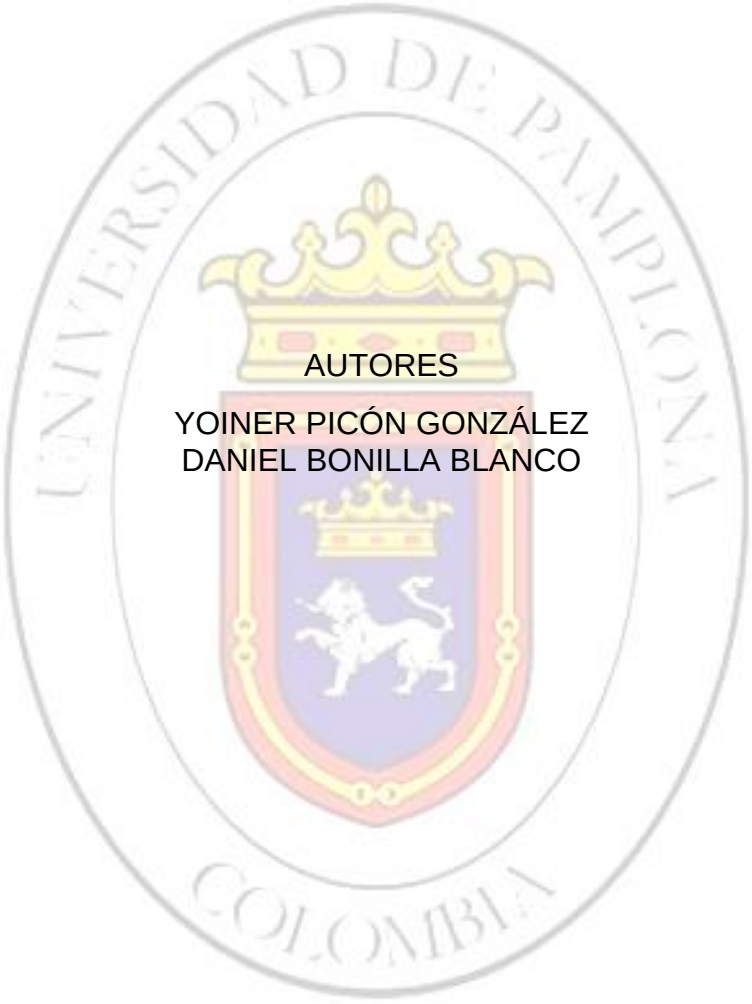
DESARROLLO DE UNA INTERFAZ GRÁFICA DE USUARIO (GUI) PARA FACILITAR EL APRENDIZAJE DELCONTROL PID AUTO SINTONIZABLE POR LÓGICA DIFUSA Y CONTROL FPD+I, CON APLICACIÓN EN UNA PLANTA DE TEMPERATURA.



AUTORES

YOINER PICÓN GONZÁLEZ
DANIEL BONILLA BLANCO

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE MECÁNICA, MECATRÓNICA E INDUSTRIAL
INGENIERÍA MECATRÓNICA
VILLA DEL ROSARIO

2022

DESARROLLO DE UNA INTERFAZ GRÁFICA DE USUARIO (GUI) PARA FACILITAR EL APRENDIZAJE DELCONTROL PID AUTO SINTONIZABLE POR LÓGICA DIFUSA Y CONTROL FPD+I, CON APLICACIÓN EN UNA PLANTA DE TEMPERATURA.

AUTORES

YOINER PICÓN GONZÁLEZ
DANIEL BONILLA BLANCO

TESIS DE INVESTIGACIÓN

DIRECTOR
ING. OSCAR MANUEL DUQUE SUÁREZ.

CODIRECTOR
ING. JAIR ELIAS ARAUJO VARGAS.

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE MECÁNICA, MECATRÓNICA E INDUSTRIAL
INGENIERÍA MECATRÓNICA
VILLA DEL ROSARIO
2022

DEDICATORIA

Esta tesis la dedicamos a nuestras familias por todo su esfuerzo en prestarnos las condiciones para alcanzar un logro importante en nuestras vidas, y a los docentes que con los mejores deseos y sus conocimientos hacen que los estudiantes crezcamos como personas y como profesionales.

AGRADECIMIENTOS

Primero que todo agradecerle a Dios por haber logrado cursar esta ingeniería, agradecerles a nuestras familias por todo su apoyo durante nuestros estudios y a los docentes de ingeniería mecatrónica porque gracias a su calidad adquirimos muy buenos conocimientos para enfrentarnos a la vida laboral.

RESUMEN

En este documento se presenta el desarrollo de una Interfaz Gráfica de Usuario (GUI) para facilitar el aprendizaje sobre el control PID auto sintonizable por lógica difusa y el control FPD+I (Proporcional-derivativo fuzzy + integral), para apreciar las ventajas y desventajas de dichos controladores la GUI también cuenta con opciones de control clásico, gráficas de simulaciones, graficas comparativas y las opciones de control para la planta de temperatura que consta de una resistencia calefactora plana, un sensor infrarrojo, un ventilador y un arduino en modo esclavo. Una vez realizada la identificación de la planta se procedio a la sintonización de los diferentes controladores con mejoras experimentales en algunos casos. Después de obtener los controladores y de realizar la GUI en Matlab y verificar que funcione correctamente se realizo la documentación de un manual de uso y/o guía para el aprendizaje con el paso a paso de cómo se debe realizar la identificación de la planta y la sintonización de todos los controladores de la GUI.

Palabras claves: control PID auto sintonizable, control FPD+I, control clásico, interfaz gráfica de usuario.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN

El presente proyecto muestra el desarrollo del diseño de una interfaz gráfica para facilitar el aprendizaje de dos importantes estrategias de control que son aplicadas en un pequeño prototipo de planta de temperatura que consta de un arduino, circuito de potencia, una resistencia calefactora y sensor infrarrojo de temperatura.

La interfaz cuenta con un curso en formato pdf con la explicación para el diseño de los dos controladores, opciones de identificación y sintonización de la planta, graficas comparativas, diagramas de bloques en simulink para la aplicación de cada controlador en el prototipo y simulaciones; también incluye un manual de uso para evitar confusiones en el manejo de la interfaz.

1 IDENTIFICACIÓN DEL PROYECTO

1.1 TÍTULO

Desarrollo de una Interfaz Gráfica de Usuario (GUI) para facilitar el aprendizaje del control PID auto sintonizable por lógica difusa y control FPD+I, con aplicación en una planta de temperatura.

1.2 PLANTEAMIENTO DEL PROBLEMA

Hoy en día es difícil aprender durante una carrera todos los tipos de controladores que existen, pero se puede ampliar el conocimiento, y para esto es posible utilizar herramientas que permitan avanzar con mucha más rapidez en el proceso de aprendizaje, en este caso siendo el control PID auto sintonizable y FPD+I estrategias de control inteligente con importantes aplicaciones en la industria, para lograr un aprendizaje más completo y teniendo en cuenta sus ventajas y desventajas, es necesario disponer de herramientas que faciliten y/o mejoren la identificación, sintonización, aplicación del control en una planta real, comparación de resultados, ahorrando costos y tiempo.

1.3 OBJETIVOS

1.3.1 Objetivo general

Diseñar una Interfaz Gráfica de Usuario (GUI) para facilitar el aprendizaje del control PID auto sintonizable por lógica difusa y control FPD+I, con aplicación en una planta de temperatura.

1.3.2 Objetivos específicos

- Construir un prototipo para implementar los controladores PID auto sintonizable por lógica difusa y FPD+I en el control de temperatura para una resistencia calefactora.
- Realizar la identificación de la planta seleccionando de algunos métodos el más adecuado.

- Realizar la sintonización de los diferentes controladores, incluyendo el clásico que ayudara en el análisis de resultados y se utilizara como referencia para los valores del control PID auto sintonizable y FPD+I.

- Diseñar el entorno visual en GUIDE de Matlab sencillo de comprender y usar, junto con su respectivo código en el lenguaje de programación propio de Matlab, que permita implementar cada uno de los controladores en el prototipo y facilite el análisis de los resultados reales y de simulaciones.

- Documentar un manual de uso y/o guía para el aprendizaje con el paso a paso de cómo se debe realizar la identificación de la planta y la sintonización de todos los controladores de la GUI.

1.4 JUSTIFICACIÓN Y LIMITACIONES

1.4.1 Justificación

El presente proyecto permitirá a los estudiantes de ingeniería mecatrónica ampliar sus conocimientos en cuanto a los tipos de controladores, en este caso al control PID auto sintonizable por lógica difusa y al control FPD+I, ayudando no solo en el aprendizaje del paso a paso para obtener dichos controladores, sino en adquirir conocimientos en cuanto a sus ventajas y desventajas observando su comportamiento en las simulaciones y en la planta real, lo cual ayudaria a definir en qué situaciones es conveniente utilizar estos dos tipos de control en la industria.

1.4.2 Limitaciones

La potencia máxima que puede suministrarse a la resistencia calefactora es de 96w y un voltaje máximo 12v (Los valores que se requieren pueden ser menores a los señalados, pero no mayores). La GUI está diseñada para estudiantes con conocimientos previos sobre control, principalmente difuso y en Matlab-Simulink.

2 ESTADO DEL ARTE

“Interactive educational tool for the design of compensators using frequency response analysis.”

En este artículo se desarrolla una interfaz gráfica para mejorar la enseñanza y aprendizaje del diseño de compensadores en donde se permite al usuario investigar varios parámetros involucrados en el diseño del compensador requerido entre los que se encuentra el sobre impulso, error constante entre otros, y se pueden obtener varias gráficas como la gráfica de Bode y la gráfica del lugar de las raíces. [1]

“Improved MATLAB GUIs for Learning Frequency Response Methods.”

Este documento muestra el diseño de dos GUI desarrolladas en UKACCControl2012 para diagramas de Bode asintóticos y la identificación a partir de los diagramas, todo esto con el objetivo de mejorar el aprendizaje de los métodos de respuesta de frecuencia. [2]

“Mehmet Cınar y Asım Kaygusuz, (2021) The practical power flow analysis program based on matlab GUI developed for educational purposes for students in smart grids: Oyamatlab. “

En este documento se realizó un programa educativo para el análisis de flujo de carga óptimo en la GUI de Matlab para que los estudiantes puedan utilizar el programa en materias relacionadas con el análisis de flujo de cargas en redes eléctricas. Este programa realiza funciones objetivas como pérdidas de energía, costo de combustible, y ajuste de voltaje mientras hace análisis de distribución de la carga eléctrica. [3]

“Chandani Sharma y Anamika Jain, (2014) Tuning of gains in basis weight using fuzzy controllers.”

El presente artículo muestra el diseño y funcionamiento del control FPD+I utilizando Simulink y recalca que los controladores de nueva generación combinan el control fuzzy con el control convencional para combinar las ventajas que ofrecen estos tipos de control. [4]

“L. Isaza, J. Vargas, F Velásquez,(2015), Diseño de una interfaz gráfica de usuario para el control de un prototipo de banda seleccionadora de piezas industriales.”

El artículo sintetiza el trabajo desarrollado al interior del grupo Macrypt que consistió en el diseño de una interfaz gráfica de usuario para el control de un prototipo de banda seleccionadora de piezas industriales , permitiendo abordar temáticas relacionadas con los sistemas de telecontrol, redes de comunicaciones industriales y aportando a usos industriales con tecnologías abiertas y flexibles. [5]

3 MARCO TEÓRICO

3.1 INTERFAZ GRÁFICA DE USUARIO (GUI)

Una Interfaz Gráfica de Usuario, son los elementos gráficos que nos ayudan a comunicarnos con un sistema o estructura. También se le conoce como el medio de interacción entre el hombre y la máquina. [5]

3.1.1 Características de una interfaz

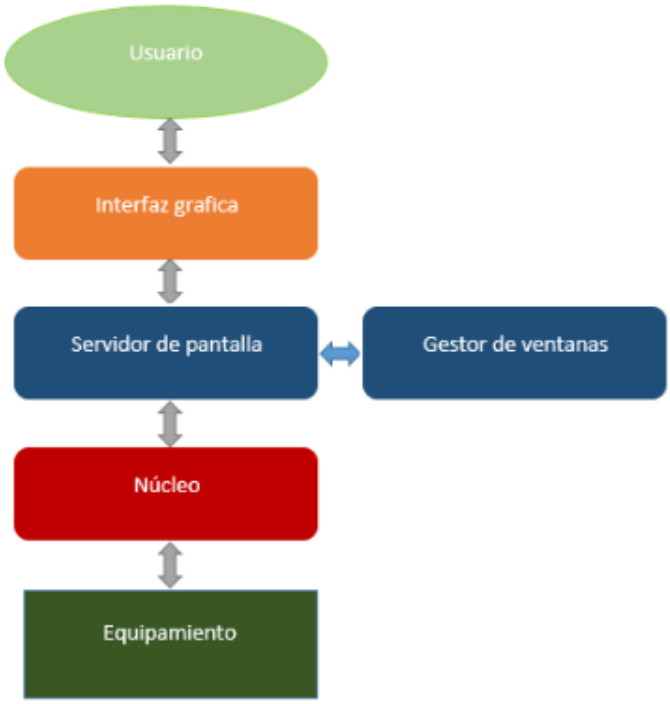
Una buena interfaz se caracteriza por:

- Ser sencilla de comprender y usar.
- La curva de aprendizaje es acelerada y es fácil recordar su funcionamiento.
- Los elementos principales son muy identificables.
- Facilitar y predecir las acciones más comunes del usuario.
- La información está adecuadamente ordenada mediante menús, iconos, barras, etc.
- La navegabilidad y la usabilidad son óptimas.
- Contiene elementos de orientación o ayuda para el usuario.

3.1.2 Capas de una interfaz gráfica

Una GUI basada en sistema de ventanas para cualquier proceso de control debe contar con las siguientes capas de la figura 1, en donde el servidor de pantalla y el núcleo vienen integrados en una computadora, el gestor de ventanas es el software donde se diseña el entorno visual, y el equipamiento es el hardware incluyendo sensores y actuadores. [6]

Figura 1 Capas de interfaz gráfica



Fuente: Daniel Sánchez, 2014. [6]

3.1.3 Puntos de vista distintos de la GUI

Existen tres puntos de vista distintos en una GUI: el Modelo del Usuario, el Modelo del Diseñador y el Modelo del Programador. [7]

- Modelo del Usuario: el usuario tiene su propia visión del sistema y espera que se comporte de determinada manera.

- Modelo del Diseñador: el diseñador es quién se encarga de unir las ideas, necesidades y deseos del usuario, con las herramientas que dispone el programador para desarrollar el software. Éste modelo consta de tres partes: la Presentación que es lo primero que llama la atención del usuario; luego adquiere más importancia la Interacción que es donde el usuario constata si el producto satisface sus expectativas. La tercera y última parte es la de Relaciones entre objetos aquí es donde se define la relación entre el modelo mental del usuario y los objetos de la Interfaz.

- Modelo del Programador: es el modelo más fácil de visualizar porque se puede especificar formalmente. Este modelo consta de los objetos que

manipula el programador, distintos a los que maneja el usuario (el programador maneja una base de datos, el usuario la llama agenda o contactos). El usuario no ve los objetos que maneja el programador. Si bien el programador conoce de plataforma de desarrollo, sistema operativo, lenguajes y herramientas de programación, especificaciones; no significa que tenga la habilidad de proporcionar al usuario modelos más adecuados.

Los distintos modelos nos permiten conocer cómo visualizan la GUI cada uno de los 'actores'. Cada uno de éstos son protagonistas al momento del diseño. Conocer cada punto de vista permite comprender los principios y reglas.

3.1.4 Principios de diseño

Existe gran cantidad de información respecto a principios de diseño de la GUI pero se resaltan a continuación algunos de los más importantes. [8]

- Familiaridad del usuario: significa que la interfaz debe utilizar términos e imágenes conocidos por el usuario; y los objetos que manipula el sistema deben estar relacionados con el ámbito de trabajo.

- Recuperación de estados: éste es uno de los principios más importantes al diseñar una Interfaz. Es inevitable cometer errores, por lo tanto el sistema le debe proporcionar al usuario la manera de subsanarlos o volver a estados anteriores. Éste principio involucra varias acciones como pedir al usuario que confirme acciones destructivas, que el usuario pueda deshacer, etc.

- Guía de usuarios: la Interfaz debe proporcionar al usuario asistencia, ayuda. No sólo cuando se cometen errores sino también cuando no se sabe qué hacer o cómo hacer alguna tarea. Esta ayuda debe estar integrada al sistema.

- Realimentación: la interfaz debe dar inmediatamente alguna respuesta a cualquier acción del usuario. Por ejemplo: movimiento del cursor, resaltar la opción elegida de un menú, comunicar el éxito o fracaso de una operación, reflejar el estado de los objetos.

- Eficiencia: se debe considerar la productividad como ideal a lograr. El usuario no debe esperar la respuesta del sistema por tiempo prolongado; los mensajes de ayuda, menús y etiquetas deben ser sencillos y deben utilizar palabras claves para poder transmitir fácilmente a qué hacen referencia.

- Potenciar la sensación de control del usuario sobre el sistema, especialmente para los usuarios sin experiencia: que la interfaz sea intuitiva (utilizar iconos, modelos, métodos, etc. consistentes con otras aplicaciones y con el mundo real), facilitar la exploración (todas las operaciones deben ser accesibles desde el menú principal), permitir cancelar y deshacer operaciones, etc.

3.1.5 GUIDE de Matlab

Graphical User Interface Development Enviroment (GUIDE) es un entorno de programación visual disponible en MATLAB que permite diseñar interfaces gráficas y ejecutar programas. Esta herramienta es muy útil cuando se tiene un programa MATLAB que necesita un ingreso continuo de datos o la interacción activa con el usuario. Con GUIDE es posible crear de forma sencilla ventanas y diálogos que contengan los componentes básicos, como son etiquetas, campos de texto, botones, contenedores, entre otros. [9]

La siguiente tabla 1 muestra una descripción de los componentes disponibles en el entorno de desarrollo.

Tabla 1 Componentes del entorno de desarrollo GUIDE

Control	***Valor de estilo***	***Descripción***
Check box	'checkbox'	Indica el estado de una opción o atributo
Editable Text	'edit'	Caja para editar texto
Pop-up menu	'popupmenu'	Provee una lista de opciones
List Box	'listbox'	Muestra una lista deslizable
Push Button	'pushbutton'	Invoca un evento inmediatamente
Radio Button	'radio'	Indica una opción que puede ser seleccionada
Toggle Button	'togglebutton'	Solo dos estados, "on" o "off"
Slider	'slider'	Usado para representar un rango de valores
Static Text	'text'	Muestra un string de texto en una caja
Panel button		Agrupa botones como un grupo
Button Group		Permite exclusividad de selección con los radio button

Fuente: Diego Orlando Barragán Guerrero.(2008). [10]

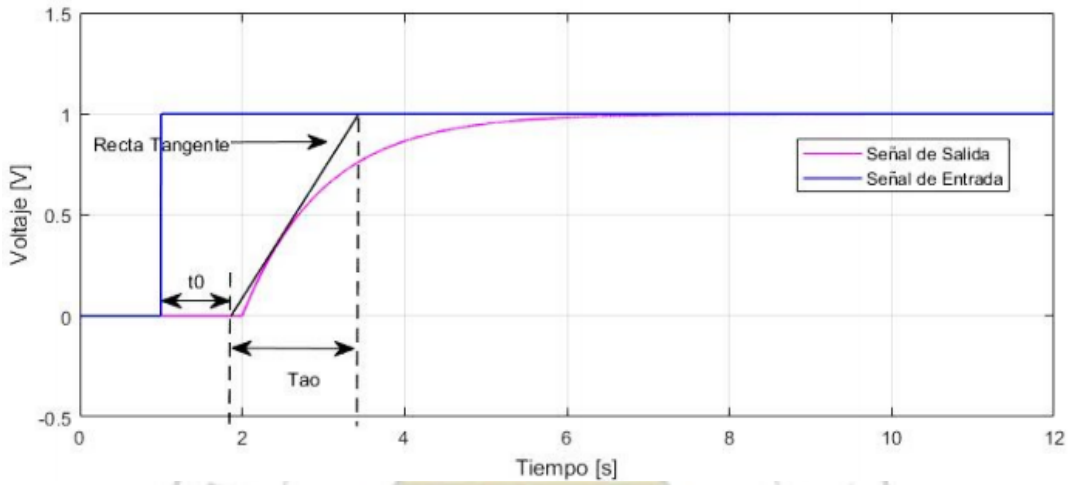
3.2 MÉTODOS DE IDENTIFICACIÓN DE LA PLANTA

3.2.1 Método de Ziegler y Nichols

Este método consiste en trazar una línea tangente a la respuesta del sistema o curva de reacción en el punto de inflexión o punto de máxima pendiente. Para

obtener el modelo, se debe identificar la ganancia del sistema, el tiempo de estabilización y el tiempo muerto como se muestra en la figura 2. [11]

Figura 2 Método de identificación de Ziegler-Nichols



Fuente: Jorge Novoa, Wuemdell Martínez, 2019. [11]

3.2.1.1 Calcular recta tangente

Si $f(x)$ y $f'(x)$ son derivables en (a), se dice que (a) es un punto de inflexión si se cumple que $f''(a) = 0$ y $f''' \neq 0$.

Para calcular el punto de inflexión se halla la derivada segunda y se calcula sus raíces, luego calculamos la imagen de x reemplazando en $f(x)$ como se muestra en el siguiente ejemplo con las ecuaciones (3.1), (3.2) y (3.3) para la función $f(x) = x^3 - 3x + 2$

$$f'(x) = 3x^2 - 3 \quad (3.1)$$

$$f'(x) = 6x \quad (3.2)$$

$$x = 0, es\ la\ unica\ raiz\ de\ f'(x)$$

Se reemplaza x en $f(x)$

$$f(x) = (0)^3 - 3(0) + 2 \quad (3.3)$$

La función tiene punto de inflexión en (0,2).

Ahora con la ecuación (3.4) de la recta tangente y siendo $x = 0$ se obtiene los valores de $f(x_0)$ y $f'(x_0)$.

$$y = f(x_0) + f'(x_0)(x - x_0) \tag{3.4}$$

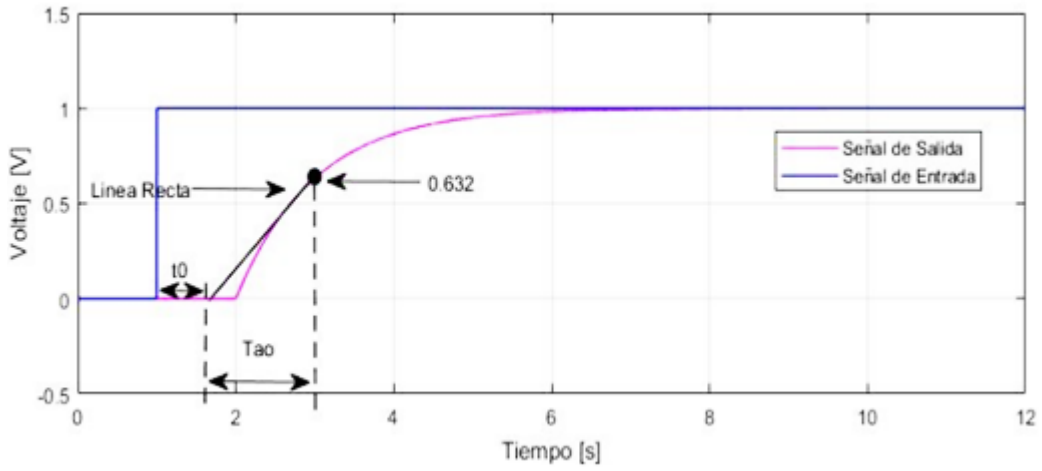
Resolviendo la ecuación (3.4) tenemos la ecuación (3.5) para la recta tangente.

$$y = 2 - 3x \tag{3.5}$$

3.2.2 Método de Miller

Miller describe este método como una variación del método de Ziegler-Nichols, en donde también se debe trazar una línea tangente en el punto de inflexión y el tiempo muerto también se calcula desde el inicio del sistema hasta donde inicia la recta tangente, el cambio se presenta en el tiempo de estabilización, ya que este sería desde donde inicia la recta tangente hasta cuando se llega al 63.2% de la respuesta del sistema como en la figura 3. [11]

Figura 3 Método de identificación de Miller



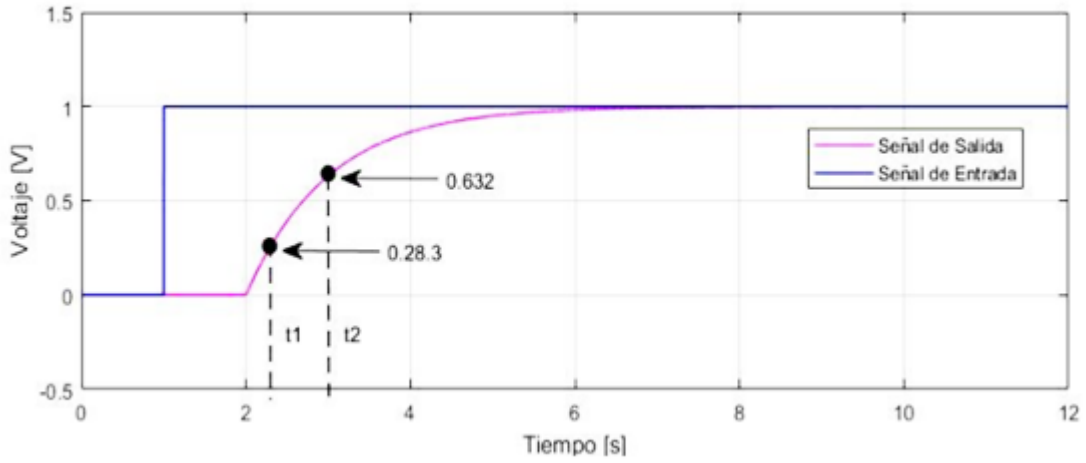
Fuente: Jorge Novoa, Wuemdell Martínez, 2019. [11]

3.2.3 Método de Smith

(Smith 1991) menciona que esta técnica no requiere que se trace la recta tangente en el punto de inflexión; el autor propone la selección de dos puntos en la curva de reacción del proceso para hallar τ y $t0$, para esto se construye dos ecuaciones con dos incógnitas seleccionando dos puntos en la curva de reacción. Smith propuso

que estos puntos fueran el 28.3% y 63.2% del valor final del proceso como se observa en la figura 4. [11]

Figura 4 Método de identificación de dos puntos de Smith

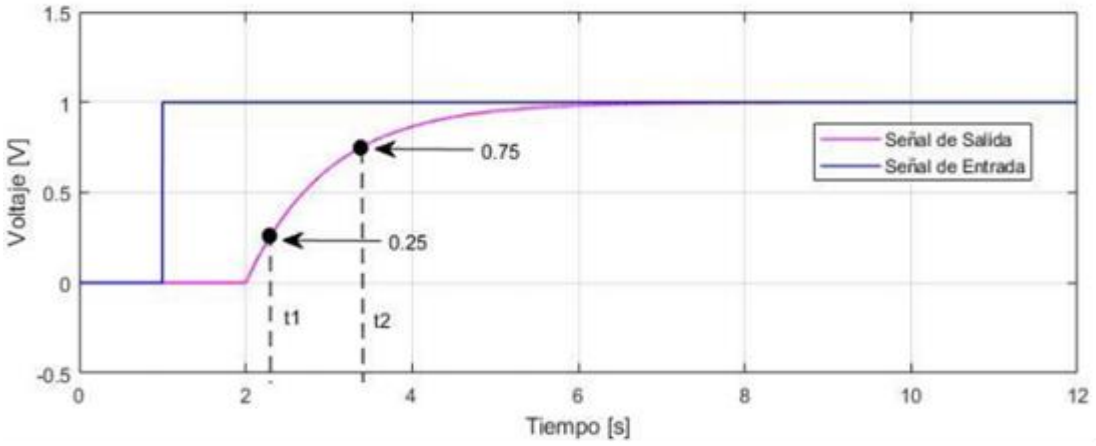


Fuente: Jorge Novoa, Wuemdell Martínez, 2019. [11]

3.2.4 Método de Ho et al

Como se observa en la figura 5, el autor propone a diferencia de Smith escoge como puntos en la curva de reacción cuando se alcance el 35% y el 85% del valor final del proceso, para hallar τ y $t0$.

Figura 5 Método de identificación de Ho et al.



Fuente: Jorge Novoa, Wuemdell Martínez, 2019. [11]

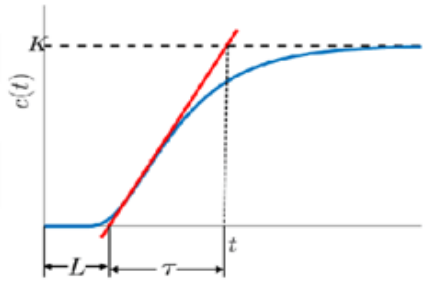
3.3 MÉTODOS DE SINTONIZACIÓN PID DE LA PLANTA

3.3.1 Primer método de Ziegler y Nichols

El Método consiste en obtener los parámetros del sistema de primer orden más tiempo muerto trazando la recta tangente en el punto de inflexión de la curva y determinar las intersecciones de la línea tangente con el eje del tiempo y el valor en el que la planta se estabiliza, ver figura 6. [8]

Figura 6 Primer método de sintonización Ziegler-Nichols

Controlador	K_p	τ_i	τ_d
P	$\frac{\tau}{KL}$	∞	0
PI	$0.9\frac{\tau}{KL}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{\tau}{KL}$	$2L$	$0.5L$



Fuente: Sergio Castaño, 2020. [12]

3.3.2 Método de Chien, Horns y Reswick

Este método de sintonización propone dos opciones de desempeño para el controlador PID, La respuesta más rápida posible sin sobre impulso y la respuesta más rápida posible con 20% de sobre impulso; en la figura 7 se muestra el método para la respuesta más rápida posible sin sobre impulso.

Figura 7 Método de sintonización CHR

Controlador	K_c	τ_i	τ_d
P	$\frac{0.3\tau}{K\theta}$	—	—
PI	$\frac{0.6\tau}{K\theta}$	4θ	—
PID	$\frac{0.95\tau}{K\theta}$	2.375θ	0.421θ

Fuente: Sergio Castaño, 2020. [13]

3.3.3 Método en Lazo Abierto de Cohen y Coon

Se emplea el mismo test que el método de Chien, Horns y Reswick. La sugerencia para los parámetros tiene en cuenta el grado de autorregulación de la planta, mensurado por la relación: $R = L/\tau$ para calcular de acuerdo a la figura 8.

Figura 8 Método de sintonización de Cohen y Coon

CONTROLADOR	Kc	T_I	T_D
P	$\frac{1}{KR}\left[1+\frac{1}{3}R\right]$	∞	0
PI	$\frac{1}{KR}\left[0.9+\frac{1}{12}R\right]$	$L\left[\frac{30+3R}{9+20R}\right]$	0
PD	$\frac{1}{KR}\left[\frac{5}{4}+\frac{1}{6}R\right]$	∞	$L\left[\frac{6-2R}{22+3R}\right]$
PID	$\frac{1}{KR}\left[\frac{4}{3}+\frac{1}{4}R\right]$	$L\left[\frac{32+6R}{13+8R}\right]$	$L\left[\frac{4}{11+2R}\right]$

Fuente: Universidad nacional de Tucumán, 2017.[14]

3.4 CONTROL PID AUTOSINTONIZABLE POR LÓGICA DIFUSA

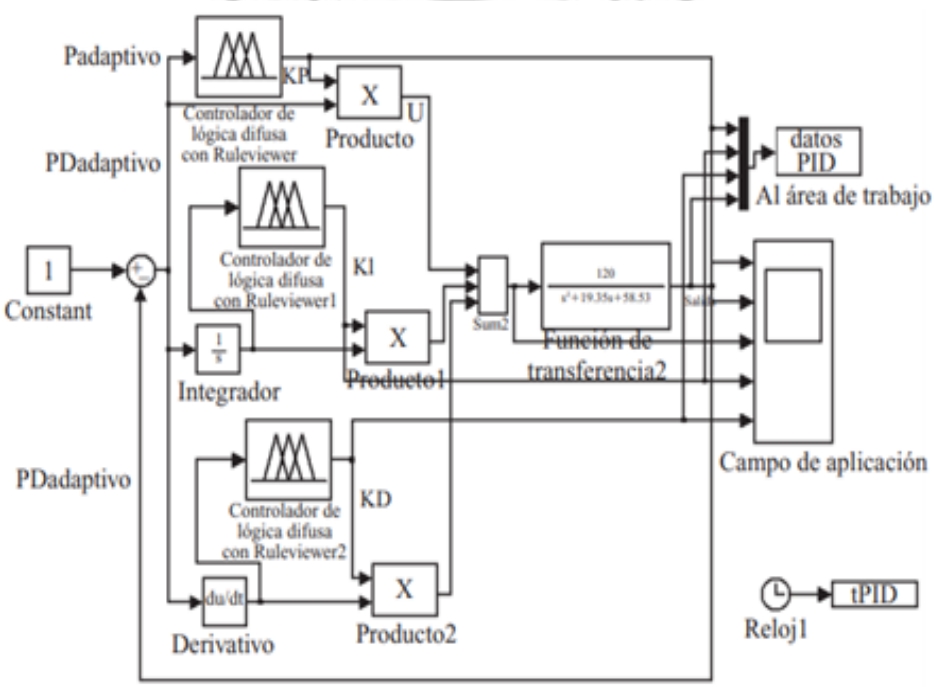
En esencia un controlador lógico difuso contiene un algoritmo que es capaz de convertir una estrategia de control lingüística en una estrategia de control automático. Con la lógica difusa se puede diseñar aplicaciones para que las máquinas respondan con mayor inteligencia a la imprecisión y a las condiciones del mundo exterior, con lo que se busca imitar el comportamiento humano. La creación de una máquina con lógica difusa es forjar un sistema experto, en donde el comportamiento de la máquina va a estar basado totalmente en el conocimiento del experto o de la persona que aporta sus conocimientos empíricos para el funcionamiento de ésta. [15]

El término esencial es el término proporcional, el cual multiplica directamente al "error" por una constante Kp (ganancia) y la alimenta a la planta; conforme el error decrece, el producto del mismo por la Kp también lo hace. En ocasiones es imposible que un sistema llegue al estado deseado simplemente con un término de Kp; es ahí cuando entra la acción integradora, es decir, el término I. Para esto es necesario integrar el error durante cierto tiempo y sumar a la acción proporcional esta acción integradora. De este modo el error de estado estable se elimina. Por

último, la acción derivativa o término “D” es alimentado por la derivada o cambio del error, de este modo el control puede hasta cierto punto predecir la tendencia del sistema para anticiparse y eliminar el sobretiro del sistema. Tanto los términos P, I y D requieren de una ganancia para actuar en forma apropiada. [15]

El controlador de la figura 9 reemplaza mediante el bloque fuzzy las constantes del control clásico Kp, Ki y Kd por variables que cambian su valor de acuerdo a la dinámica del sistema, por ejemplo en el caso proporcional el valor de Kp varía de acuerdo al valor del error. [15]

Figura 9 PID auto sintonizable difuso.

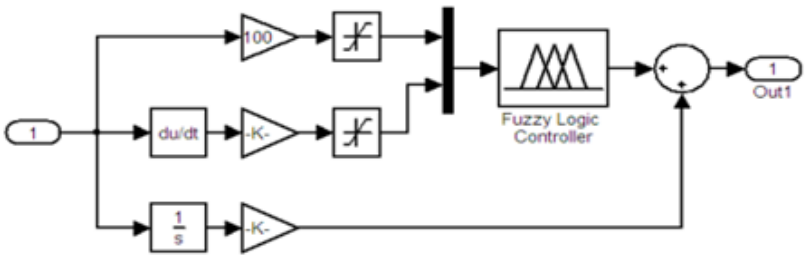


Fuente: Pedro Ponce. 2010. [15]

3.5 CONTROL FPD+I

En la estrategia de control difuso PD+I, la acción de control resultante de la parte proporcional y derivativa está definida por lógica difusa, y para evitar principalmente error en estado estable se suma la acción integral como se muestra en la figura 10.

Figura 10 Controlador FPD+I



Fuente: Abdullah Almeshal. 2013. [16]

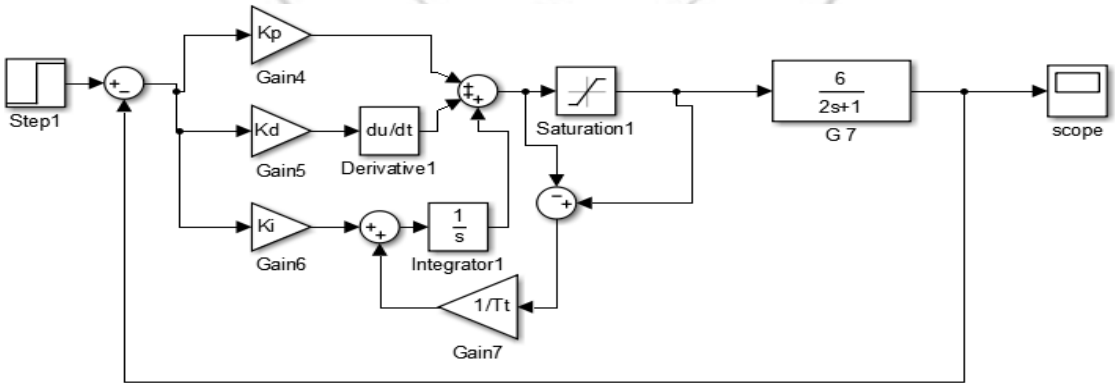
3.6 CONTROL CON ANTI-WINDUP

El fenómeno windup en los controladores PID convencionales se presenta cuando se tiene una planta con entrada acotada y la acción de control supera los límites del actuador; por consiguiente, la integral del error se acumula continuamente hasta degradar la respuesta del controlador.

La ventaja de este esquema anti-windup de la figura 11 es que los cambios en la integral no son abruptos cuando se presenta saturación, debido a que la retroalimentación permite hacer seguimiento de la salida del actuador a través de la ganancia Tt definida por la ecuación (3.6). [17]

$$Tt = \sqrt{td * ti} \tag{3.6}$$

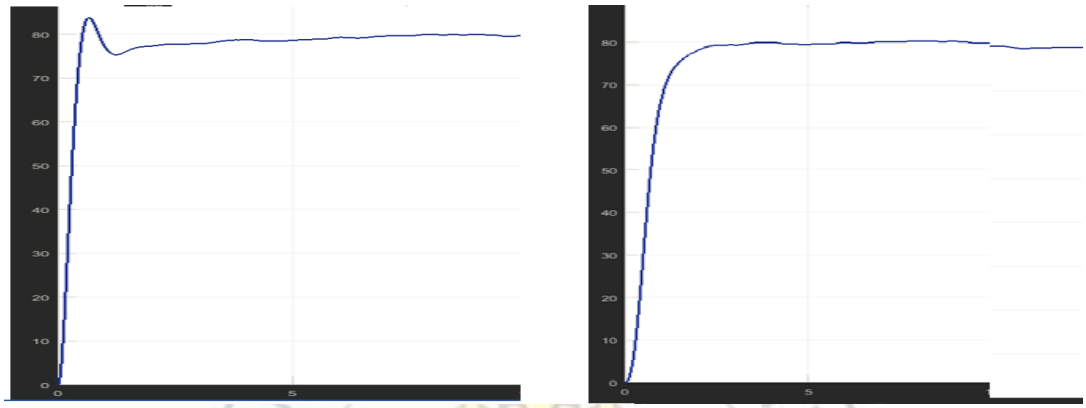
Figura 11 Esquema anti-windup



Fuente: Autores

En la figura 12 se compara la respuesta del mismo proceso con y sin anti-windup.

Figura 12 Respuesta con y sin anti-windup



Fuente: Autores

4 CRITERIOS DE DISEÑO DEL PROTOTIPO

4.1 CRITERIOS DE SELECCIÓN DE RESISTENCIA CALEFACTORA

Para la selección de la resistencia calefactora se debe tener en cuenta el tamaño tratando de seleccionar la más pequeña, que no exceda 96w y máximo 12 voltios de alimentación; es importante que la temperatura se acerque a los 100 °C y tendrá mejor porcentaje de puntuación.

Para cada una de las características se establece un porcentaje máximo de puntuación.

Tamaño: 25%

Voltaje de alimentación: 15%

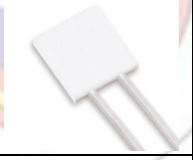
Potencia: 20%

Temperatura máxima: 25%

Costo: 15%

De acuerdo a la tabla 2, se selecciona la resistencia calefactora con mejor puntuación porcentual.

Tabla 2 Características de selección resistencia calefactora

Característica	Modelo 1 	Modelo 2 	Modelo 3 	Modelo4 
Tamaño (mm)	35*21*5	10*10*1	3*13	35*20*5
Voltaje	12	12	12	12
Potencia (w)	10	5	12	30
Temperatura máx. (°C)	80	110	600	120
Costo	90.000	178.000	62.000	53.000
Puntuación (%)	73	88	70	80

Fuente: EBay, Mercado libre, Amazon.

De la tabla 2 se puede inferir el modelo 2 como la mejor opción, presentando mejor tamaño y temperatura máxima de trabajo cercana a 100 °C, además con una potencia baja de 5w evita que disipe demasiado calor, permitiendo ser menos exigente en cuanto al material de la carcasa del prototipo.

4.2 CRITERIOS DE SELECCIÓN DEL SENSOR DE TEMPERATURA

El sensor infrarrojo aunque ofrece ventajas importantes sobre los demás tipos, no es el más utilizado en los sistemas de medición de temperatura; para este caso en particular por su tamaño y no hacer contacto con la resistencia calefactora facilita su instalación y se opta por seleccionar la mejor opción dentro del tipo de sensor infrarrojo.

En la selección del sensor se debe tener en cuenta que el rango de temperatura sea superior al de la resistencia calefactora seleccionada y preferiblemente un voltaje de alimentación de 3 a 5 voltios para conectarlo directamente al arduino, además debe ser del menor tamaño posible.

Para cada una de las características se establece un porcentaje máximo de puntuación.

Tamaño: 25%

Voltaje de alimentación: 20%

Angulo de visión: 10%

Rango de temperatura: 25%

Costo: 20%

De acuerdo a la tabla 3, se selecciona el sensor de temperatura con mejor puntuación porcentual.

Tabla 3 Características de selección del sensor

Característica	AMG8833 IR	D203S	MLX90614	EK1254
				
Tamaño (mm)	25*25*6	10*25*5	27*11	35*32*14

Voltaje	5	5	5	5
Angulo de visión	60	70	20	35
Rango Temperatura (°C)	0 a 80	-30 a 70	-20 a 120	0 a 100
Costo	103.000	178.000	89.000	57.000
Puntuación (%)	73	70	90	85

Fuente: EBay, Ferretrónica.

De la tabla 3, el sensor de temperatura más apropiado es el MLX90614 con el mejor tamaño y mayor rango de temperatura.

4.3 HARDWARE DE ADQUISICIÓN DE DATOS

Teniendo en cuenta que existen diferentes dispositivos de adquisición de datos con diferentes estándares de comunicación se opta por seleccionar uno de entre los más comunes y utilizado por los estudiantes.

Para seleccionar el más adecuado, se debe tener en cuenta las características del sensor que requiere de los pines SDA, SCL, alimentación al sensor de 5v, 3 tierras y 2 PWM.

Para cada una de las características se establece un porcentaje máximo de puntuación; como ejemplo, en el caso del tamaño la puntuación sería mayor entre más pequeña sea la tarjeta y menos exceso presente de acuerdo a los requerimientos de la planta.

Tamaño: 20%

Voltaje de alimentación: 20%

Pines PWM: 15%

Frecuencia: 20%

Corriente de salida pin: 10%

Costo: 15%

De acuerdo a la tabla 4, la tarjeta de adquisición de datos con mejor puntuación porcentual es el arduino pro 328 con el mejor tamaño, pero por ser más común entre los estudiantes se utiliza el arduino mega 2560; la tarjeta de desarrollo raspberry pi 4 es la menos indicada al ofrecer un exceso de capacidad frente a los requerimientos de la planta, generando un costo innecesario.

Tabla 4 Características de selección de tarjeta de adquisición de datos

Característica	Arduino Mega 2560 	Arduino pro 328 	Raspberry Pi 4 
Tamaño (mm)	101*53	18*33	85*53
Voltaje	7	5	5
Pines PWM	15	6	4
Frecuencia (Mhz)	16	16	1.5Ghz
Corriente salida pin (mA)	40	40	16
Costo	82.000	32.000	297.000
Puntuación (%)	77	90	50

Fuente: Ferretrónica, EBay.

4.4 DISEÑO E IMPLEMENTACIÓN DE LA ESTRUCTURA DEL PROTOTIPO, CIRCUITO DE POTENCIA

4.4.1 Diseño de circuito de potencia

Para evitar posibles daños al arduino o incluso a la computadora se aisló el circuido de potencia mediante un opto acoplador PC817A como se observa en la figura 13, en donde R12(1) representa la salida PWM del arduino.

Teniendo en cuenta que el voltaje v_d debe estar entre 1.2v y 1.4v para el opto acoplador de acuerdo al anexo 6 se considera un valor promedio de 1.3v, el voltaje

de entrada v_{in} de la salida del arduino es de 5v, y la resistencia equivalente para el diodo interno es de 78Ω aproximadamente, se calcula la resistencia R12 despejando de la ecuación (4.1) y obteniendo el valor en la ecuación (4.2).

$$v_d = v_{in} * \frac{78\Omega}{R_{12}+78\Omega} \quad (4.1)$$

$$R_{12} = \frac{v_{in}*78\Omega}{v_d} - 78\Omega = 222\Omega \quad (4.2)$$

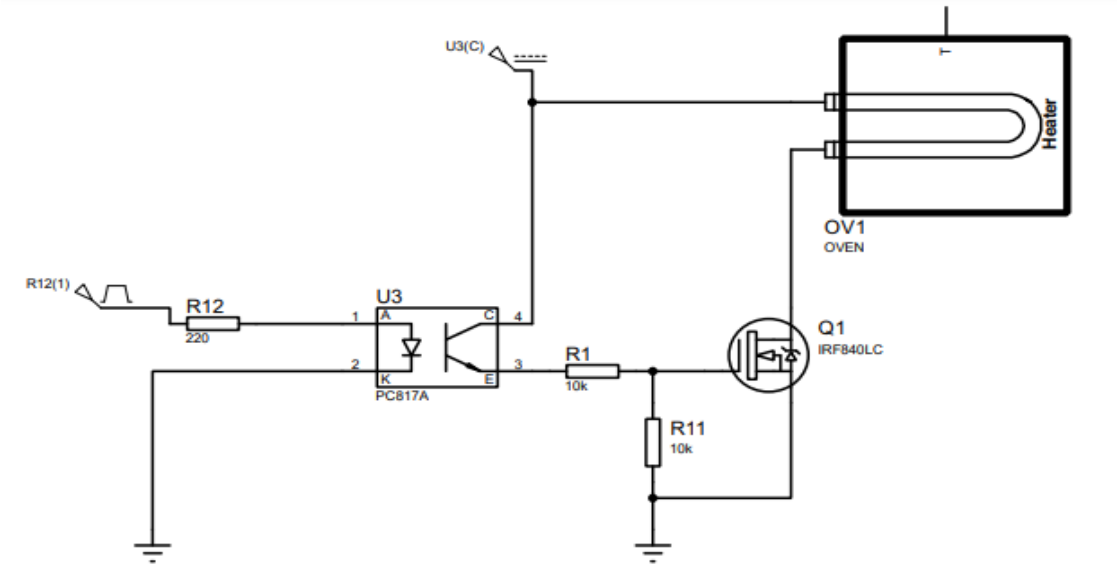
En el caso del mosfet IRF840 que necesita entre 2 a 4 voltios en su compuerta para permitir el paso de corriente de acuerdo al anexo 7, sabiendo que U3(C) genera un voltaje de entrada de 12v y se desea hacer llegar 6v a la compuerta del mosfet, se calcula R1 y R11 en 10KΩ como se muestra en la figura 13.

Se debe tener en cuenta para seleccionar el mosfet la corriente y el voltaje de la resistencia calefactora de acuerdo a la ecuación (4.3) y (4.4), en donde v_{DD} es el voltaje de la fuente, v_D es el voltaje sobre el mosfet, I_R y R_R la corriente y el valor de la resistencia calefactora.

$$v_D = v_{DD} - I_R * R_R \quad (4.3)$$

$$v_D = 12v - 0.326A * 36\Omega = 264mv \quad (4.4)$$

Figura 13 Circuito de potencia



Fuente: Autores

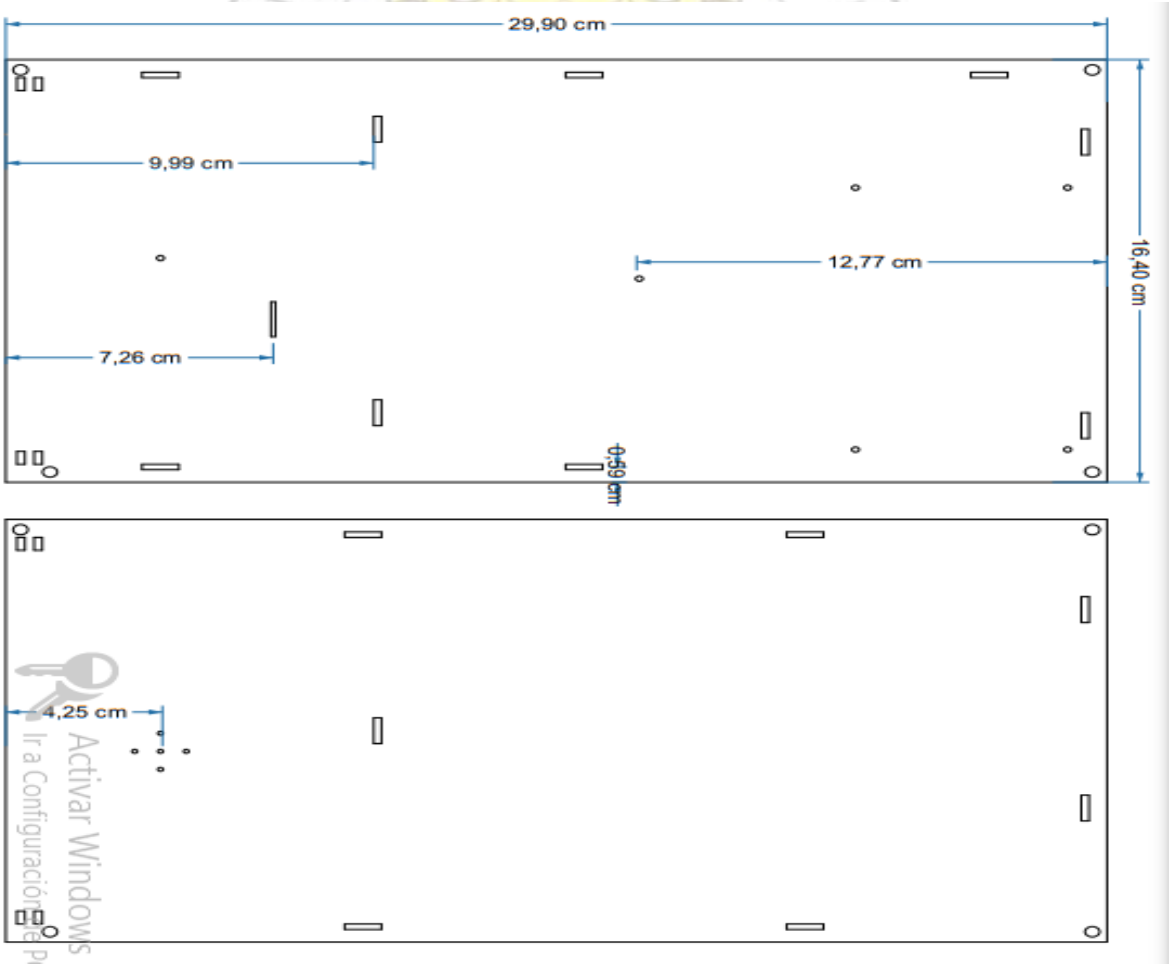
4.4.2 Diseño de la estructura del prototipo.

Teniendo en cuenta que el punto máximo de temperatura que soporta el acrílico es de 120°C para el ablandamiento, y se recomienda no exceder los 80°C de temperatura continua; se realiza el diseño manteniendo una distancia considerable entre el acrílico y la resistencia calefactora.

Los planos de diseño se realizaron en el programa CorelDRAW, para realizar el corte en acrílico. En la figura 14 se muestra la parte inferior y superior de la carcasa, en la figura 15 la lámina lateral derecha e izquierda, en la figura 16 la lámina posterior y frontal, en la figura 17 las partes de la carcasa para el circuito de potencia.

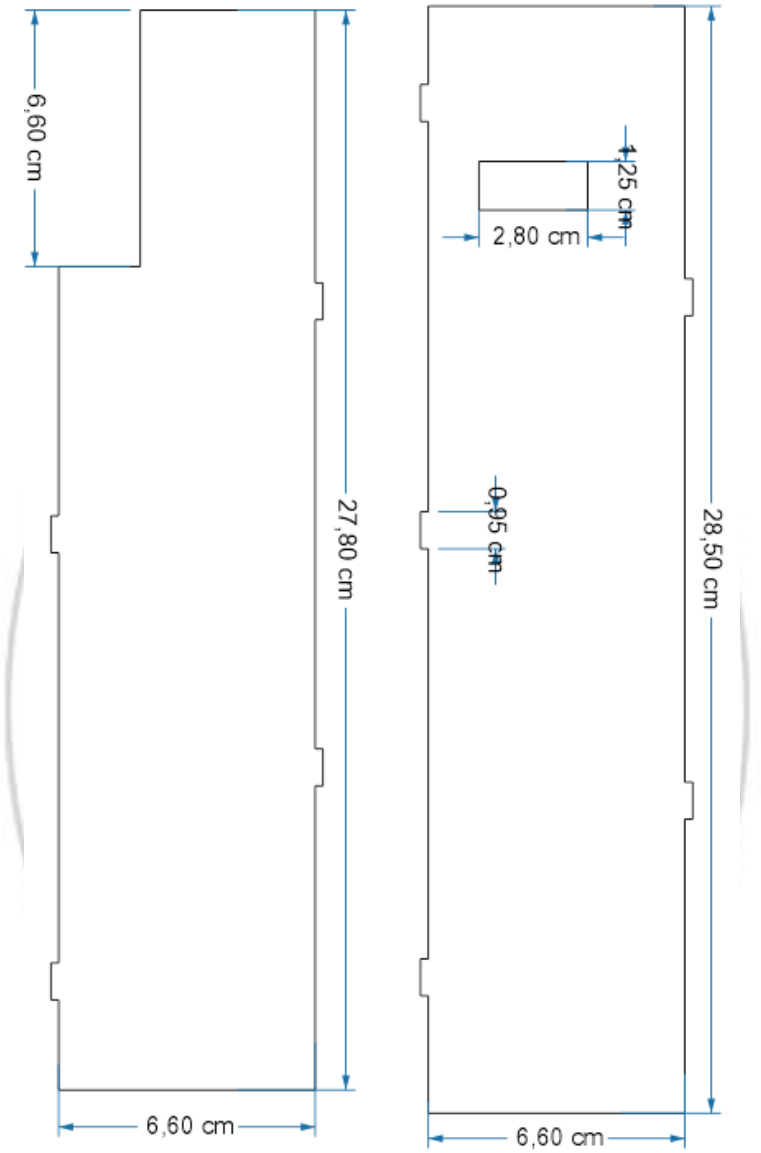
La vista en 3D para la carcasa del circuito se muestra en la figura 18, y para la estructura completa en la figura 19.

Figura 14 Lámina inferior y superior



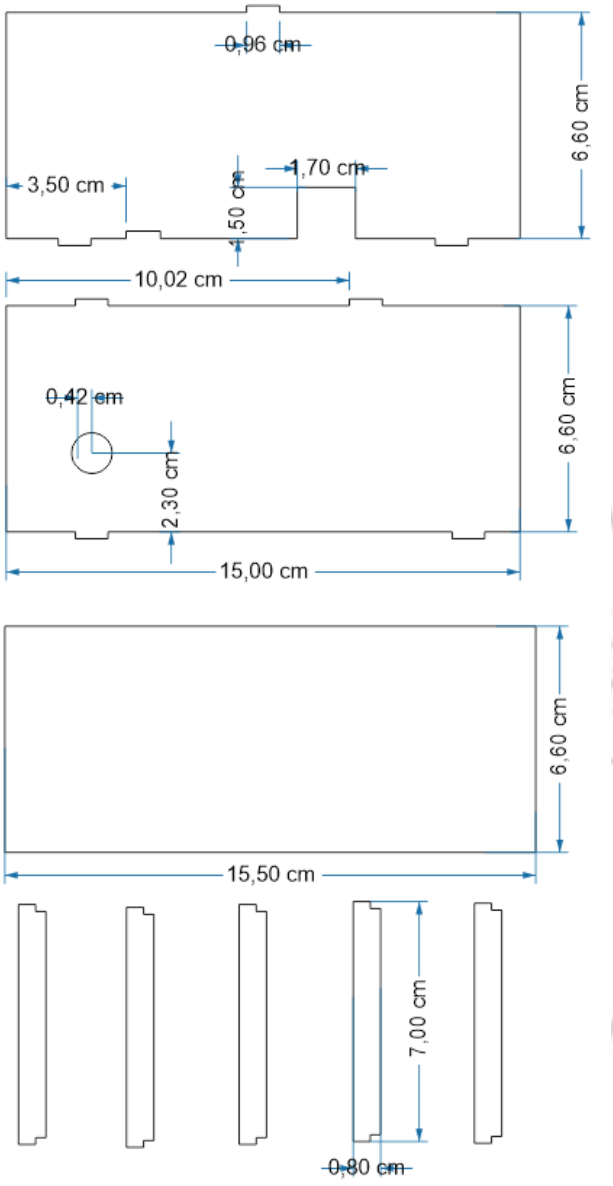
Fuente: Autores.

Figura 15 Lámina lateral derecha e izquierda



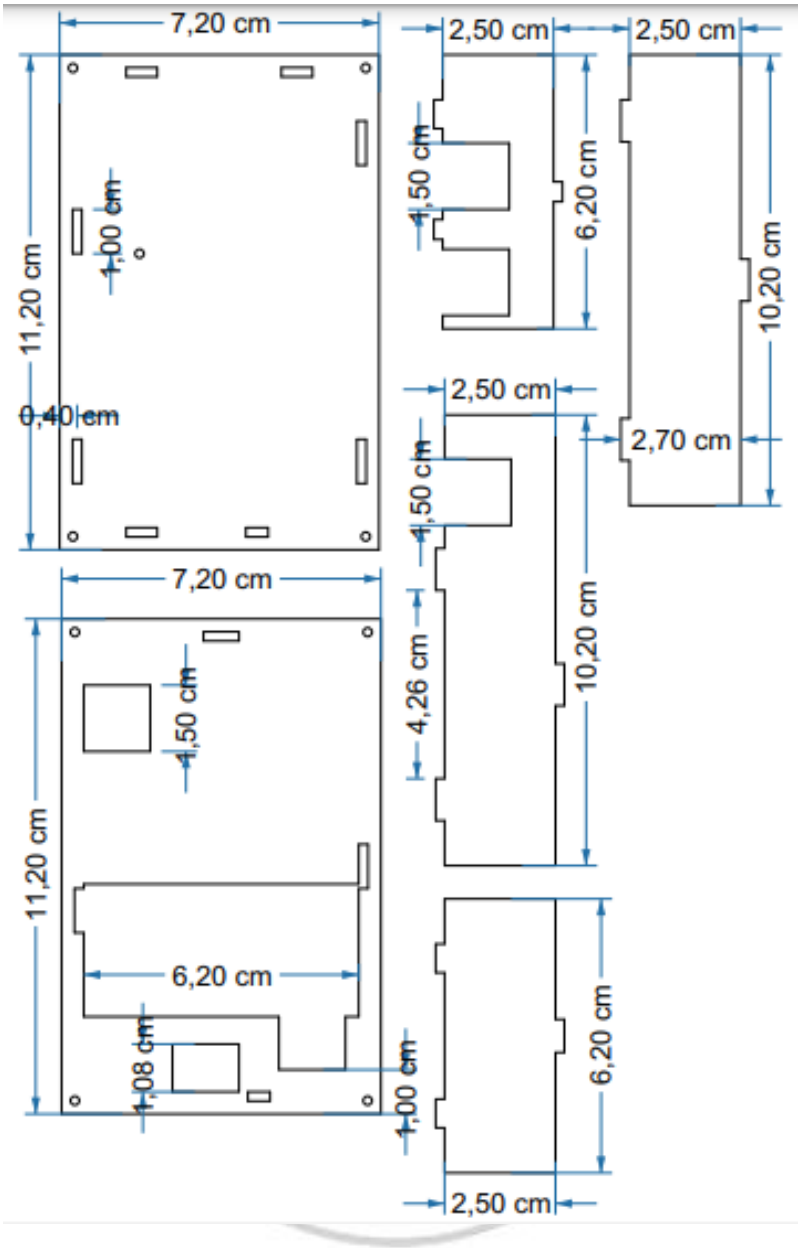
Fuente: Autores.

Figura 16 Separador central, lamina posterior y frontal



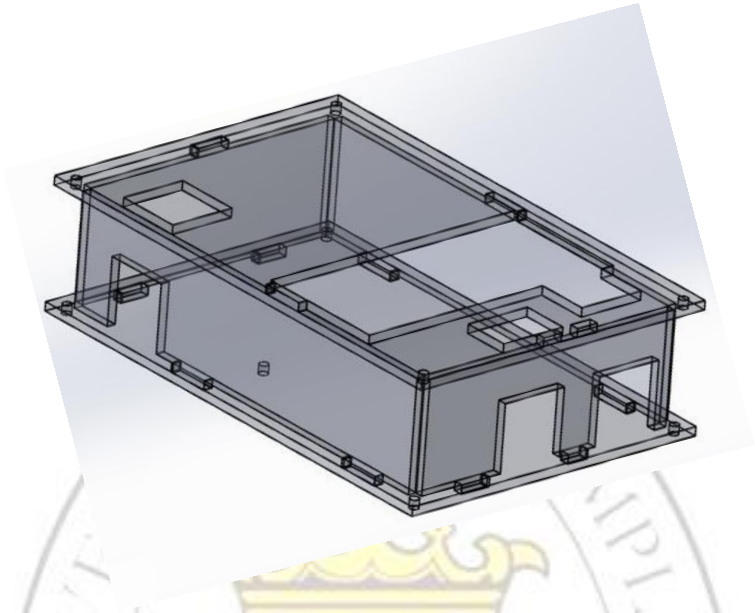
Fuente: Autores.

Figura 17 Partes de la carcasa para circuito de potencia



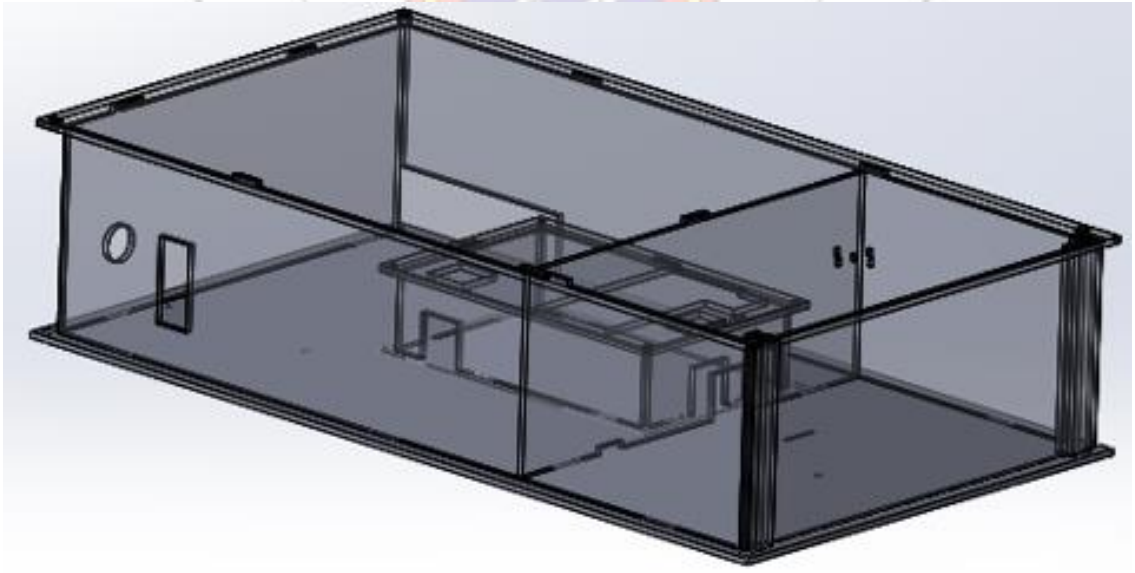
Fuente: Autores.

Figura 18 3D carcasa de circuito de potencia



Fuente: Autores

Figura 19 3D Estructura completa



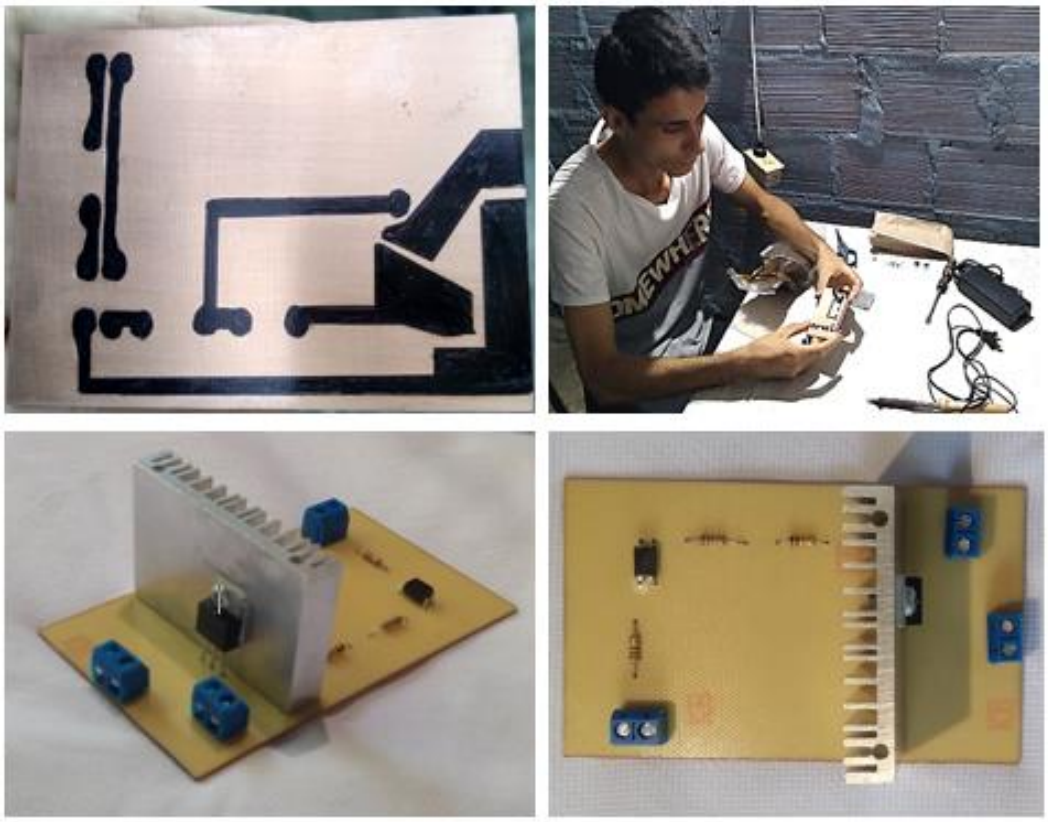
Fuente: Autores

4.4.3 Implementar el diseño del circuito y estructura

4.4.3.1 Montaje del circuito

En la figura 20 está ilustrado el montaje del circuito en la baquelita.

Figura 20 Montaje del circuito

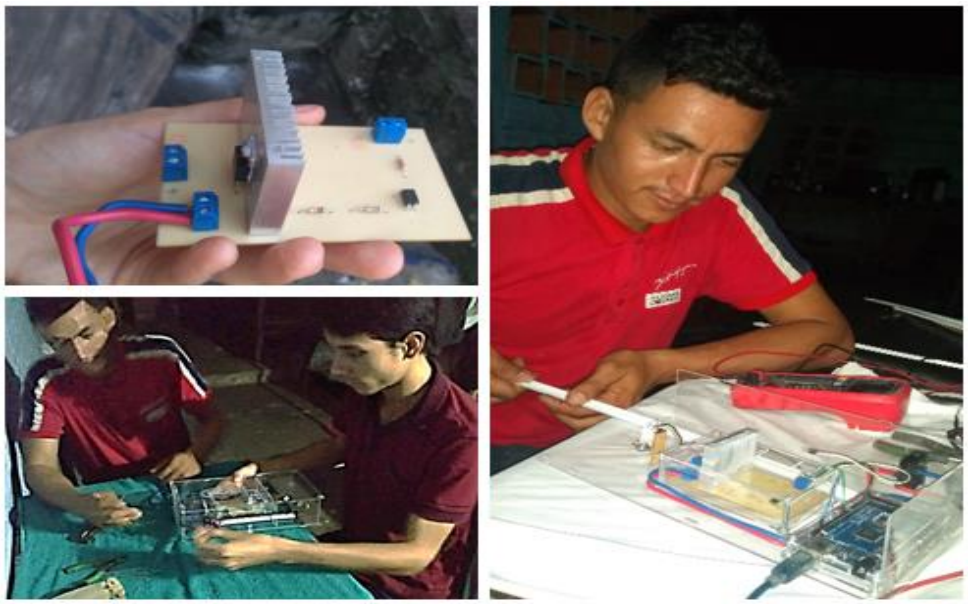


Fuente: Autores

4.4.3.2 Montaje de estructura del prototipo

En la figura 21 se muestra el montaje de la carcasa del prototipo y el circuito.

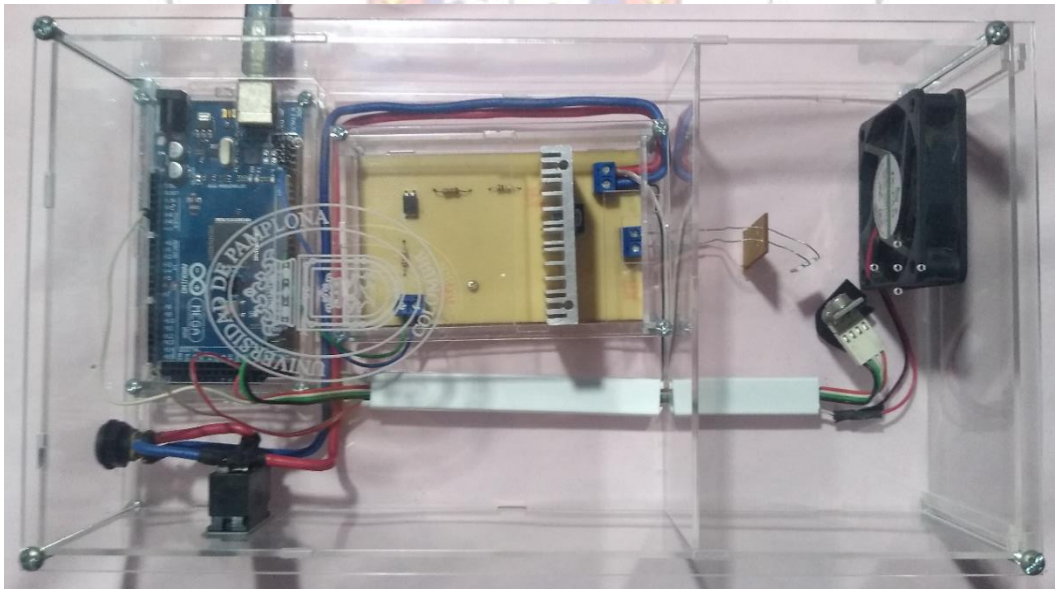
Figura 21 Montaje del prototipo



Fuente: Autores.

En la figura 22 se ilustra la vista superior del montaje del prototipo.

Figura 22 Vista superior del prototipo



Fuente: Autores

5 DISEÑO DE INTERFAZ GRÁFICA DE USUARIO

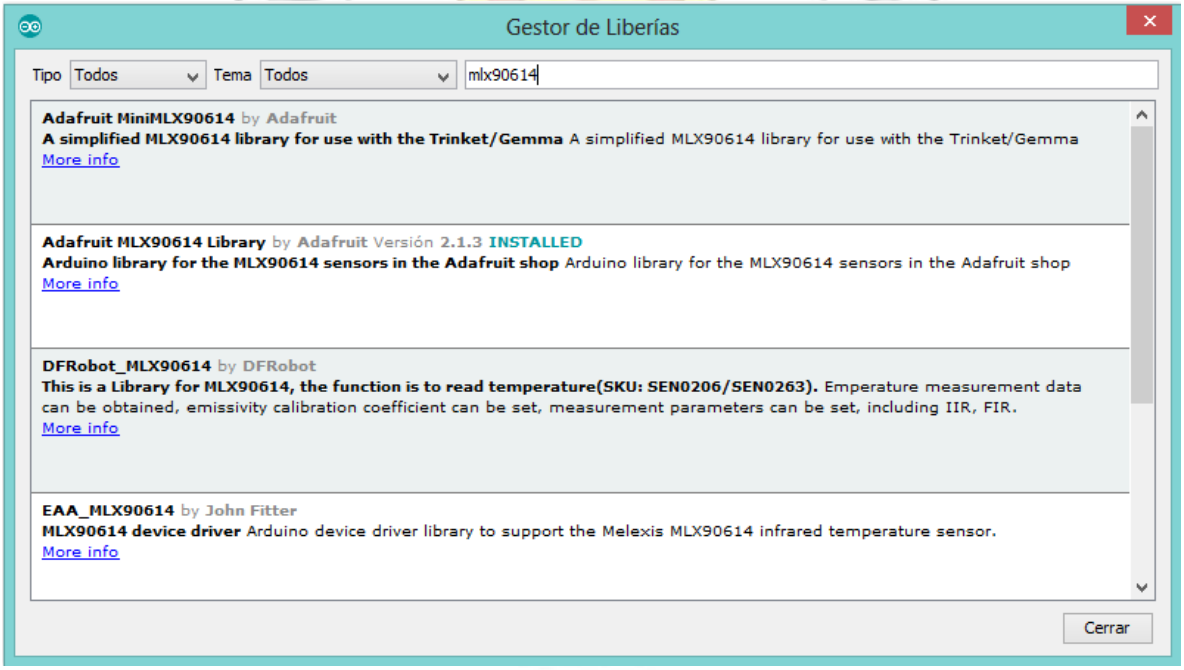
5.1 COMUNICACIÓN ENTRE ARDUINO Y SIMULINK

5.1.1 Requerimientos

Para establecer la comunicación y sincronización entre el arduino y simulink se requiere el paquete de soporte de Matlab para arduino y verificar que esten instaladas las librerías simulink Real-Time e Instrument Control Toolbox.

Teniendo en cuenta que el sensor MLX90614 tiene su propio protocolo de comunicación I2C, se requiere instalar la librería Adafruit MLX90614 directamente del gestor de librerías de arduino que se muestra en la figura 23.

Figura 23 Gestor de librerías de arduino



Fuente: Autores.

5.1.2 Enviar datos desde la tarjeta arduino a Simulink

Mediante una matriz de tipo byte se envía el valor de la temperatura que proporciona la librería del MLX90614 al puerto serial de la manera que se realiza en la figura 24.

Figura 24 Matriz de datos de temperatura

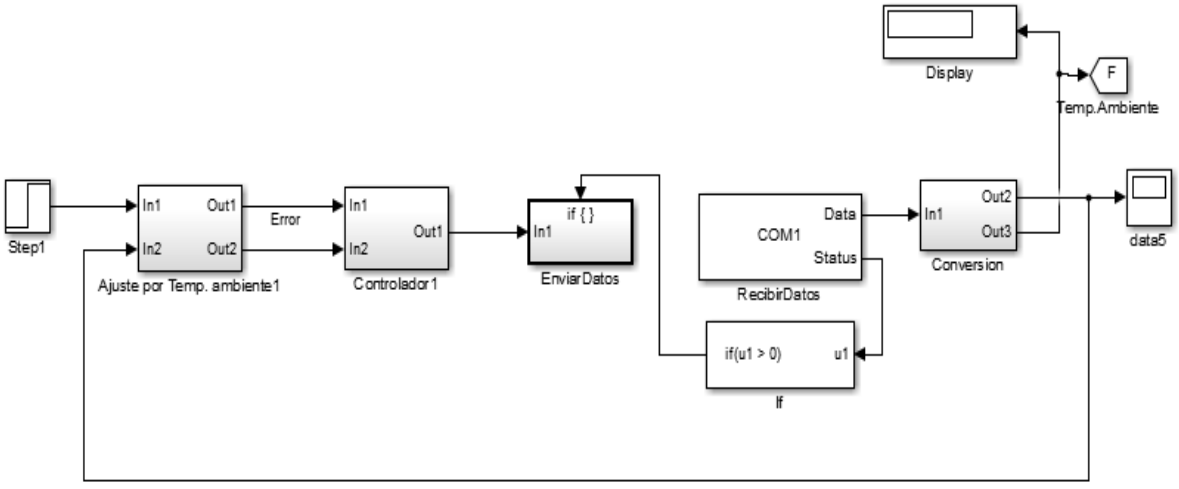
```
buffer_enviar[0]='E';
buffer_enviar[1]= entero;
buffer_enviar[2]= decimal;
buffer_enviar[3]= entAmb;
buffer_enviar[4]= decAmb;
buffer_enviar[5]='\n';
Serial.write(buffer_enviar,6);
```

Fuente: Autores.

5.1.3 Recibir y enviar datos en Simulink

De la librería Instrument Control Toolbox se obtiene el bloque Serial Receive para recibir los datos y posteriormente convertir la matriz que se envía desde el arduino en los valores de temperatura en °C. De la misma librería se obtiene el bloque Serial Send para enviar los datos al arduino. Se debe tener en cuenta que mientras se recibe datos no se puede enviar al mismo tiempo debido a que la comunicación entre el arduino y Simulink es half dúplex, por tal motivo para evitar errores de comunicación se agrega un bloque condicional if de la figura 25 para establecer que mientras se reciben datos no se envían datos.

Figura 25 Diagrama de bloques simulink



Fuente: Autores.

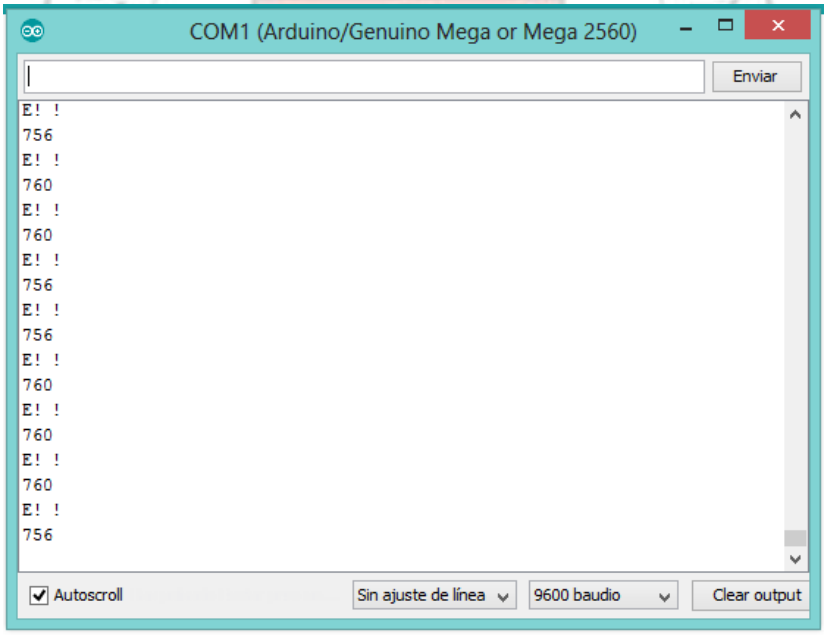
Se requiere de un bloque de conversión de datos seleccionando uint8 antes del bloque Serial Send, debido a que Simulink utiliza el tipo de números double y se debe cambiar por el tipo de datos para arduino.

5.1.4 Tiempo de ejecución del código de arduino.

Cada instrucción dentro de un código tiene un tiempo de ejecución que depende del dispositivo donde se ejecuta, dependiendo de esto y de la complejidad del código es posible que un sistema de control sea afectado si no se tiene en cuenta. Para este caso en especial el tiempo de ejecución del código es en promedio de 760μs, por lo tanto no es significativo y se puede despreciar.

Para obtener el tiempo de ejecución del código se requiere de la función micros() como se muestra en lo resaltado con color verde en el anexo 2 y su resultado se evidencia en la figura 26.

Figura 26 Tiempo de ejecución del código de arduino



Fuente: Autores.

5.2 ENTORNO DE DESARROLLO GUIDE

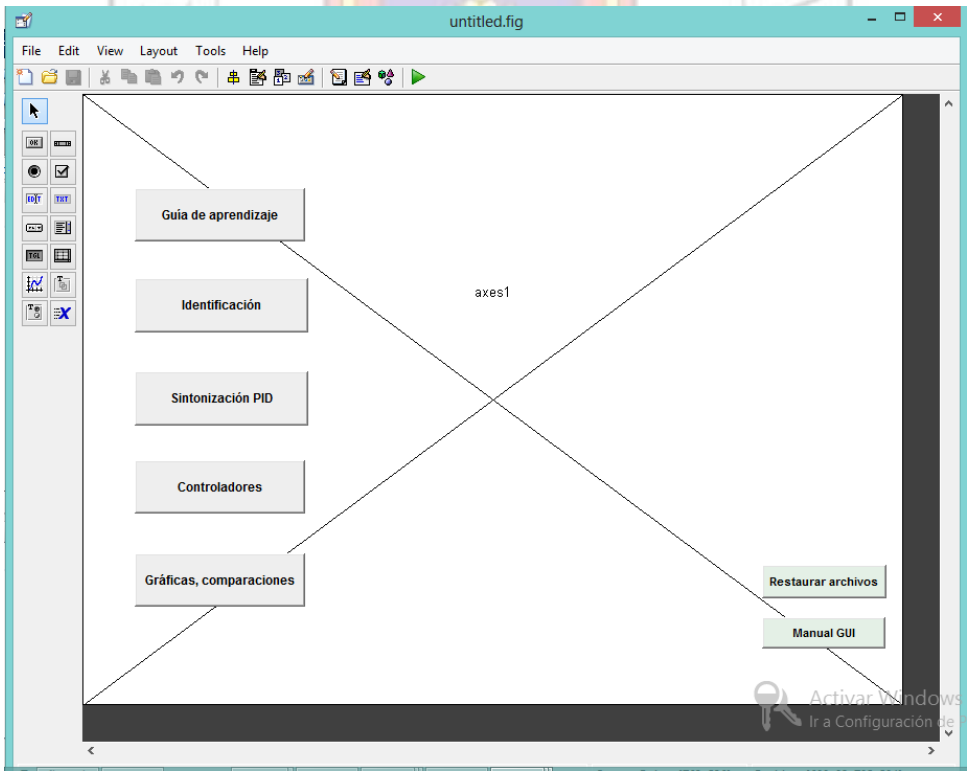
5.2.1 Diseño de figura principal

Teniendo en cuenta las características de una interfaz que figuran en el inciso 3.1.1 y el orden establecido en el diagrama de flujo de funcionalidad de la interfaz del anexo 1 en el cual se muestra cada una de las Figuras o ventanas principales con su respectiva función u opciones que contiene, se procede a diseñar las ventanas GUIDE.

Para mantener un orden y una óptima navegabilidad del usuario a través de la interfaz se establecieron las opciones de aprendizaje, identificación, sintonización, aplicación de los controladores y graficas comparativas en un orden lógico que se ve plasmado en la guía de aprendizaje y manual de uso.

Para cada una de las Figuras creadas, como en el caso de la ventana principal de la figura 27, se utilizó un “axes” para imprimir una imagen de fondo y luego se agregaron los demás componentes.

Figura 27 Figura de ventana principal en entorno de desarrollo

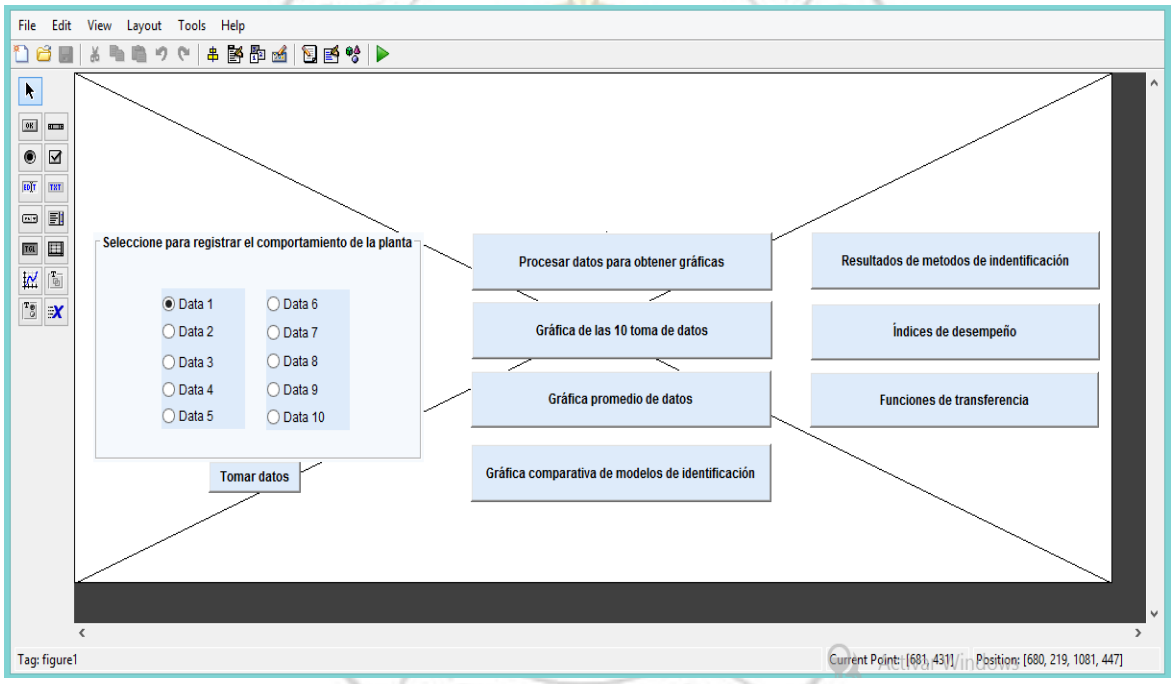


Fuente: Autores

Cada uno de los componentes (botones) de la ventana principal abre una nueva Figura (ventana) a excepción de los botones “Curso pdf” y “Manual GUI” que abren un documento cada uno de estos.

En el caso de las Figuras de “Identificación” y “sintonización” se encuentran componentes (botones) para realizar cálculos y guardar datos en archivos Excel que son los utilizados para almacenar la información que carga la interfaz al iniciarse, y de esta manera evitar inconvenientes como por ejemplo el de perder las gráficas de toma de datos anterior; un ejemplo de estos componentes es el “Tomar datos” que se observa en la figura 28.

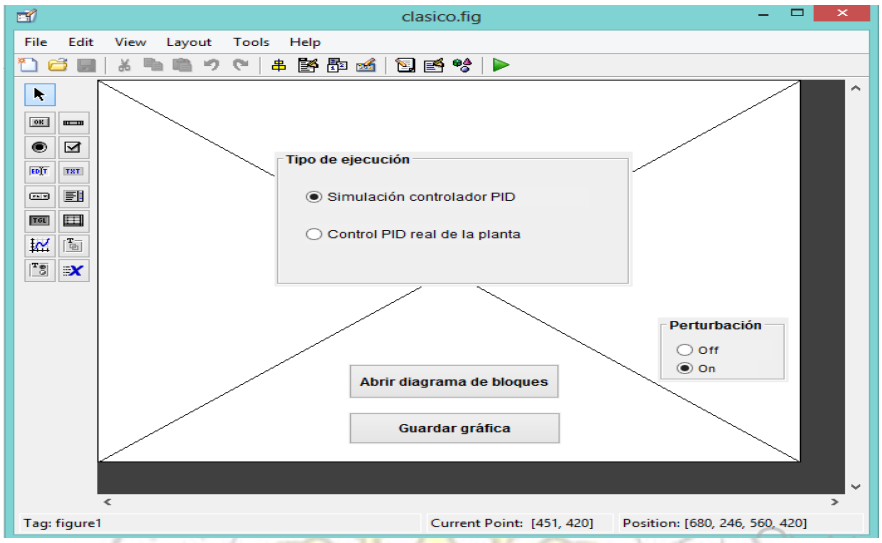
Figura 28 Figura de Identificación



Fuente: Autores

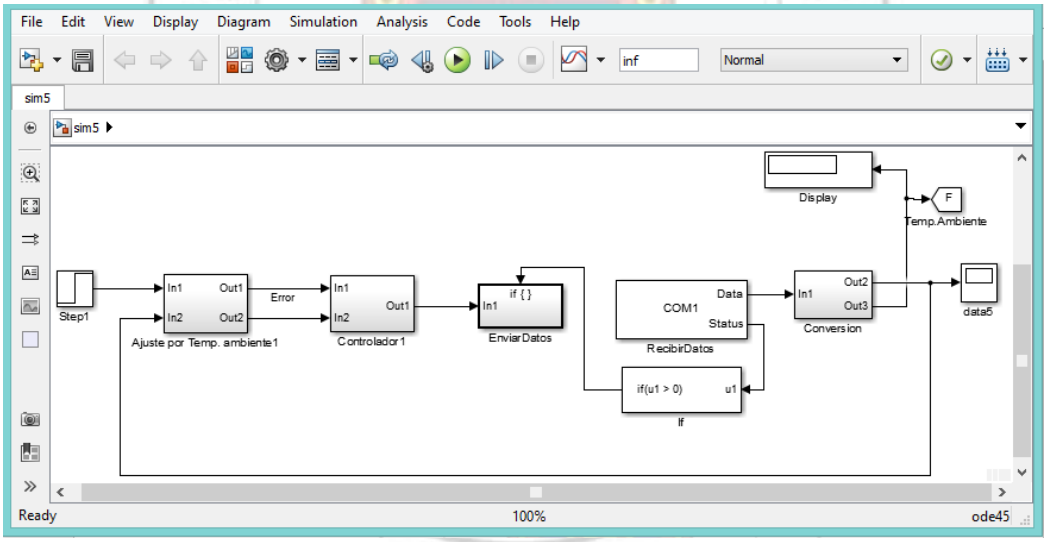
Para las opciones de controladores como “Abrir diagrama de bloques” de la figura 29, esta abre un diagrama de bloques de simulink como el de la figura 30, dependiendo de la opción que se seleccione, puede ser solo una simulación o puede conectarse con el arduino para aplicar el control en la planta real.

Figura 29 Figura de control clásico



Fuente: Autores

Figura 30 Bloque Simulink



Fuente: Autores

Otros componentes dentro de las Figuras (ventanas de interfaz) tienen la función de abrir gráficas de Simulink o mostrar solamente información en una ventana.

5.3 ORGANIZACIÓN DE ARCHIVOS Y CÓDIGOS DE FIGURAS

5.3.1 Inicialización de GUI

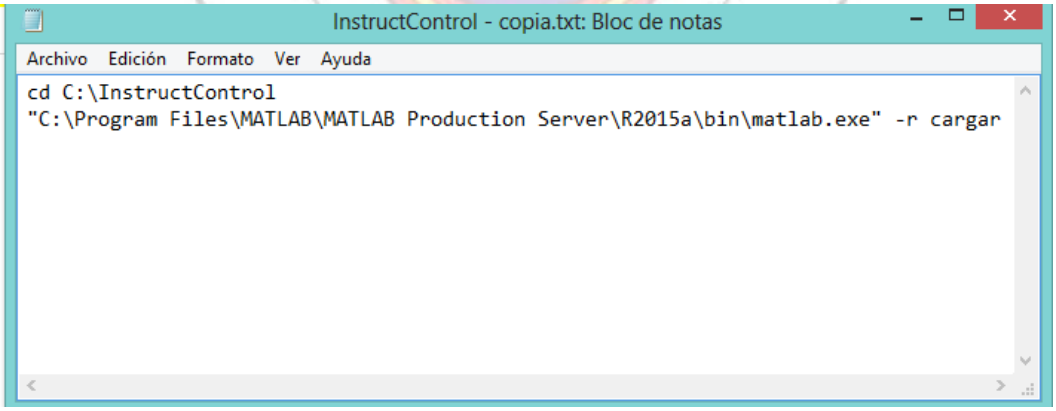
Para la interfaz se creó un ejecutable que da inicio a Matlab y a INSTRUCTCONTROL (nombre de la GUI) con el fin de iniciar los dos a la vez debido a que se depende de Matlab para conectar fácilmente con el arduino, el icono del ejecutable se ilustra en la figura 31.

Figura 31 Icono ejecutable .exe

Fuente: Autores

En la creación del ejecutable se utilizó un archivo .bat con los comandos que se observan en la figura 32 y se convirtió en un ejecutable .exe mediante el programa "Advanced Bat To Exe Converter v4.23".

Figura 32 Comandos para archivo .bat

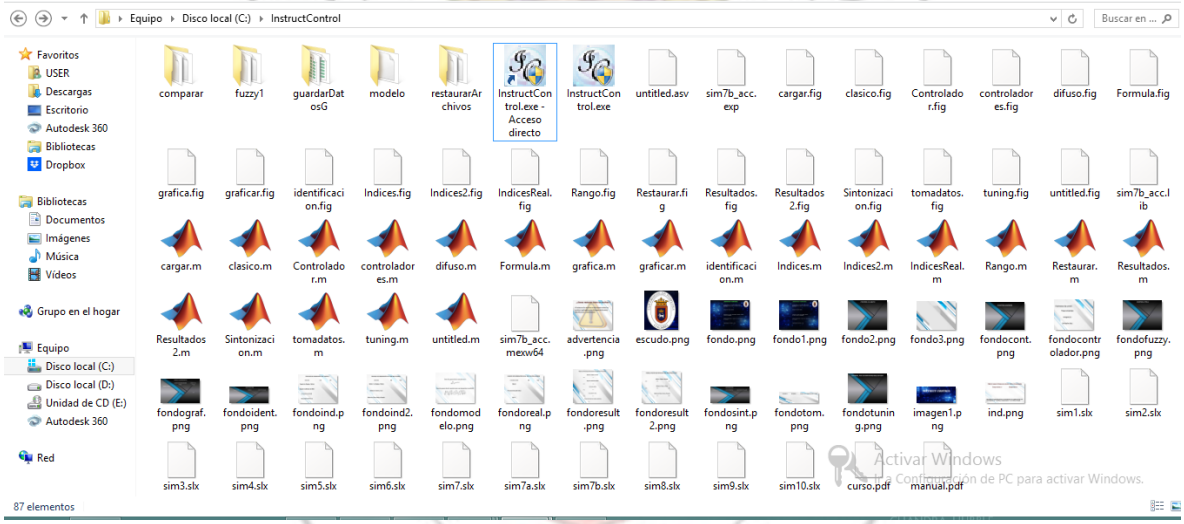


Fuente: Autores

5.3.2 Organización de archivos

Para el funcionamiento de la interfaz se requiere que la carpeta de archivos de nombre “InstructControl” se encuentre ubicada en el disco local C de la computadora, dentro de ella se encuentran los archivos de las Figuras de interfaz y sus respectivos códigos, además de imágenes y archivos de simulink como se observa en la figura 33. Dentro de dicha carpeta se encuentran 5 carpetas más, la primera contiene archivos simulink de graficas comparativas, la segunda carpeta contiene los archivos de lógica difusa, la tercera carpeta contiene archivos Excel para almacenar información, la cuarta carpeta corresponde a archivos simulink relacionados con modelos de identificación y sintonización, y la última carpeta corresponde archivos de simulink y Excel para ser copiados en caso de restaurar archivos desde la interfaz.

Figura 33 Archivos de interfaz



Fuente: Autores.

5.3.3 Códigos de Figuras de interfaz

Cada una de las Figuras o ventanas de la interfaz tienen un código por separado disponible en el anexo 3. en donde cada vez que se abre una de ellas se ejecuta una función dentro de su código que permite definir unas características iniciales como por ejemplo la imagen de fondo, para luego quedar en espera de las órdenes del usuario.

5.4 GRÁFICAS, OPCIONES DE SIMULACIÓN Y DE APLICACIÓN REAL EN LA PLANTA

5.4.1 Selección entre simulación y control real

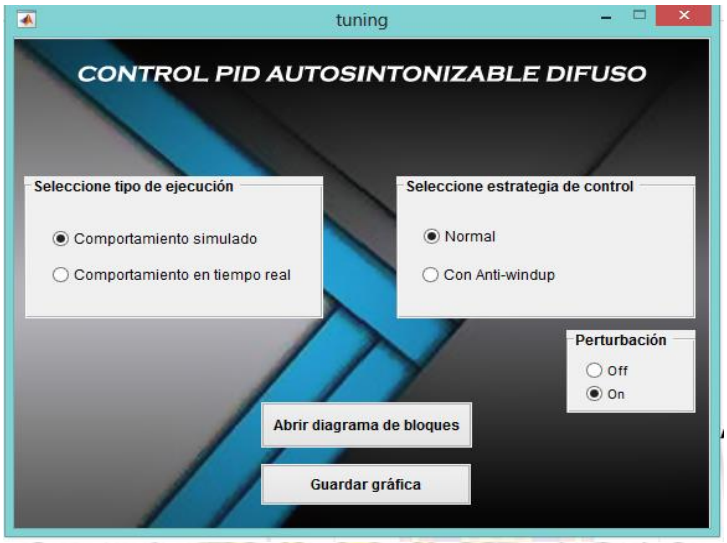
Dentro de la ventana “Controladores” que se observa en la figura 34 se encuentran tres botones para cada tipo de control.

Figura 34 Ventana de controladores

Fuente: Autores.

Con el fin de obtener la característica de usabilidad de la interfaz, para cada una de las figuras o ventanas de los tres tipos de control, se diseñó con un menú de opciones en el cual se puede seleccionar el tipo de ejecución si es simulación o control real en la planta, y si se desean perturbaciones. Finalizada la ejecución se puede guardar la gráfica, todas estas opciones se aprecian en la figura 35.

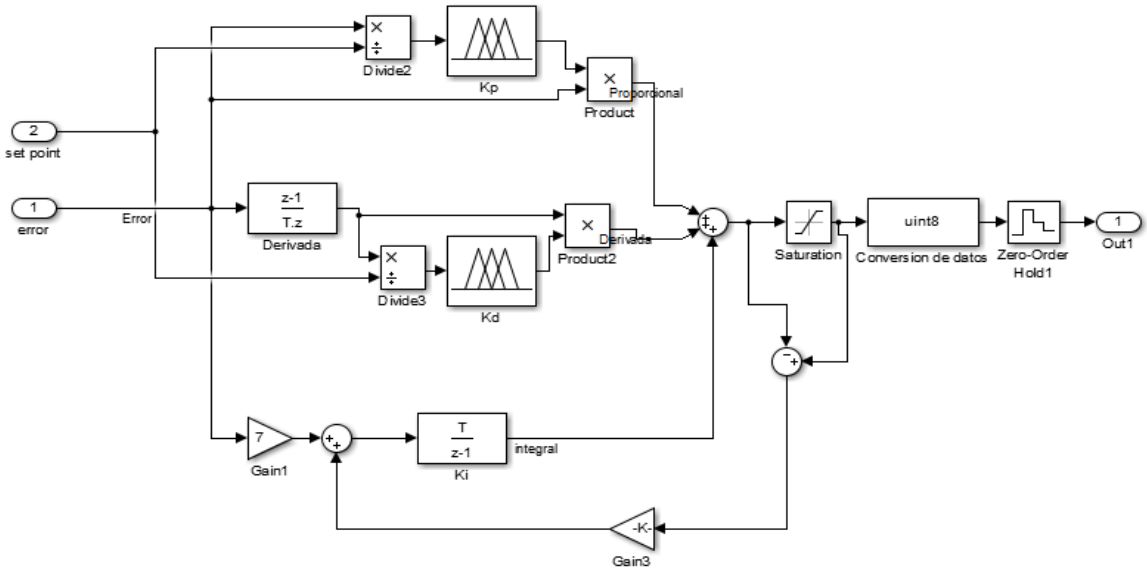
Figura 35 Opciones de control auto sintonizable



Fuente: Autores.

Al dar clic en “Abrir diagrama de bloques” de la figura 35, se abre el archivo simulink con el respectivo lazo de control de acuerdo a las opciones seleccionadas, siendo el ejemplo de seleccionar el comportamiento en tiempo real para el controlador PID auto sintonizable difuso, obtendríamos el diagrama de bloques de la figura 30, el cual contiene el controlador de la figura 36.

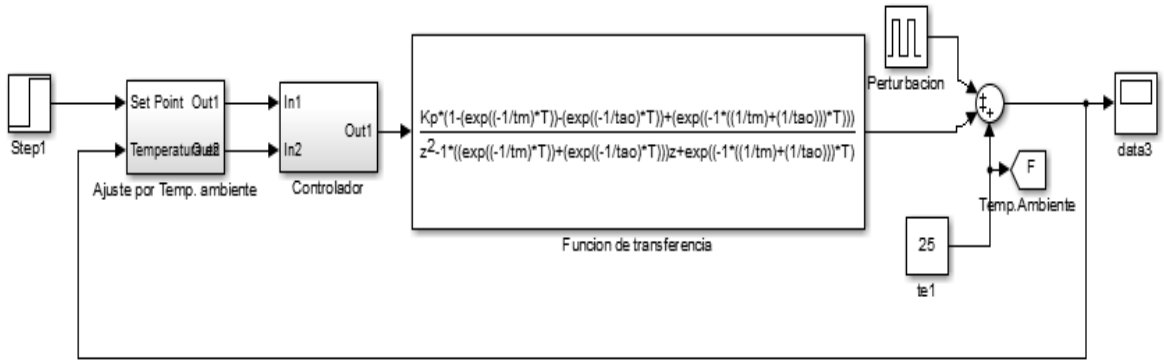
Figura 36 Controlador con anti-windup



Fuente: Autores.

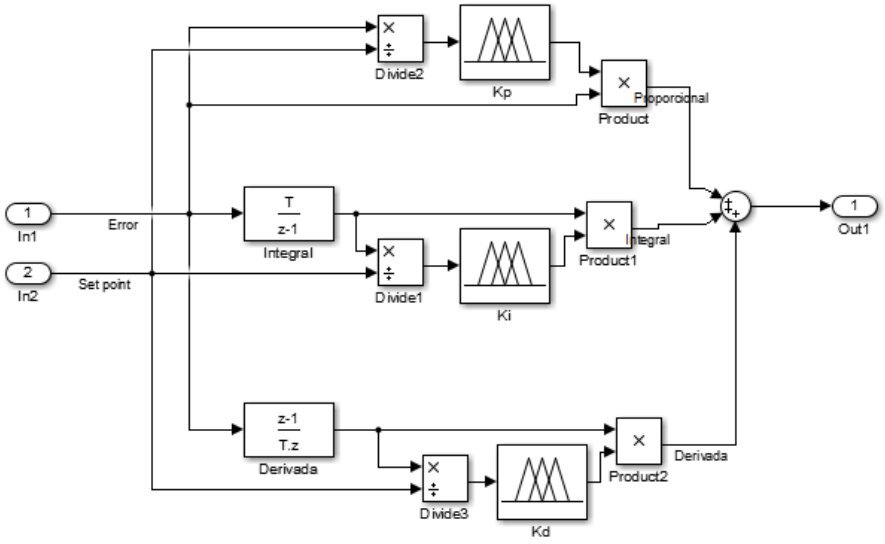
En el caso de seleccionar el comportamiento simulado se obtendría el diagrama de bloques de la figura 37 con su respectivo controlador de la figura 38.

Figura 37 Diagrama de bloques de simulación



Fuente: Autores.

Figura 38 Controlador de simulación



Fuente: Autores.

5.4.2 Gráficas

La ventana de “Gráficas, comparaciones” de la figura 39 se diseñó para poder mostrar en una sola grafica las selecciones de todos los controladores guardados después de su ejecución, sean simulados o real en la planta y de esta manera facilitar la comparación y análisis de resultados.

Figura 39 Menu de gráficas a comparar

Fuente: Autores.

En las ventanas de “Identificación” y “Sintonización” se encuentra la opción de graficas comparativas junto con herramientas de análisis y de información como lo son “Índices de desempeño” y “Resultados de métodos de identificación” que se pueden apreciar en la figura 40.

Figura 40 Ventana de identificación

Fuente: Autores.

5.5 OPCIONES DE AYUDA AL USUARIO

En la ventana principal se encuentra la opción para abrir el manual de uso de la interfaz, también se cuenta con la opción de “Restaurar archivos”, al ingresar se abre la ventana de la figura 41 diseñada con la característica de alerta, la cual solicita confirmar la acción de restaurar archivos. Al hacer clic en aceptar se restablecerán los archivos de simulink, por lo tanto es de utilidad cuando se ha trabajado y guardado cambios en los diagramas de bloques y se desea restaurar los diagramas; a su vez también se restablece la información guardada en archivos Excel en la cual se guarda la toma de datos, graficas de simulaciones y de control real de la planta, cálculos de identificación y sintonización.

Figura 41 Confirmar restauración

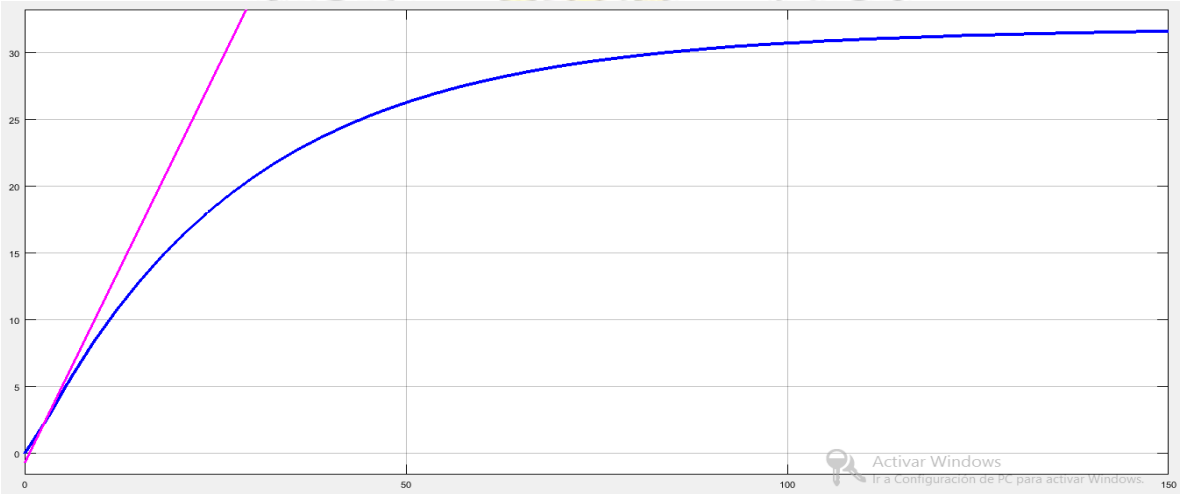
Fuente: Autores.

6 IDENTIFICACIÓN Y SINTONIZACIÓN DE LA PLANTA

6.1 IDENTIFICACIÓN

Para obtener el modelo de la planta se realizan 10 toma de datos del comportamiento de la planta a lazo abierto con 2 set points de PWM de 76 y 178 PWM (el 100% del PWM representa 255 PWM en el ardino, dando como resultado 12 v en el elemento final de control) y con las 10 toma de datos se obtiene una gráfica promedio y se calcula la recta tangente al punto de inflexión como se observa en la figura 42.

Figura 42 Respuesta promedio a lazo abierto



Fuente: Autores.

Los métodos de identificación utilizados con sus respectivos resultados de las ecuaciones (6.1) a (6.4) son:

- Tangente de Ziegler-Nichols.

$$\frac{0.313}{27.31\,s+1}\cdot e^{-0.6108\,s} \tag{6.1}$$

- Método de la tangente modificado de Miller.

$$\frac{0.313}{28.09\ s+1}.\ e^{-0.6108\ s} \tag{6.2}$$

- Método dos puntos de Smith.

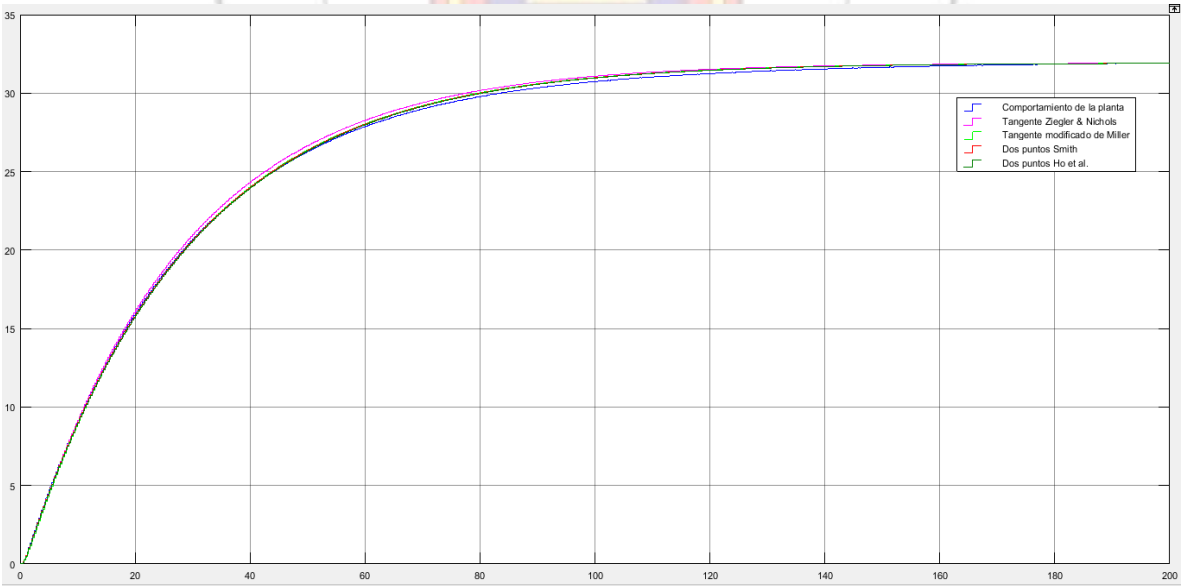
$$\frac{0.313}{28.25\ s+1}.\ e^{-0.4478\ s} \tag{6.3}$$

- Método dos puntos de Ho et al.

$$\frac{0.313}{28.34\ s+1}.\ e^{-0.4887\ s} \tag{6.4}$$

Al comparar todos los modelos obtenidos con la respuesta real de la planta a lazo abierto en la figura 43, todos presentan buenos resultados, y el mejor modelo seleccionado teniendo en cuenta índices de desempeño es el de Ho et al.

Figura 43 Resultados de identificación



Fuente: Autores.

Siendo el modelo de la planta de primer orden más tiempo muerto se obtiene una función de tranferencia general en tiempo discreto en la ecuación (6.5)

$$\frac{K(1-e^{-\frac{1}{tm}*T}-e^{-\frac{1}{\tau}*T}+e^{-\left(\frac{1}{tm}+\frac{1}{\tau}\right)*T})}{z^2-z(e^{-\frac{1}{tm}*T}+e^{-\frac{1}{\tau}*T})+e^{-\left(\frac{1}{tm}+\frac{1}{\tau}\right)*T})} \quad (6.5)$$

6.2 SINTONIZACIÓN

Los métodos de sintonización utilizados y sus respectivos resultados de observan en la tabla 5

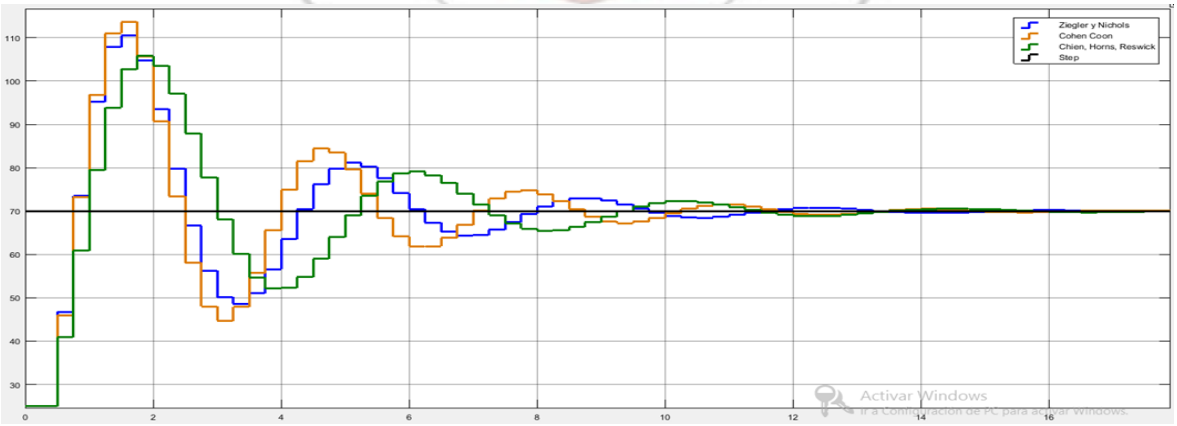
Tabla 5 Resultados de Sintonización

Método	Kp	Ki	Kd
Primer Método de Ziegler y Nichols	222.33	227.45	54.33
Lazo abierto de Cohen y Coon	247.83	207.52	43.9
Método de de Chien, Horns y Reswick.	176	151.63	36.21

Fuente: Autores.

La figura 44 muestra la respuesta de cada uno de los métodos de sintonización, con la ayuda de indicadores de desempeño se selecciona el método de Ziegler y Nichols siendo el mas apropiado para ser utilizado como referencia en el diseño de los controladores.

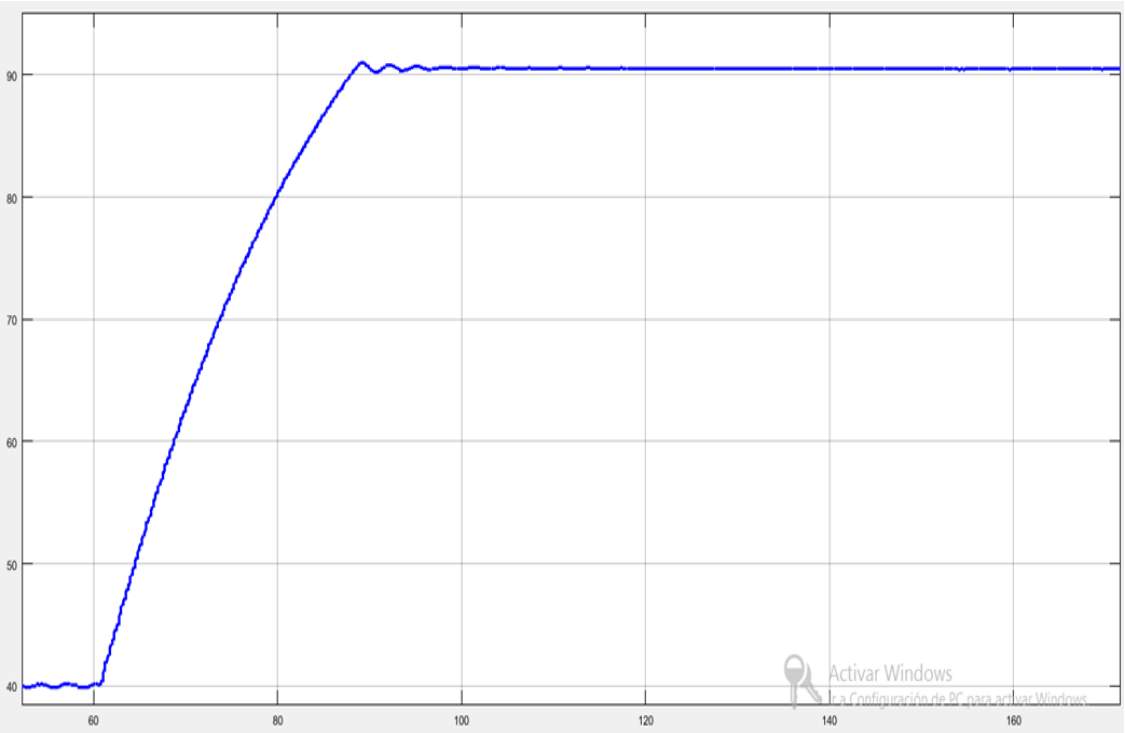
Figura 44 Respuesta de métodos de identificación



Fuente: Autores.

Teniendo en cuenta que el estudio de los controladores se enfoca principalmente en la aplicación real en la planta, al controlador clásico PID seleccionado se realizaron ajustes experimentales obteniendo el resultado de la figura 45 con Kp=200, Ki=0.4, Kd=3.

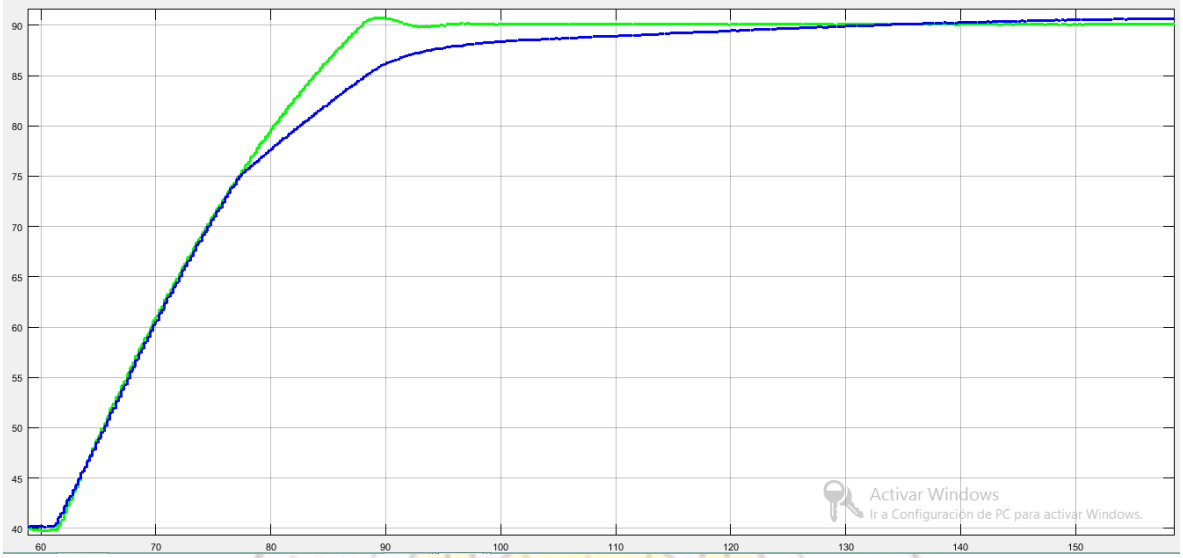
Figura 45 Comportamiento PID real con ajuste



Fuente: Autores.

Utilizando como referencia los valores del controlador PID clásico con ajuste, se realizo el controlador PID autosintonizable de la figura 9 y el controlador FPD+I de la figura 10 con ajustes experimentales obteniendo el comportamiento de la figura 46 en color verde y azul respectivamente.

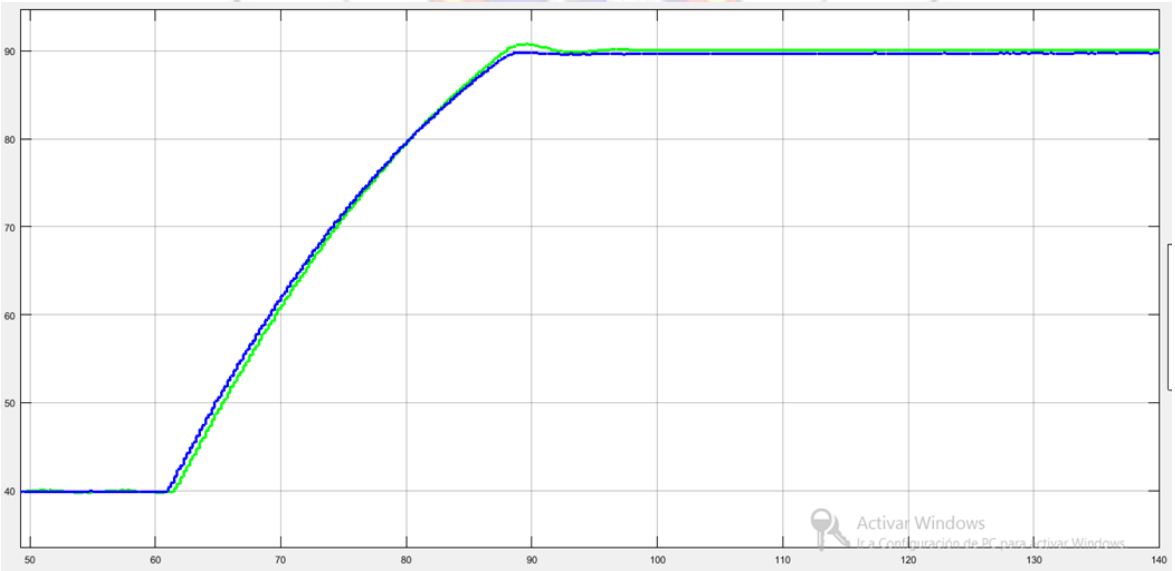
Figura 46 Respuesta de controladores principales



Fuente: Autores.

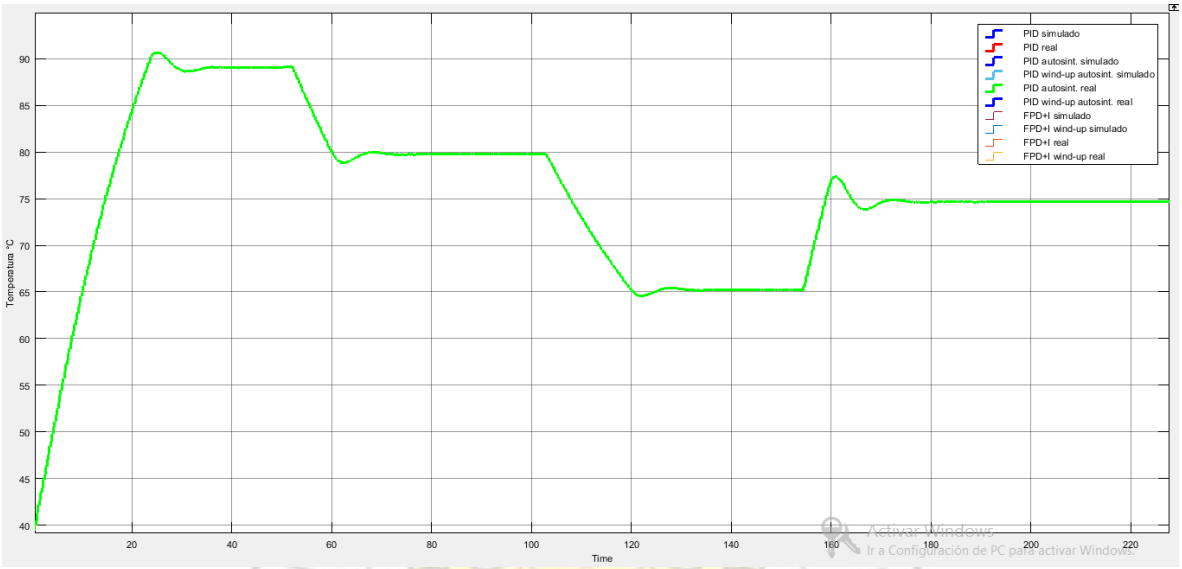
Para los dos controladores principales se realizo con anti-windup, y se resalta el resultado del PID autosintonizable con y sin anti-windup en color verde y azul respectivamente en la figura 47.

Figura 47 PID autosintonizable con y sin anti-windup



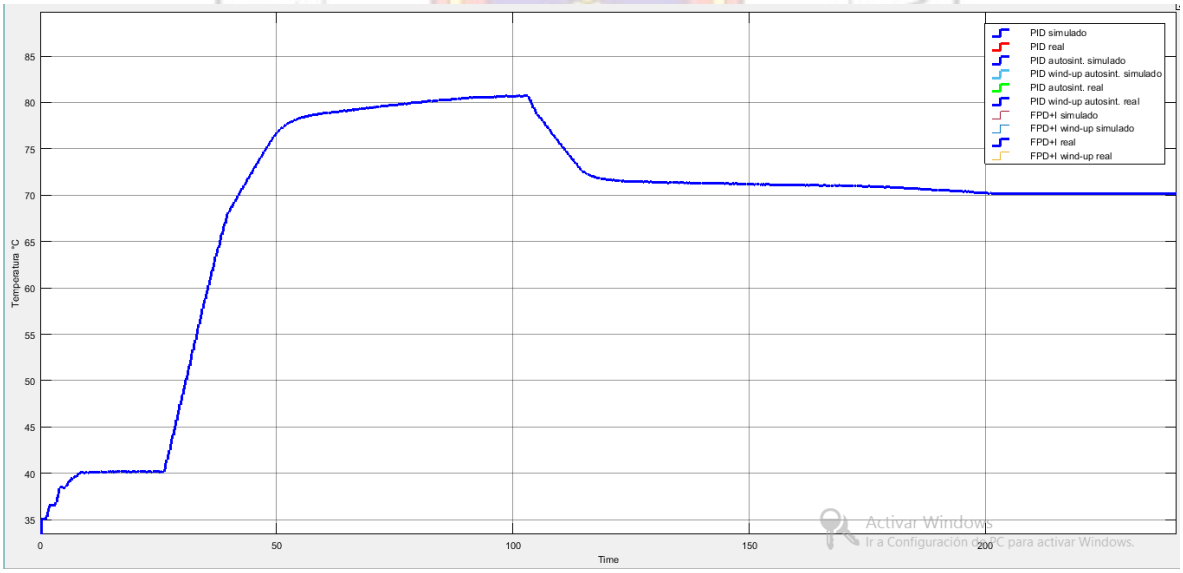
Fuente: Autores.

Figura 48 PID auto sintonizable a diferentes set points



Fuente: Autores.

Figura 49 FPD+I a diferentes set points



Fuente: Autores.

En la figura 48 y 49 se observa el comportamiento del controlador PID auto sintonizable por lógica difusa y FPD+I para diferentes set points respectivamente, demostrando el correcto funcionamiento de utilizar la estrategia que se explica en

el anexo 5 para reemplazar un polinomio utilizado en controladores difusos para hacer que el controlador logre llevar la respuesta al set point deseado.

Tabla 6 Resultados reales de controladores

Controlador	Máximo pico porcentual (Mp%)	Tiempo de estabilización (Ts)	Tiempo de crecimiento (Tr)	ISE	ITAE
PID clásico	2	44	28	2594.4	2003.2
PID auto sintonizable	1.52	39	28	2971	1121.8
PID auto sintonizable con anti-windup	0	35	29	2844.2	1610.9
FPD+I	1.3	100	72.5	3606	3041
FPD+I con anti-windup	1.7	82	64	3505.9	3441.4

Fuente: Autores.

De acuerdo a los resultados de la tabla 6, el mejor controlador es el PID autosintonizable por lógica difusa con y sin anti-windup, y en el caso del controlador FPD+I no se obtuvieron los mismos resultados al implementar el anti-windup, incluso el sobreimpulso aumento, al igual que un error en estado estable del 1.6%, teniendo en cuenta que existe una complejidad para la compensación exacta al utilizar anti-windup.

La identificación de la planta y sintonización con todos los pasos y métodos utilizados, asi como el empleo de indicadores de desempeño, y detalles sobre ajustes experimentales se pueden ver en la guía de aprendizaje del anexo 5.

7 MANUAL DE USO DE LA INTERFAZ GRÁFICA Y GUÍA DE APRENDIZAJE

Para el orden en el diseño de la interfaz, se estableció un diagrama de flujo de funcionalidad en el anexo 1 en el cual se plantea un orden lógico que el usuario debería de seguir, de igual manera para el desarrollo de la guía de aprendizaje se estableció el mismo orden, al igual que el manual de usuario.

En la guía de aprendizaje del anexo 5 se decidió combinar el paso a paso con la teoría de la identificación, sintonización, control PID auto sintonizable por lógica difusa y el control FPD+I, explicando el concepto o la estructura de cada controlador en su respectivo momento antes de cada paso con el fin de mantener mayor claridad en cuanto a que se está haciendo en cada procedimiento.

Dentro del contenido del manual de uso del anexo 4 se incluyen los requerimientos de la computadora y el código para grabar en el arduino para que la interfaz se pueda ejecutar correctamente, además se incluye un inconveniente que surgió en algún momento durante la ejecución de la interfaz con simulink en tiempo real, que es un error de sincronización con el arduino, y la manera de darle solución.

8 CONCLUSIONES

Desarrollando el diseño se dio cumplimiento a requerimientos importantes como lo son la rapidez en realizar las pruebas de cada controlador y la sencillez del prototipo que se adquirió gracias al empleo de matrices de selección que facilito escoger la mejor opción en base a criterios establecidos. Con la selección de los componentes más importantes se diseñó el circuito de potencia ajustándose a los requerimientos de la señal de control y del elemento final de control.

Se establecen los métodos de identificación y sintonización que se observa en el anexo 5 para poder deducir el modelo y controlador más adecuado, corroborando estas deducciones y selecciones mediante métricas cuantitativas para evaluar el desempeño de cada uno de los métodos planteados, teniendo en cuenta que la sintonización va de la mano con adaptaciones obtenidas durante pruebas experimentales que nos ofrecen la información para realizar ciertos ajustes en determinadas variables o constantes de cada uno de los controladores.

Cada una de las dos estrategias de control, sea el control PID auto sintonizable por lógica difusa o el control FPD+I, presentan ciertas ventajas y desventajas entre si y frente al control clásico, resaltando entre estos el controlador PID auto sintonizable que permite manipular de una mejor manera las características de la respuesta del controlador con respecto a las variables Kp, Ki y Kd que en el caso del control clásico son constantes. El controlador FPD+I no mostro muy buenos resultados, siendo el de mayor tiempo de establecimiento; y el controladar PID auto sintonizable comparado con el clásico presenta mejor tiempo de establecimiento y menor sobreimpulso, pero frente a perturbaciones el controlador clásico se estabiliza con mayor rapidez.

Llevado a cabo el diseño de la interfaz, se logra dar cumplimiento a las características más importantes de una GUI, como lo son la facilidad de uso, navegabilidad, orden y en especial la usabilidad al ofrecer herramientas para obtener cada uno de los modelos planteados para la identificación de la planta y la sintonización PID clásico de referencia y de tal manera enfocar más la atención en la sintonización de los dos controladores objeto de estudio, de igual manera la usabilidad se ve reforzada al combinar su funcionamiento con Simulink de Matlab que es un entorno de programación visual de alto nivel.

En la guía de aprendizaje se logró plasmar con facilidad cada uno de los pasos hasta llegar a la sintonización de cada controlador con sus respectivos ajustes gracias a las propias herramientas de la GUI. Con la realización del manual de uso se obtiene la información necesaria para el manejo de la GUI manteniendo un orden específico de acuerdo al desarrollo de la guía de aprendizaje evitando confusiones en el uso de cada una de las opciones de la interfaz.

9 RECOMENDACIONES

1. Dado el caso que se requiera de cambiar la resistencia calefactora, se bebe tener en cuenta el mismo tamaño, voltaje y potencia para evitar daños en la estructura del prototipo debido a que su diseño es para bajas temperaturas. Para utilizar resistencias calefactoras de mayor potencia se requiere cambiar el material de la estructura o carcasa.

2. Hacer las prácticas en un computador con procesador superior o igual al Core i5, o cualquiera equivalente al Core i5 y superiores para evitar problemas de sincronización en tiempo real con el arduino.

3. La estructura del prototipo cuenta con la facilidad de armar y desarmar, por tal motivo en caso de requerir cambiar algún componente se realiza con facilidad.

10 BIBLIOGRAFÍA

Prototipo.

[1] DRISHTI YADAV Y RN PATEL, (2018). Interactive educational tool for the design of compensators using frequency response analysis. {En línea}.{10 septiembre de 2022}.disponible.en:(https://journals.sagepub.com/doi/10.1177/0020720917750958)

[2] R.J MITCHELL, (2013) Improved MATLAB GUIs for Learning Frequency Response Methods. {En línea}.{10 septiembre de 2022}. Disponible en: (https://www.sciencedirect.com/science/article/pii/S1474667015341185).

[3] MEHMET CINAR Y ASIM KAYGUSUZ, (2021) The practical power flow analysis program based on matlab GUI developed for educational purposes for students in smartgrids:Oyamatlab.{En.línea}{10 septiembre de 2022}. Disponible.en:(https://journals.sagepub.com/doi/abs/10.1177/00207209211013471 ?journalCode=ijea)

[4] CHANDANI SHARMA Y ANAMIKA JAIN, (2014) Tuning of gains in basis weight using fuzzy controllers. {En línea}{10 septiembre de 2022}..Disponible en:(https://www.paperpublications.org/upload/book/TUNING%20OF%20GAINS%20IN%20BASIS%20WEIGHT-59.pdf)

[5] L. ISAZA, J. VARGAS, F. VELÁSQUEZ,(2015), Diseño de una interfaz gráfica de usuario para el control de un prototipo de banda seleccionadora de piezas industriales. {En línea}{11 septiembre de 2022}..Disponible en:(https://www.paperpublications.org/upload/book/TUNING%20OF%20GAINS%20IN%20BASIS%20WEIGHT-59.pdf)

[6] DANIEL SANCHEZ, (2014). Diseño de interfaces gráficas de usuario.{En línea}{11septiembre.de.2022}.Disponible.en:(https://www.paperpublications.org/upload/book/TUNING%20OF%20GAINS%20IN%20BASIS%20WEIGHT-59.pdf).

[7] GÓMEZ, LEOPOLDO SEBASTIÁN M. "Diseño de Interfaces de Usuario Principios, Prototipos y Heurísticas para Evaluación." Disponible.en:(https://www.angelfire.com/journal2/lead/DIU.pdf).

[8] CLAUDIA ALBORNOZ. Diseño de Interfaz Gráfica de Usuario Disponible.en:(http://sedici.unlp.edu.ar/bitstream/handle/10915/41578/Documento_completo.pdf?sequence=1).

[9] JOSÉ ANTONIO OTERO HERNÁNDEZ. Interfaces Gráficas de Usuario en MATLAB.{En.línea}{15septiembre.de.2022}.Disponible.en:(https://metodosnumericoscem.weebly.com/uploads/2/5/9/7/25971049/guia_simple_guide_matlab.pdf).

[10] DIEGO ORLANDO BARRAGÁN GUERRERO.(2008). .{En.línea}{19septiembre.de.2022}.Disponible.en:(https://www.utm.mx/~vero0304/HCPM/MATLAB_GUIDE.pdf).

[11] JORGE NOVOA, WUEMDELL MERTINEZ, (2019). Identificación y sintonización de controladores PID para procesos de integración. .{En línea}{12 septiembre.de.2022}.Disponible.en:(https://repositorio.cuc.edu.co/bitstream/handle/11323/5557/Identificaci%C3%B3n%20y%20sintonizaci%C3%B3n%20de%20controladores%20PID%20para%20procesos%20de%20integraci%C3%B3n.pdf?sequence=1&isAllowed=y).

[12] SERGIO CASTAÑO, (2020).Todo sobre Ziegler y Nichol sintonía de controladorPID.{En.línea}{12septiembre.de.2022}.Disponible.en:(https://controlautomaticoeducacion.com/control-realimentado/ziegler-nichols-sintonia-de-control-pid/).

[13] SERGIO CASTAÑO, (2019). Sintonia PID por el método CHR. {En.línea}{13septiembre.de.2022}.Disponible.en:(https://controlautomaticoeducacion.com/control-realimentado/sintonia-pid-por-el-metodo-chr/).

[14] UNIVERSIDAD NACIONAL DE TUCUMAN, (2017). .{En.línea}{13septiembre.de.2022}.Disponible.en:(https://catedras.facet.unt.edu.ar/controldeprocesos/wp-content/uploads/sites/85/2016/02/Tabla-Metodos-de-Sintonizacion-2017.pdf).

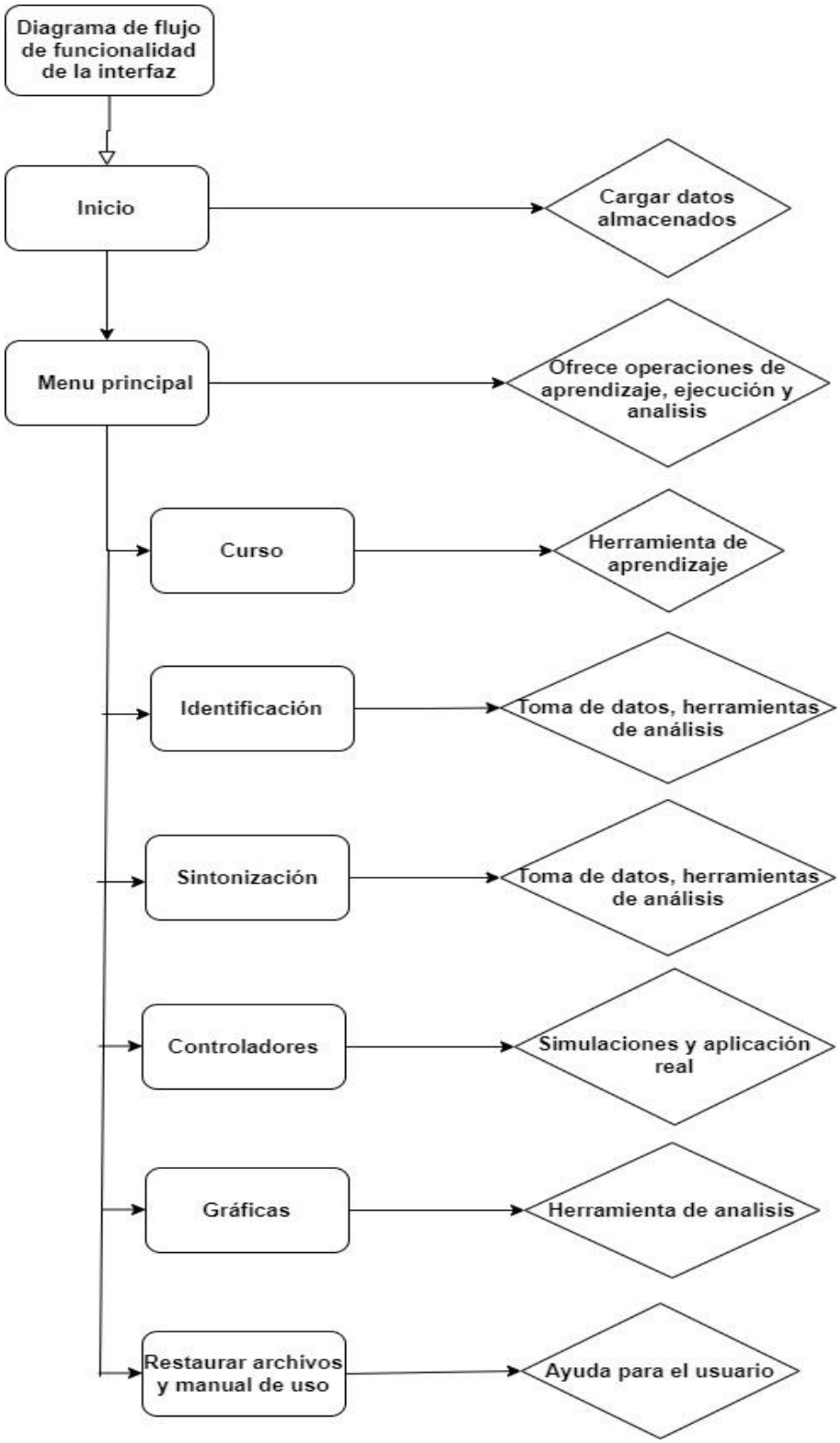
[15] PEDRO PONCE. (2010). Inteligencia artificial con aplicaciones a la ingeniería: Mexico. P 22.

[16] ABDULLAH ALMESHAL. (2013).MODELLING AND CONTROL OF TWO.WHEELED VEHICLE WITH EXTENDABLE INTERMEDIATE BODY ON AN INCLINED.SURFACE.{En.línea}{13septiembre.de.2022}.Disponible.en:(https://www.researchgate.net/publication/264084046_Modelling_And_Control_Of_A_Two-Wheeled_Vehicle_With_Extendable_Intermediate_Body_On_An_Inclined_Surface).

[17] K. J. ASTROM, AND T. HÄGGLUND,(2006). ADVANCED PID CONTROL. P 23

11 ANEXOS

ANEXO 1 DIAGRAMA DE FLUJO DE FUNCIONALIDAD DE INTERFAZ



ANEXO 2 CÓDIGO PARA EL ARDUINO

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614 ();
int pwm = 3;
int pwmrecibido;
int cont=0;
int pert = 31;
int perturbacion;
byte buffer_enviar[6];
byte buffer_recibir[3];
static long temp;
float dato;
int entero;
int decimal;
double lectura;
int entAmb;
int decAmb;
double lectAmb;
int i=1;
unsigned long t1;
unsigned long t2;
unsigned long ts1;
unsigned long ts2;
void setup() {
 Serial.begin(9600);
 mlx.begin();
 pinMode(3,OUTPUT);
 pinMode(31,OUTPUT);
 t1=millis();
}

void loop() {
ts1=micros();
  if(cont==0){
    lectAmb=(mlx.readAmbientTempC());
    entAmb=(int)lectAmb;
    decAmb=(lectAmb-entAmb)*100;
     cont=1;
    }

   if(Serial.available()){

    for(byte i=0;i<=2;i++){
      buffer_recibir[i]=Serial.read();
      }
    if(buffer_recibir[0]==82 && buffer_recibir[2]==10){
      pwmrecibido=buffer_recibir[1];
      analogWrite(pwm,pwmrecibido);
      digitalWrite(pert,LOW);
      }
```

```
      t2=millis();
    if(buffer_recibir[0]==83 && buffer_recibir[2]==10){
      pwmrecibido=buffer_recibir[1];
      analogWrite(pwm,pwmrecibido);

      if(t2<=80000){
      digitalWrite(pert,LOW);
      }
       if(t2>80000 && t2<83000){
      digitalWrite(pert,HIGH);
      }
       if(t2>=83000 && t2<95000){
      digitalWrite(pert,LOW);
      }
       if(t2>=95000 && t2<100000){
      digitalWrite(pert,HIGH);
      }
       if(t2>100000){
      digitalWrite(pert,LOW);
      }
      }
    }

     lectura=(mlx.readObjectTempC());
     entero=(int)lectura;
     decimal=(lectura-entero)*100;

    buffer_enviar[0]='E';
    buffer_enviar[1]= entero;
    buffer_enviar[2]= decimal;
    buffer_enviar[3]= entAmb;
    buffer_enviar[4]= decAmb;
    buffer_enviar[5]='\n';
    Serial.write(buffer_enviar,6);
    ts2=micros();
    ttotal=ts2-ts1;
    Serial.println(ttotal);
    delay(249);
 }
```

ANEXO 3 CÓDIGOS DE FIGURAS DE INTERFAZ

CÓDIGO COMUN AL PRINCIPIO DE TODOS LOS CÓDIGOS DE FIGURAS

```
function varargout = cargar(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn', @cargar_OpeningFcn, ...
                   'gui_OutputFcn',  @cargar_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

CÓDIGO FIGURA “CARGAR”

```
function cargar_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1);
imdata=imread('imagen1.png');
imshow(imdata);

T=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',15);
assignin('base','T',T);
%datos de controladores
data1=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',1);
assignin('base','data1',data1);
data2=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',2);
assignin('base','data2ar',data2);
data3=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',3);
assignin('base','data3',data3);
data4=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',4);
assignin('base','data4',data4);
data5=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',5);
assignin('base','data5ar',data5);
data6=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',6);
assignin('base','data6ar',data6);
data7=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',7);
assignin('base','data7',data7);
data8=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',8);
assignin('base','data8',data8);
data9=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',9);
assignin('base','data9ar',data9);
data10=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',10);
assignin('base','data10ar',data10);
errorclasico=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',16);
assignin('base','errorclasico',errorclasico);
errorauto=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',17);
assignin('base','errorauto',errorauto);
errorfpd=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',18);
assignin('base','errorfpd',errorfpd);
r1=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',19);
assignin('base','r1',r1);
r2=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',20);
```

```
assignin('base','r2',r2);
fpd7=readfis('C:\InstructControl\fuzzy1\fpd7.fis');
assignin('base','fpd7',fpd7);
fpd8=readfis('C:\InstructControl\fuzzy1\fpd8.fis');
assignin('base','fpd8',fpd8);
fpd9=readfis('C:\InstructControl\fuzzy1\fpd9.fis');
assignin('base','fpd9',fpd9);
fpd10=readfis('C:\InstructControl\fuzzy1\fpd10.fis');
assignin('base','fpd10',fpd10);

error3=readfis('C:\InstructControl\fuzzy1\error3.fis');
assignin('base','error3',error3);
error4=readfis('C:\InstructControl\fuzzy1\error4.fis');
assignin('base','error4',error4);
error5=readfis('C:\InstructControl\fuzzy1\error5.fis');
assignin('base','error5',error5);
error6=readfis('C:\InstructControl\fuzzy1\error6.fis');
assignin('base','error6',error6);
integral3=readfis('C:\InstructControl\fuzzy1\integral3.fis');
assignin('base','integral3',integral3);
integral5=readfis('C:\InstructControl\fuzzy1\integral5.fis');
assignin('base','integral5',integral5);
integral6=readfis('C:\InstructControl\fuzzy1\integral6.fis');
assignin('base','integral6',integral6);
derivada3=readfis('C:\InstructControl\fuzzy1\derivada3.fis');
assignin('base','derivada3',derivada3);
derivada4=readfis('C:\InstructControl\fuzzy1\derivada4.fis');
assignin('base','derivada4',derivada4);
derivada5=readfis('C:\InstructControl\fuzzy1\derivada5.fis');
assignin('base','derivada5',derivada5);
derivada6=readfis('C:\InstructControl\fuzzy1\derivada6.fis');
assignin('base','derivada6',derivada6);
data1=xlsread('C:\InstructControl\guardarDatosG\data.xlsx',1);
assignin('base','data1',data1);
% datos lazo abierto
data7b1=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',1);
assignin('base','data7b1',data7b1);
data7b2=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',2);
assignin('base','data7b2',data7b2);
data7b3=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',3);
assignin('base','data7b3',data7b3);
data7b4=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',4);
assignin('base','data7b4',data7b4);
data7b5=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',5);
assignin('base','data7b5',data7b5);
data7b6=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',6);
assignin('base','data7b6',data7b6);
data7b7=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',7);
assignin('base','data7b7',data7b7);
data7b8=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',8);
assignin('base','data7b8',data7b8);
data7b9=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',9);
assignin('base','data7b9',data7b9);
data7b10=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',10);
```

```
assignin('base','data7b10',data7b10);
tiempo1=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',11);
assignin('base','tiempo1',tiempo1);
tiempo3=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',12);
assignin('base','tiempo3',tiempo3);
set1=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',13);
assignin('base','set1',set1);
set2=xlsread('C:\InstructControl\guardarDatosG\data2.xlsx',14);
assignin('base','set2',set2);
setpoint1=set2-set1;
assignin('base','setpoint1',setpoint1);

%grafica promedio lazo abierto
dataprom=xlsread('C:\InstructControl\guardarDatosG\dataset2.xlsx',11);
assignin('base','dataprom',dataprom);
rectat=xlsread('C:\InstructControl\guardarDatosG\dataset2.xlsx',12);
assignin('base','rectat',rectat);

%grafica sintonizacion lazo abierto
datamodelo=xlsread('C:\InstructControl\guardarDatosG\dataset2.xlsx',13);
assignin('base','datamodelo',datamodelo);
rectat2=xlsread('C:\InstructControl\guardarDatosG\dataset2.xlsx',14);
assignin('base','rectat2',rectat2);

%Datos de identificacion
ZiNiMi=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',1,'A1:A3')
;
assignin('base','ZiNiMi',ZiNiMi);
Smith=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',1,'B1:B4');
assignin('base','Smith',Smith);
Ho=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',1,'C1:C5');
assignin('base','Ho',Ho);
modelo=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',2,'A1:A3')
;
assignin('base','modelo',modelo);
metodo1=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',2,'B1:B3'
);
assignin('base','metodo1',metodo1);
metodo2=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',2,'C1:C3'
);
assignin('base','metodo2',metodo2);
metodo3=xlsread('C:\InstructControl\guardarDatosG\metodos.xlsx',2,'D1:D3'
);
assignin('base','metodo3',metodo3);

inic=0;
assignin('base','inic',inic);

perturbacion=0;
assignin('base','perturbacion',perturbacion);
handles.output = hObject;

guidata(hObject, handles);
```

```
function varargout = cargar_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function figure1_CloseRequestFcn(hObject, eventdata, handles)

delete(hObject);


function pushbutton1_Callback(hObject, eventdata, handles)

untitled;
```

CÓDIGO FIGURA “UNTITLED” VENTANA PRINCIPAL

```
function Restaurar_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1);
imdata=imread('advertencia.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);

function varargout = Restaurar_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
close (Restaurar);

function pushbutton2_Callback(hObject, eventdata, handles)
copyfile ('C:\InstructControl\restaurarArchivos','C:\InstructControl');
```

CÓDIGO FIGURA “IDENTIFICACIÓN”

```
function identificacion_OpeningFcn(hObject, eventdata, handles, varargin)
m2=evalin('base','m2');
if (m2==0)
close (untitled);
else
end
m3=evalin('base','m3');
if (m3==1)
    m2=0;
    assignin('base','m2',m2);
    m3=0;
```

```
    assignin('base','m3',m3);
else
end
axes(handles.axes1);
imdata=imread('fondoident.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);

function varargout = identificacion_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
function d1_Callback(hObject, eventdata, handles)

function d2_Callback(hObject, eventdata, handles)

function d3_Callback(hObject, eventdata, handles)

function d4_Callback(hObject, eventdata, handles)

function d5_Callback(hObject, eventdata, handles)

function d6_Callback(hObject, eventdata, handles)

function d7_Callback(hObject, eventdata, handles)

function d8_Callback(hObject, eventdata, handles)

function tomardatos_Callback(hObject, eventdata, handles)
find_system('name','sim7b');
open_system('sim7b');
open_system('sim7b/data7b');
tomadatos;

function procesar_Callback(hObject, eventdata, handles)
syms p1 p2 p3 p4 x y yec
tiempo1=evalin('base','tiempo1');
tiempo3=evalin('base','tiempo3');
T=evalin('base','T');
cd1= (floor(tiempo1/T))+1 ;
cd3= floor(tiempo3/T) ;
cd2=cd3-(cd1-1);

data7b1=evalin('base','data7b1');
dat1=data7b1(cd1:cd3,1:2);
assignin('base','dat1',dat1);
data7b2=evalin('base','data7b2');
dat2=data7b2(cd1:cd3,1:2);
assignin('base','dat2',dat2);
data7b3=evalin('base','data7b3');
dat3=data7b3(cd1:cd3,1:2);
assignin('base','dat3',dat3);
```

```
data7b4=evalin('base','data7b4');
dat4=data7b4(cd1:cd3,1:2);
assignin('base','dat4',dat4);
data7b5=evalin('base','data7b5');
dat5=data7b5(cd1:cd3,1:2);
assignin('base','dat5',dat5);
data7b6=evalin('base','data7b6');
dat6=data7b6(cd1:cd3,1:2);
assignin('base','dat6',dat6);
data7b7=evalin('base','data7b7');
dat7=data7b7(cd1:cd3,1:2);
assignin('base','dat7',dat7);
data7b8=evalin('base','data7b8');
dat8=data7b8(cd1:cd3,1:2);
assignin('base','dat8',dat8);
data7b9=evalin('base','data7b9');
dat9=data7b9(cd1:cd3,1:2);
assignin('base','dat9',dat9);
data7b10=evalin('base','data7b10');
dat10=data7b10(cd1:cd3,1:2);
assignin('base','dat10',dat10);
x1=(data7b1((cd1):(cd3), 1))-tiempo1;
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat1,1);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat2,2);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat3,3);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat4,4);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat5,5);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat6,6);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat7,7);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat8,8);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat9,9);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dat10,10);

Msum=(dat1(:,2))+(dat2(:,2))+(dat3(:,2))+(dat4(:,2))+(dat5(:,2))+(dat6(:,
2))+(dat7(:,2))+(dat8(:,2))+(dat9(:,2))+(dat10(:,2));
inic=(Msum(1,1))/10;
Mprom=(Msum/10)-inic;
Mprom(1,1)=0;
x1r=(x1(1:12,1));
Mpromr=Mprom(1:12);
assignin('base','Mpromr',Mpromr);%eliminar
assignin('base','Mprom',Mprom);%eliminar
assignin('base','x1',x1);%eliminar
assignin('base','x1r',x1r);%eliminar
dataprom=horzcat(x1,Mprom);
assignin('base','dataprom',dataprom);
%obtener recta tangente
f3=fit(x1r,Mpromr,'poly3');
fd=formula(f3);
f=polyfit(x1r,Mpromr,3); % matriz valores p del polinomio
y2=polyval(f,x1r); % evalua los valores de x1 en el polinomio
assignin('base','y2',y2);%eliminar
dprimera=diff(fd,x);
dsegunda=diff(dprimera,x);
```

```
d1=polyder(f);
d2=polyder(d1);
Xo=roots(d2);
assignin('base','Xo',Xo);%eliminar
g1=subs(fd,p1,f(1,1)); %sustituir variable para f(Xo)
g2=subs(g1,p2,f(1,2));
g3=subs(g2,p3,f(1,3));
g4=subs(g3,p4,f(1,4));
fXo=subs(g4,x,Xo);
%fXo=solve(-y + g5);
assignin('base','fXo',fXo);%eliminar
h1=subs(dprimera,p1,f(1,1)); %sustituir variable para f'(Xo)
h2=subs(h1,p2,f(1,2));
h3=subs(h2,p3,f(1,3));
dfXo=subs(h3,x,Xo);

recta= fXo + dfXo*(x-Xo); % ecuacion de la recta tangente

Yt= double(subs(recta,{x},x1));                    % y de recta tangente
rectat= horzcat(x1,Yt);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',rectat,12);
xlswrite('C:\InstructControl\guardarDatosG\dataset2.xlsx',dataprom,11);
assignin('base','dataprom',dataprom);
assignin('base','rectat',rectat);

% IDENTIFICACION
%Datos para metodos de la tangente
ecuacion= fXo + dfXo*(x-Xo);
tm1=double(solve(ecuacion,x));  % tiempo muerto para Z&N y Miller

yec= dataprom(cd2,2);
assignin('base','yec',yec);
ecuacion2= -yec + fXo + dfXo*(x-Xo);
taotm=double(solve(ecuacion2,x)); % valor de tao mas tiempo muerto
obtenido de recta tangente
tao=taotm-tm1; % tao Z&N
% Interpolar para mayor precision en el tiempo 63.2% Miller
Ym=yec*0.632; % valor del 63.2% Miller
assignin('base','Ym',Ym);
bv=find(Mprom==Ym);
if isempty(bv)
yP2i=find(Mprom>Ym,1); % Buscar el indice del primer valor mayor a Ym
(Punto2)
yP1i=yP2i-1; % Indice del valor anterior (Punto1)
%obtener valores para los puntos
xP2=x1(yP2i);
yP2=Mprom(yP2i);
xP1=x1(yP1i);
yP1=Mprom(yP1i);
xin=[xP1 xP2];
yin=[yP1 yP2];
```

```
taotm2=interp1(yin,xin,Ym,'linear'); % tao mas tiempo muerto metodo
Miller (63.2%)--- t2 metodo de Smith
tao2=taotm2-tm1;    % tao Miller
else
taotm2=x1(bv);
tao2=taotm2-tm1;    % tao Miller
end

% Smith
Yms=yec*0.283; % valor del 28.3% Smith
bv=find(Mprom==Yms);
if isempty(bv)
yP2i=find(Mprom>Yms,1); % Buscar el indice del primer valor mayor a Ym
(Punto2)
yP1i=yP2i-1; % Indice del valor anterior (Punto1)
%obtener valores para los puntos
xP2=x1(yP2i);
yP2=Mprom(yP2i);
xP1=x1(yP1i);
yP1=Mprom(yP1i);
xin=[xP1 xP2];
yin=[yP1 yP2];
t1s=interp1(yin,xin,Yms,'linear'); % t1 metodo Smith (28.3%)
else
  t1s=x1(bv);  % t1 Smith
end
taos=(-1.5*t1s)+(1.5*taotm2);   % tao Smith
tms=(1.5*t1s)-(0.5*taotm2);     % tiempo muerto Smith

% Ho et al.
Ymh=yec*0.35; % valor del 35%
bv=find(Mprom==Ymh);
if isempty(bv)
yP2i=find(Mprom>Ymh,1); % Buscar el indice del primer valor mayor a Ym
(Punto2)
yP1i=yP2i-1; % Indice del valor anterior (Punto1)
%obtener valores para los puntos
xP2=x1(yP2i);
yP2=Mprom(yP2i);
xP1=x1(yP1i);
yP1=Mprom(yP1i);
xin=[xP1 xP2];
yin=[yP1 yP2];
t1h=interp1(yin,xin,Ymh,'linear'); % t1 metodo Ho et al. (35%)
else
  t1h=x1(bv);  % t1  Ho et al.
end

Ymh=yec*0.85; % valor del 85%
bv=find(Mprom==Ymh);
if isempty(bv)
yP2i=find(Mprom>Ymh,1); % Buscar el indice del primer valor mayor a Ym
(Punto2)
yP1i=yP2i-1; % Indice del valor anterior (Punto1)
```

```
%obtener valores para los puntos
xP2=x1(yP2i);
yP2=Mprom(yP2i);
xP1=x1(yP1i);
yP1=Mprom(yP1i);
xin=[xP1 xP2];
yin=[yP1 yP2];
t2h=interp1(yin,xin,Ymh,'linear'); % t2 metodo Ho et al. (85%)
else
  t2h=x1(bv);  % t2  Ho et al.
end
taoh=(-0.67*t1h)+(0.67*t2h);  % tao Ho et al.
tmh=(1.3*t1h)-(0.29*t2h);    % tiempo muerto Ho et al.
set1=evalin('base','set1');
set2=evalin('base','set2');
setpoint=set2-set1;
assignin('base','set1',set1);
assignin('base','set2',set2);
Kp=yec/setpoint; % Ganancia para todos los modelos anteriores.

%Guardar valores de metodos de identificacion.
%Tangente (Ziegler y Nichols)(Miller)
ZiNiMi=[tm1 ; tao ; tao2];
assignin('base','ZiNiMi',ZiNiMi);
Smith=[t1s ; taotm2 ; taos ; tms];
assignin('base','Smith',Smith);
Ho=[t1h ; t2h ; taoh ; tmh ; Kp];
assignin('base','Ho',Ho);
Kp=Ho(5);
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',ZiNiMi,1,'A1:A3'
);
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',Smith,1,'B1:B4')
;
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',Ho,1,'C1:C5');

function pushbutton3_Callback(hObject, eventdata, handles)
dataprom=xlsread('C:\InstructControl\guardarDatosG\dataset2.xlsx',11);
assignin('base','dataprom',dataprom);
rectat=xlsread('C:\InstructControl\guardarDatosG\dataset2.xlsx',12);
assignin('base','rectat',rectat);
find_system('name','C:\InstructControl\comparar\Gprom');
open_system('C:\InstructControl\comparar\Gprom','loadonly');
set_param('Gprom','SimulationCommand','Start');
open_system('Gprom/ScopProm');

function pushbutton4_Callback(hObject, eventdata, handles)

ZiNiMi=evalin('base','ZiNiMi'); % Z&N Miller
Smith=evalin('base','Smith');
Ho=evalin('base','Ho');
setpoint1=evalin('base','setpoint1');
tm1=ZiNiMi(1);
tao=ZiNiMi(2);
tao2=ZiNiMi(3);
```

```
taos=Smith(3);
tms=Smith(4);
taoh=Ho(3);
tmh=Ho(4);
Kp=Ho(5);
assignin('base','tm1',tm1);
assignin('base','tao',tao);
assignin('base','tao2',tao2);
assignin('base','taos',taos);
assignin('base','tms',tms);
assignin('base','taoh',taoh);
assignin('base','tmh',tmh);
assignin('base','Kp',Kp);
open_system('C:\InstructControl\comparar\compMet','loadonly');
pause(1);
set_param('compMet/Step','After',num2str(setpoint1));
set_param('compMet','SimulationCommand','Start');
open_system('compMet/Scop');
function pushbutton5_Callback(hObject, eventdata, handles)

function pushbutton6_Callback(hObject, eventdata, handles)

function d9_Callback(hObject, eventdata, handles)

function d10_Callback(hObject, eventdata, handles)

function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)

if hObject == handles.d1
    bdata=1;
    assignin('base','bdata',bdata);
elseif hObject== handles.d2
    bdata=2;
    assignin('base','bdata',bdata);
elseif hObject== handles.d3
    bdata=3;
    assignin('base','bdata',bdata);
elseif hObject== handles.d4
    bdata=4;
    assignin('base','bdata',bdata);
elseif hObject== handles.d5
    bdata=5;
    assignin('base','bdata',bdata);
elseif hObject== handles.d6
    bdata=6;
    assignin('base','bdata',bdata);
elseif hObject== handles.d7
    bdata=7;
    assignin('base','bdata',bdata);
elseif hObject== handles.d8
    bdata=8;
    assignin('base','bdata',bdata);
elseif hObject== handles.d9
```

```
    bdata=9;
    assignin('base','bdata',bdata);
elseif hObject== handles.d10
    bdata=10;
    assignin('base','bdata',bdata);
else
end

function figure1_CloseRequestFcn(hObject, eventdata, handles)
m1=1;
assignin('base','m1',m1);
pause(1);
m2=evalin('base','m2');
if (m2==0)
untitled;
else
end
delete(hObject);

function pushbutton7_Callback(hObject, eventdata, handles)
open_system('C:\InstructControl\comparar\data','loadonly');
set_param('data','SimulationCommand','Start');
open_system('data/datai');

function pushbutton8_Callback(hObject, eventdata, handles)
Resultados;

function pushbutton9_Callback(hObject, eventdata, handles)
ZiNiMi=evalin('base','ZiNiMi'); % Z&N Miller
Smith=evalin('base','Smith');
Ho=evalin('base','Ho');
setpoint1=evalin('base','setpoint1');
tm1=ZiNiMi(1);
tao=ZiNiMi(2);
tao2=ZiNiMi(3);
taos=Smith(3);
tms=Smith(4);
taoh=Ho(3);
tmh=Ho(4);
Kp=Ho(5);
assignin('base','tm1',tm1);
assignin('base','tao',tao);
assignin('base','tao2',tao2);
assignin('base','taos',taos);
assignin('base','tms',tms);
assignin('base','taoh',taoh);
assignin('base','tmh',tmh);
assignin('base','Kp',Kp);
open_system('C:\InstructControl\comparar\compMet','loadonly');
pause(1);
set_param('compMet/Step','After',num2str(setpoint1));
set_param('compMet','SimulationCommand','Start');
Indices;
```

```
function pushbutton10_Callback(hObject, eventdata, handles)
Formula;
```

CÓDIGO FIGURA “SINTONIZACIÓN”

```
function Sintonizacion_OpeningFcn(hObject, eventdata, handles, varargin)
m2=1;
assignin('base','m2',m2);
close (untitled);
axes(handles.axes1);
imdata=imread('fondosint.png');
imshow(imdata);
handles.output = hObject;

guidata(hObject, handles);

function varargout = Sintonizacion_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function pushbutton1_Callback(hObject, eventdata, handles)
T=evalin('base','T');
mod=evalin('base','mod');
Ho=evalin('base','Ho');
if mod==1
    ZiNiMi=evalin('base','ZiNiMi');
    tm=ZiNiMi(1);
    assignin('base','tm',tm);
    tao=ZiNiMi(2);
    assignin('base','tao',tao);
    Kp=Ho(5);
    assignin('base','Kp',Kp);
    modelo=[tm ; tao ; Kp];
    assignin('base','modelo',modelo);
elseif mod==2
   ZiNiMi=evalin('base','ZiNiMi');
    tm=ZiNiMi(1);
    assignin('base','tm',tm);
    tao=ZiNiMi(3);
    assignin('base','tao',tao);
    Kp=Ho(5);
    assignin('base','Kp',Kp);
    modelo=[tm ; tao ; Kp];
    assignin('base','modelo',modelo);
elseif mod==3
     Smith=evalin('base','Smith');
    tm=Smith(4);
```

```
    assignin('base','tm',tm);
    tao=Smith(3);
    assignin('base','tao',tao);
    Kp=Ho(5);
    assignin('base','Kp',Kp);
    modelo=[tm ; tao ; Kp];
    assignin('base','modelo',modelo);
elseif mod==4
    Ho=evalin('base','Ho');
    tm=Ho(4);
    assignin('base','tm',tm);
    tao=Ho(3);
    assignin('base','tao',tao);
    Kp=Ho(5);
    assignin('base','Kp',Kp);
    modelo=[tm ; tao ; Kp];
    assignin('base','modelo',modelo);
else
end

% Metodo 1 Z&N
Kpc1=(1.2/Kp)*(tao/tm);
Kic1=Kpc1/(2*tm);
Kdc1=Kpc1*0.5*tm;
 metodo1=[Kpc1 ; Kic1 ; Kdc1];
 assignin('base','metodo1',metodo1);
% Metodo de cohen coon
Kpc2=(tao/(Kp*tm))*((4/3)+(tm/(4*tao)));
Kic2=Kpc2/((tm*(32*tao+6*tm))/(13*tao+8*tm));
Kdc2=Kpc2*((4*tm*tao)/(11*tao+2*tm));
metodo2=[Kpc2 ; Kic2 ; Kdc2];
assignin('base','metodo2',metodo2);
% Metodo Chien-Horns-Reswick
Kpc3=(0.95*tao)/(Kp*tm);
Kic3=Kpc3/(2.375*tm);
Kdc3=Kpc3*0.421*tm;
metodo3=[Kpc3 ; Kic3 ; Kdc3];
assignin('base','metodo3',metodo3);
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',metodo1,2,'B1:B3
');
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',metodo2,2,'C1:C3
');
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',metodo3,2,'D1:D3
');
xlswrite('C:\InstructControl\guardarDatosG\metodos.xlsx',modelo,2,'A1:A3'
);

function pushbutton2_Callback(hObject, eventdata, handles)
Resultados2;

function pushbutton3_Callback(hObject, eventdata, handles)
modelo=evalin('base','modelo');
tm=modelo(1);
tao=modelo(2);
```

```
Kp=modelo(3);
assignin('base','tm',tm);
assignin('base','tao',tao);
assignin('base','Kp',Kp);
metodo1=evalin('base','metodo1');
metodo2=evalin('base','metodo2');
metodo3=evalin('base','metodo3');
Kpc1=metodo1(1);
Kic1=metodo1(2);
Kdc1=metodo1(3);
Kpc2=metodo2(1);
Kic2=metodo2(2);
Kdc2=metodo2(3);
Kpc3=metodo3(1);
Kic3=metodo3(2);
Kdc3=metodo3(3);
assignin('base','Kpc1',Kpc1);
assignin('base','Kic1',Kic1);
assignin('base','Kdc1',Kdc1);
assignin('base','Kpc2',Kpc2);
assignin('base','Kic2',Kic2);
assignin('base','Kdc2',Kdc2);
assignin('base','Kpc3',Kpc3);
assignin('base','Kic3',Kic3);
assignin('base','Kdc3',Kdc3);
open_system('C:\InstructControl\comparar\compMetSint','loadonly');
set_param('compMetSint','SimulationCommand','Start');
Indices2;

function pushbutton4_Callback(hObject, eventdata, handles)
modelo=evalin('base','modelo');
tm=modelo(1);
tao=modelo(2);
Kp=modelo(3);
assignin('base','tm',tm);
assignin('base','tao',tao);
assignin('base','Kp',Kp);
metodo1=evalin('base','metodo1');
metodo2=evalin('base','metodo2');
metodo3=evalin('base','metodo3');
Kpc1=metodo1(1);
Kic1=metodo1(2);
Kdc1=metodo1(3);
Kpc2=metodo2(1);
Kic2=metodo2(2);
Kdc2=metodo2(3);
Kpc3=metodo3(1);
Kic3=metodo3(2);
Kdc3=metodo3(3);
assignin('base','Kpc1',Kpc1);
assignin('base','Kic1',Kic1);
assignin('base','Kdc1',Kdc1);
assignin('base','Kpc2',Kpc2);
assignin('base','Kic2',Kic2);
```

```
assignin('base','Kdc2',Kdc2);
assignin('base','Kpc3',Kpc3);
assignin('base','Kic3',Kic3);
assignin('base','Kdc3',Kdc3);
open_system('C:\InstructControl\comparar\compMetSint','loadonly');
set_param('compMetSint','SimulationCommand','Start');
open_system('compMetSint/Scop');
function pushbutton5_Callback(hObject, eventdata, handles)
Controlador;

function figure1_CloseRequestFcn(hObject, eventdata, handles)
m3=1;
assignin('base','m3',m3);
untitled;
delete(hObject);

function radiobutton5_Callback(hObject, eventdata, handles)
function radiobutton6_Callback(hObject, eventdata, handles)
function radiobutton7_Callback(hObject, eventdata, handles)
function md1_Callback(hObject, eventdata, handles)
function md2_Callback(hObject, eventdata, handles)
function md3_Callback(hObject, eventdata, handles)
function id4_Callback(hObject, eventdata, handles)
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)

if hObject==handles.id1
    mod=1;
    assignin('base','mod',mod);
elseif hObject==handles.id2
    mod=2;
    assignin('base','mod',mod);
elseif hObject==handles.id3
    mod=3;
    assignin('base','mod',mod);
elseif hObject==handles.id4
    mod=4;
    assignin('base','mod',mod);
else
end

function uibuttongroup2_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.md1
    bd=1;
    assignin('base','bd',bd);
elseif hObject== handles.md2
    bd=2;
    assignin('base','bd',bd);
elseif hObject== handles.md3
    bd=3;
    assignin('base','bd',bd);
else
end
function id1_Callback(hObject, eventdata, handles)
function id2_Callback(hObject, eventdata, handles)
```

```
function id3_Callback(hObject, eventdata, handles)
function pushbutton7_Callback(hObject, eventdata, handles)
Rango;
```

CÓDIGO FIGURA "CONTROLADORES"

```
function controladores_OpeningFcn(hObject, eventdata, handles, varargin)
m2=evalin('base','m2');
if (m2==0)
close (untitled);
else
end
m3=evalin('base','m3');
if (m3==1)
    m2=0;
    assignin('base','m2',m2);
    m3=0;
    assignin('base','m3',m3);
else
end
modelo=evalin('base','modelo');
tm=modelo(1);
tao=modelo(2);
Kp=modelo(3);
assignin('base','tm',tm);
assignin('base','tao',tao);
assignin('base','Kp',Kp);

axes(handles.axes1);
imdata=imread('fondocont.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);
function varargout = controladores_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function clasico_Callback(hObject, eventdata, handles)
clasico;
function tuning_Callback(hObject, eventdata, handles)
tuning;
function difuso_Callback(hObject, eventdata, handles)
difuso;
function figure1_CloseRequestFcn(hObject, eventdata, handles)
m1=1;
assignin('base','m1',m1);
pause(1);
m2=evalin('base','m2');
if (m2==0)
untitled;
else
end
delete(hObject);
```

CÓDIGO FIGURA “GRAFICA”

```
function grafica_OpeningFcn(hObject, eventdata, handles, varargin)
m2=1;
assignin('base','m2',m2);
close (untitled);
m7=0;
assignin('base','m7',m7);
s1=0;
assignin('base','s1',s1);
s2=0;
assignin('base','s2',s2);
s3=0;
assignin('base','s3',s3);
s4=0;
assignin('base','s4',s4);
s5=0;
assignin('base','s5',s5);
s6=0;
assignin('base','s6',s6);
s7=0;
assignin('base','s7',s7);
s8=0;
assignin('base','s8',s8);
s9=0;
assignin('base','s9',s9);
s10=0;
assignin('base','s10',s10);
axes(handles.axes1);
imdata=imread('fondograf.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);

function varargout = grafica_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function bot1_Callback(hObject, eventdata, handles)
s1=get(hObject,'Value');
assignin('base','s1',s1);
function bot2_Callback(hObject, eventdata, handles)
s2=get(hObject,'Value');
assignin('base','s2',s2);
function bot3_Callback(hObject, eventdata, handles)
s3=get(hObject,'Value');
assignin('base','s3',s3);
function bot4_Callback(hObject, eventdata, handles)
s4=get(hObject,'Value');
assignin('base','s4',s4);
function bot5_Callback(hObject, eventdata, handles)
s5=get(hObject,'Value');
assignin('base','s5',s5);
function bot6_Callback(hObject, eventdata, handles)
s6=get(hObject,'Value');
```

```
assignin('base','s6',s6);
function bot7_Callback(hObject, eventdata, handles)
s7=get(hObject,'Value');
assignin('base','s7',s7);
function bot8_Callback(hObject, eventdata, handles)
s8=get(hObject,'Value');
assignin('base','s8',s8);
function bot9_Callback(hObject, eventdata, handles)
s9=get(hObject,'Value');
assignin('base','s9',s9);
function bot10_Callback(hObject, eventdata, handles)
s10=get(hObject,'Value');
assignin('base','s10',s10);
function comparar_Callback(hObject, eventdata, handles)
    open_system('C:\InstructControl\comparar\SimulacionData','loadonly');
    s1=evalin('base','s1');
    s2=evalin('base','s2');
    s3=evalin('base','s3');
    s4=evalin('base','s4');
    s5=evalin('base','s5');
    s6=evalin('base','s6');
    s7=evalin('base','s7');
    s8=evalin('base','s8');
    s9=evalin('base','s9');
    s10=evalin('base','s10');
set_param('SimulacionData/c1','Gain',num2str(s1));
set_param('SimulacionData/c2','Gain',num2str(s2));
set_param('SimulacionData/c3','Gain',num2str(s3));
set_param('SimulacionData/c4','Gain',num2str(s4));
set_param('SimulacionData/c5','Gain',num2str(s5));
set_param('SimulacionData/c6','Gain',num2str(s6));
set_param('SimulacionData/c7','Gain',num2str(s7));
set_param('SimulacionData/c8','Gain',num2str(s8));
set_param('SimulacionData/c9','Gain',num2str(s9));
set_param('SimulacionData/c10','Gain',num2str(s10));
set_param('SimulacionData','SimulationCommand','Start');
open_system('SimulacionData/Scope7');
m7=1;
assignin('base','m7',m7);
function figure1_CloseRequestFcn(hObject, eventdata, handles)
m7=evalin('base','m7');
if (m7==1)
save_system('SimulacionData');
close_system('SimulacionData');
else
end
m1=1;
assignin('base','m1',m1);
untitled;
delete(hObject);
```

CÓDIGO FIGURA “RESTAURAR”

```
function Restaurar_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1);
imdata=imread('advertencia.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);

function varargout = Restaurar_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
close (Restaurar);

function pushbutton2_Callback(hObject, eventdata, handles)
copyfile ('C:\InstructControl\restaurarArchivos','C:\InstructControl');
```

CÓDIGO FIGURA “RANGO”

```
function Rango_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1);
imdata=imread('ind.png');
imshow(imdata);
handles.output = hObject;

guidata(hObject, handles);
function varargout = Rango_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function rango1_Callback(hObject, eventdata, handles)
function rango1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rango2_Callback(hObject, eventdata, handles)

function rango2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton1_Callback(hObject, eventdata, handles)
T=evalin('base','T');
errorclasico=evalin('base','errorclasico');
errorauto=evalin('base','errorauto');
errorfpd=evalin('base','errorfpd');
r1=evalin('base','r1');
r2=evalin('base','r2');
rango1s=get(handles.rango1,'String');
```

```
rango2s=get(handles.rango2,'String');
rango1n=str2num(rango1s);
rango2n=str2num(rango2s);
if rango2n>rango1n
r1=rango1n/T;
r2=rango2n/T;
else
end
dfg=r2-r1;
tiempo4=rango2n-rango1n;
if dfg>30
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',r1,19);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',r2,20);
errorclasicor=errorclasico(r1:r2,1:2);
errorclasicor=horzcat((errorclasicor(:,1)-(rango1n-
0.25)),errorclasicor(:,2));
assignin('base','errorclasicor',errorclasicor);
errorautor=errorauto(r1:r2,1:2);
errorautor=horzcat((errorautor(:,1)-(rango1n-0.25)),errorautor(:,2));
assignin('base','errorautor',errorautor);
errorfpdr=errorfpd(r1:r2,1:2);
errorfpdr=horzcat((errorfpdr(:,1)-(rango1n-0.25)),errorfpdr(:,2));
assignin('base','errorfpdr',errorfpdr);
assignin('base','tiempo4',tiempo4);
else
end
find_system('name','compReal');
open_system('C:\InstructControl\comparar\compReal','loadonly');
set_param('compReal','SimulationCommand','Start');
pause(10);
IndicesReal;
close (Rango);
function figure1_CloseRequestFcn(hObject, eventdata, handles)
delete(hObject);
```

CÓDIGO FIGURA “INDICESREAL”

```
function IndicesReal_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1);
imdata=imread('fondoreal.png');
imshow(imdata);
itaecla=evalin('base','itaecla');
itaeauto=evalin('base','itaeauto');
itaedif=evalin('base','itaedif');
isecla=evalin('base','isecla');
iseauto=evalin('base','iseauto');
isedif=evalin('base','isedif');
tam=size(isecla);
tam=tam(1);
valorisecla=isecla(tam);
tam=size(iseauto);
tam=tam(1);
```

```
valoriseauto=iseauto(tam);
tam=size(isedif);
tam=tam(1);
valorisedif=isedif(tam);
tam=size(itaecla);
tam=tam(1);
valoritaecla=itaecla(tam);
tam=size(itaeauto);
tam=tam(1);
valoritaeauto=itaeauto(tam);
tam=size(itaedif);
tam=tam(1);
valoritaedif=itaedif(tam);
set(handles.edit1,'String',valorisecla);
set(handles.edit2,'String',valoriseauto);
set(handles.edit3,'String',valorisedif);
set(handles.edit4,'String',valoritaecla);
set(handles.edit5,'String',valoritaeauto);
set(handles.edit6,'String',valoritaedif);
handles.output = hObject;
guidata(hObject, handles);
function varargout = IndicesReal_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

CÓDIGO FIGURA “TUNING”

```
function tuning_OpeningFcn(hObject, eventdata, handles, varargin)
m2=1;
assignin('base','m2',m2);
tipo=0;
assignin('base','tipo',tipo);
estrategia=1;
assignin('base','estrategia',estrategia);
close (controladores);
axes(handles.axes1);
imdata=imread('fondotuning.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);
function varargout = tuning_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function checkbox1_Callback(hObject, eventdata, handles)
function but1_Callback(hObject, eventdata, handles)
function but2_Callback(hObject, eventdata, handles)
function radiobutton3_Callback(hObject, eventdata, handles)
function radiobutton4_Callback(hObject, eventdata, handles)
function butok_Callback(hObject, eventdata, handles)
estrategia=evalin('base','estrategia');
tipo=evalin('base','tipo');
grafica=estrategia+tipo;
assignin('base','grafica',grafica);
```

```
grafica=evalin('base','grafica');
perturbacion=evalin('base','perturbacion');
if (grafica==1)

    find_system('name','sim3');
    open_system('sim3');
    gg=3;
    assignin('base','gg',gg);
elseif (grafica==2)

    find_system('name','sim4');
    open_system('sim4');
    gg=4;
    assignin('base','gg',gg);
elseif (grafica==3)

    find_system('name','sim5');
    open_system('sim5');
     set_param('sim5/EnviarDatos/SerialSend','Header',perturbacion);
     gg=5;
    assignin('base','gg',gg);
elseif (grafica==4)

    find_system('name','sim6');
    open_system('sim6');
     set_param('sim6/EnviarDatos/SerialSend','Header',perturbacion);
     gg=6;
    assignin('base','gg',gg);
else
end

function figure1_CloseRequestFcn(hObject, eventdata, handles)
m3=1;
assignin('base','m3',m3);
controladores;
delete(hObject);

function uibuttongroup2_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.but1
    tipo=0;
    assignin('base','tipo',tipo);
elseif hObject== handles.but2
    tipo=2;
     assignin('base','tipo',tipo);
else
end

function but1_KeyPressFcn(hObject, eventdata, handles)
function uibuttongroup3_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.but3
    estrategia=1;
    assignin('base','estrategia',estrategia);
elseif hObject== handles.but4
```

```
    estrategia=2;
    assignin('base','estrategia',estrategia);
else
end

function butok_KeyPressFcn(hObject, eventdata, handles)
function uibuttongroup4_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.d1
    perturbacion='R';
    assignin('base','perturbacion',perturbacion);
elseif hObject== handles.d2
     perturbacion='S';
    assignin('base','perturbacion',perturbacion);
else
end
function pushbutton2_Callback(hObject, eventdata, handles)
gg=evalin('base','gg');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',cell(4000,2),gg,'A1
:B4000');
sum=0;
if gg==3
data3=evalin('base','data3');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data3,3);
elseif gg==4
data4=evalin('base','data4');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data4,4);
elseif gg==5
data5=evalin('base','data5');  %convertir de struct a array
errorautos=evalin('base','errorautos');
ys1=[];
ys2=[];
filas=size(data5.time,1);
    tmax= data5.time(filas,1);
    ndatos=floor(tmax/0.25);

for i=1:ndatos+1
xs1(i,1)=sum;
xs2(i,1)=sum;
posicion=find(data5.time==sum);
ys1(i,1)= data5.signals.values(:,:,posicion);
ys2(i,1)= errorautos.signals.values(posicion);
sum=sum+0.25;
end
data5= horzcat(xs1,ys1);
errorauto=horzcat(xs2,ys2);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',cell(4000,2),17,'A1
:B4000');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',errorauto,17);
assignin('base','errorauto',errorauto);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data5,5);
assignin('base','data5ar',data5);
elseif gg==6
data6=evalin('base','data6');  %convertir de struct a array
ys1=[];
```

```
filas=size(data6.time,1);
      tmax= data6.time(filas,1);
      ndatos=floor(tmax/0.25);
for i=1:ndatos+1
xs1(i,1)=sum;
posicion=find(data6.time==sum);
ys1(i,1)= data6.signals.values(:,:,posicion);
sum=sum+0.25;
end
data6= horzcat(xs1,ys1);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data6,6);
assignin('base','data6ar',data6);
else
end
```

CÓDIGO FIGURA “CLASICO”

```
function clasico_OpeningFcn(hObject, eventdata, handles, varargin)
m2=1;
assignin('base','m2',m2);
tipo=0;
assignin('base','tipo',tipo);
close (controladores);
axes(handles.axes1);
imdata=imread('fondo2.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);

function varargout = clasico_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function pushbutton1_Callback(hObject, eventdata, handles) % abrir
diagrama de bloques
tipo=evalin('base','tipo');
perturbacion=evalin('base','perturbacion');
if (tipo==1)
    find_system('name','sim1');
    open_system('sim1');
 gg=1;
 assignin('base','gg',gg);
    %open_system('sim1/data1');
elseif (tipo==2)
    find_system('name','sim2');
    open_system('sim2');
    set_param('sim2/EnviarDatos/SerialSend','Header',perturbacion);
    gg=2;
    assignin('base','gg',gg);
    %open_system('sim2/data2');
else
end
function but3_Callback(hObject, eventdata, handles)
function but4_Callback(hObject, eventdata, handles)
```

```
function but5_Callback(hObject, eventdata, handles)
function but1_Callback(hObject, eventdata, handles)
function but2_Callback(hObject, eventdata, handles)
function figure1_CloseRequestFcn(hObject, eventdata, handles)
m3=1;
assignin('base','m3',m3);
controladores;
delete(hObject);
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.but1
    tipo=1;
    assignin('base','tipo',tipo);
elseif hObject== handles.but2
    tipo=2;
     assignin('base','tipo',tipo);
else
end

function uibuttongroup3_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.d1
    perturbacion='R';
    assignin('base','perturbacion',perturbacion);
elseif hObject== handles.d2
    perturbacion='S';
    assignin('base','perturbacion',perturbacion);
else
end

function pushbutton2_Callback(hObject, eventdata, handles)
gg=evalin('base','gg');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',cell(4000,2),gg,'A1
:B4000');
sum=0;
if gg==1
data1=evalin('base','data1');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data1,1);
elseif gg==2
data2=evalin('base','data2');  %convertir de struct a array
errorclasicos=evalin('base','errorclasicos');
ys1=[];
ys2=[];
filas=size(data2.time,1);
    tmax= data2.time(filas,1);
    ndatos=floor(tmax/0.25);

for i=1:ndatos+1
xs1(i,1)=sum;
xs2(i,1)=sum;
posicion=find(data2.time==sum);
ys1(i,1)= data2.signals.values(:,:,posicion);
ys2(i,1)= errorclasicos.signals.values(posicion);
sum=sum+0.25;
end
data2= horzcat(xs1,ys1);
```

```
errorclasico=horzcat(xs2,ys2);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',cell(4000,2),16,'A1
:B4000');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',errorclasico,16);
assignin('base','errorclasico',errorclasico);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data2,2);
assignin('base','data2ar',data2);
else
end
```

CÓDIGO FIGURA “DIFUSO”

```
function difuso_OpeningFcn(hObject, eventdata, handles, varargin)
m2=1;
assignin('base','m2',m2);
tipo=0;
assignin('base','tipo',tipo);
estrategia=1;
assignin('base','estrategia',estrategia);
close (controladores);
axes(handles.axes1);
imdata=imread('fondofuzzy.png');
imshow(imdata);
handles.output = hObject;
guidata(hObject, handles);
function varargout = difuso_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function pushbutton1_Callback(hObject, eventdata, handles)
estrategia=evalin('base','estrategia');
tipo=evalin('base','tipo');
grafica=estrategia+tipo;
assignin('base','grafica',grafica);
grafica=evalin('base','grafica');
perturbacion=evalin('base','perturbacion');
if (grafica==1)
    find_system('name','sim7');
    open_system('sim7');
    gg=7;
    assignin('base','gg',gg);
    %open_system('sim7/data7');
elseif (grafica==2)
    find_system('name','sim8');
    open_system('sim8');
    gg=8;
    assignin('base','gg',gg);
    %open_system('sim8/data8');
elseif (grafica==3)
    find_system('name','sim9');
    open_system('sim9');
     set_param('sim9/EnviarDatos/SerialSend','Header',perturbacion);
     gg=9;
    assignin('base','gg',gg);
    %open_system('sim9/data9');
```

```
elseif (grafica==4)
    find_system('name','sim10');
    open_system('sim10');
    set_param('sim10/EnviarDatos/SerialSend','Header',perturbacion);
    gg=10;
    assignin('base','gg',gg);
    %open_system('sim10/data10');
else
end

function figure1_CloseRequestFcn(hObject, eventdata, handles)
m3=1;
assignin('base','m3',m3);
controladores;
delete(hObject);

function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.but1
    tipo=0;
    assignin('base','tipo',tipo);
elseif hObject== handles.but2
    tipo=2;
     assignin('base','tipo',tipo);
else
end

function uibuttongroup2_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.but3
    estrategia=1;
    assignin('base','estrategia',estrategia);
elseif hObject== handles.but4
    estrategia=2;
    assignin('base','estrategia',estrategia);
else
end

function uibuttongroup3_SelectionChangedFcn(hObject, eventdata, handles)
if hObject == handles.d1
     perturbacion='R';
    assignin('base','perturbacion',perturbacion);
elseif hObject== handles.d2
     perturbacion='S';
    assignin('base','perturbacion',perturbacion);
else
end

function pushbutton2_Callback(hObject, eventdata, handles)
gg=evalin('base','gg');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',cell(4000,2),gg,'A1
:B4000');
sum=0;
sum2=0;
if gg==7
```

```
data7=evalin('base','data7');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data7,7);
elseif gg==8
data8=evalin('base','data8');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data8,8);
elseif gg==9
data9=evalin('base','data9');  %convertir de struct a array
errorfpds=evalin('base','errorfpds');
ys1=[];
ys2=[];
filas=size(data9.time,1);
    tmax= data9.time(filas,1);
    ndatos=floor(tmax/0.25);
for i=1:ndatos+1
xs1(i,1)=sum;
xs2(i,1)=sum;
posicion=find(data9.time==sum);
ys1(i,1)= data9.signals.values(:,:,posicion);
ys2(i,1)= errorfpds.signals.values(posicion);
sum=sum+0.25;
end
data9= horzcat(xs1,ys1);
errorfpd=horzcat(xs2,ys2);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',cell(4000,2),18,'A1
:B4000');
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',errorfpd,18);
assignin('base','errorfpd',errorfpd);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data9,9);
assignin('base','data9ar',data9);
elseif gg==10
data10=evalin('base','data10');  %convertir de struct a array
ys1=[];
filas=size(data10.time,1);
      tmax= data10.time(filas,1);
      ndatos=floor(tmax/0.25);
for i=1:ndatos+1
 xs1(i,1)=sum;
posicion=find(data10.time==sum);
ys1(i,1)= data10.signals.values(:,:,posicion);
sum=sum+0.25;
end
data10= horzcat(xs1,ys1);
xlswrite('C:\InstructControl\guardarDatosG\data.xlsx',data10,10);
assignin('base','data10ar',data10);
else
end
```

ANEXO 4 MANUAL DE USO

MANUAL DE USO INTERFAZ GRÁFICA DE USUARIO

1 Introducción

El objetivo de esta interfaz es facilitar el diseño del control PID auto sintonizable y FPD+I ahorrando tiempo mediante herramientas de identificación, sintonización y análisis.

2 Requerimientos de la interfaz

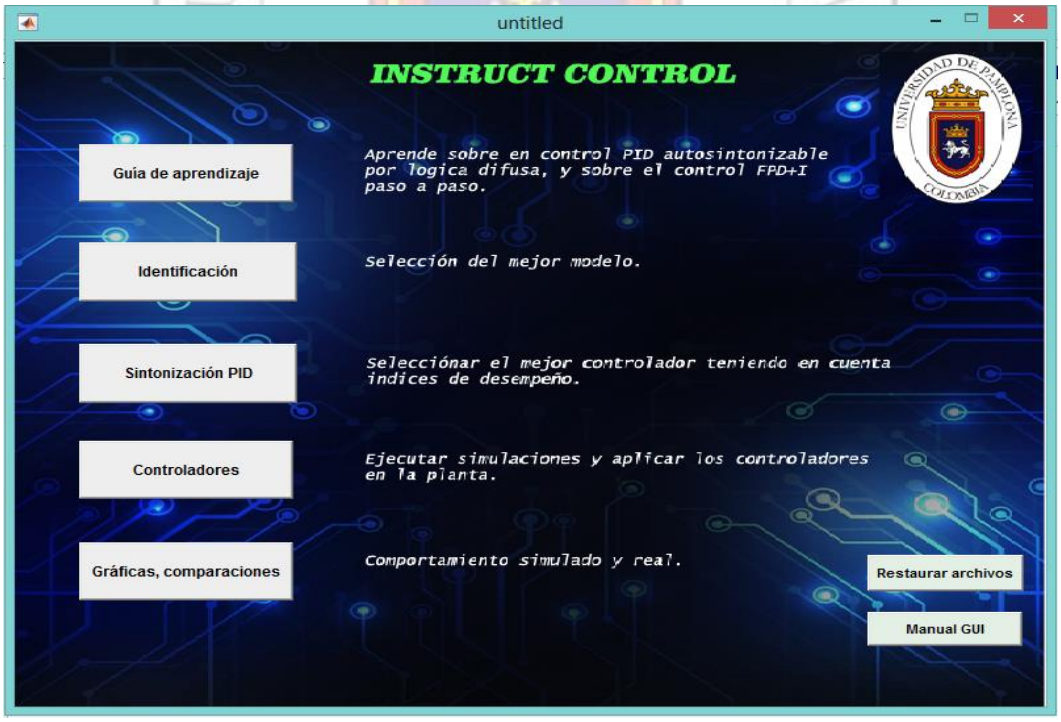
Para poder conectar simulink con arduino se debe instalar el paquete de soporte de Matlab para arduino y verificar que se tenga instalada la librería Simulink Real-Time; también es necesario tener grabado en la placa de arduino el código para la comunicación con simulink, y se encuentra en el anexo al final de este manual. Preferiblemente que la computadora donde se va a usar la interfaz tenga un procesador Core i5 o superior, o su equivalente.

3 Contenido de GUIDE

3.1 Ventana principal

En la ventana principal de la figura 1 se encuentran las opciones para el aprendizaje, ejecución y análisis.

Figura 1 Ventana principal



Fuente: Autores

3.2 Curso PDF

Al seleccionar “Curso PDF” abrirá un documento con el paso a paso desde la identificación de la planta, sintonización, graficas comparativas y uso de indicadores de rendimiento.

3.3 Identificación

Al seleccionar “Identificación” se abre la siguiente ventana de la figura 2.

Figura 2 Ventana de identificación

Fuente: Autores

1. Toma de datos.

Figura 3 Toma de datos

Fuente: Autores

En toma de datos de la figura 3 se debe registrar 10 graficas de la planta al mismo set point a lazo abierto.

Se selecciona el respectivo registro y al dar clic en “Tomar datos” figura 4, se abre la siguiente ventana en donde se debe ingresar los set point y el tiempo.

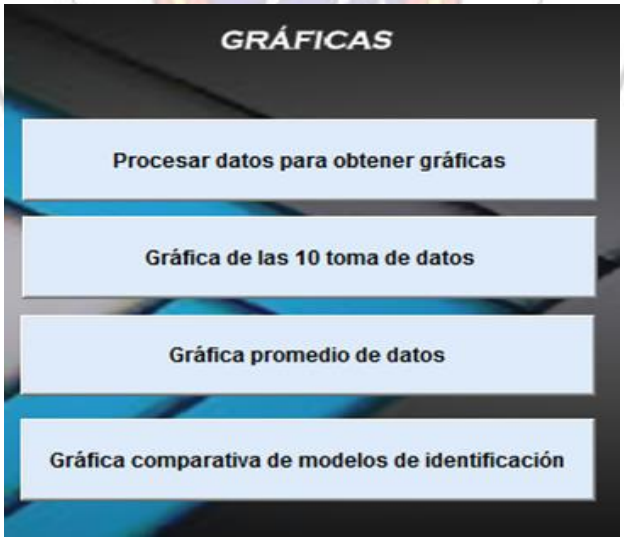
Figura 4 Tiempo toma de datos

Fuente: Autores

2. Gráficas

Antes de abrir cualquier gráfica dentro de “Identificación”, se debe seleccionar “Procesar datos para obtener gráficas” de la figura 5.

Figura 5 Gráficas de identificación



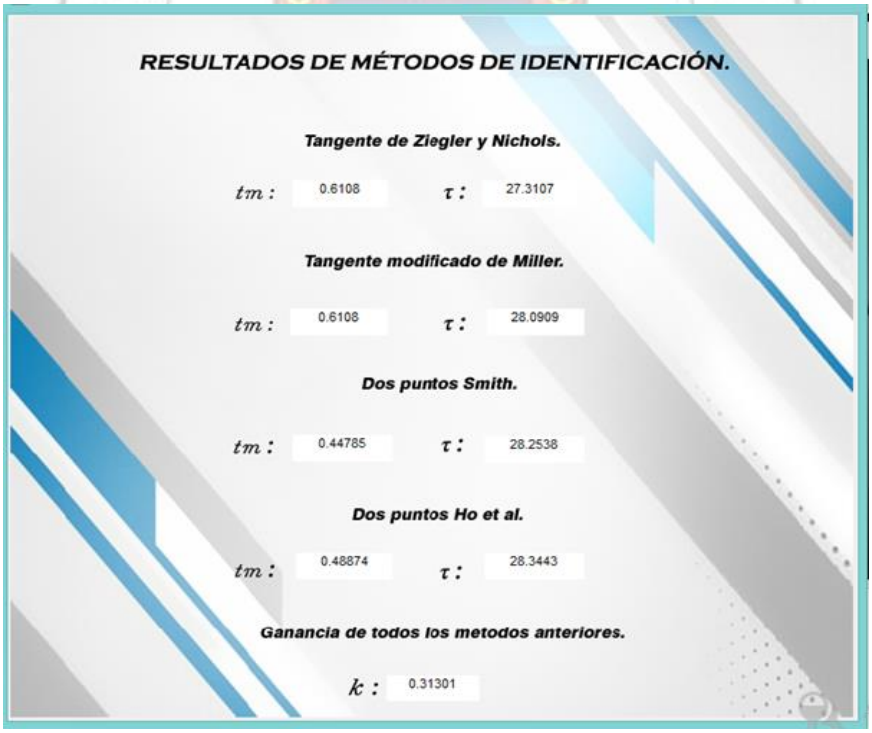
3. Datos y estadísticas figura 6.

Figura 6 Datos y estadísticas

Fuente: Autores

Al dar clic en “Resultados de métodos de identificación.” Se abre la siguiente ventana de la figura 7.

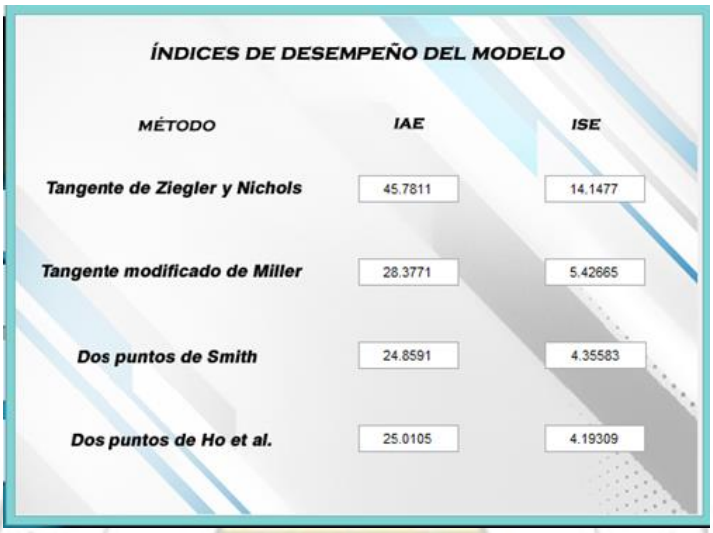
Figura 7 Resultados de identificación



Fuente: Autores

Al dar clic en “Índices de desempeño” Se abre la siguiente ventana, figura 8.

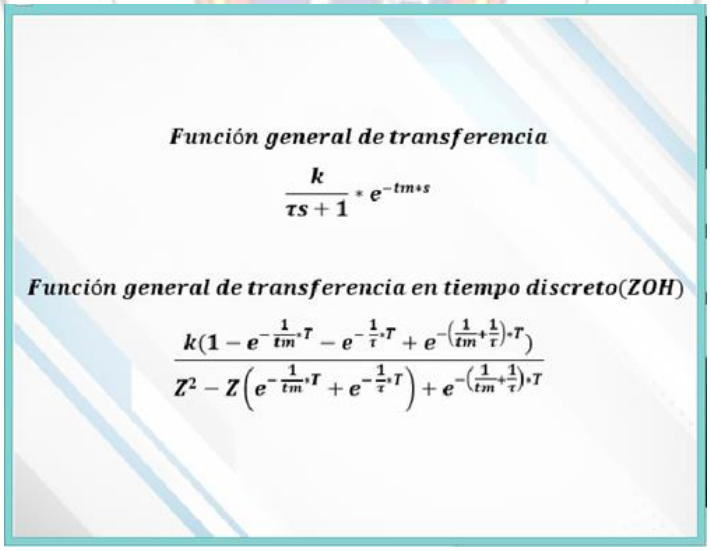
Figura 8 Índices de desempeño



Fuente: Autores

Al dar clic en “Funciones de transferencia” Se abre la siguiente ventana, figura 9.

Figura 9 Función de transferencia



Fuente: Autores

3.4 Sintonización

Al dar clic en “Sintonización” de la ventana principal, se abre el siguiente menú de opciones, figura 10.

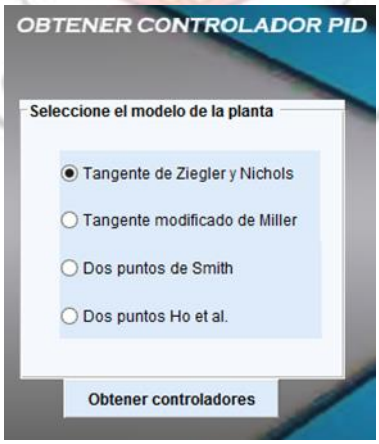
Figura 10 Ventana de sintonización

Fuente: Autores

1. Obtener controlador PID

Se debe escoger el modelo para la planta y dar clic en “Obtener controladores”, figura 11.

Figura 11 Obtener controladores

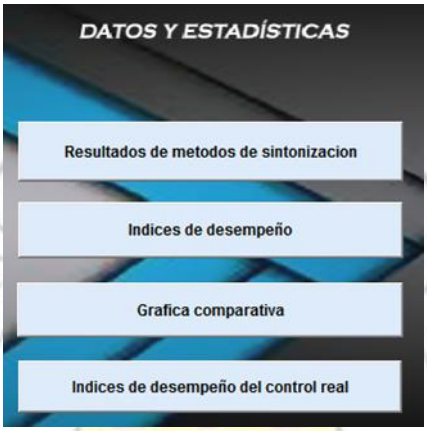


Fuente: Autores

2. Datos y estadísticas.

En datos y estadísticas se encuentran índices de desempeño de los controladores, resultados de los métodos y una gráfica comparativa, figura 12.

Figura 12 Datos y estadísticas



Fuente: Autores

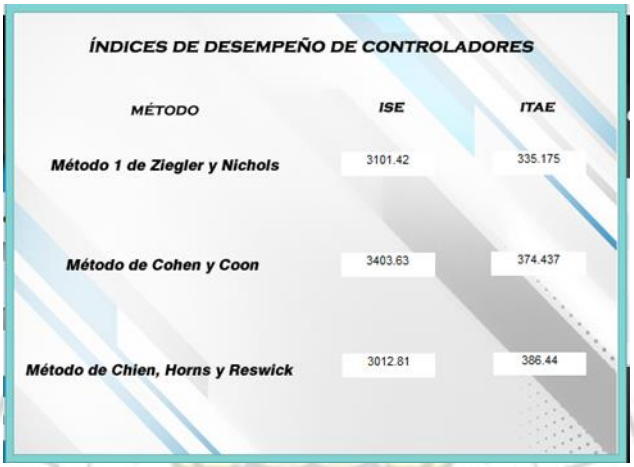
Al dar clic en “Resultados de métodos de sintonización.” Se abre la siguiente ventana, figura 13.

Figura 13 Resultados de sintonización

Fuente: Autores

Al dar clic en “Índices de desempeño.” Se abre la siguiente ventana, figura 14.

Figura 14 Índices de desempeño

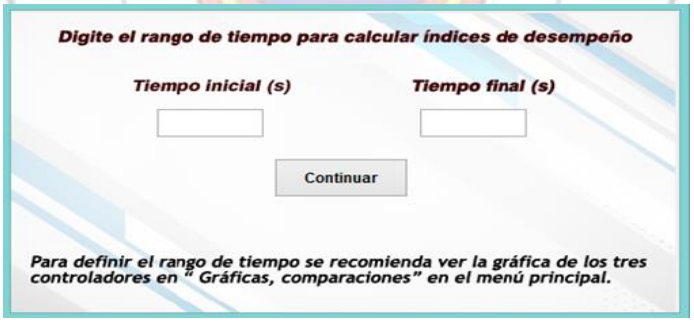


Fuente: Autores

Al dar clic en “Grafica comparativa” se abrirá una gráfica de simulink de los tres métodos de sintonización.

Al dar clic en “Índices de desempeño del control real.” Se abre la siguiente ventana figura 15. En él se debe ingresar el rango de tiempo en segundos para calcular los índices de desempeño.

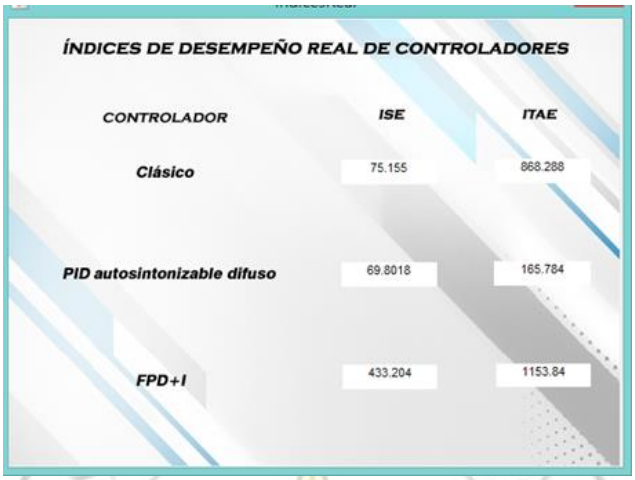
Figura 15 Rango de tiempo para índices



Fuente: Autores

Luego de ingresar los tiempos y dar clic en continuar después de unos segundos se muestran los resultados, figura 16.

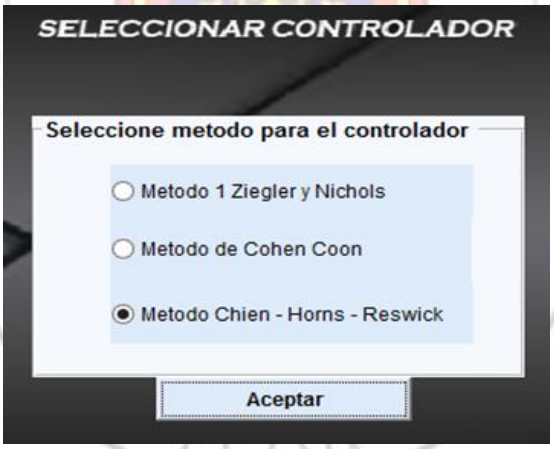
Figura 16 Índices reales



Fuente: Autores

3. En Seleccionar controlador se escoge el mejor método.

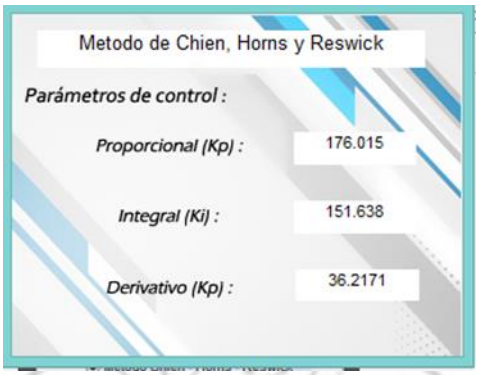
Figura 17 Seleccionar método de controlador



Fuente: Autores

Después de seleccionar el controlador y dar clic en aceptar se muestran los valores calculados para el método de sintonización, figura 18.

Figura 18 Valores de método seleccionado



Fuente: Autores

3.5 Controladores

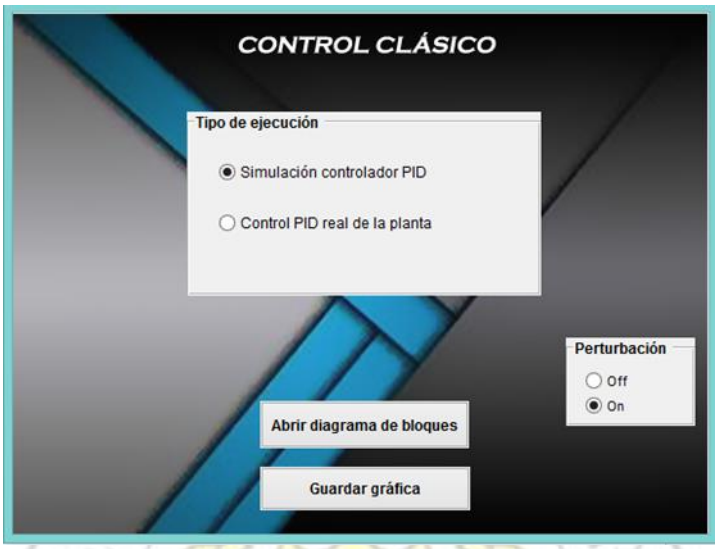
Al dar clic en “Controladores” de la ventana principal, se abre la siguiente ventana, figura 16.

Figura 19 Controladores

Fuente: Autores

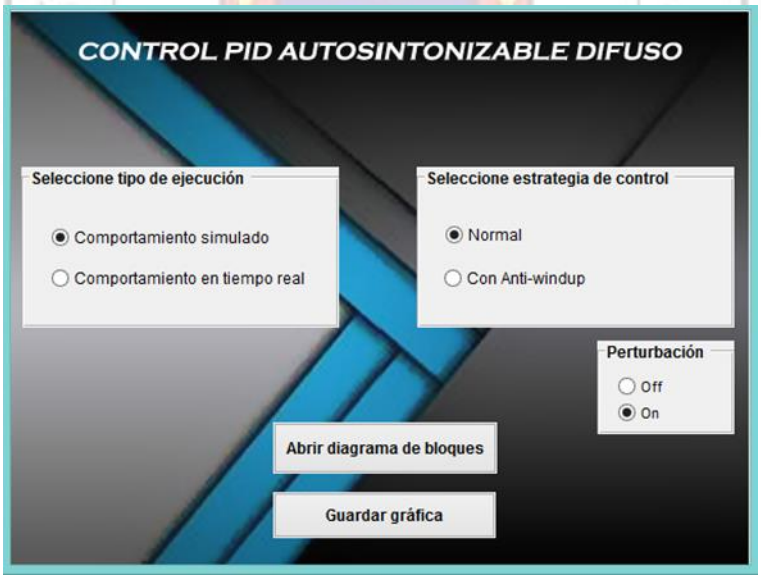
En las ventanas del “control clásico”, “Control PID auto sintonizable difuso” y “ Control FPD+I” de la figura 20, 21 y 22 se encuentra el tipo de ejecución, si es simulación o directamente en la planta y también si desea que existan perturbaciones, después de seleccionar se da clic en “Abrir diagrama de bloques“ y se abrirá simulink con el respectivo controlador seleccionado.

Figura 20 Control clásico



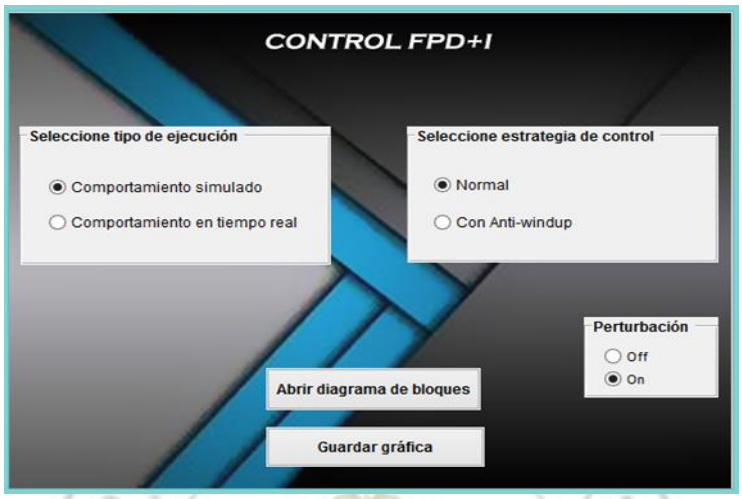
Fuente: Autores

Figura 21 Control auto sintonizable



Fuente: Autores

Figura 22 Control FPD+I

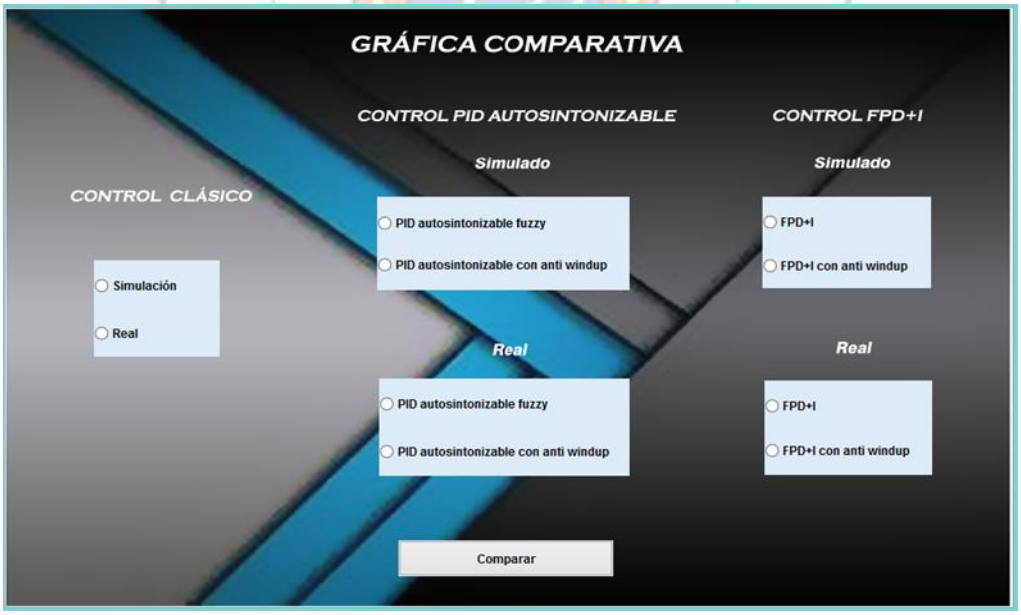


Fuente: Autores

3.6 Gráficas, comparaciones

Al dar clic en “Graficas, comparaciones” de figura 23, se abrirá la siguiente ventana con todos los controladores tanto de simulaciones como los resultados reales en la planta. Se debe seleccionar los controladores que se deseen ver y dar clic en “Comparar” para obtener una gráfica de simulink.

Figura 23 Gráfica comparativa



Fuente: Autores

3.7 Restaurar archivos

Al dar clic en “Restaurar archivos” de la ventana principal, se abrirá la siguiente ventana de confirmación, figura 24.

Figura 24 Restaurar archivos

Fuente: Autores

Al hacer clic en aceptar se restablecerán los archivos de simulink, por lo tanto es de utilidad cuando se ha trabajado y guardado cambios en los diagramas de bloques y se desea restaurar los diagramas; a su vez también se restablece la información guardada en archivos Excel en la cual se guarda la toma de datos, graficas de simulaciones y de control real de la planta, cálculos de identificación y sintonización.

3.8 Posibles problemas

1. Tarda demasiado en abrir simulink.

Por lo general si es la primera vez en abrir simulink desde que inicio la interfaz, es muy probable que tarde un poco en abrir, esto depende del computador que se esté utilizando, lo recomendable en este caso es primero abrir un archivo simulink directamente desde Matlab y no por medio de la interfaz.

2. No se puede sincronizar simulink con arduino, figura 25.

Figura 25 Error de sincronización

Fuente: Autores

Los problemas de sincronización en tiempo real entre arduino y simulink como el que se observa en la figura 25, se solucionan configurando la computadora en alto rendimiento, en el panel de control o en la configuración de ahorro de energía. Si el problema persiste reinicie el equipo.

4 CÓDIGO DEL ARDUINO

Para la comunicación se requiere que en el arduino este grabado el siguiente código y tener instalada la librería Adafruit MLX90614 en el programa de arduino de la computadora.

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614 ();
int pwm = 3;
int pwmrecibido;
int cont=0;
int pert = 31;
int perturbacion;
byte buffer_enviar[6];
byte buffer_recibir[3];
static long temp;
float dato;
int entero;
int decimal;
double lectura;
int entAmb;
int decAmb;
double lectAmb;
int i=1;
unsigned long t1;
```

```
unsigned long t2;
void setup() {
 Serial.begin(9600);
 mlx.begin();
 pinMode(3,OUTPUT);
 pinMode(31,OUTPUT);
 t1=millis();
}
void loop() {
  if(cont==0){
    lectAmb=(mlx.readAmbientTempC());
    entAmb=(int)lectAmb;
    decAmb=(lectAmb-entAmb)*100;
     cont=1;
    }
   if(Serial.available()){
    for(byte i=0;i<=2;i++){
      buffer_recibir[i]=Serial.read();
      }
    if(buffer_recibir[0]==82 && buffer_recibir[2]==10){
      pwmrecibido=buffer_recibir[1];
      analogWrite(pwm,pwmrecibido);
      digitalWrite(pert,LOW);
      }
      t2=millis();
    if(buffer_recibir[0]==83 && buffer_recibir[2]==10){
      pwmrecibido=buffer_recibir[1];
      analogWrite(pwm,pwmrecibido);

      if(t2<=80000){
      digitalWrite(pert,LOW);
      }
       if(t2>80000 && t2<83000){
      digitalWrite(pert,HIGH);
      }
       if(t2>=83000 && t2<95000){
      digitalWrite(pert,LOW);
      }
       if(t2>=95000 && t2<100000){
      digitalWrite(pert,HIGH);
      }
       if(t2>100000){
      digitalWrite(pert,LOW);
      }
      }
    }
     lectura=(mlx.readObjectTempC());
     entero=(int)lectura;
     decimal=(lectura-entero)*100;
    buffer_enviar[0]='E';
    buffer_enviar[1]= entero;
    buffer_enviar[2]= decimal;
    buffer_enviar[3]= entAmb;
    buffer_enviar[4]= decAmb;
```

```
buffer_enviar[5]='\n';
Serial.write(buffer_enviar,6);
delay(249);}
```

ANEXO 5 GUÍA DE APRENDIZAJE

GUÍA DE APRENDIZAJE

1 Objetivo General

- Aprender sobre el control PID auto sintonizable difuso y control FPD+I

2 Objetivos específicos

- Aprender a realizar una mejor identificación de la planta con la ayuda de indicadores de desempeño.
- Aprender a sintonizar teniendo en cuenta indicadores de desempeño y resultados experimentales.
- Comprender la estructura básica de un controlador PID auto sintonizable y controlador FPD+I.
- Discernir sobre el comportamiento de cada controlador.

3 Introducción

En el presente documento se plantea un paso a paso para aprender sobre estos dos controladores, iniciando con la identificación de la planta, teniendo en cuenta algunos índices de desempeño, luego con la sintonización tanto en simulaciones como en el control real en la planta. La sintonización se inicia con el control clásico que será la referencia para el PID auto sintonizable y FPD+I, de acuerdo a los resultados de cada controlador se realizan ajustes experimentales.

4 Recomendaciones

A continuación cada uno de los pasos esta explicado incluyendo fórmulas sin enfocarse en el manejo de la interfaz, por tal motivo es necesario complementar este curso con el manual de uso de InstructControl.

Antes de continuar es importante tener en cuenta que se deben tener conocimientos previos sobre control clásico, difuso y manejo de Matlab Simulink.

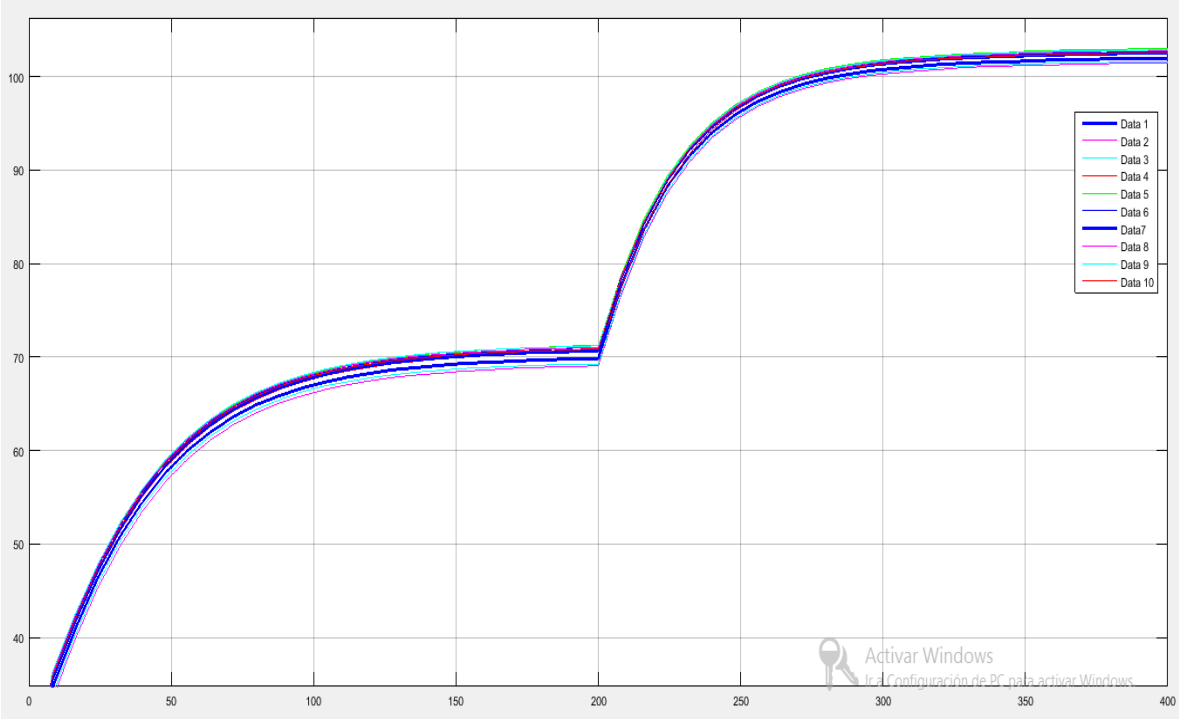
5 GUÍA

5.1 Identificación

5.1.1 Procedimiento

1. Se debe obtener 10 gráficas del comportamiento de la planta a lazo abierto con dos set point y el segundo será el de utilidad para obtener el modelo, como se observa en la figura 1. En este caso el primer set point de 76 PWM y segundo set point de 178 PWM (el 100% del PWM representa 255 PWM en el ardino, dando como resultado 12 v en el elemento final de control).

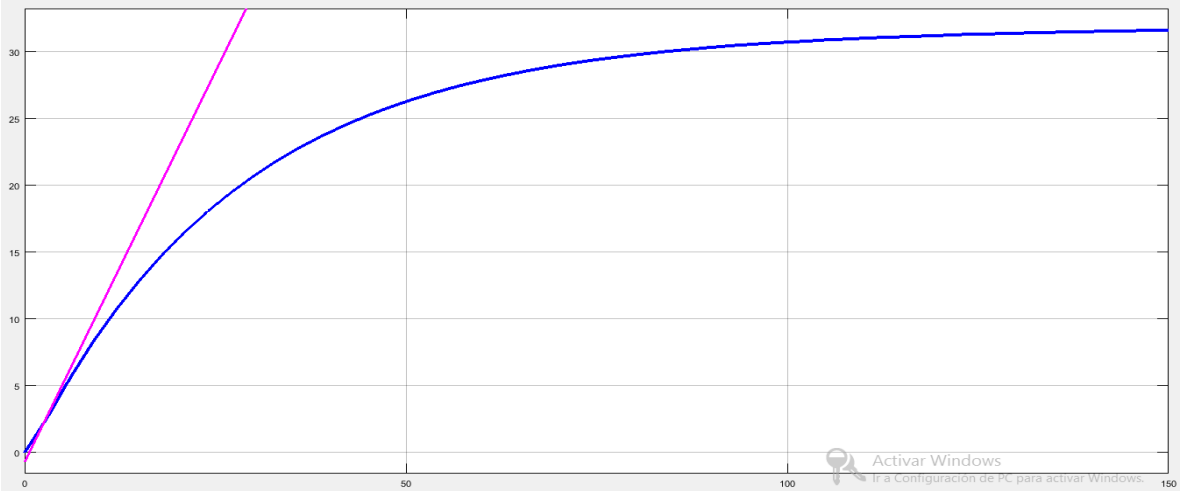
Figura 1 Grafica de toma de datos



Fuente: Autores

2. La gráfica promedio de las 10 tomas de dato se observa en la figura 2.

Figura 2 Grafica promedio



Fuente: Autores

3. Trazar la recta tangente a la curva de reacción.

Para obtener la recta tangente se debe representar el rango de la gráfica donde se encuentra el punto de inflexión con un polinomio mediante la función polyfit en Matlab, `f=polyfit(x,y,3)` siendo `y` la temperatura y `x` el tiempo.

Teniendo en cuenta la ecuación (6.1) de la recta tangente.

$$y = f(x_0) + f'(x_0) * (x - x_0) \qquad (6.1)$$

Se obtiene x_0 despejando x de la segunda derivada del polinomio que representa la gráfica y se reemplaza en la ecuación (6.1), graficando esta ecuación se obtiene la recta tangente como se puede observar en la figura 2.

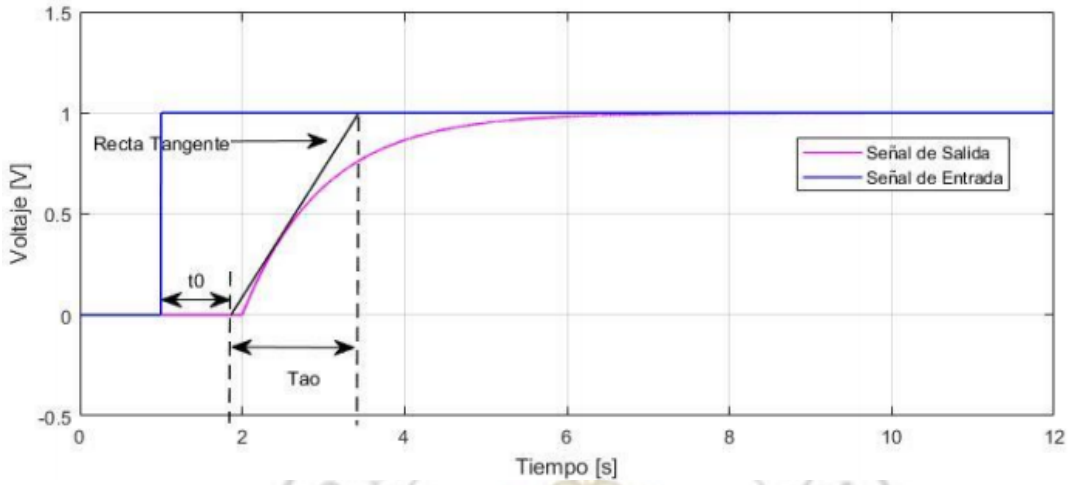
4. El modelo para la planta es de primer orden con tiempo muerto, por lo tanto la función de transferencia la ecuación (6.2).

$$\frac{K}{\tau.s+1}.e^{-t_m.s} \qquad (6.2)$$

5. Método de identificación Tangente de Ziegler y Nichols.

De la gráfica de la figura 3 se obtiene la constante proporcional K, la constante de tiempo Tao y tiempo muerto.

Figura 3 Identificación Ziegler y Nichols

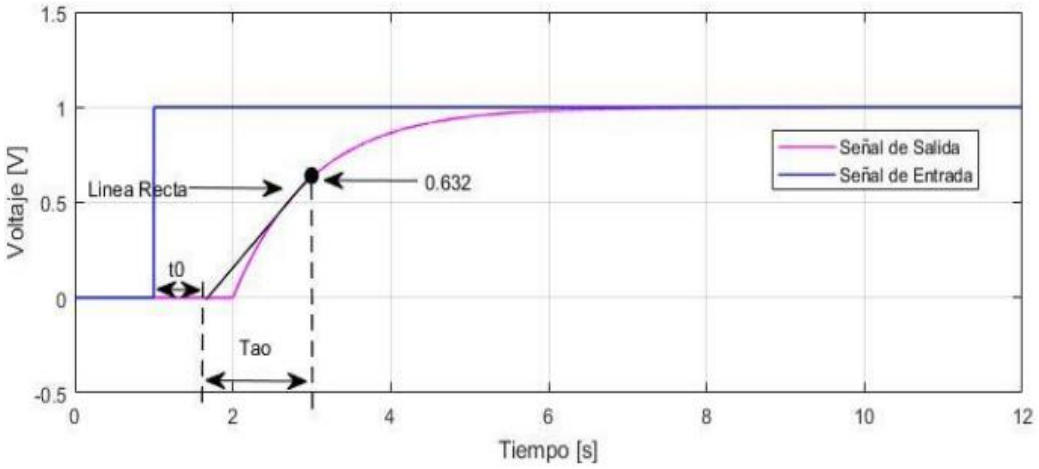


Fuente: Jorge Novoa, Wuemdell Martínez, 2019.

6. Método de identificación Tangente modificado de Miller.

Se diferencia del método anterior en que la constante de tiempo Tao se obtiene restando el tiempo muerto del tiempo cuando se alcanza el 63.2% como se observa en la figura 4.

Figura 4 Método modificado de Miller

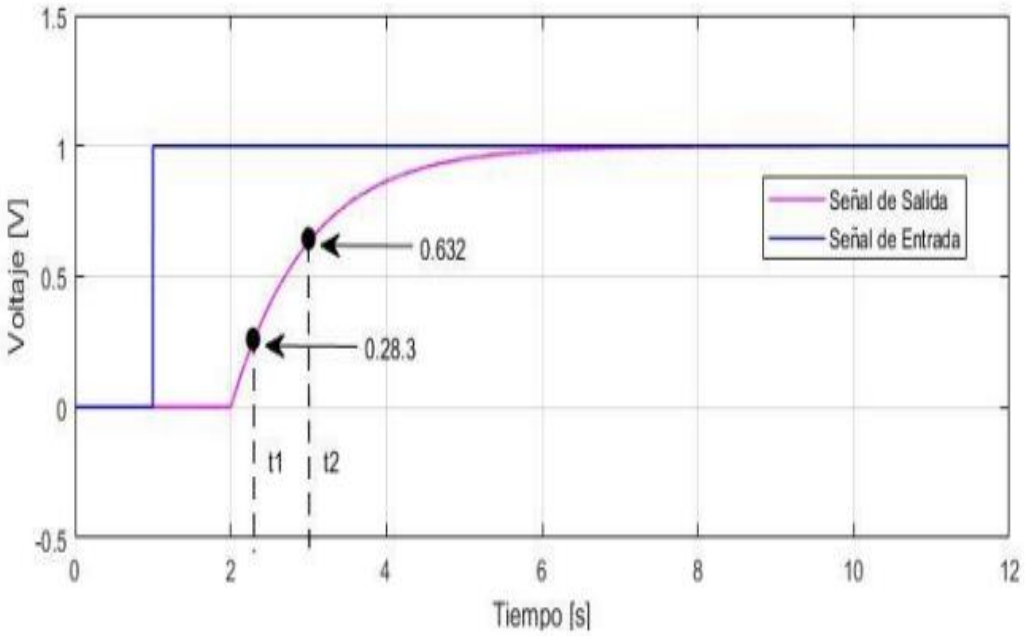


Fuente: Jorge Novoa, Wuemdell Martínez, 2019.

7. Método de identificación dos puntos de Smith.

Se establece un 28.3% para el primer tiempo y un 63.2 para el segundo tiempo según la figura 5 y teniendo en cuenta las ecuaciones (5.1) y (5.2).

Figura 5 Dos puntos de Smith



Fuente: Jorge Novoa, Wuemdell Martínez, 2019.

$$\tau = at_1 + bt_2 \quad (5.1)$$

$$t_m = ct_1 + dt_2 \quad (5.2)$$

$$a = -1.5$$

$$b = 1.5$$

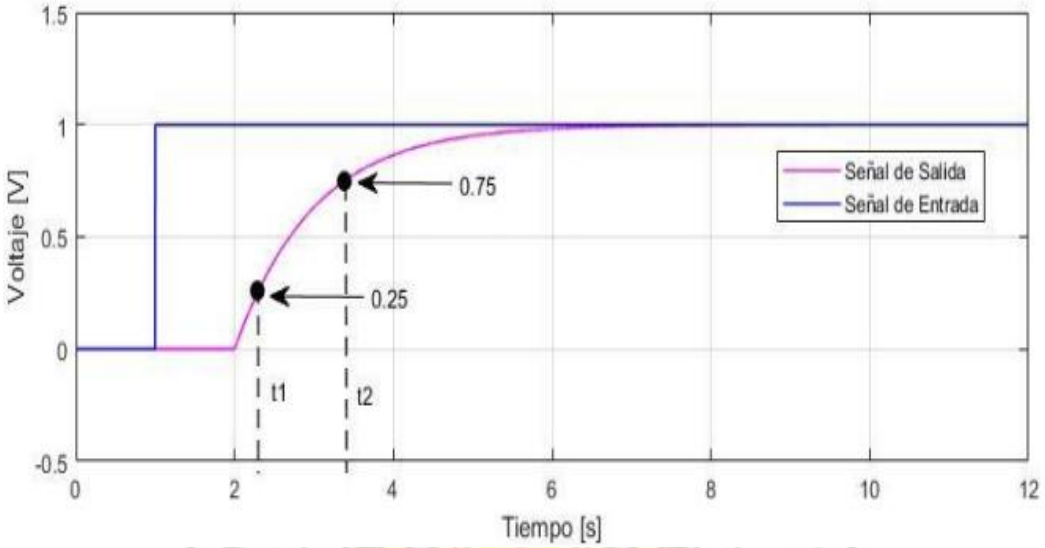
$$c = 1.5$$

$$d = -0.5$$

8. Método de identificación dos puntos de Ho et al.

Se establece un 25% para el primer tiempo y un 75% para el segundo tiempo como en la figura 6 obteniendo el tiempo muerto y tao de las ecuaciones (5.3) y (5.4).

Figura 6 Dos puntos Ho et al.



Fuente: Jorge Novoa, Wuemdell Martínez, 2019.

$$\tau = at_1 + bt_2 \quad (5.3)$$

$$t_m = ct_1 + dt_2 \quad (5.4)$$

$$a = -0.67$$

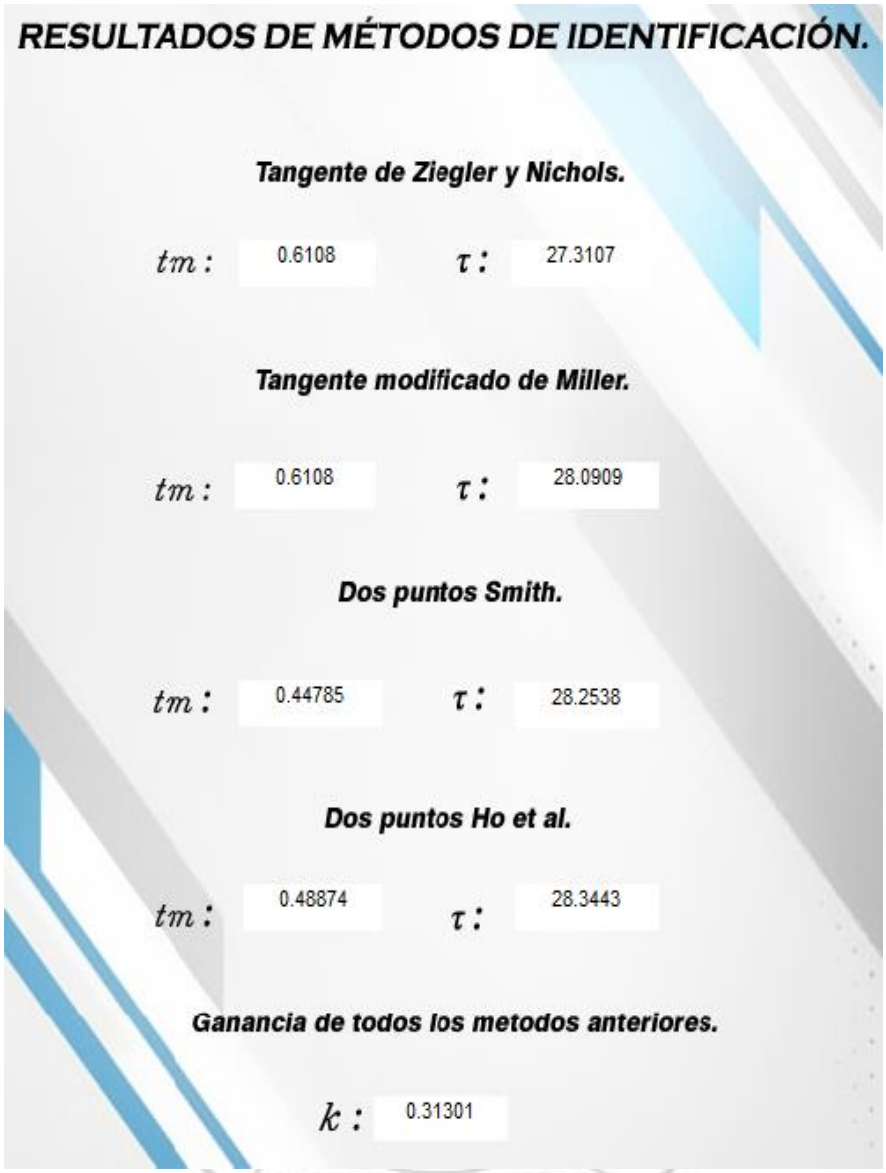
$$b = 0.67$$

$$c = 1.3$$

$$d = -0.29$$

9. Con los 4 métodos anteriores se obtienen los siguientes resultados de la figura 7.

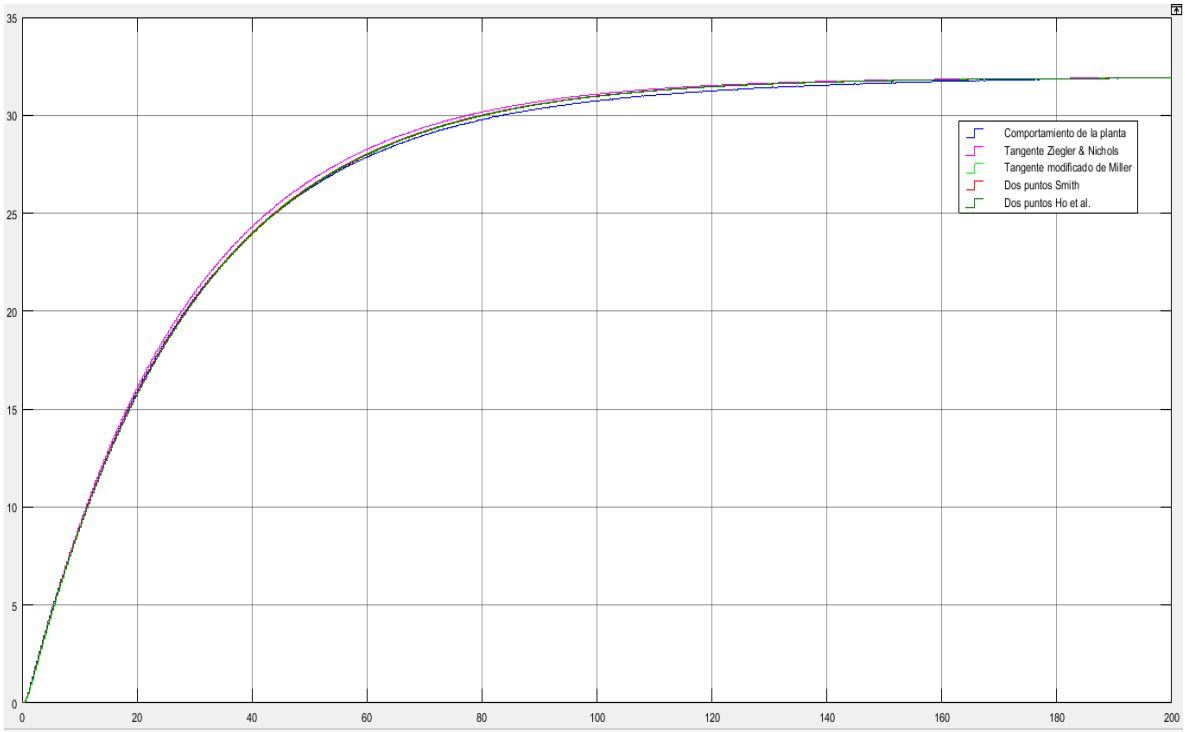
Figura 7 Resultados de métodos de identificación



Fuente: Autores

10. Al graficar los modelos obtenidos se observa en la siguiente grafica de la figura 8 que todos presentan buenos resultados.

Figura 8 Resultados de identificación



Fuente: Autores

11. Selección del mejor modelo teniendo en cuenta los índices de desempeño de la tabla 7.

Tabla 7 Índices de desempeño para el modelo

índice de desempeño	Descripción	Expresión
ISE	Integral square-error criterion (criterio de la integral del error al cuadrado)	$\int_0^T e^2(t)dt$
IAE	Integral Absolute Error Criterion (Criterio de la integral del valor absoluto del error)	$\int_0^T \|e(t)\|dt$

Fuente: Sergio Castaño, 2020.

El índice IAE representa el área diferencial entre la respuesta de la planta y la del modelo, si IAE=0 entonces la respuesta de la planta y la del modelo son iguales.

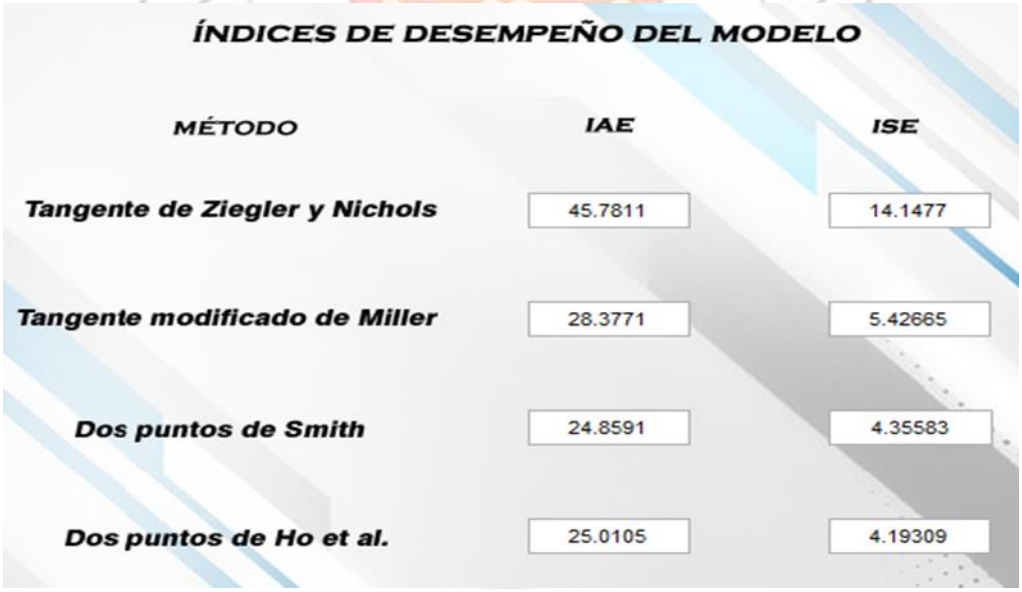
Con el índice de desempeño ISE en este caso se haría notar si el modelo tiene partes en las que el error se incrementa más que en otras. Suponiendo que se tienen dos modelos para representar dicho comportamiento de la planta, y a estos dos modelos se le calcula el índice de desempeño IAE dando como resultado el mismo valor, luego se calcula para estos dos modelos el índice de desempeño ISE dando como resultado valores diferentes, esto indica que el modelo con mayor valor presenta en alguna parte un error más brusco.

Todos los índices utilizados en este documento indican mejor desempeño en cuanto menor sea el valor obtenido.

12. Índices de desempeño

Observando la gráfica anterior de la figura 8 es un poco complicado definir el mejor modelo, por lo tanto nos enfocaremos en los índices de desempeño de la figura 9.

Figura 9 Índices de desempeño del modelo



Fuente: Autores

Los índices de desempeño para el método de Smith y Ho et al presentan los mejores resultados, pero dependiendo de las exigencias para el modelado de la planta se debe seleccionar teniendo en cuenta que aspectos o características serian de mayor prioridad; en este caso se escoge el modelo de Ho et al con el menor ISE.

5.1.2 Sintonización

5.1.2.1 Procedimiento

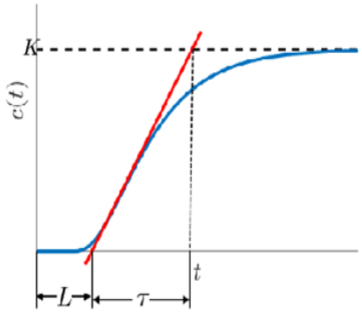
1. El modelo de la planta seleccionado es el de Ho et al y se muestra en la ecuación (5.5).

$$\frac{0.313}{28.34\,s+1}.e^{-0.4887\,s} \qquad (5.5)$$

2. Primer método sintonización de Ziegler y Nichols se muestra en la figura 10.

Figura 10 Sintonización Z&N

Controlador	K_p	τ_i	τ_d
P	$\frac{\tau}{KL}$	∞	0
PI	$0.9\frac{\tau}{KL}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{\tau}{KL}$	$2L$	$0.5L$



Fuente: Sergio Castaño, 2020.

3. Método de sintonización en Lazo Abierto de Cohen y Coon.

Se emplea el mismo test que el método anterior. La sugerencia para los parámetros tiene en cuenta el grado de autorregulación de la planta, mensurado por la relación R = L/τ y se calculan de acuerdo a la figura 11.

Figura 11 Sintonización Cohen y Coon

CONTROLADOR	Kc	T_I	T_D
P	$\frac{1}{KR}\left[1+\frac{1}{3}R\right]$	∞	0
PI	$\frac{1}{KR}\left[0.9+\frac{1}{12}R\right]$	$L\left[\frac{30+3R}{9+20R}\right]$	0
PD	$\frac{1}{KR}\left[\frac{5}{4}+\frac{1}{6}R\right]$	∞	$L\left[\frac{6-2R}{22+3R}\right]$
PID	$\frac{1}{KR}\left[\frac{4}{3}+\frac{1}{4}R\right]$	$L\left[\frac{32+6R}{13+8R}\right]$	$L\left[\frac{4}{11+2R}\right]$

Fuente: Universidad nacional de Tucumán, 2017.

4. Método de sintonización de Chien, Horns y Reswick.

Este método de sintonización propone dos opciones de desempeño para el controlador PID, La respuesta más rápida posible sin sobre impulso y la respuesta más rápida posible con 20% de sobre impulso. En este caso para la planta se selecciona el controlador con la respuesta más rápida posible sin sobre impulso y se calcula según la figura 12.

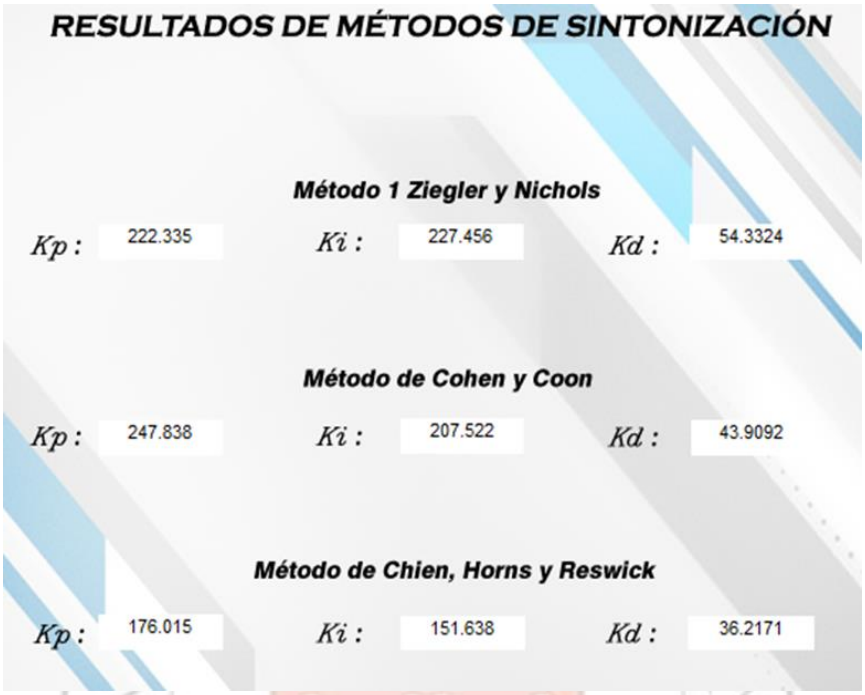
Figura 12 Sintonización CHR

Controlador	K_c	τ_i	τ_d
P	$\frac{0.3\tau}{K\theta}$	—	—
PI	$\frac{0.6\tau}{K\theta}$	4θ	—
PID	$\frac{0.95\tau}{K\theta}$	2.375θ	0.421θ

Fuente: Sergio Castaño, 2020.

5. Resultados de métodos de sintonización, figura 13.

Figura 13 Resultados métodos de sintonización



Fuente: Autores

6. Discretización

La función de transferencia general es la ecuación (5.6).

$$\frac{K}{\tau.s+1}.e^{-t_m.s} \tag{5.6}$$

Para poder incluir el tiempo muerto en la función de transferencia se aplica Taylor, ecuación (5.7).

$$e^{-t_m.s} = \frac{1}{t_m.s+1} \tag{5.7}$$

Dando como resultado la ecuación (5.8)

$$\frac{K}{\tau.s+1} * \frac{1}{t_m.s+1} \tag{5.8}$$

Siendo T el tiempo de muestreo, se aplica transformada Z y retenedor de orden cero (ZOH) a las dos fracciones anteriores por separado para facilitar los cálculos como se muestra en la ecuación (5.9).

$$\frac{K-K.e^{-\frac{1}{\tau}*T}}{z-e^{-\frac{1}{\tau}*T}} * \frac{1-e^{-\frac{1}{tm}*T}}{z-e^{-\frac{1}{tm}*T}} \tag{5.9}$$

Obteniendo como resultado la siguiente función de transferencia general en tiempo discreto en la ecuación (5.10)

$$\frac{K(1-e^{-\frac{1}{tm}*T}-e^{-\frac{1}{\tau}*T}+e^{-\left(\frac{1}{tm}+\frac{1}{\tau}\right)*T})}{z^2-z(e^{-\frac{1}{tm}*T}+e^{-\frac{1}{\tau}*T})+e^{-\left(\frac{1}{tm}+\frac{1}{\tau}\right)*T})} \tag{5.10}$$

Para el controlador PID, la parte integradora está dada por:

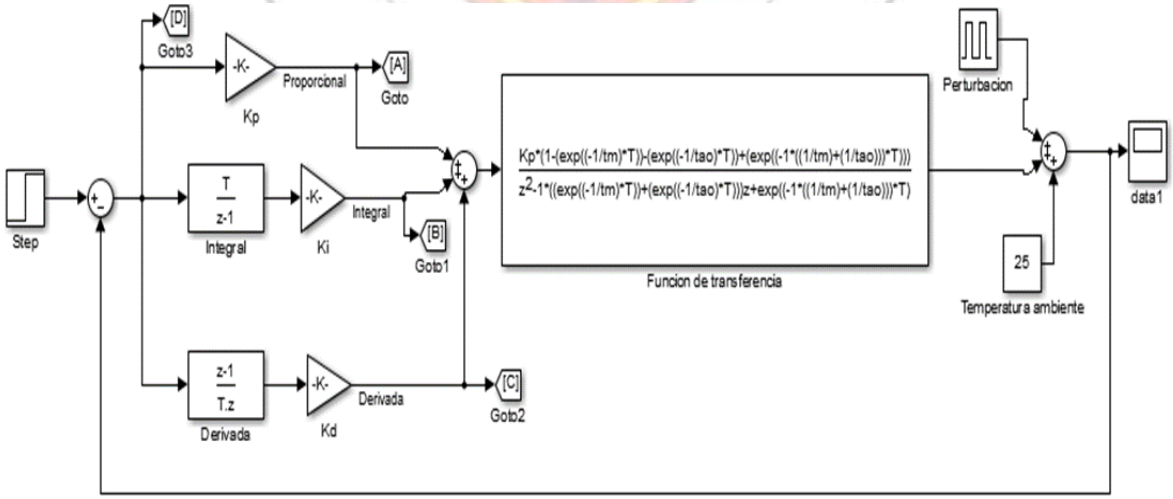
$$\frac{T}{z-1}$$

La parte derivativa sugerida por Matlab viene dada por:

$$\frac{Z-1}{T.z}$$

7. Ingresando la función de transferencia y el controlador en tiempo discreto obtenemos el siguiente diagrama de bloques para el controlador PID de la figura 14.

Figura 14 Diagrama de bloques PID



Fuente: Autores

8. Selección del mejor modelo de sintonización teniendo en cuenta los índices de desempeño de la tabla 8.

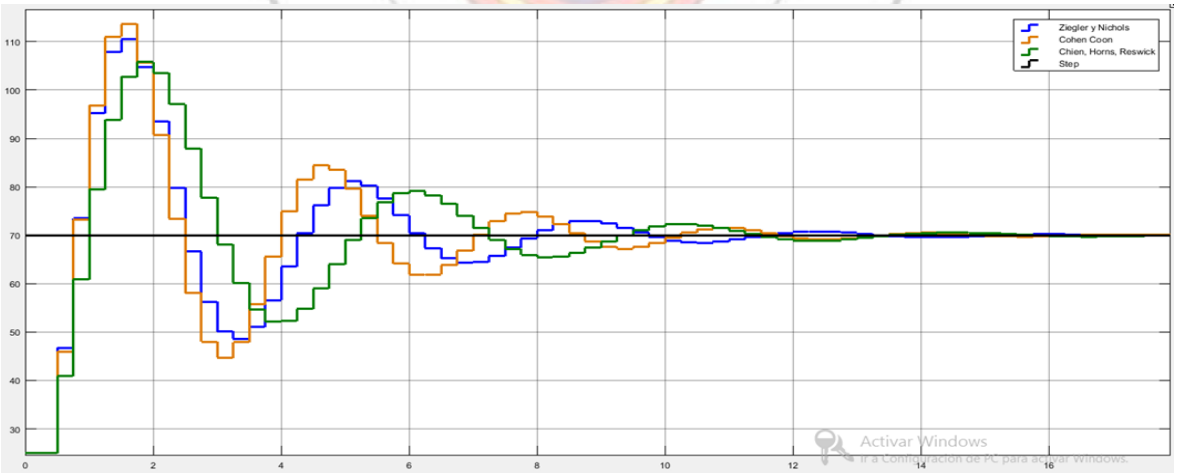
Tabla 8 Índices de desempeño para sintonización

índice de desempeño	Descripción	Expresión
ISE	Integral square-error criterion (criterio de la integral del error al cuadrado)	$\int_0^T e^2(t)dt$
ITAE	Integral of time multiplied Absolute Error Criterion (Criterio de la integral del valor absoluto del error multiplicado por el tiempo)	$\int_0^T t\|e(t)\|dt$

Fuente: Sergio Castaño, 2020.

El índice de desempeño ISE discrimina entre sistemas excesivamente sobre amortiguados y excesivamente subamortiguados, en este caso para nuestra planta resaltara los controladores con mayor sobre impulso siendo el más alto el de Cohen Coon como se muestra en la siguiente figura 15.

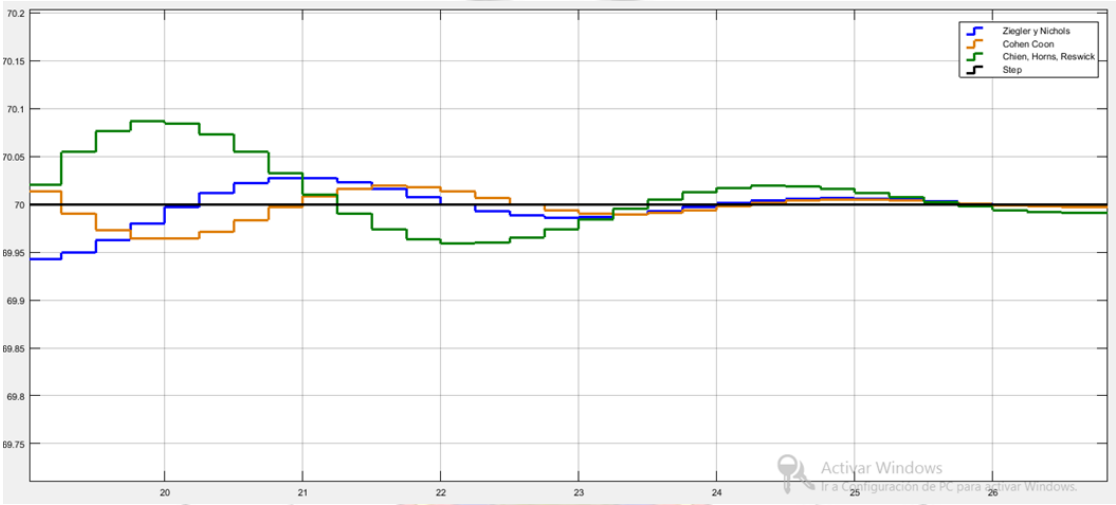
Figura 15 Grafica de métodos de sintonización



Fuente: Autores

En el índice de desempeño ITAE el error inicial es poco penalizado, pero entre más tiempo trascurra se va penalizando cada vez con mayor intensidad, por lo tanto el controlador que mayor ITAE presente, sería el de mayor error antes de estabilizarse o el que mayor tiempo necesite para estabilizarse, en este caso sería el método de Chien, Horns y Reswick en color verde el que más tiempo necesita para estabilizarse como se muestra en la figura 16.

Figura 16 Grafica para analizar con indicadores



Fuente: Autores

Teniendo en cuenta todas las indicaciones anteriores y como prioridad el ITAE, el mejor controlador según los índices de desempeño es el de Ziegler y Nichols, pero escoger el mejor controlador depende de las exigencias o las especificaciones más importantes para el sistema, es decir si nuestro sistema necesitara del controlador con menor pico en el primer sobreimpulso sin importar lo demás, entonces el más indicado sería el de Chien, Horns y Reswick.

Para esta planta se escoge el controlador de Ziegler y Nichols y sus indicadores de desempeño se muestran en la figura 17.

Figura 17 Índices de desempeño de controladores

Fuente: Autores

9. Ajustes para mejorar el desempeño del controlador seleccionado.

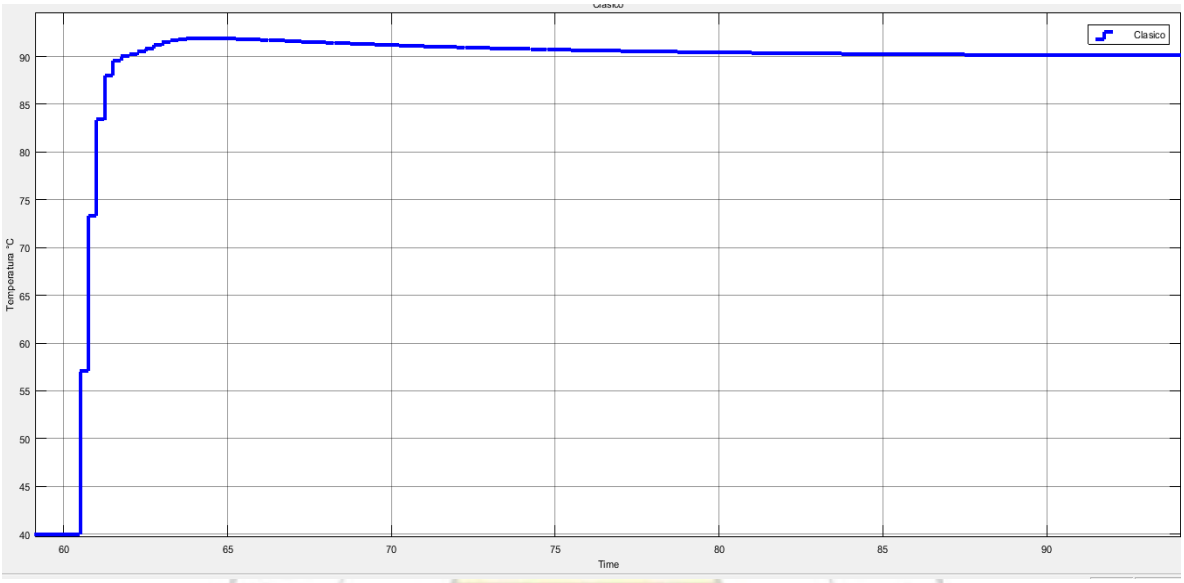
Cambiando los valores para las constantes del controlador anterior de Ziegler y Nichols de acuerdo a pruebas experimentales con los resultados de la tabla 9 con el fin de mejorar la respuesta con un sobre impulso menor al 5%, se obtiene Kp= 110, Ki= 10, Kd=50, y su comportamiento se observa en la figura 18.

Tabla 9 Ajustes experimentales

Constantes	Valor	Sobre impulso (%)	Tiempo de establecimiento (s)
Kp1	222.33	190	15
Ki1	227.45	190	15
Kd1	54.33	190	15
Kp2	120	30	11
Ki2	50	30	11
Kd2	40	30	11
Kp3	110	4	18
Ki3	10	4	18
Kd3	50	4	18

Fuente: Autores

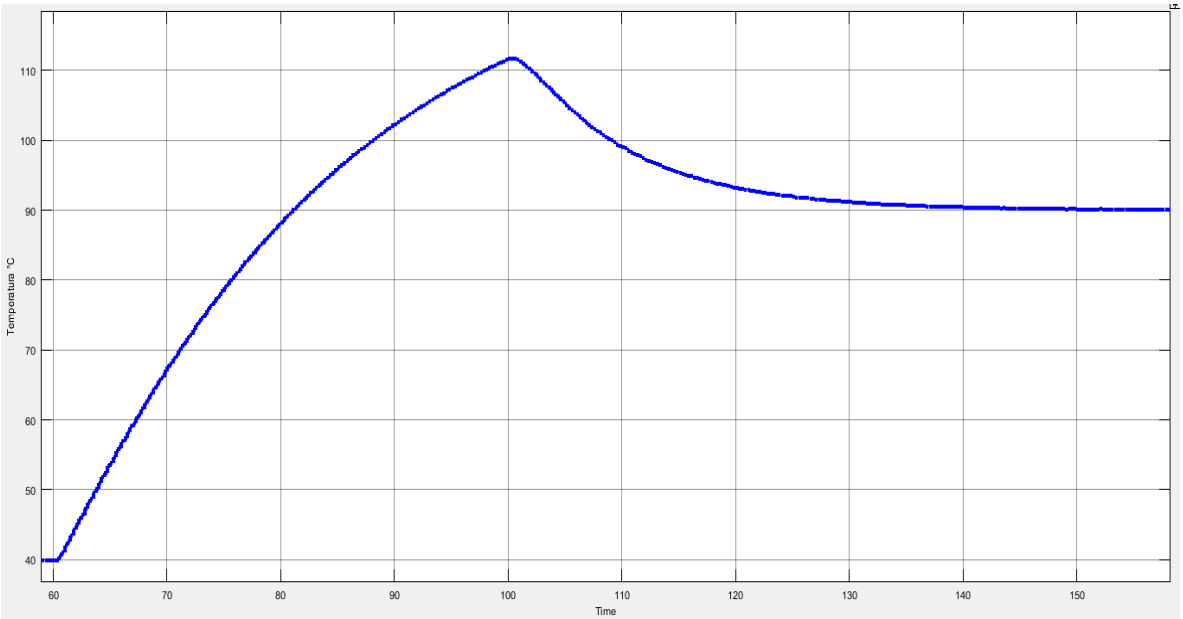
Figura 18 Simulación PID ajustado



Fuente: Autores

10. Al aplicar el controlador anterior en la planta real, se obtiene el siguiente comportamiento de la figura 19, para un set point de 90 °C.

Figura 19 Grafica comportamiento PID real



Fuente: Autores

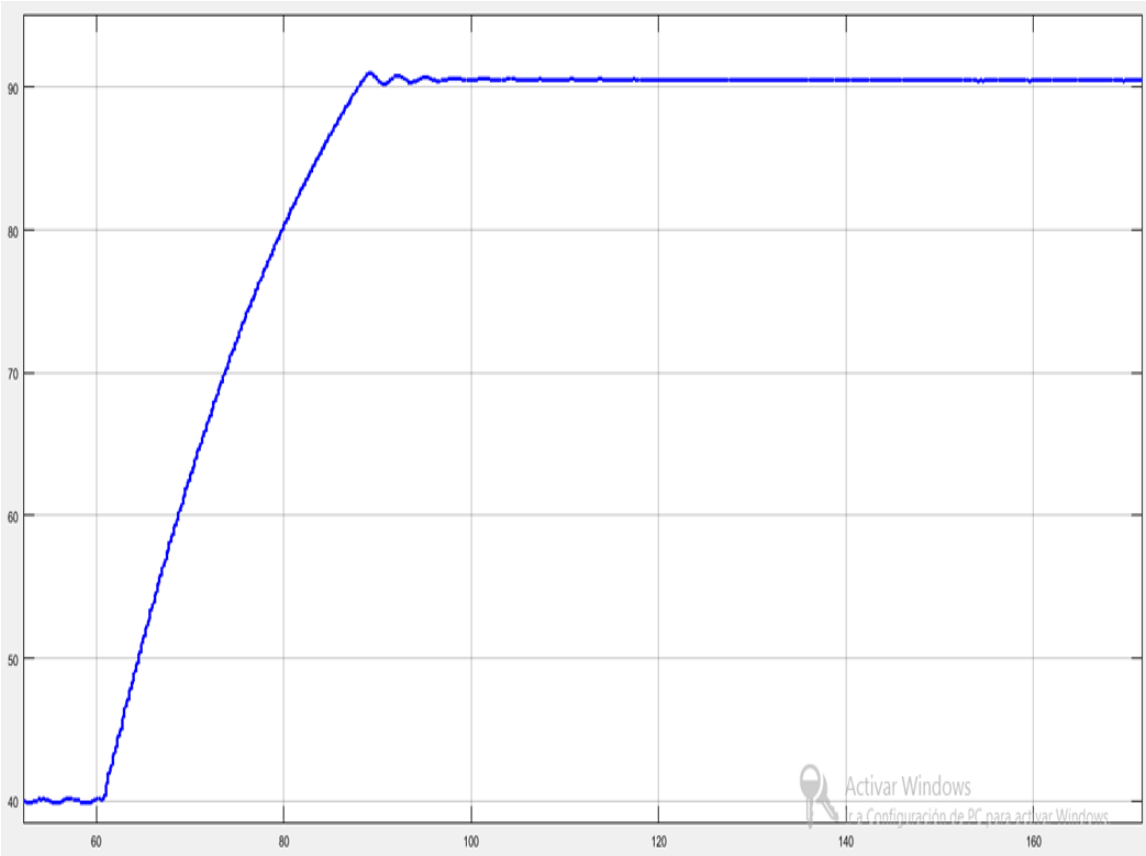
11. En la figura 19, se puede observar que el controlador real presenta demasiado sobre impulso, y mayor tiempo de establecimiento, por lo tanto se requiere de ajuste y se registra en la tabla 10. De manera experimental se obtuvo Kp=200, Ki=0.4, Kd=3. Dando como resultado el comportamiento de la figura 20, para un set ponit de 90 °C.

Tabla 10 Ajuste experimental PID real

Constantes	Valor	Sobre impulso (%)	Tiempo de establecimiento (s)
Kp1	110	43	80
Ki1	10	43	80
Kd1	50	43	80
Kp2	150	26	85
Ki2	5	26	85
Kd2	10	26	85
Kp3	200	1.2	42
Ki3	0.4	1.2	42
Kd3	3	1.2	42

Fuente: Autores

Figura 20 Comportamiento PID real ajustado

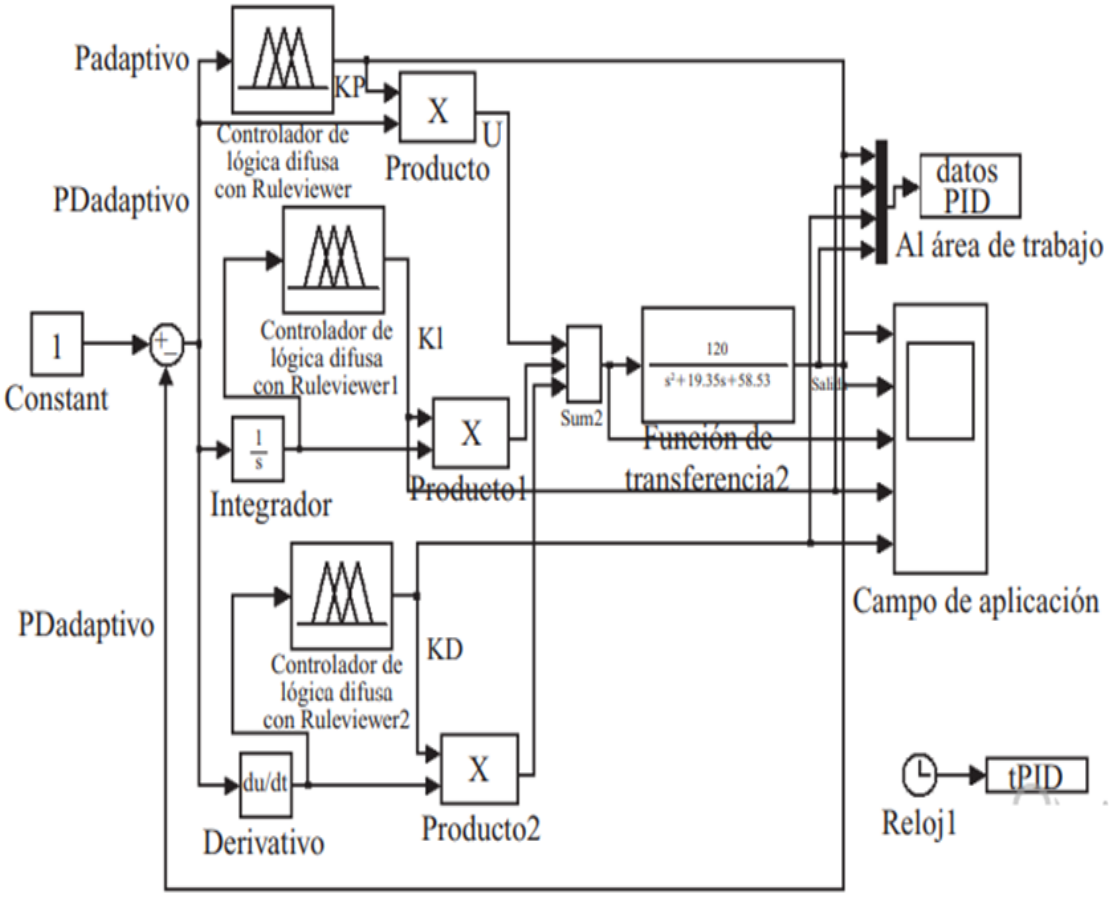


Fuente: Autores

5.1.3 Controlador PID auto sintonizable por lógica difusa.

Esta estrategia plasmada en la figura 21 consiste en hacer que las constantes Kp, Ki y Kd varíen de acuerdo al error, integral del error y derivada del error respectivamente, con el fin de que cada acción tenga un comportamiento más adecuado de acuerdo a la dinámica del sistema.

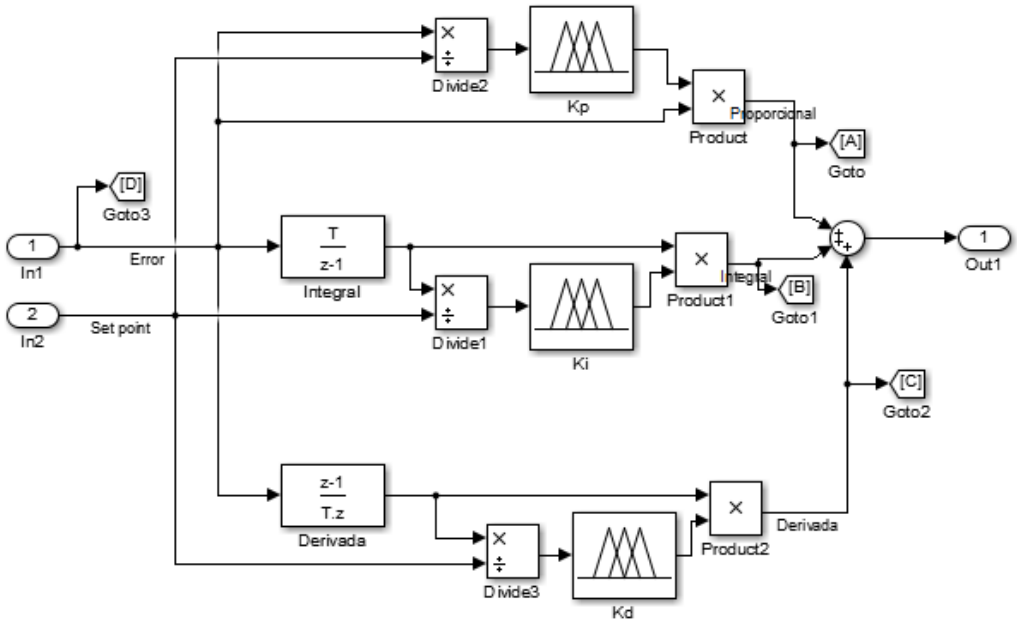
Figura 21 Diagrama de controlador PID auto sintonizable



Fuente: Pedro Ponce. 2010.

Al emplear controladores difusos por lo general se requiere de un polinomio para hacer que logre alcanzar cualquier set point, en este caso como se muestra en la figura 22, en lugar de ese polinomio para la parte proporcional se divide el error entre el set point, y este resultado sería la entrada para el bloque fuzzy proporcional; de igual manera para la parte integral se divide la integral del error sobre el set point y se divide la derivada del error sobre el set point para la entrada del bloque derivativo fuzzy.

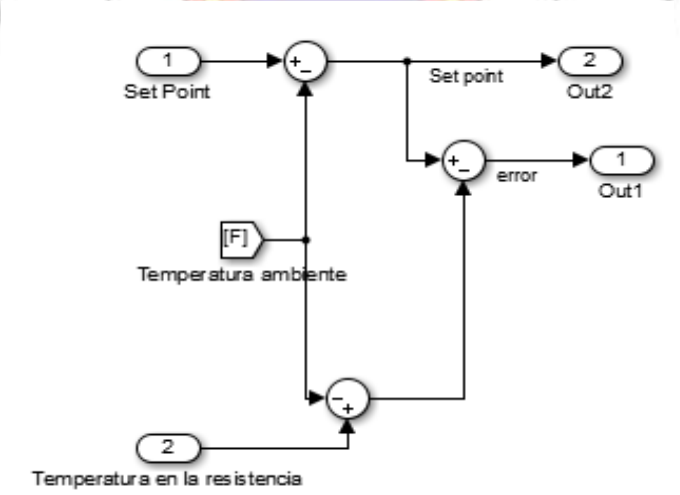
Figura 22 Ajuste de estrada PID auto sintonizable



Fuente: Autores

Se debe tener en cuenta el ajuste de la figura 23, debido a la temperatura ambiente, y es necesario para que funcione correctamente esta estrategia de control.

Figura 23 Ajuste por temperatura ambiente

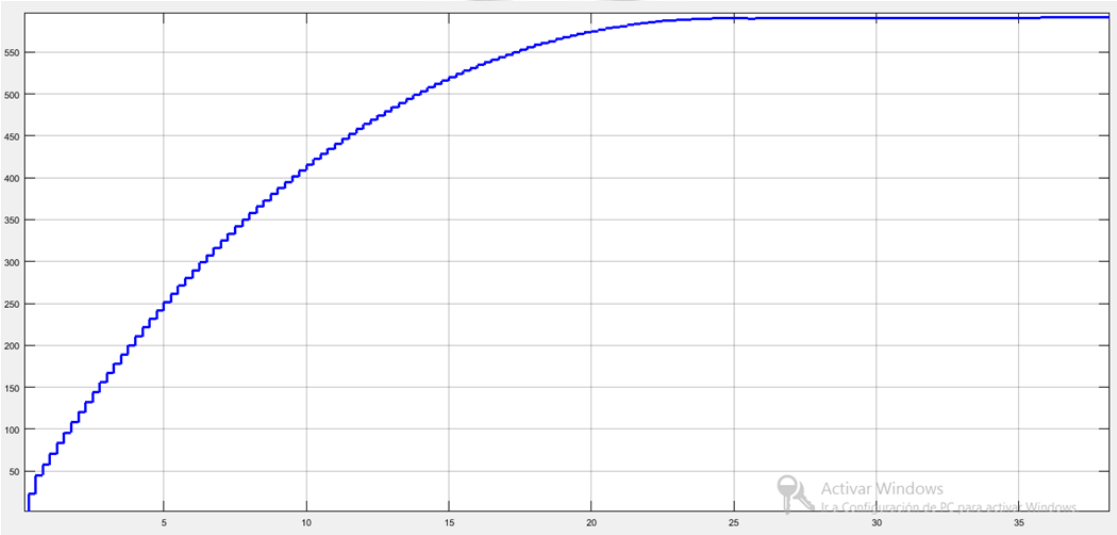


Fuente: Autores

5.1.3.1 Procedimiento para obtener controlador PID auto sintonizable por lógica difusa.

1. Para poder diseñar el controlador PID auto sintonizable por lógica difusa se debe observar mediante un Scope los valores máximos de la integral del error y la derivada del error.

Figura 24 Grafica de integral del error



Fuente: Autores

2. Rango de entrada fuzzy proporcional, integral y derivativa.

En la figura 24 se puede observar que el valor máximo para la integral del error es de 590, por lo tanto el rango para la integral seria desde -590 hasta 590. El rango para la derivada del error sería de -155 hasta 155.

Las entradas para cada uno de los bloques fuzzy se obtiene dividiendo el rango del error, integral del error y derivada del error entre el set point para el controlador, teniendo en cuenta que el punto inicial es la temperatura ambiente, es decir:

Set point para el controlador = 90°C – temperatura ambiente.

Dando como resultados los siguientes rangos para las entradas de cada bloque fuzzy:

Fuzzy proporcional: desde -1 hasta 1

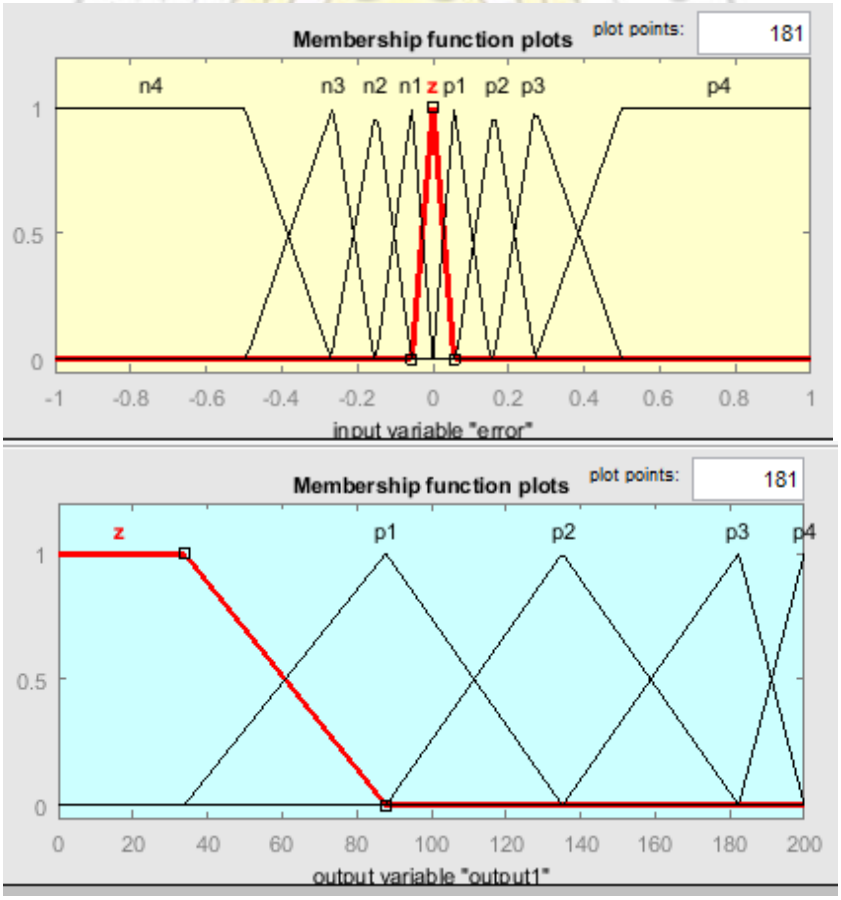
Fuzzy Integral: desde -9.5 hasta 9.5

Fuzzy derivativo: desde -1.77 hasta 1.77

3. Para las entradas de cada bloque fuzzy se toman los rangos anteriores, siendo el caso del bloque proporcional de la figura 30, el rango de la entrada es de -1 hasta 1. En el caso de los rangos de entrada del fuzzy integral y derivativo, estos son de referencia, y lo recomendable es utilizar un rango más amplio.

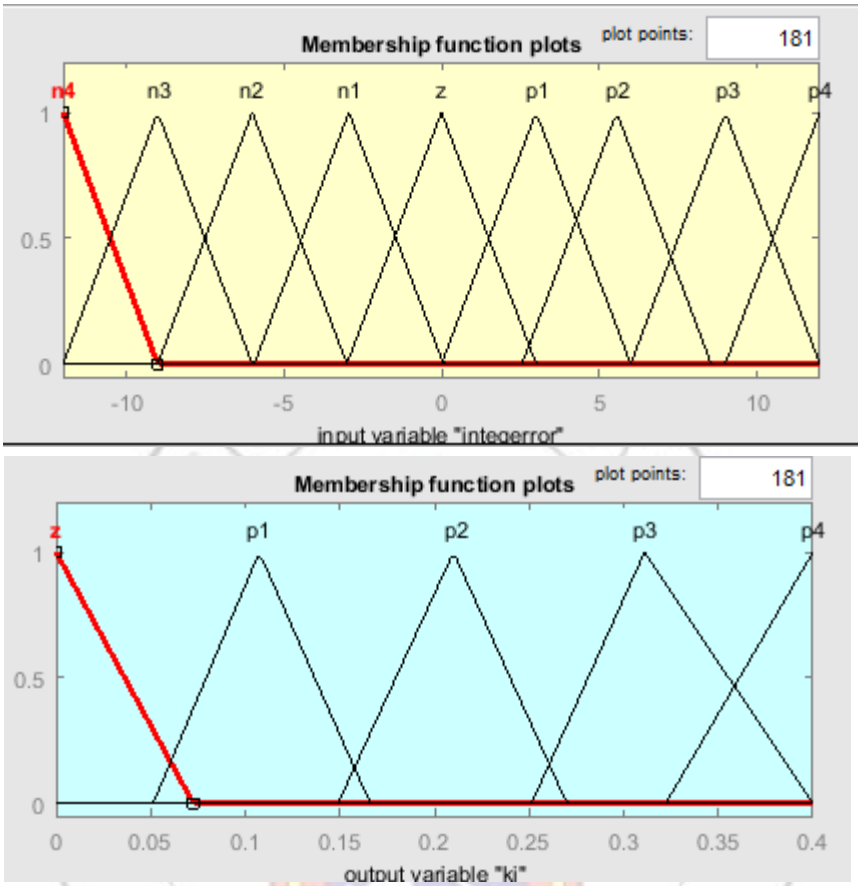
4. Antes de obtener un buen resultado como el de la figura 34 se debió realizar un ajuste experimental en las funciones de membresía de la figura 25, 26,27 y 28, de las entradas y salidas de las funciones de membresía de la parte proporcional, integral y derivativa, debido a que la respuesta de la figura 29 no muestra los resultados esperados.

Figura 25 Funciones de membresía de entrada y salida del error



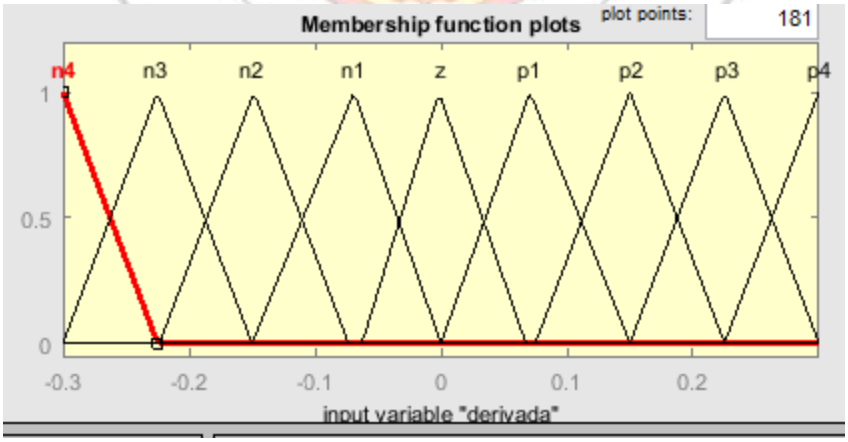
Fuente: Autores

Figura 26 Funciones de membresía de entrada y salida de la integral



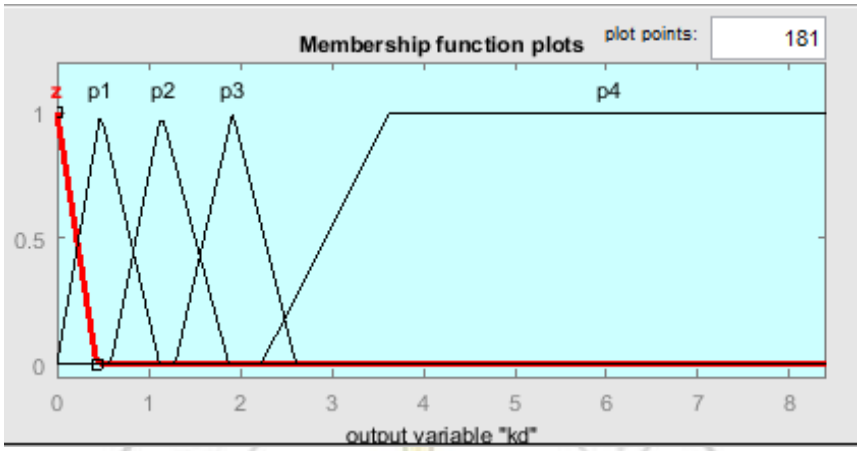
Fuente: Autores

Figura 27 Funciones de membresía de entrada de la derivada



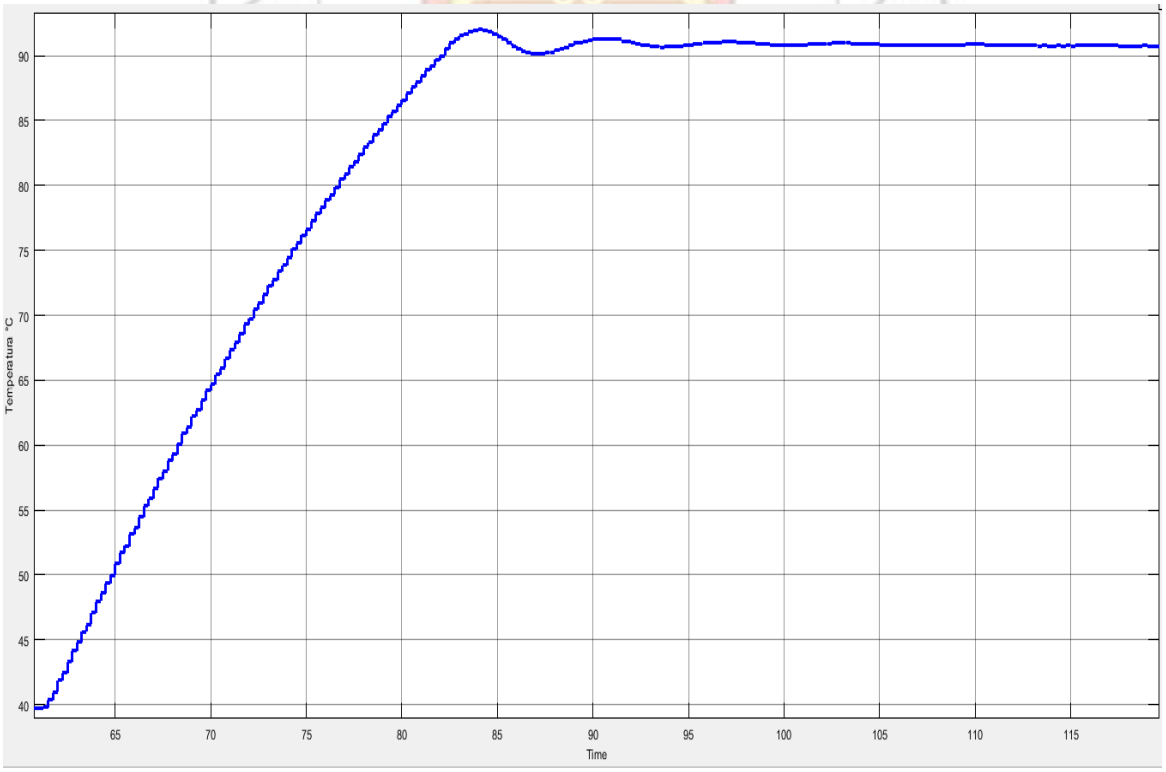
Fuente: Autores

Figura 28 Funciones de membresía de salida de la derivada



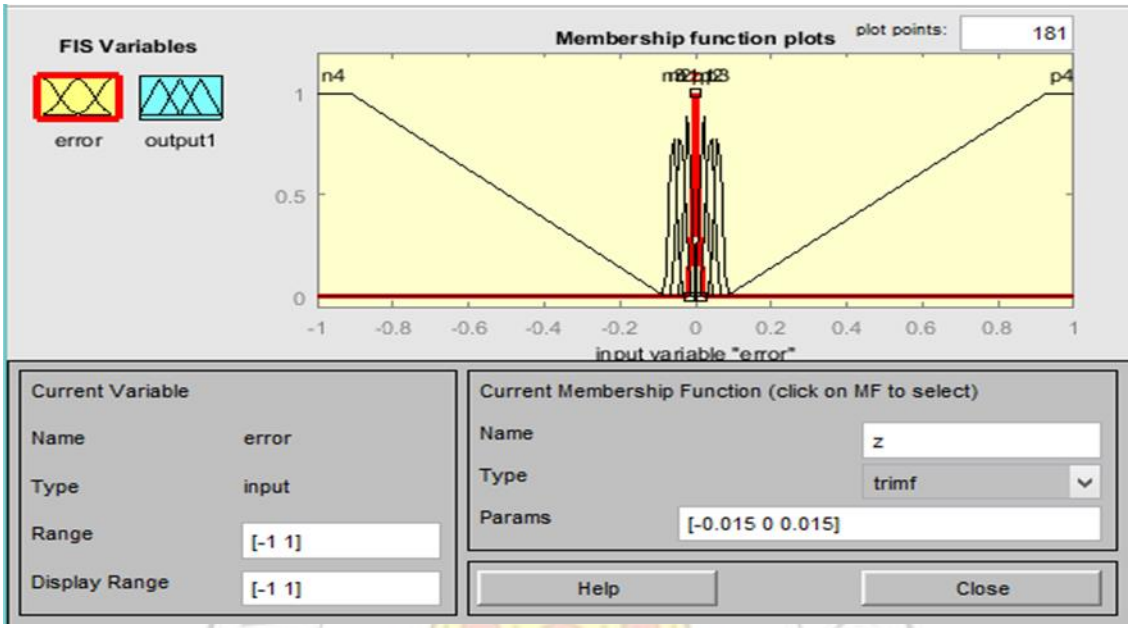
Fuente: Autores

Figura 29 Grafica PID auto sintonizable



Fuente: Autores

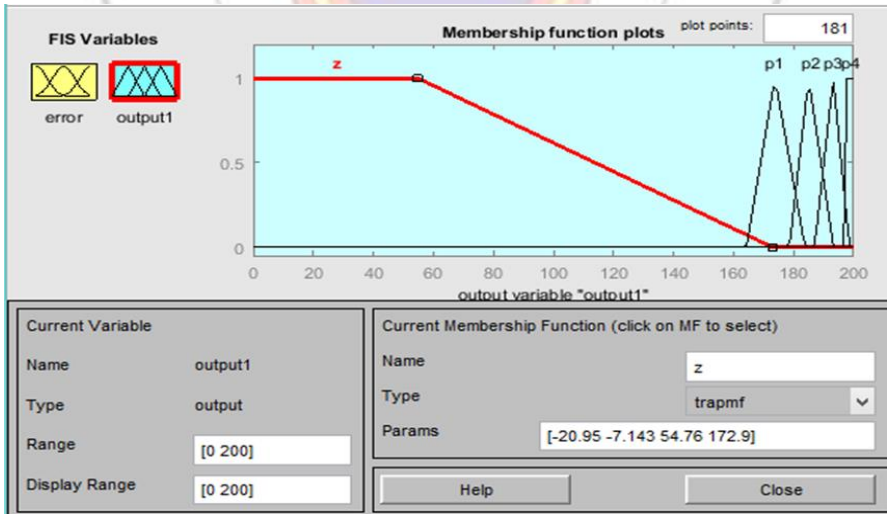
Figura 30 Bloque fuzzy proporcional, entrada del error



Fuente: Autores

Para las salidas de cada bloque fuzzy se toman los mismos valores de Kp, Ki y Kd del controlador clásico real y se ingresan todos los valores en los bloques fuzzy como en el caso del bloque proporcional de la figura 31, el rango de la salida es 0 hasta 200.

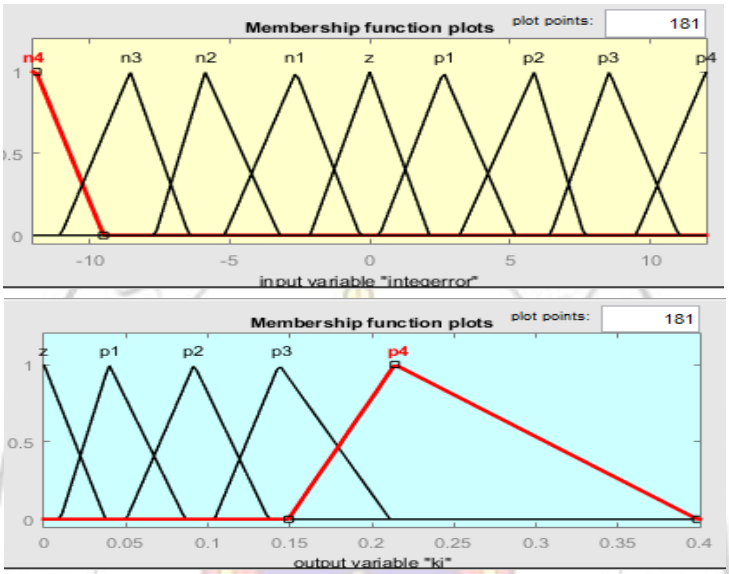
Figura 31 Bloque fuzzy proporcional, salida del error



Fuente: Autores

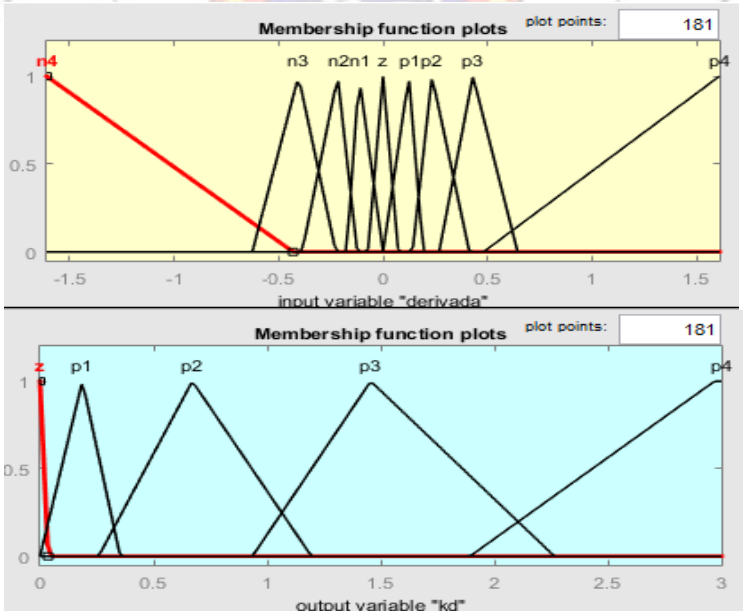
1. Para obtener el controlador PID auto sintonizable definitivo se establecieron las funciones de membresía de entrada y salida para los bloques proporcional, integral y derivativo de las figuras 30, 31, 32 y 33 respectivamente.

Figura 32 Funciones de membresía de entrada y salida integral



Fuente: Autores

Figura 33 Funciones de membresía de entrada y salida derivativa



Fuente: Autores

2. En la tabla 11 se definen las reglas difusas para poder definir a cada entrada su respectiva salida.

Tabla 11 Pertenencia de funciones de membresía

ENTRADA FUZZY Kp	SALIDA FUZZY Kp	ACCION DE CONTROL PROPORCIONAL
n4	p4	Valor negativo
n3	p3	Valor negativo
n2	p2	Valor negativo
n1	p1	Valor negativo
z	z	Z (positivo o negativo)
p1	p1	Valor positivo
p2	p2	Valor positivo
p3	p3	Valor positivo
p4	p4	Valor positivo

Fuente: Autores

3. El comportamiento del controlador PID auto sintonizable por lógica difusa de la figura 34, da como resultado un comportamiento similar al controlador PID clásico. Es importante saber que para obtener el siguiente resultado es necesario ajustar las funciones de membresía que han sido ajustadas de manera experimental, teniendo en cuenta las siguientes características básicas de cada acción de control.

Acción proporcional:

- Disminuye el tiempo de respuesta del sistema.
- En sistemas de segundo orden o superior puede hacer la respuesta más oscilatoria e incluso inestable.

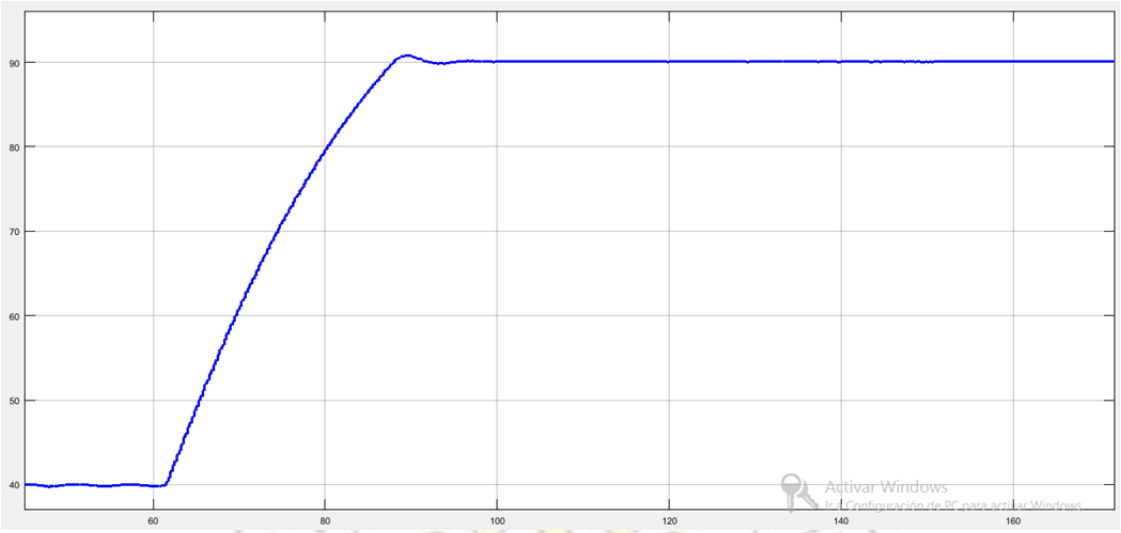
Acción integral:

- Valores muy altos aumenta el sobre impulso y el tiempo de establecimiento del sistema haciéndolo más oscilatorio.
- Elimina el error estacionario.

Acción derivativa:

- Reduce el tiempo de establecimiento y el sobre impulso

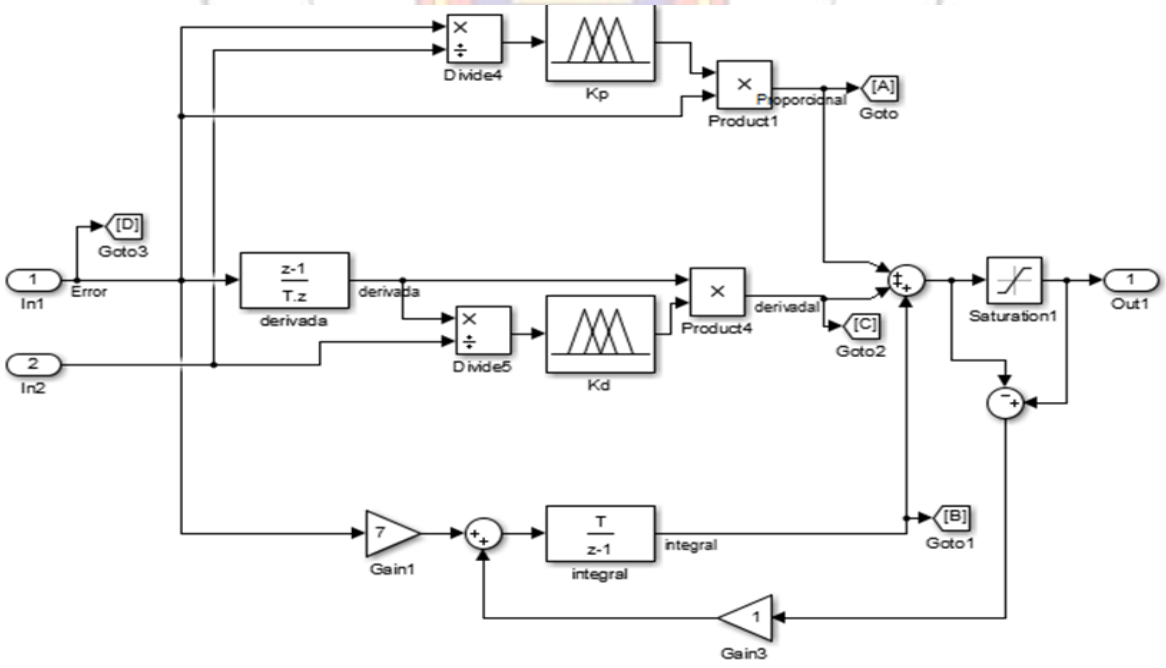
Figura 34 Comportamiento de PID auto sintonizable real



Fuente: Autores

4. Reemplazamos del controlador anterior la parte integral auto sintonizable, por el conjunto anti-windup como se observa en la figura 35.

Figura 35 Diagrama PID auto sintonizable con anti-windup



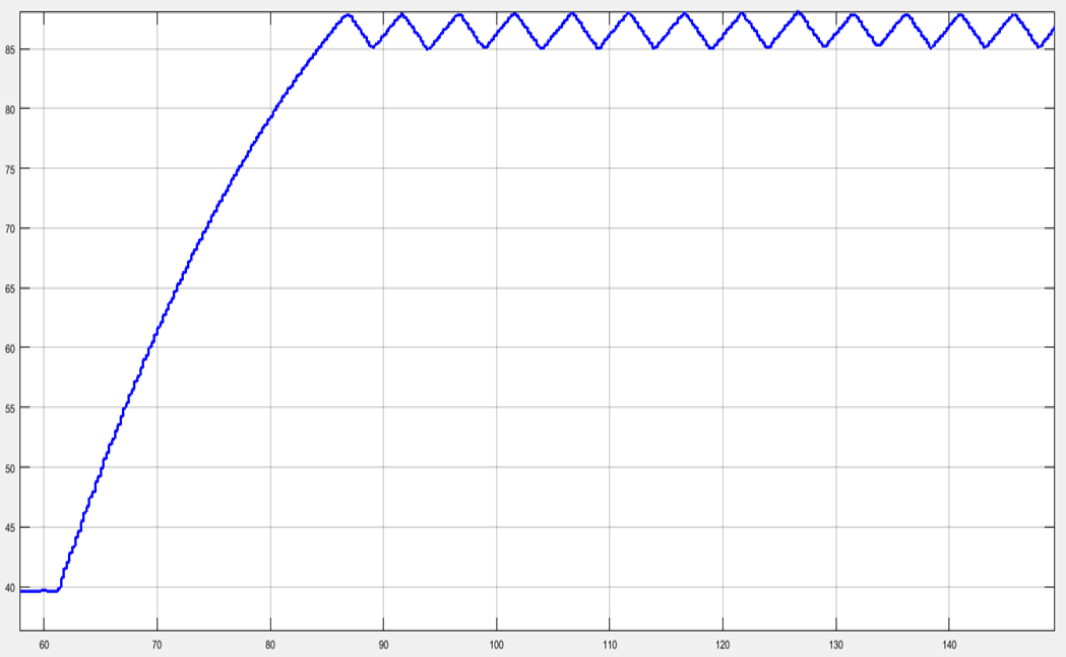
Fuente: Autores

Para desarrollar el control con anti windup, se necesita hallar Tt mediante la ecuación (5.11).

$$Tt = \sqrt{td * ti} = 0.00475 \quad (5.11)$$

Con el valor de Ki= 0.4 obtenemos la respuesta de la figura 36.

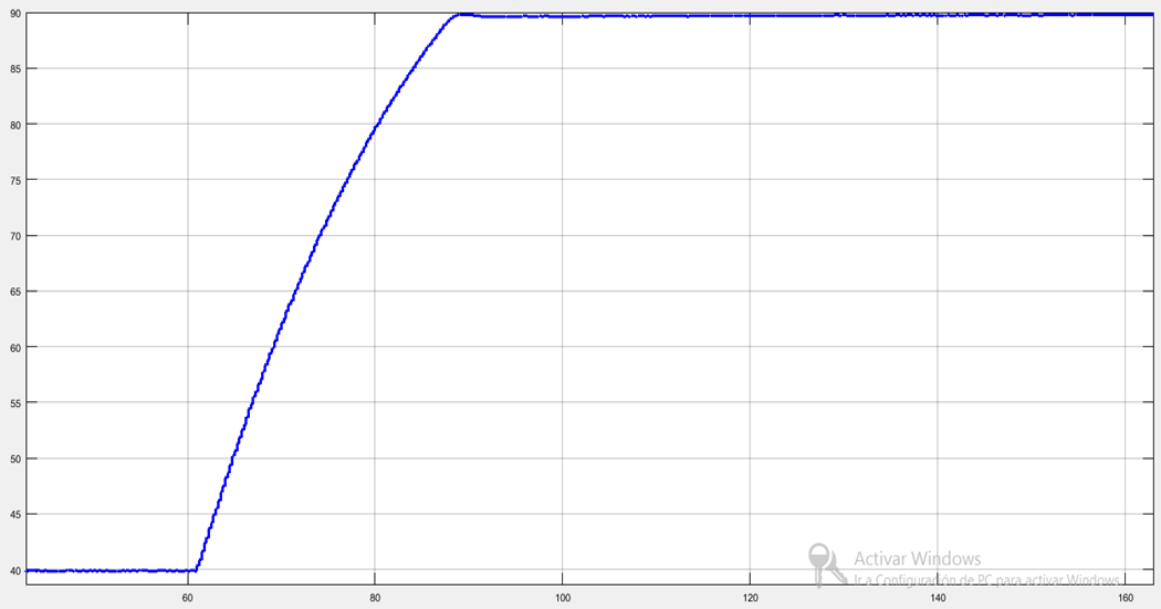
Figura 36 Respuesta PID auto sintonizable con anti-windup sin ajuste



Fuente: Autores

5. En este caso al utilizar el valor anterior de Tt y Ki, no se obtienen los resultados esperados, por lo tanto se realiza un ajuste experimental dando como resultado Tt=1 y Ki=7, con el comportamiento de la figura 37.

Figura 37 Respuesta PID auto sintonizable anti-windup con ajuste

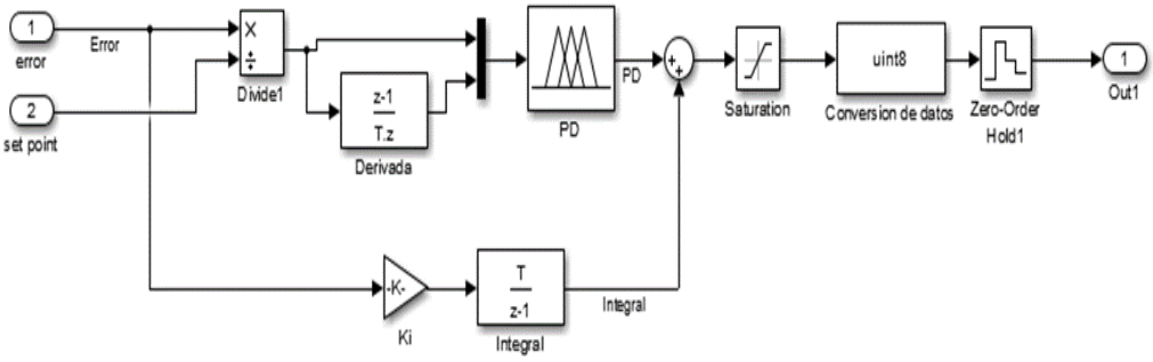


Fuente: Autores

5.1.4 Controlador FPD+I

El control FPD+I consta de un bloque difuso para la parte proporcional y derivativa, y se suma la acción integral como se ilustra en la figura 38.

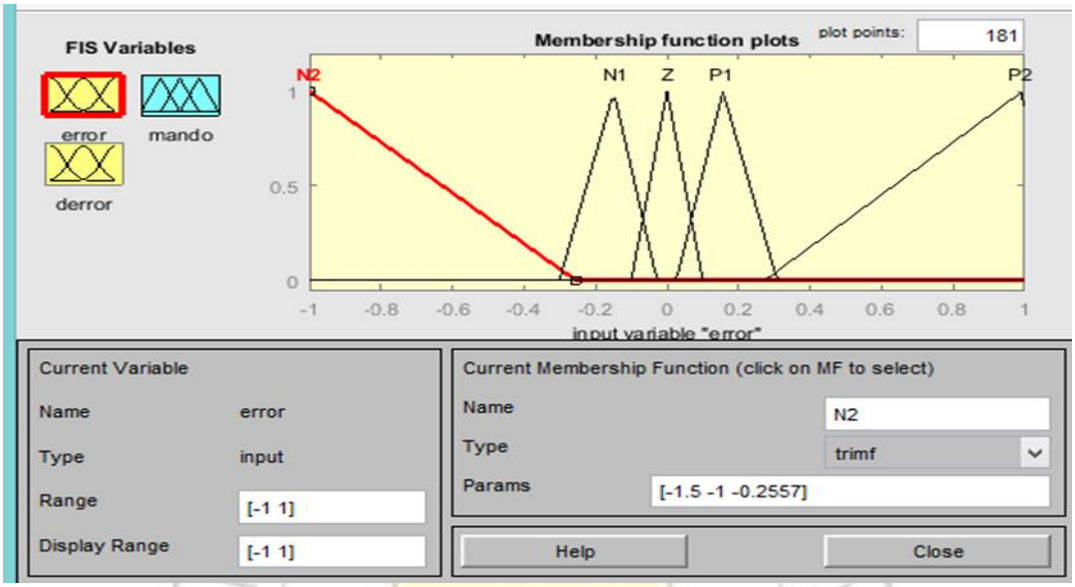
Figura 38 Controlador FPD+I



Fuente: Autores

1. Se ajustan las funciones de membresía entrada del error, derivada del error y mando.

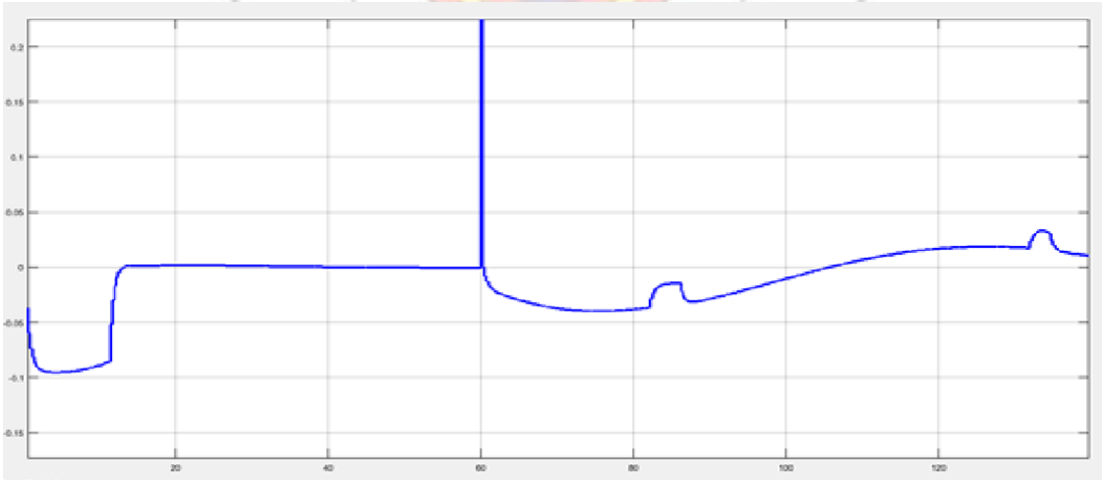
Figura 39 Funciones de membresía entrada



Fuente: Autores

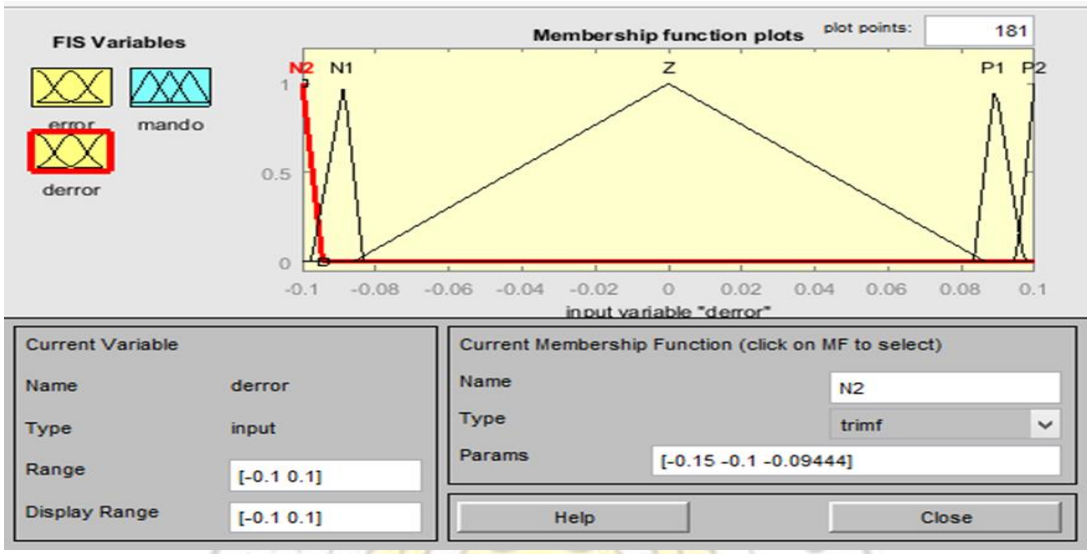
Para definir la entrada de derivada del error se observa la gráfica en la figura 40, de la derivada del error del controlador PID real para definir el rango de la entrada y ajustar las funciones de membresía como en la figura 41.

Figura 40 Derivada del error del PID



Fuente: Autores

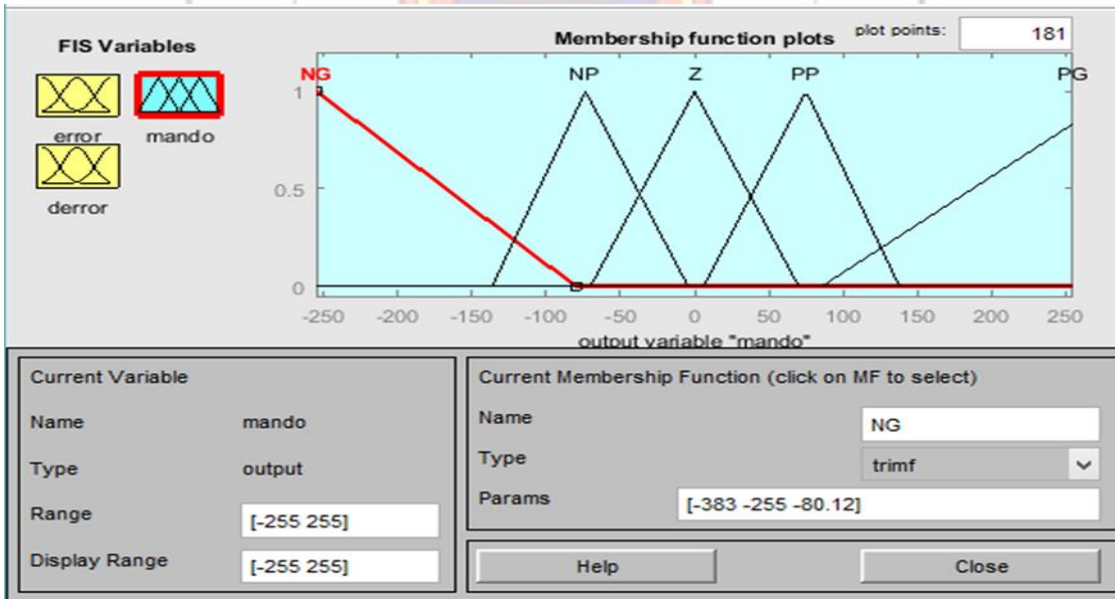
Figura 41 Entrada fuzzy derivada del error



Fuente: Autores

Se establecen las funciones de membresía para el mando o salida fuzzy PD en la figura 42.

Figura 42 Salida fuzzy PD



Fuente: Autores

2. Se define las reglas difusas que determinan de acuerdo a las entradas su respectiva salida o acción de control PD en la tabla 12.

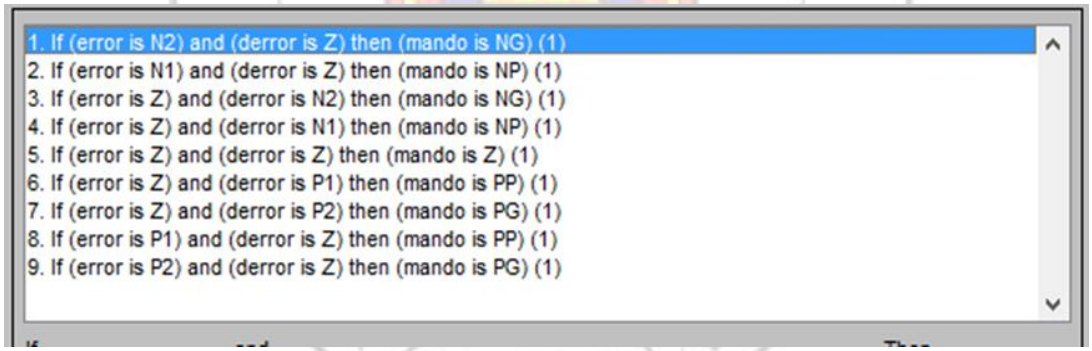
Tabla 12 Base de conocimiento para reglas difusas

		ERROR				
		N2	**N1**	**Z**	**P1**	**P2**
DELTA ERROR	**N2**	NG	NG	NG	PG	PG
	N1	NG	NP	NP	PP	PG
	Z	NG	NP	Z	PP	PG
	P1	NG	NP	PP	PP	PG
	P2	NG	NG	PG	PG	PG

Fuente: Autores

Seleccionando las reglas que actúan en puntos críticos se establecen las reglas en la figura 43.

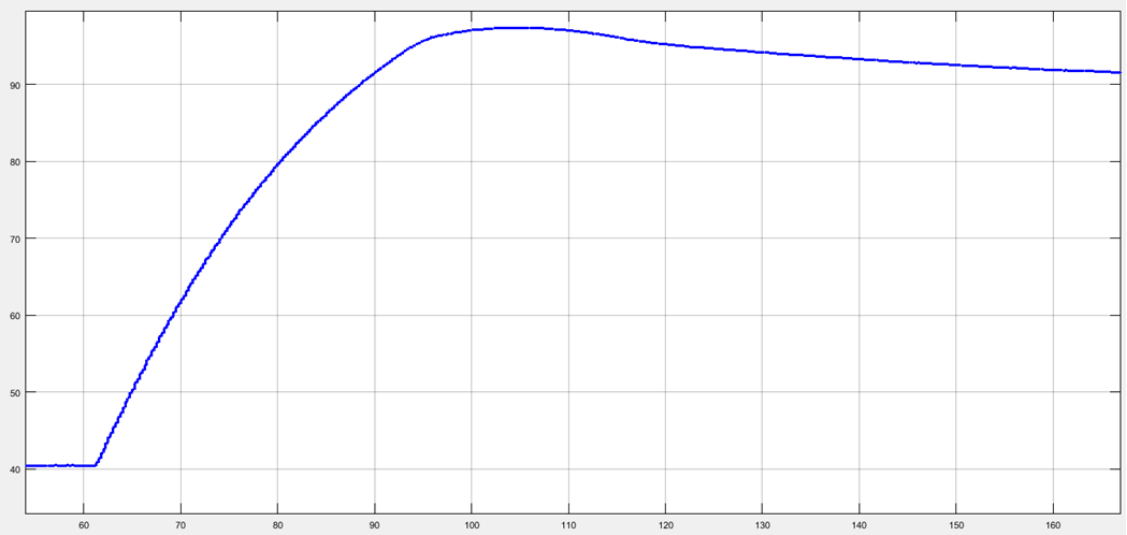
Figura 43 Reglas difusas



Fuente: Autores.

3. Se obtiene la respuesta de la figura 44, del controlador con Ki de referencia de 0.4 con un sobre impulso que supera el 10% del set point

Figura 44 Respuesta FPD+I sin ajuste

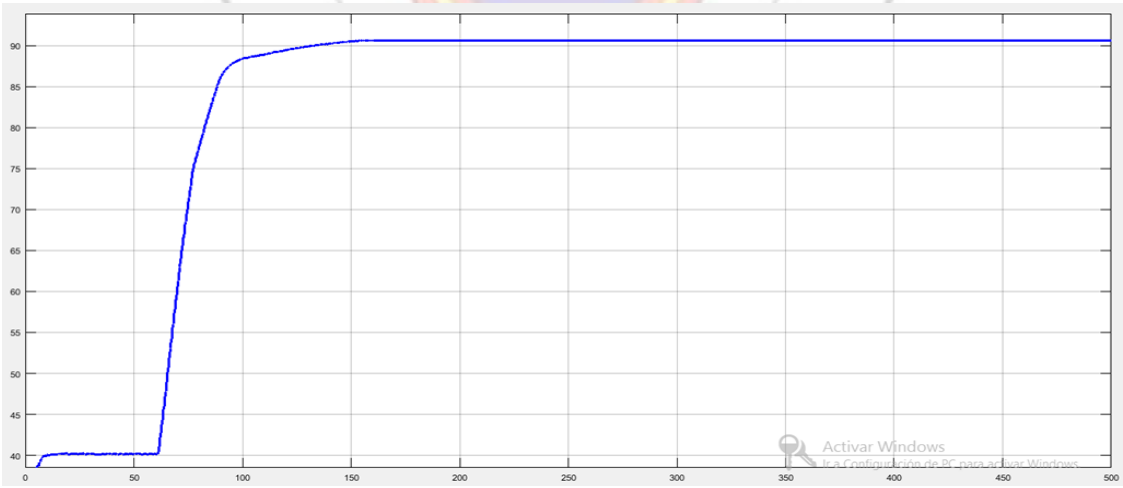


Fuente: Autores

.

4. Observando la figura 44 se puede deducir que para disminuir ese 10% se debe reducir la acción integral en donde el valor más apropiado es para Ki = 0.171, sin necesidad de cambiar las funciones de membresía se obtiene la respuesta de la figura 45.

Figura 45 Respuesta FPD+I con ajuste

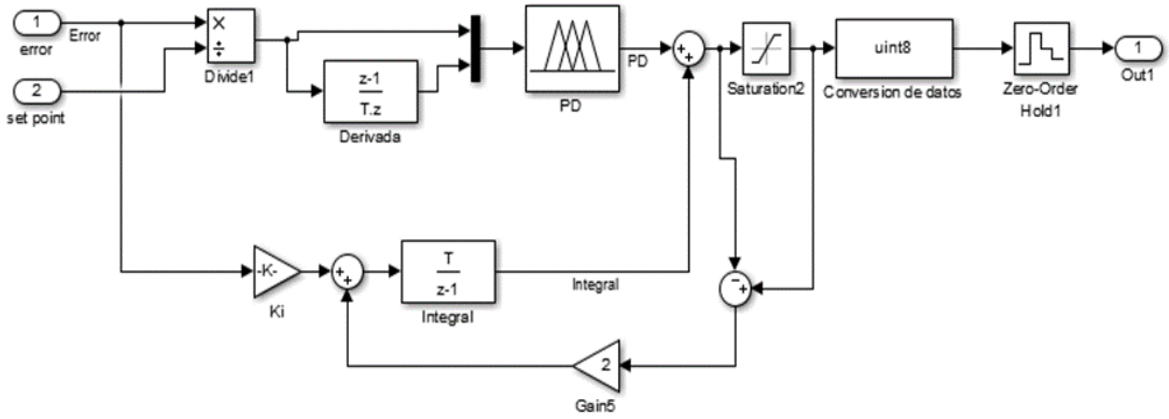


Fuente: Autores

5. Control FPD+I con anti-windup.

Reemplazamos del controlador anterior la parte integral, por el conjunto anti-windup como se observa en la siguiente figura 46.

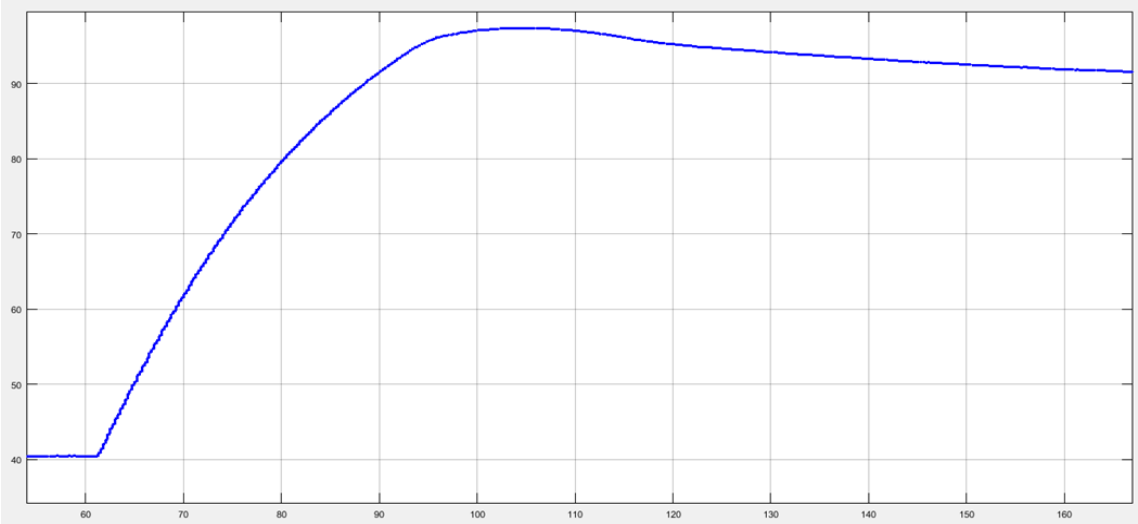
Figura 46 Control FPD+I con anti-windup



Fuente: Autores

6. Se obtiene la respuesta de la figura 47 de acuerdo con los valores de referencia Tt=0.00475 y Ki=0.4 con un sobre impulso prolongado del 9.5% del set point.

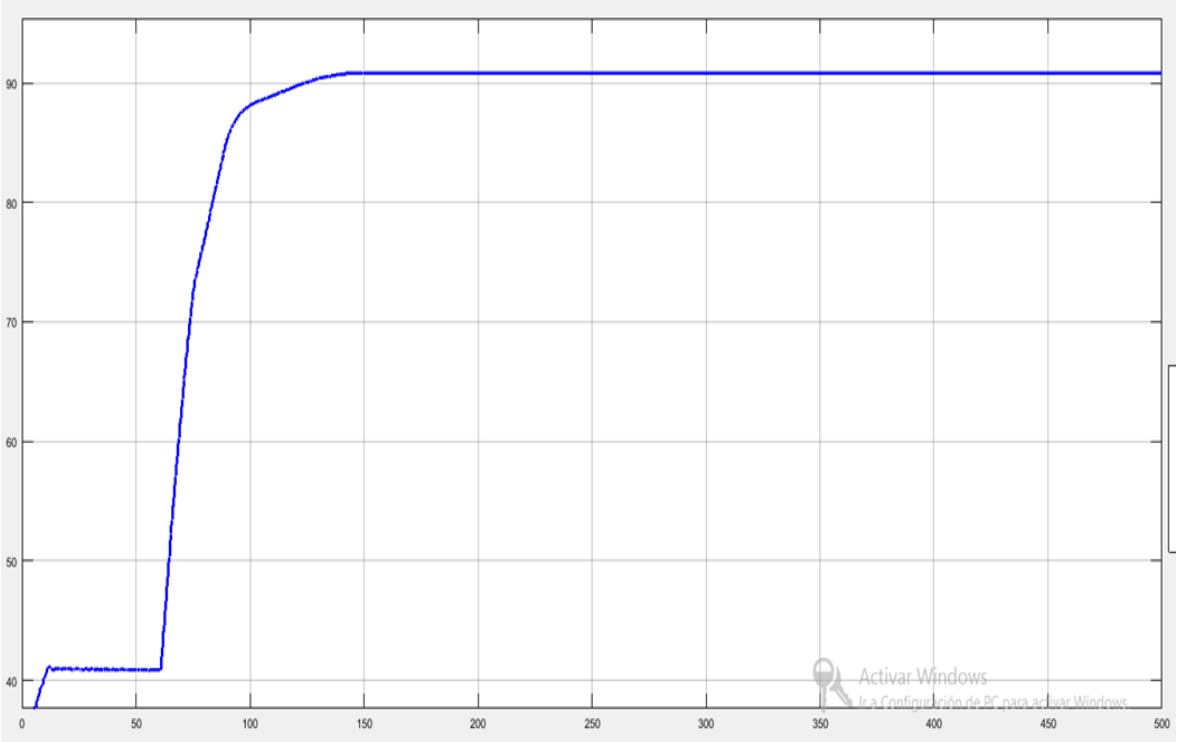
Figura 47 Respuesta FPD+I con anti- windup sin ajuste



Fuente: Autores

7. Teniendo en cuenta ajustes experimentales para Tt=0.18, Ki = 2, y las funciones de membresía del bloque fuzzy, se obtiene la siguiente respuesta de la figura 48 mejorada en el controlador real en la planta.

Figura 48 Respuesta FPD+I con anti-windup con ajuste



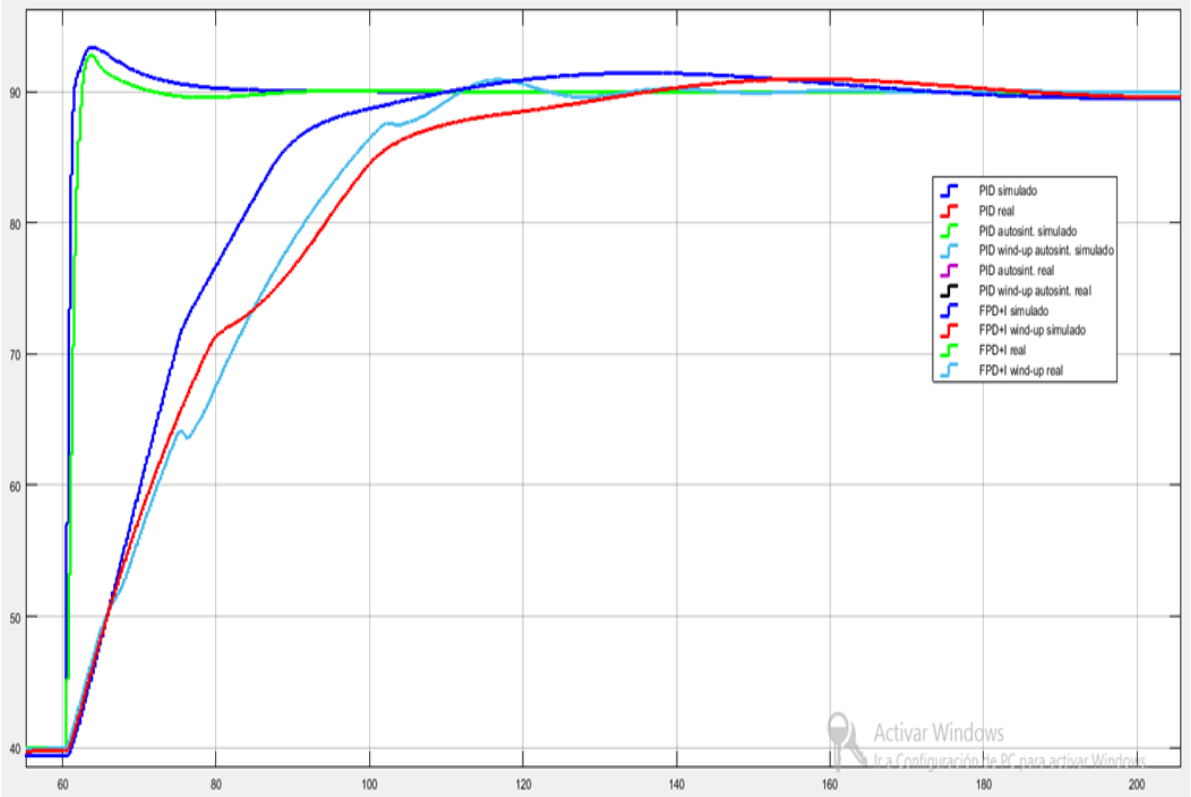
Fuente: Autores

.

5.1.5 Graficas comparativas

1. En la figura 49 se tiene las gráficas de simulaciones de control PID, PID auto sintonizable fuzzy, PID auto sintonizable fuzzy con anti-windup, FPD+I, FPD+I con anti-windup.

Figura 49 Grafica comparativa de simulaciones

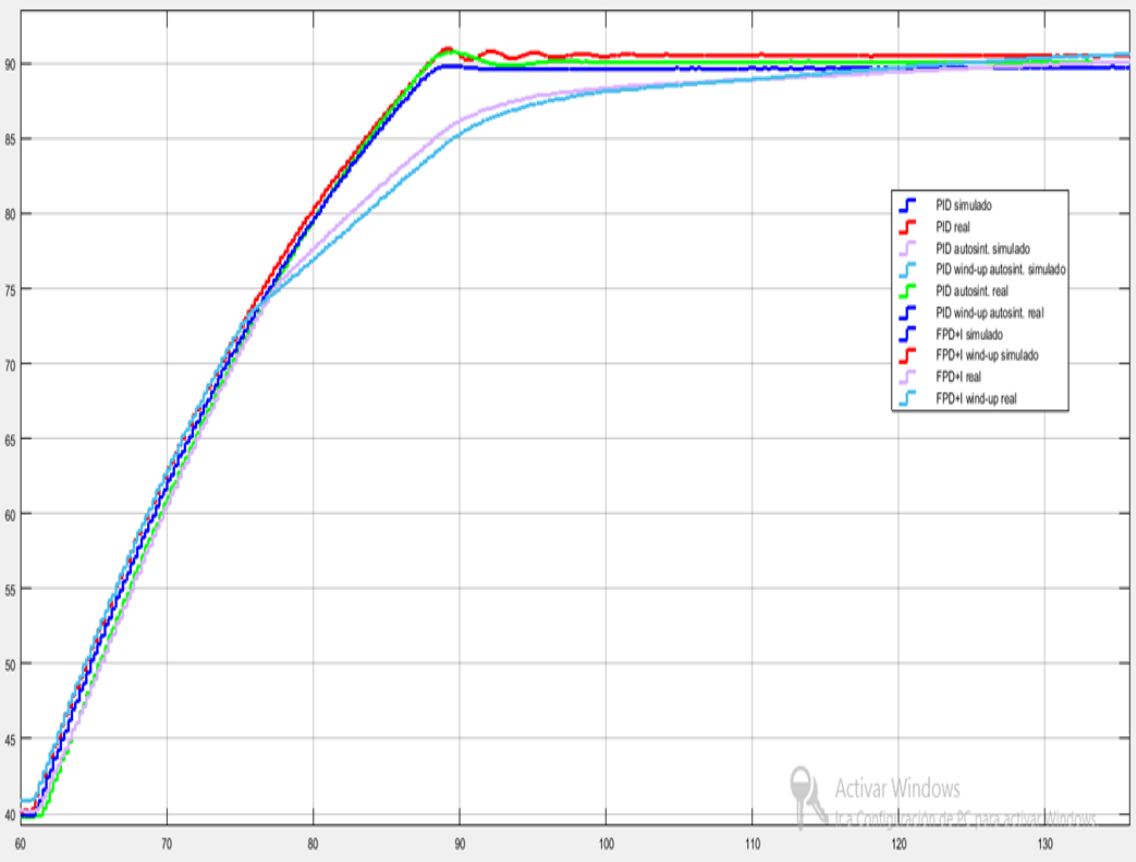


Fuente: Autores

En la gráfica de la figura 49, el controlador más rápido es el PID auto sintonizable, pero presenta un sobre impulso del 5%, por otra parte el controlador PID auto sintonizable con anti-windup presenta un tiempo de establecimiento mayor, pero con un sobre impulso del 1.8% del set point.

2. En la figura 50 se tiene las gráficas de control real en la planta, PID, PID auto sintonizable fuzzy, PID auto sintonizable fuzzy con anti-windup, FPD+I, FPD+I con anti-windup.

Figura 50 Graficas de control real de la planta



Fuente: Autores

De la figura 50, los dos mejores controladores con el menor tiempo de establecimiento son el control PID auto sintonizable fuzzy con un sobre impulso del 1.5% del set point, y el PID auto sintonizable con anti-windup sin sobre impulso.

3. Índices de desempeño

Antes de seleccionar el mejor controlador, no está de más hacer un mayor análisis con la ayuda de indicadores de desempeño. Centrando la atención en el control PID real de referencia y los dos controladores reales principales PID auto sintonizable y FPD+I obtenemos con la ayuda de la interfaz los valores del indicador de desempeño ISE e ITAE para el rango de tiempo de nuestro interés, del segundo 72 al 140.

Figura 51 Índices de desempeño de control real

ÍNDICES DE DESEMPEÑO REAL DE CONTROLADORES

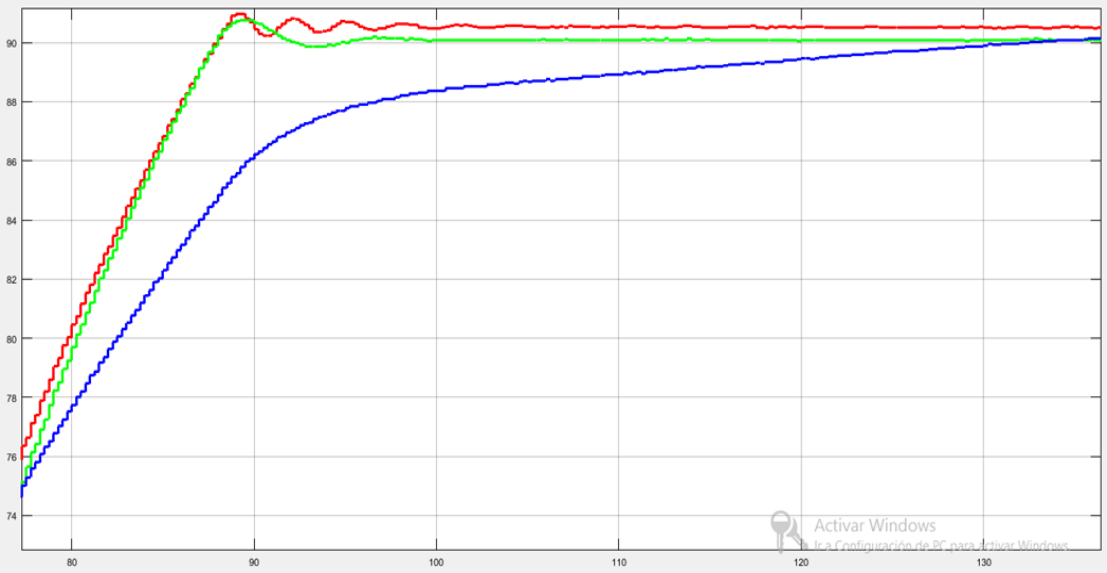
CONTROLADOR	ISE	ITAE
Clásico	2594.41	2003.25
PID auto sintonizable difuso	2971.07	1121.89
FPD+I	3606	3041.01
PID auto sintonizable difuso con anti-windup	2844.28	1610.9
FPD+I con anti-windup	3505.99	3441.44

Fuente: Autores

De acuerdo a los valores de la figura 51 el controlador FPD+I con y sin anti-windup presentan un ISE y un ITAE mucho mayor, y esto se puede verificar también al observar la gráfica, siendo este el que mayor tiempo de establecimiento presenta.

En la figura 52 se observa que el controlador PID clásico tiene la desventaja de presentar un error estable, lo que lo hace tener un ITAE mucho mayor que el PID auto sintonizable.

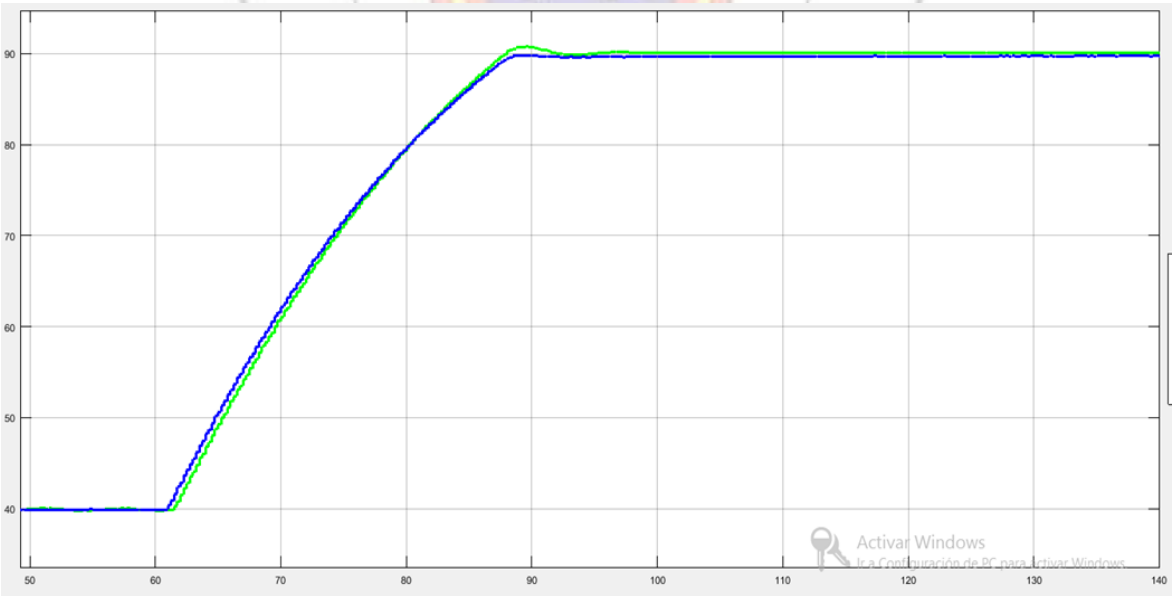
Figura 52 Grafica de los tres controladores principales



Fuente: Autores

Siendo el controlador PID auto sintonizable el de menor ITAE y un ISE no muy superior al valor mas bajo, podemos decir que es el mejor controlador y por lo tanto si se requiere de un controlador sin sobre impulso, podemos optar por el anti-windup para este controlador, obteniendo la siguiente respuesta en color azul de la figura 53.

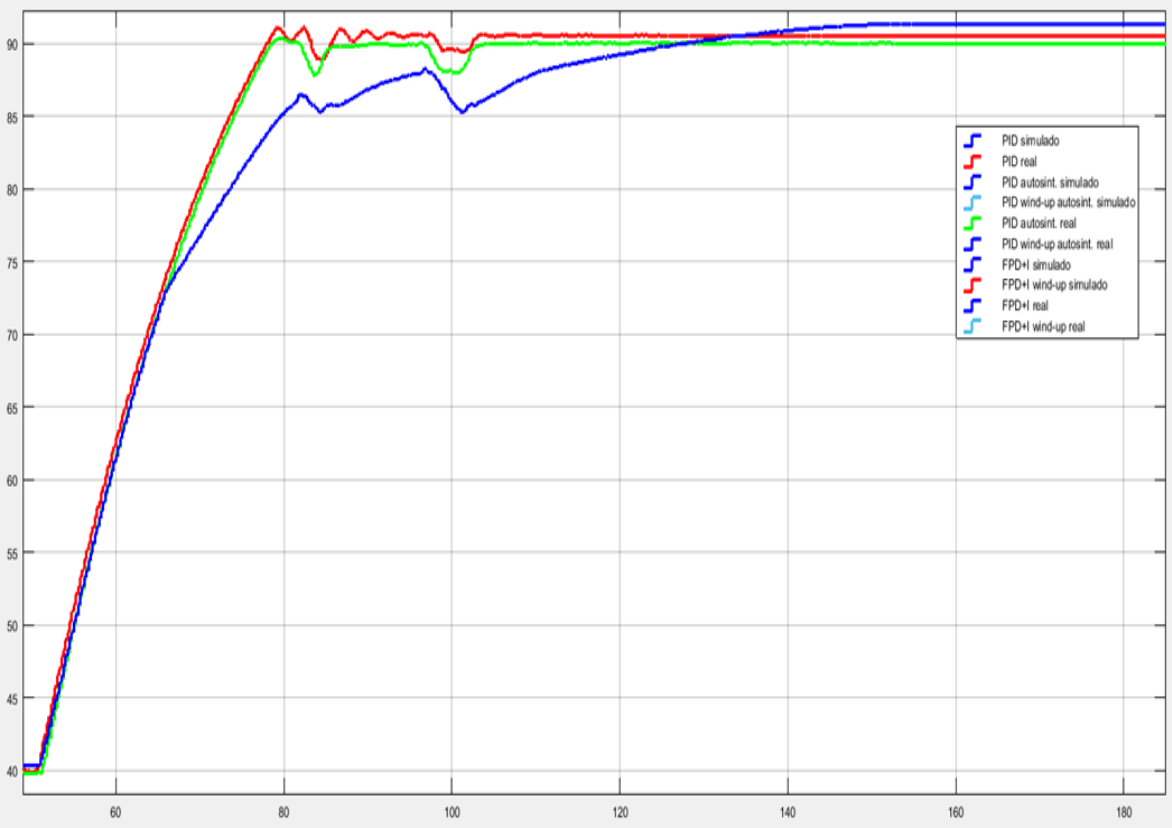
Figura 53 Control PID auto sintonizable con y sin anti-windup



Fuente: Autores

4. En la figura 54 se ilustra la gráfica de controladores principales real en la planta, PID, PID auto sintonizable fuzzy y FPD+I frente a perturbaciones.

Figura 54 Grafica de controladores principales frente a perturbaciones



Fuente: Autores

De acuerdo a la gráfica de la figura 54, se observa que el controlador PID es afectado en menor medida por las perturbaciones, pero mantiene un error en estado estable de 0.6 °C, a diferencia del auto sintonizable que tiende el error en estado estable a 0 °C.

Tabla 13 Resultados de controladores reales

Controlador	Máximo pico porcentual (Mp%)	Tiempo de estabilización (Ts)	Tiempo de crecimiento (Tr)	ISE	ITAE
PID clásico	2	44	28	2594.4	2003.2
PID auto sintonizable	1.52	39	28	2971	1121.8
PID auto sintonizable con anti-windup	0	35	29	2844.2	1610.9
FPD+I	1.3	100	72.5	3606	3041
FPD+I con anti-windup	1.7	82	64	3505.9	3441.4

De acuerdo a los resultados de la tabla 13, el mejor controlador es el PID autosintonizable por lógica difusa con y sin anti-windup, y en el caso del controlador FPD+I no se obtuvieron los mismos resultados al implementar el anti-windup, incluso el sobreimpulso aumento, al igual que un error en estado estable del 1.6%, teniendo en cuenta que existe una complejidad para la compensación exacta al utilizar anti-windup.

ANEXO 6 DATASHEET OPTOACOPLADOR PC817A

■ **Electro-optical Characteristics** (T_a=25°C)

	Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input	Forward voltage		V_F	I_F=20mA	–	1.2	1.4	V
	Peak forward voltage		V_{FM}	I_{FM}=0.5A	–	–	3.0	V
	Reverse current		I_R	V_R=4V	–	–	10	μA
	Terminal capacitance		C_t	V=0, f=1kHz	–	30	250	pF
Output	Collector dark current		I_{CEO}	V_{CE}=50V, I_F=0	–	–	100	nA
	Collector-emitter breakdown voltage		BV_{CEO}	I_C=0.1mA, I_F=0	*5 80	–	–	V
	Emitter-collector breakdown voltage		BV_{ECO}	I_E=10μA, I_F=0	6	–	–	V
Transfer characteristics	Collector current		I_C	I_F=5mA, V_{CE}=5V	2.5	–	30.0	mA
	Collector-emitter saturation voltage		$V_{CE(sat)}$	I_F=20mA, I_C=1mA	–	0.1	0.2	V
	Isolation resistance		R_{ISO}	DC500V, 40 to 60%RH	5×10^{10}	1×10^{11}	–	Ω
	Floating capacitance		C_f	V=0, f=1MHz	–	0.6	1.0	pF
	Cut-off frequency		f_c	V_{CE}=5V, I_C=2mA, R_L=100Ω, –3dB	–	80	–	kHz
	Response time	Rise time	t_r	V_{CE}=2V, I_C=2mA, R_L=100Ω	–	4	18	μs
		Fall time	t_f		–	3	18	μs

*5 From the production Date code "J5" (May 1997) to "P7" (July 2002), however the products were screened by BV_{CEO}≥70V.

Fuente: Electronica Sharp

ANEXO 7 DATASHEET MOSFET IRF840

SPECIFICATIONS (T_J = 25 °C, unless otherwise noted)							
PARAMETER	**SYMBOL**	**TEST CONDITIONS**		**MIN.**	**TYP.**	**MAX.**	**UNIT**
Static							
Drain-source breakdown voltage	V_{DS}	V_{GS} = 0 V, I_D = 250 µA		500	-	-	V
V_{DS} temperature coefficient	$\Delta V_{DS}/T_J$	Reference to 25 °C, I_D = 1 mA		-	0.78	-	V/°C
Gate-source threshold voltage	$V_{GS(th)}$	V_{DS} = V_{GS}, I_D = 250 µA		2.0	-	4.0	V
Gate-source leakage	I_{GSS}	V_{GS} = ± 20 V		-	-	± 100	nA
Zero gate voltage drain current	I_{DSS}	V_{DS} = 500 V, V_{GS} = 0 V		-	-	25	µA
		V_{DS} = 400 V, V_{GS} = 0 V, T_J = 125 °C		-	-	250	
Drain-source on-state resistance	$R_{DS(on)}$	V_{GS} = 10 V	I_D = 4.8 A [b]	-	-	0.85	Ω
Forward transconductance	g_{fs}	V_{DS} = 50 V, I_D = 4.8 A [b]		4.9	-	-	S
Dynamic							
Input capacitance	C_{iss}	V_{GS} = 0 V, V_{DS} = 25 V, f = 1.0 MHz, see fig. 5		-	1300	-	pF
Output capacitance	C_{oss}			-	310	-	
Reverse transfer capacitance	C_{rss}			-	120	-	
Total gate charge	Q_g	V_{GS} = 10 V	I_D = 8 A, V_{DS} = 400 V, see fig. 6 and 13 [b]	-	-	63	nC
Gate-source charge	Q_{gs}			-	-	9.3	
Gate-drain charge	Q_{gd}			-	-	32	
Turn-on delay time	$t_{d(on)}$	V_{DD} = 250 V, I_D = 8 A, R_g = 9.1 Ω, R_D = 31 Ω, see fig. 10 [b]		-	14	-	ns
Rise time	t_r			-	23	-	
Turn-off delay time	$t_{d(off)}$			-	49	-	
Fall time	t_f			-	20	-	
Internal drain inductance	L_D	Between lead, 6 mm (0.25") from package and center of die contact		-	4.5	-	nH
Internal source inductance	L_S			-	7.5	-	
Gate input resistance	R_g	f = 1 MHz, open drain		0.6	-	2.8	Ω

Fuente: Electronica Vishay.

ANEXO 8 MATERIALES Y COSTOS

Costos del prototipo de temperatura

	Material	Cantidad	Costo
1	Arduino Mega 2560	1	$ 82.000,00
2	Resistencia calefactora	1	$ 178.000,00
3	Sensor MLX90614	1	$ 89.000,00
4	Cooler	1	$ 13.000,00
5	Baquelita	1	$ 14.000,00
6	Impresión de circuito	1	$ 6.000,00
7	Interruptor	1	$ 6.000,00
8	Regleta	1	$ 11.000,00
9	Socket Dc	1	$ 7.000,00
10	Cable de color	5	$ 5.500,00

11	Conector bornera	3	$	3.000,00
12	Disipador de calor	1	$	5.600,00
13	Optoacoplador PC817A	1	$	2.000,00
14	Mosfet IRF840	1	$	6.000,00
15	Resistencias 10K, 220 ohm	4	$	800,00
16	Lamina de acrilico	1	$	60.000,00
17	Corte laser	1	$	52.000,00
18	Tornillos y tuercas	9	$	4.600,00
19	Carcasa para arduino	1	$	9.000,00
20	Fuente de alimentación	1	$	41.000,00
21	Estaño	1	$	2.000,00
22	Transporte	3	$	25.000,00
23	Otros	1	$	55.000,00
		Total	$	677.500,00

Fuente: Autores