Universidad Católica de Santa María
Facultad de Ciencias e Ingenierías Físicas y Formales
Escuela Profesional de Ingeniería Electrónica

"DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR DIFUSO UNIVERSAL PARA HORNOS ELÉCTRICOS SEMI-INDUSTRIALES"

Tesis presentada por el Bachiller:
Nuñez Mantilla, Jianfranco Enmanuel

para optar el Título Profesional de
Ingeniero Electrónico.

Con especialidad en:
Automatización y Control.

Asesor:
Dr. Ing. Sulla Torres, Raúl Ricardo

Arequipa - Perú

2021

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA ELECTRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 29 de Mayo del 2021

Dictamen: 000661-C-EPIE-2021

Visto el borrador del expediente 000661, presentado por:

2012600721 - NUÑEZ MANTILLA JIANFRANCO ENMANUEL

Titulado:

DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR DIFUSO UNIVERSAL PARA HORNOS ELÉCTRICOS SEMI-INDUSTRIALES

Nuestro dictamen es:

APROBADO

1546 - DELGADO BARRA LUCY ANGELA
DICTAMINADOR

1767 - SULLA TORRES RAUL RICARDO
DICTAMINADOR

2465 - ZEGARRA GAGO HENRY CHRISTIAN
DICTAMINADOR

Dedicatorias

Dedico este trabajo a mis padres, por su incondicional apoyo, sus concejos y ser aquel soporte para formar al hombre que hoy soy.

A mis amigos, compañeros, e instructores, por todas las enseñanzas y prepararme para la vida.

G. R. por ser mi inspiración y soporte para mejorar día a día.

.

.

RESUMEN

En la industria se puede encontrar una gran variedad de procesos donde se requiere una acción de control e instrumentación para encontrar la condición optima de trabajo, en este proyecto, nos vamos a enfocar en los procesos térmicos. En un sector de este tipo de procesos nos encontramos con los hornos eléctricos.

Podemos encontrar hornos en diversos rubros y categorías, cada uno de diferente tamaño y potencia, es por esto que estos equipos requieren de un control sintonizado específicamente diseñado para cada uno de ellos si se busca que trabajen en las mejores condiciones de control, en otros casos, encontramos con sistemas sin realimentación o en su defecto con control On/Off.

El presente proyecto de investigación desarrolla una comparativa entre el método de control PI que se encuentra presente ampliamente en la industria y el método de control difuso, que maneja una lógica basada en el conocimiento del operador y no requiere del modelo matemático del sistema a controlar. Para demostrar esta comparativa, se desarrollará e implementarán tanto un controlador difuso y un controlador PI en un horno eléctrico semi industrial. Y se presentarán las conclusiones del caso.

Palabras claves:

Control difuso, sistemas de control avanzado, controladores PI, sistemas térmicos.

ABSTRACT

In the national industry, we can find a great variety of process where each one requires to take an action of control & instrumentation to work properly, in this project, we are going to focus on the thermal process. Especially over the electric oven.

The electric oven is present in many factories and we can find it in some categories, each one in different size and power, this is why we have to design a specific controller for each type, by the other hand, some ovens don't have a sensor to determinate the correct temperature or just it has an On/Off controller.

This research project will explain the comparison between a PI controller, which is present in a lot of kinds of process and a fuzzy controller, who have a different way of work, this controller doesn't have to know the mathematical equation of the system to control and work efficiently. For this, we are going to implement both of them in an electric oven for the semi industrial purposes.

Key words:
Fuzzy logic, modern control system, thermal system, PI controllers.

INTRODUCCIÓN

En los últimos años, la ingeniería de control ha optado como misión, la simplificación de los procesos industriales, durante la última gran revolución, se dio el paso hacia la industria 4.0, que además de ser el salto hacia los dispositivos inalámbricos, o integradores de bolsillo, ha optado por dispositivos inteligentes que, en su medida, tomen decisiones en rendimiento del proceso basadas en un algoritmo y no en la vigilia constante de un operador. Aplicándose a los dispositivos de campo, en mayoría, estos basan su comportamiento según lo dicte un dispositivo de control supervisorio o una indicación de un operador de campo. La diferencia entre la técnica de control entre estos dispositivos y el controlador maestro, es que estos dispositivos de campo basan su funcionamiento, en su mayoría, con una configuración matemática, un algoritmo de control como lo es el controlador PI. En cambio, los tipos de controlador supervisorio muestran un comportamiento similar al que lo haría un operador, entendiendo el problema físico por parámetros difusos.

Es por esto que esta tesis de investigación, propone realizar un controlador difuso que actúe sobre una variable física, en este caso, temperatura, tal cual lo haría un controlador de línea en un proceso industrial, con un rango de salida proporcional de 0 a 100%. Además, se presenta una comparativa con un clásico controlador como lo es el PI digital.

Se pretende observar la respuesta en el tiempo entre ambos controladores, la respuesta transitoria y el costo de implementar un gabinete de control basado en microcontrolador frente a un controlador semi-industrial que se encuentra en el mercado.

ÍNDICE

Lista de figuras

Lista de tablas

CAPITULO I
PLANTEAMIENTO TEÓRICO

CAPITULO I

1. PLANTEAMIENTO TEÓRICO

1.1.Descripción del problema

En la industria existe una gran variedad de procesos desde los cuales, se logra transformar una materia en un producto final. De los factores más importantes, resaltamos la calidad y costo.

La calidad de estos productos, depende entre muchos factores, según la precisión con la cual fue diseñado y elaborado. Para esto debemos considerar que los equipos encargados de desarrollar el producto, deben de cumplir con los requisitos de precisión.

En los procesos térmicos, una herramienta común son los hornos industriales, específicamente los hornos eléctricos industriales y semi industriales. Uno de los principales inconvenientes de los hornos eléctricos supone la gran demanda energética que este requiere al momento de iniciar operaciones, en algunos casos, al no tener una técnica de control establecido, los hace susceptibles a perturbaciones o lo que se percibe como una perdida, tanto energética como de calidad, lo que desemboca en una pérdida económica. Este tipo de sistemas no pueden repeler efectivamente perturbaciones al sistema y no siguen un setpoint de manera confiable, por otro lado, el consumo energético es superior, al no monitorear la planta, el sistema se basa en la intervención humana para suministrar la energía necesaria, la cual en muchas ocasiones es mayor a la requerida.

1.2 OBJETIVOS

OBJETIVO PRINCIPAL

Diseñar e implementar un controlador difuso universal para hornos semi-industriales.

OBJETIVOS ESPECÍFICOS

- Aplicar la teoría de control avanzado para demostrar las técnicas de control difuso frente al control PI.
- Establecer una interfaz hombre maquina donde se pueda monitorear las variables de proceso y observar un gráfico de tendencia.
- Programar un microcontrolador de 8 bits en lenguaje C.
- Implementar un tablero eléctrico para el control de hornos en campo.

1.3ALCANCES

El alcance de este proyecto abarca a aquellas industrias donde se utilicen hornos de tipo eléctrico, que generen calor a partir del uso de resistencias térmicas y busquen optimizar el proceso de fabricación de sus diferentes tipos de productos o donde se busque precisión de temperatura ante perturbaciones.

Para este proyecto de investigación se implementarán ambas técnicas de control en un horno eléctrico de la industria alimentaria, y automatizar el proceso con la técnica de control difusa para crear un dispositivo comercial de control.

La potencia del horno no debe superar la capacidad de la etapa de potencia del diseño y se debe encontrar como mínimo en el rango semi-industrial (5kW) tal como se encuentra en el área de mantenimiento industrial.

El prototipo deberá cumplir con los estándares de calidad internacional y se busca el desarrollo modular, esto significa que un mismo controlador acoplado a una variedad de etapas de potencia podrá cumplir su función con un amplio rango de hornos industriales de diferente potencia y mantendrá los costos bajos para los equipos que no requieran de potencias elevadas.

1.4 VARIABLES

1.4.1 VARIABLES INDEPENDIENTES

Energía eléctrica, temperatura ambiental, modo de trabajo.

1.4.2 VARIABLES DEPENDIENTES

Temperatura de cámara, movimiento másico de aire en cámara, potencia de trabajo.

CAPITULO II
MARCO TEÓRICO

CAPITULO II

2. MARCO TEÓRICO

2.1. Hornos industriales

En la industria existen diversos tipos de hornos, podemos clasificarlos según el combustible que utilicen para calentar la cámara de cocción, como hornos a gas, gasoil o eléctricos. En este proyecto vamos a enfocarnos en estos últimos.

2.2. Principio de funcionamiento de hornos eléctricos

Existe una gran cantidad de hornos eléctricos en los que se les puede aplicar esta técnica de control, básicamente todos los que utilicen como actuador resistencias eléctricas.

Este tipo de horno es ampliamente usado en la industria debido a la capacidad de alcanzar altas temperaturas fácilmente, la seguridad que ofrece al no utilizar hidrocarburos y mayor tiempo de vida sin mantenimiento comparado a sus semejantes basados en hidrocarburos.

a. Horno cerámico eléctrico

Es ampliamente utilizado en las unidades de investigación científica, instituciones de enseñanza superior, empresas industriales y mineras ideal experimentos y equipo de producción. Es Utilizado principalmente para la alta temperatura de recocido, la micro cristalización, esmalte de cerámica, metalurgia de polvos, polvo de plástico preparación, molde recocido experimento, la sintonización de los nano-materiales, piezas de metal Temple, Y todos los requisitos tecnológicos de tratamiento térmico a alta temperatura. [1]

b. Horno en la industria del Cemento

Horno rotatorio es ampliamente utilizado para la piedra caliza calcinación para industrias como la metalurgia, química, materiales de construcción, de acuerdo con diferentes materiales, que se puede dividir. En la figura 1 podemos encontrar un ejemplo gráfico.

El horno de cal se utiliza para asar cal activa en obras de acero y ferroaleaciones plantas y tostado ligero dolomita.

Horno en metalurgia química es ampliamente utilizado para magnético tostado de pobres de mineral de hierro y oxidantes.

Horno de cemento se utiliza para la fabricación de cemento Clinker hay métodos seco y húmedo para hacer cemento Clinker.

Fig. 1 Ejemplo de hornos eléctricos industriales
Fuente, [1]

2.3. Resistencias térmicas

El alambre nicrom es una aleación de níquel y cromo, este material es altamente resistivo, tiene un punto de fusión de 1400°C y tiene gran durabilidad ante la corrosión, por lo que se utiliza ampliamente como calefactor.

El principio de funcionamiento se basa en la ley de Joule, que explica la generación de calor a partir del trabajo, en este caso, a partir de la intensidad eléctrica.

La ecuación de Joule está dada por: [2]

$$Q = 0.2392 * R * I^2 * t \ (1)$$

Donde:

Q = Calor en calorías.

R = Resistencia en Ohmios.

t = Unidad de tiempo en segundos.

0.2392 = constante K.

2.4.Sistemas de medición

Estos dispositivos son ampliamente usados en la industria, podemos distinguirlas debido al principio de funcionamiento.

2.4.1. Sensores resistivos

Un sensor industrial altamente recomendado por su linealidad y precisión es el PT100, el nombre deriva de Platino y el valor 100 en Ohmios a 0 °C. En la figura 2 se puede observar la curva de resistencia relativa.

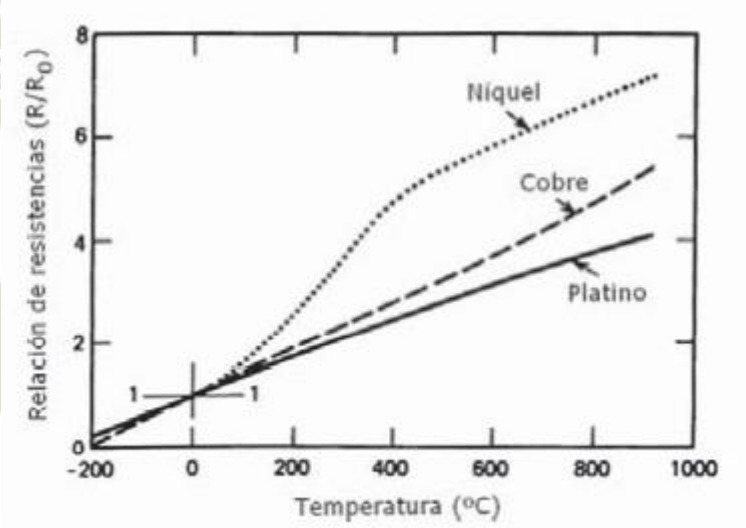


Fig. 2 Curva de resistencia relativa en diferentes materiales, Fuente [2]

2.4.2. Termopares

El principio de funcionamiento se basa en el efecto seebeck que consta en la generación de electricidad a partir del calor, en el orden de los mili-voltios. [2]

Algunas consideraciones con estos dispositivos son:

- Compensación por unión fría.
- Empalme de hilos debe de ser del mismo material que el termopar.
- Para sistemas de control, se recomienda utilizar un transmisor de alta impedancia.

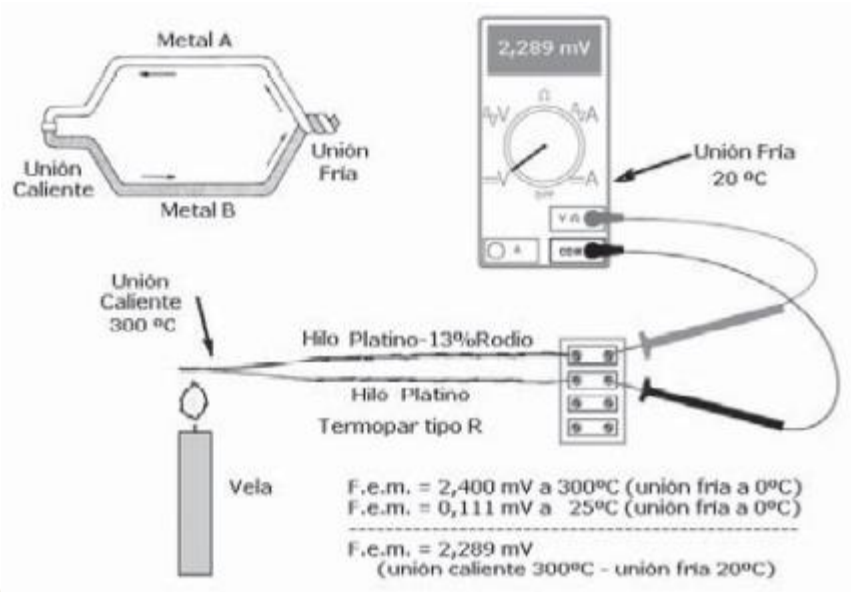


Fig. 3 Ilustración del funcionamiento de un termopar,
Fuente [2]

Los termopares más comunes son:

Termopar tipo J, de Hierro/Constatan, Es adecuado en atmosferas inertes, su rango de temperatura es entre -200°C y 1.200°C. Uno de sus principales inconvenientes es la oxidación del Hierro a partir de los 500°C por lo que no se recomienda en ambientes húmedos. En la figura 3 se ilustra el funcionamiento de un termopar mientras que en la figura 4 se muestra la curva característica de diferentes termopares.

Termopar tipo K, de Níquel-Cromo/Níquel-Aluminio, Se recomienda su uso en Atmósferas oxidantes, su rango de medición es entre -40°C y 1100°C, para atmósferas sulfurosas de requiere de la utilización de un tubo protector. [2]

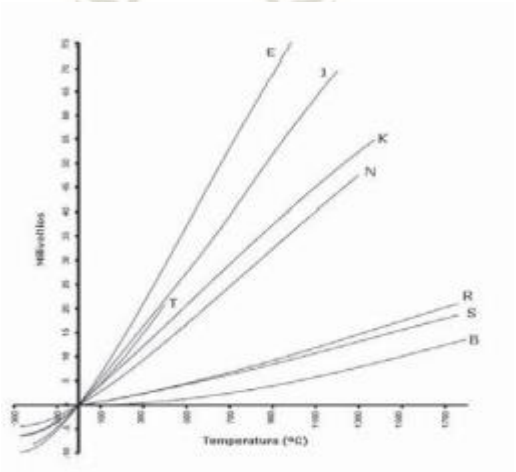


Fig. 4 Curva característica de los termopares
Fuente [2]

2.5. Controladores industriales

Se reconoce como controladores a todos aquellos sistemas que regulan o controlan un proceso, en la industria podemos encontrar controladores de todo tipo, algunos de los más simples como un controlador On/Off y dependiendo de la complejidad del sistema pasamos por controles PID, Predictivos, o hasta Redes Neuronales.

Para poder implementarlos, se necesita en muchos casos diferentes niveles de control y sistemas capaces de poder procesar el algoritmo de control. En la actualidad, la técnica de control discreto se ha implementado ampliamente en controladores industriales como PLC, DCS, Sistemas embebidos, o incluso en computadoras industriales, teniendo así cada sistema sus ventajas y desventajas.

2.5.1. PLC

Es un ordenador industrial basado en microprocesador, sus siglas vienen del inglés, Programmable Logic Controller, existen diferentes versiones para cada tipo de aplicación, se pueden encontrar versiones basadas en FPGA para el cálculo de diferentes procesos en simultáneo, así como una gran variedad de módulos que amplían las características, así como mayores prestaciones en nivel de comunicación y expansiones en puertos I/O. En la figura 5 se muestra la estructura interna de un PLC.

Una de las definiciones más acertadas es la dada por la NEMA según [3]:

"Instrumento electrónico, que utiliza memoria programable para guardar instrucciones sobre la implementación de determinadas funciones, como operaciones lógicas, secuencias de acciones, especificaciones temporales, contadores y cálculos para el control mediante módulos de E/S analógicos o digitales sobre diferentes tipos de máquinas y de procesos".

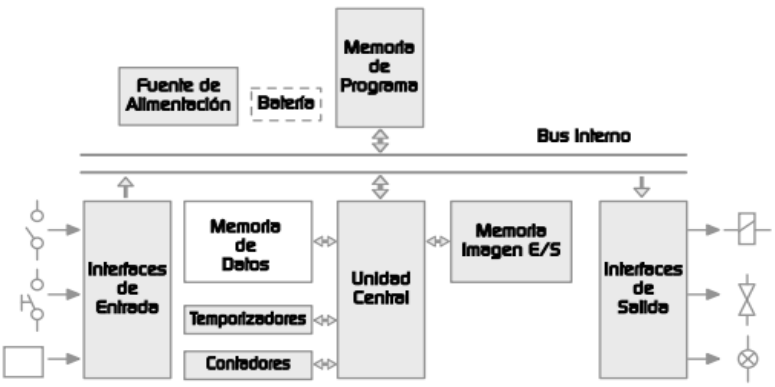


Fig. 5 Estructura interna de un PLC, Fuente [4]

2.5.2. Sistemas embebidos

Se considera a un sistema embebido cuando ha sido diseñado para un propósito específico, a diferencia del PLC, estos sistemas están diseñados exclusivamente para cumplir una tarea o proceso específico.

En comparativa, tienen una arquitectura interna similar al PLC, con la diferencia que comúnmente, se diseña en base a un microcontrolador, igualmente necesita diseñar una placa de control, de acondicionamiento de señal y una etapa de potencia, así como etapas específicas como comunicaciones y comparadores de señal.

El cerebro de estos sistemas, encargado de la parte lógica y del control, son comúnmente microcontroladores, se puede escoger dependiendo la complejidad del sistema. Existen microcontroladores de 8, 16 y 32bits, algunos de ellos con la capacidad de incorporar un DSP internamente para el tratamiento de señales analógicas.

En la figura 6, se muestra la arquitectura interna de un microcontrolador PIC de la serie 18F:

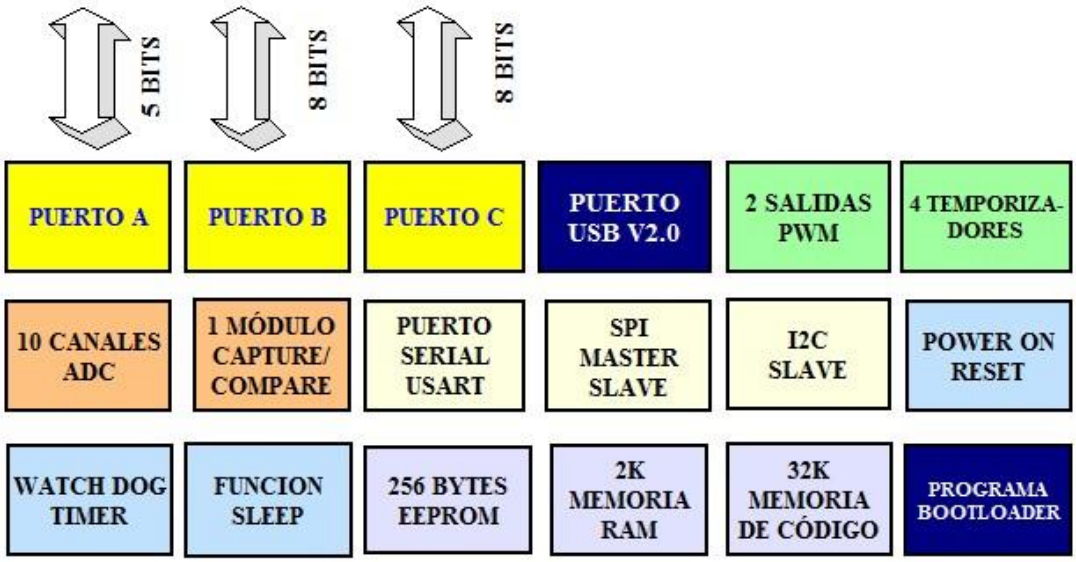


Fig. 6 Diagrama de bloques interno de uC PIC,
Fuente, [5]

Estos microcontroladores se pueden programar en distintos lenguajes de programación, entre los cuales, destacan ASM y C.

La programación en Assembler nos permite crear un código optimizado, preciso y tener un control real sobre cada registro en cada operación, pero tiene la desventaja de ser un proceso lento y poco adaptable para otro microcontrolador.

La programación en C, nos permite crear códigos de programación complejos de manera rápida y efectiva, la nueva gama de microcontroladores de Microchip, vienen optimizados para compiladores en C, así como el compilador gratuito de Microchip XC8, XC16 para la gama de 16bits y XC32 para los microcontroladores de 32 bits.

Basándonos en una etapa de control y agregando etapas según lo requerido, podemos diseñar e implementar circuitos embebidos capaces de solucionar de manera efectiva una determinada tarea o proceso de una manera eficiente y económica.

2.6. Control y automatización

"*Las teorías de control que se utilizan habitualmente son la teoría de control clásica (también denominada teoría de control convencional), la teoría de control moderno y la teoría de control robusto.*" [6]

Automatizar es la acción de modificar un proceso para que no requiera intervención humana, por lo que, en este contexto, sería la utilización de un dispositivo electrónico capaz de controlar un proceso de manera óptima sin la necesidad de supervisión constante de parte de un operador.

2.6.1. Control On Off

Es la técnica más primitiva de control realimentado, basado en mi experiencia, junto con el controlador PID, se encuentran ampliamente en la industria. Esta técnica de control no es recomendable debido a que no es una técnica precisa y produce desgaste prematuro en los actuadores del sistema.

2.6.2. Control PID

Esta técnica de control consiste en la suma de tres etapas de acción de control, como su nombre lo indica, la acción Proporcional, que se encarga de la ganancia general del controlador, la acción Integrativa que incorpora al valor acumulativo del error y la acción Derivativa, que se encarga de derivar el error, muy útil en procesos rápidos.

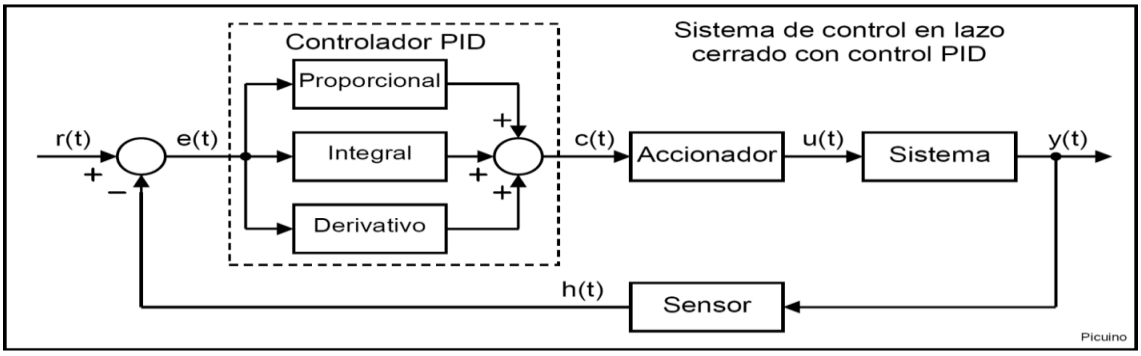


Fig. 7 Diagrama de controlador PID,
Fuente [7]

Como se muestra en la figura 7, el principio de funcionamiento se basa en la suma combinada de sus tres etapas, la cual da una señal de control c(t), el uso de un actuador u(t), y el uso de sensores para medir los parámetros físicos que controlamos entregando este último una señal de realimentación h(t).

a. Etapa proporcional

Como se muestra en la figura 8, esta etapa es representada por la constante Kp, y es directamente proporcional a la velocidad de respuesta del sistema, si se eleva demasiado disminuye la estabilidad y también es utilizada para disminuir el error el régimen estable. [7]

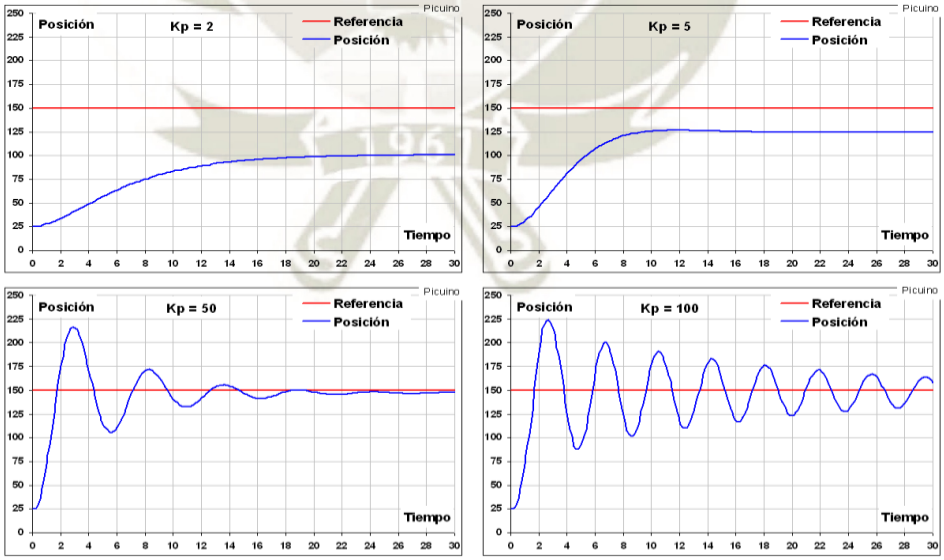


Fig. 8 Respuesta de un sistema con control Proporcional,
Fuente [7]

b. Etapa derivativa

Por otro lado, como se muestra en la figura 9, esta etapa también es conocida como regulación de velocidad, utiliza la constante Kd, se encarga de incrementar la derivada del error, es inversamente proporcional a la velocidad de respuesta del sistema, no afecta al error en el régimen permanente y aumenta la estabilidad del sistema. [7]

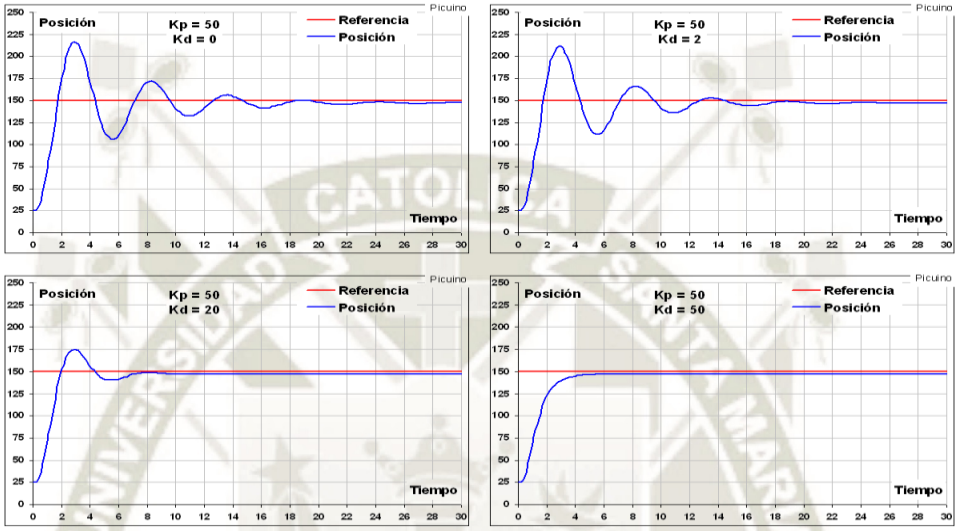


Fig. 9 Respuesta al escalón de un sistema con control PD
Fuente [7]

c. Etapa Integradora

Esta etapa tiene como constante a Ki, como el nombre lo indica, se encarga de realizar la integral del error mediante sumas acumuladas, se utiliza para reducir el error en el régimen permanente e incrementar la velocidad del sistema. Aumentar mucho esta ganancia puede generar inestabilidad. [7]

En la figura 10 se puede observar la respuesta de un controlador PI.

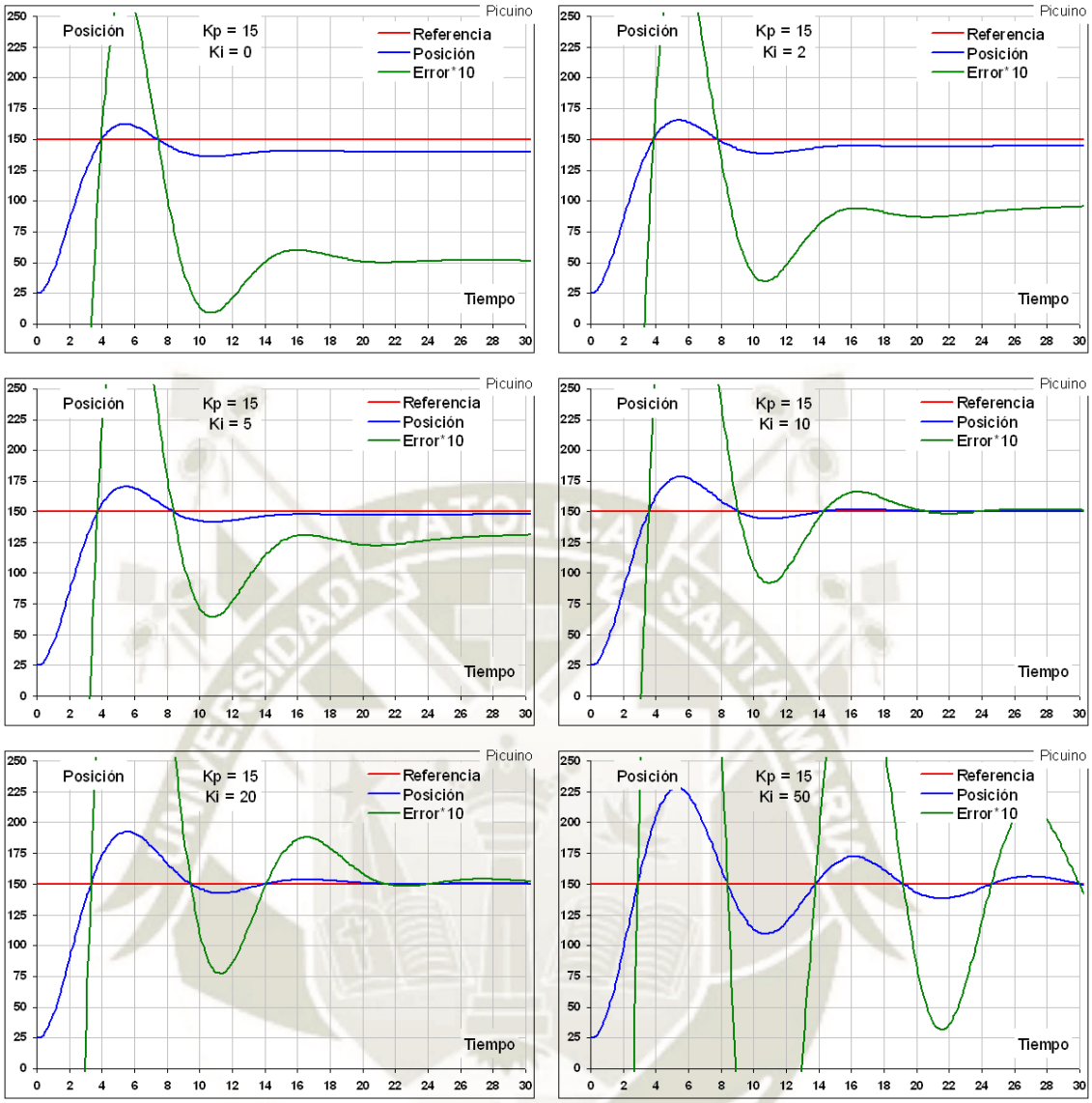


Fig. 10 Respuesta al escalón de un sistema con control PID
Fuente [7]

En la figura 10, se muestran los datos de referencia y error en un sistema con regulador PI, este con una ganancia Kp fija y en cada cuadro se modifica el valor Ki. Como podemos observar un valor alto en Ki mejora la exactitud del sistema, pero si el valor es demasiado alto, el sistema se vuelve oscilatorio en su etapa transitoria. [7]

2.6.3. Control PI en procesos térmicos

En muchos procesos térmicos se suele prescindir de la etapa derivativa de los reguladores PID, esto se debe a que la naturaleza del sistema, al ser un proceso lento, no requiere precisamente de esta función.

Los controladores industriales, como el que se muestra en la figura 11, suelen integrar la etapa derivativa en su sistema de control para evitar interferencias, así como la caída brusca de temperatura en el proceso.

Fig. 11 Controlador PID Autonics TZN4S
Fuente [7]

Aunque en general la aplicación de ambos controladores es aceptable, en procesos que requieran precisión se recomienda no omitir la etapa derivativa debido a su acción ante interferencias. En la figura 12 se puede observar la comparativa entre estos 3 controladores.

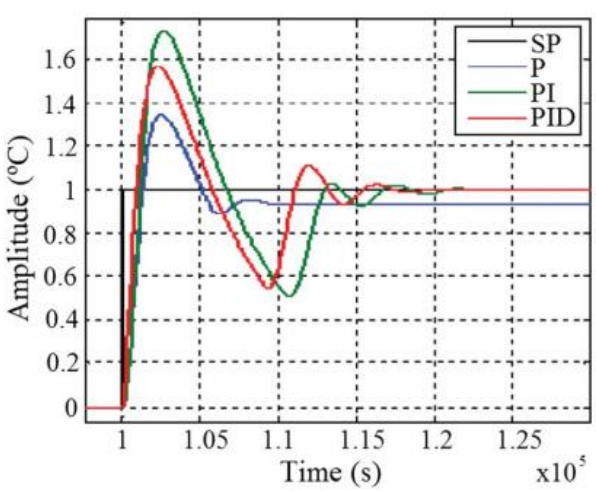


Fig. 12 Respuesta de proceso térmico con masa de aire con diferentes controladores

Fuente: [8]

2.6.4. Control Difuso

Se conoce como control difuso a todo controlador que trabaje con lógica difusa, este tipo de control moderno está basado en cómo se mira el mundo real, esta optimizado para sistemas MIMO (múltiple entrada y múltiple salida), además de ser un control no lineal, ofrece la posibilidad de interactuar directamente con los parámetros de control.

En este tipo de control se tiene muy poco o nulo conocimiento del modelo matemático de la planta a controlar, y su funcionamiento se basa en el conocimiento de un operador experto, incluso mediante la implementación de redes neuronales y lógica condicional.

La lógica difusa está basada en como vemos el mundo real, teniendo porcentajes de un nivel, como por ejemplo la humedad en la ropa: seca, un poco seca, un poco mojada o muy mojada. Los humanos solemos medir nuestro entorno por porcentajes y esta lógica lo hace de igual manera.

2.6.4.1. Conjuntos difusos

Para entender la lógica difusa es necesario conocer los conceptos de un conjunto difuso, un conjunto clásico está conformado por un nivel de pertenencia de verdadero o falso, en cambio para un conjunto difuso se tiene un grado proporcional de pertenencia, en la figura 13 se puede plasmar esta idea en referencia a una encuesta en cuando inicia el fin de semana y la comparativa entre un conjunto tradicional y uno difuso.

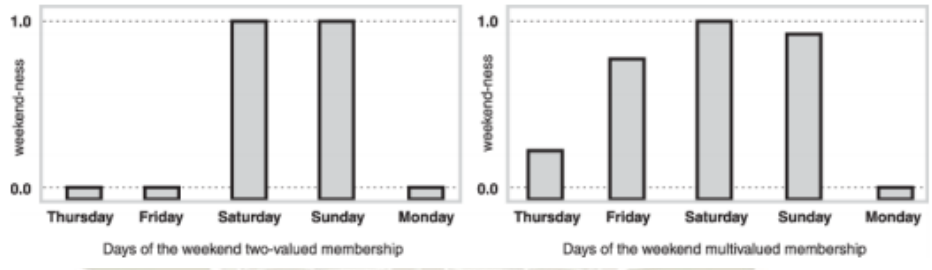


Fig. 13 Comparativa entre un conjunto clásico y uno difuso, Fuente [9]

Entonces, teniendo estos datos, se pueden expresar mediante una función matemática para determinar el nivel de membresía. Ahora podemos observar en la figura 14 que en una representación de tiempo continuo se tiene la representación gráfica de la siguiente manera:

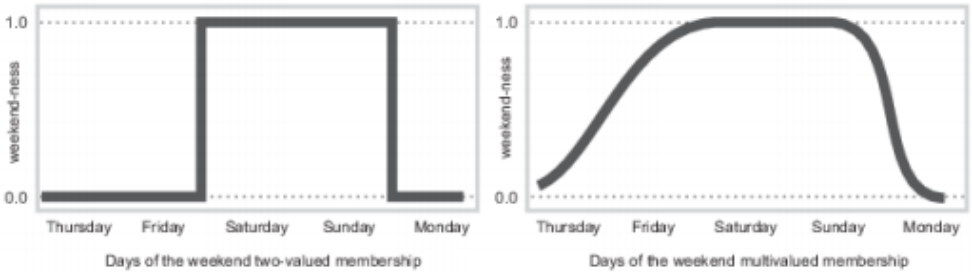


Fig. 14 Representación gráfica en tiempo continuo de un grupo clásico y uno difuso, Fuente [9]

Para la realización de una función de membresía, se tienen varios grupos entre los cuales se asigna nuestra variable de entrada, así como la forma de asociarlos. En la figura 15 tenemos como entrada la fecha del año actual y se aprecia como en la realidad se asigna a una etapa del año representado por un conjunto difuso a comparación del conjunto clásico.

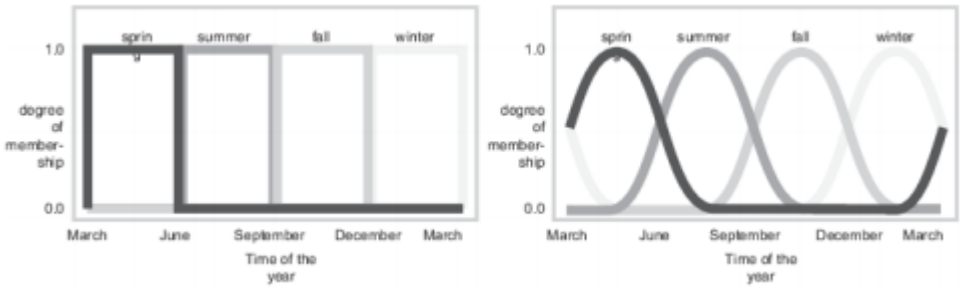


Fig. 15 Representación de la estación climática en conjuntos clásico y difuso, Fuente [9]

2.6.4.2. Funciones de membresía

En la figura 15 se puede observar a entrada con 4 funciones de membresía, estas funciones pueden estar representadas mediante funciones matemáticas y se muestran las más comunes en la figura 16.

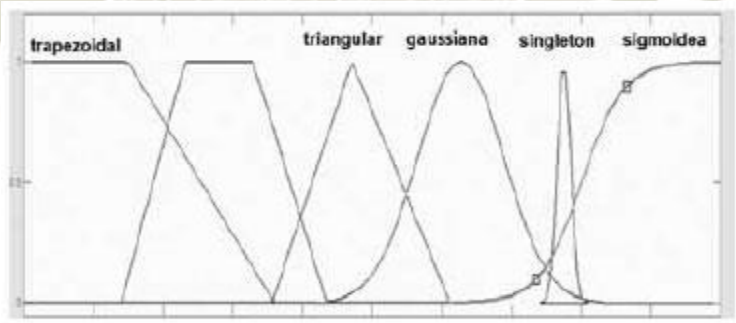


Fig. 16 Tipos de funciones de membresía Fuente [9]

Las funciones más simples de interpretar y e implementar son las basadas en rectas, como las trapezoidales y triangulares, mientras que las funciones gaussianas y sinodales se pueden implementar según los requerimientos del sistema, todo esto se define según el vago concepto que adquiera el valor de entrada y en que gráfica se complemente mejor su valor.

2.6.4.3. Reglas de control

Para las reglas de control, se tienen operadores lógicos como AND, OR, y NOT, en la figura 17 se muestran dichas reglas, siguiendo este concepto, en la lógica clásica se tiene un valor total al comparar 1 y 0, mientras que, en la lógica difusa, estos valores varían entre 0 y 1, para poder utilizar estos operadores se hace una analogía entre estos operadores y el concepto de máximo, mínimo y negado.

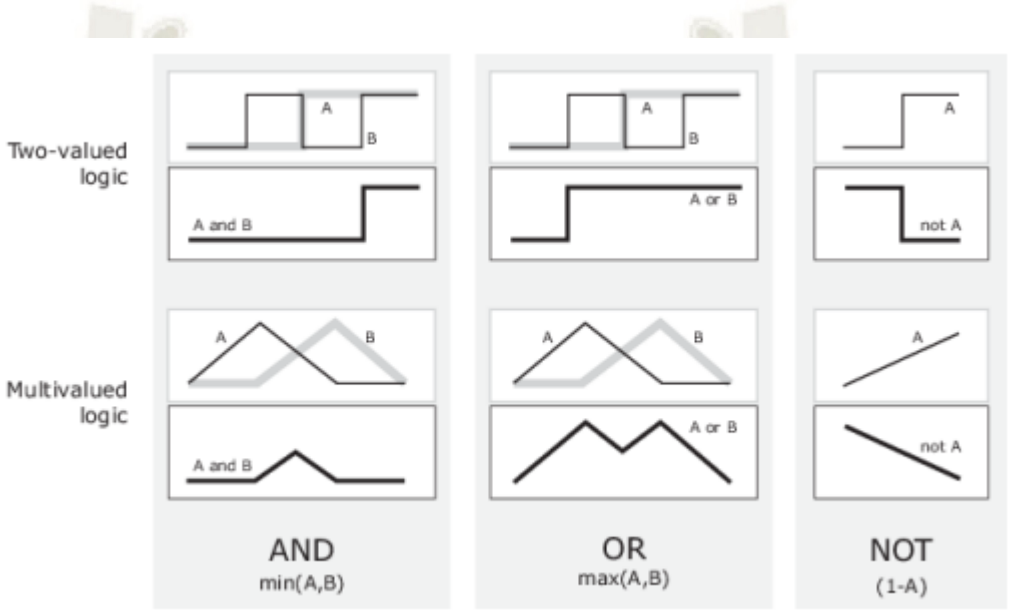


Fig. 17 Relación de conceptos de lógica clásica y difusa, Fuente [9]

Teniendo este concepto de los operadores lógicos, pasamos a ver cómo interactúan entre las funciones de membresía y la toma de decisiones, para esto se utilizan los conceptos de IF – THEN.

Teniendo un dato difuso, se aplica una regla condicional para determinar la acción a ejecutar, esto se logra utilizando los condicionales IF y THEN, por lo que se aplica para cada función de membresía a la que pertenezca el dato difuso.

En general, se requiere de la aplicación de varios condicionales para ejecutar un control preciso, por lo que la calidad del controlador depende de la capacidad de procesamiento y el número de condicionales que se consideren.

2.6.4.4. Proceso de control

Para el proceso de control se tienen definidas tres etapas, fuzzificación de las entradas, reglas de inferencia y defuzzificación de la salida, en la figura 18 se tiene el ejemplo de un proceso

difuso para la decisión del monto de propina en un restaurante basándose en la calidad de la comida y del servicio.

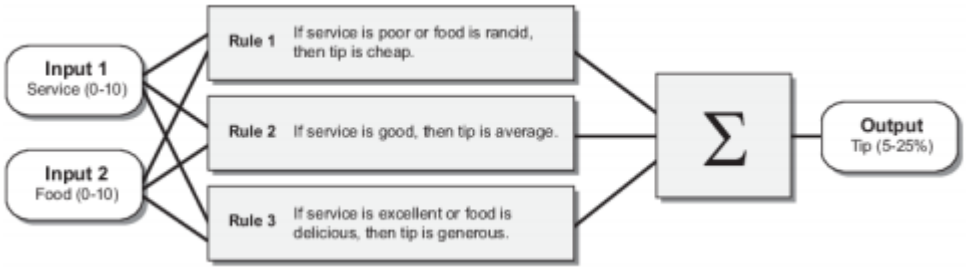


Fig. 18 Proceso difuso con 2 entradas, 3 reglas de inferencia y una salida.
Fuente [9]

Para llegar a un resultado no difuso, se muestra en la figura 19, el proceso de control que se toma, se tienen 3 funciones para la entrada 1, 2 para la segunda, 3 reglas y 3 de salida. Para la toma de decisión se eligió el operador OR o Max entre las dos entradas.

Como podemos observar, en el paso 1 se toman los datos del servicio y se evalúan en las primeras funciones de membresía, se puede ver que al pertenecer muy levemente a este grupo, se tiene un resultado bajo en el grupo de salida “cheap”.

En el paso 2, se tiene una integración mayor en el grupo de calidad de servicio, por lo que la salida en este caso es en el grupo “average” lo que indica que pertenece en mayor medida a este grupo hasta ahora, no se toma en cuenta el segundo dato de entrada debido a que no cuenta con una función asociada.

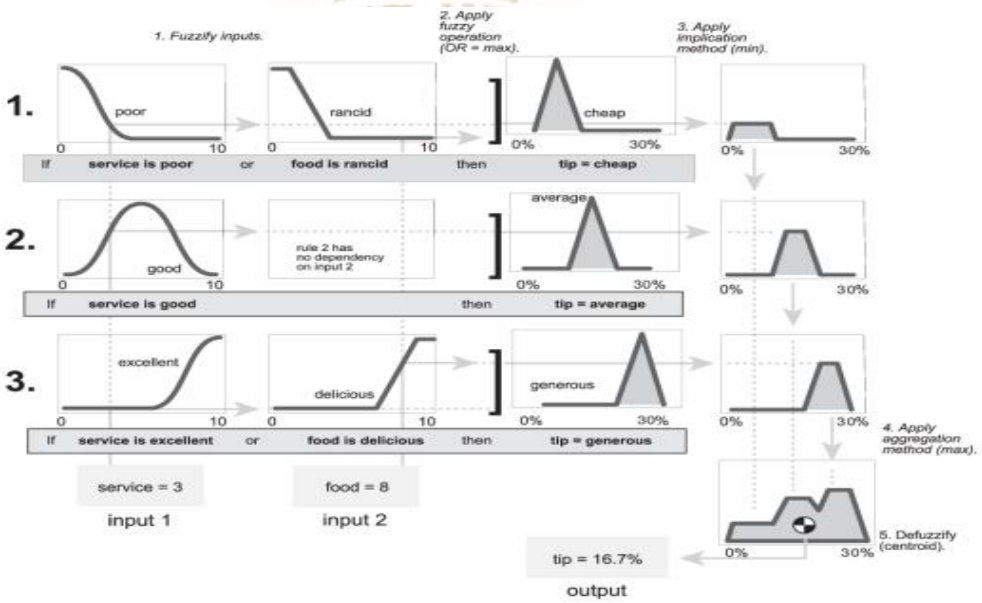


Fig. 19 Proceso de control del sistema difuso de ejemplo,
Fuente [9]

Para el tercer caso, se tiene que el servicio no pertenece al grupo de excelente, pero en la segunda entrada se tiene que la comida fue deliciosa, por lo que, al ser el operador Max entre ambas entradas, se eligió la pertenencia al tercer grupo de salida.

El resultado final se observa como una gráfica de la suma de las funciones anteriormente señaladas, por lo que se procede a la defuzzificación del proceso para poder obtener un resultado no difuso.

Para defuzzificar el resultado, se pueden aplicar diversos métodos, en el ejemplo se aplicó el método del centroide, que halla el centro de masa del grafico basándose en la suma sucesiva o integral de aproximación. Existen otros métodos como: bisector, medio del máximo, mayor y menor del máximo.

2.7. Electrónica de potencia

Para el control de equipos que requieran una gran cantidad de potencia, se tienen diseñados una gran variedad de dispositivos electrónicos, entre ellos los transistores y tiristores. Para este proyecto nos centramos en la descripción de estos últimos, en especial en su variante TRIAC.

2.7.1. Tiristores

Los tiristores son dispositivos que en su funcionamiento ideal trabajan como switches permitiendo el paso de corriente eléctrica, en el sistema real, estos dispositivos presentan una minúscula caída de voltaje. [10]

"Un tiristor es un dispositivo semiconductor con cuatro capas de estructura pnpn, con tres uniones pn. Tiene tres terminales; ánodo, cátodo y compuerta." [10]

Un punto a considerar, es que estos dispositivos pueden activar la conducción por las siguientes perturbaciones:

- **Térmica**, si la temperatura del tiristor sobrepasa el rango adecuado de funcionamiento, hay un aumento en la cantidad de pares electrón-hueco, que aumenta la corriente de fuga y consecuentemente genera la activación.
- **Luz,** un haz de luz directo sobre la oblea de silicio, genera el mismo efecto descrito anteriormente y desencadena en la activación del tiristor, cabe

mencionar que existen componentes como los MOC3021 que basan su funcionamiento en este principio.

- **dV/dT,** si es un valor elevado, la corriente de carga de las uniones capacitivas pueden activar el tiristor, asi mismo un valor alto de corriente puede dañarlo.

2.7.1.1. TRIAC

El tiristor de triodo bidireccional, puede conducir la corriente eléctrica en ambas polaridades, se utiliza comúnmente para el control por fase, como dos SCR conectados en anti paralelo. [10]

En la figura 20 se especifican sus terminales como MT1, MT2 y Gate, esto debido a que sus terminales no son únicamente ánodos o cátodos.

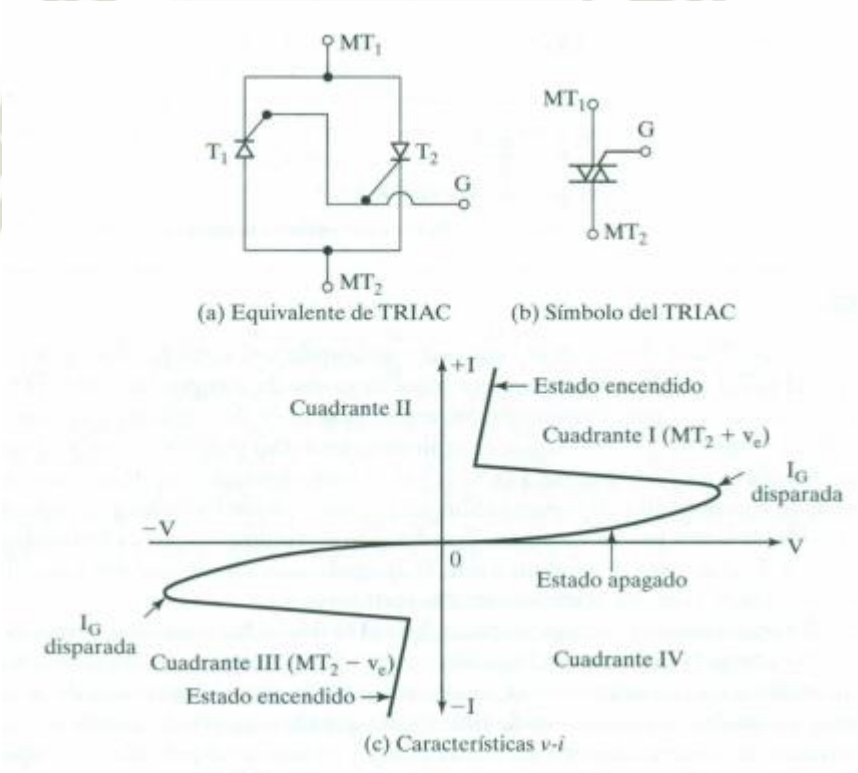


Fig. 20 Esquema de un TRIAC,
Fuente [10]

2.8.Dispositivos de interfaz Hombre-Maquina

Para poder controlar el proceso, se requiere de la interacción hombre – máquina, para esto se tiene una variedad de dispositivos capaces de mostrar datos importantes, como la variable de proceso, error, y rango de control, así como el estado general del sistema.

Para esto se muestran algunas de las interfaces disponibles para realizar este proceso, que a su vez funcionan como sistema supervisor de control y adquisición de datos, SCADA.

A diferencia de los dispositivos de interfaz, un HMI es capaz de controlar el equipo desde una sala de control, o de manera remota. Estos son normalmente implementados mediante software para aplicativos de PC o dispositivos móviles.

Según [3], un HMI debe constar de las siguientes partes:

- Configuración remota de la unidad de control en campo.
- Visualización de datos en tiempo real.
- Visualización de línea de tendencia del proceso.
- Capacidad para recopilar la data del proceso.

2.8.1. Dispositivos de interfaz

Son aquellos que permiten operar el dispositivo desde el punto de trabajo, consta de dispositivos de ingreso de datos como: teclados, interruptores, pulsadores, selectores, mandos de control, etc. Asi como dispositivos informativos como: pantallas LCD, indicadores luminosos, indicadores sonoros, etc.

Estos dispositivos permiten la libre operación del tablero de control en modo manual u offline, son esenciales en el diseño de un tablero de control. [3]

CAPITULO III
MODELAMIENTO DEL SISTEMA Y SIMULACIÓN

CAPITULO III

3. MODELAMIENTO DEL SISTEMA Y SIMULACIÓN

3.1. Diagrama de bloques del sistema

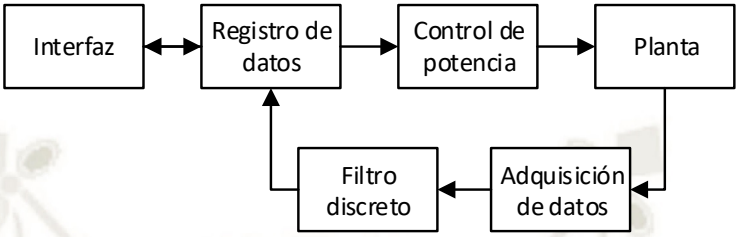


Fig. 21 Diagrama de bloques del sistema de adquisición de datos
Fuente: Elaboración propia

Como se observa en la figura 21, se tiene un sistema de adquisición de datos realimentado el cual, a diferencia de un controlador, no basa la salida de control en función de los datos obtenidos, únicamente establece un valor determinado de escalón para así obtener la respuesta del sistema y posterior procesamiento para exportar esta data y poder realizar un análisis matemático del mismo.

3.2. Adquisición de datos del sistema real

Para poder modelar el sistema real, vamos a utilizar la técnica de modelamiento basada en la respuesta del sistema.

Para esto necesitamos de un circuito de adquisición de datos, el cual se implementó basado en el microcontrolador PIC16F887 cuyo funcionamiento se basa en el diagrama de flujo mostrado en la figura 22, el sensor utilizado es una termocupla tipo K, en conjunto con el transductor IC MAX 6675, el cual entrega una señal de 16bits, serial síncrono bajo el protocolo SPI, con compensación de unión fría y detección de falso contacto de sensor.

Los datos serán almacenados en la memoria interna del microcontrolador y serán transmitidos con protocolo RS232 al CI HC06, el cual retransmitirá los datos basándose en el protocolo IEEE 802.15.

Los datos serán visualizados y procesados mediante el software MATLAB.

Para esto, se puede observar en la figura 22 el diagrama de flujo de este proceso.

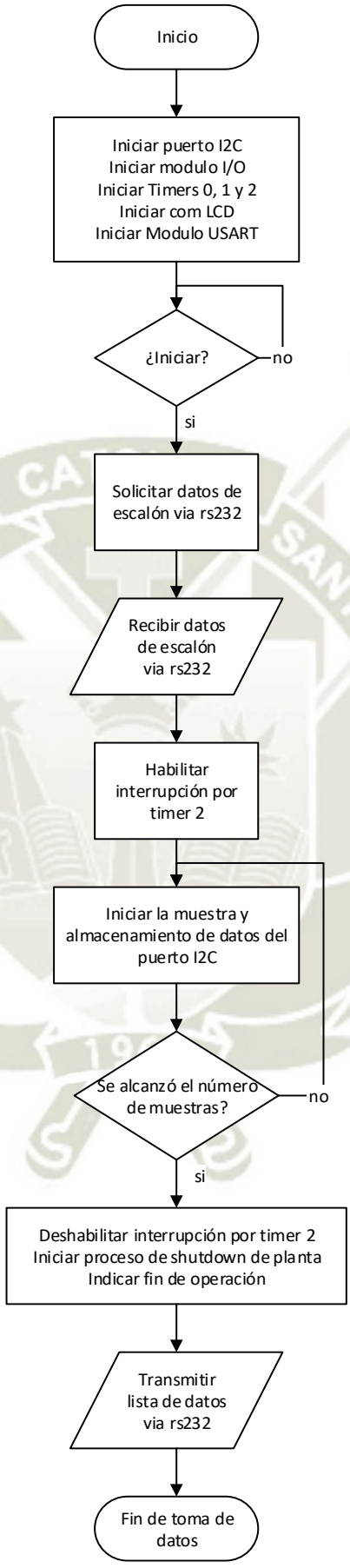


Fig. 22 Diagrama de flujo de toma de datos,
Fuente: Elaboración propia.

3.2.1. Filtrado de señal

Para el correcto modelamiento de un sistema, se deben tener instrumentos capaces de entregar información confiable, en este caso, un sensor de temperatura de calidad.

Para aumentar la confiabilidad y calidad de un sistema de control, es muy importante que cuente con sensores de alta exactitud y precisión, la primera se logra calibrando el instrumento, mientras que para mejorar la precisión es necesario un tratamiento digital de señales.

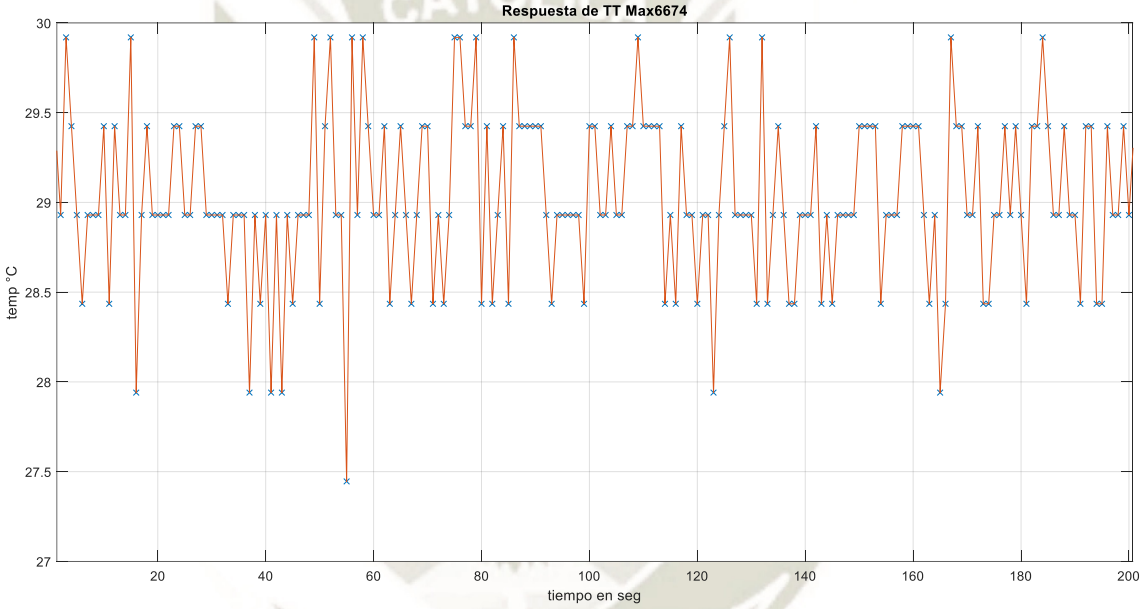


Fig. 23 Respuesta en el tiempo del transmisor de temperatura Max6674 sin filtrar,
Fuente: Elaboración propia

Como se observa en la figura 23, se tiene una gran perturbación que se expresa como la imprecisión del valor de temperatura actual, teniendo una variación de 1.2°C.

Tras realizar el análisis en frecuencia se obtiene la gráfica 24, en ella se muestra como componente principal una señal en 0.01 Hz, se visualizan además las perturbaciones no significantes, por lo que no se recomienda implementar un filtro digital ya que el costo computacional es demasiado elevado y no justifica la varianza.

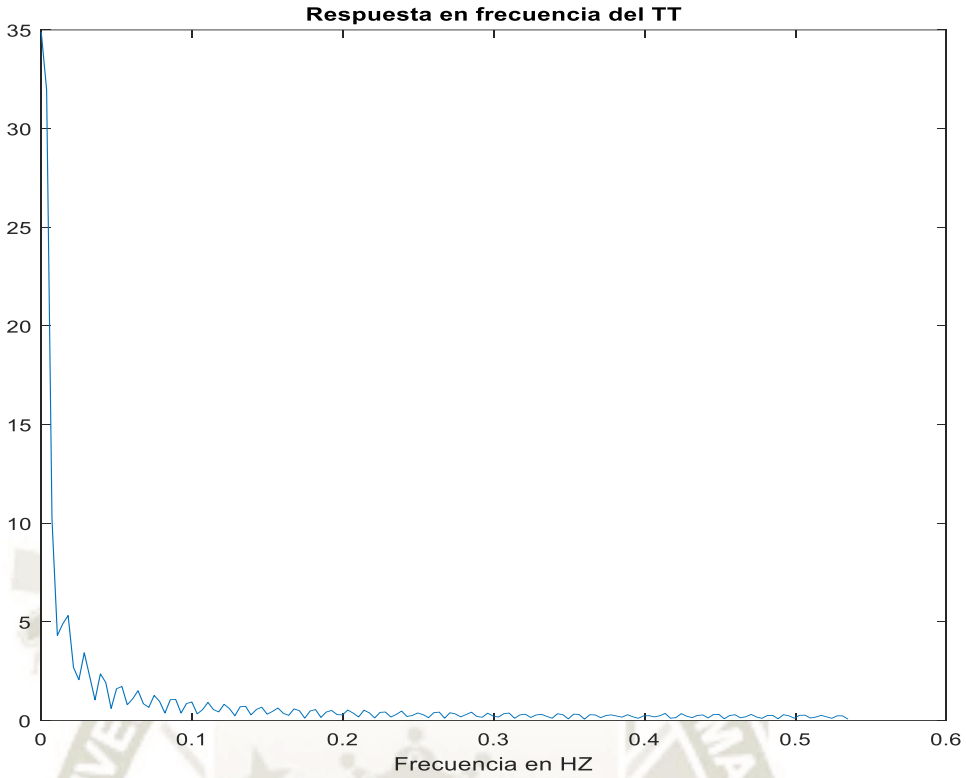


Fig. 24 Análisis en frecuencia del TT Max6674,
Fuente: Elaboración propia

Método estadístico de media móvil

La media móvil es una técnica estadística ampliamente usada en el tratamiento de datos, para filtrar nuestra señal, vamos a aplicar la media móvil exponencial, cuya ecuación se describe a continuación:

$$Vfiltrado = Vsensado * k + Vfiltrado_{anterior} * (1 - k)$$

Donde:

- Vfiltrado es el valor que entrega la media móvil.
- k es una constante que varía entre 0 y 1.
- Vsensado es el valor que entrega el TT sin procesar.

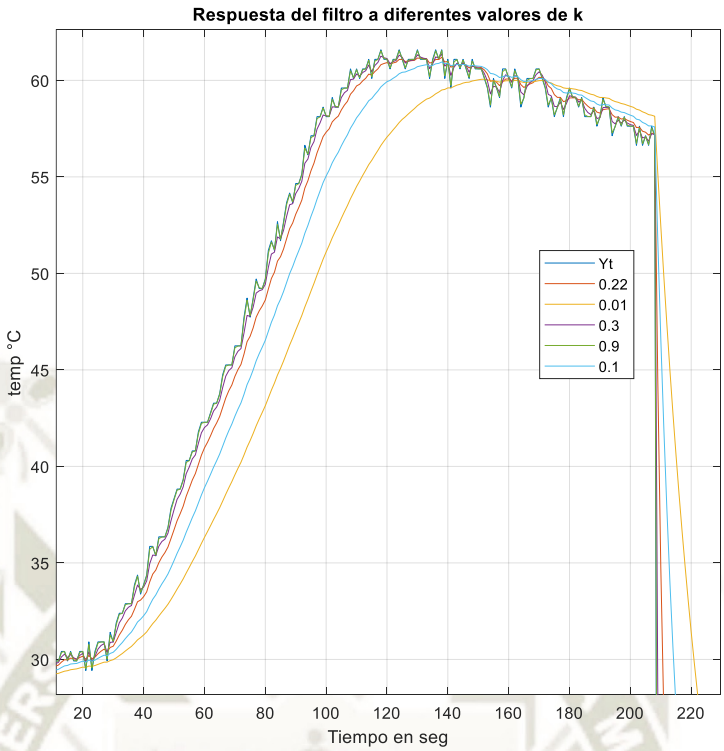


Fig. 25 Respuesta del TT para diferentes valores de k,
Fuente: Elaboración propia

Como se puede observar en la figura 25, mientras más alto sea el valor de k, la señal tiende a seguir la señal de entrada sin filtrar. Para este diseño se ha establecido un valor de k = 0.23, cuya respuesta se puede observar en la figura 26:

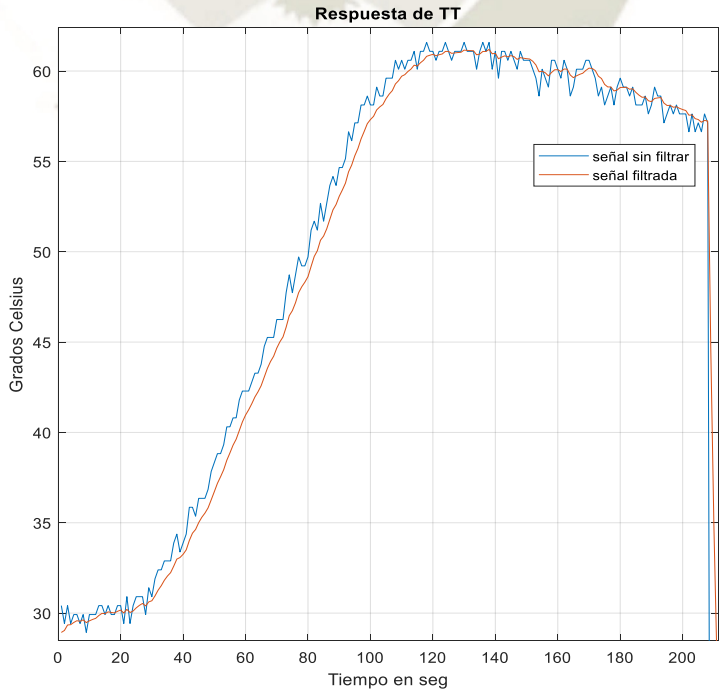


Fig. 26 Respuesta del TT para un valor de k=0.23,
Fuente: Elaboración propia

3.3.Modelado de sistema

El sistema térmico está descrito por ser un sistema de segundo orden sobre amortiguado, [6]

$$G(s) = \frac{{\omega_0}^2}{S^2 + 2\varepsilon\omega_0 S + {\omega_0}^2}$$

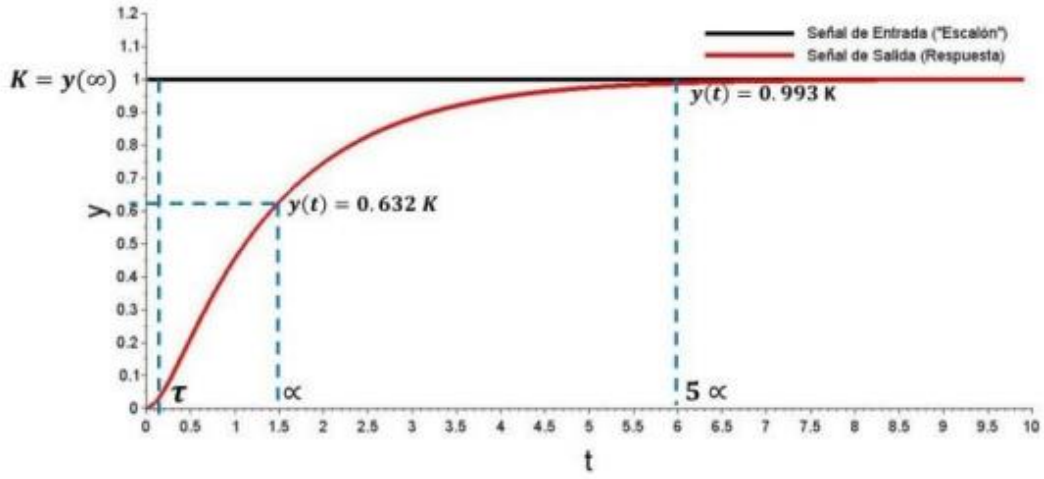


Fig. 27 Respuesta transitoria de un sistema de segundo orden sobre amortiguado,

Fuente [10]

Este sistema se describe como sobre amortiguado debido a que el coeficiente ε es mayor a 1 y se representa mediante la figura 27, se puede representar por un sistema de primer orden con retardo, descrito bajo la siguiente función de transferencia.

$$G(s) = \frac{K}{\propto S + 1} e^{-tS}$$

Donde:

- t representa al tiempo de retardo.
- K la ganancia del sistema.
- $\propto$ el valor de 0.632K.

Entonces aplicando esta ecuación en nuestra gráfica representada en la figura 28 obtenemos los valores necesarios para modelar el sistema asimilándolo a un sistema de primer orden con retardo.

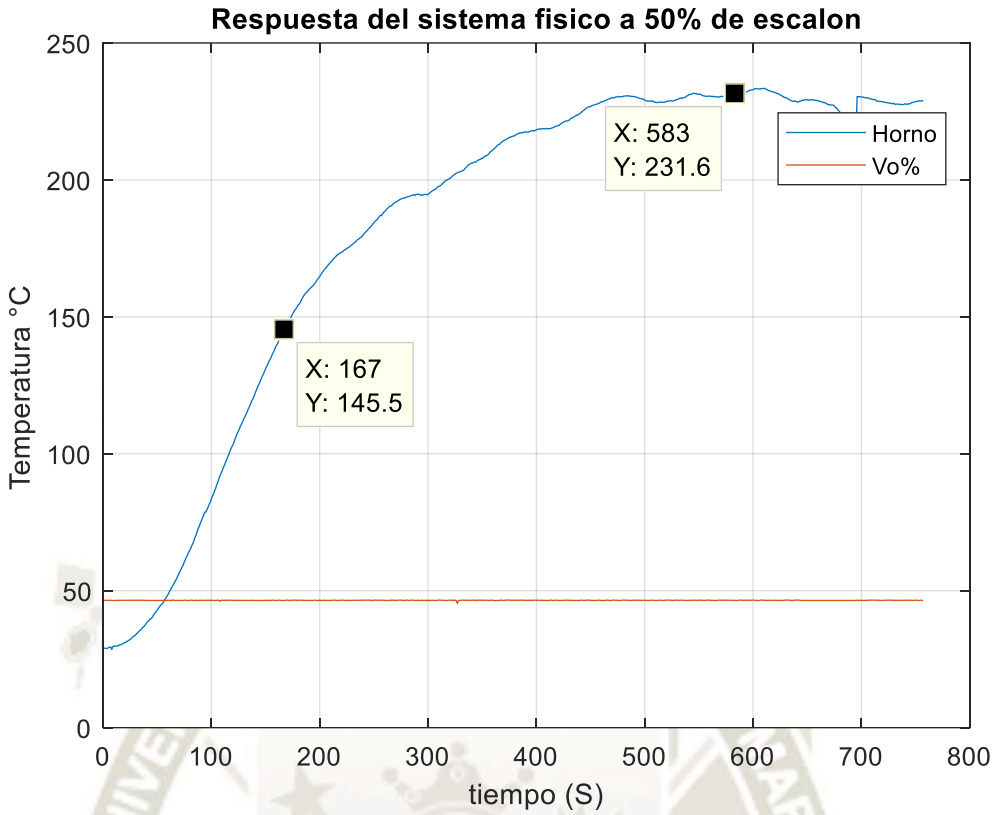


Fig. 28 Respuesta transitoria de nuestro horno que se asemeja a un sistema de primer orden con retardo, Fuente: Elaboración propia

A continuación, reemplazamos los datos obtenidos en nuestro modelo de función de transferencia y establecemos la función con los siguientes valores:

$$K = 5.23$$

$$\propto = 167$$

$$t = 42$$

$$G(s) = \frac{K}{\propto S + 1} e^{-tS} = \frac{5.23e^{-42S}}{167S + 1}$$

Comparamos así el modelo de nuestra función de transferencia con el modelo obtenido mediante el toolbox embebido en Matlab mostrado en la figura 29.

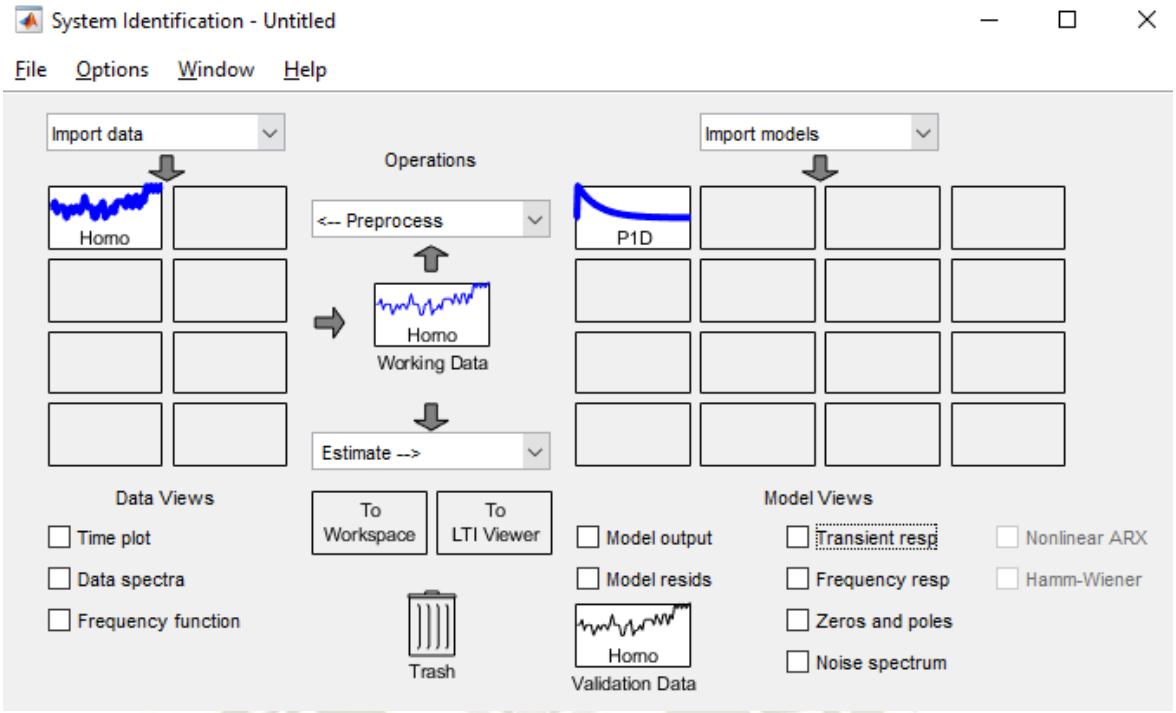


Fig. 29 System Identification, toolbox de Matlab en prueba de datos,
Fuente: Elaboración propia

Por lo tanto, obtenemos la siguiente función de transferencia:

Process model with transfer function:

$$G(s) = \frac{Kp}{1+Tp1*s} * \exp(-Td*s)$$

Kp = 5.1684
Tp1 = 170.59
Td = 30

Como podemos observar, la función obtenida mediante el método experimental y con el toolbox de Matlab, son similares. Para el diseño de simulación se utilizó el modelo obtenido mediante la aplicación model identification por ser más precisa.

Además, se realizó el modelo en dominio del tiempo utilizando el Toolbox, se procede a comparar la respuesta de estos dos sistemas cuya respuesta se muestra en la figura 30.

tf1 =

$$\frac{0.013\ s + 0.000854}{s^2 + 0.02865\ s + 0.00017}$$

Name: tf1
Continuous-time identified transfer function.

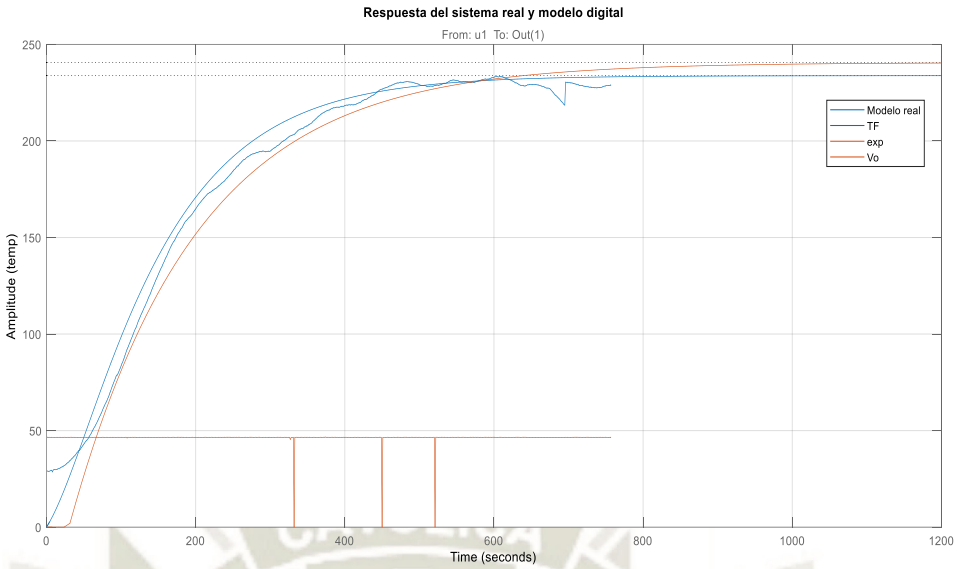


Figura 30 Respuesta escalón del modelo real y sistema simulado
Fuente: Elaboración propia

3.4. Sintonización de sistema para control PI

Para esto nos basamos en el método de sintonización de la primera regla de Ziegler-Nichols, donde se explica la relación entre la respuesta transitoria de un sistema a una entrada escalón y los valores obtenidos para determinar la ubicación de los polos y ceros de la función de transferencia del controlador.

"Este método se puede aplicar si la respuesta muestra una curva con forma de S. Tales curvas de respuesta escalón se pueden generar experimentalmente o a partir de una simulación dinámica de la planta." [6].

Por lo que podemos proceder con la sintonización a partir de los datos obtenidos previamente. En la siguiente tabla se muestran los valores recomendados para utilizar en la ecuación de nuestro controlador.

Tabla 1 Reglas de sintonía de Ziegler-Nichols basados en la respuesta escalón de planta, Fuente [6]

Tipo de controlador	Kp	Ti	Td
P	$\frac{T}{L}$	inf	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Para un ajuste en el uso de variables, en nuestra ecuación 2 se tiene:

$$G(s) = \frac{K}{\propto S + 1} e^{-tS} = \frac{5.23e^{-42S}}{167S + 1} \quad (2)$$

Por lo que para una correcta utilización de las reglas de sintonía de la tabla 1, se declara la similitud del sistema en:

$$G(s) = \frac{K}{\propto S + 1} e^{-tS} = \frac{K}{TS + 1} e^{-LS} \quad (3)$$

Entonces el controlador C, será:

$$C(S) = Kp * \left(1 + \frac{1}{TiS} + TdS\right) \quad (4)$$

Aplicando las reglas de la tabla 1 tenemos:

$$C(S) = 1.2\frac{T}{L} * \left(1 + \frac{1}{2LS} + 0.5LS\right)$$

$$C(S) = 0.6T * \frac{\left(S + \frac{1}{L}\right)^2}{S} \quad (5)$$

3.5.Diseño y simulación del sistema con controlador PI

Para esta sección vamos a utilizar el entorno de simulación de Matlab, Simulink. Por lo que procedemos a declarar los bloques de control preestablecidos en las ecuaciones 2 y 5

De la ecuación 2, obtenemos la función de transferencia en tiempo continuo de nuestro horno:

$$G(s) = \frac{5.23e^{-42S}}{167S + 1}$$

De la ecuación 5, obtenemos el modelo de nuestro controlador PID, el cual, al reemplazar los datos, obtenemos:

$$C(s) = 0.6T * \frac{\left(s + \frac{1}{L}\right)^2}{s} \quad (6)$$

$$C(s) = 0.6 * 167 * \frac{\left(s + \frac{1}{42}\right)^2}{s}$$

$$C(s) = 100.2 * \frac{(s + 0.024)^2}{s} \quad (7)$$

Siendo un sistema térmico, lo recomendable es utilizar un control PI, por lo que procedemos a calcular el controlador sin la parte derivativa.

$$C(S) = Kp * \left(1 + \frac{1}{TiS}\right) \quad (8)$$

$$C(S) = 0.9\frac{T}{L} * \left(1 + \frac{1}{\frac{L}{0.3}S}\right)$$

$$C(S) = 0.9\frac{167}{42} * \left(1 + \frac{1}{\frac{42}{0.3}S}\right)$$

$$C(s) = 3.58 + \frac{0.0256}{S} \quad (9)$$

Utilizando el entorno de Simulink, declaramos las funciones y analizamos la respuesta transitoria del sistema para una entrada escalón que se muestra en la figura 31.

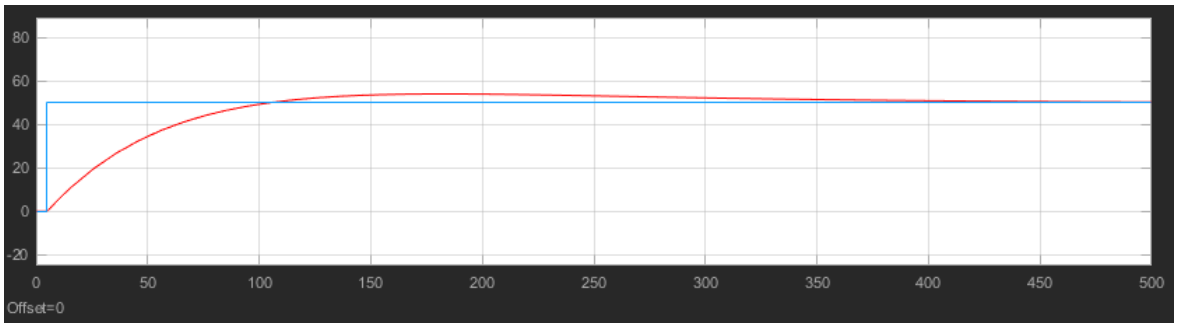


Fig. 31 Respuesta del sistema digital con controlador PI en simulink
Fuente: Elaboración propia

Como se puede observar en la figura 32, se tiene un sistema de segundo orden amortiguado, a continuación, procedemos a hallar sus parámetros:

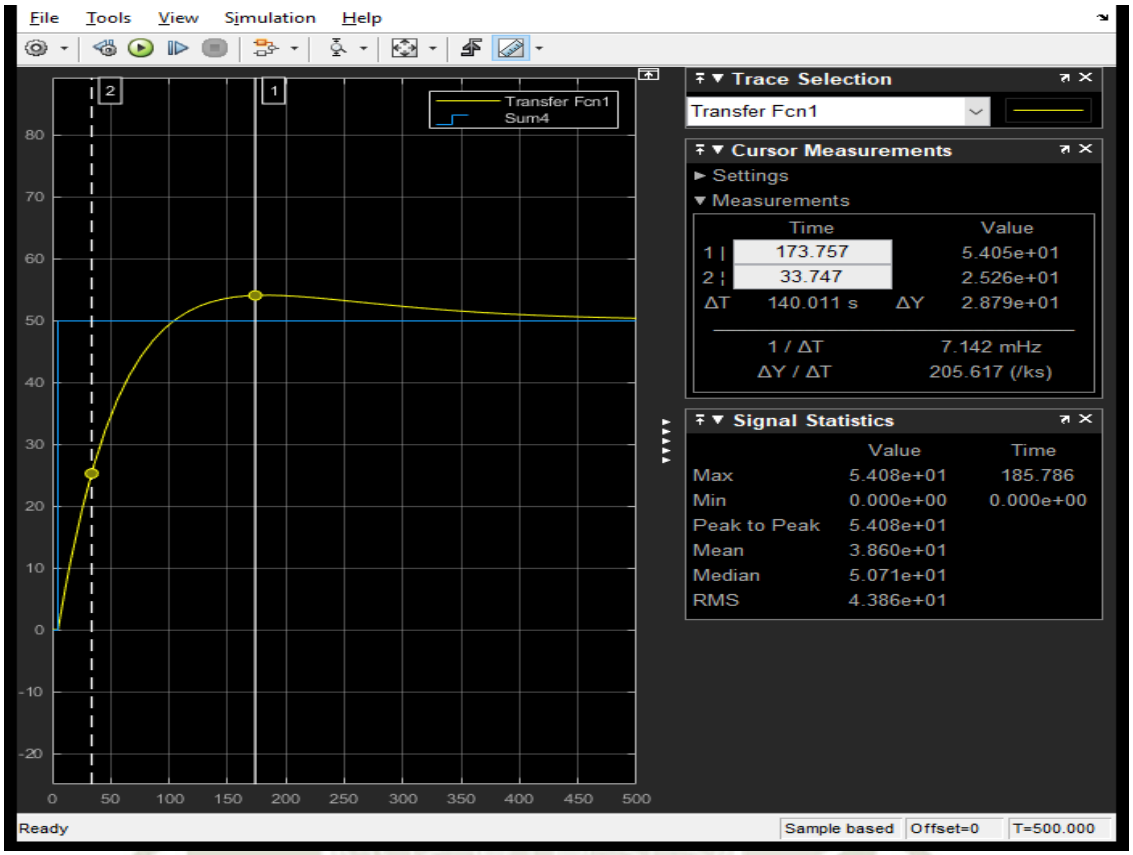


Fig. 32 Análisis de respuesta transitoria en simulink
Fuente: Elaboración propia

Tabla 2 Valores de respuesta con controlador PI

Parámetro	Fórmula	Valor
Mp	$Mp = 100 * \frac{Vmax - Vss}{Vss}$	8%
Tp	$Tp = \frac{\pi}{\omega n\sqrt{1-\varepsilon^2}}$	185s
Ts (5%)	$Ts = \frac{3}{\varepsilon\omega n}$	284s
Ts (2%)	$Ts = \frac{4}{\varepsilon\omega n}$	396s
Td	Según gráfica	33s

Fuente: Elaboración propia

3.6.Diseño y simulación del sistema con controlador difuso

Para el diseño del controlador difuso en Matlab, vamos a utilizar la herramienta embebida Fuzzy Logic Designer, mostrado en la figura 33, mediante la cual procedemos a declarar nuestras entradas, reglas de control y salida. Para este diseño se ha tomado en cuenta las siguientes características:

- 2 entradas: Error y Derivada del error.
- Una salida de control.
- Rango de entrada Error: -100 a 100°C.
- Rango de entrada en derivada de error: -10 a 10.
- Reglas de inferencia de tipo AND.
- Inferencia de control: Control Difuso tipo Sugeno.
- Técnica de defuzzificación: Valor promedio entre 5 conjuntos con valor constante.

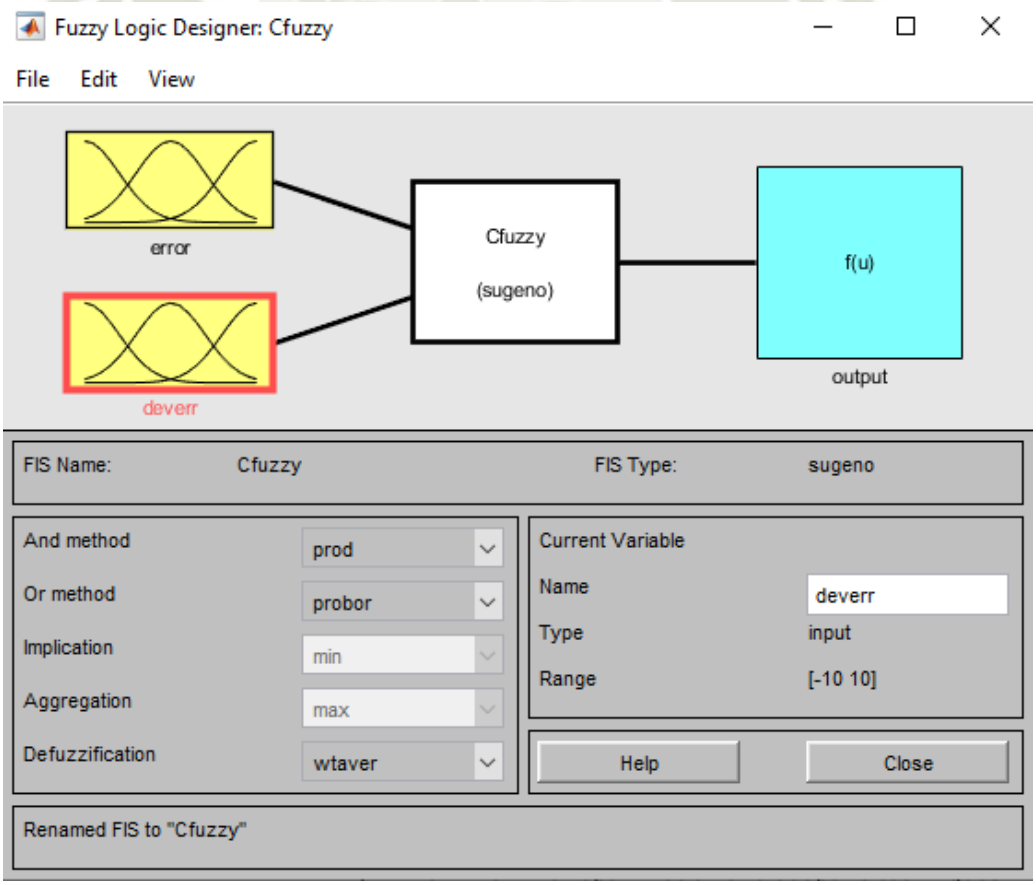


Fig. 33 Menú de Fuzzy logic Designer del controlador Cfuzzy,
Fuente: Elaboración propia

En la figura 34 mostraremos los conjuntos difusos de error, sobre los cuales se realiza la fuzzificación de los datos, como podemos observar, tenemos 5 conjuntos en la entrada error, que se muestran como muy negativo, negativo, mínimo, positivo y muy positivo, como se puede observar, el rango del conjunto mínimo, como su nombre lo indica, es muy pequeño en comparación con los otros dos conjuntos, por lo que se define que el error del controlador es proporcional al rango de este conjunto.

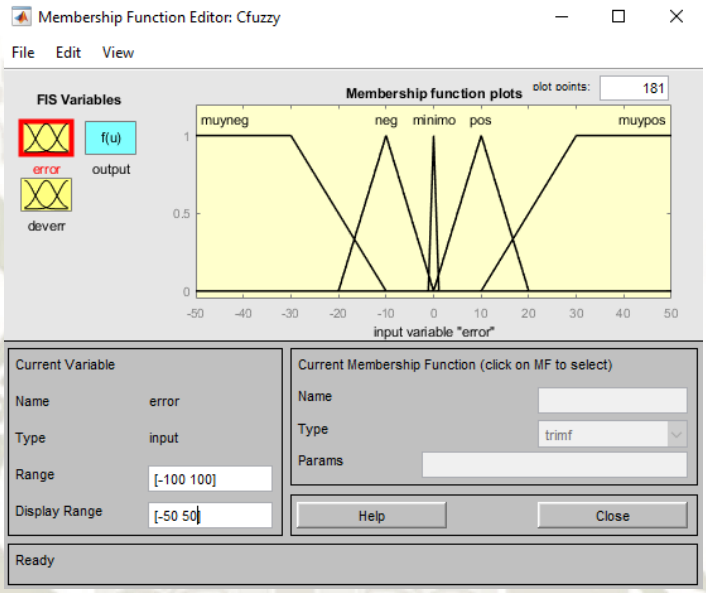


Fig. 34 Conjuntos difusos para la entrada error del controlador Cfuzzy,
Fuente: Elaboración propia

Por otro lado, en la figura 35 observamos únicamente 3 conjuntos, dos de los cuales con forma trapezoidal y como forma central una figura triangular, facilitando así el procesamiento matemático.

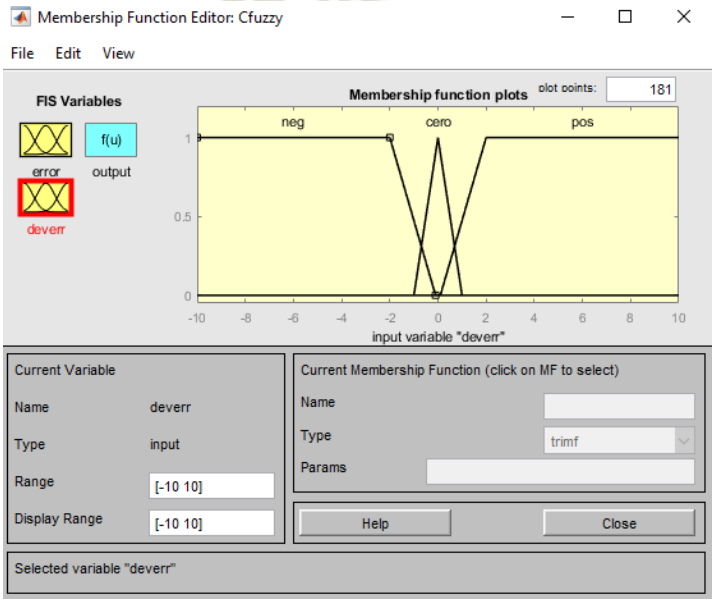


Fig. 35 Conjuntos difusos de la entrada deverr,
Fuente: Elaboración propia

Una vez que hemos definido nuestras entradas, se procede con la implementación de las reglas de control, estas están basadas en el operador lógico AND y condicionales de uso. Para poder observar mejor, contamos con el asistente del fuzzy logic designer, rule viewer que se muestra en la figura 36.

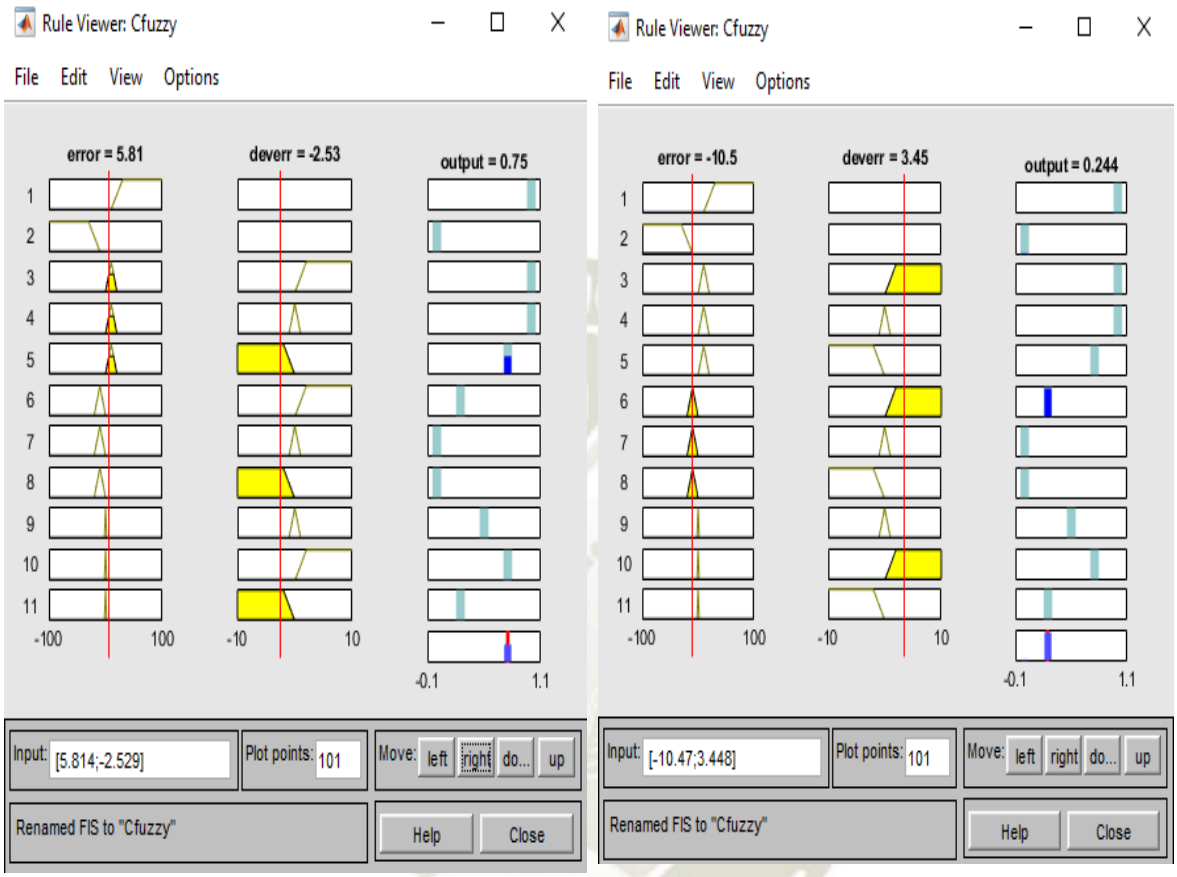


Fig. 36 Comparativa de las reglas de control con valores de entrada diferentes
Fuente: Elaboración propia

Para poder comprender las reglas de control, se tiene una gráfica tridimensional donde se muestran los valores de entrada y salida, para esto utilizaremos el Surface Viewer que se puede apreciar en la figura 37, como se puede observar, muestra un comportamiento de control no lineal. Se puede observar que el control en los extremos representa un valor constante, sea salida máxima o mínima según el caso, el cual se mantiene mientras el valor de salida (error) sea muy diferente al setpoint establecido. Así mismo se observa que el rango de control se establece en los valores cercanos a 0.

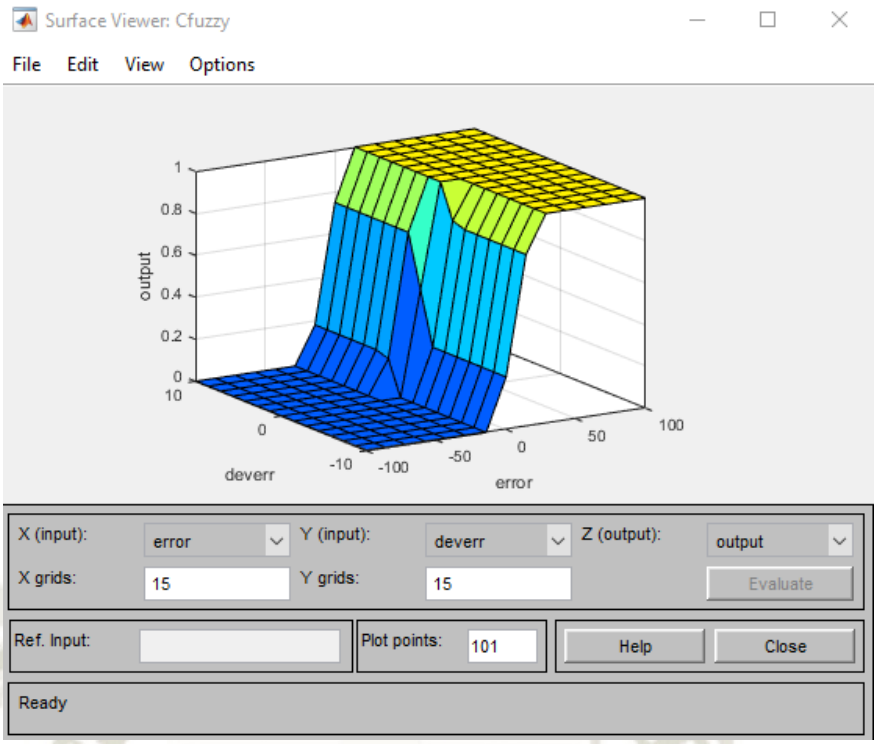


Fig. 37 Surface del controlador Cfuzzy,
Fuente: Elaboración propia

Para poner a prueba este controlador, se utilizó el entorno Simulink, a continuación, en la figura 38 se muestra el diagrama de bloques implementado en donde se muestran dos señales de setpoint establecidas en tiempos distintos, así como la implementación de la derivación y adaptación de señal para las entradas difusas y dos señales de interferencia en diferentes partes del proceso.

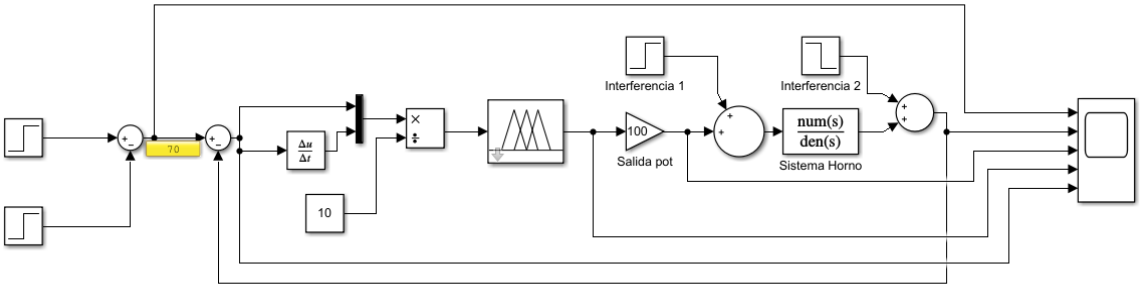


Fig. 38 Diagrama de bloques del sistema térmico con controlador difuso y perturbaciones
Fuente: Elaboración propia

La prueba de simulación se realizó bajo los siguientes parámetros:

- Setpoint inicial: 100°C en t = 2.
- Cambio de setpoint: 70°C en t = 300.
- Dos señales de interferencia con pico positivo y negativo en t = 500 y t = 600.

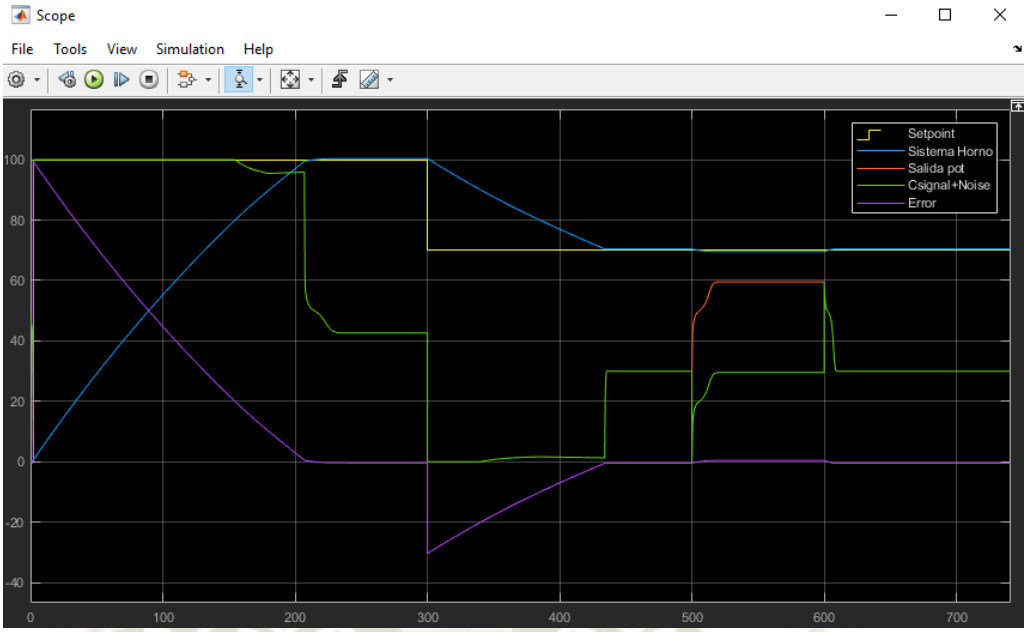


Fig. 39 Respuesta transitoria del sistema térmico con controlador difuso
Fuente: Elaboración propia

Como se puede observar en la figura 39, el sistema responde de manera exitosa y eficiente tanto a cambios en el setpoint como a perturbaciones, la respuesta del controlador es en todo momento precisa, sin sobre impulsos por otro lado, el error en tiempo estacionario tiende a 0.

En una segunda prueba, mostrada en la figura 40, se tiene un cambio brusco de temperatura de cámara, el sistema responde de manera eficiente igualmente, implementando la salida de control óptima para mantener el setpoint.

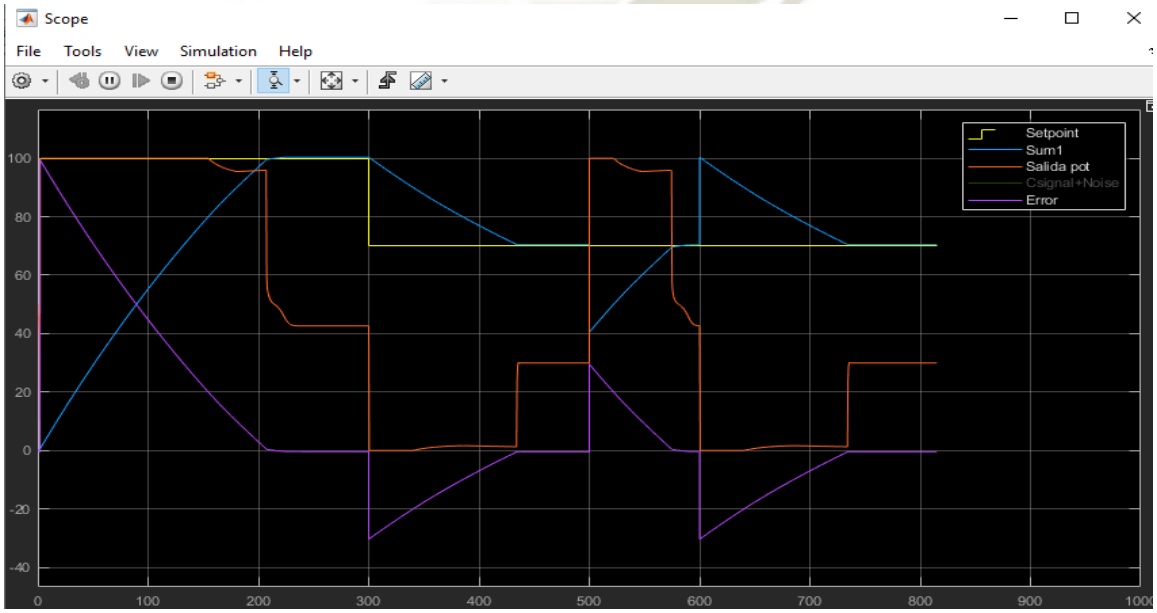


Fig. 40 Respuesta transitoria del sistema térmico con interferencia directa en temperatura de cámara y controlador difuso
Fuente: Elaboración propia

CAPITULO IV
DISEÑO E IMPLEMENTACIÓN DE CONTROL

CAPITULO IV

4. DISEÑO E IMPLEMENTACIÓN

En este capítulo de diseño, se toman en cuenta las condiciones físicas donde sobre las que actuará nuestro controlador, separamos el diseño por etapas y se implementará en un gabinete de control. El comportamiento del sistema esta descrito en el diagrama de bloques mostrado en la figura 41, la misma que se encuentra amplificada en la sección de anexos.

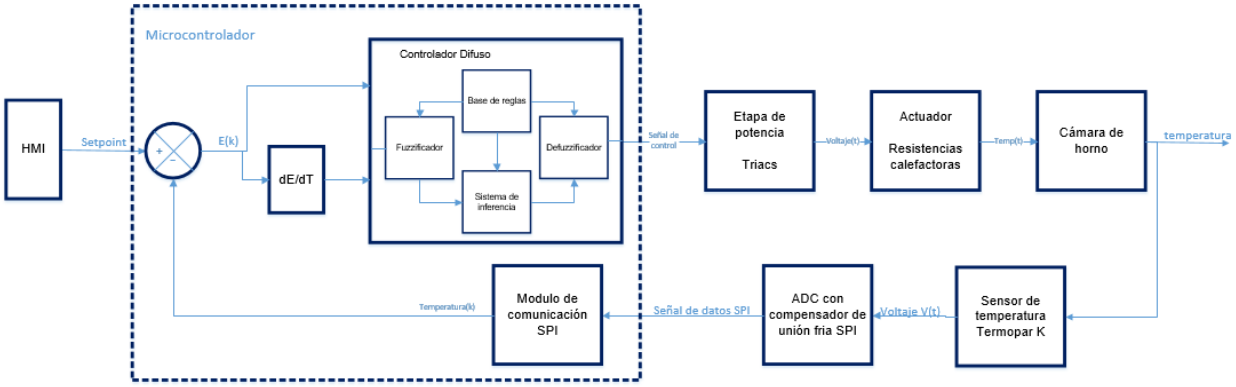


Fig. 41 Diagrama de bloques del proceso de control difuso,

Fuente: Elaboración propia

A continuación, se desarrollan las etapas de manera independiente, cabe resaltar que, durante el desarrollo de este control, se implementará un control alternativo PI con el fin de comparar resultados y demostrar la optimización de un control moderno ante una técnica de control clásica, para esto, lo único que cambia en el diagrama mostrado en la figura 41 es el bloque de control difuso y derivada del error, que ambos serán reemplazados por el bloque de control PI discreto.

Restricciones iniciales

- Sistema modular basado en 2 microcontroladores de 8 bits.
- Placa de potencia de 3 SSR independientes a 12 Amperes, 220V.
- Establecer un sistema de control con menos de 2% de sobre impulso y con un tiempo de respuesta menor a 180 segundos.

4.1 Diseño de placa de control

Para el diseño del controlador primero necesitamos elegir un dispositivo de control que se adapte a nuestras necesidades. Para esto primero enumeramos en la tabla 2 la lista de entradas y salidas necesarias, el propósito de cada una y el número mínimo de GPIO del controlador, módulos de comunicaciones y velocidad de procesamiento.

Tabla 3 Descripción de I/O del sistema de control

Descripción de entradas y salidas mínimas del sistema de control					
Entradas	**9**	**Salidas**	**16**	**Comunicaciones**	**5**
Descripción	número	Descripción	número	Descripción	número
IntRB0	1	LCD	7	SPI	3
Start/Stop	1	Vo1	1	RS232	2
Sel/remo	1	Vo2	1		
Sel/PID/FU	1	Vo3	1		
Sel/vent	1	Luz/cam	1	**Mínimo total:**	**30**
Sel/Rtop	1	Vent	1		
SensorPuert	1	LedRem	1		
Pot /AN0	1	LedPID/FU	1		
LM35/AN1	1	LedVent	1		
		LedRtop	1		

Fuente: Elaboración propia

Teniendo en consideración el mercado local, y a fin de optimizar el sistema, se ha decidido la implementación de un segundo procesador, esta configuración dual-core funciona como un sistema con control redundante de seguridad, garantizando la operatividad y correcto funcionamiento del sistema en general.

Por lo tanto, se diseña un microcontrolador que se encarga del procesamiento de la variable del proceso, algoritmo de control y comunicación, y otro microcontrolador encargado del tratamiento y análisis de datos, señales I/O, y la interacción con los dispositivos HMI, ambos trabajaran en una configuración de Master-Slave basada en el protocolo I2C

Para elegir el controlador primario, se va a necesitar un dispositivo con la cantidad de memoria necesaria para almacenar las instrucciones de control y etapas del algoritmo del controlador difuso, debe de disponer de módulos Timer capaces de entregar una resolución de escalón mínima para la implementación de control y la velocidad de procesamiento para realizar todo el algoritmo de control dentro del tiempo de muestreo y control.

Por lo tanto, tenemos los siguientes requerimientos para el controlador maestro:

- Tiempo de muestreo: 1s.
- Comunicación: SPI, I2C, UART.
- Timers disponibles: 3.
- GPIO: 8.

Mientras tanto para el controlador esclavo necesitamos:

- Módulo ADC: 10 bits.
- Comunicación: I2C. UART.
- Puerto de interrupción en cambio de estado.
- GPIO: 16.

Para la selección del controlador se procede a calcular la resolución de la señal de control, siendo esta proporcional a la resolución de los Timer hacemos el cálculo en base a una frecuencia de trabajo de 60Hz y una velocidad de operación mínima de 20MHz.

$$T = \frac{1}{F} \quad (10)$$

$$T_1 = \frac{1}{60} = 16.6\, mS$$

$$T_2 = \frac{T_1}{2} = \mathbf{8.3\ mS} \quad (11)$$

Donde:

- T representa el periodo.
- T_1 el periodo de una onda completa de 60Hz.
- T_2 representa el hemiciclo de control sobre el que actuará la etapa de control explicada en el capítulo 4.2.

Teniendo este dato, debemos seleccionar un desborde de Timer en un periodo no menor a 8.3mS para asegurar un rango de control en todo el espectro angular. En la figura 42 se puede observar el diagrama de bloques de un Timer 0 de la gama media 16F, donde se puede comprender mejor el funcionamiento de este módulo embebido.

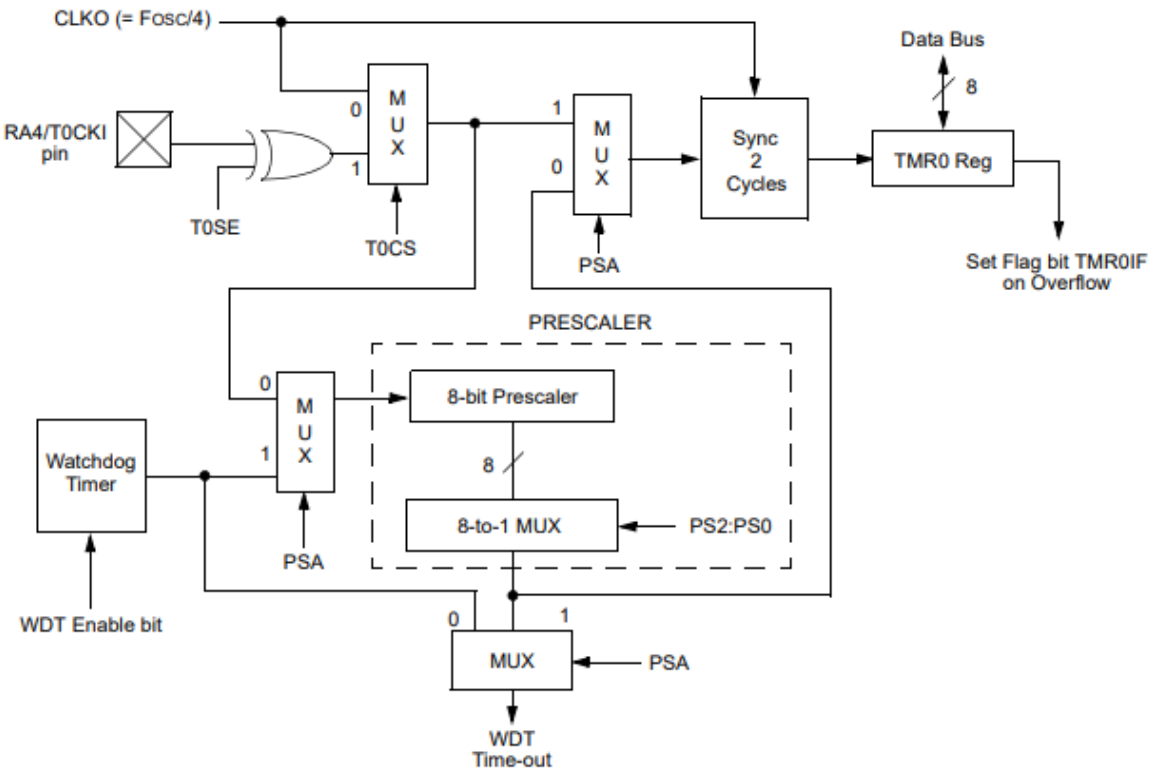


Fig. 42 Diagrama de bloques del Timer0 en la gama 16F, Fuente: [12]

Como se puede observar, el comportamiento se basa en la configuración del registro OPTION_REG, específicamente en los valores que se establezcan los datos de T0CS, T0SE, PSA y PS2:PS0, tomando en cuenta una configuración basada en alimentación via CLKO sin WDT, se tiene como resolución mínima de trabajo:

$$CLKO = \frac{Fosc}{4} = \frac{20MHz}{4} = 5MHz = 0.2uS \ (12)$$

$$TMR0\ Overflow = CLKO * 2^8 = 0.2nS * 256 = 51.2uS \ (13)$$

Como podemos observar, se necesita configurar el prescaler para alcanzar el minimo de 8mS, por lo tanto, con la configuración de prescaler = 1:256, recién se logra sobrepasar el tiempo mínimo. Procedemos entonces a calcular la resolución y el tiempo exacto.

$$TMR0\ Overflow = CLKO * 256 * 2^8 = \ 13.1mS$$

$$TMR0\ res = \frac{TMR0\ Overflow}{2^8} = 51.2uS \ (14)$$

Para calcular el número de pasos disponibles con una resolución de 51,2uS procedemos con la siguiente ecuación:

$$n\ Pasos = \frac{8.3mS}{51.2uS} = 162\ pasos\ (15)$$

Esta resolución satisface los requisitos de control de manera muy limitada que se demuestra en la etapa de potencia, por lo que se opta por implementar un controlador con al menos 3 timers con resolución de 16 bits como mínimo, asegurando así la calidad y confiabilidad del sistema.

Basándonos en este requisito, se opta por elegir la gama 18F, al ser la gama media mejorada, tiene mejores prestaciones y mantiene un costo relativamente bajo frente a las gamas superiores.

En esta gama tenemos el PIC18F2550, que es un dispositivo popular en el mercado local y entre sus prestaciones obtenemos una velocidad de procesamiento de hasta 12MIPS a una velocidad de reloj de 48MHz, 4 Timer, 3 de ellos de 16 bits y los módulos de comunicación necesarios para nuestro proyecto.

Procedemos a realizar el cálculo de resolución y el número de pasos como parámetro de diseño en la etapa de potencia. La figura 43 muestra el diagrama lógico interno.

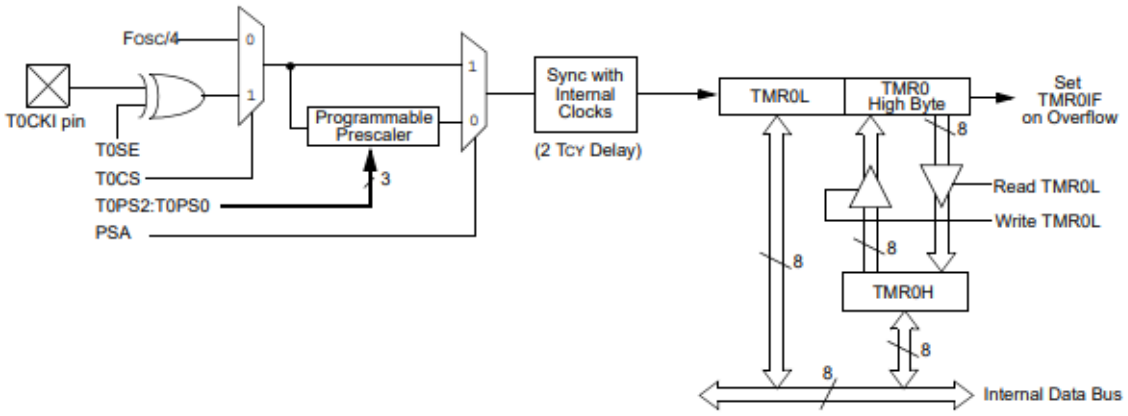


Fig. 43 Diagrama de bloques del TMR0 de 16 bits,
Fuente [12]

$$CLKO = \frac{Fosc}{4} = \frac{48MHz}{4} = 12MHz = 83.3nS\ (16)$$

$$TMR0\ Overflow = CLKO * Prescaler * 2^{16}$$

$$TMR0\ Overflow = \ 83.3nS * 2 * 2^{16} = \ 10.9mS\ (17)$$

$$TMR0\ res = \frac{TMR0\ Overflow}{2^{16}} = 166nS\ (18)$$

$$n\ Pasos = \frac{8.3mS}{166nS} = 49800\ pasos \quad (19)$$

Como podemos observar, en esta configuración se cumple a la perfección los requisitos para el controlador principal, para el controlador esclavo, se plantea utilizar el microcontrolador PIC16F887, que es un dispositivo potente y comercial que cumple con los requisitos dados para el manejo de los dispositivos HMI.

Continuando con el diseño de la etapa de la placa de control, se tiene en la figura 44 el diagrama esquemático donde se muestra la interacción entre este sistema y los dispositivos de instrumentación, potencia y visualización.

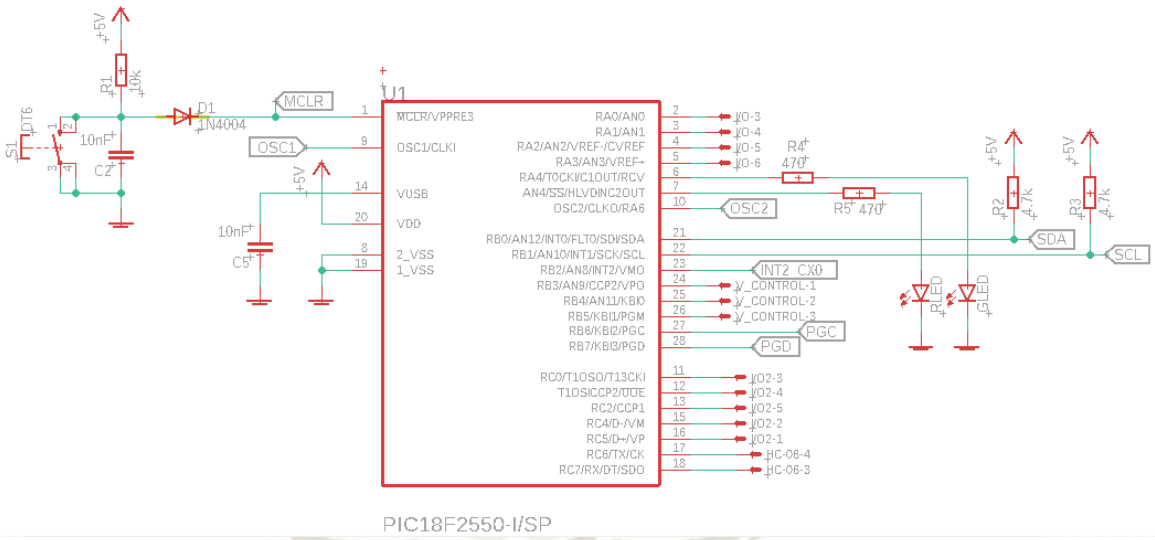


Fig. 44 Diagrama esquemático de conexión del controlador maestro.
Fuente: Elaboración propia

4.2 Diseño de etapa de Potencia

Para el diseño de esta etapa mostrada en la figura 45, se necesita realizar el cálculo de potencia que va a manejar y analizar el comportamiento del sistema para conocer los parámetros de control y resolución de salida.

Para esto analizaremos según los datos de la planta a controlar:

- Potencia nominal: 2KW x 2.
- Voltaje de alimentación: 220Vac.

- Corriente nominal: 9.09A por resistencia.

Debido a estas consideraciones, se propone utilizar el BTA16-600B que es un TRIAC que cumple de manera sobresaliente con los requisitos anteriormente descritos, ya que según [13] cuenta con:

- IT(RMS) = 16A.
- VDRM = 600V.
- IGT = 25mA.

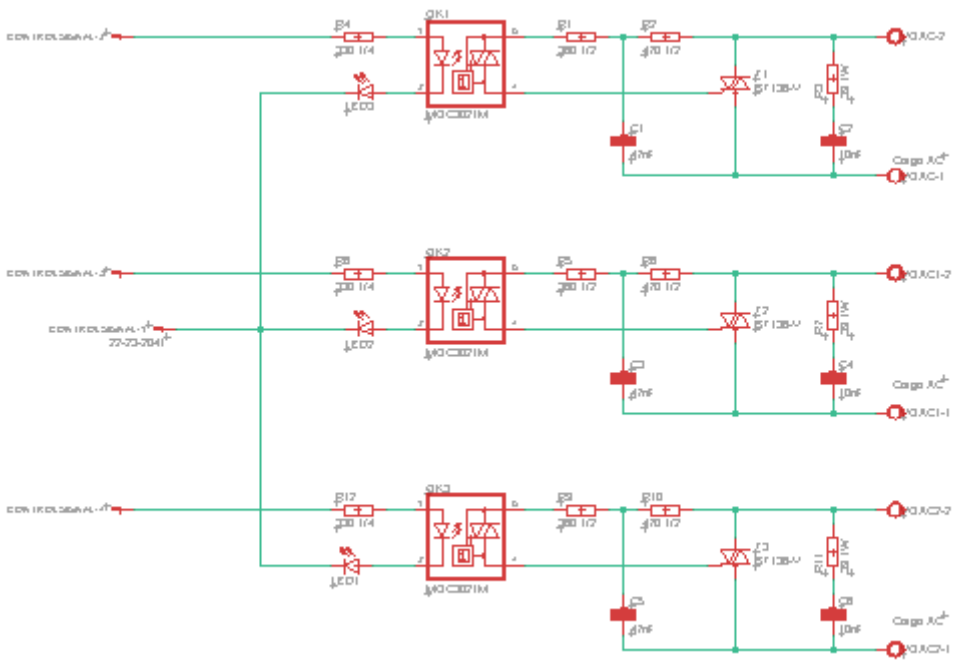


Fig. 45 Diagrama esquemático de control de potencia mediante ángulo de disparo, Fuente: [13]

El método para controlar la potencia, se basa en el control por ángulo de disparo de Triac para controlar el flujo de corriente hacia una determinada carga a partir de una señal de voltaje alterna.

Para esto, en la figura 46, analizaremos el voltaje RMS generado para una onda sinusoidal de 60Hz y 310Vp.

Siguiendo las indicaciones de la norma NTP 370.252, la sección adecuada de cable aislado con compuesto termoestable de hasta 450V y 10A es de: 2.1 mm2 que corresponde a Cable THW 14 AWG, con una corriente nominal de hasta 25A en un ducto o canaleta.

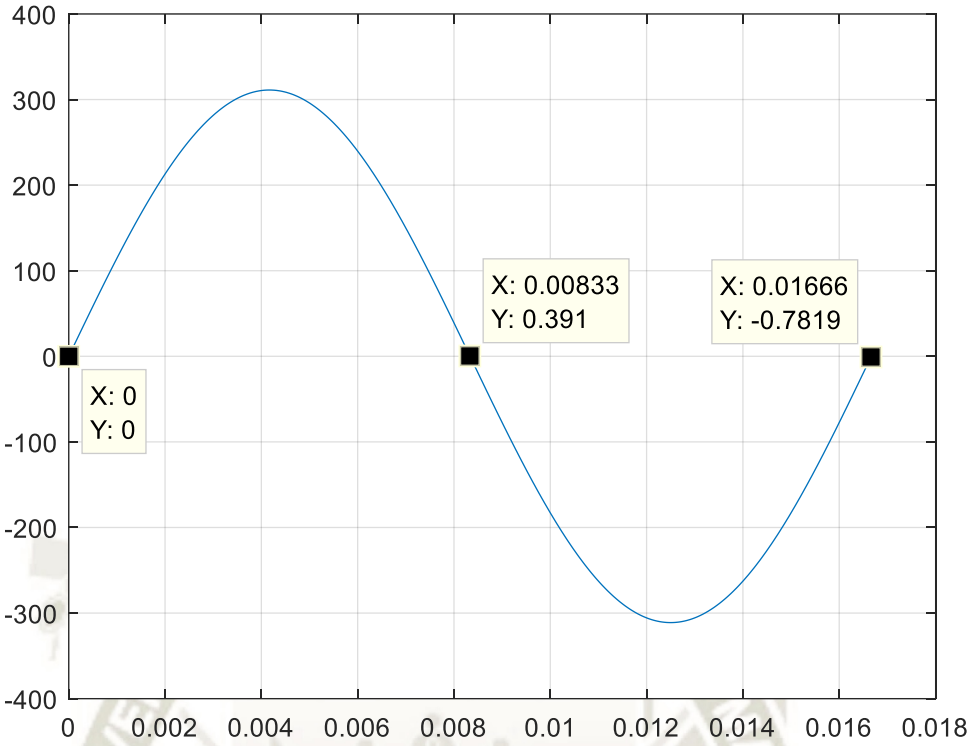


Fig. 46 Onda sinusoidal de 60Hz y 311Vp
Fuente: Elaboración propia

Para el cálculo del voltaje rms, primero definimos nuestros valores:

- Vp = 310v.
- $\omega = 2 * \pi * f.$
- $f = 60Hz.$
- $V(t) = Vp * sen(\omega t).$

  Entonces, partiendo de la siguiente ecuación, [10]

$$Vrms = \sqrt{\frac{1}{T} * \int_a^b V(t)^2 dt} \quad (20)$$

Donde en una onda sinusoidal perfecta, a = 0 y b = τ.

$$Vrms = \sqrt{\frac{1}{2\pi} * 2 \int_a^{\tau} Vp^2 * sen(\omega t)^2 dt} \quad (21)$$

$$Vrms = \sqrt{\frac{Vp^2}{\pi} \int_a^{\tau} sen(\omega t)^2 dt}$$

$$Vrms = \sqrt{\frac{Vp^2}{\pi}\int_a^{\tau} \frac{1-cos(2\omega t)}{2} dt}$$

$$Vrms = \sqrt{\frac{Vp^2}{2\pi}\int_a^{\tau} 1-cos(2\omega t) dt}$$

$$Vrms = \sqrt{\frac{Vp^2}{2\pi} * \left|\omega t - \frac{sen(2\omega t)}{2}\right|_a^{\tau}}$$

$$Vrms = \sqrt{\frac{Vp^2}{2\pi} * \left|\omega t - \frac{sen(2\omega t)}{2}\right|_a^{\tau}}$$

$$Vrms = \sqrt{\frac{Vp^2}{2\pi} * (\tau - \frac{sen(2\tau)}{2} - a + \frac{sen(2a)}{2})} \quad (22)$$

De la ecuación anterior, se puede deducir que, para los valores de $\tau = \pi$ y $a = 0$, obtenemos el valor Vrms de una onda senoidal simplificado.

$$Vrms = \sqrt{\frac{Vp^2}{2\pi}(\pi - \frac{sen(2\pi)}{2} - 0 + \frac{sen(0)}{2})} \quad (23)$$

$$Vrms = \sqrt{\frac{Vp^2}{2\pi}(\pi - 0 - 0 + 0)} = \sqrt{\frac{Vp^2}{2}}$$

$$Vrms = \frac{Vp}{\sqrt{2}} \quad (24)$$

Este proyecto, al utilizar TRIACs utiliza el principio de funcionamiento por ángulo de disparo, lo que significa que el valor τ se mantiene constante con el valor de π y lo único que varía es el valor de inicio de onda, la variable a. Por lo tanto, para determinar el valor de voltaje rms que se vaya a aplicar en función del ángulo de disparo tenemos la siguiente ecuación:

$$Vrms = \sqrt{\frac{Vp^2}{2\pi}(\pi - \frac{sen(2\pi)}{2} - a + \frac{sen(2a)}{2})}$$

$$Vrms = \sqrt{\frac{Vp^2}{2\pi}(\pi - a + \frac{sen(2a)}{2})}$$

Conociendo el voltaje pico de alimentación y despreciado la caída de tensión en los tiristores, tenemos el voltaje rms en carga igual a:

$$Vl_{rms} = \sqrt{\frac{220^2}{\pi}(\pi - a + \frac{sen(2a)}{2})}v \ (25)$$

Con estos datos, obtenemos la gráfica de respuesta de voltaje rms en función del tiempo de disparo dado por el microcontrolador en la figura 47.

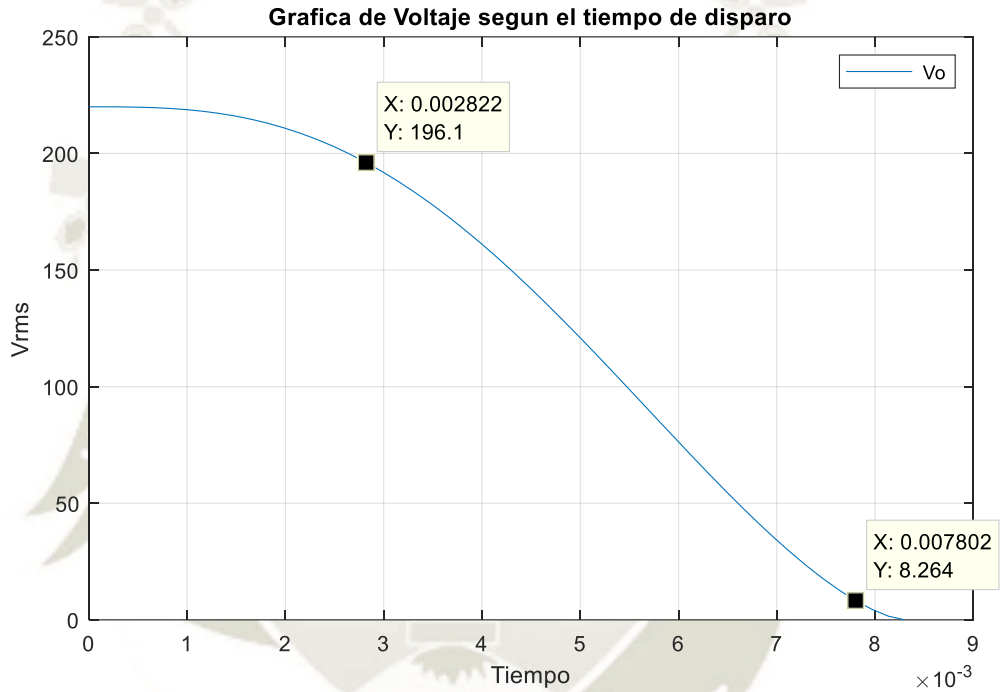


Fig. 47 Respuesta de Vrms en funcion al tiempo de disparo
Fuente: Elaboración propia

Según estos datos obtenemos podemos observar un comportamiento lineal entre los valores de 2.8mS y 7.8mS que corresponden a 196 y 8 Vrms respectivamente, a continuación, se podría linealizar la salida del sistema delimitando el rango de acción, pero gracias a las capacidades de procesamiento de nuestro controlador, podemos implementar una función de 3er orden que cumpla con los valores empleados. En la figura 48 se muestra la respuesta en el tiempo mediante un polinomio de aproximación de 4to orden.

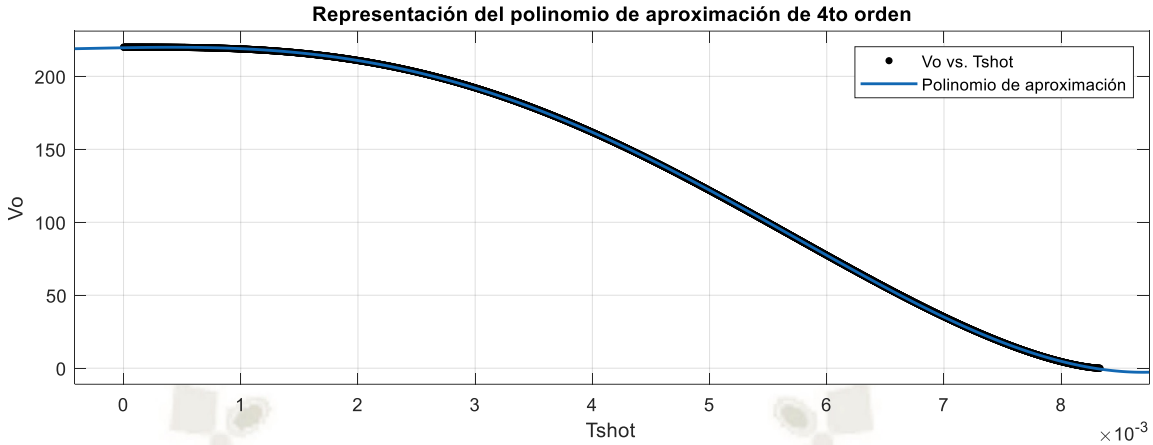


Fig. 48 Representación del polinomio de aproximación de 4to orden vs respuesta de la etapa de potencia
Fuente: Elaboración propia

Los datos obtenidos de esta función son:

Linear model Poly4:
f(x) = p1*x^4 + p2*x^3 + p3*x^2 + p4*x + p5
where x is normalized by mean 0.004167 and std 0.002406
Coefficients (with 95% confidence bounds):
p1 = 3.629 (3.627, 3.63)
p2 = 8.563 (8.561, 8.565)
p3 = -26.27 (-26.27, -26.26)
p4 = -89.26 (-89.26, -89.26)
p5 = 155.6 (155.6, 155.6)

Goodness of fit:
SSE: 987.4
R-square: 1
Adjusted R-square: 1
RMSE: 0.1403

$$F(x) = 3.629X^4 + 8.563X^3 - 26.27X^2 - 89.26X + 155.6 \ (26)$$

Dado estos cálculos, se decide utilizar el TRIAC BTA16-600B, de la marca STMicroelectronics, los cuales presentan las siguientes características: [13]

- $I_{T(RMS)}$ = 16A (Tc=86°C).
- I_{TSM} = 168A (60Hz).
- dV/dT = 40v/uS.
- $R_{th(jc)}$ = 2.1°C/W.
- $R_{th(ja)}$ = 60°C/W.

Primero, podemos definir que la corriente máxima de trabajo encaja en nuestras especificaciones, ya que nuestra carga, el grupo de resistencias laterales ocupan en conjunto 3KW, y en nuestro circuito de salida ocupamos 3 controladores de potencia del tipo SSR.

Para hallar el voltaje máximo RMS en una región de trabajo con disparo continuo, tenemos:

$$I_L = \frac{P}{V} \quad (27)$$

$$I_L = \frac{3000W}{220Vrms} = 13.63A$$

Con este dato, basándonos en la curva de corriente en funcion de la temperatura extraída del datasheet del TRIAC BTA16 y mostrada en la figura 49, podemos definir que el rango de temperatura en el que el TRIAC va a poder manejar cargas de hasta 13.63 Amperios va desde los 0 – 90°C, esto significa, que la temperatura de case para que este dispositivo maneje la capacidad total sin sufrir una disminución de sus capacidades conductivas es de 90°C.

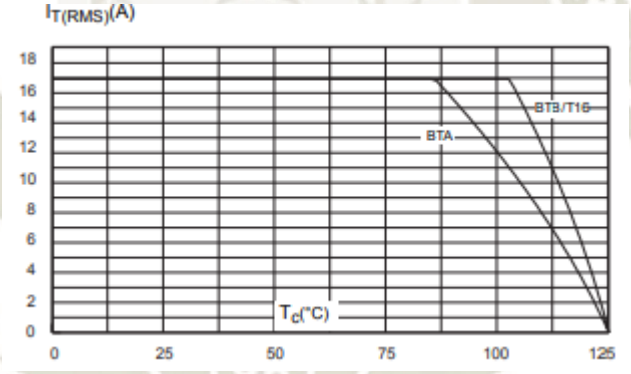


Fig. 49 Curva de corriente RMS en funcion de la temperatura de case
Fuente: Elaboración propia

Conociendo este parámetro descrito en la figura 49, es necesario conocer la potencia disipada del TRIAC con el propósito de determinar la necesidad de la implementación de un disipador de calor y de ser el caso, seleccionar uno adecuado.

Para esto, nos basamos en la figura 50, que muestra una curva que responde al valor de la potencia disipada en función a la corriente RMS, utilizaremos este valor para encontrar la temperatura de trabajo en base al cálculo por aproximación entre los circuitos eléctricos y los circuitos térmicos.

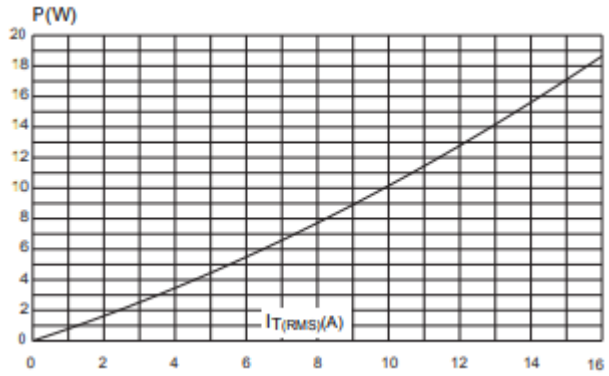


Fig. 50 Relación entre potencia disipada y corriente de trabajo en un TRIAC BTA16
Fuente: Elaboración propia

Para explicar esta relación entre los circuitos eléctricos y térmicos, procedemos a explicar la ley de ohm térmica:

$$Tj - Ta = P * R_{th(ja)} \ (28)$$

Donde:

- Tj = temperatura máxima de unión (°C).
- Ta = temperatura ambiental (°C).
- P = potencia disipada por el componente (W).
- $R_{th(ja)}$ = Resistencia térmica total entre la unión y el ambiente (°C/W).

Con estos datos, procedemos a hallar la temperatura de unión estimada para una carga de 13.63A.

- P(13.64A) = 15W.
- Ta = 25°C.
- $R_{th(ja)}$ = 60°C/W.

Por lo que reemplazando los valores obtenemos:

$$Tj(estimada) = P * R_{th(ja)} + Ta \ (29)$$

$$Tj(estimada) = 15 * 60 + 25 = 925°C$$

Como podemos observar, sobrepasa ampliamente el valor máximo de Tj = (90°C) por lo que es necesario instalar un disipador de calor. Para esto, asociamos el circuito térmico equivalente mostrado en la figura 51.

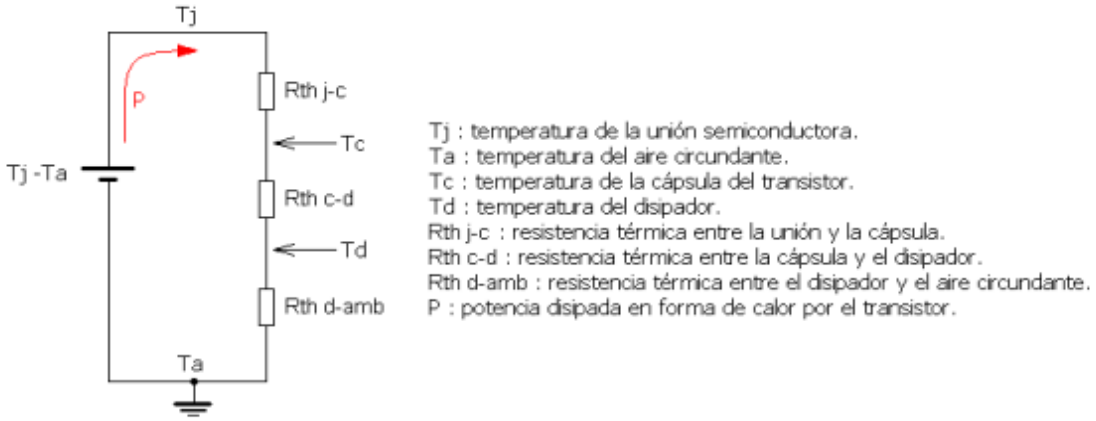


Fig. 51 Circuito térmico equivalente para una fuente de calor.
Fuente [14]

Un punto a considerar, es la utilización de diferentes componentes, Triac, con un mismo disipador, por lo que, tomando en consideración el uso de un actuador auxiliar de una potencia de 1.5KW se tiene el circuito térmico mostrado en la figura 52:

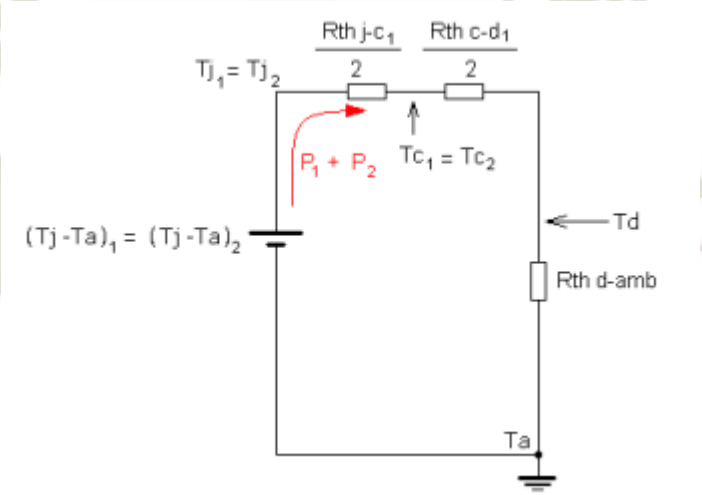


Fig. 52 Circuito térmico equivalente para dos fuentes de calor
Fuente [14]

Por lo que, reemplazando, tendríamos los siguientes datos:

- P1(13.64A) = 15W.
- P2(6.8A) = 6.5W.
- Ta = 25°C.
- Tj = Tjmax*0.6 = 80°C.
- $R_{1th(jc)}$ = 2.1°C/W.
- $R_{2th(jc)}$ = 2.1°C/W.

- $R_{th(cd)}$ = 0.12°C/W (con silicona termo conductora).

Y aplicando la Ley de Ohm térmica, tenemos que la resistencia de disipador es:

$$R_{th(da)} = \frac{Tj - Ta}{P1 + P2} - \frac{R_{th(jc)} + R_{th(cd)}}{2}$$

$$R_{th(da)} = \frac{80 - 25}{15 + 6.5} - \frac{2.1 + 0.12}{2}$$

$$R_{th(da)} = 1.45°C/W$$

Por lo tanto, debemos seleccionar un disipador de calor con un valor de resistencia inferior al mencionado, contamos con un disipador de 0.4°C/W, por lo que procedemos a calcular la temperatura a la que se encontrará la unión y la superficie del mismo.

$$Tj = Ta + (P1 + P2) * \left(R_{th(da)} + \frac{R_{th(jc)} + R_{th(cd)}}{2}\right) (30)$$

$$Tj = 25 + (15 + 6.5) * \left(0.4 + \frac{2.1 + 0.12}{2}\right)$$

$$Tj = 57.4°C$$

Mientras que la temperatura del disipador será de:

$$Td = (P1 + P2) * Rthd + Ta \ (31)$$

$$Td = 21.5 * 0.4 + 25 = 28.44°C$$

4.3 Diseño de etapa de instrumentación

Esta etapa consta de la detección mediante pulsos del cruce por cero de una señal de voltaje alterno, la utilización de sensores de temperatura como el termopar tipo k y la adaptación de esta señal.

4.3.1 Detección del cruce por cero

La detección del cruce por cero de una onda alterna senoidal se realiza para poder controlar cargas mediante el control por ángulo de disparo, al detectar que una señal alterna ha llegado al valor de amplitud 0, se emite un pulso que detecta el microcontrolador y define el inicio de una nueva onda, teniendo asi la oportunidad de activar el Triac en el tiempo adecuado.

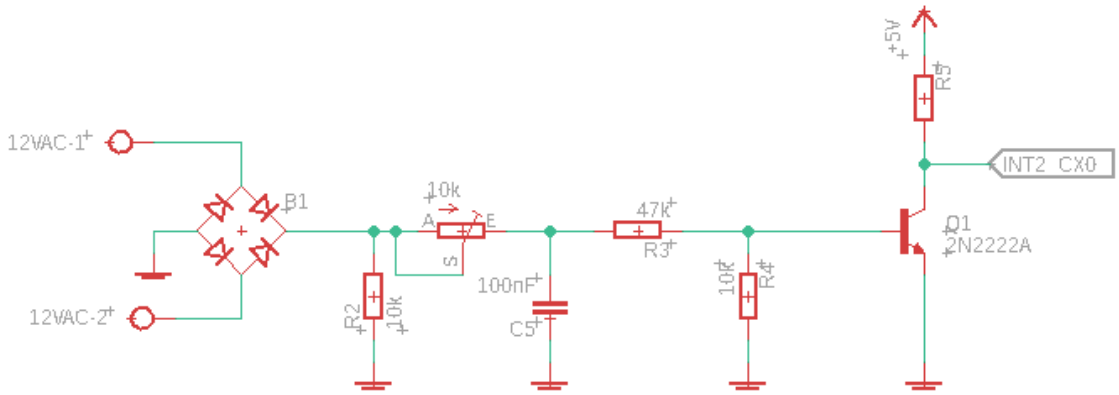


Fig. 53 Diagrama esquemático del circuito de detección del pulso de cruce por cero
Fuente: [15]

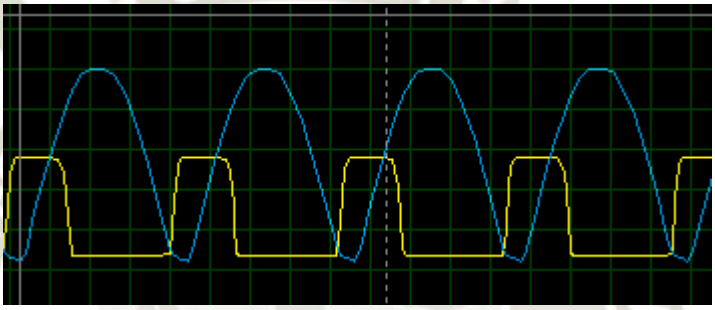


Fig. 54 Respuesta del sistema descrito en la figura 45 ante una entrada senoidal de 12v de amplitud
Fuente: Elaboración propia

Como se puede observar en las figuras 53 y 54, el sistema cumple con el propósito de detectar mediante un pulso cuando la amplitud de la señal de entrada es 0, podemos observar además que el pulso se mantiene constante durante un periodo de tiempo, por lo que se debe configurar el microcontrolador para detectar únicamente cuando se haya recibido una señal de interrupción con flanco ascendente.

4.3.2 Sensor y Transmisor de temperatura

Uno los sensores de temperatura más usados son las Termocuplas, estos dispositivos transforman la energía calorífica en energía eléctrica mediante el efecto Joule, dependiendo del material de fabricación, obtenemos un tipo diferente de termopar, para este proyecto, utilizamos la termocupla tipo K, que es un termopar muy popular en el mercado, lo cual facilita la implementación y mantenimiento, y su rango de operatividad se adecúa a nuestro proyecto.

Debido a que las termocuplas deben tener un circuito de acondicionamiento por su bajo voltaje de salida, y el inconveniente de que la temperatura de unión en sus terminales difiere

con la temperatura del proceso, se requiere la implementación de un circuito de acondicionamiento por unión fría.

Para esto, se ha seleccionado el circuito integrado Max6675, que como se puede observar en la figura 55, consta de un amplificador de señal, un filtro pasa bajos, un bloque de compensación de unión fría que registra la temperatura a la que se encuentra el integrado, que a su vez es similar a la de unión, y digitaliza esta señal mediante un bloque conversor análogo – digital con salida de comunicación SPI.

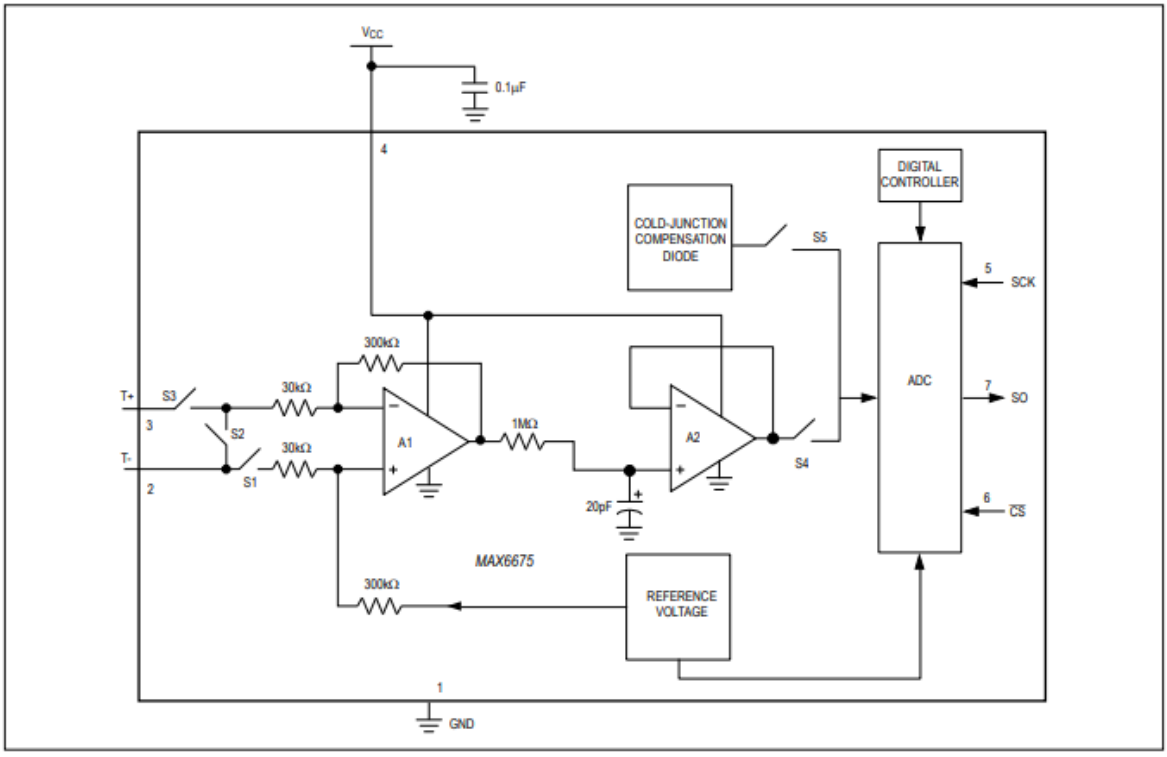


Fig. 55 Diagrama de bloques del IC Max6675
Fuente: [16]

Por lo tanto, este circuito integrado funciona como un transmisor de temperatura esclavo de un controlador principal, el modo de operación se muestra en la figura 56, donde se observa que si el pin CS, cambia a estado lógico 0, inicia el proceso de comunicación, recibiendo la señal de reloj por el pin SCK y transmitiendo 2 bytes de información mediante el pin SO.

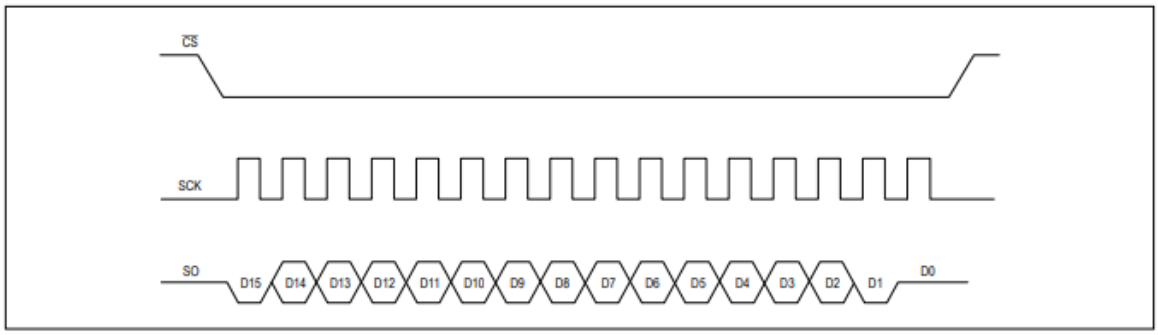


Fig. 56 Protocolo de comunicación SPI
Fuente: [16]

4.4 Diseño de etapa HMI

Para este proyecto, se pretende diseñar dos etapas de interacción hombre – máquina, la primera es mediante un circuito de control y visualización desde el panel de control, la segunda será implementada mediante el software App designer, utilizando el lenguaje de programación de Matlab, que en conjunto con el compilador de aplicaciones remotas dadas en el software, podemos diseñar un aplicativo móvil multiplataforma, que nos dé mayor control sobre nuestro dispositivo en una gran variedad de terminales que tengan acceso a la aplicación, dicho software permite graficar una línea de tendencia y preservar los datos.

4.4.1 Interfaz HMI in situ

El diseño de la primera etapa se realizó en base a una pantalla LCD de 16 caracteres por 2 filas, la implementación de indicadores luminosos y sonoros para la salida de datos y para el ingreso de datos la implementación de selectores tipo pulsadores y el uso de un potenciómetro analógico para definir el setpoint.

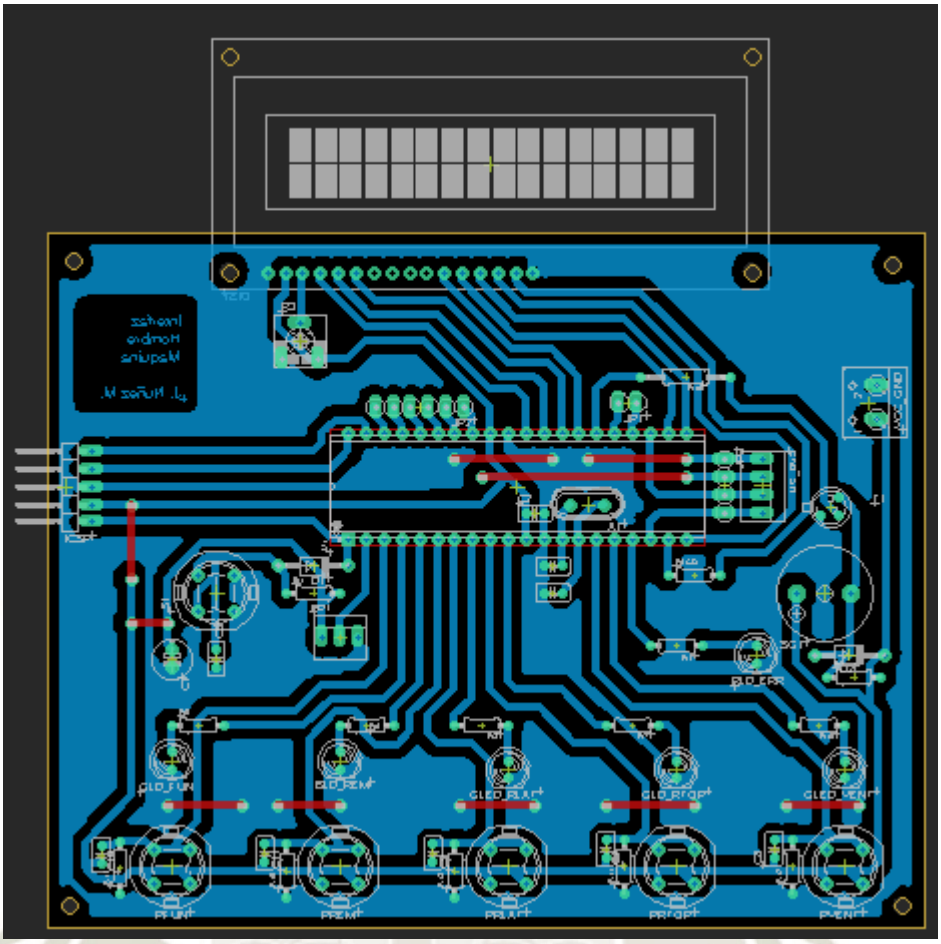


Fig. 57 PCB de la etapa HMI en software Eagle CAD
Fuente: Elaboración propia

En la figura 57, se muestra el diseño en 2D de la placa de circuito impreso, diseñada mediante el software Autodesk Eagle. Esta placa secundaria, funciona basada en un microcontrolador de la gama media de microchip 16F, el modo de operación se describe en el diagrama de flujo mostrado en la figura 58, trabajando como un sistema esclavo del controlador principal, conectado mediante el protocolo de comunicación I2C.

Para este proyecto, se ha seleccionado el microcontrolador PIC16F887, que es un dispositivo potente, posee el número de entradas y salidas necesarias, los módulos internos de comunicación y la velocidad de procesamiento requerida para esta aplicación.

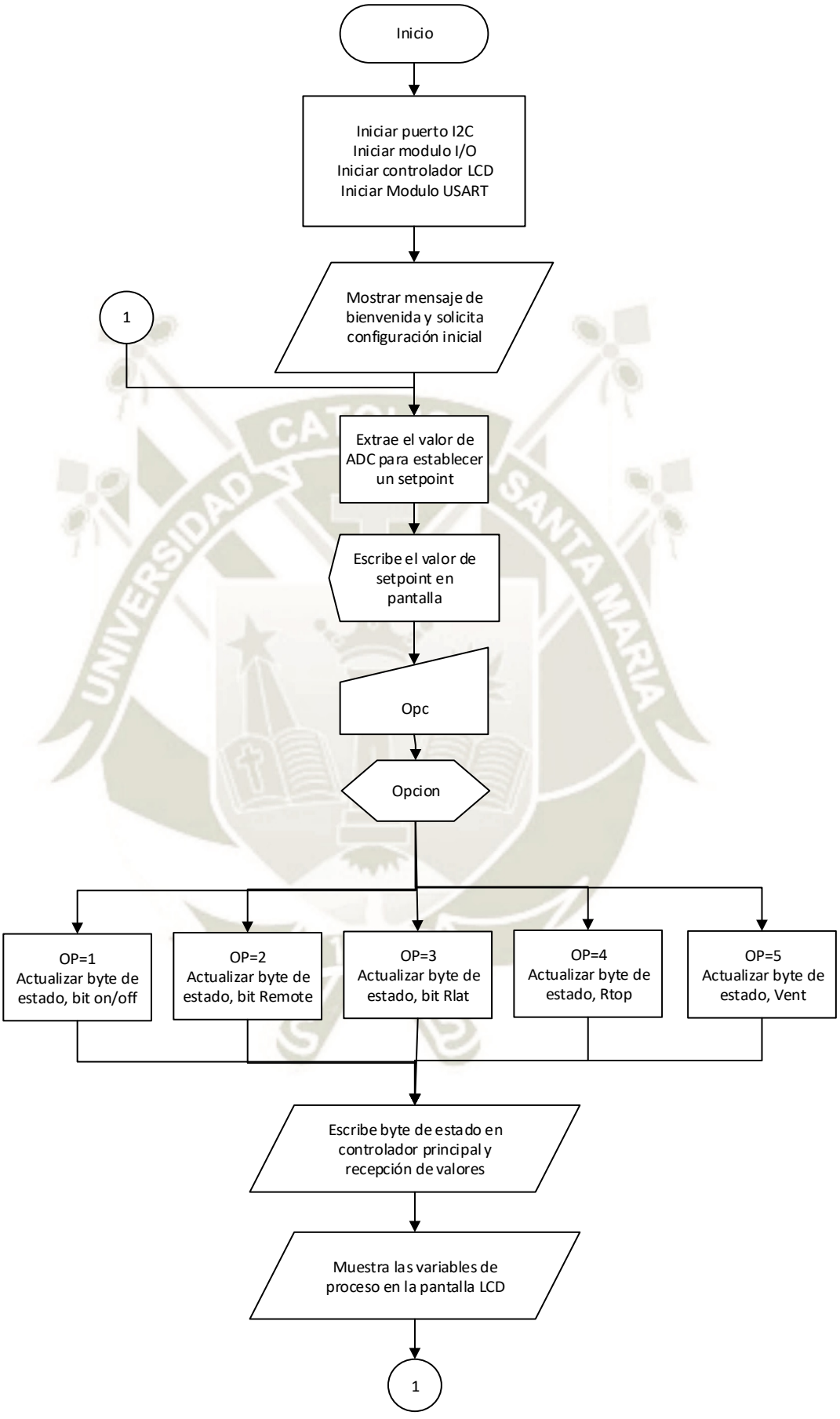


Fig. 58 Diagrama de flujo del controlador HMI
Fuente: Elaboración propia

4.5 Software y lógica de control

Esta sección comprende al algoritmo de control que se ha implementado en la placa de control principal basada en el PIC18F2550, este microcontrolador se ha programado en lenguaje C, los valores de control a implementar han sido calculados y sintonizados en el capítulo anterior y en este nos basamos en la adaptación de estos valores a un microcontrolador de 8bits.

4.5.1 Implementación de control PI

El algoritmo de control para PI se desarrolla en base al siguiente diagrama de flujo mostrado en la figura 59.

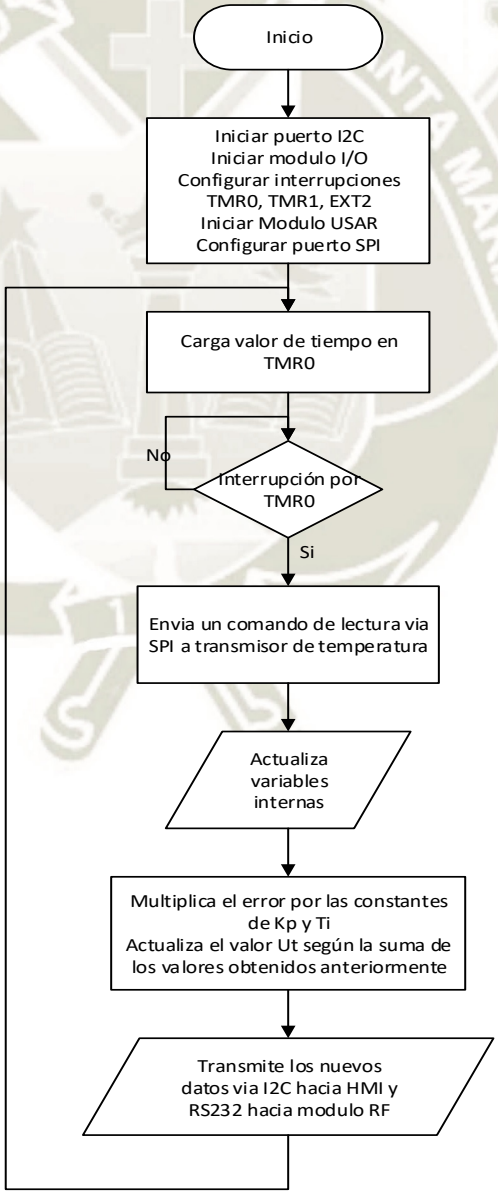


Fig. 59 Algoritmo de control PI
Fuente Elaboración propia

Este tipo de control digital necesita una señal discreta, para esto se implementa el uso de TMR0 para realizar la señal de control a una determinada frecuencia.

El tiempo de muestreo está determinado por:

$$T < \frac{L}{2}, donde \frac{L}{4} = \ 10.5s \ (32)$$

$$T = \frac{t}{20} = 8.35s$$

Se procede entonces a generar una rutina en TMR0 con fin de iniciar la señal de control cada 8 segundos.

Lo siguiente, es expresar la señal de control PI en tiempo discreto.

$$C(z^{-1}) = \frac{u(k)}{e(k)} = \frac{q0 + q1z^{-1} + q2z^{-2}}{1 - z^{-1}} \ (33)$$

Donde:

$$q0 = Kp * \left[1 + \frac{T}{2Ti} + \frac{Td}{T}\right] \ (34)$$

$$q1 = -Kp * \left[1 - \frac{T}{2Ti} + \frac{2Td}{T}\right]$$

$$q2 = \frac{KpTd}{T}$$

Despejando, para obtener la ley de control u(k), procedemos:

$$u(k)(1 - z^{-1}) = (q0 + q1z^{-1} + q2z^{-2}) * \big(e(k)\big) \ (35)$$

$$u(k) = u(k)z^{-1} + q0e(k) + q1z^{-1}e(k) + q2z^{-2}e(k) \ (36)$$

Tras aplicar la transformada Z a la ecuación anterior se tiene la ecuación de control a implementar en el microcontrolador en lenguaje C para un controlador PID

$$u(k) = u(k-1) + q0e(k) + q1e(k-1) + q2e(k-2) \ (37)$$

Así obtenemos una ecuación donde la señal de control en el instante k es igual a la señal de control anterior adicionada al error en el instante k multiplicada por q0, al error anterior multiplicado por q1 y al error en k-2 multiplicado por q2.

Como en este tipo de procesos no es relevante la acción derivativa, procesos térmicos y lentos, se puede omitir optimizando asi la programación del microcontrolador.

Por lo tanto, las constantes quedan determinadas por Td = 0.

$$q0 = Kp * \left[1 + \frac{T}{2Ti}\right] \quad (38)$$

$$q1 = -Kp * \left[1 - \frac{T}{2Ti}\right] \quad (39)$$

$$q2 = 0$$

Teniendo la ecuación de control a implementar igual a:

$$u(k) = u(k-1) + q0e(k) + q1e(k-1)$$

$$q0 = 3.58 * \left[1 + \frac{167}{2 * 140}\right] = 5.712$$

$$q1 = -3.58 * \left[1 - \frac{167}{2 * 140}\right] = -1.444$$

$$u(k) = u(k-1) + 5.712 * e(k) - 1{,}444 * e(k-1) \quad (40)$$

4.5.2 Implementación de control difuso

Para implementar este tipo de control, nos basamos en 4 principales funciones, la toma de datos, la fuzzificación de los mismos, la utilización de las reglas de inferencia y la defuzzificación de la señal.

a. Acondicionamiento de datos

El acondicionamiento de datos se implementa por software, tanto las variables de proceso como las condiciones de control son procesadas por otros dispositivos esclavos los cuales transmiten estos datos al controlador principal. Mediante un algoritmo de integración se establece un filtro digital que mejora la precisión.

b. Fuzzificación

Para la fuzzificación en lenguaje C, implementamos las funciones de membresía mostradas en las figuras 26 y 27, que corresponden a las entradas de nuestro sistema difuso, error y

derivada del error. Asi mismo se ha dividido cada funcion en diferentes rectas, como se observa en la figura 60, las cuales se describen a continuación:

$$R1 = Emuy_{neg} = 1 \quad (41)$$

$$R2 = Emuy_{neg} = -0.05 * Et - 0.5 \quad (42)$$

$$R3 = Eneg = 0.1 * Et + 2 \quad (43)$$

$$R4 = Eneg - 0.1 * Et \quad (44)$$

$$R5 = Emin = 2 * Et + 1 \quad (45)$$

$$R6 = Emin = -2 * Et + 1 \quad (46)$$

$$R7 = Epos = 0.1 * Et \quad (47)$$

$$R8 = Epos = -0.1 * Et + 2 \quad (48)$$

$$R9 = Emuy_{pos} = 0.05 * Et - 0.5 \quad (49)$$

$$R10 = Emuy_{pos} = 1 \quad (50)$$

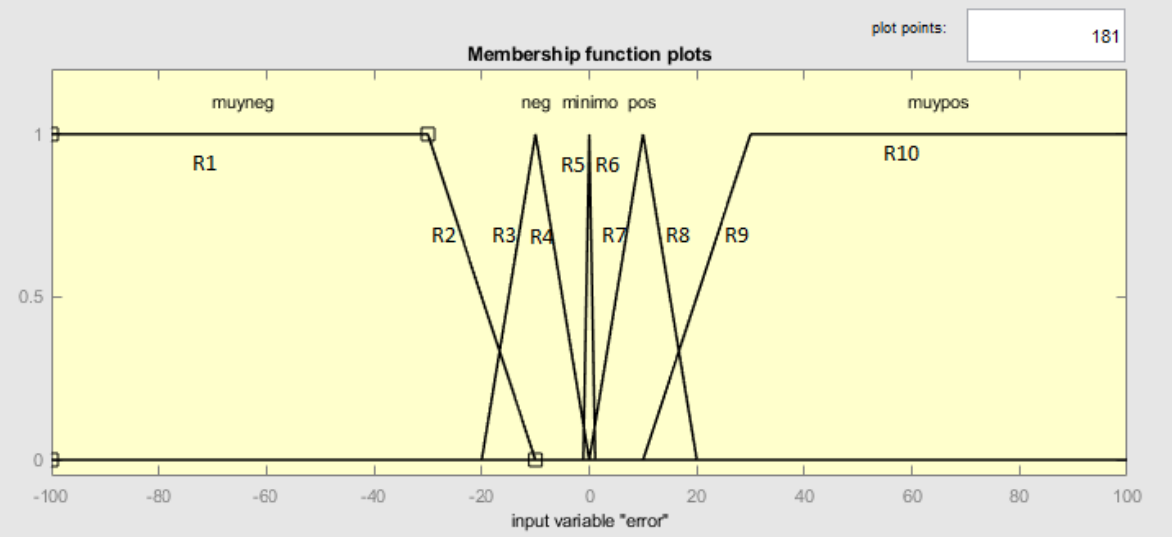


Fig. 60 Funciones de membresía para la variable ERROR,
Fuente: Elaboración propia

La clasificación del error se hace mediante una serie de instrucciones condicionales IF y comparadores, clasificando así el error encontrado de la variable de proceso y el setpoint en una o más funciones de membresía. Esta funcion de fuzzificación, entrega 5 valores para las variables: Emuy_neg, Eneg, Emin, Epos, Emuy_pos, estas variables son de tipo flotante y el valor oscila entre 0 y 1.

Del mismo modo se tiene la funcion de fuzzificación para la variable derivada del error, que como su nombre lo indica, es la derivada del error en el tiempo, esto nos sirve para que el sistema pueda predecir el comportamiento del sistema.

En la figura 61 se muestran estas rectas y a continuación sus respectivas ecuaciones.

$$R1 = dneg = 1 \quad (51)$$

$$R2 = dneg = -0.52632 * dEt - 0.052632 \quad (52)$$

$$R3 = dmin = dEt + 1 \quad (53)$$

$$R4 = dmin = -dEt + 1 \quad (54)$$

$$R5 = dpos = 0.52632 * dEt - 0.052632 \quad (55)$$

$$R6 = dpos = 1 \quad (56)$$

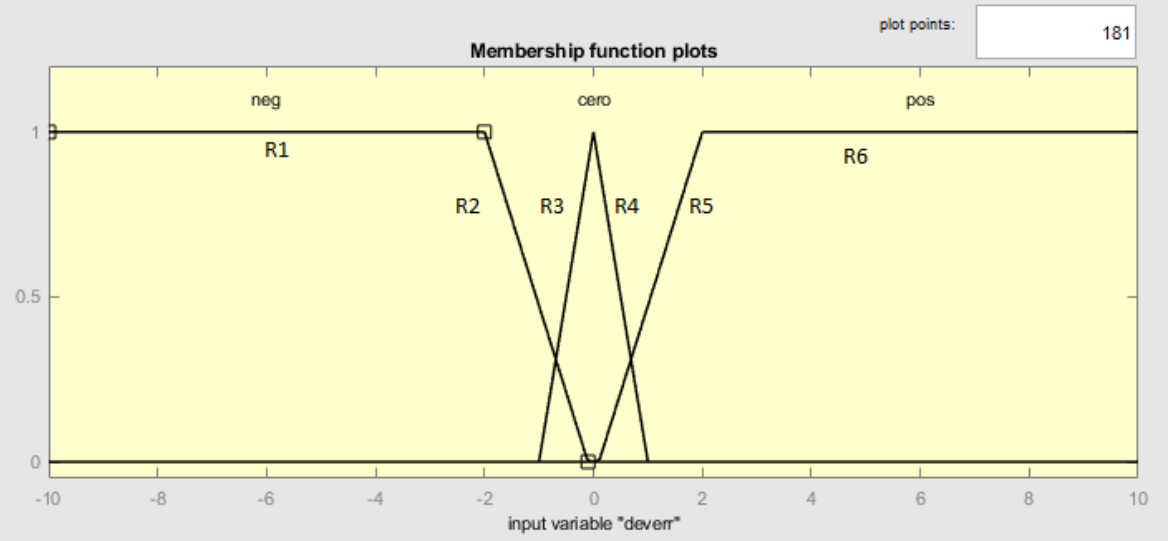


Fig. 61 Funciones de membresía para la variable devErr,
Fuente: Elaboración propia

Una vez procesadas ambas variables, se tienen valores difusos que serán procesados como conjuntos en la etapa siguiente: Reglas de inferencia.

c. Reglas de inferencia

Para esta etapa, se tienen las reglas mostradas en la figura 61, como se puede observar dictan el comportamiento del sistema y su función es asociar cada conjunto difuso y el valor obtenido con una salida difusa, esto se logra tomando el valor máximo de cada variable siempre y cuando se cumpla una regla y multiplicando ese valor por la constante de salida.

Como se puede observar en la figura 62, se obtienen dos valores, W que es el peso del valor máximo entre ambas funciones y el valor Z, que en este caso es la constante del valor de la

funcion de membresia de salida difusa, este valor también puede ser una ecuación lineal, depende del ingeniero de control determinar el método de fuzzificación adecuado para su proceso.

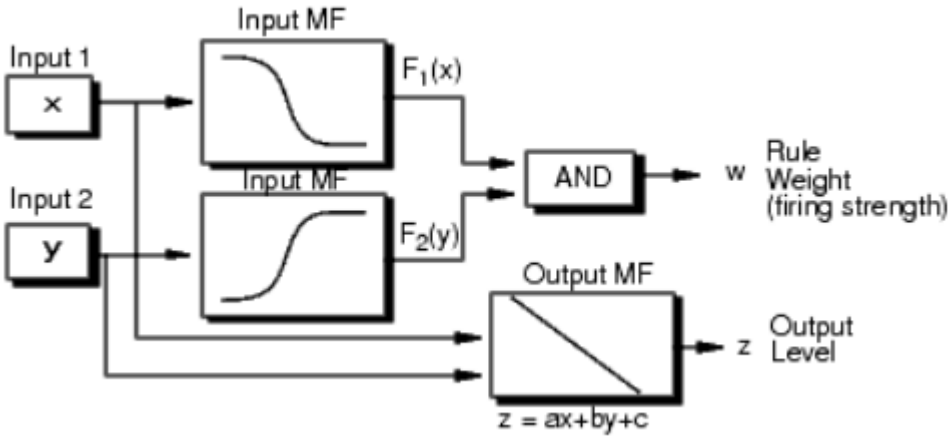


Fig. 62 Diagrama de bloques del proceso de inferencia con reglas difusas para dos entradas, Fuente: [9]

En la figura 63 se puede observar las 11 reglas difusas que rigen este controlador.

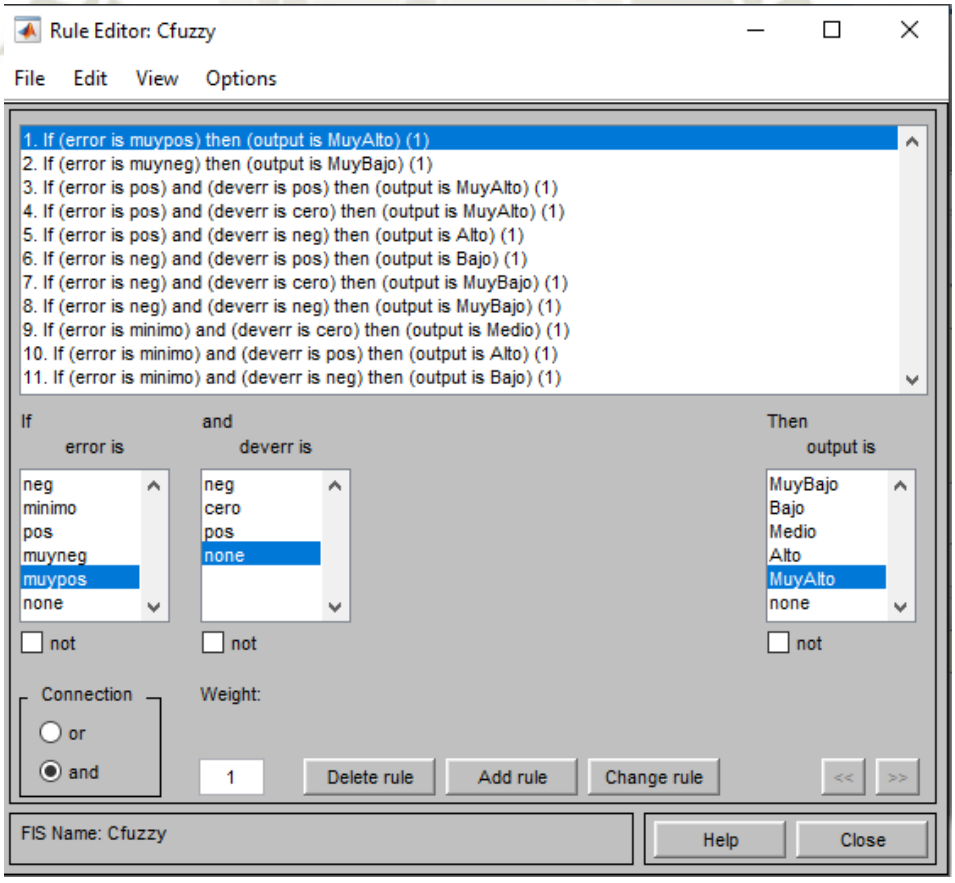


Fig. 63 Reglas de inferencia en software Matlab. Fuente: Elaboración propia

d. Defuzzificación

Esta etapa consiste en descifrar el valor de salida difusa en una salida numérica, en definición porcentual, por lo tanto, se procede a evaluar los datos conseguidos en la etapa anterior.

Para esto vamos a utilizar la siguiente ecuación:

$$Ut = \frac{\sum_{i=1}^{N} wi * zi}{\sum_{i=1}^{N} wi} \quad (57)$$

A continuación, procedemos a utilizar el software de regresión lineal de Excel para graficar el valor a implementar en los diferentes TMR del controlador para obtener el ángulo de disparo del Triac deseado. Como se puede observar en la figura 64, la ecuación de regresión lineal es:

$$F(x) = 498 * x + 15735 \quad (58)$$

Donde:

- F(x) es el valor a asignar en TMR.
- X es el valor de salida del control difuso.
- 65535 es el valor máximo en TMR de 16bits.

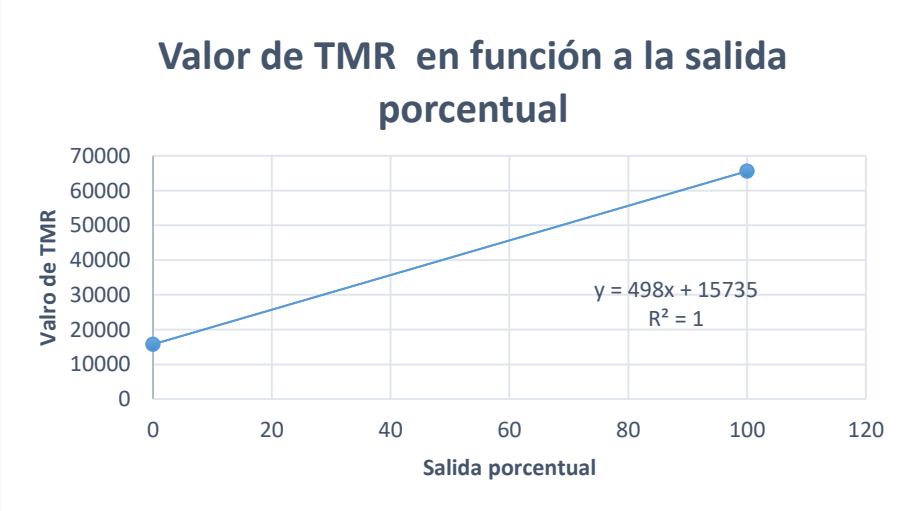


Fig. 64 Valor de asignación a TMR en función a salida defusificada,
Fuente: Elaboracion propia

4.6 Implementación del tablero de control en el horno eléctrico

El horno sobre el que se va a implementar el controlador se muestra en la figura 65, tiene entre sus características una potencia de 5kW, y una masa volumétrica de 0.35m3, que puede alcanzar una temperatura de 600°C.

Como elementos calefactores se utilizaron dos resistencias eléctricas de tipo níquel-cromo, cada una de 2,5KW.

Como sensor de temperatura se utiliza la termocupla tipo k, ampliamente utilizada en la industria.

Fig. 65 Interior de cámara de calor en horno eléctrico
Fuente: Elaboración propia

El diseño del tablero se realizó en modelo CAD y se implementó bajo los estándares de la norma IEC 61439 y se realizó una inspección de calidad de parte propia.

El tablero se encuentra dividido en 4 bloques principales, los cuales son:

- Etapa de control y medición.
- Etapa de potencia.
- Interfaz de control (control interno y visualización exterior).
- Etapa de seguridad.

A continuación, se muestra en las figuras 66 y 67 el tablero de control implementado por dentro y fuera respectivamente.

Fig. 66 Vista exterior del panel de control,
Fuente: Elaboración propia

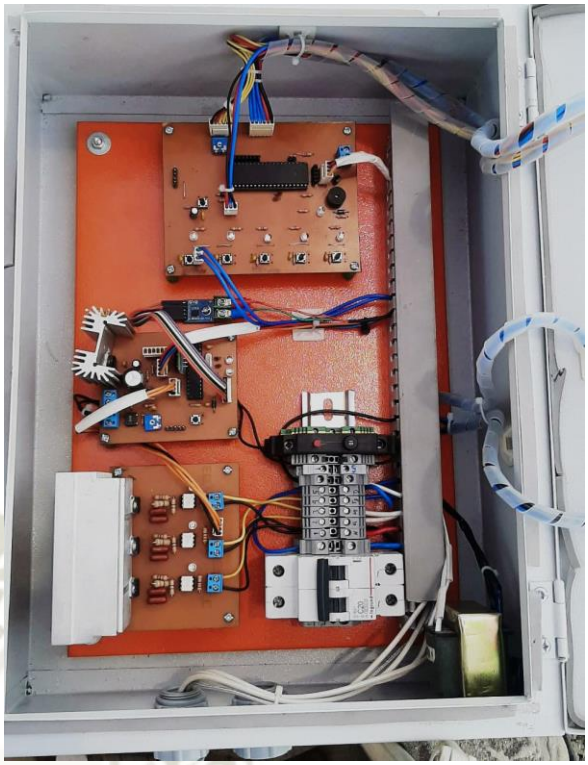


Fig. 67 Vista interna del panel de control,
Fuente: Elaboración propia

4.7 Implementación de la interfaz HMI

Para poder visualizar un proceso se pueden implementar diferentes métodos, estos pueden ser desde programas de computador o dispositivos móviles, hasta pantallas in situ que muestren información relevante.

Para este proyecto, decidí diseñar un aplicativo web que pueda funcionar en cualquier ordenador y móvil sin necesidad de instalar, utilizando el navegador web. A continuación, se muestran los pasos para diseñar este aplicativo mediante la ayuda de la herramienta app designer de Matlab.

4.7.1 App Designer

Para acceder a esta herramienta basta con escribir appdesigner en la ventana de comandos de Matlab, a continuación en la figura 68, nos aparece un entorno similar al siguiente:

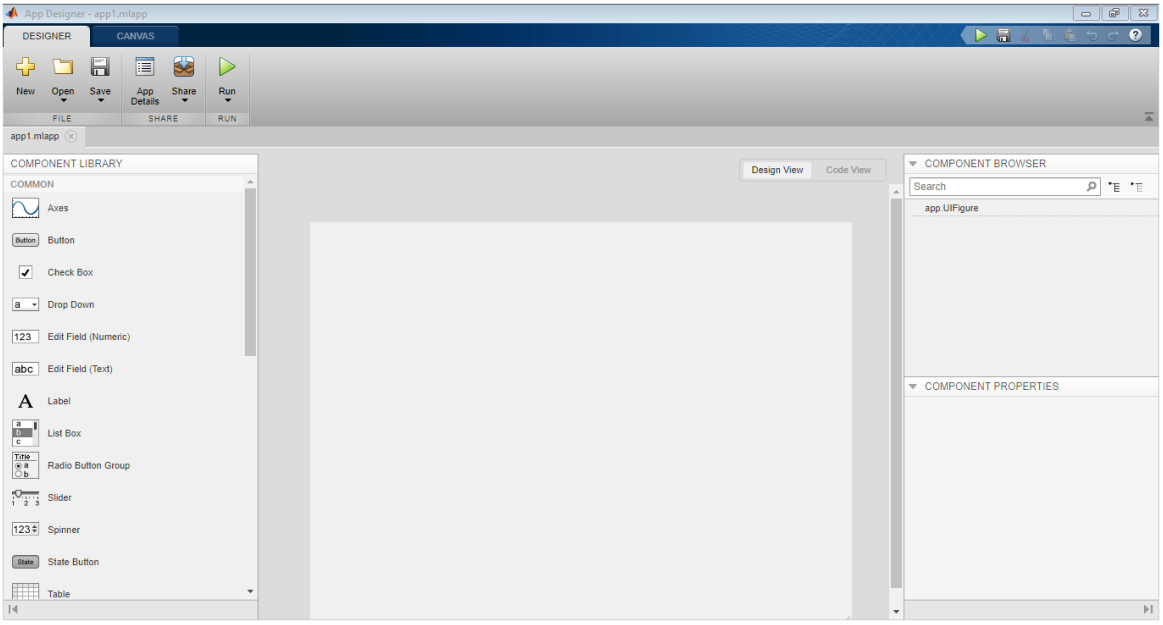


Fig. 68 Entorno de trabajo de AppDesigner
Fuente: Elaboración propia

Como podemos observar de la figura 68, se distinguen 4 secciones de trabajo que explicaremos a continuación:

a. Bloque de configuración

En esta sección se puede configurar la app que vamos a diseñar, asi como guardar, compartir y compilar, mostrado en la figura 69.

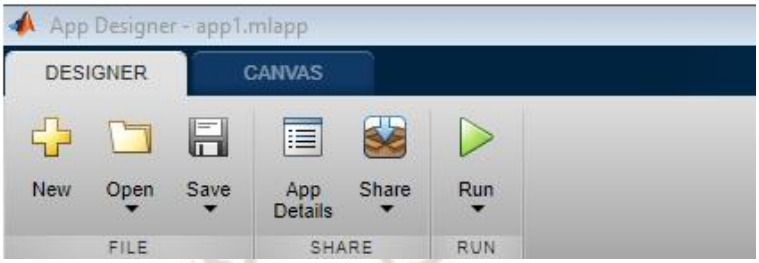


Fig. 69 Bloque de configuración app designer
Fuente: Elaboración propia

b. Bloque de diseño

En este bloque, se tienen diferentes componentes para diseñar el aplicativo, estos pueden actuar como entradas y salidas de datos tanto analógicas como datos numéricos, se tienen además indicadores y la opción de construir pestañas y gráficos, mostrado en la figura 70.

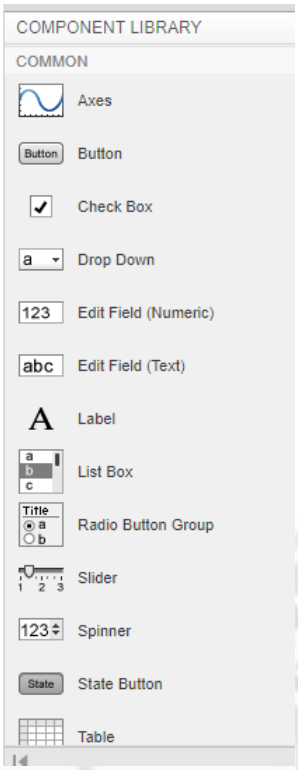


Fig. 70 Bloque de diseño
Fuente: Elaboración propia

c. Pre visualización

Esta pantalla es donde se desarrollará la aplicación, se arrastran los componentes del bloque de diseño y se establece un orden según el criterio del desarrollador, cabe mencionar que esta pantalla está dividida en dos pestañas, una gráfica y otra de entorno de programación, de la cual hablaremos más adelante, mostrado en la figura 71.

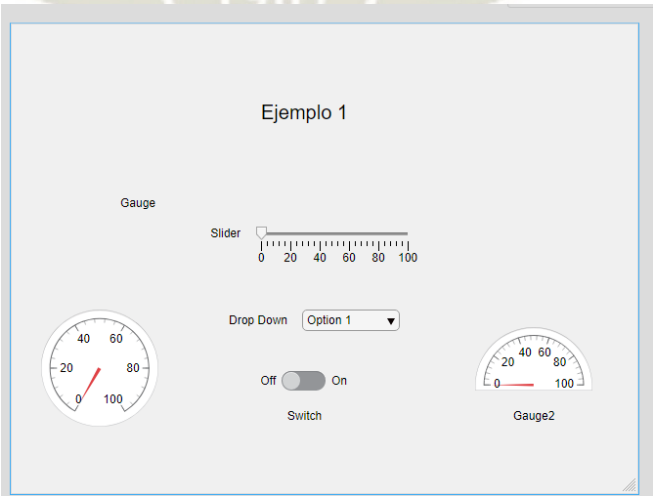


Fig. 71 Entorno gráfico de app designer
Fuente: Elaboración propia

d. Bloque de propiedades

En la figura 72 se encuentran las propiedades de cada uno de los componentes que colocamos en la pantalla de diseño, podemos configurar el nombre de etiqueta, asi como los valores que van a representar, colores y tema de presentación.

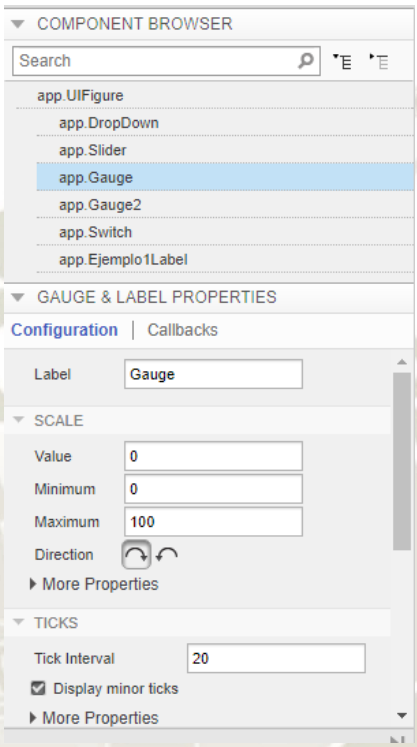


Fig. 72 Propiedades de una circular gauge con valor inicial 0
Fuente: Elaboración propia

4.7.2 Desarrollo gráfico

Para este proyecto, como se muestra en la figura 73, nuestra interfaz debe cumplir con las siguientes características:

- Configurar el puerto serie desde la app.
- Mostrar valores en tiempo real e indicar si existe algún error en el proceso.
- Enviar datos para modificar el set-point y técnica de controlador.
- Visualizar datos como error, SP, Vo, Yt en una gráfica acumulativa en tiempo real.
- Exportar una tabla de datos a Excel para posterior análisis.

Por lo tanto, procedemos a diseñar el entorno gráfico de la siguiente manera, mostrado en la figura 73, la primera pestaña será de configuración e indicadores visuales, la segunda pestaña, mostrará los datos en un gráfico pudiendo mostrar únicamente los datos seleccionados y con la opción a poder hacer zoom a un determinado objetivo. La ultima pestaña, mostrara una tabla con los datos obtenidos y el tiempo entre ellos, además de un botón que permita exportar dicha tabla a Excel, mostrado en la figura 75.

Fig. 73 Pantalla de configuración y visualización de sistemas térmicos
Fuente: Elaboración propia

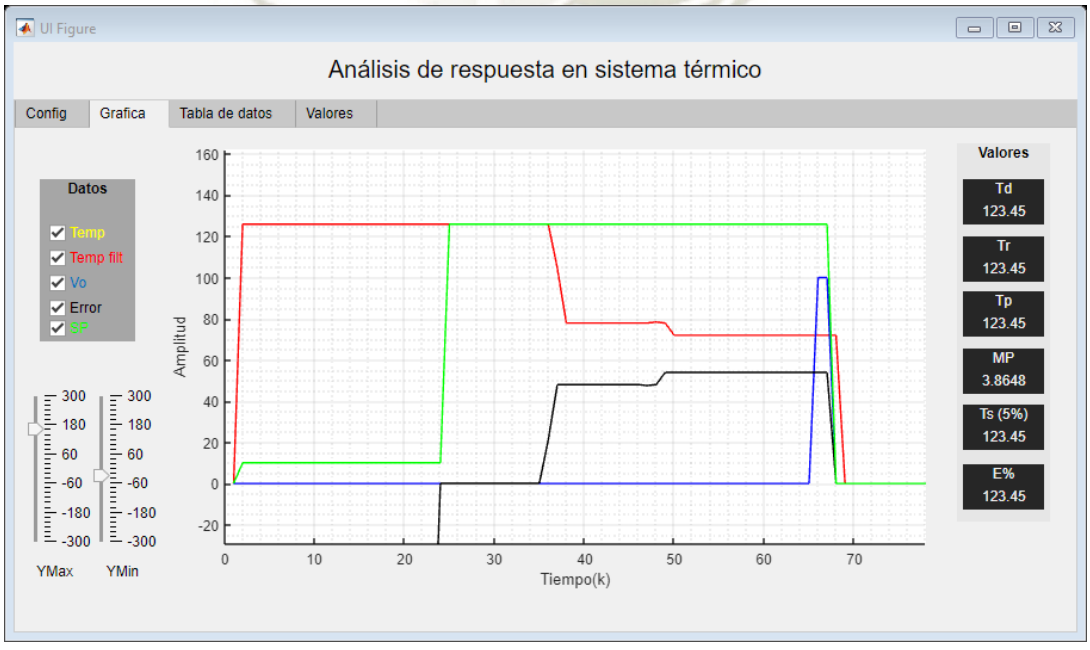
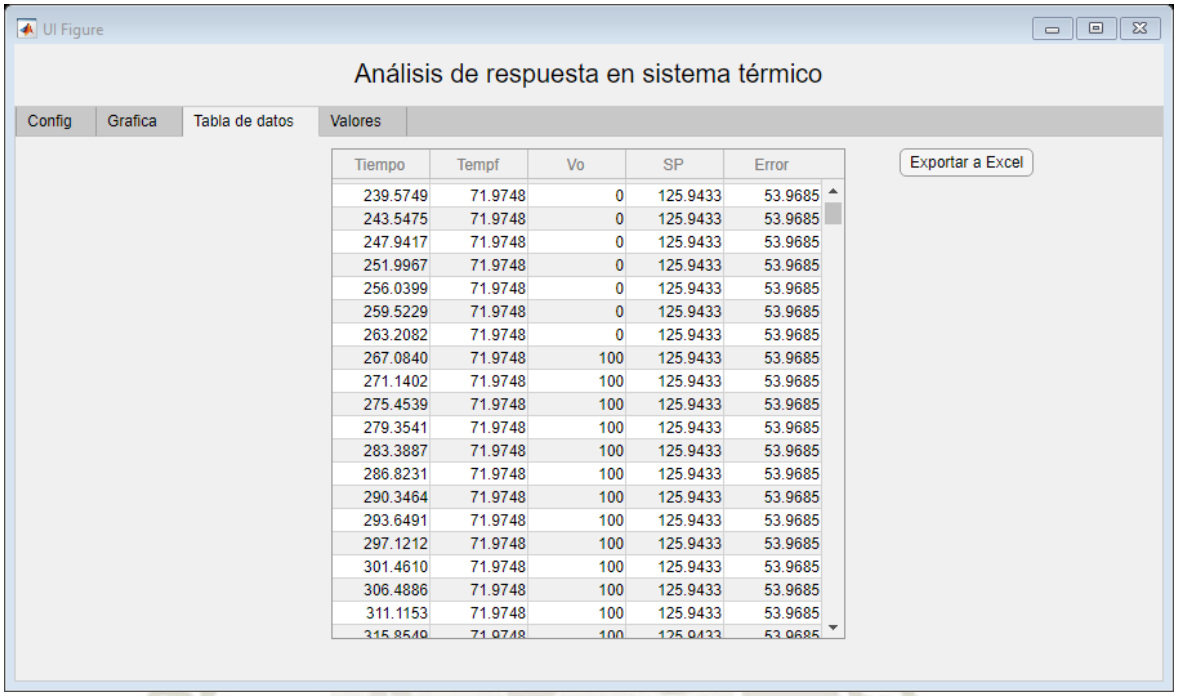


Fig. 74 Grafica de datos en tiempo real
Fuente: Elaboración propia

Como se puede observar en la figura 74, se tienen seleccionados diferentes tipos de señales, a un costado se pueden observar valores, los cuales se actualizan una vez que el sistema haya culminado el régimen transitorio.



Tiempo	Tempf	Vo	SP	Error
239.5749	71.9748	0	125.9433	53.9685
243.5475	71.9748	0	125.9433	53.9685
247.9417	71.9748	0	125.9433	53.9685
251.9967	71.9748	0	125.9433	53.9685
256.0399	71.9748	0	125.9433	53.9685
259.5229	71.9748	0	125.9433	53.9685
263.2082	71.9748	0	125.9433	53.9685
267.0840	71.9748	100	125.9433	53.9685
271.1402	71.9748	100	125.9433	53.9685
275.4539	71.9748	100	125.9433	53.9685
279.3541	71.9748	100	125.9433	53.9685
283.3887	71.9748	100	125.9433	53.9685
286.8231	71.9748	100	125.9433	53.9685
290.3464	71.9748	100	125.9433	53.9685
293.6491	71.9748	100	125.9433	53.9685
297.1212	71.9748	100	125.9433	53.9685
301.4610	71.9748	100	125.9433	53.9685
306.4886	71.9748	100	125.9433	53.9685
311.1153	71.9748	100	125.9433	53.9685
315.8549	71.9748	100	125.9433	53.9685

Fig. 75 Pestaña de visualización de tabla de datos HMI
Fuente: Elaboración propia

4.7.3 Programación

Para poder hacer funcionar todo este sistema, basta con programar cada bloque en lenguaje de Matlab, mezcla entre Java y C, la lógica de programación se explica en la figura 76, el método de programación está basado en la programación orientada a objetos, con una etapa gráfica y otra de funciones de control. Se puede diseñar la pantalla de visualización utilizando el panel de elementos y asignarle una acción mediante el código de programación.

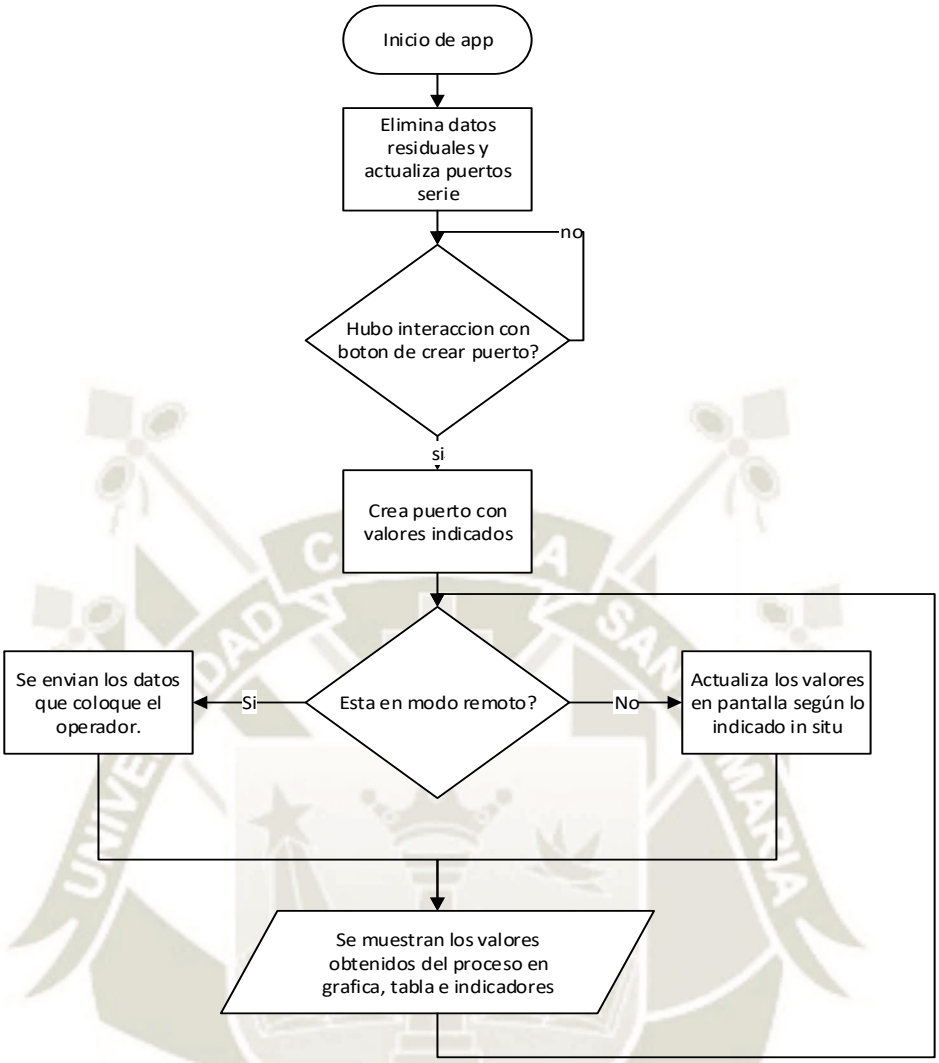


Fig. 76 Diagrama de flujo de interfaz HMI
Fuente: Elaboración propia

A continuación, se muestra un extracto donde se configura un botón mediante la programación de Matlab:

```
% Button pushed function: PararButton
        function PararButtonPushed(app, event)
            global captura
            captura = 0;
            app.CREATEButton.Enable = 'On';
            if strcmpi(app.BT.Status,'open')
                fclose(app.BT);
                app.ComunicacionLamp.Color = 'yellow';
            end
        end
```

Como se puede observar, trabaja entorno a funciones, cada acción que desarrollamos en el aplicativo, desencadena en una funcion que según lo configuremos, accionará de una u otra manera.

Para poder crear un puerto serie, utilizamos las propiedades del sistema en general, para esto declaramos un objeto que llamaremos BT.

```
properties (Access = private)
        BT
    end
```

Este objeto es de uso común en toda la app. Para poder interactuar con él, basta con declararlo en otra funcion.

```
function CREATEButtonPushed(app, event)
            delete(instrfind('Port',(app.Puertos.Value)));
            app.BT = serial(app.Puertos.Value);
            app.BT.BaudRate = str2double(app.BAUDRATE.Value);
            app.ComunicacionLamp.Color = 'yellow';
        end
```

Ahora que tenemos desarrollada nuestra aplicación, podemos pasar a la siguiente etapa, visualización de datos.

CAPITULO V
PRUEBAS E INTERPRETACIÓN DE RESULTADOS

CAPITULO V

5. PRUEBA Y ANÁLISIS DE RESULTADOS

Para la toma de datos se implementó un algoritmo de Matlab mostrado en la figura 77, que recepciona los datos del tablero de control y los procesa de modo que puede publicar los valores actuales de temperatura, señal de control porcentual, error y modo de trabajo. El muestreo y actualización de datos se da cada segundo.

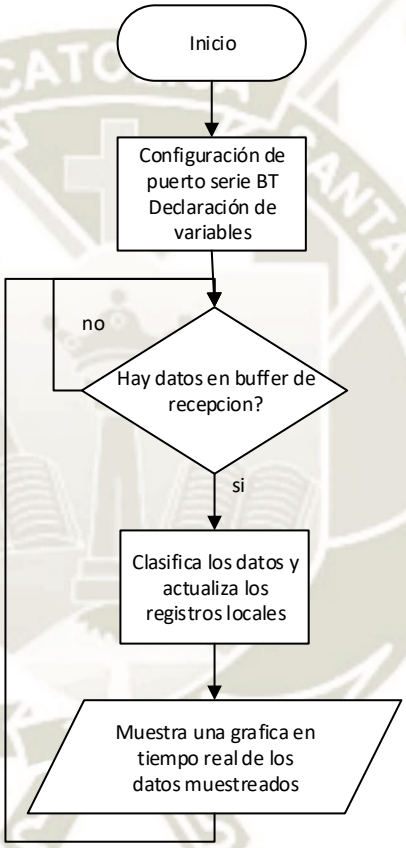


Fig. 77 Diagrama de flujo de procesamiento de datos
Fuente: Elaboración propia

5.1 Toma de datos del proceso con controlador PI

Al inicio se configuro el controlador PI mediante el método de ZN, no obstante, el sistema entregó una respuesta diferente a la esperada por lo que se tuvieron que modificar los parámetros de control, como se muestra en la figura 78, se pudo identificar que el controlador no estaba actuando en el tiempo previsto, por lo que se decidió modificar el tiempo de muestreo a través de interrupciones controladas por un timer de 16 bits.

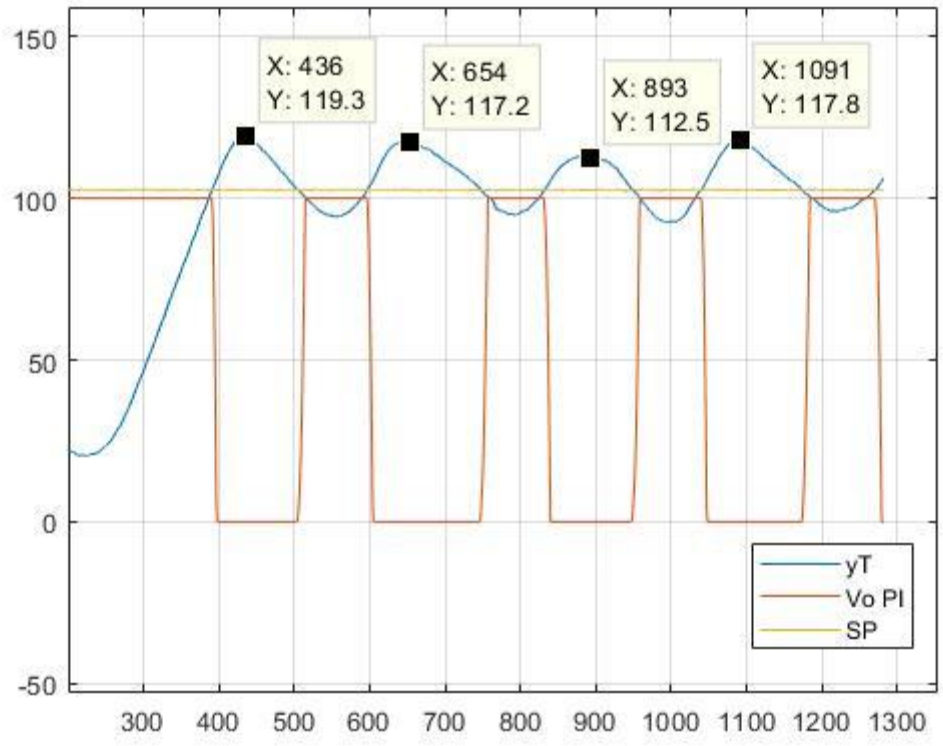


Fig. 78 Señal de respuesta oscilante PI
Fuente: Elaboración propia

Como se puede observar, se tiene una señal oscilante, con un periodo de oscilación promedio de 218 segundos o 4.5mHertz. Se procede a generar un tiempo de muestreo igual a 0.3Hz.

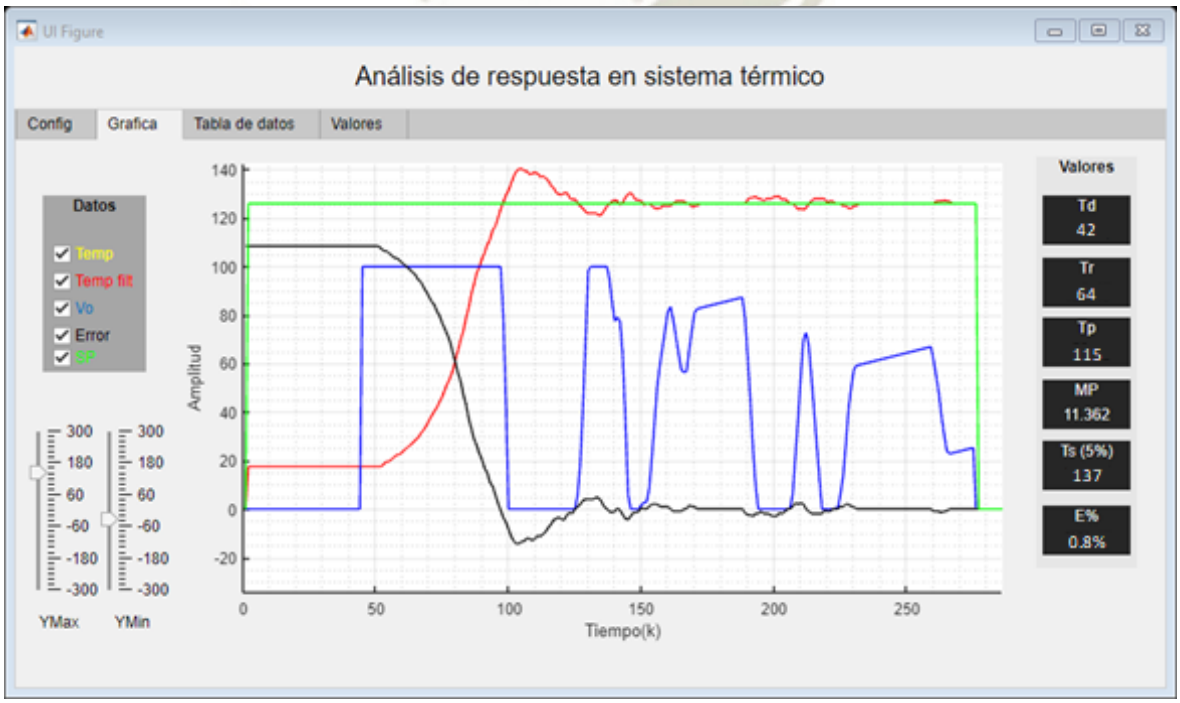


Fig. 79 Respuesta a control PI con HMI
Fuente: Elaboración propia

Como podemos observar de la figura 79, se tienen los valores ya definidos, como es el caso del tiempo de retardo (Td), Tiempo de subida (Tr), Tiempo pico (Tp), máximo sobre impulso (MP), tiempo de establecimiento (Ts) y error porcentual (E).

- Td = 42.
- Tr = 64.
- Tp = 115.
- MP = 11.32%.
- Ts = 137.
- Error = 0.8%.

5.2 Toma de datos del proceso con controlador Difuso

Durante el procesamiento de datos con este sistema, se identificó la salida de control variable en función al más mínimo error del sensor, como se puede observar en la figura 80, se tiene un controlador que responde al sistema de manera inmediata, manteniendo un error mínimo durante el régimen estable, eludiendo además perturbaciones.

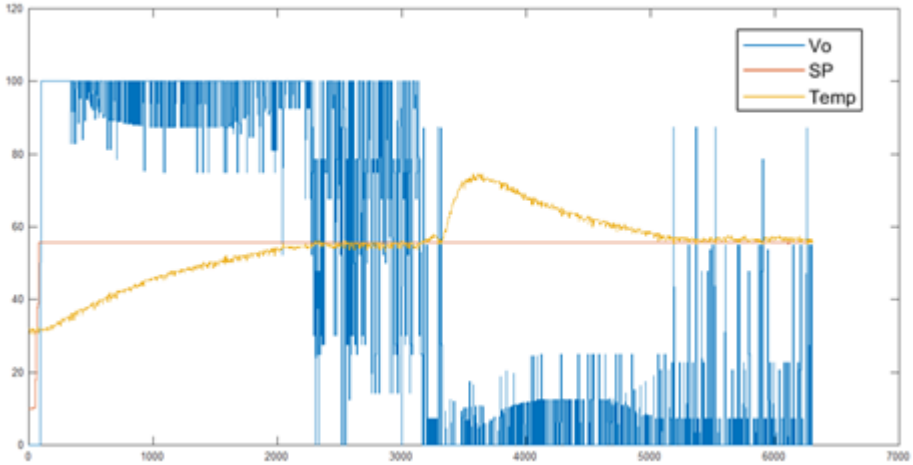


Fig. 80 Respuesta de un controlador difuso
Fuente: Elaboración propia

En una prueba con carga variable, inducción al error con un cambio en las condiciones del sensor, se identificó una frecuencia de muestreo de 10Hz, Como se puede observar, se tiene una respuesta sin sobre impulso y con una respuesta estable a partir de la muestra 1800 aproximadamente, lo que se traduce en 180 segundos.

Se tiene una perturbación en el sensor que se traduce como una alteración en la medición, ante la cual, la respuesta del sistema reacciona oportunamente manteniendo el sistema bajo control en el set point.

Por otro lado, tenemos una respuesta con picos de control, lo cual se debe a la variación del error derivada del error porcentual del transmisor de temperatura empleado, que a grandes rasgos se muestra la velocidad de acción del controlador frente a pequeños cambios en el sistema, manteniendo en promedio una salida adecuada para ejecutar el control y mantener el proceso en su set point.

Siguiendo las mismas condiciones que en la prueba con el controlador PI, se procede a utilizar el software desarrollado para analizar la respuesta al control difuso, lo que se muestra en la figura 81. Se puede observar una salida sub amortiguada y una respuesta de control estable.

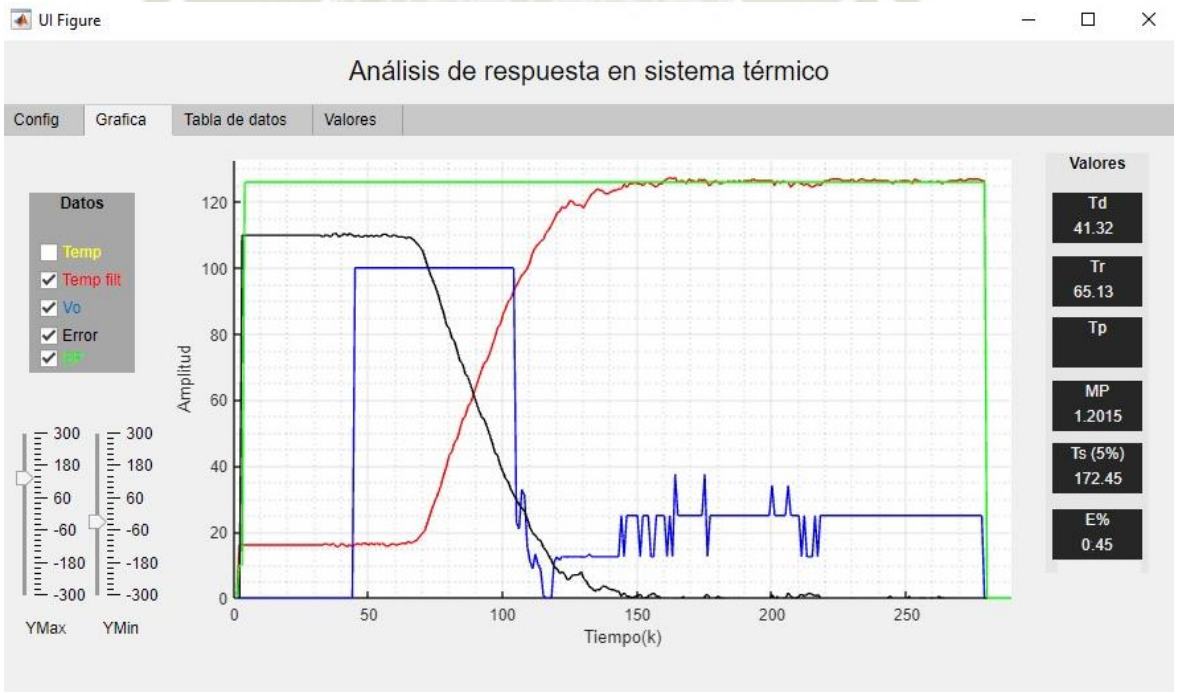


Figura 81 Respuesta difusa con filtro de ruido y sin perturbaciones.
Fuente: Elaboración Propia

5.3 Análisis de datos y gráficas comparativas

Para un análisis de respuesta, se va a considerar el tiempo de establecimiento, máximo sobre impulso y error porcentual.

A continuación, en la tabla 4 se van a comparar estos criterios.

Tabla 4 Tabla comparativa

Parámetro	Control PI	Control Difuso
Tipo de respuesta	Sub amortiguada	Sobre amortiguada
Tiempo de establecimiento	137 segundos	172 segundos
Máximo sobre impulso	11.32%	1.2%
Error porcentual	0.8%	0.45%

Fuente: Elaboración propia

Como se puede observar, si bien el controlador PI es más rápido, este presenta un error porcentual más elevado, así como un pico máximo de 11.32%, cosa que no sucede con el controlador difuso, por otro lado, el segundo ha demostrado ser un controlador robusto, capaz de soportar una perturbación sin presentar un pico negativo y manteniendo un error en régimen estable menor.

CAPITULO VI
COSTOS Y PRESUPUESTO

CAPITULO VI

6 COSTOS Y PRESUPUESTO

En un tema de costos, se plantea un precio base del equipo, tomando en consideración la estructura de fabricación modular, se puede considerar un rango de potencia variable.

Los costos de fabricación están dados en la siguiente tabla:

Tabla 5 Costos monetarios del proyecto

Equipo	Costo/Hora	Horas	Total
Diseño de circuitos en software CAD	S/ 40.00	12	S/ 480.00
Implementación de circuitos en placa física	S/ 40.00	4	S/ 160.00
Desarrollo de ingeniería de control	S/ 50.00	48	S/ 2,400.00
Programación de microcontroladores	S/ 50.00	24	S/ 1,200.00
Programación de software HMI	S/ 60.00	48	S/ 2,880.00
Armado de tablero de control	S/ 40.00	4	S/ 160.00
Materiales	-	-	S/ 350.00
Herramientas, software e instrumentos	-	-	S/ 1,500.00
Total			S/ 9,130.00

Fuente: Elaboración Propia

En la tabla anterior se describen los costos de fabricación del prototipo de control, para una producción en masa este costo se reduciría considerablemente ya que los costos de programación y software se dividirían entre una cantidad de equipos a implementar, dejando un porcentaje de los S/. 9,130.00 nuevos soles que se asume en el prototipo a una fracción del producto en masa que incluye como minimo los S/. 350.00 nuevos soles de materiales y el pago fracturado de los demás ítems.

Los costos de software se basan en las horas dedicadas a la programación del algoritmo sin considerar un monto adicional de pago por especialidad.

Estos costos también se pueden expandir al tipo de control y respuesta del sistema, al ser una técnica de control con intervalos de salida neutra y time off, se logra minimizar el estrés generado en los actuadores de la planta, reduciendo asi el tiempo de macro

funcionamiento y alargando el tiempo de vida del instrumento. Esto se puede demostrar en especial con un sensor de mejor calidad, mediante el cual, se presentan menores variaciones en la lectura de temperatura y permite un control de mayor precisión.

Como un ejemplo, se puede calcular que el precio de un lote de 10 equipos de control se vería calculado por:

Tabla 6 Costo por equipo para un lote de 10 unidades

Número de equipos	10
Materiales	S/ 350.00
Implementación	S/ 320.00
Diseño y programación/equipos	S/ 696.00
Herramientas y software/equipo	S/ 150.00
Costo por equipo	S/ 1,516.00

Fuente: Elaboración propia

Lo cual se calcula manteniendo el costo de materiales e implementación indiferente al número de unidades y dividendo los costos de diseño, programación y uso de herramientas según el número de unidades, quedando así un precio final de S/ 1,516.00 por equipo.

CAPITULO VII
CONCLUSIONES

CAPITULO VII

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

a. Se diseñó un controlador difuso universal para hornos semi-industriales del tipo eléctrico en un tablero de control, se demostró el funcionamiento y se observó mediante la respuesta transitoria que el controlador es estable, preciso y robusto.

b. Se aplicó la teoría de control moderna para demostrar su efectividad en régimen transitorio, sin sobre impulsos, frente a los clásicos controladores PI en un proceso de control de temperatura.

c. Se desarrolló una interfaz gráfica capaz de recopilar datos del proceso sin la necesidad de utilizar cables hacia el tablero de control, utilizando el protocolo IEEE 802.15.

d. Se demostró que se puede implementar un sistema inteligente basado en lógica programable en microcontrolador de 8 bits, controlando el proceso in situ, sin la necesidad de intervención de los operadores en sala de control y a su vez transmitiendo datos hacia una interfaz gráfica para su posterior análisis y registro

e. Se implementó un gabinete de control con 3 placas de circuito impreso independientes, funcionando como un sistema modular, las cuales son: etapa de control, interfaz y etapa de potencia. Esta última se puede intercambiar según las necesidades del dispositivo a controlar. Asi mismo, cuenta con borneras de protección en la etapa de control y potencia.

f. Se desarrollaron 5 conjuntos difusos para la entrada error y 3 para la entrada derivada del error, los cuales fueron utilizados para desarrollar 11 reglas difusas que permiten predecir la tendencia del error en un proceso lento, incrementando así la exactitud del controlador.

g. Se decidió utilizar la técnica de defusificación tipo Sugeno, ya que esta técnica permite un rápido procesamiento, lo cual es un factor importante en nuestro sistema embebido basado en un microcontrolador de 8 bits.

h. Se aplicaron diferentes conceptos teóricos y prácticas aprendidas durante mi formación en la escuela profesional de ingeniería electrónica, tales como: modelamiento de sistemas y control avanzado, electrónica de potencia, sensores y actuadores, comunicación de datos, programación en C. lógica de microcontroladores, física y matemática avanzada.

7.2 Recomendaciones para trabajos futuros

a. Se recomienda utilizar microcontroladores certificados bajo la norma IEC 61439, que permite que los circuitos diseñados trabajen bajo el concepto de seguridad y calidad industrial. Existen microcontroladores de la familia microchip que cumplen con esta certificación y se podría implementar estos algoritmos de programación en estos dispositivos sin complicaciones.

b. Se recomienda investigar el control PID difuso para futuras aplicaciones, hacer pruebas y contrastar resultados.

c. Para mayor precisión y reducir el desgaste sobre el actuador, se recomienda utilizar RTD en lugar de termocuplas, en específico la RTD modelo PT100, cuyo rango de trabajo normalizado se ubica entre 0 y 400 grados, así como la aplicación de transmisores de temperatura industriales.

d. Se recomienda la fabricación de las placas de circuitos en forma masiva utilizando los servicios de empresas especializadas para reducir el ruido generado en la etapa de potencia y que pueda influir en el circuito de control.

e. Se recomienda realizar pruebas de resistividad y aislamiento en los componentes a controlar previo a la instalación de este tablero de control.

f. Para ventas se recomienda realizar un manual de usuario, donde se explique la instalación del aplicativo HMI y configuración del mismo.

REFERENCIA

[1] SENA, “Instalaciones eléctricas ley de Joule”, Colombia, 2017.

[2] A. Creus, “Instrumentación Industrial”, 8va Edición, editorial Alfa Omega.

[3] SENA, “Controladores Lógicos Programables”, Colombia, 2015.

[4] S. Salas, “Todo sobre sistemas embebidos”. Lima: Universidad Peruana de Ciencias Aplicadas, 2015.

[5] P. Flotante, (2013), “Sistema Bolt 18F2550”, Available: https://www.puntoflotante.net/SISTEMA-BOLT-18F2550-INFORMACION-TECNICA.pdf

[6] K. Ogata, “Ingeniería de control moderna”, 5ta edición, Editorial: PEARSON

[7] C. Pardo, (Marzo 2021), “Descripción de señales de control PID en un sistema”, Available: https://www.picuino.com/es/arduprog/control-pid.html

[8] S. Álvarez de Miguel, J. Mollocana, C. García, M. (2017), “Identification model and PI and PID controller design for a novel electric air heater”, Automatika

[9] Matlab, “fuzzy logic toolbox user’s guide”, 2020

[10] M. H. Rashid, “Electrónica de Potencia, circuitos, dispositivos y aplicaciones”, 3ra edición, Editorial: PEARSON

[11] A. VERGARA, E. SALAZAR, “Obtención de la función de transferencia de un motor de DC mediante el análisis de la curva de reacción”. Revista de Aplicación Científica y Técnica. 2017

[12] Microchip, “PIC18F2550”, Data Sheet, DS39632, 2009

[13] STMicroelectronics, “Snubberless logic leven and standard 16 A Triacs”, Data Sheet, DS2114, May 2018.

[14] A. P. Malvino, (2000). “Principios de Electrónica”, McGraw Hill, ISBN 84-481-2568

[15] “Electrónica de Potencia”, nota de clase, Programa Profesional de Ingeniería Electrónica, Universidad Católica de Santa María, 2017 II.

[16] Maxim Integrated, “MAX6675 Cold-Juntion-Compensated”, Data sheet, 19-223, 2015.

ANEXOS

A. DIAGRAMA DE BLOQUES

a. DIAGRAMA DE BLOQUES DEL SISTEMA EN GENERAL

B. DIAGRAMAS ESQUEMÁTICOS

a. CONTROLADOR PRINCIPAL
b. INTERFAZ
c. ETAPA DE POTENCIA

C. CÓDIGOS DE PROGRAMACIÓN

a. CÓDIGO DE TOMA Y ANÁLISIS DE DATOS
b. CÓDIGO DE CONTROL PRINCIPAL
c. CÓDIGO DE INTERFAZ
d. CÓDIGO DE HMI

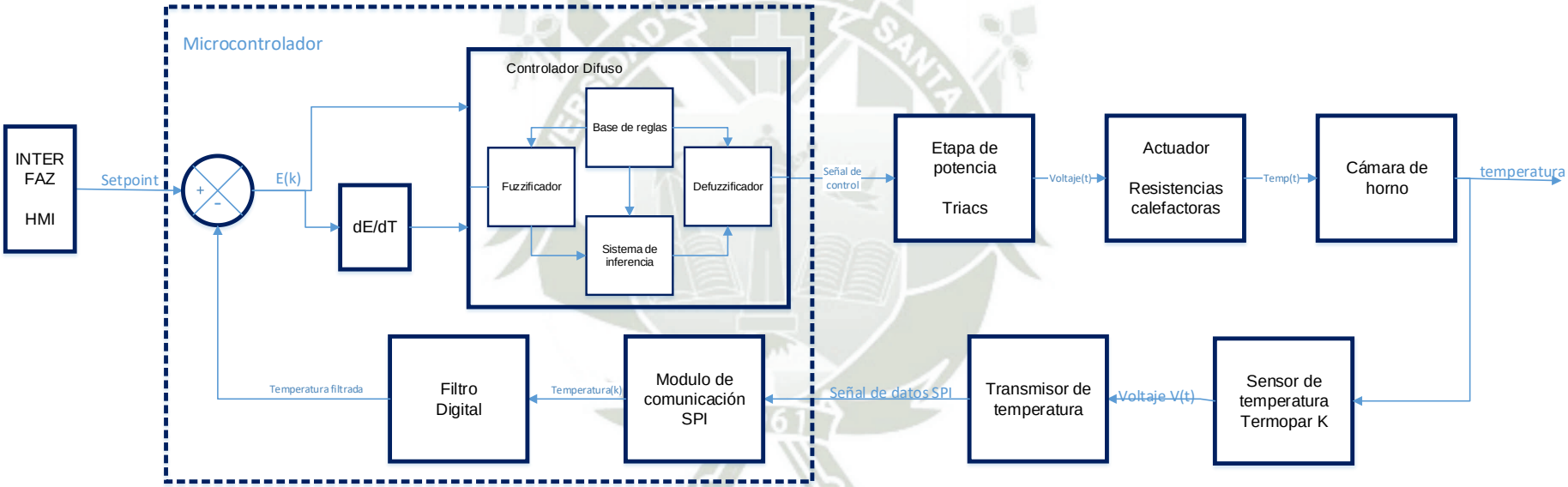
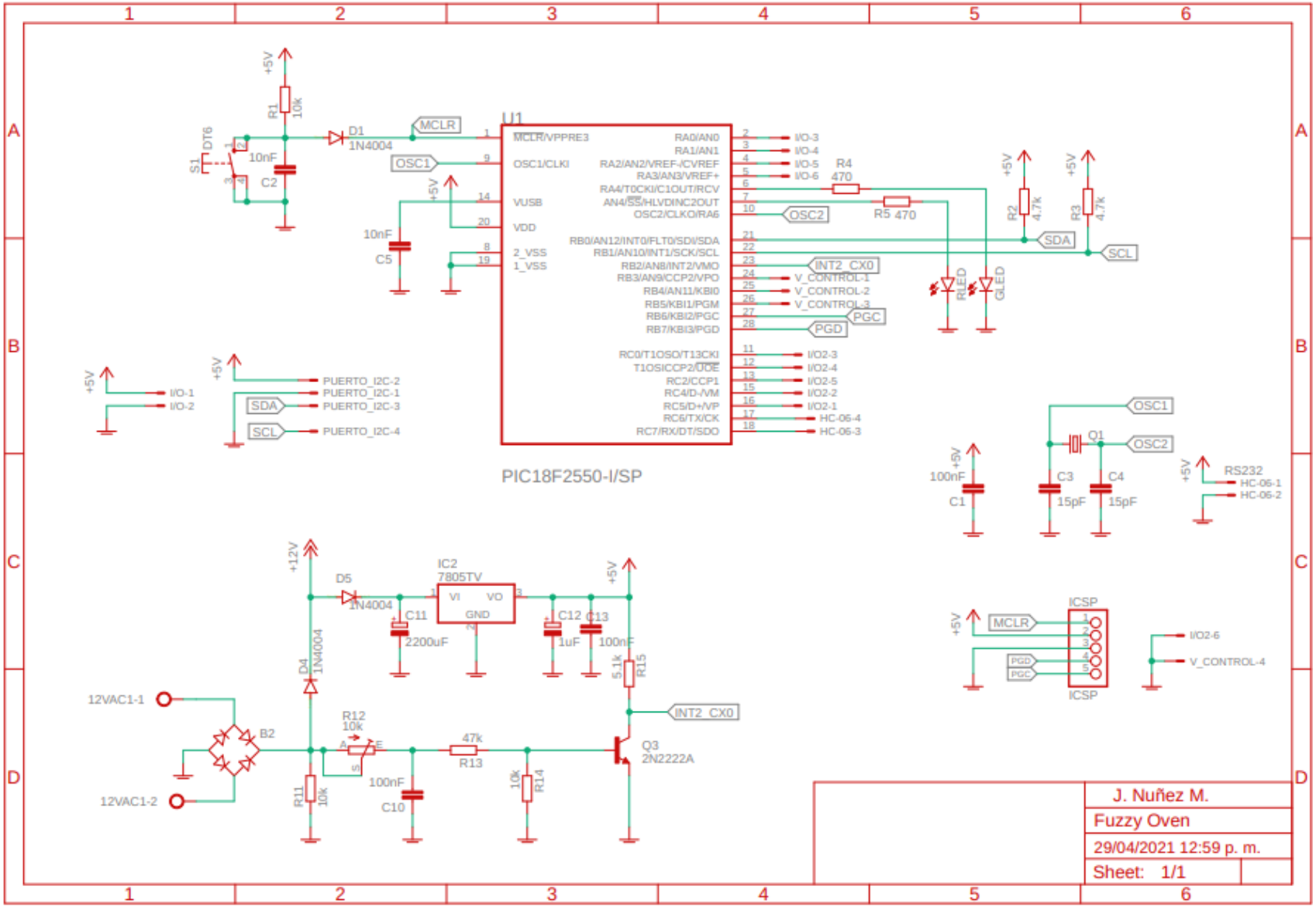
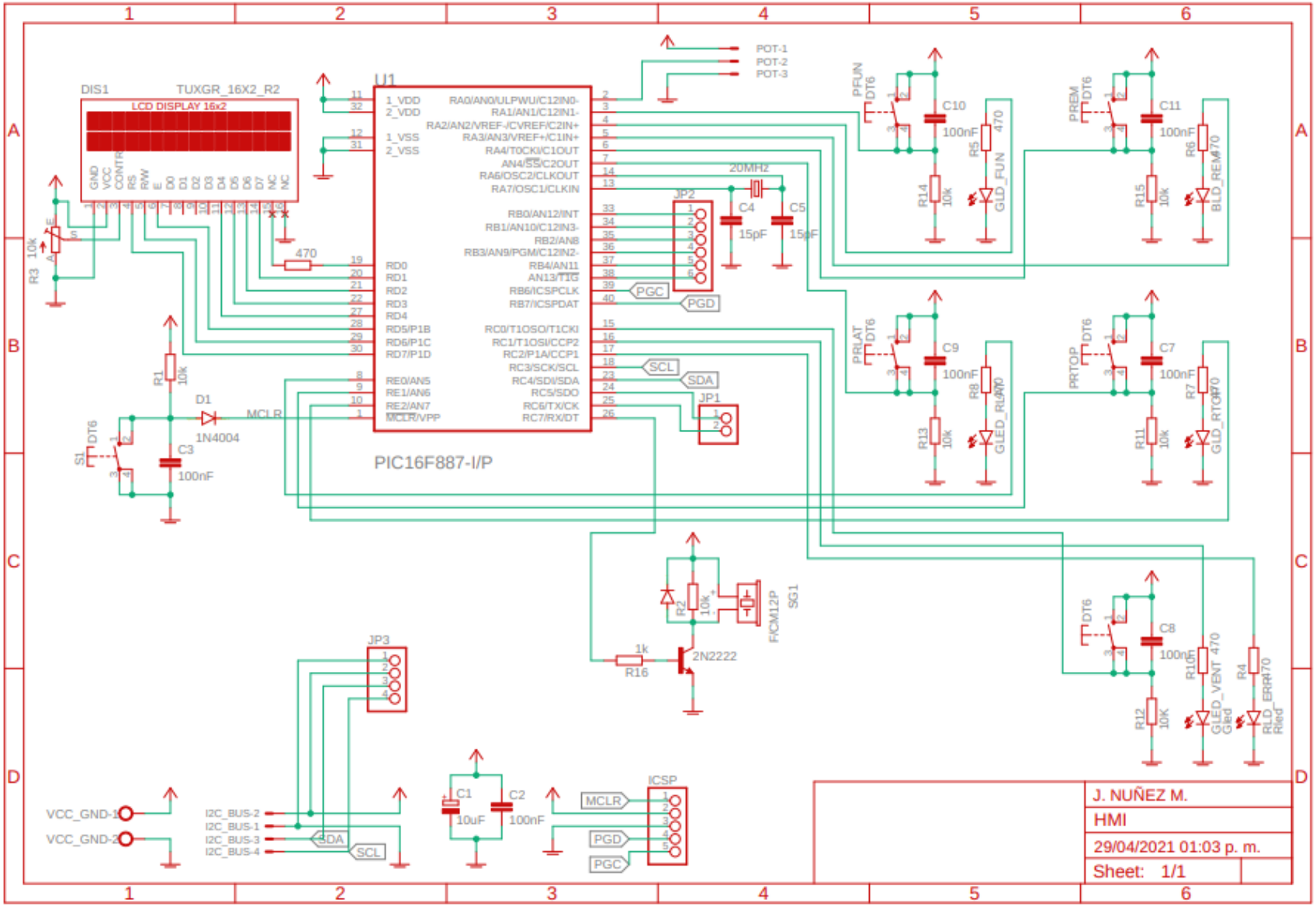
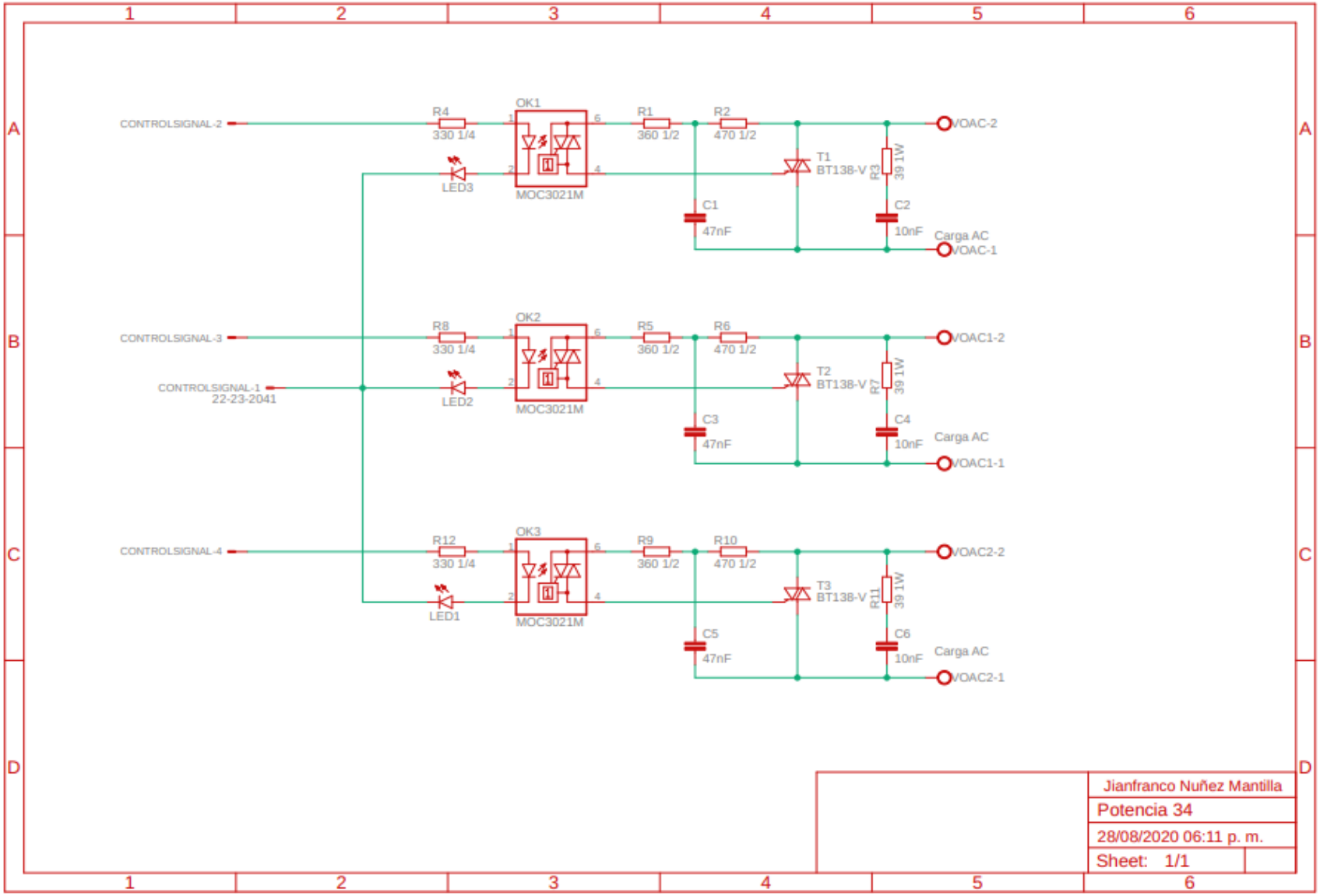


Figura 82 Diagrama de bloques del sistema
Fuente: Elaboración propia


PIC18F2550-I/SP
U1
MCLR
OSC1
OSC2
SDA
SCL
INT2_CX0
PGC
PGD
V_CONTROL-1
V_CONTROL-2
V_CONTROL-3
PUERTO_I2C-1
PUERTO_I2C-2
PUERTO_I2C-3
PUERTO_I2C-4
I/O-1
I/O-2
I/O-3
I/O-4
I/O-5
I/O-6
I/O2-1
I/O2-2
I/O2-3
I/O2-4
I/O2-5
I/O2-6
HC-06-1
HC-06-2
HC-06-3
HC-06-4
RS232
ICSP
IC2
7805TV
VI
VO
GND
Q1
Q3
2N2222A
B2
12VAC1-1
12VAC1-2
D1
D4
D5
1N4004
R1
R2
R3
R4
R5
R11
R12
R13
R14
R15
C1
C2
C3
C4
C5
C10
C11
C12
C13
10nF
100nF
2200uF
1uF
15pF
10k
4.7k
470
5.1k
47k
RLED
GLED
+5V
+12V
J. Nuñez M.
Fuzzy Oven
29/04/2021 12:59 p. m.
Sheet: 1/1


DIS1
TUXGR_16X2_R2
LCD DISPLAY 16x2
U1
PIC16F887-I/P
POT-1
POT-2
POT-3
20MHz
C4
15pF
C5
15pF
JP2
PGC
PGD
SCL
SDA
JP1
MCLR
R1
10k
D1
1N4004
S1
C3
100nF
470
R3
10k
PFUN
PREM
PRAT
PRTOP
C10
C11
C9
C7
C8
100nF
R14
R15
R13
R11
R12
10k
R5
R6
R8
R7
R16
470
GLD_FUN
BLD_REM
GLED_RLAT
GLD_RTOP
GLED_VENT
RLD_ERR
R2
10k
1k
2N2222
FCM12P
SG1
JP3
I2C_BUS-1
I2C_BUS-2
I2C_BUS-3
I2C_BUS-4
VCC_GND-1
VCC_GND-2
C1
10uF
C2
100nF
ICSP
J. NUÑEZ M.
HMI
29/04/2021 01:03 p. m.
Sheet: 1/1


1
2
3
4
5
6
A
B
C
D
CONTROLSIGNAL-2
R4
330 1/4
OK1
MOC3021M
LED3
R1
360 1/2
R2
470 1/2
T1
BT138-V
R3
39 1W
C1
47nF
C2
10nF
VOAC-2
Carga AC
VOAC-1
CONTROLSIGNAL-3
R8
330 1/4
OK2
MOC3021M
LED2
R5
360 1/2
R6
470 1/2
T2
BT138-V
R7
39 1W
C3
47nF
C4
10nF
VOAC1-2
Carga AC
VOAC1-1
CONTROLSIGNAL-1
22-23-2041
CONTROLSIGNAL-4
R12
330 1/4
OK3
MOC3021M
LED1
R9
360 1/2
R10
470 1/2
T3
BT138-V
R11
39 1W
C5
47nF
C6
10nF
VOAC2-2
Carga AC
VOAC2-1
Jianfranco Nuñez Mantilla
Potencia 34
28/08/2020 06:11 p. m.
Sheet: 1/1

A. Código de controlador principal

```
#include <18F2550.h>
#include <stdlib.h>

#FUSES HSPLL, PLL5, CPUDIV2
#FUSES NOWDT                 //No Watch Dog Timer
#FUSES NOIESO                //Internal External Switch Over mode disabled
#FUSES NOBROWNOUT              //No brownout reset
#FUSES NOVREGEN              //USB voltage regulator disabled
#FUSES NOPBADEN              //PORTB pins are configured as digital I/O on RESET
#FUSES NOLPT1OSC             //Timer1 configured for higher power operation
#FUSES NOLVP                 //No low voltage programing,
#FUSES NOXINST

#use delay(clock=48000000,crystal=20000000)

#use rs232(STREAM=Blue,baud=38400,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
#use i2c(Master,fast=100000,sda=PIN_B0,scl=PIN_B1,FORCE_HW)

#define ID_HMI_W    0xA0
#define ID_HMI_R    0xA1
#define TC_CLK    PIN_A0
#define TC_CS     PIN_A1
#define TC_DATA   PIN_A2
#include "max6675.c"

#define Vo1       PIN_B3
#define Vo2       PIN_B4
#define Vo3       PIN_B5
#define Gled      PIN_A4
#define Rled      PIN_A5

#BYTE SSPSTAT = 0xFC7
#BYTE SSPBUF = 0xFC9
#bit  BUFFSTAT = SSPSTAT.0
#bit  RWSTAT = SSPSTAT.2

//Variables globales
 float SP,Et,Et0,dEt,Yt,Ut,Ut0,Ytf;
 float time;
 int16 TIEMPO;
 int ESTADO;
 float q0 = 5.712;
 float q1 = -1.444;
 int contador;

#byte PULSADORES = ESTADO
#bit  Pvent = PULSADORES.4
#bit  Pfupid = PULSADORES.3
```

```
#bit  Popenloop = PULSADORES.2
#bit  Prem = PULSADORES.1
#bit  Ponoff = PULSADORES.0

 float emuy_neg=0;
 float eneg=0;
 float emin=0;
 float epos=0;
 float emuy_pos=0;
 float dneg=0;
 float dmin=0;
 float dpos=0;
 int16 SETP;
 float w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11;

 int1 FUPID,ONOFF,REM,VENT,OPENLOOP;
 int16 x=0;

 #define muyA  100
 #define Alto  75
 #define Medio 50
 #define Bajo  25
 #define MuyB  0

#INT_EXT2
void EXT2_isr(void)
{
  x++;
  output_low(Vo1);
  output_low(Vo2);
  output_low(Vo3);
  set_timer0(TIEMPO);
  //set_timer3(TIEMPO);
}

#INT_TIMER0
void  TIMER0_isr(void)
{
  output_high(Vo1);
  output_high(Vo2);
  output_high(Vo3);
}

void actualiza_var()
{
   char PV[6];
   CHAR BUFF[6];

   Yt=do_everything();
   Ytf = (0.23*Yt) + (Ytf*0.77);
   sprintf(PV,"%3.2f",Ytf);
```



```
    i2c_start();
    i2c_write(ID_HMI_W); //escribe
    i2c_write(PV[0]);
    i2c_write(PV[1]);
    i2c_write(PV[2]);
    i2c_write(PV[3]);
    i2c_write(PV[4]);
    i2c_write(PV[5]);
    i2c_stop();
    delay_ms(10);
    i2c_start();
    i2c_write(ID_HMI_R);
    ESTADO = i2c_read();
    BUFF[0] = i2c_read();
    BUFF[1] = i2c_read();
    BUFF[2] = i2c_read();
    BUFF[3] = i2c_read(0);
    i2c_stop();

   SETP = atol(BUFF);

   SP = 0.28348 * SETP + 10;
   Et0 = Et;
   Et  = SP-Ytf;
   dEt = Et-Et0;
}

float max(float x,float y){
   float resultado;
   if(x>y) resultado=x;
   else resultado=y;
   return(resultado);
}

void Fuzzyficacion_err()
{
   emuy_neg=0;
   eneg=0;
   emin=0;
   epos=0;
   emuy_pos=0;

   if(Et<-40)             emuy_neg=1;
   if(Et>=-40 && Et<=-15)  emuy_neg=-0.04*Et-0.6;
   if(Et>=-50 && Et<=-30)  eneg=0.05*Et+2.5;
   if(Et>=-30 && Et<=-10)  eneg=1;
   if(Et>=-10 && Et<=-1)   eneg=-0.1111*Et-0.1111;
   if(Et>=-10 && Et<=0)    emin=0.1*Et+1;
   if(Et>=0   && Et<=10)   emin=-0.1*Et+1;
   if(Et>=1   && Et<=10)   epos=0.1111*Et-0.1111;
```



```
  if(Et>=10  && Et<=30)   epos=1;
  if(Et>=30  && Et<=50)   epos=-0.05*Et+2-5;
  if(Et>=15  && Et<=40)   emuy_pos=0.04*Et-0.6;
  if(Et>40)               emuy_pos=1;
}

void Fuzzyficacion_der()
{
  dneg=0;
  dmin=0;
  dpos=0;

  if(dEt<-2)              dneg=1;
  if(dEt>=-2 && dEt<=-1)  dneg=-dEt-1;
  if(dEt>=-3 && dEt<=-1)  dmin=0.5*dEt+1.5;
  if(dEt>=-1 && dEt<=1)   dmin=1;
  if(dEt>=1  && dEt<=3)   dmin=-0.5*dEt+1.5;
  if(dEt>=1  && dEt<=2)   dpos=dEt-1;
  if(dEt>2)               dpos=1;
}

void Reglas_inferencia()
{
  w1=0;
  w2=0;
  w3=0;
  w4=0;
  w5=0;
  w6=0;
  w7=0;
  w8=0;
  w9=0;
  w10=0;
  w11=0;

  if(emuy_neg>0)       w1=emuy_neg;       //muy bajo
  if(eneg>0 && dneg>0) w2=max(eneg,dneg); //muy bajo
  if(eneg>0 && dmin>0) w3=max(eneg,dmin); //muy bajo
  if(eneg>0 && dpos>0) w4=max(eneg,dpos); //baja
  if(emin>0 && dneg>0) w5=max(emin,dneg); //baja
  if(emin>0 && dmin>0) w6=max(emin,dmin); //bajo
  if(emin>0 && dpos>0) w7=max(emin,dpos); //media
  if(epos>0 && dneg>0) w8=max(epos,dneg); //alta
  if(epos>0 && dmin>0) w9=max(epos,dmin); //alta
  if(epos>0 && dpos>0) w10=max(epos,dpos);//alta
  if(emuy_pos>0)       w11=emuy_pos;      //muy alta
}

void Defuzzificacion(){
  Ut=(Bajo*(w4+w5+w6) + (w7)*Medio + Alto*(w8,w9,w10) + muyA*(w11));
  Ut=Ut/(w1+w2+w3+w4+w5+w6+w7+w8+w9+w10+w11);
```



```
  if(Ut>99.9) Ut = 100;
  if(Ut<0.1) Ut = 0;
}

void Open_loop(){
  Ut = 0.0978 * SETP;
  if(Ut>99.9) Ut = 100;
}

void PID(){
  Ut0 = Ut;
  Ut = Ut0 + q0 * Et - q1 * Et0;
  if(Ut>99.9) Ut = 100;
  if(Ut<0.1) Ut = 0;
}

void main()
{
  setup_adc_ports(NO_ANALOGS);
  set_tris_A(0b01001100);
  set_tris_B(0b11000101);
  set_tris_C(0b10111111);

  output_A(0x00);
 // output_B(0x00);
  output_C(0x00);

  output_high(Rled);
  output_high(Gled);

  delay_ms(1500);

  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);      //10.9 ms overflow
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_2);      //10.9 ms overflow
  setup_timer_3(T3_INTERNAL | T3_DIV_BY_2);

  EXT_INT_EDGE(L_TO_H);
  enable_interrupts(INT_EXT2);
  enable_interrupts(INT_TIMER0);
  enable_interrupts(INT_TIMER1);

  i2c_init(true);
  i2c_start();
  i2c_stop();

  output_low(Rled);
  output_low(Gled);
  printf("inicio\n\r");

  while(TRUE)
  {
```



```
if(Ponoff) ONOFF = 1;
else ONOFF = 0;
if(Prem) REM = 1;
else REM = 0;
if(Pfupid) FUPID = 1;
else FUPID = 0;
if(Popenloop) OPENLOOP = 1;
else OPENLOOP = 0;
if(Pvent) VENT = 1;
else VENT = 0;

if(ONOFF){

 output_low(Rled);
 output_high(Gled);

 if(OPENLOOP){
   actualiza_var();
   Open_loop();
   printf("OPEN LOOP\n\r");
 }
 else{
  if(!FUPID){
   actualiza_var();
   Fuzzyficacion_err();
   Fuzzyficacion_der();
   Reglas_inferencia();
   Defuzzificacion();
   printf("FUZZY\n\r");
    }
  else{
   actualiza_var();
   PID();
   printf("PID\n\r");
 }
 }
 enable_interrupts(GLOBAL);

 time = 498*Ut + 15735;
 TIEMPO = time;
 if(TIEMPO<15735) TIEMPO=15735;
 if(TIEMPO>65435) TIEMPO=65435;
}

if(!ONOFF){
  actualiza_var();
  disable_interrupts(GLOBAL);
  output_low(Vo2);
  output_low(Vo1);
  output_low(Vo3);
  output_high(Rled);
```

```
      output_low(Gled);
      printf("OFF\n\r");
     }

    if(VENT){
      enable_interrupts(INT_TIMER3);
     }
    else disable_interrupts(INT_TIMER3);

     printf("Vo=%3.6f\n\r",Ut);
     printf("Er=%3.6f\n\r",Et);
     printf("Yt=%3.6f\n\r",Yt);
     printf("Ft=%3.6f\n\r",Ytf);
     printf("SP=%3.6f\n\r",SP);
     printf("Estado=%u\n\r",ESTADO);

     delay_ms(723);
   }
}
```

B. Código de Interfaz

```
 #include <16F887.h>
#fuses NOWDT
#FUSES HS
#FUSES NOPUT
#FUSES NOBROWNOUT
#FUSES NOLVP
#FUSES NOPROTECT

#device ADC=10
#use delay(clock=20000000)

#use i2c(SLAVE,sda=pin_C4,scl=pin_C3,address=0xA0,FORCE_HW)

#define LCD_ENABLE_PIN PIN_D5
#define LCD_RS_PIN PIN_D7
#define LCD_RW_PIN PIN_D6
#define LCD_DATA4 PIN_D4
#define LCD_DATA5 PIN_D3
#define LCD_DATA6 PIN_D2
#define LCD_DATA7 PIN_D1
#define LCD_LED PIN_D0

#include <lcd.c>

#define PFUN   PIN_A1
#define LFUN   PIN_A2
#define LREM   PIN_A3
#define PREM   PIN_A4
#define PRLAT  PIN_A5
#define LRLAT  PIN_E0
#define PRTOP  PIN_E1
#define LRTOP  PIN_E2
#define PVENT  PIN_C0
#define LVENT  PIN_C1
#define LERROR PIN_C2
#define BUZZER PIN_C7

#byte SSPSTAT = 0xFC7
#bit BFBIT=SSPSTAT.0
#bit STARTBIT=SSPSTAT.3
#bit STOPBIT=SSPSTAT.4

#byte SSPCON1=0xFC6

//VARIABLES
INT16 SETPOINT;
INT SPL,SPM;
INT ESTADO=0;
char PV[6];
```



```
unsigned int8 address, bufferI2C[6];

#INT_SSP

void i2c_isr()
{
  int state;
  int j=0;
  state = i2c_isr_state();
  if(state==0)
    i2c_read();
  if(state == 0x80)
    i2c_read(2);
  if(state>=0x80) {
    i2c_write(bufferI2C[state-0x80]);
     }
  else if(state>0){
    PV[state-1] = i2c_read();
 }
}


void main()
{
  set_tris_A(0b00110011);
  set_tris_B(0b11111111);
  set_tris_C(0b01111001);
  set_tris_D(0b00000000);
  set_tris_E(0b00000010);

  lcd_init();
  i2c_init(true);
  setup_adc_ports(AN0);
  setup_adc(ADC_CLOCK_DIV_32);

  output_high(LFUN);
  output_high(LREM);
  output_high(LRLAT);
  output_high(LRTOP);
  output_high(LVENT);
  output_high(LERROR);
  output_high(BUZZER);

  printf(LCD_PUTC,"\fProControl\n\r CONTROL DIFUSO");
  delay_ms(200);
  output_low(BUZZER);
  delay_ms(1500);
  set_adc_channel(0);

  output_low(LFUN);
  output_low(LREM);
```

```
output_low(LRLAT);
output_low(LRTOP);
output_low(LVENT);
output_low(LERROR);
output_low(BUZZER);

SSPCON1 = 0x3E;//0b00111110;
enable_interrupts(INT_SSP);
enable_interrupts(GLOBAL);

float SP;

delay_ms(400);

while(TRUE){

 SETPOINT=read_adc();
 SPL = SETPOINT & 0x00FF;
 SPM = SETPOINT >> 8;
 SP = 0.28348 * SETPOINT + 10;

 sprintf(bufferI2C,"%u%Lu",ESTADO,SETPOINT);
 LCD_PUTC("\f");
 if((ESTADO & 0b00000001) == 0x01){
   LCD_GOTOXY(15,1);
   printf(LCD_PUTC,"ON");
   }
 else{
   LCD_GOTOXY(14,1);
   printf(LCD_PUTC,"OFF");
 }

 if((ESTADO & 0b00000100) == 0x04){
   LCD_GOTOXY(1,1);
   printf(LCD_PUTC,"MODO OPN L");
   }
 else IF((ESTADO & 0b00001000) == 0x08){
   LCD_GOTOXY(1,1);
   printf(LCD_PUTC,"MODO PID");
 }
 ELSE{
   LCD_GOTOXY(1,1);
   printf(LCD_PUTC,"MODO FUZZY");
 }


   LCD_GOTOXY(1,2);
   printf(LCD_PUTC,"S:%3.2f
T:%c%c%c%c%c",SP,PV[0],PV[1],PV[2],PV[3],PV[4]);

   if(input(PFUN)){
```



```
      delay_ms(30);
      if((ESTADO & 0b00000001) == 0x01){
        output_low(LFUN);
        ESTADO = ESTADO & 0b11111110;}
      else{
        ESTADO = ESTADO | 0b00000001;
        output_high(LFUN);}
    }
    if(input(PREM)){
      delay_ms(30);
      if((ESTADO & 0b00000010) == 0x02){
        output_low(LREM);
        ESTADO = ESTADO & 0b11111101;}
      else{
        ESTADO = ESTADO | 0b00000010;
        output_high(LREM);}
    }
    if(input(PRLAT)){
      delay_ms(30);
      if((ESTADO & 0b00000100) == 0x04){  //open loop
        output_low(LRLAT);
        ESTADO = ESTADO & 0b11111011;}
      else{
        ESTADO = ESTADO | 0b00000100;
        output_high(LRLAT);}
    }
    if(input(PRTOP)){
      delay_ms(30);
      if((ESTADO & 0b00001000) == 0x08){
        output_low(LRTOP);
        ESTADO = ESTADO & 0b11110111;}
      else{
        ESTADO = ESTADO | 0b00001000;
        output_high(LRTOP);}
    }
    if(input(PVENT)){
      delay_ms(30);
      if((ESTADO & 0b00010000) == 0x10){
        output_low(LVENT);
        ESTADO = ESTADO & 0b11101111;}
      else{
        ESTADO = ESTADO | 0b00010000;
        output_high(LVENT);}
    }

    delay_ms(400);
}
}
```

C. Código de HMI

```
classdef HMItermico < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        AnlisisderespuestaensistematrmicoLabel  matlab.ui.control.Label
        TabGroup                        matlab.ui.container.TabGroup
        ConfigTab                       matlab.ui.container.Tab
        TemperaturaGaugeLabel           matlab.ui.control.Label
        TemperaturaGauge                matlab.ui.control.Gauge
        SetpointCKnobLabel              matlab.ui.control.Label
        SetpointCKnob                   matlab.ui.control.Knob
        ErrorGaugeLabel                 matlab.ui.control.Label
        ErrorGauge                      matlab.ui.control.Gauge
        SalidadecontrolGaugeLabel       matlab.ui.control.Label
        SalidadecontrolGauge            matlab.ui.control.Gauge
        TipodecontrolKnobLabel          matlab.ui.control.Label
        TipodecontrolKnob               matlab.ui.control.DiscreteKnob
        ArranqueSwitchLabel             matlab.ui.control.Label
        ArranqueSwitch                  matlab.ui.control.RockerSwitch
        ContSeg                         matlab.ui.control.Label
        SegundosLabel                   matlab.ui.control.Label
        ONOFFLampLabel                  matlab.ui.control.Label
        ONOFFLamp                       matlab.ui.control.Lamp
        ERRORLampLabel                  matlab.ui.control.Label
        ERRORLamp                       matlab.ui.control.Lamp
        ComunicacinLampLabel            matlab.ui.control.Label
        ComunicacionLamp                matlab.ui.control.Lamp
        BAUDRATEDropDownLabel           matlab.ui.control.Label
        BAUDRATE                        matlab.ui.control.DropDown
        PuertosCOMLabel                 matlab.ui.control.Label
        Puertos                         matlab.ui.control.DropDown
        REFRESHButton                   matlab.ui.control.Button
        CREATEButton                    matlab.ui.control.Button
        PuertoSERIELabel                matlab.ui.control.Label
        REMOTOLampLabel                 matlab.ui.control.Label
        REMOTOLamp                      matlab.ui.control.Lamp
        RemotoSwitchLabel               matlab.ui.control.Label
        RemotoSwitch                    matlab.ui.control.RockerSwitch
        SetpointText                    matlab.ui.control.NumericEditField
        DELETEButton                    matlab.ui.control.Button
        PararButton                     matlab.ui.control.Button
        InicioButton                    matlab.ui.control.Button
        GraficaTab                      matlab.ui.container.Tab
        Grafica                         matlab.ui.control.UIAxes
        DatosLabel                      matlab.ui.control.Label
        TempCheckBox                    matlab.ui.control.CheckBox
        TempfiltCheckBox                matlab.ui.control.CheckBox
        VoCheckBox                      matlab.ui.control.CheckBox
        ErrorCheckBox                   matlab.ui.control.CheckBox
        ValoresLabel_2                  matlab.ui.control.Label
        TdLabel                         matlab.ui.control.Label
        TdLbl                           matlab.ui.control.Label
```

```
        TrLabel                     matlab.ui.control.Label
        TrLbl                       matlab.ui.control.Label
        TpLabel                     matlab.ui.control.Label
        TpLbl                       matlab.ui.control.Label
        MPLabel_4                   matlab.ui.control.Label
        MpLbl                       matlab.ui.control.Label
        Ts5Label                    matlab.ui.control.Label
        TsLbl                       matlab.ui.control.Label
        ELabel                      matlab.ui.control.Label
        ErrLbl                      matlab.ui.control.Label
        SPCheckBox                  matlab.ui.control.CheckBox
        YMaxSliderLabel             matlab.ui.control.Label
        YMaxSlider                  matlab.ui.control.Slider
        YMinSliderLabel             matlab.ui.control.Label
        YMinSlider                  matlab.ui.control.Slider
        Tabladedatos                matlab.ui.container.Tab
        UITable                     matlab.ui.control.Table
        ExportaraExcelButton        matlab.ui.control.Button
        ValoresTab                  matlab.ui.container.Tab
        DatosPIDLabel               matlab.ui.control.Label
        DatosFuzzyLabel             matlab.ui.control.Label
        Label                       matlab.ui.control.Label
    end
    properties (Access = private)
        BT
    end

    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app)
            app.Puertos.Items = seriallist;
            app.ONOFFLamp.Color = 'black';
            app.ERRORLamp.Color = 'black';
            app.ComunicacionLamp.Color = 'black';
            app.REMOTOLamp.Color = 'black';
            global cont Vo SP Yt Ytf Er tiempo datos
            Yt =zeros(1,11000);
            Ytf=zeros(1,11000);
            Vo =zeros(1,11000);
            Er =zeros(1,11000);
            SP =zeros(1,11000);
            tiempo =zeros(1,11000);
            datos = zeros(5,11000);
            cont = 1;
        end
        % Callback function
        function BAUDRATEValueChanged(app, event)

        end
        % Callback function: Puertos, REFRESHButton
        function PuertosValueChanged(app, event)
            app.Puertos.Items = seriallist;
        end
        % Button pushed function: CREATEButton
        function CREATEButtonPushed(app, event)
```

```
    %delete(instrfind('Port',(app.Puertos.Value)));
    app.BT = serial(app.Puertos.Value);
    app.BT.BaudRate = str2double(app.BAUDRATE.Value);
    app.ComunicacionLamp.Color = 'yellow';
end
% Value changing function: SetpointCKnob
function SetpointCKnobValueChanging(app, event)
    changingValue = event.Value;
    app.SetpointText.Value = round(changingValue);
end
% Value changed function: SetpointText
function SetpointTextValueChanged(app, event)
    app.SetpointCKnob.Value = app.SetpointText.Value;
end
% Value changed function: RemotoSwitch
function RemotoSwitchValueChanged(app, event)
    global SP
    if strcmpi(app.RemotoSwitch.Value,'Off')
        app.REMOTOLamp.Color = 'black';
        app.SetpointText.Value = round(app.SetpointText.Value,3);
    elseif strcmpi(app.RemotoSwitch.Value,'On')
        app.REMOTOLamp.Color = 'blue';
        SP = app.SetpointText.Value;
    end
end
% Size changed function: GraficaTab
function GraficaTabSizeChanged(app, event)
    position = app.GraficaTab.Position;

end
% Value changed function: ArranqueSwitch
function ArranqueSwitchValueChanged(app, event)
   tic;

end
% Button pushed function: DELETEButton
function DELETEButtonPushed(app, event)
    delete(instrfind('Port',(app.Puertos.Value)));
    app.ComunicacionLamp.Color = 'red';
end
% Button pushed function: ParарButton
function ParарButtonPushed(app, event)
   global captura
   captura = 0;
   app.CREATEButton.Enable = 'On';
   if strcmpi(app.BT.Status,'open')
       fclose(app.BT);
       app.ComunicacionLamp.Color = 'yellow';
   end
end
% Button pushed function: InicioButton
function InicioButtonPushed(app, event)
    global Yt Ytf Vo Er SP cont Error captura tiempo datos
    captura = 1;
    while (captura == 1)
```

```
if captura == 0
    break;
end
if strcmpi(app.BT.Status,'closed')
    fopen(app.BT);
    app.CREATEButton.Enable = 'Off';
end
if strcmpi(app.BT.Status,'open')
    app.ComunicacionLamp.Color = 'green';
    Var = fscanf(app.BT);

    if(find(Var=="FUZZY"))
        app.TipodecontrolKnob.Value = 'Fuzzy';
    elseif(find(Var == "PID"))
        app.TipodecontrolKnob.Value = 'PID';
    elseif find(Var == "Open")
        app.TipodecontrolKnob.Value = 'Open Loop';
    end
    if(find(Var=='='))
        ini=find(Var=='='); %Busca el retorno de carro
(Primer dato)

        ini=ini(1)+1;
        fin=find(Var==10); %Busca operador grados (ultimo
dato)

        fin= fin(fin>ini)-1;
        fin=fin(1);
        tempC=char(Var(ini:fin));
        temp=str2double(tempC);
        if(find(Var=='Y'))
            tiempo(cont) = toc;
            Temperatura = temp;
            Yt(cont) = Temperatura;
            app.ContSeg.Text = num2str(toc);
            cont = cont+1;
        elseif(find(Var=='F'))
            Tempfilt = temp;
            Ytf(cont) = Tempfilt;
            app.TemperaturaGauge.Value = Tempfilt;
        elseif(find(Var=='V'))
            Salida = temp;
            Vo(cont) = Salida;
            app.SalidadecontrolGauge.Value = Salida;
        elseif(find(Var=='r'))
            Error = temp;
            Er(cont) = Error;
            app.ErrorGauge.Value = Error;
        elseif(find(Var=='S'))
            SetPoint = temp;
            SP(cont) = SetPoint;
            app.SetpointText.Value = SetPoint;
        end

        if app.TempCheckBox.Value
            plot(app.Grafica,Yt,'yellow');
        end
```

```
                        if app.TempfiltCheckBox.Value
                            plot(app.Grafica,Ytf,'red');
                        end
                        app.Grafica.NextPlot = 'add';
                        if app.VoCheckBox.Value
                            plot(app.Grafica,Vo,'blue');
                        end
                        if app.ErrorCheckBox.Value
                            plot(app.Grafica,Er,'black');
                        end
                        if app.SPCheckBox.Value
                            plot(app.Grafica,SP,'green');
                        end
                        app.Grafica.NextPlot = 'replaceall';

                        app.Grafica.XLim = [0 max(cont)+10];
                        app.Grafica.YLim = [app.YMinSlider.Value
app.YMaxSlider.Value];
                        app.Grafica.XGrid = 'on';
                        app.Grafica.YGrid = 'on';
                        app.Grafica.XMinorGrid = 'on';
                        app.Grafica.YMinorGrid = 'on';
                        app.Grafica.MinorGridColor = [0.651 0.651 0.651];
                        app.Grafica.GridColor = [0.502 0.502 0.502];
                        xlabel(app.Grafica,'Tiempo(k)');
                        ylabel(app.Grafica,'Amplitud');

                        %datos = [tiempo;Ytf;Vo;SP;Er];
                        %app.UITable.Data = datos';


                        app.MpLbl.Text = num2str(((max(Ytf)-SP)/SP)*100);
                    end
                else
                    app.ComunicacionLamp.Color = 'red';
                end
                pause(0.1);
            end
        end
        % Button pushed function: ExportaraExcelButton
        function ExportaraExcelButtonPushed(app, event)
            global datos
            writematrix(datos','datos.xls');

        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 892 495];
            app.UIFigure.Name = 'UI Figure';
            % Create AnlisisderespuestaensistematrmicoLabel
```



```
            app.AnlisisderespuestaensistematrmicoLabel =
uilabel(app.UIFigure);
            app.AnlisisderespuestaensistematrmicoLabel.HorizontalAlignment
= 'center';
            app.AnlisisderespuestaensistematrmicoLabel.FontSize = 20;
            app.AnlisisderespuestaensistematrmicoLabel.Position = [259 461
375 24];
            app.AnlisisderespuestaensistematrmicoLabel.Text = {'Análisis de
respuesta en sistema térmico'; ''};
            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [1 1 892 447];
            % Create ConfigTab
            app.ConfigTab = uitab(app.TabGroup);
            app.ConfigTab.Title = 'Config';
            % Create TemperaturaGaugeLabel
            app.TemperaturaGaugeLabel = uilabel(app.ConfigTab);
            app.TemperaturaGaugeLabel.HorizontalAlignment = 'center';
            app.TemperaturaGaugeLabel.Position = [58 223 73 22];
            app.TemperaturaGaugeLabel.Text = 'Temperatura';
            % Create TemperaturaGauge
            app.TemperaturaGauge = uigauge(app.ConfigTab, 'circular');
            app.TemperaturaGauge.Limits = [0 300];
            app.TemperaturaGauge.MajorTicks = [0 50 100 150 200 250 300];
            app.TemperaturaGauge.ScaleColors = [1 0 0];
            app.TemperaturaGauge.ScaleColorLimits = [250 300];
            app.TemperaturaGauge.Position = [33 260 120 120];
            % Create SetpointCKnobLabel
            app.SetpointCKnobLabel = uilabel(app.ConfigTab);
            app.SetpointCKnobLabel.HorizontalAlignment = 'center';
            app.SetpointCKnobLabel.Position = [149 30 66 22];
            app.SetpointCKnobLabel.Text = 'Setpoint °C';
            % Create SetpointCKnob
            app.SetpointCKnob = uiknob(app.ConfigTab, 'continuous');
            app.SetpointCKnob.Limits = [0 300];
            app.SetpointCKnob.ValueChangingFcn = createCallbackFcn(app,
@SetpointCKnobValueChanging, true);
            app.SetpointCKnob.Position = [148 89 60 60];
            app.SetpointCKnob.Value = 20;
            % Create ErrorGaugeLabel
            app.ErrorGaugeLabel = uilabel(app.ConfigTab);
            app.ErrorGaugeLabel.HorizontalAlignment = 'center';
            app.ErrorGaugeLabel.Position = [240 223 32 22];
            app.ErrorGaugeLabel.Text = 'Error';
            % Create ErrorGauge
            app.ErrorGauge = uigauge(app.ConfigTab, 'circular');
            app.ErrorGauge.Limits = [-50 50];
            app.ErrorGauge.MajorTicks = [-50 -40 -30 -20 -10 0 10 20 30 40
50];
            app.ErrorGauge.ScaleColors = [0 1 0;1 0 0;1 0 0];
            app.ErrorGauge.ScaleColorLimits = [-5 5;30 50;-50 -30];
            app.ErrorGauge.Position = [195 260 120 120];
            % Create SalidadecontrolGaugeLabel
            app.SalidadecontrolGaugeLabel = uilabel(app.ConfigTab);
            app.SalidadecontrolGaugeLabel.HorizontalAlignment = 'center';
```

```
            app.SalidadecontrolGaugeLabel.Position = [371 223 95 22];
            app.SalidadecontrolGaugeLabel.Text = 'Salida de control';
            % Create SalidadecontrolGauge
            app.SalidadecontrolGauge = uigauge(app.ConfigTab, 'circular');
            app.SalidadecontrolGauge.ScaleColors = [1 0 0];
            app.SalidadecontrolGauge.ScaleColorLimits = [90 100];
            app.SalidadecontrolGauge.Position = [357 260 120 120];
            % Create TipodecontrolKnobLabel
            app.TipodecontrolKnobLabel = uilabel(app.ConfigTab);
            app.TipodecontrolKnobLabel.HorizontalAlignment = 'center';
            app.TipodecontrolKnobLabel.Position = [333 30 84 22];
            app.TipodecontrolKnobLabel.Text = 'Tipo de control';
            % Create TipodecontrolKnob
            app.TipodecontrolKnob = uiknob(app.ConfigTab, 'discrete');
            app.TipodecontrolKnob.Items = {'PID', 'Fuzzy', 'Open Loop'};
            app.TipodecontrolKnob.Position = [333 86 60 60];
            app.TipodecontrolKnob.Value = 'Fuzzy';
            % Create ArranqueSwitchLabel
            app.ArranqueSwitchLabel = uilabel(app.ConfigTab);
            app.ArranqueSwitchLabel.HorizontalAlignment = 'center';
            app.ArranqueSwitchLabel.Position = [539 33 55 22];
            app.ArranqueSwitchLabel.Text = 'Arranque';
            % Create ArranqueSwitch
            app.ArranqueSwitch = uiswitch(app.ConfigTab, 'rocker');
            app.ArranqueSwitch.ValueChangedFcn = createCallbackFcn(app,
@ArranqueSwitchValueChanged, true);
            app.ArranqueSwitch.Position = [557 102 20 45];
            % Create ContSeg
            app.ContSeg = uilabel(app.ConfigTab);
            app.ContSeg.BackgroundColor = [0.902 0.902 0.902];
            app.ContSeg.FontSize = 18;
            app.ContSeg.Position = [528 291 76 22];
            app.ContSeg.Text = '00:00:00';
            % Create SegundosLabel
            app.SegundosLabel = uilabel(app.ConfigTab);
            app.SegundosLabel.HorizontalAlignment = 'center';
            app.SegundosLabel.VerticalAlignment = 'bottom';
            app.SegundosLabel.Position = [504 226 103 151];
            app.SegundosLabel.Text = 'Segundos';
            % Create ONOFFLampLabel
            app.ONOFFLampLabel = uilabel(app.ConfigTab);
            app.ONOFFLampLabel.HorizontalAlignment = 'right';
            app.ONOFFLampLabel.Position = [720 268 57 22];
            app.ONOFFLampLabel.Text = 'ON / OFF';
            % Create ONOFFLamp
            app.ONOFFLamp = uilamp(app.ConfigTab);
            app.ONOFFLamp.Position = [792 268 20 20];
            % Create ERRORLampLabel
            app.ERRORLampLabel = uilabel(app.ConfigTab);
            app.ERRORLampLabel.HorizontalAlignment = 'right';
            app.ERRORLampLabel.Position = [728 236 49 22];
            app.ERRORLampLabel.Text = 'ERROR';
            % Create ERRORLamp
            app.ERRORLamp = uilamp(app.ConfigTab);
            app.ERRORLamp.Position = [792 236 20 20];
```

```
            app.ERRORLamp.Color = [0.851 0.3294 0.102];
            % Create ComunicacinLampLabel
            app.ComunicacinLampLabel = uilabel(app.ConfigTab);
            app.ComunicacinLampLabel.HorizontalAlignment = 'right';
            app.ComunicacinLampLabel.Position = [695 299 82 22];
            app.ComunicacinLampLabel.Text = 'Comunicación';
            % Create ComunicacionLamp
            app.ComunicacionLamp = uilamp(app.ConfigTab);
            app.ComunicacionLamp.Position = [792 299 20 20];
            % Create BAUDRATEDropDownLabel
            app.BAUDRATEDropDownLabel = uilabel(app.ConfigTab);
            app.BAUDRATEDropDownLabel.HorizontalAlignment = 'right';
            app.BAUDRATEDropDownLabel.Position = [676 120 70 22];
            app.BAUDRATEDropDownLabel.Text = 'BAUDRATE';
            % Create BAUDRATE
            app.BAUDRATE = uidropdown(app.ConfigTab);
            app.BAUDRATE.Items = {'2400', '4800', '9600', '14400', '19200',
'28800', '38400', '57600', '115200'};
            app.BAUDRATE.Position = [763 120 100 22];
            app.BAUDRATE.Value = '38400';
            % Create PuertosCOMLabel
            app.PuertosCOMLabel = uilabel(app.ConfigTab);
            app.PuertosCOMLabel.HorizontalAlignment = 'right';
            app.PuertosCOMLabel.Position = [667 151 78 22];
            app.PuertosCOMLabel.Text = 'Puertos COM';
            % Create Puertos
            app.Puertos = uidropdown(app.ConfigTab);
            app.Puertos.Items = {'COM3', 'COMx'};
            app.Puertos.ValueChangedFcn = createCallbackFcn(app,
@PuertosValueChanged, true);
            app.Puertos.Position = [763 151 100 22];
            app.Puertos.Value = 'COM3';
            % Create REFRESHButton
            app.REFRESHButton = uibutton(app.ConfigTab, 'push');
            app.REFRESHButton.ButtonPushedFcn = createCallbackFcn(app,
@PuertosValueChanged, true);
            app.REFRESHButton.Position = [667 81 70 22];
            app.REFRESHButton.Text = 'REFRESH';
            % Create CREATEButton
            app.CREATEButton = uibutton(app.ConfigTab, 'push');
            app.CREATEButton.ButtonPushedFcn = createCallbackFcn(app,
@CREATEButtonPushed, true);
            app.CREATEButton.Position = [745 80 58 22];
            app.CREATEButton.Text = 'CREATE';
            % Create PuertoSERIELabel
            app.PuertoSERIELabel = uilabel(app.ConfigTab);
            app.PuertoSERIELabel.Position = [741 183 80 22];
            app.PuertoSERIELabel.Text = 'Puerto SERIE';
            % Create REMOTOLampLabel
            app.REMOTOLampLabel = uilabel(app.ConfigTab);
            app.REMOTOLampLabel.HorizontalAlignment = 'right';
            app.REMOTOLampLabel.Position = [719 331 58 22];
            app.REMOTOLampLabel.Text = 'REMOTO';
            % Create REMOTOLamp
            app.REMOTOLamp = uilamp(app.ConfigTab);
```



```
            app.REMOTOLamp.Position = [792 331 20 20];
            % Create RemotoSwitchLabel
            app.RemotoSwitchLabel = uilabel(app.ConfigTab);
            app.RemotoSwitchLabel.HorizontalAlignment = 'center';
            app.RemotoSwitchLabel.Position = [486 33 48 22];
            app.RemotoSwitchLabel.Text = 'Remoto';
            % Create RemotoSwitch
            app.RemotoSwitch = uiswitch(app.ConfigTab, 'rocker');
            app.RemotoSwitch.ValueChangedFcn = createCallbackFcn(app,
@RemotoSwitchValueChanged, true);
            app.RemotoSwitch.Position = [499 101 20 45];
            % Create SetpointText
            app.SetpointText = uieditfield(app.ConfigTab, 'numeric');
            app.SetpointText.Limits = [0 300];
            app.SetpointText.ValueDisplayFormat = '%.3f';
            app.SetpointText.ValueChangedFcn = createCallbackFcn(app,
@SetpointTextValueChanged, true);
            app.SetpointText.HorizontalAlignment = 'center';
            app.SetpointText.BackgroundColor = [0.9608 0.9608 0.9608];
            app.SetpointText.Position = [144 12 67 22];
            % Create DELETEButton
            app.DELETEButton = uibutton(app.ConfigTab, 'push');
            app.DELETEButton.ButtonPushedFcn = createCallbackFcn(app,
@DELETEButtonPushed, true);
            app.DELETEButton.Position = [816 80 62 22];
            app.DELETEButton.Text = 'DELETE';
            % Create PararButton
            app.PararButton = uibutton(app.ConfigTab, 'push');
            app.PararButton.ButtonPushedFcn = createCallbackFcn(app,
@PararButtonPushed, true);
            app.PararButton.Position = [777 33 72 22];
            app.PararButton.Text = 'Parar';
            % Create InicioButton
            app.InicioButton = uibutton(app.ConfigTab, 'push');
            app.InicioButton.ButtonPushedFcn = createCallbackFcn(app,
@InicioButtonPushed, true);
            app.InicioButton.Position = [700 33 71 22];
            app.InicioButton.Text = 'Inicio';
            % Create GraficaTab
            app.GraficaTab = uitab(app.TabGroup);
            app.GraficaTab.SizeChangedFcn = createCallbackFcn(app,
@GraficaTabSizeChanged, true);
            app.GraficaTab.Title = 'Grafica';
            % Create Grafica
            app.Grafica = uiaxes(app.GraficaTab);
            title(app.Grafica, 'Grafica de respuesta')
            xlabel(app.Grafica, 'Tiempo')
            ylabel(app.Grafica, 'Temp')
            app.Grafica.PlotBoxAspectRatio = [1 0.543889845094664
0.543889845094664];
            app.Grafica.GridColor = [0.502 0.502 0.502];
            app.Grafica.MinorGridColor = [0.651 0.651 0.651];
            app.Grafica.Box = 'on';
            app.Grafica.NextPlot = 'add';
            app.Grafica.XGrid = 'on';
```

```
app.Grafica.XMinorGrid = 'on';
app.Grafica.YGrid = 'on';
app.Grafica.YMinorGrid = 'on';
app.Grafica.Position = [131 37 639 373];
% Create DatosLabel
app.DatosLabel = uilabel(app.GraficaTab);
app.DatosLabel.BackgroundColor = [0.651 0.651 0.651];
app.DatosLabel.HorizontalAlignment = 'center';
app.DatosLabel.VerticalAlignment = 'top';
app.DatosLabel.FontWeight = 'bold';
app.DatosLabel.Position = [22 244 80 136];
app.DatosLabel.Text = 'Datos';
% Create TempCheckBox
app.TempCheckBox = uicheckbox(app.GraficaTab);
app.TempCheckBox.Text = 'Temp';
app.TempCheckBox.FontColor = [1 1 0];
app.TempCheckBox.Position = [30 324 51 22];
% Create TempfiltCheckBox
app.TempfiltCheckBox = uicheckbox(app.GraficaTab);
app.TempfiltCheckBox.Text = 'Temp filt';
app.TempfiltCheckBox.FontColor = [1 0 0];
app.TempfiltCheckBox.Position = [30 303 67 22];
% Create VoCheckBox
app.VoCheckBox = uicheckbox(app.GraficaTab);
app.VoCheckBox.Text = 'Vo';
app.VoCheckBox.FontColor = [0 0.451 0.7412];
app.VoCheckBox.Position = [30 282 36 22];
% Create ErrorCheckBox
app.ErrorCheckBox = uicheckbox(app.GraficaTab);
app.ErrorCheckBox.Text = 'Error';
app.ErrorCheckBox.Position = [30 261 49 22];
% Create ValoresLabel_2
app.ValoresLabel_2 = uilabel(app.GraficaTab);
app.ValoresLabel_2.BackgroundColor = [0.902 0.902 0.902];
app.ValoresLabel_2.HorizontalAlignment = 'center';
app.ValoresLabel_2.VerticalAlignment = 'top';
app.ValoresLabel_2.FontWeight = 'bold';
app.ValoresLabel_2.Position = [791 93 78 317];
app.ValoresLabel_2.Text = 'Valores';
% Create TdLabel
app.TdLabel = uilabel(app.GraficaTab);
app.TdLabel.BackgroundColor = [0.149 0.149 0.149];
app.TdLabel.HorizontalAlignment = 'center';
app.TdLabel.VerticalAlignment = 'top';
app.TdLabel.FontColor = [0.9412 0.9412 0.9412];
app.TdLabel.Position = [796 342 68 38];
app.TdLabel.Text = 'Td';
% Create TdLbl
app.TdLbl = uilabel(app.GraficaTab);
app.TdLbl.FontColor = [0.9412 0.9412 0.9412];
app.TdLbl.Position = [812 342 42 22];
app.TdLbl.Text = '123.45';
% Create TrLabel
app.TrLabel = uilabel(app.GraficaTab);
app.TrLabel.BackgroundColor = [0.149 0.149 0.149];
```

```
app.TrLabel.HorizontalAlignment = 'center';
app.TrLabel.VerticalAlignment = 'top';
app.TrLabel.FontColor = [0.9412 0.9412 0.9412];
app.TrLabel.Position = [796 294 68 38];
app.TrLabel.Text = 'Tr';
% Create TrLbl
app.TrLbl = uilabel(app.GraficaTab);
app.TrLbl.FontColor = [0.9412 0.9412 0.9412];
app.TrLbl.Position = [812 294 42 22];
app.TrLbl.Text = '123.45';
% Create TpLabel
app.TpLabel = uilabel(app.GraficaTab);
app.TpLabel.BackgroundColor = [0.149 0.149 0.149];
app.TpLabel.HorizontalAlignment = 'center';
app.TpLabel.VerticalAlignment = 'top';
app.TpLabel.FontColor = [0.9412 0.9412 0.9412];
app.TpLabel.Position = [796 248 68 38];
app.TpLabel.Text = 'Tp';
% Create TpLbl
app.TpLbl = uilabel(app.GraficaTab);
app.TpLbl.FontColor = [0.9412 0.9412 0.9412];
app.TpLbl.Position = [812 248 42 22];
app.TpLbl.Text = '123.45';
% Create MPLabel_4
app.MPLabel_4 = uilabel(app.GraficaTab);
app.MPLabel_4.BackgroundColor = [0.149 0.149 0.149];
app.MPLabel_4.HorizontalAlignment = 'center';
app.MPLabel_4.VerticalAlignment = 'top';
app.MPLabel_4.FontColor = [0.9412 0.9412 0.9412];
app.MPLabel_4.Position = [796 200 68 38];
app.MPLabel_4.Text = 'MP';
% Create MpLbl
app.MpLbl = uilabel(app.GraficaTab);
app.MpLbl.FontColor = [0.9412 0.9412 0.9412];
app.MpLbl.Position = [812 200 42 22];
app.MpLbl.Text = '123.45';
% Create Ts5Label
app.Ts5Label = uilabel(app.GraficaTab);
app.Ts5Label.BackgroundColor = [0.149 0.149 0.149];
app.Ts5Label.HorizontalAlignment = 'center';
app.Ts5Label.VerticalAlignment = 'top';
app.Ts5Label.FontColor = [0.9412 0.9412 0.9412];
app.Ts5Label.Position = [796 153 68 38];
app.Ts5Label.Text = 'Ts (5%)';
% Create TsLbl
app.TsLbl = uilabel(app.GraficaTab);
app.TsLbl.FontColor = [0.9412 0.9412 0.9412];
app.TsLbl.Position = [812 153 42 22];
app.TsLbl.Text = '123.45';
% Create ELabel
app.ELabel = uilabel(app.GraficaTab);
app.ELabel.BackgroundColor = [0.149 0.149 0.149];
app.ELabel.HorizontalAlignment = 'center';
app.ELabel.VerticalAlignment = 'top';
app.ELabel.FontColor = [0.9412 0.9412 0.9412];
```

```
            app.ELabel.Position = [796 103 68 38];
            app.ELabel.Text = 'E%';
            % Create ErrLbl
            app.ErrLbl = uilabel(app.GraficaTab);
            app.ErrLbl.FontColor = [0.9412 0.9412 0.9412];
            app.ErrLbl.Position = [812 103 42 22];
            app.ErrLbl.Text = '123.45';
            % Create SPCheckBox
            app.SPCheckBox = uicheckbox(app.GraficaTab);
            app.SPCheckBox.Text = 'SP';
            app.SPCheckBox.FontColor = [0 1 0];
            app.SPCheckBox.Position = [30 244 38 22];
            % Create YMaxSliderLabel
            app.YMaxSliderLabel = uilabel(app.GraficaTab);
            app.YMaxSliderLabel.HorizontalAlignment = 'right';
            app.YMaxSliderLabel.Position = [14 40 36 22];
            app.YMaxSliderLabel.Text = 'YMax';
            % Create YMaxSlider
            app.YMaxSlider = uislider(app.GraficaTab);
            app.YMaxSlider.Limits = [-300 300];
            app.YMaxSlider.Orientation = 'vertical';
            app.YMaxSlider.Position = [17 76 3 122];
            app.YMaxSlider.Value = 120;
            % Create YMinSliderLabel
            app.YMinSliderLabel = uilabel(app.GraficaTab);
            app.YMinSliderLabel.HorizontalAlignment = 'right';
            app.YMinSliderLabel.Position = [72 40 33 22];
            app.YMinSliderLabel.Text = 'YMin';
            % Create YMinSlider
            app.YMinSlider = uislider(app.GraficaTab);
            app.YMinSlider.Limits = [-300 300];
            app.YMinSlider.Orientation = 'vertical';
            app.YMinSlider.Position = [72 76 3 122];
            % Create Tabladedatos
            app.Tabladedatos = uitab(app.TabGroup);
            app.Tabladedatos.Title = 'Tabla de datos';
            % Create UITable
            app.UITable = uitable(app.Tabladedatos);
            app.UITable.ColumnName = {'Tiempo'; 'Tempf'; 'Vo'; 'SP';
'Error'};
            app.UITable.RowName = {};
            app.UITable.Position = [246 33 399 381];
            % Create ExportaraExcelButton
            app.ExportaraExcelButton = uibutton(app.Tabladedatos, 'push');
            app.ExportaraExcelButton.ButtonPushedFcn =
createCallbackFcn(app, @ExportaraExcelButtonPushed, true);
            app.ExportaraExcelButton.Position = [687 392 104 22];
            app.ExportaraExcelButton.Text = 'Exportar a Excel';
            % Create ValoresTab
            app.ValoresTab = uitab(app.TabGroup);
            app.ValoresTab.Title = 'Valores';
            % Create DatosPIDLabel
            app.DatosPIDLabel = uilabel(app.ValoresTab);
            app.DatosPIDLabel.Position = [45 371 60 22];
            app.DatosPIDLabel.Text = 'Datos PID';
```



```
            % Create DatosFuzzyLabel
            app.DatosFuzzyLabel = uilabel(app.ValoresTab);
            app.DatosFuzzyLabel.Position = [45 169 72 22];
            app.DatosFuzzyLabel.Text = 'Datos Fuzzy';
            % Create Label
            app.Label = uilabel(app.ValoresTab);
            app.Label.Position = [83 341 35 22];
        end
    end
    methods (Access = public)
        % Construct app
        function app = HMItermico
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```