DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR PID ADAPTIVO BASADO EN UN PROCESADOR DIGITAL DE SEÑAL dsPIC APLICADO AL CONTROL DE PROCESOS INDUSTRIALES

JOHN EVERT BARCO JIMENEZ
EDISSON ESCOBAR ROSERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA, DEPARTAMENTO DE ELECTRONICA
INGENIERIA ELECTRONICA
SAN JUAN DE PASTO
2009

DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR PID ADAPTIVO BASADO EN UN PROCESADOR DIGITAL DE SEÑAL dsPIC APLICADO AL CONTROL DE PROCESOS INDUSTRIALES

JOHN EVERT BARCO JIMENEZ
EDISSON ESCOBAR ROSERO

Trabajo de grado presentado como requisito parcial para optar el título de Ingeniero Electrónico

Director de Proyecto:
MSC Darío Fernando Fajardo Fajardo

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA, DEPARTAMENTO DE ELECTRONICA
INGENIERIA ELECTRONICA
SAN JUAN DE PASTO
2009

RESPONSABILIDAD

"Las ideas y conclusiones aportadas en el proyecto de grado son responsabilidad exclusiva de sus autores".

Artículo Primero del Acuerdo No. 324 de octubre 11 de 1966, emanado del Honorable Concejo Directivo de la Universidad de Nariño.

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del jurado

_____
Firma del jurado

San Juan de Pasto, Noviembre de 2009

El primer autor:

“Para Hernán y Matilde a quienes les debo, entre otras cosas, la increíble experiencia de estar vivo”.

“*A mi novia Paola por su infinita sabiduría que radica en su corazón, cómplice y amiga que siempre ha sabido apoyarme*”

El Segundo autor:

“Dedicado a mi madre Marnely, mi hermana Flor y por supuesto a mi novia Lady, por su motivación y constante apoyo en la realización de mis metas”

AGRADECIMIENTOS

A Dios por hacer que todo esto sea posible.

Ing. Darío Fernando Fajardo Fajardo, Director del trabajo de grado, por sus buenos consejos y orientación, su atención, colaboración y gran apoyo para realizar este proyecto.

A la Universidad de Nariño por brindarnos las herramientas suficientes para llevar este proyecto a feliz término.

Finalmente a todos nuestros compañeros y amigos quienes nos acompañaron o participaron en la realización de éste proyecto.

CONTENIDO

LISTA DE TABLAS

Pag.

LISTA DE FIGURAS

Pag.

LISTA DE ANEXOS

Pag.

ANEXO N. Librería para manejo de retardos, RETARDOS.INC (Archivo anexo en CD)

ANEXO O. Librería para manejo de teclado 4x4, TECLADO.INC (Archivo anexo en CD)

ANEXO P. Librería de mensajes, Mensajes2.inc (Archivo anexo en CD)

GLOSARIO

A/D: conversión de una señal analógica a digital.

D/A: conversión de una señal digital a analógica.

BUFFER: es una ubicación de la memoria en un instrumento digital reservada para el almacenamiento temporal de información.

DEFUSIFICAR: convertir del campo difuso al real.

DINAMICA: descripción de la evolución en el tiempo de un sistema físico en relación a las causas que provocan los cambios de estado físico y/o estado de movimiento.

DSP: es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad.

DSPIC o DSC: circuito integrado desarrollado por Microchip Technology Inc. denominados como controladores digital de señal, están basados en la arquitectura y el repertorio de instrucciones de los microcontroladores a lo cual se le añade prestaciones y recursos de los DSP.

DUAL: que reúne dos tipos de acciones de distintas pero reciprocas.

ESTATICA: estudio del sistema en equilibrio.

FIRMWARE: bloque de instrucciones de un programa para propósitos específicos, grabado en una memoria tipo ROM, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo es en parte hardware, pero también es software, ya que proporciona lógica y se dispone en algún tipo de lenguaje de programación.

FUSIFICAR: convertir del campo real al difuso.

FUZZY: (Difuso), relativo a la lógica difusa, permite representar de forma matemática los conceptos o conjuntos imprecisos. Esta proporciona las herramientas para analizarlos y sacar una conclusión como un humano lo haría.

GANANCIAS: constantes que permiten variar el desempeño del controlador PID, por ejemplo: k_p, k_i, y k_d. Que a su vez regulan la dimensión de la acción con la cual están relacionadas.

IMPLICACION: sirve para establecer una relación, o derivación entre una condición y su condicionado, o el establecimiento de una afirmación hipotética. Si las premisas son verdaderas lo es también la conclusión.

INFERENCIA: es una evaluación que realiza la mente entre conceptos. Acción o efecto de deducir o sacar una conclusión.

MAPEO: evaluar un vector teniendo en cuenta la correspondencia de los datos del vector con su índice.

MAPPING: relativo al mapeo.

MOSFET: transistor de efecto de campo basado en la estructura MOS (semiconductor de oxido de metal).

OFFSET: desfase o diferencia de tiempo entre dos señales similares.

ON-LINE: en línea. Acción que se refiere cuando se está ejecutando el proceso en el mismo instante.

PID: (Proporcional Integral Derivativo), mecanismo de control por realimentación que se utiliza en sistemas de control industriales, un controlador PID corrige el error entre un valor medido y el valor que se quiere obtener calculándolo y luego sacando una acción correctora que puede ajustar al proceso acorde. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral, y el derivativo.

PULL-UP: se refiere a la acción de elevar la tensión de salida de un circuito lógico, desplazando el punto de trabajo.

RESET: reiniciar, ir al inicio de una lista de instrucciones que conforman un programa.

SETPOINT: valor o punto de referencia al cual se quiere igualar.

SPI: (Serial Peripheral Interface), es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier electrónica digital que acepte un flujo de bits serie regulado por un reloj.

TIMER: módulo del microcontrolador que permite generar o medir periodos de tiempo en forma precisa.

TRANSITORIO: que dura un periodo de tiempo determinado finito.

WINDUP: exceso de la acción de integral del controlador PID.

RESUMEN

Este trabajo consiste en el diseño e implementación de un controlador PID discreto y programable con la capacidad de adaptar sus parámetros dependiendo del proceso con ayuda de un sistema difuso; cuenta con una interfaz de entrada por teclado para el punto de referencia y algunos datos para el controlador.

Se aplica al control de un motor, no obstante el sistema se puede modificar en su actuador para adecuarse a cualquier tipo de planta. El dispositivo se ha implementado en un controlador digital de señal (dsPIC).

La adaptabilidad se hace en base a la respuesta transitoria del sistema y a un mecanismo de inferencia lógico difuso. En este caso se presenta un nuevo mecanismo difuso, que se clasifica dentro del control inteligente, y es una solución relativamente sencilla para dotar al controlador de adaptabilidad en comparación con los métodos a priori de control o algunos del control moderno, como son la identificación de parámetros on-line. El controlador PID y el mecanismo de inferencia difuso de Mandani se simulo en Simulink de Matlab obteniéndose buenos resultados. Este nuevo mecanismo permite reducir recursos de hardware pudiéndose implementar en un DSC sin comprometer sus características adaptivas.

Palabras clave: Controlador PID, Parámetros Adaptivos, Lógica Difusa, Sistemas de Control

ABSTRACT

This work consists on the design and a controller's implementation discreet and programmable PID with the capacity to adapt its parameters depending on the process with the help of a fuzzy system, this it has an interface for keyboard for the entrance of data like they are the reference point and some parameters.

It is applied to the control of a three-phase motor, nevertheless the system can modify in its driver to be adapted to any plant type. The device has been implemented in a digital controller of sign (dsPIC) that is a family of microcontroller with characteristic of a digital processor of sign (DSP).

The adaptability is made based on the transitory response of the system and to a fuzzy logic inference mechanism. In this case it shows up a new mechanism based on fuzzy logic, this mechanism is classified inside the intelligent control, and it is a priori a relatively simple solution to endow the controller of adaptability in comparison with the methods of control or some of the modern control, like they are the on-line identification of parameters. The controller PID and the fuzzy inference mechanism of Mandani one simulates in Simulink of Matlab being obtained good results. This new mechanism allows to reduce hardware resources being able to implement in a DSC without committing its characteristic adaptive.
To prove their acting you implements the acting to manage a motor industrial three-phase, the regulator uses the method of control of feedback speed to climb, its operation is based on the modulation of space vectorial sinusoidal, this with the purpose of being able to demonstrate its application in an efficient way.

Keywords: PID Controller, Adaptive Parameter, Fuzzy Logic, Control Systems.

INTRODUCCION

Los sistemas de control se encuentran en casi la totalidad de todos los sectores de la industria, tales como líneas de ensamble automático, control de maquinas-herramienta, tecnología espacial, sistemas de armas, sistemas de potencia, robótica y muchos otros. La automatización y el control son los pilares que permiten el avance de los procesos industriales de forma que su eficiencia aumenta el doble y algunas veces aun más. Una herramienta esencial a través de todo este proceso ha sido el controlador PID que ha sido "el pan y mantequilla" de la ingeniería de control; el controlador PID soluciona muchos de los problemas que se encuentran en los procesos industriales, razón por la cual fue el más utilizado en la industria teniendo un 95% de participación y aun hoy en día ha sobrevivido a cambios tecnológicos y es considerado en gran parte de procesos implementado en microprocesadores.

A través de los años la ingeniería de control ha avanzado y haciendo su aparición el control moderno, donde se clasifica el control adaptivo o robusto, la definición dada para este tipo de controlador es, "*un controlador adaptivo es un controlador con parámetros ajustables y un mecanismo para ajustar los parámetros*"[1], el control robusto ha enfocado la adaptabilidad en adquirir información del proceso, de forma que se tenga una idea del modelo de la planta a la que se está controlando, estos sistemas cuentan con gran robustez a un costo de requerir una información detallada del proceso; debido a esta característica, es necesario emplear un elevado consumo de recursos de procesamiento y tiempo en el lazo de ajuste de parámetros, entonces, ¿No sería mejor contar con un controlador que no consuma tantos recursos y enfoque su procesamiento de forma más acertada?.

De esta forma nace el control inteligente, de la combinación de controladores como es el controlador PID y herramientas de inteligencia artificial, como son la lógica difusa. De ninguna manera se trata de hacer a un lado al control robusto, sino de encontrar un controlador que enfoque la adaptabilidad de manera más sencilla, en donde el procesamiento no se concentre solamente en conocer información a priori del proceso, también aprender del mismo, de esta forma se reparte mas acertadamente los recursos de procesamiento, en consecuencia se ha obtenido resultados iguales o mejores en algunos tipos de plantas que con el control robusto. La lógica difusa hace posible el diseño del mecanismo adaptivo, pues provee los medios para convertir una estrategia de control lingüística basada en el conocimiento de un experto en una señal de control. El controlador desarrollado en esta investigación se basa en el uso del controlador PID dotado de adaptabilidad por medio de un mecanismo difuso propuesto; para probar su funcionamiento se hicieron varias pruebas regulando la velocidad de un motor, sometido a diferentes

[1] ASTROM, Karl Johan y WITTENMARK, Bjorn. Adaptive Control. 2ed. United States of America: Addison Wesley, 1995. p.1.

formas de trabajo, no obstante se puede modificar su actuador según el tipo de planta en el que se implemente.

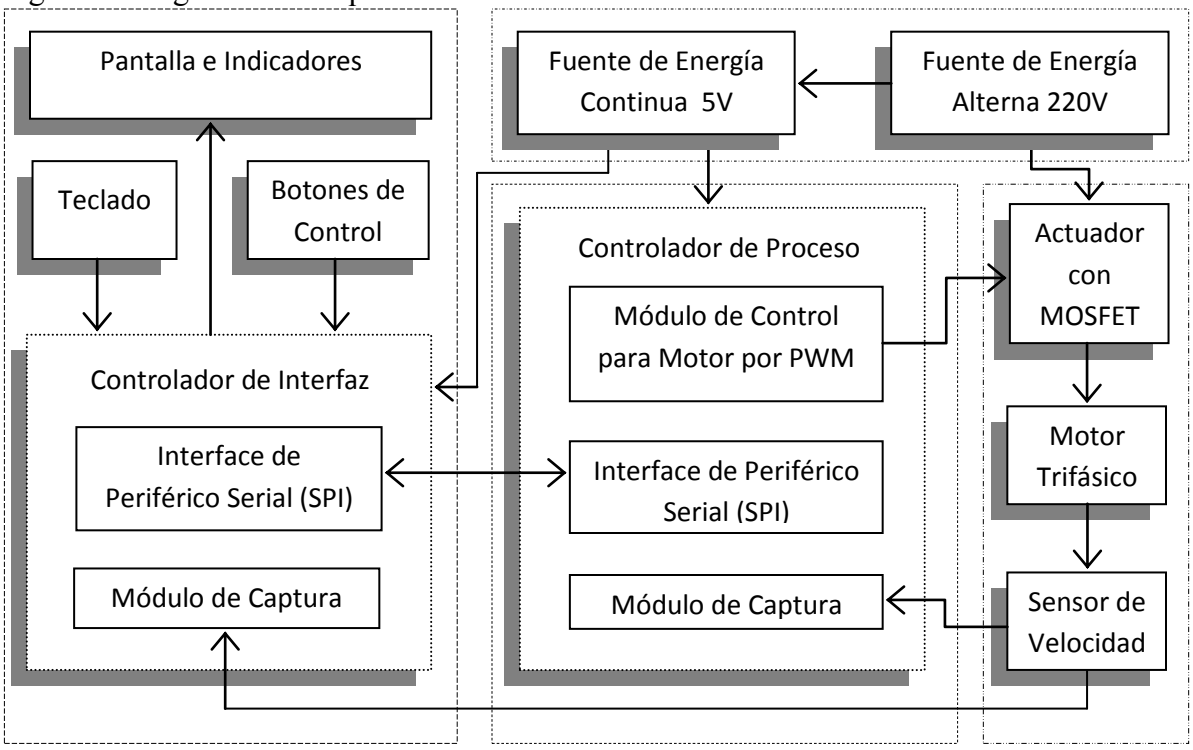
El documento inicia con una presentación general del dispositivo, después se abordara la explicación más detallada de cada uno de sus componentes, así mismo se describe el proceso de investigación que incluye las simulación en computador con MATLAB, la emulación del dispositivo y finalmente se presentaran las pruebas con los resultados de la implementación, seguidos de las conclusiones correspondientes al trabajo realizado. El presente trabajo de investigación pretende contribuir al estudio de sistemas de control en el grupo de investigación de instrumentación y sistemas inteligentes permitiendo profundizar en el control inteligente.

1. DESCRIPCION GENERAL DEL SISTEMA

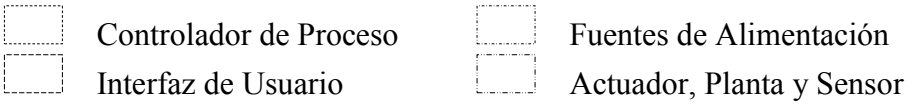
El controlador PID adaptivo que se presenta tiene como objetivo ser utilizado en la regulación de velocidad de un motor en el control de procesos, por ejemplo: el transporte de productos sub-terminados en bandas, mezclador de soluciones, sistemas de calibración rotatorios, tornos automáticos, etc. En general procesos o maquinas donde la precisión de la velocidad de giro y la potencia sean necesarias. No obstante, también se resuelven algunos problemas en este tipo de controladores como son: la portabilidad, facilidad en la instalación y uso, facilidad de calibración de parámetros (adaptabilidad) y costo de adquisición. El trabajo de investigación se centra en dotar al controlador de la característica de adaptabilidad, para esto se propone un mecanismo adaptivo basado en lógica difusa.

El sistema se puede describir por el diagrama de bloques de la siguiente figura.

Figura 1. Diagrama de bloques del controlador



Se puede observar en el diagrama que el controlador se puede dividir en 4 partes principales:



- **Controlador de proceso.** El controlador de proceso se encarga de ejecutar dos tareas importantes, la ejecución del algoritmo del controlador PID y el algoritmo del mecanismo adaptivo basado en lógica difusa.

El hardware que se utiliza para la implementación es el dispositivo dsPIC de Microchip, este cumple con todos los requisitos necesarios en relación a velocidad de procesamiento, en cuanto a sensores y actuadores; existe una distribución de dsPIC con unos periféricos específicos para el control de motores.

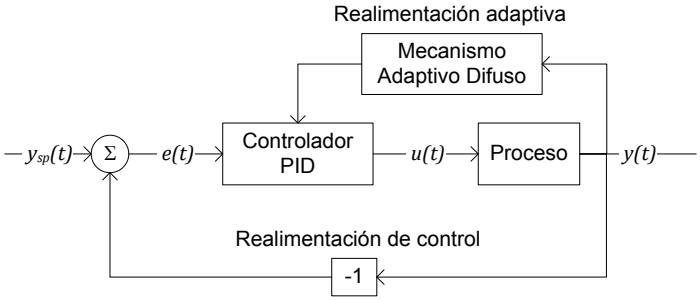
- **Interfaz de usuario.** Es la encargada de interactuar con el usuario para ingresar los datos de proceso. Cuenta con hardware independiente basada en un microcontrolador PIC que maneja de forma sencilla el teclado y la pantalla.

- **Fuentes de alimentación, actuador, planta y sensor.** La fuente provee la energía a los componentes del sistema, el actuador es el encargado de manejar el motor trifásico dependiendo del la señal generada por el controlador, el sensor mide la velocidad de la planta motor trifásico.

2. CONTROLADOR DE PROCESO

El objetivo de un sistema de control es regular la salida de un proceso cumpliendo unas especificaciones aplicando una estrategia prescrita de control, haciendo uso de los elementos del sistema. El sistema de control está conformado por sensores, actuadores y el controlador de proceso, en este caso como se ha mencionado el tipo de control es adaptivo, por definición, "Un controlador adaptivo es un controlador con parámetros ajustables y un mecanismo para ajustar los parámetros"[2]. Con base a la definición anterior un sistema adaptivo cuenta con dos lazos, el lazo normal de realimentación entre el proceso y el controlador y el lazo de ajuste de parámetros.

Figura 2. Diagrama de bloques del controlador PID adaptivo difuso



Evidentemente el controlador con parámetros ajustables y que proporciona el lazo de realimentación es el controlador PID que cuenta con un diseño robusto y simple. El sistema que se utiliza en el lazo de ajuste de parámetros se enmarca en la teoría de la inteligencia artificial (AI) y se hace con ayuda de la lógica difusa, esta se encarga de proporcionar la realimentación adaptiva, la cual recopila información del proceso y dependiendo de un conocimiento base genera una respuesta que cambia los parámetros del controlador. El estudio del controlador de proceso se puede dividir en tres partes.

- Los Procesos y el controlador PID
- Diseño del mecanismo adaptivo difuso
- Funcionamiento del controlador PID adaptivo difuso

[2] Ibit., p.1.

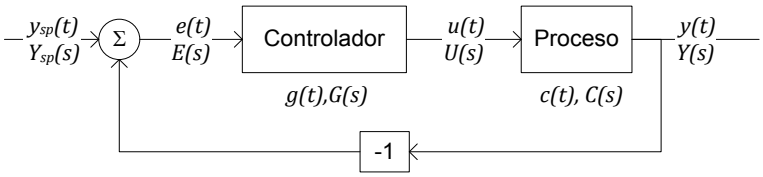
2.1 LOS PROCESOS Y EL CONTROLADOR PID

El controlador PID tiene varias funciones importantes, provee realimentación, tiene la habilidad para eliminar el error en estado estable por medio de la acción integral, genera una señal para una corrección anticipativa con ayuda de la acción derivativa, el controlador PID soluciona la mayoría de problemas en el control de procesos a bajo costo. Como se sabe este controlador tiene tres parámetros ajustables según el tipo de proceso de los cuales depende su desempeño. Es evidente que el estudio detallado entre la relación del controlador y las características del proceso es la clave del desempeño del controlador.

2.1.1 Los procesos y la realimentación. La siguiente figura muestra un lazo de control, donde se pueden identificar dos componentes principales, el proceso, sistema o planta y el controlador, el proceso tiene como entrada la variable manipulada o variable de control que se denota como $u(t)$, la señal de salida del proceso o variable de proceso se denota como $y(t)$, el valor deseado para la variable de proceso se conoce como punto de referencia o setpoint se denota como $y_{sp}(t)$. La señal de error que se denota como $e(t)$, es la diferencia entre el punto de referencia y la variable de proceso. En la siguiente figura también se muestra como el proceso y el controlador están conectados en un lazo de realimentación cerrado. El propósito del sistema es mantener la variable de proceso cerca del valor deseado, esto se consigue por el lazo de realimentación. La idea de realimentación es simple y aun extremadamente poderosa, esta se puede expresar de la siguiente manera. Aumentar la variable manipulada cuando la variable de proceso es más pequeña que el punto de referencia y disminuir la variable manipulada cuando la variable de proceso es más grande que el punto de referencia. Este tipo de realimentación es llamada realimentación negativa porque la variable manipulada se mueve en el sentido opuesto a la variable de proceso[3]. La razón de porque los sistemas realimentados son interesantes es porque la realimentación hace que la variable de proceso este cerca del punto de referencia a pesar de los disturbios y la variación de las características del proceso, en contraste con los sistemas en lazo abierto los cuales no tienen en cuenta la medida de la señal de salida como tal no cuentan con lazo de realimentación y no se compara la variable de proceso con la de error en el punto de sumatoria.

[3] ASTROM, Karl Johan y HAGGLUND, Tore. PID Controllers: Theory, Design and Tunnig. 2 ed. Unites States of America: Instrument Society of America, 1995. p.60.

Figura 3. Diagrama de bloques de un sistema con realimentación unitaria



Fuente: KUO, Benjamin. Sistemas de Control Automático, 1996.

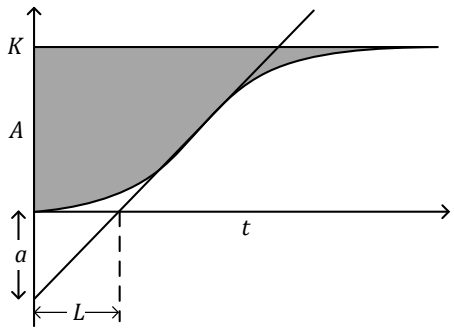
Los controladores generalmente tienen varios parámetros que pueden ser ajustados, el desempeño de estos depende de que tan apropiadamente se hayan escogido estos parámetros en función del proceso que controlan, el procedimiento para escoger estos parámetros se llama sintonía.

2.1.1.1 Modelo general de un proceso. Un modelo simple de la dinámica del proceso es el modelo de los dos parámetros, este se determina después de aplicar un escalón unitario y grabar la variable de proceso, todo este proceso se debe realizar en lazo abierto sin hacer uso de la realimentación. El proceso se puede aproximar por el siguiente modelo según el tiempo muerto (L) que se describe con la siguiente ecuación.

$$G_b(s) = \frac{a}{sL} e^{-sL} \qquad (1)$$

Este modelo corresponde a un integrador con tiempo muerto, está caracterizado por dos parámetros, a y L, que se determinan gráficamente desde la respuesta al escalón mostrada en la siguiente figura, donde la tangente a la respuesta al escalón es la mayor pendiente encontrada y la intersección de esta tangente con los ejes vertical y horizontal dan como resultado a y L respetivamente[4]. Este modelo es el utilizado para el procedimiento de sintonía de un controlador PID por el método de Ziegler-Nichols.

Figura 4. Modelo de los dos parámetros



Fuente: ASTROM, Karl. PID Controllers: Theory, Design and Tuning, 1995.

[4] Ibit., p.14.

2.1.1.2 Especificaciones de diseño para un proceso. Antes que se diseñe un controlador se deben tener en cuenta los requerimientos u objetivos que este debe cumplir al controlar un proceso, estos requerimientos se pueden observar al analizar los dos tipos de respuesta que conforman la respuesta total de un proceso.

- **Respuesta estática de un proceso.** Uno de los objetivos principales de los controladores es que la variable de proceso siga al punto de referencia de forma exacta en estado estable. Un proceso entra en estado estable cuando el efecto transitorio debido a una estrada se ha terminado completamente. La diferencia entre la variable de proceso y el punto de referencia en estado estable se define como error en estado estable (e_{ss}), en el mundo real debido a la fricción y otras imperfecciones, la respuesta de salida en estado estable raramente concuerda exactamente con la referencia[5]. Considerando la figura 3 se define e_{ss} así:

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{s \to 0} s \cdot E(s) = \lim_{s \to 0} \frac{s \cdot Y_{sp}(s)}{1 + C(s)G(s)} \qquad (2)$$

Si $Y_{sp}(s)$ es una función escalón con magnitud A, entonces $Y_{sp}(s) = A/s$, y e_{ss} se define así:

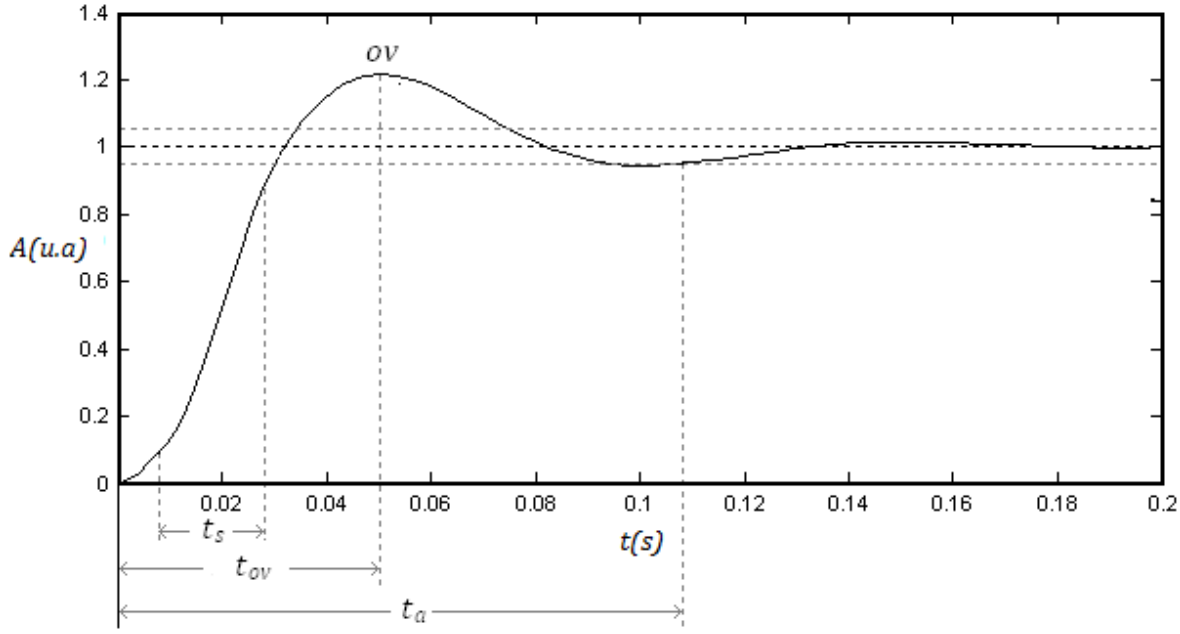
$$e_{ss} = \lim_{s \to 0} \frac{s \cdot Y_{sp}(s)}{1 + C(s)G(s)} = \lim_{s \to 0} \frac{A}{1 + C(s)G(s)} = \frac{A}{1 + \lim_{s \to 0} C(s)G(s)} \qquad (3)$$

Teniendo en cuenta que el número de polos en cero define el orden del proceso, para procesos de tipo 0 o con ningún polo en cero e_{ss} se define como indica la anterior ecuación, no obstante también se observa que para que e_{ss} sea cero a entradas escalón unitario, $G(s)$ o en su defecto $C(s)$ debe poseer un polo en $s = 0$, como lo puede proporcionar un controlador que posea una acción integradora.

- **Respuesta dinámica de un proceso.** La respuesta transitoria de un proceso en el tiempo es aquella que tiende a cero cuando el tiempo aumenta, la respuesta transitoria de un sistema es importante ya que debe mantenerse dentro de los límites tolerables. Para un sistema de control lineal la caracterización de la respuesta transitoria se obtiene aplicando la función escalón unitario en lazo cerrado.

[5] KUO, Benjamin. Sistemas de Control Automático. 7 ed. México: Prentice Hall Hispanoamérica S.A, 1996. p.362.

Figura 5. Respuesta típica de un proceso al escalón unitario



Donde, "ov" es el sobrepaso máximo, es el mayor valor de la variable de proceso después de superar a $y_{sp}(t)$ y el tiempo en el que ocurre se denomina tiempo de sobrepaso (t_{ov}), "t_s" es el tiempo de subida, es el tiempo requerido para que la respuesta al escalón se eleve del 10 al 90% de su valor final, "t_a" es el tiempo de asentamiento, es el tiempo requerido para que la respuesta al escalón disminuya y permanezca dentro de un porcentaje especifico, puede ser del 2% o en este caso del 5%.

2.1.2 Controlador PID. Desarrollado empíricamente, reúne tres acciones que lo hacen robusto y flexible para el manejo de procesos. La suma de las acciones proporcional, integral y derivativa constituyen una estructura sencilla y eficiente. Para entender el funcionamiento de este controlador se puede analizar cada acción de forma independiente, el algoritmo PID se describe como:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \tag{4}$$

Función de transferencia del controlador en el dominio de Laplace (s)

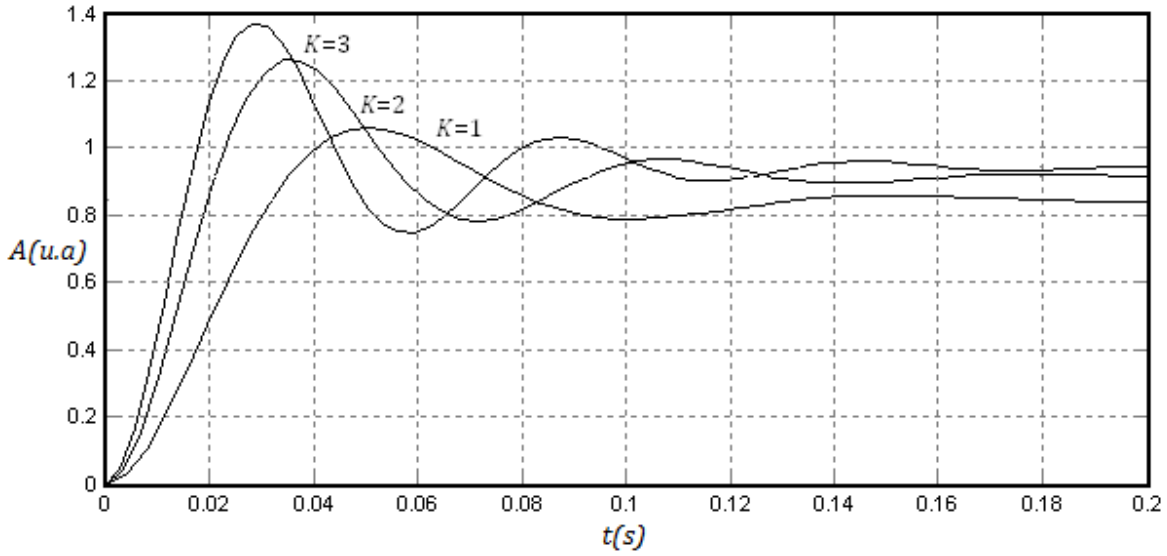
$$C(s) = K\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{5}$$

Donde, $u(t)$ es la variable de control y $e(t)$ es el error de control, la variable de control es la suma de tres términos, el termino proporcional al error (P), el termino proporcional a la integral del error (I) y el termino proporcional a la derivada del error (D), los parámetros del controlador son la ganancia proporcional (K), el tiempo integral (T_i), y el tiempo derivativo (T_d). La clave para lograr determinar una buena combinación de parámetros y en efecto mejorar el desempeño del controlador es conocer cómo afectan cada una de las tres acciones a un proceso en general.

2.1.2.1 Acción proporcional. En el caso del controlador proporcional, la acción de control es simplemente proporcional al error como se observa en la siguiente ecuación.

$$u_P(t) = K \cdot e(t) \tag{6}$$

Figura 6. Simulación de un controlador proporcional



$$G(s) = \frac{3950}{(s + 29.72)(s + 24.47)} \tag{7}$$

Cuando se considera la dinámica del sistema se introducen otras propiedades sobre el comportamiento del sistema en lazo cerrado. Un sistema en lazo cerrado normalmente será inestable si se eligen altas ganancias de lazo, un ejemplo típico de un control proporcional se muestra en la anterior figura, esta muestra el comportamiento de la salida de proceso con función de transferencia como indica la ecuación 7, después de un cambio al escalón en la señal de referencia. El error en estado estable decrece cuando se incrementa la ganancia del controlador, se nota también que se vuelve más oscilatoria perdiendo estabilidad pero

mejorando su dinámica en cuanto a tiempo de subida[6]. Si se iguala $k_p = K$ donde k_p se conoce como la ganancia proporcional se puede concluir que los efectos de la acción proporcional en función de la variación de su ganancia se pueden resumir según la siguiente tabla.

Tabla 1. Efectos del Controlador Proporcional

	Tiempo de subida	Sobrepaso	Tiempo de asentamiento	Error en estado estable	Estabilidad
Incremento k_p	*Disminuye*	Aumenta	Disminución menor	Disminuye	Degrada

Fuente: JANTZEN, Jan. Tuning of Fuzzy PID Controllers, 1998.

2.1.2.2 Acción integral. La función principal de la acción integral es asegurar que la salida del proceso concuerde con el punto de referencia en estado estable, con el control proporcional normalmente existe error en estado estable, con la acción integral un pequeño error positivo llevara a incrementar la señal de control y un pequeño error negativo llevara a decrementar la señal de control sin importar que tan pequeño sea. La función de transferencia de un controlador proporcional integral es:

$$C_{PI}(s) = K\left(1 + \frac{1}{s \cdot T_i}\right) \quad (8)$$

Como se observa la acción integral del controlador añade un polo en $s = 0$ a la función de transferencia del proceso incrementando el tipo de sistema en uno, en general el error en estado estable mejora en un orden, es decir, si el error en estado estable a una entrada dada es constante, la acción integral lo reduce a cero. El problema surge en escoger T_i y K adecuados para que la respuesta transitoria cumpla los requerimientos y sea satisfactoria.

Las propiedades de la acción integral se ilustran en la siguiente figura, donde el proceso tiene como función de transferencia la indicada en la ecuación 7. La ganancia proporcional se mantiene en $K = 1$ y el tiempo integral se varia, se puede observar que los tiempos grandes de integración hacen que la respuesta dirija lentamente hacia el punto de referencia y los tiempos pequeños hacen más oscilatorio e inestable el proceso aumentando el sobrepaso y disminuyendo el tiempo de subida. De la ecuación anterior se puede observar que si $T_i \rightarrow \infty$ la acción integral se anula y cuando toma valores finitos remueve el error en estado estable. Si se relaciona $k_i = K/T_i$ donde k_i es la ganancia integral, se puede concluir que los efectos de la acción integral en función de la variación de su ganancia se pueden resumir según la siguiente tabla.

[6] ASTROM y HAGGLUND, Op. cit., p.65.

Figura 7. Simulación de un controlador proporcional integral

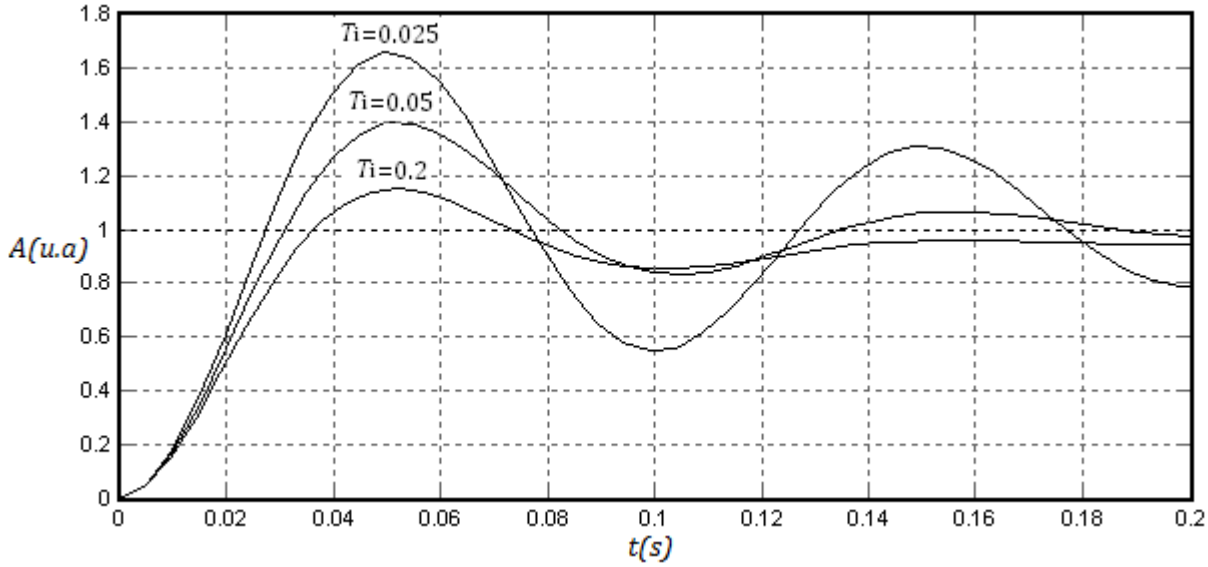


Tabla 2. Efectos de la acción integral

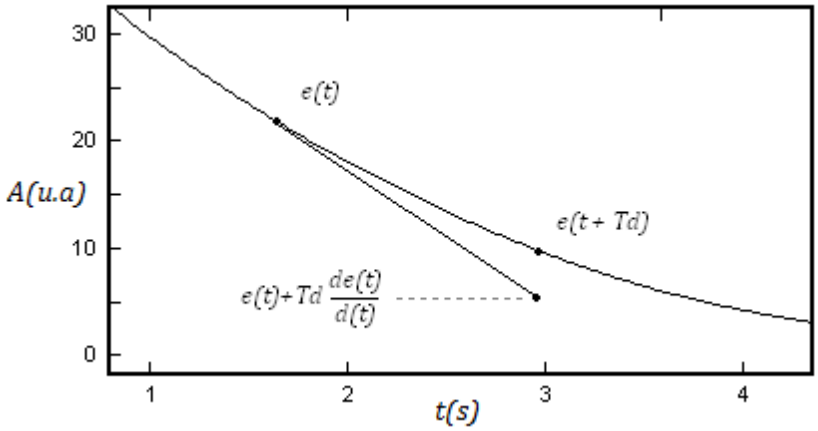
	Tiempo de subida	Sobrepaso	Tiempo de asentamiento	Error en estado estable	Estabilidad
Incremento k_i	Disminución menor	Aumenta	Aumenta	*Disminución considerable*	Degrada

Fuente: JANTZEN, Jan. Tuning of Fuzzy PID Controllers, 1998.

2.1.2.3 Acción derivativa. La acción de un controlador proporcional derivativo se puede interpretar como si esta fuese hecha para predecir la salida del proceso, la predicción se hace por extrapolación del error en la dirección de la tangente a su curva respectiva.

$$u_{PD}(t) = K\left[T_d \frac{de(t)}{dt}\right] \quad (9)$$

Figura 8. Interpretación de la acción derivativa como control predictivo



Fuente: ASTROM, Karl. PID Controllers: Theory, Design and Tuning, 1995.

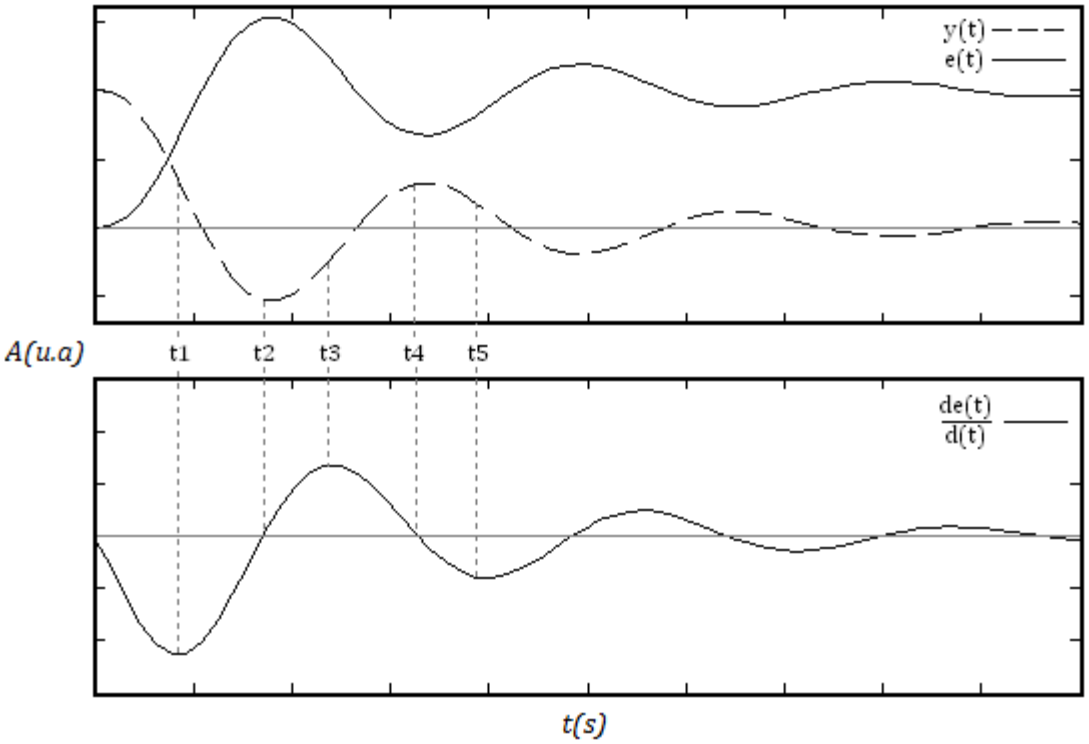
La estructura básica de un controlador PD está dada por la anterior ecuación, si se hace la expansión de series de Taylor de $e(t + T_d)$, el resultado es:

$$e(t + T_d) \approx e(t) + T_d \frac{de(t)}{dt} \tag{10}$$

De esta manera, la señal de control es proporcional a un estimado del error de control en un tiempo T_d hacia adelante. Donde el estimado se obtiene mediante la extrapolación lineal como indica la anterior figura.

En la siguiente figura se puede observar la respuesta de un proceso a una entrada escalón unitario, la cual tiene sobrepaso y se muestra oscilatoria, la señal de control de error correspondiente y su derivada en el tiempo, siguiendo de cerca la derivada del error se puede analizar los beneficios de la acción derivativa.

Figura 9. Análisis de la derivada del error de control



Fuente: KUO, Benjamin, Sistemas de Control Automático, 1996.

Al considerar la derivativa del error según la figura anterior, se puede decir que los factores que contribuyen al sobrepaso grande son: el error de control es muy grande en el intervalo $0 < t < t1$, y este mismo en el intervalo $t1 < t < t2$ es inadecuado, por tanto, para reducir el sobrepaso se debería disminuir el error de control en el intervalo $0 < t < t1$ y aumentarlo en el intervalo $t1 < t < t2$. De forma similar en el intervalo $t2 < t < t4$, el control de error en el intervalo $t2 < t < t3$ se debe reducir porque genera un sobrepaso y aumentarlo en el intervalo $t3 < t < t4$ para corregir adecuadamente el sobrepaso negativo. Por tanto, se puede deducir que la acción derivativa produce el efecto de compensación requerido, así:

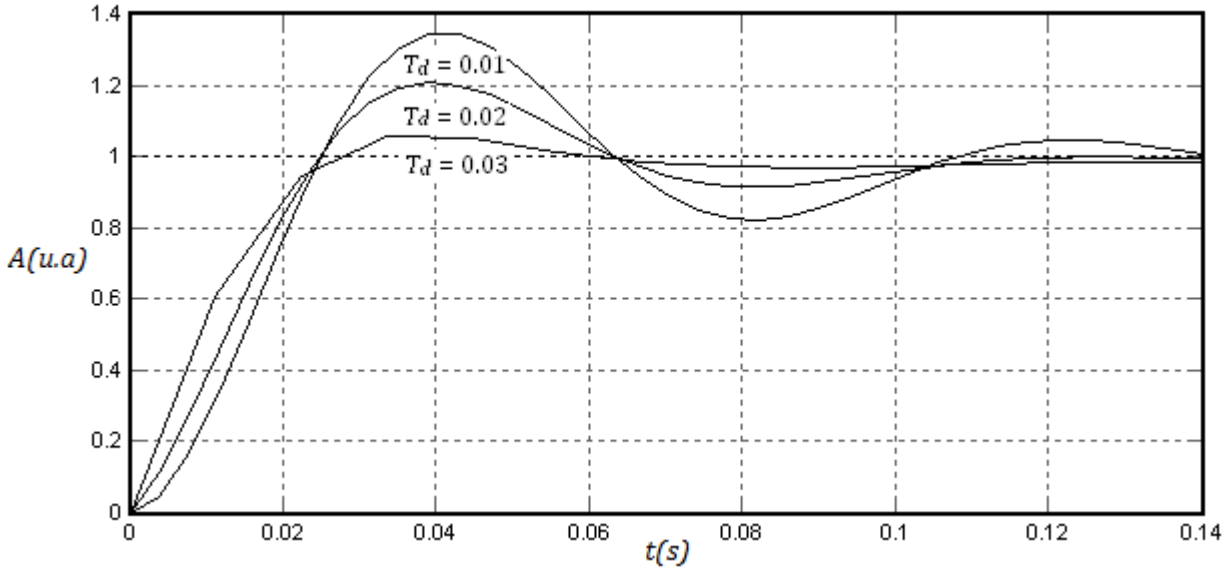
- Para el intervalo 0 < t < t1 la acción $\frac{de(t)}{dt}$ es negativa, por tanto se reducirá el error de control.

- Para el intervalo t1 < t < t2 tanto $e(t)$ como $\frac{de(t)}{dt}$ son negativos, por tanto se aumentara el error de control.

- Para el intervalo t2 < t < t3, $e(t)$ y $\frac{de(t)}{dt}$ tienen signos opuestos por tanto se reducirá el error de control y contrarrestara el sobrepaso negativo[7].

La siguiente figura muestra la simulación de un sistema con control PID sobre un proceso que tiene como función de transferencia la definida en la ecuación 7. Las ganancias proporcional e integral se mantienen constantes y el tiempo derivativo varia. Se puede deducir, a medida que aumenta el tiempo derivativo el amortiguamiento se incrementa o sea disminuyen las oscilaciones, el sobrepaso se reduce considerablemente y el tiempo de subida disminuye en menor medida.

Figura 10. Simulación de un controlador proporcional derivativo e integral



Si se relaciona $k_d = K \cdot T_d$ donde k_d es la ganancia derivativa, se puede concluir que los efectos de la acción derivativa en función de la variación de su ganancia se pueden resumir según la siguiente tabla.

Tabla 3. Efectos de la acción derivativa

	Tiempo de subida	Sobrepaso	Tiempo de asentamiento	Error en estado estable	Estabilidad
Incremento k_d	Disminución menor	*Disminución considerable*	Disminuye	Cambio Menor	Mejora

Fuente: JANTZEN, Jan. Tuning of Fuzzy PID Controllers, 1998.

[7] KUO, Op. cit., p.675.

El controlador PID es la suma de las tres acciones, proporcional, integral y derivativa, cada una de estas tiene dominancia en alguna característica en particular, la acción derivativa regula evidentemente el sobrepaso y la acción integral el error en estado estable, sin olvidar la acción proporcional que puede asociarse con la rapidez del proceso según este lo requiera. La siguiente ecuación del controlador PID resultante, tiene en cuenta la nueva notación de las ganancias k_p, k_i, y k_d, donde la variación de alguna de estas no afecta a la otra, de manera que trabajan en forma no interactiva, esta característica es importante ya que es favorable para el mecanismo adaptivo que se explicara más adelante.

$$u_{PID}(t) = k_p \cdot e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \qquad (11)$$

2.1.2.4 Método de sintonía de respuesta al escalón según Ziegler y Nichols (Z&N). Este método está basado en el registro de la respuesta al escalón basado en el modelo de los dos parámetros. Ziegler y Nichols propusieron los parámetros PID y una estimación del periodo de muestreo (h) como función de a y L. Estos se muestran en la siguiente tabla.

Tabla 4. Parámetros del controlador PID obtenidos por el método de sintonía de respuesta al escalón según Ziegler y Nichols

	Ganancia proporcional (K)	Tiempo integral (T_i)	Tiempo derivativo (T_d)	Periodo de muestreo (h)
Parámetros PID	$1.2/a$	$2L$	$L/2$	$3.4L$

Fuente: ASTROM, Karl. PID Controllers: Theory, Design and Tuning, 1995.

2.1.2.5 Criterio de la integral absoluta del error ponderada por el tiempo ITAE. El criterio ITAE se define para verificar la calidad de control de la variable, también sirve para juzgar el seguimiento del punto de referencia o setpoint.

$$ITAE = \int_0^\infty t|e(t)|dt \qquad (12)$$

2.1.2.6 Modificaciones del algoritmo. Debido a que la acción derivativa y la integral pueden presentar dificultades, se hace necesario prestar atención a estos aspectos.

- **Limitación de la ganancia derivativa.** La acción derivativa presenta dificultades si existe ruido de medición de alta frecuencia, si $n(t)$ es un ruido de medición senoidal.

$$n(t) = a \sin wt \tag{13}$$

Si el ruido se aplica a la ecuación de la acción derivativa, el resultado sería:

$$u_{PD}(t) = KT_d \frac{dn(t)}{dt} = aKT_d w \cos wt \tag{14}$$

De la anterior ecuación, la amplitud de la señal de control puede ser arbitrariamente grande si el ruido tiene alta frecuencia, se debe limitar esta dificultad de la ganancia de alta frecuencia, esto se logra implementando el término derivativo con ayuda de un filtro de primer orden con un constante de tiempo $\frac{T_d}{N}$ como indica la siguiente ecuación, esta aproximación actúa como una derivada para señales con componentes de baja frecuencia, los valores típicos de N son de 8 a 20[8].

$$D(s) = -\frac{KT_d s}{1 + \frac{T_d}{N} s} Y(s) \tag{15}$$

En el dominio del tiempo

$$D(t) = -\frac{T_d}{N} \frac{dD(t)}{dt} - KT_d \frac{dy(t)}{dt} \tag{16}$$

- **El efecto windup en la acción integral.** En un proceso se puede presentar en su actuador un efecto no lineal como la saturación el error permanece, cuando se usa un controlador con acción integral el error continua siendo integrado y se incrementa, esto significa que termino integral hace windup causando transitorios grandes[9]. Para evitar el efecto windup se maneja un modelo de referencia del actuador conociendo sus características, de forma que se condiciona la acción integral para que nunca sobrepase los límites.

2.1.2.7 Requerimientos para la implementación digital. Se utilizara un sistema de procesamiento digital, debido a la diferencia de naturaleza de los sistemas, entre proceso analógico y controlador digital, se debe tener en cuenta tres aspectos importantes:

- Muestreo y reconstrucción de las señales
- Discretización del algoritmo

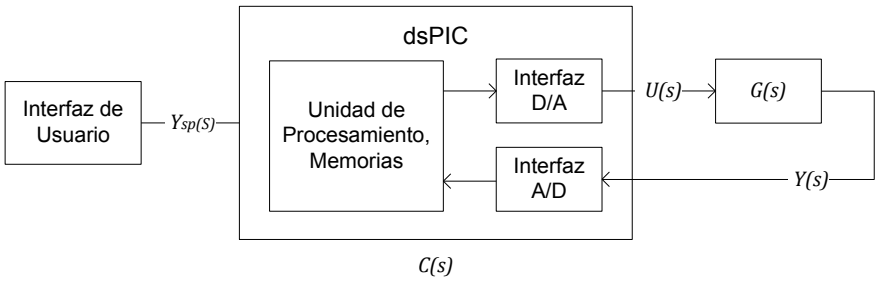
[8] ASTROM y HAGGLUND, Op. cit., p.77.

[9] Ibit., p.81.

- Cuantificación

En función de estos aspectos se debe hacer una elección adecuada del procesador digital, en este caso se utiliza un procesador digital de señal dsPIC que cuenta con las herramientas específicas y adecuadas para el control de procesos.

Figura 11. Diagrama de bloques de un controlador digital aplicado a un proceso analógico



- **Muestreo y reconstrucción de la señal.** Para enlazar dos tipos de sistemas diferentes analógico (A) y digital (D) se hace necesario de una interfaz A/D, para adquirir la variable de proceso $y(t)$ la cual se procesara y generara una señal de control hacia la planta, en este ultimo procedimiento inverso se necesita una interfaz D/A para convertir las señales discretas del controlador en analógicas y así poder aplicarlas a la planta. En resumen se necesitan dos tipos de interfaces A/D y D/A.

- **La interfaz A/D.** Esta realiza el procedimiento de muestreo, se toma una muestra de la variable de proceso periódicamente con un periodo h donde interviene el sensor, el acondicionador de señal y el conversor A/D. Este mecanismo de muestreo introduce algunos fenómenos inesperados, para evitar este tipo de problemas de debe asegurar un periodo de muestreo adecuado dependiendo de la constante de tiempo de la planta y un filtrado para evitar componentes indeseables.

- **La interfaz D/A.** Esta realiza una conversión de la señal de control digital generada a analógica, en este procedimiento se hace necesario un circuito retenedor que se encarga de decodificar y reconstruir la señal digital logrando una señal en tiempo continuo, en este proceso se realiza un filtrado de las componentes de alta frecuencia inherente de las señales discretas, en este proceso intervienen tres componentes que son el acondicionador de señal, el actuador y el conversor D/A. Para proporcionar la señal de control a la planta se genera en primera instancia por el procesador digital, para esto el dsPIC cuenta con un módulo propio de control generador de señal PWM que hace la conversión D/A, después pasa al acondicionador de señal, y finalmente al actuador, de esta forma esta lista para ser aplicada a la planta o proceso.

- **Discretización de algoritmo PID.** Para implementar las leyes del control en el tiempo continuo del controlador PID en un dispositivo digital, se hace necesario aproximar la derivada y la integral para que tengan la mayor similitud con el modelo continuo. Al término proporcional de la siguiente ecuación se le reemplaza la variable de tiempo continua por la versión muestreada,

$$P(t) = k_p \cdot e(t) \tag{17}$$

Versión muestreada,

$$P(k) = k_p \cdot e(k) \tag{18}$$

Para el término integral de la siguiente ecuación

$$I(t) = k_i \int_0^t e(\tau) d\tau \tag{19}$$

Se puede decir que

$$\frac{dI(t)}{dt} = k_i \cdot e(t) \tag{20}$$

Existen muchas formas de implementar esta ecuación, sin embargo la aproximación en diferencias hacia atrás es una de las técnicas que se apega al modelo real y que presenta buenos resultados en la simulación.

$$\frac{I(k) - I(k-1)}{h} = k_i \cdot e(k) \tag{21}$$

Despejando el término integral de la anterior ecuación

$$I(k) = I(k-1) + k_i h \cdot e(k) \tag{22}$$

Donde "h" es el periodo de muestreo y es constante. Debido a que este término es constante y no altera las reglas de variación observada en la tabla 2, se tomara la nueva ganancia k_i como el producto de $k_i h$, para la implementación digital la ecuación que define la acción integral es:

$$I(k) = I(k-1) + k_i \cdot e(k) \tag{23}$$

La aproximación de la acción derivativa teniendo en cuenta la limitación de la ganancia de la derivada dada en la ecuación 16 se puede realizar de la misma forma como se hizo con el termino integral

$$\frac{T_d}{N}\frac{D(k)-D(k-1)}{h}+\ D(k)=\ -KT_d\frac{(y(k)-y(k-1))}{h} \tag{24}$$

Que se puede reescribir así:

$$D(k)=\ a_dD(k-1)-b_d(y(k)-y(k-1)) \tag{25}$$

$$a_d=\frac{T_d}{T_d+Nh} \tag{26}$$

$$b_d=\frac{KT_dN}{T_d+Nh} \tag{27}$$

Donde, a_d es un coeficiente relacionado con la limitación de la ganancia derivativa, b_d es un coeficiente proporcional k_d, evidentemente sigue de igual forma las reglas de variación de la ganancia observada en la tabla 3, por esto se considera la nueva ganancia derivativa y denota como k_d.

- **Cuantificación.** Un computador digital solamente permite precisión finita en los cálculos, en algunos casos se hace difícil implementar el termino integral en procesadores que manejan palabras de longitud corta, para evitar el redondeo de este término que se presenta cuando el error o la ganancia integral son muy pequeños, es necesario tener una precisión de al menos 23 bits, en este caso el dsPIC permite usar variables tipo flotante con precisión de 32 bits de la siguiente manera:

Tabla 5. Formato de variable tipo flotante de precisión simple de 32 bits

Tamaño en bits	1	8	23
	Signo	Exponente	Mantisa
Índice de bits	31	30 23	22 0

Un número de punto flotante de precisión simple se almacena en una palabra de 32 bits, donde el signo se maneja con un bit, el exponente maneja 8 bits y la mantisa 23 bits. El signo se maneja de forma que 0 = positivo y 1 = negativo, los valores 0 y 255 manejan un significado especial, cero e infinito respectivamente. Entonces los números se expresan de la siguiente forma

$$v=s\times 2^{exp}\times m \tag{28}$$

Donde, v es el valor del numero, exp es el exponente y m la mantisa.

Las variables que se utilizan en la construcción de la señal de control y otras auxiliares pueden ser de tipo de flotante o pueden ser simplemente de tipo carácter de 8 bits o tipo entero de 16 bits, las cuales pueden o no manejar signo, en tipo de datos no se necesita alguna codificación especial.

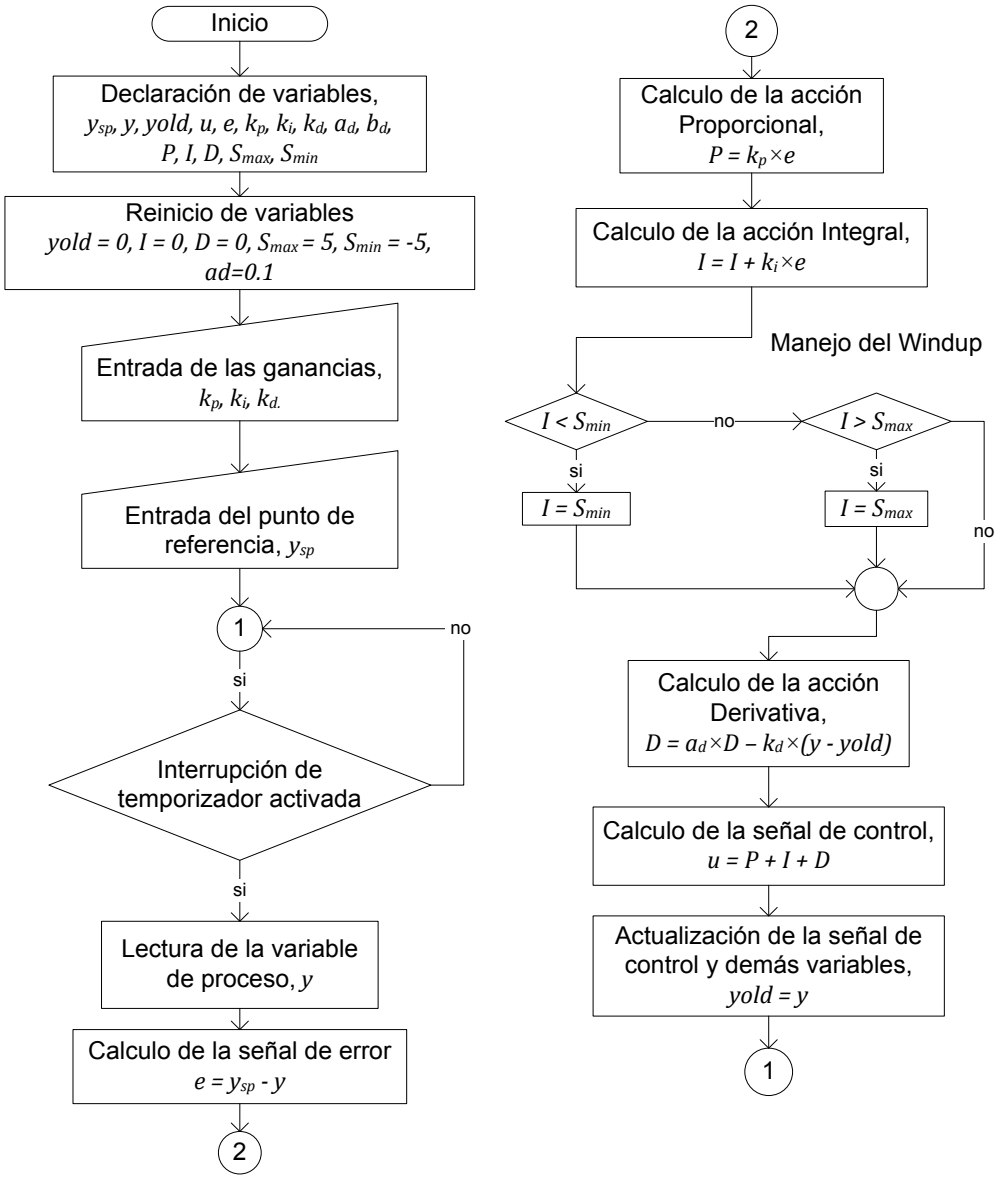
- **Algoritmo de control.** Al implementar una estrategia de control en un sistema digital, teniendo en cuenta los aspectos que se mencionaron anteriormente, se sigue esta secuencia básica periódica de operaciones:

  - Esperar la señal periódica de interrupción
  - Leer la variable de proceso
  - Procesar la señal de control (algoritmo PID)
  - Enviar la señal de control al actuador
  - Actualizar todas las variables
  - Ir al inicio

El algoritmo se explica con detalle en el diagrama de bloques de la siguiente figura, el algoritmo inicia con las rutinas básicas, declaración y reinicio de variables, seguido de las rutinas, lectura de las ganancias (k_p, k_i, y k_d) y lectura del punto de referencia (y_{sp}) que se ejecutan con ayuda de la interfaz de usuario que se explicara más adelante. Es necesario que la acción de control se ejecute en tiempos precisos de la misma magnitud y de forma periódica, para esto se hace uso de la interrupción del temporizador con periodo h, después se hace la lectura de $y(t)$, se calcula $e(t)$, y en base al error y las ganancias del controlador PID se calculan las acciones del controlador, proporcional (P), integral (I) y derivativa (D), evidentemente en este cálculo se tiene en cuenta la limitación de la acción derivativa y el manejo del windup con ayuda de las variables S_{max} y S_{min} que contienen los limites máximo y mínimo permitidos respectivamente del actuador. La señal de control $u(t)$ obtenida con la suma de los términos P, I, y D se actualiza y se envía al proceso por medio del acondicionador de señal y el actuador. Así finaliza el ciclo para ser ejecutado nuevamente según el temporizador de interrupción.

Este algoritmo de control PID está preparado para ser programado en el procesador digital de señal dsPIC, cabe destacar que se hicieron pruebas para escoger el algoritmo con mayor desempeño y que se acogiera con mayor similitud a las reglas observadas en las tablas de comportamiento de variación de las ganancias PID. Para que el controlador sea adaptivo solo resta adicionar el mecanismo adaptivo difuso propuesto.

Figura 12. Diagrama de flujo del controlador PID

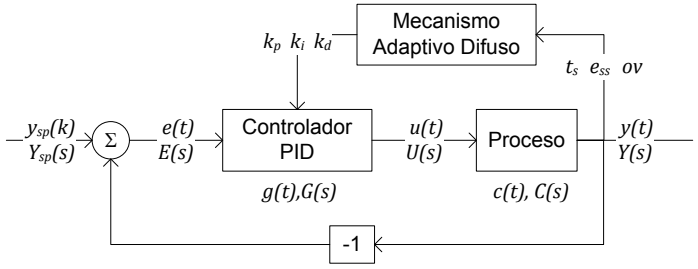


2.2 MECANISMO ADAPTIVO DIFUSO (MAD)

El mecanismo adaptivo difuso proporciona la realimentación adaptiva, estos se diseñan para que se adapte a la dinámica del proceso o solamente para proporcionar sintonía; algunos de los controladores establecidos son: el de ganancia planeada, el regulador autosintonizable, por modelo de referencia y el control dual[10].

El éxito de los controladores adaptivos mencionados se basa en la determinación de los parámetros del proceso en línea (*on-line*), para lo cual se hace necesario manejar un algoritmo de identificación de parámetros como el algoritmo de estimación de mínimos cuadrados que requiere para su cálculo operaciones con matrices, la implementación de este tipo de algoritmos requiere bastante recursos de procesamiento, se ha tratado de reducirlos o simplificarlos, aun así el costo de procesamiento continua siendo alto, esto se debe a que su enfoque es el conocimiento preciso de los parámetros por esto se debe utilizar sistemas computacionales con amplia capacidad de procesamiento. En la actualidad se han desarrollado mecanismos adaptivos que enfocan de formas distintas la adaptación, el procesamiento se centra en aprender del proceso o interactuar con este por medio de un sistema inteligente. Los recursos de procesamiento se tratan de repartir de forma más adecuada, obteniendo un equilibrio entre significancia y precisión de la información, así surge un nuevo tipo de control, el control inteligente y es donde se enmarca el mecanismo adaptivo propuesto.

Figura 13. Diagrama de bloques del controlador PID adaptivo difuso



"Un sistema de control inteligente se puede definir como una combinación de la teoría del control clásico con la inteligencia artificial (IA), bajo esta simple definición cualquier sistema de control el cual involucre lógica difusa, redes neuronales, planes de aprendizaje externo, algoritmos genéticos, programación genética o cualquier combinación de estos es designado como control inteligente"[11]. En este caso en particular la herramienta de la inteligencia artificial utilizada es la lógica difusa que se encarga de la realimentación

[10] ASTROM y WITTENMARK, Op. cit., p.18.

[11] ZILOUCHIAN, Ali y JAMSHIDI, Mo. Intelligent Control Systems Using Soft Computing Methodologies. United States of America: CRC Press LLC, 2001. p.212.

adaptiva. La sintonía del controlador PID se basa en una calibración adecuada de las ganancias, esta tarea se ejecuta por un operador experto en el proceso, siguiendo unas reglas basadas en la experiencia, el MAD reemplaza la tarea del operador, que sigue las mismas reglas ejecutadas por el sistema difuso dotando al controlador de sintonía automática y adaptabilidad. La sintonía automática se refiere a encontrar las ganancias apropiadas del controlador PID para que después permanezcan constantes. La adaptabilidad se refiere a variación de las ganancias a lo largo del proceso (on-line) respondiendo a los transitorios cumpliendo los requerimientos de sobrepaso, rechazo de disturbios en la carga y tiempos de proceso. La capacidad de adaptarse al sistema siguiendo las reglas que usa un operador experto se debe a la lógica difusa que provee los medios para emular el razonamiento humano.

2.2.1 Lógica difusa. Se encuentra lógica difusa en varias ocasiones cotidianas, cuando se maneja una situación o experiencia y se hace uso del conocimiento humano, que generalmente no es perfecto debido a que se tiene una habilidad limitada para percibir el mundo, otra limitación es la vaguedad inherente del lenguaje natural que se usa para describir y compartir la información. La lógica difusa sirve para representar la vaguedad e imprecisión de los conceptos empleados en el lenguaje natural (conocimiento) o sea sirve para modelar la imprecisión propia de los conceptos humanos y así emular el razonamiento humano, Zadeh dice: "La lógica difusa trata de copiar la forma en que los humanos toman decisiones, lo curioso es que, aunque baraja información imprecisa, esta lógica es en cierto modo muy precisa, se puede aparcar un coche en muy poco espacio sin darle a de atrás, suena a paradoja pero es así."[12]

Para el manejo de esta lógica se introduce el concepto de conjunto difuso, bajo el que reside la idea de que los elementos sobre los que se construye el pensamiento humano no son números sino etiquetas lingüísticas, así permite representar el conocimiento común, que es mayoritariamente del tipo lingüístico cualitativo y no necesariamente cuantitativo, no obstante la lógica difusa permite trabajar a la vez con datos numéricos y términos lingüísticos, los términos lingüísticos son inherentemente menos precisos que los datos numéricos pero en muchas ocasiones aportan una información más útil para el razonamiento humano, un ejemplo se muestra en la siguiente figura.

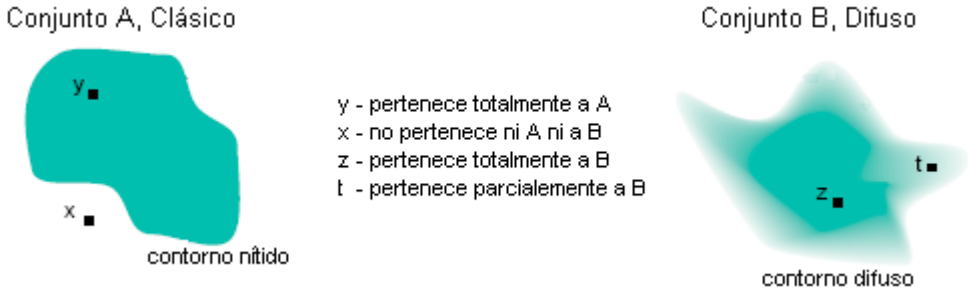
[12] Fuzzy Logic Toolbox User's Guide. The Mathworks Inc. Disponible en Internet: URL: http://www.mathworks.com/access/helpdesk/help/pdf_doc/fuzzy/fuzzy.pdf, p.1.

Figura 14. Importancia de la significancia sobre la precisión

Fuente: The Mathwork Inc. Fuzzy Logic Toolbox User's Guide, 2007.

2.2.1.1 Teoría de los conjuntos difusos. En la teoría de conjuntos se parte de las nociones, elemento y conjunto, un conjunto es una colección de elementos los cuales tienen una propiedad en común que les hace susceptibles de pertenecer al conjunto, entre el conjunto y sus elementos se define una relación de pertenencia.

Figura 15. Comparación entre un conjunto clásico y difuso



Fuente: Schneider Electric. Technique notebook 191 - Fuzzy Logic, 2007.

En lógica clásica un elemento (x), pertenece $(\in)$ o no pertenece $(\notin)$ a un conjunto (A), tiene pertenencia absoluta, la expresión matemática mediante una función característica binaria $\varphi\{0,1\}$ es la siguiente:

$$\varphi_A(x) = \begin{cases} 1 \ si \ x \in A \\ 0 \ si \ x \notin A \end{cases} \quad (29)$$

En lógica difusa el elemento (x) no tiene pertenencia absoluta sino gradual al conjunto (A), y su función característica no adopta valores en el conjunto discreto $\{0, 1\}$, sino en el intervalo cerrado [0, 1] de su función de pertenencia μ_A, mediante notación matemática se define un conjunto borroso como:

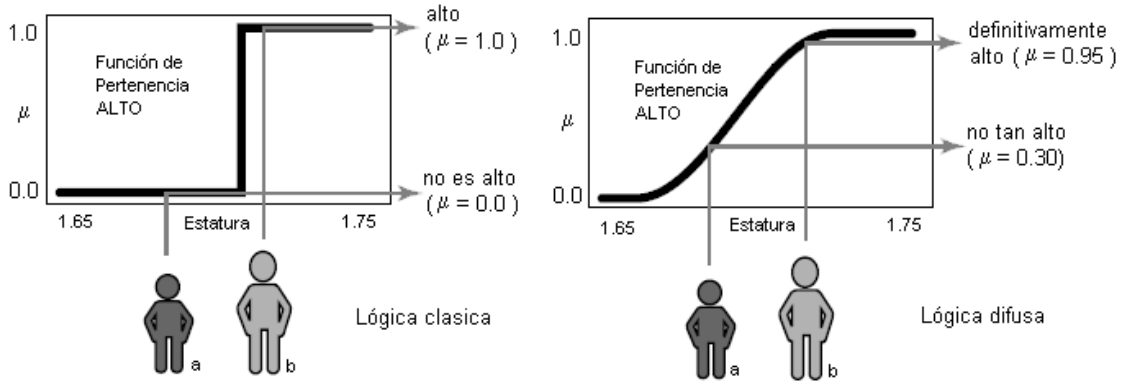
$$A = \{(x, \mu_A(x)) | x \in S\} \quad (30)$$

Donde A es el conjunto que se puede definir con un identificador lingüístico, S es el universo del discurso y μ_A es la función de pertenencia.

2.2.1.2 Función de pertenencia. Se define como $\mu_A: S \rightarrow [0, 1]$, de modo que $\mu_A(x) \in [0, 1]$ es el grado con que el elemento x pertenece al conjunto borroso A, cuando $\mu_A(x) = 0$ el elemento no pertenece al conjunto, y cuando $\mu_A(x) = 1$ pertenece totalmente.

Del ejemplo en ambos casos, el universo del discurso es "Estatura" y el conjunto se describe con el identificador lingüístico "ALTO". Según la lógica clásica la pertenencia del sujeto a es: $\mu_{ALTO}(a) = 0$ no es alto, del sujeto b, $\mu_{ALTO}(b) = 1$ es alto, en lógica difusa, $\mu_{ALTO}(a) = 0.30$ no es tan alto, $\mu_{ALTO}(b) = 0.95$ definitivamente alto, en lógica difusa el resultado es un concepto con cierto grado de vaguedad, igual a los que se usan en el razonamiento humano.

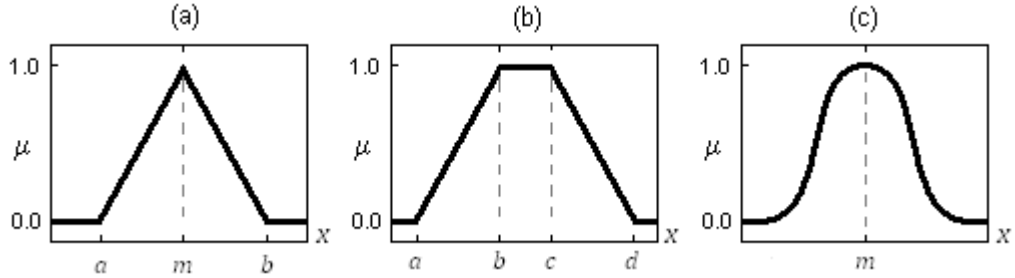
Figura 16. Comparación de funciones de pertenencia



Fuente: The Mathwork Inc. Fuzzy Logic Toolbox User's Guide, 2007.

La forma de la función de pertenencia tiene un cierto componente subjetivo en contraste con la forma rígida objetiva de las funciones características en lógica clásica. Estas funciones pueden adquirir diversas formas y se eligen con un amplio grado de libertad por parte del diseñador, lo que puede traducirse como la posibilidad de incluir cierto conocimiento experto. No obstante en la práctica se tiende a trabajar con las funciones de pertenencia estándares que se muestran en la siguiente figura, ya que son simples de analizar.

Figura 17. Funciones de pertenencia típicas



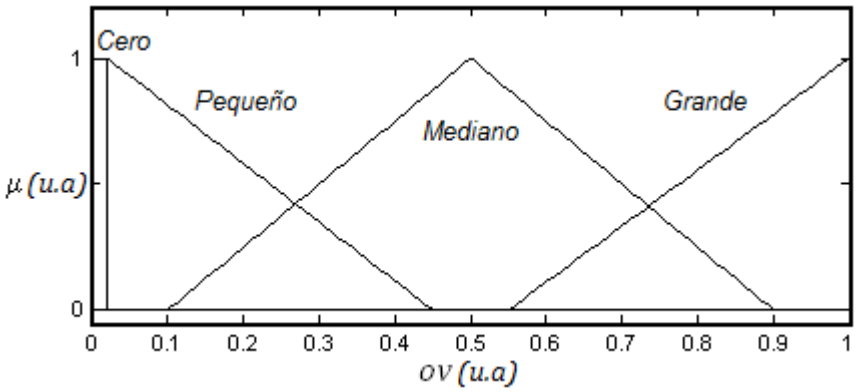
Donde (a) es una función de pertenencia triangular, (b) es una función de pertenencia trapezoidal, (c) es una función de pertenencia gaussiana.
Fuente: JANTZEN, Jan. Design of Fuzzy Controllers, 1998.

Se define la función de pertenencia triangular que se utilizara en el desarrollo del mecanismo difuso que se muestra en la figura 17 (a) con la siguiente expresión.

$$\mu(x) = \begin{cases} 0 & si\ x \leq a \\ \dfrac{x-a}{m-a} & si\ x \in (a, m] \\ \dfrac{b-x}{b-m} & si\ x \in (m, b) \\ 0 & si\ x \geq b \end{cases} \tag{31}$$

Varios conjuntos difusos se pueden definir sobre la misma variable o universo del discurso, por ejemplo en la siguiente figura se observa la definición de los conjuntos, "Cero", "Pequeño", "Mediano" y "Grande" sobre la variable de discurso "ov" o sobrepaso normalizado a la unidad, cada conjunto se define por su función de pertenencia.

Figura 18. Función de pertenencia, variable y término lingüístico



Al diseñar un sistema difuso se deben definir las variables de discurso de entrada y salida, escogiendo sus respectivos conjuntos difusos, se debe tener en cuenta dos aspectos al

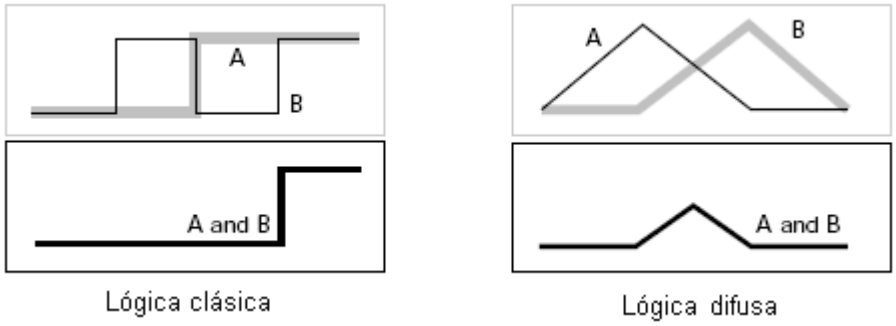
definir los conjuntos, primero, determinar la forma y el ancho del conjunto, segundo, determinar cuántos conjuntos son necesarios. De acuerdo a la teoría de conjuntos difusos el primer aspecto es subjetivo, no obstante se recomienda la función de pertenencia triangular, el ancho se debe escoger de forma que permita sensibilidad al sistema. El segundo aspecto es escoger el número de conjuntos, esto se escoge dependiendo de su ancho, es deseable que los conjuntos se traslapen entre sí alrededor de un 50%, de otra forma los elementos de la variable pueden tomar un grado de verdad pobre en algunos conjuntos y no retornar una conclusión bien definida[13].

2.2.1.3 Operaciones entre conjuntos difusos. Se han definido tres operaciones básicas sobre los conjuntos difusos que operan el grado de pertenencia o grado de verdad $\mu(x)$.

- **Intersección (and).** El operador lógico correspondiente a la intersección de conjuntos es and. El grado de verdad de la proposición $A\ and\ B$ es el mínimo (min) de los grados de verdad de los elementos de A y B, se denota muestra la siguiente expresión:

$$\mu_{A\ and\ B}(x) = min\ (\mu_A(x), \mu_B(x)) \tag{32}$$

Figura 19. Comparación de la operación intersección entre lógica clásica y difusa



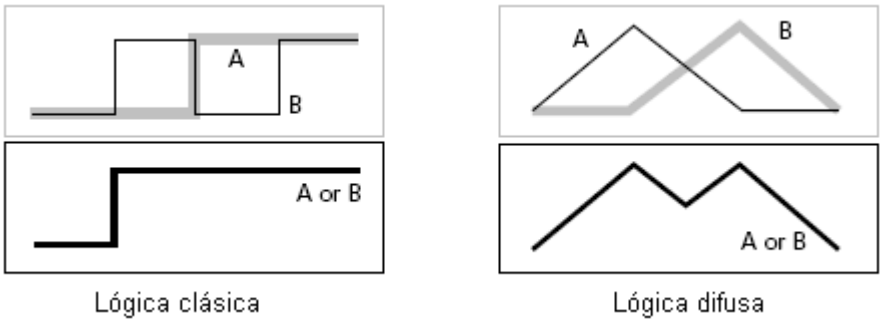
Fuente: The Mathwork Inc. Fuzzy Logic Toolbox User's Guide, 2007.

- **Unión *(or).*** El operador lógico correspondiente a la unión de conjuntos es or. El grado de verdad de la proposición $A\ or\ B$ es el máximo de los grados de verdad de los elementos de A y B, se denota como muestra la siguiente expresión:

$$\mu_{A\ or\ B}(x) = max\ (\mu_A(x), \mu_B(x)) \tag{33}$$

[13] JANTZEN, Jan. Design of Fuzzy Controllers. Denmark: Technical University of Denmark, 1998. p.11.

Figura 20. Comparación de la operación unión entre lógica clásica y difusa

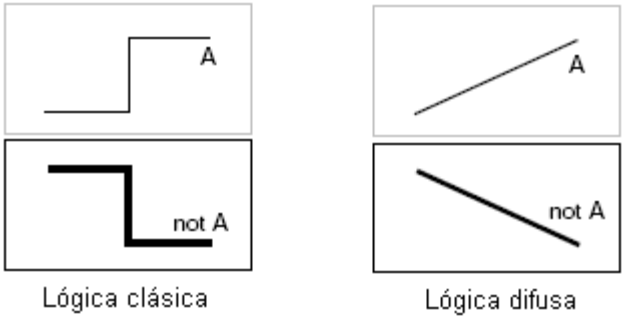


Fuente: The Mathwork Inc. Fuzzy Logic Toolbox User's Guide, 2007.

- **Complemento (not).** El operador lógico correspondiente al complemento es not, el grado de verdad de la proposición $not\ A$ es lo que le falta al grado de verdad para ser uno, se denota como muestra la siguiente expresión:

$$\mu_{not\ A}(x) = 1 - \mu_A(x) \tag{34}$$

Figura 21. Comparación de la operación complemento entre lógica clásica y difusa



Fuente: The Mathwork Inc. Fuzzy Logic Toolbox User's Guide, 2007.

2.2.1.4 Reglas "SI-ENTONCES". La lógica difusa tiene por objetivo poner en práctica la manera de razonar de un ser humano, por tanto se puede considerar parte de la inteligencia artificial. En este caso el conocimiento se redacta en una base de reglas difusas que está compuesta de reglas "SI- ENTONCES", donde las variables de discurso, los conjuntos difusos y los operadores son los sujetos y verbos de la lógica difusa, las reglas "SI-ENTONCES" se usan para formular declaraciones condicionales de la siguiente manera: SI "predicado" ENTONCES "conclusión". Las bases de las reglas difusas al igual que los sistemas expertos clásicos, se apoyan sobre una base de conocimientos que se obtienen de la experiencia humana, es por eso que la lógica difusa puede emular la decisión de un experto humano. Cabe destacar que por medio de estas reglas se puede implementar sistemas una entrada y una salida (SISO) o entrada múltiple y salida múltiple (MIMO).

Una regla difusa tiene dos componentes, predicado y conclusión.

- **Predicado.** Llamado premisa o condición, se compone de las variables de discurso de entrada y sus respectivos conjuntos difusos formando proposiciones que se pueden enlazar con los operadores and o or. Por ejemplo, si se define las variables de discurso de entrada con sus conjuntos difusos así: Sobrepaso (ov) = {"Muy Pequeño" (MP), "Pequeño" (P), "Mediano" (M), "Grande" (G)}se puede formar el predicado:

"Sobrepaso es Muy Pequeño"

- **Conclusión.** Se compone de las variables de discurso de salida y sus respectivos conjuntos difusos formando proposiciones unidas por el operador and, solo se utiliza el operador and debido que al usar alguno de los otros operadores se introduce incertidumbre en el conocimiento. Continuando el ejemplo, si se define la variable de discurso de salida con sus conjuntos así: Incremento de k_d ($INCk_d$)= {"Muy Pequeño" (MP), "Pequeño" (P), "Mediano" (M), "Grande" (G) }, se puede completar la regla agregando la conclusión,

SI "Sobrepaso es Muy Pequeño" ENTONCES "Incremento de k_d es Muy Pequeño"

La anterior regla se basa en un conocimiento adquirido por la experiencia, en este caso se trata la corrección del sobrepaso incrementando la ganancia derivativa.

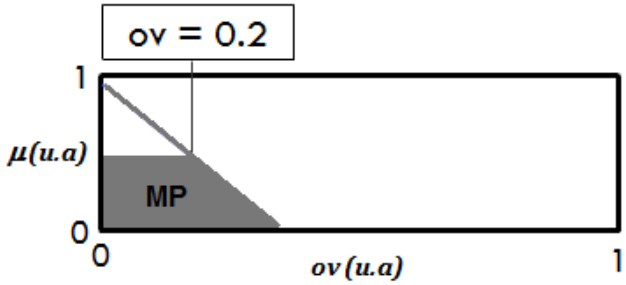
2.2.1.5 Interpretación de una regla "SI-ENTONCES". Interpretar una regla involucra dos partes funcionales, primero, evaluar el predicado el cual implica la fusificación de las entradas y la aplicación de los operadores difusos, segundo, aplicar el método de la implicación obteniendo el resultado de la conclusión y en efecto de la regla.

- **Fusificación y aplicación de operadores.** La fusificación permite pasar los valores de la variable de discurso de entrada del campo real al campo difuso, consiste en determinar el grado de verdad de los valores en su conjunto correspondiente como se especifique en las proposiciones que componen el predicado de la regla, después se opera los grados de verdad de las proposiciones según indique el operador. Tomando como ejemplo el siguiente predicado:

SI "Sobrepaso es Muy Pequeño" ENTONCES "Incremento de k_d es Muy Pequeño"

Si se define el rango de la variable de entrada, Sobrepaso [0, 1], la fusificación del valor Sobrepaso = 0.2 según la anterior regla es:

Figura 22. Fusificación del predicado de la regla

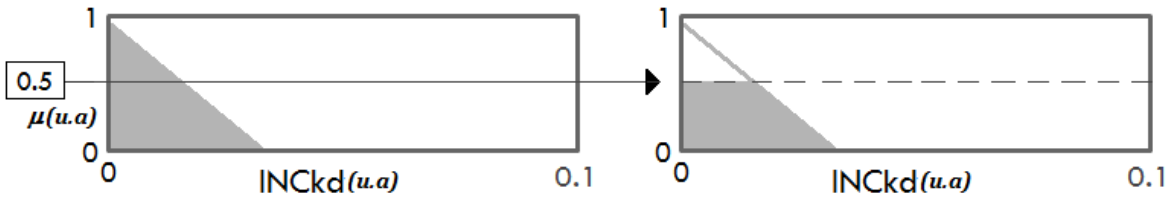


Para la proposición "Sobrepaso es Muy Pequeño", $\mu(0.2) = 0.5$. En resumen se obtiene el grado de verdad del predicado; para obtener el resultado de la regla se debe conocer cuál es la implicación del predicado sobre la conclusión.

- **Método de la implicación.** Indica la medida que el predicado afecta la conclusión, debido a que el predicado tiene un grado de verdad entonces la conclusión también se hace verdadera en ese mismo grado, este método consiste en igualar los grados de verdad del predicado y de la conclusión, en efecto, modifica el grado de pertenencia de la variable de discurso de salida y su conjunto difuso según especifique la regla. El operador utilizado para el método de la implicación es el operador min. El resultado de la implicación es un conjunto difuso afectado por el grado de verdad del predicado.

Continuando el ejemplo: si se define el rango de la variable de salida incremento de k_d ($INCk_d$)[0, 0.1], la implicación de la regla, SI "Sobrepaso es Muy Pequeño" ENTONCES "Incremento de k_d es Muy Pequeño", se obtiene aplicando el operador min entre el grado de verdad del predicado (0.5) y la conclusión. $0.5 = min\,(0.5,\ INCk_d)$. Solo resta convertir el conjunto resultado del campo difuso al real para solucionar el sistema y obtener la respuesta, esto es a lo que se conoce como defusificación.

Figura 23. Aplicación del método de implicación



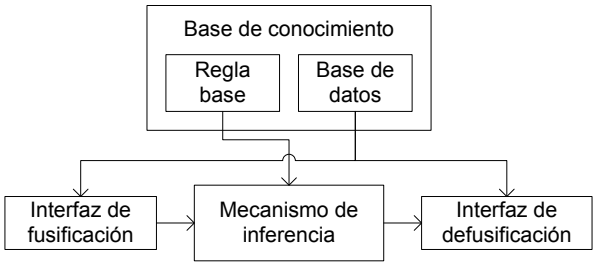
En general una regla por sí sola no hace mucho, un sistema difuso se compone por un regla base, formada por varias reglas "SI-ENTONCES". Para sacar una conclusión de una regla base se debe analizar en conjunto todas sus reglas, esto se hace por medio de un mecanismo de inferencia establecido, el mecanismo de inferencia difusa más utilizado es el de

Mamdani, este se encarga de interpretar cada una de las reglas que conforman el sistema, como se explico anteriormente, sintetizando el resultado de todas las reglas en un conjunto difuso y eventualmente obtener la respuesta aplicando la defusificación.

2.2.1.6 Mecanismo de inferencia difusa de Mamdani. El mecanismo de inferencia es el proceso que formula la relación de las entradas con las salidas, este proceso involucra todas las piezas descritas anteriormente, conjuntos, funciones de pertenencia, operadores y reglas "SI-ENTONCES". Permite analizar el efecto de las entradas sobre cada regla que compone la regla base, se analiza todas las reglas en grupo obteniendo un conjunto de conclusión para cada regla, todos los conjuntos se agregan conformando solamente uno, el cual se debe defusificar para pasar del campo difuso al real obteniendo la respuesta final del sistema para una entrada en particular.

Si se hace un barrido de las entradas, con ayuda del mecanismo de inferencia se puede construir la grafica de correspondencia entre las entradas y salidas, según el número de entradas de dos o tres dimensiones, esto se conoce como la superficie del sistema.

Figura 24. Estructura de un sistema difuso



Fuente: The Mathwork Inc. Fuzzy Logic Toolbox User's Guide, 2007.

De la grafica anterior, la base conocimiento es la experiencia del experto redactada, en una regla base, conformada por reglas "SI-ENTONCES", y un base de datos que contiene la definición de los conjuntos difusos de las variables de entrada y salida. La interfaz de fusificación se refiere a la conversión del campo real al difuso, la interfaz de defusificación es proceso inverso. Teniendo en cuenta la grafica anterior el proceso del mecanismo de inferencia según Mamdani se puede dividir en 5 pasos, que son: Paso 1 Fusificación de las entradas, Paso 2 Aplicación del operador difuso, Paso 3 Aplicación del método de implicación, Paso 4 Agregación de todos los conjuntos obtenidos, y Paso 5 Defusificación.

- **Paso 1 y 2 Fusificación de las entradas y aplicación del operador.** Se fusifica cada entrada y después se aplica el operador según especifique la regla, se hace de igual forma al análisis de una regla "SI-ENTONCES" (…ver numeral 2.2.1.5…), pero en este caso se realiza en grupo para todas las reglas que conforman la regla base, de esta forma se obtiene

el grado de verdad del predicado para cada regla dependiendo de las entradas, y se puede continuar con el siguiente paso.

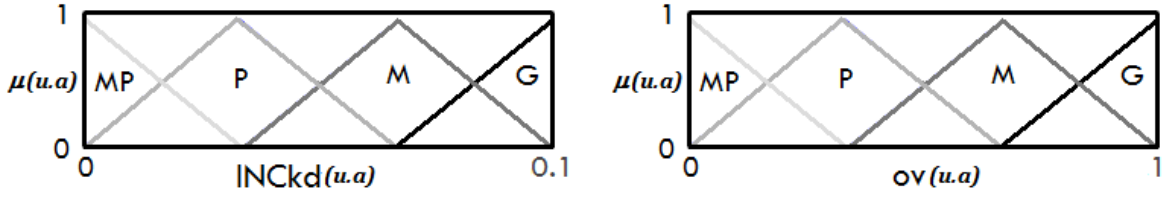
- **Paso 3 Aplicación del método de implicación.** Se aplica el operador "min" entre el predicado y la conclusión de cada regla, se hace de igual forma como se explico en el método de implicación, pero en este caso se realiza en grupo para todas las reglas, el resultado es un conjunto afectado por el grado de verdad del predicado por cada regla del sistema como indica la figura 26. Continuando con el ejemplo, si se define la regla base así:

Tabla 6. Reglas para el sistema difuso determinación de $INCk_d$

Regla 1	SI ov es MP ENTONCES $INCk_d$ es MP.
Regla 2	SI ov es P ENTONCES $INCk_d$ es P.
Regla 3	SI ov es M ENTONCES $INCk_d$ es M.
Regla 4	SI ov es G ENTONCES $INCk_d$ es G.

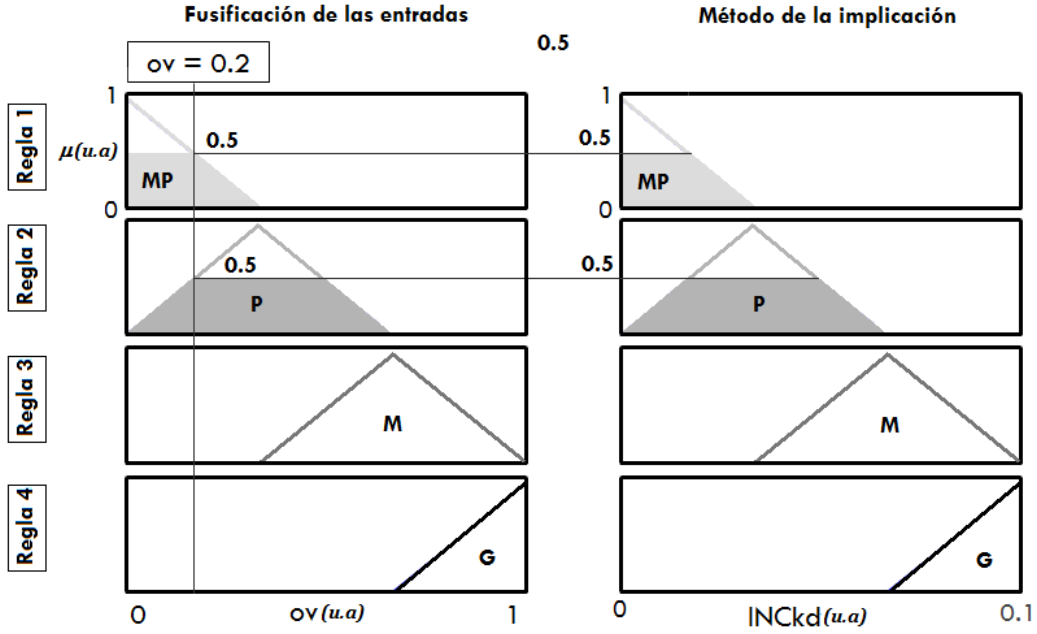
Y se define la forma de los conjuntos como indica la siguiente figura:

Figura 25. Conjuntos difusos para las variables de entrada y salida



Se resume aplicación de los pasos 1 al 3, para las entradas, Sobrepaso = 0.2 en la siguiente figura.

Figura 26. Aplicación de los pasos 1 al 3, del método de inferencia de Mamdani



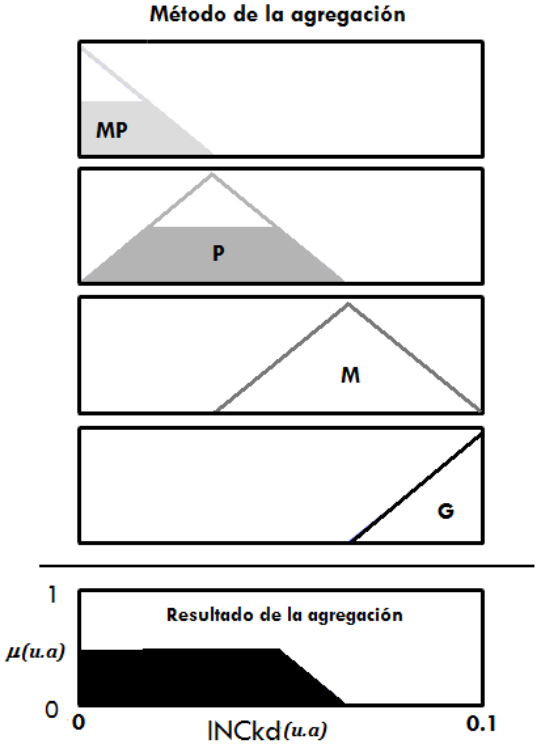
En la anterior figura se observa que el resultado de la implicación son los dos conjuntos de la variable de discurso de salida $INCk_d$ afectados por el grado de verdad del predicado correspondiente según la regla.

- **Paso 4 Agregación.** La decisión se basa en el peso de todas las reglas del sistema difuso, las salidas de las reglas se deben combinar de alguna manera para tomar una decisión, la agregación es el proceso por el cual los conjuntos difusos que representan las salidas de cada regla se combinan en un único conjunto difuso.

La agregación solamente ocurre una vez para cada variable de salida, la entrada del proceso de agregación son los conjuntos truncados que retorna el método de implicación por cada regla, el resultado del proceso de agregación es un conjunto difuso por cada variable de salida. El método de la agregación utiliza el operador "max" o lo que es igual realizar la unión de conjuntos.

Continuando con el ejemplo anterior, se aplica el método de agregación uniendo todos los conjuntos para obtener uno solo, como indica la siguiente figura.

Figura 27. Aplicación del paso 4 la agregación, del método de inferencia de Mamdani



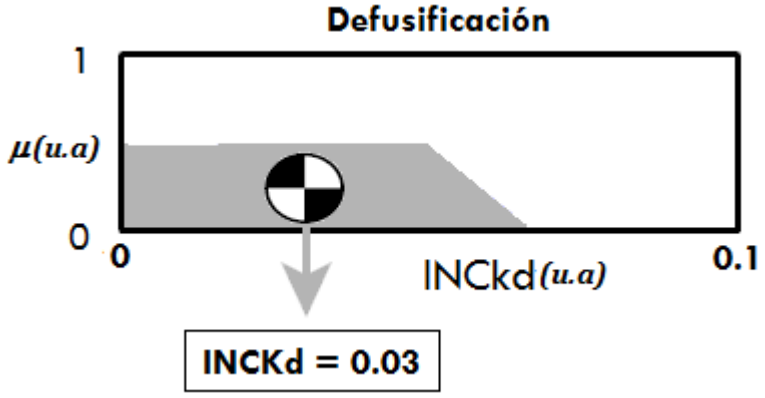
- **Paso 5 Defusificación.** La entrada para el proceso de defusificación es un conjunto difuso y la salida es un número escalar, el método de defusificación que más se utiliza es el método del centro de gravedad o del centroide. El valor de salida (u) es la abscisa bajo el centro de gravedad del conjunto difuso. Numéricamente se calcula de la siguiente forma.

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \tag{35}$$

Donde x_i es un punto del universo discurso y $\mu(x_i)$ es su grado de pertenencia, la expresión se puede interpretar como el peso promedio de los elementos del conjunto resultado. Este método tiene buena complejidad computacional.

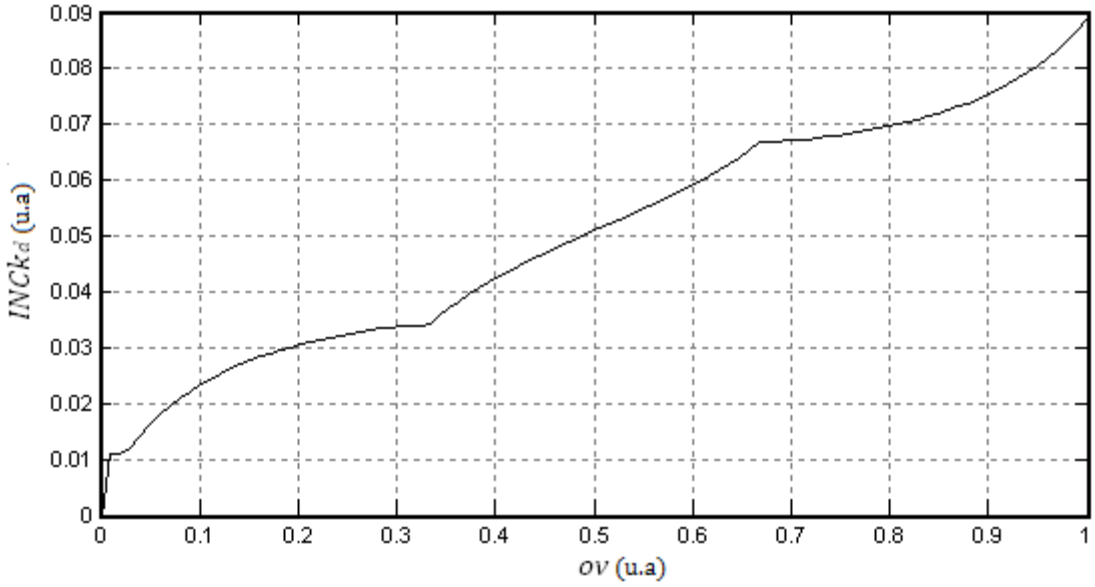
Para finalizar el ejemplo, solucionando el sistema para las entradas determinadas, se defusifica el conjunto resultado de la agregación, para esto se toma una resolución de 30 puntos y se aplica el método del centroide con ayuda de Matlab el resultado es como indica la siguiente figura.

Figura 28. Resultado de la defusificación por el método del centroide



Por medio del ejemplo se siguió el proceso del mecanismo de inferencia difusa de Mamdani, para construir la grafica del sistema se hace un barrido en todo su rango de la entrada "Sobrepaso" y con ayuda de Matlab se construye una grafica de correspondencia con la salida "Incremento de k_d"

Figura 29. Grafica de correspondencia entre las entradas y la salida del sistema difuso



2.2.2 Mecanismo adaptivo difuso 1 (MAD1). El objetivo del mecanismo adaptivo es encontrar de alguna forma las ganancias adecuadas para que el controlador cumpla su estrategia de control regulando la salida de proceso dentro de las especificaciones en cuanto a estabilidad (Est), sobrepaso (ov), error en estado estable (e_{ss}) y tiempo de subida (t_s).

El MAD1 se encarga de variar k_p, k_i, y k_d, en base de incrementos. Cuando se varía las ganancias del controlador PID se afectan en forma significativa algunas características de la variable de proceso $y(k)$ como ov, e_{ss}, y t_s, para mejorar las características del proceso el MAD1 calcula una nueva ganancia según el incremento que se obtiene del sistema difuso, la base de conocimiento del sistema difuso se basa en el estudio practico de los efectos del controlador PID sobre los procesos, así se escoge cada incremento cómo lo haría un experto en función de la evolución del proceso, por consiguiente el mecanismo que se propone varia las ganancias del controlador emulando el razonamiento de un experto en procesos.

El éxito del mecanismo adaptivo (MA) se basa en el cálculo de los incrementos por parte del sistema difuso, y del sistema difuso en la redacción del conocimiento base. Los componentes que integran el sistema difuso, la regla base, las variables, y conjuntos se pueden deducir de la siguiente tabla.

Tabla 7. Efectos de la variación de las ganancias k_p, k_i, y k_d sobre un proceso

	Tiempo de subida	Sobrepaso	Tiempo de asentamiento	Error en estado estable	Estabilidad
Incremento k_p	*Disminuye*	Aumenta	Disminución menor	Disminuye	Degrada
Incremento k_i	Disminución menor	Aumenta	Aumenta	*Disminución considerable*	Degrada
Incremento k_d	Disminución menor	*Disminución considerable*	Disminuye	Cambio Menor	Mejora

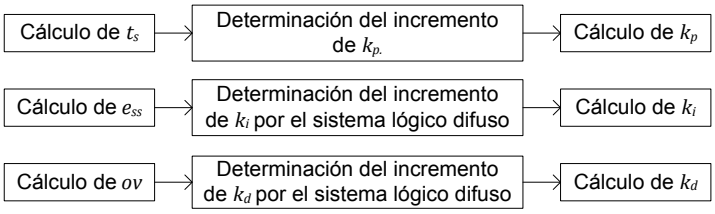
Fuente: JANTZEN, Jan. Tuning of Fuzzy PID Controllers, 1998.

En la tabla anterior se observa que cada una de las ganancias varía todas las características del proceso, sin embargo también es evidente que cada ganancia tiene dominio sobre una característica en particular:

- La ganancia proporcional (k_p) sobre el tiempo de subida (t_s)
- La ganancia integral (k_i) sobre el error en estado estable (e_{ss})
- La ganancia derivativa (k_d) sobre el sobrepaso (ov)

De la anterior deducción, cada ganancia se adapta utilizando un MA independiente, k_p según t_s, k_i según e_{ss}, y k_d según ov, ya que cumplen la correspondencia en forma dominante. De igual forma se define un sistema difuso para cada ganancia.

Figura 30. Diagrama de bloques del mecanismo adaptivo propuesto



Se nota un detalle particular en las entradas del sistema difuso, es que el tiempo de subida no es una medida relativa como si lo son el error en estado estable y el sobrepaso que se pueden definir como un porcentaje, debido a esto el tiempo de subida tendrá un tratamiento adaptivo diferente. Teniendo en cuenta esto se divide el MAD propuesto se divide en tres partes:

- Mecanismo adaptivo de k_p
- Mecanismo adaptivo difuso de k_i y k_d
- Funcionamiento del MAD1

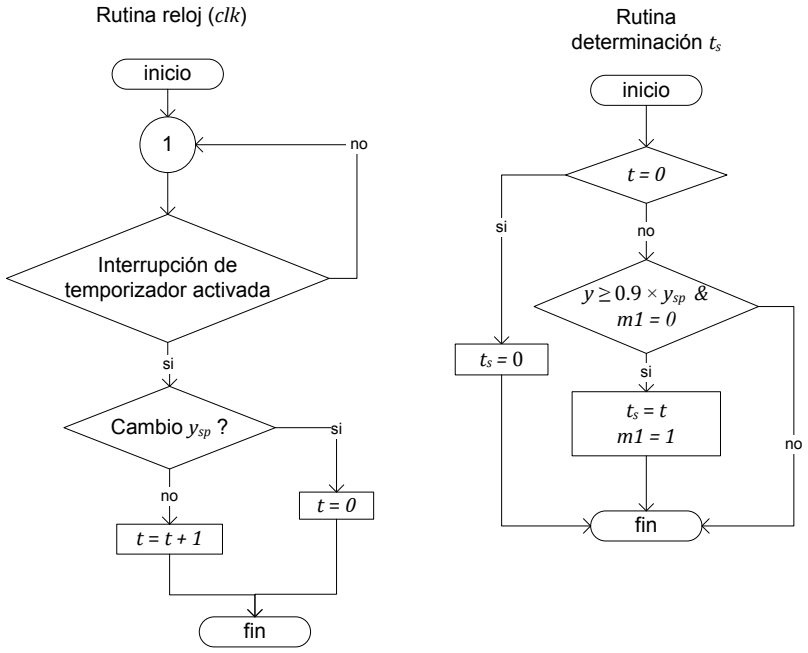
2.2.2.1 Mecanismo adaptivo de k_p. Se encarga de encontrar el k_p más adecuado dependiendo de t_s, debido a que t_s no se puede expresar como una medida absoluta sino relativa, entonces se emplea un método distinto al difuso para adaptar esta ganancia, como lo sugiere la anterior figura el proceso se divide en tres partes, organizadas así:

- Cálculo de t_s
- Determinación del incremento de k_p $(INCk_p)$ y cálculo de k_p

• **Cálculo t_s.** El tiempo de subida se define como "el tiempo requerido para que la respuesta al escalón se eleve del 10 al 90% de su valor final"[14], en este caso para determinar el valor de t_s, se hace una pequeña variación de su definición, sin embargo esto es manejable para el sistema difuso, t_s se toma como el tiempo desde que el escalón inicia hasta que la señal llega al 90% de su valor final. Para el cálculo del tiempo de subida se hace necesario manejar un reloj (t), cada ves que ocurre un periodo de muestreo (h) el reloj se incrementa y cuando un escalón ocurre este se reinicia, el reloj lleva la cuenta del tiempo real entre una entrada escalón y otra, o lo que es lo mismo mide el tiempo real entre transitorios.

[14] KUO, Op. cit., p.386.

Figura 31. Diagrama de flujo de la rutina reloj y la rutina cálculo de t_s



Se pueden distinguir dos rutinas, la rutina reloj se encarga de medir el tiempo real entre un cambio de setpoint (y_{sp}), para contar con precisión la base de tiempo (h) se maneja con un temporizador configurado con interrupción. La rutina cálculo de t_s es simplemente la captura de t cuando se cumple la definición de tiempo de subida. Se maneja una variable auxiliar m_1 esta se encarga de que una vez calculado t_s, este no se modifique para valores de $y > 0.9$.

- **Determinación del incremento de k_p ($INCk_p$) y cálculo de k_p.** El incremento de k_p afecta al tiempo de subida de forma que este disminuye, intuitivamente se debe incrementar k_p hasta que el tiempo de subida sea optimo, para esto se elabora una ecuación en el cual se tiene en cuenta el progreso de t_s dependiendo $INCk_p$, entonces, cuando t_s no muestre progreso ante los incrementos de k_p no tendría sentido que se siga con el incremento, esto se entiende como una sintonía adecuada de k_p en función a su característica particular t_s.

Según la tabla 7 k_p tiene cierta dominancia sobre ov, siendo directamente proporcionales, esta relación no se tiene en cuenta cuando se calcula $INCk_p$ sin que tenga importancia ya que el sobrepaso se controla por medio su ganancia dominante k_d. Como no se puede generalizar un tiempo de subida adecuado para todas las plantas, porque tienen constantes de tiempo diferentes, solo se puede calcular un porcentaje de mejoramiento, el sistema de determinación de $INCk_p$ cuenta con dos pasos, primero, se mide el t_s del primer transitorio

y se hace un incremento inicial de k_p, segundo se mide el t_s del segundo transitorio y se comparan como una razón (r).

$$r = \frac{t_s}{t_{so}} \tag{36}$$

Donde t_s es el tiempo de subida del transitorio reciente y t_{so} es el tiempo de subida del transitorio pasado. Cuando t_s mejora ósea disminuye, $r < 1$, cuando t_s no mejora o aumenta, $r \geq 1$. Se deduce de la anterior ecuación, que el término $r \rightarrow 1$ cuando t_s ya no tiende a mejorar, teniendo en cuenta esto se elabora la siguiente ecuación:

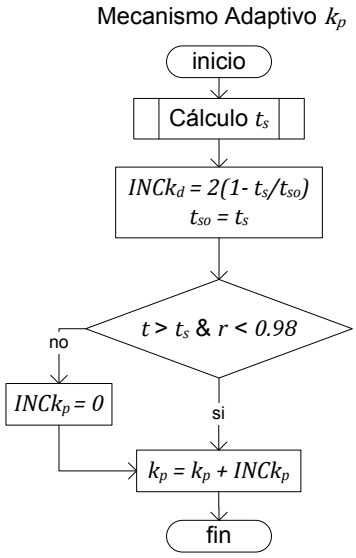
$$INCk_p = \alpha(1 - r) \tag{37}$$

La resta hace que el incremento disminuya cuando t_s ya no tienda a mejorar, el coeficiente α es una constante de proporción del incremento, solo se puede deducir de forma experimental y su función es aumentar la velocidad de adaptación de k_p. En este caso $\alpha = 2$ y k_p se calcula como indica la siguiente ecuación.

$$k_p = k_p + INCk_p \tag{38}$$

El cálculo de $INCk_p$ y por ende de k_p, se realiza una vez cada transitorio, sin embargo se necesita una condición para que k_p no se incremente indefinidamente, el incremento k_p es cero si t_s y t_{so} son iguales, esto no sucede idealmente en todos los transitorios, entonces se condiciona la ecuación 37 con el termino $r < 0.98$, como se muestra en la siguiente figura, cuando se cumple esta condición t_s empieza a mostrar una razón de mejora leve, esto significa que aunque k_p se incremente t_s no puede mejorar significativamente, entonces no tendría sentido incrementar k_p, esto se entiende como una sintonía satisfactoria de k_p.

Figura 32. Diagrama de bloques del mecanismo adaptivo de k_p



En el diagrama anterior se describe los pasos que ejecuta el mecanismo adaptivo de k_p, la rutina que calcula k_p por medio de la anterior ecuación, que se observa en el diagrama de bloques que se debe condicionar para que se ejecute solamente una vez cada transitorio.

2.2.2.2 Mecanismo adaptivo difuso de k_i y k_d. Se explica el MAD de k_i y k_d en forma paralela ya que son muy similares aunque cada uno es independiente del otro, como sugiere la deducción de la tabla 7, los MAD se encargan de encontrar las ganancias k_i y k_d más adecuadas dependiendo de e_{ss} y ov respectivamente, de igual forma las entradas para los sistemas difusos que determinan los incrementos de k_i ($INCk_i$) y de k_d ($INCk_d$) son e_{ss} y ov respectivamente, los procesos se divide en dos partes:

- Cálculo de e_{ss} y ov

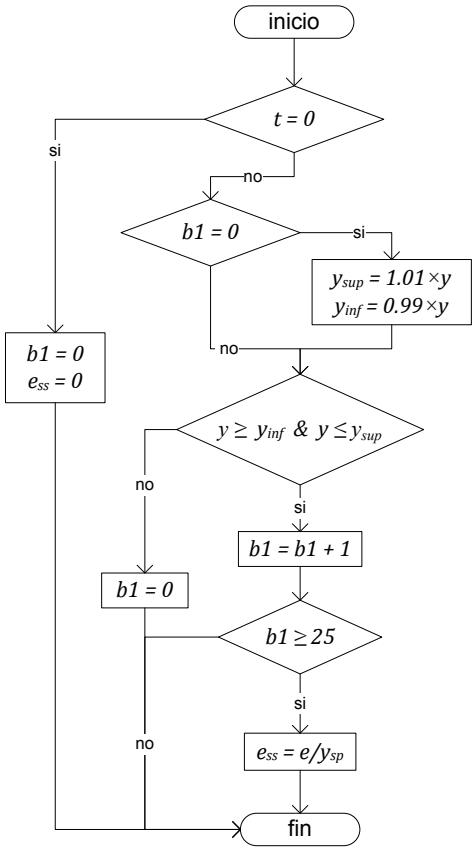
- Determinación del incremento de k_i ($INCk_i$) y de k_d ($INCk_d$) por el sistema difuso y cálculo de k_i y k_d.

• **Cálculo de e_{ss}.** "La diferencia entre la variable de proceso $y(k)$ y el punto de referencia (y_{sp}) en estado estable se define como error en estado estable (e_{ss})"[15]. El estado estable del proceso se define cuando la variable llega a un equilibrio y el efecto transitorio llega a su final. Para efectos de cálculo se define los límites de variación de la variable $y(k)$ en estado estable, entre un 2%, esto se realiza con ayuda de las variables y_{sup} y y_{inf}, para

[15] Ibit., p.365.

el cálculo se utiliza una variable adicional b_1, a medida que $y(k)$ se mantenga entre el 2% de variacion b_1 se incrementa en 1, entonces sí y solo si $b_1 > 25$, se considera que $y(k)$ entra en estado estable, b_1 actúa como una variable de manejo de error, así pues el cálculo se protege sobre medidas erróneas y se asegura que realmente el transitorio ha terminado. Después de cumplir esta condición se puede calcular e_{ss} fácilmente, esto se hace igualando $e(k) = e_{ss}$, y se divide entre y_{sp} para normalizar su valor. Se calcula e_{ss} una vez cada transitorio utilizando nuevamente la rutina de reloj (clk) para su sincronización.

Figura 33. Diagrama de flujo para el cálculo de e_{ss}

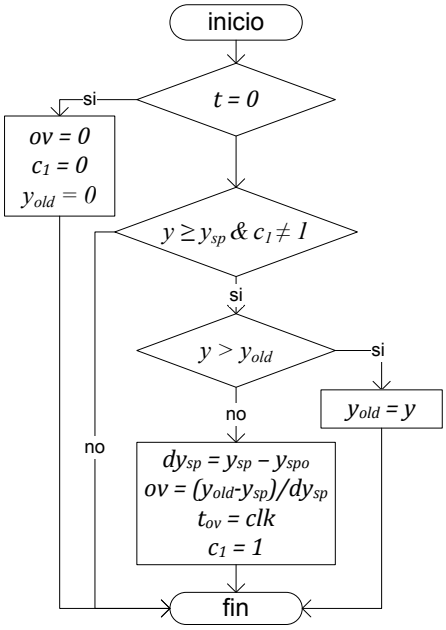


- **Cálculo de ov.** "ov es el sobrepaso máximo, se define como el mayor valor que ocurre en variable de proceso $y(k)$ después de superar a y_{sp}"[16], el tiempo en el que ocurre se denomina tiempo de sobrepaso (t_{ov}), ov se calcula para cada transitorio, se sincroniza el cálculo con ayuda de la rutina reloj (clk) reiniciando las variables.
Cuando $y(k)$ sobrepasa a y_{sp} entonces la definición se cumple y se compara $y(k)$ actual con el anterior (y_{old}), cuando se obtiene el mayor valor de $y(k)$ se aplica la ecuación

[16] Ibit., p.385.

$ov = y_{old} - y_{sp}$ y se captura t_{ov} con ayuda de clk , la variable dy_{ss} es la diferencia entre el escalón anterior y el actual, esta medida es necesaria para normalizar el sobrepaso, también se utiliza una variable auxiliar c_1 para evitar que el cálculo se haga más de una vez en cada transitorio.

Figura 34. Diagrama de flujo para el cálculo de ov



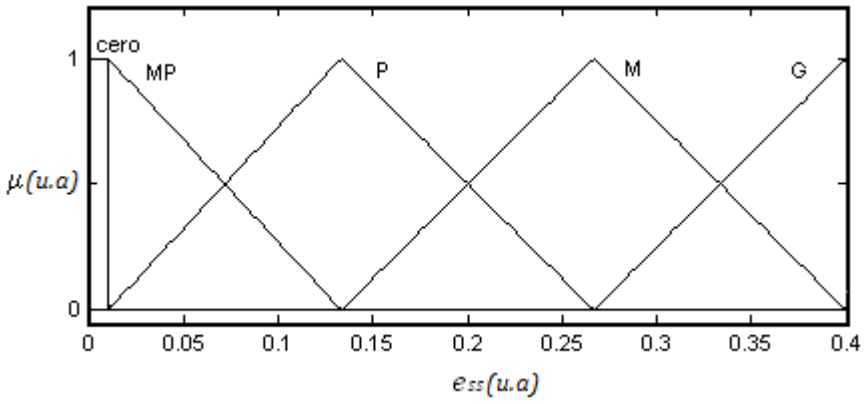
- **Determinación del incremento de k_i ($INCk_i$) y de k_d ($INCk_d$) por el sistema difuso y cálculo de k_i y k_d.** Se han escogido los siguientes conjuntos difusos para las variables de entrada:

  - e_{ss} = {"Cero", "Muy Pequeño" (MP), "Pequeño" (P), "Mediano" (M), "Grande" (G)}

  - ov = {"cero", "Muy Pequeño" (MP), "Pequeño" (P), "Mediano" (M), "Grande" (G)}

Cada variable se describe por 4 identificadores lingüísticos o conjuntos difusos significativos, como se puede observar se agrega un quinto conjunto que es el conjunto "Cero", este tiene una particularidad y sirve para indicar que la característica esta dentro del rango normal permitido. Como se observa en las figuras 35 y 36 para las dos variables, se utiliza funciones de pertenencia triangulares de forma que se traslapen alrededor del 50%, esto siguiendo las normas de diseño recomendadas.

Figura 35. Conjuntos difusos de la variable de entrada e_{ss}



Como se observa en la figura anterior el rango establecido del error en estado estable es de [0, 0.4], esto es debido a que se ha normalizado, evidentemente la normalización tiene un rango de [0, 1], pero ya que es muy inusual encontrar un sistema con error en estado estable igual al 100%, no hay necesidad de colocar conjuntos difusos en el rango excluido, si por algún motivo en el sistema el valor de esta variable se sale del rango establecido, estas se saturan al valor mínimo y máximo permitido.

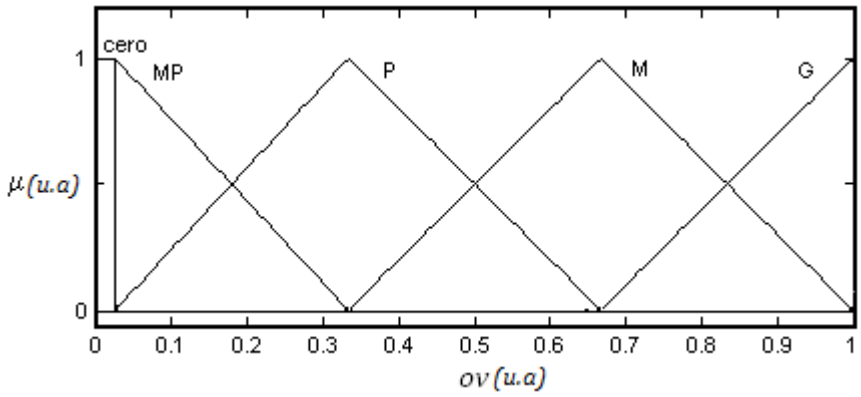
Existe una particularidad en este sistema, el conjunto difuso "Cero" que tiene un rango de [0, 0.01), en este caso este conjunto es necesario para indicar al sistema difuso que el error en estado estable esta dentro de los límites permitidos, y eventualmente k_i no sufra incrementos indeseados, entiéndase esto como si el error en estado estable es normal y no hay necesidad de incrementar k_i ya que se ha adaptado satisfactoriamente.

Tabla 8. Rango de los conjuntos difusos de la variable e_{ss}

Cero	Muy pequeño	Pequeño	Mediano	Grande
[0, 0.01)	[0.01, 0.133)	(0.01, 0.266)	[0.133, 0.4)	(0.266, 0.4]

Estos conjuntos como se explico anteriormente se definen para describir la variable en forma difusa, y dependiendo de cada regla calcular su implicación.

Figura 36. Conjuntos difusos de la variable de entrada ov



Como se observa en la figura anterior el rango del sobrepaso es de [0,1], esto es debido a que se ha normalizado, si por algún motivo en el sistema el valor del sobrepaso se sale del rango establecido, estas se saturan al valor mínimo y máximo permitido.

Existe una particularidad en este sistema; el conjunto difuso cero que tiene un rango [0, 0.01), en este caso este conjunto se hace necesario para indicar al sistema difuso que el sobrepaso esta dentro de los límites permitidos como indica la siguiente tabla, o sea, el sobrepaso es normal y no se necesita incrementar k_d porque se ha adaptado satisfactoriamente.

Tabla 9. Rango de los conjuntos difusos de la variable ov

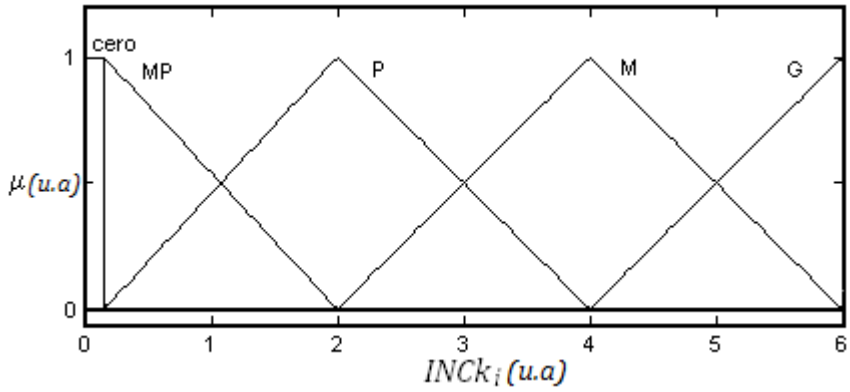
Cero	Muy pequeño	Pequeño	Mediano	Grande
[0, 0.01)	[0.01, 0.333)	(0.01, 0.666]	[0.333, 1)	(0.666, 1]

Evidentemente, la variable ov se relaciona con k_d y e_{ss} se relaciona con k_i, esto como ya se explico debido a la dominancia correspondiente de cada una de las ganancias. Asi pues las variables de salida son, incremento de k_i ($INCk_i$) y de k_d ($INCk_d$), se han seguido las mismas pautas de diseño mencionadas anteriormente, describiendo las variables de salida con los mismos identificadores lingüísticos, pues resulta más intuitivo al definir las reglas "SI-ENTONCES".

- $INCk_i$ = {"Cero", "Muy Pequeño" (MP), "Pequeño" (P), "Mediano" (M), "Grande" (G)}

- $INCk_d$= {"Cero", "Muy Pequeño" (MP), "Pequeño" (P), "Mediano" (M), "Grande" (G)}

Figura 37. Conjuntos difusos de la variable de salida $INCk_i$

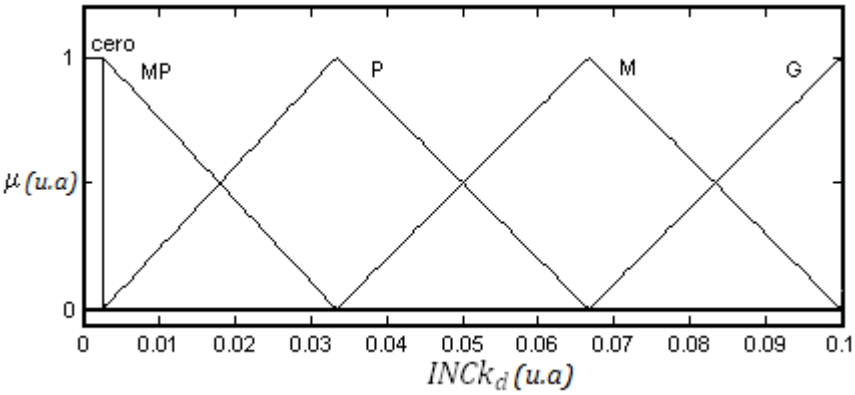


El rango de la variable $INCk_i$ es [0, 6], este valor se ha es aproximadamente igual al 10% del mayor valor que puede llegar a tomar k_i. También existe un conjunto "cero", este indica que el incremento debe ser nulo, y hace correspondencia al conjunto cero de la variable de entrada e_{ss}.

Tabla 10. Rango de los conjuntos difusos de la variable $INCk_i$

Cero	Muy pequeño	Pequeño	Mediano	Grande
0	[0, 2)	(0, 4)	[2, 6)	(4, 6]

Figura 38. Conjuntos difusos de la variable de salida $INCk_d$



El rango de la variable de salida $INCk_d$ es [0, 0.1], este valor se ha escogido como el 10% del mayor valor que puede llegar a tomar k_d. También existe un conjunto "cero", este indica que el incremento debe ser nulo, y hace correspondencia al conjunto cero de la variable de entrada ov.

Tabla 11. Rango de los conjuntos difusos de la variable $INCk_d$

Cero	Muy pequeño	Pequeño	Mediano	Grande
0	[0, 0.0333)	(0, 0.0666)	[0.0333, 0.0666)	(0.0666, 0.1]

- **Reglas "SI-ENTONCES".** Teniendo en cuenta la tabla 7, efectos de variación de las ganancias, se redacta las siguientes reglas:

Tabla 12. Reglas para el sistema difuso determinación de $INCk_i$

Regla 1	SI e_{ss} es "cero" ENTONCES $INCk_i$ es "cero".
Regla 2	SI e_{ss} es MP ENTONCES $INCk_i$ es MP.
Regla 3	SI e_{ss} es P ENTONCES $INCk_i$ es P.
Regla 4	SI e_{ss} es M ENTONCES $INCk_i$ es M.
Regla 5	SI e_{ss} es G ENTONCES $INCk_i$ es G.

Tabla 13. Reglas para el sistema difuso determinación de $INCk_d$

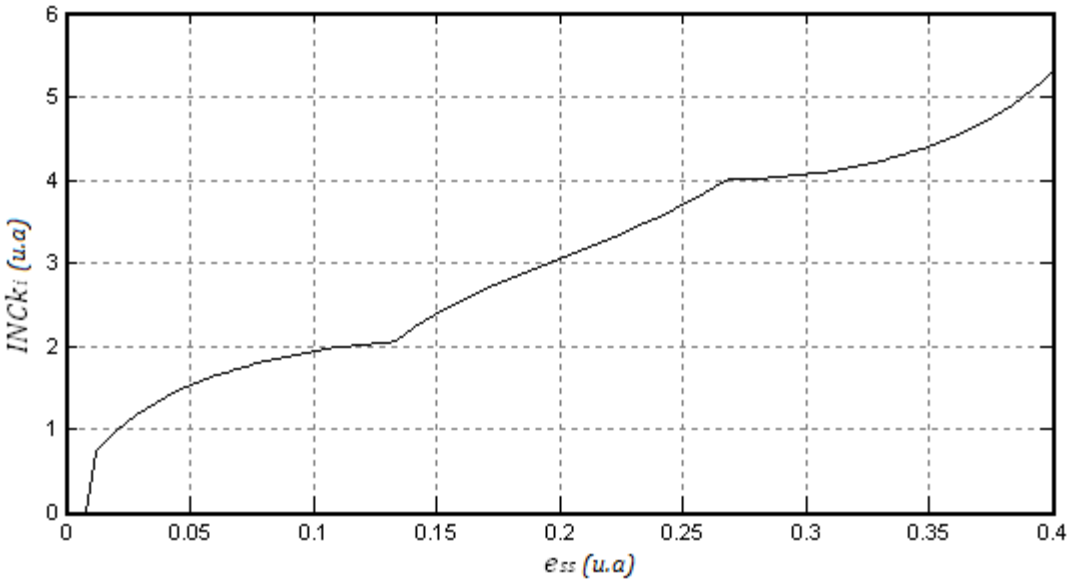
Regla 1	SI ov es "cero" ENTONCES $INCk_d$ es "cero".
Regla 2	SI ov es MP ENTONCES $INCk_d$ es MP.
Regla 3	SI ov es P ENTONCES $INCk_d$ es P.
Regla 4	SI ov es M ENTONCES $INCk_d$ es M.
Regla 5	SI ov es G ENTONCES $INCk_d$ es G.

- **Solución del sistema difuso por el método de mapeo.** Una forma de implementar el sistema difuso propuesto, se hace por medio del mapeo o mapping, debido a que existe una entrada y una salida se puede calcular el valor específico correspondiente, similar a evaluar una función utilizando el mecanismo de inferencia de Mamdani. Teniendo en cuenta la implementación digital, con ayuda de Matlab se aplica el mecanismo de inferencia de Mamdani a cada sistema, el mapping de la entrada y la salida se graba utilizando vectores, donde el coeficiente tenga una correspondencia con la variable independiente y la variable dependiente sea el valor de la variable de salida en este caso del valor del incremento.

Mapping de e_{ss} vs $INCk_i$: Se considera una resolución de 100 puntos, se hace una correspondencia de la variable e_{ss} [0, 0.4] y el coeficiente del vector i [1, 100], $i = f(e_{ss})$.

$$i(e_{ss}) = 247.5e_{ss} + 1 \tag{39}$$

Figura 39. Mapping de e_{ss} vs $INCk_i$



De esta forma la nueva variable independiente es el coeficiente i, este se debe redondear, pues la entrada del vector es un número entero comprendido en el rango [1, 100]. Para esto se puede utilizar una función de redondeo del lenguaje de programación.

Tabla 14. Vector de mapping de e_{ss} vs $INCk_i$

$VINCk_i$ [0, 6]	0	0	0	0.723	0.854	…	4.91	4.99	5.08	5.19	5.31
i [1, 100]	1	2	3	4	5	…	96	97	98	99	100
e_{ss} [0, 0.4]	0	0.004	0.008	0.012	0.016	…	0.383	0.387	0.391	0.396	0.4

Implementación por vector de ov vs $INCk_d$: Se considera una resolución de 100 puntos, se hace una correspondencia de la variable ov[0, 1] y el coeficiente del vector i[1, 100], $i = f(ov)$.

$$i(ov) = 99ov + 1 \tag{40}$$

De esta forma la nueva variable independiente es el coeficiente i, este se debe redondear, pues la entrada del vector es un número entero comprendido en el rango [1, 100]. Para esto se puede utilizar una función de redondeo del lenguaje de programación, el mapping de ov vs $INCk_d$ se puede observar en la figura 29.

Tabla 15. Vector de mapping de ov vs $INCk_d$

$VINCk_d$ [0, 0.1]	0	0.010	0.010	0.012	0.014	…	0.081	0.083	0.085	0.087	0.089
i [1, 100]	1	2	3	4	5	…	96	97	98	99	100
ov [0, 1]	0	0.010	0.020	0.030	0.040	…	0.959	0.969	0.979	0.989	1

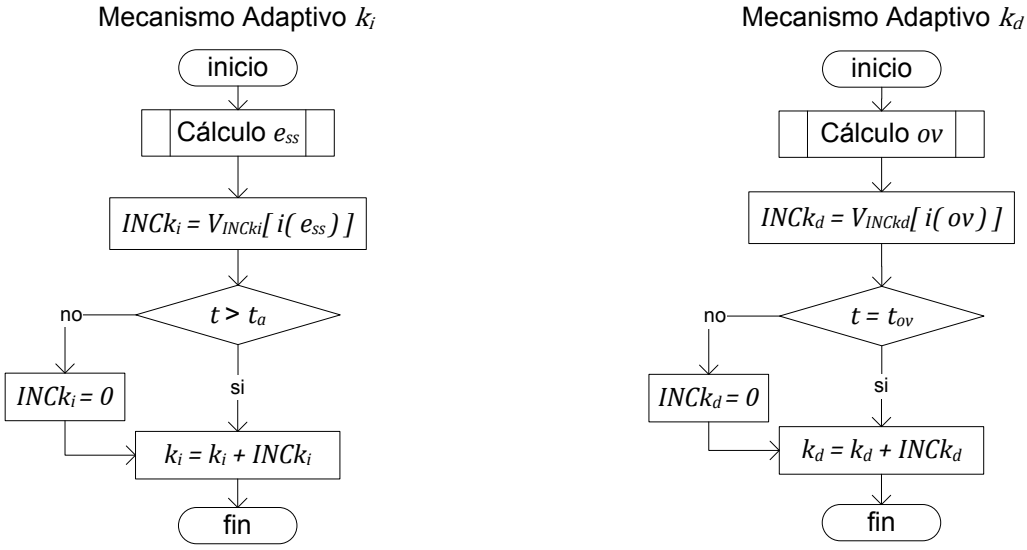
Por medio del sistema difuso se obtiene los incrementos para cada ganancia siguiendo la base de conocimiento de un experto en proceso, El MAD obtiene las ganancias utilizando la siguiente formulas:

$$k_i = k_i + INCk_i \tag{41}$$

$$k_d = k_d + INCk_d \tag{42}$$

Para resumir todas las operaciones que realizan los mecanismos adaptivos difusos de k_i y k_d se utilizan los siguientes diagramas de flujo

Figura 40. Diagramas de bloques de los mecanismo adaptivos de k_i y k_d

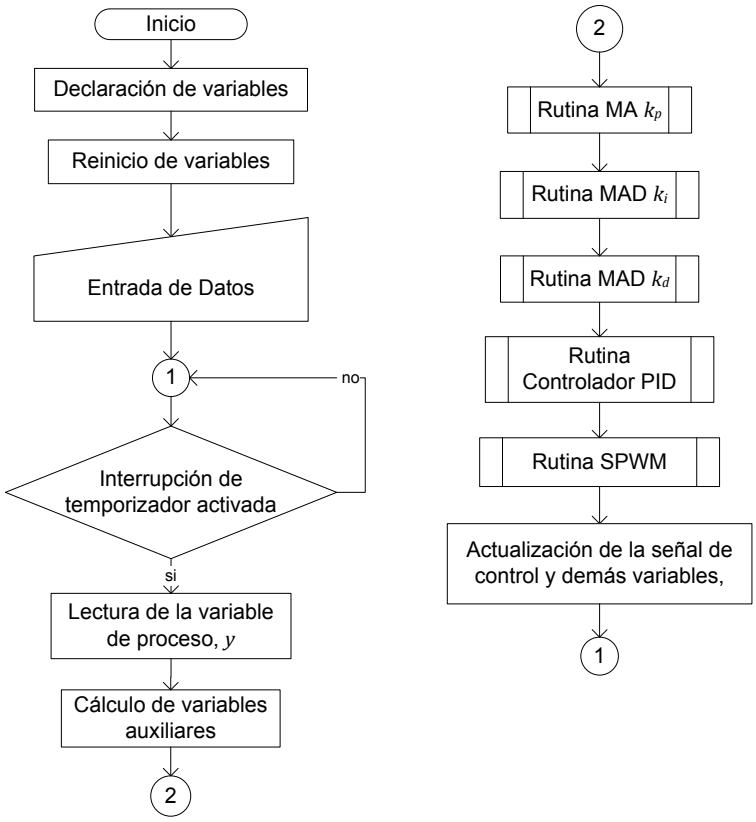


El cálculo de k_i se ajusta más de una vez cada transitorio, es conveniente para corregir de forma eficaz el error en estado estable, a diferencia del cálculo de k_d que se realiza una vez cada transitorio debido a la característica inherente del sobrepaso.

2.2.2.3 Funcionamiento del MAD1. El MAD1 se utiliza para sintonizar el controlador PID digital a una planta especifica, el controlador parte con las ganancias sin sintonizar de tal forma que el controlador no tenga gran efecto en el proceso, esto se refiere a que el controlador parte con las acciones proporcional, integral y derivativa casi nulas, en este caso el controlador es un lazo de realimentación con un controlador PID lento, estos parámetros iníciales se escogen con el fin de encargar la tarea de encontrar los incrementos de las ganancias más adecuados al sistema difuso partiendo desde el valor mínimo de un ganancia hasta el valor idóneo de esta misma.

Cada transitorio adquiere información de la planta, esta interactúa con el sistema difuso del MAD1 de manera que se obtiene un parámetro resultado, cuando el sistema cumple los requerimientos establecidos de diseño, que son eliminar el error en estado estable, mejorar el tiempo de subida y evitar en la mayor medida posible el sobrepaso, este detiene el incremento de parámetros, resultado en una sintonía exitosa.
Este tipo se sistemas son sensibles a la desestabilización debido a las perturbaciones de carga, no es conveniente utilizar el MAD1 durante la operación de la planta, el MAD1 se utiliza para escoger los parámetros adecuados cada vez que el proceso lo requiera, por ejemplo cuando se instale por primera vez un controlador o cuando la planta tenga una sintonía pobre, los parámetros se guardan en la memoria y el dispositivo permanece trabajando como un controlador PID estático o manual.

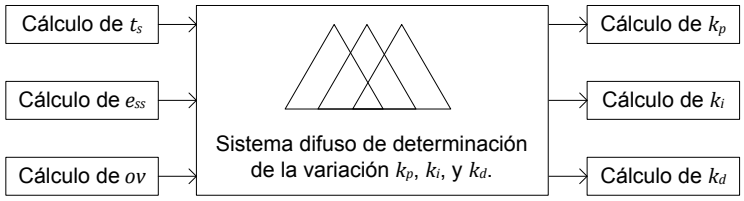
Figura 41. Diagrama de bloques del controlador adaptivo PID con el MAD1



En el diagrama anterior se puede observar de forma general las rutinas del controlador PID adaptivo con el MAD1, que está conformado por la rutina del controlador PID (figura 12) y las rutinas de los mecanismos adaptivos de k_p, k_i y k_d, que se ubican antes de la rutina del controlador, estas se ejecutan como se ha explicado anteriormente, este diagrama está listo para ser implementado en el sistema digital dsPIC.

2.2.3 Mecanismo adaptivo difuso 2 (MAD2). La principal diferencia con el MAD1 es que este varía las ganancias del controlador por medio de incrementos y decrementos logrando mayor estabilidad dándole la característica de controlador de proceso adaptivo on-line durante la operación, igual que el MAD1 está basado en el conocimiento experto de la tabla 7, pero se tiene en cuenta además de las relaciones dominantes entre ganancias del controlador y características del proceso otras relaciones que tienen cierto grado de significancia reflejándose en una regla base de mayor tamaño. El sistema difuso que conforma el MAD2 es del tipo MIMO (*Multiple input multiple output*) en el cual algunas salidas dependen de varias entradas.

Figura 42. Diagrama de bloques del mecanismo adaptivo difuso 2



Como sugiere el anterior grafico, el mecanismo adaptivo difuso 2 se divide en tres partes y una parte adicional donde se explica el funcionamiento del MAD2.

- Cálculo de las características t_s, e_{ss}, y ov
- Sistema difuso de determinación de la variación de k_p, k_i, y k_d
- Cálculo de las ganancias k_p, k_i, y k_d

Debido a la similitud de funcionamiento de los mecanismos MAD1 y MAD2, las características t_s, e_{ss}, y ov se calculan de igual forma en ambos.

2.2.3.1 Sistema difuso para determinar la variación de k_p, k_i, y k_d. Se encarga de encontrar la variación adecuada de las ganancias dependiendo de las entradas y la regla base, en este caso la variación de las ganancias es la salida del sistema, que puede ser un incremento o un decremento, la explicación del sistema se puede dividir en tres partes:

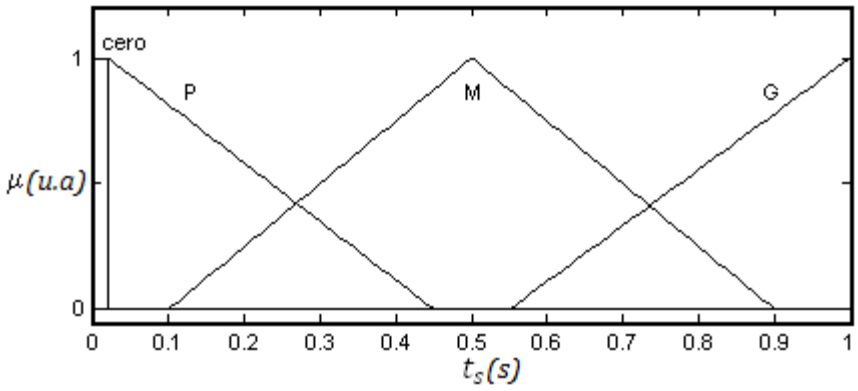
- Definición de los conjuntos difusos de las entradas y salidas
- Descripción de la regla base
- Procesamiento del sistema difuso

- **Definición de los conjuntos difusos de las entradas y salidas.** Las entradas del sistema son las características t_s, e_{ss}, y ov, las mismas del MAD1, se describen de una forma similar.

La variable de entrada tiempo de subida t_s se describe por los siguientes conjuntos difusos.

- t_s = {“Cero”, “Pequeño” (P), “Mediano” (M), “Grande” (G)}

Figura 43. Conjuntos difusos de la variable de entrada t_s



Maneja un rango de [0, 1] que se mide en segundos, esta característica se puede cuantificar de esta forma, pues este es el tiempo característico de una planta motor de baja potencia, el conjunto "Cero" indica que el tiempo está en el rango adecuado.

Tabla 16. Rango de los conjuntos difusos de la variable t_s

Cero	Muy pequeño	Pequeño	Mediano
[0, 0.02]	(0.02, 0.45]	[0.1, 0.9]	[0.55, 1]

El error en estado estable e_{ss}, se describe por los siguientes conjuntos difusos, el rango de esta característica esta normalizado, pero es más útil describirlo en el rango [0, 04], igual que en la característica anterior el conjunto "Cero" indica que e_{ss} está en el rango permitido.

- e_{ss} = {"Cero", "Pequeño" (P), "Mediano" (M), "Grande" (G)}

Figura 44. Conjuntos difusos de la variable de entrada e_{ss}

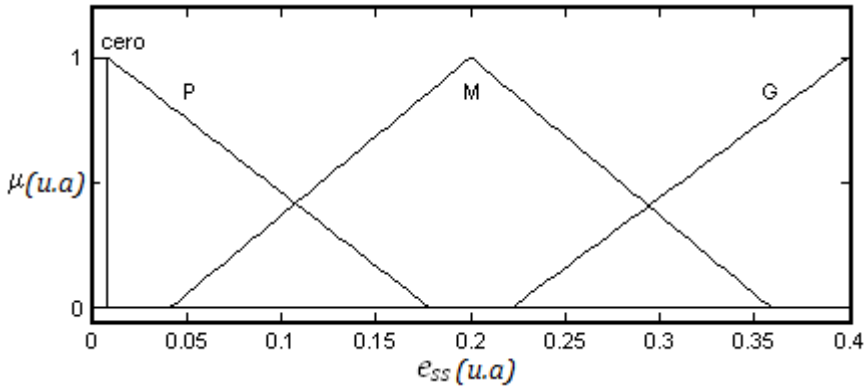


Tabla 17. Rango de los conjuntos difusos de la variable e_{ss}

Cero	Muy pequeño	Pequeño	Mediano
[0, 0.01]	(0.01, 0.18]	[0.04, 0.36]	[0.22, 0.4]

El sobrepaso máximo ov, se describe por los siguientes conjuntos difusos, el rango de esta característica esta normalizado [0, 1], igual que en la característica anterior el conjunto "Cero" indica que ov está en el rango permitido.

- ov = {"Cero", "Pequeño" (P), "Mediano" (M), "Grande" (G)}

Figura 45. Conjuntos difusos de la variable de entrada ov

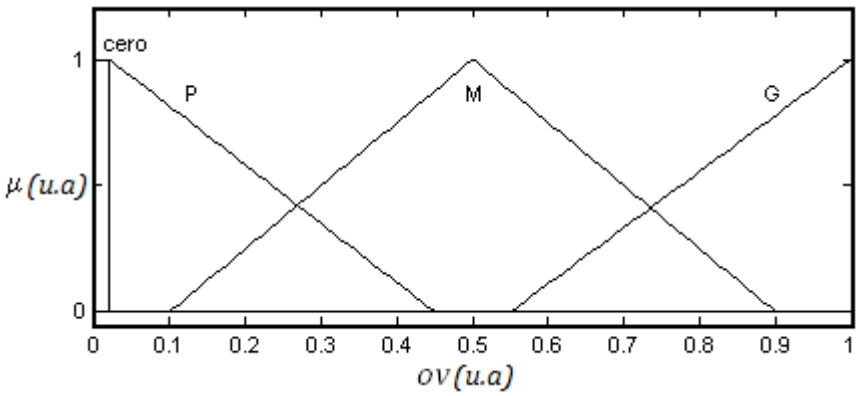


Tabla 18. Rango de los conjuntos difusos de la variable ov

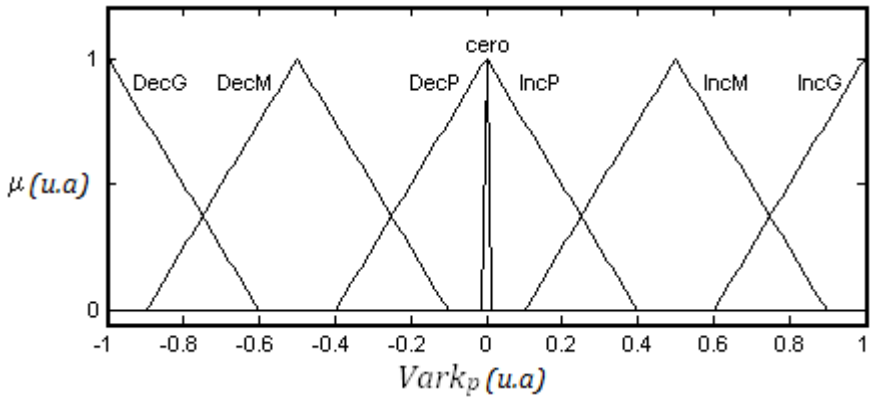
Cero	Muy pequeño	Pequeño	Mediano
[0, 0.01]	(0.01, 0.45]	[0.1, 0.9]	[0.55, 1]

Las variables de salida del sistema difuso indican un incremento o decremento de la ganancia en particular, en general una variación, igual que las variables de entrada son tres, una variable de salida para cada ganancia en particular.

La variable de salida variación de k_p ($Vark_p$) se puede describir por los siguientes conjuntos difusos:

- $Vark_p$ = {"Decremento Grande" (DecG), "Decremento Mediano" (DecM), "Decremento Pequeño" (DecP), "cero", "Incremento Pequeño" (IncP), "Incremento Mediano" (IncM), "Incremento Grande" (IncG)}

Figura 46. Conjuntos difusos de la variable de salida $Vark_p$



Maneja tres conjuntos graduales para incrementar y otros tres para decrementar, también cuenta con el conjunto "Cero" que indica una variación nula de $Vark_p$, la deducción de los rangos de estos conjuntos es experimental.

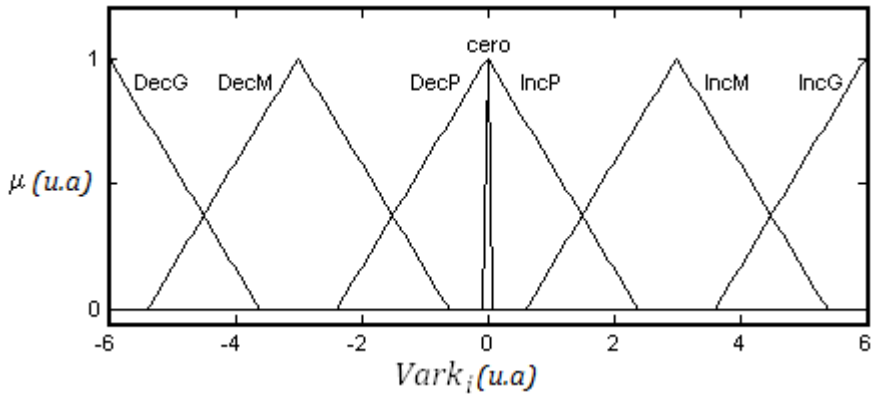
Tabla 19. Rango de los conjuntos difusos de la variable $Vark_p$

Decremento Grande	Decremento Mediano	Decremento Pequeño	Cero	Incremento Pequeño	Incremento Mediano	Incremento Grande
[-1, -0.6]	[-0.9, -0.1]	[-0.4, 0)	0	(0, 0.4]	[0.1, 0.9]	[0.6, 1]

La variable de salida variación de k_i ($Vark_i$) se puede describir por los siguientes conjuntos difusos distribuidos en un rango de [-6, 6], igual que la variable anterior cuenta con tres conjuntos para incrementar y tres conjuntos para decrementar, el conjunto "Cero" indica una variación nula de $Vark_i$, entiéndase esto como si la ganancia se ha adaptado satisfactoriamente.

- $Vark_i$ = {“Decremento Grande” (DecG), “Decremento Mediano” (DecM), “Decremento Pequeño” (DecP), “cero”, “Incremento Pequeño” (IncP), “Incremento Mediano” (IncM), “Incremento Grande” (IncG)}

Figura 47. Conjuntos difusos de la variable de salida $Vark_i$



La variable de salida variación de k_d ($Vark_d$) se puede describir por los siguientes conjuntos difusos distribuidos en un rango de [-0.05, 0.05], igual que la variable anterior cuenta con tres conjuntos para incrementar y tres conjuntos para decrementar, el conjunto “Cero” indica variación nula de $Vark_d$, entiéndase esto como si la ganancia se ha adaptado satisfactoriamente.

Tabla 20. Rango de los conjuntos difusos de la variable $Vark_i$

Decremento Grande	Decremento Mediano	Decremento Pequeño	Cero	Incremento Pequeño	Incremento Mediano	Incremento Grande
[-6, -3.6]	[-5.4, -0.6]	[-2.4, 0)	0	(0, 2.4]	[0.6, 5.4]	[3.6, 6]

Figura 48. Conjuntos difusos de la variable de salida $Vark_d$

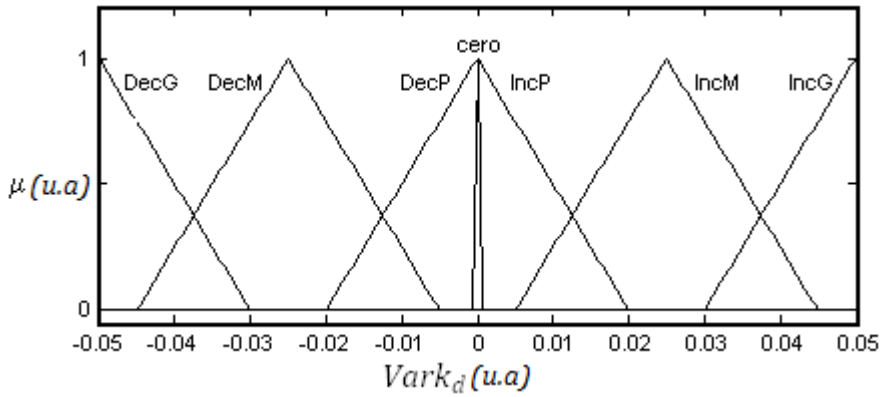


Tabla 21. Rango de los conjuntos difusos de la variable $Vark_d$

Decremento Grande	Decremento Mediano	Decremento Pequeño	Cero	Incremento Pequeño	Incremento Mediano	Incremento Grande
[-0.05, -0.03]	[-0.045, -0.005]	[-0.02, 0)	0	(0, 0.02]	[0.005, 0.045]	[0.03, 0.05]

- **Descripción de la regla base.** Es la redacción del conocimiento experto de la tabla 7, por medio de reglas "SI-ENTONCES", estas definen la variación de las ganancias en relación con las características del proceso, a diferencia con la regla base del MAD1 esta tiene en cuenta varios niveles de significancia en la relación entre cada ganancia y cada característica. La característica del sobrepaso ov se ve afectada en mayor dominancia por k_d y en menor medida por k_i y k_p, el error en estado estable e_{ss} se maneja de forma absoluta por k_i y en menor medida por k_p, el tiempo de subida se maneja con dominancia por k_p y casi en igual manera por k_d.

Tabla 22. Regla base para el sistema difuso del MAD2

Regla 1	SI ov es G ENTONCES k_p es decP and k_i es decP and k_d es incG
Regla 2	SI ov es M ENTONCES k_d es incM
Regla 3	SI ov es P ENTONCES k_d es incP
Regla 4	SI e_{ss} es G ENTONCES k_p es incM and k_i es incG
Regla 5	SI e_{ss} es M ENTONCES k_p es incP and k_i es incM
Regla 6	SI e_{ss} es P ENTONCES k_i es incP
Regla 7	SI t_s es G ENTONCES k_p es incG and k_d decP
Regla 8	SI t_s es M ENTONCES k_p es incM and k_d decP
Regla 9	SI t_s es P ENTONCES k_p es incP and k_d decP
Regla 10	SI ov es cero ENTONCES k_d es cero
Regla 11	SI e_{ss} es cero ENTONCES k_i es cero
Regla 12	SI t_s es cero ENTONCES k_p es cero and k_d es cero

Al variar las ganancias algunas características relacionadas se afectan positivamente mientras que otras se afectan de manera contraria, para definir la variación correcta de las ganancias dependiendo de las características se tienen en cuenta las relaciones secundarias de la tabla 7, determinando si se debe incrementar o decrementar las ganancias, el nivel de variación se define con ayuda del proceso experimental, cuando el sobrepaso es grande, se corrige con un incremento grande de k_d, de igual forma se define para los niveles mediano y pequeño, aunque un sobrepaso grande también se ocasiona debido a valores altos en las ganancias k_p y k_i, intuitivamente se determina que al obtener un sobrepaso grande se decremente k_p y k_i, de forma experimental se deduce que el nivel de decremento sea pequeño.

El error en estado estable en sus tres niveles grande, mediano, y pequeño se corrige con el incremento correspondiente de k_i, ya que es la ganancia dominante, no obstante el error en estado estable disminuye con el incremento de la ganancia k_p, y se relacionan como indica la tabla anterior.

El tiempo de subida en sus tres niveles se maneja con incrementos correspondientes de la ganancia k_p, haciendo que el tiempo de subida disminuya, pero la ganancia k_d en altos valores puede ocasionar un gran amortiguamiento aumentando considerablemente t_s, debido a esto, se decrementa k_d cuando el tiempo de subida no es adecuado.

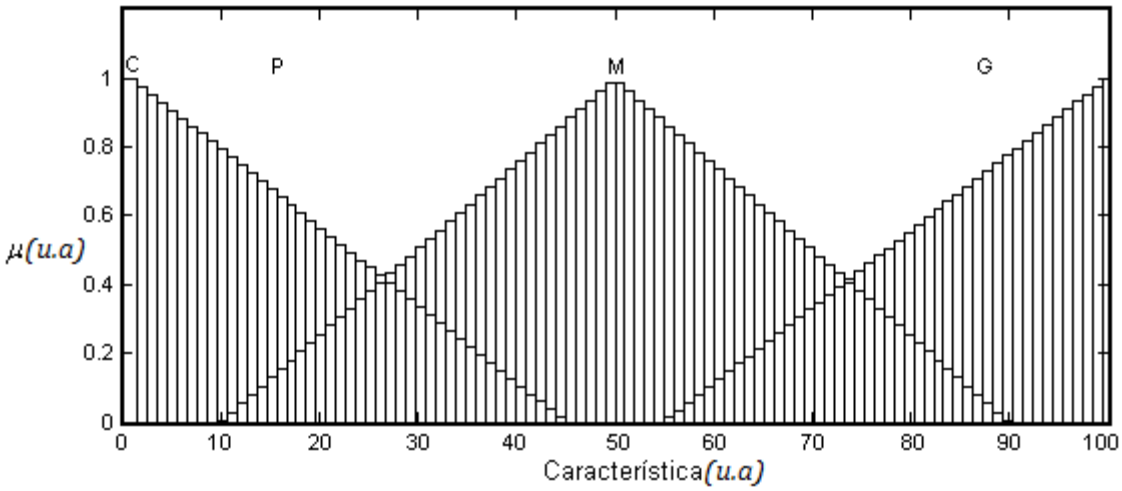
La anterior descripción es una explicación de la regla base que se observa en la tabla anterior en donde se explica las relaciones entre las ganancias y características del mecanismo adaptivo propuesto.

- **Procesamiento del sistema difuso del MAD2.** Debido a que el sistema difuso es MIMO no es viable implementarlo por el método de mapping ya que consume altos

recursos de memoria. Este sistema difuso con varias entradas y salidas se ha implementado procesando de forma numérica el método de Mamdani explicado anteriormente, logrando un balance entre recursos de memoria y velocidad de procesamiento, de esta forma la implementación no se recarga completamente sobre los recursos de memoria del sistema.

El tiempo en el cual se debe resolver el sistema difuso es menor al tiempo de muestreo ($h = 0.002s$), dependiendo de este parámetro se escoge la resolución de los conjuntos difusos de entrada y salida, entonces, se define 50 puntos para las variables de salida y 100 puntos para las variables de entrada, como indica la siguiente grafica:

Figura 49. Discretización de los conjuntos difusos para las variables de entrada



- **Fusificación de las variables de entrada.** Los conjuntos difusos de entrada cero (C), pequeño (P), mediano (M) y, grande (G) se utilizan para fusificar cada variable de entrada obteniendo la implicación del predicado de cada regla, los conjuntos se implementan en vectores independientes debido al traslapamiento de estos, el índice del vector corresponde a la variable independiente y el valor destino donde apunta el índice es la variable dependiente, en la cual se almacena el valor de la función de pertenencia. En la grafica se muestra el rango del índice [0, 99] del total de los conjuntos difusos, para el ahorro de memoria se fusifica todas las variables de entrada utilizando los mismos conjuntos, simplemente se modifica cada rango de entrada para que coincida con el rango del índice de los conjuntos difusos.

Tabla 23. Correspondencia de los rangos de las variables de entrada con los rangos de los índices de los vectores que representan los conjuntos difusos

Rango de la variable de entrada	Ecuación de correspondencia	Rango del índice del vector
$t_s[0,1]$	$i(t_s) = 99 \times t_s$ (43)	$i(t_s)[0,99]$
$e_{ss}[0,0.4]$	$i(e_{ss}) = 247.5 \times e_{ss}$ (44)	$i(e_{ss})[0,99]$
$ov[0,1]$	$i(ov) = 99 \times ov$ (45)	$i(ov)[0,99]$

La implementación por vector de los conjuntos difusos para fusificar las variables de entrada es una manera precisa y rápida con la cual se determina la implicación de cada regla, esta consiste en una multiplicación y obtener el valor de la implicación apuntando con el índice al vector que indique la regla. Debido a que los conjuntos difusos toman valores solamente a lo largo de un tramo del rango total, se pueden definir dentro de un rango efectivo para evitar el uso de memoria, al acortar su rango se puede confundir el inicio en el rango real [0, 99] para esto se declaran con un nuevo rango y un offset para definirlos dentro del rango total, como indica la siguiente tabla.

Tabla 24. Rangos de los conjuntos difusos para las variables de entrada

Conjunto	Rango efectivo	Nuevo rango	Offset
Cero (C)	[0]	[0]	0
Pequeño (P)	[1, 45]	[0, 44]	1
Mediano (M)	[10, 89]	[0, 79]	10
Grande (G)	[55, 99]	[0, 44]	55

Tabla 25. Definición de los vectores que representan los conjuntos difusos de entrada

i	0	1	2	3	4	…	75	76	77	78	79
M[0, 79]	0.01	0.03	0.05	0.07	0.1	…	0.1	0.07	0.05	0.03	0.01
i	0	1	2	3	4	…	40	41	42	43	44
P[0,44]	0.99	0.97	0.95	0.93	0.91	…	0.1	0.08	0.05	0.03	0.01
G[0, 44]	0.01	0.03	0.05	0.07	0.1	…	0.91	0.93	0.95	0.97	1

Los conjuntos difusos de salida conforman el conjunto resultado aplicando el método de la implicación según indique el predicado y fusificación de la regla; para evitar consumo de memoria se describen todas la variables de salida utilizando los mismos conjuntos difusos, igual que en los conjuntos de entrada también se debe redimensionar el valor de salida por medio de una ecuación de correspondencia, se tiene en cuenta el rango total que recorren los conjuntos y el rango de cada una de las variables de salida como indica la siguiente tabla.

Figura 50. Discretización de los conjuntos difusos para las variables de salida

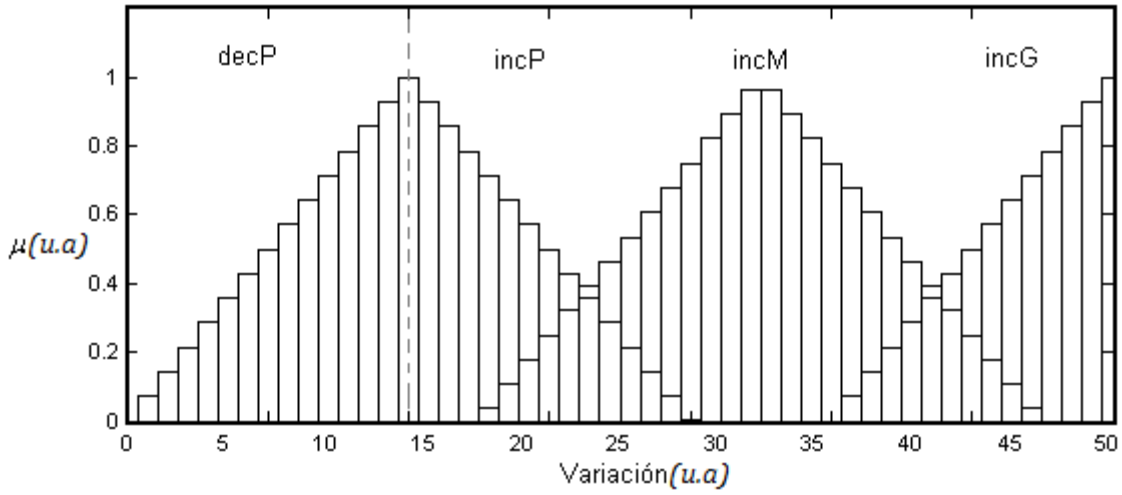


Tabla 26. Correspondencia del rango de los índices de los vectores con los rangos de las variables de salida

Rango del índice del vector	**Ecuación de correspondencia**	**Rango de la variable de salida**
$i\_VARk_p[0,50]$	$Vark_p = i\_VARk_p \times 0.0285 - 0.428$ (46)	$VARk_p[-0.4, 1]$
$i\_VARk_i[0,50]$	$Vark_i = i\_VARk_i \times 0.171 - 2.57$ (47)	$VARk_i[-2.4, 6]$
$i\_VARk_d[0,50]$	$Vark_d = i\_VARk_d \times 0.00142 - 0.0214$ (48)	$VARk_d[-0.02, 0.05]$

Igual que los conjuntos difusos de entrada, los conjuntos de salida también se definen con un nuevo rango y un offset, como indica la siguiente tabla.

Tabla 27. Rangos de los conjuntos difusos para las variables de salida

Conjunto	**Rango Efectivo**	**Nuevo Rango**	**Offset**
Decremento pequeño (decP)	[1, 14]	[0, 13]	0
Incremento pequeño (incP)	[15, 28]	[0, 13]	15
Incremento mediano (incM)	[18, 45]	[0, 27]	18
Incremento grande (incG)	[36, 49]	[0, 13]	36

Como se puede observar en la figura 50 y en la tabla anterior, el conjunto y su vector correspondiente, “decP” y “incG”, son iguales en forma, tienen el mismo número de elementos siendo equivalentes, con un offset distinto, se puede utilizar “decP” como “incG” evaluando con el offset correspondiente, de esta forma solo se hace necesario declarar uno de los dos, en este caso “decP”.

Tabla 28. Definición de los vectores que representan los conjuntos difusos de salida

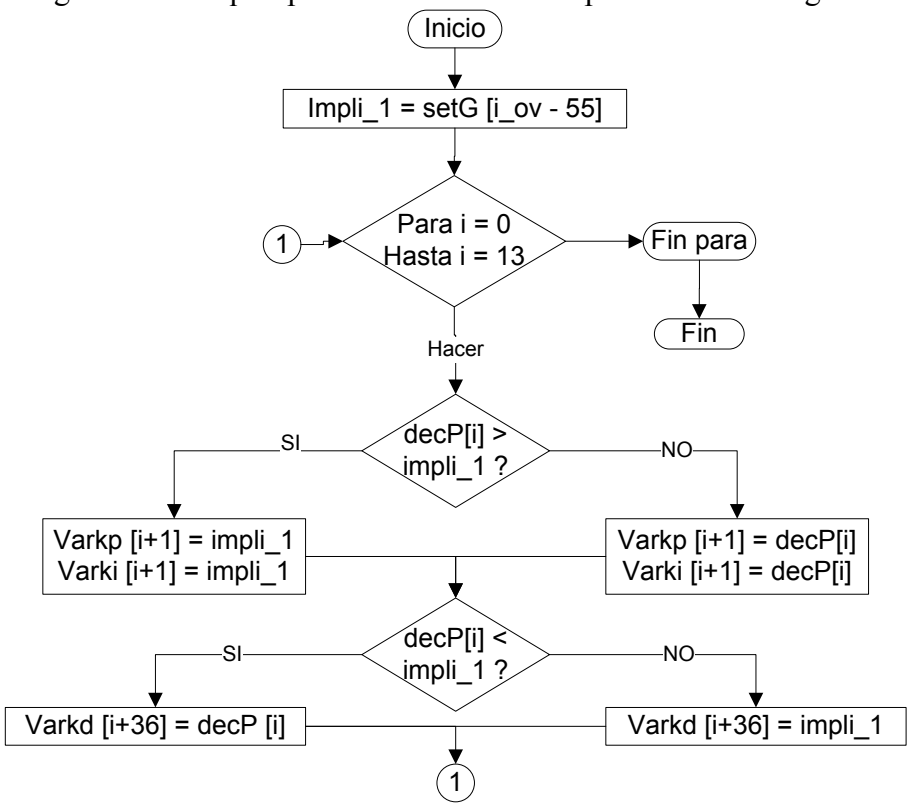
i	0	1	2	3	4	…	23	24	25	26	27
incM[0, 27]	0.03	0.10	0.17	0.25	032	…	0.32	0.25	0.17	0.10	0.03
i	0	1	2	3	4	…	9	10	11	12	13
decP[0, 13], incG[0, 13]	0.07	0.14	0.21	0.28	0.35	…	0.71	0.78	0.85	0.92	1.00
incP[0, 13]	0.92	0.85	0.78	0.71	0.64	…	0.35	0.28	0.21	0.14	0.07

- **Aplicación del método de la implicación y agregación.** Una vez se obtiene la fusificación del predicado de cada regla aplicando las formula anteriormente descritas y direccionando el vector o los vectores correspondientes, se puede aplicar el método de la implicación a cada regla y también ejecutar la agregación, de la siguiente forma, se toma como ejemplo la regla 1:

SI "ov es G" ENTONCES "k_p es decP" and "k_i es decP" and "k_d es incG", se supone que ov toma un valor igual a 0.8, primero se fusifica la entrada, $ov = 0.8$, se reemplaza en la ecuación 45 y se obtiene el valor del índice, $i(ov) = 99 \times 0.8$, se hace la operación y se redondea a un numero entero, $i(ov) = 79$, el índice obtenido se evalúa en el vector que representa el conjunto difuso grande (G) como indica la regla, para fusificar el valor obteniendo el grado de verdad del predicado. El vector del conjunto G, se define, con rango[0, 44] con offset de 55, para evaluar correctamente el vector, el índice se resta con el offset antes de ser evaluado, así, $i(ov) = 79 - 55 = 24$, evaluando G(24) = 0.50, siendo esta la fusificación de la entrada y como no hay ningún operador, también el grado de verdad del predicado de la regla 1 (implicación_1 = 0.50).

Para aplicar el método de la implicación se debe truncar los conjuntos "decP" de $VARk_p$, "decP" de $VARk_i$, y incG de $VARk_d$ como indica la regla 1 según implicación_1, esta operación se realiza con ayuda de un ciclo, como indica la siguiente figura, el código de esta rutina se puede observar en el literal (d) del ANEXO I.

Figura 51. Diagrama de bloques para el cálculo de la implicación de la regla 1



El anterior diagrama se encarga de conformar los conjuntos $VARk_p$, $VARk_d$, y $VARk_p$ según indique la implicación de la regla sobre los conjuntos "decP" e "incG", se tiene en cuenta la alineación de los vectores según el coeficiente, el traslapamiento de los conjuntos entre si y la truncación de los conjuntos según la regla. Cada regla tiene su algoritmo para calcular la implicación, a medida que se procesa cada regla, también se hace el proceso de agregación modificando los conjuntos $VARk_p$, $VARk_i$, y $VARk_d$.

- **Método de defusificación.** El método empleado es el mismo que se explico en (…ver numeral 2.2.1.6…) la entrada para aplicar el método de defusificación COG es un conjunto resultado, en este caso son tres, $VARk_p$, $VARk_i$, y $VARk_d$, se defusifican implementando la ecuación 35 como indica el siguiente diagrama. El código de esta rutina se puede observar en el literal (d) del ANEXO I.

Figura 52. Diagrama de bloques del método de defusificación

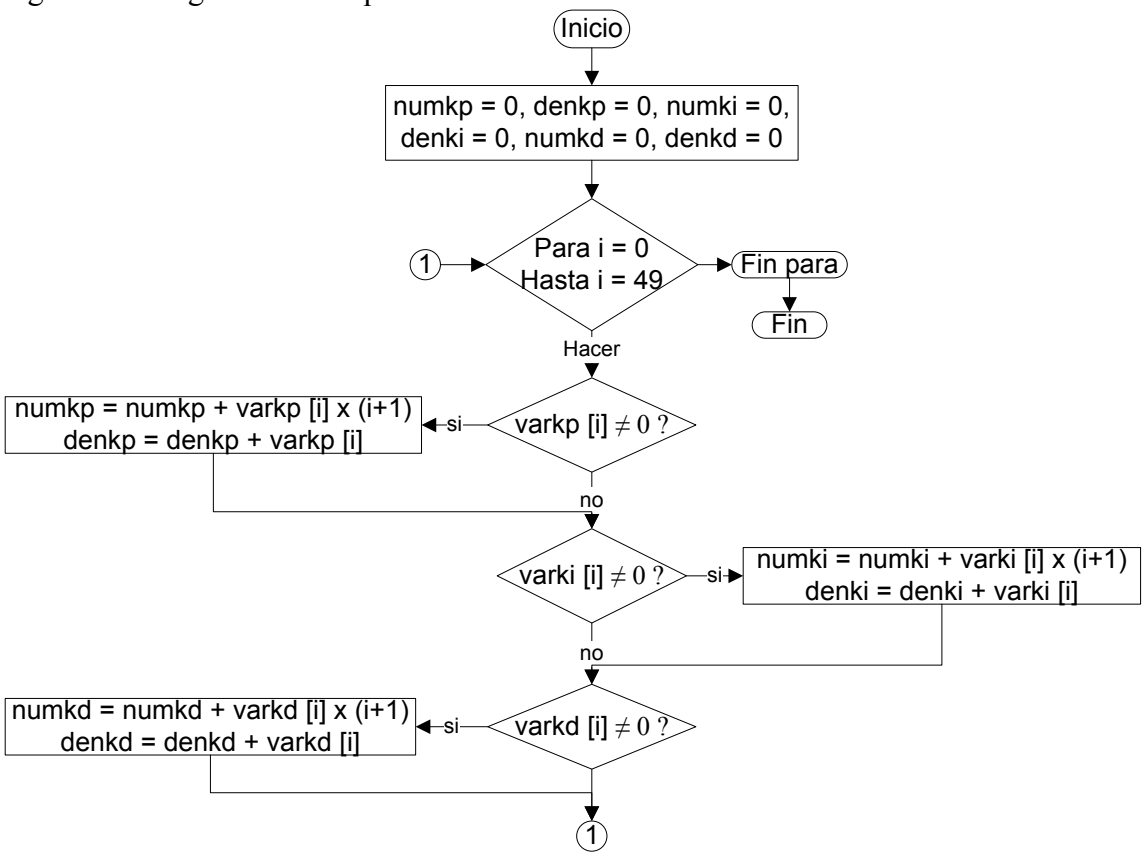
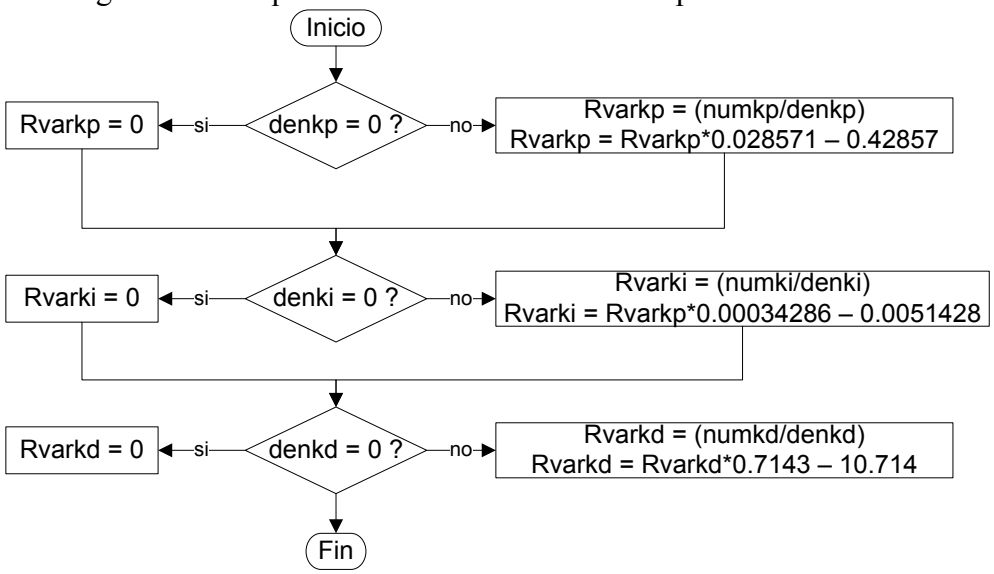


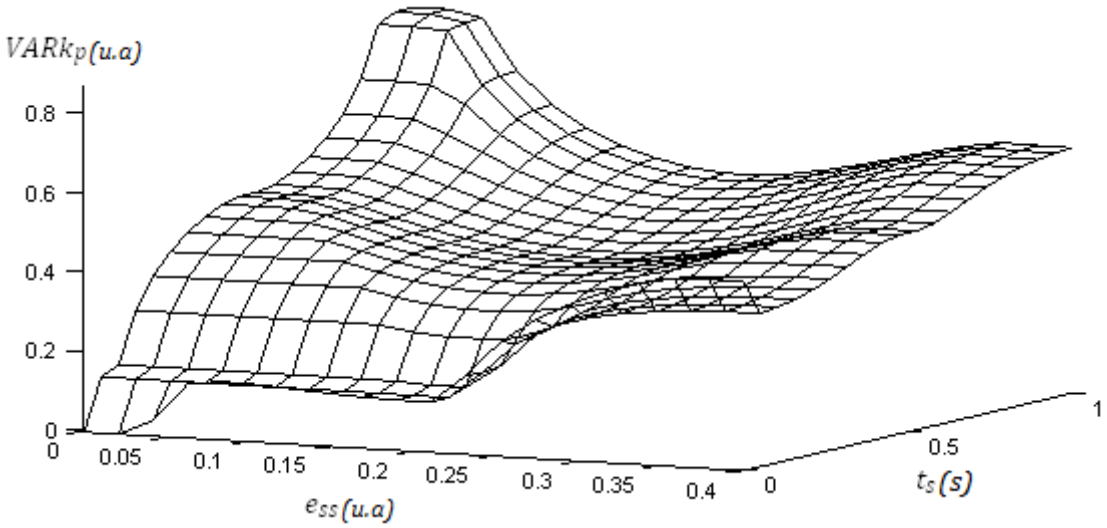
Figura 53. Diagrama de bloques de las ecuaciones de correspondencia



-

- **Superficies del sistema difuso de MAD2.** Con ayuda de Matlab se grafica las superficies de correspondencia entre las entradas y salidas del sistema difuso. La variable de salida $VARk_p$ depende las entradas t_s, ov y, e_{ss}, graficar esta función no es posible ya que es un función de tres variables, $VARk_p = f(t_s, ov, e_{ss})$, se grafica en pares para obtener una graficas en función de dos variables.

Figura 54. Superficie de correspondencia entre la salida $VARk_p$ y las entradas e_{ss} y t_s



Para graficar la superficie se uso una resolución de 400 puntos, y se resolvió el sistema con ayuda del la herramienta fuzzy de Matlab. Estas graficas reflejan el comportamiento de las salidas según indica la regla base, estas representan la variación de cada ganancia en función de las características de entrada, estas asemejan el razonamiento de un operador experto al manejar un controlador adaptivo de forma manual.

Figura 55. Superficie de correspondencia entre la salida $VARk_p$ y las entradas e_{ss} y ov

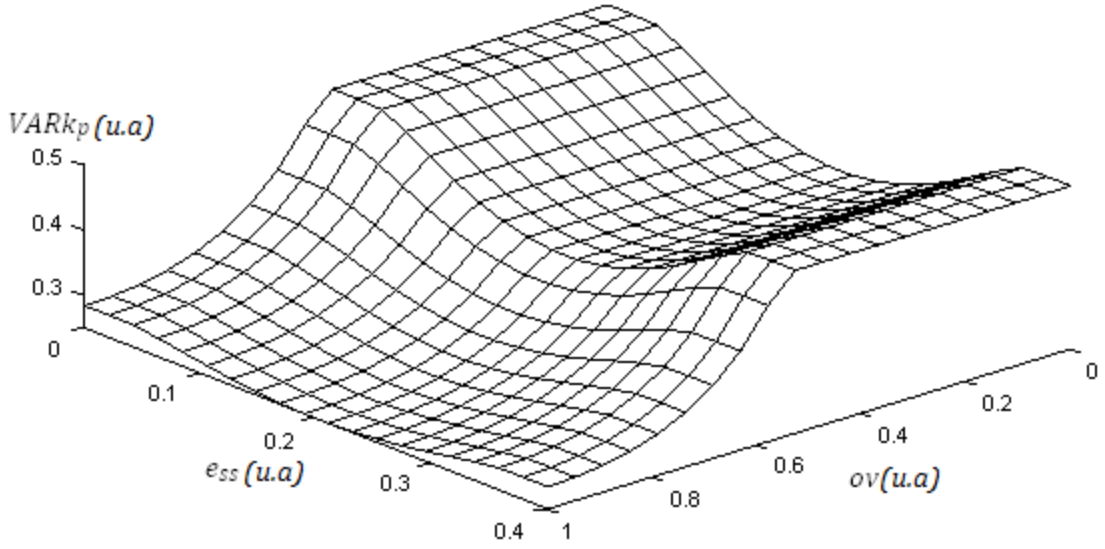
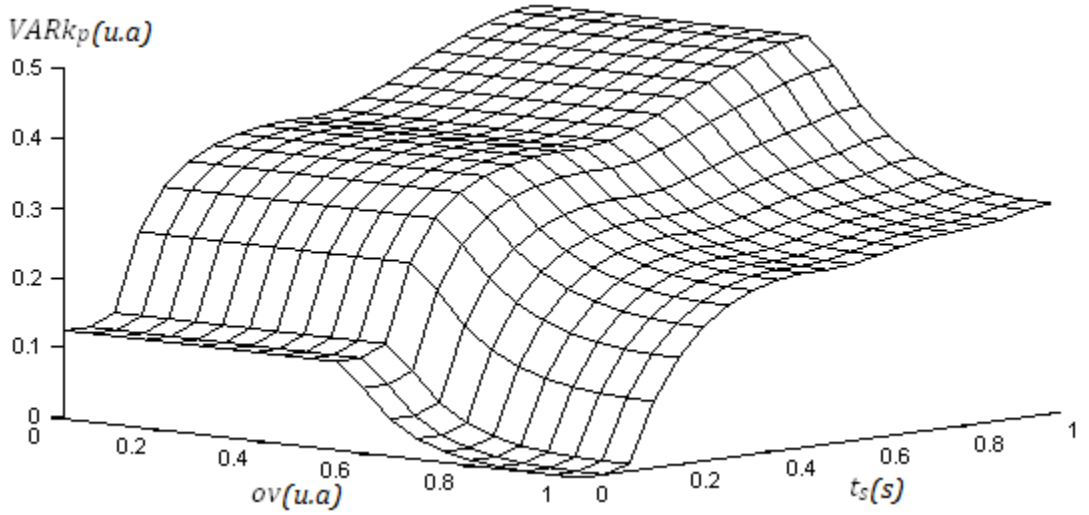


Figura 56. Superficie de correspondencia entre la salida $VARkp$ y las entradas ov y t_s



Cada salida $VARk_p$, $VARk_i$ y, $VARk_d$ tienen dependencia de las tres características, sin embargo solo se graficaron las más representativas, donde se muestran variaciones evidentes, como característica general en las graficas de las superficies se puede observar que la variación es alta cuando alguna ganancia toma un valor alto, y viceversa para la variable del eje perpendicular corroborando la regla base.

Figura 57. Superficie de correspondencia entre la salida $VARk_i$ y las entradas ov y e_{ss}

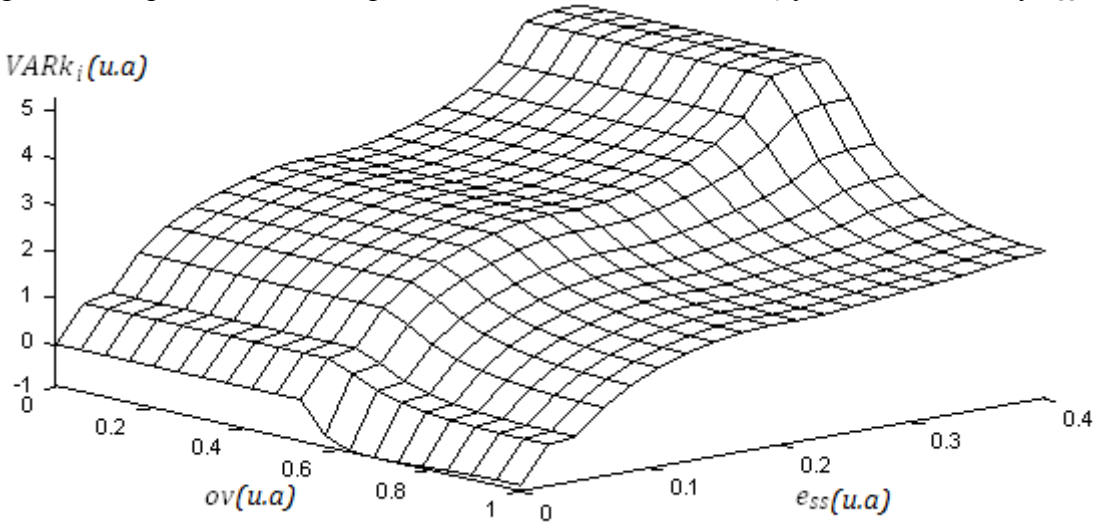
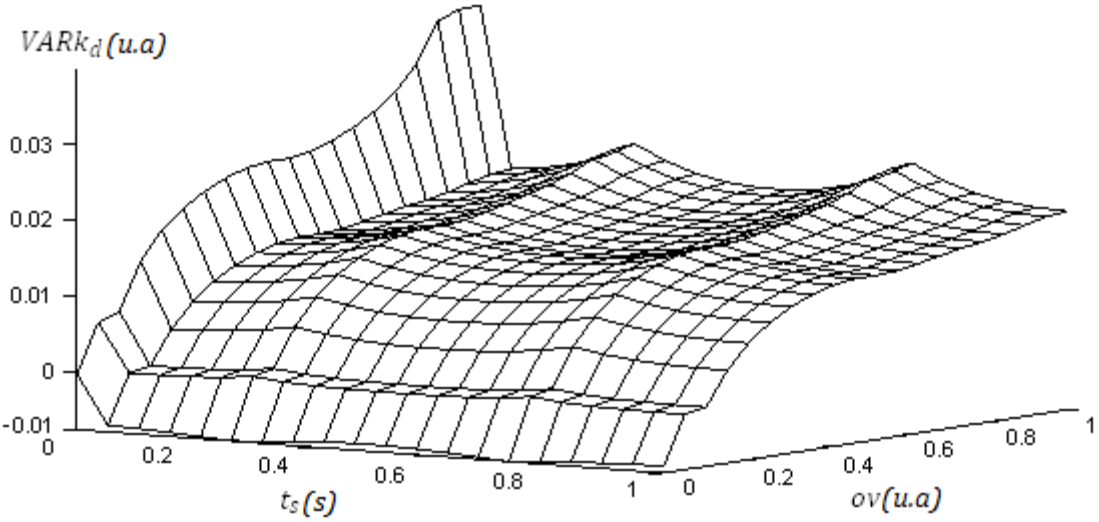


Figura 58. Superficie de correspondencia entre la salida $VARk_d$ y las entradas t_s y ov



2.2.3.2 Cálculo de las ganancias k_p, k_i, y k_d. Determinando la variación de cada ganancia se puede calcular la ganancia, que es la labor del MAD2, es importante sincronizar el tiempo de actualización de las ganancias, el sistema difuso determina la variación de las ganancias para cada transitorio pero para esto necesita que todas las variables de entrada se hayan calculado antes, entonces los eventos que ocurren en un transitorio citados en orden son: el tiempo de subida, el sobrepaso y el error en estado estable, el ultimo evento que ocurre es el estado estable o cuando el tiempo de asentamiento ha ocurrido, en conclusión la determinación de la variación y el cálculo de las ganancias se realiza cuando el sistema

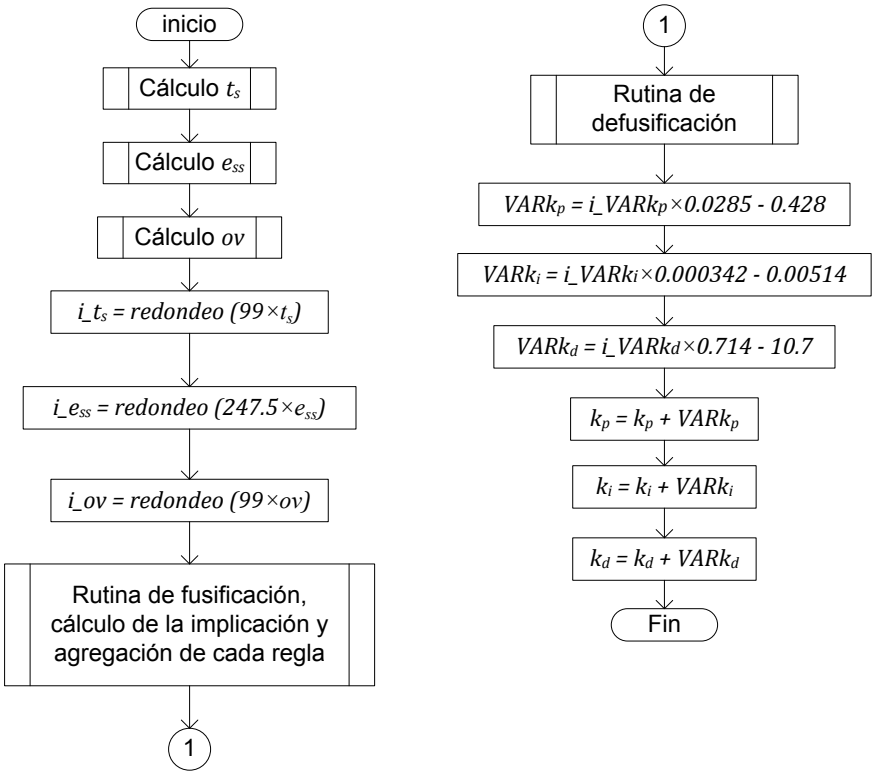
está en estado estable, con mayor exactitud cuando el reloj es igual al tiempo de asentamiento, para el cálculo se utilizan las siguientes formulas:

$$k_p = k_p + VARk_p \tag{49}$$

$$k_i = k_i + VARk_i \tag{50}$$

$$k_d = k_d + VARk_d \tag{51}$$

Figura 59. Diagrama de bloques del MAD2



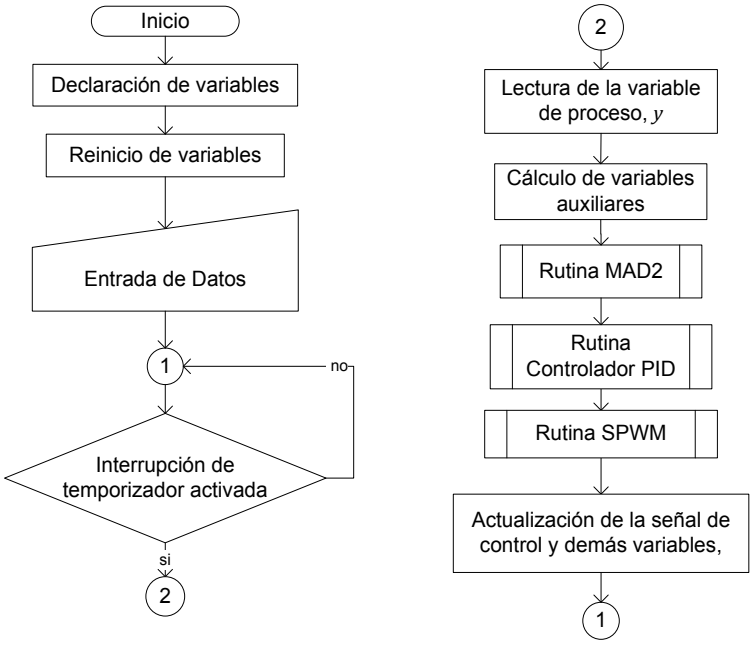
El anterior diagrama de bloques resume las operaciones que ejecuta el MAD2, esta rutina se debe sincronizar con el reloj de proceso y el tiempo se asentamiento.

2.2.3.3 Funcionamiento del MAD2. Funciona de forma similar que el MAD1, puede cumplir las mismas funciones del MAD1 pero tiene unas diferencias que le permiten funcionar on-line con el proceso en operación, la función de este es analizar cada transitorio y dar un resultado variando las ganancias, cuando la respuesta del proceso está dentro de los requerimientos las variaciones son nulas, aunque cuando ocurre un evento de carga o

una situación que altere el proceso, como un efecto no lineal el controlador adaptivo balancea sus parámetros seguir el punto de referencia.

El siguiente diagrama de bloques integra las rutinas del controlador PID y el mecanismo adaptivo difuso 2 que conforman el controlador adaptivo, se puede observar de forma general las operaciones o subrutinas que debe ejecutar el sistema digital al implementar el algoritmo, la ubicación de la subrutina MAD2 es la misma en la que se ubico el MAD1, su operación general es muy similar aunque como se observo son mecanismos adaptivos muy diferentes. Este diagrama de bloques describe en forma completa el controlador adaptivo con el MAD2 (PIDAD2) el cual está listo para ser implementado en el sistema digital dsPIC.

Figura 60. Diagrama de bloques del controlador adaptivo PID con el MAD2



2.3 IMPLEMENTACIÓN DEL CONTROLADOR DE PROCESO

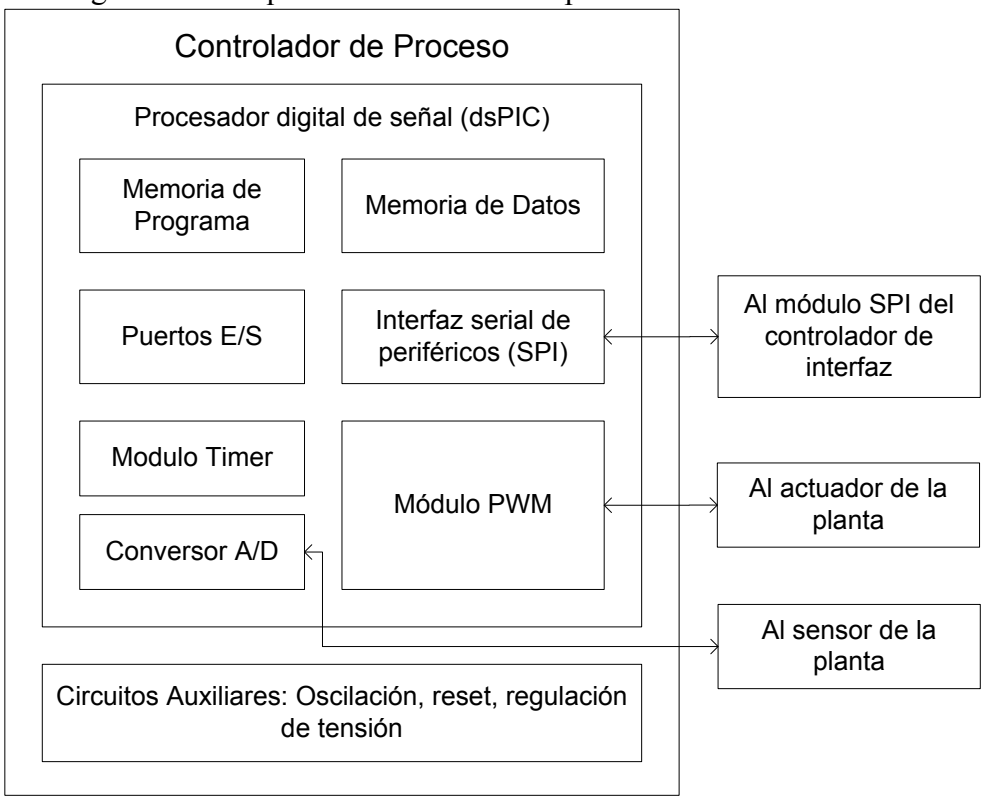
El controlador de proceso se implementa en el controlador digital de señal (DSC), un controlador digital de señal es un chip, controlador embebido que de manera uniforme integra los atributos de control de un microcontrolador (MCU) con la capacidad de computacion y manejo de informacion de un procesador de señal digital (DSP) en un solo nucleo.
Microchip ofrece cada cosa que se deberia esperar de un potente 16-bit MCU: rapido, sofisticado y con manejo de interrupcion flexible, un gran arreglo de funciones digitales y analogicas, manejo de potencia, opciones de reloj flexibles, reinicio, seguridad de codigo,

emulacion en tiempo real, etc. Ademas cuenta con una alta capacidad de memoria de programa y datos además de tener los periféricos necesarios para comunicación y control. Su programación se puede hacer en lenguaje a bajo nivel (ensamblador) y lenguaje C30 que está basado en el tradicional lenguaje C. La explicación de la interfaz de puede abordar en dos partes:

- Hardware del controlador de proceso

- Programación del firmware del controlador de proceso

Figura 61. Diagrama de bloques del controlador de proceso



2.3.1 Hardware del controlador de proceso. Está basado en el controlador dsPIC30F4012, se escogió principalmente porque pertenece a una distribución de DSC de Microchip Inc. destinada a la implementación de controladores de proceso industriales, este controlador es de alto desempeño con arquitectura Harvard modificada, tiene un set de 83 instrucciones optimizado para el compilador C, es de tecnología CMOS de bajo consumo con rango de voltaje de operación de 2.5 – 5V, velocidad de operación de 40Mhz y hasta 120Mhz para entrada de PLL de 4x, 8x, 16x, con la opción de trabajar cerca de los 30MIPS (Millón de instrucciones por segundo).

La memoria de programa tiene un espacio de 16Kx24 bits de palabras de instrucciones, la memoria de datos tiene un tamaño de 2Kx16 bits, en la cual se ubican los registros de funciones especiales, como el de estatus que contiene los bits indicadores del resultado de las operaciones aritméticas, también se pueden destacar el grupo de registros de trabajo de 16 bits que pueden ejecutar labores de direccionamiento indirecto entre ellos permitiendo ejecutar instrucciones DSP que requieran varios operandos.

Cuenta con 5 puertos, el puerto B con 6 pines los cuales se pueden configurar como entradas o salidas (E/S) digitales o como entradas analógicas para el módulo A/D, el puerto C con 3 pines los cuales se pueden configurar como E/S digital, manejan el módulo de interrupción por cambio de estado, el puerto D con 2 pines configurables como E/S digital, el puerto E con 7 pines configurables como E/S digital, maneja el módulo PWM y un pin es parte del módulo SPI y el puerto F con 2 pines configurables como E/S digital, maneja el módulo SPI.

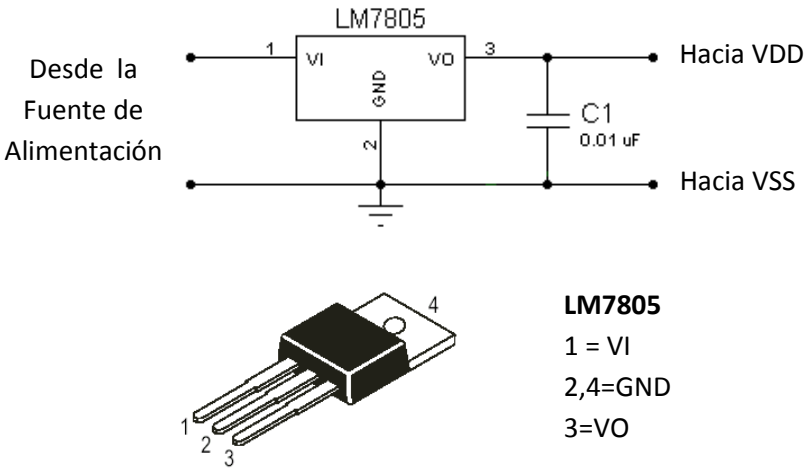
Tiene un módulo temporizador de 16 bits el cual puede generar interrupción en un periodo preprogramado, en este caso se utiliza para generar el tiempo de muestreo. También cuenta con un módulo de interfaz serial de periféricos (SPI) de 8 bits, útil para la comunicación de datos con entre la interfaz de usuario y el controlador de proceso, para establecer comunicación son necesarios 3 pines, un reloj serial SCK, una entrada de datos SDI y una salida de datos SDO.

El módulo de notificación de cambio provee la habilidad de generar una interrupción cuando se detecta un cambio de nivel en uno de lo pines del puerto, En este caso se habilita la interrupción al pin RC13 del DSC la cual se utiliza para manejar la rutina de STOP del proceso la cual se explicara más adelante.

El módulo de conversión A/D, cuenta con conversor A/D de 10 bits de aproximaciones sucesivas, con un error de cuantificación de ± ½ bit, un error de offset de ± 1 bit, un error de ganancia de ± 5 bits. El módulo A/D se utiliza en la emulación para desarrollar el controlador adaptivo, este se configura con dos entradas, la entrada AN0 para la salida de la planta y la entrada AN3 para el setpoint análogo, el voltaje de referencia positivo es el voltaje de alimentación (VCC) y negativo la tierra (GND), los tiempos de conversión que se escogen dan como máximo un tiempo de muestreo y conversión de 5uS, en la aplicación con el motor se utiliza la misma configuración con una sola entrada AN0.

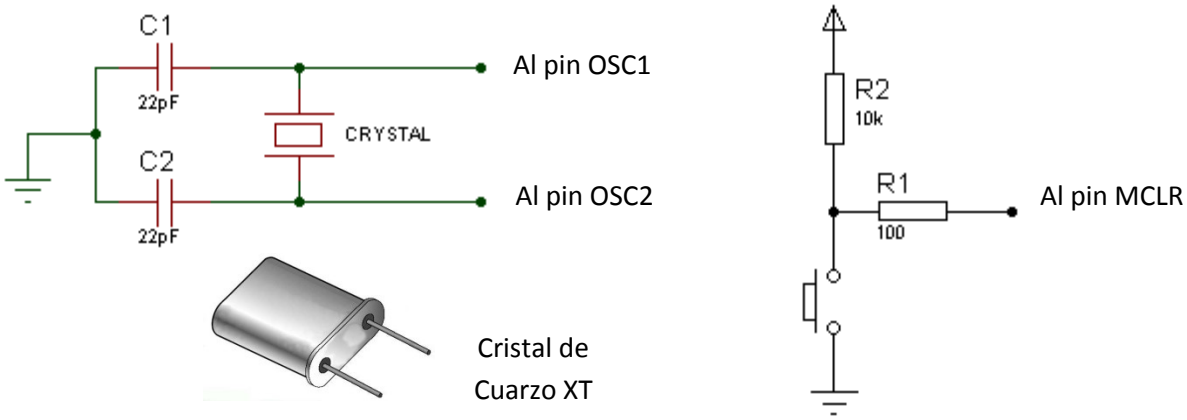
2.3.1.1 Circuito de alimentación. El DSC tiene como voltaje de operación un rango de 2.0V-5.5V DC, que se aplica en los pines VDD y VSS que son alimentación y tierra. Este voltaje se obtiene de la fuente de alimentación y se regula a la entrada de la tarjeta del controlador de interfaz por medio del regulador de tensión LM7805, que es un regulador de tres terminales positivo en encapsulado T0220, tiene un limitador de corriente interno, un interruptor de apagado por temperatura que permite operar en un rango seguro, puede entregar 1A de corriente a la salida.

Figura 62. Circuito auxiliar de alimentación con regulador LM7805



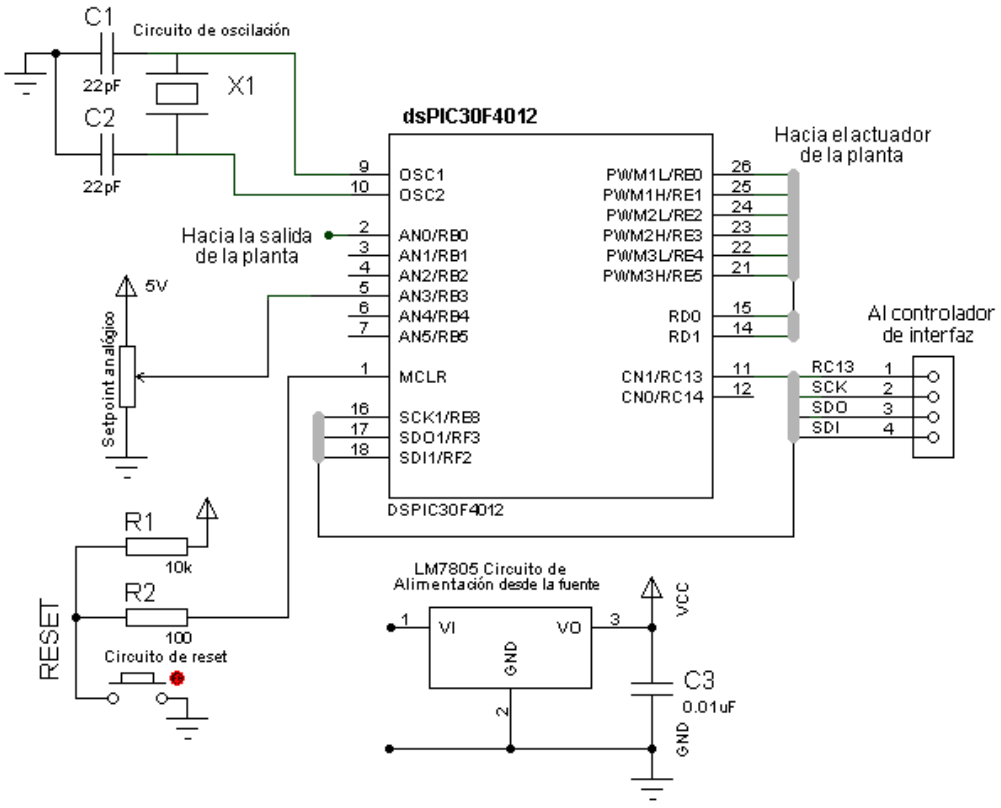
2.3.1.2 Circuito de oscilación y reinicio. El DSC requiere un circuito que le indique la velocidad de trabajo o frecuencia de oscilación (Fosc), que genere una onda cuadrada para sincronizar las operaciones del sistema, es simple pero vital para el funcionamiento. Este microcontrolador permite varios tipos de osciladores, en este caso se utiliza el oscilador de cristal de cuarzo HT con Fosc de 10Mhz, el dsPIC tiene un multiplicador PLL que permite aumentar la velocidad de trabajo, en este caso se aumenta 8X, en este caso opera igual que con un cristal de 80Mhz. La conexión se hace como indica la siguiente figura, los condensadores se escogen de acuerdo a la recomendación de la hoja de datos del microcontrolador, en este caso c1=c2=22pF. El ciclo instrucción es lo que tarda en una instrucción estándar en ser ejecutada, en este caso cada ciclo de instrucción tarda 4 ciclos del oscilador, Fosc/4, en este caso 1 ciclo de instrucción tarda 50nS.
El circuito de reinicio el dispositivo carga el contador de programa con el valor 00H, en este estado todos los elementos internos del DSC toman un valor conocido, el pin de reinicio es el pin 1 se denomina MCLR (*Máster Clear*) y produce este estado cuando se aplica un nivel bajo.

Figura 63. Circuito de oscilación y de reinicio



En la siguiente figura se puede visualizar el hardware controlador de proceso, donde se integran los circuitos anteriormente mencionados.

Figura 64. Circuito del controlador de proceso



2.3.2 Programación del firmware del controlador de proceso. El firmware que complementa el hardware anterior, se encarga de configurar y manejar los periféricos y módulos que conforman el controlador de proceso. Principalmente se distingue 2 partes en el programa del controlador de proceso, la configuración de módulos, periféricos y la implementación del programa que se muestra en el diagrama de bloques de la figura 41 correspondiente al PIDAD1 y la figura 60 para el PIDAD2. La programación se ha realizado en 2 tipos de lenguajes, lenguaje ASM30 y C30.

El lenguaje ASM30 es un tipo de lenguaje ensamblador para dsPIC y PIC24 de Microchip Inc. este lenguaje comprende alrededor de 83 instrucciones, que se agrupan en instrucciones de movimiento de datos, matemáticas, lógicas, de rotación, de bit, de comparación, de flujo de programas, para manejo de pila, de control y para el módulo DSP. Estas instrucciones pueden manejar hasta 4 operandos entre direcciones y literales, las cuales proporcionan una gran flexibilidad para el operador además de significar un aumento en la eficiencia del programa. Se maneja con MPLAB IDE que permite generar el archivo ejecutable (.COF) desde el archivo fuente (.S) para poder probar el funcionamiento del

programa, también genera el archivo hexadecimal (.HEX) para se pueda grabar en el hardware.

El lenguaje C30 está basado en ANSI x3.159-1989, que resulta en una plataforma de desarrollo de código C, el compilador es puerto del compilador GCC de la fundación del software libre. El compilador produce un archivo ensamblador, a este se pueden enlazar archivos objeto y librerías para producir el archivo ejecutable (.COF) que se puede cargar en MPLAB IDE donde se puede testear, y también se puede hacer la conversión al formato hexadecimal (.HEX) para que se pueda grabar en el hardware. La grabación del dispositivo se realiza utilizando el grabador GTP-USB lite y el software WINPIC800 (disponible en http://www.winpic800.com/), las opciones de trabajo escogidas son: oscilador XT con PLL de 8X, y sin código de protección en la memoria de datos y programa.

De lo anterior, el lenguaje ASM30 se resume en una lista de instrucciones muy flexibles y eficaces; C30 tiene una sintaxis muy similar al lenguaje tradicional C que nos permite mucha facilidad para implementar algoritmos complejos. Sin olvidar a MPLAB que es la herramienta que nos brinda la oportunidad de combinar los dos lenguajes, compilar, simular y generar el archivo .HEX para grabar en el hardware dsPIC.

2.3.2.1 Programación de la configuración de módulos y periféricos. Conforma la primera parte del programa de controlador de proceso, donde se declara las variables y se les asigna el espacio en la memoria de datos, se inicializa las variables y se configura e inicializa los periféricos y módulos, todo se realiza en el lenguaje ASM30. No se entrara en detalles técnicos de programación ya que no es el objetivo del trabajo. Para un mayor análisis se puede dirigir al literal (a) del ANEXO I donde se encuentran las líneas de programa con comentarios explicativos.

Se configura los puertos, puerto B como entrada analógica para el setpoint y la salida de la planta (en la emulación), puerto C como bits de control del proceso, puerto D como salidas para el actuador (en la emulación), puerto E, como bits de comunicación para el módulo SPI y como bits de salida para el actuador, puerto F para bits del módulo SPI.

El conversor A/D que se utiliza en la emulación se configura, con dos canales de entrada analógica, para el setpoint analógico y la salida de la planta, con un tiempo de muestreo de un ciclo de instrucción, conversión automática después del muestreo y un tiempo de conversión de 24 ciclos de instrucción.

El módulo SPI se configura en modo de 8 bits, con muestreo de bit en la mitad, estado de reposo en bajo y con flanco de inicio ascendente.

El módulo TIMER se configura como generador de interrupción con un periodo de 2mS, para controlar un tiempo de muestreo preciso.

Las interrupciones se configuran así: en el módulo SPI para informar que un dato ha llegado y necesita ser leído, en el módulo de notificación de cambio para interrumpir el flujo del programa, necesario para el botón de control de STOP y para el módulo TIMER.

2.3.2.2 Programación del algoritmo PIDAD. Después de la inicialización de la variables y configuración de periféricos y módulos, continua la programación del algoritmo del controlador PIDAD, se hace en su mayor parte en lenguaje C30 para los algoritmos de control PID y el mecanismo adaptivo difuso 1 y 2, la conversión de cifras a número flotante y viceversa, que son los datos de usuario que entran por teclado y se envían entre el controlador de interfaz y proceso. Algunas partes de programa se hacen en ASM30, por ejemplo, la atención de interrupciones, la lectura del conversor A/D, la comunicación SPI y el manejo del módulo de notificación de cambio. Sin distinción del lenguaje de programación se puede ordenar el algoritmo PIDAD en las siguientes subrutinas.

- **Subrutina de interrupción de comunicación SPI.** Esta subrutina da el inicio de la operación de control, se maneja dentro de la interrupción esto para lograr una sincronía inicial, o sea se puede llamar en cualquier momento de la ejecución del programa, esta rutina se encarga de recibir y enviar datos del controlador de interfaz, Se reciben los datos de usuario como son: velocidad y voltaje nominal del motor, modo de operación manual o adaptivo, modo de ejecución periódica o no periódica, tipo de setpoint digital o analógico, parámetros del controlador, k_p, k_i y k_d, los setpoints con sus respectivos tiempos de operación, todos en paquetes de 8 bits. Al finalizar el proceso se envían los datos del controlador de proceso hacia el controlador de interfaz en este caso las ganancias del controlador, k_p, k_i y k_d, esta comunicación ocurre siempre que el dispositivo controlador se configure como adaptivo. Por medio de esta subrutina se puede alterar el flujo del programa, dando al usuario la posibilidad de cambiar el setpoint del proceso. Para más detalles del código de la subrutina ver el literal (b) del ANEXO I.

- **Subrutina de conversión de cifra a número flotante.** Esta rutina se ejecuta después de la rutina anterior, tiene una función simple, la conversión de los datos recibidos por el controlador de interfaz a números flotante para que se puedan procesar adecuadamente. Se maneja con el comando selector *"switch"* de esta forma se pueden ordenar cada numero pulsado por el usuario en el lugar que corresponde, si es unidad, decena, centena o miles, o también decima, centésima o milésima. Por ejemplo:

1. Selector (cifra)
2. Caso 0, c2 = 0, ir a 12
3. Caso 1, c2 = 0.1, ir a 12
4. Caso 2, c2 = 0.2, ir a 12
5. Caso 3, c2 = 0.3, ir a 12
6. Caso 4, c2 = 0.4, ir a 12

7. Caso 5, c2 = 0.5, ir a 12
8. Caso 6, c2 = 0.6, ir a 12
9. Caso 7, c2 = 0.7, ir a 12
10. Caso 8, c2 = 0.8, ir a 12
11. Por defecto, c2 = 0.9, ir a 12
12. Fin selector

Si "cifra" se refiere a la cifra pulsada, y se ha identificado que debe ser una decima el pseudocódigo del anterior ejemplo, indica como otorgarle el valor que le corresponde, en relación con el ejemplo anterior, se suman las unidades, decimas, centésimas y milésimas y se obtiene le numero en punto flotante. Esta se implementa con la función de selección "*switch*" debido a que utiliza mucho menos tiempo de procesamiento que una multiplicación, el factor tiempo se tiene en cuenta debido a que esta rutina puede ser llamada en medio del proceso y se trata de reducir su duración al mínimo. Para más detalles del algoritmo ver el literal (e) del ANEXO I.

- **Subrutina de conversión de número flotante a cifra.** Hace lo contrario de la rutina anterior, convierte los números flotantes en cifras, para esto se aplican una serie de divisiones o multiplicaciones al número en proceso hasta desglosarlo en sus cifra correspondientes, esta tarea se realiza al finalizar el proceso, como tal el tiempo no es tan crítico, para más detalles del algoritmo ver el literal (f) del ANEXO I.

- **Subrutina de interrupción del TIMER.** La característica de interrupción de esta rutina sincronizada por el módulo TIMER hace que sea precisa y estable, la atención de esta rutina da paso a la ejecución de las rutinas de lectura de la planta, del mecanismo adaptivo y del controlador PID. Se debe tener en cuenta que las tres rutinas anteriormente mencionadas duren un tiempo menor al periodo del TIMER que coordina la ejecución de la interrupción, en este caso de 2mS. Para más detalles ver el literal (a) del ANEXO I.

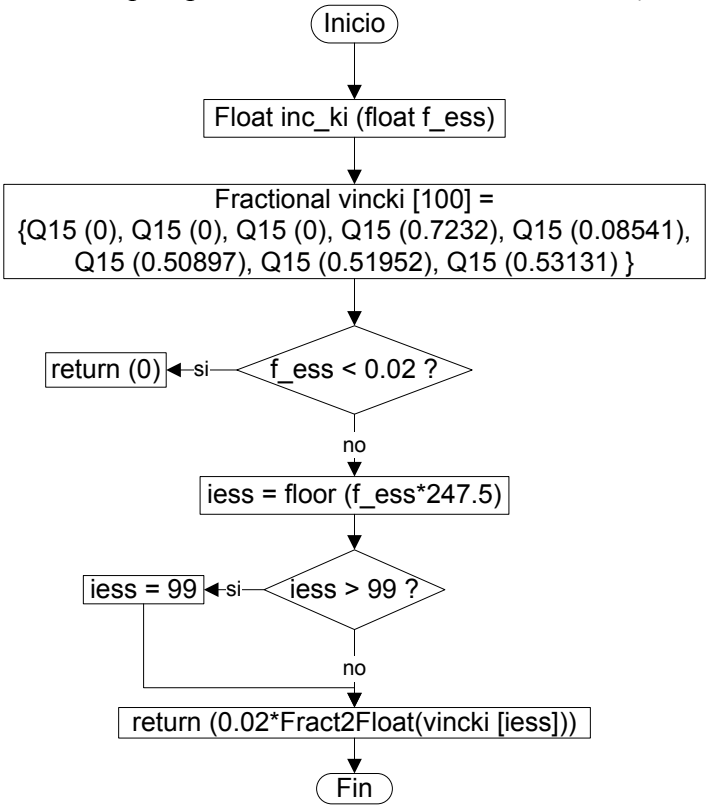
- **Subrutina de lectura de la planta.** Se ejecuta la lectura de la salida planta, utilizando el conversor A/D para conocer el valor de la salida de la planta que se manifiesta en forma de voltaje. La subrutina consiste en la lectura del registro donde se almacena el valor previamente muestreado y convertido. Para más detalles ver el literal (a) del ANEXO I.

- **Subrutina del algoritmo PID.** Se encarga de ejecutar el algoritmo de controlador PID, se implementa programando en lenguaje C30 el diagrama de flujo de la figura 12, para más detalles de la rutina ver el literal (b) del ANEXO I.

- **Subrutina del algoritmo del mecanismo de MAD.** Esta rutina inicia con el cálculo de las características del proceso, el cual es común para el MAD1 y 2, y finaliza con el cálculo de los incrementos o variación de las ganancias tanto para MAD1 y 2. La implementación del cálculo de las características se hace en lenguaje C30 siguiendo los diagramas de flujo de las figuras 31, 33, y 34, para t_s, e_{ss}, y ov respectivamente. Para más detalles de estas subrutinas ver el literal (b) del ANEXO I. Después de calcular el valor de las características de la salida de planta, se puede calcular la variación de las ganancias por medio de los mecanismos adaptivos.

  - **Programación del cálculo de la variación de las ganancias según MAD1.** Se calculan los incrementos de k_p, k_i, y k_d. Para el cálculo del incremento de k_p simplemente se codifica el diagrama de bloques de la figura 32 en lenguaje que no representa mucha dificultad, para el implementar el cálculo del incremento de k_i y k_d se utiliza el método del mapeo, que en programación del lenguaje C30 significa evaluar un vector haciendo corresponder la característica de entrada con el índice del vector. A continuación se muestra el diagrama.

Figura 65. Diagrama de bloques para el cálculo del incremento de k_i



"Fractional vincki" indica la declaración del vector de tamaño 100 del tipo "*fractional*", la función "*floor*" se utiliza para redondear el número flotante a entero, debido a que tiene que actuar como índice, y la función "Frac2Float" indica una conversión a numero "*fractional*" a flotante, ya que el incremento es de tipo flotante y los vectores prefabricados se declaran en tipo "*fractional*" o en formato Q15. En este formato los valores tienen que estar en el rango (0, 1) pero pueden tener hasta 6 cifras decimales, esta codificado en 16 bits, se hace muy útil para ahorrar memoria de datos al declarar vectores o matrices de tipo no entero. La implementación del cálculo del incremento de k_d se hace de igual forma. Para más detalles de estas subrutinas ver los literales (b) y (c) del ANEXO I.

- **Programación del cálculo de la variación de las ganancias según MAD2.** La explicación de la programación del MAD2 se ha abordado en la sección procesamiento del sistema difuso del MAD2 (…ver numeral 2.2.3…), la cual se describe en su totalidad en el diagrama de bloques de la figura 59. El código de esta rutina se puede observar en los literales (b) y (d) del ANEXO I.

- **Subrutina de interrupción de STOP.** Esta rutina con su característica de interrupción puede alterar el flujo del programa en cualquier momento y detenerlo, esta es opción útil y necesaria para mayor seguridad.

- **Organización de las subrutinas.** Las subrutinas anteriormente mencionadas conforman el algoritmo de control PID adaptivo difuso que se ejecutan en el controlador de proceso.

- **Rutinas auxiliares.** Estas rutinas se ejecutan de forma asincrónica y pueden alterar el flujo normal del programa en cualquier instante como el usuario lo necesite, por ejemplo, cambiar la velocidad, parada de emergencia del proceso.

1. Subrutina de interrupción de comunicación SPI (Inicio)
    a. Subrutina de conversión de cifra a número flotante

2. Subrutina de interrupción del TIMER

3. Subrutina de interrupción de STOP (Fin)
    a. Subrutina de conversión de número flotante a cifra

Las anteriores rutinas se encargan de dar el inicio (1), controlar un periodo de muestreo estable (2) y dar el fin (3) para la ejecución de las rutinas de control.

- **Rutinas de control.** Esta rutinas se ejecutan dependiendo del periodo de muestreo en forma periódica, conforman el algoritmo de control del controlador PID adaptivo difuso representando el flujo normal del programa, estas se apoyan en las rutinas auxiliares para, manejar el inicio y el fin de proceso, entrada de datos y generar el periodo de muestreo. En orden de ejecución son:

1. Subrutina de lectura de la planta

2. Subrutina del algoritmo del mecanismo de MAD1 y 2

3. Subrutina del algoritmo PID

4. ir a 1 (sincronizada con la subrutina de interrupción del TIMER)

3. INTERFAZ DE USUARIO

La interfaz es la encargada de interactuar con el usuario, se encarga de configurar los parámetros del proceso y ponerlo en marcha, además se comunica con el controlador de proceso para enviar los datos adquiridos.

3.1 FUNCIONAMIENTO DE LA INTERFAZ DE USUARIO

Permite una fácil comunicación con el usuario desplegando la información en una pantalla LCD textual y algunos indicadores de proceso, para lograr un fácil accionamiento cuenta con un teclado matricial y un panel de botones de control, para las entradas numéricas y el control de flujo del proceso, respectivamente. El control de los periféricos de entrada salida que se mencionan se hace por medio del controlador de interfaz implementado con un microcontrolador PIC que también se encarga del almacenamiento de datos y sincronización de la comunicación serial SPI con el controlador de proceso. El funcionamiento se divide en dos partes:

- Interacción con el usuario

- Comunicación serial SPI con el controlador de proceso

3.1.1 Interacción con el usuario. Es la entrada de datos por medio del teclado según la información que se indique en la pantalla antes de poner en marcha el proceso. Para que el dispositivo sea fácil, rápido de configurar y operar, se ha divido los datos de entrada en dos tipos, datos de configuración del controlador y proceso y datos de operación del proceso, como indica la siguiente tabla.

Los datos de configuración del controlador y del proceso se digitan una vez cada nuevo proceso, se entiende que esta configuración no cambia cada vez que se pretende operar la planta, en este modo se configuran datos específicos de la planta que se controlara (Velocidad y voltaje nominal del motor), además se configura algunas opciones del controlador (Modo de operación, parámetros del controlador y tipo de setpoint). A diferencia del anterior modo los datos de operación del proceso cambian según el usuario lo requiera cada vez que se inicia el proceso, en este modo se entran datos como el punto de referencia, el tiempo de operación y se escoge si la ejecución es periódica o no.

Tabla 29. Datos de entrada de la interfaz

Datos de configuración del controlador y del proceso	**Datos de operación del proceso**
Velocidad nominal del motor trifásico (500 – 6000 RMP)	Velocidades de operación, Velocidad 1, 2 y 3, [500, 5999]
Voltaje nominal del motor trifásico (100 – 230 VAC)	Tiempo de duración del proceso, Tiempo 1, 2 y 3, [1, 59]
Modo de operación del controlador, manual o adaptivo	Ejecución periódica
Parámetros del controlador PID, Ganancia k_p[100, 4999] Ganancia k_i[1, 4999] Ganancia k_d[100, 4999]	
Tipo de setpoint, digital o analógico	

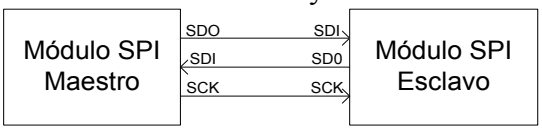
La interfaz permite navegar entre los dos modos mencionados, el funcionamiento de la interfaz de usuario y específicamente la interacción con el usuario con la entrada de datos se describe de forma precisa en el diagrama de flujo de la figura 68. La información digitada por el usuario se debe almacenar de forma permanente hasta que el usuario decida cambiarla, para esto se debe utilizar una memoria del tipo permanente EEPROM la cual se opera con en el controlador de interfaz.

Después de introducir los datos del proceso por medio de la interfaz y grabarlos en la memoria permanente, estos se envían al controlador de proceso que se prepara para iniciar la operación, para que el controlador entre en operación la interfaz se encarga de iniciarlo por medio de un indicador, durante el proceso la interfaz debe realizar algunas tareas como son: mostrar el tiempo de proceso, permitir intervención del usuario en el proceso, por ejemplo, un cambio de velocidad o una parada de proceso obligatoria. Estas opciones de interfaz durante el proceso se describen específicamente en la rutina de duración de proceso y cambio de velocidad; el diagrama de flujo de la figura 69 muestra cuales son los pasos para el funcionamiento de esta rutina, esta cuenta con algunas subrutinas, la rutina reloj de proceso se encarga de manejar un reloj de tiempo real el cual se visualiza en la pantalla y además permite detener el proceso según se haya programado, la subrutina enviar datos al controlador de proceso hace referencia a la comunicación serial entre el controlador de interfaz y el controlador de proceso.

3.1.2 Comunicación serial SPI con el controlador de proceso. Los datos de operación y configuración que tienen almacenados en el controlador de interfaz se deben transmitir hacia el controlador de proceso para iniciar el ciclo de proceso y posteriormente al finalizar el proceso el controlador de proceso debe transmitir hacia el controlador de interfaz los parámetros del controlador obtenidos, esta comunicación se logra por medio de la interfaz SPI, esta comunicación se maneja en un bus de tres líneas, dos líneas transfieren datos

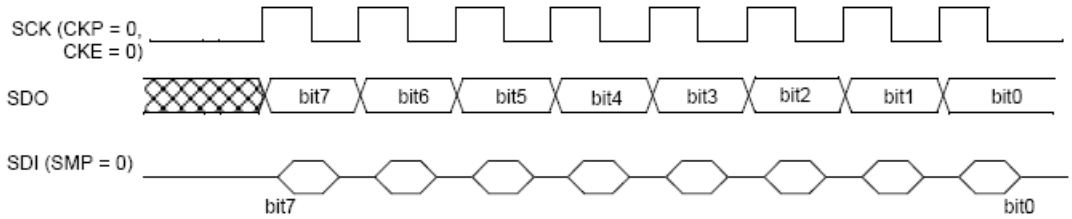
(SDI, SDO) uno en cada dirección y la tercer línea es la de reloj (SCK), las cuales sincronizan y transmiten paquetes de información de 8 bits.

Figura 66. Conexión entre módulos SPI maestro y esclavo



Generalmente los dos dispositivos que se comunican se configuran como maestro y esclavo donde el primero inicia la transferencia, maneja el reloj y transmite datos por SDO, y recibe simultáneamente por SDI. Existen dos aspectos a tener en cuenta, la polaridad y fase del reloj, la polaridad determina el estado de reposo de esta línea, que puede ser en estado alto o bajo, de igual forma para la fase del reloj la cual puede ser configurada por flanco de bajada o de subida indicando el instante en que los datos son desplazados, estos dos aspectos se deben configurar de forma correspondiente en los módulos SPI del controlador de interfaz.

Figura 67. Diagrama de tiempos de las líneas del módulo SPI



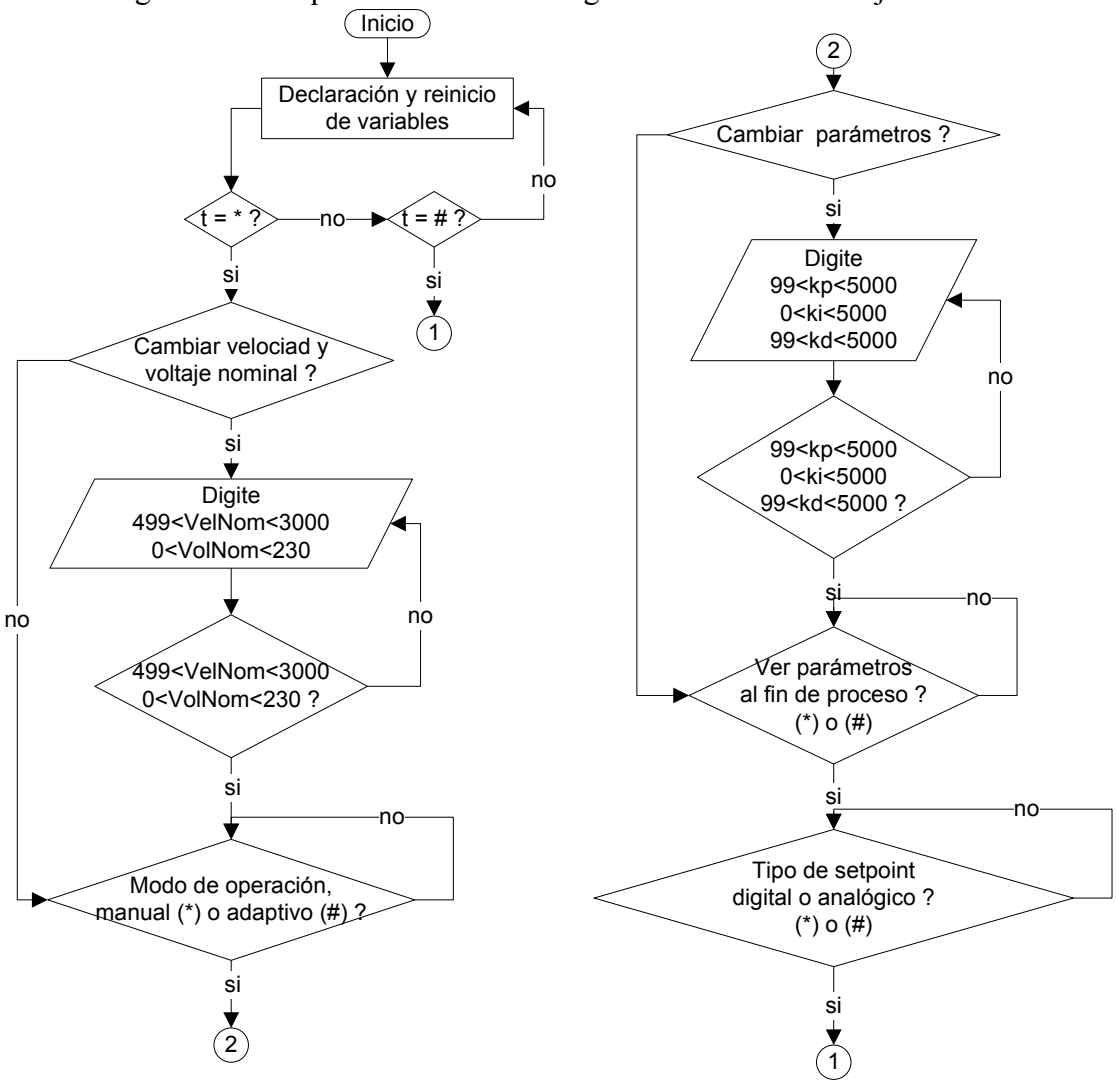
Fuente: Microchip Technology Inc. PIC16F877A Data Sheet, 2001.

En la comunicación el controlador de interfaz se define como maestro y el controlador de proceso como esclavo, antes de iniciar el proceso se hace una comunicación entre el controlador de interfaz hacia el controlador de proceso para transmitir en paquetes de 8 bits las siguientes variables: velocidad y voltaje nominal del motor, modo de operación manual o adaptivo, modo de ejecución periódica o no periódica, tipo de setpoint digital o analógico, parámetros del controlador, k_p, k_i y k_d, los setpoints con sus respectivos tiempos de operación, al finalizar el proceso, se establece comunicación entre el controlador de proceso hacia el controlador de interfaz y se trasmiten las variables: ganancias del controlador, k_p, k_i y k_d, esta comunicación ocurre siempre que el dispositivo controlador se configure como adaptivo, de lo contrario no tiene sentido transmitir los parámetros.

3.2 IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

El hardware de la interfaz de usuario se puede dividir en: periféricos de entrada y salida y controlador de interfaz de usuario. El firmware de la interfaz de usuario se programa según los diagramas de bloques de las figuras 68 y 69, estos se adaptaran según los módulos del hardware del controlador de interfaz, en este caso el microcontrolador PIC16F877A.

Figura 68. Diagrama de bloques de la rutina de ingreso de datos de trabajo del controlador



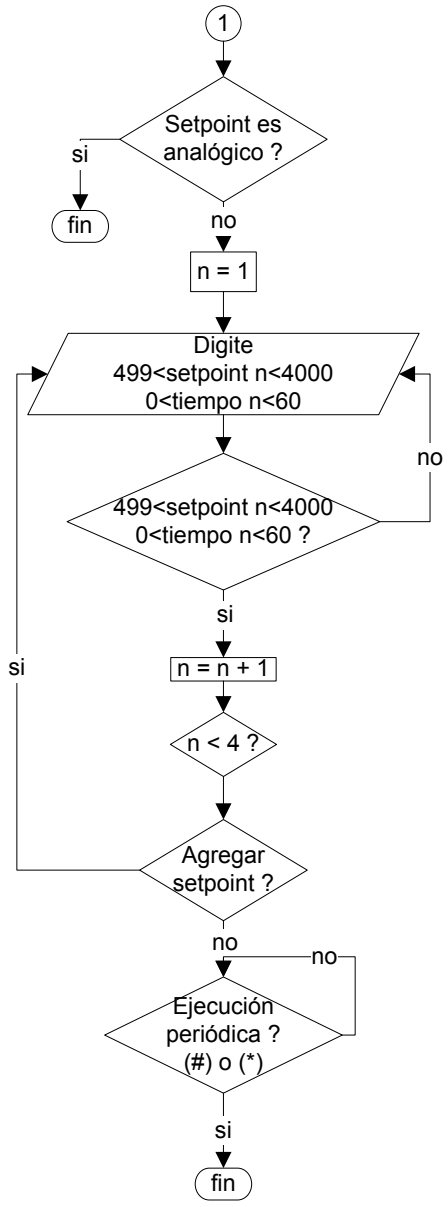
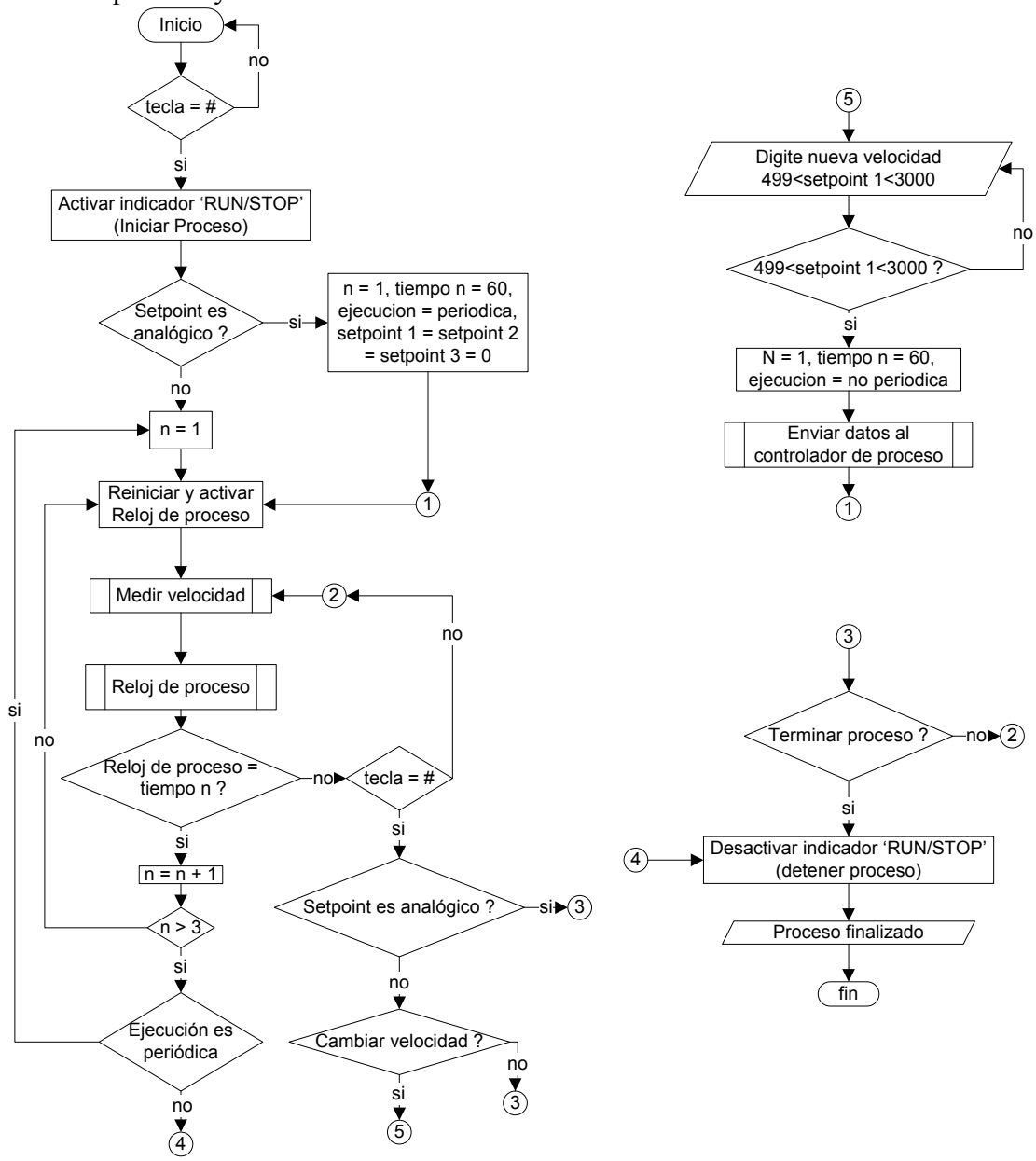

1
Setpoint es analógico ?
si
fin
no
n = 1
Digite
499<setpoint n<4000
0<tiempo n<60
no
499<setpoint n<4000
0<tiempo n<60 ?
si
si
n = n + 1
n < 4 ?
Agregar setpoint ?
no
no
Ejecución periódica ?
(#) o (*)
si
fin

Figura 69. Diagrama de bloques de la rutina de control de visualización de variables, duración de proceso y cambio de velocidad



3.2.1 Periféricos de entrada y salida. Permite la visualización de los datos como también la entrada de estos, facilitando la interacción con el usuario, se compone de teclado matricial 4x4 para la entrada precisa de datos, unos botones de control *STOP/RUN* y RESET para alterar el flujo normal del proceso, la pantalla de cristal liquido (LCD) de

formato alfanumérico de 16x2 con referencia QY-162A[17], y los indicadores de proceso que muestran el estado del proceso *STOP/RUN*.

3.2.2 Controlador de interfaz de usuario. Se encarga de manejar los periféricos de entrada y salida mencionados por medio de sus puertos, de forma que coordina la información que se muestra en la pantalla e indicadores y su respuesta dada por el teclado o botones de proceso.

Cuenta con una memoria predefinida no volátil donde se almacenan los mensajes que conforman los menús que se muestran en la pantalla, una memoria temporal donde se guardan los datos digitados y las opciones escogidas por el usuario, un módulo de interfaz serial para comunicación con el controlador de proceso, y lo más importante una memoria *flash* (memoria *EEPROM* desarrollada) de programa donde se almacena el software controlador de interfaz o firmware que coordina todos los periféricos y los módulos mencionados. Para mayor facilidad, la explicación se abordara en dos partes elementales.

- Hardware de controlador de interfaz de usuario

- Programación del firmware del controlador de interfaz de usuario

3.2.2.1 Hardware de controlador de interfaz de usuario. Está conformado principalmente por el microcontrolador PIC16F877A de Microchip, conectores y algunos circuitos auxiliares para el microcontrolador.

- **Microcontrolador PIC16F877A.** Es un microcontrolador de alto desempeño con una CPU (Unidad Central de Proceso) RISC (*Reduced Instruction Set Core* - Núcleo con Set Reducido de Instrucciones) y organizado con arquitectura *Harvard*, fabricado por Microchip Technology Inc, es de tecnología CMOS con bajo consumo de potencia, se puede programar con un set de 35 instrucciones, tiene una velocidad de operación de 4Mhz, amplio rango de voltaje de operación 2.0V-5.5V DC, corriente de 25mA como fuente y sumidero, programación vía serial ICSP (*In-Circuit Serial Programming*), y encapsulado PDIP *(Plastic Dual in Line Pin)* de 40 pines[18].

La memoria de programa es de tipo flash de 8Kx14 bits con un contador de programa de 13 bits, la memoria RAM tiene un tamaño de 368x8 bits de memoria de datos donde se ubican los registros de funciones especiales como el de estatus y opciones que permiten conocer el

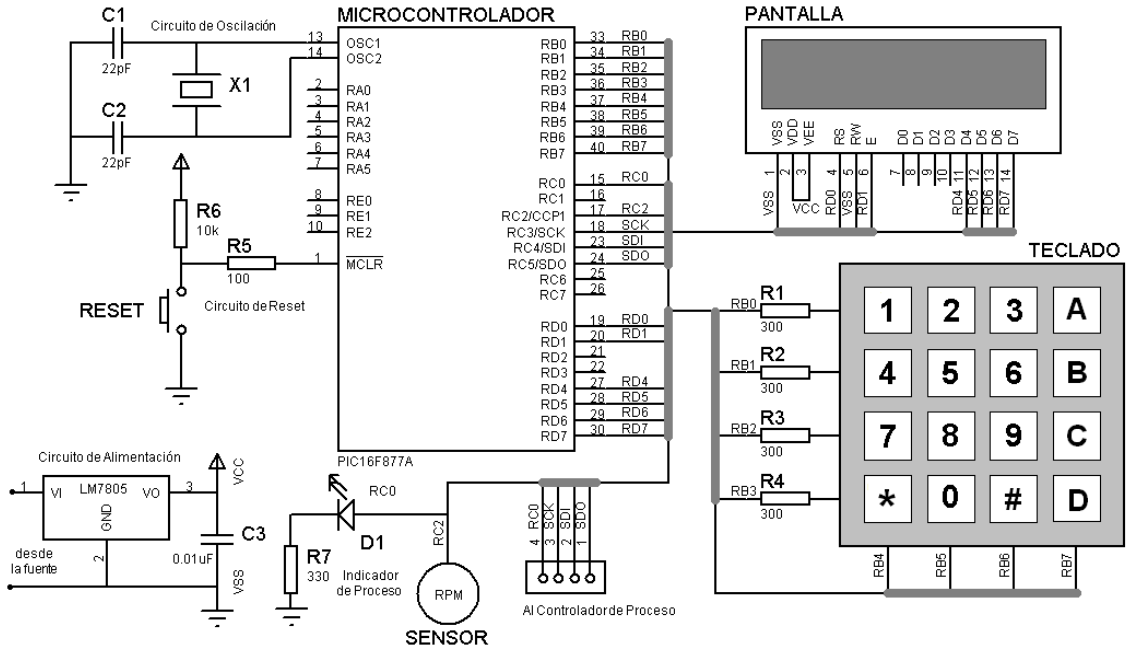
[17] HD44780U (LCD-II). Hitachi, Ltd., 1998, Disponible en Internet: URL: http:semiconductor.hitachi.com/
[18] PIC16F877A Data Sheet, Microchip Technology Inc, 2003, Disponible en Internet: URL http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf, 235p.

estado de indicadores según el resultado de las operaciones aritméticas y modificar opciones adicionales. La memoria no volátil EEPROM tiene 256x8 bytes.

Este microcontrolador cuenta con 5 puertos configurables como entradas y salidas, el puerto B con 8 líneas digitales que permiten interrupción, estas tiene resistencias pull-up internas que facilitan la conexión del teclado, el puerto C con 8 líneas digitales, tres de las cuales permiten utilizar el módulo SPI o interfaz serial de periféricos que se utiliza para comunicar el controlador de interfaz de usuario con el controlador de proceso. El módulo SPI al ser serial utiliza 3 pines de puerto solamente, SCK (*Serial Clock*) es RC3, SDI (*Serial Data input*) es RC4 y SDO es RC5 (*Serial Data Output*), RC0 se configura como salida y se conecta hacia el controlador de proceso este se usa como bandera de alerta de los estados RUN o STOP del proceso. El puerto D tiene 8 líneas bidireccionales, 6 líneas de este puerto se encargan de manejar el LCD.

En la siguiente figura se muestra la interconexión de los elementos y circuitos auxiliares que conforman el hardware del controlador de interfaz, al igual que el hardware del controlador de proceso se utilizan los mismos circuitos auxiliares de reinicio, de alimentación y de oscilación (…ver numeral 2.3.1…), la única diferencia es en la velocidad del oscilación, en este hardware el cristal de oscilación que marca la velocidad del procesador es de 4Mhz con ciclo de instrucción de 1uS.

Figura 70. Circuito del controlador de interfaz



El módulo temporizador 1 es un contador de 16 bits preescalable, que permite generar interrupción según se programe su periodo, este se utiliza para generar la base de tiempo y controlar el tiempo de duración del proceso.

La memoria EEPROM permite realizar dos tipos de operaciones, operación de escritura y lectura, ambas se ejecutan manipulando el registro de control de la memoria, esta es útil para mantener de forma permanente la configuración de usuario y los datos digitados, así pues, cuando el mecanismo adaptivo encuentra unos valores aptos para la planta estos se guardan en la memoria. La escritura de estos datos en la memoria es de forma opcional. De esta forma el controlador no pierde la experiencia ganada al interactuar con la planta.

Igual que el controlador de proceso este microcontrolador también tiene el módulo de interfaz serial de periféricos (SPI), se utiliza para comunicar los datos de proceso al controlador de proceso, de forma que este módulo se comunica como dispositivo maestro, genera la señal de reloj y sincroniza las operaciones de envió y recepción de bits. El módulo SPI del controlador de interfaz se configura en modo maestro tanto para transmitir datos como para recibirlos, de tal forma que este genera la señal de reloj por el pin SCK, la polaridad del reloj en nivel bajo y el flanco para este caso es ascendente. Estas configuraciones deben coincidir en el módulo SPI esclavo o en el controlador de proceso, el registro que almacena el dato recibido es el registro de 8 bits SSPBUF en este también se alista el dato para ser enviado, cuando se termina la escritura de este registro la transmisión o recepción inicia automáticamente.

3.2.2.2 Programación del firmware del controlador de interfaz. El firmware de interfaz es la parte complementaria del hardware anteriormente descrito, este se encarga de manejar los periféricos y módulos que conforman la interfaz de usuario. Este se encarga de desplegar los menús de configuración del controlador, interactuar con la respuesta del operador de forma simple y cómoda, grabar las opciones de usuario, comunicarse con el controlador de proceso, controla y visualiza la duración del proceso, manejar las interrupciones de proceso y permite visualizar la variable del proceso, todo se realiza bajo las instrucciones de programación.

- **Lenguaje ensamblador.** El software para programar en lenguaje ensamblador es proporcionado por la empresa productora del dispositivo *Microchip*, en este caso se utilizo MPLAB, que también cuenta con el programa ensamblador que es MPASMWIN el cual pasa se encarga de traducir el programa escrito en ensamblador con extensión .ASM a lenguaje de maquina con extensión .HEX. Este archivo se puede grabar dentro de la memoria de programa mediante un grabador. Para desarrollar el programa de interfaz de usuario se utilizo MPLAB IDE v8.00, las pruebas y simulaciones se hicieron en ISIS v7. El programa principal utiliza 4 librerías de usuario con el fin de facilitar el manejo de periféricos y ordenar el código del programa.

- **Librería de retardo.** Cuenta con varios tiempos de temporización desde 4us hasta 20s, Esta librería se utiliza para controlar y sincronizar el tiempo que tardan en ejecutarse algunas acciones.

- **Librería de manejo de teclado y LCD.** Estas librerías auxiliares reúnen grupos de instrucciones que ejecutan una acción propia para cada periférico, estos grupos se nombran de forma que sean un apoyo nemotécnico en el desarrollo del programa principal, ahorran en gran medida la extensión del programa.

- **Librería de mensajes.** En esta librería se agrupan los mensajes individuales más utilizados para no extender el código de programa y además se manejan algunos efectos de visualización.

- **Grabador de memoria de programa y bits de configuración.** con los bits de configuración se elige configuración de oscilador, tipos de reinicio, habilitación de WDT, código de protección, opción, etc. se aloja en la dirección 2007h y se conoce como palabra de configuración. Para grabar la memoria de programa del microcontrolador se usa el software WinPIC800 el cual se carga el archivo .HEX resultado de la compilación, el hardware para grabar el dispositivo es GTP-USB.

- **Descripción del diagrama de flujo de la interfaz de usuario.** Se hace necesario una rutina principal de interfaz de usuario que integre los diagramas de interacción con el usuario de las figuras 68, 69, y además maneje en detalle las variables y módulos para la operación de los periféricos, la siguiente figura describe de forma completa la operación de la interfaz de usuario, para observar con mayor detalle la programación se puede referir al firmware en lenguaje de maquina en el ANEXO K, librerías y archivos auxiliares (ver ANEXOS L – Q).

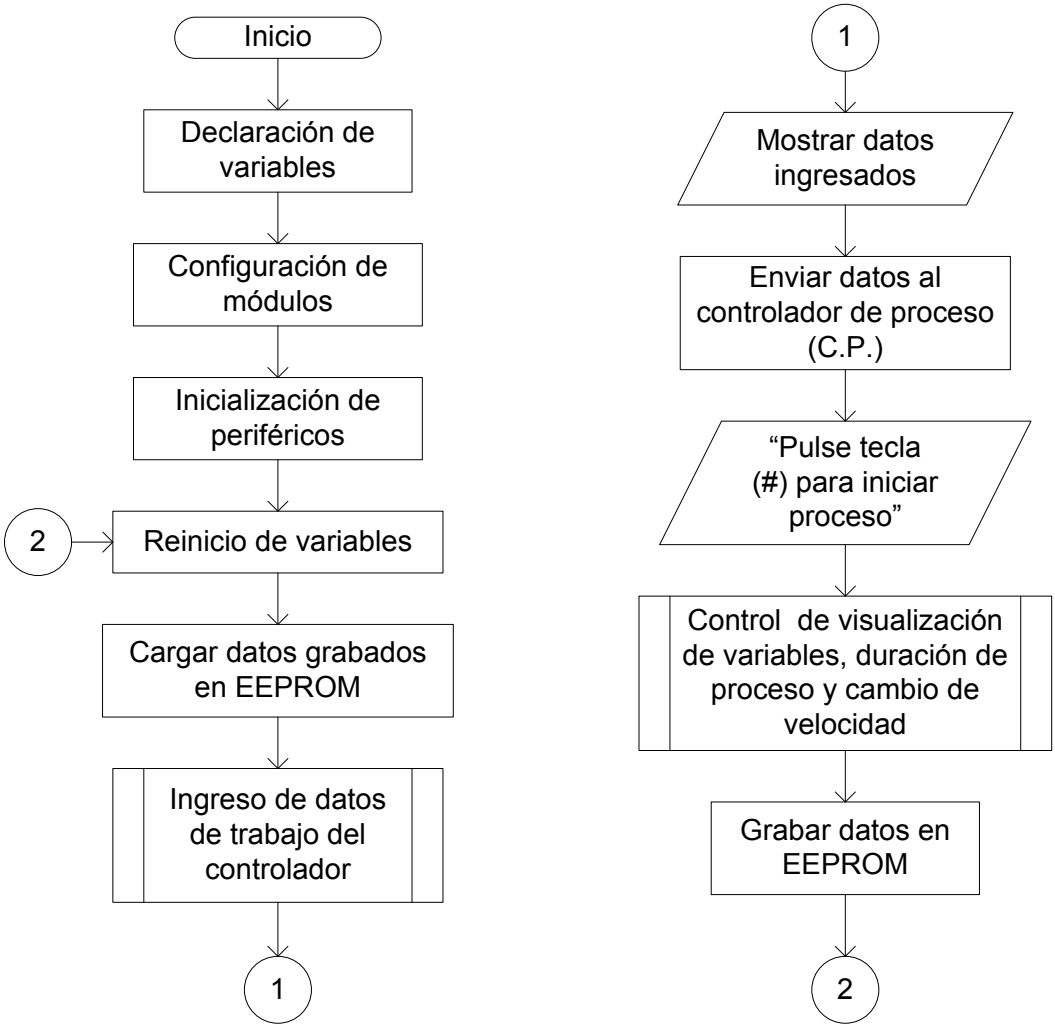
- **Rutina principal de interfaz de usuario.** El programa inicia con la declaración de variables donde se les asigna el espacio correspondiente en la memoria de datos, también se asigna el espacio correspondiente de los mensajes en la memoria de programa, se continua con la configuración de módulos, en este paso se configuran los módulos anteriormente descritos, puertos de entrada salida, el temporizador TMR1, módulo de captura CCP1, el módulo de interfaz serial SPI y se configura el módulo de interrupciones (RB4:7, CCP1). Se inicializa los periféricos de salida y entrada como son el teclado y la pantalla, se reinician a su valor por defecto todas las variables declaradas. Se cargan los datos k_p, k_i, k_d de la memoria EEPROM, se ejecuta la rutina de ingreso de datos donde se pide al usuario digitar todos los datos necesarios para iniciar el proceso, después se muestran algunos datos ingresados y se envían estos al controlador

de proceso haciendo uso del módulo SPI. Se pide al usuario que confirme el inicio del proceso y empieza a ejecutar la rutina de control de visualización que controla la duración del proceso y las interrupciones de flujo de este. Al finalizar el proceso, se guardan los parámetros PID enviados por el controlador de proceso si esta en modo adaptivo o simplemente se guardan los parámetros PID digitados por el usuario.

- **Rutina ingreso de datos de trabajo del controlador.** Maneja la interacción con el usuario, se llenan los datos necesarios para iniciar el proceso, como son: Modo de operación, parámetros PID, tipo de setpoint o punto de referencia, en caso de punto de referencia digital, se requiere los valores de los puntos de referencia con sus respectivos tiempos y el tipo de ejecución. Esta hace uso de la subrutina de interrupción de teclado en esta se atiende la interrupción por RB4:7 y se identifica la letra presionada, de esta forma puede dar valor a una variable, visualizarla en la pantalla y almacenarla.

- **Rutina de control de visualización de variables, duración de proceso y cambio de velocidad.** Se da la orden para iniciar el controlador de proceso por medio del indicador RUN/STOP, en esta rutina se maneja la duración del proceso con ayuda del TMR0 para mayor precisión y también la visualización del tiempo, el cambio de velocidad en caso de ser punto de referencia digital y también la posibilidad de terminar el proceso de forma obligatoria por medio del teclado.

Figura 71. Diagrama de bloques rutina principal del controlador de interfaz

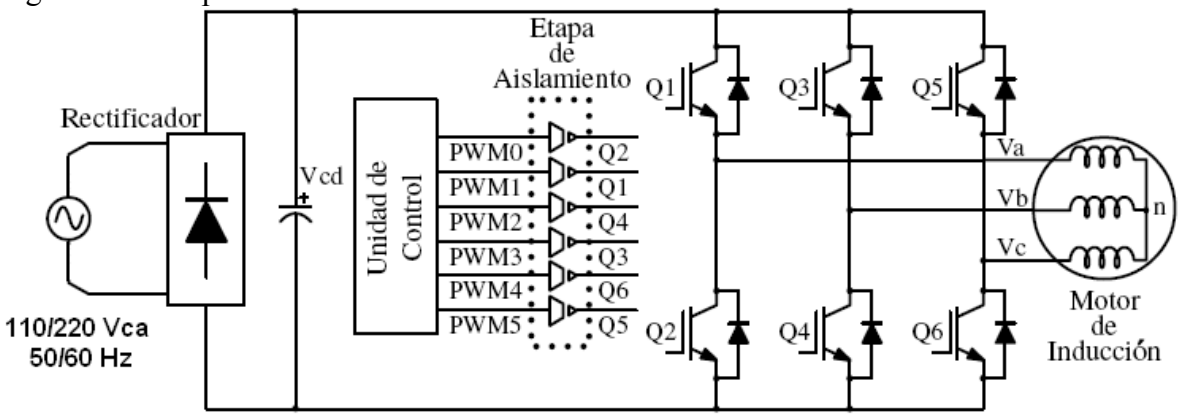


4. ACTUADOR, SENSOR Y FUENTE DE ALIMENTACIÓN DEL PROCESO

4.1 CONTROL VOLTAJE/FRECUENCIA A MOTOR DE INDUCCIÓN TRIFÁSICO USANDO MODULACIÓN EN ESPACIO VECTORIAL

El control escalar de velocidad es el método más común encontrado en variadores de velocidad para motores de inducción. Se basa en el principio de que la velocidad base de un motor es proporcional a la frecuencia del voltaje de alimentación e inversamente proporcional al número de polos de su estator. Sin embargo, para asegurar una operación adecuada a velocidades menores a la nominal, es necesario reducir linealmente el voltaje entre las terminales del motor. Los componentes usuales de un variador de velocidad se muestran en la siguiente figura. El puente rectificador de diodos y los capacitores convierten la entrada de corriente alterna a corriente directa con un rizo despreciable. El inversor, bajo el control de un microcontrolador (μC) eléctricamente aislado, sintetiza el voltaje de corriente directa (CD) en 3 voltajes de corriente alterna (CA) trifásico de amplitud y frecuencia variables[19].

Figura 72. Componentes usuales de un variador de velocidad



Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

En el prototipo construido, el voltaje y corriente del bus CD son monitoreados para detectar condiciones de falla. Adicionalmente, la velocidad real del rotor es retroalimentada por medio de un codificador (encoder) digital, ligado al eje del motor de inducción a ser controlado

[19] BIMAL, Bose. Modern Power Electronics and AC drivers, Prentice Hall, 2001. 501p.

La modulación en espacio vectorial (SPWM) de los inversores trifásicos y especialmente aplicados a los motores de inducción para el control de la velocidad a cupla constante, constituye un sistema de control de alta eficiencia. Ejecuta un algoritmo sofisticado que proporciona un 15% adicional en el voltaje de salida, comparado con el algoritmo de modulación sinusoidal típico; mitigando la distorsión armónica y las perdidas por conmutación. Es una técnica basada en el hecho de que los voltajes trifásicos el motor de inducción pueden convertirse en un solo vector rotatorio, compatible con el control escalar de velocidad.

Es decir que el sistema trifásico simétrico de tensiones de variación sinusoidal en el tiempo como el dado por las ecuaciones 52 a 54 queda reemplazado por una expresión vectorial en la ecuación 58 con la cual se realizan todos los estudios de este control.

El sistema trifásico de tensiones de red, se representa por:

$$V_R = V_m sen(\omega t) \tag{52}$$

$$V_S = V_m sen(\omega t - 2\pi/3) \tag{53}$$

$$V_T = V_m sen(\omega t - 4\pi/3) = V_m sen(\omega t + 2\pi/3) \tag{54}$$

Donde V_m es el valor de pico o amplitud de cada tensión y su valor eficaz es $V = V_m/\sqrt{2}$ y $\omega = 2\pi f$ es la velocidad angular en rad/s.

Al estudiar el motor asincrónico se observa que el flujo o la inducción magnética en el entrehierro del motor, puede representarse por un vector que gira a la velocidad angular sincrónica ω y tiene un módulo de valor máximo constante que vale $B = 1{,}5B_{fase}$, es decir su módulo es mayor que el de una fase en el factor 3/2 .

En la modulación vectorial el análisis presenta algunos aspectos similares pero requiere ser adaptado a los valores de tensiones que se desean obtener, para lo cual resulta conveniente trabajar con operadores complejos, ya que en definitiva el vector espacial es un número complejo.

Para el análisis del control vectorial, es preferible expresar el sistema trifásico de tensiones en función del coseno, a efectos de obtener ecuaciones más simples.

A su vez se toman las tensiones simples de fase que entrega el inversor en a, b y c (bornes), con respecto al centro o n (neutro) de la carga equilibrada conectada en estrella, por lo cual se tiene:

$$V_a(t) = V_m \cos(\omega t) \tag{55}$$

$$V_b(t) = V_m \cos(\omega t - 2\pi/3) \tag{56}$$

$$V_c(t) = V_m \cos(\omega t - 4\pi/3) = V_m \cos(\omega t + 2\pi/3) \quad (57)$$

El vector espacial que representa al sistema trifásico mencionado se obtiene con la transformación de Park, así

$$\bar{V} = \frac{1}{c}\left[V_a(t)e^{j0} + V_b(t)e^{j\frac{2\pi}{3}} + V_c(t)e^{j\frac{4\pi}{3}}\right] \quad (58)$$

Donde los operadores complejos, ubican a tres vectores fijos en el plano complejo y desfasados 120º entre sí y las tres funciones indican que dichos vectores no son constantes en magnitud, sino variables en el tiempo conforme al sistema de ecuaciones anterior. El coeficiente c es una constante de proporcionalidad, vale 3/2 para poder conservar la magnitud V_m de las tensiones dadas en el sistema de ecuaciones anterior, en la expresión final del vector dado por la ecuación anterior. Si el análisis se hace sobra la base de Potencia constante, entonces es $c = \sqrt{3/2}$. Escribiendo las funciones trigonométricas en forma exponencial, basadas en la identidad de Euler y reemplazando en ecuación anterior, se obtiene:

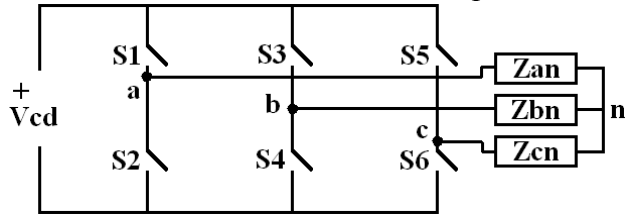
$$\bar{V} = \frac{2}{3}V_m\left[\frac{e^{j\omega t} + e^{-j\omega t}}{2} + \frac{e^{j\omega t} + e^{-j\left(\omega t + \frac{4\pi}{3}\right)}}{2} + \frac{e^{j\omega t} + e^{-j\left(\omega t + \frac{8\pi}{3}\right)}}{2}\right] \quad (59)$$

$$\bar{V} = \frac{2}{3}V_m\frac{3}{2}e^{j\omega t} \quad (60)$$

$$\bar{V} = V_m e^{j\omega t} = \sqrt{2}V_{efic}.e^{j\omega t} \quad (61)$$

La anterior ecuación nos da el vector espacial que gira en el plano complejo, con módulo V_m constante y velocidad sincrónica constante ω y representa al sistema trifásico simétrico de tensiones, siempre que alimente a una carga trifásica equilibrada. El paso siguiente es relacionar la anterior ecuación con las tensiones que entrega el inversor y el vector espacial.

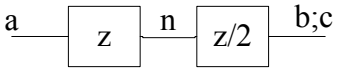
Figura 73. El inversor con las referencias de tensiones simples



Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

El inversor opera siempre con tres elementos en conducción y tres abiertos y siendo la carga en estrella equilibrada, la impedancia que ve el inversor en todo momento es como muestra la siguiente figura, para el estado particular de las llaves S1, S4, S6 cerradas.

Figura 74. Valores de impedancias y las respectivas tensiones válidas durante la conducción de los elementos S1, S4, S6



$$V_{an} = (2/3)E \tag{62}$$

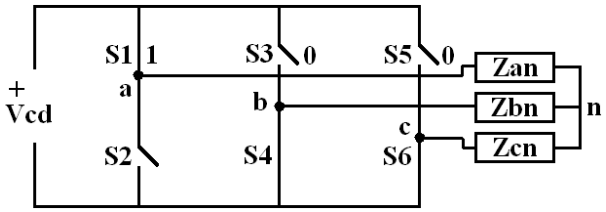
$$V_{bn} = V_{cn} = -(1/3)E \tag{63}$$

$$V_{ab} = E\ V_{bc} = 0\ V_{ca} = -E \tag{64}$$

Estos valores de tensión no son valores eficaces, los cuales se calculan integrando la onda de salida en el período completo.

Al ser las tensiones simétricas y la carga equilibrada, no existe corriente por el conductor neutro. Neutro se mantiene flotante, no se conecta a ningún potencial. El estado de conducción mencionado se sintetiza en la siguiente figura, y corresponde al vector espacial para ese estado de conducción, al cual se denomina $\bar{V}_1$.

Figura 75. Combinación 100 de las llaves del inversor



Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

Para reconocer cada estado de conducción, es suficiente identificar a los tres elementos superiores del puente. Se indica con un 1 cuando están cerrados y con un 0 cuando están abiertos, en consecuencia para el estado de la anterior figura será: $\bar{V}_1 = 100$, Observando las figuras 74 y 75 se tiene que la tensión $E = V_{an} + V_{n0}$ y como $V_{an} = (2/3)E$, resulta $V_{n0} = (1/3)E$ con estos valores, se intuye que el valor V_m del vector espacial de la ecuación 61 es $(2/3)E$, pero esto debe ser demostrado, lo cual se hace trabajando con la

ecuación 58 y los valores V_{an}, V_{bn}, V_{cn} que entrega el puente. De la figura anterior y reemplazando los voltajes de fase en la ecuación 58, se obtiene:

$$\bar{V}_1 = \frac{2}{3}\left[\frac{2E}{3}e^{j0} - \frac{E}{3}e^{j\frac{2\pi}{3}} - \frac{E}{3}e^{j\frac{4\pi}{3}}\right] = \frac{2}{3}E \tag{65}$$

Este es el valor y posición del vector espacial en el plano complejo para el estado de conducción de la figura 75. Se debe repetir este análisis para los restantes estados de conducción del inversor. Los 8 posibles estados de conmutación del inversor, y sus correspondientes voltajes fase-neutro se muestran en la siguiente tabla. Durante los estados activos del 1 al 6, la energía de la fuente de CD es transferida hacia las terminales del motor. Lo opuesto ocurre durante los estados inactivos 0 y 7. En consecuencia para cada combinación de llaves, utilizando la ecuación 58 se obtienen los 8 vectores básicos con su valor y ubicación en el plano complejo.

Tabla 30. Estados de conmutación del inversor y voltaje correspondientes

Transistores	V_{an}	V_{an}	V_{an}	V_S	Vector
Q2,Q4,Q6	0	0	0	0	V_0
Q1,Q4,Q6	$2V_{CD}/3$	$-V_{CD}/3$	$-V_{CD}/3$	$2V_{CD}/3$	V_1
Q1,Q3,Q6	$V_{CD}/3$	$V_{CD}/3$	$-2V_{CD}/3$	$2V_{CD}e^{j\frac{\pi}{3}}/3$	V_2
Q3,Q2,Q6	$-V_{CD}/3$	$2V_{CD}/3$	$-V_{CD}/3$	$2V_{CD}e^{j\frac{2\pi}{3}}/3$	V_3
Q2,Q3,Q5	$-2V_{CD}/3$	$V_{CD}/3$	$V_{CD}/3$	$2V_{CD}e^{j\frac{\pi}{3}}/3$	V_4
Q2,Q4,Q5	$-V_{CD}/3$	$-V_{CD}/3$	$2V_{CD}/3$	$2V_{CD}e^{j\frac{4\pi}{3}}/3$	V_5
Q1,Q4,Q5	$V_{CD}/3$	$-2V_{CD}/3$	$V_{CD}/3$	$2V_{CD}e^{j\frac{5\pi}{3}}/3$	V_6
Q1,Q3,Q5	0	0	0	0	V_7

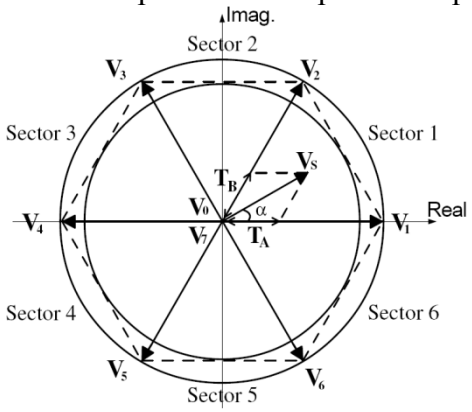
Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

Estos 8 vectores llamados básicos o directores, son los únicos vectores que puede generar el inversor y por tanto son vectores fijos en el plano complejo y se ubican como muestra la siguiente figura, donde los 6 vectores activos tienen el mismo módulo $2E/3$ y están desfasados 60º entre sí, en cambio los vectores nulos solo pueden indicarse como un punto en el centro del plano complejo ya que su módulo es cero y aún no tienen referencia de duración. La acción de cada uno de los 6 vectores activos dura como máximo 60º, mientras que los 2 vectores nulos solo accionarán cuando se introduzca un espacio de tiempo muerto, es decir cuando se reduzca la acción de los vectores activos, introduciendo un tiempo muerto ya sea con V_0 o con V_7. Los 6 vectores activos forman un hexágono equilátero que encierra un círculo tangente a sus lados. El punto de tangencia del círculo con los lados del hexágono determina el valor máximo del vector modulado (V_{max}), para

lograr una modulación vectorial de tensiones senoidales. Este vector V_{max} al girar describe el círculo mencionado.

El espacio exterior al círculo inscripto en el hexágono corresponde a la sobremodulación, la cual se interpreta como en el control escalar, de índice de modulación $m > 1$ y por tanto deja de ser lineal la relación entre la tensión de salida y dicho índice.

Figura 76. Ubicación de los vectores espaciales en el plano complejo



Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

Vector V_s del sector I, modulado entre V_1, V_2 y V_0, el valor de V_{max} es:

$$Vmax = \frac{2}{3}Ecos(\pi/3) = \frac{2}{3}E\frac{\sqrt{3}}{2} = \frac{E}{\sqrt{3}} \qquad (66)$$

El anterior resultado es el máximo valor que se puede obtener para el vector modulado con trayectoria sinusoidal.

4.2 MODULACION DEL VECTOR ESPACIAL

De la figura anterior el vector Vs resulta al conmutar entre los estados del inversor a una frecuencia F_{PWM}, determinando el tiempo de muestreo $T_S = 1/F_{PWM}$.

Tomando como referencia la figura anterior, $\bar{V}_S$ puede ser representado como en la siguiente ecuación, al encontrarse en el primer sector y mantener un ángulo α con respecto al eje x:

$$\bar{V}_S = \left(\frac{T_A}{T_S} \times \bar{V}_1\right) + \left(\frac{T_B}{T_S} \times \bar{V}_2\right) + \left(\frac{T_{0/7}}{T_S} \times \bar{V}_{0/7}\right) \quad (67)$$

Los intervalos de tiempo T_A, T_B, y $T_{0/7}$ se calculan de tal forma que el voltaje promedio por segundo, producido por los vectores $\bar{V}_1$, $\bar{V}_2$, $\bar{V}_{0/7}$, a lo largo de los ejes $x - y$ resulte igual al producido por el vector espacial de referencia $\bar{V}_S$. El índice de modulación se define como: $m = \frac{|\bar{V}_S|}{2E/3}$. Donde $|\bar{V}_S|$ es la amplitud del vector $\bar{V}_S$. Proyectando $\bar{V}_S$ a lo largo de los ejes, se obtiene:

$$2E/3 \times sen(\pi/3) \times T_B = |\bar{V}_S| \times sen(\alpha) \times T_S \quad (68)$$

$$(2E/3) \times T_A + ((2E/3) \times cos(\pi/3) \times T_B) = |\bar{V}_S| \times cos(\alpha) \times T_S \quad (69)$$

Donde α representa al ángulo formado por el vector $\bar{V}_S$ con respecto a $\bar{V}_1$. Resolviendo para T_A y T_B se obtiene:

$$\frac{T_A}{T_S} = \frac{2}{\sqrt{3}} \times m \times sen\left(\frac{\pi}{3} - \alpha\right) \quad (70)$$

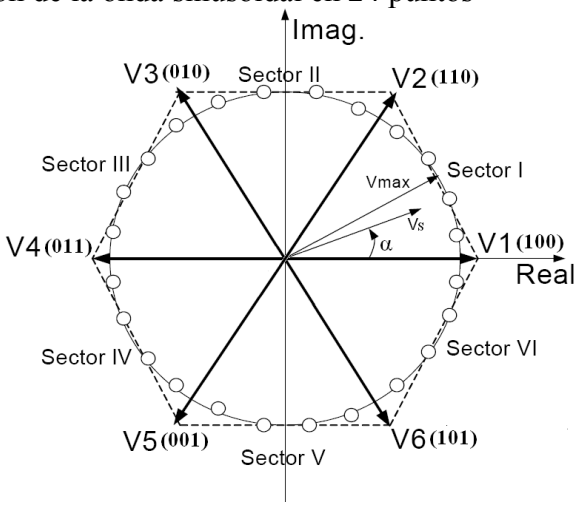
$$\frac{T_B}{T_S} = \frac{2}{\sqrt{3}} \times m \times sen(\alpha) \quad (71)$$

El algoritmo SPWM requiere de este último par de ecuaciones para calcular los tiempos de conmutación en cada nuevo periodo PWM.

4.2.1 Funcionamiento en "*Six-Step*". Conceptualmente si cada uno de los vectores, $\bar{V}_1$ al $\bar{V}_6$ se lo mantiene activo con el valor máximo dado por la ecuación 66, durante el tiempo máximo $T_S = T/6$ correspondiente a un sextante del hexágono, se obtendrá un funcionamiento parecido al de un inversor tradicional E-180º. Esta forma de funcionamiento en control vectorial se llama "*six-step*" y no presenta aún modulación alguna ya que los vectores nulos no intervienen. En consecuencia la trayectoria sinusoidal pretendida quedará discretizada con solo 6 puntos dados por los 6 vectores activos de la figura 76. Con este tipo de control la tensión a la salida del inversor no se puede variar y por tanto no permite variar la velocidad del motor ya que si se disminuye la frecuencia, teniendo en cuenta la relación V/F se producirá una saturación del núcleo. No obstante puede lograrse un funcionamiento especial en "*six-step*", introduciendo tiempos muertos al principio y final de cada sextante, en la proporción adecuada para lograr V/F constante. De todas maneras, sus resultados no son muy aceptables y si se pretende un control de esta naturaleza es preferible implementarlo con el control escalar SPWM de un solo pulso.

4.2.2 Funcionamiento en modulacion vectorial. Conceptualmente consiste en generar nuevos vectores con los 8 que entrega el inversor, lo cual se logra reduciendo el tiempo de acción de los 6 vectores activos fijos intercalando tiempos muertos con los vectores nulos. Esta acción se lleva a cabo en cada sextante con los dos vectores activos que encierran al mismo y los dos vectores nulos. Para este fin se parte con la informacion del numero de puntos de discretizacion de la onda senoidal que se pretende obtener, y un sentido de giro del vector modulado. La siguiente figura muestra 24 puntos de discretizacion con los respectivos vectores directores y sus combinaciones de llaves de los transistores superiores del puente inversor; donde el angulo α corresponde a la ubicación del vector a generar en cada sector.

Figura 77. Discretizacion de la onda sinusoidal en 24 puntos

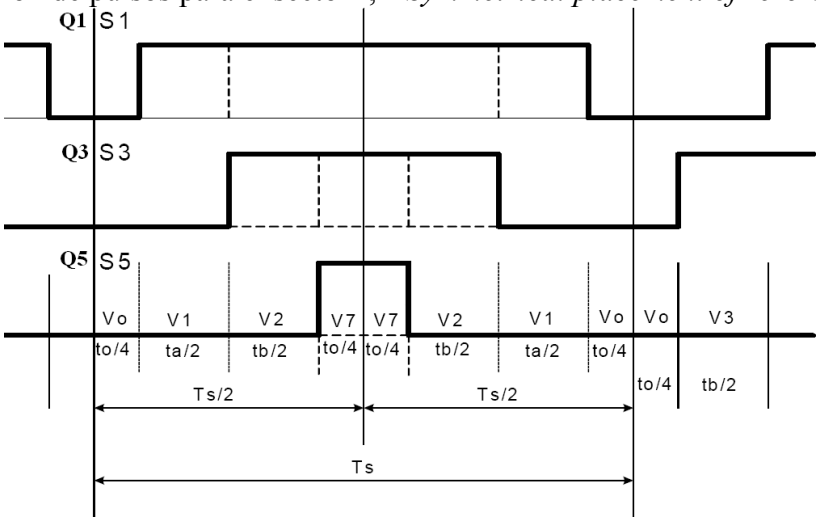


Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

Cuanto más puntos se ubiquen mejor será la tensión de salida, con un menor contenido armónico, pero se debe tener en cuenta que el inversor para cada punto debe generar un vector haciendo una serie de combinaciones con sus llaves. Esto trae consigo una cantidad muy elevada de conmutaciones para el inversor y en consecuencia deben calcularse las pérdidas por conmutación de los transistores de potencia, y evaluar si realmente se obtiene un rendimiento total que supere a un funcionamiento con menor cantidad de puntos que si bien dará un contenido armónico mayor que el anterior, tendrá menos pérdidas por conmutación. La ubicación de estos vectores y la secuencia de conmutación necesaria para crear los patrones de las señales de disparo para cada sextante, determinará la duración (T_A, T_B, T_0) de cada vector director. Existen varios tipos de secuencia patrones que se pueden implementar, aquí se presenta una secuencia simétrica respecto a la ubicación de los vectores nulos llamada "*Symmetrical placement of zero vectors*" para la generación de los pulsos patrones, como muestra la siguiente figura para el sector I. La simetría de la secuencia con respecto $T_S/2$, se logra tomando la mitad de los tiempos respectivos en cada

mitad de T_S. Puede verse que el vector nulo está ubicado en el origen; medio y final de cada secuencia en forma simétrica, se indican además los vectores directores que intervienen en el sector I para generar los 4 vectores de dicho sector. Debe tenerse presente que el inversor puede entregar solamente un estado de llaves por vez, asumiendo el menor número de conmutaciones de los dispositivos electrónicos que conforman el puente inversor.

Figura 78. Patrón de pulsos para el sector I, "*Symmetrical placement of zero vectors*"



Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

Una operación de mínimas conmutaciones y reducido contenido armónico se logra cambiando el orden de los vectores de los sectores pares, así para el sector II, el vector director V_3 comienza la secuencia en lugar de V_2.

Las figuras de las secuencias para los seis sextantes se pueden deducir de la siguiente tabla.

Tabla 31. Secuencia de los vectores directores para cada sextante

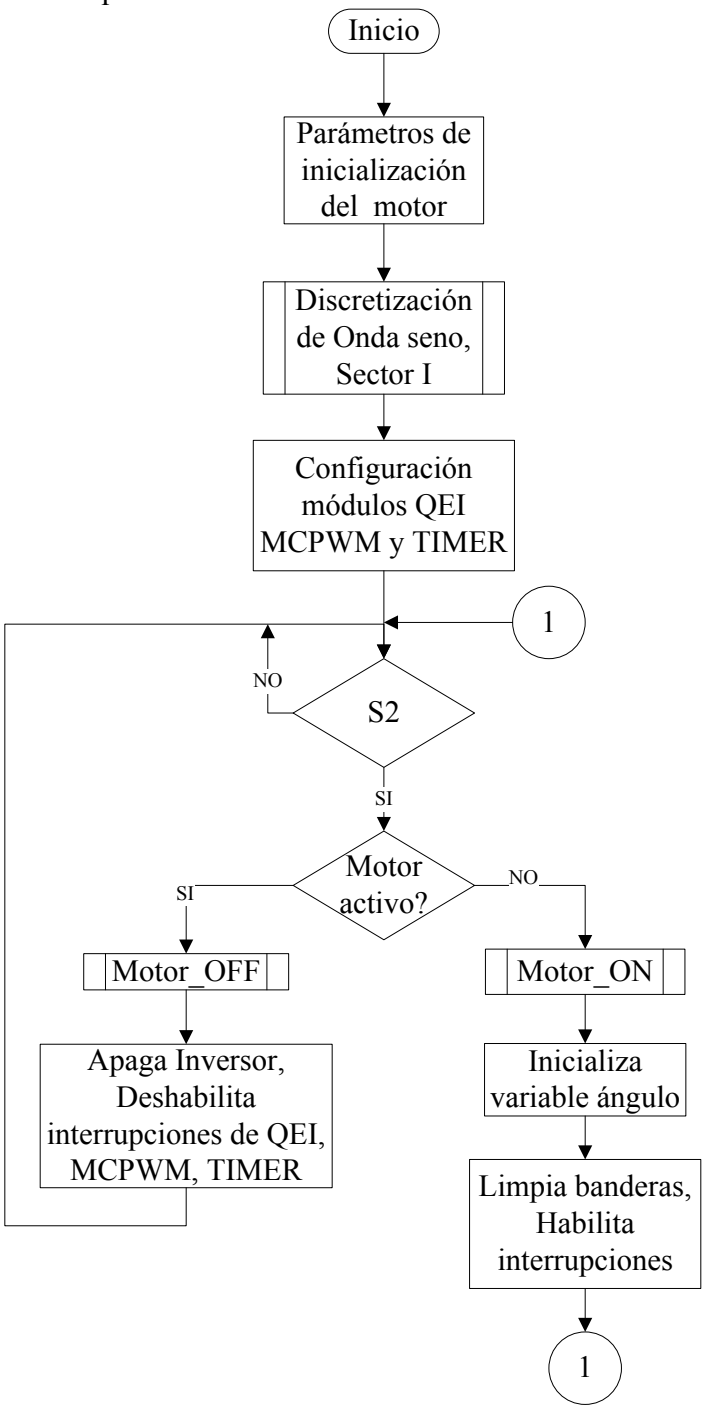
Sector	I	II	III	IV	V	VI
T_A	V_1	V_3	V_3	V_5	V_5	V_1
T_B	V_2	V_2	V_4	V_4	V_6	V_6

Fuente: Microchip Technology Inc. VF Control of 3-Phase Induction Motor Using Space Vector Modulation, 2005.

Como se ve, el tiempo T_A afecta a las tres llaves superiores del inversor y T_B a las tres inferiores, cada vector director interviene en dos sectores contiguos, mientras que V_0 se

ubica siempre en el comienzo y final de cada vector y V_7 en el centro. Precisamente, estos datos de secuencia y discretización se deberán ingresar al programar el dsPIC.

Figura 79. Diagrama de bloques de la rutina SPWM

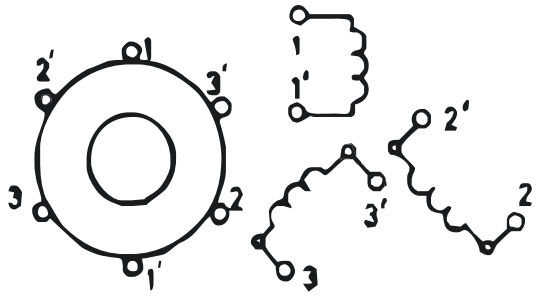


4.3 EL MOTOR DE INDUCCION

Como paso previo a la lectura de las alternativas para controlar la velocidad de un motor de inducción, es conveniente hacer un repaso a los conceptos básicos de los motores asíncronos de jaula de ardilla. Las maquinas asíncronas, como el motor de inducción, se basan en el principio de la acción de un campo magnético giratorio sobre un arrollamiento en cortocircuito.

4.3.1 Generalidades de los motores trifásicos. Son motores que se caracterizan porque son mecánicamente sencillos de construir, lo cual los hace muy robustos y sencillos, apenas requieren mantenimiento, son baratos y, en el caso de motores trifásicos, no necesitan y no se ven sometidos a vibraciones por efecto de la transformación de energía eléctrica en mecánica, ya que la potencia instantánea absorbida por una carga trifásica es constate e igual a la potencia activa.

Figura 80. Motor trifásico de inducción



Fuente: Motores trifásicos de inducción: generalidades. Disponible en: URL: http://www-app.etsit.upm.es/departamentos/teat/asignaturas/lab-ingel/motor%20asincrono%20trifasico.pdf

Estas son las principales ventajas que hacen que sea ampliamente utilizado en la industria. Como inconvenientes, se puede mencionar que son motores que tienen bajos pares de arranque, que presentan una zona inestable de funcionamiento y que el control de velocidad en amplios rangos es complejo.

4.3.2 Control escalar de la máquina de inducción. A este método de control se le conoce el método de flujo constante. Del ANEXO J, se obtiene la siguiente ecuación.

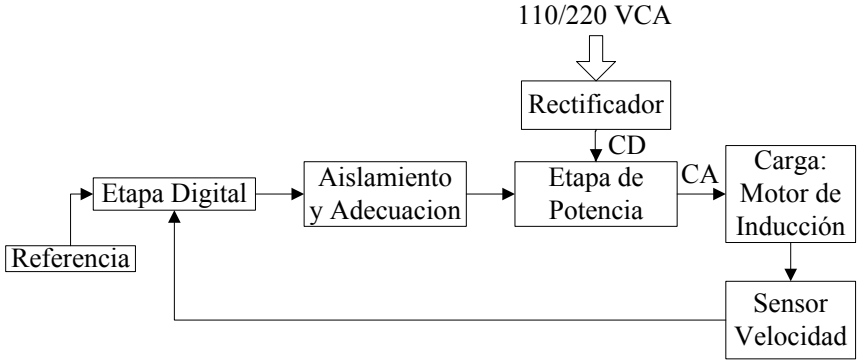
$$\phi = \frac{V}{2\pi N f} \qquad (72)$$

La relación anterior muestra que la tensión aplicada al motor debe variar en forma proporcional a la frecuencia para mantener el flujo constante.

4.4 IMPLEMENTACION VARIADOR DE FRECUENCIA (PROTOTIPO EXPERIMENTAL)

EL prototipo experimental se compone de los siguientes circuitos: digital, aislamiento y adecuación de señal, circuito de potencia, circuitos sensores de sobre-corriente, sobre-voltaje y encoder para medir las R.P.M. (revoluciones por minuto) del motor de inducción trifásico. La siguiente figura ilustra los bloques del circuito experimental. Como puede observarse es un sistema de lazo cerrado. En las siguientes secciones se explicara el objetivo, las características y desarrollo de los bloques. Durante la explicación de los circuitos se harán referencias a segmentos del diagrama para localizar los componentes del circuito descrita.

Figura 81. Diagrama de bloques del circuito experimental del variador de frecuencia



4.4.1 Determinación de los elementos a emplear. Para el diseño se parte de la necesidad de construir un inversor trifásico que entregue la potencia nominal a la carga. La carga es un motor de inducción trifásico de jaula de ardilla de $1/2$ HP con un voltaje nominal 220/440VCA y corriente 2/1A respectivamente. Marca The Louis Allis Co, con una velocidad nominal de 1740 RPM a una frecuencia de voltaje de 60 Hz.

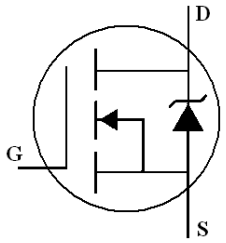
4.4.1.1 Puente rectificador. La señal de CA de red domestica se convierte a CD con el uso de un puente rectificador no controlado de cuatro diodos.

Considerando la corriente nominal del motor igual a 2A, los diodos deben soportar de 3 a 6 veces este valor al momento de arranque, usando así un rectificador de 10A y capaz de soportar un voltaje de pico inverso igual $220VCA * \sqrt{2} = 311V$.

Un capacitor de valor 1880 μF filtra residuos de la componente de CA a la salida del puente rectificador.

4.4.1.2 Mosfet. Es un interruptor controlado por voltaje que conduce en un solo sentido. Para una carga resistiva este tipo de interruptor no presenta ningún problema, sin embargo el motor es una carga inductiva que almacena energía de la fuente mientras el mosfet conduce, y la libera cuando el mosfet termina de conducir, para que circule esta corriente inversa se utiliza un diodo en antiparalelo entre el D (Drenaje) y la S (Fuente). El arreglo del mosfet y el diodo en antiparalelo permite la conducción bidireccional de la corriente y así evitar el daño de los seis transistores que conforman el Inversor.

Figura 82. Símbolo del transistor mosfet NPN.



Fuente: International Rectifier. IRFP360 Data Sheet, 2000.

Para la selección del transistor mosfet adecuado se tiene en cuenta principalmente la corriente y el voltaje. Debe soportar al menos 220VCA de red rectificados, es decir 311VCD y una corriente de 3 a 6 veces la corriente nominal de 2A del motor.

Características principales del mosfet IRFP360.

- Voltaje de alimentación $V_{DS} = 400V$
- Resistencia de drenaje a fuente ($R_{DS(ON)} = 0.20\Omega$)
- Corriente de drenaje $I_D = 23A$.

Otras características que se tienen en cuenta son la resistencia $R_{DS(ON)}$ existente entre drenaje y fuente cuando el transistor se encuentra en conducción, y la frecuencia de conmutación. Con el primer parámetro se pueden llegar a presentar perdidas de potencia nada despreciables al momento de ahorrar energía. Por otro lado la frecuencia de conmutación exigida para el inversor no supera los 20KHz de manera que no hay problema con este tipo de transistores que superaran fácilmente dicha frecuencia.

Los diodos de protección contra corrientes inversas son seleccionados según sus características de t_{rr} (tiempo de recuperación inversa) y voltaje soportado entre sus terminales, que para efectos del prototipo se empleo la referencia SF54, con especificaciones técnicas de $t_{rr} = 35ns$, de voltaje 200V y corriente 5A.

Adicionalmente se configuraron capacitores de desacople en paralelo a los mosfet's de valor 1µF/250V, como elementos de protección contra cambios peligrosos de voltaje dv/dt y mitigar los posibles daños debida a la conmutación de los propios Mosfet's.

4.4.1.3 Driver. El driver suministra la señal de voltaje adecuada que en este caso es de valor 15VCD para excitar la G (compuerta) de los dispositivos conmutadores mosfet's y de esta manera garantizar que la señal de control active o desactive de forma adecuada los transistores.

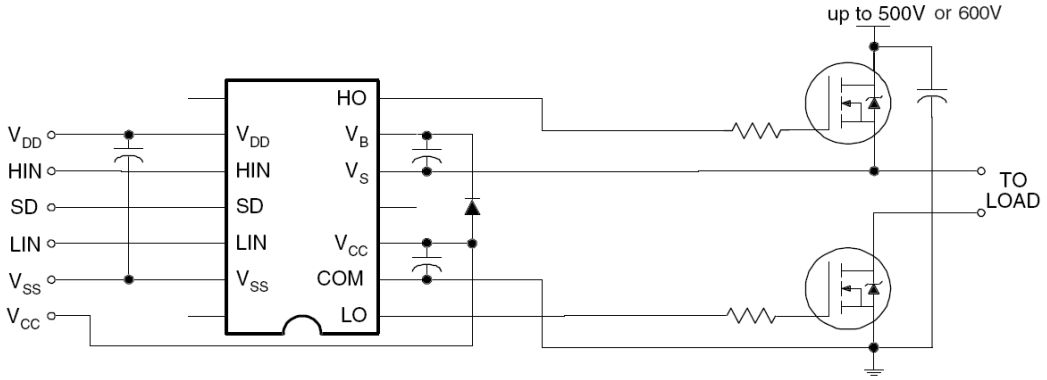
Para tomar la decisión del tipo de driver a utilizar se considera en primera instancia el tiempo de encendido y apagado, es decir debe ser lo más rápido posible con el fin de evitar retardos en la propagación de la señal de control.

El IR2110 es un driver doble, capaz de manejar dos transistores con una misma fuente de alimentación mediante la utilización de un circuito de carga de capacitor o "*bootstrap*" como se muestra en la siguiente figura.

Características principales del driver IR2110:

- Canal flotante diseñado para operación "*bootstrap*", máximo a 500V.
- Tolerancia a transitorios de voltaje dv/dt de voltaje negativo.
- Voltaje de salida de 10 a 20V
- Monitoreo de bajo voltaje para ambos canales
- Corriente de salida +2A/-2A
- Tiempo de encendido/apagado 120 y 94 ns respectivamente.
- Retardo entre canales de 10 ns

Figura 83. Conexión típica del driver IR2110



Fuente: International Rectifier. IR2110 Data Sheet, 2004.

Simbología:

- VDD voltaje de lógica, VSS tierra lógica
- HIN entrada lógica para el manejo de la compuerta del lado superior (HO), en fase
- SD entrada lógica para apagado general del IR2110
- LIN entrada lógica para el manejo de la compuerta del lado inferior (LO), en fase
- VB voltaje flotante del lado superior
- HO salida de manejo de la compuerta del lado superior
- VS retorno o tierra del voltaje flotante del lado superior
- VCC voltaje del lado inferior
- LO salida de manejo de la compuerta del lado inferior
- COM retorno o tierra del lado inferior

• **Componentes "*bootstrap*".** El circuito de "*bootstrap*" se calcula dependiendo de la frecuencia de trabajo y la carga de la compuerta del mosfet, basado en las especificaciones de diseño propuestas por el fabricante IRF[20], de donde se toma la siguiente ecuación

[20] An978 Application Note, International Rectifier, 2007, Disponible en Internet: URL: http://www.irf.com/technical-info/appnotes/an-978.pdf, p6.

$$C \geq \frac{2\left[2Q_g + \frac{I_{qbs\,(max)}}{f} + Q_{ls} + \frac{I_{Cbs\,(leak)}}{f}\right]}{V_{cc} - V_f - V_{LS} - V_{Min}} \tag{73}$$

Donde:

- Q_g es la carga de la compuerta del Mosfet en Coulombs
- $I_{qbs\,(max)}$ es la corriente de reposo en el lado superior del driver en Amperes
- Q_{ls} es el nivel de carga requerida por ciclo en Coulombs
- $I_{Cbs\,(leak)}$ es la corriente de fuga del capacitor de "*bootstrap*" en Amperes
- f es la frecuencia de operación en Hertz
- V_{CC} es el voltaje de alimentación del driver en Volts
- V_f es el voltaje de caída a través del diodo en Volts
- V_{LS} es el voltaje de caída a través del transistor inferior en Volts
- V_{min} es el voltaje mínimo entre los terminales V_B y V_S.

Teniendo en cuenta que la frecuencia de conmutación utilizada en el desarrollo del prototipo del inversor no supera los 10KHz, y sustituyendo los valores suministrados por el fabricante del driver y de los mosfet's, para cada una de las variables especificas en la anterior ecuación, se obtiene el valor del capacitor de "*bootstrap*".

$$C \geq \frac{2\left[2(210nC) + \frac{230\mu A}{10KHz} + 5nC + \frac{5\mu A}{10KHz}\right]}{15V - 0.95V - 4V - 9.7V} = 2.56\mu F \tag{74}$$

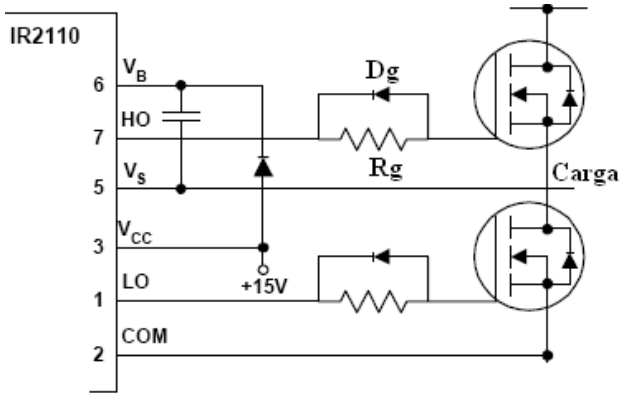
Entonces el valor del capacitor debe ser mayor a $2.56\mu F$, sin embargo se recomienda multiplicar este resultado por un factor de 15 veces, por lo que el valor optimo seria $38\mu F$, el valor comercial seleccionado es de $33\mu F/50V$. En cuanto al diodo de "*bootstrap*", según lo establecido en el manual técnico DT-98-2[21], este debe tener un t_{rr} (tiempo de recuperación inversa) menor a 100ns, soportar el voltaje de alimentación del inversor y la corriente de carga del circuito, la cual depende de la carga de compuerta del transistor y la frecuencia de conmutación. De manera que el diodo elegido es el de referencia SF54, el cual soporta hasta 400V entre sus terminales, tiene un t_{rr} de 35ns y soporta una corriente

[21] DT98-2 Design Tip, International Rectifier, 2001, Disponible en internet: URL: http://www.irf.com/technical/info/designtp/dt98-2.pdf, p4.

de 5A; con lo que se cumplen con todos los requerimientos para el correcto funcionamiento del driver.

- **Red resistencia-diodo.** El propósito de esta red (Rg-Dg) es proporcionar un retardo adicional al encendido del transistor, sin afectar el apagado del mismo. También se usa para reducir el inevitable pico de voltaje durante el tiempo de recuperación inversa en el transistor. Esto tiene un impacto en pérdidas de potencia, como también dv/dt y EMI (interferencia electromagnética).

Figura 84. Red resistencia-diodo



Fuente: International Rectifier. IR2110 Data Sheet, 2004.

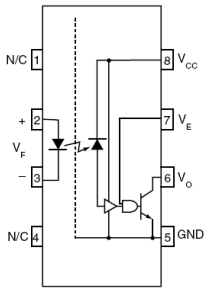
De acuerdo a lo indicado en manual técnico DT-98-2[22] el valor de la resistencia Rg adecuado se encuentra alrededor de 27Ω. Y el diodo a usar con un tiempo de recuperación inverso menor a 100ns para el cual se opto por la referencia 1N4148.

4.4.1.4 Opto-acoplador. El opto-acoplador aísla la referencia (GND) del μC de la referencia (GND) correspondiente al driver IR2110 y la etapa de potencia. Al igual que el IR2110 un parámetro determinante es el retardo de la propagación de la señal para el cual se opto por usar el 6N137.

El opto-acoplador es compatible con dispositivos TTL/CMOS, lo que facilita el acoplamiento directo con los pines del μC. De las características eléctricas del μC se obtiene que la corriente en un bit es $I_{in} = 25mA$; es decir, la máxima corriente por bit que el μC puede absorber es 25mA; comparada con la máxima corriente $I_{in} = 2mA$ que absorbe la entrada del opto-acoplador se concluye que el μC puede consumir esta corriente sin exceder los valores máximos de corriente para los puertos PWM del μC.

[22] An978 Application Note, Op. cit., p13.

Figura 85. Símbolo del opto-acoplador 6N137



Fuente: Fairchild, 6N137 Data Sheet, 2001.

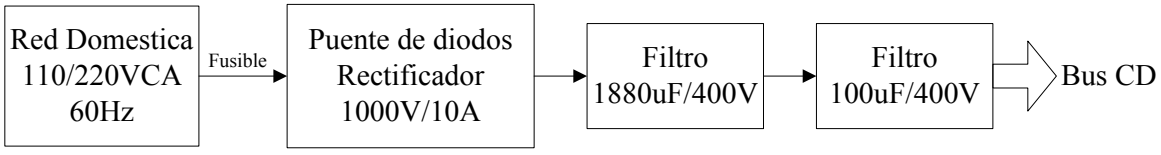
Características principales del opto-acoplador 6N137

- Alta velocidad 10 MBits/s
- CMR de 10 kV/μs
- Voltaje máximo de emisor 5.5V
- Voltaje máximo de detector 7.0 V (máximo un minuto)

4.4.2 Construcción de etapas del inversor.

4.4.2.1 Modulo de potencia. El circuito de potencia puede analizarse en dos partes, la primera correspondiente a la etapa CA-CD del inversor en la que se obtiene la fuente V_{CD}, a lo que se denomina bus de CD, y la segunda es la del puente inversor como tal.

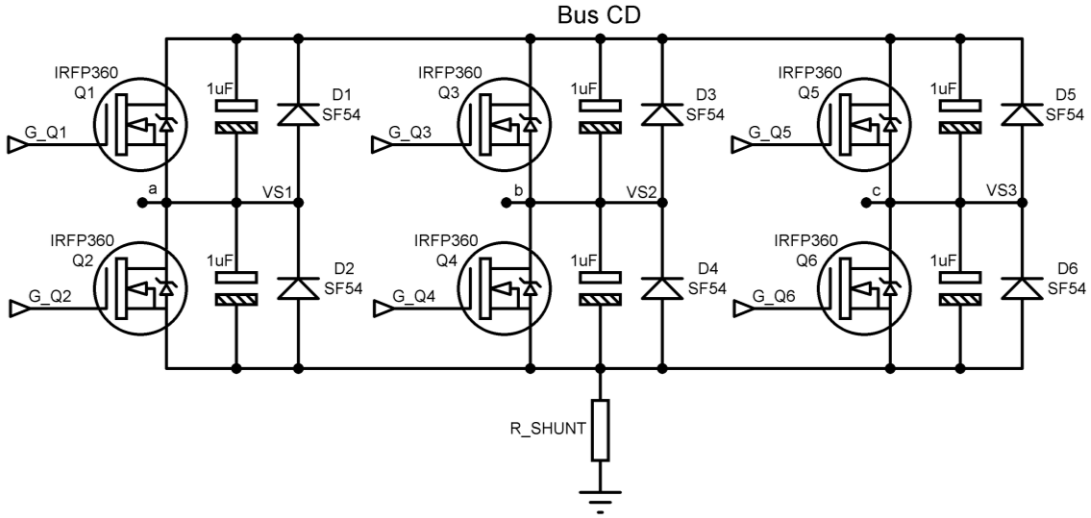
Figura 86. Diagrama de bloques de fuente de potencia



El Bus CD alimenta el inversor configurado como se muestra en la figura a continuación. Donde la red "*snubber*" que se aprecia en paralelo con cada transistor formada por un diodo y un capacitor, cumple la función de atenuar los dv/dt peligrosos y proteger a los transistores de corrientes inversas respectivamente por los elementos que la conforman. Con un diodo rápido SF54 y un capacitor de 1μF/250V se logra dicha red.

También es importante detallar que la resistencia "*R_Shunt*" simplifica y economiza la función de medir la corriente de retorno que está consumiendo el inversor y así monitorear constante esta variable.

Figura 87. Esquema del puente inversor conformado por 6 transistores la red "*snubber*" y la resistencia "*R_shunt*"



4.4.2.2 Modulo de aislamiento y adecuación de señal. Este modulo aísla las señales PWM provenientes de la etapa digital y que controlan la etapa de potencia, proporcionando protección contra cortocircuito; así como una mayor inmunidad al ruido que pudiera afectar el normal funcionamiento del µC. Del mismo modo se adecuan las señales PWM a fin de cumplir los parámetros de funcionamiento de los transistores del Inversor Trifásico, garantizando así su correcto control de conducción.

En la siguiente figura se aprecia el esquema de conexión del opto-acoplador 6N137, donde la señal PWM1H y PWM1L son las señales que controlan las compuertas del transistor superior e inferior respectivamente que corresponde a una rama o fase del puente inversor y que consecuentemente aísla el 6N137 y ahora las señales a la salida de este dispositivo se llaman HIN y LIN.

Finalmente el paso siguiente en este modulo es adecuar la señal el cual se logra con el driver IR2110, proporcionando el nivel de voltaje y la impedancia necesaria para el normal funcionamiento en el control de las compuertas de los transistores es y ahora las señales de control se llaman HO y LO.

Figura 88. Esquema de conexión del opto-acoplador 6N137

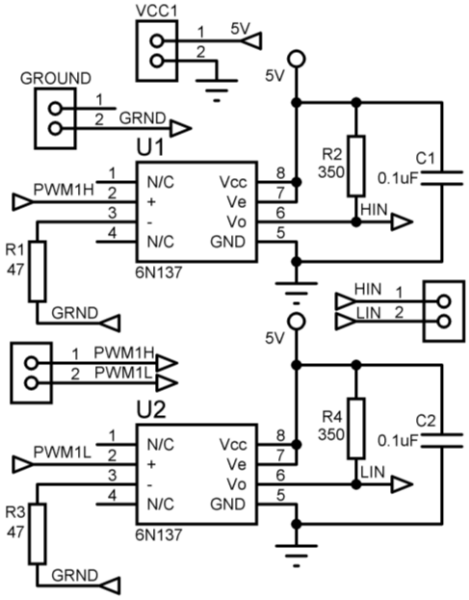
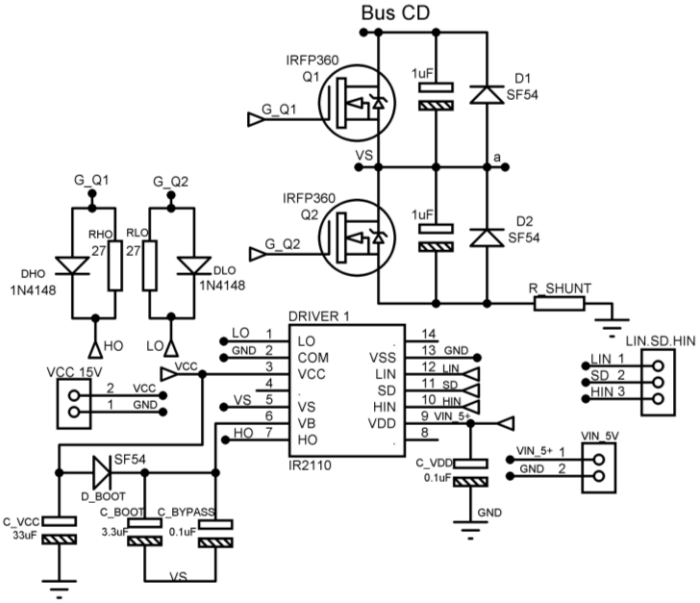
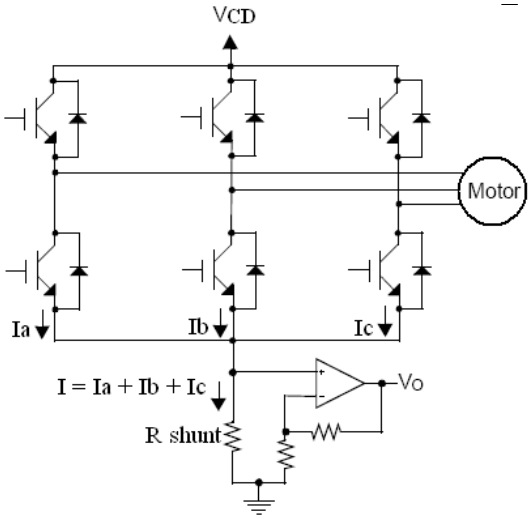


Figura 89. Esquema de conexión del driver IR2110



4.4.3 Sensor de sobrecorriente y sobrevoltaje. La necesidad de sensar la corriente de retorno del inversor es con el fin de monitorear a cada instante una posible sobrecorriente por cortocircuito o exigencia del motor que lleve a un pico de corriente no permitido dentro de las especificaciones de los elementos que conforman el inversor para el normal funcionamiento.

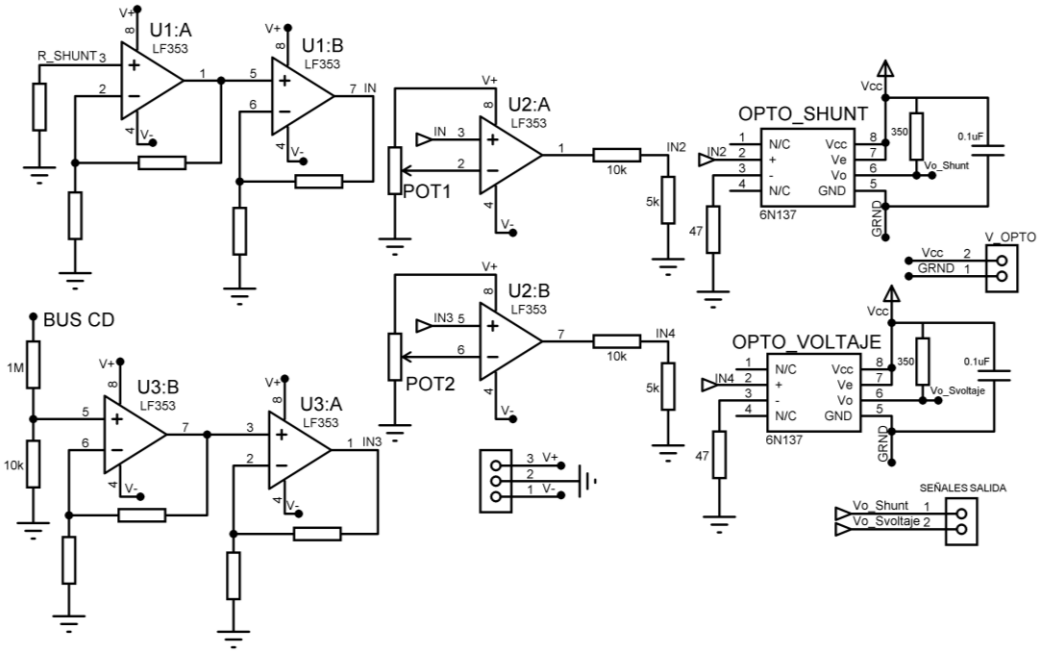
Figura 90. Configuración de conexión del sensor de corriente "*R_Shunt*"



Fuente: Microchip Technology Inc. Motor Control Sensor Feedback Circuits, 2003.

Ahora de la figura 91, el voltaje obtenido en "*R_shunt*" es amplificado generándose el voltaje Vo que antes de llevarse directamente al modulo ADC del µC es comparado con un umbral predefinido y calibrado por POT1 además de aislar la señal generada por el comparador por medio de un opto-acoplador rápido 6N137. De la misma manera opera el sensor de voltaje con la diferencia que la entrada es el resultado de una divisor de voltaje aplicada el Bus de CD.

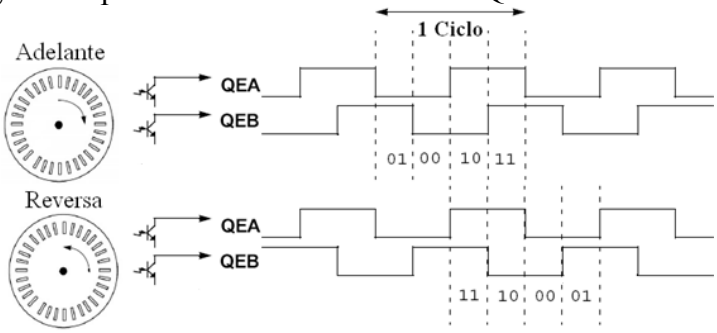
Figura 91. Circuito de amplificación del voltaje "*R_Shunt*" (superior) y circuito de amplificación del divisor de voltaje de bus CD (inferior)



4.4.4 Sensor de velocidad. El sensor de velocidad seleccionado es el encoder Q9898 que genera dos señales en cuadratura. La lectura de los pulsos son 504 por vuelta, sin embargo el modulo QEI (interfaz de enconder en cuadratura) del µC se lo configuro en modo 4x, para así multiplicar por un factor de cuatro la señal de los pulsos y obtener un total de 2016 por vuelta.

La forma de las dos señales fase A y fase B son como se muestra en la siguiente figura, donde es posible detectar el sentido de giro hacia adelante o hacia atrás del motor, de acuerdo a la forma en que lleguen las señales al modulo QEI del µC.

Figura 92. Fase a y fase b para el modulo de cuadratura QEI

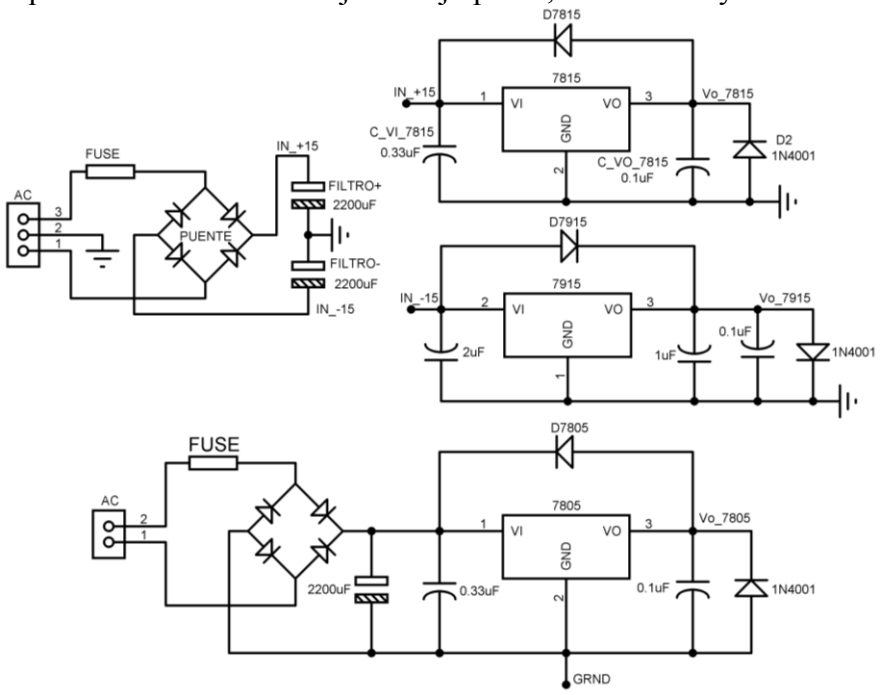


Fuente: Microchip Technology Inc. Quadrature Enconder Interface (QEI), 2004.

4.4.5 Etapa digital. La etapa digital consiste del μC dsPIC 30F4012 que almacena el programa SPWM. La elección de un μC para esta aplicación y no de un sistema digital destinado para tal propósito se debió al conocimiento previo del μC, fácil programación, dar aplicación al μC así como el costo.

4.4.6 Fuente de bajo poder.

Figura 93. Esquema de fuente de voltaje de bajo poder, +15V/-15V y +5V



De la figura anterior la fuente dual de +15/-15V tiene la función de alimentar el pin de V_{CC} del driver IR2110, y también a los operaciones LF353.

Por otro lado la fuente de +5V alimenta lo correspondiente a la etapa digital y a los dos opto-acopladores que aíslan las señales de alarma de sobrevoltaje y sobrecorriente en su pin V_{CC}.

4.5 PRUEBAS Y RESULTADOS

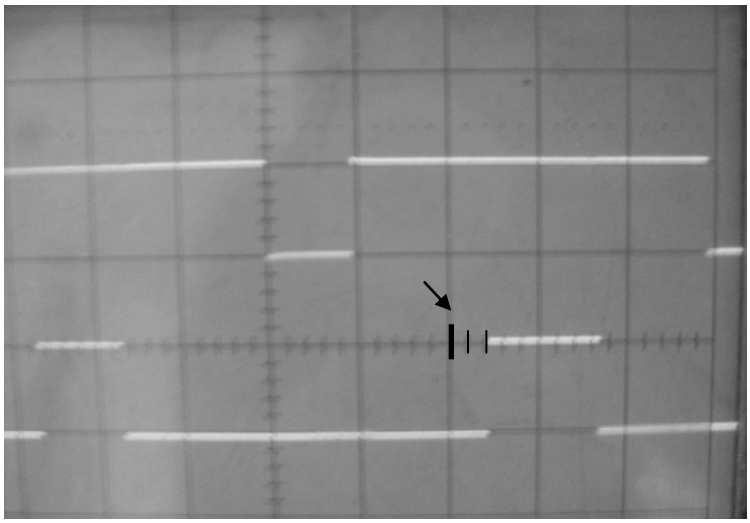
Despues de la implementacion de la modulacion SPWM y del sistema en su totalidad; se muestran las pruebas realizadas al sistema, asi como los resultados arrojados por las

mismas. Se realizaron pruebas con carga resistiva conectada en estrella sin filtro a la salida del inversor.

Los resultados obtenidos se observaron utilizando un osciloscopio SCOPE.

Cabe mencionar que antes de realizar la prueba, se verificaron las señales de control generadas por el μC para ca uno de los transistores; asi como el tiempo muerto entre las señales de control de un transistor superior e inferior de una misma fase del inversor. De esta manera se verificaron que las señales de control entre fases del inversor estuvieran desfasadas 120 grados una con respecto de las otras; ademas, que las señales de control entre transistores de una misma fase se encontraran complementadas y existiera entre estas el tiempo muerto establecido. Tal y como muestra la siguiente figura donde el valor del tiempo muerto es equivalente a 2μs ademas de su respectivo complemento del transistor superior e inferior de la misma fase.

Figura 94. Tiempo muerto entre señales de transistores de una misma fase. 5μs/div



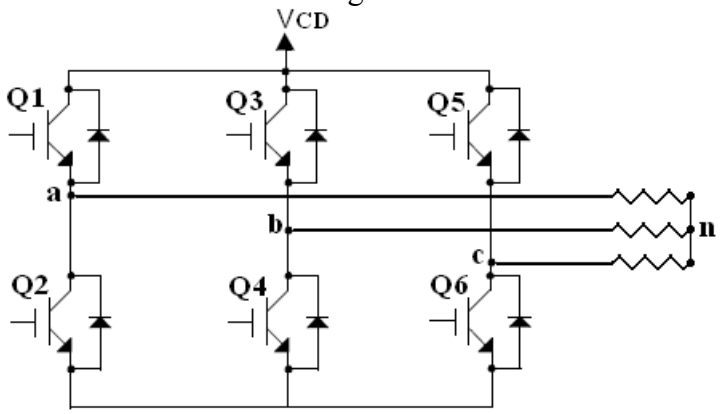
Fuente: Investigacion

4.5.1 Pruebas con carga resistiva en estrella sin filtro de salida. Las pruebas con carga resistiva conectada en estrella se realizaron utilizando el diagrama de conexiones mostrado en la siguiente figura, bajo las siguientes condiciones de prueba:

- Voltaje de entrada alterna = 110 V
- Voltaje de Bus DC = 155 V
- Frecuencia de conmutacion = 3600 KHz

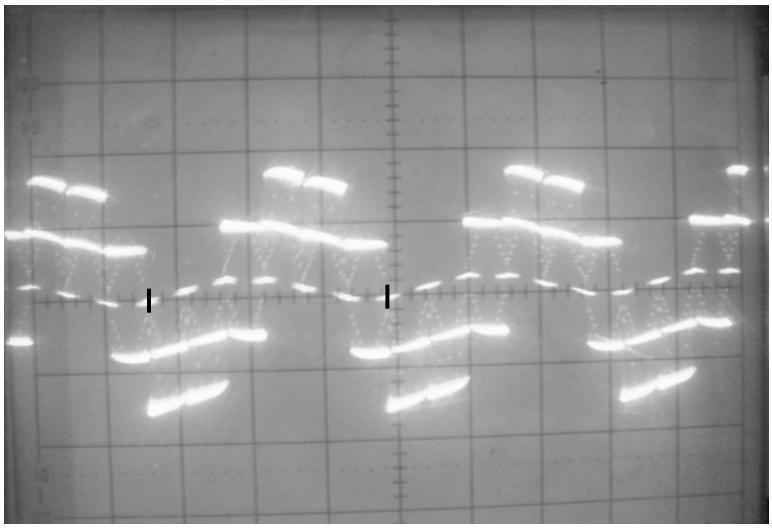
- Tiempo muerto = 2 μs
- Carga Resistiva = 100 Ω
- Indice de modulacion maxima = $\sqrt{3}/2$

Figura 95. Configuracion en estrella de la carga resistiva



Fuente: Investigacion

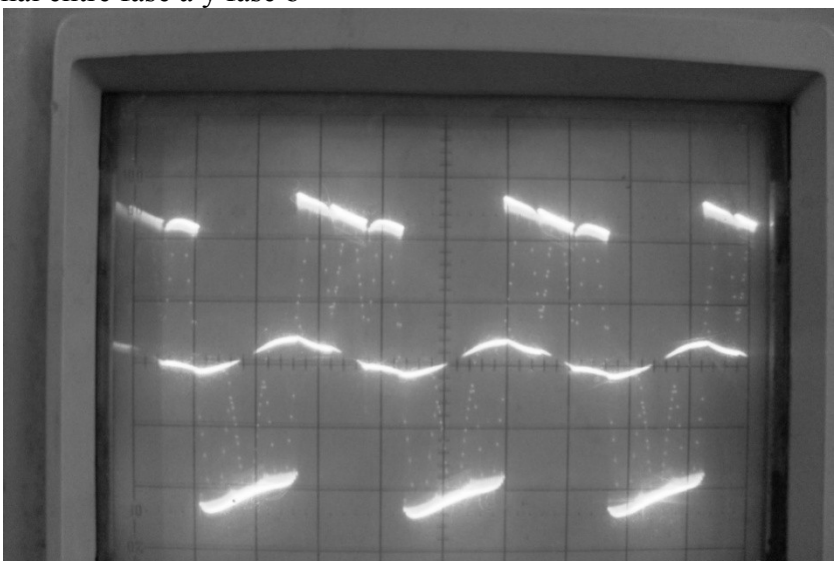
Figura 96. Fase a. 5 ms/div



La señal sinusoidal obtenida de frecuencia igual a 60Hz se puede constatar entre las líneas demarcadas en la anterior figura.

De la misma manera se obtuvo la señal entre la fase a y la fase b como se aprecia en la siguiente figura.

Figura 97. Señal entre fase a y fase b



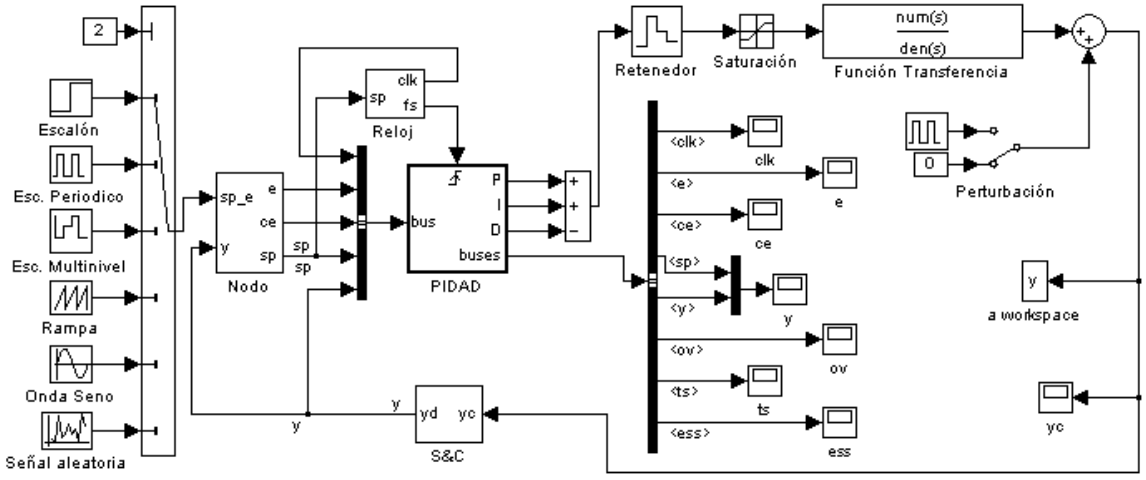
5. SIMULACIONES, EMULACIONES, APLICACIÓN Y ANALISIS DE RESULTADOS

5.1 SIMULACIONES Y ANALISIS DE RESULTADOS

Por medio de la simulación se representa el controlador PID adaptivo con su mecanismo difuso, el objetivo de construir un modelo de software del controlador es determinar la eficiencia de los algoritmos, ya que la simulación facilita la experimentación por medio del computador, estas se realizan en Simulink de Matlab.

5.1.1 Construcción del modelo software en Simulink. Este software cuenta con los "*toolboxes*" (caja de herramientas) adecuados para simular sistemas de control y difusos, permite una construcción sistemática y que permite avanzar por escalones en la investigación por medio de la integración de bloques.

Figura 98. Modelo software del controlador PID adaptivo difuso (PIDAD)



En el diagrama de bloques general del modelo de simulación del PIDAD, está compuesto por los siguientes subbloques:

- Setpoint y multiplexor
- Nodo

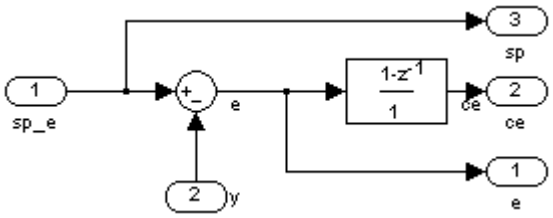
- Reloj
- Muestreador y cuantificador (S&C)
- Retenedor y saturación
- Función de transferencia
- Perturbación
- Demultiplexor y variables
- Subbloque PIDAD

Para comprender el funcionamiento del PIDAD simulado se da una definición de cada bloque que lo compone.

5.1.1.1 Setpoint y multiplexor. Es el conjunto de señales de referencia, escalón unitario, escalón unitario periódico, escalón multinivel, rampa periódica, onda senoidal y la señal aleatoria, esta se eligen según la constante en la parte superior del multiplexor.

5.1.1.2 Subbloque nodo. En este se calcula la señal de error (e) dependiendo del setpoint (y_{sp}) y la salida (y), también se calcula el cambio del error (ce) por medio del bloque derivativo que se muestra en la siguiente figura, los bloques rectangulares con esquinas redondeadas indican las entradas y salidas del subbloque.

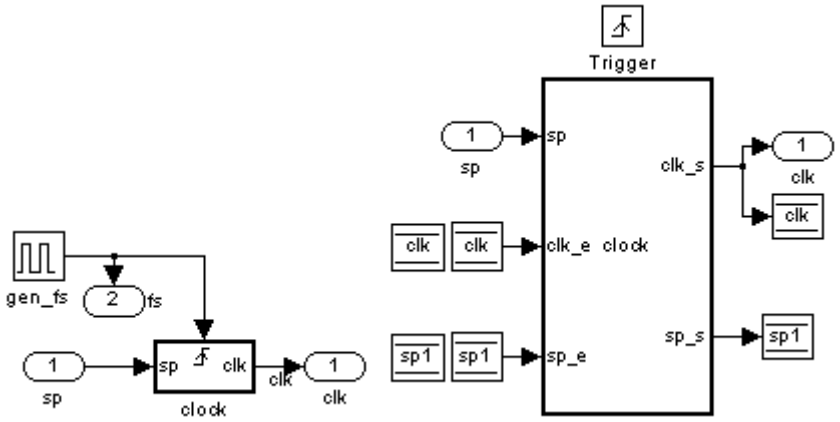
Figura 99. Subbloque nodo



5.1.1.3 Reloj. Este se encarga de medir el tiempo real, se debe reiniciar ante el cambio de nivel de un escalón, para esto tiene una tolerancia de 1%. En la siguiente figura se pueden observar dos partes (a) y (b), la parte (b) representa un bloque de subfunción que esta contenido dentro del bloque "*clock*" de la parte (a); el generador "*gen_fs*" genera una onda con un periodo de 0.002 segundos haciendo que el subbloque "*clock*" se ejecute de forma

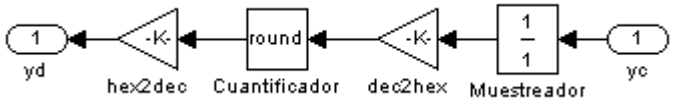
periódica y en efecto la subfunción. Los cuadros que encierran una variable entre dos líneas horizontales son memorias, estas son necesarias para manejar una variable de forma global, cada memoria se declara con tres bloques, un bloque de estado inicial que no tiene conexión, y dos bloques, de lectura y escritura de la memoria. Para detalles del código contenido en el subbloque reloj referirse al numeral (a) del ANEXO H.

Figura 100. Subbloque reloj



5.1.1.4 Muestreador y cuantificador (S&C). Este simula un conversor analógico a digital de 10bits, y un muestreador con tiempo de 0.002s para la señal de salida "*y*", esto se logra muestreando la señal "*y*", multiplicándola por la constante "dec2hex = 1023/5" para convertir de voltios a decimal según CAD de 10 bits, se redondea el resultado y se vuelve a convertir a voltios multiplicando por la constante "hex2dec = 5/1023".

Figura 101. Subbloque del Muestreador y cuantificador (S&C)



5.1.1.5 Retenedor y saturación. Simula un retenedor de orden cero, que convierte la señal de digital a analógica para aplicar a la planta de naturaleza analógica, el modelo de saturación simula las limitaciones del actuador en este caso maneja un rango de 5 voltios, 3v máximo y -2v mínimo.

5.1.1.6 Función de transferencia. Esta indica la ecuación de la función de transferencia de la planta que se trabaje, esta se expresa como un polinomio en el dominio de la variable de

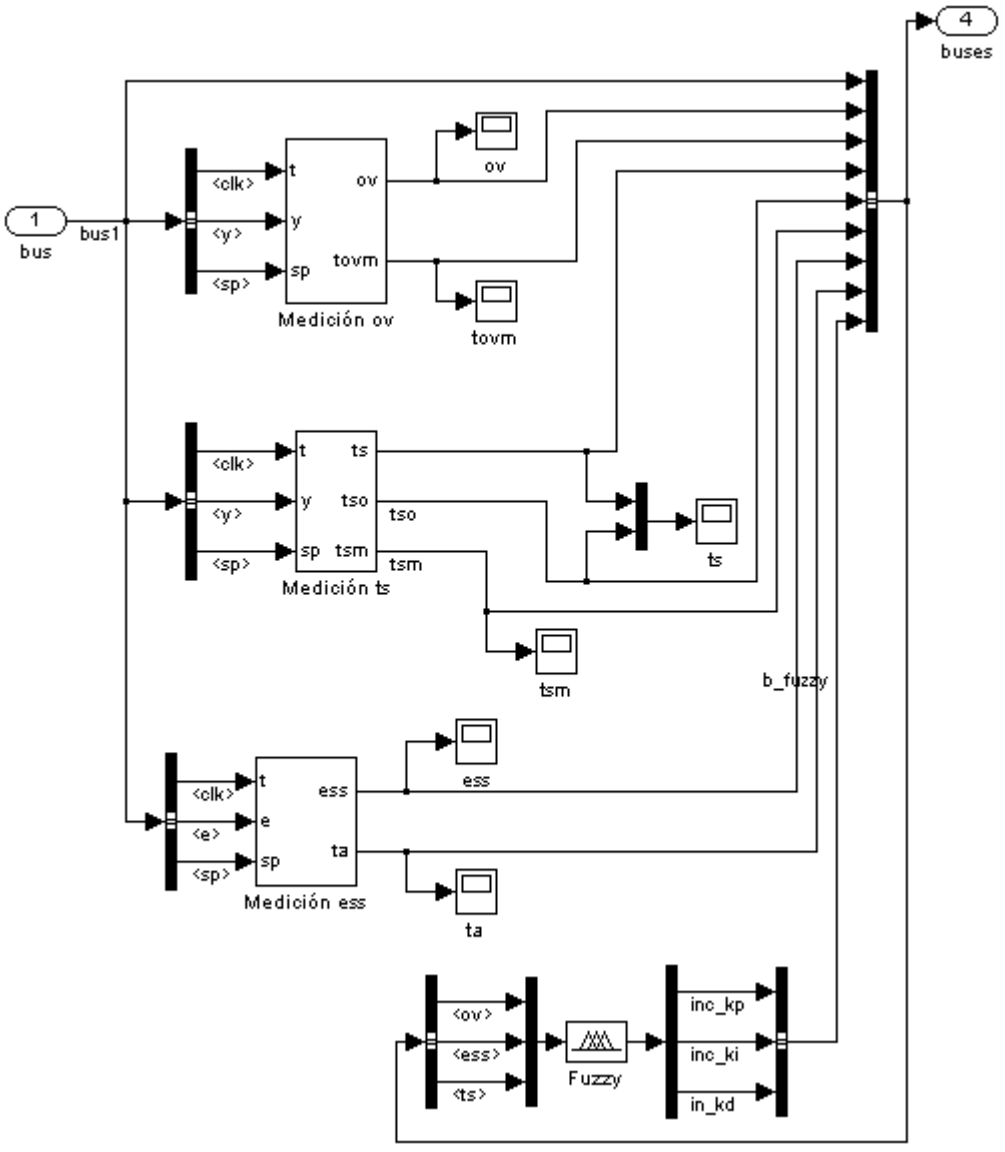
Laplace (s) numerador y denominador, esta se debe crear previamente antes de iniciar la simulación ya que se lee automáticamente del área de trabajo (*workspace*) de Matlab.

5.1.1.7 Perturbación. Este es un modelo que sirve para simular una perturbación al sistema de forma que se suma una función impulso o escalón a la señal de salida (y), se activa con un interruptor manual.

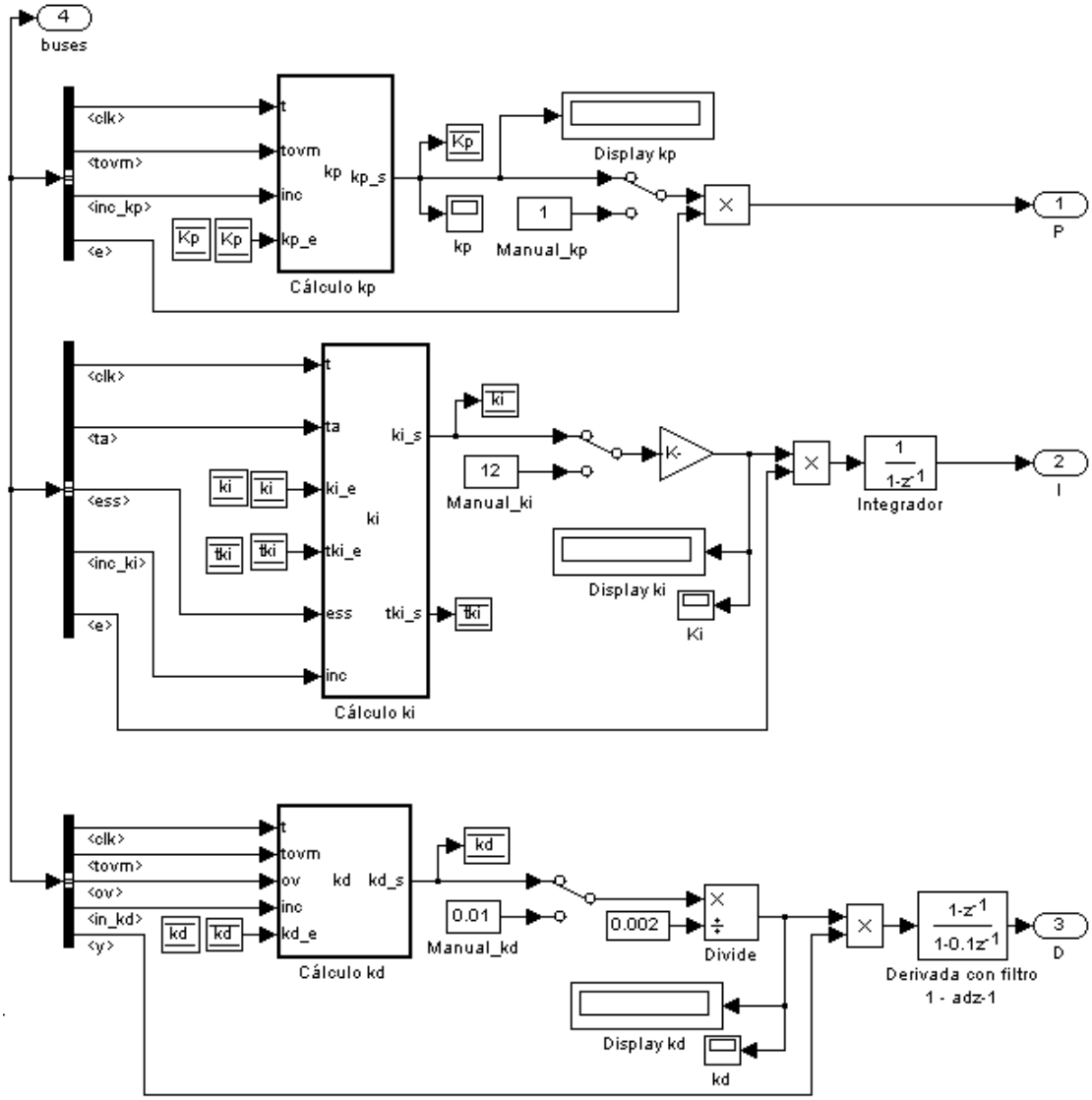
5.1.1.8 Demultiplexor y variables. Es una forma de agrupar las variables importantes del sistema para visualizarlas fácilmente.

5.1.1.9 Subfunción PIDAD. Representa el bloque del controlador PID y el mecanismo adaptivo basado en lógica difusa, el contenido de este subbloque se muestra en la siguiente figura.

Figura 102. Diagrama de bloques de la subfunción PIDAD



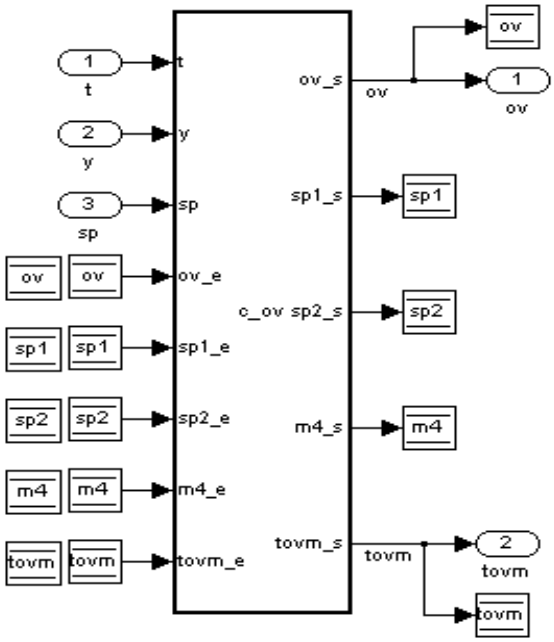
(a)



(b)

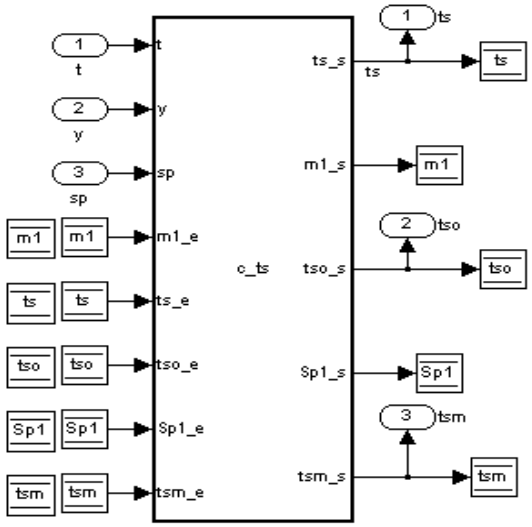
5.1.1.10 Subbloque medición del sobrepaso (*ov*). Se implementa el diagrama de flujo de cálculo de "*ov*" de la figura 34 en lenguaje de Matlab, como se indica en el siguiente código, el cual está contenido en el subbloque de la siguiente figura. Para detalles del código contenido en el subbloque de medición de sobrepaso "*ov*" referirse al numeral (b) del ANEXO I.

Figura 103. Subbloque de medición de "ov"



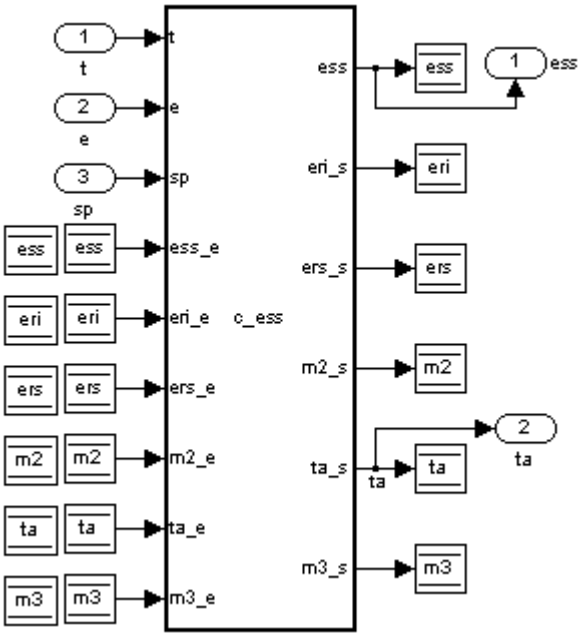
- **Subbloque de medición t_s.** Se implementa el diagrama de flujo de cálculo de "t_s" de la figura 31 en lenguaje de Matlab, como se indica en el siguiente código, el cual está contenido en el subbloque de la siguiente figura. Para detalles del código contenido en el subbloque de medición de "t_s" referirse al numeral (c) del ANEXO H.

Figura 104. Subbloque de medición de "t_s"



- **Subbloque de medición e_{ss}.** Se implementa el diagrama de flujo de cálculo de "e_{ss}" de la figura 33 en lenguaje de Matlab, el cual está contenido en el subbloque de la siguiente figura. Para detalles del código contenido en el subbloque de medición de "e_{ss}" referirse al numeral (d) del ANEXO H.

Figura 105. Subbloque de medición de "e_{ss}"



- **Cálculo k_p, k_i, y k_d.** Los bloques de cálculo de k_p, k_i, y k_d que se muestran en la figura 102(b) implementan las ecuaciones 49, 50 y 51 en lenguaje Matlab.

- **Algoritmo de control PID.** El controlador PID se implementa de forma muy sencilla, el término proporcional se implementa por medio de un multiplicador y la constante, el término derivativo se implementa con el bloque de derivación y su filtro y el término integral con un bloque integral. El sistema funciona como se explico en los diagramas de bloques del mecanismo adaptivo difuso y el controlador PID. La simulación se debe realizar con la planta sin controlador, con la planta sintonizada según Z&N.

- **Mecanismo adaptivo difuso.** el bloque que se observa con el nombre "*fuzzy*" hace referencia al modelo difuso detallado en la sección 3, este se construye a partir de las reglas difusas y conjuntos con la ayuda del toolbox fuzzy de Matlab. Los dos controladores adaptivos propuestos difieren solamente en el mecanismo adaptivo, MAD1 y MAD2, para fines de simulación el cambio entre estos se hace fácilmente modificando el bloque fuzzy.

5.1.2 Pruebas de simulación del controlador PID con MAD1 (PIDAD1). El mecanismo difuso adaptivo 1 se utiliza como herramienta para sintonizar el controlador PID. El controlador PIDAD1 se aplica a una planta con una función de transferencia típica de un motor, para ejecutar la rutina de sintonía la planta se somete a una función escalón unitario periódica en la cual el MAD1 dependiendo de la respuesta transitoria trata de llevarla hacia un rango de operación adecuado balanceando los parámetros del controlador PID. Es importante mencionar que los parámetros con los que parte el controlador, casi anulan la acción integral y derivativa, los valores son:

$k_p = 1$ $k_i = 0.0001$ $k_d = 0.0001$

Una vez la sintonía ha sido realizada el controlador PID puede trabajar como un controlador PID manual sintonizado al valor de los parámetros obtenidos.

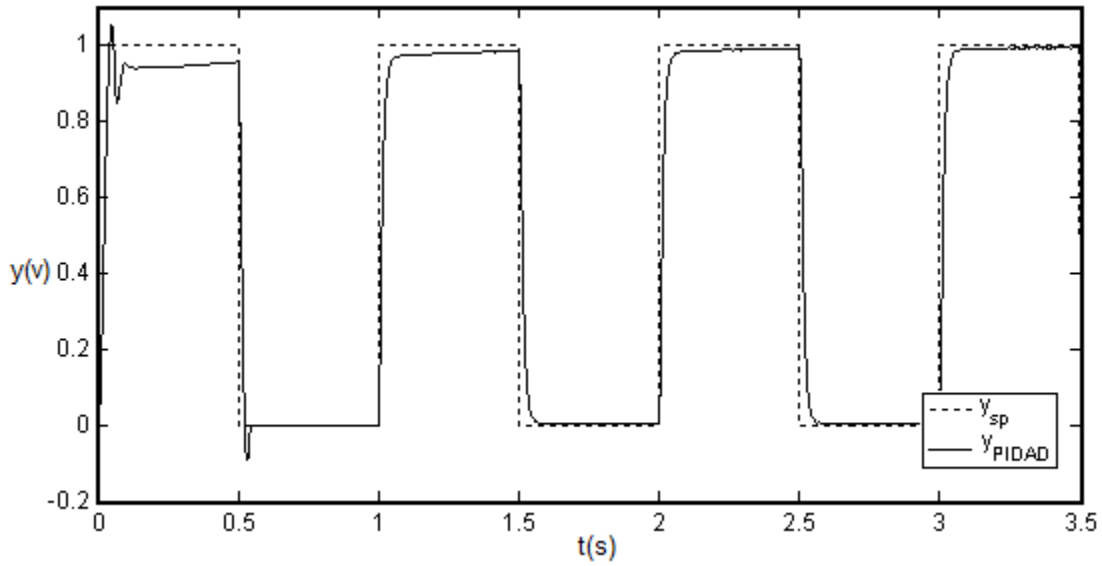
5.1.2.1 PIDAD1 aplicado a la planta 1. La planta 1 representa la función de transferencia $G_1(s)$ de un motor de corriente continua real. El modelo que se utiliza se obtuvo de forma experimental, adquiriendo la señal de entrada y salida aplicada a la planta para posteriormente ser procesadas con ayuda de la función "*Idem*" de Matlab determinando la función de transferencia del motor[23].

$$G_1(s) = \frac{3950}{(s + 29.72)(s + 24.47)} \qquad (75)$$

El escalón periódico unitario es la señal adecuada para realizar la rutina de sintonía del controlador, el sistema trata de llevar la respuesta al escalón a unos límites de trabajo aceptables, reduciendo el error en estado estable y el sobrepaso.

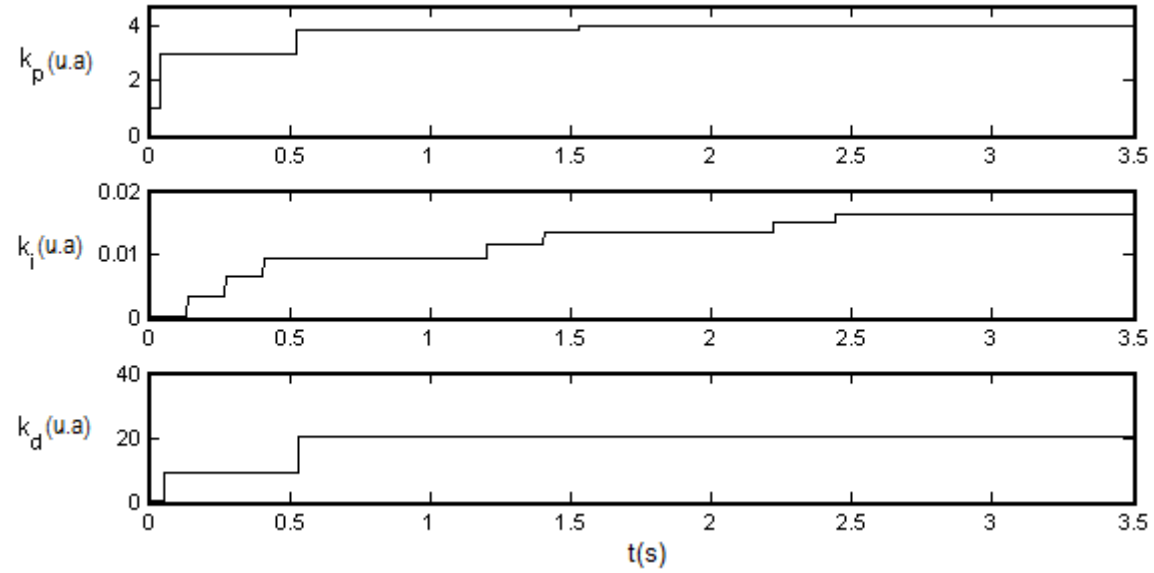
[23] Diseño e implementación de un sistema de control digital de posición para un motor DC. Universidad Santo Tomas de Bucaramanga, 2006.

Figura 106. Respuesta de la planta 1 con controlador PIDAD1 a una entrada escalón unitario periódico



En el primer transitorio el sistema mide el sobrepaso y el error en estado estable, en el tercer transitorio el sobrepaso se corrige por completo, y en el séptimo el error en estado estable.

Figura 107. Evolución de parámetros PID del controlador PIDAD1 a una e entrada escalón unitario periódico aplicado a la planta 1



En el cuarto transitorio el sistema considera que el tiempo de subida esta balanceado según la variación de k_p, de tal forma que este parámetro converge en ese instante. El parámetro k_i se considera sintonizado al sexto transitorio, obviamente cuando el error en estado estable desaparece después del tiempo de asentamiento. El parámetro k_d converge de forma rápida al segundo transitorio, ya que en el tercero el sobrepaso desaparece completamente. La convergencia de los parámetros del controlador PIDAD1 es:

$k_p = 3.967$ $k_i = 0.0165$ $k_d = 20.21$

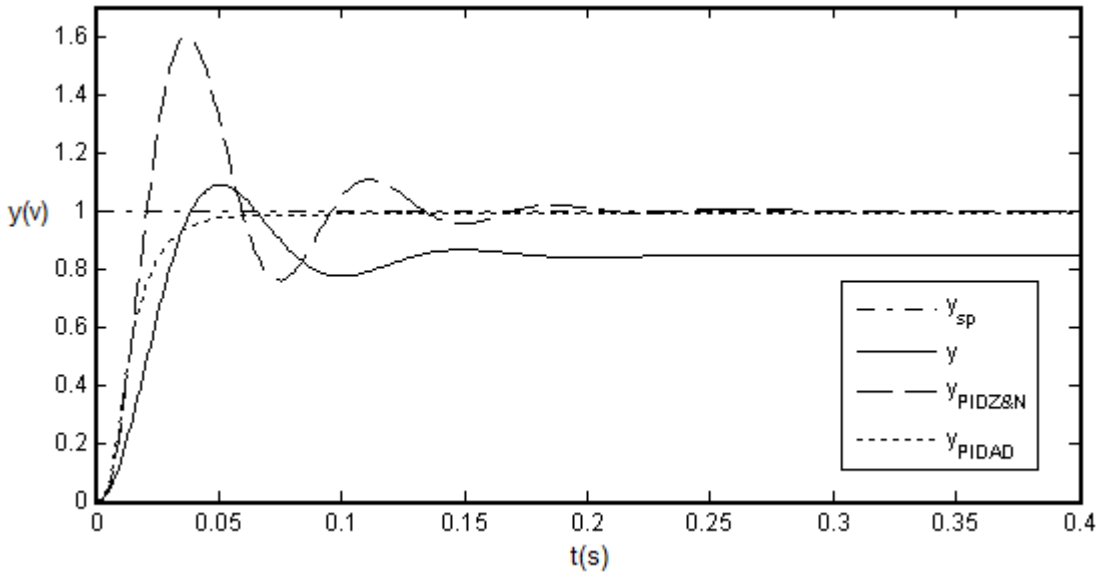
Para tener una referencia de comparación se utilizara un controlador sintonizado según Ziegler y Nichols (Z&N), para esto es necesario obtener el modelo de los dos parámetros que describe la planta,

$K_{es} = 5.540$ $a = 0.5554$ $L = 0.0104.$

Con ayuda de la tabla de sintonía de controladores PID según Z&N (PIDZ&N) se puede determinar los parámetros sintonizados:

$k_p = 2.160$ $k_i = 0.2085$ $k_d = 5.597$

Figura 108. Comparación de las respuestas de la planta 1 según los controladores aplicados



Evidentemente el controlador PIDAD1 tiene una respuesta al escalón más apropiada que el controlador PIDZ&N, se puede observar que ofrece una solución satisfactoria debido a que se cumple los requerimientos mínimos de funcionamiento de un proceso.

Tabla 32. Comparación cuantitativa de la respuesta transitoria de la planta 1 según los controladores a una entrada escalón unitario

Controlador	Tiempo de asentamiento (5%) $t_a(s)$	Error en estado estable e_{ss}	Sobrepaso máximo ov	Tiempo de subida $t_s\ (s)$
Sin control	0.1490	0.1560	25.70%	0.0284
PID Z&N	0.1540	0.0005	59.93%	0.0200
PIDAD	0.0420	0.0001	0%	0.0310

Tabla 33. Criterio ITAE de la planta 1 cuando este es operado por cada controlador

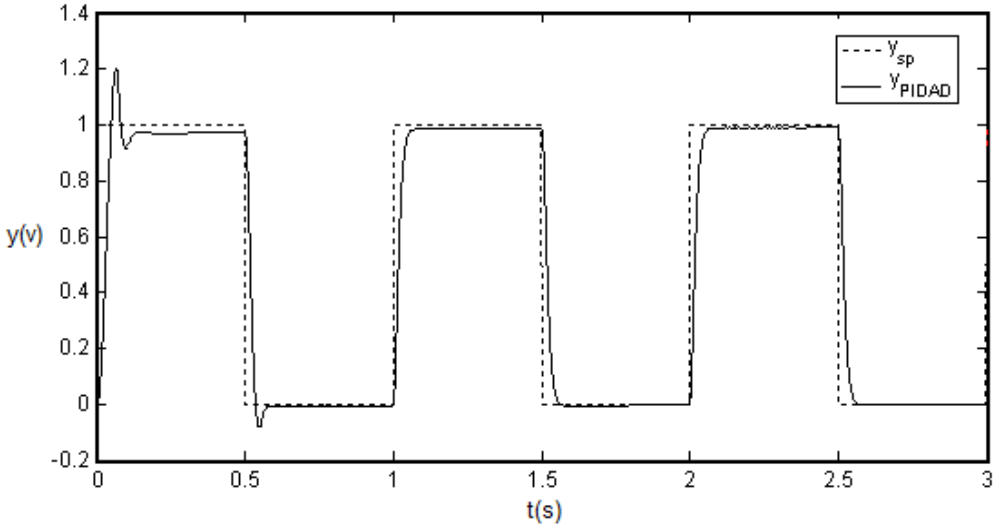
Controlador	ITAE (0.2s)	ITAE (0.4s)
Sin control	0.0033	0.0131
PID Z&N	0.0018	0.0020
PIDAD	0.0004	0.0010

El criterio del ITAE confirma el funcionamiento satisfactorio del controlador PIDAD1 por encima de los demás controladores.

5.1.2.2 PIDAD1 aplicado a la planta 2. La planta 2 representada por la función de transferencia $G_2(s)$, esta planta es una modificación de la planta 1, se ha duplicado el tiempo de retraso al doble de la planta 1, aumentando la dificultad para el controlador. Este modelo se simula para demostrar que el mecanismo adaptivo puede sintonizar de forma correcta aun cuando cambien los parámetros de la planta, en este caso el tiempo muerto (L) se aumenta al doble y los polos se acercan más al eje imaginario, aumentando su dominancia.

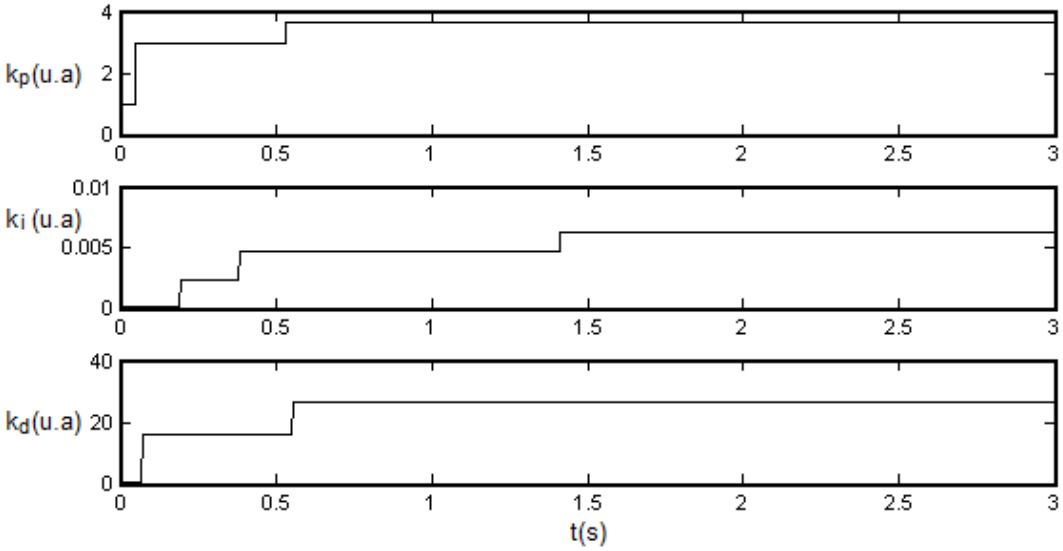
$$G_2(s) = \frac{19750}{(s + 14.86)(s + 12.24)} \tag{76}$$

Figura 109. Respuesta de la planta 2 con un controlador PIDAD1 a una entrada escalón unitario periódico



Cada efecto transitorio del escalón periódico unitario le sirve al controlador para mejorar su experiencia en el proceso, como se observa en la anterior figura el sobrepaso se corrige completamente en el segundo transitorio, el error en estado estable se considera corregido o dentro de los limites en el cuarto transitorio. El tiempo de subida que depende de k_p se considera estable en el segundo transitorio.

Figura 110. Evolución de parámetros PID del controlador PIDAD1 a una e entrada escalón unitario periódico aplicado a la planta 2



La convergencia de los parámetros del controlador PIDAD1 es:

$k_p = 3.636$ $k_i = 0.0062$ $k_d = 26.39$

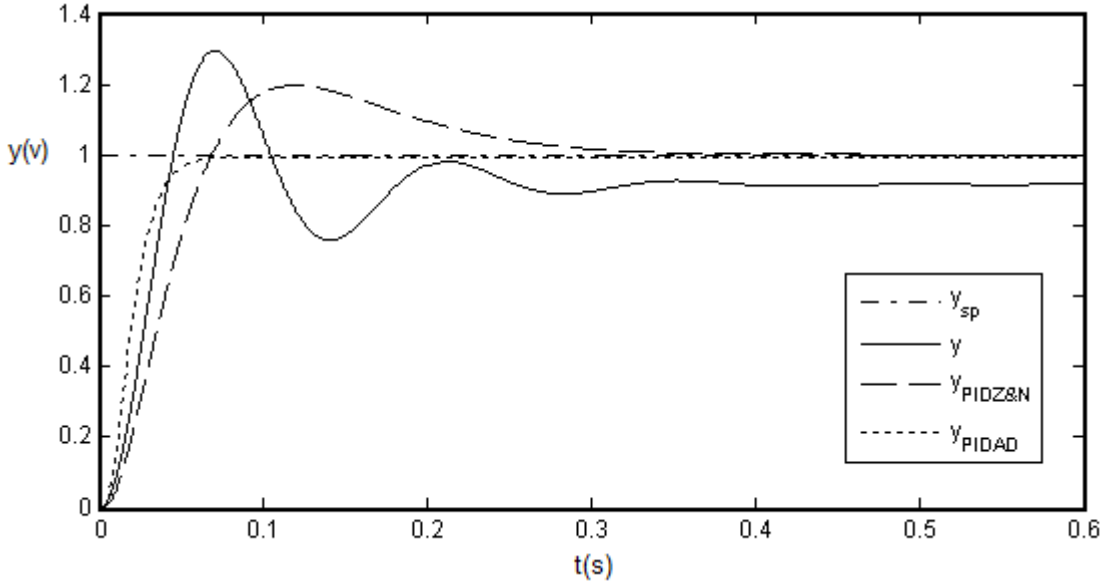
Para tener una referencia de comparación se utilizara un controlador sintonizado según Ziegler y Nichols, para esto es necesario obtener el modelo de los dos parámetros que describe la planta,

$K_{es} = 10.8$ $a = 1.3103$ $L = 0.0205$

Con ayuda de la tabla de sintonía de controladores PID según Z&N (PIDZ&N) se puede determinar los parámetros sintonizados:

$k_p = 0.9158$ $k_i = 0.0186$ $k_d = 11.29$

Figura 111. Comparación de las respuestas de la planta 2 según los controladores aplicados



Se puede observar que el controlador PIDAD1 responde de manera satisfactoria al sintonizar dos plantas muy diferentes, y es capaz de llevar la respuesta del proceso a los límites de funcionamiento adecuados sin importar el incremento de dificultad del proceso.

Tabla 34. Comparación cuantitativa de la respuesta transitoria de la planta 1 según los controladores aplicados a una entrada escalón unitario

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\ (s)$
Sin control	0.2820	0.0840	38.00%	0.0385
PID Z&N	0.2397	0.0004	19.69%	0.0591
PIDAD	0.0480	0.0008	0.00%	0.0390

Tabla 35. Criterio ITAE de la planta 2 cuando este es operado por cada controlador

Controlador	**ITAE (0.3s)**	**ITAE (0.6s)**
Sin control	0.0052	0.0166
PID Z&N	0.0045	0.0048
PIDAD	0.0006	0.0014

El criterio del ITAE confirma el funcionamiento satisfactorio del controlador PIDAD1 por encima de los demás controladores.

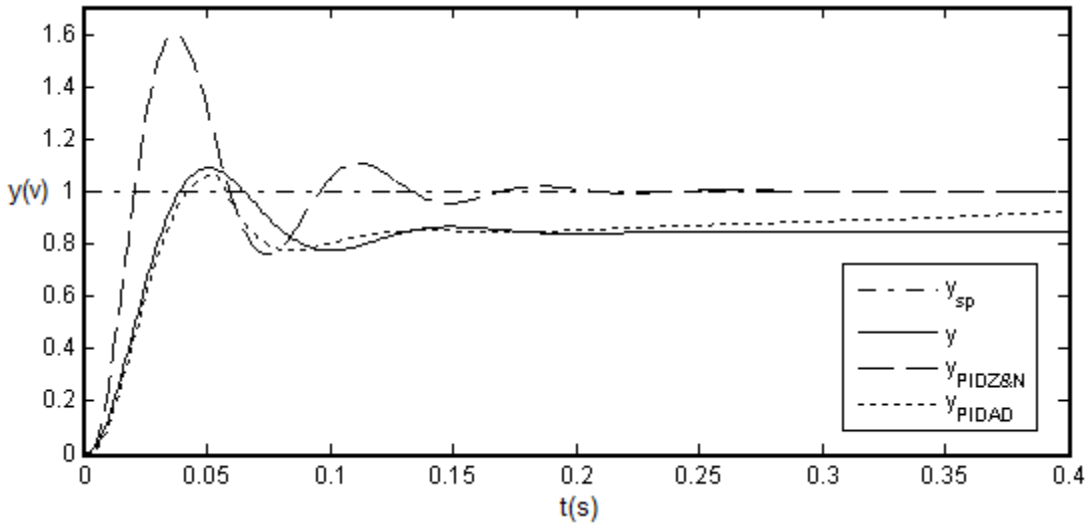
5.1.3 Pruebas de simulación del controlador PID con MAD2 (PIDAD2). El mecanismo difuso adaptivo 2 se puede utilizar junto con el controlador PID como un controlador adaptivo, dicho de otra forma, es capaz de adaptarse a la dinámica del proceso según este lo requiera. Este controlador se prueba con varias señales de entrada, partiendo con los parámetros sin sintonizar:

$k_p = 1$ $k_i = 0.0001$ $k_d = 0.0001$

El MAD2 al igual que el anterior trata de llevar la respuesta transitoria a un rango de operación adecuado variando los parámetros, cuando ya no se haga necesario la variación de estos parámetros estos convergen en un valor.

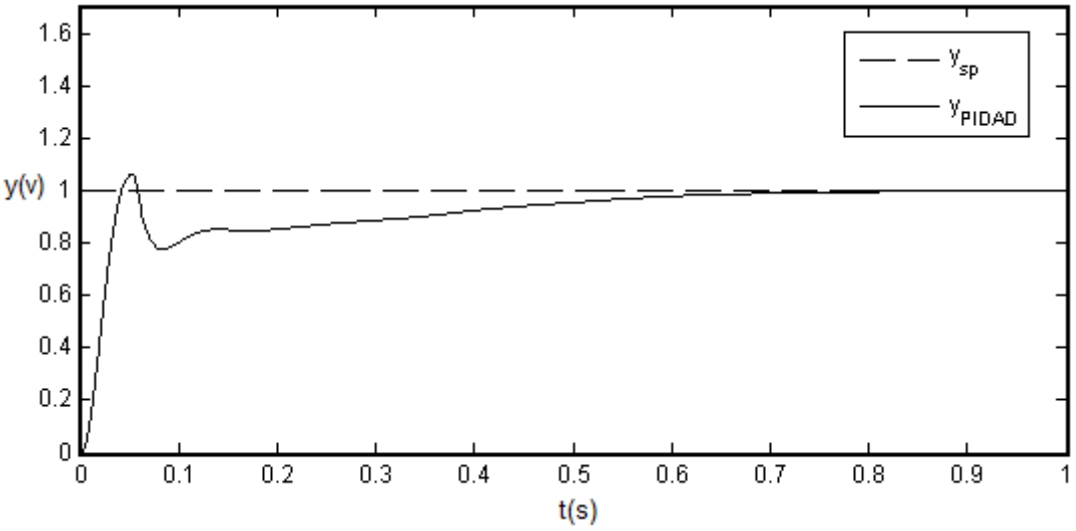
5.1.3.1 PIDAD2 aplicado a la planta 1 a una entrada escalón unitario. La señal de entrada del sistema en este caso es un escalón unitario, esto significa que el mecanismo adaptivo solo experimenta un transitorio.

Figura 112. Comparación de las respuestas de la planta 1 según los controladores aplicados a una entrada escalón unitario



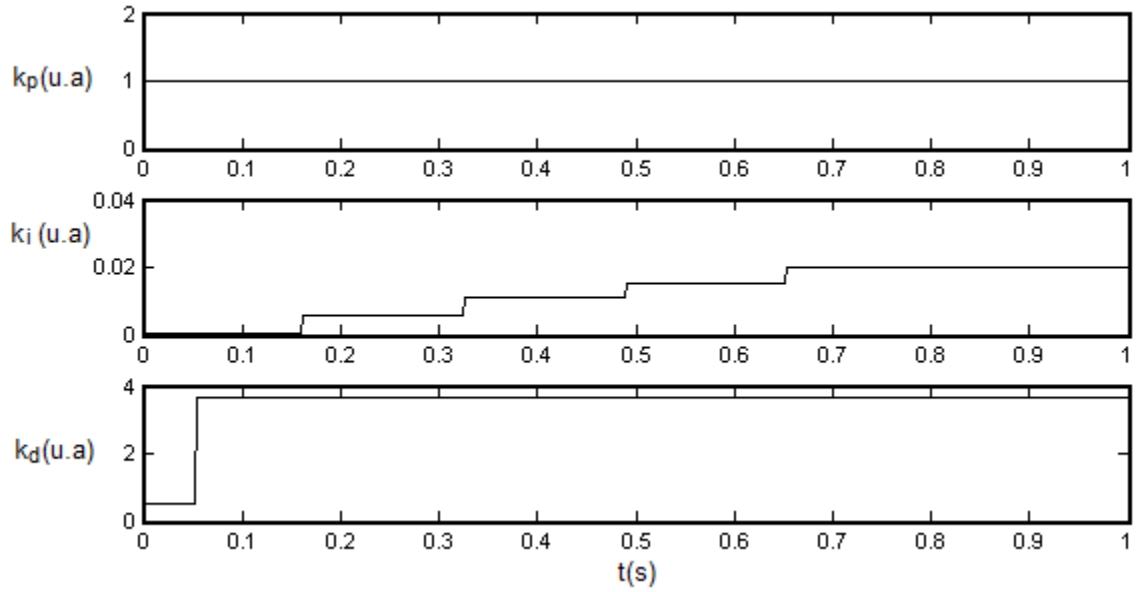
El controlador PIDAD2 trata de corregir el error en estado estable a medida que avanza el proceso.

Figura 113. Respuesta de la planta 1 con controlador PIDAD2 a una entrada escalón unitario



El error en estado estable se corrige completamente en el tiempo 0.8 segundos, mostrando un comportamiento adaptivo.

Figura 114. Evolución de parámetros PID del controlador PIDAD2 a una e entrada escalón unitario periódico aplicado a la planta 1



Convergencia de los parámetros

$k_p = 1.000$ $k_i = 0.02016$ $k_d = 3.685$

Como se esperaba la mejora se presenta en el error en estado estable y en el sobrepaso, corrigiendo el error estable en gran medida y el sobrepaso en menor medida disminuyéndolo 5 veces a lo habitual del proceso. Cabe resaltar que solamente se ha aplicado la experiencia de un transitorio, y aun se puede mejorar más la respuesta del proceso con la aplicación de escalones.

Tabla 36. Comparación cuantitativa de la respuesta transitoria de la planta 1 según los controladores aplicados a una entrada escalón unitario

Controlador	Tiempo de asentamiento 5% $t_a(s)$	Error en estado estable e_{ss}	Sobrepaso máximo ov	Tiempo de subida $t_s(s)$
Sin control	0.1139	0.1560	25.70%	0.0284
PID Z&N	0.1540	0.0005	59.93%	0.0200
PIDAD	0.5220	0.0002	5.90%	0.0360

5.1.3.2 PIDAD2 aplicado a la planta 1 a una entrada escalón unitario periódico. La entrada del sistema es un escalón periódico unitario, el controlador parte con las mismas condiciones iníciales. Se hace pruebas para el controlador PIDZ&N con el fin de tener un punto de referencia, así como también a la planta sin control.

Figura 115. Respuestas de la planta 1 a una entrada escalón unitario periódico según cada controlador

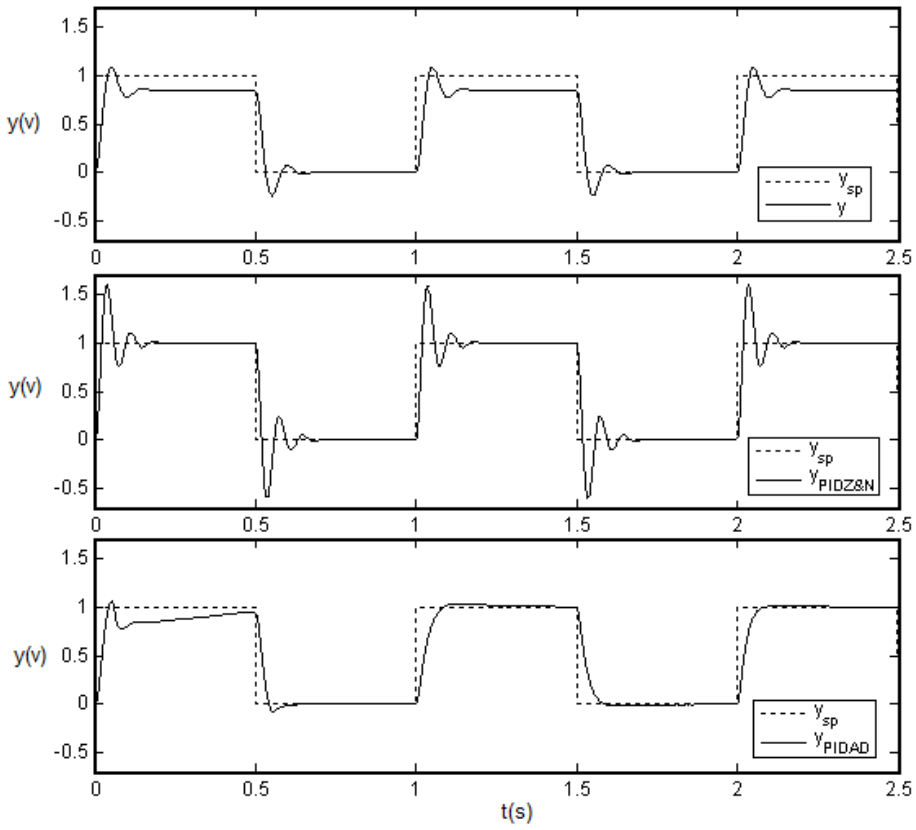
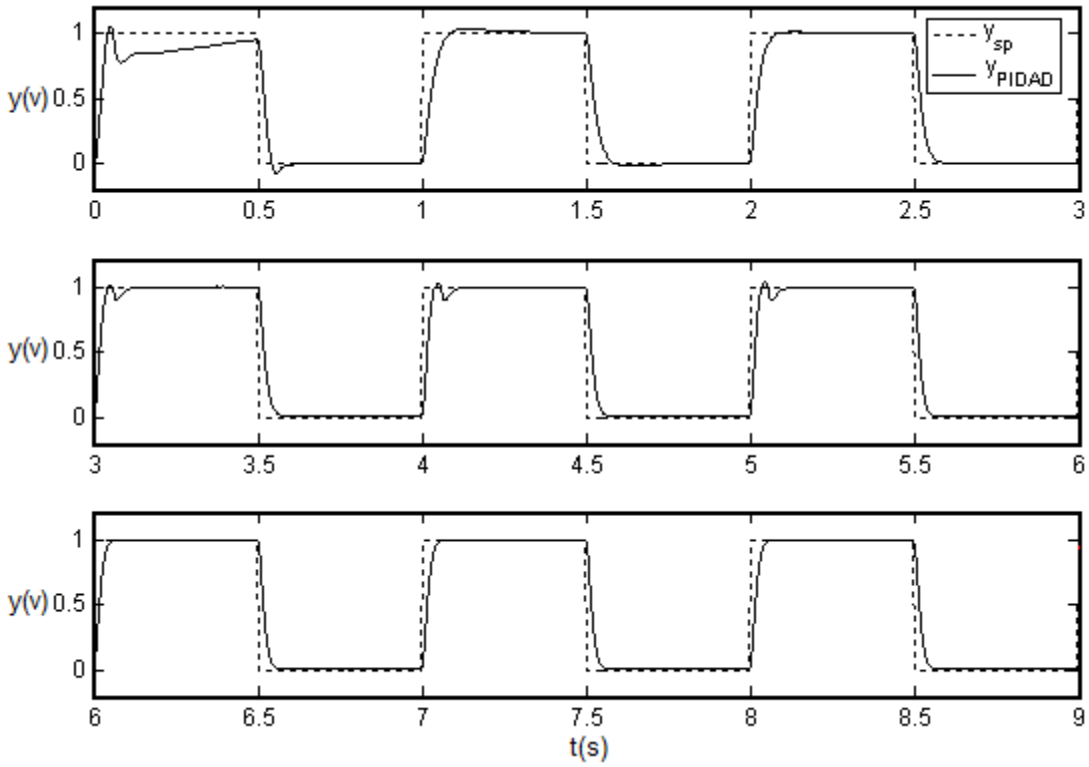
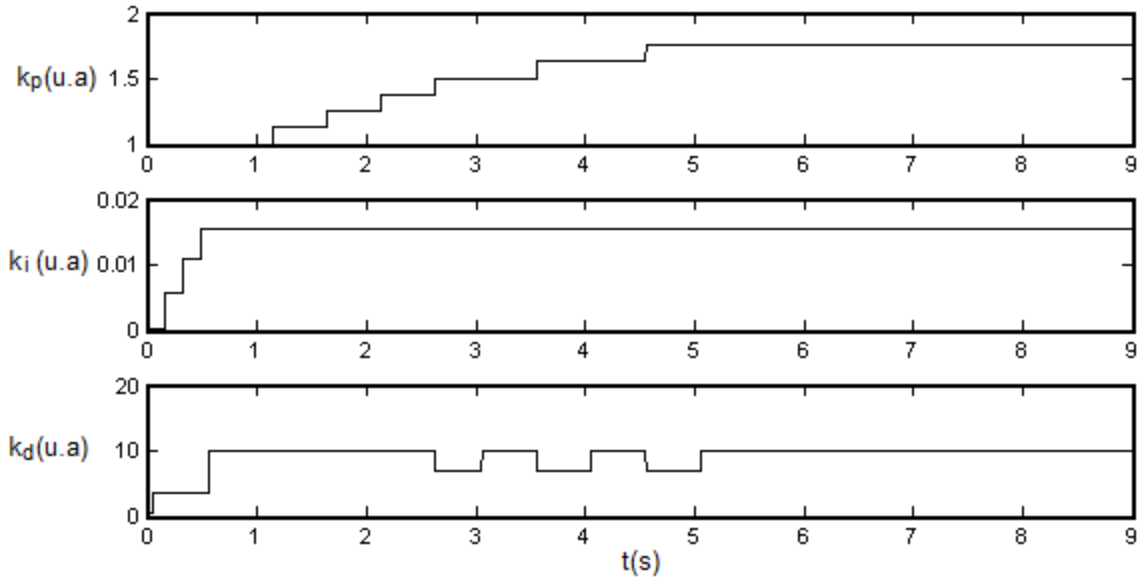


Figura 116. Respuesta de la planta 1 con un controlador PIDAD2 a una entrada escalón unitario periódico



El controlador PIDAD2 es capaz de llevar el sobrepaso dentro de los límites permitidos en el tercer transitorio como también el error en estado estable. El tiempo de subida por el contrario se demora en estabilizar los parámetros k_p y k_d los cuales varían hasta nivelarse encontrando un tiempo de subida adecuado para el proceso según el controlador PIDAD2.

Figura 117. Evolución de los parámetros k_p, k_i y k_d del controlador PIDAD2 a la entrada escalón periódico aplicado a la planta 1



Convergencia de los parámetros

$k_p = 1.761$ $k_i = 0.01547$ $k_d = 10.130$

Las oscilaciones de k_d suceden para probar el tiempo de subida a varios valores de k_p, finalmente ambos consiguen llegar a un equilibrio mejorando el tiempo de subida.

Figura 118. Comparación de las respuestas de la planta 1 a una entrada escalón unitario según los controladores aplicados

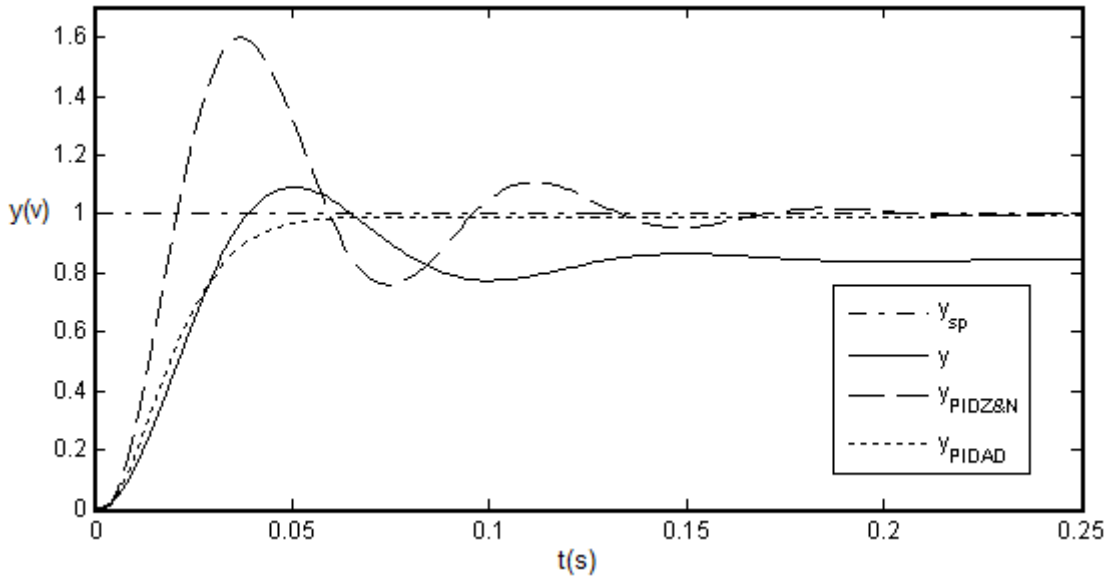


Tabla 37. Comparación cuantitativa de la respuesta transitoria de la planta 1 a una entrada escalón unitario periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento 5%** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s(s)$
Sin control	0.1356	0.1560	25.70%	0.0284
PID Z&N	0.1540	0.0005	59.93%	0.0200
PIDAD	0.0470	0.0004	0.04%	0.0390

Tabla 38. Criterio ITAE de la planta 1 cuando este es operado por cada controlador

Controlador	**ITAE (0.125s)**	**ITAE (0.25s)**
Sin control	0.0013	0.0051
PID Z&N	0.0014	0.0019
PIDAD	0.0004	0.0006

5.1.3.3 PIDAD2 aplicado a la planta 1 a una entrada escalón multinivel periódico. La entrada del sistema es un escalón multinivel periódico, el controlador parte con las mismas

condiciones iníciales. Se toman como controladores de referencia el proceso sin control y el controlador PIDZ&N

Figura 119. Respuestas de la planta 1 a una entrada escalón multinivel periódico según cada controlador

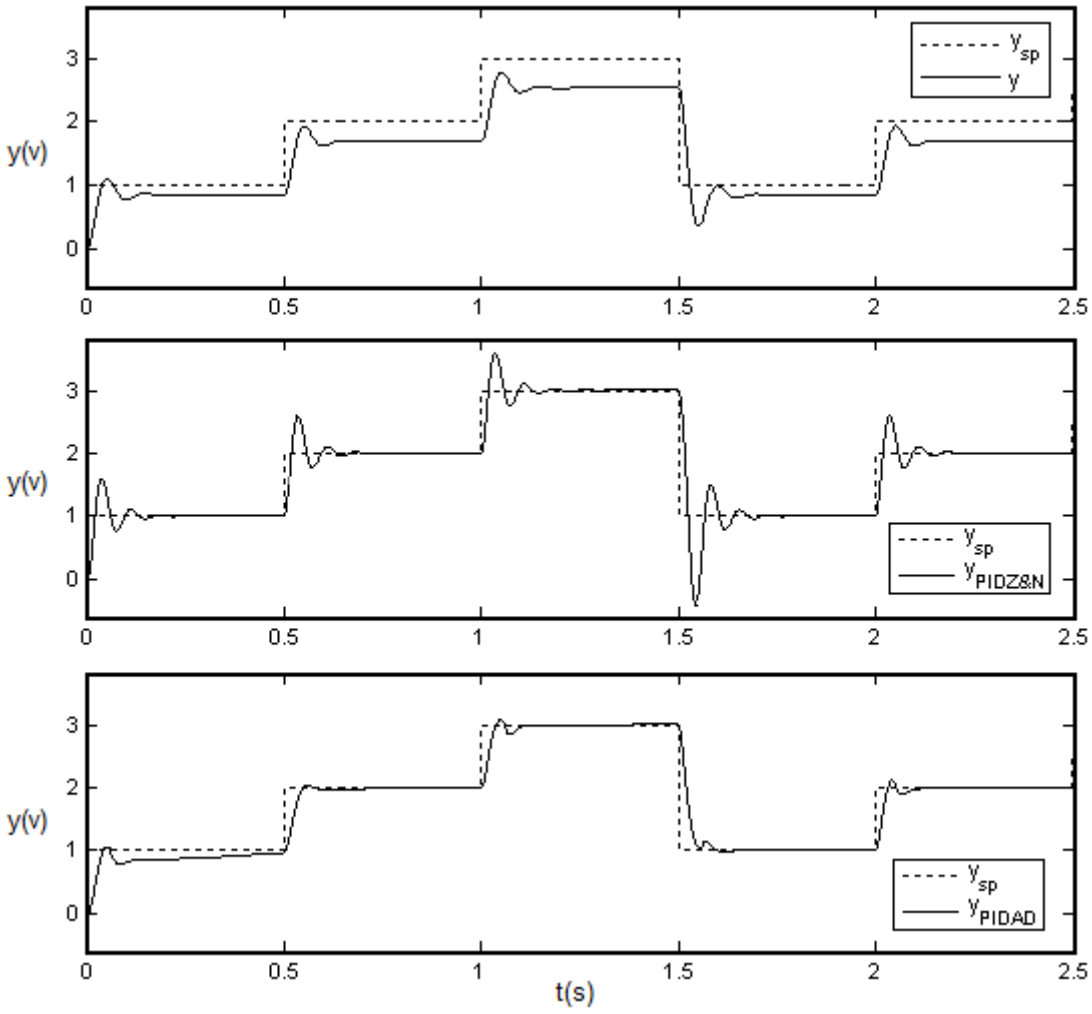
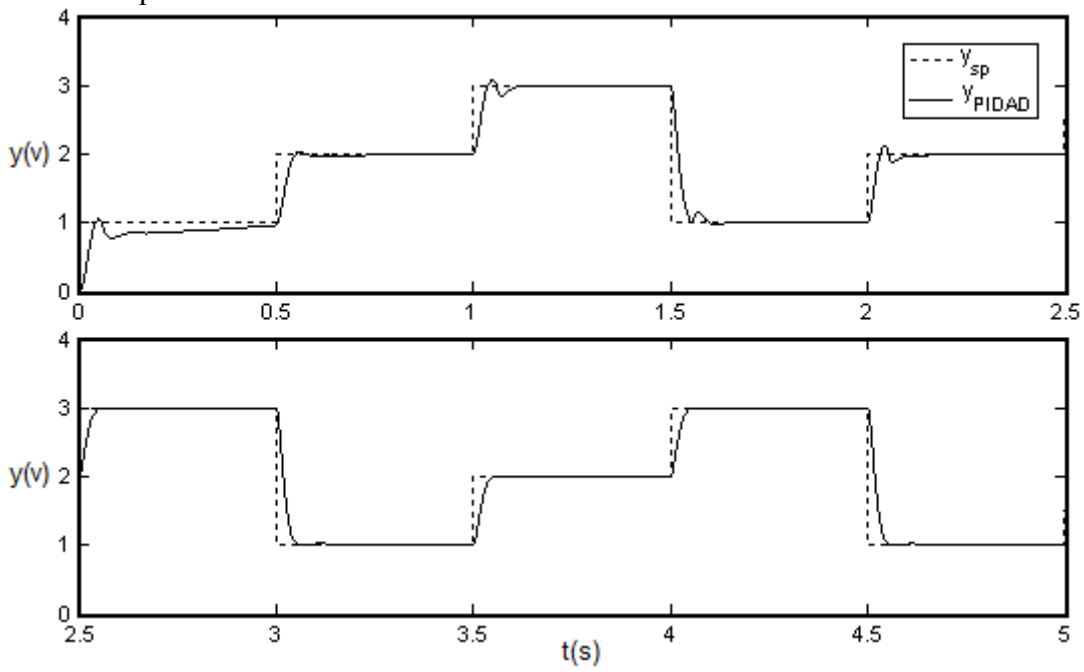
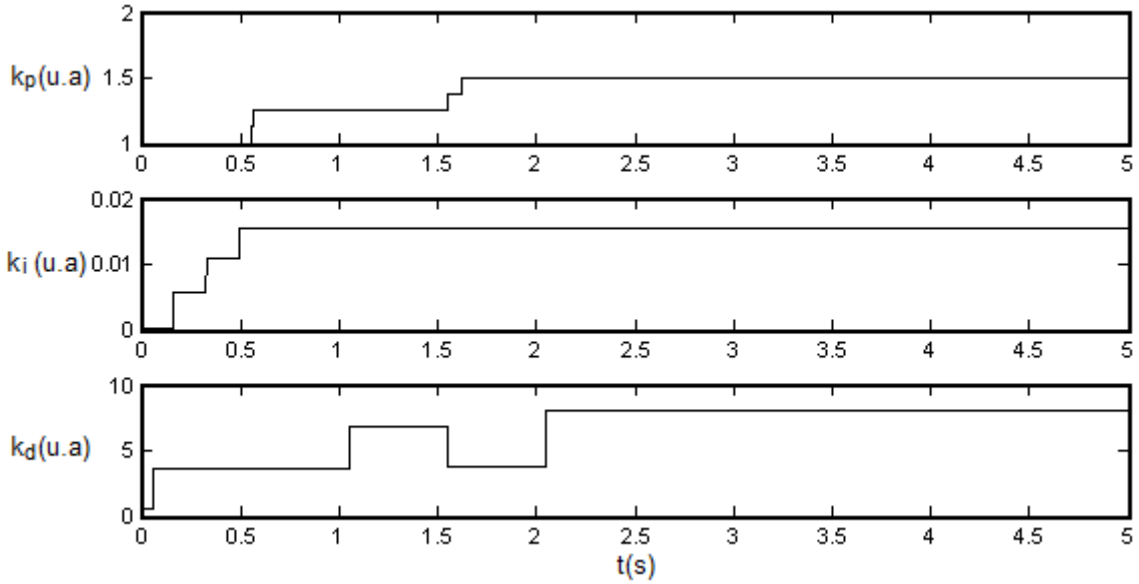


Figura 120. Respuesta de la planta 1 con un controlador PIDAD2 a una entrada escalón multinivel periódico



La señal escalón periódico multinivel prueba la versatilidad del controlador PIDAD2 sobre el PIDAD1, este es mucho más estable ya que cuenta con un nivel de decisión más complejo debido a su sistema difuso. Los resultados obtenidos muestran un control del proceso satisfactorio ya que lleva la respuesta del proceso a los límites permitidos.

Figura 121. Evolución de los parámetros k_p, k_i y k_d del controlador PIDAD2 a la entrada escalón multinivel periódico aplicado a la planta 1



Convergencia de los parámetros

$k_p = 1.507$ $k_i = 0.01547$ $k_d = 8.030$

Igual que en la prueba anterior se muestra una oscilación en el parámetro k_d, esto debido a la consecución del equilibrio con k_p, para llevar a un mejor valor al tiempo de subida. El sobrepaso y el error en estado estable se corrigen de forma satisfactoria.

Figura 122. Comparación de las respuestas de la planta 1 a una entrada escalón unitario según los controladores aplicados

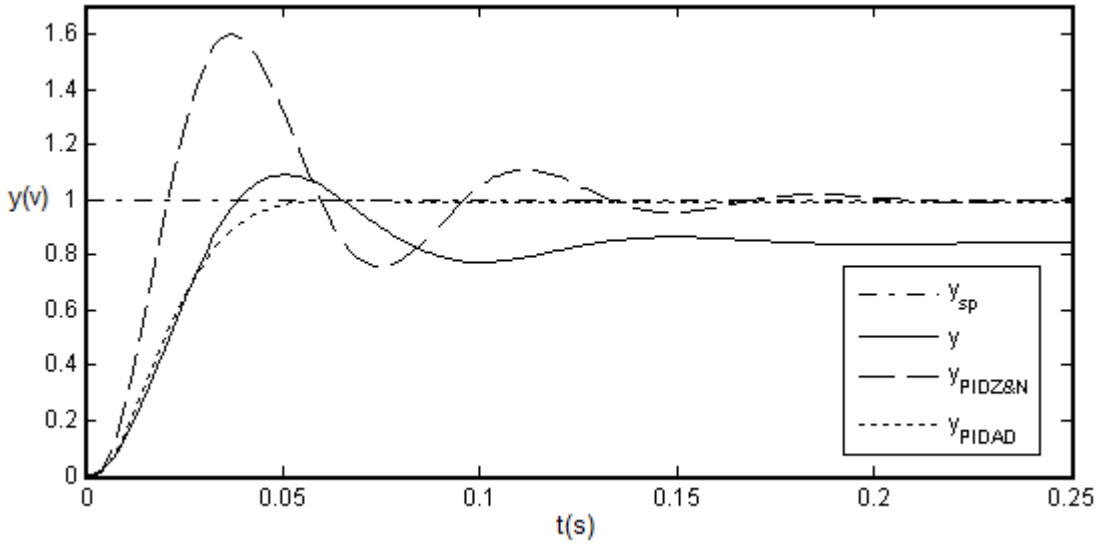


Tabla 39. Comparación cuantitativa de la respuesta transitoria de la planta 1 a una entrada escalón multinivel periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%) $t_a(s)$**	**Error en estado estable e_{ss}**	**Sobrepaso máximo ov**	**Tiempo de subida t_s (s)**
Sin control	0.1178	0.1560	25.70%	0.0284
PID Z&N	0.1540	0.0005	59.93%	0.0200
PIDAD	0.0450	0.0002	0.02%	0.0390

De nuevo se puede observar que el controlador PIDAD2 tiene una respuesta satisfactoria superando al controlador PIDZ&N.

5.1.3.4 PIDAD2 aplicado planta 1 a una entrada escalón y perturbación. Se ha aplicado una perturbación de 0.5 de magnitud en el tiempo 0.3 segundos a los controladores de referencia y al controlador PIDAD2 con los parámetros en los que sistema converge. El criterio del ITAE es muy preciso para evaluar el desempeño del controlador, se observa que el controlador PIDAD2 con un ITAE de 0.0007 responde mejor por el doble del ITAE del controlador PIDZ&N el cual tiene un ITAE = 0.0013 en los primeros 3 segundos del proceso, cuando la perturbación ocurre esta ventaja se mantiene aunque reduce respecto al controlador PIDZ&N.

Figura 123. Comparación de las respuestas de la planta 1 a una entrada escalón unitario con perturbación según los controladores aplicados

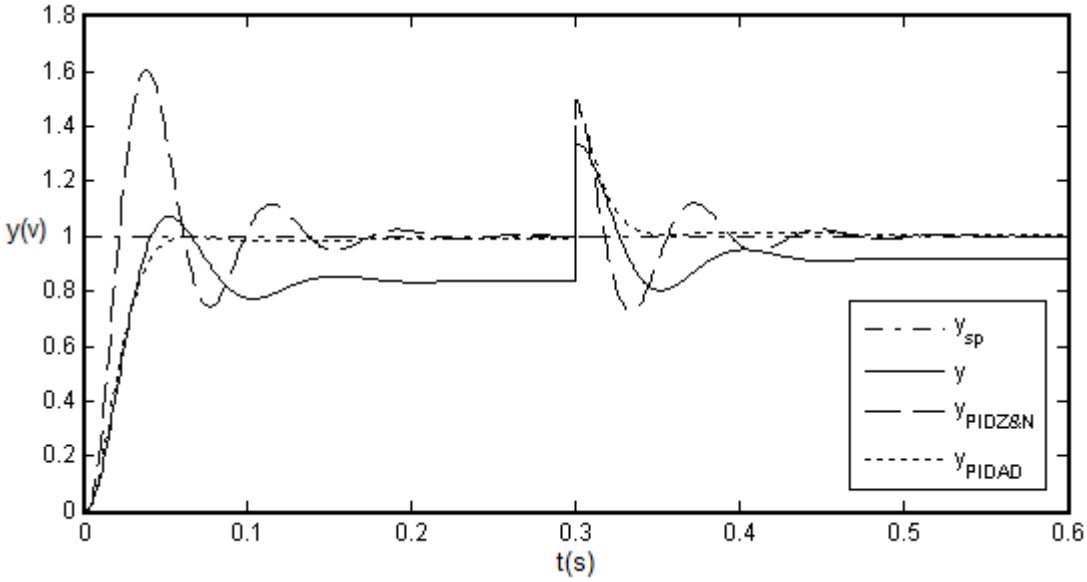


Tabla 40. Criterio ITAE de la planta 1 cuando este es operado por cada controlador

Controlador	**ITAE (0.3s)**	**ITAE (0.6s)**
Sin control	0.0074	0.0204
PID Z&N	0.0013	0.0061
PIDAD	0.0007	0.0042

5.1.3.5 PIDAD2 aplicado a la planta 1 a una entrada rampa periódica y seno. Las señales de rampa periódica y seno, no son entradas que se utilizan regularmente en los procesos, aun así resulta interesante observar el comportamiento de la planta con el controlador PIDAD2.

Figura 124. Respuesta de la planta 1 a una entrada rampa periódica con un controlador PIDAD2

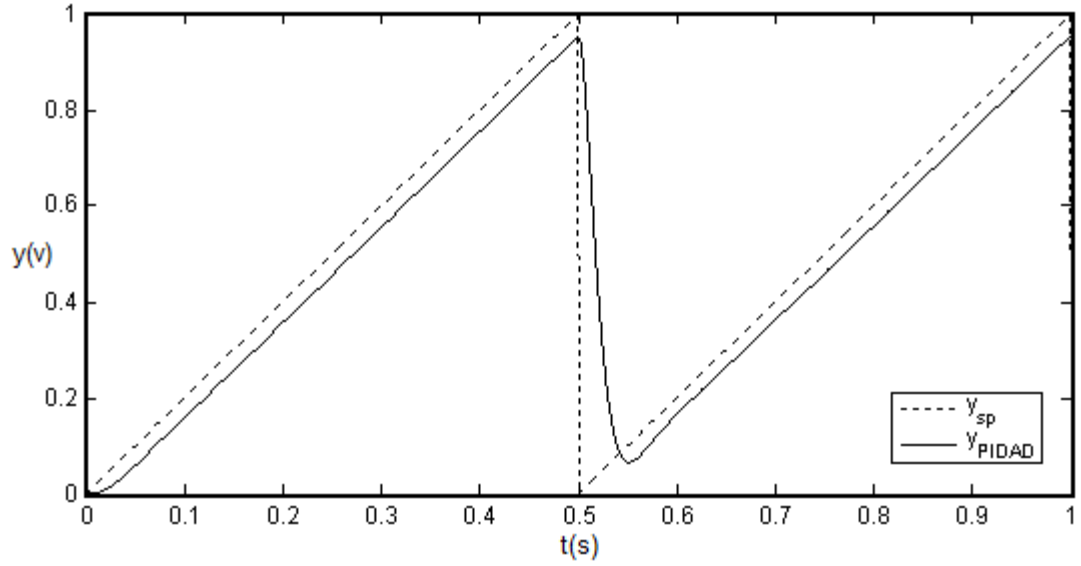
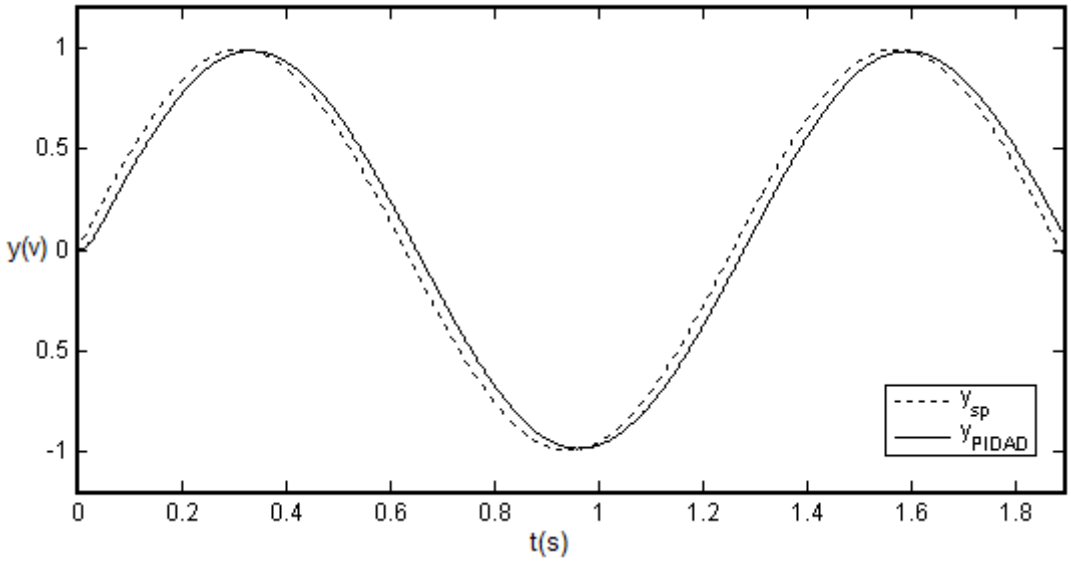


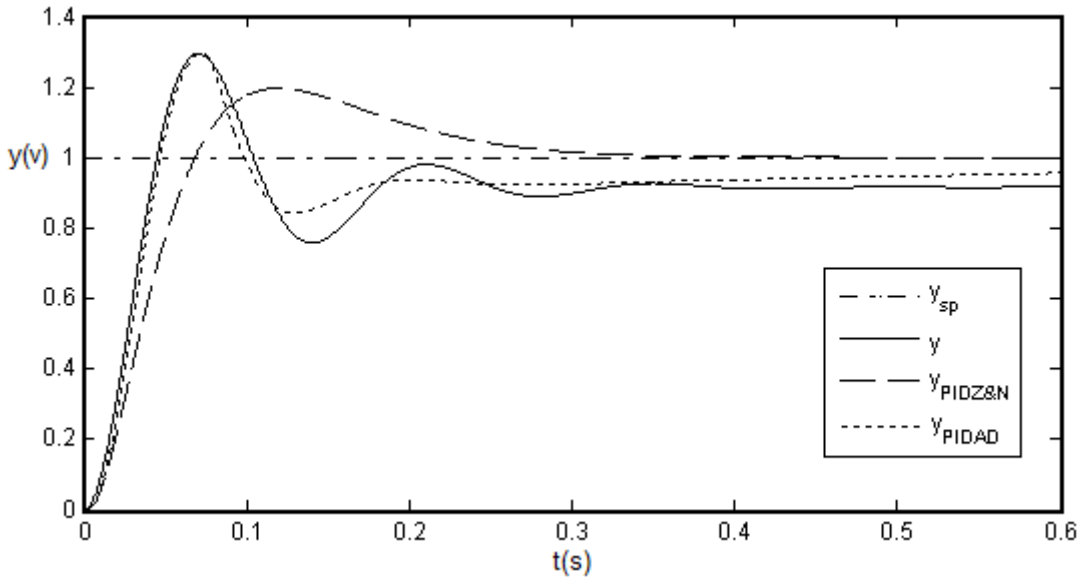
Figura 125. Respuesta de la planta 1 a una entrada seno con un controlador PIDAD2



Se observa que el controlador muestra un comportamiento de seguimiento de la señal, estas señales sirven para observar su comportamiento a diferentes entradas aunque se enfoco el diseño del controlador para adaptarse a señales escalón debido a que son las utilizadas en los procesos industriales.

5.1.3.6 PIDAD2 aplicado a la planta 2 a una entrada escalón unitario. La señal de entrada del sistema en este caso es un escalón unitario, esto significa que el mecanismo adaptivo solo experimenta un transitorio.

Figura 126. Comparación de las respuestas de la planta 2 según los controladores aplicados a una entrada escalón unitario



Como ya se menciono la planta 2 cuenta con mayor dificultad para el controlador, no obstante se observa en la figura anterior que el controlador responde corrigiendo el error en estado estable y disminuyendo el sobrepaso.

Figura 127. Respuesta de la planta 2 con controlador PIDAD2 a una entrada escalón unitario

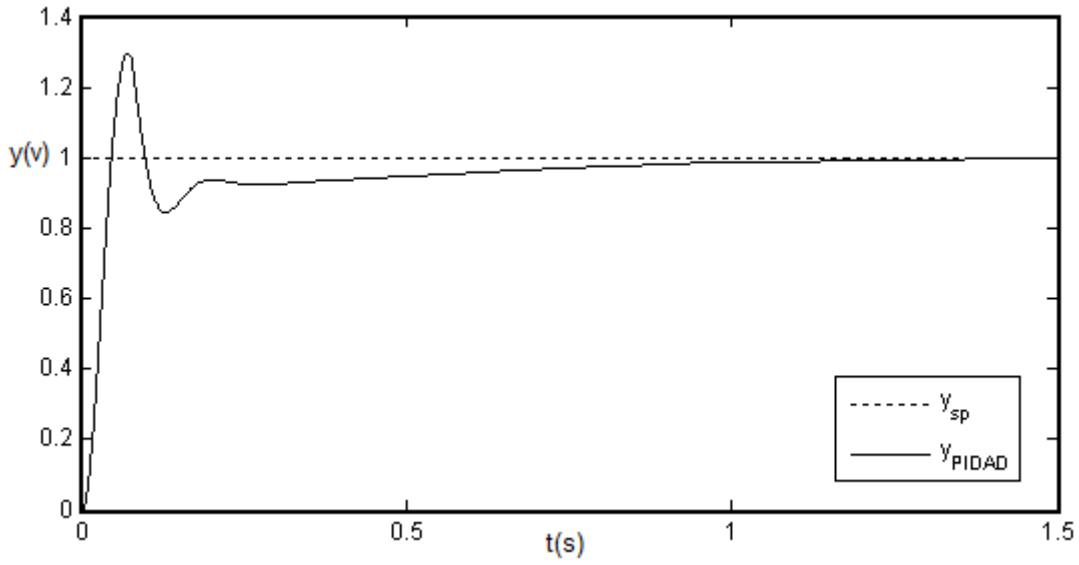
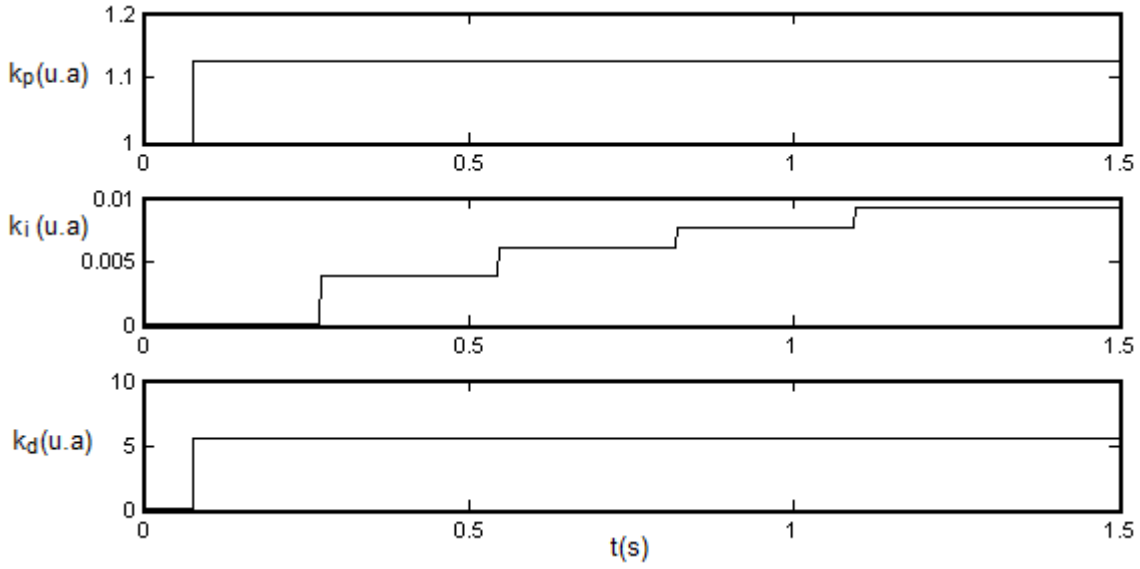


Tabla 41. Comparación cuantitativa de la respuesta transitoria de la planta 2 según los controladores aplicados a una entrada escalón unitario

Controlador	**Tiempo de asentamiento (5%)** $t_a\,(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\,(s)$
Sin control	0.1985	0.0840	38.00%	0.0385
PID Z&N	0.2397	0.0004	19.69%	0.0591
PIDAD	0.6233	0.0004	29.53%	0.0427

Figura 128. Evolución de los parámetros k_p, k_i y k_d del controlador PIDAD2 a la entrada escalón multinivel periódico aplicado a la planta 2

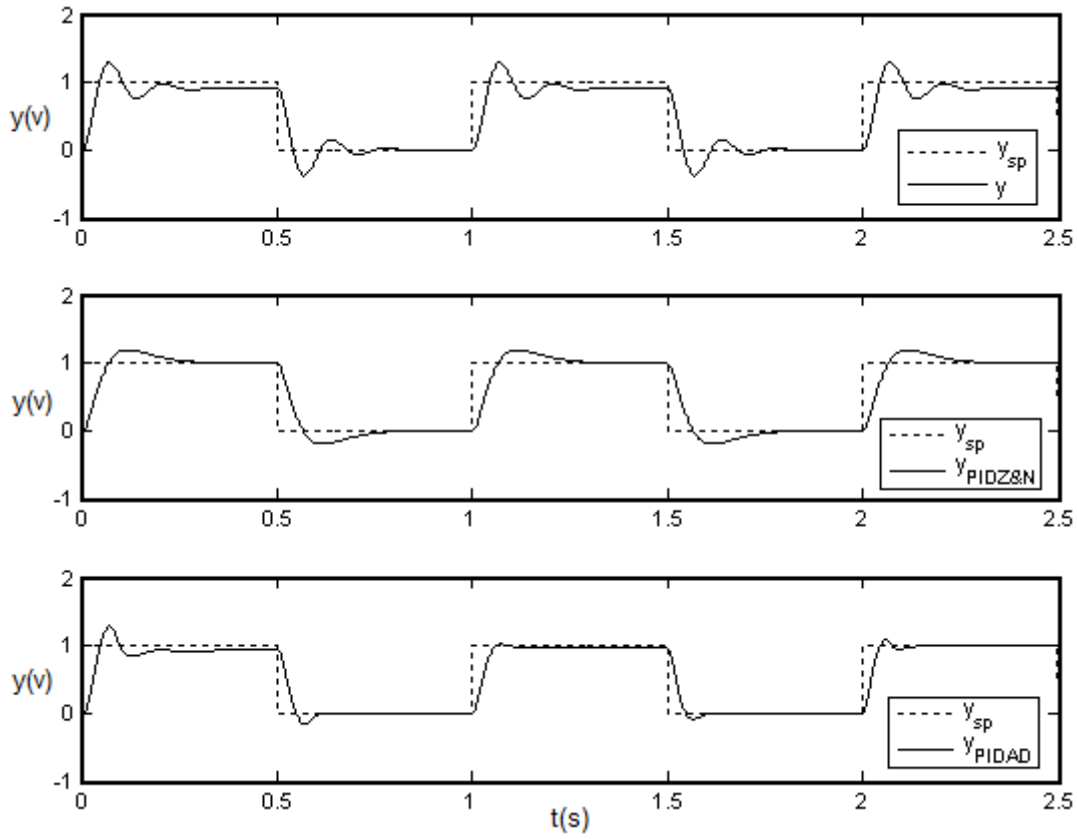


Convergencia de los parámetros

$k_p = 1.127$ $k_i = 0.0077$ $k_d = 5.601$

5.1.3.7 PIDAD2 aplicado a la planta 2 a una entrada escalón unitario periódico. La entrada del sistema es un escalón periódico unitario, el controlador parte con las mismas condiciones iníciales.

Figura 129. Respuestas de la planta 2 a una entrada escalón unitario periódico según cada controlador



El controlador PIDAD2 responde de manera que puede llevar el error en estado estable reduciéndolo a niveles aceptables en el tercer sobrepaso, sin embargo se nota al tratar el sobrepaso tarda un poco mas debido a que necesita una combinación de k_p y k_d para nivelar el tiempo de subida, el controlador converge después de unos transitorios obteniendo una respuesta satisfactoria dentro de los límites permitidos.

Figura 130. Respuesta de la planta 2 con un controlador PIDAD2 a una entrada escalón unitario periódico

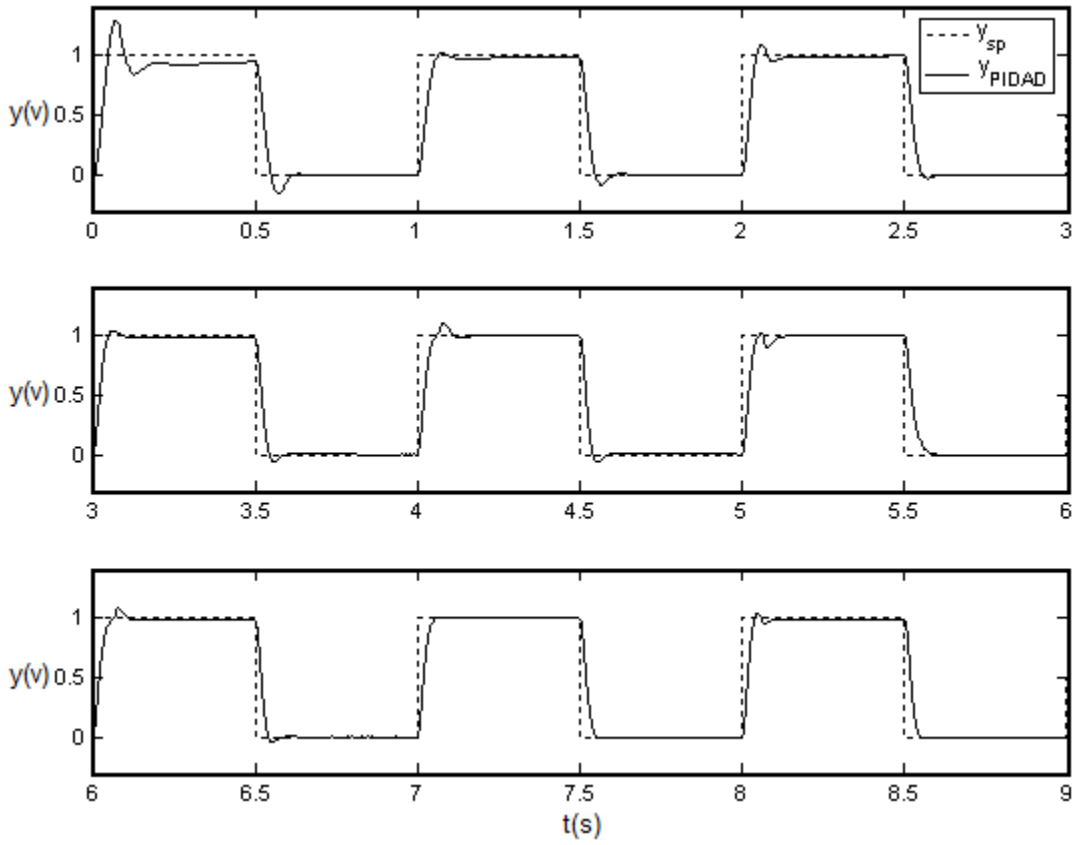
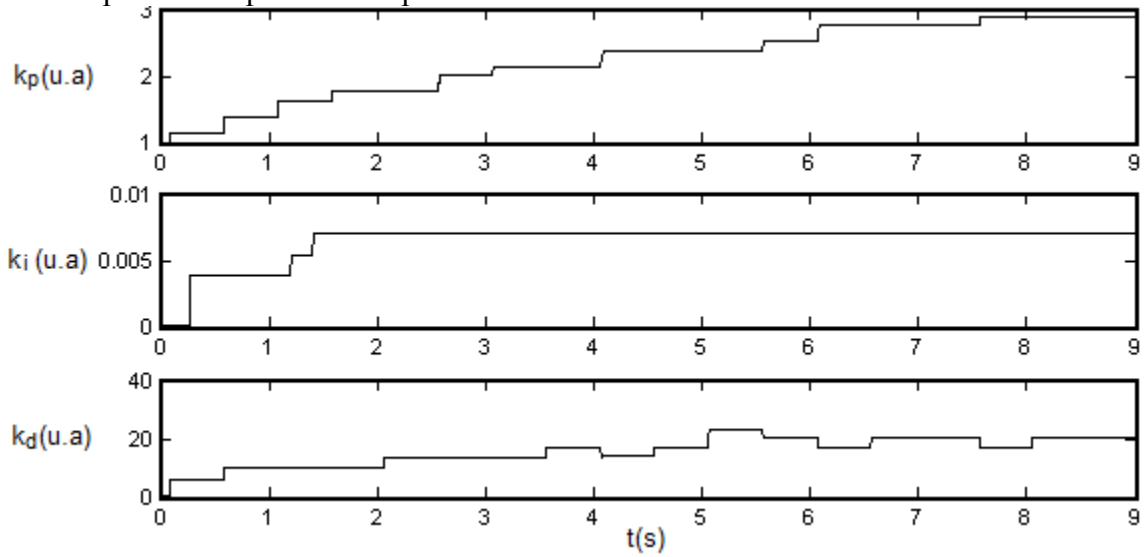


Figura 131. Evolución de los parámetros k_p, k_i y k_d del controlador PIDAD2 a la entrada escalón periódico aplicado a la planta 2



Convergencia de los parámetros

$k_p = 2.901$ $k_i = 0.0070$ $k_d = 20.28$

Figura 132. Comparación de las respuestas de la planta 2 a una entrada escalón unitario según los controladores aplicados

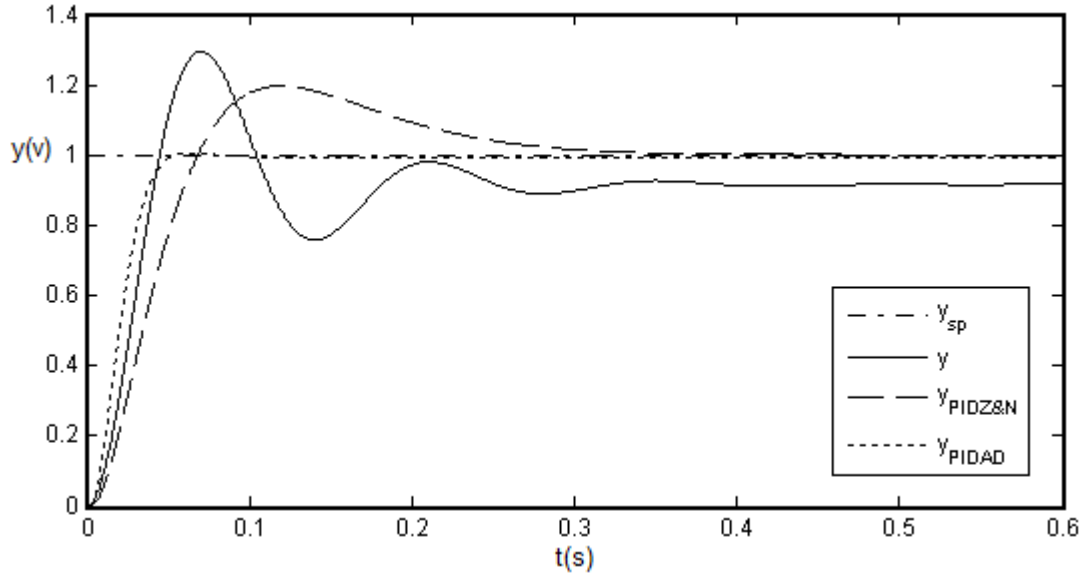


Tabla 42. Comparación cuantitativa de la respuesta transitoria de la planta 2 a una entrada escalón unitario periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\ (s)$
Sin control	0.2730	0.0840	38.00%	0.0385
PID Z&N	0.2397	0.0004	19.69%	0.0591
PIDAD	0.0438	0.0003	0.72%	0.0375

Tabla 43. Criterio ITAE de la planta 2 cuando este es operado por cada controlador

Controlador	**ITAE (0.3s)**	**ITAE (0.6s)**
Sin control	0.0052	0.0166
PID Z&N	0.0045	0.0048
PIDAD	0.0005	0.0012

El controlador PIDAD2 muestra clara ventaja en la sintonía de sus parámetros respecto al controlador PID, el cual supera su eficiencia según el ITAE en 4 veces.

5.1.3.8 PIDAD2 aplicado a la planta 2 a una entrada escalón multinivel periódico. La entrada del sistema es un escalón multinivel periódico, el controlador parte con las mismas condiciones iníciales.

Figura 133. Respuestas de la planta 2 a una entrada escalón multinivel periódico según cada controlador

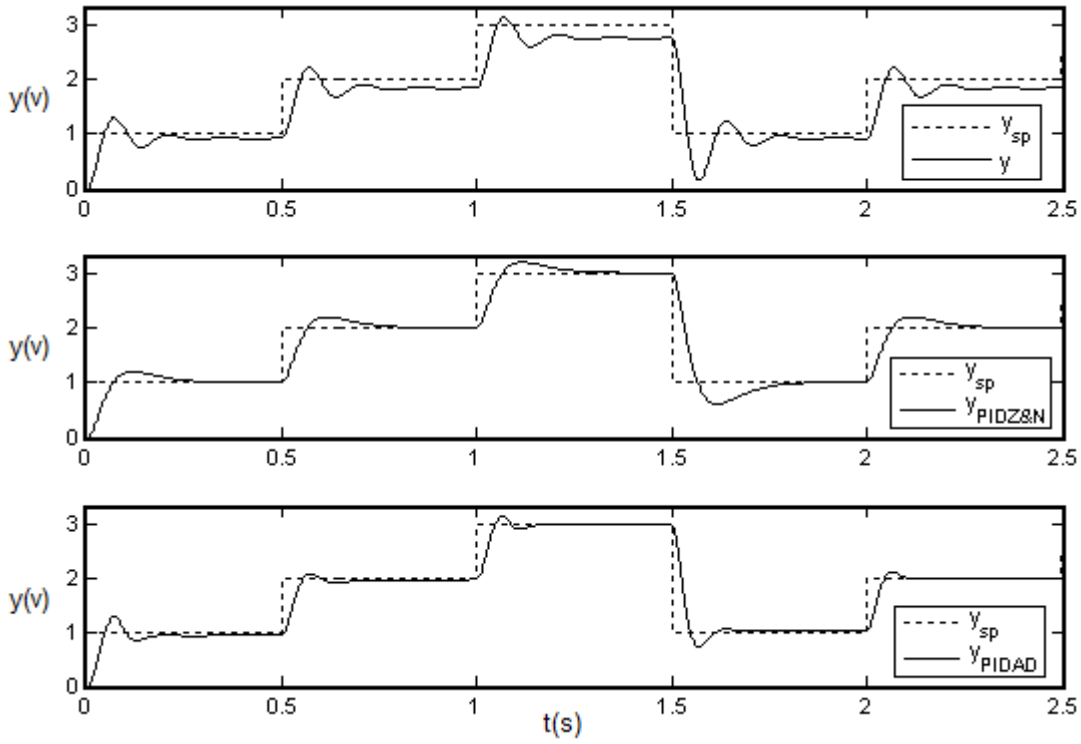


Figura 134. Respuesta de la planta 2 con un controlador PIDAD2 a una entrada escalón multinivel periódico

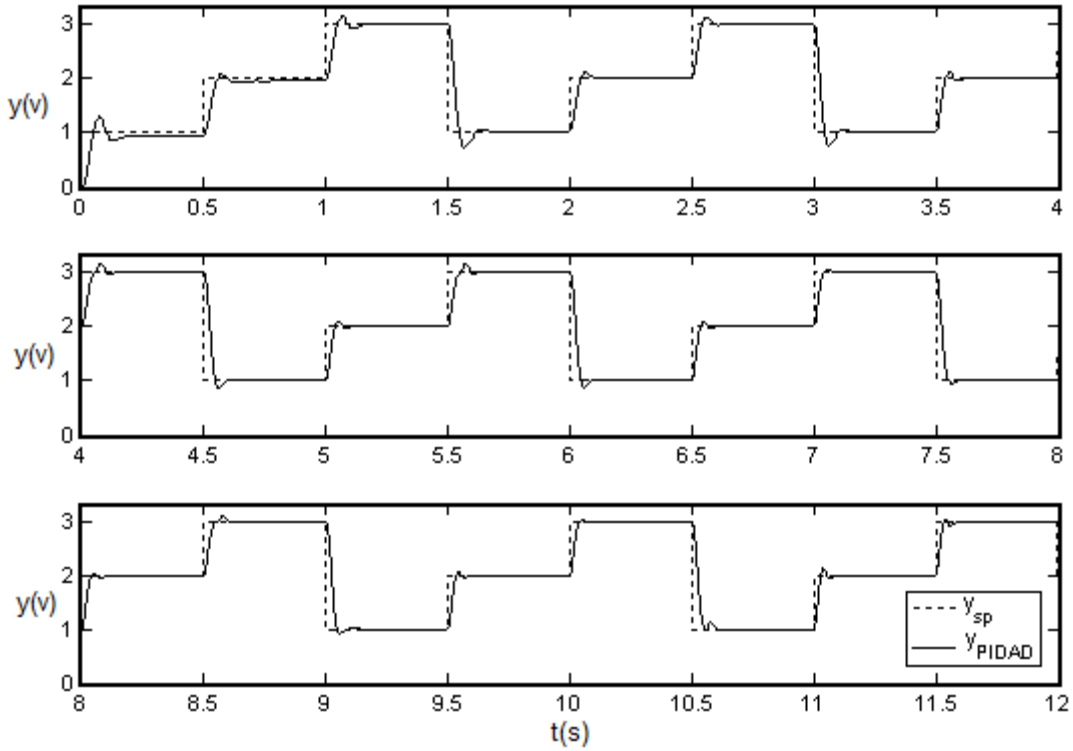
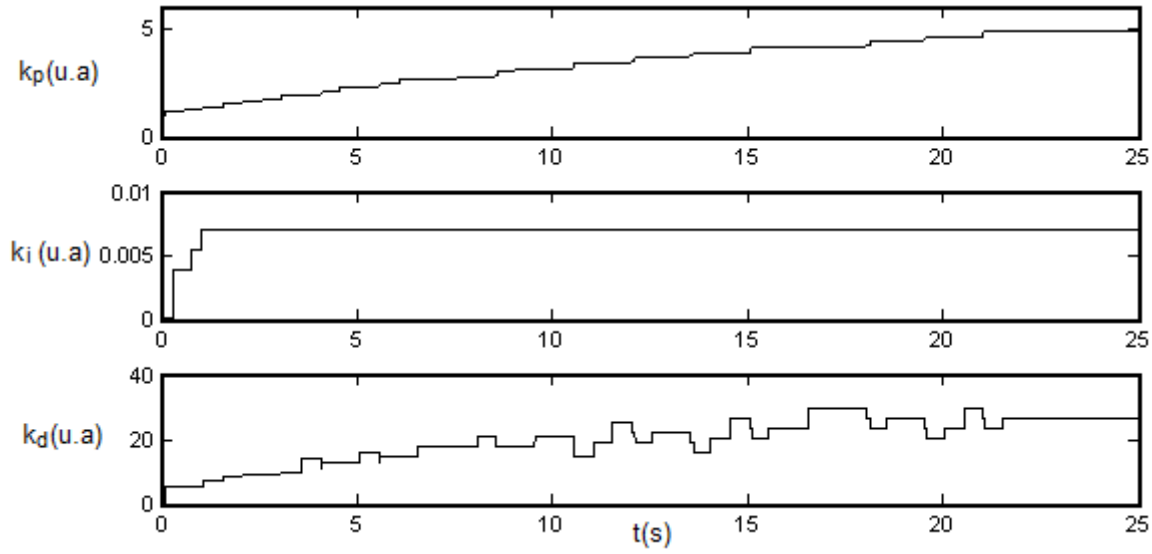


Figura 135. Evolución de los parámetros k_p, k_i y k_d del controlador PIDAD2 a la entrada escalón multinivel periódico aplicado a la planta 2



Convergencia de los parámetros

$k_p = 4.927$ $k_i = 0.0070$ $k_d = 26.50$

Igual que en el caso anterior de la entrada escalón periódico unitario se nota que el controlador demora en nivelar los parámetros, no obstante consigue la nivelación obteniendo una respuesta óptima.

Figura 136. Comparación de las respuestas de la planta 2 a una entrada escalón unitario según los controladores aplicados

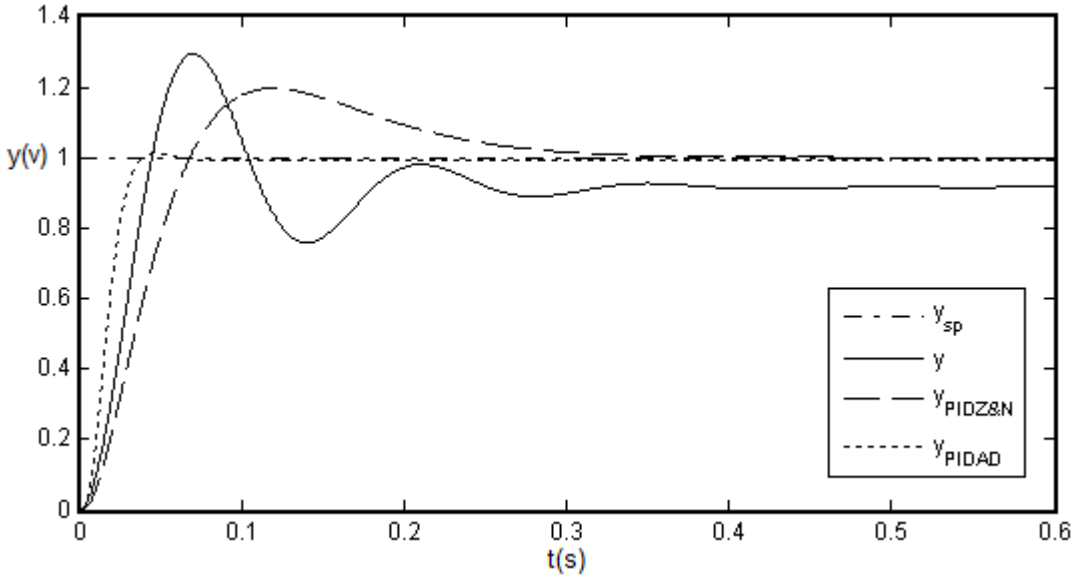


Tabla 44. Comparación cuantitativa de la respuesta transitoria de la planta 2 a una entrada escalón multinivel periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento 5% $t_a(s)$**	**Error en estado estable e_{ss}**	**Sobrepaso máximo ov**	**Tiempo de subida $t_s(s)$**
Sin control	0.2730	0.0840	38.00%	0.0385
PID Z&N	0.2397	0.0004	19.69%	0.0591
PIDAD	0.0333	0.0013	1.30%	0.0292

5.1.3.9 PIDAD2 aplicado planta 2 a una entrada escalón y perturbación. Se ha aplicado una perturbación de 0.5 de magnitud en el tiempo 0.3 segundos a los controladores de referencia y al controlador PIDAD2 con los parámetros en los que ha convergido.

Figura 137. Comparación de las respuestas de la planta 2 a una entrada escalón unitario con perturbación según los controladores aplicados

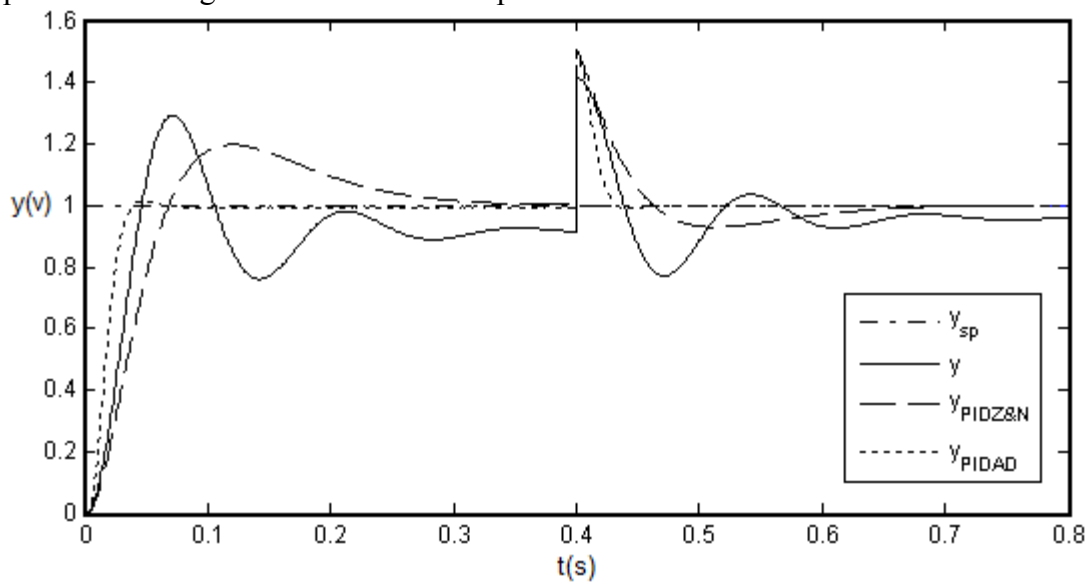


Tabla 45. Criterio ITAE de la planta 2 cuando este es operado por cada controlador

Controlador	**ITAE (0.4s)**	**ITAE (0.8s)**
Sin control	0.0081	0.0255
PID Z&N	0.0046	0.0152
PIDAD	0.0006	0.0042

En este caso se puede observar que los parámetros que el controlador PIDAD2 ha balanceado responden de una manera optima en comparación con el controlador PIDZ&N, debido a que su eficiencia mejora en 13 veces respecto a la planta sin control 7 veces respecto al controlador PIDZ&N hasta los 4 segundos antes de la perturbación y hasta el segundo 8 supera casi 4 veces la eficiencia del controlador PIDZ&N.

5.1.3.10 PIDAD2 aplicado a la planta 2 a una entrada rampa periódica y seno. Las señales de rampa periódica y seno, no son entradas que se utilizan regularmente en los procesos, aun así resulta interesante observar el comportamiento de la planta con el controlador PIDAD2.

Figura 138. Respuesta de la planta 2 a una entrada rampa periódica con un controlador PIDAD2

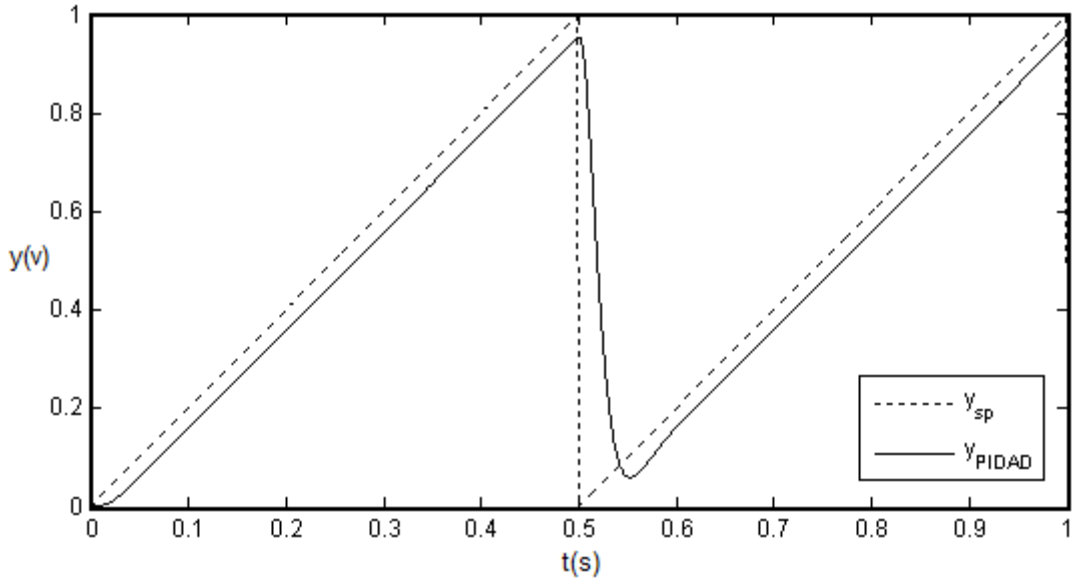
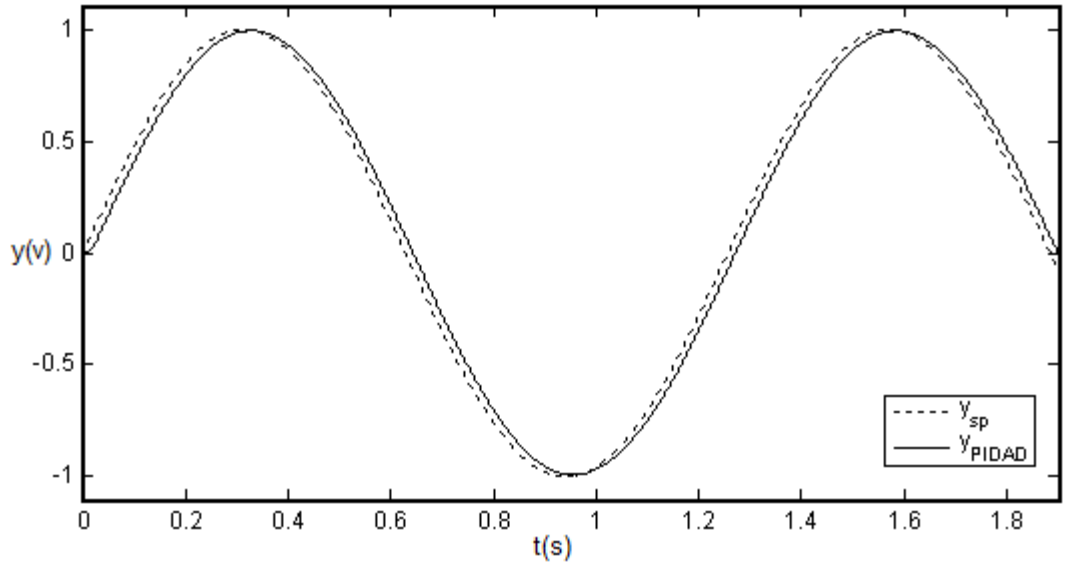


Figura 139. Respuesta de la planta 2 a una entrada seno con un controlador PIDAD2



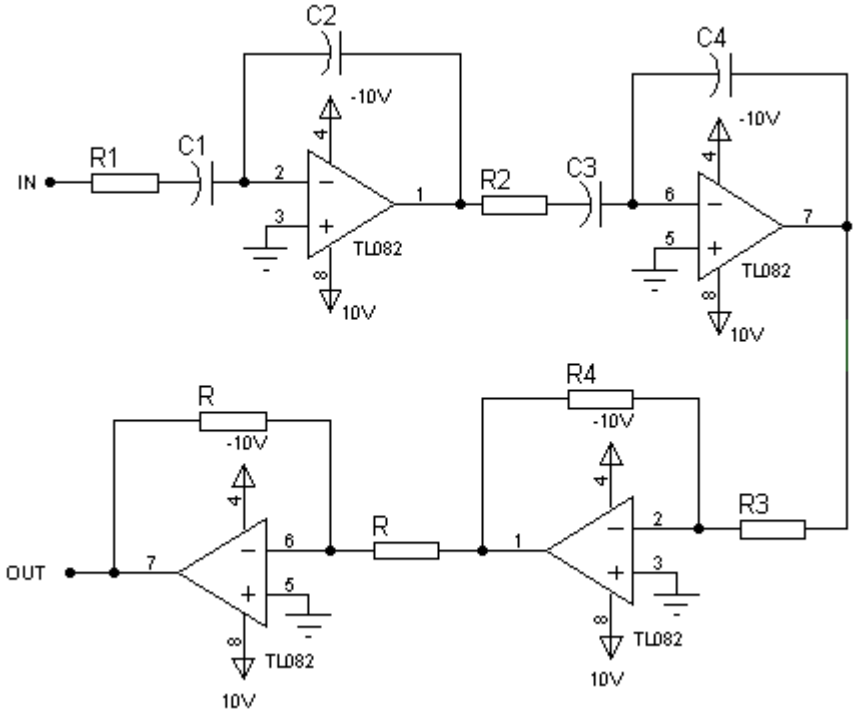
Al igual que en el anterior proceso, el controlador PIDAD2 muestra un buen comportamiento ante entradas rampa y seno, ya que muestra un seguimiento aceptable. Como ya se menciono, estas pruebas solo son para observar el comportamiento como fin teórico, ya que en la planta real se encuentra una señal de referencia de este tipo en muy pocos casos.

5.2 EMULACIONES Y ANALISIS DE RESULTADOS

5.2.1 Construcción de la planta para la emulación. Las plantas que se propusieron para la simulación son las más adecuadas para ser emuladas, se ha construido un modelo físico a base de amplificadores operaciones, componentes pasivos y activos. La siguiente ecuación muestra la función de transferencia del circuito que representa el modelo físico.

$$G(s) = \frac{1}{R_1 R_2 C_2 C_4} \times \frac{1}{s + 1/R_1 C_1} \times \frac{1}{s + 1/R_2 C_3} \times \frac{R_4 R}{R_3 R} \qquad (77)$$

Figura 140. Modelo físico de una planta construida con amplificadores operacionales y componentes activos y pasivos



Fácilmente cambiando algunos componentes se puede variar la función de transferencia G(s).

$$G(s) = \frac{\alpha}{(s + p_1)(s + p_2)} \qquad (78)$$

$$\alpha = \frac{R_4}{R_1 R_2 R_3 C_2 C_4} \tag{79}$$

$$p_1 = 1/R_1 C_1 \tag{80}$$

$$p_2 = 1/R_2 C_3 \tag{81}$$

En la siguiente tabla se listan los componentes específicos para lograr la función de transferencia de las plantas simuladas que se han emulado.

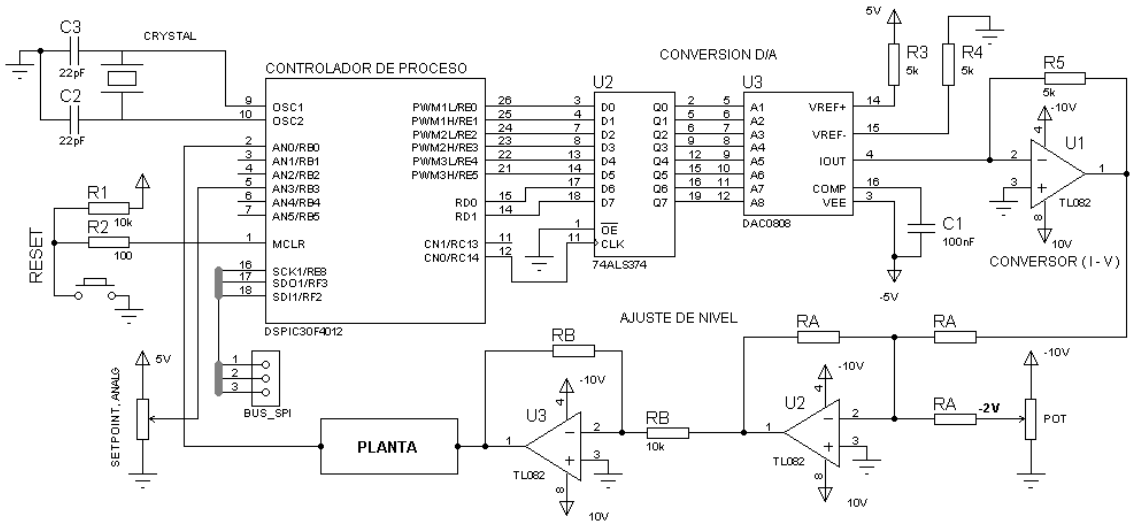
Tabla 46. Componentes para la construcción del modelo físico de las plantas

	R_1	R_2	R_3	R_4	C_1	C_2	C_3	C_4
Planta 1	449Ω	437Ω	4KΩ	11.78Ω	91.14uF	42.32uF	77.50uF	95.66uF
Planta 2	449Ω	437Ω	4KΩ	11.78Ω	182.10uF	80.07uF	155.32uF	95.66uF

La emulación es un equivalente a una implementación en la investigación, esta facilita trabajar con el proceso como si fuera el real ya que el modelo físico es equivalente, en este caso lo componentes que almacenan energía que son los condensadores en conjunto con los amplificadores operacionales, logran que el comportamiento de efecto transitorio sea el mismo del real pues estos permiten generar los polos de la función de transferencia. La emulación tiene una ventaja en la investigación y es que permite trabajar los detalles y ajustes necesarios del controlador de manera fácil y rápida, ya que se necesitan elementos de trabajo mínimos y no se necesita trabajar la investigación en el lugar donde la planta está instalada.

En la siguiente figura se puede observar el controlador de proceso, este puede ejecutar los algoritmos PIDAD1 y PIDAD2, los cuales utilizan el mismo circuito de implementación del controlador de proceso diferenciándose solamente en el programa del controlador. Se pueden identificar los componentes básicos que conforman el sistema que son el circuito de reinicio, de oscilación, de conversión digital analógica que actúa como acondicionador de señal y actuador, el circuito de ajuste de nivel que permite al controlador a generar voltajes negativos para la señal de control.
También cuenta con el conector de bus SPI para comunicarse con la interfaz de usuario que permite programar las señales de referencia.

Figura 141. Circuito del controlador de proceso utilizado en la emulación



- **Tarjeta de adquisición de señal NI USB-6008.** Esta tarjeta fabricada por la empresa National Instrumets (NI) se utilizo para adquirir la señal de salida de la planta. Tiene 8 entradas analógicas a 12 bits con un rango de voltaje de [±1 , ±20], el muestreo puede ser de 48KS/s, también cuenta con 12 entradas y salidas digitales, un contador de 32 bits y posee la opción de compuerta digital de disparo. Trabaja con el software de LabView aunque también tiene compatibilidad con Visual Basic .NET. Esta tarjeta de NI es una mejora e innovación en el campo de las aplicaciones de medición, ya que cuenta con una característica que la hace muy versátil y fácil de usar que es la conexión USB. Entre las aplicaciones recomendadas están el uso como grabador de datos (*data logging*), para uso académico en laboratorios o para entrada y salida de datos de sistema embebidos.

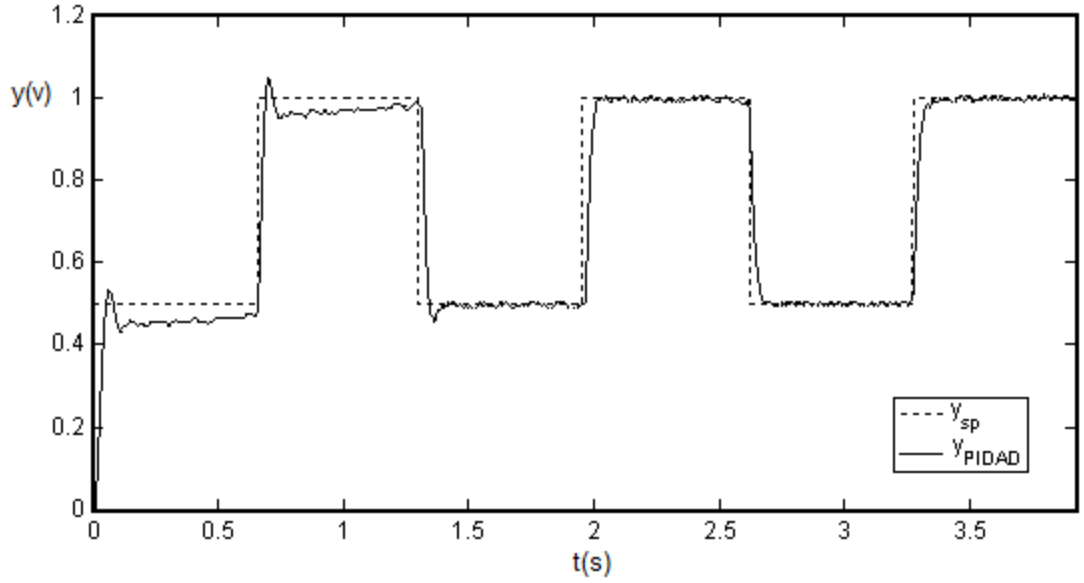
5.2.2 Pruebas de emulación del controlador PID con MAD1 (PIDAD1). El mecanismo difuso adaptivo 1 se utiliza como herramienta para sintonizar el controlador PID. El controlador PIDAD1 se aplica a la planta 1 con una función de transferencia típica de un motor, para ejecutar la rutina de sintonía la planta se somete a una función escalón unitario periódica en la cual el MAD1 dependiendo de la respuesta transitoria trata de llevarla hacia un rango de operación adecuado balanceando los parámetros del controlador PID. Es importante mencionar que los parámetros con los que parte el controlador, casi anulan la acción integral y derivativa, los valores son:

$k_p = 1$ $k_i = 0.0001$ $k_d = 0.0001$

5.2.2.1 Emulación del PIDAD1 aplicado a la planta 1. El escalón periódico unitario es la señal adecuada para realizar la rutina de sintonía del controlador, el sistema trata de llevar

la respuesta al escalón a unos límites de trabajo aceptables, reduciendo el error en estado estable y el sobrepaso.

Figura 142. Respuesta de la planta 1 con controlador PIDAD1 a una entrada escalón unitario periódico



Convergencia de los parámetros

$k_p = 2.320$ $k_i = 0.0104$ $k_d = 18.13$

De la grafica se observa que el sistema responde de forma muy satisfactoria incluso, se observa que se tiene una convergencia de parámetros en el cuarto transitorio, donde se ha corregido el error en estado estable y el sobrepaso.

Figura 143. Comparación de las respuestas de la planta 1 según los controladores aplicados

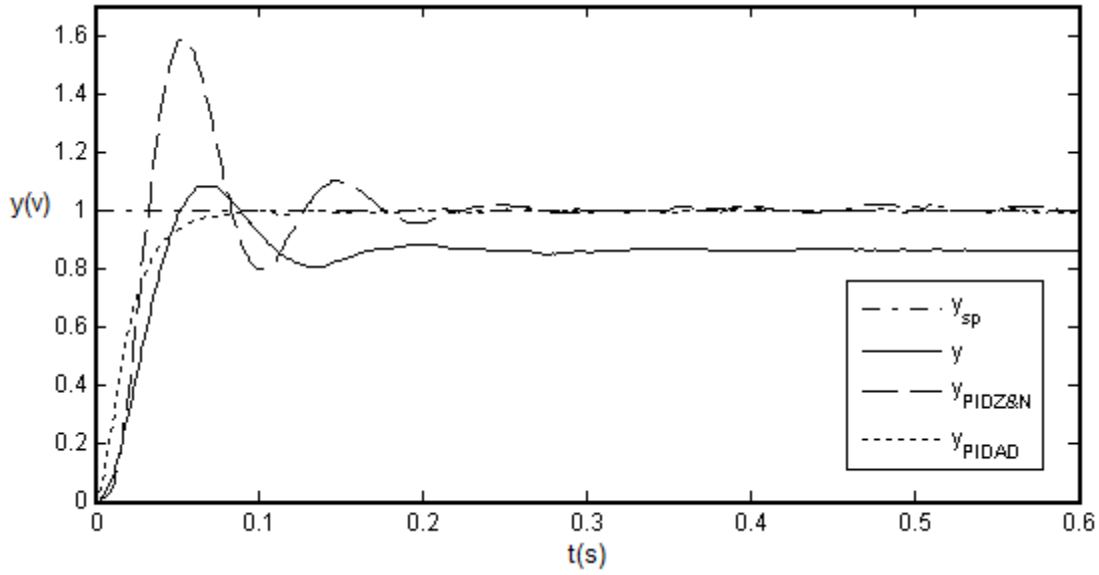


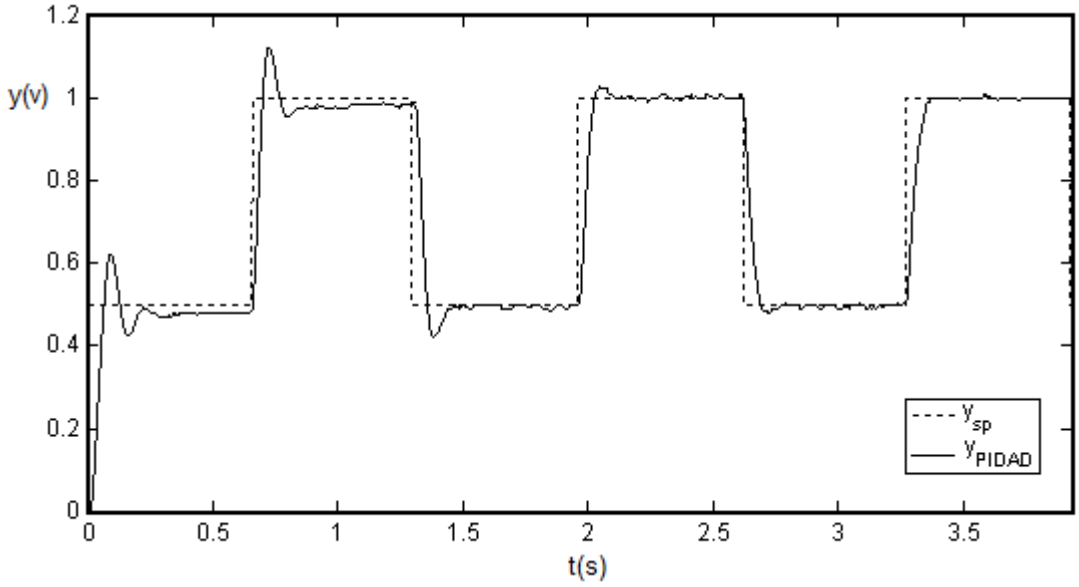
Tabla 47. Comparación cuantitativa de la respuesta transitoria de la planta 2 a una entrada escalón multinivel periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento 5% $t_a(s)$**	**Error en estado estable e_{ss}**	**Sobrepaso máximo ov**	**Tiempo de subida $t_s(s)$**
Sin control	0.1533	0.0840	38.00%	0.0385
PID Z&N	0.2397	0.0004	19.69%	0.0591
PIDAD	0.0581	0.0004	0.520%	0.0430

El controlador PIDAD1 muestra una respuesta muy satisfactoria por encima del controlador PIDZ&N pues reduce el sobrepaso al 0,5% y corrige el error en estado estable, sin perder velocidad de respuesta en cuanto al tiempo de subida.

5.2.2.2 Emulación del PIDAD1 aplicado a la planta 2. Se ejecuta la misma rutina de sintonía para la planta 2, el escalón periódico unitario resulta adecuado para brindar la experiencia de varios transitorios.

Figura 144. Respuesta de la planta 2 con controlador PIDAD1 a una entrada escalón unitario periódico



Convergencia de los parámetros

$k_p = 1.960$ $k_i = 0.0057$ $k_d = 26.29$

Se puede observar que el controlador estabiliza los parámetros en el sexto transitorio, esto se debe a que se ha aumentado la dificultad en la planta 2 respecto la planta 1, sin embargo el sistema es capaz de corregir de forma muy satisfactoria el error en estado estable y el sobrepaso.

Figura 145. Comparación de las respuestas de la planta 1 según los controladores aplicados

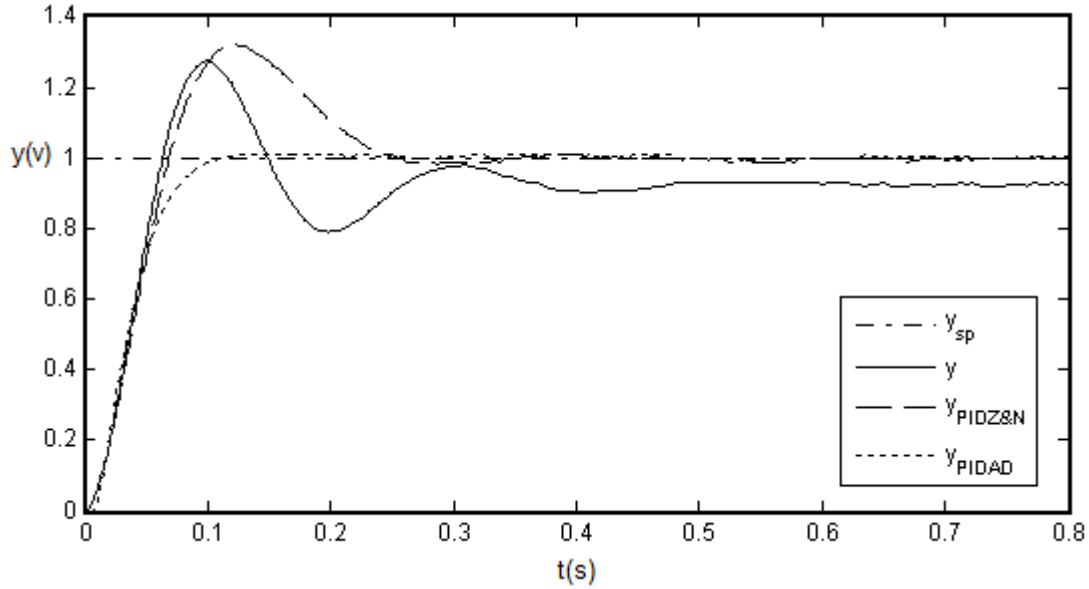


Tabla 48. Comparación cuantitativa de la respuesta transitoria de la planta 2 a una entrada escalón multinivel periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\,(s)$
Sin control	0.3413	0.0840	38.00%	0.0385
PID Z&N	0.2397	0.0004	19.69%	0.0591
PIDAD	0.0871	0.0088	1.07%	0.0720

De igual forma el resultado de la respuesta de controlador PIDAD1 está por encima del controlador PIDZ&N, debido a que reduce el sobrepaso a 1% y el error en estado estable a los niveles muy bajos y permitidos para los procesos industriales, el tiempo de subida se aumenta un poco respecto al controlador PIDZ&N esto debido a que el sistema tiene como prioridad el manejo del error en estado estable y el sobrepaso.

5.2.3 Pruebas de emulación del controlador PID con MAD2 (PIDAD2). El mecanismo difuso adaptivo 2 se puede utilizar junto con el controlador PID como un controlador adaptivo, dicho de otra forma, es capaz de adaptarse a la dinámica del proceso según este lo requiera. Este controlador se prueba con varias señales de entrada, partiendo con los parámetros sin sintonizar:

$k_p = 1$ $k_i = 0.0001$ $k_d = 0.0001$

El MAD2 al igual que el anterior trata de llevar la respuesta transitoria a un rango de operación adecuado variando los parámetros, cuando ya no se haga necesario la variación de estos parámetros estos convergen en un valor.

5.2.3.1 PIDAD2 aplicado a la planta 1 a una entrada escalón unitario. La señal de entrada del sistema en este caso es un escalón unitario, esto significa que el mecanismo adaptivo solo experimenta un transitorio.

Figura 146. Comparación de las respuestas de la planta 1 según los controladores aplicados a una entrada escalón unitario

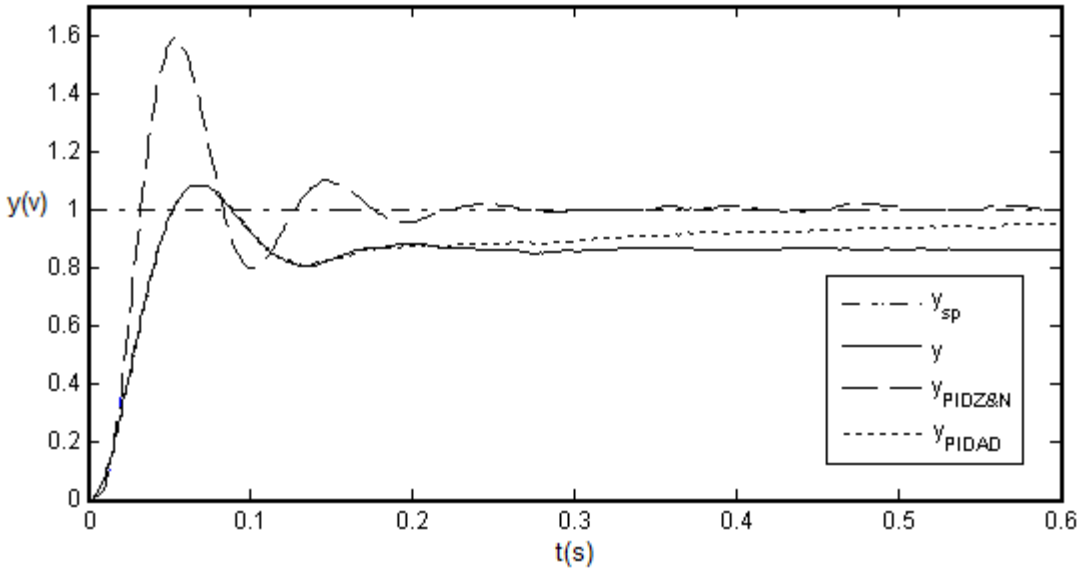
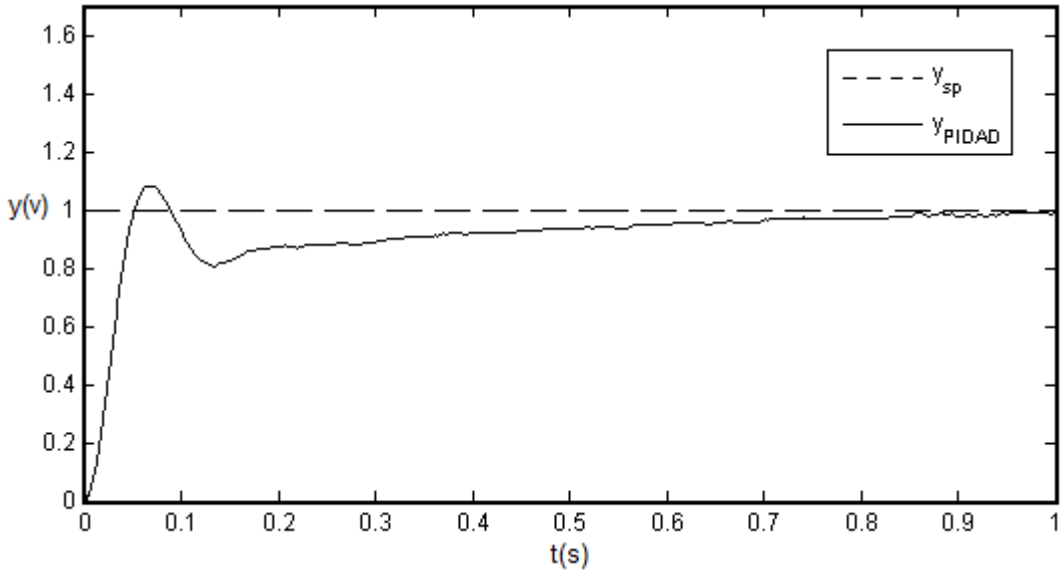


Tabla 49. Comparación cuantitativa de la respuesta transitoria de la planta 1 según los controladores aplicados a una entrada escalón unitario

Controlador	**Tiempo de asentamiento 5% $t_a(s)$**	**Error en estado estable e_{ss}**	**Sobrepaso máximo ov**	**Tiempo de subida $t_s(s)$**
Sin control	0.1330	0.1420	26.42%	0.0384
PID Z&N	0.1641	0.0009	59.15%	0.0310
PIDAD	0.6494	0.0009	8.26%	0.0450

Figura 147. Respuesta de la planta 1 con controlador PIDAD1 a una entrada escalón unitario



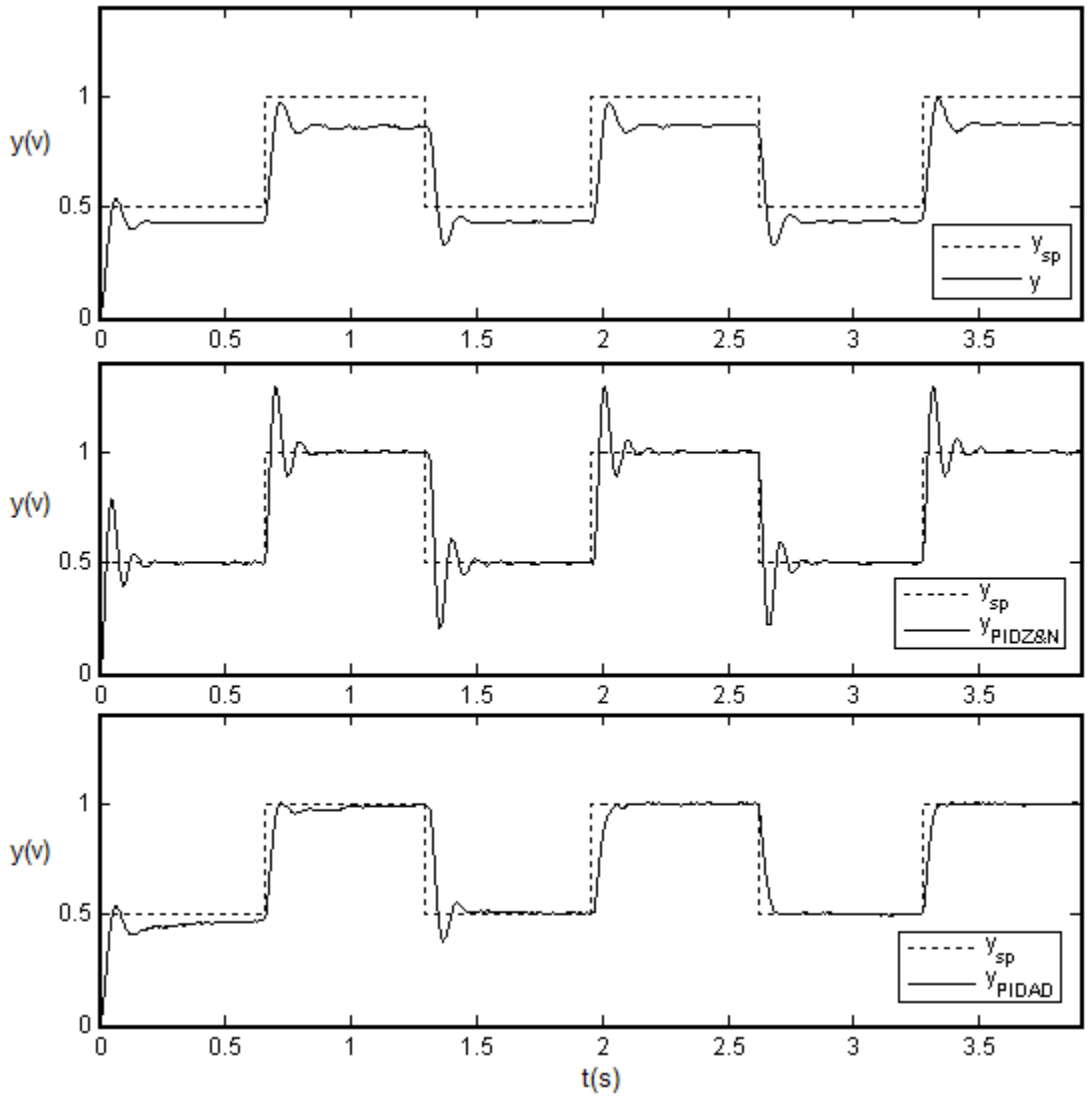
Convergencia de los parámetros

$k_p = 1.310$ $k_i = 0.0078$ $k_d = 4.580$

El error en estado estable se corrige completamente en el tiempo 1 segundo, mostrando un comportamiento adaptivo. Como se esperaba el error en estado estable se corrige completamente y se reduce el sobrepaso, cabe destacar que el resultado emulación se asemeja mucho a la simulación, ya que la forma de onda de la salida del proceso muestra el mismo comportamiento que la vista en la simulación.

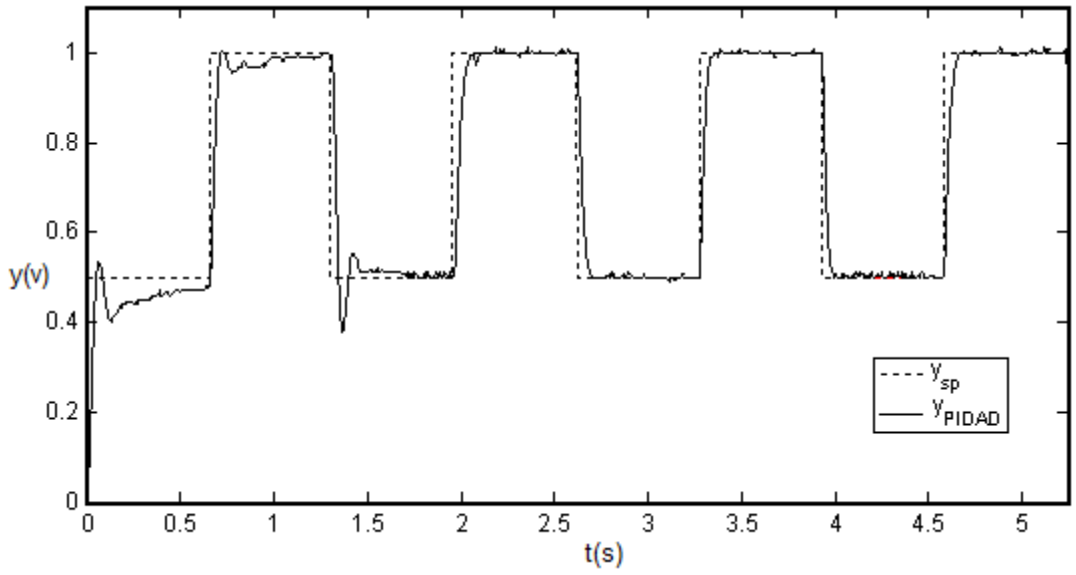
5.2.3.2 PIDAD2 aplicado a la planta 1 a una entrada escalón unitario periódico. La entrada del sistema es un escalón periódico unitario, el controlador parte con las mismas condiciones iníciales anteriores. Se hace pruebas para el controlador PIDZ&N con el fin de tener un punto de referencia, así como también a la planta sin control.

Figura 148. Respuestas de la planta 1 a una entrada escalón unitario periódico según cada controlador



El controlador PIDAD2 es capaz de llevar el sobrepaso dentro de los límites permitidos en el cuarto transitorio como también el error en estado estable.

Figura 149. Respuesta de la planta 1 con un controlador PIDAD2 a una entrada escalón unitario periódico



Convergencia de los parámetros

$k_p = 2.430$ $k_i = 0.0107$ $k_d = 16.57$

Figura 150. Comparación de las respuestas de la planta 1 a una entrada escalón unitario según los controladores aplicados

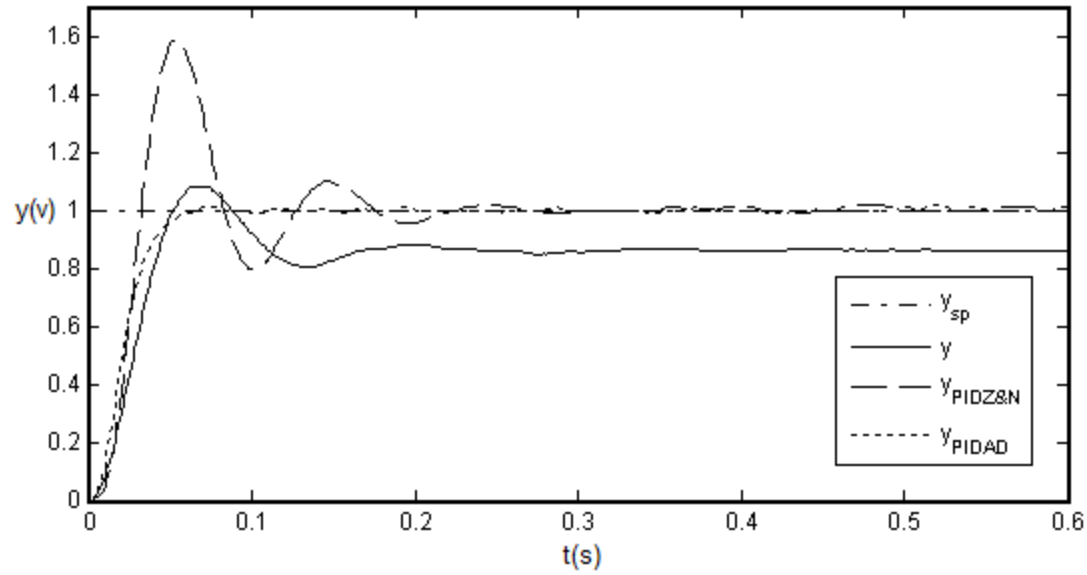


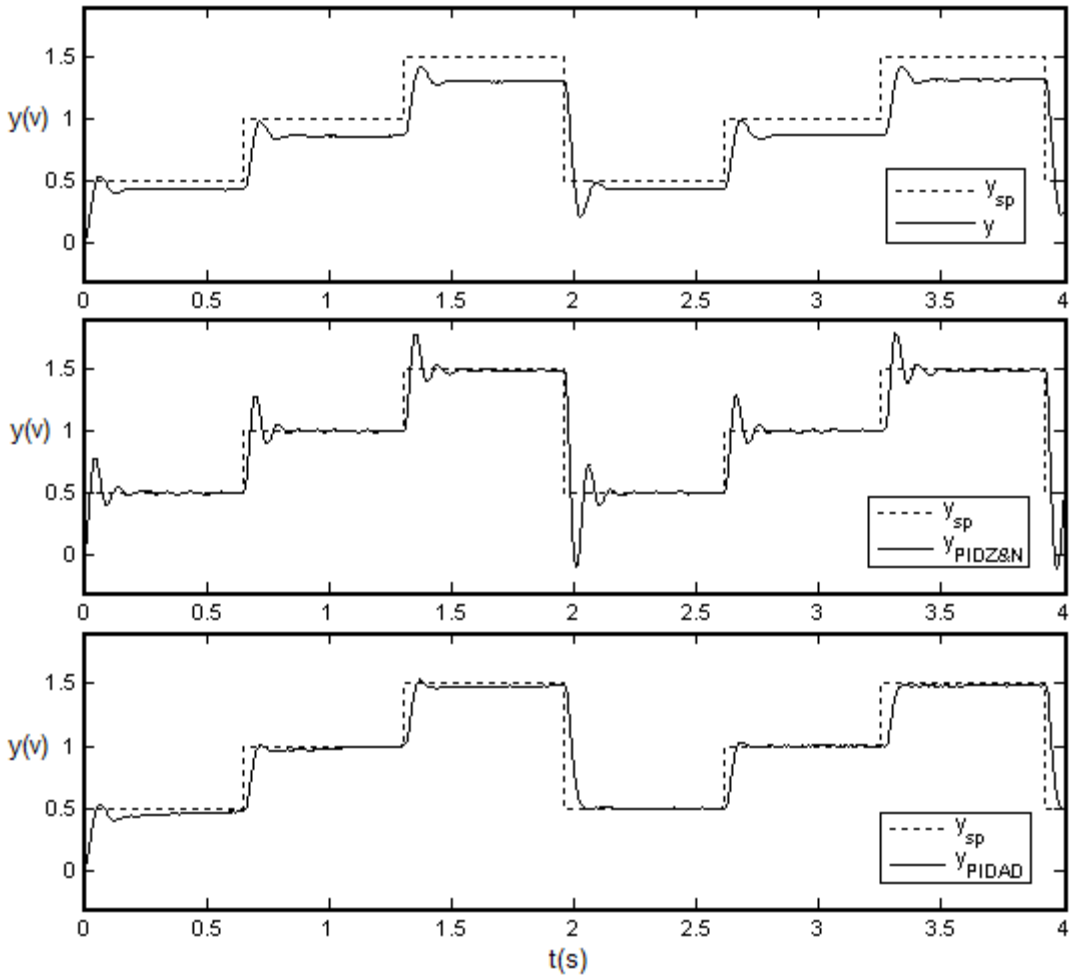
Tabla 50. Comparación cuantitativa de la respuesta transitoria de la planta 1 a una entrada escalón unitario periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s(s)$
Sin control	0.1330	0.1420	26.42%	0.0384
PID Z&N	0.1641	0.0009	59.15%	0.0310
PIDAD	0.0490	0.0009	1.93%	0.0410

La respuesta al escalón del proceso con el controlador PIDAD2 es optima en comparación con el controlador PIDZ&N, ya que reduce el sobrepaso menor al 2% y corrige el error en estado estable sin alterar su velocidad de respuesta o tiempo de subida de forma considerable.

5.2.3.3 PIDAD2 aplicado a la planta 1 a una entrada escalón multinivel periódico. La entrada del sistema es un escalón multinivel periódico, el controlador parte con las mismas condiciones iníciales, se toman como controladores de referencia el proceso sin control y el controlador PIDZ&N.

Figura 151. Respuestas de la planta 1 a una entrada escalón multinivel periódico según cada controlador



Convergencia de los parámetros

$k_p = 2.31$ $k_i = 0.0167$ $k_d = 22.49$

De igual forma que en la anterior prueba el controlador PIDAD2 tiene una respuesta de transitorio mucho mejor que el controlador PIDZ&N, en este caso en particular se ha obtenido un sobrepaso de 5% y un error en estado estable dentro de los limites, esta

situación del sobrepaso al 5% puede surgir si los parámetros varían y no logran disminuirlo, el sistema decide converger para no desestabilizar el controlador.

Figura 152. Comparación de las respuestas de la planta 1 a una entrada escalón unitario según los controladores aplicados

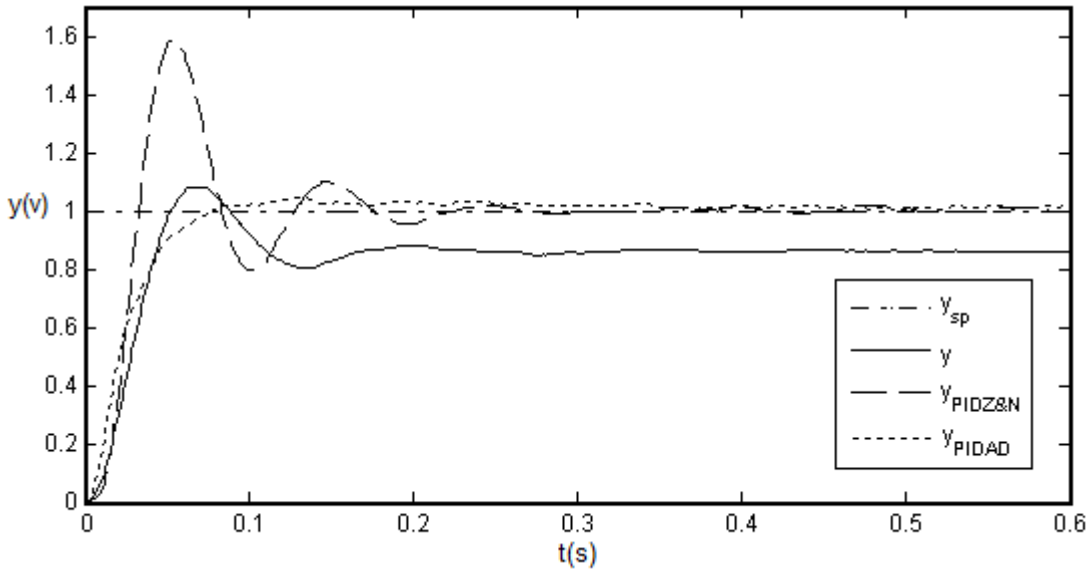


Tabla 51. Comparación cuantitativa de la respuesta transitoria de la planta 1 a una entrada escalón multinivel periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s(s)$
Sin control	0.1330	0.1420	26.42%	0.0384
PID Z&N	0.1641	0.0009	59.15%	0.0310
PIDAD	0.1320	0.0005	5.19%	0.0500

5.2.3.4 PIDAD2 aplicado a la planta 2 a una entrada escalón unitario. La señal de entrada del sistema en este caso es un escalón unitario, esto significa que el mecanismo adaptivo solo experimenta un transitorio.

Figura 153. Comparación de las respuestas de la planta 2 según los controladores aplicados a una entrada escalón unitario

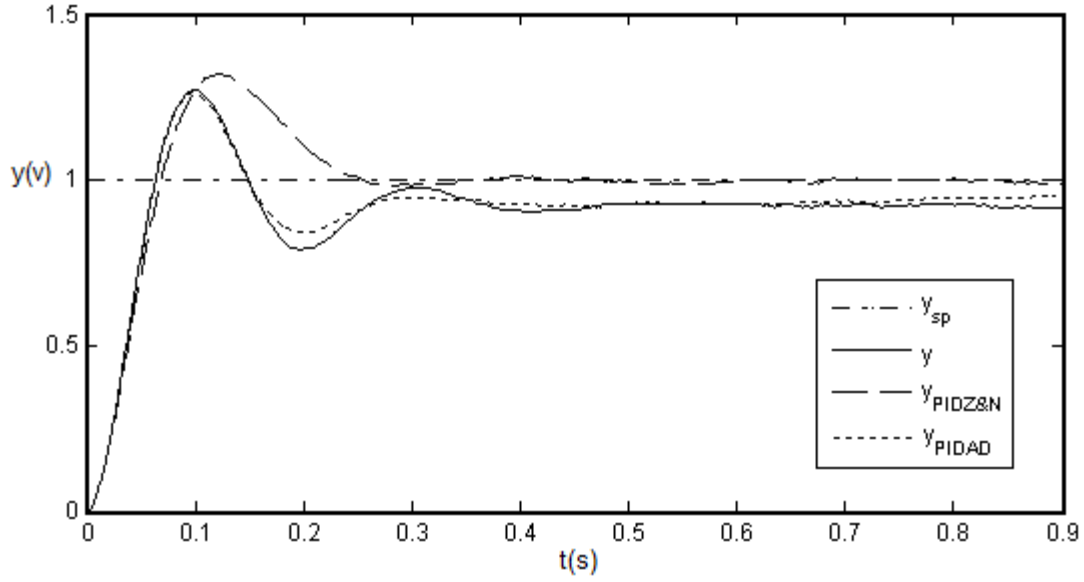


Figura 154. Respuesta de la planta 2 con controlador PIDAD2 a una entrada escalón unitario

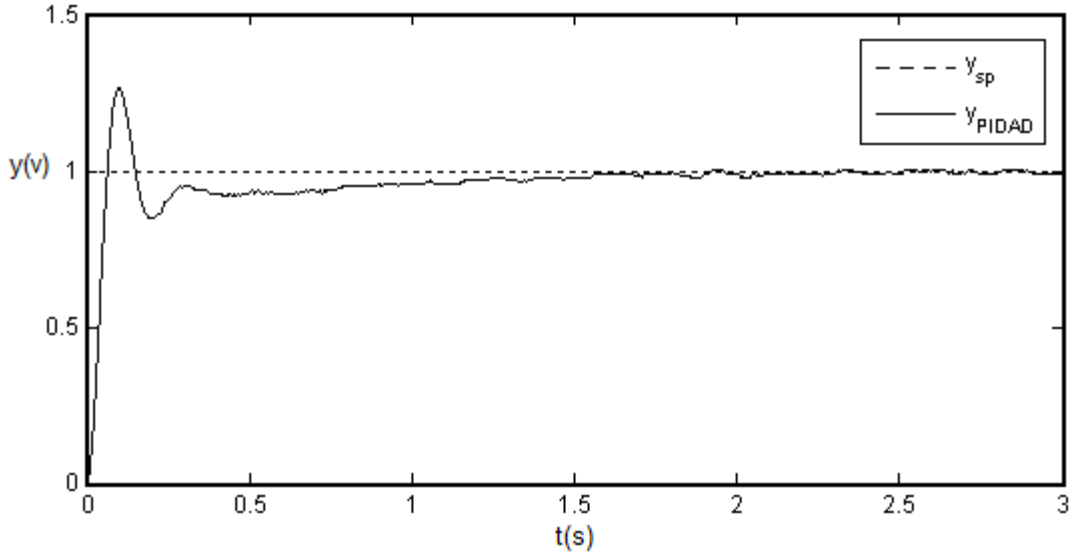


Tabla 52. Comparación cuantitativa de la respuesta transitoria de la planta 2 según los controladores aplicados a una entrada escalón unitario

Controlador	Tiempo de asentamiento (5%) $t_a(s)$	Error en estado estable e_{ss}	Sobrepaso máximo ov	Tiempo de subida $t_s\ (s)$
Sin control	0.3200	0.0715	37.41%	0.0569
PID Z&N	0.2221	0.0001	32.08%	0.0620
PIDAD	1.1463	0.0020	26.56%	0.0570

El controlador responde de forma satisfactoria corrigiendo el error en estado estable alrededor de los 1.5 segundos, observando un comportamiento adaptivo.

5.2.3.5 PIDAD2 aplicado a la planta 2 a una entrada escalón unitario periódico. La entrada del sistema es un escalón periódico unitario, el controlador parte con las mismas condiciones iníciales. Se toman como controladores de referencia el proceso sin control y el controlador PIDZ&N.

Figura 155. Respuestas de la planta 2 a una entrada escalón unitario periódico según cada controlador

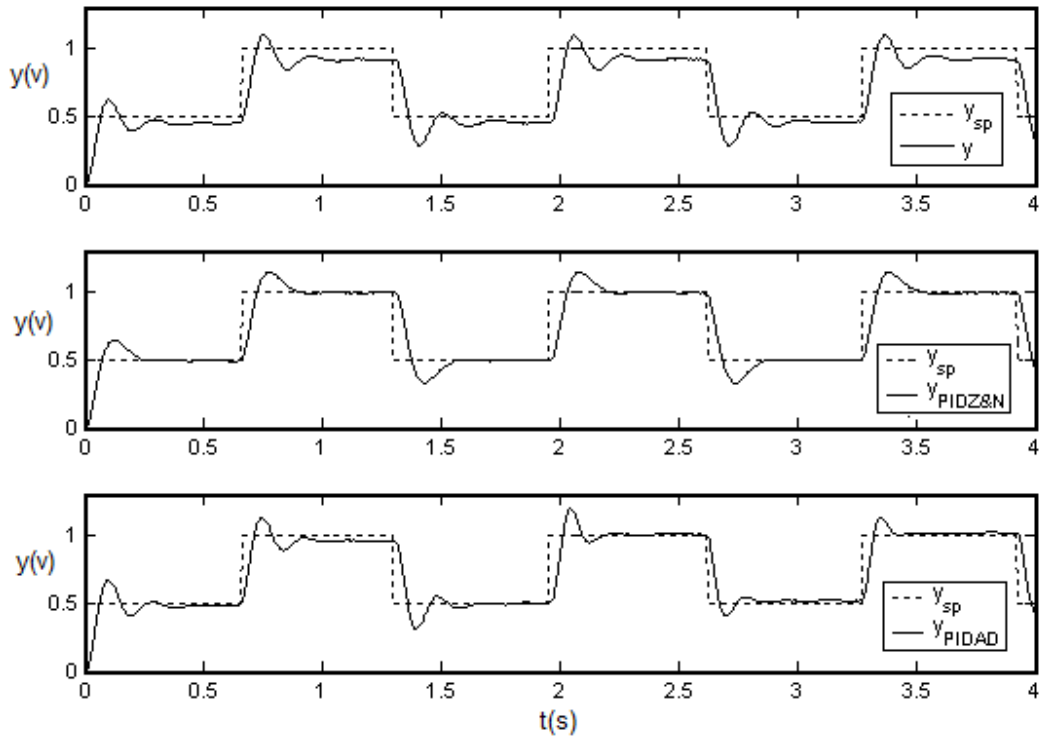
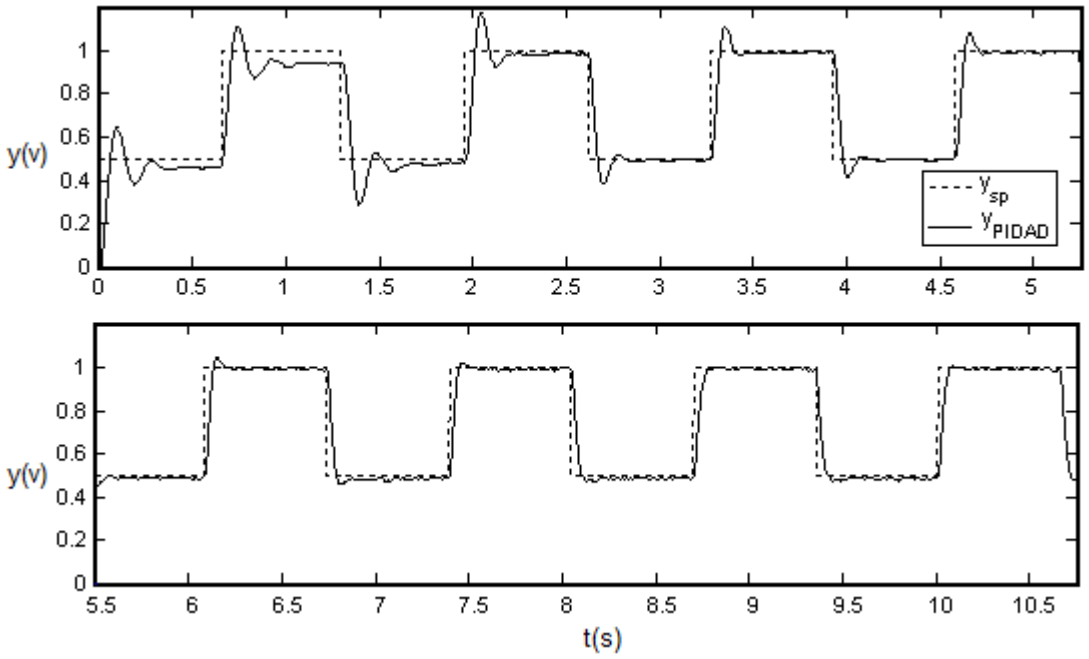


Figura 156. Respuesta de la planta 2 con un controlador PIDAD2 a una entrada escalón unitario periódico



Convergencia de los parámetros

$k_p = 2.74$ $k_i = 0.0080$ $k_d = 29.76$

Debido a la dificultad que conlleva el control de la planta 2, el sistema tarda un poco en balancear los parámetros, sin embargo logra llevar la respuesta del proceso dentro de los límites establecidos.

Figura 157. Comparación de las respuestas de la planta 2 a una entrada escalón unitario según los controladores aplicados

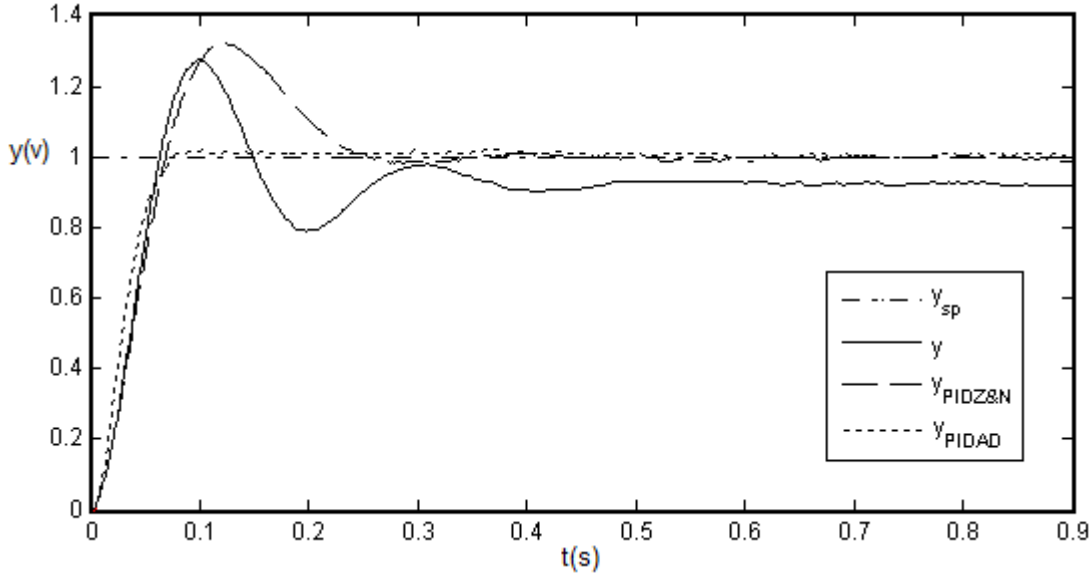


Tabla 53. Comparación cuantitativa de la respuesta transitoria de la planta 2 a una entrada escalón unitario periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento 5%** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\ (s)$
Sin control	0.3200	0.0715	37.41%	0.0569
PID Z&N	0.2221	0.0001	32.08%	0.0620
PIDAD	0.0660	0.0001	2.03%	0.0570

Sin importar la dificultad del proceso el controlador PIDAD2 consigue llevar el sobrepaso y el error en estado dentro de los límites permitidos, claramente se observa que el controlador tiene una respuesta optima por encima del controlador PIDZ&N incluso mejorando el tiempo de subida.

5.2.3.6 PIDAD2 aplicado a la planta 2 a una entrada escalón multinivel periódico. La entrada del sistema es un escalón multinivel periódico, el controlador parte con las mismas condiciones iníciales. Se toman como controladores de referencia el proceso sin control y el controlador PIDZ&N

Figura 158. Respuestas de la planta 2 a una entrada escalón multinivel periódico según cada controlador

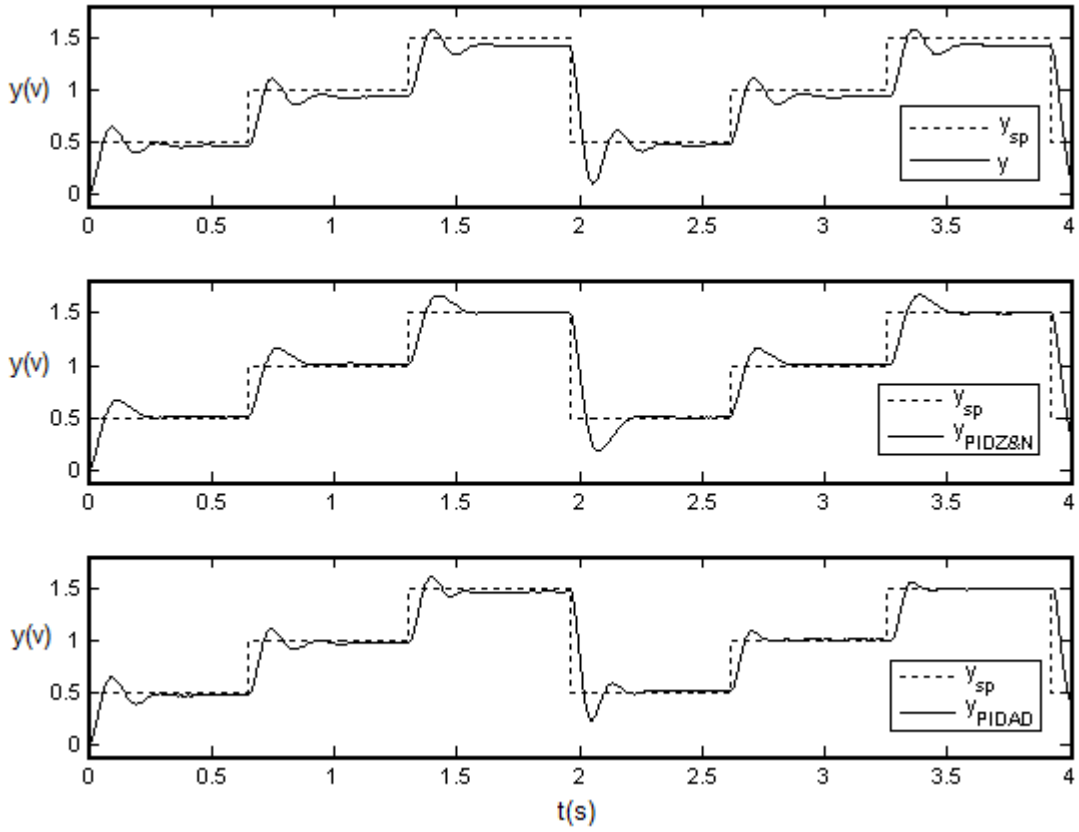
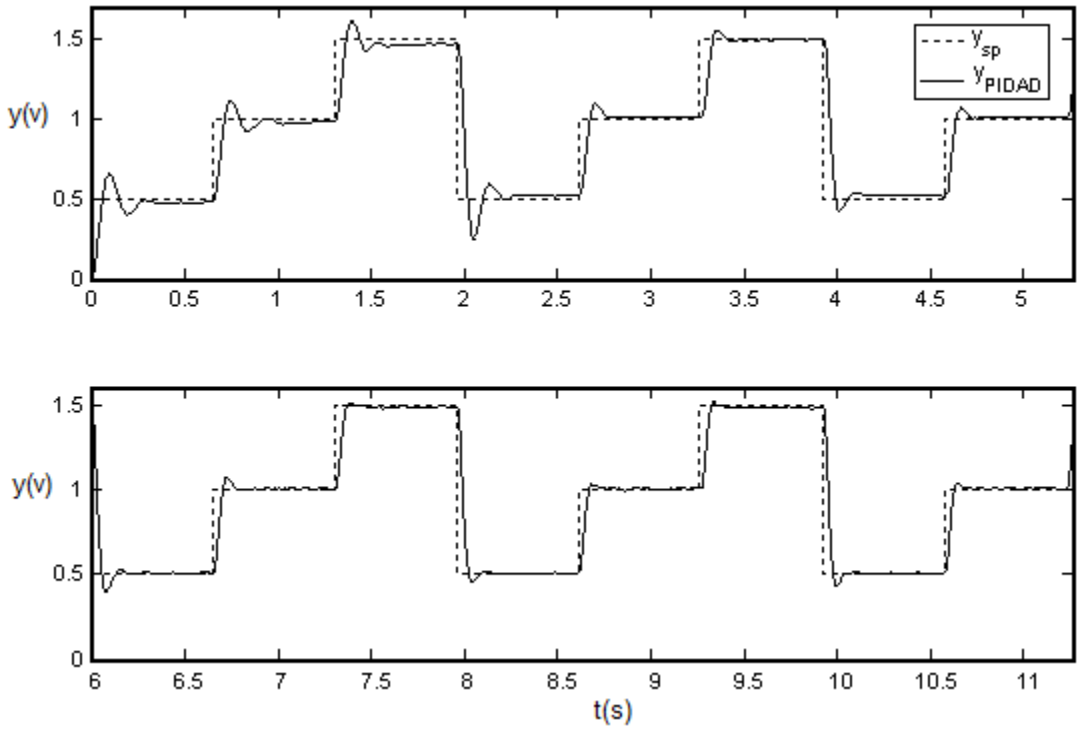


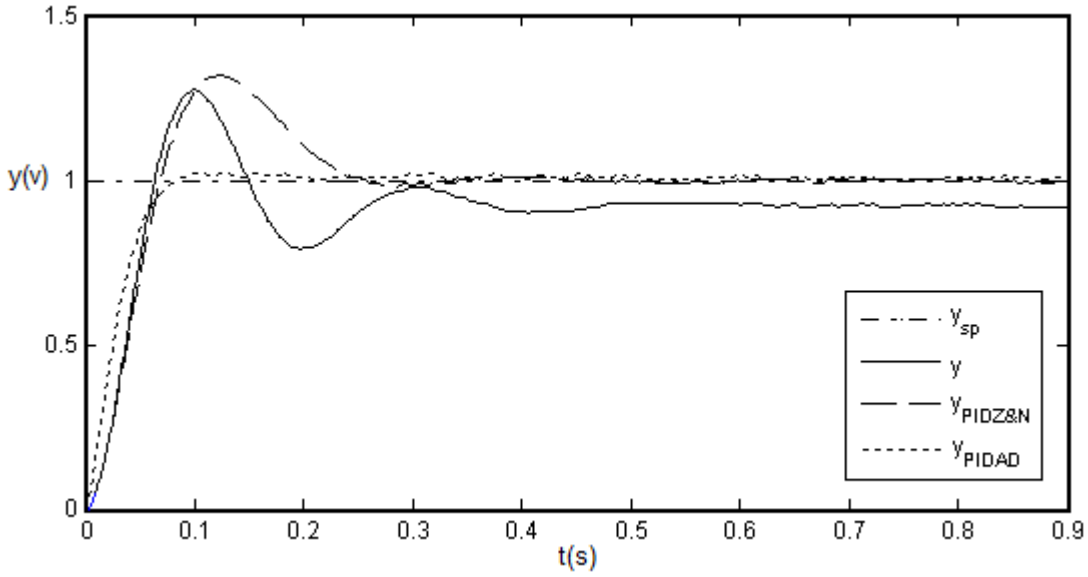
Figura 159. Respuesta de la planta 2 con un controlador PIDAD2 a una entrada escalón multinivel periódico



Convergencia de los parámetros

$k_p = 2.53$ $k_i = 0.0078$ $k_d = 30.12$

Figura 160. Comparación de las respuestas de la planta 2 a una entrada escalón unitario según los controladores aplicados



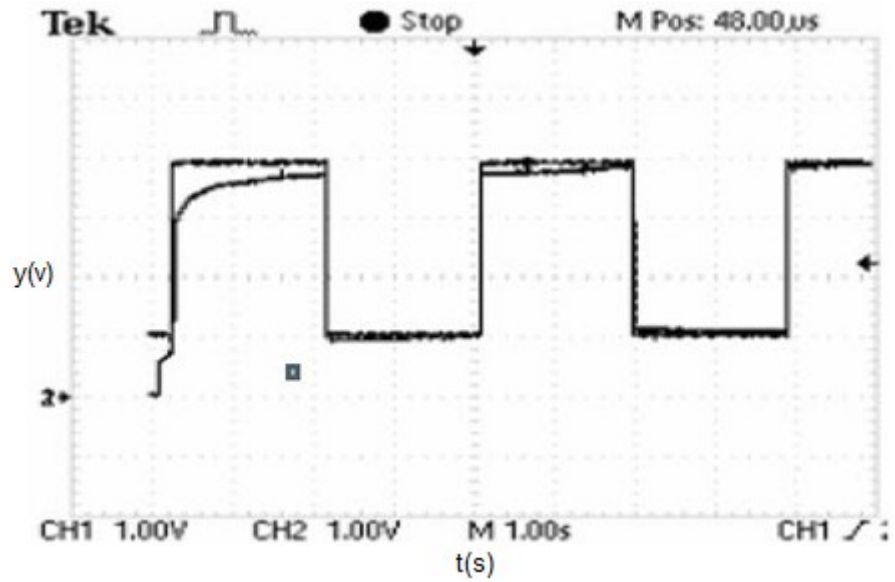
Como se esperaba el sistema tarda en balancear los parámetros debido a la dificultad que se ha aumentado en la planta 2, sin embargo el sistema consigue llevar la respuesta del proceso dentro de los límites permitidos y por encima del controlador PIDZ&N, reduciendo un sobrepaso a un 2%, eliminando el error en estado estable y mejorando el tiempo de subida.

Tabla 54. Comparación cuantitativa de la respuesta transitoria de la planta 2 a una entrada escalón unitario periódico según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\ (s)$
Sin control	0.3200	0.0715	37.41%	0.0569
PID Z&N	0.2221	0.0001	32.08%	0.0620
PIDAD	0.0680	0.0005	2.044%	0.0560

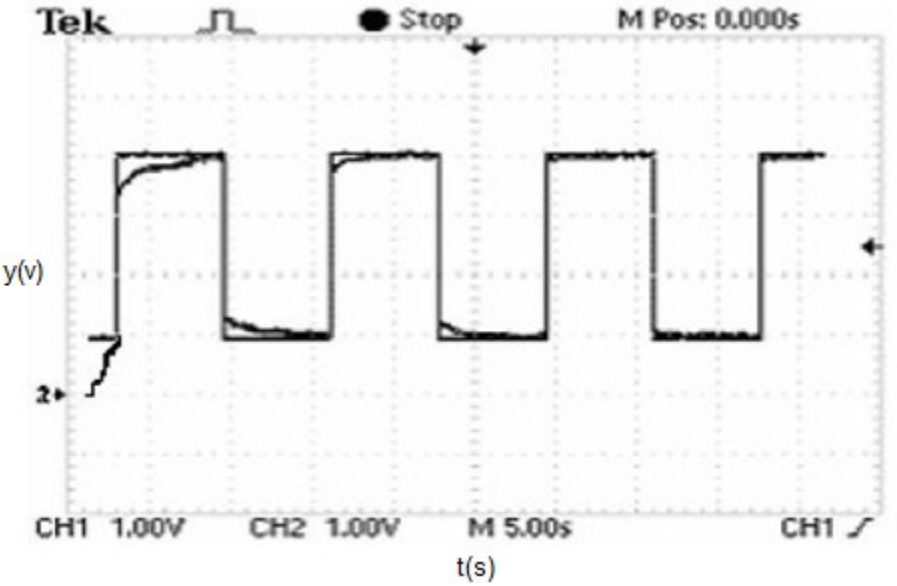
Las figuras que se muestra a continuación pertenecen a un trabajo muy similar al que se ha presentado en este documento, se trata de un controlador PID adaptivo que tiene un mecanismo adaptivo difuso de Takagi y Sugeno, diferente al utilizado.

Figura 161. Seguimiento de una señal de referencia rectangular en un proceso de primer orden



Fuente: CHAINHO, Joao; PEREIRA, Pedro; RAFAEL, Silviano y PIRES, A.J. *A simple PID controler with adaptive parameter in a dsPIC, 2005.*

Figura 162. Seguimiento de una señal de referencia rectangular en un proceso de segundo orden



Fuente: CHAINHO, Joao; PEREIRA, Pedro; RAFAEL, Silviano y PIRES, A.J. *A simple PID controler with adaptive parameter in a dsPIC, 2005.*

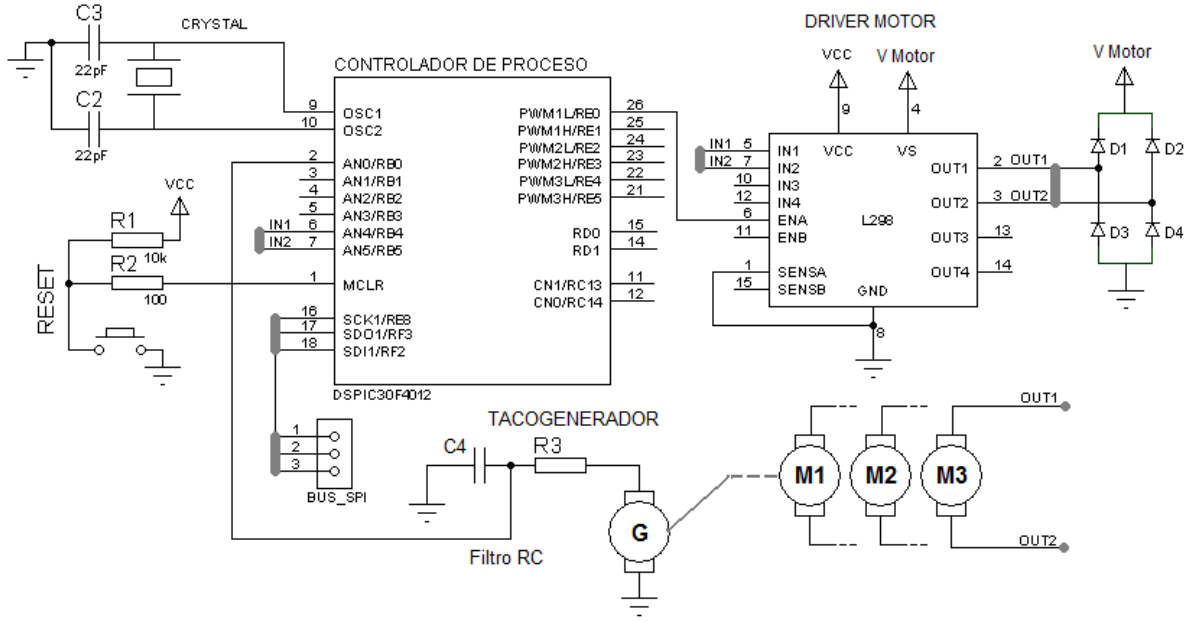
En las dos figuras anteriores se puede observar que el comportamiento del controlador es similar al planteado, al igual que el presentado también existe una relación entre la duración de la adaptación y la frecuencia de la señal de referencia. Como se puede observar el controlador adaptivo se termina adaptando a la señal, demostrando la eficiencia de este tipo de controladores.

5.3 APLICACIÓN Y ANALISIS DE RESULTADOS

Debido al elevado costo de los materiales para la implementación del actuador de un motor trifásico y la consecución mismo motor, se aplica el controlador desarrollado, al control de velocidad de un motor de baja potencia DC, se eligió esta aplicación porque cuenta con un modelo matemático muy similar, obteniendo una planta equivalente a la planta motor AC, con la ventaja de tener un menor costo de implementación. Se aplica el controlador a 3 motores DC de diferentes potencias, que manejan voltajes de 24VDC hasta 7VDC. Se propone el control de 3 motores de parámetros electro-mecánicos muy diferentes para observar la versatilidad del controlador propuesto y poder concluir acerca de su característica adaptiva.

Figura 163. Controlador PID adaptivo difuso PIDAD

Figura 164. Circuito de implementación del controlador PIDAD aplicado al control de velocidad de motores DC



El circuito está constituido por tres partes principales, el controlador de proceso, el driver de motor y el tacogenerador.

- **Driver de motor.** Es un manejador de motor DC con referencia L298 que maneja de hasta 184Watts cumpliendo con la necesidad de aislar la etapa de potencia con la etapa de control; cuenta con dos entradas de control compatible a niveles lógicos TTL que permiten inversión de giro y frenado rápido, además posee una entrada adicional llamada "*enable*" o habilitador que facilita el control de velocidad por PWM.

- **Tacogenerador.** Es un motor DC que se acopla a un eje para medir su velocidad angular, el principio de funcionamiento se basa en generar una corriente proporcional al movimiento de rotación a partir de la bobina ubicada en un campo magnético fijo, representándose en un voltaje, es recomendable aplicar un filtro RC para acondicionar la señal, el filtro aplicado cuenta con una constante de tiempo RC = 11.28mS. Posteriormente la señal es medida por el módulo ADC de 10 bits del controlador de proceso con una precisión ± 5 RPM (Revoluciones por minuto), un error de offset de ±19 RPM y un error de ganancia de ± 47 RPM de acuerdo a la hoja de datos del controlador utilizado. En este punto se hace necesario conocer la función característica del sensor tacogenerador para realizar el ajuste equivalente entre RPM/1000 y voltios.

Con ayuda de LabView 8.5 versión estudiantil y un medidor de revoluciones por segundo (RPS) se caracterizo el tacogenerador con los siguientes datos obtenidos:

Tabla 55. Valores obtenidos de Voltaje de tacogenerador vs RPS

Voltaje Tacogenerador	RPS
0	0
0.402	12.7226
0.801	25.3807
1.192	37.8787
1.607	50.7614
1.999	63.2911
2.390	75.7575
2.800	89.2857
3.210	101.0101
3.610	114.9425

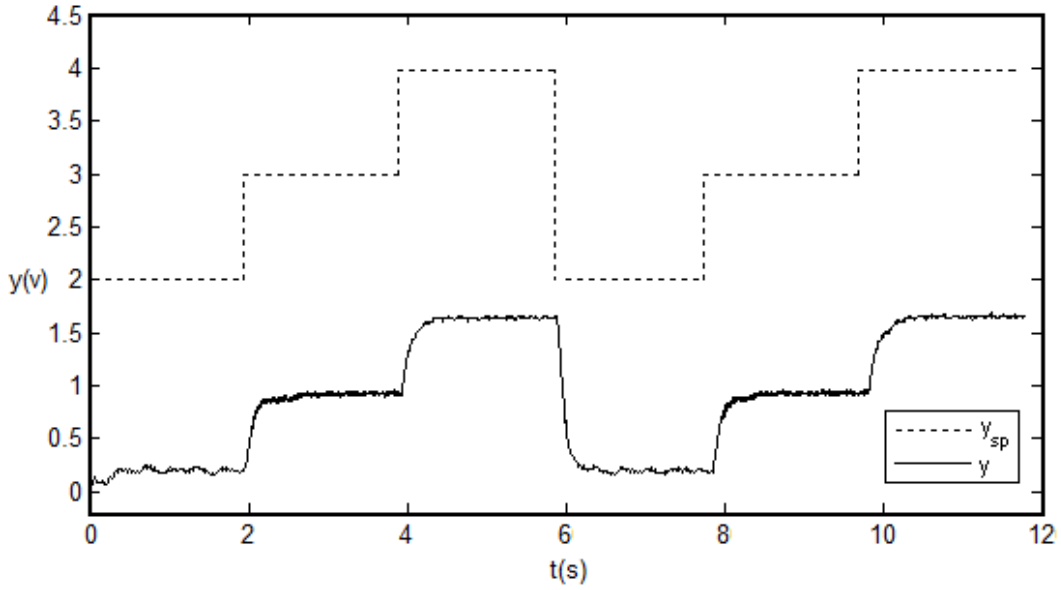
Con ayuda de Matlab se hizo la regresión lineal respectiva obteniendo la función que más se ajusta a los puntos de dispersión obtenidos.

$$\frac{RPM}{1000} = 1.903758 * V_{TACO} - 0.00267 \quad (82)$$

Esta linealización se resuelve de forma fácil en el controlador de proceso.

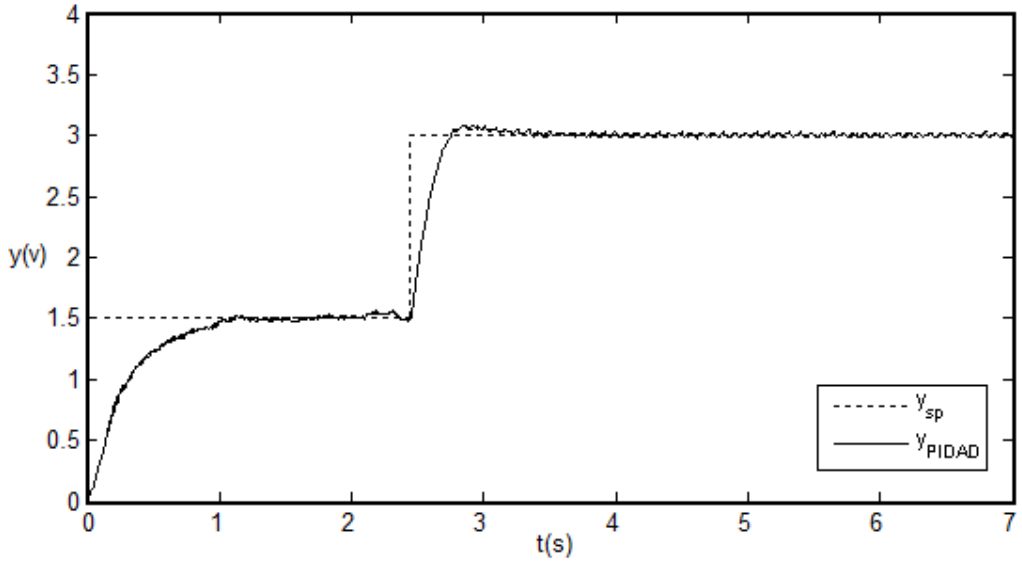
5.3.1 Pruebas de aplicación en el motor 1. Se realizaron pruebas con diferentes entradas y tipo de control para su posterior comparación.

Figura 165. Respuesta del motor 1 con realimentación unitaria a una entrada escalón multinivel



Se aplico una entrada escalón multinivel de 2000, 3000, 4000 RPM como indica la anterior grafica, se puede observar que la respuesta del motor 1 tiene un error en estado estable considerablemente grande y una respuesta transitoria aceptable.

Figura 166. Respuesta del motor 1 con PIDAD2 a una entrada escalón multinivel como rutina adaptiva



Despues de ejecutar la rutina adaptiva los parámetros convergen en:

$k_p = 1.44$ $k_i = 0.0106$ $k_d = 1.00$

Figura 167. Comparación de las respuestas del motor 1 según los controladores aplicados

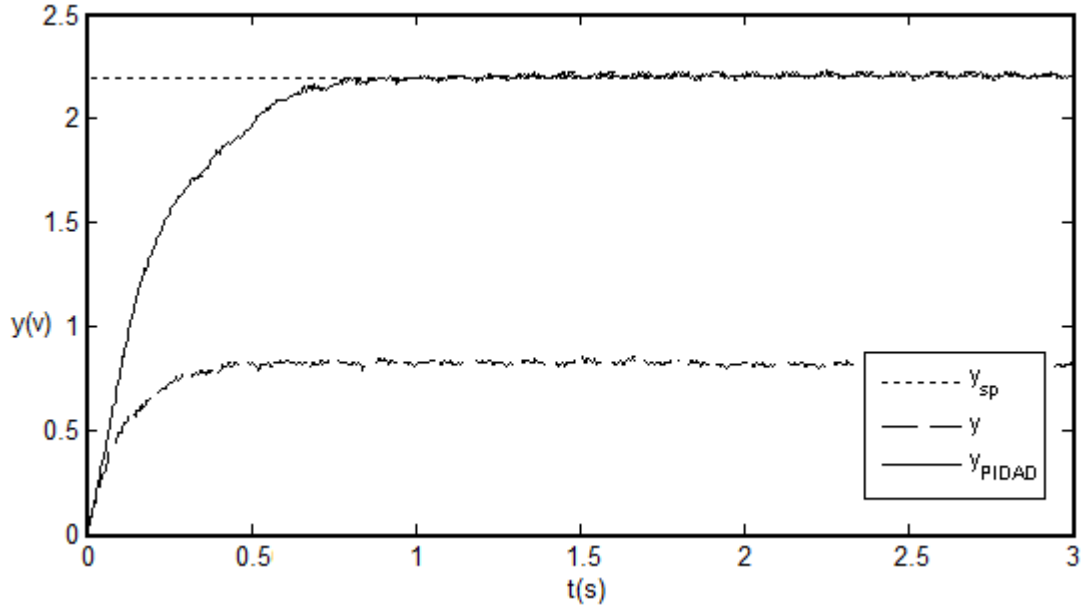


Tabla 56. Comparación cuantitativa de la respuesta transitoria del motor 1 a una entrada escalón unitario según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%) $t_a(s)$**	**Error en estado estable e_{ss}**	**Sobrepaso máximo ov**	**Tiempo de subida $t_s\ (s)$**
Realimentación unitaria	0.840	0.6250	3.17%	0.269
PIDAD	0.626	0.0008	1.49%	0.510

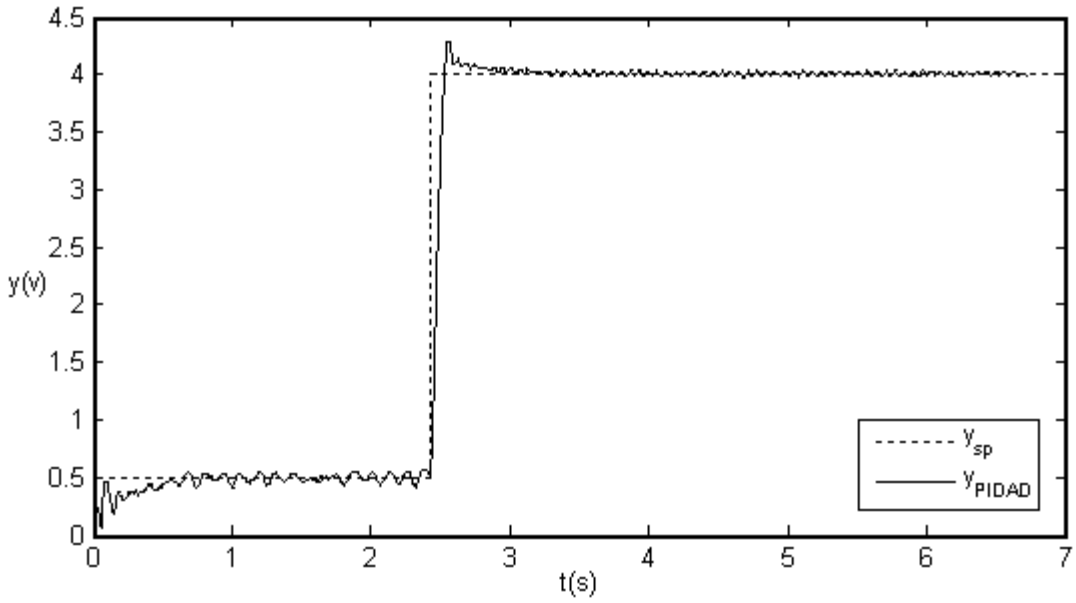
.

Se puede observar que el controlador PIDAD2 lleva la respuesta transitoria a unos niveles de control muy aceptables para la aplicación en un proceso real, corrigiendo el error en estado estable totalmente y llevando el sobrepaso a un nivel máximo de 1.49%.

5.3.2 Pruebas de aplicación en el motor 2. Se realizaron pruebas con diferentes entradas y tipo de control para su posterior comparación.

La respuesta del motor 2 con realimentación unitaria a una entrada escalón multinivel no represento ningún valor de voltaje medido por el tacogenerador es decir el motor en ningún momento experimento movimiento

Figura 168. Respuesta del motor 2 con PIDAD2 a una entrada escalón multinivel como rutina adaptiva



Despues de ejecutar la rutina adaptiva los parámetros convergen en:

$k_p = 1.58$ $k_i = 0.0106$ $k_d = 1.00$

Figura 169. Respuesta del motor 2 con PIDAD2 a una entrada escalón multinivel

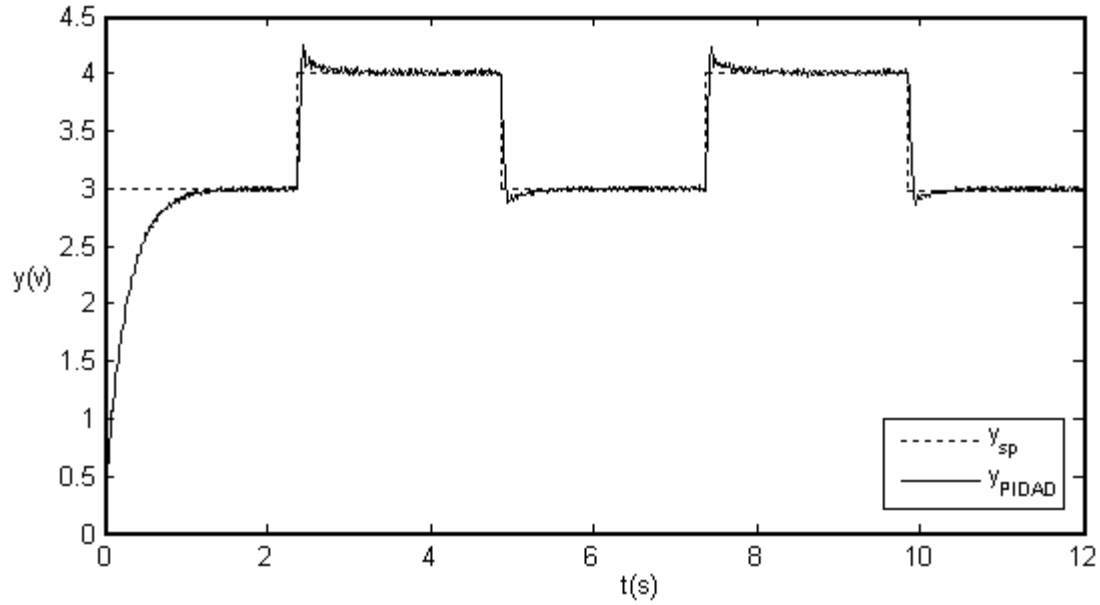


Figura 170. Respuesta del motor 2 con PIDAD2 a una entrada escalón unitario

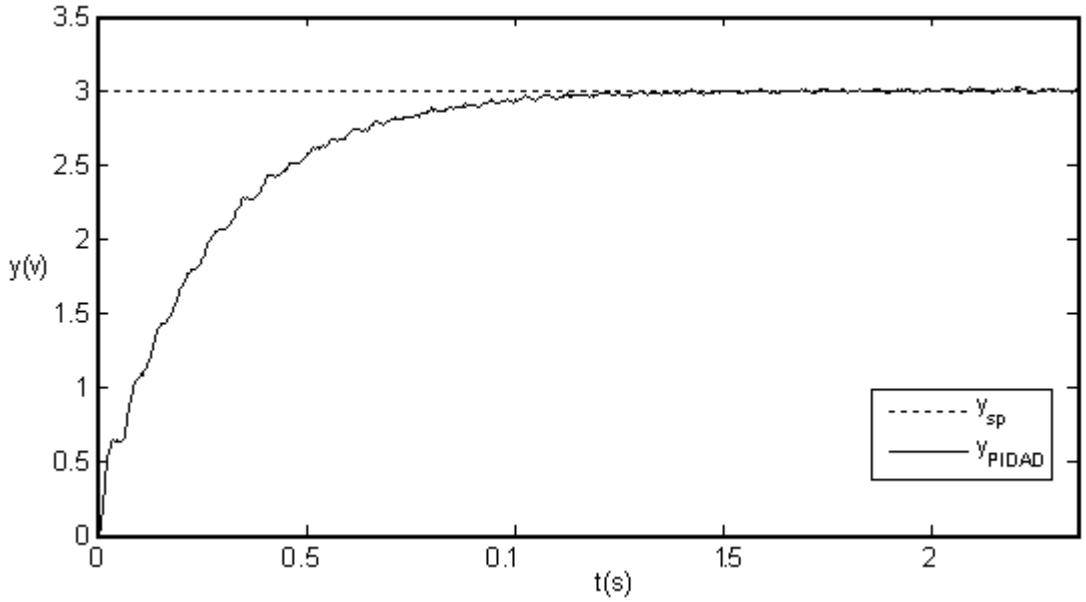


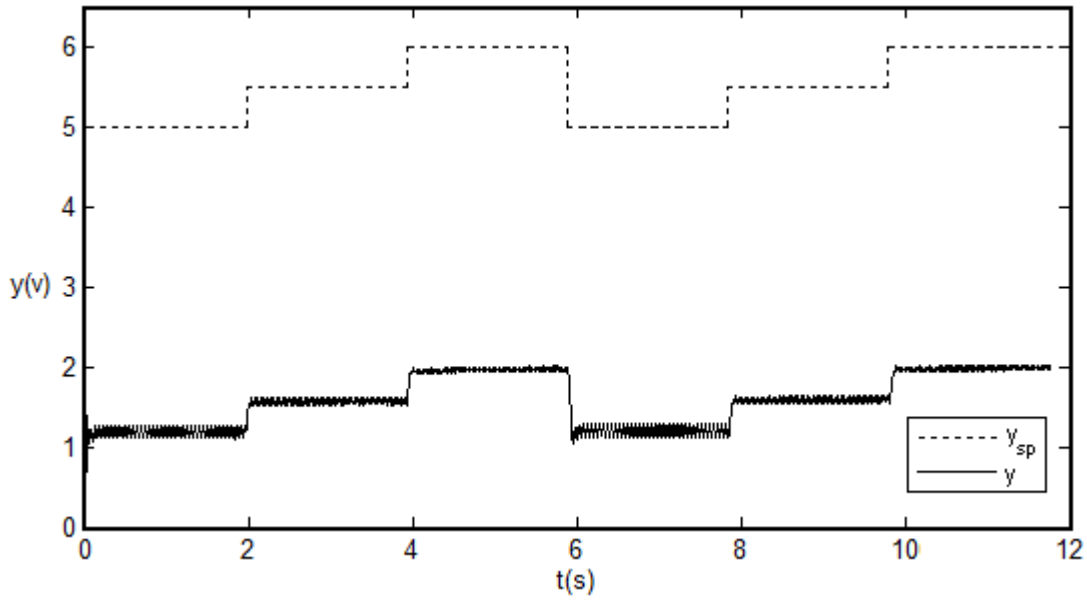
Tabla 57. Comparación cuantitativa de la respuesta transitoria del motor 2 a una entrada escalón unitario según los controladores aplicados

Controlador	Tiempo de asentamiento (5%) $t_a(s)$	Error en estado estable e_{ss}	Sobrepaso máximo ov	Tiempo de subida $t_s\ (s)$
Realimentación unitaria	-	-	-	-
PIDAD	0.868	0.0002	0.78%	0.601

Se puede observar que el controlador PIDAD2 lleva la respuesta transitoria a unos niveles de control muy aceptables para la aplicación en un proceso real, corrigiendo el error en estado estable totalmente y llevando el sobrepaso a un nivel máximo de 0.78%.

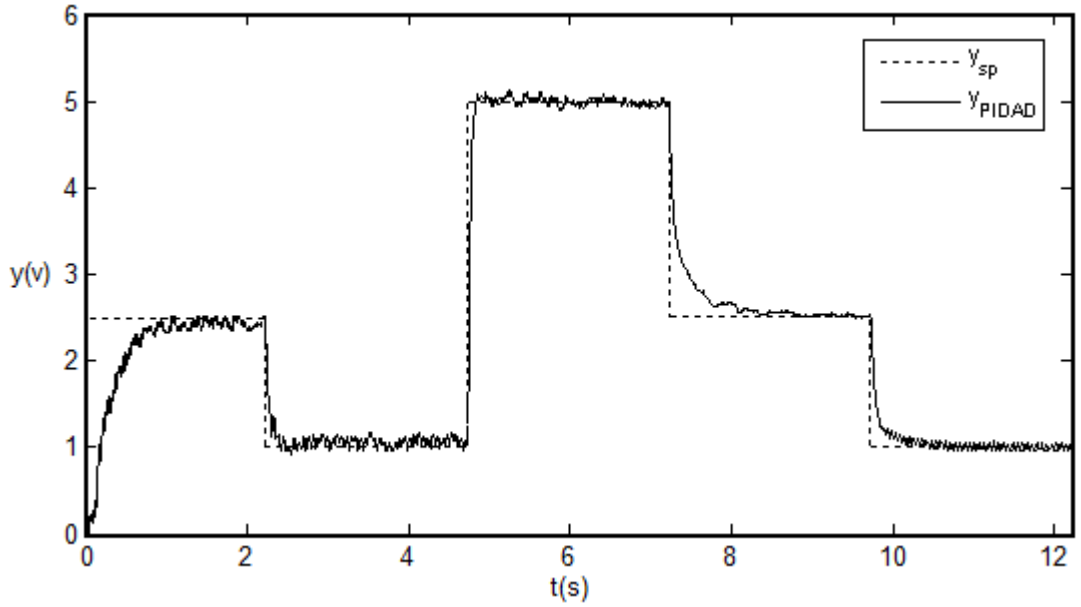
5.3.3 Pruebas de aplicación en el motor 3. Se realizaron pruebas con diferentes entradas y tipo de control para su posterior comparación.

Figura 171. Respuesta del motor 3 con realimentación unitaria a una entrada escalón multinivel



Se aplico una entrada escalón multinivel de 5000, 5500 y 6000 RPM como indica la anterior grafica, se puede observar que la respuesta del motor 2 tiene un error en estado estable considerablemente grande y por el contrario una respuesta transitoria aceptable.

Figura 172. Respuesta del motor 3 con PIDAD2 a una entrada escalón multinivel como rutina adaptiva



Despues de ejecutar la rutina adaptiva los parámetros convergen en:

$k_p = 1.29$ $k_i = 0.0085$ $k_d = 14.33$

Figura 173. Comparación de las respuestas del motor 3 según los controladores aplicados

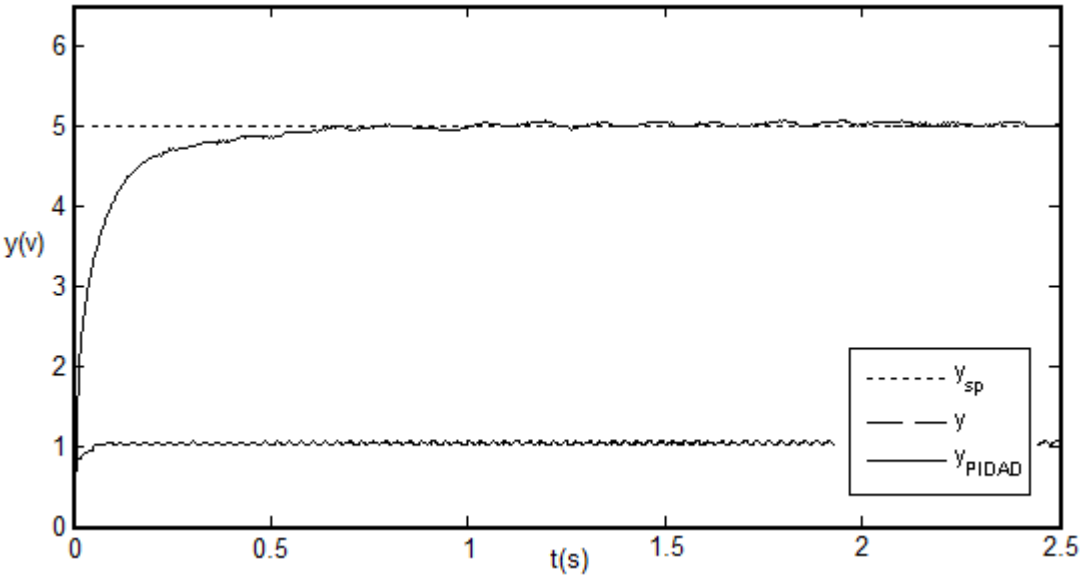


Tabla 58. Comparación cuantitativa de la respuesta transitoria del motor 3 a una entrada escalón unitario según los controladores aplicados

Controlador	**Tiempo de asentamiento (5%)** $t_a(s)$	**Error en estado estable** e_{ss}	**Sobrepaso máximo** ov	**Tiempo de subida** $t_s\ (s)$
Sin control	0.180	0.7885	10.43%	0.0048
PIDAD	0.361	0.0052	1.61%	0.166

Se puede observar que el controlador PIDAD2 lleva la respuesta transitoria a unos niveles de control muy aceptables para la aplicación en un proceso real, corrigiendo el error en estado estable totalmente y llevando el sobrepaso a un nivel máximo de 1.61%.

6. CONCLUSIONES

Este trabajo presenta la inclusión de la teoría de lógica difusa enfocada al control de procesos donde el mecanismo difuso propuesto que varia los parámetros del controlador PID, soluciona de una manera satisfactoria la sintonización automática a un proceso, demostrando que es una estrategia adecuada para abordar el problema de sintonía y adaptividad. Para este tipo de sistemas adaptivos no se hace necesario un conocimiento a priori de las características de la planta, como también se muestra que para que exista adaptabilidad no es completamente necesario conocer los parámetros de la planta, tal y como se observa en la aplicación del controlador sobre los tres procesos trabajados (…ver numeral 5.3…) sintetizando los resultados en las tablas 56, 57 y 58 donde se evidencia una reducción completa del error en estado estable, además de un sobrepaso alrededor de un 1% logrando que la respuesta transitoria se lleve a unos niveles óptimos de trabajo para la aplicación en un proceso real.

El controlador obtenido tiene la característica de ser robusto y flexible en cuanto rutinas de adaptividad como se puede observar en las figuras 153, 155 y 158, correspondientes a la pruebas de emulación en las cuales se propone señales de referencia en escalón unitario, escalón unitario periódico y escalón periódico multinivel, el controlador resuelve de manera muy satisfactoria el proceso de adaptación, ya que comparativamente con otros controladores como el PIDZ&N lleva la respuesta transitoria de la planta a unos niveles de requerimiento óptimos, asegurando que el sobrepaso se mantenga alrededor del 2% y el error en estado estable se reduzca menos del 0.1%, sin olvidar que es una planta (planta 2) con un grado medio de exigencia en cuanto a constantes de tiempo (tiempo muerto) para un controlador PID, no obstante la prueba realizada para una planta con constantes de tiempo correspondientes a las del proceso propuesto (planta 1, motor) que se pueden observar en las figura 148 presenta los siguientes resultados, sobrepaso alrededor del 1%, tiempo de asentamiento 0.0490s, tiempo de subida 0.0410s y error en estado estable por debajo del 0.1%, mejorando en 30 veces el sobrepaso, 3 veces el tiempo de asentamiento, y manteniendo el mismo error en estado estable y el mismo tiempo de subida en comparación con el controlador PIDZ&N que es un tipo de controlador PID base para la comparación en desempeño. Demostrando robustez y flexibilidad en cuanto a adaptividad y desempeño como controlador.

El control inteligente donde clasifica el controlador PIDAD1 y 2 ofrecen una solución sencilla en comparación con los controladores adaptivos que requieren información específica y precisa del proceso, el enfoque de adaptividad de los controladores PIDAD es mucho más sencillo en donde el procesamiento no se concentra solamente en conocer información del proceso, sino en aprender del mismo, de esta forma se reparte mas acertadamente los recursos de procesamiento y resulta más eficiente en algunos procesos

donde los controladores adaptivos requieren en detalle las características de la planta, ejecutando algoritmos con elevado consumo de recursos de procesamiento y tiempo en el lazo de ajuste de parámetros. Así también se demuestra que la lógica difusa es una excelente solución para elaborar un algoritmo de sintonía donde la importancia de la significancia y precisión puede variar, donde la exactitud no es tan relevante como el grado de importancia de los datos.

Se implemento un controlador digital para procesos industriales que cumple con unos requerimientos mínimos de procesamiento en tiempo real, demostrándose que un dsPIC es una herramienta con una amplia gama de módulos dedicados a tareas especificas de la automatización convirtiéndolo en un elemento robusto que puede será aplicado para el control de procesos.

El controlador PIDAD diseñado e implementado con tecnología al alcance en la región cuenta con iguales características de funcionamiento a un controlador adaptivo adquirido en el mercado industrial, con el valor agregado de tener un costo menor, representado en ser una solución eficiente para la implementación de control de procesos en la región.

7. TRABAJO FUTURO

Debido a que el controlador desarrollado que se presenta en este trabajo puede tener limitaciones en sistemas de órdenes superiores con grandes tiempos muertos, cabe anotar que esto se debe a la estructura inherente de un controlador PID, sin embargo se puede utilizar una parte del controlador propuesto para trabajar con múltiples controladores conmutados sintonizado los controladores PID utilizados con el mecanismo adaptivo propuesto, sin modificar el mecanismo de conmutación como es el modelo de dinámica de replicaciones[24]. Esperando conseguir mejores resultados para el manejo de procesos que presenten al control gran dificultad.

Otro trabajo que sería de gran interés en su realización, es utilizar una de las herramientas de la inteligencia artificial como las redes neuronales o la lógica difusa para desarrollar un mecanismo de decisión para ser utilizado en la estructura de control por conmutación[25], en el cual el controlador desarrollado y el mecanismo de decisión tengan un alto desempeño en el control de procesos de gran dificultad, sin olvidar que la implementación se pueda desarrollar en un sistema embebido.

[24] MUÑOZ, I y QUIJANO, N. Teoría de Juegos Evolutiva en Sistemas de Control Conmutado. En: VIII Congreso de la Asociación Colombiana de Automática, (2009); p.2.
[25] Op. cit., p.2.

8. RECOMENDACIONES

Establecer un protocolo de comunicación con el fin de conectar el sistema de control desarrollado directamente a una red industrial y de esta manera se pueda controlar un proceso dentro de la misma red en cuestión; los protocolos que pueden ser de aplicación son los protocolos estándar como son el profibus o CAN.

Implementar un protocolo para comunicarse e intercambiar datos con dispositivos autómatas programables, por ejemplo PLC's, o microcontroladores. Para esta labor se pueden utilizar protocolos RS-232, RS-485 o el I2C.

Implementar una carcasa inmune a los impactos y ruidos mecánicos y eléctricos que se pueden presentar en un ambiente industrial.

BIBLIOGRAFIA

16-BIT LANGUAGE TOOLS GETTING STARTED, Microchip Technology Inc, 2006, Disponible en Internet: URL: http://ww1.microchip.com/downloads/en/DeviceDoc/70094E.pdf, 60p.

AMESTEGUI, Mauricio. Apuntes de control PID. La Paz: Universidad Mayor de San Andrés, 2001. 134p.

An978 Application Note, International Rectifier, 2007, Disponible en Internet: URL: http://www.irf.com/technical-info/appnotes/an-978.pdf, 30p.

ANGULO, José; GARCIA, Begoña; MARTINEZ, Ignacio y SAENS, Javier. Microcontroladores Avanzados dsPIC. España: Thomson Editores, 2006. 768p.

ANGULO, José; ROMERO, Susana y ANGULO, Ignacio. Microcontroladores PIC Diseño práctico de aplicaciones. 3 ed. España: Mc Graw Hill, 2003. 231p.

ASTROM, Karl Johan y HAGGLUND, Tore. Automatic tuning of simple regulators with specifications on phase and amplitude margins. En: Automatica. Vol. 20, No. 5 (1984); p. 645–651.

_____ PID Controllers: Theory, Design and Tuning. 2 ed. Unites States of America: Instrument Society of America, 1995. 343p.

ASTROM, Karl Johan y WITTENMARK, Bjorn. Adaptive Control. 2ed. United States of America: Addison Wesley, 1995. 574p.

BASILIO, J. C. y MATOS S. R. Design of PI and PID Controllers With Transient Performance Specification. En: IEEE Transactions on Education. Vol. 45, No. 4 (2002); 7p.

CHAINHO, Joao; PEREIRA, Pedro; RAFAEL, Silviano y PIRES, A.J. A simple PID controller with adaptive parameter in a dsPIC. Setubal, Portugal: Escola Superior de Tecnología de Setubal, 2005. 5p.

CHING-HUNG, L. y CHING-CHENG T. Calculation of PID controller parameters by using a fuzzy neural network. En: ISA Transactions. Vol. 42 (2003); p. 391-400.

Conjuntos y Sistemas Difusos: Lógica Difusa y Aplicaciones. Universidad de Málaga. Disponible en Internet: URL:
http://www.lcc.uma.es/~ppgg/FSS/, 138p.

Diseño e implementación de un sistema de control digital de posición para un motor DC. Universidad Santo Tomas de Bucaramanga, 2006.

DT98-2 Design Tip, International Rectifier, 2001, Disponible en internet: URL:
http://www.irf.com/technical/info/designtp/dt98-2.pdf, 5p.

dsPIC30F4011/12 Data Sheet, Microchip Technology Inc, 2007, Disponible en Internet: URL:
http://ww1.microchip.com/downloads/en/DeviceDoc/70135E.pdf, 236p.

FADAEI, A. y SALAHSHOOR, K. Design and implementation of a new fuzzy PID controller for networked control systems. En: ISA Transactions. Vol. 47, No. 4 (2008); p. 351-361.

FENG, Gang y LOZANO, Rogelio. Adaptive Control Systems. Great Britain: Reed Educational, 1999. 330p.

Fuzzy Logic Toolbox User's Guide. The Mathworks Inc. Disponible en Internet: URL:
http://www.mathworks.com/access/helpdesk/help/pdf_doc/fuzzy/fuzzy.pdf, 343p.

Getting Started with dsPIC30F Digital Signal Controllers User's Guide. Microchip Technology Inc, 2005, Disponible en Internet: URL:
http://ww1.microchip.com/downloads/en/devicedoc/70151a.pdf, 132p.

GOMÁRIZ, Spartacus; BIEL, Domingo; MATAS, José y REYES, Miguel. Teoría de control y diseño electrónico. España: Ediciones Universidad Politécnica de Catalunya, 1998. 392p.

HD44780U (LCD-II). Hitachi, Ltd., 1998, Disponible en Internet: URL: http:semiconductor.hitachi.com/

JANTZEN, Jan. Design of Fuzzy Controllers. Denmark: Technical University of Denmark, 1998. 27p.

_____ Tunning of Fuzzy PID Controllers. Denmark: Technical University of Denmark, 1998. 22p.

KUO, Benjamin. Sistemas de Control Automático. 7 ed. México: Prentice Hall Hispanoamérica S.A, 1996. 898p.

LI, Hongxing; CHEN, Philip y HUANG, Han-Pang. Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Application in Engineering. United States of America: CRC Press LLC, 2001. 369p.

La Lógica difusa: Cuaderno Técnico No 191. Schneider Electric. Disponible en Internet: URL: http://www.schneiderelectric.es. 32p.

MOHAN, B. y SINHA, Arpita. Mathematical models of the simplest fuzzy PI/PD controllers with skewed input and output fuzzy sets. En: ISA Transactions. Vol. 47, No. 3 (2007); p 300-310.

MPLAB ASM30 MPLAB LINK30 AND UTILITIES USER'S GUIDE, Microchip Technology Inc, 2005, Disponible en internet: URL: http://ww1.microchip.com/downloads/en/devicedoc/Asm30_Link_Util_51317e.pdf, 280p.

MPLAB C30 C COMPILER USER'S GUIDE. Microchip Technology Inc, 2003, Disponible en Internet: URL: http://asl.epfl.ch/education/courses/MicroInfo/doc/51284D.pdf, 199p.

MUÑOZ, I y QUIJANO, N. Teoría de Juegos Evolutiva en Sistemas de Control Conmutado. En: VIII Congreso de la Asociación Colombiana de Automática, (2009); 6p.

OGATA, Katsuhiko. Sistemas de Control en Tiempo Discreto. 2 ed. México: Prentice Hall Hispanoamérica S.A, 1996. 745p.

MUDI, Rajani; DEY Chanchal y LEE Tsu-Tian. An improved auto-tuning scheme for PI controllers. En: ISA Transactions. Vol. 47, No. 1 (2008); p. 45-52.

PIC16F877A Data Sheet, Microchip Technology Inc, 2003, Disponible en Internet: URL: http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf, 235p.

RODRIGUEZ, Francisco y LOPEZ, Manuel. Control Adaptativo y Robusto. España: Secretariado de publicaciones de la universidad de Sevilla, 1996. 365p.

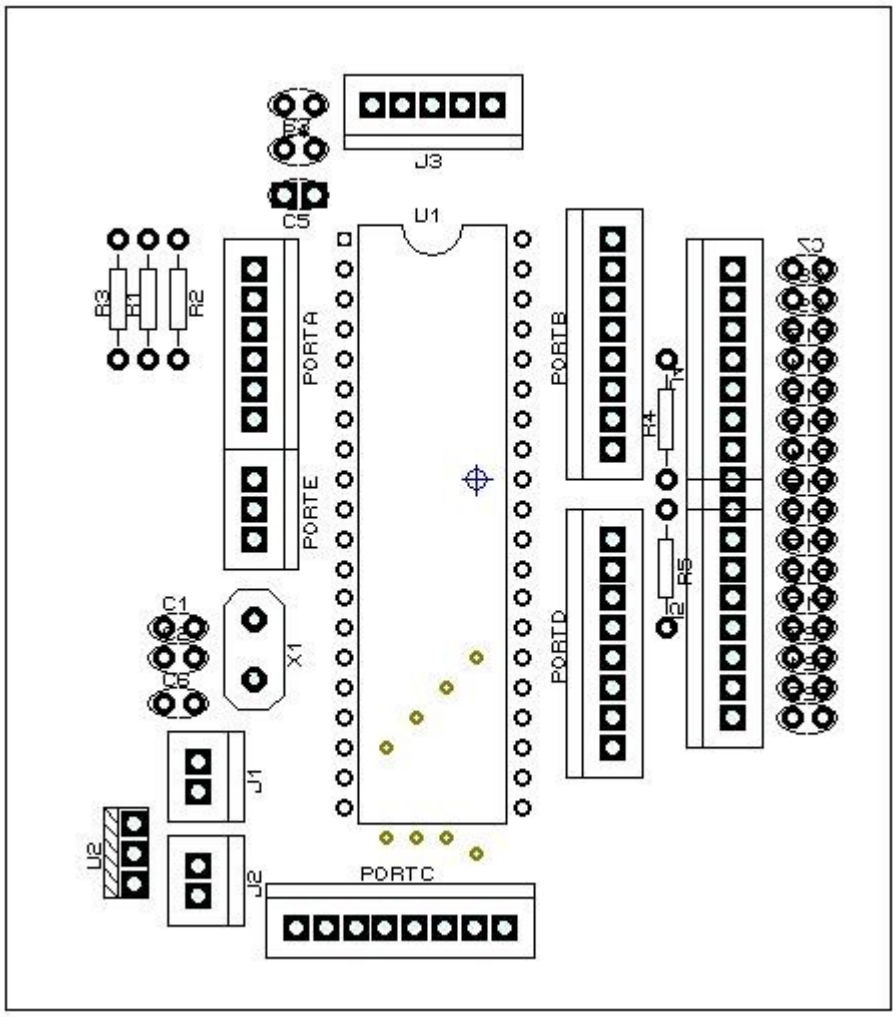
ROJAS, Luis Felipe. Controlador PID comerciales, Costa Rica: Universidad de Costa Rica, 2007. 96p.

TAO, Gang. Adaptive Control Design and Analysis. New Jersey: John Wiley & Sons, Inc., 2003. 640p.

VODA, A. y LANDAU I. A Method for the Auto-calibration of PID Controllers. En: Automatica. Vol. 31, No. 1 (1995); p. 41-53.

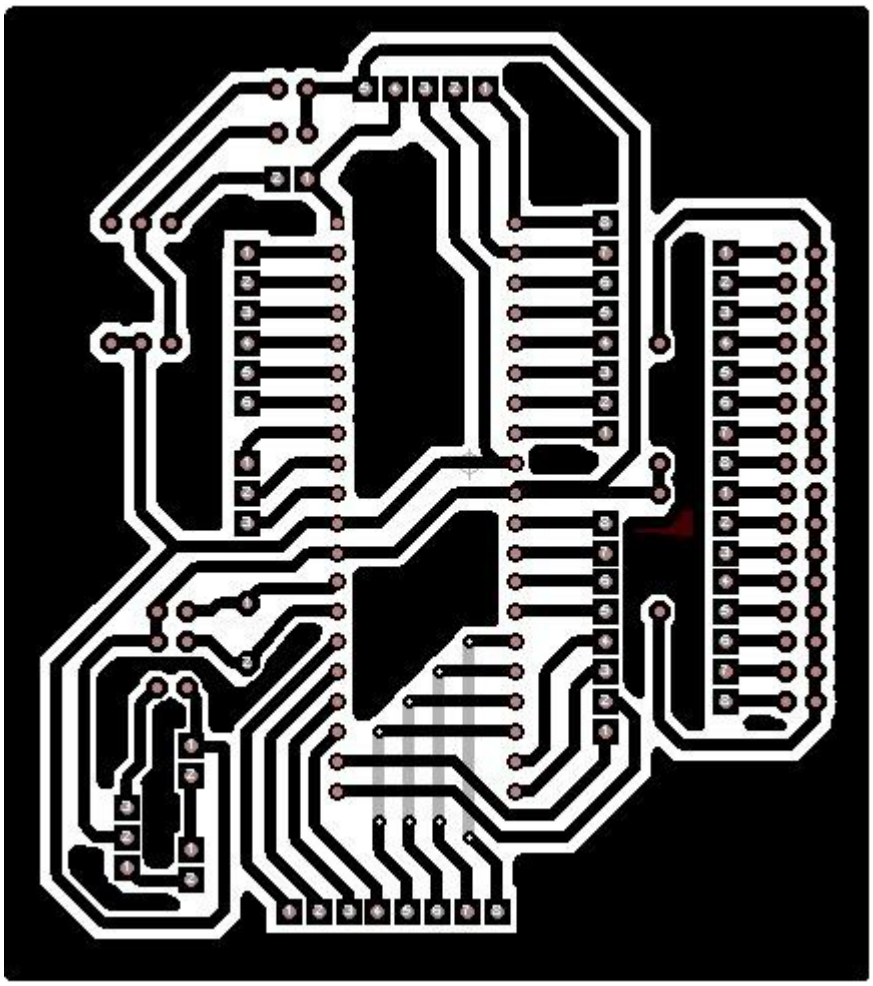
ZILOUCHIAN, Ali y JAMSHIDI, Mo. Intelligent Control Systems Using Soft Computing Methodologies. United States of America: CRC Press LLC, 2001. 493p.

ANEXO A. Esquema del circuito impreso del controlador de interfaz

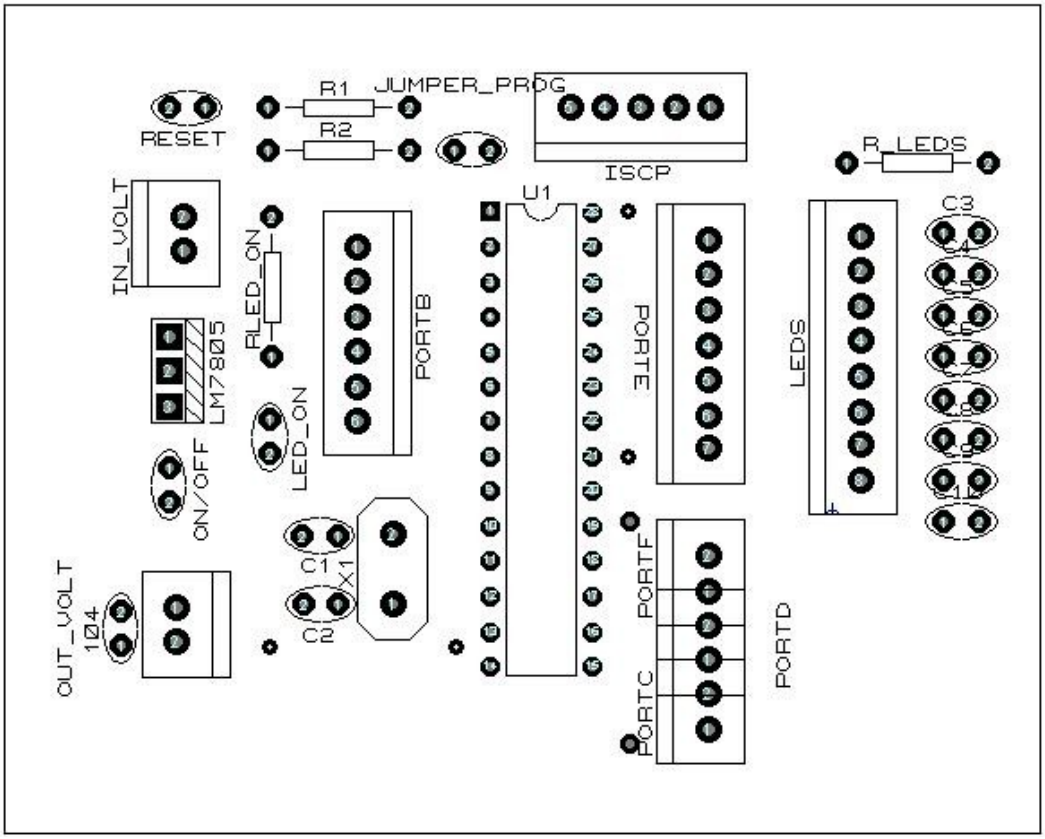


Vista de componentes del circuito impreso del controlador de interfaz

Vista de pistas del circuito impreso del controlador de interfaz

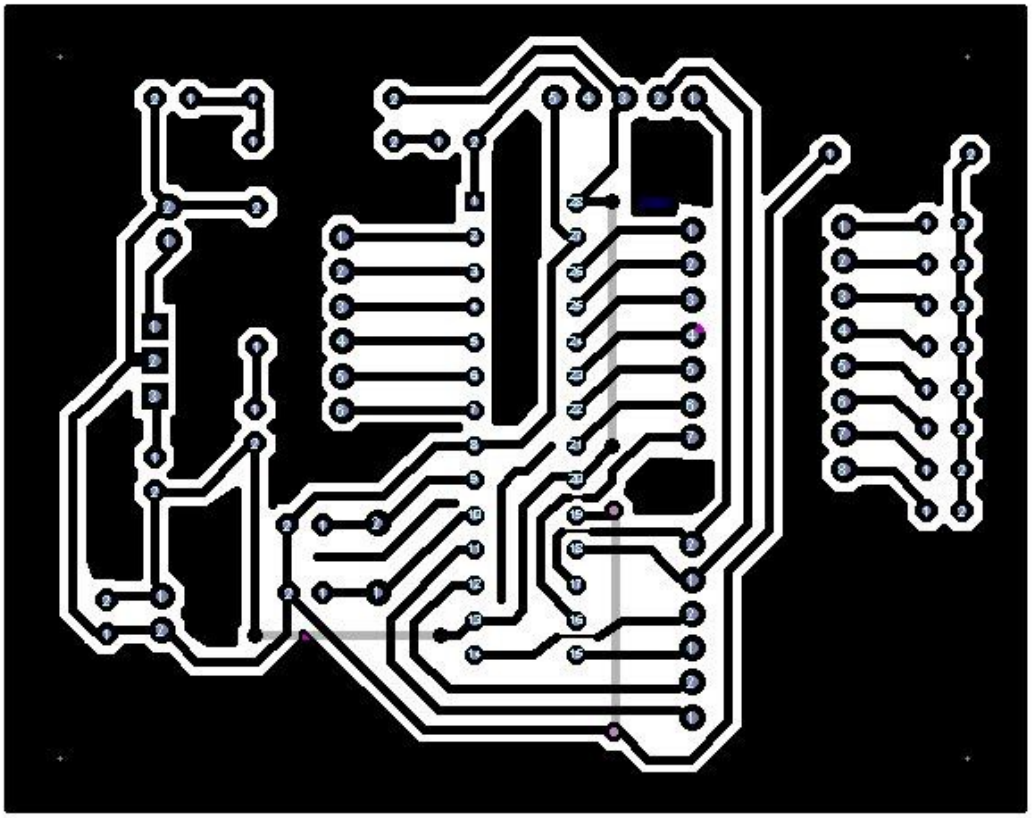


ANEXO B. Esquema del circuito impreso del controlador de proceso

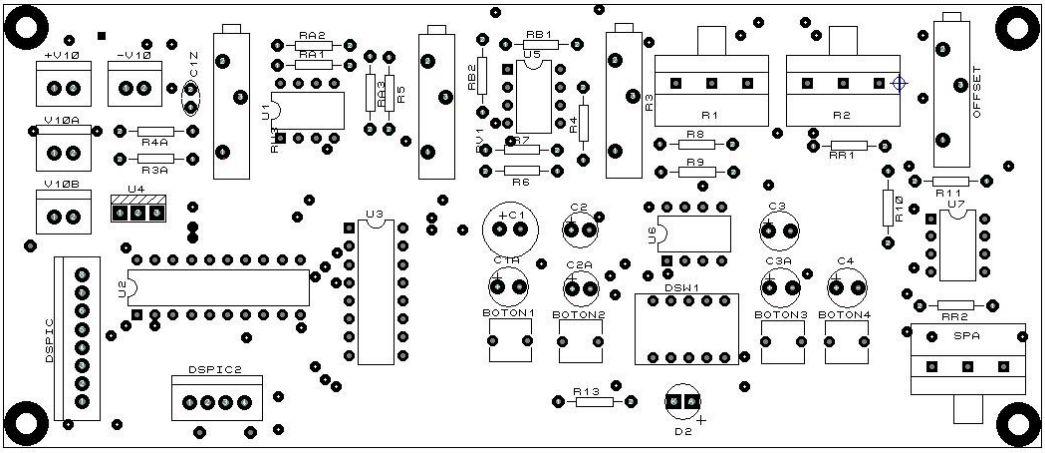


Vista de componentes del circuito impreso del controlador de proceso

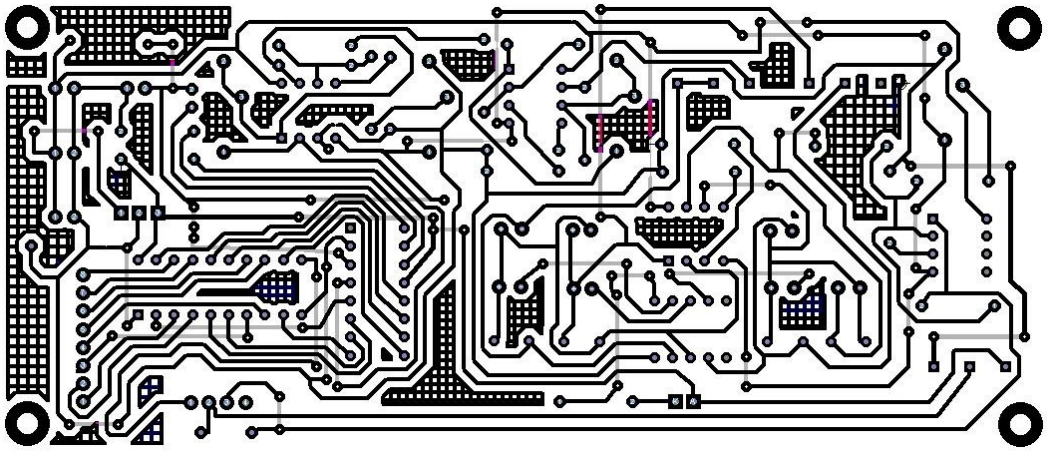
Vista de pistas del circuito impreso del controlador de proceso



ANEXO C. Esquema del circuito impreso de la emulación de la planta

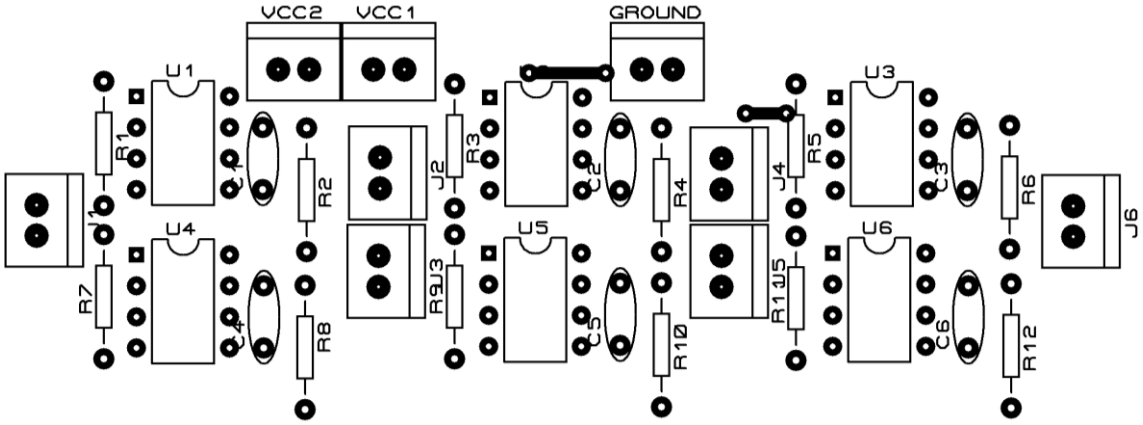


Vista de componentes del circuito impreso de la emulación de la planta

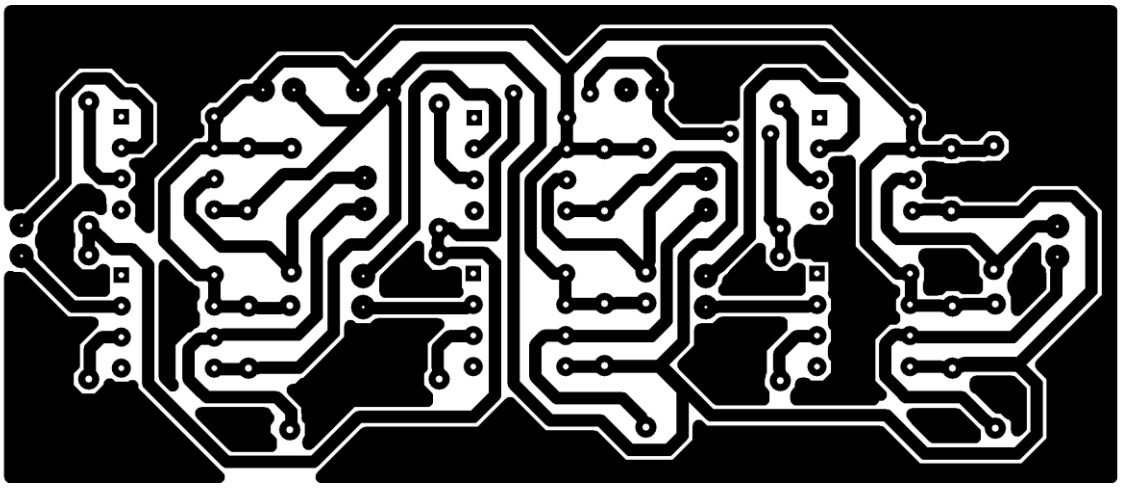


Vista de pistas del circuito impreso de la emulación de la planta

ANEXO D. Esquema del circuito impreso de los opto-acopladores

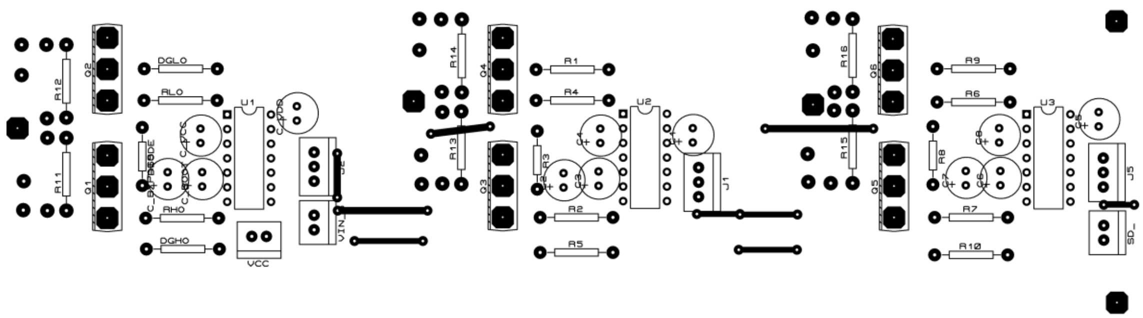


Vista de componentes del circuito impreso de los opto-acopladores

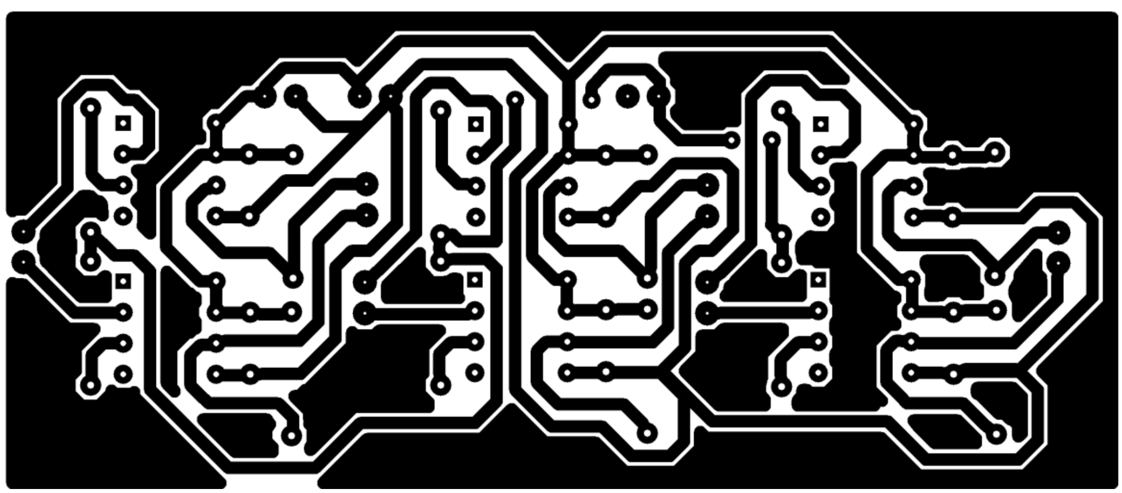


Vista de pistas del circuito impreso de los opto-acopladores

ANEXO E. Esquema del circuito impreso del puente inversor con los respectivos drivers

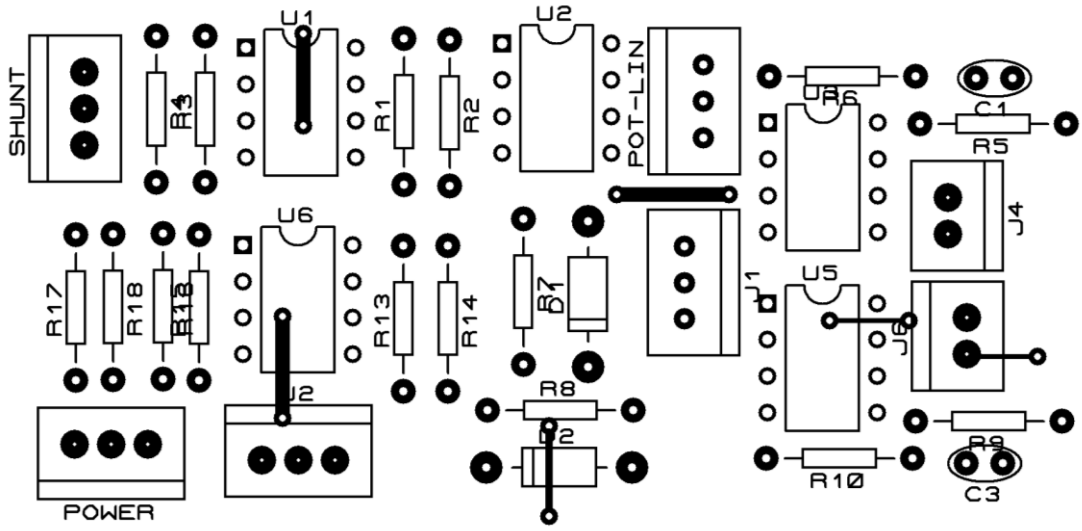


Vista de componentes del circuito impreso del puente inversor con los respectivos drivers

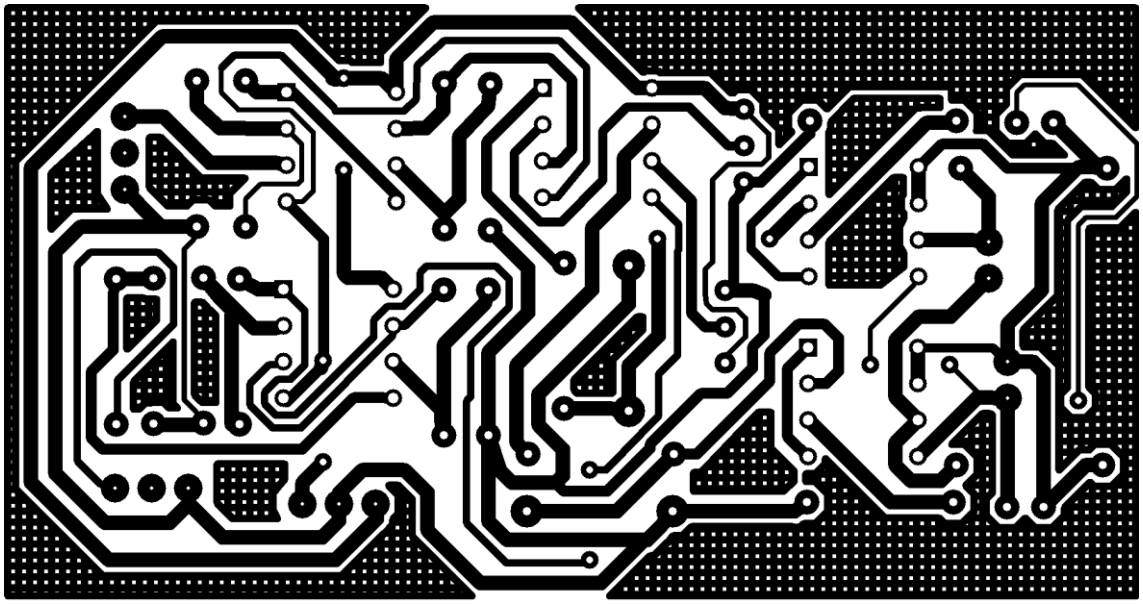


Vista de pistas del circuito impreso del puente inversor con los respectivos drivers

ANEXO F. Esquema del circuito impreso de los sensores de sobrecorriente y sobrevoltaje

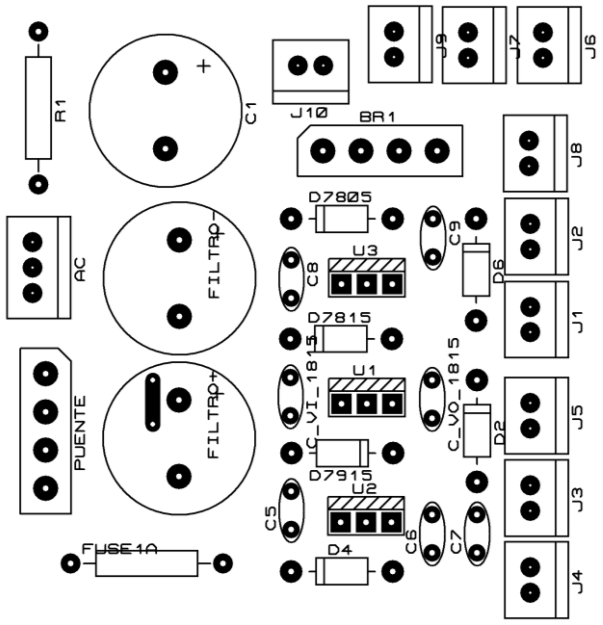


Vista de componentes del circuito impreso de los sensores de sobrecorriente y sobrevoltaje

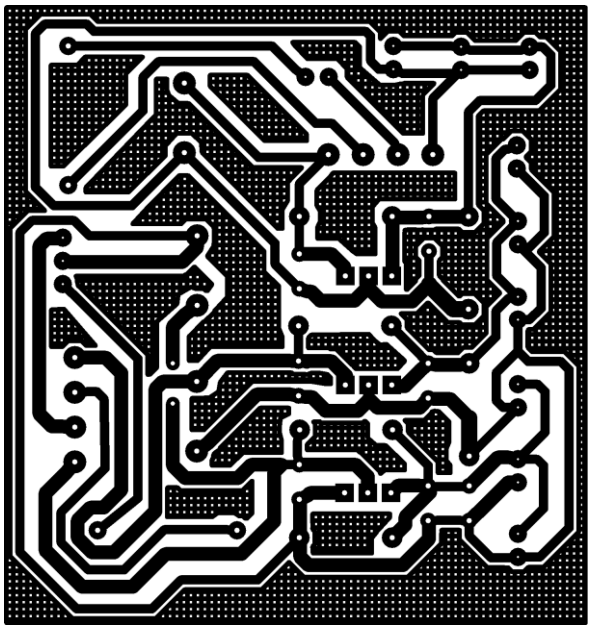


Vista de pistas del circuito impreso de los sensores de sobrecorriente y sobrevoltaje

ANEXO G. Esquema del circuito impreso de la fuente de bajo poder



Vista de componentes del circuito impreso de la fuente de bajo poder



Vista de pistas del circuito impreso de la fuente de bajo poder

ANEXO H. Código fuente de los bloques de SIMULINK

a. Código de la subfunción contenida en "*clock*"

```
function [clk_s, sp_s] = clock(sp, clk_e, sp_e)
%inicialización de variables
sp_s = sp_e;
% Clock
% Generación de reloj, este se debe reiniciar
% a cualquier cambio de setpoint con una tolerancia de 1%
clk_s = clk_e + 0.002;
%inicialización del sp referencia
if (clk_e == 0)
     sp_s = sp;
else
     if (sp > sp_e*1.01 || sp < sp_e*0.99)
     clk_s = 0;
     end
end
```

b. Código del subbloque de medición de "*ov*"

```
function [ov_s, sp1_s, sp2_s,m4_s, tovm_s]  = c_ov(t, y, sp,
ov_e, sp1_e, sp2_e, m4_e,tovm_e)
%Inicialización de Variables
ov_s = ov_e;
m4_s = m4_e;
tovm_s = tovm_e;
sp1_s = sp1_e;
sp2_s = sp2_e;
spr = 0;
trs = 0;
 %sincronización con t
if (t == 0)
    ov_s = 0;
    m4_s = 0;
elseif(t == 0.001)
    sp2_s = sp1_s;
    sp2_e = sp1_s;
    sp1_s = sp;
    sp1_e = sp;
else
    %   Sobrepaso (ov)
```

```
    spr = sp1_e - sp2_e;
    if (spr > 0)
            ov_s = (y - sp)/spr; %(sp - y)/-spr
    end
    if (ov_e > ov_s)
        ov_s = ov_e;
        %tiempo de ov max
        if ( m4_e == 0)
        tovm_s = t;
        m4_s = 1;
        end
    end
end
```

c. Código del subbloque de medición de "t_s"

```
function [ts_s, m1_s, tso_s, Sp1_s, tsm_s] = c_ts(t, y, sp,
m1_e, ts_e, tso_e, Sp1_e, tsm_e)
%inicialización de variables
m1_s = m1_e;
ts_s = ts_e;
tso_s = tso_e;
Sp1_s = Sp1_e;
um = 0;
spr = 0;
tsm_s = tsm_e;
% Tiempo de Subida (ts)
if (t == 0)
    m1_s = 0;
    m1_e = 0;
    tso_s = ts_e;
    ts_s = 0;
else
    if (m1_e == 0)
        spr = sp - Sp1_e;
        um = 0.9*spr + Sp1_e;
            if (spr > 0)
                if ( y >= um)
                    ts_s = t;
                    ts_e = t;
                    m1_s = 1;
                    Sp1_s = sp;
                end
            else %spr < 0
```

```
                if ( y <= um)
                    ts_s = t;
                    ts_e = t;
                    m1_s = 1;
                    Sp1_s = sp;
                end
            end
        %tiempo subida menor
        if ((ts_e < tsm_e || tsm_e == 0) && ts_e ~= 0)
            tsm_s = ts_e;
        end
    end
end
```

d. Código del subbloque de medición de "e_{ss}"

```
function [ess, eri_s, ers_s, m2_s, ta_s, m3_s] = c_ess(t, e,
sp, ess_e, eri_e, ers_e, m2_e, ta_e, m3_e)
%inicializacion de variables
ess = ess_e;
eri_s = eri_e;
ers_s = ers_e;
m2_s = m2_e;
ta_s = ta_e;
m3_s = m3_e;
% se comprueba, 5 muestras consecutivas de e
if (t == 0)
    ess = 0;
elseif(t == 0.001)
    ess = 0;
    m2_s = 0;
    m2_e = 0;
    m3_s = 0;
    m3_e = 0;
    ta_s = 0;
    ta_e = 0;
else
    %lb y la, refieren a la oscilación mínima permitida
    % para decir que e está en estado estable
    if (m2_e == 0)
        eri_s = 0.9*e;
        ers_s = 1.1*e;
        ers_e = ers_s;
        eri_e = eri_s;
```

```
    end
    if (e <= ers_e && e >= eri_e)
        m2_s = m2_e + 1;
    else
        m2_s = 0;
        m2_e = 0;
    end
        if (m2_e >= 20)
         if (sp <= 0)
            ess = 0;
        else
            ess = e/sp;
        end
            m2_s = 30;
            m2_e = 30;
         %tiempo asentamiento
        if (m3_e == 0 && ess ~= 0)
            ta_s = t;
            m3_s = 1;
        end
     end
end
```

ANEXO I. Código fuente del controlador de proceso

a. Código de la rutina principal del controlador de proceso (ASM30)

```
;lazo de control PID Adaptivo
;lazo PID de posición paralelo con Diferencias hacia atrás
.equ __30F4012, 1
.include"p30f4012.inc"
; --> Configuración de Bits
config __FOSC, CSW_FSCM_OFF & XT_PLL8
config __FWDT, WDT_OFF
config __FBORPOR, PBOR_OFF & MCLR_EN & RST_PWMPIN & PWMxH_ACT_LO &
PWMxL_ACT_LO
config __FGS, CODE_PROT_OFF

; --> Declaración de variables globales
.global _main
.global __T1Interrupt
.global __SPI1Interrupt
.global __CNInterrupt
.global _mk
.global _yk
.global _giro
.global _SPoint
.global _OutCero

.equ REG_DAEjeModo,   0x0A58
.equ Dir_DatosSPI,    0x0A00
.equ Dir_Vel1,        0x0A28
.equ Dir_Parametros,  0x0A10
.equ CVel,            0x0800

.data
_yk:                  .word 0x0000
_mk:                  .word 0x0000
_SPoint:              .word 0x0000
REG_Dato:             .word 0x0000
REG_Puntero:          .word 0x0000
REG_DatoSPI:          .word 0x0000
REG_CantReg:          .word 0x0000
REG_RespuestaINT:     .word 0x0000
REG_Cmd:              .word 0x0000
```

```
_giro:          .byte 0x00

.text
_main:
; --> CONFIGURACION DE PUERTOS
     mov  #0x0001, W0 ;b'00'0001', RB0=y(k)(in) ,RB3=SetPoint(in),
RB4-RB5(outs) giro
     mov  W0, TRISB
     mov  #0x2000, W0 ;b'0010'0000'
     mov  W0, TRISC ;RC14=clk_374(out), RC13=Start/stop(in)
     mov  #0x0000, W0
     mov  W0, TRISD ;RD0=mk(D7)(out), RD1=mk(D8)(out)
     mov  #0x0100, W0
     mov  W0, TRISE ;RE(0-5)=mk(D1-D6)(outs), RE8=SCK1(in)
     mov  #0x0004, W0 ; b'0100'
     mov  W0, TRISF ; RF2=SDI1(in), RF3=SDO1(out)

; --> CONFIGURACION ADC
     call _ADC_Inicializa

; --> CONFIGURACION SPI
     mov  #0x0000, W0 ;b'0000'0000'0000'0000'
     mov  W0, SPI1CON
     mov  #0x8000, W0 ;b'1000'0000'0000'0000'
     mov  W0, SPI1STAT

; --> CONFIGURACION DE TIMER
     ;T1CON {TON(15), TSIDL(13), TGATE(6), TCKPS(5-4), TSYNC(2),
TCS(1)}
     mov  #0x0000, W0 ;
     mov  W0, T1CON
     clr  TMR1
     mov  #0x9C40, W0 ; Carga el Registro de Periodo
     mov  W0, PR1    ; con XT 10Mhz PLL X8 --> 80Mhz --> 9C40h =
2ms, 4E20h = 1ms, HS = 20Mhz --> TMR1 PR1 = 1ms = 1388h, 2ms =
2710

     call _InitMCPWM

; --> CONFIGURACION DE INTERRUPCIONES
     bclr INTCON1, #NSTDIS ;Habilitar Interrupcion Anidada
     ;Prioridad de la interrupcion, Prioridad CPU = 0
     mov  #0x1000, W0 ;Prioridad TMR1 = 1
```

```
      mov  W0, IPC0
      mov  #0x0001, W0 ;Prioridad SPI1 = 1
      mov  W0, IPC2
      mov  #0x1000, W0 ;Prioridad CN = 1
      mov  W0, IPC3 ;Habilitacion de la interrupcion
      bset CNEN1, #CN1IE ;Habilit Int.CN1=RC13
      bset IEC0, #CNIE ;Habilita Int.CN
      bset IEC0, #SPI1IE ;habilita Int. SPI1
      bset IEC0, #T1IE ;Habilita Int. TMR1
      ;Flags Int. limpia software
      bclr IFS0, #T1IF ;bit3 0x0008
      bclr IFS0, #SPI1IF ;bit8 0x0100
      bclr IFS0, #CNIF ;bit15 0x8000

; --> Reinicio de Variables
_ReinicioVariables:
;Perifericos
      bclr T1CON, #TON ; Timer1 OFF
      clr  TMR1
      bclr ADCON1, #DONE
;OVDCON y giro
      mov  #0x0100, W0
      mov  W0, OVDCON
      bclr PTCON, #PTEN
      bclr PORTB, #0x4
      nop
      bclr PORTB, #0x5
;Puertos
      clr  PORTC ; RC14=clk_374
      ;clr  PORTE ; m(k)

;Registros
      clr  REG_Puntero
      clr  REG_CantReg
      clr  REG_RespuestaINT
      clr  REG_Cmd
;Reinicio OffSet mk y PID
      clr  REG_Dato ;inicializa REG_Dato en 2v
      call _SacarDato  ; Se resta con el Amp. Op.
      clr  PORTB
      call _Reinicio_PID ; Reinician variables Subrutina PID
;Inicio Perifericos
      bset ADCON1, #ADON ;ADC On
```

```
; --> Programa Principal

_CP1_EsperarInterrupcion:
     nop
     bclr  SPI1STAT, #SPIROV
     pwrsav     #1
     nop
     goto  _CP1_EsperarInterrupcion
; Fin Programa Principal
; --------------------------------------

; --> --> --> --> --> --> --> --> --> -->
; --> INTERRUPCIONES
; --> --> --> --> --> --> --> --> --> -->

__T1Interrupt:
     btss  PORTC, #0xD ;Start/Stop
     goto  _FinInterrupcion_T1
; --------------------------------------
; --> Manejo Interrupcion  T1
;Actualizar Dato en Lazo de control PID
;Int_T1:   mov   REG_Dato, W1
     call  _SacarDato

; --------------------------------------
; Leer entrada Analogica
_LeerPlanta:
     bset  ADCON1, #SAMP ;Inicia Muestreo, autoconversion
     nop
     nop
     nop
     nop
     nop

_CI1:btss  ADCON1, #DONE ;indica fin de conversion
     goto  _CI1
     nop
     bclr  ADCON1, #DONE ;SAMP se limpia, al iniciar conversion
     nop
     nop
Leer_yk:   mov   ADCBUF0, W1 ;capturo el valor de y(k)
     mov   W1, _yk
; --------------------------------------
```

```
; Adaptacion Parametros
_Adaptive:
     call  _Conversion_Variables_Clock
     btss  REG_DAEjeModo, #0 ; 1=Adap, 0=Man
     goto  _PIDControl
Adaptacion_kpkikd:
     call  _Calcular_ess_ta
     call  _Medir_ov_tovm
     call  _Medir_ts
     call  _Adaptar_kpkikd
;Fin Adaptacion Parametros

; ---------------------------------------
; Lazo de Control PID
_PIDControl:
     call  _PID_Pos_Dif_Atras_Ad
     mov   _mk, W1
     mov   W1, REG_Dato ;Dato actualiza proxima INT
;goto _FinInterrupcion_T1

;Fin Lazo de Control PID
; ---------------------------------------
_FinInterrupcion_T1:
     bclr  IFS0, #ADIF
     bclr  IFS0, #T1IF
     retfie
; ---------------------------------------

; --> --> --> --> --> --> --> --> --> -->
; --> MANEJO INTERRUPCION CN
; --> --> --> --> --> --> --> --> --> -->
__CNInterrupt:
     btss  PORTC, #0xD
     goto  FinProceso
;Inicio Motor DC
     bset  PTCON, #PTEN ;Habilitar PWM
;OVDCON = 0x3F00; Habilita, entrega comando al módulo PWM
     call  _c2f_datosSPI
     call  _Reinicio_Var_Clock
     call  _Reinicio_PID
     bset  T1CON, #TON
     clr   TMR1
     goto  FinInterrupcion_CN
```

```
FinProceso:
     bclr  T1CON, #TON
     bclr  PTCON, #PTEN ;desHabilitar PWM
     clr   PDC1
     clr   REG_Dato
     clr   PORTB
     clr   TMR1
     clr   REG_Cmd
     bclr  SPI1STAT, #SPIEN
     mov   #0x0000, W0 ;b'0000'0000'0000'0000'
     mov   W0, SPI1CON
     mov   #0x8000, W0 ;b'1000'0000'0000'0000'
     mov   W0, SPI1STAT
     call  _kp_f2c
     call  _ki_f2c
     call  _kd_f2c
FinInterrupcion_CN:
     bclr  IFS0, #CNIF ;Flag Int CN
     retfie
; ---------------------------------------
; --> --> --> --> --> --> --> --> --> -->
; --> MANEJO INTERRUPCION SPI1
__SPI1Interrupt:
;    btss  IFS0, #SPI1IF
;    goto  _FinInterrupcion_SPI1
     mov   SPI1BUF, W0
; ---------------------------------------
;Comando o Dato
     cp    W0, #0x0A ;W0 - A
     btss  SR, #C
     goto  Dato
; ---------------------------------------
Comando:  asr   W0, #0x04, W0
     bra   W0
     goto  _FinInterrupcion_SPI1
     goto  Cmd_EnvioDatos  ;0x10-20
     goto  Cmd_CambioVelocidad  ;0x30-40        ;0x30-40
     goto  Cmd_RegresenDatos    ;0x50-60
     goto  _FinInterrupcion_SPI1      ;0xF0
; ---------------------------------------
Cmd_EnvioDatos: ;0x10-20
     mov   #Dir_DatosSPI, W0
     mov   W0, REG_Puntero
```

```
     mov  #0x0001, W0
     mov  W0, REG_Cmd
     goto _FinInterrupcion_SPI1
; -------------------------------------
Cmd_CambioVelocidad:;0x30-40
     mov  #Dir_Vel1, W0 ;Vel1
     mov  W0, REG_Puntero
     mov  #0x0001, W0
     mov  W0, REG_Cmd
     mov  W0, CVel
     goto _FinInterrupcion_SPI1
; -------------------------------------
Cmd_RegresenDatos:   ;0x50-60
;convierto los parametros kp,ki,kd en cifra x registro
     mov  #Dir_Parametros, W0
     mov  W0, REG_Puntero ;alista 1er Dato
     mov.b [W0], W1
     mov  W1, SPI1BUF
     inc2 REG_Puntero
     mov  #0x03, W0
     mov  W0, REG_Cmd
     goto _FinInterrupcion_SPI1
; -------------------------------------
Dato: mov  REG_Cmd, W1
     bra  W1
     goto _FinInterrupcion_SPI1
     goto EnvioDatos
; -------------------------------------
;RegresenDatos
     mov  REG_Puntero, W1
     mov.b [W1], W0
     mov  W0, SPI1BUF
     inc2 REG_Puntero
     goto _FinInterrupcion_SPI1
; -------------------------------------
EnvioDatos: mov REG_Puntero, W1
     mov.b W0, [W1]
     inc2 REG_Puntero
_FinInterrupcion_SPI1:
     bclr IFS0, #SPI1IF ;flag Int. SPI
     retfie
; =====================================
; --> SUBRUTINAS DE USUARIO
```

```
; ----------------------------------------
_SacarDato:
     mov  REG_Dato, W1
     mov  W1, PDC1
     bset  PORTB, #4
     nop
     bclr  PORTB, #5
     return
```

b. Código de la rutina PID adaptivo (C30)

```
// Declaracion de Prototipos Funciones
#include<p30f4012.h>
#include<math.h>
#include<dsp.h>

// --------------------------------
// Conversion de Variables y clock
// --------------------------------
extern float sp;
extern int yk, SPoint, Modo_Dig_Anal;
struct DAEjeModo{
     unsigned MO:1; //Modo de Operacion, 1=Adap, 0=Man
     unsigned EP:1; //EjecucionPeriodica, 1=Si, 0=No
     unsigned DA:1; //Digital o Analogico, 1=Anal, 0=Dig
} DAEjeModo __attribute__((address(0xA58)));
unsigned int __attribute__((address(0x0800))) CVel = 0;

float y_k=0, ek=0, sp_a=0, sp=0;
unsigned long t=0;
float tf = 0;
unsigned long tiempo_ms[3];
unsigned int punt_t = 0, veces = 0;
extern float setpoint[3];
extern float tiempo[3];

void Conversion_Variables_Clock(void)
{
//Manejo de reloj
t = t + 2; tf = tf + 0.002;
if (t > tiempo_ms[punt_t]){
     punt_t++;
```

```
      veces++;
      if(veces > 2){
            DAEjeModo.MO = 0;}

      if(punt_t == 1){
            if(tiempo[punt_t] == 0){
                  punt_t = 0; }}

      if(punt_t == 2){
            if(tiempo[punt_t] == 0){
                  if(DAEjeModo.EP){
                        punt_t = 0; }
                  else{
                        punt_t = 1;}}}

      if(punt_t > 2){
            if(DAEjeModo.EP){
                  punt_t = 0; }
            else{
                  punt_t = 2;}}

      t = 2; tf = 0.002;
      sp_a = sp;
      sp = setpoint[punt_t];}

//convertir a HEX a voltage
y_k = 0.009307478*yk - 0.000019941;
//Nodo de Resta
if(DAEjeModo.DA){
sp = 0.00488758*SPoint;}
ek = sp - y_k;
}

void Reinicio_Var_Clock(void)
{
veces = 0;
punt_t = 0;
t = 0; tf = 0;

if(CVel == 1){
      tiempo[0] = 59;
      tiempo[1] = 0;
      tiempo[2] = 0;
```

```
      setpoint[1] = 0;
      setpoint[2] = 0;
      CVel = 0;}
tiempo_ms[0] = tiempo[0]*1000;
tiempo_ms[1] = tiempo[1]*1000;
tiempo_ms[2] = tiempo[2]*1000;
sp = setpoint[0];
sp_a = 0;
}
// Fin Conversion, clock

// --------------------------
// - INTEGRAL
// - Calcular ess y ta
// --------------------------
float yr_sup = 0, yr_inf = 0, ess = 0;
unsigned int m1 = 0, m2 = 0;
unsigned long ta = 0;

void Calcular_ess_ta(void)
{
if (t == 2)
{ess = 0; m1 = 0; m2 = 0; ta = 0;}
else
{
      if(m1 == 0)
            {yr_sup = 1.1*y_k; yr_inf = 0.9*y_k;}

      if (y_k >= yr_inf && y_k <= yr_sup)
            {m1 = m1 + 1;}
      else
            {m1 = 0;}

      if (m1 >= 25){
            if (sp <= 0.2)
                  {ess = 0;}
            else
                  {ess = ek/sp;}
            m1 = 25;
            // Tiempo de Asentamiento (ta)
            if ( m2 == 0 && ess != 0)
            {ta = t; m2 = 1;}}
}
```

```
}

// -------------------------------
// - DERIVATIVO
// - Medir ov tovm
// -------------------------------
float ov = 0, ov_a = 0, spr = 0, ov_m = 0;
unsigned long tovm = 0;
unsigned int md0 = 0;
void Medir_ov_tovm(void)
{
if(t == 2){
     ov = 0; ov_a = 0; ov_m = 0; md0 = 0; spr = 0; tovm = 0;}
else{
     spr = sp - sp_a;
     if(spr > 0){
          if(y_k > sp && spr != 0){
               ov = (y_k - sp)/spr;}}
     else{
          if(y_k < sp && spr != 0){
               ov = (y_k - sp)/spr;}}

     if(ov_a > ov){
          if(md0 == 0){
               tovm = t; md0 = 1; ov_m = ov_a;}}
     ov_a = ov;}
}

// -------------------------------
// - PROPORCIONAL
// - Medir ts
// -------------------------------
float um = 0;
unsigned int mp_1 = 0, mp_2 = 0;
unsigned long  ts = 0, tso = 0, tsm = 0;
float tsf = 0, tsof = 0, tsmf = 0;
void Medir_ts(void)
{
if(t == 2){
     mp_1 = 0; tso = ts; tsof = tsf; ts = 0; tsf = 0; um = 0;}
else{
     if(mp_1 == 0){
          um = 0.9*spr + sp_a;
```

```
            if(spr > 0){
                  if(y_k >= um){
                        ts = t; tsf = tf; mp_1 = 1;}}
            else{
                  if(y_k <= um){
                        ts = t; tsf = tf; mp_1 = 1;}}

            if((ts < tsm || tsm == 0) && ts != 0)
            {tsm = ts; tsmf = tsf;}} //tsm = tsmin
}
}
// -------------------------------
// - Mecanismo Adaptivo difuso
// - Adaptor kp ki kd
// -------------------------------

extern float Rvarkp, Rvarki, Rvarkd;
extern float kp, ki, kd;

void Adaptar_kpkikd(void)
{
if(t == ta && ta != 0){
      mamdani();
      kp = kp + Rvarkp;
      ki = ki + Rvarki;
      kd = kd + Rvarkd;}

if(kp < 1){kp = 1;}
if(kp >= 50){kp = 49.99;}
if(ki >= 0.5){ki = 0.4999;}
if(ki < 0.0001){ki = 0.0001;}
if(kd >= 50){kd = 49.99;}
if(kd < 1){kd = 1;}
}

// -------------------------------
// - Controlador PID
// -------------------------------
extern unsigned int mk;
//extern char giro;
float m_k=0;
float P=0, I=0, D=0;
float yold=0, Delta_yk=0;
```

```
float Smax=6.6061, Smin=0;
float ad=0;
//sintonizado con Matlab PID_Poscional_VariacionOgata_Parametros

void PID_Pos_Dif_Atras_Ad(void)
{
//calculo Delta_y(k) = y(k) - y(k-1)
Delta_yk = y_k - yold;

//Proporcional
//P = k(b*ysp(k) - y(k))
P = kp*ek;

//Integral
I = I + ki*ek;
//Manejo Windup
if (I < Smin )
{I = Smin;}
if (I > Smax )
{I = Smax;}

//Derivativo
ad = 0.1;
D = ad*D - kd*Delta_yk;
m_k = P + I + D;
//actualizo variables
yold = y_k;

if (m_k < Smin )
{m_k = Smin;}
if (m_k > Smax )
{m_k = Smax;}

mk = m_k*605.5009;

//convertir voltage a HEX, 204.6 para 10bits, 51 para 8bits

}

void Reinicio_PID(void)
{
yold = 0;
P = 0;
```

```
I = 0;
D = 0;
tsm = 0;
tsmf = 0;
mk = 0;
}

void ADC_Inicializa(void)
{
      ADCON1= 0x00E0; //
      ADCON3bits.SAMC = 0x02; //1TAD
//SAMC, bits de Auto muestreo x TAD
      ADCON3bits.ADRC = 0; //System clock
      ADCON3bits.ADCS = 0x0A; //bits de conversion segun TAD
      ADCON2 = 0x0000;
      ADCHS = 0x0000;
      ADPCFG = 0xFFFE;
      ADCSSL = 0x0001;
}

void InitMCPWM(void)
{
/****REGISTRO DE CONTROL DE FALLA __FLTACON__****/
FLTACON = 0;
/****REGISTRO OVERRIDE __OVDCON__****/
OVDCON = 0x0000;// Deshabilita todas las salidas PWM.
/****REGISTRO DE PERIODO __PTPER__****/
PTPER = 2000; // (20000000/5000) >> 1;//
/****REGISTRO 1 DE CONTROL __PWMCON1__****/
PWMCON1 = 0B00001110111;// habilita salidas PWM en modo
/****REGISTRO 2 DE CONTROL __PWMCON2__****/
PWMCON2 = 0B111100000010;
/****CONTROL BASE DE TIEMPO PWM __PTCON__****/
PTCON = 0B1000000000000010; //Inicia PWM en modo alineado al
      return;
}
```

c. Código de la rutina mecanismo adaptivo difuso 1 (MAD1) (C30)

```
#include "p30f4012.h"
#include<dsp.h>
#include<math.h>
```

```
unsigned int iess, iov;
extern vincki[100], vinckd[100];

float inc_ki(float f_ess)
{
     if(f_ess < 0.03){
     return (0);}
     iess = floor(f_ess*247.5);
     if (iess>99){iess=99;}
     return(0.01*Fract2Float(vincki[iess]));
}

float inc_kd(float f_ov)
{
     if(f_ov < 0.02){
     return(0);}
     iov = floor(f_ov*99);
     if (iov>99){iov=99;}
     return(22*Fract2Float(vinckd[iov]));
}
```

d. Código de la rutina mecanismo adaptivo difuso 2 (MAD2) (C30)

```
#include "p30f4012.h"
#include <math.h>
#include <dsp.h>

unsigned char
setP[45]={0,255,249,243,237,232,226,220,214,208,202,196,191,185,17
9,173,167,161,155,150,144,138,132,126,120,114,109,103,97,91,85,79,
73,68,62,56,50,44,38,32,27,21,15,9,3};
unsigned char
setM[80]={1,7,14,20,26,33,39,46,52,59,65,71,78,84,91,97,104,110,11
7,123,129,136,142,149,155,162,168,175,181,187,194,200,207,213,220,
226,232,239,245,252,252,245,239,232,226,220,213,207,200,194,187,18
1,175,168,162,155,149,142,136,129,123,117,110,104,97,91,84,78,71,6
5,59,52,46,39,33,26,20,14,7,1};
unsigned char
setG[45]={3,9,15,20,26,32,37,43,49,55,60,66,72,78,83,89,95,100,106
,112,118,123,129,135,141,146,152,158,163,169,175,181,186,192,198,2
03,209,215,221,226,232,238,244,249,255};
```

```
unsigned char
decP[14]={18,36,55,73,91,109,127,146,164,182,200,219,237,255};
unsigned char
incP[14]={237,219,200,182,164,146,128,109,91,73,55,36,18,0};
unsigned char
incM[28]={9,27,46,64,82,100,118,137,155,173,191,209,228,246,246,22
8,209,191,173,155,137,118,100,82,64,46,27,9};
//incG = decP
unsigned char varkp[50];
unsigned char varki[50];
unsigned char varkd[50];
unsigned char i, impli_1, impli_2, impli_3, impli_4, impli_5,
impli_5, impli_6, impli_7, impli_8, impli_9;
extern float tsf, ess, ov_m;
float numkp, denkp, numki, denki, numkd, denkd;
float Rvarkp, Rvarki, Rvarkd;
unsigned char i_ts, i_ess, i_ov;

void mamdani(void)
{
//METODO DE IMPLICACION MAMDANI
//ts = 0.1; //ts[0 1]
if(tsf <= 0.05){tsf = 0;}
if(tsf > 1){tsf = 1;}
i_ts = floor(99*tsf);
//ess = 0.2; //ess[0 0.4]
if(ess < 0.02){ess = 0;} //0.02
if(ess > 0.4){ess = 0.4;}
i_ess = floor(247.5*ess);
//ov = 0.1; //ov[0 1]
if(ov_m < 0.02){ov_m = 0; i_ts = 0;}
if(ov_m > 1){ov_m = 1;}
i_ov = floor(99*ov_m);

for(i=0;i<=49;i=i+1){
     varkp[i] = 0;
     varki[i] = 0;
     varkd[i] = 0;}

//Regla 1. SI ov es G ENTONCES k_p es decP and k_i es decP and k_d
es incG
if (i_ov > 54){
    impli_1 = setG[i_ov - 55];
```

```
    for(i=0;i<=13;i=i+1){
        if(decP[i] > impli_1){
            varkp[i+1] = impli_1;
            varki[i+1] = impli_1;}
        else{
            varkp[i+1] = decP[i];
            varki[i+1] = decP[i];}
        if(decP[i] < impli_1){
            varkd[i+36] = decP[i];}
        else{
            varkd[i+36] = impli_1;}}}

//Regla 2. SI ov es M ENTONCES k_d es incM
if (i_ov > 9 && i_ov < 90){
    impli_2 = setM[i_ov - 10];
    for(i=0;i<=27;i=i+1){
        if(impli_2 > varkd[i+18] && incM[i] > varkd[i+18]){
            if(incM[i] < impli_2){
                varkd[i+18] = incM[i];}
            else{
                varkd[i+18] = impli_2;}}}}

//Regla 3  SI ov es P ENTONCES k_d es incP
if (i_ov > 0 && i_ov < 45){
    impli_3 = setP[i_ov];
     for(i=0;i<=13;i=i+1){
        if(impli_3 > varkd[i+15] && incP[i] > varkd[i+15]){
            if(incP[i] < impli_3){
                varkd[i+15] = incP[i];}
            else{
                varkd[i+15] = impli_3;}}}}

//Regla 4  SI e_ss es G ENTONCES k_p es incM and k_i es incG
if (i_ess > 54){
    impli_4 = setG[i_ess - 55];
     for(i=0;i<=27;i=i+1){
        if(impli_4 > varkp[i+18] && incM[i] > varkp[i+18]){
            if(incM[i] < impli_4){
                varkp[i+18] = incM[i];}
            else{
                varkp[i+18] = impli_4;}}
    if(i <= 13){
        if(impli_4 > varki[i+36] && decP[i] > varki[i+36]){
```

```
            if(decP[i] < impli_4){
                varki[i+36] = decP[i];}
            else{
                varki[i+36] = impli_4;}}}}}

//Regla 5  SI e_ss es M ENTONCES k_p  es incP and k_i es incM
if(i_ess > 9 && i_ess < 90){
    impli_5 = setM[i_ess - 10];
     for(i=0;i<=27;i=i+1){
        if(impli_5  > varki[i+18] && incM[i] > varki[i+18]){
            if(incM[i] < impli_5){
                varki[i+18] = incM[i];}
            else{
                varki[i+18] = impli_5;}}
    if(i <= 13){
        if(impli_5 > varkp[i+15] && incP[i] > varkp[i+15]){
            if(incP[i] < impli_5){
                varkp[i+15] = incP[i];}
            else{
                varkp[i+15] = impli_5;}}}}}

//Regla 6  SI e_ss es P ENTONCES k_i es incP
if (i_ess > 0 && i_ess < 45){
    impli_6 = setP[i_ess];
     for(i=0;i<=13;i=i+1){
        if(impli_6 > varki[i+15] && incP[i] > varki[i+15]){
            if(incP[i] < impli_6){
                varki[i+15] = incP[i];}
            else{
                varki[i+15] = impli_6;}}}}

//Regla 7  SI t_s es G ENTONCES k_p es incG and k_d decP
if (i_ts > 54){
    impli_7 = setG[i_ts - 55];
     for(i=0;i<=13;i=i+1){
        if(impli_7 > varkp[i+36] && decP[i] > varkp[i+36]){
            if(decP[i] < impli_7){
                varkp[i+36] = decP[i];}
            else{
                varkp[i+36] = impli_7;}}
        if(impli_7 > varkd[i+1] && decP[i] > varkd[i+1]){
            if(decP[i] < impli_7){
                varkd[i+1] = decP[i];}
```

```
            else{
                varkd[i+1] = impli_7;}}}}

//Regla 8  SI t_s es M ENTONCES k_p es incM and k_d decP
if (i_ts > 9 && i_ts < 90){
    impli_8 = setM[i_ts - 10];
     for(i=0;i<=27;i=i+1){
        if(impli_8 > varkp[i+18] && incM[i] > varkp[i+18]){
            if(incM[i] < impli_8){
                varkp[i+18] = incM[i];}
            else{
                varkp[i+18] = impli_8;}}
    if(i <= 13){
        if(impli_8 > varkd[i+1] && decP[i] > varkd[i+1]){
            if(decP[i] < impli_8){
                varkd[i+1] = decP[i];}
            else{
                varkd[i+1] = impli_8;}}}}}

//Regla 9  SI t_s es P ENTONCES k_p es incP and k_d decP
if (i_ts > 0 && i_ts < 45){
    impli_9 = setP[i_ts];
     for(i=0;i<=13;i=i+1){
        if(impli_9 > varkp[i+15] && incP[i] > varkp[i+15]){
            if(incP[i] < impli_9){
                varkp[i+15] = incP[i];}
            else{
                varkp[i+15] = impli_9;}}
        if( impli_9 > varkd[i+1] && decP[i] > varkd[i+1]){
            if(decP[i] < impli_9){
                varkd[i+1] = decP[i];}
            else{
                varkd[i+1] = impli_9;}}}}

//DEFUSIFICACION
numkp=0;
denkp=0;
numki=0;
denki=0;
numkd=0;
denkd=0;

for(i=0;i<=49;i=i+1){
```

```
    if(varkp[i] != 0){
        numkp = numkp + varkp[i]*(i+1);
        denkp = denkp + varkp[i];}
    if(varki[i] != 0){
        numki = numki + varki[i]*(i+1);
        denki = denki + varki[i];}
    if(varkd[i] != 0){
        numkd = numkd + varkd[i]*(i+1);
        denkd = denkd + varkd[i];}}

if(denkp == 0){Rvarkp = 0;}
else {
Rvarkp=(numkp/denkp);
Rvarkp= Rvarkp*0.025714 - 0.38571;} //0.0285x – 0.428
if(denki == 0){Rvarki = 0;}
else{
Rvarki=(numki/denki);
Rvarki= Rvarki*0.00034286 - 0.0051428;} //0.171x – 2.57
if(denkd == 0){Rvarkd = 0;}
else{
Rvarkd=(numkd/denkd);
Rvarkd= Rvarkd*0.7143 - 10.714;} //0.00142x – 0.0214
}
```

e. Código de la rutina convertir datos de cifras a flotantes (C30)

```
#include "p30f4012.h"
// -------------------------------
// Convertir Datos SPI , cifra2float
// -------------------------------

unsigned int __attribute__ ((address(0xA00))) DatosSPI[44];
float c1, c2, c3, c4, fact_c2f;
float kp, ki, kd, VelN, VoltN;
float setpoint[3];
float tiempo[3];

float c2f(unsigned int punt_c2f, unsigned int div_c2f){
c1 = DatosSPI[punt_c2f];
switch (DatosSPI[punt_c2f + 1]){
case 0: c2 = 0; break;
case 1: c2 = 0.1; break;
```

```
case 2: c2 = 0.2; break;
case 3: c2 = 0.3; break;
case 4: c2 = 0.4; break;
case 5: c2 = 0.5; break;
case 6: c2 = 0.6; break;
case 7: c2 = 0.7; break;
case 8: c2 = 0.8; break;
default: c2 = 0.9;}
switch (DatosSPI[punt_c2f + 2]){
case 0: c3 = 0; break;
case 1: c3 = 0.01; break;
case 2: c3 = 0.02; break;
case 3: c3 = 0.03; break;
case 4: c3 = 0.04; break;
case 5: c3 = 0.05; break;
case 6: c3 = 0.06; break;
case 7: c3 = 0.07; break;
case 8: c3 = 0.08; break;
default: c3 = 0.09;}
switch (DatosSPI[punt_c2f + 3]){
case 0: c4 = 0; break;
case 1: c4 = 0.001; break;
case 2: c4 = 0.002; break;
case 3: c4 = 0.003; break;
case 4: c4 = 0.004; break;
case 5: c4 = 0.005; break;
case 6: c4 = 0.006; break;
case 7: c4 = 0.007; break;
case 8: c4 = 0.008; break;
default: c4 = 0.009;}
switch (div_c2f){
case 0: fact_c2f = 1; break;
case 1: fact_c2f = 1000; break;
case 2: fact_c2f = 10; break;
case 3: fact_c2f = 0.1; break;
case 4: fact_c2f = 0; break;
default: fact_c2f = 1;}
return ((c1 + c2 + c3 + c4)*fact_c2f);} //

void c2f_datosSPI (void)
{
VelN = c2f(0,0);
VoltN = c2f(4,1);
```

```
kp = c2f(8,2);
ki = c2f(12,3);
kd = c2f(16,2);
setpoint[0] = c2f(20,0);
tiempo[0] = c2f(24,1);
setpoint[1] = c2f(28,0);
tiempo[1] = c2f(32,1);
setpoint[2] = c2f(36,0);
tiempo[2] = c2f(40,1);
}

unsigned int i_SPI = 0;
void Reinicio_DatosSPI(void)
{
for(i_SPI = 0; i_SPI < 44; i_SPI++){
     DatosSPI[i_SPI] = 0;}
}
```

f. Código de la rutina convertir datos de flotante a cifra (C30)

```
#include "p30f4012.h"
#include<math.h>

// PROPORCIONAL (kp)
extern float kp;
extern int DatosSPI[44];
float kp_aux;
void kp_f2c(void)
{
//solo admite valores [1.00 - 49.99] de kp
if (kp < 1){kp = 1;}
if (kp >= 50){kp = 49.99;}
kp_aux = kp;
if(kp_aux<10)
{
DatosSPI[8] = 0x0000;
DatosSPI[9]=floor(kp_aux);
kp_aux = (kp_aux - DatosSPI[9])*10;
DatosSPI[10]=floor(kp_aux);
kp_aux = (kp_aux - DatosSPI[10])*10;
DatosSPI[11]=floor(kp_aux);
}
```

```
else
{
kp_aux=kp/10;
DatosSPI[8]=floor(kp_aux);
kp_aux = (kp_aux - DatosSPI[8])*10;
DatosSPI[9]=floor(kp_aux);
kp_aux = (kp_aux - DatosSPI[9])*10;
DatosSPI[10]=floor(kp_aux);
kp_aux = (kp_aux - DatosSPI[10])*10;
DatosSPI[11]=floor(kp_aux);
}
}
//INTEGRAL (ki)
extern float ki;
float ki_aux;
void ki_f2c(void)
{
//solo admite valores de [0.0001 - 0.4999] de ki
if (ki < 0.0001){ki = 0.0001;}
if (ki >= 0.5){ki = 0.4999;}
ki_aux = ki;
if(ki_aux<0.001)
{DatosSPI[12]=0;DatosSPI[13]=0;DatosSPI[14]=0;
ki_aux=ki*10000;
DatosSPI[15]=floor(ki_aux);}
else
      if(ki_aux<0.01)
      {DatosSPI[12]=0;DatosSPI[13]=0;
      ki_aux=ki*1000;
      DatosSPI[14]=floor(ki_aux);
      ki_aux=(ki_aux-DatosSPI[14])*10;
      DatosSPI[15]=floor(ki_aux);}
      else
            if(ki_aux<0.1)
            {DatosSPI[12]=0;
            ki_aux=ki*100;
            DatosSPI[13]=floor(ki_aux);
            ki_aux=(ki_aux-DatosSPI[13])*10;
            DatosSPI[14]=floor(ki_aux);
            ki_aux=(ki_aux-DatosSPI[14])*10;
            DatosSPI[15]=floor(ki_aux);}
            else
                  if(ki_aux>=0.1)
```

```
                {ki_aux=ki*10;
                DatosSPI[12]=floor(ki_aux);
                ki_aux=(ki_aux-DatosSPI[12])*10;
                DatosSPI[13]=floor(ki_aux);
                ki_aux=(ki_aux-DatosSPI[13])*10;
                DatosSPI[14]=floor(ki_aux);
                ki_aux=(ki_aux-DatosSPI[14])*10;
                DatosSPI[15]=floor(ki_aux);}
}

// DERIVATIVA (kd)
extern float kd;
float kd_aux;
void kd_f2c(void)
{
//solo admite valores [1.00 - 49.99] de kd
if (kd < 1){kd = 1;}
if (kd >= 50){kd = 49.99;}
kd_aux = kd;
if(kd_aux<10)
{
DatosSPI[16] = 0x0000;
DatosSPI[17]=floor(kd_aux);
kd_aux = (kd_aux - DatosSPI[17])*10;
DatosSPI[18]=floor(kd_aux);
kd_aux = (kd_aux - DatosSPI[18])*10;
DatosSPI[19]=floor(kd_aux);
}
else
{
kd_aux=kd/10;
DatosSPI[16]=floor(kd_aux);
kd_aux = (kd_aux - DatosSPI[16])*10;
DatosSPI[17]=floor(kd_aux);
kd_aux = (kd_aux - DatosSPI[17])*10;
DatosSPI[18]=floor(kd_aux);
kd_aux = (kd_aux - DatosSPI[18])*10;
DatosSPI[19]=floor(kd_aux);
}
}
```

ANEXO J. Deducción de la relación proporcional voltaje/frecuencia

La ecuación de la fase a del estator es:

$$V_a(t) = R_a i_a + \frac{d\psi_a}{dt}$$

Donde ψ_a es el flujo total abrazado por la fase a del estator. Despreciando las perdidas del estator en la anterior ecuación.

$$v_a(t) = \frac{d\psi_a}{dt}$$

En el caso sinusoidal

$$\psi_a(t) = \sqrt{2}\psi_{ef}\, sen(\omega t)$$

$$v_a(t) = \sqrt{2}\psi_{ef}\, \omega sen(t)$$

$$V_{aef} = \frac{\sqrt{2}\psi_{ef}\, \omega}{\sqrt{2}}$$

Donde, $\omega = 2\pi f$ es frecuencia angular y $\psi = N\phi$ es el flujo total

Por lo tanto la ecuación para el flujo se puede escribir como

$$\psi = \frac{V}{\omega} = \frac{V}{2\pi f}$$

$$\phi = \frac{V}{2\pi N f}$$