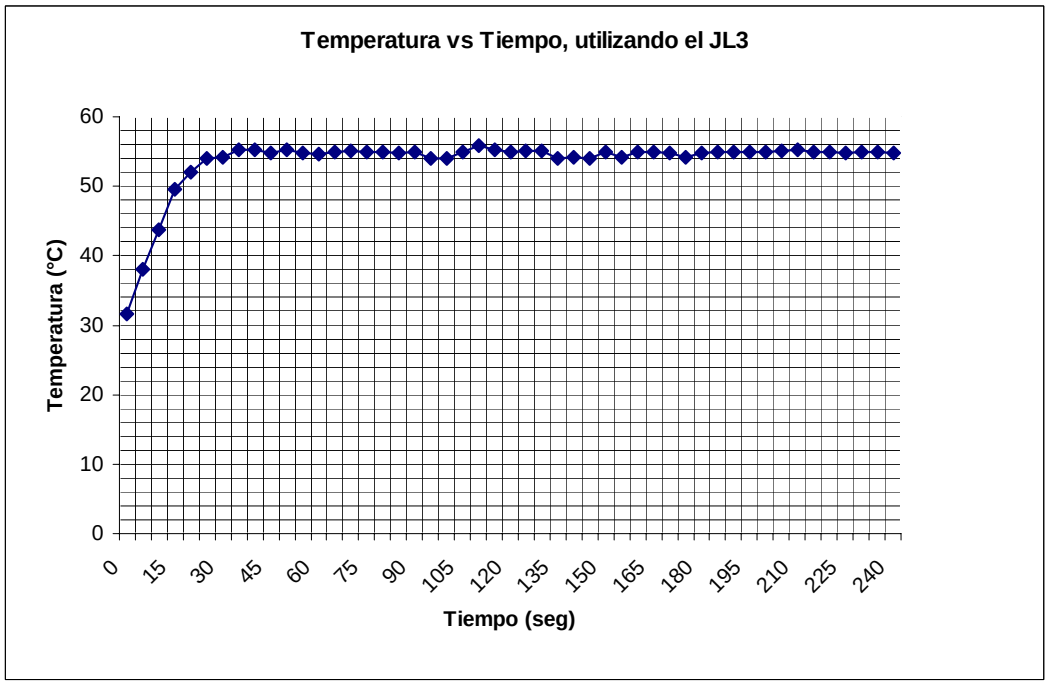
6.1. PRUEBA CON LOS MICROCONTROLADORES

Estas pruebas se realizan utilizando el archivo de caracterización Fuzzy denominado "fuzzypic_+-16°_21.fc".

6.1.1. Primera prueba: *Setpoint* de Temperatura: 55° Celsius. Temperatura inicial del Conversor A/D: 28 °C. Utilizando como elemento calefactor un bombillo incandescente de 120 Vac, 100 Vatios.
A continuación se presentan los datos de temperatura registrados en la Tarjeta de Control Fuzzy con los microcontroladores (PIC y MOTOROLA) durante tres (3) minutos en intervalos constantes de tiempo de cinco (5) segundos. Las gráficas 72 y 73 muestran la evolución de la temperatura del elemento calefactor ante condiciones ambientales estables (28 ° Celsius).



Figuras 72 (arriba), 73 (abajo)



Tiempo en el cual se alcanzó el 95 % del *setpoint* fue aproximadamente 25 segundos.

Máximo valor de rizo fue de 1.2 ° Celsius.

6.1.2. **Segunda prueba**: Para varios puntos de control diferentes y con *Setpoint* de Temperatura: 55° Celsius. Temperatura inicial del Conversor A/D: 28 °C se muestra la gráfica del voltaje aplicado a la carga.

En cada punto de control la variable variación del error es uno y se calcula la variable error (*Setpoint* de Temperatura- Temperatura del Conversor A/D), a continuación se busca que reglas se cumplen en este punto.

- Punto de control uno

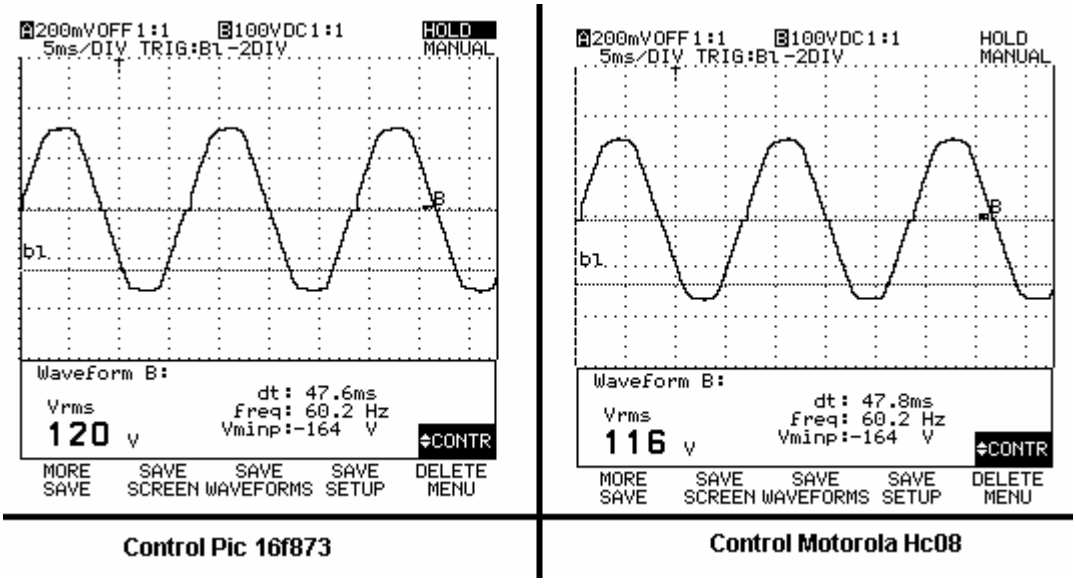
  Variable Error = 20 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 35° Celsius

Graficas de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3 par un Error igual a 20º C .



Las reglas que se cumplen en este instante son la 23 y 24 las cuales dicen.

Regla 23:si Error es grande (1) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24:si Error es grande (1) y Varerror es Peqpos (0.5) entonces la potencia es max. La salida del controlador es cero.

- Punto de control dos

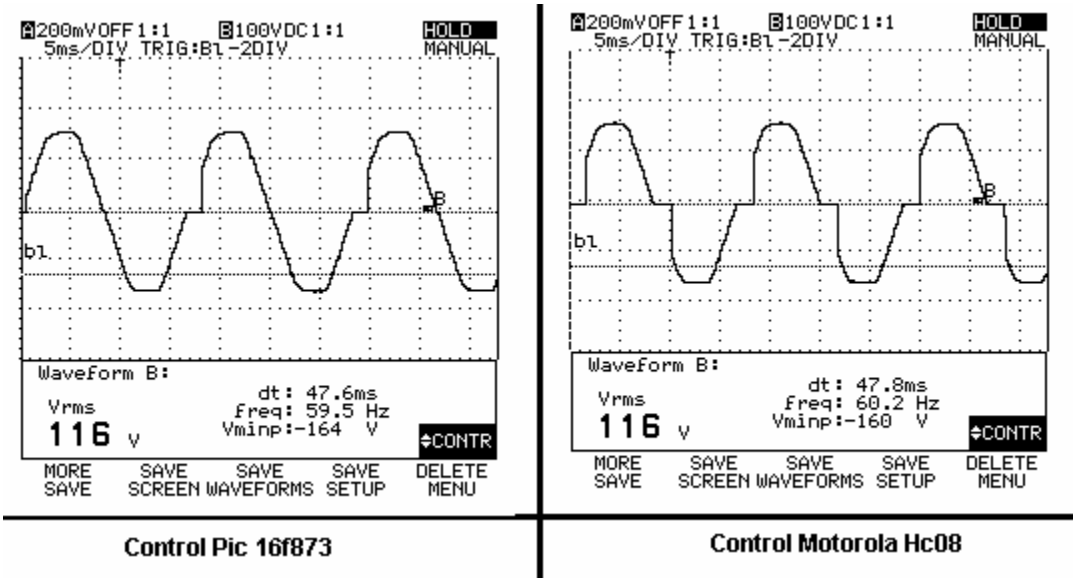
  Variable Error = 15 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 40° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



Las reglas que se cumplen en este instante son las 18,19, 23 y 24 las cuales dicen.

Regla 18: si Error es Med (0.1) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.1) y Varerror es Peqpos (0.5) entonces la potencia es Med.

Regla 23:si Error es grande (0.9) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24:si Error es grande (0.9) y Varerror es Peqpos (0.5) entonces la potencia es max. La salida del controlador es 14.5

- Punto de control tres.

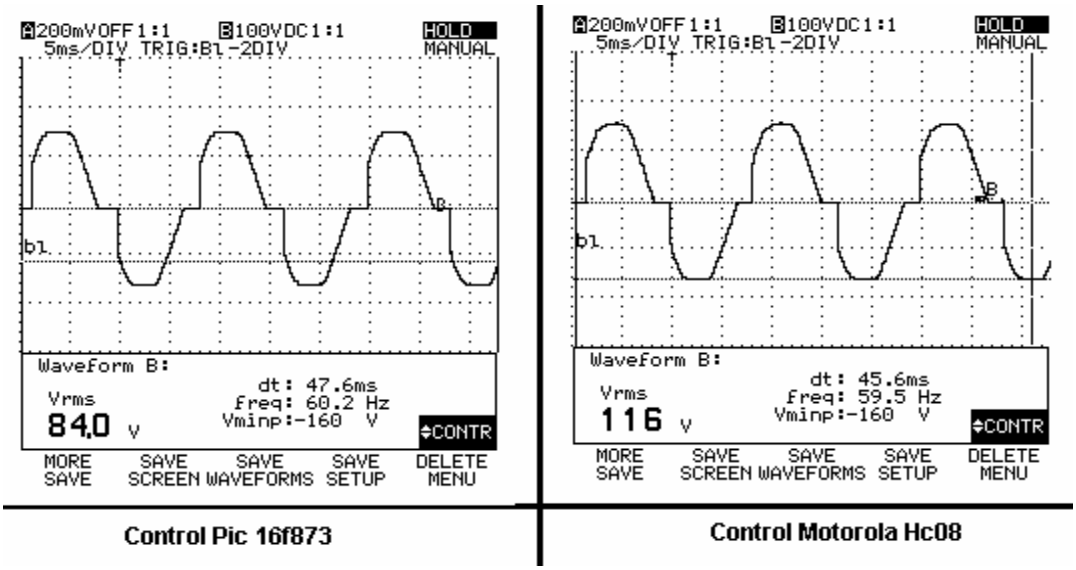
  Variable Error = 10 ° Celsius

Variable Varerror= 1° Celsius.

Setpoint= 55° Celsius.

Temperatura leída (A/D)= 45° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



Las reglas que se cumplen en este instante son las 18,19, 23 y 24 las cuales dicen.

Regla 18: si Error es Med (0.6) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.6) y Varerror es Peqpos (0.5) entonces la potencia es Med.

Regla 23:si Error es grande (0.4) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24:si Error es grande (0.4) y Varerror es Peqpos (0.5) entonces la potencia es max.

La salida del controlador es 49.91

- Punto de control cuatro

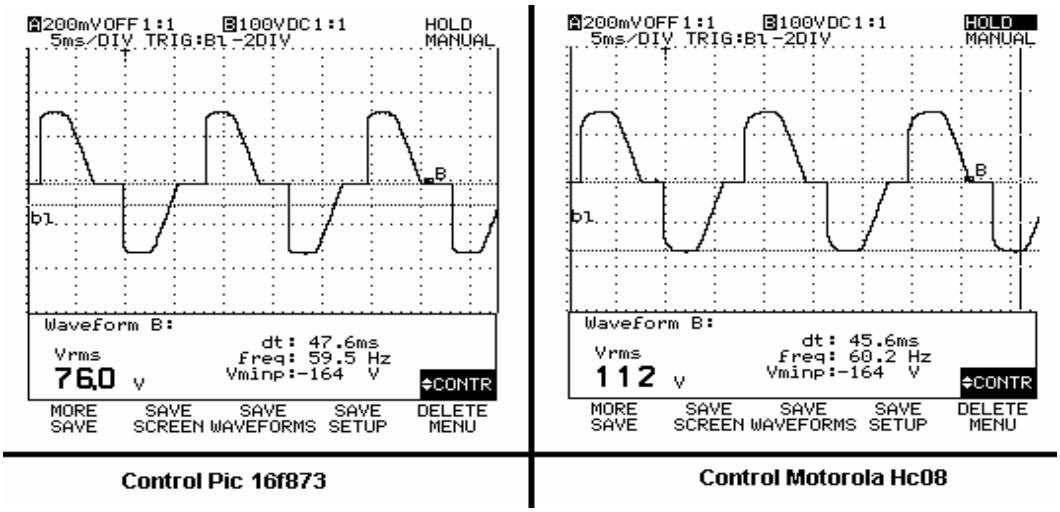
  Variable Error = 8 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55°Celsius.

  Temperatura leída (A/D)= 57° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



Las reglas que se cumplen en este instante son las 18,19, 23 y 24 las cuales dicen.

Regla 18: si Error es Med (0.8) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.8) y Varerror es Peqpos (0.5) entonces la potencia es Med.

Regla 23:si Error es grande (0.2) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24:si Error es grande (0.2) y Varerror es Peqpos (0.5) entonces la potencia es max.

La salida del controlador es 64.29

- Punto de control cinco.

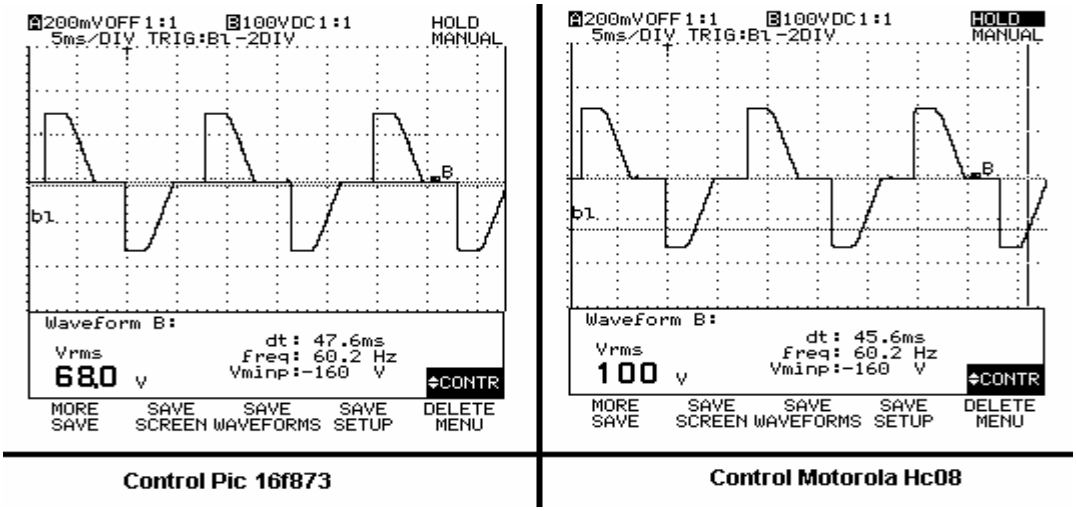
  Variable Error =6 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 49 ° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



Control Pic 16f873 | **Control Motorola Hc08**

Las reglas que se cumplen en este instante son las 13,14,18 y 19 las cuales dicen.

Regla 13:si Error es Peq (0.01) y Varerror es cero (0.5) entonces la potencia es Medb.

Regla 14:si Error es Peq (0.01) y Varerror es Peqpos (0.5) entonces la potencia es Medb.

Regla 18: si Error es Med (0.99) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.99) y Varerror es Peqpos (0.5) entonces la potencia es Med.

La salida del controlador es 90.11

- Punto de control seis

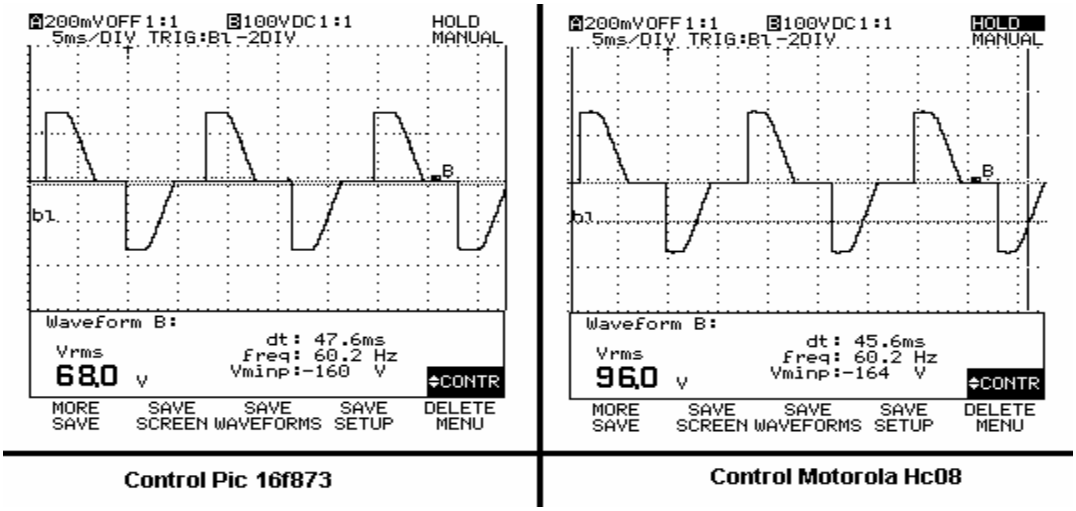
  Variable Error = 4 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55°Celsius.

Temperatura leída (A/D)= 51 ° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



Las reglas que se cumplen en este instante son las 13,14,18,19 las cuales dicen.
Regla 13:si Error es Peq (0.67) y Varerror es cero (0.5) entonces la potencia es Medb.
Regla 14:si Error es Peq (0.67) y Varerror es Peqpos (0.5) entonces la potencia es Medb.
Regla 18: si Error es Med (0.33) & Varerror es cero (0.5) entonces la potencia es Med.
Regla 19: si Error es Med (0.33) y Varerror es Peqpos (0.5) entonces la potencia es Med.
La salida del controlador es 96

- Punto de control siete

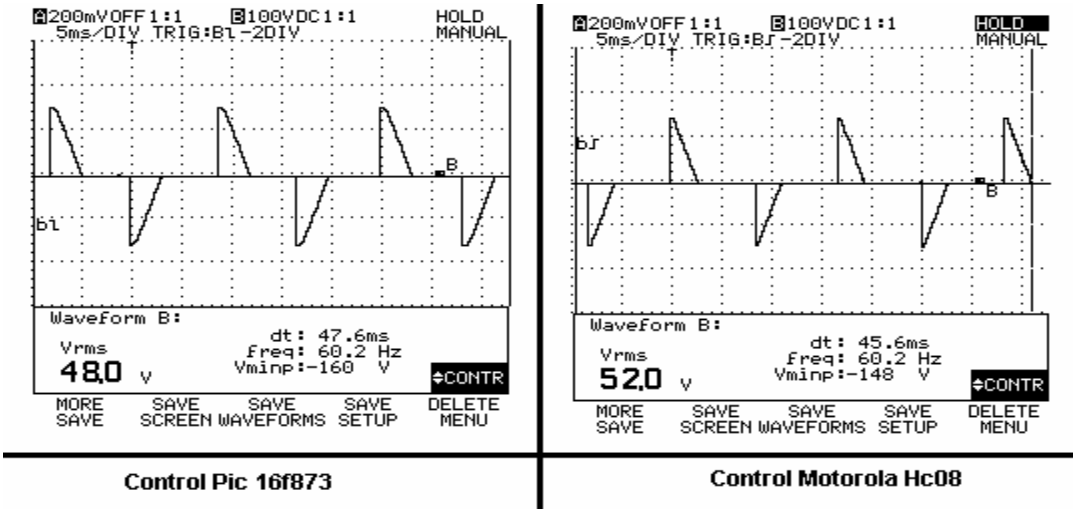
  Variable Error = 1 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 54 ° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



Control Pic 16f873 **Control Motorola Hc08**

Las reglas que se cumplen en este instante son las 3,4,8,9 las cuales dicen.

Regla 3:si Error es cero (0.01) y Varerror es cero (0.5) entonces la potencia es cero.

Regla 4:si Error es cero (0.01) y Varerror es Peqpos (0.5) entonces la potencia es cero.

Regla 8: si Error es Min (0.99) & Varerror es cero (0.5) entonces la potencia es Medb.

Regla 9: si Error es Min (0.99) y Varerror es Peqpos (0.5) entonces la potencia es cero.

La salida del controlador es 125

- Punto de control ocho.

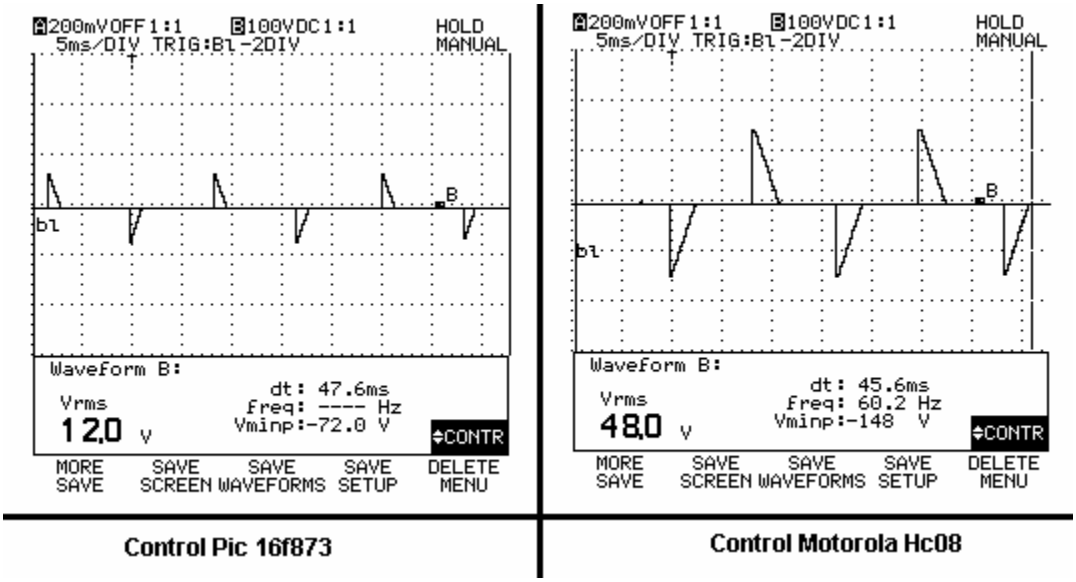
  Variable Error = 0 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55°Celsius.

  Temperatura leída (A/D)= 55° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el microcontrolador PIC 16f873 o el Motorola HC08JL3.



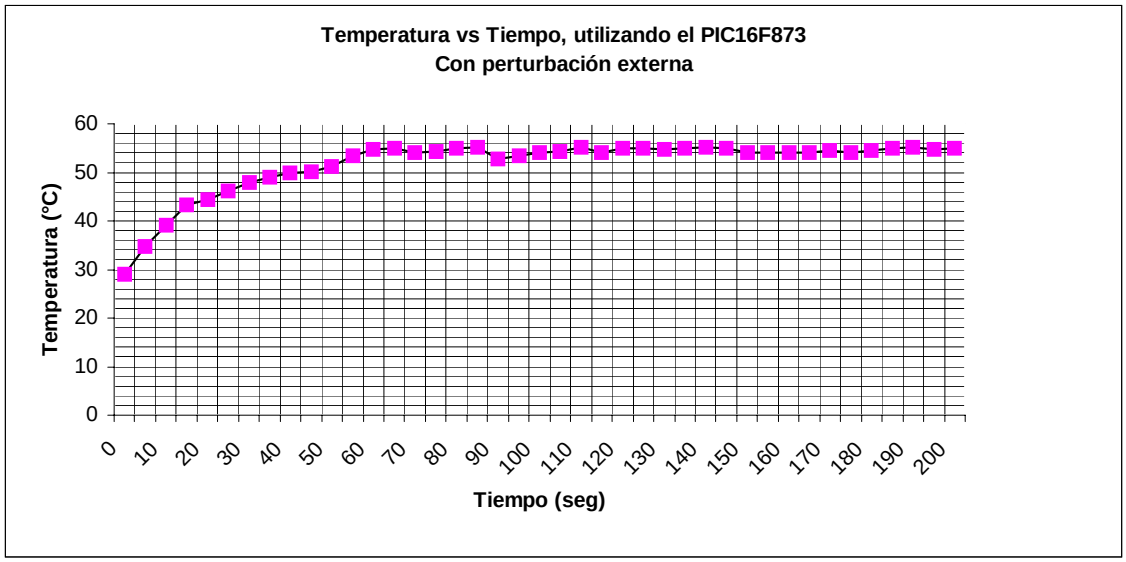
Las reglas que se cumplen en este instante son las 3 y 4 las cuales dicen.

Regla 3:si Error es cero (0.01) y Varerror es cero (0.5) entonces la potencia es cero.

Regla 4:si Error es cero (0.01) y Varerror es Peqpos (0.5) entonces la potencia es cero.

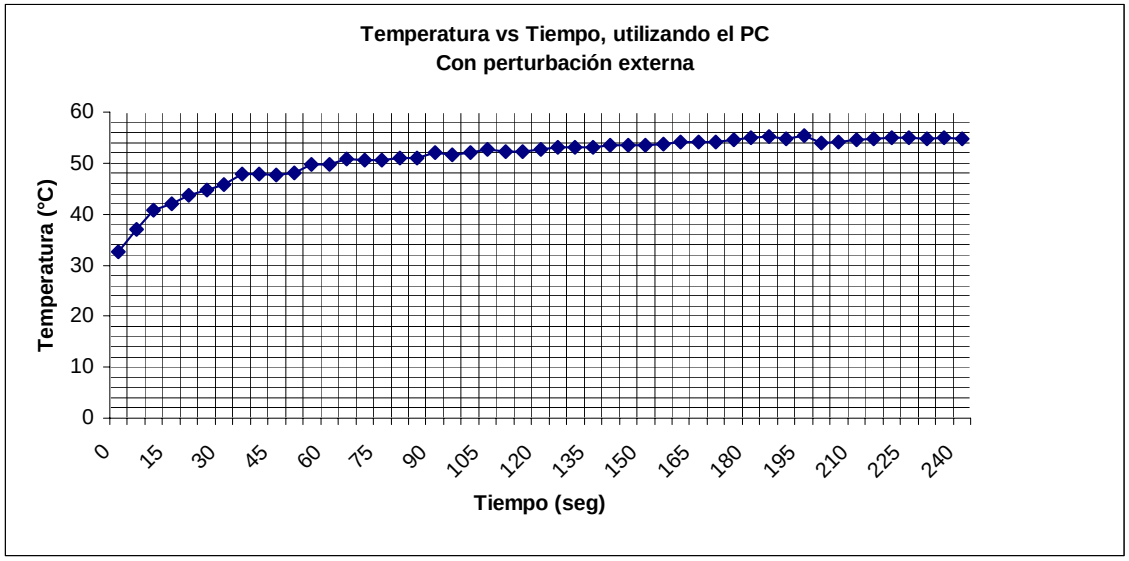
La salida del controlador es 150.

6.1.3. **Tercera prueba**: *Setpoint* de Temperatura: 55° Celsius. Temperatura inicial del Conversor A/D: 28 °C. Cambiando el elemento calefactor (reflector de 75 Vatios, 120 Vac) y adicionando el efecto de un factor externo como lo es la brisa generada por un extractor de computador (12 Vdc, 120 mA; 7.5 cm de diámetro) colocado en la parte frontal del elemento calefactor, luego de un (1) minuto de iniciada la prueba (tiempo en el cual se estabiliza la temperatura a menos del 1.1 ° Celsius de rizo). La gráfica 7.3 nos muestra la forma como se recupera la estabilidad de la curva de temperatura ante la distorsión (la brisa se deja permanentemente a partir del punto en el cual se inicia).



6.2. PRUEBA UTILIZANDO EL COMPUTADOR

La tendencia de temperatura vs. Tiempo para el sistema de calefacción con el bombillo incandescente utilizando el controlador Fuzzy basado en el Computador se muestra en la siguiente gráfica:



Para varios puntos de control diferentes y con *Setpoint* de Temperatura: 55° Celsius.

Temperatura inicial del Conversor A/D: 28 °C se muestra la gráfica del voltaje aplicado a la carga.

En cada punto de control la variable variación del error es uno y se calcula la variable error (*Setpoint* de Temperatura- Temperatura del Conversor A/D), a continuación se busca que reglas se cumplen en este punto.

- Punto de control uno

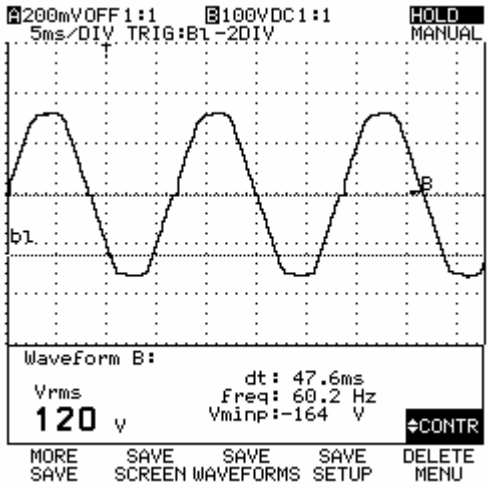
  Variable Error = 20 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 35° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el computador y la adquisición de temperatura el microcontrolador PIC 16f873.



Las reglas que se cumplen en este instante son la 23 y 24 las cuales dicen.

Regla 23: si Error es grande (1) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24: si Error es grande (1) y Varerror es Peqpos (0.5) entonces la potencia es max.

La salida del controlador es cero.

- Punto de control dos.

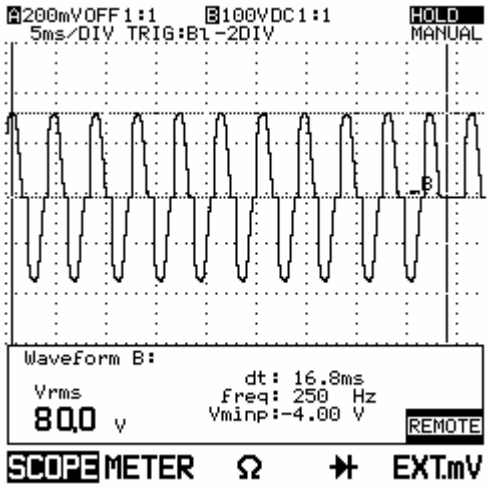
  Variable Error = 15 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 35° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el computador y la adquisición de temperatura el microcontrolador PIC 16f873.



Las reglas que se cumplen en este instante son las 18,19, 23 y 24 las cuales dicen.

Regla 18: si Error es Med (0.1) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.1) y Varerror es Peqpos (0.5) entonces la potencia es Med.

Regla 23: si Error es grande (0.9) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24: si Error es grande (0.9) y Varerror es Peqpos (0.5) entonces la potencia es max.

La salida del controlador es 14.5

- Punto de control tres.

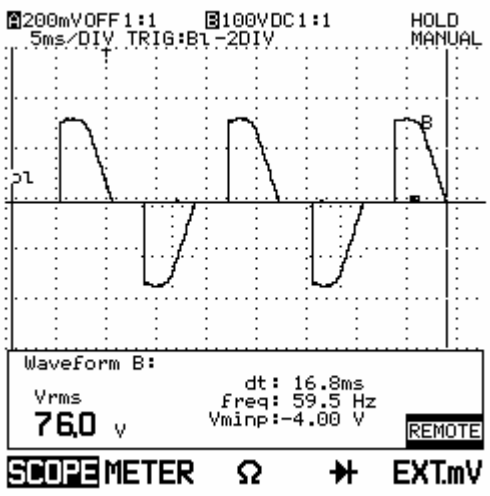
  Variable Error = 10 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

Temperatura leída (A/D)= 35° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el computador y la adquisición de temperatura el microcontrolador PIC 16f873.



Las reglas que se cumplen en este instante son las 18,19, 23 y 24 las cuales dicen.

Regla 18: si Error es Med (0.6) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.6) y Varerror es Peqpos (0.5) entonces la potencia es Med.

Regla 23: si Error es grande (0.4) y Varerror es cero (0.5) entonces la potencia es max.

Regla 24: si Error es grande (0.4) y Varerror es Peqpos (0.5) entonces la potencia es max.

La salida del controlador es 49.91

- Punto de control cuatro

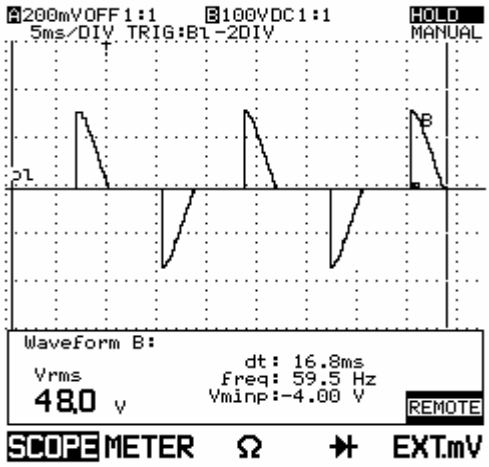
  Variable Error = 4 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 35° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el computador y la adquisición de temperatura el microcontrolador PIC 16f873.



Las reglas que se cumplen en este instante son las 13,14,18,19 las cuales dicen.

Regla 13:si Error es Peq (0.67) y Varerror es cero (0.5) entonces la potencia es Medb.

Regla 14:si Error es Peq (0.67) y Varerror es Peqpos (0.5) entonces la potencia es Medb.

Regla 18: si Error es Med (0.33) & Varerror es cero (0.5) entonces la potencia es Med.

Regla 19: si Error es Med (0.33) y Varerror es Peqpos (0.5) entonces la potencia es Med.

La salida del controlador es 96

- Punto de control cinco.

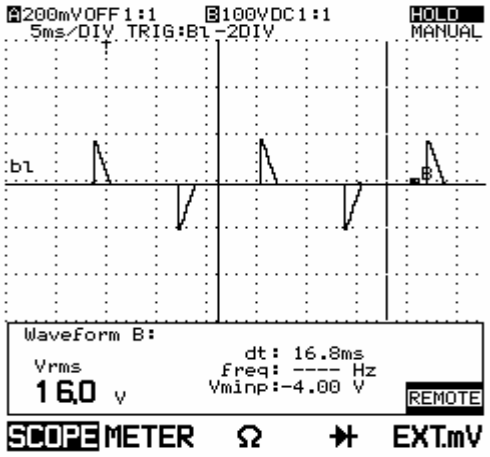
  Variable Error = 1 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 35° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el computador y la adquisición de temperatura el microcontrolador PIC 16f873.



Las reglas que se cumplen en este instante son las 3,4,8,9 las cuales dicen.

Regla 3:si Error es cero (0.01) y Varerror es cero (0.5) entonces la potencia es cero.

Regla 4:si Error es cero (0.01) y Varerror es Peqpos (0.5) entonces la potencia es cero.

Regla 8: si Error es Min (0.99) & Varerror es cero (0.5) entonces la potencia es Medb.

Regla 9: si Error es Min (0.99) y Varerror es Peqpos (0.5) entonces la potencia es cero.

La salida del controlador es 125

- Punto de control seis.

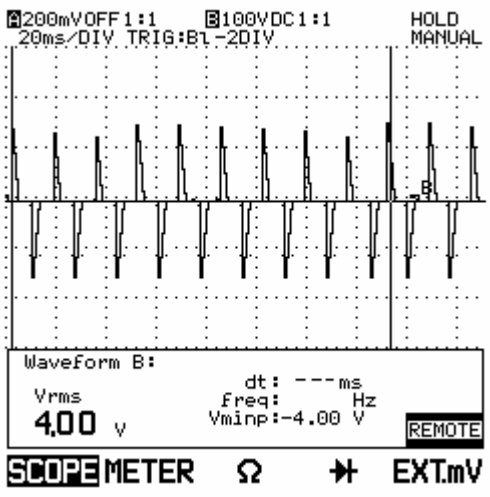
  Variable Error = 0 ° Celsius

  Variable Varerror= 1° Celsius.

  Setpoint= 55° Celsius.

  Temperatura leída (A/D)= 35° Celsius

Grafica de voltaje presente en la carga cuando el control lo realiza el computador y la adquisición de temperatura el microcontrolador PIC 16f873.



Las reglas que se cumplen en este instante son las 3 y 4 las cuales dicen.

Regla 3:si Error es cero (0.01) y Varerror es cero (0.5) entonces la potencia es cero.

Regla 4:si Error es cero (0.01) y Varerror es Peqpos (0.5) entonces la potencia es cero.

La salida del controlador es 150.

7. CONCLUSIONES

- Al implementar un controlador *fuzzy* con un microcontrolador de ocho (8) bits de propósito general, se deben tener en cuenta las restricciones de dicho dispositivo (como por ejemplo en capacidad de memoria y frecuencia de operación) pues estas pueden determinar la no aplicabilidad a un sistema de control dado. En cuánto a software, una restricción propia del formato de control generado con "FuzzyE3T" es que sólo permite dos variables de entrada y una de salida; sin embargo, este programa puede ser colocado como una subrutina para ser llamado en diferentes partes de un programa principal y así conseguir que el control se extienda a un mayor número de variables de entrada y salida.

- El control de temperatura implementado en este proyecto con lógica Fuzzy, posee las ventajas de no necesitar un modelamiento matemático preciso del sistema a controlar, tiene alto rechazo al ruido inherente del sistema y la capacidad para mantener el control ante factores externos que tienden a cambiar la naturaleza misma del sistema (como por ejemplo, el enfriamiento que produce una fuerte brisa sobre el elemento calefactor); esto, debido a la forma como se representa el conocimiento. Como desventaja se puede citar la necesidad de hacer ajustes al controlador por prueba y error, más aún si no se tiene una experiencia considerable en el manejo del proceso a controlar.

- El programa en LabVIEW de generación de código ensamblador diseñado en este proyecto, permite alcanzar el propósito intrínseco de proporcionar una herramienta de desarrollo de controladores con Lógica Fuzzy construidos en hardware de bajo costo y tamaño reducido. Además, al generar el código en lenguaje ensamblador se obtiene un control más eficiente sobre los dispositivos ya que no se incluyen bucles

innecesarios que aumenten el tamaño del código y el tiempo de ejecución de los programas.

- El controlador Fuzzy basado en el Computador (LabVIEW) y una tarjeta de adquisición y control es idóneo para sistemas centralizados y de una complejidad considerable, sin embargo para sistemas distribuidos más sencillos los controladores Fuzzy basados en microcontroladores (Assembler) demuestran su versatilidad y adaptabilidad a un costo notablemente reducido.

- En cuánto a la selección del microcontrolador, se deben tener en cuenta las características propias de la aplicación en particular. La elección del microcontrolador debe ser un compromiso entre las ventajas y desventajas de cada uno con respecto al otro. Aunque no fue un objetivo de esta tesis hacer esta comparación, se pueden dar los siguiente tópicos como referencia:

1. Si se requiere una mayor resolución en la conversión de datos de A/D el microcontrolador PIC16F873 es superior ya que dispone de 10 bits (1024 estados posibles) mientras el MC68HC908JL3 solo posee 8 bits (256 estados posibles).

2. En cuánto a tamaño de código, en el MC68HC908JL3 se optimiza más el espacio en memoria debido a que posee rutinas lógico-matemáticas (comparaciones, multiplicación de 8x8 bits, suma de 16 bits y división de 16, entre otros) más completas dentro de su repertorio de instrucciones. En el PIC16F873 se deben realizar las rutinas por el usuario.

3. El PIC16F873 cuenta con un módulo de Comunicación serie (sincrono y asíncrono), mientras en el MC68HC908JL3 el usuario debe implementar las rutinas por software para este fin.

4. El MC68HC908JL3 es más versátil para implementar aplicaciones de PWM a 60 Hz utilizando frecuencias de oscilación (de reloj) mayores que el PIC16F873. Esto se debe

principalmente a que un módulo de PWM en el JL3 consta de 16 bits mientras que en el PIC sólo de 10 bits.

- Por último, es importante decir que el formato Fuzzy para microcontroladores de 8 bits diseñado en este proyecto es fácilmente adaptadable de un dispositivo a otro (en este caso, se realizó primero para el PIC y luego se adaptado al MOTOROLA). Esto permite sugerir que en nuevos trabajos de grado se adopte este formato como guía para la implementación de controladores Fuzzy en otros dispositivos como por ejemplo los DSP's.

Dos razones principalmente se tuvieron en cuenta para la utilización de LabVIEW como plataforma de desarrollo (En lugar de C, por ejemplo): La programación grafica proporciona mayores facilidades en tiempo de desarrollo de una aplicación que su homologa en líneas de programa (nemonicos). La otra razón es que LabVIEW se ha adoptado en la E3T como uno de los pilares de la Instrumentación y control, por todos los beneficios que ofrece y por la legalidad de su licencia para la escuela.

Es importante comparar el trabajo hecho en esta tesis con lo mejor del mercado (FUZZY E3T vs FUZZYTECH, por ejemplo) para tener una visión de lo que se quiere lograr con este tipo de proyectos. Para un tipo de aplicaciones de baja complejidad se puede considerar adecuado el modelo de controlador de dos entradas y una salida (con seis términos linguisticos cada variable) como el caso de FUZZY E3T, sin embargo es recomendable extender el programa al doble de variables de entrada y salida para abarcar una cantidad superior de aplicaciones. Sin embargo, extenderse demasiado en él numero de variables haría poco practico el ejercicio de trabajar con los tipos de microcontroladores utilizados en esta tesis, en cuyo caso seria mejor utilizar microcontroladores de gama superior que ya incluyen las funciones de lógica fuzzy necesarias para el control.

Por ultimo, se debe tener en cuenta que el programa Fuzzy E3T queda en su primera versión con el código fuente disponible para ser retomado en otros proyectos para ser

llevado a un nivel mas alto de desarrollo de tal forma que pueda hacerse comercial. Por el momento se deben guardar las proporciones a la hora de hacer las comparaciones con los programas que llevan mas de 10 años en el mercado, y entender que en la medida que se apoye la generación de tecnología propia en nuestro entorno (especialmente en la universidad) en lugar del solo consumismo que nos rodea, podremos dar nuestro granito de arena al desarrollo de nuestro país. Sin embargo, se cumple el propósito inicial planteado en este proyecto de dar una herramienta en software y hardware para el diseño e implementación de controladores fuzzy en microcontroladores de gama media y básica de MICROCHIP y MOTOROLA; para aplicaciones de bajo costo.

8. REFERENCIAS BIBLIOGRÁFICAS

MENDEL, Jerrry M. Uncertain Rule_Based Fuzzy Logic Systems : Introduction And New Directions. Saddle River, Nj. Prentice Hall, 1997.

KLIR, George J. YUAN, Bo. *Fuzzy sets and fuzzy logic: theory and applications.* Upper Saddle River, Nj. Prentice Hall, 1995.

TERANO, Toshiro. SUGENO, Michio. ASAI, Kiyoji. Fuzzy Systems Theory And Its Applications. Second Edition. Boston, Ma. Academic Press, 1992.

PIMIENTA OSORIO, Carlos Alberto. ROMERO BAYONA, Rosa. Prototipado De Un Controlador *Fuzzy* Utilizando *Sie_Fuzzy* a través de una Tarjeta Daq *National Instruments*. Tesis . Bucaramanga. Universidad Industrial De Santander, 1998.

National Instruments. Manual de Usuario *LabView*. PDF, U.S.A. Edition *Part Number* 320999C-01, Julio 2002

Microchip Technology Inc. *Data Sheet* PIC16F87X. PDF, USA. Numero de Inventario Interno DS30292C. *Microchip Technology* Inc. 2001

Palmer, Mark. *Using the CCP Module(s*). USA. Nota de Aplicación Pdf AN594. Numero de Inventario Interno DS00594B. *Microchip Technology* Inc. 1997

Palacherla, Amar. PIC16C5X / PIC16CXXX *Math Utility Routines*. USA. Nota de Aplicación Pdf AN526. Numero de Inventario Interno DS00526E. *Microchip Technology* Inc. 1997

Palacherla, Amar. *Math Utility Routines*. USA. Nota de Aplicación Pdf AN544. Numero de Inventario Interno DS00544D. *Microchip Technology* Inc. 1997

Motorola, Inc. *Technical Data* MC68HC908JL3. Manual PDF MC68HC908JL3. Numero de Inventario Interno MC68H(R)C908JL3-Rev. 1.0. Motorola, Inc 1999

Motorola, Inc. Cpu08 *Central Processor Unit Reference Manual*. Manual PDF CPU08RM/AD. Numero de Inventario Interno CPU08RM/AD-Rev. 2. Motorola, Inc 1996

Motorola, Inc. Tim08 *Timer Interface Module Reference Manual*. Manual PDF TIM08RM/AD. Numero de Inventario Interno TIM08RM/AD-Rev. 1.0. Motorola, Inc 1996

Motorola, Inc. *AdcAnalog-To-Digital ConverterReference Manual*. Numero de Inventario Interno Adc *Reference Manual*. Motorola, Inc

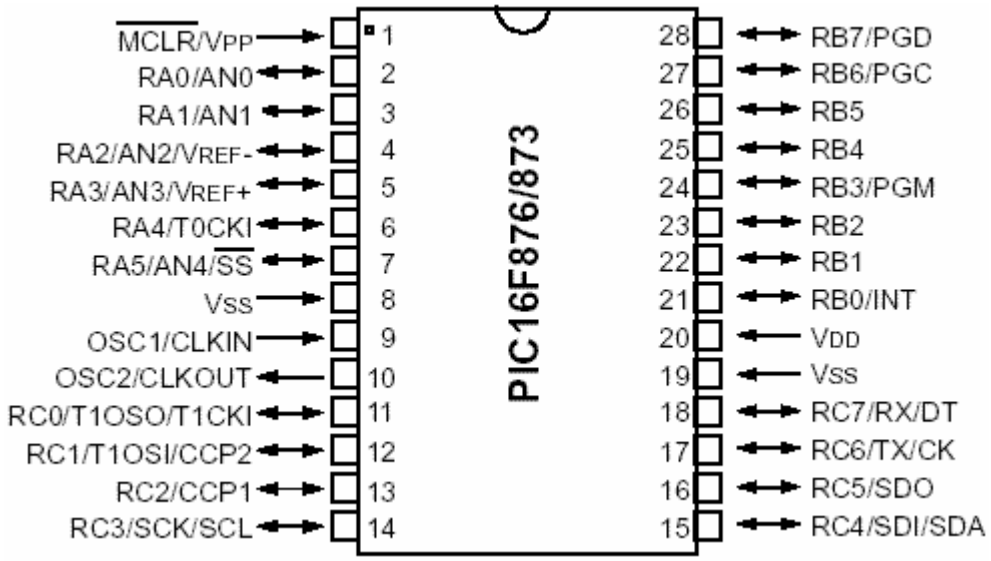
Jonson, Mark. M68HC08 *Integer Math Routines*. Nota de aplicación Pdf AN1219/D. Numero de Inventario Interno AN1219/D -Rev 1.0. Austin, Texas. Motorola, Inc 1996

National Instruments. Web Site: www.ni.com/labview
National Instruments. Web Site: www.ni.com/manuals
Microchip Technology Inc. Web Site: http://www.microchip.com/
Motorola, Inc. Web Site: http://www.motorola.com/sps/

ANEXO I

8.1. CARACTERISTICAS DE LOS MICROCONTROLADORES UTILIZADOS EN LA APLICACIÓN (16F873, MCHC908JL3).

8.2. I.1 MICROCONTROLADOR PIC16F873:



- Cpu tipo RISC

- ALU de 8 bits y registro de trabajo W del que normalmente recibe un operando que puede ser cualquier registro, memoria, puerto de Entrada/Salida o el propio código de instrucción.

- Contador de programa de 13 bits (lo que permite direccionar 4 kilo *byte* de memoria implementada).

  - Todas las instrucciones toman un ciclo sencillo excepto los saltos que toman dos ciclos.
  - Velocidad de operación: DC – Hasta 20 Mhz reloj de entrada depende del modelo.

- DC - 200 ns ciclo de instrucción.

- 4K de memoria de programa *FLASH*.

- Memoria de datos dividida en 2 áreas:
  - Memoria de datos *EEPROM* 128 bytes x 8 bits.
  - Memoria de datos *RAM* 192 bytes x 8 bits. Formada por 22 registros de propósito específico (SFR) y 36 de propósito general (GPR).

- Capacidades de interrupción (14 fuentes).

- Pila de ocho niveles de profundidad.

- Modos de direccionamiento Directo, indirecto y relativo.

- Reset al encendido (POR).
- *Timer Power-up* (PWRT) .

- *Watchdog Timer* (WDT) con oscilador propio RC dentro del dispositivo.

- Protección de código programable.

- Modo de SLEEP.

- Opciones de selección del oscilador.

- Tecnología de bajo consumo y alta velocidad CMOS *FLASH/EEPROM.*

- Programación serial *In-Circuit* (ICSP) vía dos pines

- Capacidad de programación serial con fuente de 5V *In-Circuit*.

- *Debugging In-Circuit* via dos pines.

- Rango de voltaje de operación : 2.0V a 5.5V

- Rangos Comerciales e Industriales de temperatura:

- Bajo consumo de potencia:
  - < 0.6 mA típica a 3V, 4 MHz
  - 20 mA típica a 3V, 32 Khz.

PERIFERICOS

- *Timer*0: 8-bits timer/contador con preescalador de 8-bits.

- *Time*r1: 16-bits timer/contador con preescalador de 8-bits. Puede ser incrementado durante el modo *SLEEP* vía señal externa reloj (cristal).

- *Timer*2: 8-bits timer/contador con 8-bits, preescalador y postescalador.

- Dos módulos de captura, comparación y *PWM*.
  - Resolución máx. de captura 16 bits a 12.5 ns.
  - Resolución máx. de comparación 16 bits a 200 ns.
  - Resolución máx. de Pwm es 10-bits
- Conversor análogo-digital 10-bits con 5 canales de entrada.

- Puerto serial sincrono (SSP) con modos : SPI(modo maestro) y I2Ç(maestro esclavo).

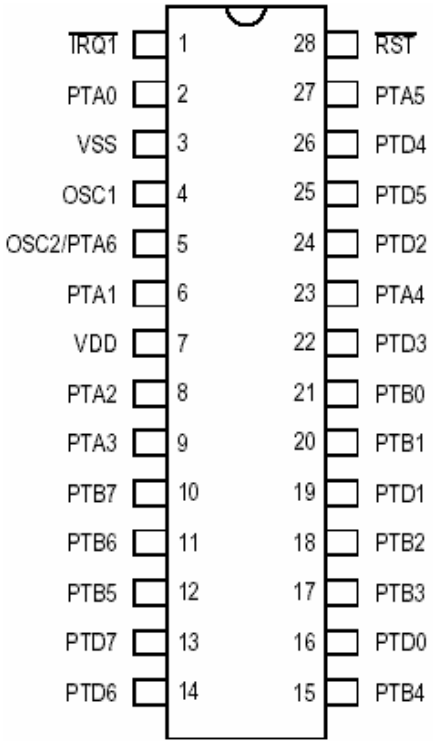
- Receptor transmisor universal sincrono asíncrono (USART/SCI).

8.3. I.2. CARACTERÍSTICAS MICROCONTROLADOR MC68HC908JL3:

- Cpu 08 tipo
- Acumulador de 8 bits, registro índice (h::x) de 16 bits para instrucciones de manipulación
- Puntero de stack de 16 bits
- Contador de programa de 16 bits
- Frecuencia de operación del bus de Cpu 8 MHZ
- Instrucción de multiplicación 8x8.
- Instrucción de división 16/8.
- Transferencia de datos de memoria a memoria
- 4096 bits de memoria de programa *FLASH*
- Memoria de datos *RAM* 128 bytes x 8 bits
- Compatible con la familia HC05
- Opciones de oscilador ,circuito RC o cristal hasta 32 Mhz
- Voltaje de operación a 5 voltios o 3 voltios

PERIFERICOS

- Modulo TIM contador de dos canales a 16 bits (captura, comparación y Pwm)
- Conversor de 12 canales de 8 bits de resolución
- 23 puertos de entrada salida de propósito general
  - Siete pines con interrupción para manejar teclado
  - Diez pines para manejar leds
- Modulo COP contra fallos *reset system*
- Pin de interrupción externa
- Pin de reset con pull-up interno
- 28 pines PDIP



Terminales microcontrolador Mchc908jl3

ANEXO I I

FORMATO DE PROGRAMA CONTROL *FUZZY* EN LENGUAJE ENSAMBLADOR PARA LOS MICROCONTROLADORES (16F873, MCHC908JL3).

8.4. II.1. MICROCONTROLADOR MOTOROLA MCHC908JL3.

Este formato se realizo en base al archivo de caracterización *fuzzypic*_+-16°_21.fc cuya documentación se muestra en el anexo III

- Variables utilizadas por la rutina fuzzye3t : Almacenadas en memoria *Ram* del microcontrolador:

```
s_max       ds 1
s_alta      ds 1
s_meda      ds 1
s_med       ds 1
```

```
s_medb          ds 1
s_cero          ds 1
SDEN_1          ds 1
MSNUM_1         ds 1
MSNUM_2         ds 1
error_lsb       ds 1  ;guarda datos de entrada
varerror_lsb    ds 1  ;guarda datos de entrada
Q1_lsb          ds 1
Q2_lsb          ds 1
Q3_1            ds 1
N_TERM_0        ds 1
N_TERM_1        ds 1
N_FUNC          ds 1
DNUMDEN_1       ds 1
DNUMDEN_2       ds 1
N_VERT          ds 1
N_VERT_0        ds 1
leftbottom_lsb  ds 1
rightbottom_lsb ds 1
lefttop_lsb     ds 1
righttop_lsb    ds 1
```

- **Vértices utilizados por rutina fuzzye3t: Almacenados en memoria de programa**

```
VERTICES
;*******error*varerror*************************************
        FCB  $00
        FCB  $00
        FCB  $00
        FCB  $10
        FCB  $00
        FCB  $10
        FCB  $10
        FCB  $30
        FCB  $10
        FCB  $30
        FCB  $30
```

```
        FCB  $60
        FCB  $30
        FCB  $60
        FCB  $60
        FCB  $FF
        FCB  $60
        FCB  $FF
        FCB  $FF
        FCB  $FF
        FCB  $00
        FCB  $00
        FCB  $00
        FCB  $60
        FCB  $00
        FCB  $60
        FCB  $60
        FCB  $80
        FCB  $60
        FCB  $80
        FCB  $80
        FCB  $9F
        FCB  $80
        FCB  $9F
        FCB  $9F
        FCB  $FF
        FCB  $9F
        FCB  $FF
        FCB  $FF
        FCB  $FF
```

- **Rutina principal fuzzye3t: Almacenada en memoria de programa**

```
FUZZYE3T
;*************************************
   ;**********Fuzificación*************
        clr    N_VERT
        clr    N_TERM_0
PER_0   mov    error_lsb,Q1_lsb
        jsr    MEMBR
        brset  7,DNUMDEN_2,S1
```

```
            jmp     INCR_0
S1          mov     DNUMDEN_1,error_lsb
            clr     N_TERM_1
        mov     N_VERT,N_VERT_0
        mov     #$20,N_VERT       ;*********remplazar en LabVIEW***************
PER_1   mov     vararror_lsb,Q1_lsb
        jsr     MEMBR
        brset   7,DNUMDEN_2,S2
        jmp     INCR_1
S2      mov     DNUMDEN_1,Q1_lsb
        mov     error_lsb,Q2_lsb
        jsr     REGLAS
INCR_1  inc     N_TERM_1
        lda     N_TERM_1
        cmp     #$05; ;numero indica la cantidad  de términos vararror
        bhs     S3
        jmp     PER_1
S3      mov     N_VERT_0,N_VERT
INCR_0  inc     N_TERM_0
        lda     N_TERM_0
        cmp     #$05 ;numero indica la cantidad  de términos error
        bhs     CENTROID
        jmp     PER_0
;**************************************
;       Defuzificación
CENTROID
            clra
            clr     MSNUM_1
            clr     MSNUM_2
            clr     SDEN_1
            clr     DNUMDEN_1
*MSNUM=(s_max*a1)+(s_alta*a2)+(s_meda*a3)+(s_med*a4)+(s_medb*a5)+(s_cero*a6)**
*SDEN_1=(h1+h2+h3+h4+h5+h6)=(s_max+s_alta+s_meda+s_med+s_medb+s_cero)**
            lda     SDEN_1
            add     s_Max
            sta     SDEN_1
            lda     s_Max
            ldx     #$00
            mul
            jsr     suma16
```

```
            lda     SDEN_1
            add     s_Alta
            sta     SDEN_1
            lda     s_Alta
            ldx     #$47
            mul
            jsr     suma16
            lda     SDEN_1
            add     s_Meda
            sta     SDEN_1
            lda     s_Meda
            ldx     #$5E
            mul
            jsr     suma16
            lda     SDEN_1
            add     s_Med
            sta     SDEN_1
            lda     s_Med
            ldx     #$80
            mul
            jsr     suma16
            lda     SDEN_1
            add     s_Medb
            sta     SDEN_1
            lda     s_Medb
            ldx     #$8E
            mul
            jsr     suma16
            lda     SDEN_1
            add     s_Cero
            sta     SDEN_1
            lda     s_Cero
            ldx     #$D4
            mul
            jsr     suma16
*** DNUMDEN = (MSNUM/SDEN_1) **************
        ldhx    MSNUM_2
        lda     MSNUM_1
        ldx     SDEN_1
        div
```

```
      sta    DNUMDEN_1
  rts
;***EL DATO DE SALIDA DE ESTA RUTINA ESTA AL MACENADO EN DNUMDEN_1****
```

8.5.

8.6. SUBRUTINAS UTILIZADAS POR LA RUTINA FUZZYE3T: ALMACENADA EN MEMORIA DE PROGRAMA.

```
;*******************************************************
;              SUMA Q1=Q1+Q2 de 16 bits
;*******************************************************
SUMA16                    ;Q1+Q2
      add    MSNUM_1
      sta    MSNUM_1
      bcc    SALTO
      inc    MSNUM_2
SALTO   txa
      add    MSNUM_2
      sta    MSNUM_2
  rts
;*******************************************************
;     SUBRUTINA DE GRADO DE MEMBRESÍA
MEMBR
      clrh
      ldx    N_VERT
            ldx    VERTICES,x
            stx    leftbottom_lsb
            inc    N_VERT
            ldx    N_VERT
            ldx    VERTICES,x
            sta    lefttop_lsb
            inc    N_VERT
            ldx    N_VERT
            ldx    VERTICES,x
            stx    righttop_lsb
            inc    N_VERT
            ldx    N_VERT
            ldx    VERTICES,x
            stx    rightbottom_lsb
            inc    N_VERT
```

```
IF_0
        lda     Q1_lsb
        cmp     leftbottom_lsb
        blo     SALE_1
IF_1
        cmp     rightbottom_lsb
        bhi     SALE_1
IF_2
    cmp     lefttop_lsb
    blo     RECT_A
    jmp     IF_3
RECT_A
        lda     lefttop_lsb
        sub     leftbottom_lsb
        sta     Q3_1
        lda     Q1_lsb
        sub     leftbottom_lsb
        ldx     #$80
        mul
        pshx
        pulh
        ldx     Q3_1
        div
        sta     DNUMDEN_1
        jmp     SALE_0
IF_3
        cmp     righttop_lsb
        bhi     RECT_D
        jmp     CTE
RECT_D
        lda     rightbottom_lsb
        sub     righttop_lsb
        sta     Q3_1
        lda     rightbottom_lsb
        sub     Q1_lsb
        ldx     #$80
        mul
        pshx
        pulh
        ldx     Q3_1
```

```
            div
            sta    DNUMDEN_1
            jmp    SALE_0
CTE
       mov   #$80,DNUMDEN_1
SALE_0   bset  7,DNUMDEN_2
       jmp   FUERA
SALE_1
       bclr  7,DNUMDEN_2
FUERA
  rts
;**************************************
;      SUBRUTINA DE MINIMO MAXIMO (retorna el max en Q1, el min en Q2)
MINMAX
            lda  Q1_lsb
            cmp  Q2_lsb
            bhs  NO_INTERC
            lda  Q1_lsb
            mov  Q2_lsb,Q1_lsb
            sta  Q2_lsb
NO_INTERC
  rts
;;**************************************
;      SUBRUTINA DE REGLAS
REGLAS              jsr    MINMAX
            lda    N_TERM_0
            ldx    #!05
            mul
            add    N_TERM_1
           cbeqa   #!0,Medb
           cbeqa   #!1,Cero
           cbeqa   #!2,Cero
           cbeqa   #!3,Cero
           cbeqa   #!4,Cero
           cbeqa   #!5,Alta
           cbeqa   #!6,Medb
           cbeqa   #!7,Medb
           cbeqa   #!8,Cero
           cbeqa   #!9,Cero
           cbeqa   #!10,Max
```

```
        cbeqa   #!11,Alta
        cbeqa   #!12,Medb
        cbeqa   #!13,Medb
        cbeqa   #!14,Medb
        cbeqa   #!15,Max
        cbeqa   #!16,Max
        cbeqa   #!17,Med
        cbeqa   #!18,Med
        cbeqa   #!19,Med
        cbeqa   #!20,Max
        cbeqa   #!21,Max
        cbeqa   #!22,Max
        cbeqa   #!23,Max
        cbeqa   #!24,Alta
Max
        mov     s_Max,Q1_lsb
        jsr     MINMAX
        mov     Q1_lsb,s_Max
        jmp     RETORNA
Alta
        mov     s_Alta,Q1_lsb
        jsr     MINMAX
        mov     Q1_lsb,s_Alta
        jmp     RETORNA
Meda
        mov     s_Meda,Q1_lsb
        jsr     MINMAX
        mov     Q1_lsb,s_Meda
        jmp     RETORNA
Med
        mov     s_Med,Q1_lsb
        jsr     MINMAX
        mov     Q1_lsb,s_Med
        jmp     RETORNA
Medb
        mov     s_Medb,Q1_lsb
        jsr     MINMAX
        mov     Q1_lsb,s_Medb
        jmp     RETORNA
Cero
```

```
            mov     s_Cero,Q1_lsb
            jsr     MINMAX
            mov     Q1_lsb,s_Cero
            jmp     RETORNA
RETORNA
 rts
```

8.7. II.2. MICROCONTROLADOR PIC 16F873.

Este formato se realizo en base al archivo de caracterización *fuzzypic*_+-16°_21.fc cuya documentación se muestra en el anexo III

- Variables utilizadas por la rutina fuzzye3t : Almacenadas en memoria *Ram* del microcontrolador:

```
error0_lsb
error0_msb
error_lsb
error_msb
varerror_lsb
varerror_msb
s_Max_lsb
s_Max_msb
s_Alta_lsb
s_Alta_msb
s_Meda_lsb
s_Meda_msb
s_Med_lsb
s_Med_msb
s_Medb_lsb
s_Medb_msb
s_Cero_lsb
s_Cero_msb
Q1_lsb
Q1_msb
Q2_lsb
Q2_msb
Q3_1
Q3_2
Q3_3
```

SDEN_1
SDEN_2
SDEN_3
MSNUM_1
MSNUM_2
MSNUM_3
INDIC@
N_TERM_0
N_TERM_1
DNUMDEN_1
DNUMDEN_2
REST_1
REST_2
REST_3
N_VERT
N_VERT_0
M@
RR
leftbottom_lsb
leftbottom_msb
rightbottom_lsb
rightbottom_msb
lefttop_lsb
lefttop_msb
righttop_lsb
righttop_msb

- Vértices utilizados por rutina fuzzye3t: Almacenados en memoria de programa

8.8.

```
VERTICES
        addwf   PCL,1
        retlw   0x00
        retlw   0x00
        retlw   0x00
        retlw   0x00
        retlw   0x00
        retlw   0x00
        retlw   0x40
        retlw   0x00
        retlw   0x00
        retlw   0x00
        retlw   0x40
        retlw   0x00
        retlw   0x00
        retlw   0x00
        retlw   0x40
        retlw   0x00
        retlw   0xC0
        retlw   0x00
        retlw   0x40
        retlw   0x00
```

```
retlw   0xC0
retlw   0x00
retlw   0xC0
retlw   0x00
retlw   0x80
retlw   0x01
retlw   0xC0
retlw   0x00
retlw   0x80
retlw   0x01
retlw   0x80
retlw   0x01
retlw   0xFF
retlw   0x03
retlw   0x80
retlw   0x01
retlw   0xFF
retlw   0x03
retlw   0xFF
retlw   0x03
retlw   0xFF
retlw   0x03
retlw   0x00
retlw   0x00
retlw   0x00
retlw   0x00
retlw   0x00
retlw   0x00
retlw   0x80
retlw   0x01
retlw   0x00
retlw   0x00
retlw   0x80
retlw   0x01
retlw   0x80
retlw   0x01
retlw   0x00
retlw   0x02
retlw   0x80
retlw   0x01
```

```
retlw   0x00
retlw   0x02
retlw   0x00
retlw   0x02
retlw   0x7F
retlw   0x02
retlw   0x00
retlw   0x02
retlw   0x7F
retlw   0x02
retlw   0x7F
retlw   0x02
retlw   0xFF
retlw   0x03
retlw   0x7F
retlw   0x02
retlw   0xFF
retlw   0x03
retlw   0xFF
retlw   0x03
retlw   0xFF
retlw   0x03
```

8.9.

- **Rutina principal fuzzye3t: Almacenada en memoria de programa**

8.10.

```
FUZZYE3T
          clrf      s_Max_lsb
          clrf    s_Max_msb
          clrf      s_Alta_lsb
          clrf    s_Alta_msb
          clrf      s_Meda_lsb
          clrf    s_Meda_msb
          clrf      s_Med_lsb
          clrf    s_Med_msb
          clrf      s_Medb_lsb
          clrf    s_Medb_msb
          clrf      s_Cero_lsb
          clrf    s_Cero_msb
     bcf     PCLATH,3
     bcf     PCLATH,4
;=========================
          clrf    N_TERM_0
          clrf    N_VERT
PERT_0
          movf     error_lsb,0
          movwf    Q1_lsb
          movf     error_msb,0
          movwf    Q1_msb
          call     MEMBR
          btfss    DNUMDEN_2,7
          goto     INCR_0
          movf     DNUMDEN_1,0
          movwf   error0_lsb
          movf     DNUMDEN_2,0
          andlw    b'00000011'
          movwf    error0_msb
          clrf    N_TERM_1
          movf     N_VERT,0
          movwf    N_VERT_0
          movlw    0x28       ;~~~~~~~~~~~~~~~~~~~
          movwf    N_VERT
```

```
PERT_1
        movf    varerror_lsb,0
        movwf   Q1_lsb
        movf    varerror_msb,0
        movwf   Q1_msb
        call    MEMBR
        btfss   DNUMDEN_2,7
        goto    INCR_1
        movf    DNUMDEN_1,0
        movwf   Q1_lsb
        movf    DNUMDEN_2,0
        andlw   b'00000011'
        movwf   Q1_msb
        movf    error0_lsb,0
        movwf   Q2_lsb
        movf    error0_msb,0
        movwf   Q2_msb
        call    REGLAS
INCR_1
        incf    N_TERM_1,1
        movlw    0x05
        subwf   N_TERM_1,0
        btfss   STATUS,0
        goto    PERT_1

        movf    N_VERT_0,0
        movwf   N_VERT
INCR_0
        incf    N_TERM_0,1
        movlw   0x05
        subwf   N_TERM_0,0
        btfss   STATUS,0
        goto    PERT_0
;======================
CENTROID
        clrf    MSNUM_1
        clrf    MSNUM_2
        clrf    MSNUM_3
        clrf    SDEN_1
        clrf    SDEN_2
```

```
        clrf    SDEN_3
        clrf    DNUMDEN_1
        clrf    DNUMDEN_2
c_Max
        movf    s_Max_lsb,0
        movwf   Q1_lsb
        movf    s_Max_msb,0
        movwf   Q1_msb
        iorwf   s_Max_lsb,0
        btfsc   STATUS,2
        goto    c_Alta
        call    SUM@I
        movlw   0x00
        movwf   Q2_lsb
        movlw   0x00
        movwf   Q2_msb
        call    MULSUM@S
c_Alta
        movf    s_Alta_lsb,0
        movwf   Q1_lsb
        movf    s_Alta_msb,0
        movwf   Q1_msb
        iorwf   s_Alta_lsb,0
        btfsc   STATUS,2
        goto    c_Meda
        call    SUM@I
        movlw   0x1C
        movwf   Q2_lsb
        movlw   0x01
        movwf   Q2_msb
        call    MULSUM@S
c_Meda
        movf    s_Meda_lsb,0
        movwf   Q1_lsb
        movf    s_Meda_msb,0
        movwf   Q1_msb
        iorwf   s_Meda_lsb,0
        btfsc   STATUS,2
        goto    c_Med
        call    SUM@I
```

```
        movlw   0x77
        movwf   Q2_lsb
        movlw   0x01
        movwf   Q2_msb
        call    MULSUM@S
c_Med
        movf    s_Med_lsb,0
        movwf   Q1_lsb
        movf    s_Med_msb,0
        movwf   Q1_msb
        iorwf   s_Med_lsb,0
        btfsc   STATUS,2
        goto    c_Medb
        call    SUM@I
        movlw   0x00
        movwf   Q2_lsb
        movlw   0x02
        movwf   Q2_msb
        call    MULSUM@S
c_Medb
        movf    s_Medb_lsb,0
        movwf   Q1_lsb
        movf    s_Medb_msb,0
        movwf   Q1_msb
        iorwf   s_Medb_lsb,0
        btfsc   STATUS,2
        goto    c_Cero
        call    SUM@I
        movlw   0x38
        movwf   Q2_lsb
        movlw   0x02
        movwf   Q2_msb
        call    MULSUM@S
c_Cero
        movf    s_Cero_lsb,0
        movwf   Q1_lsb
        movf    s_Cero_msb,0
        movwf   Q1_msb
        iorwf   s_Cero_lsb,0
        btfsc   STATUS,2
```

```
        goto    c_Salida?
        call    SUM@I
        movlw    0x54
        movwf   Q2_lsb
        movlw    0x03
        movwf   Q2_msb
        call    MULSUM@S
c_Salida?
        call  DIV@
;<<EL DATO DE DNUMDEN_1 Y DNUMDEN_2>>>SALIDA DE 10BITS DE ESTA RUTINA SE ALMACENA EN LAS VARIABLES> ;<<
```

8.11.

8.12. SUBRUTINAS UTILIZADAS POR LA RUTINA FUZZYE3T: ALMACENADAS EN MEMORIA DE PROGRAMA.

8.13.

```
;***********************************************
REGLAS
        call    MINMAX
        movf    N_TERM_0,0
        addwf   N_TERM_0,0
        addwf   N_TERM_0,0
        addwf   N_TERM_0,0
        addwf   N_TERM_0,0
        addwf   N_TERM_1,0
        addwf   PCL,1
        goto    Medb
        goto    Cero
        goto    Cero
        goto    Cero
        goto    Cero
        goto    Alta
        goto    Medb
        goto    Medb
        goto    Cero
        goto    Cero
        goto    Max
        goto    Alta
        goto    Medb
        goto    Medb
```

```
        goto    Medb
        goto    Max
        goto    Max
        goto    Med
        goto    Med
        goto    Med
        goto    Max
        goto    Max
        goto    Max
        goto    Max
        goto    Alta
Max
        movf    s_Max_lsb,0
        movwf   Q1_lsb
        movf    s_Max_msb,0
        movwf   Q1_msb
        call    MINMAX
        movf    Q1_lsb,0
        movwf   s_Max_lsb
        movf    Q1_msb,0
        movwf   s_Max_msb
        return
Alta
        movf    s_Alta_lsb,0
        movwf   Q1_lsb
        movf    s_Alta_msb,0
        movwf   Q1_msb
        call    MINMAX
        movf    Q1_lsb,0
        movwf   s_Alta_lsb
        movf    Q1_msb,0
        movwf   s_Alta_msb
        return
Meda
        movf    s_Meda_lsb,0
        movwf   Q1_lsb
        movf    s_Meda_msb,0
        movwf   Q1_msb
        call    MINMAX
        movf    Q1_lsb,0
```

```
        movwf   s_Meda_lsb
        movf    Q1_msb,0
        movwf   s_Meda_msb
        return
Med
        movf    s_Med_lsb,0
        movwf   Q1_lsb
        movf    s_Med_msb,0
        movwf   Q1_msb
        call    MINMAX
        movf    Q1_lsb,0
        movwf   s_Med_lsb
        movf    Q1_msb,0
        movwf   s_Med_msb
        return
Medb
        movf    s_Medb_lsb,0
        movwf   Q1_lsb
        movf    s_Medb_msb,0
        movwf   Q1_msb
        call    MINMAX
        movf    Q1_lsb,0
        movwf   s_Medb_lsb
        movf    Q1_msb,0
        movwf   s_Medb_msb
        return
Cero
        movf    s_Cero_lsb,0
        movwf   Q1_lsb
        movf    s_Cero_msb,0
        movwf   Q1_msb
        call    MINMAX
        movf    Q1_lsb,0
        movwf   s_Cero_lsb
        movf    Q1_msb,0
        movwf   s_Cero_msb
        return
;*********************************************
MEMBR
        movf    N_VERT,0
```

```
        call    VERTICES
        movwf   leftbottom_lsb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   leftbottom_msb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   lefttop_lsb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   lefttop_msb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   righttop_lsb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   righttop_msb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   rightbottom_lsb
        incf    N_VERT,1
        movf    N_VERT,0
        call    VERTICES
        movwf   rightbottom_msb
        incf    N_VERT,1
IF_0
        movf    leftbottom_lsb,0
        movwf   Q2_lsb
        movf    leftbottom_msb,0
        movwf   Q2_msb
        call    COMP@
        xorlw   0x02
        btfss   STATUS,2
        goto    IF_1
```

```
        bcf     DNUMDEN_2,7
        return
IF_1
        movf    rightbottom_lsb,0
        movwf   Q2_lsb
        movf    rightbottom_msb,0
        movwf   Q2_msb
        call    COMP@
        xorlw   0x01
        btfss   STATUS,2
        goto    IF_2
        bcf     DNUMDEN_2,7
        return
IF_2
        movf    lefttop_lsb,0
        movwf   Q2_lsb
        movf    lefttop_msb,0
        movwf   Q2_msb
        call    COMP@
        xorlw   0x02
        btfss   STATUS,2
        goto    IF_3
RECT_A
        movf    leftbottom_lsb,0
        movwf   Q2_lsb
        movf    leftbottom_msb,0
        movwf   Q2_msb
        call    REST@
        movlw    0x03
        movwf   Q2_msb
        movlw    0xFF
        movwf   Q2_lsb
        call    M_MSNUM
        movf    lefttop_lsb,0
        movwf   Q1_lsb
        movf    lefttop_msb,0
        movwf   Q1_msb
        movf    leftbottom_lsb,0
        movwf   Q2_lsb
        movf    leftbottom_msb,0
```

```
        movwf   Q2_msb
        call    REST@
        movf    Q1_lsb,0
        movwf   SDEN_1
        movf    Q1_msb,0
        movwf   SDEN_2
        call    DIV@
        bsf     DNUMDEN_2,7
        return
IF_3
        movf    righttop_lsb,0
        movwf   Q2_lsb
        movf    righttop_msb,0
        movwf   Q2_msb
        call    COMP@
        xorlw   0x01
        btfss   STATUS,2
        goto    CTE
RECT_D
        movf    Q1_lsb,0
        movwf   Q2_lsb
        movf    Q1_msb,0
        movwf   Q2_msb
        movf    rightbottom_lsb,0
        movwf   Q1_lsb
        movf    rightbottom_msb,0
        movwf   Q1_msb
        call    REST@
        movlw    0x03
        movwf   Q2_msb
        movlw    0xFF
        movwf   Q2_lsb
        call    M_MSNUM
        movf    rightbottom_lsb,0
        movwf   Q1_lsb
        movf    rightbottom_msb,0
        movwf   Q1_msb
        movf    righttop_lsb,0
        movwf   Q2_lsb
        movf    righttop_msb,0
```

```
        movwf   Q2_msb
        call    REST@
        movf    Q1_lsb,0
        movwf   SDEN_1
        movf    Q1_msb,0
        movwf   SDEN_2
        call    DIV@
        bsf     DNUMDEN_2,7
        return
CTE
        movlw    0x03
        movwf   DNUMDEN_2
        movlw    0xFF
        movwf   DNUMDEN_1
        bsf     DNUMDEN_2,7
        return
;***********************************************
COMP@   movf    Q2_msb,0
        subwf   Q1_msb,0
        btfss   STATUS,0
        goto    Q1MENOR
        btfss   STATUS,2
        goto    Q1MAYOR
        movf    Q2_lsb,0
        subwf   Q1_lsb,0
        btfss   STATUS,0
        goto    Q1MENOR
        btfss   STATUS,2
        goto    Q1MAYOR
        goto    Q1IGUAL
Q1MENOR  retlw   0x02
Q1MAYOR  retlw   0x01
Q1IGUAL  retlw   0x00
;***********************************************
MINMAX
        movf    Q2_msb,0
        subwf   Q1_msb,0
        btfss   STATUS,0
        goto    INTERC
        btfss   STATUS,2
```

```
        goto    NO_INTERC
        movf    Q2_lsb,0
        subwf   Q1_lsb,0
        btfsc   STATUS,0
        goto    NO_INTERC
INTERC
        movf    Q1_msb,0
        movwf   Q3_3
        movf    Q2_msb,0
        movwf   Q1_msb
        movf    Q3_3,0
        movwf   Q2_msb
        movf    Q1_lsb,0
        movwf   Q3_3
        movf    Q2_lsb,0
        movwf   Q1_lsb
        movf    Q3_3,0
        movwf   Q2_lsb
NO_INTERC
        Return
;***********************************************
REST@
        comf    Q2_msb,1
        comf    Q2_lsb,1
        incf    Q2_lsb,1
        btfsc   STATUS,2
        incf    Q2_msb,1
        movf    Q2_lsb,0
        addwf   Q1_lsb,1
        btfsc   STATUS,0
        incf    Q1_msb,1
        movf    Q2_msb,0
        addwf   Q1_msb,1
        return
;***********************************************
SUM@I
        movf    Q1_lsb,0
        addwf   SDEN_1,1
        btfsc   STATUS,0
        incf    SDEN_2,1
```

```
        movf    Q1_msb,0
        addwf   SDEN_2,1
        return
;***********************************************
M_MSNUM
        clrf    MSNUM_3
        clrf    MSNUM_2
        clrf    MSNUM_1
        movlw   .16
        movwf   INDIC@
MLAZO_A
        bcf     STATUS,0
        rlf     MSNUM_1,1
        rlf     MSNUM_2,1
        rlf     MSNUM_3,1
        bcf     STATUS,0
        rlf     Q2_lsb,1
        rlf     Q2_msb,1
        btfss   STATUS,0
        goto    NSUM@_A
        movf    Q1_lsb,0
        addwf   MSNUM_1,1
        btfsc   STATUS,0
        incf    MSNUM_2,1
        movf    Q1_msb,0
        addwf   MSNUM_2,1
        btfsc   STATUS,0
        incf    MSNUM_3,1
NSUM@_A
        decfsz  INDIC@,1
        goto    MLAZO_A
        return
;***********************************************
MULSUM@S
        clrf    Q3_3
        clrf    Q3_2
        clrf    Q3_1
        movlw   .16
        movwf   INDIC@
MLAZO
```

```
        bcf     STATUS,0
        rlf     Q3_1,1
        rlf     Q3_2,1
        rlf     Q3_3,1
        bcf     STATUS,0
        rlf     Q2_lsb,1
        rlf     Q2_msb,1
        btfss   STATUS,0
        goto    NSUM@
        movf    Q1_lsb,0
        addwf   Q3_1,1
        btfsc   STATUS,0
        incf    Q3_2,1
        movf    Q1_msb,0
        addwf   Q3_2,1
        btfsc   STATUS,0
        incf    Q3_3,1
NSUM@
        decfsz  INDIC@,1
        goto    MLAZO
        movf    Q3_1,0
        addwf   MSNUM_1,1
        btfsc   STATUS,0
        incf    MSNUM_2,1
        movf    Q3_2,0
        addwf   MSNUM_2,1
        btfsc   STATUS,0
        incf    MSNUM_3,1
        movf    Q3_3,0
        addwf   MSNUM_3,1
        return
;*********************************************
DIV@
        movlw   .24
        movwf   INDIC@
        clrf    DNUMDEN_2
        clrf    DNUMDEN_1
        clrf    REST_3
        clrf    REST_2
        clrf    REST_1
```

```
        clrf    SDEN_3
DLAZO
        bcf     STATUS,C
        rlf     MSNUM_1,1
        rlf     MSNUM_2,1
        rlf     MSNUM_3,1
        rlf     REST_1,1
        rlf     REST_2,1
        rlf     REST_3,1
        movf    SDEN_3,0
        subwf   REST_3,0
        btfss   STATUS,2
        goto    PAS@1
        movf    SDEN_2,0
        subwf   REST_2,0
        btfss   STATUS,2
        goto    PAS@1
        movf    SDEN_1,0
        subwf   REST_1,0
PAS@1
        btfss   STATUS,0
        goto    PAS@2
        movf    SDEN_1,0
        subwf   REST_1,1
        btfss   STATUS,0
        decf    REST_2,1
        movf    SDEN_2,0
        subwf   REST_2,1
        btfss   STATUS,0
        decf    REST_3,1
        movf    SDEN_3,0
        subwf   REST_3,1
        bsf     STATUS,0
PAS@2
        rlf     DNUMDEN_1,1
        rlf     DNUMDEN_2,1
        decfsz  INDIC@,1
        goto    DLAZO
```

8.14. RETURN

8.15. ;*

ANEXO III

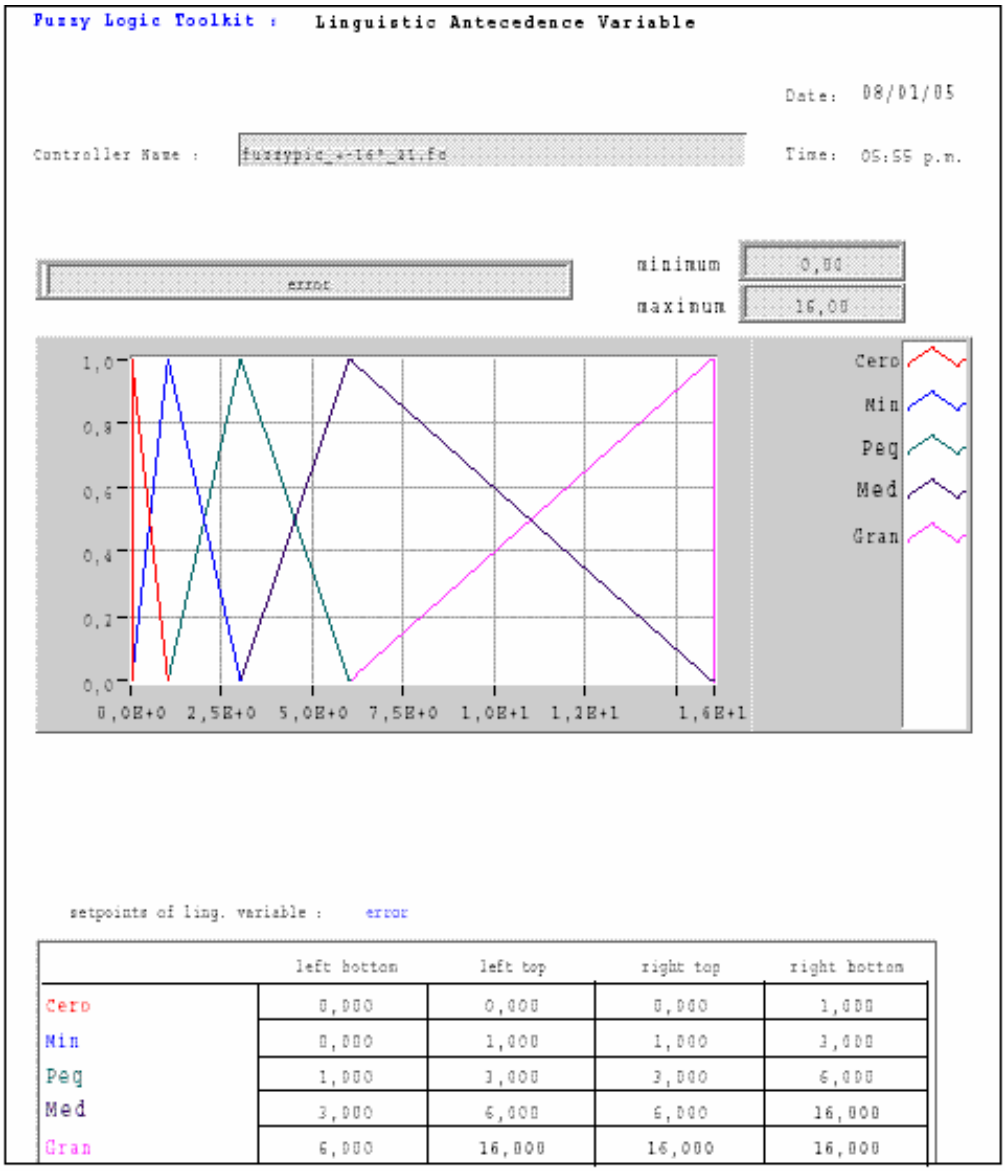
DOCUMENTACION ARCHIVO FC UTILIZADO EN LA APLICACIÓN.

La caracterización del sistema que se uso en la aplicación se guardo en el archivo fuzzypic_+-16°_21.fc, se muestra a continuación las variables de entrada, la variable de salida y las reglas.

8.16. III.1. VARIABLES LINGÜÍSTICAS DE ENTRADA

También llamadas antecedentes son Error y Varerror

- Error



	left bottom	left top	right top	right bottom
Cero	0,000	0,000	0,000	1,000
Min	0,000	1,000	1,000	3,000
Peq	1,000	3,000	3,000	6,000
Med	3,000	6,000	6,000	16,000
Gran	6,000	16,000	16,000	16,000

- Varerror

Fuzzy Logic Toolkit : Linguistic Antecedence Variable

Date: 08/01/05

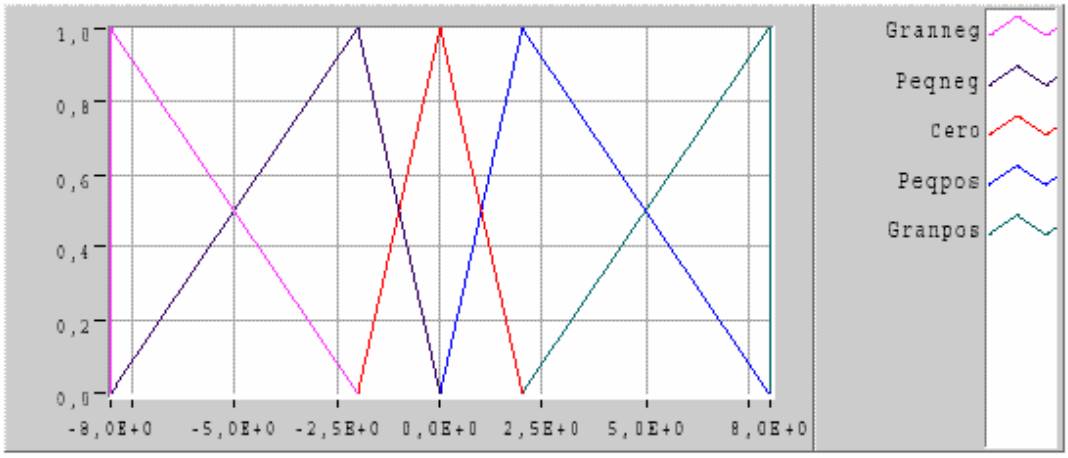
Controller Name : fuzzypic_+-16°_21.fc

Time: 06:28 p.m.

varerror

minimum -8,00

maximum 8,00



setpoints of ling. variable : varerror

	left bottom	left top	right top	right bottom
Granneg	-8,000	-8,000	-8,000	-2,000
Peqneg	-8,000	-2,000	-2,000	0,000
Cero	-2,000	0,000	0,000	2,000
Peqpos	0,000	2,000	2,000	8,000
Granpos	2,000	8,000	8,000	8,000

III .2. VARIABLE LINGÜÍSTICA DE SALIDA

Tambien llamada consecuencias a esta variable se le dio el nombre de Pot_angulo

Fuzzy-Logic Toolkit : Linguistic Consequence Variable

Date: 08/01/05

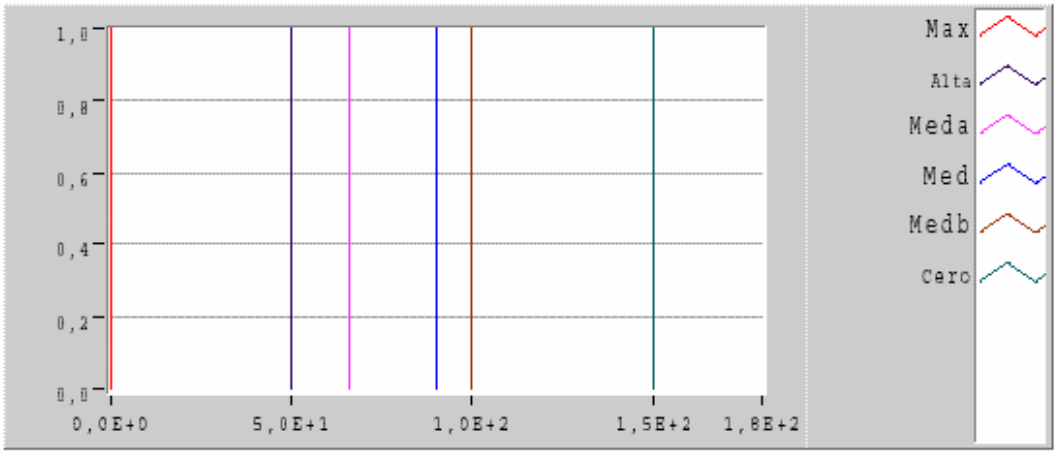
Controller Name : fuzzypic_+-16°_21.fc

Time: 05:54 p.m.

Potang

minimum 0,00

maximum 180,00



Inferencemethod : Max-Min

Defuzzifymethod : Center of Maximum

Modify : yes

Consequence if no rule is firering : last value

setpoints of ling. variable : Potang

	left bottom	left top	right top	right bottom
Max	0,000	0,000	0,000	0,000
Alta	50,000	50,000	50,000	50,000
Meda	66,000	66,000	66,000	66,000
Med	90,000	90,000	90,000	90,000
Medb	100,000	100,000	100,000	100,000
Cero	150,000	150,000	150,000	150,000

8.17. III .3. CONJUNTO DE REGLAS UTILIZADAS

Fuzzy Logic Toolkit : **Rules**

Date: 05:56 p.m.

Controller Name : fuzzypic_+-16^_21.fc

Time: 08/01/05

Rule No.	IF error	AND vararror	THEN Potang	DoS
1	Cero	Granneg	Medb	1,00
2	Cero	Peqneg	Cero	1,00
3	Cero	Cero	Cero	1,00
4	Cero	Peqpos	Cero	1,00
5	Cero	Granpos	Cero	1,00
6	Min	Granneg	Alta	1,00
7	Min	Peqneg	Medb	1,00
8	Min	Cero	Medb	1,00
9	Min	Peqpos	Cero	1,00
10	Min	Granpos	Cero	1,00
11	Peq	Granneg	Max	1,00
12	Peq	Peqneg	Alta	1,00
13	Peq	Cero	Medb	1,00
14	Peq	Peqpos	Medb	1,00
15	Peq	Granpos	Medb	1,00
16	Med	Granneg	Max	1,00
17	Med	Peqneg	Max	1,00
18	Med	Cero	Med	1,00
19	Med	Peqpos	Med	1,00
20	Med	Granpos	Med	1,00
21	Gran	Granneg	Max	1,00
22	Gran	Peqneg	Max	1,00
23	Gran	Cero	Max	1,00
24	Gran	Peqpos	Max	1,00
25	Gran	Granpos	Alta	1,00

ANEXO IV

PROGRAMA EN LENGUAJE ENSAMBLADOR PARA LOS MICROCONTROLADORES (16F873, MCHC908JL3) UTILIZADOS EN LA APLICACIÓN.

8.18. IV.1. MICROCONTROLADOR MOTOROLA MCHC908JL3.

Este programa realiza control Fuzzy de la aplicación utilizando solo el microcontrolador. Las rutinas utilizadas por Fuzzye3t se encuentran en el anexo II.

```
*******************************************************************************
*Programa Control Fuzzy Desde UC*
*Manejo De Teclado Y Lcd Fuzzy, Microcontrolador Motorola Hc08jl3- Xtal 5 Mhz*
*Muestra Temperatura  Y Setpoint
*Entra Tres Datos Y Acepta Con La Tecla Asterisco Del Teclado
* FC fuzzypic_+-16°_21.fc
*******************************************************************************
RAM     EQU  $0080
Rom     EQU  $EC00   ; Valida para JL3, JK3, JK1
Vector  EQU  $FFDE
*******************************************************************************
$Include 'C:\pemicro\ics08jlz\jl3regs.inc'
*******************************************************************************
   org Ram
cent            ds 1
dec             ds 1
uni             ds 1
caracter        ds 1
cuenta_bit      ds 1
columna         ds 1
fila            ds 1
PA              ds 1
R0C             ds 1
R0D             ds 1
R0E             ds 1
TEMP            ds 1
C               ds 1
E               EQU  $5
RS              EQU  $4
```

```
BUFFER   ds 1
yes      ds 1
vram     ds 1
bin_l    ds 1
bin_h    ds 1
*******************************************************************************
t        ds 1
*******************************************************************************
H_Byte   ds 1      ;16-bit valor de entrada Binario>BCD
L_Byte   ds 1
R1       ds 1
R2       ds 1   ;BCD salida Binario->BCD
R0       ds 1
count    ds 1
temp1    ds 1
temp2    ds 1
BCDX1    ds 1
BCDX10   ds 1
BCDX100  ds 1
NUMERADOR ds 1
TECLA    ds 1
PWMH     ds 1
PWML     ds 1
error_msb ds 1
*******************************************************************************
Insertar aquí las variables utilizadas por fuzzye3t para microcontrolador Motorola que se encuentran en el anexo II
*******************************************************************************
  org Rom
VÉRTICES
;*********************************************************
;******INSERTAR  AQUÍ  LA TABLA DE VÉRTICES***********
;*********************************************************
main:
        rsp
        bset 0,CONFIG1   ;deshabilita modulo cop (watchdog)
        bset 1,INTSCR    ;DESABILITA INTERRUPCION PIN IRQ
        jsr  PUERTOS_init
        JSR  INICIA_VARIAB
        jsr  LCD_INIT
        jsr  MENSAJE1
```

```
            jsr teclado_init
            cli             ;ACTIVA ATENCION A INTERRUPCIONES  (teclado)
main_2        BRSET 0,yes,sig1
            jmp  main_2
sig1          JSR ASCC-BCD
            JSR BCD-BIN
       MOV   bin_l,TECLA
**************************
*MUESTRA MENSAJE TEMPERATURA
**************************
          ldx #$01
          BCLR 1,C
          jsr DALTO
          MOV #$4,TEMP
          jsr SIGLO
          ldx #%10000000 ;POSICION 0
          BCLR 1,C
          jsr DALTO
          clrx
          mov #$0,R0C
CICLO     CLRH
          ldX R0C
          LDX TABLA_LCD,X
          BSET 1,C
          jsr DALTO
          mov #$FF,R0D
RETA      jsr RETARDO
          dbnz R0D,RETA
          inc R0C
          lda R0C
          SUB #$0F
          BCS CICLO
         jsr init_pwm
          ldx #$01
          BCLR 1,C
          jsr DALTO
          MOV #$4,TEMP
          jsr SIGLO
loop      jsr  AD_init    ; conversión sencilla
         CLI             ;limpiar bits 3 del CCR habilita la atención de interrupción
```

```
main_3    BRCLR 0,yes,sig2 ;
        bra  main_3
sig2      jsr temp_actual
        jmp loop
PUERTOS_init
           mov  #$30,DDRA         ;puerto A0-A3 entrada columnas del teclado
                                  ;puerto A4-A5 SALIDAS LCD   (enable, RS)
           clr  porta
           mov  #$7F,ddrd         ;puerto D salida LDC y filas del teclado Y PWM
           clr  PORTD             ;puerto D Lsb en cero
           mov  #$FF,DDRB
           CLR  PORTB
         MOV #$FF,TEMP ;***********RETARDOS PARA***********
         jsr SIGLO ;**************INICIALIZACION************
         MOV #$FF,TEMP ;*************LCD*****************
         jsr SIGLO
         MOV #$FF,TEMP
         jsr SIGLO
         MOV #$FF,TEMP
         jsr SIGLO
         MOV #$FF,TEMP
         jsr SIGLO
         MOV #$FF,TEMP
         jsr SIGLO
         MOV #$FF,TEMP
         jsr SIGLO
         MOV #$FF,TEMP
         jsr SIGLO
           rts
*******************************************************************************
*RUTINA DE INICIALIZACION DE TECLADO*
*******************************************************************************
Teclado_init   bset $1,KBSCR          ;imaskk =1
           mov  #$0F,KBIER        ;habilita cada una columnas pta0-pta3
           bset $2,KBSCR          ;ACKK=1 limpia cualquier falsa interrupción
           mov #%00000001,KBSCR          ;imaskk =0
          rts
*******************************************************************************
*RUTINA DE INICIALIZACION DE  PWM*
*******************************************************************************
```

```
init_pwm      bset 5,tsc
              bset 4,tsc
              mov #$28,tmodh  ; periodo28
              mov #$00,tmodl  ; periodoB0
              mov #$24,tch0h  ; ancho de pulso14
              mov #$00,tch0l  ; ancho de pulso58
              mov #$1E,tsc0   ;%00011110  ;PWM SINBUFER (ON),NIVEL INICIAL BAJO
            rts
*******************************************************************************
*RUTINA DE INICIALIZACION DE VARIABLES
*******************************************************************************
INICIA_VARIAB
              clr C         ;VAR PARA LCD C=0 REGISTRO CONTROL C=1 REGISTRO DATOS
              clr yes       ;VAR PAR TECLADO
              clr BUFFER    ;VAR PAR TECLADO
              CLR uni       ;VAR PARA BCD_BIN
              CLR dec       ;VAR PARA BCD_BIN
              CLR cent      ;VAR PARA BCD_BIN
              mov #$80,vram  ;VAR PAR TECLADO
              CLR s_max
              CLR s_alta
              CLR s_meda
              CLR s_med
              CLR s_medb
              CLR s_cero
              CLR SDEN_1
              CLR MSNUM_1
              CLR MSNUM_2
              CLR error_lsb
              CLR varerror_lsb
              CLR error2
              CLR error0_lsb
              CLR Q1_lsb
              CLR Q2_lsb
              CLR Q3_1
              CLR N_TERM_0
              CLR N_TERM_1
              CLR N_FUNC
              CLR DNUMDEN_1
              CLR DNUMDEN_2
```

```
          CLR N_VERT
          CLR N_VERT_0
          CLR leftbottom_lsb
          CLR rightbottom_lsb
          CLR lefttop_lsb
          CLR righttop_lsb
          CLR TECLA
  RTS
*******************************************************************************
*RUTINA DE INICIALIZACION DE LCD*
*******************************************************************************
LCD_INIT   ldx #%00000010            ;LIMPIA  HOME
        BCLR 1,C
        jsr DALTO
        MOV #$4,TEMP
        jsr SIGLO
        ldx #%00101000    ;BUS DE DATOS (4 BITS) (2 LINEA EN PANTALLA) TAMAÑO CARACTER 5*7
        BCLR 1,C
        jsr DALTO
        ldx #%00001110            ; PANTALLA ACTIVA Y CURSOR NO PARPADEA
        BCLR 1,C
        jsr DALTO
        MOV  #$40,PORTD           ;puerto D Lsb en cero
          rts
**************************
*MUESTRA MENSAJE DE (CONTROL FUZZY)*
**************************
MENSAJE1   ldx #$01
        BCLR 1,C
        jsr DALTO
        MOV #$4,TEMP
        jsr SIGLO
        ldx #%10000010 ;POSICION 2
        BCLR 1,C
        jsr DALTO
        clrx
        mov #$0,R0C
CICLO1     CLRH
        ldx R0C
        LDX TABLA_LCD1,X
```

```
        BSET 1,C
        jsr DALTO
        mov #$FF,R0D
RETA1   jsr RETARDO
        dbnz R0D,RETA1
        inc R0C
        lda R0C
        SUB #$0F
        BCS CICLO1
        ldx #%11000110 ;POSICION $46
        BCLR 1,C
        jsr DALTO
        RTS
*****************************************************************
TABLA_LCD1
   DB "C"
   DB "O"
   DB "N"
   DB "T"
   DB "R"
   DB "O"
   DB "L"
   DB " "
   DB "F"
   DB "U"
   DB "Z"
   DB "Z"
   DB "Y"
   DB " "
   DB " "
   DB "U"
   DB "C"
*****************************************************************
TABLA_LCD
   DB " "
   DB " "
   DB "T"
   DB "E"
   DB "M"
   DB "P"
```

```
    DB "E"
    DB "R"
    DB "A"
    DB "T"
    DB "U"
    DB "R"
    DB "A"
    DB " "
    DB " "
    DB " "
*******************************************************************************
AD_init         mov #$47,adscr ;$47=(1000111) habilita la interrupción, conversión UNICA conversión on canal
ADC7=PTB7
            mov #$80,ADCLK ; velocidad del reloj para la conversión 16CICLOS*0.8uS*16=204.8uS => 1 conversión
cada 4uS
            BSET 0,yes
            rts
*******************************************************************************
*RUTINA DE ATENCION A INTERRUPCION AD/LCD*
*******************************************************************************
INTER_CONVERSION:
            PSHA
            PSHH
            PSHX
            MOV adr,CARACTER    ;GUARDA EN VARIABLE EL DATO CONVERTIDO
            clrh
            LDX CARACTER          ;CONVIERTE DATO  FORMATO 0-150 (BIN_L*150/255)
            LDA #$96
            MUL
            PSHX
            PULH
            LDX #$FF
            DIV
            STA CARACTER
            lda TECLA
            CMP CARACTER
            BHS TSMAYOR        ; SALTA SI TECLA >= QUE A/D
            lda CARACTER
            SUB TECLA
            sta error_lsb
```

```
        sta error_msb
        JMP NOFUZZ
TSMAYOR
        lda TECLA
        SUB CARACTER
        STA error_lsb
        CMP #$0F
        BGE PRENDE
        LDX #$10
        MUL
        CBEQX #$00,WE
        JMP  PRENDE
WE          sta error_lsb
        sta error_msb
*****************************************************
VAERR
;*** VARERROR=(TEMP ACTUAL)-(TEMP ANTERIOR)*****
        LDA CARACTER
        CMP error2
        BHS VARMAYOR        ; SALTA SI A/D(ACTUAL) >= QUE A/D(ANTERIOR)
        lda error2
        SUB CARACTER
        ADD #$8
        LDX #$10
        MUL
        sta varerror_lsb
        JMP FUZZ
VARMAYOR      lda CARACTER
        SUB error2
        ADD #$8
        LDX #$10
        MUL
        sta varerror_lsb
FUZZ        MOV CARACTER,error2
        LDA error_lsb
       ; CBEQA #$00,NOFUZZ
        jsr FUZZYE3T                    ; SALTA SUBRRUTINA FUZZY
        JMP TERMINAR
NOFUZZ       bset 5,tsc
        bset 4,tsc
```

```
        mov  #$1F,tsc0   ;%00011111  ;PWM SINBUFER (ON),NIVEL INICIAL BAJO, 100% CICLO UTIL
        bclr 5,tsc      ; activa contador y comienza Pwm
        JMP  TERMINAR
PRENDE      mov  #$00,PWMH   ;ANCHO DE PULSO DE LA SEÑAL PWM (MSB)
        mov  #$00,PWML   ;ANCHO DE PULSO DE LA SEÑAL PWM (LSB)
        mov #$1E,tsc0   ;%00011111  ;PWM SINBUFER (ON),NIVEL INICIAL BAJO, 100% CICLO UTIL
TERMINAR    PULX
        PULH
        PULA
        BCLR 1,INTSCR ;HABILITA INTERRUPCION PIN IRQ
        cli
  rti       ; retorna
*****************************************************************************
*RUTINA DE ATENCION A INTERRUPCION externa (cruce por cero)*
*****************************************************************************
Pwm         PSHA
        PSHH
        PSHX
        bset 5,tsc     ;DESACTIVA contador TIM
        bset 4,tsc     ;reset al contador TIM
        LDA  PWMH
        STA  tch0h
        LDA  PWML
        STA  tch0l
        bclr 5,tsc          ; activa contador y comienza Pwm
        bclr 0,YES
        PULX
        PULH
        PULA
       cli
  rti           ; retorna
*****************************************************************************
FUZZYE3T
;    INSERTAR  SUBRUTINA PRINCIPAL FUZZYE3T  (anexo II)
      sta    DNUMDEN_1   ;(0-255)
      cbeqa   #!255,APAGA
      ldx    DNUMDEN_1
*SE CONVIERTE EL DATO DE SALIDA DE FORMATO (0-255)*
* UTILZADO EN LOS CALCULOS A FORMATO (0-10200) UTILZADO EN EL PWM*************
     CLRH
```

```
      LDA   #!40
      MUL
      STX PWMH   ;ANCHO DE PULSO DE LA SEÑAL PWM (MSB)
      STA PWML   ;ANCHO DE PULSO DE LA SEÑAL PWM (LSB)
      mov #$1E,tsc0   ;%00011111  ;PWM SINBUFER (ON),NIVEL INICIAL BAJO, 100% CICLO UTIL
      JMP SSS
APAGA  bset 5,tsc
      bset 4,tsc
      mov #$1F,tsc0   ;%00011111  ;PWM SINBUFER (ON),NIVEL INICIAL BAJO, 100 % CICLO UTIL
      bclr 5,tsc      ; activa contador y comienza Pwm
SSS   rts
;*******************************************************
;              SUMA Q1=Q1+Q2 de 16 bits   (anexo II)
SUMA16                    ;Q1+Q2
       add    MSNUM_1
       sta    MSNUM_1
       bcc    SALTO
       inc    MSNUM_2
SALTO    txa
       add    MSNUM_2
       sta    MSNUM_2
  rts
;*******************************************************
;     INSERTAR  SUBRUTINA DE GRADO DE MEMBRESÍA  (anexo II)
;*************************************
;     INSERTAR  SUBRUTINA DE MINIMO MAXIMO (retorna el  max en Q1, el min en Q2)  (anexo II)
;     INSERTAR  SUBRUTINA DE REGLAS    (anexo II)
*******************************************************
;subrutinas varias
DALTO     stx R0E
        JSR ENMASCAR_PTO
        lda #$F0
        and R0E
        nsa
        eor PTD
        sta PTD; ENVIA PARTE ALTA DEL PRIMER DATO
        JSR HAB_ERS
DBAJO     JSR ENMASCAR_PTO
        lda #$0F
        and R0E
```

```
        eor PTD
        sta PTD  ;ENVIA PARTE BAJA DEL PRIMER DATO
        JSR HAB_ERS
  RTS
ENMASCAR_PTO
        lda #$F0
        and PTD
        sta PTD ; ENMASCARA LOS BITS MAS ALTOS DEL PTD
  RTS
**********************
HAB_ERS    BRSET 1,C,DATO
        bclr RS,PTA
        JMP SIGUE
DATO       bset RS,PTA
SIGUE      JSR RETARDO
        bset E,PTA
        JSR RETARDO
        bclr E,PTA
   RTS
*********************************
SIGLO      ldx #$FF
        dbnzx *
        DEC TEMP
        BHI SIGLO
        RTS
****************
RETARDO    ldx #$FF
        dbnzx *
        rts
*****************
temp_actual
            MOV adr,CARACTER    ;GUARDA EN VARIABLE EL DATO CONVERTIDO
              LDA CARACTER
              clrh
              ldx #!17      ;convierte el dato diviendole en 1.7 => 255/1.7= 150ºC
              div           ;es el máximo valor medido por el sensor
              ldx #!10
              mul
              STA L_Byte
              LDX H_Byte
```

```
    PSHH
    PULX
    LDA #!10
    MUL
    PSHX
    PULH
    LDX #!17
    DIV
    add L_Byte
    sta L_Byte
    PSHH
    PULX
    LDA #!100
    MUL
    PSHX
    PULH
    LDX #!17
    DIV
    STA NUMERADOR
    jsr B2_BCD
    jsr BCDASC
ldx #%10000110 ;POSICION $02
BCLR 1,C
jsr DALTO
LDX BCDX100 ;MUESTRA CENTENAS
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX BCDX10  ;MUESTRA DECENAS
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX BCDX1       ;MUESTRA UNIDADES
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX #$2E  ;.
```

```
BSET 1,C
jsr DALTO
MOV #$2,TEMP
jsr SIGLO
mov NUMERADOR,L_Byte
clr H_Byte
jsr B2_BCD
jsr BCDASC
LDX BCDX10        ;MUESTRA DECIMAS
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX #$DD  ;º
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX #$43  ;C
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
     CLR H_Byte
   clrh
   LDA TECLA
    STA L_Byte
    jsr B2_BCD
    jsr BCDASC
 ldx #%11000000 ;POSICION $40
BCLR 1,C
jsr DALTO
LDX BCDX100 ;MUESTRA CENTENAS
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX BCDX10  ;MUESTRA DECENAS
BSET 1,C
jsr DALTO
```

```
MOV #$4,TEMP
jsr SIGLO
LDX BCDX1  ;MUESTRA UNIDADES
BSET 1,C
jsr DALTO
MOV #$FF,TEMP
jsr SIGLO
    CLR H_Byte
    LDA error_msb  ;varerror_lsb
    STA L_Byte
    jsr B2_BCD
    jsr BCDASC
 ldx #%11000111 ;POSICION $48
BCLR 1,C
jsr DALTO
LDX BCDX100 ;MUESTRA CENTENAS
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX BCDX10  ;MUESTRA DECENAS
BSET 1,C
jsr DALTO
MOV #$4,TEMP
jsr SIGLO
LDX BCDX1  ;MUESTRA UNIDADES
BSET 1,C
jsr DALTO
MOV #$FF,TEMP
jsr SIGLO
LDX #$25  ;%
BSET 1,C
jsr DALTO
MOV #$FF,TEMP
jsr SIGLO
LDX #$45  ;E
BSET 1,C
jsr DALTO
MOV #$FF,TEMP
jsr SIGLO
```

```
        MOV #$ff,TEMP ;***********RETARDOS PARA***********
        Jsr SIGLO ;***************ESTABILIZACION DE TECLA************
        MOV #$ff,TEMP ;*******************************
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP ;***********RETARDOS PARA***********
        Jsr SIGLO ;***************ESTABILIZACION DE TECLA************
        MOV #$ff,TEMP ;*******************************
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
 rts
*******************************************************************************
;--Binario A BCD
;  Entrada: dosbytes de RAM; L_Byte, H_Byte
;  Salida: tres bytes de RAM; R0, R1, R2
;  Uso: poner el valor en L_Byte y H_Byte, llamar subrutina B2_BCD, resultado en
;  R2 es el LSD y en R1 es MSD: (1er nibble R2) (unidades), (2do nibble R2)
;  decenas,(1er nibble R1) centenas,(2do nibble R1) (miles)....
;   para facilidad unidades en  BCDX1 decenas en BCDX10, centenas en BCDX100
B2_BCD  clc                     ;limpia bit acarreo
       mov    #!16,count        ;contador a 16
       clr    R1
```

```
        clr    R2
        clr    R0
loop16  rol    L_Byte
        rol    H_Byte
        rol    R2
        rol    R1
        rol    R0
        dbnz   count,adjDEC
        lda    #%00001111
        and    R2
        sta    BCDX1       ; variable almacena las unidades
        lda    #%11110000
        and    R2
        NSA
        sta    BCDX10      ; variable almacena las decenas
        lda    #%00001111
        and    R1
        sta    BCDX100     ; variable almacena las centenas
        RTS                ;SALE
adjDEC  LDA    R2
        MOV    R2,TEMP2
        jsr    adjBCD
        mov    temp2,R2
        LDA    R1
        MOV    R1,TEMP2
        jsr    adjBCD
        mov    temp2,R1
        LDA    R0
        MOV    R0,TEMP2
        jsr    adjBCD
        mov    temp2,R0
        jmp    loop16
adjBCD  add    #!3
        STA    temp1
        brclr  3,temp1,Q
        mov    TEMP1,temp2
Q       LDA    TEMP2
        add    #$30
        STA    temp1
        brclr  7,temp1,W
```

```
      mov    temp1,temp2          ;salva como MSD
W     rts
;---------------------------------------------------------------------
; Rutina Bcdascii
; Convierte Los Dígitos En Bcd A Sus Correspondientes Caracteres ASCII
;---------------------------------------------------------------------
BCDASC            lda     #$30                ;ASCII = BCD + H'30'
                  ADD       BCDX1          ;CONVERT BCDX1 TO ACSX1
                  sta     BCDX1              ;CONVERT BCDX1 TO ACSX1
                  lda     #$30                ;ASCII = BCD + H'30'
                  ADD       BCDX10          ;CONVERT BCDX1 TO ACSX1
                  sta     BCDX10             ;CONVERT BCDX1 TO ACSX1
                  lda     #$30                ;ASCII = BCD + H'30'
                  ADD       BCDX100         ;CONVERT BCDX1 TO ACSX1
                  sta     BCDX100           ;CONVERT BCDX1 TO ACSX1;
   rts
**********************************************************************
**RUTINA BCD A BINARIO*
*convierte un numero bcd de tres cifras almacenado en las
*variables uni(unidades), dec(decenas), cent (centenas)
*a un numero binario salida dos variables(bin_l,bin_h)*
*********************************************************
BCD-BIN       ldx #!10
          lda dec
          mul
          add uni
          sta bin_l
          ldx #!100
          lda cent
          mul
          add bin_l
          sta bin_l
          txa
          adc #0
          sta bin_h ;FIN DE RUTINA BCD-BIN
          RTS
*****************RUTINA ASCCI A BINARIO****************************************
ASCC-BCD
           lda uni
           sub #$30
```

```
            sta uni
            lda dec
            sub #$30
            sta dec
            lda cent
            sub #$30
            sta cent
            RTS
*****************************************************************************
TABLA
    DB    $33 ;3
    DB    $36 ;6
    DB    $39 ;9
    DB    $4E ;#

    DB    $41 ;
    DB    $46 ;
    DB    $38 ;8
    DB    $30 ;0

    DB    $31 ;1
    DB    $34 ;4
    DB    $37 ;7
    DB    $45 ;E

    DB    $32 ;2
    DB    $35 ;5
    DB    $43 ;
    DB    $4D ;
*****************************************************************************
*RUTINA DE ATENCION A INTERRUPCION DE TECLADO*
TECLADO:      bset $1,KBSCR    ;imaskk =1
            ldhx #$0
            MOV #$10,TEMP    ;***********RETARDOS PARA***********
            Jsr SIGLO        ;**************ESTABILIZACION DE TECLA************
            mov  #$30,ddra   ;puerto A Lsb lectura
            bclr E,porta
            mov  porta,columna  ;averigua columna
            mov  #$01,portd   ;uno a rotar
            mov  #$00,PTAUE   ;PULL UP puerto A Lsb desconectado
```

```
        mov  #$FF,fila
LAZO_FILA
        mov  porta,PA
        lda  #$0f
        inc  fila
        and  Pa
        cbeqa #$0f,SALTO1
        asl  portd
        bra  LAZO_FILA
SALTO1
        sec
        ldx  #$4
LAZO_COLUMNA
        decx
        asr  columna
        bcc  SALTO2
        bra  LAZO_COLUMNA
SALTO2
      lda #$4
      mul
      add fila
      tax
      lda tabla,x
      sta caracter
;*************************RUTINA SI YA SOLTO TECLA**********************
      mov  #$0F,PTAUE   ;PULL UP puerto A Lsb desconectado
      MOV  #$60,PORTD       ;puerto D Lsb en cero
KEYREL     LDA  PORTA
      AND  #$0F
      CBEQA #$0F,REVISA
      JMP KEYREL
REVISA     BRCLR 5,caracter,DECIDE
      LDA BUFFER
      cmp #$3
      bhs SALE
      ldx vram
      mov caracter,x+
      stx vram
      jsr saca_caracter
      INC BUFFER       ;VARIABLE PARA SABER SI YA ESTAN LOS 3 NUMEROS
```

```
        JMP SALE
DECIDE    LDA  caracter
        CBEQA #$4E,A2
        JMP ACEPTAR
;*********************************RUTINA BORRA DATO************************
A2        LDA  BUFFER
        CBEQA #$00,SALE
        ldx #%10000     ;DESPLAZA CURSOR IZQ
        BCLR 1,C
        jsr DALTO
        MOV #$ff,TEMP
        Jsr SIGLO
        ldx #$20        ;BORRA DATO EN LA POSICION DEL CURSOR
        BSET 1,C
        jsr DALTO
        MOV #$ff,TEMP
        Jsr SIGLO
        ldx #%10000     ;DESPLAZA CURSOR IZQ
        BCLR 1,C
        jsr DALTO
        MOV #$ff,TEMP
        Jsr SIGLO
        DEC vram
        DEC BUFFER
        JMP SALE
ACEPTAR   LDA BUFFER
        cmp #$3
        bhs A1
        JMP SALE
A1        BSET 0,yes
        NOP
        jmp fin_tecl
SALE
        MOV #$ff,TEMP ;***********RETARDOS PARA***********
        Jsr SIGLO ;***************ESTABILIZACION DE TECLA************
        MOV #$ff,TEMP ;*******************************
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
```

```
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        MOV #$ff,TEMP
        Jsr SIGLO
        bset $1,KBSCR         ;imaskk =1
        mov  #$0F,KBIER       ;habilita cada una columnas pta0-pta3
        bset $2,KBSCR         ;ACKK=1 limpia cualquier falsa interrupción
        mov #%00000001,KBSCR   ;imaskk =0
        cli
fin_tecl   MOV  #$40,PORTD       ;puerto D Lsb en cero
        bclr E,porta
     rti
*******************************************************************************
saca_caracter
        LDX caracter
        BSET 1,C
        jsr DALTO
        mov #$FF,R0D
RETA2     jsr RETARDO
        dbnz R0D,RETA2
  RTS
*************************************************************
dummy_isr:
     rti          ; retorno
  org  Vector
     dw  INTER_CONVERSION  ; ADC
     dw  TECLADO      ; vector  teclado
     dw  dummy_isr    ; (No Vector Asignado $FFE2-$FFE3)
     dw  dummy_isr    ; (No Vector Asignado $FFE4-$FFE5)
     dw  dummy_isr    ; (No Vector Asignado $FFE6-$FFE7)
     dw  dummy_isr    ; (No Vector Asignado $FFE8-$FFE9)
     dw  dummy_isr    ; (No Vector Asignado $FFEA-$FFEB)
     dw  dummy_isr    ; (No Vector Asignado $FFEC-$FFED)
     dw  dummy_isr    ; (No Vector Asignado $FFEE-$FFEF)
     dw  dummy_isr    ; (No Vector Asignado $FFF0-$FFF1)
     dw  dummy_isr    ; TIM1 Overflow Vector
     dw  dummy_isr    ; TIM1 Channel 1 Vector
```

```
dw  dummy_isr   ; TIM1 Channel 0 Vector
dw  dummy_isr   ; (No Vector Asignado $FFF8-$FFF9)
dw  Pwm         ; ~IRQ1
dw  dummy_isr   ; SWI Vector
dw  main        ; Reset Vector
```

8.19.

IV.2. MICROCONTROLADOR PIC 16F873.

Este programa realiza control Fuzzy de la aplicación utilizando solo el microcontrolador Las rutinas utilizadas por Fuzzye3t se encuentran en el anexo II.

```
;***PROGRAMA DE CONTROL FUZZY PARA LA TARJETA DE APLICACIÓN CON  PIC16F873 ***
; UTILIZA EL FC fuzzypic_+-16°_21.fc)
            LIST     P=16f873
            INCLUDE  "P16F873.INC"
       cblock  0x20
;****************************************
Insertar variables utilizadas por fuzzye3t (anexo II)
;****************************************
SERIALA
ADQ_N
WORK
CONVER_1
CONVER_2
TEC_lsb
TEC_msb
caracter
columna
fila
val_col
val_fil
R0C
R0E
W1
temp
BUFFER
yes
vram
bin_l
bin_h
R1
R2              ;BCD salida Binario->BCD
count
temp1
temp2
BCDX100         ;CENTENAS
```

```
BCDX10        ;DECENAS
BCDX1         ;UNIDADES
BCDX0         ;DECIMAS DE GRADO
BCDX01        ;CENTESIMAS
BCDX001       ;MILESIMAS DE GRADO
NUMERADOR
MSNUM_4   ;MULTIPLICACION 16*16=32
ACCdHH
ACCdHL
ACCdLL
RestoHH
RestoHL
RestoLL
PWMLSB
PWMMSB
error2
p
ERRORLCD
REFRESCO ;VARIABLE REFERSCO LCD
        endc
;**************************************
            ORG  0x00
            goto INICIO
            ORG  0x04
            goto INTER
;**************************************
VERTICES
;Insertar  aquí tabla  de vértices          (anexo ii)
TABLA_TEC ADDWF 2,1
    RETLW 0x33
    RETLW 0x36
    RETLW 0x39
    RETLW 0x4E
    RETLW 0x41
    RETLW 0x46
    RETLW 0x38
    RETLW 0x30
    RETLW 0x31
    RETLW 0x34
    RETLW 0x37
```

```
    RETLW 0x45
    RETLW 0x32
    RETLW 0x35
    RETLW 0x43
    RETLW 0x4D
;*****************************************************
TABLA_LCD ADDWF 2,1
    RETLW " "
    RETLW "T"
    RETLW "E"
    RETLW "M"
    RETLW "P"
    RETLW "E"
    RETLW "R"
    RETLW "A"
    RETLW "T"
    RETLW "U"
    RETLW "R"
    RETLW "A"
    RETLW " "
    RETLW "U"
    RETLW "C"
    RETLW " "
;=======================
TABLA_LCD1 ADDWF 2,1
    RETLW " "
    RETLW "C"
    RETLW "O"
    RETLW "N"
    RETLW "T"
    RETLW "R"
    RETLW "O"
    RETLW "L"
    RETLW " "
    RETLW "D"
    RETLW "I"
    RETLW "F"
    RETLW "U"
    RETLW "S"
    RETLW "O"
```

```
    RETLW " "
    RETLW "M"
    RETLW "I"
    RETLW "C"
    RETLW "R"
    RETLW "O"
    RETLW "C"
    RETLW "O"
    RETLW "N"
    RETLW "T"
    RETLW "R"
    RETLW "O"
    RETLW "L"
    RETLW "A"
    RETLW "D"
    RETLW "O"
    RETLW " "
    RETLW " "
;*****************************INICIO DE RUTINAS FUZZY*********************
;** INSERTAR  AQUÍ SUBRUTINAS: REGLAS, MEMBR, COMP@, MINMAX,REST@,  (anexo II)
;** INSERTAR  AQUI  : SUM@I,  MULSUM@S   (anexo II)
;*****************************FIN DE RUTINAS FUZZY*********************
DIV@
        movlw   .24
        movwf   INDIC@
        clrf    DNUMDEN_2
        clrf    DNUMDEN_1
        clrf    REST_3
        clrf    REST_2
        clrf    REST_1
        clrf    SDEN_3
DLAZO
        bcf     STATUS,C
        rlf     MSNUM_1,1
        rlf     MSNUM_2,1
        rlf     MSNUM_3,1
        rlf     REST_1,1
        rlf     REST_2,1
        rlf     REST_3,1
        movf    SDEN_3,0
```

```
        subwf   REST_3,0
        btfss   STATUS,2
        goto    PAS@1
        movf    SDEN_2,0
        subwf   REST_2,0
        btfss   STATUS,2
        goto    PAS@1
        movf    SDEN_1,0
        subwf   REST_1,0
PAS@1
        btfss   STATUS,0
        goto    PAS@2
        movf    SDEN_1,0
        subwf   REST_1,1
        btfss   STATUS,0
        decf    REST_2,1
        movf    SDEN_2,0
        subwf   REST_2,1
        btfss   STATUS,0
        decf    REST_3,1
        movf    SDEN_3,0
        subwf   REST_3,1
        bsf     STATUS,0
PAS@2
        rlf     DNUMDEN_1,1
        rlf     DNUMDEN_2,1
        decfsz  INDIC@,1
        goto    DLAZO
        return
;*************************************************
;**RUTINAS ESCRITURA LCD****
;*************************************************
RETAR   movlw  0xFF  ;RUTINA DE RETARDOS NECESARIOS
        movwf  temp
DECRE   decfsz temp,1
        goto   DECRE
          return
CONTROL  bcf   PORTA,2    ;RUTINA GENERA LAS SEÑALES DE CONTROL PARA ESCRIBIR EN EL LCD
           goto  DATO2
DATO       bsf   PORTA,2
```

```
DATO2     bcf   PORTA,5
           movwf  R0E
           movwf  W1
           SWAPF  W1,1
           RLF    W1,0
           andlw  0x0E      ;GARANTIZA QUE LOS BITS 7-4 Y 0 PTB SEAN CERO
           MOVWF  PORTB     ;SACA PARTE ALTA DEL RESULTADO
           BTFSC  R0E,7
           GOTO   S1
           BCF    PORTA,4
           GOTO   ACTIVA1
S1         BSF    PORTA,4
ACTIVA1 call   RETAR
           BSF    PORTA,5
          call   RETAR
          call   RETAR
          BCF    PORTA,5
           RLF    R0E,0
           andlw  0EH     ;GARANTIZA QUE LOS BITS 7-4Y0 PTB SEAN CERO
           MOVWF  PORTB     ;SACA PARTE BAJA DEL RESULTADO
           BTFSC  R0E,3
          GOTO   S2
           BCF    PORTA,4
           GOTO   ACTIVA2
S2          BSF    PORTA,4
ACTIVA2  call   RETAR
           BSF    PORTA,5
           call   RETAR
           call   RETAR
           BCF    PORTA,5
           RETURN
SIGLO     movlw   0X20        ;RETARDO ENTRE CARACTERES
           movwf   temp2
RETA1     call    RETAR
          decfsz  temp2,1
          GOTO    RETA1
          RETURN
;*********************************
;Esta Subrutina, Espera Hasta Que Todas Las Teclas Sean Soltadas
;Espera Mediante Un Sleep
```

```
KeyRelease
        call    delay16         ;
        comf    PORTB,W         ;LEE PORTB
        bcf     INTCON,RBIF     ;LIMPIA BADERA INT TECLADO
        bsf     INTCON,RBIE     ;HABILITA INT TECLADO
        andlw   B'11110000'     ;LIMPIA LAS SALIDAS
        btfsc   STATUS,2        ;TECLA SIGUE PRESIONADA?
        return                  ;no ENTONCES RETORNA
        sleep                   ;DE LO CONTRARIO ESPERA
        bcf     INTCON,RBIE     ;DESPÌERTA
        comf    PORTB,W
        bcf     INTCON,RBIF     ;LIMPIA BANDERA
        goto    KeyRelease      ;INTENTA OTRA VEZ
;*************************************************************
;delay16 espera 16.4mSecs usando interrupción TMR0
delay16
        bsf     STATUS,RP0      ;pagina 1
        movlw   B'00000111'     ;fosc/256 --> TMR0
        movwf   OPTION_REG      ;      /
        bcf     STATUS,RP0      ;pagina 0
        clrf    TMR0
        bcf     INTCON,T0IF     ;LIMPIA BANDERA
        bsf     INTCON,T0IE     ;HABLITA INT
CheckAgain
        btfss   INTCON,T0IF     TIMER SE DESBORDA?
        goto    CheckAgain      ;REVISA DE NUEVO
        bcf     INTCON,T0IE
        bcf     INTCON,T0IF     ;LIMPIA BANDERA
        return
;********************
;delay16 espera 56.4mSecs usando interrupción TMR0
delay56
        bsf     STATUS,RP0      ;PAGINA 1
        movlw   B'00000111'     ;fosc/256 --> TMR0
        movwf   OPTION_REG      ;      /
        bcf     STATUS,RP0      ;PAGINA 0
        clrf    TMR0
        bcf     INTCON,T0IF     ;LIMPIA BANDERA
        bsf     INTCON,T0IE     ;HABLITA INT
OTRAVEZ
```

```
        btfss   INTCON,T0IF     ;TIMER SE DESBORDA?
        goto    OTRAVEZ     ;REVISA DE NUEVO
        bcf     INTCON,T0IF     ;LIMPIA BANDERA
        return
;*****************************************************************
;*RUTINA DE INICIALIZACION DE LCD*
LCD_INIT   movlw   b'00000010'   ;LIMPIA  HOME OJO DE SER CLEAR #%01
       call    CONTROL
       movlw   28H    ;BUS DE DATOS (4 BITS) (2 LINEA EN PANTALLA) TAMAÑO CARACTER 5*7
       call    CONTROL
       movlw   0CH          ; PANTALLA ACTIVA Y CURSOR NO PARPADEA
       call    CONTROL
       CLRF    PORTB
       RETURN
;********MULTIPLICACION 16*16 DOS NUMEROS SIN SIGNO*****************
;*MULTIPLICA (Q1*Q2)=MSNUM(3(mmsb),2(msb),1(lsb))*******************
M_MSNUM clrf   MSNUM_3
          clrf   MSNUM_2
          clrf   MSNUM_1
          clrf   MSNUM_4
          movlw  .24
          movwf  INDIC@
MLAZO_A
          bcf    STATUS,0
          rlf    MSNUM_4,1
          rlf    MSNUM_1,1
          rlf    MSNUM_2,1
          rlf    MSNUM_3,1
          bcf    STATUS,0
          rlf    Q2_lsb,1
          rlf    Q2_msb,1
          btfss  STATUS,0
          goto   NSUM@_A
          movf   Q1_lsb,0
          addwf  MSNUM_4,1
          btfsc  STATUS,0
          incf   MSNUM_1,1
          movf   Q1_msb,0
          addwf  MSNUM_1,1
          btfsc  STATUS,0
```

```
        incf   MSNUM_2,1
NSUM@_A
        decfsz INDIC@,1
        goto   MLAZO_A
        return
;*************************************************
;********DIVISION 24BITS(MSNUM/Q2=MSNUM)          ***
Dividir
        movlw   .24                     ;para 24 desplazamientos
        movwf   temp
        movf    MSNUM_3,W               ;lleva Dividendo al temporal ACCd
        movwf   ACCdHH
        movf    MSNUM_2,W
        movwf   ACCdHL
        movf    MSNUM_1,W
        movwf   ACCdLL
        clrf    MSNUM_3
        clrf    MSNUM_2
        clrf    MSNUM_1
        clrf    RestoHH
        clrf    RestoHL
        clrf    RestoLL
lazo
        bcf     STATUS,C
        rlf     ACCdLL,F
        rlf     ACCdHL,F
        rlf     ACCdHH,F
        rlf     RestoLL,F
        rlf     RestoHL,F
        rlf     RestoHH,F
        movf    Q2_msb,W
        subwf   RestoHL,W               ;comprueba si Divisor>Resto
        btfss   STATUS,Z
        goto    Nochk
        movf    Q2_lsb,W
        subwf   RestoLL,W               ;si msb es igual comprobar lsb
Nochk
        btfss   STATUS,C                ;carry a uno si Resto>Divisor
        goto    Nogo
        movf    Q2_lsb,W                ;Resto-Divisor a Resto
```

```
            subwf     RestoLL,F
            btfss     STATUS,C
            decf      RestoHL,F
            movf      Q2_msb,W
            subwf     RestoHL,F
            bsf       STATUS,C                ; 1 es Resultado(MSNUM)
Nogo
            rlf       MSNUM_1,F
            rlf       MSNUM_2,F
            rlf       MSNUM_3,F
            decfsz    temp,F                  ;loop
            goto      lazo
            return
;*************************************************************
;********CONVERSION BIN(16BITS) A 3 DIGITOS BCD    **********
;*****ENTRA MSNUM_2(MSB),MSNUM_1(LSB),SALE R1(MSB),R2(LSB)***
Bin_BCD     BCF       STATUS,C
          MOVLW     0X10
          MOVWF     count
              CLRF      R2
              CLRF      R1
LOOP_16     RLF       MSNUM_1,1
          RLF       MSNUM_2,1
          RLF       R2,1
          RLF       R1,1
          DECFSZ    count,1
          GOTO      AJUSTE
          RETURN
AJUSTE      MOVF      R2,0
          MOVWF     temp2
          CALL      AJUSTE_BCD
          MOVF      temp2,0
          MOVWF     R2
          MOVF      R1,0
          MOVWF     temp2
          CALL      AJUSTE_BCD
          MOVF      temp2,0
          MOVWF     R1
          GOTO      LOOP_16
AJUSTE_BCD  ADDLW     0X03
```

```
        MOVWF    temp
        BTFSC    temp,3
        GOTO     Q1
        GOTO     Q2
Q1        MOVF     temp,0
        MOVWF    temp2
Q2        MOVF     temp2,0
        ADDLW    0X30
        MOVWF    temp
        BTFSC    temp,7
        goto     Q3
        GOTO     Q4
Q3        MOVF     temp,0
        MOVWF    temp2
Q4        RETURN
;*************************************************************
;********CONVERSION 3 DIGITOS BCD A BIN (16BITS)   *************
;*****ENTRA Q2(MSB),Q2(LSB),BCDX1,BCDX100; SALE R1(MSB),R2(LSB)*
BCD_BIN    BCF    STATUS,C
        movlw  0x0a      ;diez decimal
        movwf  Q1_lsb
        clrf   Q1_msb
            call   M_MSNUM    ;MULTIPLCA DECIMAL POR 10
        MOVF   MSNUM_1,0
        ADDWF  BCDX1,0    ;SUMA UNIDADES
        MOVWF  R2
        MOVLW  0X64      ;CARGA PARTE BAJA DEL NUMERO 100 DECIMAL
            MOVWF  Q1_lsb
        CLRF   Q1_msb    ;CARGA PARTE ALTA DEL NUMERO 100 DECIMAL
        MOVF   BCDX100,0
        MOVWF  Q2_lsb
        CLRF   Q2_msb
            call   M_MSNUM    ;MULTIPLCA DECIMAL POR 10
        MOVF   MSNUM_1,0
        ADDWF  R2,1
        MOVF   STATUS,0  ;averigua valor de carry
        ANDLW  0X01      ;enmascara
        ADDWF  MSNUM_2,0
        MOVWF  R1
          return               ; BCD a binario conversión
```

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;subrutina de atención a la interrupción
INTER   btfsc INTCON,0 ;RBIF=1 interrupción CAMBIO ESTADO PUERTO B (TECLA PULSADA)
      goto  TECLA
      btfsc INTCON,INTF ;INTE=1 interrupción señal ext(cruce por cero)
      goto  CRUCE0
      btfsc PIR2,0      ;CCP2IF=1 BANDERA INTE por COMPARACION
      goto  COMPAR
      btfsc INTCON,2    ;TOIF=1 interrupción TIMER cero
      goto  TIMER2
            goto  EXIT1      ;No hay interrupción, retorna
;*********************************************************
;ESTA RUTINA ENCUENTRA CUAL DE LAS TECLAS ESTA PRESIONADA GUARDA SU VALOR
;EN UNA DE TRES VARIABLES, MUESTRA EN EL LCD, BORRA DATO GUARDADO SI LA
;TECLA N ESTA PRESIONADA, Y CONTINUA CON EL RESTO DEL PROGRAMA LUEGO DE
;SALIR DE LA INTERRUPCION SI EL BIT 0 DE LA VARIBLE yes ES '1'
;ADEMAS ESPERA HASTA QUE LAS TECLAS SEAN SOLTADAS PARA SALIR DE LA INTERRUPCION
TECLA
            bcf     INTCON,RBIE     ;deshabilita int TEC
            comf    PORTB,W         ;complementa portB Y PONE EN W
            bcf     INTCON,RBIF     ;LIMPIA BANDERA TEC
            call    delay16         ;Esperado Por 16mSecs EVITA REBOTES
            movf    PORTB,W         ;VUELVE A LEER PUERTO OTRA VEZ
            andlw   B'11110000'     ;ENMASCARA LAS SALIDAS
      addlw   0x0f
      movwf   columna
      clrf    val_col
      movlw   0x03
      movwf   val_fil
LAZO_COL bsf    STATUS,0
      rlf     columna,1
            btfss   STATUS,0
            goto    FILA
            incf    val_col,1
            goto    LAZO_COL
FILA    BCF     PORTA,4
      MOVLW   0X0E
            MOVWF   PORTB
            MOVLW   0X07
            MOVWF   temp
```

```
LAZO_FIL MOVF   PORTB,0
          andlw   B'11110000'    ;ENMASCARA LAS SALIDAS
      addlw   0x0f
      bcf    STATUS,2
      xorlw   0xff
          btfss   STATUS,2
          goto    SALE
          bsf     PORTA,4
          decf    val_fil,1
          movf    temp,0
          movwf   PORTB
          movlw   0x03
          addwf   temp,1
          goto    LAZO_FIL
SALE
      movf    val_col,0
      addwf   val_col,0
      addwf   val_col,0
      addwf   val_col,0
      addwf   val_fil,0
      call    TABLA_TEC
      movwf   caracter
      CLRF    PORTA
      CLRF    PORTB
      BTFSS   caracter,5
      GOTO    DECIDE
      MOVF    BUFFER,0
      XORLW   0X03
      BTFSS   STATUS,2
      GOTO    GUARDA
      clrf    PORTB
      bcf     PORTA,4
      BCF     PORTA,5
          call    KeyRelease     ;REVISA SI SOLTO TECLA
          BCF     INTCON,RBIE    ;DESHABILITA INT TECLADO
      GOTO    FINT
GUARDA  movf    vram,0
      movwf   FSR
      bcf     STATUS,7
      MOVF    caracter,0
```

```
      MOVWF   INDF
           call    KeyRelease     ;REVISA SI SOLTO TECLA
           BCF     INTCON,RBIE    ;DESHABILITA INT TECLADO
      INCF    BUFFER,1
      INCF    vram,1         ;incrementa variable para guardar datos
      MOVF    caracter,0     ;MUESTRA CARACTER EN LCD
      call    DATO
      clrf    PORTB
      bcf     PORTA,4
      BCF     PORTA,5
      goto    FINT
DECIDE
      call    KeyRelease     ;REVISA SI SOLTO TECLA
           BCF     INTCON,RBIE    ;DESHABILITA INT TECLADO
      movf    caracter,0
      XORLW   0X4E
      BTFSS   STATUS,2
      GOTO    ACEPTAR
      MOVF    BUFFER,0
      XORLW   0X00
      BTFSC   STATUS,2
      GOTO    FINT
      movlw   b'10000'      ;DEPLAZA CURSOR 1 PUESTO A LA IZQUIERDA
      call    CONTROL
      movlw   b'100000'     ;BORRA DATO EN LA POSICION DEL CURSOR
      call    DATO
      movlw   b'10000'      ;DEPLAZA CURSOR 1 PUESTO A LA IZQUIERDA
      call    CONTROL       ;CARACTER ESPACIO
      clrf    PORTB         ;FILAS DE TECLADO A CERO
      bcf     PORTA,4       ;FILAS DE TECLADO A CERO
      bcf     PORTA,5       ;FILAS DE TECLADO A CERO
      DECF    BUFFER,1
      DECF    vram,1
      GOTO    FINT
ACEPTAR MOVF    BUFFER,0
      XORLW   0X03
      BTFSS   STATUS,2
      GOTO    FINT
      BSF     yes,0
      clrf    PORTB       ;FILAS DE TECLADO A CERO
```

```
    bcf    PORTA,4     ;FILAS DE TECLADO A CERO
    bcf    PORTA,5     ;FILAS DE TECLADO A CERO
    GOTO   EXIT1
;**********************************
COMPAR
    BCF    T1CON,0   ;APAGA TIMER 1
    CLRF   CCP2CON
    BCF    PORTC,1
    clrf   TMR1L
    clrf   TMR1H
    MOVF   PWMLSB,0
    MOVWF  CCPR2L
    MOVF   PWMMSB,0
    MOVWF  CCPR2H
    BCF    PIR2,0    ;limpia bandera señalizacion int comparación
    MOVLW  0X08      ;MODULO CP2 EN MODO COMPARACION ACTIVA
    MOVWF  CCP2CON   ;PIN RC1 CUANDO COINCIDEN VALORES TMR1 Y CCPR2H
    BSF    STATUS,RP0   ;BANCO 1
    BSF    PIE2,0    ;HABILITA INTERUPCION COMPARACION
    BCF    STATUS,RP0   ;BANCO 0
    GOTO   EXIT1
;**********************************
TIMER2
        bcf    INTCON,T0IE     ;limpia mascara
    GOTO   EXIT1
;**********************************
CRUCE0  MOVLW  0X08     ;MODULO CP2 EN MODO COMPARACION ACTIVA PIN RC1
    MOVWF  CCP2CON   ;CUANDO COINCIDEN VALORES TMR1 Y CCPR2H-L
    BSF    T1CON,0   ;HABILITA CONTADOR TMR1
    BSF    yes,0
    BCF    PIR2,0    ;LIMPIA BANDERA
    BSF    STATUS,RP0   ;BANCO 1
    BSF    PIE2,0    ;HABILITA INTERUPCION COMPARACION
    BCF    STATUS,RP0   ;BANCO 0
    bcf    INTCON,INTF   ;limpia bandera señalizacion interrupción externa
    GOTO   EXIT1
;******************************************************
FINT
        bcf    INTCON,RBIF     ;LIMPIA BANDERA INT TECLADO
        bsf    INTCON,RBIE     ;HABILITA INT TECLADO
```

```
EXIT1
          movf  WORK,0
          retfie         ;retorna
;*********************************************************
INICIO  clrwdt              ;refresca el perro guardián
;*********************************************************
;se entra al modo de configuración del PIC
          bsf    STATUS,RP0     ;Cambia al banco 1
          bcf    STATUS,RP1     ;RP0=1 y RP1=0
;*******************************
;puerto A como ENTRADA ANALOGICA AL CONVERSOR
     movlw   b'10001110'    ;BIT7=1 JUST DERECHA, RA0 entrada conversor
     movwf   ADCON1
     BCF     STATUS,RP0     ;Cambia al banco 0
     movlw   b'10000001'    ;frec/32, canal 0 ,activa conversor pero
     movwf   ADCON0         ; no inicia conversion
          bsf    STATUS,RP0     ;Cambia al banco 1
          movlw   b'00000011'    ;RAO entrada A/D,RA1(setpoint), RA RESTANTES salida
          movwf   TRISA
;*******************************
;puerto C
;se programa la comunicación serial
          movlw   b'10000000'    ;RC7/RX como entrada y RC6/TX como salida
          movwf   TRISC          ;RC0-5 como salidas RC1 (PWM)
;************************************
;puerto B como entrada
          movlw   b'11110001'    ;RB7-4 entrada(teclado),RB3-RB1 salida(teclado),
          movwf   TRISB          ;RB0 entrada(cruce0)
     movlw   b'01000000'
          movwf   OPTION_REG
     clrf    PIE1
;************************************
;CONFIGURACION DE A/D:se realizo en configuración pta
;CONFIGURACION DE option: se realizo en configuración ptb
;************************************
;*CONFIGURACION DEL PWM ccp2-pin RC1;ccp1-pin rc2*
;*PARA GENERAR EL PWM SE UTILIZA EL TIMER 1 EN MODO COMPARACION*
;SE CARGA EN LOS REGISTROS CCPR2H-L EL CONTENIDO DEL TIEMPO OFF*
;LUEGO DE ESTE TIEMPO EL PIN RC1 PASA A ON (ANCHO DE PULSO)*
;CCPR2H-L=(8.33mS-ANCHO DE PULSO)
```

```
      BCF     STATUS,RP0  ;Cambia al banco 0
           clrf    PORTA     ;pone todos los bits del puerto A en 0
           clrf    PORTB     ;pone todos los bits del puerto B a 0
           clrf    PORTC     ;pone todos los bits del puerto C a 0
           clrf    T1CON     ;APAGA TIMER 1
      MOVLW   0X08      ;MODULO CP2 EN MODO COMPARACION ACTIVA
      MOVWF   CCP2CON   ;PIN RC1 CUANDO COINCIDEN VALORES TMR1 Y CCPR2H-L
      clrf    TMR1L
      clrf    TMR1H
           movlw   0X52      ;ILUMINACION CERO TIEMPO OFF(8.3mS)
           movwf   CCPR2L
           movlw   0X14      ;ILUMINACION CERO TIEMPO OFF(8.3mS)
           movwf   CCPR2H
      bcf     PIR2,0
;*********************************************************
           clrf    PORTA     ;pone todos los bits del puerto A en 0
           clrf    PORTB     ;pone todos los bits del puerto B a 0
           clrf    PORTC     ;pone todos los bits del puerto C a 0
           call    delay56
           call    delay56
           call    delay56
           call    delay56
;*********************************************************
      call    LCD_INIT
;inicialización de variables
VUELVE              movlw   .1
           movwf   SERIALA
           clrf    ADQ_N
           clrf    BUFFER
      MOVLW   BCDX100     ;DIRECCIONAMIENTO INDIRECTO
      MOVWF   vram        ;GUARDA CENT,DECE,UNID
           bcf     INTCON,RBIE     ;deshabilita mascara
           movf    PORTB,W       ;lee puerto
           bcf     INTCON,RBIF     ;limpia bandera
;****************************************
;*MUESTRA  MENSAJE CONTROL DIFUSO****
              movlw   01H         ;BORRA DISPLAY
         call    CONTROL
MUESTRA             clrf    R0C         ;INICIA CONTADOR DE TABLA
CICLO     movf    R0C,W       ;HACE BARRIDO DE TABLA
```

```
        call    TABLA_LCD1
        call    DATO

        incf    R0C,1
        movlw   0FH
        XORWF   R0C,W      ;PREGUNTA POR FIN DE LECTURA DE TABLA
        btfss   STATUS,Z
        GOTO    CICLO
             movlw   0XC6          ;POSICION DEL CURSOR 6 RENGLON 2 DISPLAY
        call    CONTROL
;*****************************
      clrf    PORTB
      bcf     PORTA,4
           movlw   b'10001000'    ;habilitación todas las interrupciones, bit3 int teclado
           movwf   INTCON         ;bit4 intr ext desactivada
      CLRF    yes
;*********************************************************
;El siguiente es el lazo en el que se quedara el PIC normalmente
;en caso de no entrar y aceptar el dato de temperatura
LEER
   NOP
           BTFSS  yes,0
           GOTO   LEER
;CONVIERTE EL DATO DE ASCCI A BCD
           movlw  0x30
      subwf  BCDX100,1  ;CENTENAS
      subwf  BCDX1,1    ;UNIDADES
      subwf  BCDX10,0   ;DECENAS
      movwf  BCDX10
      MOVWF  Q2_lsb
      CLRF   Q2_msb
;CONVIERTE EL DATO DE BCD A BINARIO
      call   BCD_BIN
;**********************LIMITA EL DATO (R2) ENTRE 0 Y 150ºC********
      movf   R1,0
      andlw  0xff
      BTFSS  STATUS,2
      GOTO   VUELVE
      movf   R2,0
      SUBLW  0X96
```

```
      BTFSS  STATUS,0
      GOTO   VUELVE
      bsf   STATUS,RP0   ;Selecciona el banco 1
      movlw   b'11000000' ; DESABILITA RESISTENCIAS PULL PUERTO B
              movwf   OPTION_REG
      bcf   STATUS,RP0   ;Selecciona el banco 0
;CONVIERTE EL DATO DEL TECLADO A FORMATO 0-1023([Q2_lsb]*1023/150)
      movf  R2,0
      MOVWF Q2_lsb
      CLRF  Q2_msb
      MOVLW 0X03
      MOVWF Q1_msb
      MOVLW 0XFF
      MOVWF Q1_lsb
           CALL  M_MSNUM   ;MULTIPLICA DATOA(16)*DATOB(16)
           MOVLW 0X96      ;CARGA PARTE BAJA DEL DIVISOR (150DECIMAL)
           MOVWF Q2_lsb
      CLRF  Q2_msb    ;CARGA PARTE Alta DEL DIVISOR (150 DECIMAL)
      CALL  Dividir    ;divide el dato entre el #150 (RESULTADO EN MSNUM)
      MOVF  MSNUM_1,0
      MOVWF TEC_lsb
      MOVWF Q1_lsb
      MOVF  MSNUM_2,0
      MOVWF TEC_msb
      MOVWF Q1_msb
           movlw   01H         ;BORRA DISPLAY
    call    CONTROL
;*************************INICIO CONVERSION*********************
;*****REALIZA CUATRO AQUISICIONES Y TOMA EL PROMEDIO DE ESTAS ***
BIG_LOOP  BCF   PIR1,ADIF      ;LIMPIA BANDERA FIN DE CONVERSION
       BSF   ADCON0,GO      ;INICIA CONVERSION
ESPERA    BTFSS PIR1,ADIF      ; espera fin de conversion(6.4uS*12Tad=76.8uS)
       GOTO  ESPERA
        bsf   STATUS,RP0   ;Selecciona el banco 1
             movf  ADRESL,0     ;LEE dato LSB de conversion UBICADO BANCO 1
        bcf   STATUS,RP0   ;Selecciona el banco 0
        MOVWF Q1_lsb
        movf  ADRESH,0     ;LEE dato MSB de conversion UBICADO BANCO 0
        MOVWF Q1_msb
        clrf  SDEN_2
```

```
        clrf  SDEN_1
        CALL  SUM@I
;//////////////////////////////////
       BCF   PIR1,ADIF     ;LIMPIA BANDERA FIN DE CONVERSION
       BSF   ADCON0,GO     ;INICIA CONVERSION
ESPERA1   BTFSS PIR1,ADIF      ; espera fin de conversion(6.4uS*12Tad=76.8uS)
       GOTO  ESPERA1
        bsf   STATUS,RP0   ;Selecciona el banco 1
              movf  ADRESL,0     ;LEE dato LSB de conversion UBICADO BANCO 1
        bcf   STATUS,RP0   ;Selecciona el banco 0
        MOVWF Q1_lsb
        movf  ADRESH,0     ;LEE dato MSB de conversion UBICADO BANCO 0
        MOVWF Q1_msb
        CALL  SUM@I
;//////////////////////////////////
       BCF   PIR1,ADIF     ;LIMPIA BANDERA FIN DE CONVERSION
       BSF   ADCON0,GO     ;INICIA CONVERSION
ESPERA2   BTFSS PIR1,ADIF      ; espera fin de conversion(6.4uS*12Tad=76.8uS)
       GOTO  ESPERA2
        bsf   STATUS,RP0   ;Selecciona el banco 1
             movf  ADRESL,0     ;LEE dato LSB de conversion UBICADO BANCO 1
        bcf   STATUS,RP0   ;Selecciona el banco 0
        MOVWF Q1_lsb
        movf  ADRESH,0     ;LEE dato MSB de conversion UBICADO BANCO 0
        MOVWF Q1_msb
        CALL  SUM@I
;//////////////////////////////////
       BCF   PIR1,ADIF     ;LIMPIA BANDERA FIN DE CONVERSION
       BSF   ADCON0,GO     ;INICIA CONVERSION
ESPERA3   BTFSS PIR1,ADIF      ; espera fin de conversion(6.4uS*12Tad=76.8uS)
       GOTO  ESPERA3
        bsf   STATUS,RP0   ;Selecciona el banco 1
             movf  ADRESL,0     ;LEE dato LSB de conversion UBICADO BANCO 1
        bcf   STATUS,RP0   ;Selecciona el banco 0
        MOVWF Q1_lsb
        movf  ADRESH,0     ;LEE dato MSB de conversion UBICADO BANCO 0
        MOVWF Q1_msb
        CALL  SUM@I
;//////////////////CALCULA EL PROMEDIO ////////////////////////////////////
        movf  SDEN_2,0
```

```
        MOVWF MSNUM_2
        movf  SDEN_1,0
        MOVWF MSNUM_1
        CLRF MSNUM_3
        MOVLW 0X04      ;CARGA PARTE BAJA DEL DIVISOR (682 DECIMAL)
                MOVWF Q2_lsb
                MOVLW 0X00      ;CARGA PARTE BAJA DEL DIVISOR (682 DECIMAL)
        MOVWF Q2_msb
        CALL  Dividir
        movf  MSNUM_2,0
        MOVWF Q2_msb
        MOVWF CONVER_2   ;GUARDA EL PROMEDIO DE LAS ADQUISICIONES
        movf  MSNUM_1,0
        MOVWF Q2_lsb
        MOVWF CONVER_1  ;GUARDA EL PROMEDIO DE LAS ADQUISICIONES
;**********************************ERROR**************************************
      MOVF  TEC_lsb,0
      MOVWF Q2_lsb
      MOVF  TEC_msb,0
      MOVWF Q2_msb
      call  COMP@
      MOVWF p
      btfsc p,0        ;SALTA SI TEMP CONVERSOR ES <= QUE TECLADO
      goto  NOFUZZY
      MOVF  CONVER_1,0
      subwf TEC_lsb,0
      MOVWF ERRORLCD
      MOVWF Q1_lsb
      clrf  Q1_msb
      clrf  Q2_msb
      MOVLW 0X0B
      MOVWF Q2_lsb
      call  COMP@
      MOVWF p
      btfsc p,0
      goto  PRENDE
      MOVLW 0X40
      MOVWF Q2_lsb
      clrf  Q2_msb
      CALL  M_MSNUM
```

```
      btfsc MSNUM_2,2
      goto  PRENDE
      MOVF  MSNUM_1,0
      MOVWF error_lsb
      MOVF  MSNUM_2,0
      MOVWF error_msb
;*******************************************************************************
;**AJUSTAMOS EL DATO DEL CONVERSOR AL TIPO DE SENSOR 150º MAXIMO*
;CONVIERTE EL DATO DEL CONVERSOR A FORMATO 0-150([Q2_lsb]*150/1023)
      CLRF Q1_msb
      MOVLW 0X96
      MOVWF Q1_lsb
           CALL  M_MSNUM   ;MULTIPLICA DATOA(16)*DATOB(16)
              MOVLW 0XFF     ;CARGA PARTE BAJA DEL DIVISOR (150DECIMAL)
              MOVWF Q2_lsb
      MOVLW 0X03
      MOVWF Q2_msb    ;CARGA PARTE Alta DEL DIVISOR (150 DECIMAL)
      CALL  Dividir    ;divide el dato entre el #150 (RESULTADO EN MSNUM)
      MOVF  MSNUM_1,0
      MOVWF CONVER_1
      MOVWF Q1_lsb
      MOVF  MSNUM_2,0
      MOVWF CONVER_2
      MOVWF Q1_msb
;*** VARERROR=(TEMP ACTUAL)-(TEMP ANTERIOR)*****
      MOVF CONVER_1,0
      MOVWF Q1_lsb
      MOVF CONVER_2,0
      MOVWF Q1_msb
      MOVF  error2,0
      MOVWF Q2_lsb
      CLRF  Q2_msb
      call  COMP@
      MOVWF p
      btfss p,1
      GOTO VARMAYOR ;01
      MOVF  CONVER_1,0
      subwf error2,0
      sublw 0X08
      MOVWF Q1_lsb
```

```
        MOVLW 0X40
        MOVWF Q2_lsb
        CLRF  Q1_msb
        CLRF  Q2_msb
        CALL  M_MSNUM
        MOVF  MSNUM_1,0
        MOVWF varerror_lsb
        MOVF  MSNUM_2,0
        MOVWF varerror_msb
        GOTO  FUZZ
VARMAYOR
        MOVF  error2,0
        subwf CONVER_1,0
        MOVWF Q1_lsb
        CLRF  Q1_msb
        CLRF  Q2_msb
        MOVLW 0X08
        MOVWF Q2_lsb
        call  COMP@
        MOVWF p
        btfsc p,0
        goto  NOFUZZY  ;01
        MOVF  Q1_lsb,0
        ADDLW 0X08
        MOVWF Q1_lsb
        MOVLW 0X40
        MOVWF Q2_lsb
        CLRF  Q1_msb
        CLRF  Q2_msb
        CALL  M_MSNUM
        MOVF  MSNUM_1,0
        MOVWF varerror_lsb
        MOVF  MSNUM_2,0
        MOVWF varerror_msb
FUZZ
        MOVF CONVER_1,0
        MOVWF error2
        goto  FUZZYE3T
NOFUZZY
        movlw   0X25      ;ILUMINACION CERO TIEMPO OFF(8.3mS)
```

```
      movwf   CCPR2L
      MOVWF   PWMLSB
      movlw   0XF8     ;ILUMINACION CERO TIEMPO OFF(8.3mS)
      movwf   CCPR2H
      MOVWF   PWMMSB
      movlw   0X00
      movwf   ERRORLCD
      goto  TERMINAR
PRENDE
      movlw   0X00     ;ILUMINACION CERO TIEMPO OFF(8.3mS)
      movwf   CCPR2L
      MOVWF   PWMLSB
      movlw   0X00     ;ILUMINACION CERO TIEMPO OFF(8.3mS)
      movwf   CCPR2H
      MOVWF   PWMMSB
      goto  TERMINAR
;<<SUBRUTINA FUZZYE3T>>
;***********************************************
;** INSERTAR  AQUÍ  RUTINA  FUZZYE3T     (anexo II)***
;***********************************************
          movf   DNUMDEN_1,0
          movwf  Q2_lsb        ;CARGA ANCHO DE PULSO BAJO
   MOVF   DNUMDEN_2,0    ;CARGA ANCHO DE PULSO ALTO
          movwf  Q2_msb
;*CONVIERTE DATO DE SALIDA (0-1023) A NUEVO FORMATO PWM (2605h)
;**(DNUMDEN_2,DNUMDEN_1)*2605h/1023
      MOVLW 0X26
      MOVWF Q1_msb
      MOVLW 0X05
      MOVWF Q1_lsb
          CALL  M_MSNUM   ;MULTIPLICA DATOA(16)*DATOB(16)
          MOVLW 0XFF      ;CARGA PARTE BAJA DEL DIVISOR (1023DECIMAL)
          MOVWF Q2_lsb
          MOVLW 0X03
      MOVWF Q2_msb    ;CARGA PARTE Alta DEL DIVISOR (1023 DECIMAL)
      CALL  Dividir    ;divide el dato entre el #1023 (RESULTADO EN MSNUM)
      MOVF  MSNUM_1,0
      MOVWF PWMLSB
      MOVF  MSNUM_2,0
      MOVWF PWMMSB
```

```
;*****************************FIN RUTINA FUZZY************************
TERMINAR
;******************MUESTRA DATO DEL CONVERSOR EN EL LCD*****************
            movf  CONVER_1,0
        MOVWF MSNUM_1
        movf  CONVER_2,0
        MOVWF MSNUM_2
;*******************************************************************************
     CALL  Bin_BCD    ;PASA LA PARTE ENTERA DEL DATO A BCD PARA EL LCD
;**MUEVE EL RESULTADO A LAS VARIABLES BCDX1,BCDX10,BCDX100*****
     MOVF     R2,W
     ANDLW    0X0F
     ADDLW    0X30
     MOVWF    BCDX1       ;UNIDADES
     MOVF     R2,W
     ANDLW    0XF0
     MOVWF    BCDX10
     swapf    BCDX10,1
     MOVLW    0X30
     ADDWF    BCDX10,1    ;DECENAS
     MOVF     R1,W
     ANDLW    0X0F
     ADDLW    0X30
     MOVWF    BCDX100     ;CENTENAS
;******************AJUSTE PARTE DECIMAL*******************
          MOVLW 0X92     ;CARGA PARTE ALTA DEL NUMERO 146 DECIMAL
          MOVWF Q1_lsb
     CLRF  Q1_msb  ;CARGA PARTE ALTA DEL NUMERO 146 DECIMAL
     MOVF  RestoLL,0
     MOVWF Q2_lsb      ;LEE RESIDUO LSB
     MOVF  RestoHL,0
     MOVWF Q2_msb      ;LEE RESIDUO MSB
     CALL  M_MSNUM   ;MULTIPLICA DATOA(16)*DATOB(16)
          MOVLW 0X64     ;CARGA PARTE BAJA DEL DIVISOR 100 DECIMAL
          MOVWF Q2_lsb
     CLRF  Q2_msb
     CALL  Dividir    ;divide el dato entre el #100
     CALL  Bin_BCD    ;PASA LA PARTE DECIMAL DEL DATO A BCD PARA EL LCD
;**MUEVE EL RESULTADO A LAS VARIABLES BCDX0,BCDX01,BCDX001*****
     MOVF     R2,W
```

```
ANDLW    0X0F
ADDLW    0X30
MOVWF    BCDX001     ;MILESIMAS
MOVF     R2,W
ANDLW    0XF0
MOVWF    BCDX01
swapf    BCDX01,1
MOVLW    0X30
ADDWF    BCDX01,1    ;CENTESIMAS
MOVF     R1,W
ANDLW    0X0F
ADDLW    0X30
MOVWF    BCDX0       ;DECIMAS
     movlw   0X84        ;POSICION DEL CURSOR 4 RENGLON 1 DISPLAY
     call    CONTROL
     call    delay56
     MOVF    BCDX100,0    ;MUESTRA CENTENAS
     call    DATO
     movlw   0X85        ;POSICION DEL CURSOR 5 RENGLON 1 DISPLAY
     call    CONTROL
     MOVF    BCDX10,0     ;MUESTRA DECENAS
     call    DATO
     movlw   0X86        ;POSICION DEL CURSOR 6 RENGLON 1 DISPLAY
     call    CONTROL
     MOVF    BCDX1,0      ;MUESTRA UNIDADES
     call    DATO
     call    delay56
     call    delay56
     movlw   0X87        ;POSICION DEL CURSOR 7 RENGLON 1 DISPLAY
     call    CONTROL
     MOVLW   0X2E          ;(.)
     call    DATO
     movlw   0X88        ;POSICION DEL CURSOR 8 RENGLON 1 DISPLAY
     call    CONTROL
     MOVF    BCDX0,0      ;MUESTRA DECIMAS
     call    DATO
     movlw   0X89        ;POSICION DEL CURSOR 9 RENGLON 1 DISPLAY
     call    CONTROL
     MOVLW   0XDD     ;(º)
     call    DATO
```

```
        movlw   0X8A        ;POSICION DEL CURSOR 10 RENGLON 1 DISPLAY
        call    CONTROL
        MOVLW   0X43        ;(C)
        call    DATO
        movlw   0X8B        ;POSICION DEL CURSOR 11 RENGLON 1 DISPLAY
        call    CONTROL
        MOVLW    0X20
        call    DATO
;*****************************************************************************
;CONVIERTE EL DATO DEL TECLADO A FORMATO 0-150([Q2_lsb]*15/1023)
        movf  TEC1_LCD,0
    MOVWF Q2_lsb
          movf  TEC2_LCD,0
    CLRF  Q2_msb
    MOVLW 0X03
    MOVWF Q1_msb
    MOVLW 0XFF
    MOVWF Q1_lsb
        CALL  M_MSNUM   ;MULTIPLICA DATOA(16)*DATOB(16)
        MOVLW 0X96      ;CARGA PARTE BAJA DEL DIVISOR (150DECIMAL)
        MOVWF Q2_lsb
    CLRF  Q2_msb    ;CARGA PARTE Alta DEL DIVISOR (150 DECIMAL)
    CALL  Dividir    ;divide el dato entre el #150 (RESULTADO EN MSNUM)
    MOVF  MSNUM_1,0
    MOVWF TEC_lsb
    MOVWF Q1_lsb
    MOVF  MSNUM_2,0
    MOVWF TEC_msb
    MOVWF Q1_msb
    movf  TEC1_LCD,0
    MOVWF MSNUM_1
    movf  TEC2_LCD,0
    MOVWF MSNUM_2
    CLRF  MSNUM_3
    CALL  Bin_BCD    ;PASA LA PARTE ENTERA DEL DATO A BCD PARA EL LCD
;**MUEVE EL RESULTADO A LAS VARIABLES BCDX1,BCDX10,BCDX100*****
    MOVF     R2,W
    ANDLW    0X0F
    ADDLW    0X30
    MOVWF     BCDX1       ;UNIDADES
```

```
      MOVF     R2,W
      ANDLW    0XF0
      MOVWF    BCDX10
      swapf    BCDX10,1
      MOVLW    0X30
      ADDWF    BCDX10,1    ;DECENAS
      MOVF     R1,W
      ANDLW    0X0F
      ADDLW    0X30
      MOVWF    BCDX100     ;CENTENAS
      movlw  0XC4            ;POSICION DEL CURSOR 4 RENGLON 2 DISPLAY
     call   CONTROL
          MOVF   BCDX100,0     ;MUESTRA CENTENAS
          call   DATO
          movlw  0XC5          ;POSICION DEL CURSOR 5 RENGLON 2 DISPLAY
          call   CONTROL
          MOVF   BCDX10,0      ;MUESTRA DECENAS
          call   DATO
          movlw  0XC6          ;POSICION DEL CURSOR 6 RENGLON 2 DISPLAY
          call   CONTROL
          MOVF   BCDX1,0       ;MUESTRA UNIDADES
          call   DATO
          movlw  0XC7          ;POSICION DEL CURSOR 7 RENGLON 2 DISPLAY
          call   CONTROL
          MOVLW    0X20
          call   DATO
          movlw  0XC8          ;POSICION DEL CURSOR 8 RENGLON 2 DISPLAY
          call   CONTROL
          MOVLW    0X20
          call   DATO
;**************************Rutina Refresco Lcd*****************
   DECFSZ   REFRESCO
   GOTO    RFRES1
          movlw  01H         ;BORRA DISPLAY
   call   CONTROL
   MOVLW  0X0F
   MOVWF  REFRESCO
RFRES1
;*****************************************************************************
   BCF    PIR2,0      ;limpia bandera señalizacion int comparación modulo 2
```

```
 BSF     STATUS,RP0   ;BANCO 1
BSF     PIE2,0       ;activa interrupción por comparación modulo 2
clrf    PIE1         ;desactiva todas las interrupciones de este registro A/D
BCF     STATUS,RP0   ;BANCO 0
movlw   b'11010000'    ;habilitación todas las interrupciones,
 movwf   INTCON         ;bit4 intr ext activada
GOTO BIG_LOOP       ;SALTA PARA COMENZAR UNA NUEVA CONVERSION  Y MUESTRA LCD
    END
```

8.20. IV.3. MICROCONTROLADOR PIC 16F873.

Este programa realiza la adquisición del dato de temperatura y mediante comunicación serie RS-232 es enviado al computador parta ejecutar el control Fuzzy de la aplicación este retorna al microcontrolador el dato del ancho de pulso necesario para ajustar la rutina de PWM y su posterior envio al elemento calefactor.

```
;*** FORMATO DE  PROGRAMA  MPLEMENTACION  FUZZY EN EL PC,**********
;*** EL PIC16F873 REALIZA CONVERSIÓN, TX SACA PULSO PIN RC1***********
;*** RECIBE DATO PARA PWM PRODUCCIDO EN EL CCP2 POR INTERRUPCION**
;*****************5  MHz ************************************************
            LIST    P=16f873
            INCLUDE  "P16F873.INC"
      cblock  0x20
Q1_lsb
Q1_msb
Q2_lsb
Q2_msb
PWMLSB
PWMMSB
SERIALA
ADQ_N
POTENCIA_1
POTENCIA_2
WORK
MULTI_1
MULTI_2
MULTI_3
R0C ;variables lcd
R0D ;variables lcd
R0E   ;variables lcd
W1    ;variables lcd
temp ;variables lcd
       endc
;**************************************
            ORG  0x00
            goto INICIO
            ORG  0x04
            goto INTER
;*****************************************************************
;*RUTINA DE INICIALIZACION DE LCD*
LCD_INIT   movlw   b'00000010'   ;LIMPIA  HOME
        call    CONTROL
```

```
            movlw   28H        ;BUS DE DATOS (4 BITS) (2 LINEA EN PANTALLA) TAMAÑO CARACTER 5*7
        call    CONTROL
            movlw   0CH         ; PANTALLA ACTIVA Y CURSOR NO PARPADEA
        call    CONTROL
            movlw   06H         ; PANTALLA ACTIVA Y CURSOR NO PARPADEA
        call    CONTROL
        CLRF    PORTB
        RETURN
;*****************************************************
TABLA_LCD ADDWF 2,1
    RETLW " "
    RETLW "C"
    RETLW "O"
    RETLW "N"
    RETLW "T"
    RETLW "R"
    RETLW "O"
    RETLW "L"
    RETLW " "
    RETLW "D"
    RETLW "I"
    RETLW "F"
    RETLW "U"
    RETLW "S"
    RETLW "O"
    RETLW " "
;=========================
COMP@   movf    Q2_msb,0
            subwf   Q1_msb,0
            btfss   STATUS,0
            goto    Q1MENOR
            btfss   STATUS,2
            goto    Q1MAYOR
            movf    Q2_lsb,0
            subwf   Q1_lsb,0
            btfss   STATUS,0
            goto    Q1MENOR
            btfss   STATUS,2
            goto    Q1MAYOR
            goto    Q1IGUAL
Q1MENOR  retlw   0x02
Q1MAYOR  retlw   0x01
Q1IGUAL  retlw   0x00
;*****************************************************
;Se transmite serialmente el dato que está en el registro W
```

```
TXDAT   bcf   PIR1,TXIF    ;Restaura flag del transmisor
            movwf TXREG       ;Mueve el byte a transmitir al registro de transmisión
            bsf   STATUS,RP0   ;Selecciona el banco 1
            bcf   STATUS,RP1
TXDATW  btfss TXSTA,TRMT   ;TRMT=1 el byte se transmitio, salta
            goto  TXDATW       ;No se transmitio el byte, vuelve a esperar
            bcf   STATUS,RP0   ;Si se transmitio el byte, vuelve al banco 0
      return
;*************************************************
;**RUTINAS ESCRITURA LCD****
RETAR   movlw  0xff  ;RUTINA DE RETARDOS NECESARIOS
            movwf  temp
DECRE   decfsz temp,1
            goto   DECRE
            return
CONTROL bcf   PORTA,2    ;RUTINA GENERA LAS SEÑALES DE CONTROL PARA ESCRIBIR EN EL LCD
            goto  DATO2
DATO    bsf   PORTA,2
DATO2   bCf    PORTA,5
      movwf  R0E
            movwf  W1
      SWAPF  W1,1
      RLF    W1,0
            andlw  0x0E      ;GARANTIZA QUE LOS BITS 7-4 Y 0 PTB SEAN CERO
      MOVWF  PORTB     ;SACA PARTE ALTA DEL RESULTADO
      BTFSC  R0E,7
      GOTO   S1
      BCF    PORTA,4
      GOTO   ACTIVA1
S1     BSF    PORTA,4
ACTIVA1 call   RETAR
      BSF    PORTA,5
            call   RETAR
      call   RETAR
      BCF    PORTA,5
      RLF    R0E,0
            andlw  0EH     ;GARANTIZA QUE LOS BITS 7-4Y0 PTB SEAN CERO
      MOVWF  PORTB     ;SACA PARTE BAJA DEL RESULTADO
      BTFSC  R0E,3
      GOTO   S2
      BCF    PORTA,4
      GOTO   ACTIVA2
S2     BSF    PORTA,4
ACTIVA2 call   RETAR
      BSF    PORTA,5
```

```
            call   RETAR
      call   RETAR
      BCF    PORTA,5
            RETURN
SIGLO      movlw   0XFF        ;RETARDO ENTRE CARACTERES
        movwf   R0D
RETA1      call    RETAR
        decfsz  R0D,1
        GOTO    RETA1
      RETURN
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>>>>>>ATENCION A INTERRUPCION;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;pregunta por el origen de la interrupcion
INTER   MOVWF WORK
      btfsc INTCON,INTF ;INTF=1 interrupción señal ext(cruce por cero)
      goto  CRUCE0
      btfsc PIR1,RCIF  ;RCIF=1 interrupción por recepción,salta A RECIBE
      goto  RECIBE
      btfsc   PIR2,0   ;CCP2IF=1 BANDERA INTE por COMPARACION
      goto    COMPAR
            goto  EXIT1      ;No hay interrupción, retorna
;*********************************************************
RECIBE   movf   RCREG,0
            movwf  SERIALA
A_CERO
            movlw  0x00
            xorwf  ADQ_N,0
            btfss  STATUS,2
            goto   A_UNO
            movf   SERIALA,0
            movwf  PWMLSB
            goto   SALIR
A_UNO
             movf   SERIALA,0
             movwf  PWMMSB
       BCF   PIR1,ADIF      ;LIMPIA BANDERA FIN DE CONVERSION
       BSF   ADCON0,GO      ;INICIA CONVERSION
ESPERA   BTFSS PIR1,ADIF      ; espera fin de conversion(6.4uS*12Tad=76.8uS)
       GOTO  ESPERA
                movf  ADRESH,0       ; transmite dato MSB de conversion
       call  TXDAT         ;1.25mS duracion de transmisión 10 bits
       bsf   STATUS,RP0     ;Selecciona el banco 1
                movf  ADRESL,0
       bcf   STATUS,RP0     ;Selecciona el banco 0
       call  TXDAT         ;; transmite dato LSB de conversion
```

```
            movf   PWMLSB,0
            movwf  Q2_lsb        ;CARGA ANCHO DE PULSO BAJO
      MOVF   PWMMSB,0    ;CARGA ANCHO DE PULSO ALTO
            movwf  Q2_msb
      MOVLW 0Xf8
      MOVWF Q1_lsb
      MOVLW 0X25
      MOVWF Q1_msb
      CALL  COMP@
      XORLW 0X00
      BTFSS STATUS,Z    ;SALTA SI ES IGUAL 25f8
      GOTO  SALIR
      BCF     T1CON,0   ;APAGA TIMER 1
      CLRF    CCP2CON
      BCF     PORTC,1
      clrf    TMR1L
      clrf    TMR1H
      BCF     PIR2,0    ;limpia bandera señalizacion int comparación
;*****************************************************************
SALIR       bcf PIR1,RCIF     ;limpia bandera señalizacion int recepción
            incf  ADQ_N,1      ;incrementa el número de adquisición serial
            movlw 0x02         ;si es igual a 2 la reinicializa a 0
            xorwf ADQ_N,0
            btfss STATUS,2
            goto  EXIT1
            clrf  ADQ_N
      bsf     INTCON,INTE   ;activa interrupcion externa
      goto  EXIT1
;====================
COMPAR BCF     T1CON,0   ;APAGA TIMER 1
      CLRF    CCP2CON
      BCF     PORTC,1
      clrf    TMR1L
      clrf    TMR1H
      MOVF    PWMLSB,0
      MOVWF   CCPR2L
      MOVF    PWMMSB,0
      MOVWF   CCPR2H
      BCF     PIR2,0    ;limpia bandera señalizacion int comparacion
      MOVLW   0X08      ;MODULO CP2 EN MODO COMPARACION ACTIVA
      MOVWF   CCP2CON   ;PIN RC1 CUANDO COINCIDEN VALORES TMR1 Y CCPR2H
      BSF     STATUS,RP0   ;BANCO 1
      BSF     PIE2,0    ;HABILITA INTERUPCION COMPARACION
      BCF     STATUS,RP0   ;BANCO 0
```

```
      goto  EXIT1
;****************************************************************
CRUCE0   MOVLW   0X08     ;MODULO CP2 EN MODO COMPARACION ACTIVA PIN RC1
       MOVWF   CCP2CON   ;CUANDO COINCIDEN VALORES TMR1 Y CCPR2H-L
       BSF     T1CON,0      ;HABILITA CONTADOR TMR1
       BCF     PIR2,0    ;LIMPIA BANDERA
       BSF     STATUS,RP0   ;BANCO 1
       BSF     PIE2,0    ;HABILITA INTERUPCION COMPARACION
       BCF     STATUS,RP0   ;BANCO 0
       bcf     INTCON,INTF   ;limpia bandera señalizacion interrupcion externa
EXIT1    MOVF    WORK,0
       retfie         ;retorna
;*******************************************************************************
INICIO  clrwdt              ;refresca el perro guardián
          clrf       PORTA     ;pone todos los bits del puerto A 0
          clrf       PORTB     ;pone todos los bits del puerto B a 0
          clrf       PORTC     ;Direcciona con A2 A1 A0 ningún elemento
         call  SIGLO
         call  SIGLO
;*********************************************************
;se entra al modo de configuración del PIC
          bsf     STATUS,RP0     ;Cambia al banco 1
          bcf     STATUS,RP1     ;RP0=1 y RP1=0
;puerto A como ENTRADA ANALOGICA AL CONVERSOR
      movlw   b'10001110'   ;BIT7=1 JUST DERECHA, RA0 entrada conversor
      movwf   ADCON1
      BCF     STATUS,RP0     ;Cambia al banco 0
      movlw   b'00000001'   ;frec/2, canal 0 ,activa conversor pero
      movwf   ADCON0         ; no inicia conversion
          bsf     STATUS,RP0     ;Cambia al banco 1
          movlw   b'00000011'   ;RAO entrada A/D,RA1(setpoint), RA RESTANTES salida
          movwf   TRISA
;puerto C
;se programa la comunicación serial
          movlw   b'10000000'   ;RC7/RX como entrada y RC6/TX como salida
          movwf   TRISC         ;RC0-5 como salidas RC1 (PWM)
      BCF     PORTC,1       ;BOMBILLO OFF
;puerto B como entrada
          movlw   b'11110001'   ;RB7-4 entrada(teclado),RB3-RB1 salida(teclado),
          movwf   TRISB         ;RB0 entrada(cruce0)
          movlw   b'11000000'   ;BITS0-2=1 DIV TMR0 1:256
                      ;BIT3=0 DIV FREC PARA TMR0, NO PARA WDT
                      ;BIT4=0 INC TMR0 FLANCO DESCENDENTE (TOCK1)
                      ;BIT5=0 PULSOS TMR0 DE RELOJ INTENO FOSC/4(TEMPORIZADOR),NO TOCK1
```

```
                    ;BIT6=0 FLANCO DESCENDENTE DE INT EXTERNA
                    ;BIT7=1 RESISTENCIA PULL-UP DE PUERTA B DESACTIVADAS
    movwf   OPTION_REG
    clrf    PIE1
;*************************************
;CONFIGURACION DE A/D:se realizo en configuración pta
;CONFIGURACION DE option:se realizo en configuración ptb
;*************************************
;CONFIGURACION DE RECEPCION  DE DATOS
        movlw   b'00100100'   ;modo asíncrono,SYNC=0, dato de 8 bits
        movwf   TXSTA         ;y alta velocidad BRGH=1
        movlw   d'31'         ;
        movwf   SPBRG         ;Registro SPBRG=32 para trabajo a 9600 baudios 5 Mhz
        bsf     PIE1,RCIE     ;habilita interrupción en recepción
        bcf     STATUS,RP0    ;cambio al banco 0
        movlw   b'10010000'   ;configuración del USART para recepción continua
        movwf   RCSTA         ;
;*************************************
;*CONFIGURACION DEL PWM ccp2-pin RC1;ccp1-pin rc2*
;*PARA GENERAR EL PWM SE UTILIZA EL TIMER 1 EN MODO COMPARACION*
;SE CARGA EN LOS REGISTROS CCPR2H-L EL CONTENIDO DEL TIEMPO OFF*
;LUEGO DE ESTE TIEMPO EL PIN RC1 PASA A ON (ANCHO DE PULSO)*
;CCPR2H-L=(8.33mS-ANCHO DE PULSO)
    BCF     STATUS,RP0  ;Cambia al banco 0
        clrf    PORTA     ;pone todos los bits del puerto A 0
        clrf    PORTB     ;pone todos los bits del puerto B a 0
        clrf    PORTC     ;pone todos los bits del puerto C a 0
        clrf    T1CON     ;APAGA TIMER 1
    MOVLW   0X08     ;MODULO CP2 EN MODO COMPARACION ACTIVA
    MOVWF   CCP2CON   ;PIN RC1 CUANDO COINCIDEN VALORES TMR1 Y CCPR2H-L
    clrf    TMR1L
    clrf    TMR1H
        movlw   0X05     ;ILUMINACION CERO TIEMPO OFF(8.3mS)
        movwf   CCPR2L
        movlw   0X26     ;ILUMINACION CERO TIEMPO OFF(8.3mS)
        movwf   CCPR2H
    bcf     PIR2,0
;*********************************************************
;inicialización de variables
        movlw   .1
        movwf   SERIALA
        clrf    ADQ_N
        call       LCD_INIT
;*************************************
;*MUESTRA MENSAJE TEMPERATURA****
```

```
;**************************************
          movlw   01H         ;BORRA DISPLAY
       call    CONTROL
MUESTRA                clrf    R0C         ;INICIA CONTADOR DE TABLA
CICLO     movf    R0C,W       ;HACE BARRIDO DE TABLA
       call    TABLA_LCD
       call    DATO
       incf    R0C,1
       movlw   0FH
       XORWF   R0C,W      ;PREGUNTA POR FIN DE LECTURA DE TABLA
       btfss   STATUS,Z
       GOTO    CICLO
            movlw   0XC7         ;POSICION DEL CURSOR 9 RENGLON 2 DISPLAY
       call    CONTROL
            movlw   "P"          ;POSICION DEL CURSOR 9 RENGLON 2 DISPLAY
       call    DATO
            movlw   "C"          ;POSICION DEL CURSOR 9 RENGLON 2 DISPLAY
       call    DATO
;=====================================================================
         movlw   b'11000000'    ;habilitación todas las interrupciones,bit6 int perifericos
         movwf   INTCON         ;bit4 intr ext desactivada
;*****************************************************
;El siguiente es el lazo en el que se quedara el PIC normalmente
;en caso de no producirse recepción o transmisión serial con el computador
;en este lazo lee y actualiza el tablero
LEER
     NOP
     btfss   PIR2,0   ;CCP2IF=1 BANDERA INTE por COMPARACION
     goto    LEER
     NOP
     BCF     T1CON,0   ;APAGA TIMER 1
     CLRF    CCP2CON
     BCF     PORTC,1
     clrf    TMR1L
     clrf    TMR1H
     BCF     PIR2,0    ;limpia bandera señalizacion int comparacion
     MOVLW   0X08      ;MODULO CP2 EN MODO COMPARACION ACTIVA
     MOVWF   CCP2CON   ;PIN RC1 CUANDO COINCIDEN VALORES TMR1 Y CCPR2H-L
     goto    LEER
END
```