UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

CONTROL DIGITAL PID PARA SISTEMAS TÉRMICOS BASADO EN MICROCONTROLADOR PIC

TRABAJO DE GRADUACIÓN

PRESENTADO POR

José Mario Díaz Benítez

Eri Samuel Murcia Peraza

PARA OPTAR AL GRADO DE

Ingeniero en Electrónica

Asesor:

Ing. Héctor Rubén Carías Juárez

Enero de 2006

Soyapango – El Salvador – Centro América

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

AUTORIDADES:

RECTOR
ING. FEDERICO HUGUET RIVERA

VICERRECTOR ACADÉMICO
PBRO. VÍCTOR BERMÚDEZ, sdb

SECRETARIO GENERAL
LIC. MARIO RAFAEL OLMOS

DECANO DE LA FACULTAD DE INGENIERÍA
ING. GODOFREDO GIRÓN

DIRECTOR DE ESCUELA DE ELECTRÓNICA
ING. OSCAR DURÁN VIZCARRA

ASESOR DEL TRABAJO DE GRADUACIÓN
ING. HÉCTOR RUBÉN CARÍAS JUÁREZ

JURADO EVALUADOR
ING. CARLOS GUILLERMO BRAN
ING. EDGARDO CRUZ ZELEDÓN
ING. JUAN RENÉ NÚÑEZ

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN

Ing. Carlos Guillermo Bran
JURADO

Ing. Edgardo Cruz Zeledón
JURADO

Ing. Juan René Núñez
JURADO

Ing. Héctor Rubén Carías Juárez
ASESOR

Índice.

1 Introducción. 1
2 Objetivos 2
2.1 Objetivo General. 2
2.2 Objetivos Específicos. 2
3 Alcances y Limitaciones. 3
3.1 Alcances 3
3.2 Limitaciones 4
4 Marco Teórico. 5
4.1 Filtros Digitales. [1]. 5
4.1.1 Estructuras Directas. 5
4.1.2 Módulos De Segundo Orden. 9
4.1.3 Implementación En Microcomputadora De Filtros Digitales. 11
4.2 Controladores y acciones de Control [2] 14
4.2.1 Clasificación de controladores Industriales. 14
4.2.2 Controlador Automático, actuador y sensor (elemento de medición). 14
4.2.3 Acciones de Control. 15
4.3 Controladores PID digitales [1]. 19
4.4 Ajuste empírico de controladores PID [3]. 23
4.4.1 Método de la curva de reacción de Ziegler-Nichols 24
4.5 Transductores de temperatura [4]. 25
4.5.1 Termopares o termocuplas. 27
4.6 Microcontroladores PIC [5] 28
4.7 El Servidor Web Siteplayer. 31
4.7.1 Descripción de SitePlayer 31
5 Diseño del Controlador 34
5.1 Acondicionador de señal para termocupla. 34
5.2 Interfaz con el usuario. 37
5.3 Ingeniería Humana Empleada en el diseño de la Interfaz. [7] 40
5.4 Etapa de potencia. 42
5.5 Conexión de Siteplayer con controlador PID. 43
5.6 Justificación de los Componentes Empleados. 44
5.7 Programa del controlador 46
5.8 Etapa de Autosintonía. 50
6 Conclusiones y Recomendaciones 53
6.1 Conclusiones 53
6.2 Recomendaciones 54
7 Referencias Bibliográficas 56
8 ANEXO 57

1 Introducción.

El presente documento contiene la información concerniente al trabajo de graduación “Control Digital PID Para Sistemas Térmicos Basado En Microcontrolador PIC”.

En el marco teórico se dan a conocer los principios matemáticos que sirven como base para la implementación del programa interno del PIC, donde para dicho programa se hizo uso de un filtro digital, en el cual sus coeficientes de operación están relacionados con las variables de un control PID (Kp, tv y tn). Luego se procede a dar a conocer la estructura estándar de un PID, se estudian las propiedades de cada una de las acciones de control P, PI, PD, PID y se describe el método empírico de calibración de Ziegler-Nichols, utilizado para calcular los parámetros del controlador dependiendo de la respuesta del sistema térmico específico a controlar. Posteriormente se dedica una parte a una explicación breve acerca de los transductores de temperatura en general y específicamente de la termocupla, que es transductor empleado en este trabajo de graduación más una breve descripción de los microcontroladores PIC.

Se presenta también para finalizar, el coprocesador Servidor web Siteplayer, empleado para la comunicación remota vía Ethernet de los datos del controlador.

En el diseño del controlador, se explican las partes que componen el controlador como lo son: el acondicionador de señal para la termocupla, la forma de adquisición de las señales al PIC, la implementación del filtro digital, la interfaz de usuario, la etapa de salida de señal analógica hacia el actuador, la etapa de potencia hacia el calefactor y la comunicación de los datos provenientes del controlador hacia la computadora.

2 Objetivos

2.1 Objetivo General.

Diseñar y construir un prototipo de control PID digital de bajo costo para sistemas térmicos basado en microcontrolador PIC que posea una interfaz fácil de usar para introducir parámetros, calibrar y fijar la referencia de temperatura y ser capaz de establecer comunicación con una computadora para transferir datos del comportamiento de la temperatura, dentro del equipo, en el tiempo.

2.2 Objetivos Específicos.

- ✓ Diseñar y construir un controlador PID para sistemas térmicos basado en microcontrolador PIC.
- ✓ Diseñar y construir la etapa de adquisición de datos necesaria para la medición de la variable controlada: temperatura.
- ✓ Diseñar y construir la etapa de potencia necesaria para el manejo del elemento de control (resistencia calefactora).
- ✓ Desarrollo de un programa capaz de entrar en comunicación con el PIC para adquirir los datos que estarán alojados en una memoria y poder graficarlos.

3 Alcances y Limitaciones.

3.1 Alcances

Se ha diseñado un sistema que cuando opera a lazo cerrado es capaz de controlar la temperatura de un equipo, visualizarla con una precisión de una[1] cifra decimal por medio de una interfaz digital diseñada con criterios de ingeniería humana; y que hace sencillo el proceso de ingreso de datos tanto al usuario avanzado, que cambia parámetros como constantes de operación del control (ganancia proporcional, tiempo integral y tiempo derivativo), como al operario que nada más pone la referencia de temperatura.

El sistema opera a lazo abierto (sin realimentación), y es capaz de obtener un valor de temperatura en el visualizador y se tienen cinco salidas que se activen a cinco valores de temperatura programados.

El controlador posee de un sistema de protección en caso de que el sensor de realimentación falle, incluyendo alarmas visuales en caso de sobre temperatura y temperatura baja.

Se construyó un acondicionador de señal para termocupla a la entrada analógica con el propósito de tener la señal de la variable controlada, que en este caso es la temperatura.

Se construyó la etapa de potencia para modificar el estado del elemento de control, que en este caso es la resistencia calefactora, agregando a esta los sistemas de protección y aislamiento necesarios.

Se agregaron los componentes necesarios para la comunicación vía Ethernet, así como la programación de los mismos para registrar de forma remota los datos provenientes del controlador.

[1] En el anteproyecto se ofrecieron dos cifras decimales, pero con la condición de explicar por qué no podría representarse con dos o más, en las limitaciones se da una explicación del por qué

3.2 Limitaciones

- ✓ El máximo de temperatura a controlar es de 300 ºC.
- ✓ Las resistencias calefactoras son alimentadas con 110 ó 220 VAC monofásico pudiendo elegir el usuario con qué voltaje trabajar.
- ✓ Se restringe el sensor a una termocupla.
- ✓ El microcontrolador a usar es el 16F877A por sus características de bajo costo, comunicación serie, convertidor A/D de 10 bits, suficiente memoria de programa para la aplicación y posibilidad de uso de algunas entradas y salidas digitales.
- ✓ La interfaz digital tiene como restricción una cifra decimal por la siguiente razón: Debido a que para obtener 300ºC con una resolución de 1 centésimo (300.00) sería necesario que el convertidor análogo a digital tuviera como mínimo 30000 pasos, por lo que debería tener 15 bits (esto por la fórmula $n = \frac{\log(30000)}{\log(2)} = 14.87$). El convertidor A/D del PIC tiene 8 canales de 10 bits (1024 pasos) usando los 8 canales se logra llegar hasta una resolución de 8192 pasos que no son suficientes si se quiere hacer que la medición sea real, por lo tanto se decidió hacer la medición de 300 ºC con un décimo de precisión. Con esto se asegura una medida real de la temperatura, usando tres canales analógicos del microcontrolador utilizando una resolución máxima de 3072 pasos.
- ✓ Otra de las limitaciones técnicas es que el sistema podrá ser utilizado en sistemas donde las variaciones de temperatura no sean tan críticas debido a que como puede verse en las pruebas hechas al dispositivo, siempre hay una variación de más o menos 5 grados desde la referencia, esto no es lo indicado para equipos que requieran de alta precisión como los incubadores.

4 Marco Teórico.

4.1 Filtros Digitales. [1]

En el procesamiento digital de señales interesa encontrar una función de transferencia $D(z)$ en el dominio de z la cual realice las operaciones de filtrado y control digital. La función de transferencia puede representarse en general por:

$$D(z) = \frac{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}}{1 + b_1 z^{-1} + \cdots + b_n z^{-n}} \tag{4.1}$$

a_i y b_i son coeficientes reales, y n es el máximo de los órdenes de los polinomios del denominador y numerador. Ambos polinomios pueden tener coeficientes cero en los términos de mayor orden, de modo que (4.1) describe el caso general.

La idea es describir la realización en diagrama de bloques de (4.1) usando elementos de retraso de tiempo (representados por z^{-1}), sumadores y multiplicadores. Cada diferente diagrama de bloques se llama estructura del filtro.

Existen incontables estructuras para (4.1) y se describirán brevemente las más importantes. En particular se describirán las estructuras de forma directa y módulos de segundo orden.

4.1.1 Estructuras Directas.

Las estructuras directas para filtros digitales son aquellas en las cuales los coeficientes reales, a_i y b_i de (4.1), aparecen como multiplicadores en la implementación de diagramas de bloques.

4.1.1.1. Primera Estructura Directa.

Si se representa a:

$$D(z)=\frac{\sum_{i=0}^{n} a_i z^{-i}}{\sum_{i=0}^{n} b_i z^{-i}} \tag{4.2}$$

donde $b_0 = 1$. Si $X(z)$ es la entrada al filtro, y $Y(z)$ la salida, entonces

$$\frac{Y(z)}{X(z)}=\frac{\sum_{i=0}^{n} a_i z^{-i}}{\sum_{i=0}^{n} b_i z^{-i}}$$

de modo que

$$\frac{Y(z)}{M(z)}=\sum_{i=0}^{n} a_i z^{-i}$$

$$\frac{X(z)}{M(z)}=\sum_{i=0}^{n} b_i z^{-i}$$

Ahora

$$X(z)=\sum_{i=0}^{n} b_i z^{-i} M(z)$$

$$M(z)=X(z)-\sum_{i=1}^{n} b_i z^{-i} M(z)$$

$$Y(z)=\sum_{i=0}^{n} a_i z^{-i} M(z)$$

En el dominio del tiempo

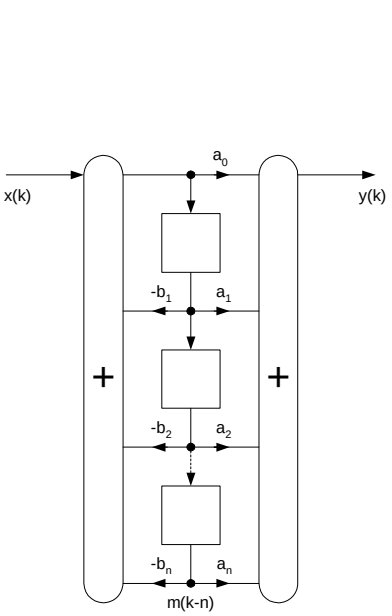
$$\left.\begin{aligned} m(k) &= x(k)-\sum_{i=1}^{n} b_i m(k-i) \\ y(k) &= \sum_{i=0}^{n} a_i m(k-i) \end{aligned}\right\} \tag{4.3}$$

Las ecuaciones (4.3) definen la primera estructura directa (1D) como se muestra en la figura 4.1a. En la figura 4.1 los retrasos de tiempo (z^{-1}) se representan por cajas rectangulares; los multiplicadores por flechas etiquetadas; los sumadores por círculos y elipses conteniendo un más (+); y los puntos de distribución de señal, por puntos en las uniones de las líneas y barras oscuras. Las

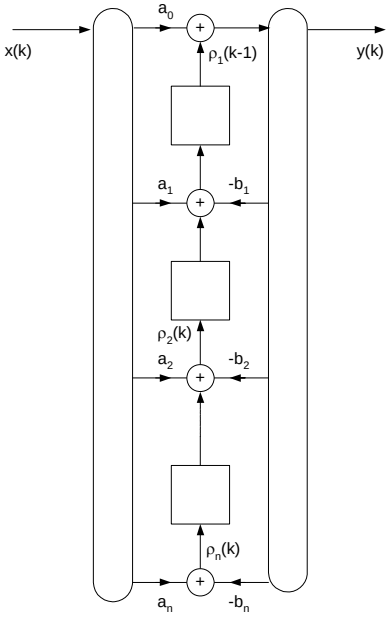
estructuras 1D se llaman **canónicas** debido a que poseen solo n elementos de retraso de tiempo, el número mínimo para una función de enésimo orden de (4.1).

4.1.1.2. Redes Transpuestas.

La estructura transpuesta de un filtro digital se forma al invertir el flujo de la señal en todas las ramas del diagrama de bloques. En consecuencia, los puntos de suma se convierten en puntos de distribución de señal, y viceversa. Las entradas llegan a ser salidas y viceversa. La transpuesta de una estructura de un filtro tiene la misma función de transferencia que la estructura original. Entonces las estructuras para filtros digitales existen como pares transpuestos. En consecuencia, se pueden usar estas propiedades de las estructuras para encontrar una segunda estructura directa (2D) para (4.1).



(a)



(b)

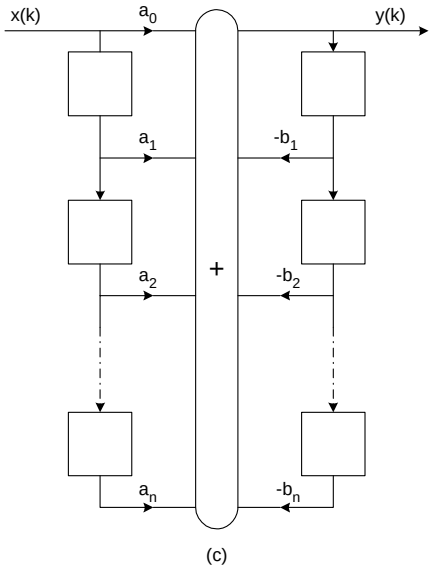


Figura 4.1 Estructuras directas: (a) 1D; (b) 2D; (c) 3D.

4.1.1.3. Segunda Estructura Directa.

Si se efectúa la transpuesta de la estructura 1D, se obtiene la estructura 2D mostrada en la figura 4.1b. También implementa (4.1), pero necesita $n+1$ ecuaciones diferencia (puntos de suma), mientras que la estructura 1D necesita solo 2.

Las ecuaciones diferencia 2D tienen la forma

$$\left.\begin{aligned} p_i(k) &= p_{i+1}(k-1) + a_i x(k) - b_i y(k) \quad i = 1, n = 1 \\ p_n(k) &= a_n x(k) - b_n y(k) \\ y(k) &= a_0 x(k) + p_1(k-1) \end{aligned}\right\} \tag{4.4}$$

Esta estructura también es canónica.

4.1.1.4. Tercera Estructura Directa.

Volviendo a 4.1 se puede escribir:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{n} a_i z^{-i}}{\sum_{i=0}^{n} b_i z^{-i}}$$

Y de este modo

$$Y(z)\sum_{i=0}^{n} b_i z^{-i} = X(z)\sum_{i=0}^{n} a_i z^{-i}$$

Por consecuencia

$$Y(z) = \sum_{i=0}^{n} a_i z^{-i} X(z) - \sum_{i=0}^{n} b_i z^{-i} Y(z)$$

En el dominio del tiempo

$$y(k) = \sum_{i=0}^{n} a_i x(k-i) - \sum_{i=0}^{n} b_i y(k-i) \qquad (4.5)$$

Esta es la ecuación en diferencia para la tercera estructura directa (3D), la cual puede verse en su diagrama a bloques en la figura 4.1(c). Debe observarse que esta estructura tiene una sola unión de suma, pero tiene $2n$ elementos de retardo de tiempo

4.1.2 Módulos De Segundo Orden.

Para evitar los problemas de sensibilidad de los coeficientes se implementa comúnmente la función de transferencia $D(z)$ de (4.1) como módulos de segundo orden en cascada o paralelo de la forma

$$D(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

La estructura de estos módulos de segundo orden pueden ser ellas mismas de la forma directa de la figura 4.1. La figura 4.2 ilustra las estructuras 1D, 2D y 3D para los módulos de segundo orden.

Las ecuaciones diferencia que describen cada estructura son:

1D:
$$\left.\begin{aligned} m(k) &= x(k) - b_1 m(k-1) - b_2 m(k-2) \\ y(k) &= a_0 m(k) + a_1 m(k-1) + a_2 m(k-2) \end{aligned}\right\} \quad (4.5)$$

2D:
$$\left.\begin{aligned} y(k) &= a_0 x(k) + p_1(k-1) \\ p_1(k) &= a_1 x(k) - b_1 y(k) + p_2(k-1) \\ p_2 &= a_2 x(k) - b_2 y(k) \end{aligned}\right\} \quad (4.6)$$

3D:
$$y(k) = a_0 x(k) + a_1(k-1) + a_2 x(k-2) - b_1 y(k-1) - b_2 y(k-2) \quad (4.7)$$

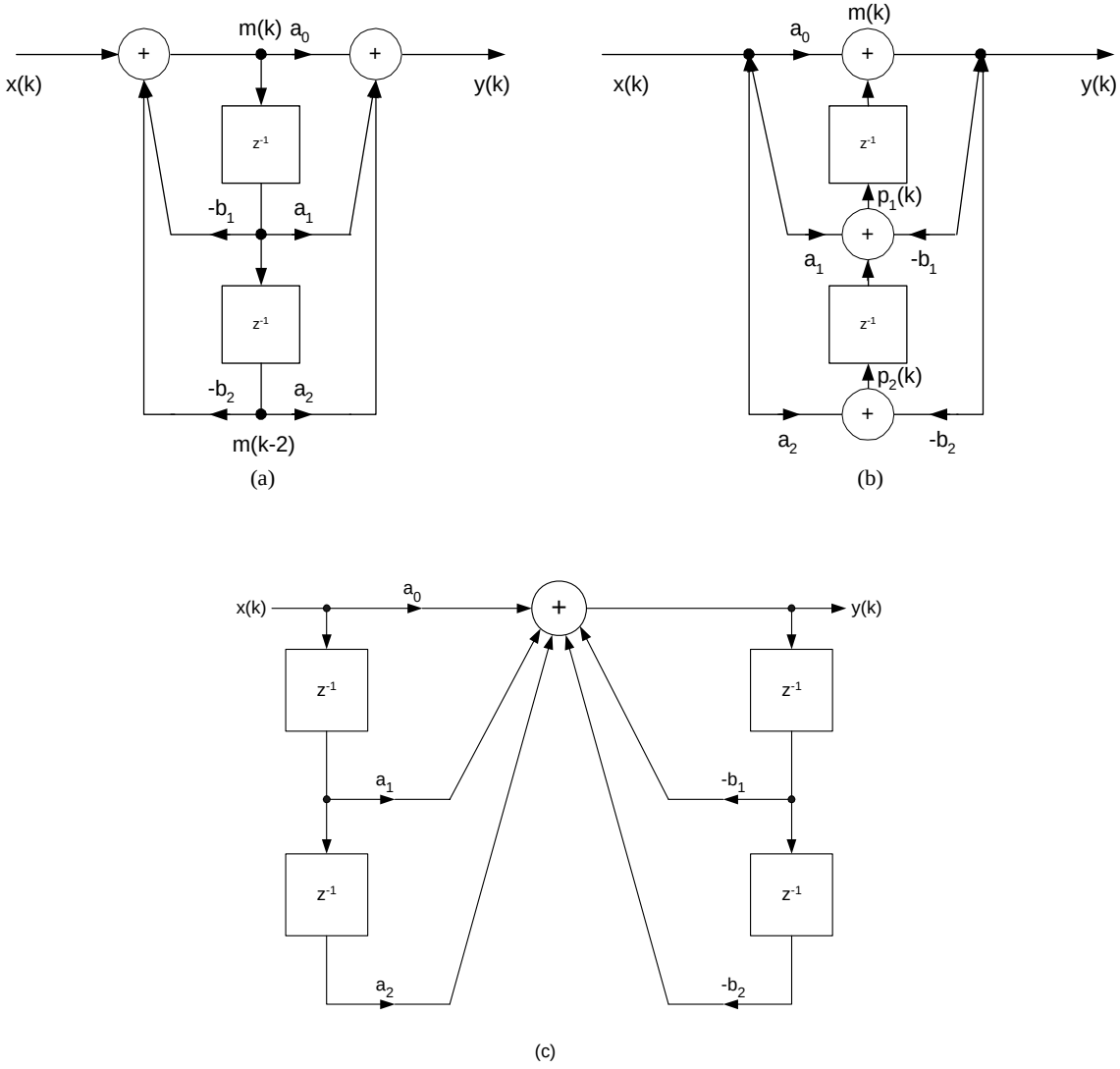


Figura 4.2 Módulos de segundo orden directos: (a) 1D; (b) 2D; (c) 3D.

Las ecuaciones se calculan en el orden apropiado. Por ejemplo, en (4.5) $m(k)$ debe obtenerse primero. En (4.6) y (4.7) $y(k)$ debe calcularse primero para minimizar los retardos de tiempo de cálculo entre la muestra de entrada $x(kT)$ y la generación de la salida $y(kT+\tau_C)$, donde τ_c representa el retardo de cálculo del filtro. Idealmente $\tau_c = 0$ pero ya que esto es irreal, minimizamos τ_c al ordenar los cálculos. En la práctica si $T >> \tau_c$ se puede despreciar τ_c.

4.1.3 Implementación En Microcomputadora De Filtros Digitales.

En el apartado anterior se examinaron varias estructuras para la realización de filtros digitales. Cada estructura puede describirse por un único conjunto de ecuaciones diferencia. Las ecuaciones requieren las operaciones de multiplicación, suma, y retraso de tiempo. En esta sección se explorará la implementación de estas ecuaciones diferencia por microcomputadoras[2]. Las microcomputadoras se programan en lenguaje ensamblador para calcular las ecuaciones diferencia de la estructura en paralelo.

El procedimiento describe la forma de implementar los módulos de segundo orden.

Considérese la figura 4.5. Todos los programas para módulos de segundo orden pueden modelarse por este flujograma, el cual representa el proceso necesario durante un intervalo de tiempo $(kT < t < kT + T)$. Por ejemplo, se tiene la estructura 1D con la ecuación.

$$\left.\begin{aligned} m(k) &= x(k) - b_1 m(k-1) - b_2 m(k-2) \\ y(k) &= a_0 m(k) + a_1 m(k-1) + a_2 m(k-2) \end{aligned}\right\} \qquad (4.5)$$

[2] El microcontrolador PIC es considerado como una microcomputadora.

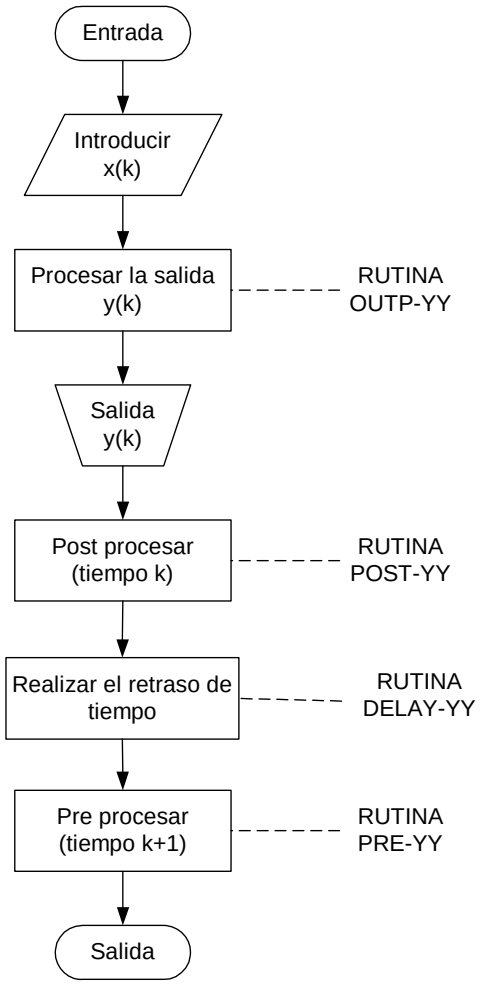


Figura 4.5 Módulos de segundo orden en general. YY = 1D, 2D, 3D, 4D, 1X, 2X.

$$\left.\begin{aligned} T_1 &= -b_1 m(k-1) - b_2 m(k-2) \\ T_2 &= a_1 m(k-1) + a_2 m(k-2) \end{aligned}\right\} \tag{4.8}$$

Si se precalcula (durante el intervalo de tiempo $kT - T \le t < kT$)

Entonces (durante el intervalo de tiempo $kT \le t < kT + T$) se puede rápidamente calcular la salida $y(k)$ al recibir la entrada $x(k)$ como sigue:

$$\left.\begin{aligned} m(k) &= x(k) + T_1 \\ y(k) &= a_0 m(k) + T_2 \end{aligned}\right\} \tag{4.9}$$

Esto completa el procedimiento de un módulo 1D, de modo que un post procesamiento no es necesario.

Las ecuaciones 4.5, 4.6 y 4.7 se reorganizan a continuación en las siguientes rutinas:

Estructura 1D:

OUTP_1D:

$$m(k) = x(k) + T_1$$
$$y(k) = a_0 m(k) + T_2$$

POST_1D: ninguno

PRE_1D:

$$T_1 = -b_1 m(k-1) - b_2 m(k-2)$$
$$T_2 = a_1 m(k-1) + a_2 m(k-2)$$

Estructura 2D

OUTP_2D:

$$y(k) = a_0 x(k) + p_1(k-1)$$

POST_2D:

$$p_1(k) = a_1 x(k) - b_1 y(k) + p_2(k-1)$$
$$p_2(k) = a_2 x(k) - b_2 y(k)$$

PRE_2D: ninguno

Estructura 3D

OUTP_3D

$$y(k) = a_0 x(k) + p_1(k-1)$$

POST_3D:

$$p_1(k) = a_1 x(k) - b_1 y(k) + p_2(k-1)$$
$$p_2(k) = a_2 x(k) - b_2 y(k)$$

PRE_3D: ninguno

4.2 Controladores y acciones de Control [2]

4.2.1 Clasificación de controladores Industriales.

Los controladores Industriales, se pueden clasificar de acuerdo con sus acciones de control, de la siguiente forma:

1. Controladores de dos posiciones o intermitentes (encendido-apagado)
2. Controladores proporcionales
3. Controladores integrales
4. Controladores proporcional-integral
5. Controladores tipo proporcional-derivativo
6. Controladores tipo proporcional-integral-derivativo

La mayoría de los controladores industriales utilizan electricidad o algún fluido, como aceite o aire a presión, a modo de fuentes de potencia. De acuerdo a esto, los controladores pueden también clasificarse según el tipo de potencia que utilizan en su operación, como neumáticos, hidráulicos o electrónicos. La clase de controlador a usar se decide en base a la naturaleza de la planta y las condiciones de operación, incluyendo consideraciones tales como seguridad, costo, disponibilidad, confiabilidad, exactitud, peso y tamaño.

4.2.2 Controlador Automático, actuador y sensor (elemento de medición).

La figura 4.6 muestra un diagrama de bloques de un sistema de control industrial que consiste en un controlador automático, un actuador o accionador, una planta y un sensor (elemento de medición). El controlador detecta la señal de error, que suele estar a un nivel de potencia muy bajo, y la amplifica a un nivel suficientemente alto. (Así, el controlador automático está constituido por un detector de error y un amplificador. También suele haber un circuito de retroalimentación adecuado, junto con un amplificador, que se utilizan para alterar la señal de error, amplificándola, y a veces diferenciándola y/o integrándola, para producir una mejor señal de control). El actuador es un dispositivo de potencia que

produce la entrada a la planta, de acuerdo con la señal de control, de modo que la señal de retroalimentación corresponda a la señal de entrada de referencia.

La salida de un controlador automático alimenta a un actuador o accionador, que bien pueden ser un motor o una válvula neumática, un motor hidráulico o uno eléctrico.

El transductor o elemento de medición es un dispositivo que convierte la variable de salida en otra variable adecuada, como un desplazamiento, presión, o voltaje, que se utilizan para comparar la salida con la señal de entrada de referencia. Este elemento es el camino de retroalimentación en el sistema de lazo cerrado.

El punto de ajuste del controlador debe convertirse en una entrada de referencia con las mismas unidades que la señal de retroalimentación del sensor o el elemento de medición.

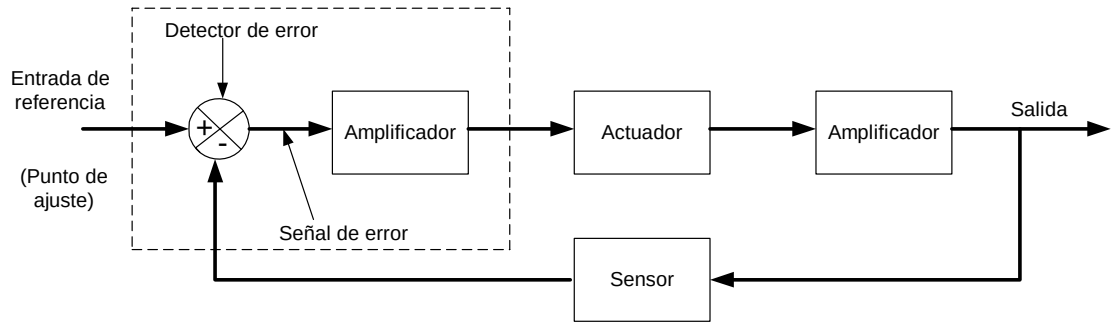


Figura 4.6 Diagrama de bloques de un sistema de control industrial

4.2.3 Acciones de Control.

Las funciones de transferencia de las acciones de control principales, que son las utilizadas para el desarrollo del presente proyecto se encuentran resumidas en las siguientes ecuaciones:

Proporcional $$\frac{U(s)}{E(s)} = K_P \qquad (4.10)$$

Proporcional e Integral $$\frac{U(s)}{E(s)} = K_p\left(1+\frac{1}{T_i s}\right) \quad (4.11)$$

Proporcional y Derivativo $$\frac{U(s)}{E(s)} = K_P + sK_P T_D \quad (4.12)$$

Proporcional, Integral y Derivativo $$\frac{U(s)}{E(s)} = K_p\left(1+\frac{1}{T_I s}+T_D s\right) \quad (4.13)$$

A continuación se describen las acciones de control de cada una de las funciones PID:

4.2.3.1. Acción de control proporcional

Para un controlador de acción de control proporcional, la relación entre la salida del controlador $u(t)$ y la señal de error $e(t)$ es

$$u(t) = K_P e(t)$$

O expresado en transformada de Laplace,

$$\frac{U(s)}{E(s)} = K_P \quad (4.14)$$

Donde K_P se denomina ganancia proporcional.

No importando el mecanismo y la potencia que lo alimenta, el controlador proporcional es esencialmente un amplificador con ganancia ajustable. Con el inconveniente de que su acción de control produce una desviación del valor en estado estacionario[3], ya que nunca llegará por más que se aumente K_P al valor de la referencia.

4.2.3.2. Acción de control integral:

Esta acción de control genera una salida del controlador que es proporcional al error acumulado, lo que implica que su modo de controlar es lento.

[3] El error de estado estacionario, es el que puede observarse cuando la planta ha estabilizado y su valor no es igual al de la referencia.

$$u(t) = K_I \int_0^t e(t)dt \qquad \frac{U(s)}{E(s)} = \frac{K_I}{s} \tag{4.15}$$

La señal de control $u(t)$ tiene un valor diferente de cero cuando la señal de error $e(t)$ es cero. Por lo que se concluye que dada una referencia constante, o perturbaciones, el error en régimen permanente es cero.

4.2.3.3. Acción de control proporcional-integral.

Esta acción se define mediante

$$u(t) = K_P e(t) + \frac{K_P}{T_I} \int_0^t e(t)dt \tag{4.16}$$

Donde T_I se denomina tiempo integral y es quien ajusta la acción integral. La función de transferencia resulta:

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s} \right) \tag{4.17}$$

Con un control proporcional, es necesario que exista error para tener una acción de control distinta de cero. Con acción integral, un error pequeño positivo siempre resulta en una acción de control creciente, y si el error es negativo la señal de control será decreciente. Este razonamiento sencillo muestra que el error en régimen permanente será siempre cero.

4.2.3.4. Acción de control proporcional-derivativa

La acción de control se define mediante:

$$u(t) = K_p e(t) + K_P T_d \frac{de(t)}{dt} \tag{4.18}$$

Donde T_D es una constante denominada tiempo derivativo. Esta acción tiene carácter de previsión, lo que hace más rápida la acción de control, aunque tiene la desventaja importante que amplifica las señales de ruido y puede provocar saturación en el actuador. La acción de control derivativa nunca se utiliza por sí sola, debido a que sólo es eficaz durante períodos transitorios. La función de transferencia de un controlador PD resulta:

$$\frac{U(s)}{E(s)} = K_P + sK_P T_D \tag{4.19}$$

Cuando una acción de control derivativa se agrega a un controlador proporcional, permite obtener un controlador de alta sensibilidad, es decir que responde a la velocidad del cambio del error y produce una corrección significativa antes de que la magnitud del error se vuelva demasiado grande. Aunque el control derivativo no afecta en forma directa al error en estado estacionario, añade amortiguamiento al sistema y, por tanto, permite un valor más grande que la ganancia K, lo cual provoca una mejora en la precisión en estado estacionario.

4.2.3.5. Acción de control proporcional-integral-derivativa

Esta acción combinada reúne las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con esta acción combinada se obtiene mediante:

$$u(t) = K_P e(t) + \frac{K_P}{T_I}\int e(t)dt + K_P T_D \frac{de(t)}{dt} \tag{4.20}$$

y su función de transferencia resulta:

$$\frac{U(s)}{E(s)} = K_P\left(1 + \frac{1}{Tis} + T_D s\right) \tag{4.21}$$

4.3 Controladores PID digitales [1].

La función de transferencia de un controlador PID en el dominio de s es:

$$C(s) = \frac{U(s)}{E(s)} = K_P\left(1 + \frac{1}{Tis} + T_D s\right) \tag{4.22}$$

La cual puede ser expresada de la forma:

$$C(s) = K_P + \frac{K_P}{T_I s} + K_P T_D s \tag{4.23}$$

O también:

$$C(s) = K_P + \frac{K_I}{s} + K_D s \tag{4.24}$$

Donde K_I y K_D son las ganancias integral y derivativa respectivamente y son ajustables.

En control digital, las funciones de transferencia se expresan en función de la variable discreta z, y del tiempo de muestreo T, para la ecuación anterior, la función discretizada se presenta de la siguiente forma:

$$D(z) = K_P + K_I \frac{T}{2}\left[\frac{z+1}{z-1}\right] + K_D\left[\frac{z-1}{Tz}\right] \tag{4.25}$$

Para la parte integral se usó la regla trapezoidal de integración numérica y para la parte diferencial se usó la regla de diferenciación numérica directa.

Esta función de transferencia puede ser implementada como es mostrado en la figura 4.7. Aquí los términos proporcional, integral y derivativo son implementados separadamente y sumados a la salida. Un método alternativo (que ha sido implementado en el programa de este proyecto) sería encontrar una función de transferencia de segundo orden para (4.25) y usar las estructuras directas presentadas anteriormente. Considérese

A partir de:

$$D(z) = K_P + K_I \frac{T}{2}\left[\frac{z+1}{z-1}\right] + K_D\left[\frac{z-1}{Tz}\right]$$

Desarrollando la expresión:

$$D(z) = \frac{K_P(z-1)(z) + (K_I T/2)(z+1)(z) + (K_D/T)(z-1)(z-1)}{(z-1)(z)}$$

$$D(z) = \frac{K_P(z^2 - z) + (K_I T/2)(z^2 + z) + (K_D/T)(z^2 - 2z + 1)}{(z^2 - z)}$$

Agrupando términos:

$$D(z) = \frac{(K_P + K_I T/2 + K_D/T)z^2 + (-K_P + K_I T/2 - 2K_D/T)z + K_D/T}{(z^2 - z)}$$

Multiplicando denominador y numerador por z^{-2} se obtiene la otra representación de los sistemas discretos, quedando de la siguiente forma:

$$D(z) = \frac{(K_P + K_I T/2 + K_D/T) + (-K_P + K_I T/2 - 2K_D/T)z^{-1} + K_D/Tz^{-2}}{1 - z^{-1} + 0z^{-2}}$$

Comparando esta expresión con la de la función general de segundo orden

$$D(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

Se puede observar que:

$$a_0 = K_P + \frac{K_I T}{2} + \frac{K_D}{T}$$

$$a_1 = -K_P + \frac{K_I T}{2} - \frac{2K_D}{T} \qquad (4.26)$$

$$a_2 = \frac{K_D}{T}$$

$$b_1 = -1$$

$$b_2 = 0$$

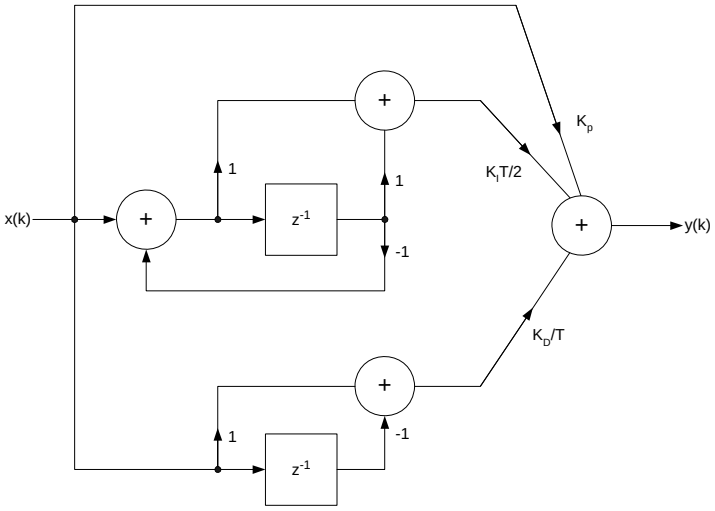


Figura 4.7 Controlador PID digital.

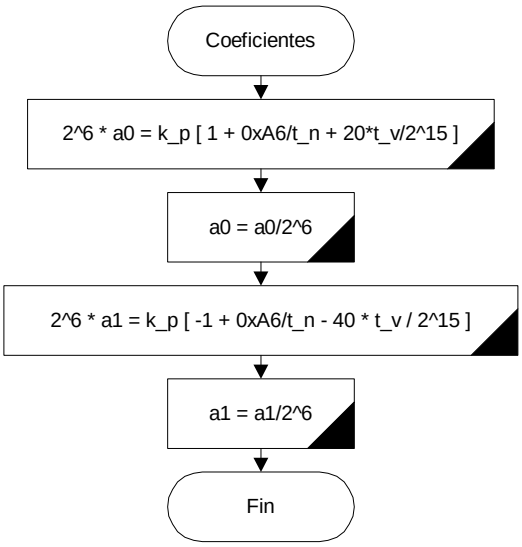


Fig. 4.8 Flujograma de cálculo de a0 y a1

Tomando en cuenta que $K_I = \frac{K_P}{T_N}$ y que $K_D = K_P(T_V)$, y evaluando las ecuaciones para un tiempo de muestreo igual a un segundo $T = 1s$ y además incluyendo las escalas para el ingreso de las variables de control se obtiene que: $a0 = K_P \frac{1000}{2^{15}}\left[1 + \frac{2^{15}}{2T_N \times 100} + T_V \frac{20}{2^{15}}\right]$ y $a1 = K_P \frac{1000}{2^{15}}\left[-1 + \frac{2^{15}}{2T_N \times 100} - 2T_V \frac{20}{2^{15}}\right]$ para que no se pierdan los datos mas pequeños en los cálculos del

microcontrolador, se normalizan las ecuaciones multiplicando por 1, en otras palabras ($\frac{2^6}{2^6}$) como sigue:

$$2^6 a0 = K_P \frac{1000}{2^9}\left[1+\frac{2^{15}}{2T_N \times 100}+T_V \frac{20}{2^{15}}\right] \text{ y } 2^6 a1 = K_P \frac{1000}{2^9}\left[-1+\frac{2^{15}}{2T_N \times 100}-2T_V \frac{20}{2^{15}}\right]$$

Para evitar cálculos innecesarios en el microcontrolador se calculan todas las constantes aparte y se aproximan,

$$2^6 a0 = K_P\left[1+\frac{0xA6}{T_N}+\frac{T_v}{0x666}\right] \text{ y } 2^6 a1 = K_P\left[-1+\frac{0xA6}{T_N}-\frac{T_V}{0x333}\right]$$ después de obtener el termino de la derecha relativamente grande, se divide entre 2^6 y queda el valor escalado y normalizado El flujograma del cálculo de a0 y a1 puede verse en la figura 4.8.

La salida ayudándonos del cálculo de la segunda estructura directa tenemos la salida: $y_{(K)} = a0\left(error_{(K)}\right)+t3_{(k-1)}$ donde $t3_{(K)} = a1\left(error_{(K-1)}\right)+K_D\left(error_{(K-2)}\right)+y_{(K-1)}$.

Los códigos de estos dos cálculos se encuentran en las rutinas de coeficientes y salida. La figura 4.9 representa el cálculo de estas variables.

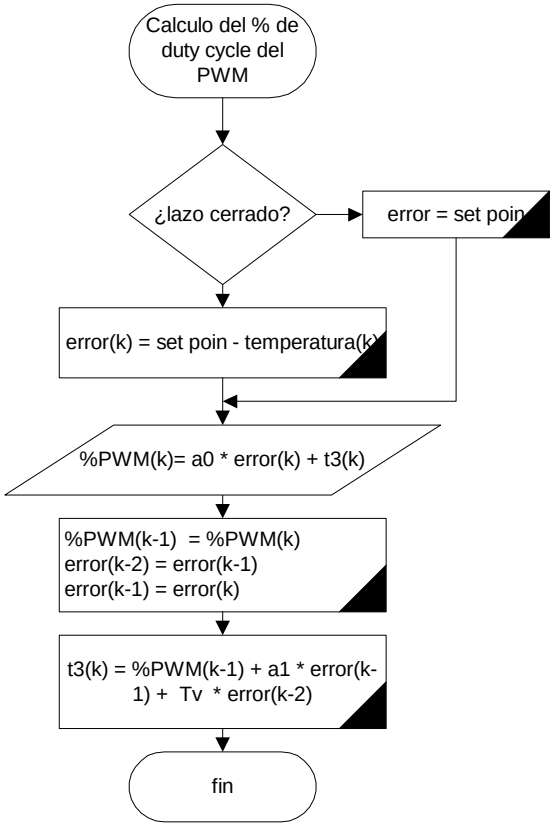


Fig. 4.9 Flujograma de cálculo de % de duty cycle en PWM[4].

4.4 Ajuste empírico de controladores PID [3].

A continuación se presenta el método de la “Curva de Reacción de Ziegler-Nichols”, ya que es el método aplicado en el proyecto, debido a su facilidad y confiabilidad para determinar los parámetros PID con los datos obtenidos de la planta; aunque existen numerosas alternativas para llevar a cabo esta tarea.

Algunos de los métodos empíricos tradicionales comenzaron a usarse por los años 1950. Algunos de estos métodos son:

- El método de oscilación de Ziegler-Nichols.
- El método de la curva de reacción de Ziegler-Nichols.

[4] PWM: Modulación de Ancho de Pulso es aplicado a la etapa de potencia ver Diseño del controlador en el apartado de Etapa de potencia.

- El método de la curva de reacción de Cohen-Coon.

Es importante saber cuál es la estructura (estándar, serie o paralelo) del PID al que se aplica el ajuste propuesto por Ziegler y Nichols. Existe cierta controversia respecto a cuál fue la estructura originalmente usada por Ziegler y Nichols; las reglas dadas aquí se proponen para la estructura estándar[5].

4.4.1 Método de la curva de reacción de Ziegler-Nichols

Muchas plantas en la práctica pueden describirse satisfactoriamente con un modelo de la forma:

$$H(s) = \mu \frac{e^{-sL}}{1+sT} \tag{4.27}$$

Una versión linealizada cuantitativa de este modelo puede obtenerse mediante un experimento a lazo abierto con el siguiente procedimiento:

1. Se lleva manualmente la planta a lazo abierto a un punto de operación normal manipulando $u(t)$ (ver figura 4.7). Supóngase que la planta se estabiliza en $y(t) = y_0$ para $U(t) = U_0$.
2. En un instante inicial t_0 se aplica un cambio al escalón en la entrada, de U_0 a U_∞ (el salto debe estar entre un 10 a 20% del valor nominal.
3. Se registra la respuesta de la salida hasta que se estabilice en el nuevo punto de operación. La figura 4.7 muestra una curva típica.
4. Se calculan los parámetros del modelo de las fórmulas:

$$\mu = \frac{y_{\max} - y_0}{U_{ref} - U_0} \tag{4.28}$$

$$L = t_1 - t_2 \tag{4.29}$$

$$T = t_2 - t_1 \tag{4.30}$$

[5] Vease Ec.4.21.

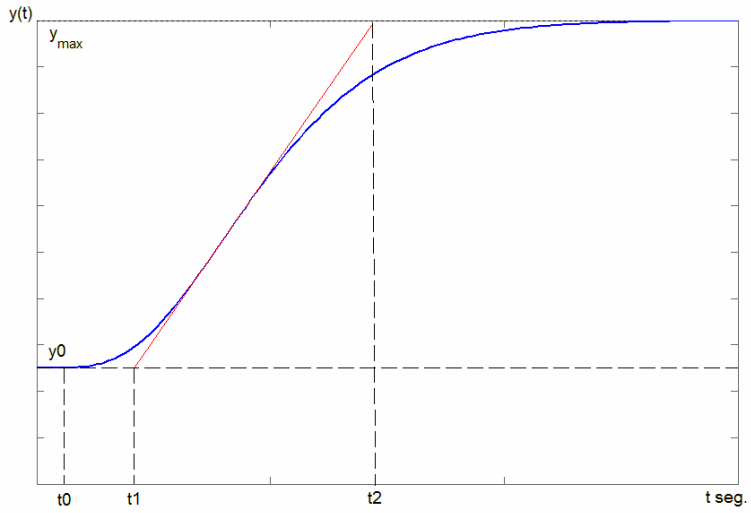


Figura 4.7: Respuesta al escalón (curva de reacción) en lazo abierto de la planta.

Los parámetros del controlador PID propuestos por Ziegler y Nichols a partir de la curva de reacción se determinan de Tabla 4.1.

	K_P	t_i	t_d
P	$\frac{T}{\mu L}$		
PI	$\frac{0.9T}{\mu L}$	$3L$	
PID	$\frac{1.2T}{\mu L}$	$2L$	$0.5L$

Tabla 4.1. Parámetros de controladores PID según el método de la curva de reacción de Ziegler-Nichols

4.5 Transductores de temperatura [4].

Cada vez que a un sistema físico se le suministra energía, su estado cambia.

La temperatura representa un índice de dicho estado.

La unidad de medida que se adopta en el Sistema Internacional es el grado Kelvin (K); la temperatura del cero absoluto[6] corresponde a 0 K.

[6] La temperatura del cero absoluto se asume que es cuando ya ha cesado todo movimiento molecular en un cuerpo. no ha sido posible llegar a ella por experimentación.

En la práctica se utilizan otras dos unidades de medida: el grado centígrado o grado Celsius (ºC), y el grado Fahrenheit (ºF).

En la tabla 4.2 puede observarse la correspondencia entre estas unidades: con respecto a las otras escalas se ve que la de grados Fahrenheit se espacia de modo diferente.

ºK	ºC	ºF
0	-273.1	-460
273.1	0	+32
373.1	100	+212
1273	1000	1832

Tabla 4.2 equivalencias entre las diferentes unidades de temperatura.

La ley de conversión es la siguiente:

$$^{o}F = \frac{9}{5}{}^{o}C + 32 \qquad (4.31)$$

En el presente proyecto se adopta la escala de los grados centígrados, cuyos puntos de referencia son muy prácticos en el uso corriente; ya que a 0ºC corresponde la temperatura del hielo que funde y a 100ºC la de ebullición del agua a nivel del mar.

En la industria y en el sector civil, la temperatura se determina con la ayuda de varios tipos de transductores más o menos complejos y precisos.

Entre todos los transductores se distinguen los de semiconductor, las termorresistencias y los termopares (o termocuplas) que, a pesar de sus sencillez tienen una alta precisión.

Ya que las dimensiones de los dispositivos mencionados anteriormente son muy pequeñas, estos pueden fácilmente intercalarse en el proceso.

4.5.1 Termopares o termocuplas.

El termopar consta de dos conductores metálicos de naturaleza distinta cuyo punto de conexión constituye un contacto galvánico (soldadura); esto es visible en la figura 4.8

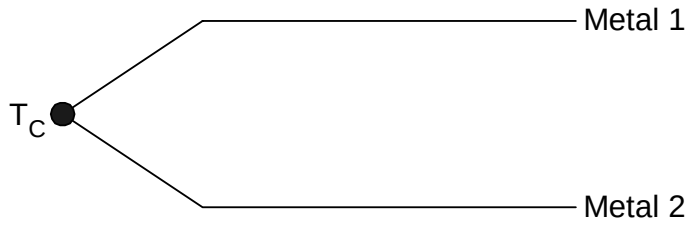


Figura 4.8 Termopar

Cuando el termopar (o “unión caliente”) alcanza la temperatura que hay que medir (por ejemplo en un horno), se conectan los conductores con el punto de medición (“unión fría”) cuya temperatura es diferente (figura 4.9).

En dicho circuito se genera una fuerza electromotriz (f.e.m.) termoeléctrica cuyo valor es dado por la diferencia Tc –Tf (efecto “Seebeck”).

Conociendo esta f.e.m. y la temperatura Tf, será posible calcular el valor de Tc.

Ya que hay que conocer la temperatura Tf para determinar Tc, conviene agregar a los hilos del termopar alambres de compensación hasta el punto en que la temperatura es conocida y queda constante.

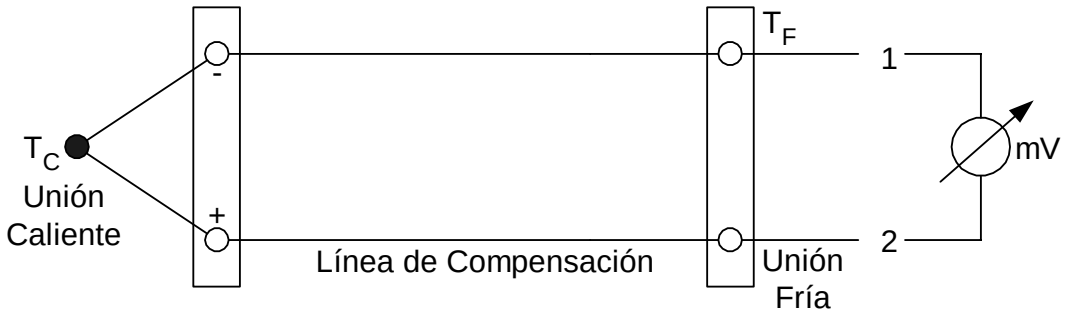


Figura 4.9 Líneas de compensación para el termopar.

Los termopares más usados son los siguientes:

- ✓ Los de Fe-Constantán (Tipo J)
- ✓ Los de Ni-NiCr (Tipo K)
- ✓ Los de Cu-Constantán (Tipo T)

Con respecto a los demás, el termopar Fe-Constantán ofrece una notable f.e.m. y una linealidad bastante buena, con un costo reducido; sin embargo su temperatura máxima de funcionamiento resulta limitada a causa del terminal de hierro (700 - 800 ºC).

El termopar de Ni-NiCr tiene una buena f.e.m. inferior a la del precedente tipo, pero puede usarse con temperaturas más elevadas (superiores a 1500 ºC).

Su característica es bastante lineal (sobre todo a las altas temperaturas), pero su costo es levemente superior al del termopar de Fe-Constantán. El principal inconveniente es su vulnerabilidad ante los gases reductores, por lo que es necesario protegerlo debidamente.

El termopar de Cu-Constantán se utiliza en un campo de temperaturas inferior al del termopar Fe-Constantán (de 500 hasta 600 ºC); tiene el mismo costo que el de este último, pero su linealidad es superior. Se adopta sobre todo en el campo de las bajas temperaturas.

Los alambres de compensación necesitan la misma característica termoeléctrica que la del termopar hasta 150 – 200 ºC con un costo inferior.

4.6 Microcontroladores PIC [5]

Definición de PIC: Peripheral Interface Control, comúnmente llamado Microcontrolador. En Japón se le llama “One Chip Microcomputer”

La figura 4.10 muestra el uso de los microcontroladores en los diversos campos de la vida moderna

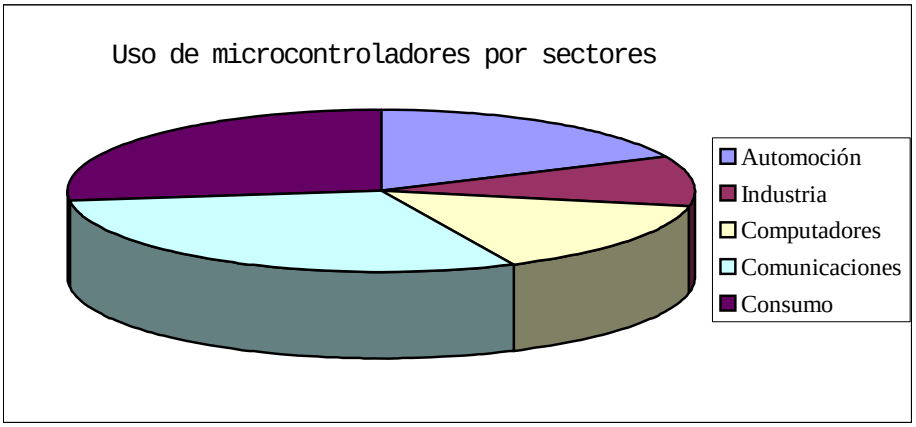


Figura 4.10 Uso de microcontroladores por sectores.

Sus características son:

- One Chip Microcomputer (La microcomputadora en un solo chip)
  - El PIC integra CPU, RAM, ROM y dispositivos de I/O.
- Puede leer variables analógicas (ciertos modelos)
  - El PIC 16F874A maneja 8 canales de entrada analógica.
- RISC (Reduced Instruction Set Computer)
  - Solo 35 instrucciones pueden ser decodificadas.
- Encapsulado pequeño (DIP 8 - 40 pines)
  - Se puede conectar en equipos pequeños.
- Tecnología de bajo consumo de potencia (0.8W) y alta velocidad
  - Se puede usar con batería y a una frecuencia máxima de 20MHz.
- Alta corriente I/O(20 - 25mA)
  - Se pueden manejar LEDs directamente.
- Bajo costo
  - Un PIC vale desde 3 dólares en EE.UU. (Pero comprado en El Salvador cuesta mas de $11.00).

Microprocesador y Microcontrolador

El microcontrolador es un sistema cerrado. No hay flexibilidad para ampliar su capacidad, pero tiene todas las funciones que necesita para controlar periféricos.

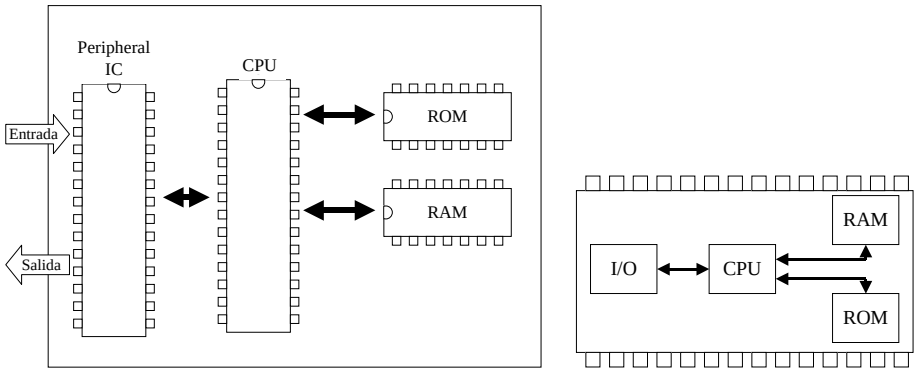


Figura 4.11 (a) sistema a microprocesador y (b) microcontrolador

La familia de los PIC de Microchip

Modelo	Pines	E/S	Memoria Programa(bit)	Memoria Datos(Byte)	Conv A/D	Precio
PIC12C509	8	6	1k X12(ONE)	41		$1.15-
PIC16C64A	40	33	2k X14(ONE)	128		$4.62-
PIC16C711	18	13	1k X14(ONE)	68	4ch 8bit	$2.30-
PIC16C715	18	13	2k X14(ONE)	128	4ch 8bit	$3.08-
PIC16C73B	28	22	4k X14(ONE)	192	5ch 8bit	$4.62-
PIC16C74B	40	33	4k X14(ONE)	192	8ch 8bit	$5.38-
PIC16F84	18	13	1k X14(EEP)	68		$3.08-
PIC16F84A	18	13	1k X14(EEP)	68		$3.59-
PIC16F873	28	22	4k X12(EEP)	192		$6.15-
PIC16F877	40	33	8k X12(EEP)	368	8ch 10bit	$7.69-

Tabla 4.3 Comparación de características entre diversos microcontroladores.

Su Forma de uso:

- ✓ Se escribe el programa en un archivo de texto con extensión .asm
- ✓ Se compila por medio de la aplicación MPLAB
- ✓ Se carga por medio de un circuito especial, el programa ya compilado en un archivo con extensión .hex
- ✓ Se prueba el programa en breadboard o en un entrenador especial

4.7 El Servidor Web Siteplayer.

4.7.1 Descripción de SitePlayer

El Siteplayer es el servidor web Ethernet más pequeño del mundo, también es el primer producto de una familia de servidores web integrados diseñados para permitir que cualquier dispositivo con microprocesador tenga acceso a Internet de manera sencilla y poco costosa. En tan sólo unos 4 cm^2, SitePlayer integra un servidor web, un controlador Ethernet 10baseT, memoria de flash para páginas web, procesador de objetos gráficos, y una interfase para dispositivos serie.

SitePlayer se trata de una interfase de comunicación serie/ethernet que también puede utilizarse como una opción para acceso web, actualización de productos o dispositivos manejados por microprocesador, o para reacondicionar productos antiguos. Entre las aplicaciones ejemplos se incluyen los aparatos de audio, termostatos, automatización del hogar, control industrial, control de procesos, equipos de prueba, aparatos médicos, automóviles, control de maquinaria, supervisión remota, y teléfonos móviles.

SitePlayer se trata de un coprocesador de servidor web, que funciona con protocolos de internet y paquetes Ethernet, es independiente del procesador propio del aparato. El tráfico Web no afecta al procesador del dispositivo, y además añade medidas de seguridad. Las comunicaciones entre SitePlayer y el dispositivo se realizan a través de objetos enviados a través de un puerto serie estándar de dos cables. No es necesario escribir ningún código TCP/IP o de red. SitePlayer puede también utilizarse en algunas aplicaciones en un modo “stand alone” (independiente) en el que se pueden realizar entradas/salidas (I/O) digitales, para ello se dispone de 8 pines I/O configurables que también pueden utilizarse como cuatro PWMs o DACs de 8 bits, generadores de frecuencia, o contadores de eventos.

SitePlayer contiene un potente sistema de objetos llamado SiteObjects™ que permite el cambio de imágenes gráficas, texto, música, enlaces, botones de radio o casillas de control basándose en datos en directo procedentes del procesador

del aparato sin necesidad de comandos CGI[7] o programación Java. Una página web puede contener un botón gráfico girado a una determinada posición, un interruptor que puede moverse hacia arriba o hacia abajo, o un enlace que puede cambiar basándose en una variable en el procesador del aparato. Para hacer las páginas para SitePlayer se utilizan herramientas de creación de páginas web estándar. Las páginas web se descargan en la memoria flash de SitePlayer a través de la interfaz de red. Las actualizaciones del firmware de SitePlayer también pueden descargarse manteniéndolo siempre al día. Además en el sitio www.siteplayer.com se encuentra una librería de botones gráficos, interruptores, LEDs y otras herramientas de interfase para el desarrollo de páginas web.

Proceso de SitePlayer para la creación de un proyecto

Un proyecto es el código que le dice a SitePlayer como funcionar y qué páginas web deben ser definidas por el fichero de definición de SitePlayer e integradas en la imagen binaria SitePlayer a través de SiteLinker (programa que hace la compilación y descarga de los archivos), estos ficheros se descargan en SitePlayer a través de una conexión Ethernet. Se puede modificar la página e interactuar con el aparato utilizando el módulo SitePlayer o SitePlayerPC[8]. Los capítulos del manual de programación se corresponden con los pasos necesarios para la creación de un proyecto SitePlayer:

1. Definir y crear sus objetos a través de un editor de texto con un fichero de definición SitePlayer Definition File (.SPD)

2. Crear sus páginas web utilizando un editor HTML

3. Recopilar y descargar el fichero binario de SitePlayer Binary file (.SPB) a través del programa SiteLinker

4. Explorar SitePlayer o SitePlayerPC a través de un navegador web

[7] CGI: Interfases de puerta de enlace común, son comandos que permiten hacer en páginas Web uso de formularios, entrada de texto, salida de texto, como los utilizados en las encuestas por Internet.
[8] SiteplayerPC es un emulador gratuito proporcionado por netmedia para su descarga en el sitio www.siteplayer.com

Para mayores detalles acerca de los ficheros de definición .SPD, .SPB y .SPI, necesarios para la programación y funcionamiento del Siteplayer consúltese el manual de programación del mismo, donde también se tratan los aspectos del flujo de datos del Siteplayer desde el dispositivo a monitorear y el web. Para consultar el manual haga clic aquí: Manual de programación Siteplayer [6]

5 Diseño del Controlador.

5.1 Acondicionador de señal para termocupla.

El microcontrolador PIC 16F877A posee entradas analógicas que pueden ser referenciadas de 0 a 5V con una resolución de 10 bits cada uno, para lograr llegar a tener una buena precisión de 1 decimal se hizo uso de tres canales analógicos para tener un mayor rango de datos al PIC para lograr llegar a los 300 ºC.

Para acondicionar la termocupla, se hizo uso del circuito integrado AD594 ya que la termocupla empleada es de tipo J, éste a su salida genera una señal de 10mV/ºC; por lo tanto para 300 ºC se tendrían como máximo 3V.

Estos 3V de salida pasan a un amplificador no inversor con ganancia 3.33, para que se tenga un voltaje de 10V a la salida, esto nos daría una razón de 33mv/ºC

El circuito acondicionador para la termocupla se presenta en la figura 5.1.

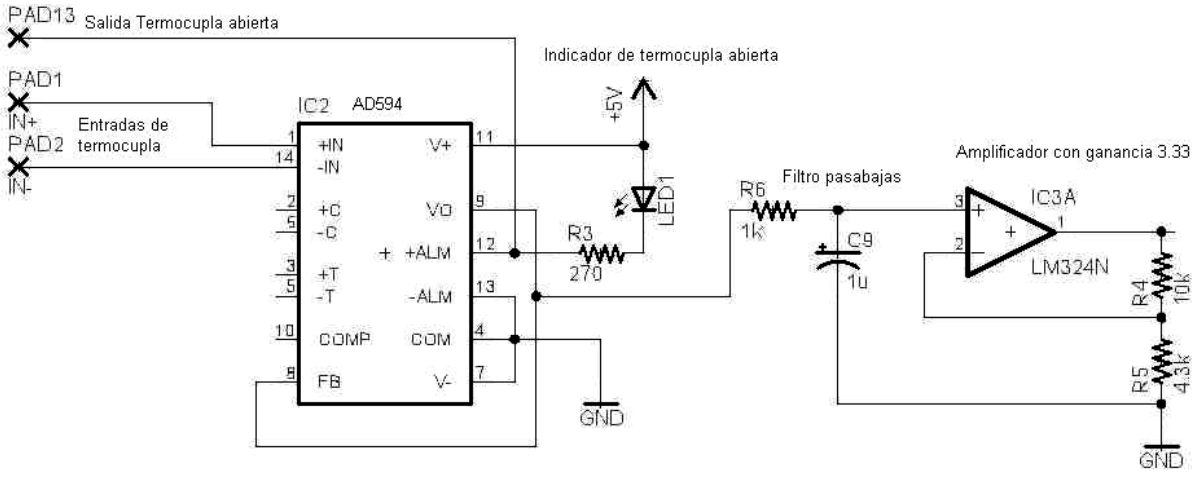


Figura 5.1 acondicionador para termocupla con AD594

Debido a que se hace uso de un amplificador no inversor la ganancia de este amplificador se calcula de la siguiente manera:

$$Ganancia = \frac{Vout}{Vin} = 1 + \frac{R_F}{R_1} \qquad (5.1)$$

Donde R_F es la resistencia de realimentación y R_1 la resistencia de entrada

Para una ganancia de 3.33 los valores resultaron en los cálculos de $R_F = 10K\Omega$ y $R_1 = 4.3K\Omega$.

Para hacer uso de los tres canales analógicos y así poder tener una precisión de 1 dígito fue necesario dividir el voltaje en rangos. La salida de voltaje de cero a diez voltios del acondicionador de termocupla es dividido entre los tres canales, el primero de los tres rangos es de cero a 3.33V (que sería aproximadamente 0 a 100 ºC) se introduce al primero de los tres canales análogos siendo previamente amplificado para obtener de cero a cinco voltios en todo el rango del canal. La ganancia de este amplificador es de 1.5 y es de tipo no inversor calculándose sus resistores de la misma forma mencionada en (5.1).

En los otros dos canales se hizo de la siguiente manera: Se utiliza un divisor de tensión para generar las referencias de 3.33V y 6.67V con tres resistores iguales a una referencia fija de 10V y se tienen dos restadores con ganancia de 1.5 donde una de sus entradas de resta, en uno de los casos, es la de 3.33V; la otra entrada proviene de la salida del acondicionador de termocupla, este restador, por lo tanto, tendrá una salida de 0 a 5 voltios cuando el acondicionador esté en el rango de 3.33V a 6.66V (100 a 200 ºC), esta salida alimentará al segundo de los tres canales analógicos.

Similarmente para el último de los canales analógicos el restador correspondiente tendrá en una de sus entradas la referencia fija de 6.67V, en la otra entrada la salida del acondicionador de termocupla, éste tiene también una ganancia de 1.5 donde se tendrá un voltaje de salida de 0 a 5 voltios cuando el acondicionador se encuentre en el rango de 6.67V a 10V (200 a 300 ºC).

La configuración de los restadores antes mencionados es la siguiente:

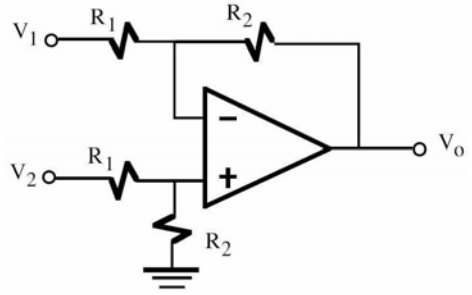


Figura 5.2 Configuración de restadores

La forma de calcular la ganancia es la siguiente:

$$Ganancia = \frac{Vout}{Vin} = \frac{R_2}{R_1}(V_2 - V_1) \qquad (5.2)$$

Para una ganancia de 1.5 en ambos restadores se utilizaron $R_2 = 15K\Omega$ y $R_1 = 10K\Omega$.

Un esquema del circuito completo se presenta en la siguiente figura:

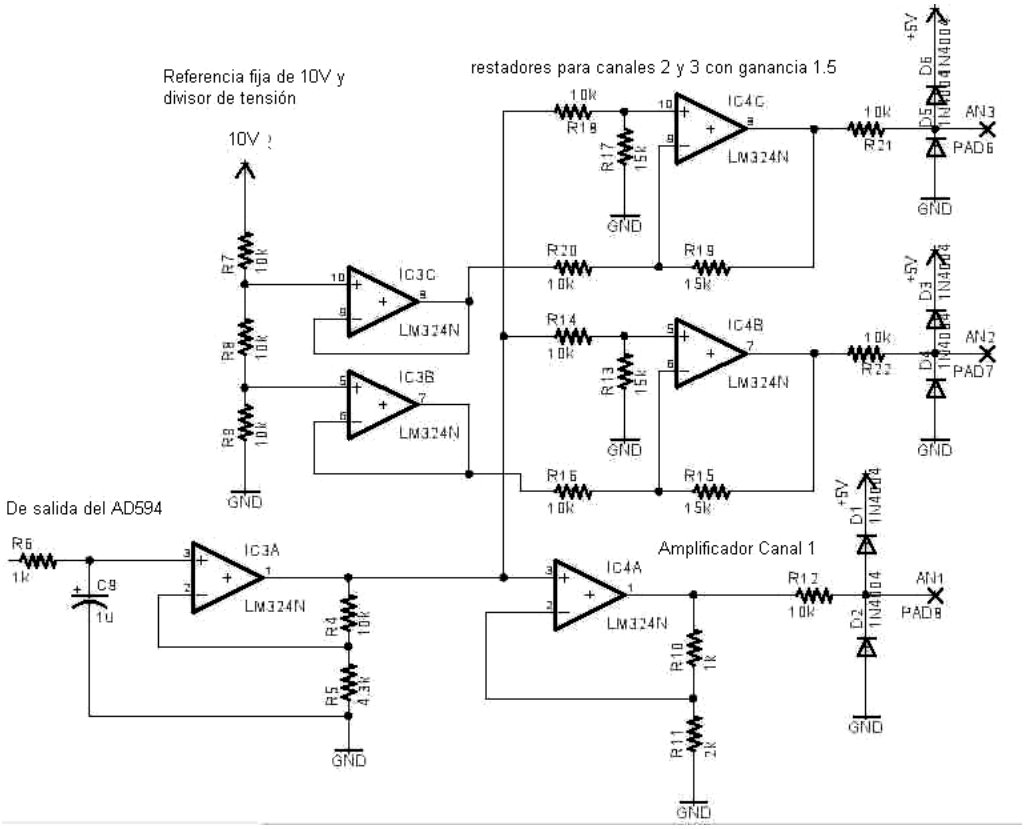


Figura 5.3 Circuito completo de canales analógicos

Los diodos a las salidas de este circuito tienen el objetivo de limitar el voltaje en la entrada analógica no sea mayor que la referencia fija de 5 V que ellos tienen.

Por medio de programa se hace que cuando se ha alcanzado el máximo voltaje del canal 1 entonces se comience a leer el canal 2 sumando un desplazamiento de 1024 a lo que se medirá; de igual forma se hace cuando se alcanza el máximo voltaje del canal 2, se mide ahora en el canal 3 sumando un desplazamiento de 2048.

5.2 Interfaz con el usuario.

Para hacer la interfaz con el usuario se hace uso de 3 botones (OK, arriba y abajo) para ingresar datos. Para visualizar la temperatura y las opciones del control, se hace uso de una pantalla LCD conectada al PIC por medio de dos puertos, uno para mandar datos que es el puerto B y el otro para mandar señales de control a lo que son las líneas del RS, E y RW. La figura 5.2 muestra un esquema básico de la conexión del PIC a la pantalla y al teclado.

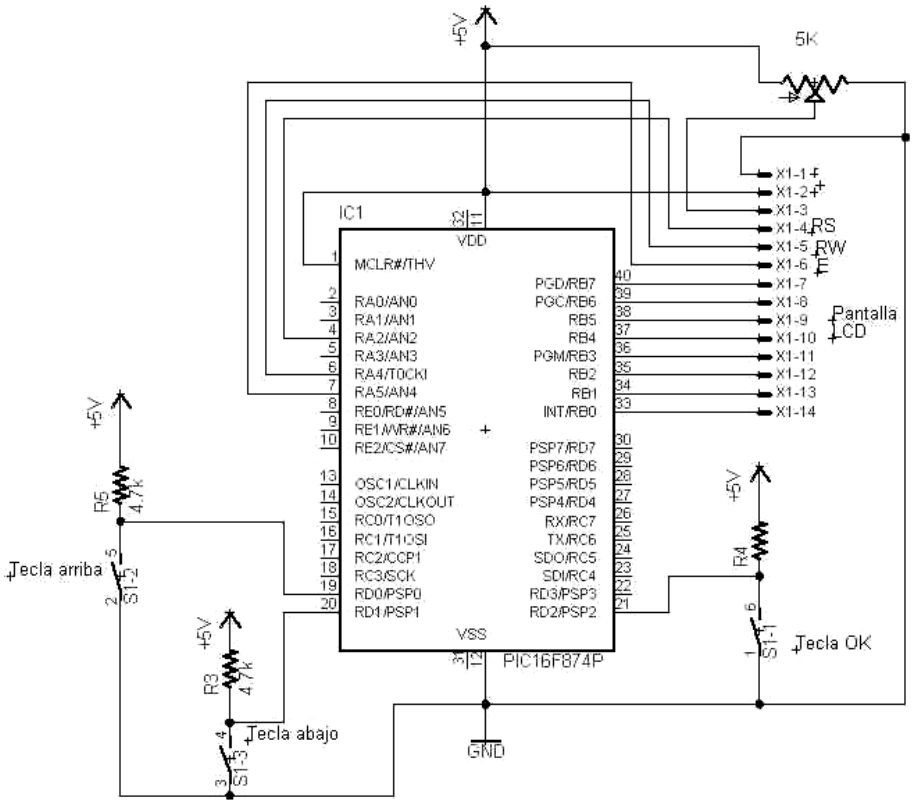


Figura 5.4 PIC con pantalla

Para el usuario experto que requiera poner manualmente los valores de operación PID (K_P, T_V y T_N) los rangos son:

K_P está en el rango de 0.03 y 1000

T_V está en el rango de 15ms y 20s

T_N está en el rango de 3.05ms y 31.99s

Los rangos fueron elegidos tratando de ajustarse a los controladores PID analógicos que hay en el laboratorio de Instrumentación y Control, aunque no son idénticos por las escalas que han tenido que ser implementadas.

Estos rangos deben tener una escala para poder ser ingresados al controlador, para eso se utilizan las siguientes fórmulas:

$$K_{P(pantalla)} = \frac{K_{P(real)} 32767}{1000}$$

$$T_{V(pantalla)} = \frac{T_{V(real)}32767}{20}$$

$$T_{N(pantalla)} = \frac{T_{N(real)}32767}{100}$$

Donde $K_{P(pantalla)}$, $T_{V(pantalla)}$ y $T_{N(pantalla)}$ son los valores a introducirse por los botones y desplegado en la pantalla.

La razón por la cual es necesario introducir de esta manera los datos es por la memoria limitada del PIC, ya que se están utilizando muchas funciones del PIC como lo son los convertidores A/D, la comunicación serie, los puertos de entrada/salida digitales (para manejo de la pantalla, alarmas y teclado) y también faltan ciertas funciones que se agregan como lo son el manejo de registros históricos de temperatura, la comunicación con la computadora, por esto se agregaron estas conversiones de los parámetros del controlador.

En las siguientes figuras se presentan los textos desplegados en la pantalla LCD en la ejecución del programa.

Figura 5.5 pantalla de saludo.

Figura 5.6 pantalla del termómetro (home).

Figura 5.7a pantalla de menú.

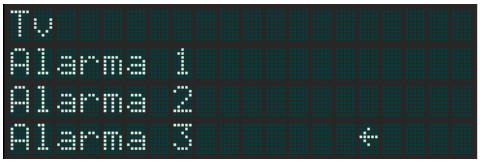


Figura 5.7b pantalla de menú.

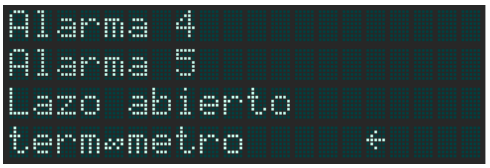


Figura 5.7c pantalla de menú.

Figura 5.8 pantalla de calibración.

Figura 5.9 pantalla de ajuste de la referencia.

Figura 5.10 pantalla de ajuste del Kp.

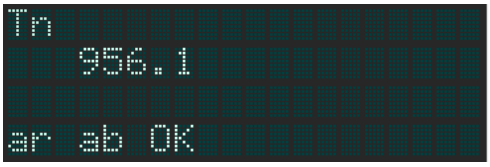


Figura 5.11 pantalla de ajuste del Tn.

Figura 5.12 pantalla de ajuste del Tv.

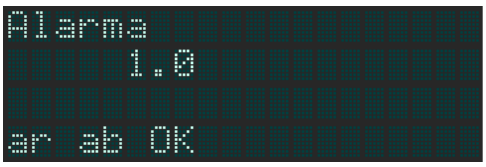


Figura 5.13 pantalla de ajuste de las alarmas. Figura 5.14 Lazo abierto.

5.3 Ingeniería Humana Empleada en el diseño de la Interfaz. [7]

Los aspectos de Ingeniería humana de la interfaz entre el proceso y el operador se están volviendo cada vez más importantes. En términos de diseño de aparatos electrónicos, la ingeniería humana se refiere al estudio de las características físicas y psicológicas del promedio de los operadores para funcionar lo más eficientemente posible. Por lo tanto, la mayor preocupación del ingeniero- diseñador es crear una interfaz eficiente.

En el desarrollo de los sistemas, Roger Pressman, considera un paso importante el tema de Ingeniería Humana1, y define así: "es una actividad multidisciplinaria que aplica un conocimiento derivado de la

psicología para especificar y diseñar la Interacción hombre máquina (IHM) de alta calidad". El proceso de la ingeniería humana comprende:

- Análisis de actividad, identificar tareas y actividades
- Análisis y diseño semántico se define cada acción requerida por el usuario y de cada acción producida por la máquina
- Diseño del entorno del usuario para formar un entorno adecuado para el usuario aparte de considerar los elementos del sistema que interactúan como son: hardware, software, base de datos y otros debería incluir facilidades físicas.
- Creación de prototipos, se debería lograr con la contribución eficaz del usuario y teniendo en cuenta los pasos anteriores.

En el análisis de la actividad, Puede decirse que la actividad a realizar por el operario será nada más el cambio de valores o parámetros de operación del controlador, esa será la única interacción que se podría tener por parte del operario.

En el análisis y diseño semántico, Se han definido cada una de las acciones a tomar por el usuario empleando palabras o abreviaturas comprensibles para la mayoría de personas, y están reflejadas en cada una de las opciones que posee el menú de la pantalla, descrito en la sección anterior.

En el diseño del entorno del usuario, Se ha querido hacer la interfaz con el usuario lo más sencilla posible, nada más manejando las opciones por medio de un menú y solamente cambiando los valores mediante tres interruptores, también la letra en la pantalla es bastante grande y legible.

En la creación de prototipos, fue creado el prototipo básico del controlador pensando en la facilidad para el usuario, aunque con algunas limitantes por la memoria de programa del PIC, por esto no se ha podido lograr un mejor nivel de interacción.

5.4 Etapa de potencia.

El Controlador a su salida genera una señal de voltaje por medio de la interfaz SPI (Interfase Puerto Serie síncrono), que es convertido a paralelo por medio de un circuito integrado 74164, este tiene el dato de salida de 8 bits que es directamente conectado a un convertidor D/A cuya señal de salida es después conectada a una entrada de un comparador que tiene una señal de rampa en la otra de sus terminales, a la salida se genera una señal PWM[9] con ancho de pulso proporcional a la salida del controlador, la señal PWM alimenta un Relé de estado sólido que estará conmutando la señal de corriente alterna hacia lo que es el calefactor, dándole más o menos potencia dependiendo del voltaje de salida.

El diagrama a bloques de la etapa de potencia se presenta en la figura 5.12

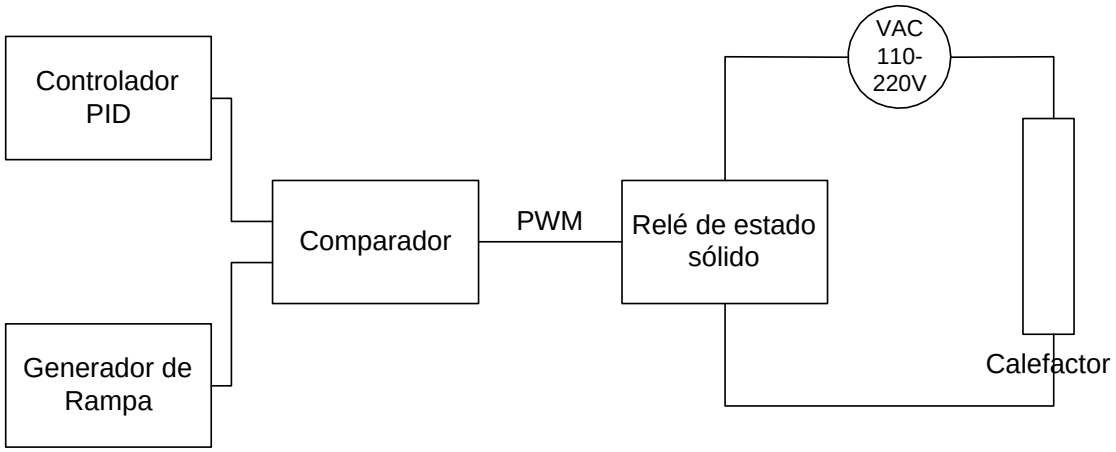


Figura 5.15 Diagrama a bloques de la etapa de potencia.

En la Figura 5.16 se muestra lo que es el generador de rampa conectado al comparador.

[9] PWM: Modulación de Ancho de Pulso, consiste en una señal de frecuencia fija que varía su tiempo en alto dependiendo en este caso del voltaje de salida del controlador.

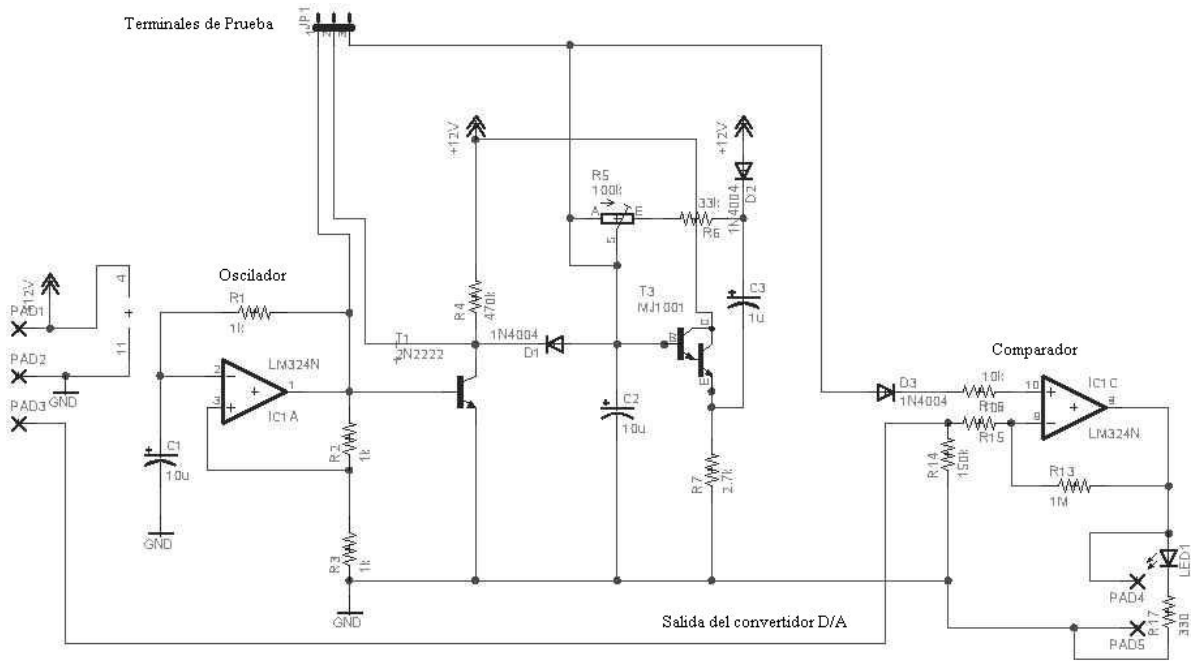


Figura 5.16 Generador PWM por medio de generador de rampa y comparador.

La razón de utilizar un PWM externo y no el que viene con el microcontrolador es porque el PWM interno tiene una frecuencia muy alta y no logra controlar eficazmente la temperatura que es una variable lenta, esto es porque cabe demasiadas veces dentro del ciclo de corriente alterna (60Hz), por ello, se hace uso de una señal que es proporcional a la variable de salida del filtro (escalada también) para hacer la generación de la señal PWM y así dejar pasar cierta cantidad de ciclos de corriente alterna dependiendo del tiempo en alto de la salida.

5.5 Conexión de Siteplayer con controlador PID.

Para hacer la comunicación de datos, se conectó el Siteplayer a la interfaz serie asíncrona que está en los pines RC6 y RC7, que son los pines 25 y 26 del 16F877A, por medio de programa se hace que se le esté enviando continuamente los datos de una tabla interna en el microcontrolador, la cual es refrescada en el sitio web del Siteplayer, que es visualizado por cualquier computadora que acceda a la dirección IP del servidor web, así se pueden estar monitoreando los datos de estado de temperatura que hay en el controlador.

Un diagrama a bloques representativo es el que se muestra en la figura 5.17

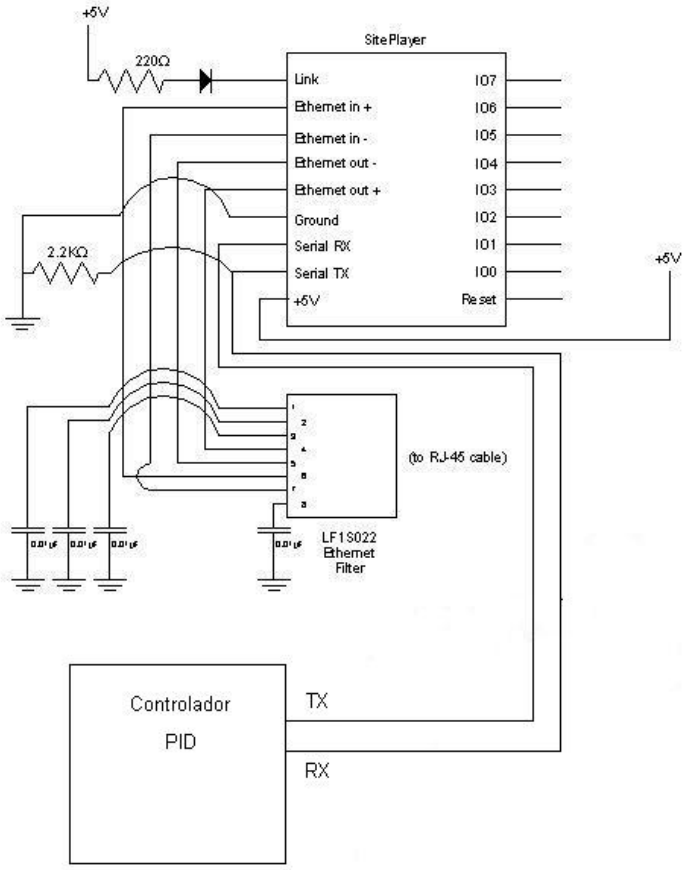


Figura 5.17. Forma de conexión del Controlador con Siteplayer.

5.6 Justificación de los Componentes Empleados.

Para la realización del proyecto se hizo uso de ciertos componentes, como por ejemplo, la termocupla tipo J, el microcontrolador PIC 16F877A, el circuito integrado AD594, los cuales fueron elegidos por ciertas razones de conveniencia, las cuales se procede a numerar a continuación:

El Microcontrolador PIC 16F877A, fue de los primeros componentes en adquirirse, y posee características que lo hacen preferirse de entre otros modelos, por ejemplo:

- Comunicación serie síncrona y asíncrona
- Suficiente memoria de programa para la aplicación

- 8 canales de conversión análogo a digital de 10 bits
- Disponible en el país
- Documentación, programas y dispositivo programador accesibles
- Bajo costo
- Suficientes puertos de entrada/salida digitales

En la siguiente tabla puede observarse una cotización hecha en el sitio http://microcontrollershop.com/default.php?cPath=112_184

PIC16F628A-I/P Microcontroller, 18 DIP, 20 MHz	PIC16F628AIP	US$3.18
PIC16F876A-I/SP Microcontroller, 28 DIP, 20 MHz	PIC16F876AIP	US$5.69
PIC16F877A-I/P Microcontroller, 40 DIP, 20 MHz	PIC16F877AIP	US$6.64
PIC18F258-I/SP Microcontroller, 28 DIP, 40 MHz, CAN, 32k Flash	PIC18F258ISP	US$7.83
PIC18F458-I/L Microcontroller, 44 PLCC, 40 MHz, CAN, 32k Flash	PIC18F458IL	US$9.27
PIC18F458-I/P Microcontroller, 40 DIP, 40 MHz, CAN, 32k Flash	PIC18F458IP	US$8.22

Tabla 5.1 comparación de precios de microcontroladores.

Puede observarse que el 16F877A mantiene un precio intermedio en comparación con los demás modelos, donde los inferiores a él tienen la desventaja de que no poseen la suficiente cantidad de pines de entrada/salida digitales que se requieren para la aplicación, los de la familia 18F, poseen funciones más avanzadas, pero que no tienen que ver con el proyecto y presentan un mayor costo aquí en el país, por lo que las capacidades intermedias del 16F877A han sido preferidas.

Termocupla tipo J, fue elegida por su capacidad de sensar la temperatura en el rango que ha sido propuesto en el presente proyecto. También su costo es menor que los de las termocuplas tipo K y tipo T descritas en la introducción teórica.
También la preferencia por una termocupla tipo J fue porque el integrado acondicionador de termocupla AD594, es más barato que el de termocupla tipo K AD595 ($13.00 de AD594 contra $21 de AD595) disponibles en el país (Cotizado en la venta de dispositivos electrónicos Electrónica J&R).

Acondicionador de señal de termocupla AD594, como ya se dijo anteriormente, este acondicionador, hace lineal la respuesta de la termocupla y que se presente

un voltaje a su salida de 10mV/ºC, tiene un costo, relativamente bajo a comparación de los circuitos equivalentes en hardware que habría que construir para lograr el mismo objetivo.

Servidor web Siteplayer, Este coprocesador ha sido necesario adquirirlo por la razón de tener que comunicar el controlador por medio de protocolo TCP/IP, para evitar el escribir demasiado código en el PIC, se emplea su puerto serie asíncrono para comunicarse con el Siteplayer, el cual hará la función de poner a disposición en protocolo TCP/IP los datos contenidos en la memoria del microcontrolador.

Relé de estado sólido, fue necesario utilizar este dispositivo debido a que se tienen que conmutar 110V ó 220V y es necesario sincronizarse con la línea de corriente alterna para tal efecto y sin producir el ruido que involucra un relé común magnético, él ya internamente posee los circuitos de potencia necesarios para hacer la sincronía con la línea y la conmutación.

5.7 Programa del controlador.

A continuación se presenta el flujograma de la función principal del programa (figura 5.17), aunque hay otras subrutinas que están detalladas en el anexo 1

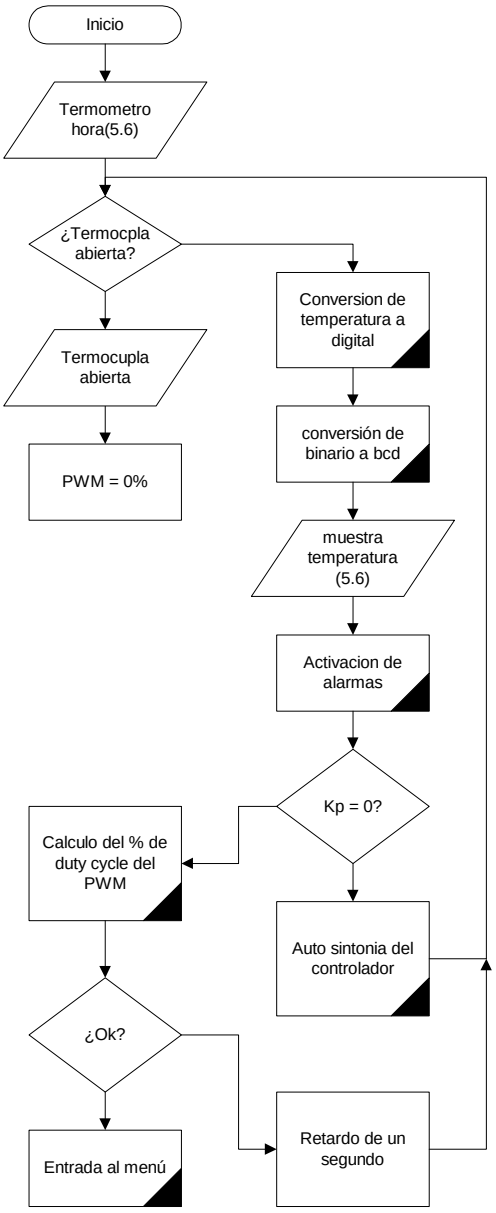


Figura 5.18 Flujograma del programa principal.

El flujograma que representa la entrada al controlador y la entrada al menú son los siguientes, en los bloques de entrada/salida se hace referencia a las pantallas que representan indicando el número de la figura entre paréntesis.

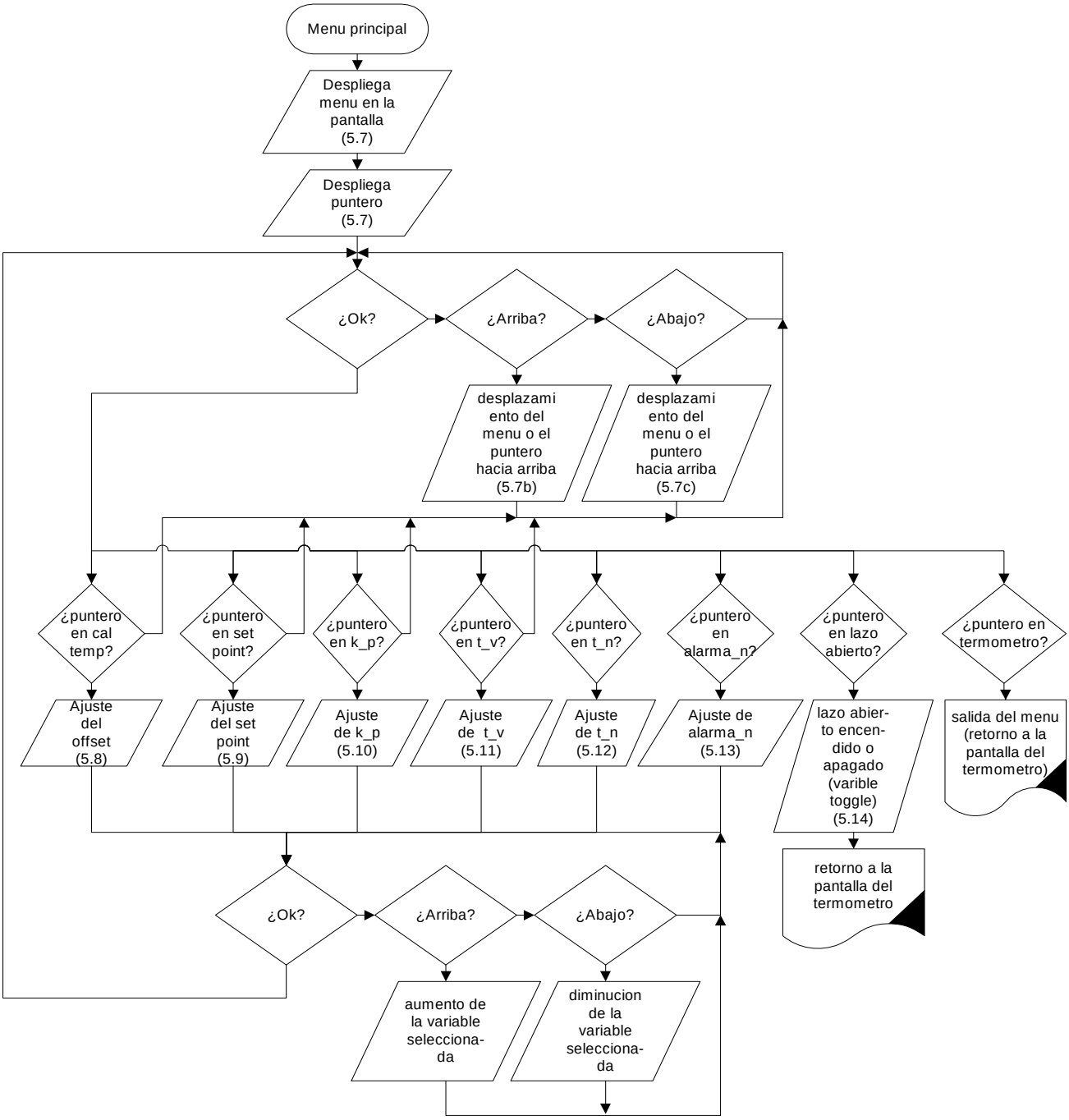


Figura 5.19 Flujograma del menú principal del PIC

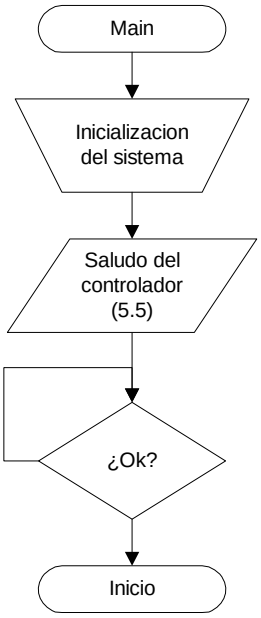


Figura 5.20 Flujograma de bienvenida del controlador

Las funciones de activación de alarmas y muestreo de temperatura se muestran a continuación en la figura 5.21.

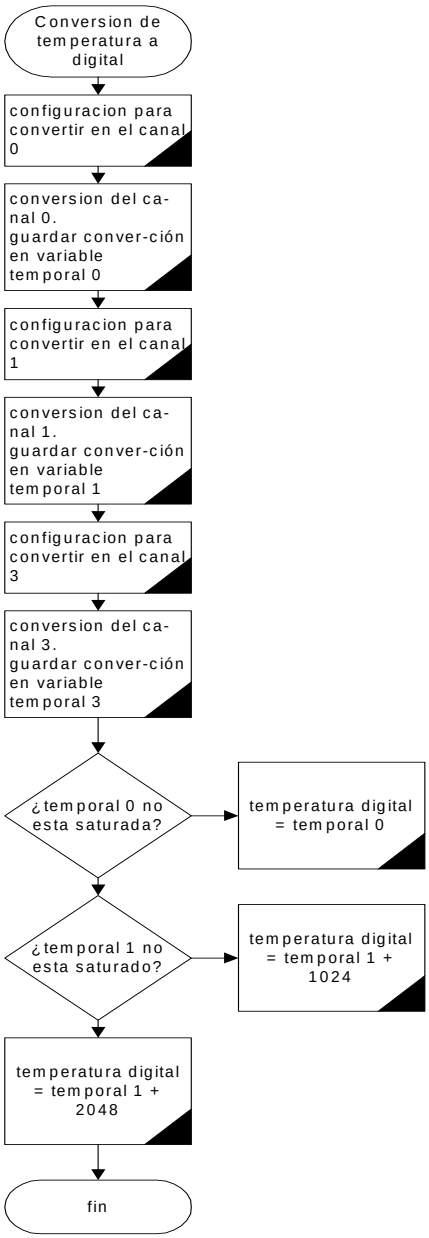


Fig. 5.21 Muestreo de temperatura y activación de alarmas.

5.8 Etapa de Autosintonía.

Haciendo referencia al apartado 4.4.1 y a las ecuaciones 4.28, 4.2 y 4.30 se conoce que $\mu = \frac{y_{\max} - y_0}{U_{ref} - U_0}$, $L = t_1 - t_0$ y $T = t_2 - t_1$.

La idea básica del algoritmo es encontrar el punto de inflexión de la respuesta de la planta, después por medio de la ecuación de punto pendiente se procede a encontrar la recta tangente a dicho punto, se evalúa para $y = y_0$ y se encuentra t_1, seguido de eso se evalúa para $y = y_{\max}$ y se obtiene t_2, y_0 es la respuesta de la planta en lazo abierto para un valor de referencia cualquiera, $y_{\max}$ es la respuesta de la planta en lazo abierto para el valor de referencia anterior más el cinco por ciento de la máxima temperatura de control, para este caso en específico, el cinco por ciento de trescientos grados es quince Celsius. U_0 es la referencia aplicada al inicio de la auto sintonía y U_{ref} es la referencia anterior más quince Celsius.

Después de haber obtenido estos datos se evalúan en las ecuaciones de la tabla 4.1 y se encuentra el K_P, t_i y t_d. El flujograma del algoritmo de auto sintonía se muestra en la figura 5.22

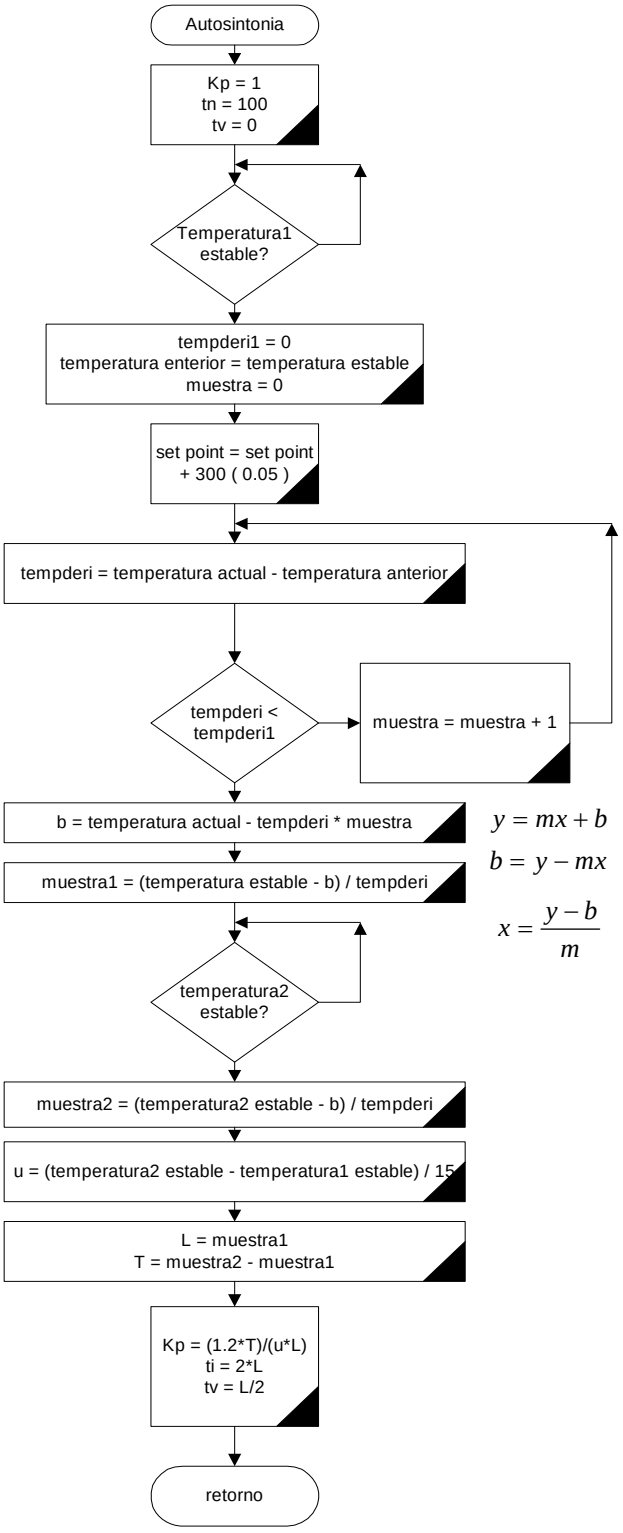


Fig. 5.22 Algoritmo de autosintonía.

6 Conclusiones y Recomendaciones

6.1 Conclusiones

- En el control de una variable física, como la temperatura, es indispensable la buena elección del transductor a emplear y de la linealidad de la conversión de dicha variable a una señal eléctrica para tal efecto, la termocupla tipo J tuvo una muy buena respuesta dentro del rango que se había especificado junto con los elementos de acondicionamiento y linealización.

- Para una lectura exacta de la variable es necesario un medio de calibración confiable, para este proyecto, se puso como opción la calibración por medio del mismo controlador, debido a las posibles desviaciones que tuviera el acondicionador y hacer todas los cambios de manera digital.

- El filtro PID con la tercera estructura directa tuvo muy buena respuesta con las plantas térmicas de prueba, con lo que pudo comprobarse que la teoría de control digital, con los algoritmos apropiados, es muy eficaz en la implementación práctica de sistemas de control a lazo cerrado.

- En todo proyecto de construcción de prototipos debe pensarse en la mejor presentación de los datos para las personas encargadas de manipular estos dispositivos, por esto, es importante aplicar cierto tipo de medidas encaminadas a mejorar la forma en la que se introducen los datos, se presentan las variables, se hace la interacción con el humano para lograr un uso eficiente y más productivo de la herramienta o equipo, por ello es importante considerar la ingeniería humana en los diseños de dispositivos para el uso industrial, o doméstico.

- El coprocesador servidor web Siteplayer tiene muchas aplicaciones en el campo de la industria, debido a que con él pueden actualizarse equipos para que puedan poseer una dirección IP y se pueda conocer su estado por Internet, sin necesidad de hacer adaptaciones con tarjetas de red comerciales o escribir todo el protocolo TCP/IP dentro de la memoria limitada de un PIC o un microprocesador y también posee una forma de configuración sencilla y económica. Cuenta ya con mucha documentación para que se puedan efectuar más proyectos en el futuro.

- Los microcontroladores PIC permiten generar aplicaciones de control con varias funciones internas, no requiriendo hardware adicional (memorias externas, interfases serie, interfases digitales paralelas, convertidores análogo a digital) necesitándose nada más de una buena administración de sus capacidades mediante su programa interno. Hay algunos que ya tienen la capacidad de comunicación vía Ethernet o USB integrada, pero su costo es bastante elevado.

6.2 Recomendaciones

- Es importante motivar a la construcción de proyectos que impliquen el manejo de variables físicas por medios electrónicos, debido a que esto familiariza a las personas que los ejecutan con dispositivos comerciales que se encontrarán después en el campo de trabajo en la industria.

- En el presente documento se encuentran dos funciones muy útiles como lo son la rutina BCD, que convierte números binarios a números BCD y también la forma de multiplicar dos números binarios con signo de 16 bits (aplicación del algoritmo de Booth), estas fueron necesarias porque se tenía que trabajar constantemente con esos formatos de números, pueden ser necesarias en caso que se tenga que trabajar como en este proyecto con eficiencia en las rutinas y que se tenga espacio limitado en memoria, por

ello se recomienda se puedan hacer uso del ejemplo aquí planteado en otros proyectos.

- Se recomienda que siempre que se haga un monitoreo remoto de variables, se cuente con algún tipo de sistema de seguridad, el Siteplayer posee dos sistemas de seguridad, uno para evitar que se descargue otro programa o página web desde la red y otro para evitar que usuarios no autorizados tengan acceso a las páginas web internas.

7 Referencias Bibliográficas

[1] Digital Control System Analysis and Design. Charles L. Phillips, H. Troy Nagle, JR. Prentice-Hall. Capítulos 12 y 13. Biblioteca UDB.

[2] Katsuhiko Ogata, “Ingeniería de Control Moderna”. Segunda Edición. Prentice Hall. 1993.

[3] Leiva Gutiérrez, Renato Alberto y Vásquez Ortiz, Carlos René. Interfaz Gráfica para la Obtención del Modelo Matemático, Diseño de Controladores y Control de una Planta. Biblioteca Central Universidad Don Bosco, 2004.

[4] ElettronicaVenetta Manual teórico experimental código TG34 de módulo G34/EV: Transductores y control de temperatura Laboratorio de Instrumentación y Control, Universidad Don Bosco.

[5] Kazuo Sato “Introducción de PIC, CPLD y FPGA, Los Dispositivos que Sostienen al Mundo Moderno”.

[6] Netmedia Inc. Manual de Programación de Siteplayer.

[7] Ing. Gloria Teresita Huamaní Huamaní, INGENIERIA HUMANA EN EL ENTORNO INFORMATICO,

http://quipu.uni.edu.pe/public/revistas/theke/8/ingenieria.htm

8 ANEXO

En este apartado se presenta el código escrito en C++ y además se explica cada función ocupada haciendo referencia al bloque que la representa en los flujogramas de la aplicación.

```
#include <system.h>
#pragma CLOCK_FREQ 4190000 //define la frecuencia de reloj a utilizar en la compilación
#define enable       asm bsf PORTA,5 ;Activa señal E pin 6 del lcd
#define disable      asm bcf PORTA,5 ;Desactiva señal E
#define leer         asm bsf PORTA,4  ;Pone LCD en Modo RD pin 5 del lcd
#define escribir     asm bcf PORTA,4 ;Pone LCD en Modo WR
#define off_comando  asm bcf PORTA,2 ;Desactiva RS (modo comando) pin 4 del lcd
#define on_comando   asm bsf PORTA,2 ;Activa RS (modo dato
//**************************          primera linea     $80-$8f*****************
//**************************          segunda           línea     $C0-$cf*****************
//**************************          tercera  línea     $94-$9f*****************
//**************************          cuarta   línea     $D4-$df*****************
//  temperatura
const char *txt = "Menu%Ctrl temp%PIC PID%Pres OK%  temperatura  %ßC%Temp Cal%Referencia%Ajustes PID%termometro%ar ab OK%Kp%Tn%Tv%Lazo abierto%Alarma %1%2%3%4%5%Lazo abierto%  TA  %Historicos% escaleado%sintonia%";
//////////////////0    5        15      23      31              47 50      59        70        82        93
102105108111       124    323436384042       155    164            175            186
char disp0 = 0;
char disp1 = 0;
char disp2 = 0;
char disp3 = 0;
char num1;
char num2;
char num3;
char x;
char i;
char j;
char l;
char m;
char n;
long z;
```

```
char tecla_pulsada;
char teclado;
char temp_1;
long temp_2;                                    //variable de carga y descarga de los coeficientes;
char temp_3;                                    //en tabla y se puede usar
char lin_pun = 0x8f;
char puntero_var = 0;
char puntero_dir[] = {0x80,0xc0,0x94,0xd4};
char logger[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char logger1[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char menu_var;
char band_menu = 0;
char menu_offset;
char historicos = 1;
char historicos1;
//********************autot********
char as = 0;
char atb;
char atb1;
char cont;
char cont1;
long bcd_num_ant;
long bcd_num_ran;
char tangente;
char tangente_ant;
char tangente_max;
char paso;
char paso1;
char u0;
char u;
//*****************************
long conver;
long conv0;
long conv1 = 0;
long conv3 = 0;
long y;
long offset = 0;
long offset_temp = 0;
```

```
long bcd_num;
char temp_bcd;
char bcd1;
char bcd2;
char bcd3;
char cod[7];
char neg = 0;
char menu_gen;
char tabla;
long dato = 0;
char tiemp_log = 0;
//**************************** controlador ********************
long k_p = 0x300;
long t_n = 0x7fff;
long t_v = 0x0;
long k_p_ant;
long t_n_ant;
long t_v_ant;
long alarma_1 = 500;
long alarma_2 = 400;
long alarma_3 = 300;
long alarma_4 = 200;
long alarma_5 = 100;
long set_p = 2500;
long set_p_ant;
char lazo_a = 0;
long sal = 0;
long sal1 = 0;
long sal_r;
long t3 = 0x0;
long t3_ant = 0;
long error = 0;
long error1 = 0x0;
long error2 = 0x0;
long a0;
long a1;
long fn;
long fp;
```

```
long fv;
long productoL;
long productoH;
char bitder;
char tmp;
char tmp1;
long signo;
char neg1;
char div;
//*****************************************************************************
//*****************************************************************************
void main_init()@0x800; //inicialización las funciones del pic
void lcd_ini(); //inicialización de la pantalla.
void lcd_addr(); //función que ocupa mensaje() para dar la dirección de un carácter en la pantalla
void lcd_write(); //función que ocupa mensaje() para escribir un carácter en la pantalla
void coeficientes(); //rutina donde se calculan los coeficientes del filtro PID
void saludo(); //despliega el saludo del controlador
void mensaje(); //función para sacar un mensaje
void chequea_ok(); //revisión de la tecla ok
void termometro_hora(); //despliega la pantalla del termómetro
void conversion(); //rutina que convierte las tres señales análogas y las presenta en conver
void entero(); //rutina que dice si un numero es positivo o negativo
void bcd(); //rutina de conversión de binario a bcd
void alarmas();  //función que activa las alarmas
void salida(); //función donde esta programado el filtro PID
void chequea_teclado(); //chequeo de las teclas arriba y abajo
void presenta_puntero(); //presentación del puntero
void menu_prins_1()@0x1000;  //despliega el menú del controlador
void clr_pun();   //función del puntero que prepara la nueva posición del puntero
void cal_menu(); //menú de calibración
void pid_ajus();
//rutina de multimenus donde se selecciona qué magnitud se cambiará con up_down()
void hs01(char); //función que manda datos por el puerto rs232 para la comunicación con el
                 //site player
void site_player(); //función que sirve para comunicar al PIC con el SITE PLAYER
void up_down(); //rutina que suma o resta las variables de los menús
void puntero();  //función que controla los movimientos del puntero para la presentación del menú
long multi(long, long)@0x1800; //función de multiplicación
```

```
void rota14r();   //función para dividir entre 2^15 o 2^6 dependiendo de div
void autot();//función donde está programada la rutina de auto sintonía
//*****************************************************************************
//*****************************************************************************
main()//rutina principal
        {
        main_init();//inicializacion de las variables
inicio:
        chequea_ok(); //si se teclea ok se sale del saludo y entra al termómetro
                        //sino se sigue viendo el saludo
        asm BANKSEL _tecla_pulsada;
        if (tecla_pulsada)
                goto t_h;
        else goto inicio;
t_h:
        termometro_hora(); //despiega la pantalla del termómetro
        puntero_var = 0;
        menu_var = 0;
        lin_pun = 0x8f;
t_h_t:
        if (!(portc & 0x01))
                {
                temp_1 = 0xc3;
                lcd_addr();
                i = 155;
                mensaje();
                sspbuf = 0;
                }
        else
                {
                conversion();//convierte la señal de temperatura de 0.0
                        //a 300ºC a valor digital desde 0 a 3069
                temp_2 = offset;
                temp_2 &= 0x8000;
                if (temp_2)
                        {
                        neg = 1;
                        }
```

```
                entero();
                bcd();    //convierte el valor binario de temperatura +
                          //el valor de calibración a código bcd
                alarmas(); //verifica si se tienen que encender las alarmas
                if (historicos)
                          {
                          historicos1 = historicos;
                          site_player();
                          }
                if (!as) //si hay un valor de kp, se activa la salida sino
                          salida(); //se pasa a la rutina de auto sintonía
                else
                          {
                          autot();
                          salida();
                          }
                }
        chequea_ok();//si se aprieta ok entra al menú
                          //sino se toma otra muestra de temperatura
        asm BANKSEL _tecla_pulsada;
        if (tecla_pulsada) //entra a la pantalla del termómetro otra vez
                {
                sspbuf = 0xff; //se pone la salida a cero para proteger la resistencia
                goto m_p;
                }
        else
        {
                delay_s(1); //retardo para esperar un segundo entre muestra y muestra

                goto t_h_t;
        }
m_p:
        menu_prins_1(); //llamada a la pantalla del menú
        }
//*************************************************************************************
//********************************* inicialización de variables **********************
//*************************************************************************************
main_init()
```

```
        {
        trisa = 0x0b; //pa7, pa6, pa5, pa4, pa2 como salidas, los otros como entradas
        trisb = 0x00; //todo el puerto b como salida
        trisd = 0x07; //pd0, pd1 y pd2 como entrada, los otros como salida
        intcon = 0x00; //sin interrupciones
        cmcon = 0x07; //el comparador del puerto a como señales ttl
        porta = 0;
        portb = 0;
        portd = 0;
        adcon0 = 0x81;
//(1Fh) frecuencia de conversión: fosc/64, selección de canal x & 0b00111000, estado
//de conversión (x & 0x4), modulo a/d on/off (x & 0x01)
        adcon1 = 0xc4;
//(9Fh)justificado a la derecha (x & 0x80), fosc/64, configuración de los canales (x & 0x0f) an3,
//an1, an0, los otros digitales vdd, vss
        trisc = 0xd1; //configuración del puerto de comunicación serial pc5 como entrada
        sspstat = 0x40;//Configure register for SPI..(b6=0). Mode 1,1 SPI, middle of output
        sspcon = 0x32;//Configure register  for SPI, Mode 1,1 SPI Master Mode, 1/16 Tosc bit
        spbrg = 26;//selecciona el Bautrate a 9,600Kbps
        set_bit (trisc,6);//habilita el pin 6 del puerto c como salida
        set_bit (trisc,7);//habilita el pin 7 del puerto c como salida
        txsta = 0x24; //transmisión habilitada
        rcsta = 0x90; //habilitación del puerto serial y habilitación de la recepción de datos
        cod[7] = 0x00;
        cod[1] = '.';
        lcd_ini(); //inicialización de la pantalla
        coeficientes();   //cálculo de los coeficientes pata el filtro
        sspbuf = 0xff;
        saludo();//pantalla de saludo
        }
salida() //filtro PID
        {
        neg = 0;
        if (lazo_a) //si se seleccionó lazo abierto no se resta la temperatura de realimentación
                error = set_p;
        else error = set_p - bcd_num;
//**********************
//**********************
```

```
//** y = salida del filtro para convertirlo en voltaje
//y = a0 * x + t3
//** a0 = constante del filtro (se calculo en coeficientes())
//** x = error actual
//********************** t3 = valor del pre-procesamiento
//**********************
//primeramente se hace y=0 para inicializar el cálculo, después se multiplica el valor del error
//actual por la constante a0 y por último se le suma algebraicamente t3 (t3 puede tomar
//valores positivos o negativos) para obtener el valor de la salida actual.
        sal = 0; //y = 0
        multi(error,3);
        if (error & 0x8000)
                {
                productoL = ~productoL;
                productoL++;
                neg = 1;
                }
        error = productoL / 0xa;
        if (neg == 1)
                {
                error = ~error;
                error++;
                }
        multi(a0,error);//a0 * x

        if (t3 & 0x8000)
                {
                set_bit (neg,1);
                }
        if (!neg)
                {
                sal = productoL + t3;

                if (sal & 0x8000)
                        {
                        sal = 0x7fff;
                        }
                }
```

```
        else
                {
                if (neg == 1)
                        {
                        sal = productoL;
                        sal = sal + t3;
                        }
                if (neg == 2)
                        {
                        sal = t3;
                        sal = sal + productoL;
                        }
                if (neg == 3)
                        sal = 0;
                if (sal & 0x8000)
                        sal = 0;
                }
        sal1 = sal;         //se guarda la salida y el error para los cálculos de t3
        t3_ant = t3;
        error2 = error1;
        error1 = error;
//*****************************************************************************
//*****************************************************************************
// t3 = a1 * x1 + a2 * x2 - b1 * y1 - b2 * y2  a2 = kd  b1 = -1  b2 = 0
// t3 = a1 * x1 + kd * x2 + y1
//*****************************************************************************
// t3 = a1 * x1 + (fv * kp)/2^6 * x2 + y1
//*****************************************************************************
//*****************************************************************************
//a1 = constante calculada en coeficientes()
//x1 = error de una muestra anterior
//x2 = error de dos muestras anteriores
//y1 = salida de una muestra anterior
//*****************************************************************************
//*****************************************************************************
//t3 es el valor de preprocesamiento y se sumará al valor de la salida en el próximo muestreo.
//primeramente se borra t3, luego se le suma el valor de la salida de dos muestras atrás,
//después se calcula el valor de tv y se multiplica por el valor del error de hace dos muestras
```

```
//para luego terminar con la suma de el valor del error de hace una muestra multiplicado
//por la constante a1 (todas las sumas son algebraicas)
        neg = 0;
        multi(a1,error1);
        t3 = productoL;

        fv = t_v / 0x200;
        multi(40,fv);
        multi(productoL,k_p);
        fv = productoL / 0x40;
        multi(fv,error2);

        t3 = t3 + productoL;

        if (t3 & 0x8000)
                neg = 1;

        t3 = t3 + sal1;

        if (!neg)
                {
                if (t3 & 0x8000)
                        {
                        t3 = 0x7fff;
                        }
                }
        sal_r = sal >> 7;
        multi (sal_r,4);
        sal_r = productoL/5;
        sal_r = 0xcc - sal_r;
        sspbuf = sal_r;
//se escalea la salida para que el valor máximo (0x7ff) sea 0xff, se multiplica  por 4/5 y se resta
//de 0xcc (4) esto se hace debido a que cuando el valor sea máximo la salida debe estar apagada
//y cuando el valor sea mínimo, la salida debe estar saturada. Se resta de 4 porque arriba de 4
//la salida siempre estará apagada y se saca en forma serial por el mssp del pic*/
regre_mssp:
        if (sspstat & 0x01 == 0)
                goto regre_mssp;
```

```
        sspbuf;
        }
//****************************** funciones de pantalla ******************************
lcd_write()
//el valor que está en temp1 lo saca por el puerto B y después le da la señal de habilitación
//a la pantalla para escribir en ella.
        {

        portb = temp_1;
                        //********
        enable;
        disable;
        delay_ms(10);
                        //********
        }
lcd_addr()
//pasa la pantalla a modo escritura y carga en ella temp1 como la dirección en la que se escribirá
        {
        off_comando;
        portb = temp_1;
        enable;
        disable;
        delay_ms(10);
        on_comando;//********
        }
lcd_ini()
//inicializa la pantalla. Primeramente le da un tiempo de espera a la pantalla para que el
//voltaje de alimentación se estabilice luego envía las palabras de configuración
//para que desaparezca el puntero, borre la pantalla, apunte a la dirección de inicio,
//e incremente de la dirección automáticamente
        {
        delay_s(1);       //********
        portb = 0x38;
        set_bit(porta,2);
        clear_bit(porta,2);//********
        delay_ms(5);
        portb = 0x38;
        set_bit(porta,2);
```

```
        clear_bit(porta,2);         //********
        delay_ms(1);
        temp_1 = 0x38;
        lcd_write();//********
        temp_1 = 0x38;
        lcd_write();
        temp_1 = 0x0c;
        lcd_write();//********
        temp_1 = 0x01;
        lcd_write();
        delay_ms(2);
        temp_1 = 0x06;//********
        lcd_write();
        }
//*****************************************************************************
//*****************************************************************************
//************************************ chequea ok *********************************
//*****************************************************************************
chequea_ok()
//debido a que en nuestro circuito la tecla ok se encuentra en el pin3 del puerto d, tomamos el
//valor del puerto d y lo enmascaramos con 0x04. Si el resultado es verdadero la tecla ok fue
//pulsada.
        {
        tecla_pulsada = 0;
        teclado = portd & 0x04;
        asm BANKSEL _teclado;
        asm     btfss _teclado,2;// ok = 1?
        goto salida_ok;
        asm BANKSEL _tecla_pulsada;
        tecla_pulsada = 1;
espera_ok:
        teclado = portd & 0x04;
        asm BANKSEL _teclado;
        asm     btfsc _teclado,2;
        goto espera_ok;
salida_ok:
        }
//*****************************************************************************
```

```
//************************************ chequea teclado ****************************
//*********************************************************************************
chequea_teclado()
//Debido a que las teclas arriba y abajo están ubicadas en el bit0 y bit1 del puerto d para
//para reconocerlas, enmascaramos el puerto d para con 0x03, si el resultado es verdadero una
//de las dos fue presionada
        {
        teclado = portd & 0x03;
        switch (teclado)
                {
                case 1:
                        tecla_pulsada = 2;
                        break;
                case 2:
                        tecla_pulsada = 3;
                        break;
                }
        }
//*********************************************************************************
//******************************** presenta puntero ********************************
//*********************************************************************************
presenta_puntero()
//Cuando se está dentro del menú, esta función presenta el puntero que es <- para esto
//primero le decimos a la pantalla la dirección donde lo deberá escribir, la dirección está
//guardada en lin_pun y después le decimos a la pantalla que deberá escribir la flecha dándole
//el valor en hexadecimal. (0x7f)
        {
        temp_1 = lin_pun;
        lcd_addr();
        temp_1 = 0x7f; //0x7f es la flecha hacia la izquierda
        lcd_write();

        }
//*********************************************************************************
//*************************** mensaje *********************************************
//*********************************************************************************
mensaje()
//Toma el valor de la tabla txt apuntada con i, para desplegar el mensaje que está en ella.
```

```
//al final la función busca el símbolo de porcentaje para poder salirse de la tabla y esperar
//el próximo mensaje que está en la misma tabla pero apuntada con otro valor de i.
        {
regre_msg:
        temp_1 = txt[i];
        if (temp_1 == '%')
                goto afuera;
        lcd_write();
        i++;
        goto regre_msg;
afuera:
        }
//************************************ saludo ****************************************
saludo() //Despliega el mensaje de saludo del controlador
        {
                                        //******* Cntrl Temp
        temp_1 = 0x80;//******* PIC PID
        lcd_addr();         //*******
        i = 5;              //******* pres OK
        mensaje();
        temp_1 = 0xc0;
        lcd_addr();
        i = 15;
        mensaje();
        temp_1 = 0xd4;
        lcd_addr();
        i = 23;
        mensaje();
        }
//********************************* termómetro hora ***********************************
termometro_hora() //Despliega la pantalla del termómetro
        {
        temp_1 = 0x01;//******** fecha hora
        lcd_addr();   //******** temperatura ºc
                      //******** "lazo abierto"
        temp_1 = 0x80;//******** Menu pres OK
        lcd_addr();
        i = 31;
```

```
        mensaje();
        temp_1 = 0xc9;
        lcd_addr();
        i = 47;
        mensaje();
        temp_1 = 0xd4;
        lcd_addr();
        i = 0;
        mensaje();
        temp_1 = 0xd9;
        lcd_addr();
        i = 23;
        mensaje();
        if(lazo_a)
//si en el controlador se habilitó la función de lazo abierto, lazo_a es verdadero y se despliega
//en la misma pantalla del termómetro el comentario (lazo abierto).
                {
                temp_1 = 0x94;
                lcd_addr();
                i = 142;
                mensaje();
                }
        if(as)
//si en el controlador se habilitó la función de auto sintonía, as es verdadero y se despliega
//en la misma pantalla del termómetro el comentario (sintonía).
                {
                temp_1 = 0x94;
                lcd_addr();
                i = 186;
                mensaje();
                }
        }
//*****************************************************************************
//************************* menú principal ***********************************
//*****************************************************************************
menu_prins_1()//función del menú principal
        {
menu_offset = 0;//***** temp cal<-
```

```
                    //***** referencia
menu_prins_1_ini://***** kp
                     //***** tn
          temp_1 = 0x01;//***** tv
          lcd_addr();   //***** alarma 1
                        //***** alarma 2
          j = 0;        //***** alarma 3
menu_aumen:             //***** alarma 4
                        //***** alarma 5
          temp_1 = puntero_dir[j];//***** lazo abierto
          lcd_addr();                 //***** termómetro
          i = 50;
          mensaje();
          j++;
menu_aumen_0:
          temp_1 = puntero_dir[j];
          lcd_addr();
          i = 59;
          mensaje();
          if (j == 3)
                    goto menu_aumen_sal;
          j++;
menu_aumen_1:
          temp_1 = puntero_dir[j];
          lcd_addr();
          i = 102;
          mensaje();
          i = 175;
          mensaje();
          if (j == 3)
                    goto menu_aumen_sal;
          j++;
menu_aumen_2:
          temp_1 = puntero_dir[j];
          lcd_addr();
          i = 105;
          mensaje();
          i = 175;
```

```
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_3:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 108;
        mensaje();
        i = 175;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_4:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 124;
        mensaje();
        i = 132;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_5:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 124;
        mensaje();
        i = 134;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_6:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 124;
```

```
        mensaje();
        i = 136;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_7:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 124;
        mensaje();
        i = 138;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_8:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 124;
        mensaje();
        i = 140;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_9:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 164;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_10:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 186;
```

```
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_11:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 142;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
        j++;
menu_aumen_12:
        temp_1 = puntero_dir[j];
        lcd_addr();
        i = 82;
        mensaje();
        if (j == 3)
                goto menu_aumen_sal;
menu_aumen_sal:
        presenta_puntero(); //presenta el puntero <-
puntero_desp:
        delay_ms(250); // tiempo necesario para evitar el rebote sin querer
        chequea_ok();  // si se teclea ok se pasa al menú interno
        chequea_teclado(); // se navega entro los menús
        if (!tecla_pulsada) // si no se pulso nada se espera a que haya algun pulsamiento
                goto puntero_desp;
        menu_gen = 1;
        puntero();
        if (band_menu == 1)
                {
                band_menu = 0;
                puntero_var = 0;
                menu_var = 0;
                lin_pun = 0x8f;
                goto menu_prins_1_ini;
                }
        goto puntero_desp;
```

```
	}
//*****************************************************************************
//********************************** puntero ************************************
//*****************************************************************************
puntero()
//Decide que hacer al detectar que una tecla fue pulsada. Si fue ok avanza en el menú, y si fue
//la tecla arriba o abajo se desplaza en el menú
	{
	switch (tecla_pulsada) //¿cual tecla fue pulsada?
		{
			case 2:
			puntero_var--; // ¿arriba?
			if (puntero_var == 0xff)
				{
				puntero_var =0;
				menu_var--;
				if (menu_var == 0xff)
					{
					puntero_var =3;
					menu_var = 10;
					}
				}
			break;
		case 3: // ¿abajo?
			puntero_var++;
			if (puntero_var == 4)
				{
				puntero_var = 3;
				menu_var++;
				if (menu_var == 11)
					{
					puntero_var =0;
					menu_var = 0;
					}
				}
			break;
		case 1: // ¿ok?
			goto menu_in;
```

```
                    break;
                default:
                    goto fin_pun;
                }
pos_pun:
        lin_pun = puntero_dir[puntero_var] + 0x0f;  //¿en cual línea se pondrá el cursor?
        switch (menu_var)//desplazamiento de los menús en la pantalla
                {
                case 0:
                    temp_1 = 0x01;
                    lcd_addr();
                    j = 0;
                    goto menu_aumen;
                    break;
                case 1:
                    temp_1 = 0x01;
                    lcd_addr();
                    j = 0;
                    goto menu_aumen_0;
                    break;
                case 2:
                    temp_1 = 0x01;
                    lcd_addr();
                    j = 0;
                    goto menu_aumen_1;
                    break;
                case 3:
                    temp_1 = 0x01;
                    lcd_addr();
                    j = 0;
                    goto menu_aumen_2;
                    break;
                case 4:
                    temp_1 = 0x01;
                    lcd_addr();
                    j = 0;
                    goto menu_aumen_3;
                    break;
```

```
        case 5:
                temp_1 = 0x01;
                lcd_addr();
                j = 0;
                goto menu_aumen_4;
                break;
        case 6:
                temp_1 = 0x01;
                lcd_addr();
                j = 0;
                goto menu_aumen_5;
                break;
        case 7:
                temp_1 = 0x01;
                lcd_addr();
                j = 0;
                goto menu_aumen_6;
                break;
        case 8:
                temp_1 = 0x01;
                lcd_addr();
                j = 0;
                goto menu_aumen_7;
                break;
        case 9:
                temp_1 = 0x01;
                lcd_addr();
                j = 0;
                goto menu_aumen_8;
                break;
        case 10:
                temp_1 = 0x01;
                lcd_addr();
                j = 0;
                goto menu_aumen_9;
                break;
        }
clr_pun(); //borra el puntero
```

```
        presenta_puntero(); // presenta el puntero en la nueva posición
        goto fin_pun;
menu_in:
        temp_3 = puntero_var + menu_var; // si la tecla pulsada fue ok
                                         // se decide a que menú entrar

        if (temp_3 == 13)  // si el cursor estaba apuntando a termómetro
                goto t_h;  // salta al termómetro
        if (temp_3 == 12)
                {
                if (lazo_a)
                        { //salta al termómetro con el controlador en lazo abierto
                        lazo_a = 0;
                        k_p = k_p_ant;
                        t_n = t_n_ant;
                        t_v = t_v_ant;
                        coeficientes();
                        }
                else //sino salta al termómetro deshabilitando la función de lazo abierto
                        {
                        lazo_a = 1;
                        k_p_ant = k_p;
                        t_n_ant = t_n;
                        t_v_ant = t_v;
                        k_p = 0x20;
                        t_n = 0x7fff;
                        t_v = 0;
                        coeficientes();
                        }
                goto t_h;
                }
        if (temp_3 == 11)
                { //si apuntaba a sintonía
                if (as)
                        { //salta al termómetro con el controlador en la función de auto sintonía
                        as = 0;
                        k_p = k_p_ant;
                        t_n = t_n_ant;
```

```
                t_v = t_v_ant;
                set_p = set_p_ant;
                coeficientes();
                }
        else //sino, si ya estaba en la función de auto sintonía la deshabilita y
                // regresa al termómetro en modo normal.
                {
                as = 1;
                k_p_ant = k_p;
                t_n_ant = t_n;
                t_v_ant = t_v;
                set_p_ant = set_p;
                k_p = 0x20;
                t_n = 0x7fff;
                t_v = 0;
                set_p = 350;
                coeficientes();
                }
        goto t_h;
        }
    if (temp_3 == 0) // si apuntaba a temp cal
        cal_menu();     // salta al menú de calibración
    if (temp_3 > 0)  // si apuntaba a cualquiera de los otros
        pid_ajus();     // salta a un menú multi función que decide que
                        // qué variable cambiar
fin_pun:
    }
//*****************************************************************************
//************************** pid ajus ****************************************
//*****************************************************************************
pid_ajus()//Función donde se encuentran los saltos hacia los menús internos del controlador tales como:
          //las alarmas, kp, Tn, Ti y calibración
    {
    z = offset;     // mete offset en una variable temporal para poder trabajar
                    // con el menú cal
pid_ajus_ini:
    temp_1 = 0x01;
```

```
        lcd_addr();
        temp_1 = 0x80;
        lcd_addr();
        if (temp_3 == 1) // si la flecha esta apuntando a referencia
                {          // se aumenta o disminuye referencia
                i = 59;
                mensaje();
                temp_1 = 0xc9;
                lcd_addr();
                i = 47;
                mensaje();
                offset = set_p;
                }
        if (temp_3 == 2) //si kp
                {
                i = 102;
                mensaje();
                offset = k_p;
                }
        if (temp_3 == 3)//si tn
                {
                i = 105;
                mensaje();
                offset = t_n;
                }
        if (temp_3 == 4)
                {
//              asm movlw _Tv; //etc...
                i = 108;
                mensaje();
                offset = t_v;
                }
        if (temp_3 == 5)
                {
                i = 124;
                mensaje();
                offset = alarma_1;
                }
```

```
if (temp_3 == 6)
        {
        i = 124;
        mensaje();
        offset = alarma_2;
        }
if (temp_3 == 7)
        {
        i = 124;
        mensaje();
        offset = alarma_3;
        }
if (temp_3 == 8)
        {
        i = 124;
        mensaje();
        offset = alarma_4;
        }
if (temp_3 == 9)
        {
        i = 124;
        mensaje();
        offset = alarma_5;
        }
if (temp_3 == 10)
        {
        i = 164;
        mensaje();
        offset = historicos;
        }
temp_1 = 0xd4;
lcd_addr();
tabla = 1;
i = 93;
mensaje();
tecla_pulsada = 3;
offset++;
up_down();
```

```
        temp_2 = offset;

c_t_pid_m:

        chequea_ok();
        chequea_teclado();
        asm BANKSEL _tecla_pulsada;
        if (tecla_pulsada == 1)
                goto fin_pid_ajus;
        if (tecla_pulsada & 0x03 > 1)
                {
                offset = temp_2;
                up_down();
                temp_2 = offset;
                goto c_t_pid_m;
                }
        else goto c_t_pid_m;
fin_pid_ajus:
        offset = z;
        if (temp_3 == 1)
                {
                set_p = temp_2;
                }
        if (temp_3 == 2)
                {
                k_p = temp_2;
                coeficientes();
                }
        if (temp_3 == 3)
                {
                t_n = temp_2;
                coeficientes();
                }
        if (temp_3 == 4)
                {
                t_v = temp_2;
                coeficientes();
                }
```

```
        if (temp_3 == 5)
                {
                alarma_1 = temp_2;
                }
        if (temp_3 == 6)
                {
                alarma_2 = temp_2;
                }
        if (temp_3 == 7)
                {
                alarma_3 = temp_2;
                }
        if (temp_3 == 8)
                {
                alarma_4 = temp_2;
                }
        if (temp_3 == 9)
                {
                alarma_5 = temp_2;
                }
        if (temp_3 == 10)
                {
                historicos = temp_2;
                }
        band_menu = 1;
        menu_var = 0;
        puntero_var = 0;
        }
//*****************************************************************************
//***************************** clr pun ***************************************
//*****************************************************************************
clr_pun()
//Borra el puntero anterior para deplegar el nuevo puntero. Cada vez que hay un
//desplazamiento entre menús se ocupa esta función
        {
        temp_1 = 0x8f;
        lcd_addr();
        temp_1 = ' ';
```

```
        lcd_write();
        temp_1 = 0xcf;
        lcd_addr();
        temp_1 = ' ';
        lcd_write();
        temp_1 = 0x9f;
        lcd_addr();
        temp_1 = ' ';
        lcd_write();
        temp_1 = 0xdf;
        lcd_addr();
        temp_1 = ' ';
        lcd_write();
        }

//*****************************************************************************
//********************************** menú de calibración ***************************
//*****************************************************************************
cal_menu()
//Primero despliega el mensaje de calibración en la pantalla, luego presenta el valor actual de
//calibración y deja libre al usuario para que se pueda aumentar o dismunuir esta variable con
//las teclas arriba y abajo. Al final se acepta el valor con la tecla ok.
        {
        temp_1 = 0x01;
        lcd_addr();
        temp_1 = 0x80;
        lcd_addr();
        i = 50;
        mensaje();
        temp_1 = 0xc9;
        lcd_addr();
        i = 47;
        mensaje();
        temp_1 = 0xd4;
        lcd_addr();
        i = 93;
        mensaje();
        offset++;
```

```
        tecla_pulsada = 3;
        up_down();
c_t_c_m:
        chequea_ok();
        chequea_teclado();
        asm BANKSEL _tecla_pulsada;
        if (tecla_pulsada == 1)
                goto fin_menu_cal;
        if (tecla_pulsada & 0x03 > 1)
                {
                up_down();
                goto c_t_c_m;
                }
        else goto c_t_c_m;
fin_menu_cal:
        band_menu = 1;
        }
//*****************************************************************************
//************************** UP_DOWN ***********************************
//*****************************************************************************
up_down()
//Esta función se ocupa para aumentar y disminuir las variables de manera rápida, de tal
//manera que si el usuario mantiene presionado el botón de arriba o abajo, se aumentará
//o disminuirá la variable hasta que se suelte la tecla.
        {
        x = 1;
        j = 0;
regre:
        y = 0;
        conver = 0;
au_dis: //aumenta o disminuye la variable de salida
if (tecla_pulsada == 2)
                {
                if (offset == 0x7fff)
                        goto tope;
                offset = offset + x;
                temp_2 = offset;
                temp_2 &= 0x8000;
```

```
                if (temp_2)
                        {
                        neg = 1;
                        entero();
                        }
                else
                        {
                        neg = 0;
                        entero();
                        }
                }
        if (tecla_pulsada == 3)
                {
                if (temp_3)
                        {
                        if (offset == 0)
                                goto tope;
                        }
                else
                        {
                        if (offset == 0x8000)
                                goto tope;
                        }
                offset = offset - x;
                temp_2 = offset;
                temp_2 &= 0x8000;
                if (temp_2)
                        {
                        neg = 1;
                        entero();
                        }
                else
                        {
                        neg = 0;
                        entero();
                        }
                }
tope:
```

```
max_min://Si la variable es natural y su valor es cero, y se presiona la tecla hacia abajo, el valor de
//la variable seguirá siendo cero; pero si la variable es entera y se presiona la tecla hacia abajo
//la variable tomará valores negativos. De igual forma si se está en el valor máximo de la
//variable y se presiona la tecla hacia arriba la variable seguirá teniendo el valor máximo.
                temp_1 = 0xc0;
                lcd_addr();
                if (neg == 1)
                        {
                        temp_1 = '-';
                        lcd_write();
                        }
                else
                        {
                        temp_1 = ' ';
                        lcd_write();
                        }
                bcd();
                goto rebote;
rebote:
        if (portd & 0x03 != 0)
                {
                if (j < 10)
                        {
                        x = 1;
                        }
                else
                        {
                        if (j < 100)
                                {
                                x = 1;
                                }
                        else
                                {
                                if (j < 1000)
                                        {
                                        x = 1;
                                        }
                                }
```

```
                    }
                if (j < 255)
                    j++;
                goto regre;
                }
        else goto fin_up_down;
fin_up_down:
}
//*****************************************************************************
//***************************** entero ****************************************
//*****************************************************************************
entero()//Convierte un número hexadecimal negativo por ejemplo 0xff que normalmente es 255 en -
1
                //de tal forma que la variable toma el valor de 1 y se enciende la bandera neg. Esto
se obtiene
                //aplicando complemento a2 a la variable original.
{
        offset_temp = offset;
        if (neg)
                {
                offset_temp = ~offset_temp;
                offset_temp++;
                }
        disp0 = offset_temp;
        asm
                {
                BANKSEL _offset_temp+d'1'
                movf _offset_temp+d'1',w;
                BANKSEL _disp1;
                movwf _disp1;
                }
}
//*****************************************************************************
//********************** multiplicación ***************************************
//*****************************************************************************
multi(long multiplicando, long multiplicador)           // rutina de multiplicación signada
        {
        bitder = 0;
```

```
        i = 0;
        productoH = 0;
        productoL = multiplicador;
        signo = 0;
inicio_mul:
        tmp = productoL & 0x0001;
        if (tmp ^ bitder == 0)
        {
                goto comun;
        }
        else
        {
                goto sumaresta;
        }
sumaresta:
        tmp1 = productoL & 0x01;
                if (tmp1 == 1)
                {
                productoH = productoH - multiplicando;
                goto comun;
                }
                else
                {
                productoH = productoH + multiplicando;
                }
comun:
                signo = productoH & 0x8000;
                clear_bit(STATUS,C);
                asm BANKSEL _productoH + d'1'
                asm     rrf _productoH + d'1'
                asm BANKSEL _productoH
                asm     rrf _productoH
                asm BANKSEL _productoL + d'1'
                asm     rrf _productoL + d'1'
                asm BANKSEL _productoL
                asm     rrf _productoL
                bitder = STATUS         & 0x01;
                productoH = signo | productoH;
```

```
                clear_bit(STATUS,C);
                i++;
        if (i == 0x10)
        {
                goto end;
        }
        else
        {
                goto inicio_mul;
        }
end:
        return productoH;
        }
//*************************************************************************************
//********************************** rota14r ******************************************
//*************************************************************************************
rota14r() //Rutina que sirve para el escaleo de las variables. Si div es cero, divide entre 16
         //si div es 1 divide entre 6.
        {
        if (div)
                {
                i = 15;
                }
        else i = 6;
rota_ini:
        clear_bit(status,0);
        asm
                {
                BANKSEL _productoH
                rrf _productoH+d'1',1;
                rrf _productoH,1
                BANKSEL _productoL
                rrf _productoL+d'1',1;
                rrf _productoL,1
                }
        i--;
        if (i)
                goto rota_ini;
```

```
        }
//*****************************************************************************
//*************************** auto sintonia ***********************************
//*****************************************************************************
autot()
//Primeramente lleva la temperatura del equipo a un valor constante de temperatura en lazo abierto
//cuando la temperatura es estable le suma un escalón a la variable de consigna. Este escalón es
//del 15%. Mientras la temperatura sube el controlador vigila la derivada de la
//función de temperatura
//y espera a obtener la máxima derivada, cuando esto ocurre; guarda el valor de temperatura
//y el tiempo
//en que esto ocurrió. Después espera a que estabilice la temperatura para obtener
//el segundo valor de
//estabilización. Luego de esto, con ayuda de la ecuación punto pendiente y despejando
//el tiempo, la evalúa
//en temperatura de estabilización 1 y estabilización 2. Con esto consigue tiempo 1 y tiempo 2.
//Ya teniendo estos valores se ajusta con la tabla de autosintonización.
        {
        cont++;
        lazo_a = 1;
        if (cont == 255)
                {
                bcd_num_ran = bcd_num_ant++;
                if (bcd_num < bcd_num_ran)
                        {
                        bcd_num_ran = bcd_num_ant--;
                        if (bcd_num > bcd_num_ran)
                                {
                                atb++;
                                }
                        else atb = 0;
                        }
                }
        else goto fin_at;
        if (atb1)
                {
                cont1++;
                tangente_ant = tangente;
```

```
                tangente = bcd_num - bcd_num_ant;
                if (tangente > tangente_ant)
                        {
                        tangente_max = tangente;
                        paso = cont1;
                        }
                }
        if (atb == 7)
                {
                if (!atb1)
                        {
                        u0 = bcd_num;
                        set_p = set_p + 45;
                        atb1++;
                        atb = 0;
                        }
                else
                        {
                        u = bcd_num - u0;
                        u = u/15;
                        paso1 = cont1 - paso;
                        multi(12,paso1);
                        k_p = productoL / 10;
                        k_p = k_p / u;
                        k_p = k_p / paso;
                        t_n = paso << 1;
                        t_v = paso >> 1;
                        coeficientes();
                        }
                }
fin_at:
        }
//*****************************************************************************
//************************** alarmas ******************************************
//*****************************************************************************
alarmas() //Compara los valores de las alarmas seteadas en el menú principal con
//el valor de temperatura actual
 //si la temperatura es mayor que una de las alarmas enciende el bit del puerto d
```

```
//correspondiente a dicha alarma.
        {
        if (bcd_num > alarma_1)
                set_bit(portd,3);
        else clear_bit(portd,3);
        if (bcd_num > alarma_2)
                set_bit(portd,4);
        else clear_bit(portd,4);
        if (bcd_num > alarma_3)
                set_bit(portd,5);
        else clear_bit(portd,5);
        if (bcd_num > alarma_4)
                set_bit(portd,6);
        else clear_bit(portd,6);
        if (bcd_num > alarma_5)
                set_bit(portd,7);
        else clear_bit(portd,7);
        }
//*****************************************************************************
//******************************** conversion ***********************************
//*****************************************************************************
conversion() //Toma el valor análogo de los tres canales de temperatura en
             //el puerto a y devuelve en la variable
//conver el valor en hexadecimal de 12 bits entre 0 y 0xbb8 equivalente a 3,000 en decimal.
        {
        adcon0 = (adcon0 & 11000111b); //setea el canal cero listo para convertir
        delay_ms(10);
        set_bit(adcon0,2); //inicia la conversión
conversion0:
        if (adcon0 & 0x04 == 4) //espera el fin de la conversión
                goto conversion0;

        asm     //ordena los 10 bits de la conversión en conv0
                {
                BANKSEL ADRESH
                movf _adresh, W
                BANKSEL _conv0+d'1'
                movwf  _conv0+d'1'
```

```
                BANKSEL ADRESL
                movf _adresl, W
                BANKSEL _conv0
                movwf  _conv0
                }
        adcon0 = (adcon0 & 11000111b) | 0x08; //setea el canal dos listo para convertir
        delay_ms(10);
        set_bit(adcon0,2); //inicia la conversión
conversion1:
        if (adcon0 & 0x04 == 4) //espera el fin de la conversión
                goto conversion1;
        asm     //ordena los 10 bits de la conversión en conv1
                {
                BANKSEL ADRESH
                movf _adresh, W
                BANKSEL _conv1+d'1'
                movwf  _conv1+d'1'
                BANKSEL ADRESL
                movf _adresl, W
                BANKSEL _conv1
                movwf  _conv1
                }
        adcon0 = (adcon0 & 11000111b) ^ 0x18; //setea el canal uno listo para convertir
        delay_ms(10);
        set_bit(adcon0,2); //inicia la conversión
conversion3:
        if (adcon0 & 0x04 == 4) //espera el fin de la conversión
                goto conversion3;

        asm //ordena los 10 bits de la conversión en conv1
                {
                BANKSEL ADRESH
                movf _adresh, W
                BANKSEL _conv3+d'1'
                movwf  _conv3+d'1'
                BANKSEL ADRESL
                movf _adresl, W
                BANKSEL _conv3
```

```
                movwf  _conv3
                }
        if ((conv0 < 1023))
                {
                if (conv1 < 2)
                        {
                        if (conv3 < 2)
                                {
                                conver = conv0;
// si es el canal 0 conver no tiene offset de canal
//y = 0;
                                }
                        }
                }
        if (conv0 > 1022)
                {
                if (conv1 < 1023)
                        {
                        if (conv3 < 2)
                                {
                                conver = conv1 + 1022;
// si es el canal 1 conver tiene un offset de 1022
//y = 1022;
                                }
                        }
                }
        if (conv0 > 1022)
                {
                if (conv1 > 1022)
                        {
                        if (conv3 < 1023)
                                {
                                conver = conv3 + 2044;
// si es el canal 3 conver tiene un offset de 2044
//y = 2044;
                                }
                        }
                }
```

```
        }
//*****************************************************************************
//********************************** bcd ***************************************
//*****************************************************************************
bcd()
//Rutina que convierte un valor binario a bcd después enmascara todos los datos
//con 0x30 para obtener
//los valores en ascci para poder ser desplegados en la pantalla.
        {
        temp_2 = offset;
        temp_2 &= 0x8000;
        if (!temp_2)
                {
                bcd_num = conver + offset_temp;
                                // El número que se mostrara como temperatura será
                goto bcd_ini;   // la suma algebraica de el valor de la conversión
                }                 // con el valor de calibración
        else
                {
                if (conver == 0)
                        {
                        bcd_num = offset_temp;
                        goto bcd_ini;
                        }
                else
                        {
                        bcd_num = conver - offset_temp;
                        }
                }
bcd_ini: // algoritmo de binario a bcd
                //                |
        i = 24;  //                |
        disp0 = 0;//               V
        disp1 = 0;
        disp2 = 0;
        disp3 = 0;
        num1 = bcd_num;
        asm
```

```
        {
                BANKSEL _bcd_num+d'1'
                movf _bcd_num+d'1',w;
                BANKSEL _num2;
                movwf _num2;
        }
        num3 = 0;
loop:
        status = 0;
        asm BANKSEL _num1;
        asm rlf _num1,1;
        asm BANKSEL _num2;
        asm rlf _num2,1;
        asm BANKSEL _num3;
        asm rlf _num3,1;
        asm BANKSEL _disp0;
        asm rlf _disp0,1;
        asm BANKSEL _disp1;
        asm rlf _disp1,1;
        asm BANKSEL _disp2;
        asm rlf _disp2,1;
        asm BANKSEL _disp3;
        asm rlf _disp3,1;
        i--;
        if (!i)
                {
                goto fin_bcd;
                }
        temp_bcd = disp3;
        disp3 = disp3 + 0x03;
        if (disp3 & 0x08 == 0)
                disp3 = temp_bcd;
        else temp_bcd = disp3;
        disp3 = disp3 + 0x30;
        if (disp3 & 0x80 == 0)
                disp3 = temp_bcd;
        temp_bcd = disp2;
        disp2 = disp2 + 0x03;
```

```
        if (disp2 & 0x08 == 0)
                disp2 = temp_bcd;
        else temp_bcd = disp2;
        disp2 = disp2 + 0x30;
        if (disp2 & 0x80 == 0)
                disp2 = temp_bcd;
        temp_bcd = disp1;
        disp1 = disp1 + 0x03;
        if (disp1 & 0x08 == 0)
                disp1 = temp_bcd;
        else temp_bcd = disp1;
        disp1 = disp1 + 0x30;
        if (disp1 & 0x80 == 0)
        disp1 = temp_bcd;
        temp_bcd = disp0;
        disp0 = disp0 + 0x03;
        if (disp0 & 0x08 == 0)
                disp0 = temp_bcd;
        else temp_bcd = disp0;
        disp0 = disp0 + 0x30;
        if (disp0 & 0x80 == 0)
                disp0 = temp_bcd;
        goto loop;
fin_bcd:// el esultado es un numero bcd de 6 cifras
                // guardadas en disp2:disp0
        cod[0] = (disp0 & 0x0f) | 0x30;
        asm BANKSEL _disp0
        asm swapf _disp0;
        cod[2] = (disp0 & 0x0f) | 0x30;
        cod[3] = (disp1 & 0x0f) | 0x30;
        asm BANKSEL _disp1
        asm swapf _disp1;
        cod[4] = (disp1 & 0x0f) | 0x30;
        cod[5] = (disp2 & 0x0f) | 0x30;
        asm BANKSEL _disp2
        asm swapf _disp2;
        cod[6] = (disp2 & 0x0f) | 0x30;
for (i = 6; i > 2; i--)
```

```
        {
        temp_1 = cod[i];
        if (temp_1 == 0x30)
                {
                cod[i] = ' ';
                }
        else break;
        }
        i = 0;
        x = 0xc7;
otro_entero:
        temp_1 = x;
        lcd_addr();
        temp_1 = cod[i];
        lcd_write();
        i++;
        x--;
        if (i < 7)
        goto otro_entero;
fin_for:
}
//*************************************************************************************
//********************************** coeficientes ***********************************
//*************************************************************************************
coeficientes() //Rutina que calcula los coeficientes a0 y a1, las fórmulas son las siguientes:
{
//*********************************************************************************************************************
//*********************************************************************************************************************
// a0 = k_p * (1000/2^15) [ 1 + 2^15/(2 * t_n * 100) + t_v *(20/2^15) ] T = 1;
// para que no se pierdan los datos más pequeños se multipicará por 2^5.
// 2^5 * a0 = k_p * (1000/2^10) [ 1 + 2^15/(2 * t_n * 100) + t_v *(20/2^15) ] 1000/2^9=1.99 = 0x1
// 2^5 * a0 = k_p [ 1 + 2^15/(2 * t_n * 100) + t_v *(20/2^15) ]
//*********************************************************************************************************************
//*********************************************************************************************************************
// a1 = k_p * (1000/2^15) [ -1 + 2^15/(2 * t_n * 100) -2 t_v *(20/2^15) ] T = 1;
// para que no se pierdan los datos más pequeños se multipicará por 2^5.
// 2^5 * a1 = k_p * (1000/2^10) [ -1 + 2^15/(2 * t_n * 100) -2 t_v *(20/2^15) ] 1000/2^9=1.99 = 0x1
// 2^5 * a1 = k_p [ -1 + 2^15/(2 * t_n * 100) -2 t_v *(20/2^15) ]
```

```
//*****************************************************************************************************
//*****************************************************************************************************
        fn = 0x28f5 / t_n;
        fv = t_v / 0x19;
        a0 = 1 + fn + fv;
        multi(a0,k_p);
        a0 = productoL;
        a0 = a0/32;
        clear_bit(status,C);
        asm     rlf _fv;
        asm     rlf _fv + d'1';
        a1 = fn - fv;
        a1 = a1 - 1;
        multi(a1,k_p);
        a1 = productoL;
        neg = 0;
        if (a1 & 0x8000)
                {
                neg = 1;
                a1 = ~a1;
                a1++;
                }
        a1 = a1/32;
        if (neg)
                {
                a1 = ~a1;
                a1++;
                }
}
//*****************************************************************************************
//************************ Comunicación rs232 ***********************************
//*****************************************************************************************
hs01(char dato)//Toma el valor en dato y lo envía por el puerto rs232 (bit 6 y 7 del puerto c)
{
espera:
if (!(pir1 & 0x10))
        goto espera;
txreg = dato;
```

```
}
//*****************************************************************************
//********************************** Comunicación con el Site Player **********
//*****************************************************************************
site_player() //Esta rutina se encarga de enviar el valor de temperatura periódicamente
              //hacia el site player
              //para que sea visto remotamente (cada segundo). Además almacena una
              //tabla de 30 datos
              //tomando una muestra cada minuto.
{
        temp_2 = bcd_num;
        logger[0] = temp_2;
        hs01 (0x81);
        delay_ms(10);
        hs01 (0x00);
        delay_ms(10);
        hs01 (logger[0]);
        delay_ms(10);
        temp_2 = temp_2 >> 8;
        logger1[0] = temp_2;
        hs01 (logger1[0]);
        delay_ms(10);
        if (tiemp_log == 60)
                {
                tiemp_log = 0;
                historicos1--;
                if (!historicos1)
                        {
                        l = 30;
tabla_sp:
                        l--;
                        logger1[l+1] = logger1[l];
                        logger[l+1] = logger[l];
                        if (l)
                                goto tabla_sp;
                        l = 0;
                        m = 0;
datos_tabla:
```

```
                    l++;
                    m++;
                    m++;
                    hs01 (0x81);
                    delay_ms(10);
                    hs01 (m);
                    delay_ms(10);
                    hs01 (logger[l]);
                    delay_ms(10);
                    hs01 (logger1[l]);
                    delay_ms(10);
                    if (l < 31)
                        goto datos_tabla;
                    }
                }
                tiemp_log++;
}
```

Código necesario para el Siteplayer

El código del archivo proyecto.spd se presenta a continuación

```
;SECCIÓN DE DEFINICIÓN

;Nombre o descripción del dispositivo
$devicename "Controlador PID Digital UDB"

;Habilita o deshabilita DHCP
$DHCP off

;Contraseña para evitar que usuarios no autorizados descarguen archivos
en el servidor
$DownloadPassword "abcdef"

;Contraseña para poder navegar en las páginas del servidor
$SitePassword ""

;Dirección IP inicial
$InitialIP "10.0.17.250"

;La ruta de la imagen binaria a ser creada
$sitefile "C:\Siteplayer\proyecto.spb"

;Directorio raíz de las páginas WEB
$sitepath "C:\Siteplayer\WEB"
```

```
;Archivos a incluir durante el proceso de compilado
$Include "C:\Archivos de programa\Siteplayer\pcadef.inc"

;SECCIÓN DE OBJETOS

org 0h
$OutputOnly
TempVar dw 0
Temp1 dw 0
Temp2 dw 0
Temp3 dw 0
Temp4 dw 0
Temp5 dw 0
Temp6 dw 0
Temp7 dw 0
Temp8 dw 0
Temp9 dw 0
Temp10 dw 0
Temp11 dw 0
Temp12 dw 0
Temp13 dw 0
Temp14 dw 0
Temp15 dw 0
Temp16 dw 0
Temp17 dw 0
Temp18 dw 0
Temp19 dw 0
Temp20 dw 0
Temp21 dw 0
Temp22 dw 0
Temp23 dw 0
Temp24 dw 0
Temp25 dw 0
Temp26 dw 0
Temp27 dw 0
Temp28 dw 0
Temp29 dw 0
Temp30 dw 0

;SECCIÓN DE EXPORTACIÓN

;Exportar definiciones de variables a Visual Basic
$ExportFormatFile "C:\Archivos de programa\Siteplayer\makevb.def"
$ExportFile "C:\Siteplayer\Visualdefs.bas"
$Export

;Exporta definiciones a lenguaje C
$ExportFormatFile "C:\Archivos de programa\Siteplayer\makec.def"
$ExportFile "C:\Siteplayer\Cdefs.h"
$Export

;Exporta definiciones a asm
$ExportFormatFile "C:\Archivos de programa\Siteplayer\makeasm.def"
$ExportFile "C:\Siteplayer\ASMdefs.asm"
$Export
```

El código del archivo .html se presenta a continuación

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
var Compteur = null;
var CompteurTourne = false;

function DemarreHorloge () {
if(CompteurTourne)
clearTimeout(Compteur);
CompteurTourne = false;
AfficheTemps();
}

function AfficheTemps () {
var Temps = new Date();
var TempsLocal = Temps.getTime()+
   (Temps.getTimezoneOffset()-60)*60;
var Maintenant = new Date(TempsLocal);
var Heure = " " + Maintenant.getHours();
var minutes = Maintenant.getMinutes();
var secondes = Maintenant.getSeconds();
Heure += ((minutes < 10) ? ":0" : ":") + minutes;
Heure += ((secondes < 10) ? ":0" : ":") + secondes;
document.Horloge.FenetreHeure.value = Heure;
var AujourdHui = " " + Maintenant.getDate();
var Mois = Maintenant.getMonth()+1;
var Annee = Maintenant.getYear();
AujourdHui += "/" + Mois + "/" + Annee;
document.Horloge.FenetreDate.value = AujourdHui;
Compteur = setTimeout("AfficheTemps()",1000);
CompteurTourne = true;
}
// -->
</SCRIPT>

<meta http-equiv="refresh" content="15" >
<TITLE ALIGN=CENTER>CONTROLADOR PID CON PIC</TITLE>
<H1 ALIGN=CENTER>Estado de la temperatura en el sistema térmico<H1>
</HEAD>
<BODY onload="DemarreHorloge()">

<P ALIGN=CENTER>La temperatura actual es: ^TempVar:4 ^TempVar:3
^TempVar:2.^TempVar:1</P>

<P ALIGN=CENTER>
```

```
<FORM name="Horloge" onSubmit="0">
<INPUT type="text" name="FenetreDate" size=12 value="">
</INPUT>
<INPUT type="text" name="FenetreHeure" size=12 value="">
</INPUT>
</FORM>
</P>

<P ALIGN=CENTER ><FONT SIZE=2>Tabla de datos<BR>
Las muestras se toman cada minuto
</FONT></P>


<P ALIGN=CENTER>
<TABLE BORDER=2>
<TR>
        <TH ALIGN=CENTER>Muestra</TH>
        <TH ALIGN=CENTER>Temperatura</TH>
</TR>
<TR>
        <TD ALIGN=CENTER>1</TD>
        <TD ALIGN=CENTER>^Temp1/10</TD>
</TR>
<TR>
        <TD ALIGN=CENTER>2</TD>
        <TD ALIGN=CENTER>^Temp2/10</TD>
</TR>
<TR>
        <TD ALIGN=CENTER>3</TD>
        <TD ALIGN=CENTER>^Temp3/10</TD>
</TR>
<TR>
        <TD ALIGN=CENTER>4</TD>
        <TD ALIGN=CENTER>^Temp4/10</TD>
</TR>
<TR>
        <TD ALIGN=CENTER>5</TD>
        <TD ALIGN=CENTER>^Temp5/10</TD>
</TR>
<TR>
        <TD ALIGN=CENTER>6</TD>
        <TD ALIGN=CENTER>^Temp6/10</TD>
</TR>
<TR>
        <TD ALIGN=CENTER>7</TD>
        <TD ALIGN=CENTER>^Temp7/10</TD>
</TR>
```

```
<TR>
	<TD ALIGN=CENTER>8</TD>
	<TD ALIGN=CENTER>^Temp8/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>9</TD>
	<TD ALIGN=CENTER>^Temp9/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>10</TD>
	<TD ALIGN=CENTER>^Temp10/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>11</TD>
	<TD ALIGN=CENTER>^Temp11/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>12</TD>
	<TD ALIGN=CENTER>^Temp12/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>13</TD>
	<TD ALIGN=CENTER>^Temp13/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>14</TD>
	<TD ALIGN=CENTER>^Temp14/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>15</TD>
	<TD ALIGN=CENTER>^Temp15/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>16</TD>
	<TD ALIGN=CENTER>^Temp16/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>17</TD>
	<TD ALIGN=CENTER>^Temp17/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>18</TD>
	<TD ALIGN=CENTER>^Temp18/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>19</TD>
	<TD ALIGN=CENTER>^Temp19/10</TD>
```

```
</TR>
<TR>
	<TD ALIGN=CENTER>20</TD>
	<TD ALIGN=CENTER>^Temp20/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>21</TD>
	<TD ALIGN=CENTER>^Temp21/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>22</TD>
	<TD ALIGN=CENTER>^Temp22/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>23</TD>
	<TD ALIGN=CENTER>^Temp23/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>24</TD>
	<TD ALIGN=CENTER>^Temp24/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>25</TD>
	<TD ALIGN=CENTER>^Temp25/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>26</TD>
	<TD ALIGN=CENTER>^Temp26/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>27</TD>
	<TD ALIGN=CENTER>^Temp27/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>28</TD>
	<TD ALIGN=CENTER>^Temp28/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>29</TD>
	<TD ALIGN=CENTER>^Temp29/10</TD>
</TR>
<TR>
	<TD ALIGN=CENTER>30</TD>
	<TD ALIGN=CENTER>^Temp30/10</TD>
</TR>

</TABLE>
```

</P>
</BODY>
</HTML>

El aspecto de la página Web en el siteplayer es el siguiente:

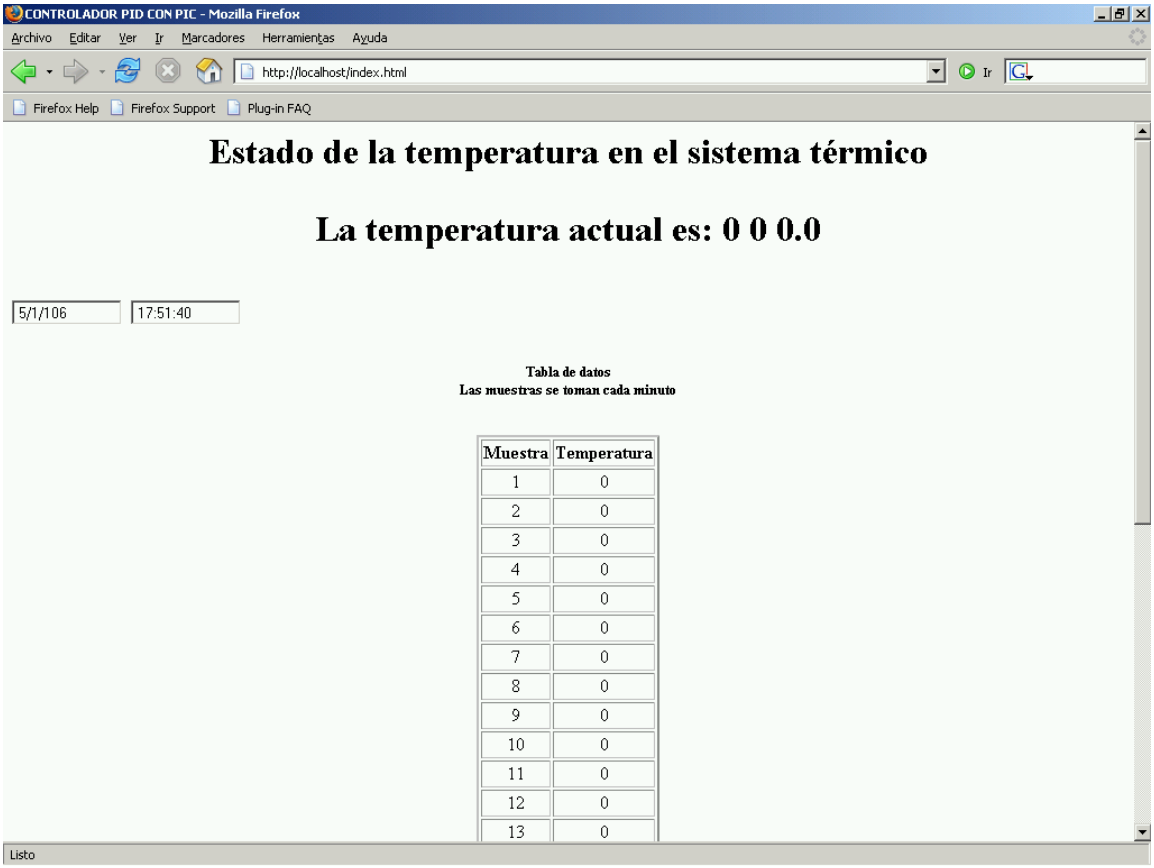


Figura 8.1 Aspecto de la página Web en el navegador Mozilla

Costos y aspectos técnicos del Controlador PID

Item	Cantidad	Descripción	Costo
1	1	Microcontrolador PIC 16F877A	$20.00
2	1	Acondicionador de termocupla AD594	$13.00
3	1	Regulador LM317	$0.55
4	1	Regulador LM7805	$0.55
5	1	Regulador LM7912	$0.55
6	1	Regulador LM7812	$0.55
7	1	74LS164	$2.50
8	1	DAC0808	$5.35

9	43	Resistencias de diversos valores ¼ W	$5.00
10	10	Diodos 1N4004	$2.50
11	17	Capacitores de diversos valores	$2.00
12	1	Módulo Siteplayer	$60.00
13	1	Filtro conector Ethernet EPJ9372	$7.50
14	1	Pantalla de Cristal líquido	$20.00
15	2	Impresos	$20.00
16	4	LM324	$5.00
17	1	Fuente de voltaje de +/- 15V	$30.00
18	1	Cable crossover	$2.00
19	1	Relé de estado sólido	$15.00
20	5	Conectores de 3 bornes	$0.60
21	8	Conectores de 2 bornes	$0.50
		TOTAL	$ 213.15

NOTA: El presupuesto aquí presentado no contempla la termocupla debido a que en los equipos comerciales sólo se vende el controlador, y la termocupla se tiene que adquirir aparte.

Los aspectos técnicos que posee el controlador son:

- Capacidad de calibración digital
- Comunicación vía Ethernet para envío de datos para monitoreo
- Capacidad de trabajar a lazo abierto
- Capacidad de auto sintonización de parámetros PID
- Capacidad para controlar equipos que tengan resistencias que trabajan a 110V ó 220V
- 5 valores de alarma programables
- Temperatura máxima 300 ºC

Puede compararse su precio y características con el CNI8C22 en Omega Internacional, consultar la página http://www.omega.com/pptst/CNI8C.html

El precio en la página antes citada: $355

Donde de las características técnicas son:

- Compacto

- Display grande con característica de cambio de color
- Protocolos de comunicación: RS-232, RS-422/485 o comunicación Modbus que puede seleccionarse mediante menú
- Ofrece una amplia cantidad de entradas que pueden ser seleccionadas mediante menú.
- Medición de voltaje o corriente del proceso
- Salida analógica
- Capacidad de control ON-OFF, PID, P, PI, PD, y auto sintonía de parámetros

En la siguiente figura se observa la forma de este controlador de temperatura PID

Figura 8.2 Controlador de temperatura CNI8C22

La hoja técnica puede verse en los documentos anexos CNI8C.pdf.

Al comparar técnicamente al prototipo, puede observarse que la interfaz con el usuario, que posee el CNI8C22 es nada más hecha a base de displays y no cuenta con un menú para hacer más fácil la interacción con el usuario, eso debe ser para poder hacerlo más compacto, las comunicaciones no consideran el uso de Ethernet en este modelo, solamente protocolos serie, donde también el prototipo construido posee el protocolo serie RS-232, con la adición de posibilidad de comunicar temperatura actual e históricos a una red de computadoras.

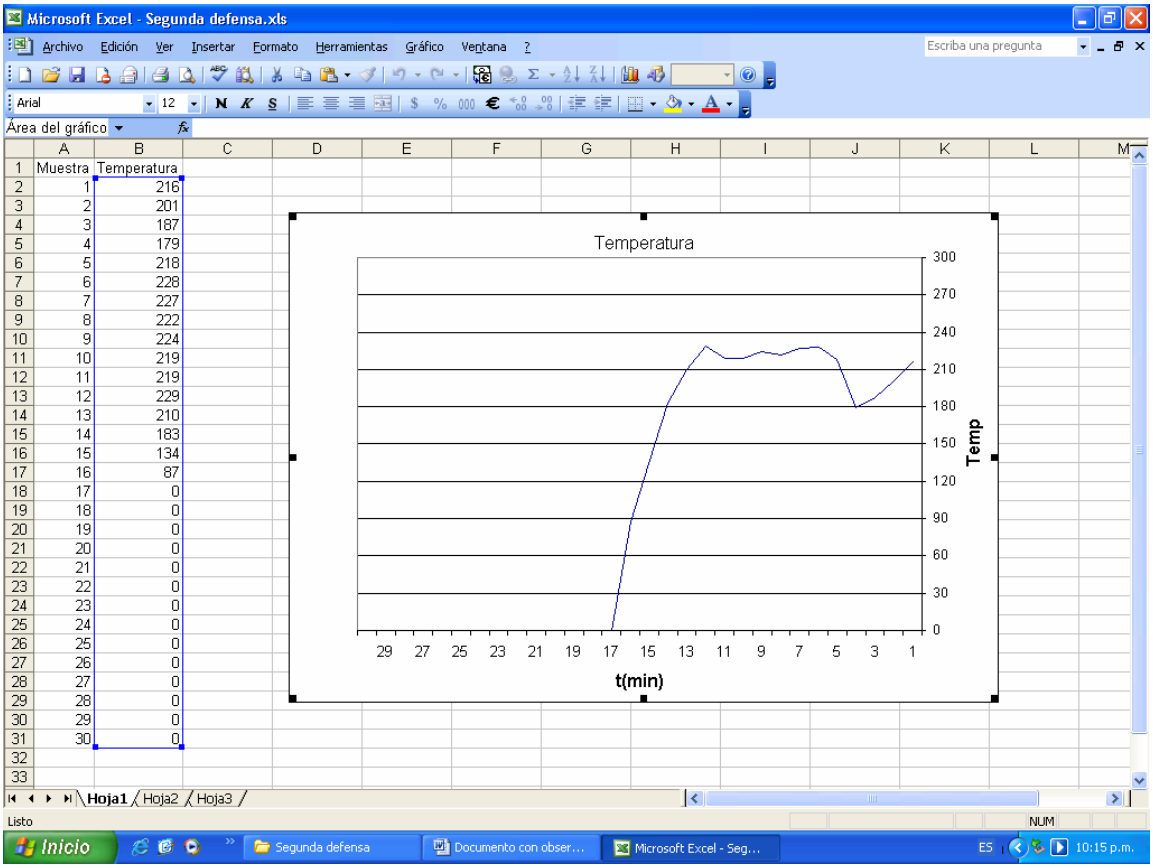
En cuanto a la capacidad de control, pueden seleccionarse las opciones que se mencionan para el CNI8C22, menos la capacidad ON-OFF.

El precio del controlador comercial es de $355, los costos del prototipo andan por debajo de esta cantidad, pero podrían rondar por la misma debido a la integración de todo el sistema en un chasis, mano de obra, costos de comercialización, etc.

Pruebas realizadas

Para dejar constancia de las pruebas realizadas, se tomaron diferentes gráficos del comportamiento del sistema por medio de la página de Excel, la cual genera un gráfico automáticamente, tomando los datos de las muestras que se encuentran en la tabla del sitio Web interno al Siteplayer.

La página de Excel tiene la siguiente apariencia:



	A	B
1	Muestra	Temperatura
2	1	216
3	2	201
4	3	187
5	4	179
6	5	218
7	6	228
8	7	227
9	8	222
10	9	224
11	10	219
12	11	219
13	12	229
14	13	210
15	14	183
16	15	134
17	16	87
18	17	0
19	18	0
20	19	0
21	20	0
22	21	0
23	22	0
24	23	0
25	24	0
26	25	0
27	26	0
28	27	0
29	28	0
30	29	0
31	30	0

Las siguientes páginas muestran los resultados de algunas pruebas con diferentes valores de parámetros.

Grafico a lazo abierto de temperatura ambiente a 45º de referencia.

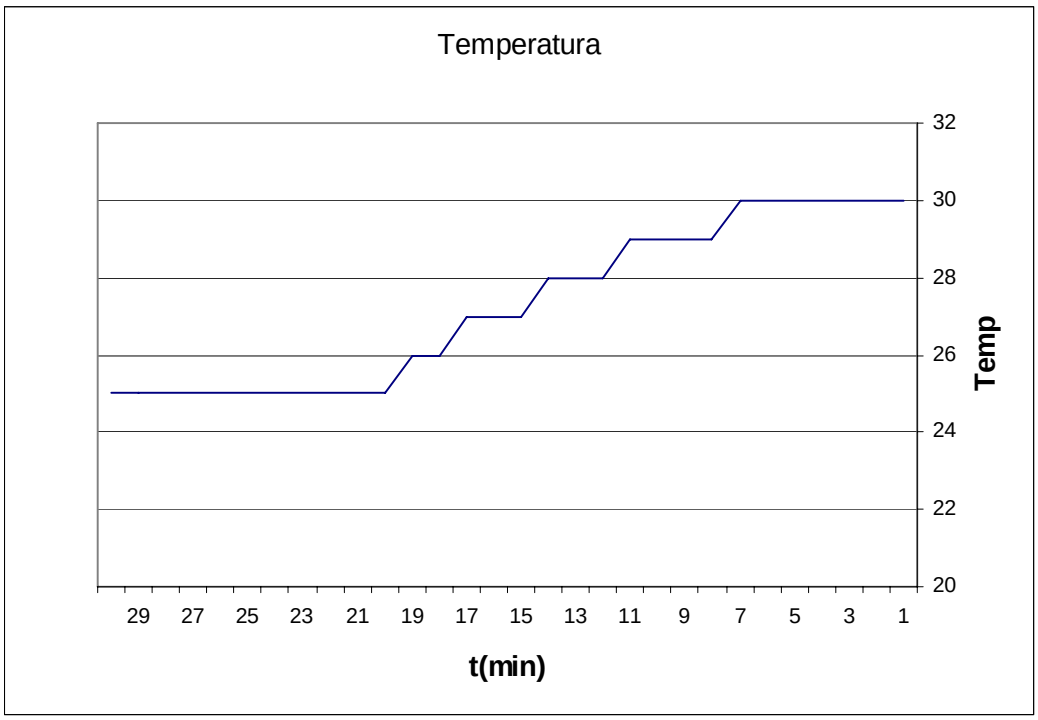
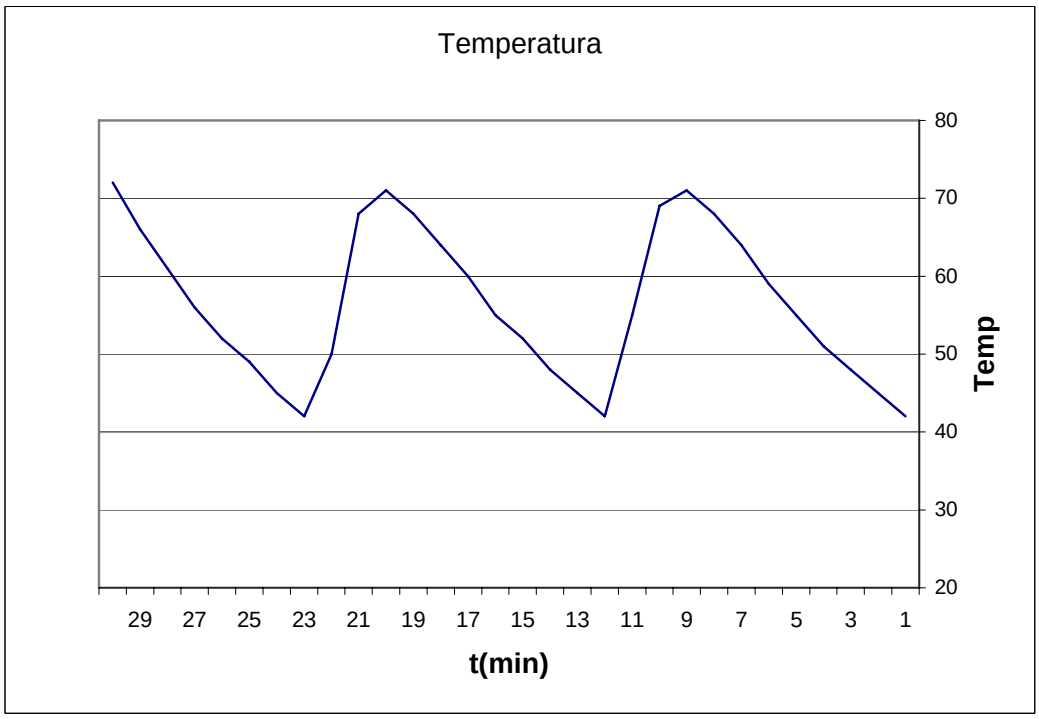
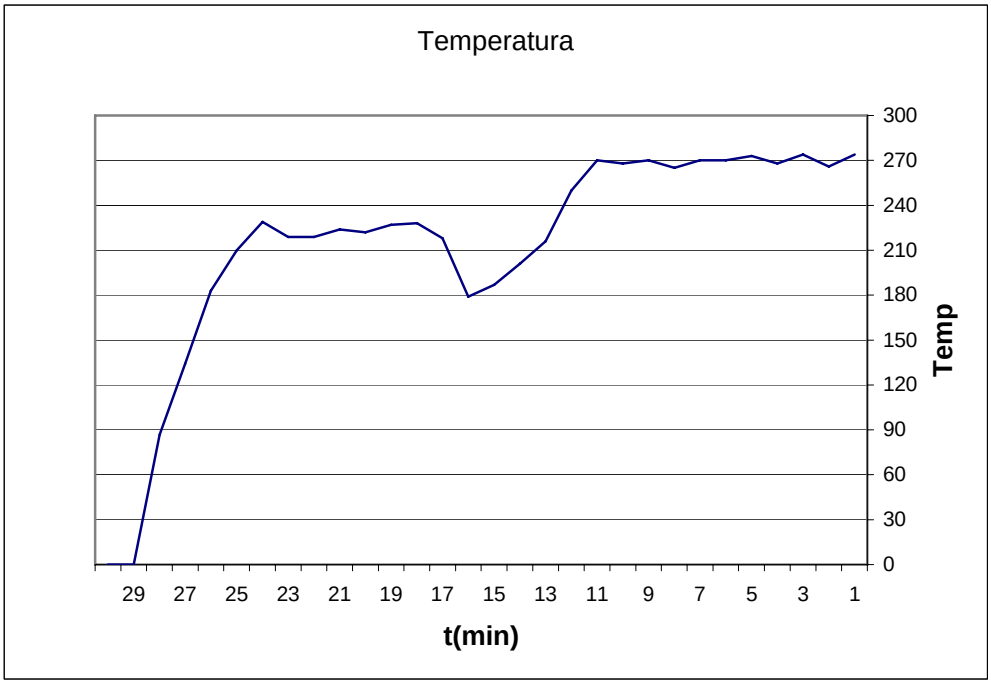


Gráfico a lazo cerrado Kp = 2 Tn = 0.5



Referencia a 230 y a 270 ºC (Kp = 32; Tv = 163.8; Tn = 10)



Referencia 270 a 290 ºC (Kp = 32; Tv = 163.8; Tn = 10)