METODOLOGÍA PARA LA IMPLEMENTACIÓN DE CONTROLADORES REGULATORIOS DIFUSOS TIPO TAKAGI-SUGENO EN SISTEMAS MICROCONTROLADOS

David Patricio Paredes Flor

Facultad de Ingeniería Electrónica, Escuela Politécnica del Ejercito

Av. El Progreso S-N, Sangolquí, Ecuador.

d.paredes.f@hotmail.com

RESUMEN.- El presente documento engloba los conceptos de lógica difusa, aplicados en una metodología para la implementación de las diferentes partes que conforman un controlador difuso tipo Takagi-Sugeno en sistemas microcontrolados, empezando por la definición de las variables involucradas, fuzzificación de las entradas, el proceso de razonamiento en el motor de inferencia difusa y el cálculo de la salida. Aplicable a cualquier modelo de microcontrolador que cumpla con ciertos requisitos básicos como lo son los puertos de E/S necesarios, memoria, velocidad, comunicaciones, etc.

PALABRAS CLAVE.- Lógica difusa, controladores difusos, fuzzificación, métodos de implicación, base de reglas, motor de inferencia, microcontrolador, sistema embebido.

I. INTRODUCCIÓN

En la actualidad los controladores difusos se han convertido en una opción fundamental al momento de implementar sistemas de control, permitiendo manejar un sinnúmero de plantas y evitando las complicaciones matemáticas propias de un desarrollo clásico de controladores. Así mismo los sistemas microcontrolados se han abierto camino en las aplicaciones de control a manera de sistemas embebidos, como lo es por ejemplo un controlador regulatorio de propósito general de dos entradas para el cual va dirigida esta metodología, esto es debido a su capacidad de procesar señales análogas y digitales así como la velocidad de procesamiento que puede superar los 40Mhz.

II. ESPECIFIACIONES DEL MICROCONTROLADOR

La base para seleccionar el microcontrolador adecuado se muestra en la siguiente tabla.

Parámetro	Especificación
Entradas	Puertos ADC necesarios en función del número de las variables del proceso a ser medidas, así mismo, los puertos digitales necesarios según los requerimientos
Salidas	Tendrá que tener los pines PWM necesarios para generar las señales de control analógicas

Velocidad	>= 20Mhz (Recomendado)
Comunicaciones	Serial RS232 o USB, I2C
Memoria de programa	>= 24 KB (Recomendado)
RAM	>= 2 KB (Recomendado)

Tabla 1: Especificaciones del microcontrolador.

En cuanto a la velocidad, existen microcontroladores capaces de trabajar a una frecuencia máxima de 48Mhz, como es el caso de la serie de los PIC´s 18F4x5x y los 18F4x5x, ideales para este tipo de aplicaciones, ya que además soportan comunicación USB 2.0 y RS232, con lo cual se hace más fácil el seteo de parámetros mediante una interface de configuración. Además un requisito no indispensable pero si útil es la comunicación I2C, con el objetivo de leer/escribir los datos de las variables en una memoria EEPROM externa como por ejemplo una 24C08 sin necesidad de volver a grabar el microcontrolador con las variables.

III. ESPECIFICACIONES DEL SISTEMA CONTROLADOR

Como base para el desarrollo de la metodología se escogió un sistema difuso PD tipo Takagi-Sugeno de dos entradas, con regulación de universos de discurso para las variables, límites de las funciones de pertenencia, edición de base de reglas y valores de los singletons de salida, el esquema básico se muestra en la figura 1.

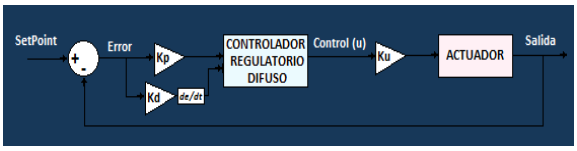

Figura 1: Esquema del sistema controlador.

Se define que las funciones de pertenencia son de tipo triangular o trapezoidal debido a su bajo coste computacional al momento de su evaluación.

IV. METODOLOGÍA DE DESARROLLO

A diferencia de los sistemas difusos tipo Mamdani, los sistemas Takagi-Sugeno no tienen el defuzzificador, en vez de este se realiza el cálculo de la salida en función de las variables de entrada. A continuación se muestra la metodología de implementación de las partes de este controlador difuso.

a. Definición de variables.

Las variables necesarias para el controlador difuso son las siguientes, donde las constantes N, N1, N2 pueden tomar cualquier valor.

Variables	Tipo	Rango
ne	int	(1 a 2)
nf	int	(1 a N)
nl	int	(1 a 4)
nr	int	(1 a ne*nf)
ns	int	(1 a N1)
nf_sel	int	(1 a N2)
Ent_[ne]	long	(0 a 1023)
Act_[ne][nf]	boolean	(true - false)
Eval_[ne][nf]	float	(0 a 1)
Lim_[nf_sel][nl]	signed long	(-1023 a 1023)
Reglas_[nr]	int	(1 a ns)
Imp_[nr]	boolean	(true - false)

Sing_[ns]	signed long	(-1023 a 1023)
WS_[ns]	float	(0 a 1)
pesoTemp	float	(0 a 1)
num	float	(0 a 1023)
den	float	(0 a 1)
crisp	signed long	(-1023 a 1023)

Tabla 2: Definición de variables

Donde, ***ne*** es el número de entradas que en este caso son 2, ***nf*** es el número de funciones de pertenencia por entrada las cuales son 5 y N puede tomar cualquier valor, ***nl*** es el número de límites para cada función de pertenencia que es igual a 4, ***nr*** número de reglas que es igual a ne*nf, ***ns*** número de singletons de salida que son 5, ***nf_sel*** es el número de función de pertenencia asociada a la entrada ne y al índice nf, **Ent_**[ne] almacena cada una de las entradas del sistema, ***Act_***[ne][nf] almacena qué funciones de pertenencia se activaron (valor lógico verdadero sí se activó y falso de lo contrario), este facilita posteriormente la evaluación de las reglas del sistema difuso, ***Eval_***[ne][nf] almacena los valores de la evaluación de las funciones de pertenencia o la certeza de cada valor lingüístico, ***Lim_***[nf_sel][nl] almacena los valores de los límites para cada función de pertenencia referente a cada entrada, ***Reglas_***[nr] almacena el valor de posición del singleton de salida para cada condición en la base de reglas, ***Impl_***[nr] almacena el método de implicación para cada regla, **Sing_**[ns] contiene las posiciones de los singletons (constantes) de las funciones de la salida, algunos posibles valores para los singletons de salida se muestran en la tabla 3.

	Índice	**Variable Ling.**	**Valor**
Sing_[]	0	GN	-1023
	1	N	-400
	2	Z	0
	3	P	400
	4	GP	1023

Tabla 3: Posibles valores para el singleton de salida.

WS_[ns] contiene los pesos calculados de cada una de las salidas, ***pesoTemp*** variable temporal almacena valores intermedios en la asignación de los valores de WS_[], ***num*** y ***den*** son las variables del numerador y denominador para el cálculo de la salida, ***crisp*** devuelve el valor de la salida del controlador difuso.

Se utiliza el conversor análogo digital del microcontrolador trabajando con una resolución de 10 bits, de modo que los valores digitales estarán el en rango de 0 a 1023 siendo su equivalente en valor analógico de 0 a 5 Vdc, así mismo las salidas de control están en el mismo rango, con signo positivo y negativo.

b. Definición y evaluación de las funciones de pertenencia.

Dado que las funciones de pertenencia permitidas son de tipo trapezoidal y triangular, implica que los límites para cada función de pertenencia tienen 4 valores como se muestra en la siguiente gráfica.

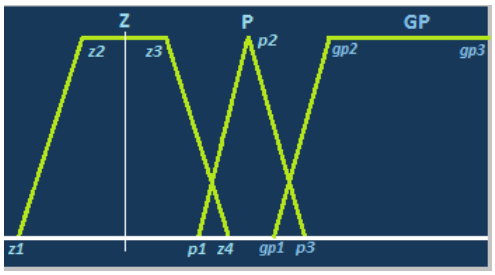

Figura 2: Funciones de pertenencia permitidas.

Donde para la primera función de pertenencia (Z) que es de tipo trapezoidal los límites son (z1, z2, z3, z4), para la segunda (P) de tipo triangular los límites son (p1, p2, p2, p4), nótese que p2 se repite dos veces, justamente eso permite identificar el tipo de función de pertenencia que se está tratando, finalmente para el tercer tipo de función de pertenencia es de tipo trapecio rectángulo (GP), donde los límites son (pg1, gp2, gp3, gp3), con los valores de los límites de las funciones de pertenencia se sabe su tipo y se especifica el algoritmo para evaluar dicha función, este algoritmo queda a criterio del lector, esta función la llamaremos ***evalFunPert()***.

En primer lugar se determina que funciones de pertenencia se han activado y su valor de pertenencia, primero, para determinar si el valor está dentro de los límites mínimo y máximo de dicha función de pertenencia se trabaja iterando ***nf*** en función de ***ne***, obteniendo el número de la función de pertenencia de referencia **nf_sel**, con este valor obtenemos los valores del vector ***Lim_***[nf_sel][nl] e iterándolos los almacenamos en el vector temporal ***vec***[], este entra como parámetro a la función ***evalFunPert()***, la cual determina qué tipo de función de pertenencia es y la evalúa devolviendo su valor, esto siempre y cuando se determine que se activó la función de pertenencia en curso, este valor booleano de respuesta se asigna en el vector Act[ne][nf].

```
long min, max;
long vec[nl];
for(int i=0; i<ne; i++){
  Ent_[i] = leer_adc(i);
  Long valor = Ent_[i];
  for(int j=0; j<nf; j++){
    nf_sel = i*j;
    for(int k=0; k<nl; k++){
        vec[k] = Lim_[nf_sel][k];
    }
    min = Lim_[nf_sel][0];
    max = Lim_[nf_sel][3];
    Act_[i][j]=estaEntre(min,max, valor);
    If(Act_[i][j]){
     Eval_[i][j]=evalFunPert(vec_,valor);
    } else {
      Eval_[i][j] = 0.00;
    }
  }
}
```

c. Definición de la base de reglas y métodos de implicación.

De igual forma para definir la base de reglas se itera ***nf*** en función de ***ne***, obteniendo la variable ***nr***, y seteando el valor deseado del singleton de salida en el vector ***Reglas_***[nr]. Un ejemplo de la base de reglas se muestra en la siguiente figura.

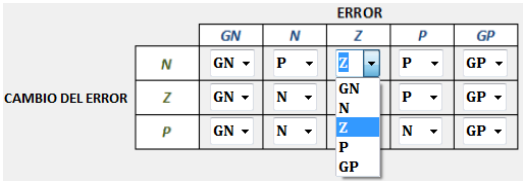

Figura 3: Base de reglas.

Donde para cada combinación de funciones de pertenencia activadas en función de las entradas se asigna un valor del vector Sing_[].

Los métodos de implicación son los operadores lógicos que relacionan una condición con otra, en otras palabras son los encargados de unir los valores fuzzificados de las entradas para generar las reglas y posteriormente las salida, en general las reglas tienen la siguiente forma:

SI Ent_[a] es FP1 Y Ent_[b] es FP2 OR Ent_[a] es FP2 Y Ent_[b] es FP2 ENTONCES DU es S [c].

Las uniones Y generalmente se aplican seleccionado el mínimo de los valores de pertenencia obtenidos al evaluar las entradas, mientras que en las uniones O se aplica el máximo entre los mínimos obtenidos por la uniones Y. Las uniones Y y O se usan para definir las reglas difusas. Al final de las reglas, el consecuente está asociado con una función de pertenencia de la salida.

Con esto se define el vector ***Imp_***[nr] para cada regla donde si es true se aplica el operador lógico AND, caso contrario se aplica OR.

Una vez calculados los vectores Act_[][] y Eval_[][], la base de reglas Reglas_[nr] y definidos los métodos de implicación para cada regla pasamos al motor de inferencia difusa.

d. Motor de inferencia difusa.

El motor de inferencia difusa trabaja en conjunto con la base de reglas, teniendo los diferentes niveles o valores de pertenencia arrojados por el fuzzificador, los mismos deben ser procesados para generar una salida difusa. Es decir, la tarea del motor de inferencia es tomar los niveles de pertenencia y apoyado en la base de reglas, generar la salida del sistema difuso. El algoritmo para el motor de inferencia difusa se muestra a continuación.

```
int r=0;
pesoTemp=0;
for(int i=0; i<nf; i++){
  for(int j=0; j<nf; j++){
    // Verif. si están activadas
    if (Act_[1][i] && Act_[0][j]) {
     //Verif. el método de implicación
     if(Imp_[r]){
      //Obtenemos el menor valor (AND)
       if (Eval_[0][i] > Eval_[0][j]) {
          pesoTemp = Eval_[0][j];
        } else {
          pesoTemp = Eval_[0][i];
        }
     } else {
      //Obtenemos el mayor valor (OR)
       if (Eval_[0][i] < Eval_[0][j]) {
          pesoTemp = Eval_[0][j];
        } else {
          pesoTemp = Eval_[0][i];
        }
    }

     // Comparamos el peso calculado con
        el peso histórico de dicha salida,
        hasta obtener el mayor

      if (WS_[Reglas_[r]] < pesoTemp) {
        WS _[Reglas [r]] = pesoTemp;
      }
     //Enceramos el peso temporal
     pesoTemp = 0;
     //Contador incremental para saltar
     de regla en regla
     r = r + 1;
    }
}
```

Con el algoritmo anterior, en base a los valores de evaluación de las funciones de pertenencia con el vector Eval_[], más el valor de sigleton de salida para cada regla en el vector Reglas_[], ya podemos obtener el peso de cada singleton de salida; lo que resta es simplemente calcular la salida de control del sistema.

e. Cálculo de la salida.

La salida del controlador regulatorio Takagi-Sugeno se calcula de la siguiente forma.

$$salida = \frac{\sum_{i=1}^{ns} WS\_[i].Sing\_[i]}{WS\_[i]}$$

Es decir, la sumatoria de los pesos multiplicado por su respectivo singleton de salida dividido para los pesos. Esta sumatoria, se aplica en el siguiente algoritmo.

```
num = 0;
den = 0;
crisp = 0;

//Bucle en función del número de singletons de salida
for (i = 0; i <= ns; i++) {
  num = num + WS_[i]*Sing_[i];
  den = den + WS_[i];
}
//Obtenemos la salida
crisp = num/den;
```

V. ANÁLSIS DE RESULTADOS

La metodología antes mostrada se aplicó en el desarrollo de un sistema regulatorio difuso, para el control de posición de pedales de automóvil para personas discapacitadas, luego de haber aplicado los algoritmos en el microcontrolador, fue necesaria la creación de una interface de configuración, la cual fue hecha en Visual Basic 2010.

Posterior al seteo de parámetros de control, ajustes y calibraciones, se obtuvo la óptima respuesta del sistema controlador, mostrada en la siguiente figura. Dond la señal de setpoint es de color naranja, mientras que la señal de respuesta del sistema está en color verde

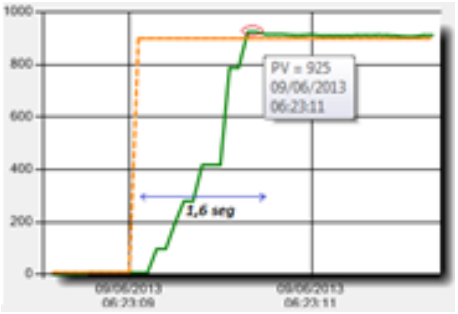

Figura 4: Desempeño del sistema regulatorio.

De la gráfica anterior, se obtienen los siguientes resultados

Parámetro	Valor
Tiempo de establecimiento	1,6 seg
Máximo pico	25 (2,44 %)
Error en estado estacionario	13 (1,27 %)

Tabla 4: Resumen de parámetros de respuesta del sistema.

Analizando el máximo pico, el error en estado estacionario y el tiempo de establecimiento del sistema, se puede

decir que el controlador regulatorio difuso implementado en un microcontrolador, con los ajustes y calibraciones adecuadas, tiene un óptimo desempeño.

VI. CONCLUSIONES

- Luego de estudiar los algoritmos de lógica difusa, se desarrolló esta metodología orientada a sistemas difusos tipo Takagi-Sugeno, ya que al ser una versión más compacta y eficiente de los sistemas tipo Mamdani, computacionalmente hablando, se obtiene un bajo coste computacional en especial al momento de calcular la salida.
- Se logró mediante esta metodología implementar un sistema de control regulatorio difuso tipo Takagi-Sugeno en un microcontrolador, demostrando que no necesariamente estos algoritmos de lógica difusa son algoritmos a ser aplicados en computadoras, más, se demostró el desempeño del mismo en un sistema microcontrolado de posición de pedales.

VII. RECOMENDACIONES

- Al utilizar esta metodología en sistemas microcontrolados, se recomienda su aplicación metódica, programando y probando cada una de las partes antes detalladas.
- Al ser aplicado en un sistema microcontrolado, se recomienda separar la parte lógica encargada de los cálculos, de la parte de potencia encargada de gobernar los actuadores en función de las señales de control, eliminando al máximo posibles fuentes de ruido que puedan afectar el desempeño del controlador

REFERENCIAS

[1] Spartacus Gomáriz, Domingo Biel. "Teoría de Control Diseño Electrónico", Diseño de sistemas de control de tiempo continuo y discreto, Ch. 3, pp. 132, 1998.

[2] Kevin M. Passino. "Fzzy Control", Fuzzy Control: The Basics, Ch. 2, pp. 51, 1998.

[3] Barragán P. Antonio. "Sistemas de Control Borroso estables por Diseño". Fundamentos de los sistemas borrosos, Ch. 2, pp. 23, 2011.

[4] Kevin M. Passino. "Fzzy Control", Fuzzy Control: Takagi-Sugeno Fuzzy Systems, Ch. 2, pp. 73, 1998.

[5] Kamyar Mehran, "Takagi-Sugeno Fuzzy Modeling for Process Control", Takagi-Sugeno fuzzy modeling, Ch. 2, pp. 2, 2008.

David Paredes nació en Quito, Ecuador, el 20 de Junio de 1988, realizó sus estudios primarios en la Escuela San Pedro Pascual, sus estudios secundarios en el Colegio Militar Eloy Alfaro, sus estudios superiores los realizó en la Escuela Politécnica del Ejercito en la carrera de Ingeniería Electrónica en Automatización y Control.