UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

FALLA DE ORIGEN

CONTROL DE SISTEMAS CON TIEMPO MUERTO LARGO

T E S I S P R O F E S I O N A L

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N :

I G N A C I O A R C E L U S D E D I E G O

E R I C L O P E Z G O M E Z

DIRECTOR: DR. YU TANG XU

MEXICO, D. F. 1995

Damos gracias a Dios por habernos provisto del espíritu necesario para llevar a cabo nuestros estudios universitarios,

a la Universidad Nacional Autónoma de México por proporcionarnos la oportunidad de educarnos para ayudar a fincar una patria llena de esperanza

y especialmente al Dr. Yu Tang Xu por su inagotable paciencia y firme apoyo.

Dedicamos este trabajo a:

mis padres y
hermanos,
la abuela,
mis amigos
y Liz

a todos por su amor.

Iñaki

Marcelo y Maru por su gran cariño y compañía.

Eric

Introducción

Los retardos ocurren frecuentemente en los sistemas químicos, biológicos, mecánicos y electrónicos, y están asociados invariablemente a los tiempos de transporte de masa o de energía. Como ejemplo tenemos los procesos químicos que tratan con fluídos, las hormonas dentro del flujo sanguíneo, las ondas sísmicas, la radiación electromagnética en el espacio o los tiempos de cómputo (necesarios para hacer análisis de composición química, procesamiento de una imagen visual, evaluación de la salida de un algoritmo de control digital, etc.)[1].

A pesar de los nuevos algoritmos de control predictivo -como el predictor de Smith- en la industria se sigue utilizando en la mayoría de aplicaciones con tiempo muerto el control PID. Esto se debe en primer lugar a que es muy robusto, y en segundo y más importante, a que los operadores están habituados a él y lo pueden sintonizar de manera sencilla en base a reglas empíricas.

Hoy, la investigación ha dejado ya treinta años atrás a los controladores PID típicos, ocupándose incluso de obtener algoritmos que sintonizan de forma automática sus ganancias y reducen la complejidad de su uso en aras de una mayor aceptación industrial.

En este trabajo presentamos un par de implementaciones de algoritmos de control en un microcontrolador de bajo costo para un proceso térmico típico y son analizados los resultados comparativos entre ellos.

Esta tesis está organizada de la siguiente manera: este capítulo presenta el problema de forma general y la manera en que se piensa solucionar, así como la descripción de los demás capítulos y las convenciones y políticas seguidas a lo largo del trabajo de investigación y la escritura.

El segundo capítulo explica de manera más profunda el problema a tratar, y plantea los alcances de la investigación. El tercero describe de manera general los métodos de solución posibles, desarrolla específicamente el método escogido y explica las razones de su selección. La aplicación de éste al problema particular que nos atañe y a las herramientas con las que contamos está contenida en el capítulo cuatro.

Finalmente los resultados son presentados apoyados en gráficas en la quinta sección y se discuten de acuerdo a lo esperado en la sexta para poder escribir las conclusiones en la séptima.

El trabajo cuenta con un par de apéndices en los que están incluídos el código fuente de todos los programas utilizados para la implantación, entre ellos las subrutinas de manejo para números de doble precisión, y los circuitos de interfaz.

ÍNDICE

1. DEFINICIÓN DEL PROBLEMA **5**

1.1 SISTEMAS CON TIEMPO MUERTO LARGO 5
1.2 CONTROL DE SISTEMAS CON TIEMPO MUERTO LARGO 7
1.3 ALCANCES DE ESTE TRABAJO 8

2. MÉTODO **9**

2.1 CONTROLADORES PROPORCIONAL-INTEGRALES 9
2.2 CONTROLADORES PREDICTIVOS 10
2.3 CONTROLADORES PI CON PREDICCION ADAPTABLE 12

3. APLICACIÓN DEL MÉTODO **15**

3.1 CARACTERÍSTICAS DE LA PLANTA 15
3.2 PARTICULARIZACION 17
3.3 CARACTERÍSTICAS DEL CONTROLADOR 19
3.4 CIRCUITERÍA 21
3.5 PROGRAMACIÓN 21
3.5.1 Filosofía de programación 21
3.5.2 Interrupciones 24

4. RESULTADOS **27**

4.1 CONDICIONES DE LOS EXPERIMIENTOS 27
4.2 GRÁFICAS 28

5. DISCUSIÓN DE RESULTADOS **39**

6. CONCLUSIONES **41**

BIBLIOGRAFÍA **43**

APÉNDICE A **A-1**

A.1 PROGRAMAS PARA EL HC11 ..... A-1
A.2 PROGRAMAS PARA SIMNON A-30

APÉNDICE B **B-1**

1. Definición del Problema

1.1 Sistemas con tiempo muerto largo

Muchos de los sistemas físicos pueden describirse con base en sus retrasos y en su ganancia estática dados por el sistema en su respuesta al escalón unitario.

El tiempo muerto aparente del sistema se compone por tres tipos de retrasos que resulta complejo describir de forma matemática, sin embargo es fácil aproximarlo haciendo combinaciones adecuadas de ellos:

- el primer tipo es el intervalo después de la aplicación de un fuerza durante el cual no se observa respuesta alguna. Esta característica no depende de la naturaleza de la fuerza aplicada y ocurre en el transporte de masa o energía a lo largo de un ruta particular. El retraso dependerá de la longitud de la ruta y la velocidad del traslado. Responde a varios nombres: tiempo muerto, retraso puro o de distancia-velocidad[2];
- para el segundo tipo la respuesta comienza inmediatamente y alcanza un punto estable cuando el tiempo se hace infinito. Cuando dicha respuesta alcanza un 63% del valor final se dice que el tiempo transcurrido desde la aplicación de la entrada es la constante de tiempo. Recibe el nombre de retraso de primer orden, exponencial, de RC, lineal simple, etc.;
- en el último caso nunca se alcanza un valor final y por ello la constante de tiempo se considera infinita. Este tipo no se presenta en la realidad pero es considerado normalmente en los modelos[3].

Para un sistema estable, la ganacia estática es el valor que alcanza la respuesta al escalón unitario en estado estable cuando el tiempo tiende a infinito, de forma práctica se determina cuando ha transcurrido un tiempo igual a cuatro constantes de tiempo, en ese momento la salida tiene una diferencia del 2% con respecto dicho valor final.

Un sistema con retrasos de los dos primeros tipos se puede describir por la función de transferencia:

$$P(s) = \frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1} e^{-Ls} \qquad \textbf{(1.1)}$$

donde:

Y(s)	Salida del sistema
U(s)	Entrada escalón unitario
K	Ganancia estática
τ	Constante de tiempo
L	Tiempo muerto

y su respuesta en malla abierta se muestra en la Ilustración 1.1:

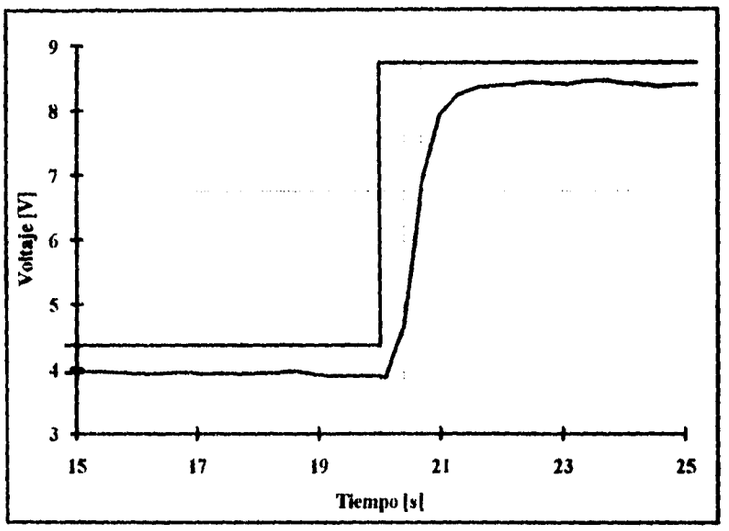


Ilustración 1.1. Sistema con tiempo muerto largo

Para manejos de control digital, es necesario discretizar la función de transferencia propuesta en la ecuación (1.1) utilizando un retén de orden cero:

$$P(z) = \frac{Y(z)}{U(z)} = \frac{bz^{-1}}{1+az^{-1}} z^{-d} \qquad (1.2)$$

siendo:

$$a = -e^{-h/\tau} \qquad (1.3)$$

$$b = \frac{K}{\tau}(1 - e^{-h/\tau}) \qquad (1.4)$$

$$d = L/h \qquad (1.5)$$

donde:

h	Tiempo de muestreo
L	Tiempo muerto

Cuando el tiempo muerto L es mayor que la constante de tiempo τ dominante en el proceso se dice que el sistema tiene tiempo muerto largo.

1.2 Control de sistemas con tiempo muerto largo

Un controlador realimentado aplica su acción correctiva basado en la observación de la salida presente, de esta manera la acción correctiva se modera mediante su efecto observable en el proceso; es por ello que al trabajar con un proceso con tiempo muerto largo la situación de control se complica, reflejándose en tiempos de asentamiento y sobretiros no deseables[4]. Además, se ha encontrado que al introducir un retardo puro a un sistema bien sintonizado, las ganancias del controlador deben disminuir para mantener la estabilidad[1]. Es por ello que para poder controlar estos sistemas eficientemente es necesario contar con un elemento predictivo[5].

En la industria el controlador tipo PID se encuentra presente en la mayor parte de las aplicaciones, incluyendo los sistemas con tiempo muerto largo; en él, la parte derivativa se puede interpretar como un mecanismo de predicción. Por desgracia la predicción a través de la derivación no es apropiada cuando el

proceso contiene tiempo muerto largo. Por lo tanto, al aplicar un control PID en este tipo de problemas la parte derivativa es prácticamente desactivada y se utiliza solamente un control PI[5].

A continuación se presenta un ejemplo de un proceso con tiempo muerto largo controlado mediante un controlador PI.

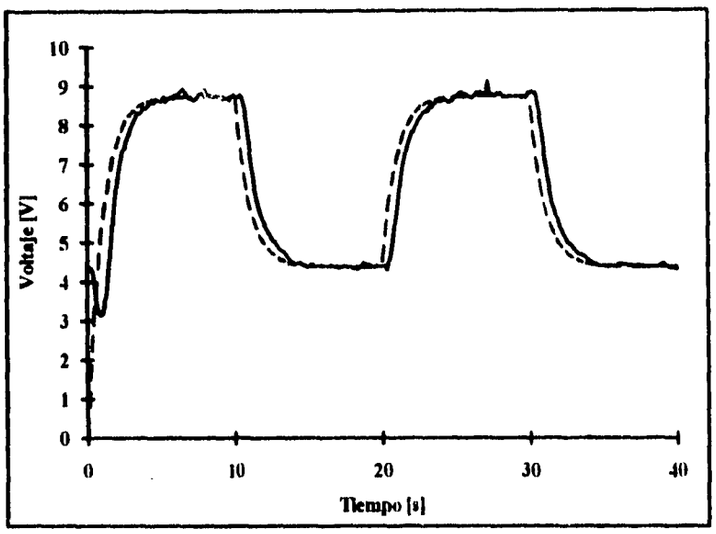


Ilustración 1.2. Control PI de un sistema con tiempo muerto largo.

Como puede verse, existe un retraso marcado entre la salida real y la referencia desdeada, éste se debe a la presencia del tiempo muerto largo en la dinámica del sistema.

1.3 Alcances de este trabajo

Existen diversas aproximaciones de solución al problema planteado, nuestro objetivo es implantar un controlador tipo PI Predictivo y otro tipo PI con Predicción Adaptable en un proceso con las características expuestas, utilizando para ello un microcontrolador de bajo costo, factible de ser usado en la industria.

2. Método

A continuación revisaremos los conceptos sobre los cuales se basará el desarrollo de la tesis. Se sigue una secuencia que comienza con las ecuaciones de los controladores PI estándar, pasa por el análisis de la predicción y lo aprovecha para explicar la forma en la que se puede adaptar.

2.1 Controladores Proporcional-Integrales

Es un controlador bien conocido y muy robusto en su comportamiento. Para el caso digital, utilizando la aproximación de Euler en la integración y con base en las señales de referencia y de salida calculamos la señal de control de acuerdo con:

$$u(t) = K_p e(t) + K_i e_i(t)$$

donde:

- u Señal de control
- e Señal de error
  $e(t) = r(t) - y(t)$
- e_i Error integral
  $e_i(t) = \frac{1}{h}\sum_{\tau=0}^{t} e(\tau)$
- K_p Constante de proporcionalidad
- K_i Constante de integración
- r Señal de referencia
- y Señal de salida
- h Período de muestreo

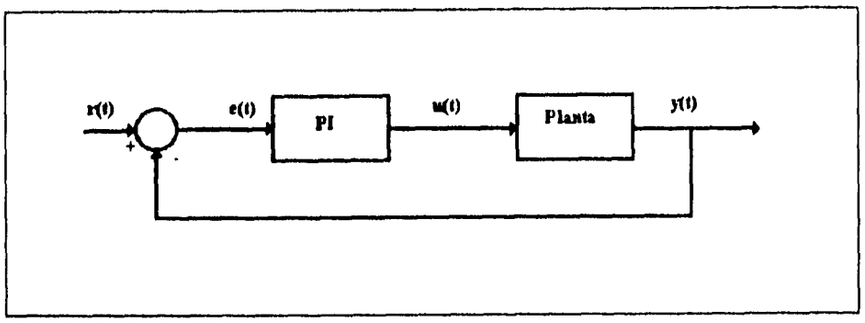


Ilustración 2.1 Controlador PI

Cuando se trata de una planta con retraso, necesitamos dar al sistema una señal de control adelantada de tal forma que:

$$u(t+d) = K_p e(t+d) + K_i e_i(t+d) \qquad (2.1)$$

donde:

d	Número de pasos de adelanto
l	Tiempo Muerto

Si analizamos

$$e(t+d) = r(t+d) - y(t+d),$$

podremos ver que necesitamos tanto de la referencia como de la salida adelantadas. Es una suposición adecuada el pensar que conocemos la referencia en todo momento. Nos queda pues, la interrogante de cómo obtener dicha salida adelantada; para ello recurriremos a las técnicas predictivas.

2.2 Controladores Predictivos

Si se conoce el modelo del sistema de interés, es posible construir predictores adecuados a partir de éste. De hecho, cuando el modelo es lineal y de dimensiones finitas, dicho predictor se puede obtener por medio de manipulaciones algebráicas sencillas del modelo[6].

De esta forma comenzaremos a efectuar dichas manipulaciones con la siguiente expresión correspondiente a la ecuación de transferencia discreta de un sistema lineal de una sola entrada y una sola salida:

$$H(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{Y(z^{-1})}{U(z^{-1})} \qquad (2.2)$$

siendo para el caso de un sistema con tiempo muerto:

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n} \qquad (2.3)$$

$$B(z^{-1}) = z^{-d}(b_1 z^{-1} + \ldots + b_m z^{-m}) = z^{-d} B'(z^{-1}) \qquad (2.4)$$

La salida del sistema en el tiempo $t+d$ se puede expresar mediante la siguiente forma predictiva:

$$y(t+d) = \alpha(z^{-1}) y(t) + \beta(z^{-1}) u(t) \qquad (2.5)$$

donde:

$$\alpha(z^{-1}) = G(z^{-1}) = \alpha_0 + \alpha_1 z^{-1} + \ldots + \alpha_{n-1} z^{-(n-1)}$$
$$\beta(z^{-1}) = F(z^{-1}) B(z^{-1}) = \beta_1 z^{-1} + \ldots + \beta_{m+d-1} z^{-(m+d-1)} \qquad (2.6)$$

y $F(z^{-1})$ y $G(z^{-1})$ son los únicos polinomios que satisfacen la ecuación de Bezout:

$$1 = F(z^{-1}) A(z^{-1}) + z^{-d} G(z^{-1}) \qquad (2.7)$$

$$F(z^{-1}) = 1 + f_1 z^{-1} + \ldots + f_{d-1} z^{-(d-1)}$$

$$G(z^{-1}) = g_0 + g_1 z^{-1} + \ldots + g_{n-1} z^{-(n-1)} \qquad (2.8)$$

Los coeficientes de $F(z^{-1})$ y $G(z^{-1})$ se pueden calcular a partir de (2.7) conociendo las características del sistema de interés.

Podemos reescribir la ecuación (2.5) de una manera más familiar:

$$y(t+d) = \phi(t)^T \theta_0 \qquad \textbf{(2.9)}$$

donde:

$$\phi(t)^T = [y(t),\ldots,y(t-n+1),u(t-1),\ldots,u(t-m-d+1)] \qquad \textbf{(2.10)}$$

$$\theta_0^T = [\alpha_0,..\alpha_{n-1},\beta_1,\ldots,\beta_{m+d-1}] \qquad \textbf{(2.11)}$$

Por último, a partir de la ecuación (2.9) podemos escribir la forma final del predictor:

$$\bar{y}(t+d,\theta) = \phi(t)^T \theta \qquad \textbf{(2.12)}$$

donde:

θ	Vector de parámetros estimados
$\bar{y}$	Salida predicha

Es fácil ver que cuando $\theta=\theta_0$, $\bar{y}(t+d,\theta) = y(t+d)$ para toda t. Esto implica que para el correcto funcionamiento del predictor es necesario conocer los parámetros del sistema. Si se da el caso de que los parámetros de la planta cambien en el tiempo, la predicción hecha no será una buena aproximación y el controlador basado en esta información no se comportará de manera óptima[7].

2.3 Controladores PI con Predicción Adaptable

El control adaptable surge a principios de los años 50 en respuesta a la necesidad de controlar sistemas cuyas características variaban en el tiempo debido al cambio del medio (en principio, pilotos automáticos para aviones). A partir de entonces fueron desarrolladas diversas aproximaciones para resolver este problema. Entre ellas existen cuatro que son las más socorridas: la programación de ganancia con base exclusivamente en mediciones auxiliares que no son la salida del sistema, lo que viene a ser una adaptación en malla abierta; la adaptación a partir de un modelo de referencia (MRAS, métodos

directos o autosintonización implícita), que puede ser de forma serie o paralela: los reguladores autosintonizables (STR, métodos indirectos o autosintonización explícita) -todos ellos son controladores determinísticos-, y finalmente los esquemas de tipo estocástico[8].

En nuestro caso, la adaptación se lleva a cabo sobre la parte predictiva del controlador. Es decir: queremos un controlador predictivo que sea capaz de adaptarse a las variaciones en los parámetros reales de la planta. Ésto se ve más claro en el diagrama de bloques contenido en la Ilustración 2.2. De él vale la pena observar que la adaptación no pretende llevar a un conocimiento exacto de los parámetros de la planta, sino a la similitud entre la salida estimada y la real.

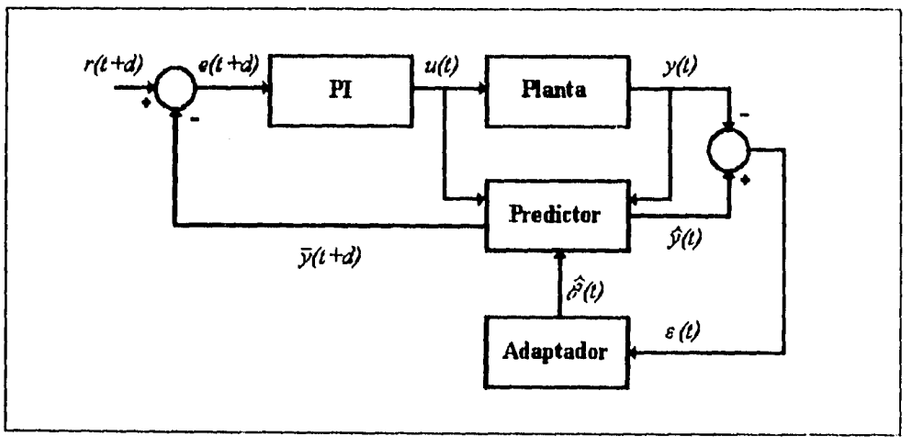


Ilustración 2.2. Control PI con Predicción Adaptable

El problema clave es obtener un mecanismo de ajuste que anule el error de estimación

$$\bar{e}(t) = \bar{y}(t) - y(t), \qquad (2.13)$$

ésto se hace utilizando el concepto de gradiente[9].

Considerando ahora el vector estimado de parámetros función del tiempo, la ecuación (2.12) pasa a:

$$\bar{y}(t+d) = \varphi^{T}(t)\hat{\theta}(t) \qquad \textbf{(2.14)}$$

La salida presente estimada -necesaria en la ecuación (2.13)- se calcula con base en la ecuación (2.14)

$$\bar{y}(t) = \varphi^{T}(t-d)\hat{\theta}(t-d), \qquad \textbf{(2.15)}$$

donde:

$\hat{\theta}$ Vector estimado de parámetros
$\bar{y}$ Salida estimada

Dado que el error en los parámetros estimados tiende asintóticamente a cero, nos es conveniente utilizar en la ecuación (2.15) los más recientes, por lo que ahora

$$\hat{y}(t) = \varphi^{T}(t-d)\hat{\theta}(t), \qquad \textbf{(2.16)}$$

haciendo una nueva definición del error de estimación a partir de (2.13) obtenemos:

$$\varepsilon(t) = \hat{y}(t) - y(t) \qquad \textbf{(2.17)}$$

Finalmente logramos obtener los parámetros estimados con ayuda del algoritmo de proyección[10]:

$$\hat{\theta}(t+1) = \hat{\theta}(t) - \gamma \frac{\varphi(t-d)\varepsilon(t)}{1+\varphi^{T}(t-d)\varphi(t-d)} \qquad \textbf{(2.18)}$$

donde:

γ Razón de adaptación ($0<\gamma<2$)

3. Aplicación del método

En este capítulo se explica detalladamente el equipo que utilizamos como medio para implementar los algoritmos de control. Dado que la programación se hace en lenguaje ensamblador, se encontró adecuado seguir una serie de políticas que nos permitieran hacer programas estructurados y modulares; dichas convenciones son también mostradas aquí. Por último dentro de los alcances de esta sección está el describir la particularización del método expuesto en el capítulo anterior para ser adaptado tanto al problema como al equipo con que contamos para resolverlo.

A grandes rasgos el sistema está dividido en tres bloques, agrupados bajo el siguiente esquema:

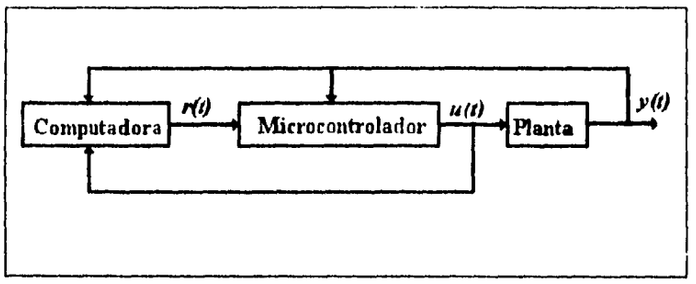


Ilustración 3.1. Esquema general del sistema

El bloque marcado como computadora es en realidad auxiliar, pues sólo proporciona la referencia y adquiere los datos de los experimentos.

3.1 Características de la planta

El equipo utilizado es un proceso térmico PT326 fabricado por Feedback Instruments Ltd., tiene las características básicas de una planta grande como son: un retardo en el sistema, comportamiento no lineal en algunos rangos,y una respuesta no muy rápida.

El proceso consiste en hacer circular aire a temperatura ambiente a través de una rejilla que se calienta y finalmente por un tubo de plástico provocando que el aire alcance una temperatura deseada. Además cuenta con un controlador

tipo proporcional, otro tipo encendido-apagado que se pueden o no conectar y conexiones para poder hacer control externo.

A la entrada del ventilador se tiene una garganta que se puede variar de forma angular de modo que se modifique la dinámica de la planta, se maneja de forma manual. Como actuador para calentar la rejilla se tiene un tiristor que se encarga de la etapa de potencia y como sensor se tiene un termistor que mide la temperatura del aire y proporciona como salida una señal de voltaje.

El intercambiador de calor presenta algunas no linealidades debidas al actuador y al sensor, que se manifiestan como variaciones de la ganancia estática del proceso en diferentes rangos de operación.

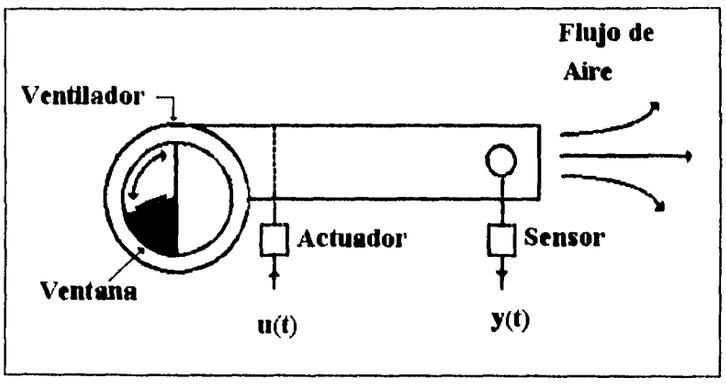


Ilustración 3.2. Proceso térmico

A pesar de lo anterior, pretenderemos trabajar en un intervalo relativamente lineal, que se ubica de forma más o menos central dentro de los límites físicos del sistema: por un lado la temperatura ambiente y por el otro la capacidad de calentamiento del flujo de aire[11].

Especificaciones		
Rango de voltaje de la señal de control	0-10	V
Rango de voltaje de la señal de salida	0-14.55	V
Resistencia mínima de carga en cualquier salida	5	kΩ
Rango de medición de temperatura	0-80	°C
Rango de control de temperatura	30-60	°C
Potencia del calentador	80	W
Rango de velocidad del aire	0.3-3	m/s

Tabla 3.1. Especificaciones del Proceso Térmico

De acuerdo con la respuesta en malla abierta de este sistema, que se puede ver en la Ilustración 1.1 podemos modelar la planta como un sistema de primer orden con tiempo muerto largo.

3.2 Particularización

Para poder aplicar el método expuesto en el capítulo anterior, es necesario referirlo a las características particulares de nuestro proceso, así podemos reescribir la ecuación (2.2) como:

$$PT(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} z^{-d} \qquad (3.1)$$

y definir dichos polinomios así:

$$B(z^{-1}) = bz^{-1} \qquad (3.2)$$

$$A(z^{-1}) = 1 + az^{-1} \qquad (3.3)$$

de donde:

$n = deg(A) = 1$

$m = deg(B) = 1$

Los valores de los coeficientes a y b se obtienen a partir de mediciones de la respuesta de malla abierta del sistema utilizando tanto la ecuación (1.3) como la ecuación (1.4).

Aplicando las restricciones marcadas por los valores de n y m a las ecuaciones (2.8), (2.10) y (2.11) llegamos a:

$$F(z^{-1}) = 1 + f_1 z^{-1} + \ldots + f_{d-1} z^{-(d-1)} \qquad (3.4)$$

$$G(z^{-1}) = g_0 \qquad (3.5)$$

$$\phi(t)^T = [y(t), u(t-1), \ldots, u(t-d)] \qquad (3.6)$$

$$\theta_0^T = [\alpha_0, \beta_1, \ldots, \beta_d] \qquad (3.7)$$

Substituyendo las ecuaciones (3.3), (3.4) y (3.5) en la de Bezout (2.7) obtenemos:

$$1 = (1 + az^{-1})(1 + f_1 z^{-1} + \ldots + f_{d-1} z^{-d+1}) + z^{-d} g_0 \qquad (3.8)$$

y de ella las siguientes igualdades y soluciones:

Igualdades	Soluciones
$1=1$	
$a + f_1 = 0$	$f_1 = -a$
$a f_1 + f_2 = 0$	$f_2 = -a f_1 = (-a)^2$
.	.
.	.
.	.
$a f_{d-2} + f_{d-1} = 0$	$f_{d-1} = -a f_{d-2} = (-a)^{d-1}$
$a f_{d-1} + g_0 = 0$	$g_0 = (-a)^d$

Substituyendo esto en la ecuación (2.6) llegamos a la definición final de:

$$\alpha(z^{-1}) = (-a)^d$$
$$\beta(z^{-1}) = bz^{-1} + b(-a)z^{-2} + \ldots + b(-a)^{d-1} z^{-d} \qquad (3.9)$$

Por otro lado, utilizando los valores ya conocidos de n y m para actualizar la ecuación (2.10) podemos conocer el número de datos del controlador y la salida que será necesario almacenar en memoria para la parte predictiva:

$$\phi^T(t) = [u(t-1), \ldots, u(t-d), y(t)] \quad \textbf{(3.10)}$$

$$\phi^T(t-d) = [u(t-d-1), \ldots, u(t-2d), y(t-d)] \quad \textbf{(3.11)}$$

3.3 Características del Controlador

Es un controlador digital basado en la tarjeta de evaluación M68HC11EVB de Motorola. Dicha tarjeta es un medio económico para depurar código del programador y evaluar sistemas incorporados al microcontrolador M68HC11, tiene la capacidad para conectarse por medio de un puerto de entrada-salida compatible con RS-232C con una computadora de la cual puede recibir código gracias a un ACIA (*Asynchronous Communications Interface Adapter*). Tiene una unidad de reemplazo de puertos, que le permite al usuario ocupar los puertos que el microcontrolador usualmente destina al direccionamiento de memoria. Además, posee -a diferencia de su predecesora inmediata, la EVBU- una memoria RAM de 8 kB para el usuario. Sus características de operación se muestran en la Tabla 3.2, a partir de ellas puede verse que nuestra tarjeta es capaz de ser utilizada sin mayores dificultades en las áreas industriales típicas.

Temperatura de almacenamiento	-40 a 85 °C
Temperatura de operación	0 a 50 °C
Humedad relativa	0 a 90 %
Alimentación	+5 Vdc @ 500 mA +12 Vdc @ 100 mA -12 Vdc @ 100 mA
Dimensiones Ancho Largo	 17.80 cm 11.75 cm

Tabla 3.2. Características de la EVB

La comunicación entre el usuario y el circuito integrado se lleva a cabo mediante un programa monitor/depurador que Motorola incluye en la tarjeta de evaluación: el BUFFALO (*Bit User Fast Friendly Aid to Logical Operations*).

Por otro lado, para compilar el código en lenguaje ensamblador y comunicarnos de manera más amigable con el microcontrolador, en la computadora utilizamos el programa IASM11.

El microcontrolador utilizado, el M68HC11A1, es un microprocesador con memoria EEPROM interna, con un conjunto de puertos de entrada-salida y ocho canales de conversión A/D.

Debemos cubrir ciertas necesidades de comunicación entre los bloques del sistema. Para ello hemos administrado los recursos de la siguiente forma:

- La señal de referencia *r(t)* es generada por la tarjeta de adquisición de datos RTI800 de Analog Devices con ayuda del programa Simnon; entra al microprocesador por uno de los canales del puerto E que contiene el convertidor A/D.
- La señal de salida *y(t)* es generada por el sensor de temperatura y escalada analógicamente antes de llegar a otro de los canales del puerto E.
- La señal de control *u(t)* es generada por el microprocesador y colocada en el puerto B, pasa por un convertidor D/A y alimenta tanto a la planta como a la tarjeta de adquisición de datos.
- El puerto B se utilizó a lo largo del desarrollo como salida digital para monitorear variables, como el error integral o la salida predicha por ejemplo.

El microcontrolador trabaja siempre en el modo expandido multiplexado, sin embargo la tarjeta tiene la capacidad de trabajar en dos modos de operación: el primero es el de depuración, en el que bajo el control del programa monitor recibe programas compilados en archivos tipo S y los ejecuta para su corrección; el segundo es el de evaluación o emulación, en el que la tarjeta emula trabajar en el modo de circuito integrado único y permite evaluar al sistema en su ambiente real de trabajo utilizando la memoria interna del microcontrolador[12].

3.4 Circuitería

Consiste básicamente en la interfaz entre la tarjeta de evaluación y el mundo real bajo las siguientes consideraciones:

- El convertidor Analógico/Digital del microcontrolador tiene como entrada un rango de 0 a 5 V.
- La salida del Proceso Térmico está entre 0 y 14.55 V
- El microcontrolador no tiene salida analógica.
- La entrada al actuador del Proceso térmico es analógica.

Las dos primeras nos llevan a construir un atenuador de señal, más específicamente un amplificador con ganancia de 0.3436. Las otras dos apuntan al uso de un convertidor Digital/Analógico. La implantación de ambos circuitos puede observarse en el apéndice B. El suministro de energía se llevó a cabo mediante fuentes de voltaje de 5, 12 y -12 V para la tarjeta y 15 V para la interfaz. Todo esto se realizó en un solo circuito impreso.

La selección del amplificador operacional utilizado es en base a su capacidad de trabajar con una sola fuente de alimentación, que se especificó de +15 V por ser el mínimo necesario para poder manejar todo el rango de voltaje de salida del PT. Por otro lado, el convertidor D/A fue elegido debido a que está diseñado para trabajar con microprocesadores y por ello exige poca corriente a la fuente de información digital.

3.5 Programación

Todos los programas se encuentra en el apéndice A, en él se explican con detalle los algoritmos que están detrás de todos ellos.

Utilizamos dos lenguajes de programación: el ensamblador del HC11 y el intérprete de comandos de simmon.

3.5.1 Filosofía de programación

Al programar seguimos un esquema modular y estructurado. La programación modular favorece el ahorro de código y la simplicidad en los programas que los utilizan, la programación estructurada ayuda a la claridad y al futuro mantenimiento y mejora de los sistemas. Para poder lograr ambas metas en lenguaje ensamblador fue necesario asumir determinadas convenciones.

Cada módulo toma los datos de la pila o la memoria y expresa sus resultados a través de esos mismos medios.

El módulo principal -en el archivo PRINCIPA.ALL- inicializa el sistema y llama mediante interrupciones al programa de control que se elija, que puede ser Proporcional Integral (PI.ALL), PI Predictivo (PIP.ALL) o PI Predictivo Adaptable (PIPA.ALL). La operaciones básicas y subrutinas de uso frecuente están juntas en el archivo OPERA.ALL.

Los programas hechos en Simmon -una planta y una Macro- definen tanto el uso de la tarjeta de adquisición de datos (entradas y salidas) como el comportamiento de la referencia y su predicción en amplitud, corrimiento de directa, frecuencia y atraso. Además, son el lugar donde se calcula el error y se llevan a cabo las acciones necesarias para tener gráficas en una sola escala relacionada con los valores reales y cumplir con las restricciones de voltaje de los otros bloques del sistema.

Al no estar definidas éstas en el lenguaje del HC11, las realizamos bajo los esquemas expuestos en la Tabla utilizando las siguientes convenciones:

Brinco	Uso de cualquiera de las instrucciones de brinco: BGT, BLE, BLS, BHI, BCLR, BRSET, BCC, BCS, BEQ, BNE, BPL, BMI, BVC, BVS, BGE, BLT
Brinco no condicionado	BRA
Operación	Operación en el HC11 que genera el cambio de bandera que brinco necesita para tomar decisión.
Negado	Se utiliza en las operaciones lógica negada con respecto a las originales, ej: Si la condición original es a=b, escribiríamos a≠b.
Etiqueta X	X es el texto de la etiqueta en el código fuente.

IF THEN *Procedimiento* END	Operación Brinco negado ETIQUETA 0 *Procedimiento* ETIQUETA 0
IF THEN *Procedimiento 1* ELSE *Procedimiento 2* END	Operación Brinco ETIQUETA 1 BRA ETIQUETA 2 ETIQUETA 1 *Procedimiento 1* BRA ETIQUETA FIN ETIQUETA 2 *Procedimiento 2* ETIQUETA FIN
CASE *Procedimiento 1* THEN END *Procedimiento 2* THEN END ... *Procedimiento n* THEN END	Operación Brinco negado ETIQUETA 1 *Procedimiento 1* BRA ETIQUETA FIN ETIQUETA 1 Operación Brinco negado ETIQUETA 2 *Procedimiento 2* BRA ETIQUETA FIN ETIQUETA 2 Operación ... ETIQUETA N-1 Operación Brinco negado ETIQUETA FIN *Procedimiento n* ETIQUETA FIN
FOR límite inf. < variable < límite sup. *Procedimiento* NEXT	LDAA límite superior STAA localidad de memoria ls LDAB límite inferior STAB localidad de memoria li ETIQUETA INICIO *Procedimiento* LDAA ls INC li SUBA li BPL ETIQUETA INICIO

Tabla 3.3. Estructuras de programación

Con el fin de hacer un programa principal fácil de entender, y facilitar el paso de datos entre subrutinas, hicimos una pila en la RAM del usuario -entre las localidades 0000h y 0035h- gracias a ello podemos utilizar el acceso de modo inmediato y de esta manera obtener mayor rapidez. El apuntador de la pila es el índice de registro Y.

Cada subrutina está documentada con una pequeña explicación al principio así como con la definición de las variables que utiliza, información sobre qué hace cada sección, entradas y salidas.

Para llevar a cabo las operaciones matemáticas, utilizamos el sistema de números signados de doble precisión. Cada número estará compuesto de dieciséis bits, siendo el más significativo el de signo; consideramos el punto decimal a la izquierda del bit ocho. Esto hace necesario incluir en las subrutinas de E/S un escalamiento, además de definir las operaciones básicas en el microprocesador, que está pensado para trabajar con números de ocho bits.

La definición de los parámetros de la planta (*a, b* y tiempo muerto), del periodo de muestreo *h*, razón de adaptación γ y las constantes de proporcionalidad e integración k_p y k_i se hace directamente en la memoria, en las localidades llamadas respectivamente: A, B, L, Ts, gamma, kp y ki.

3.5.2 Interrupciones

Una interrupción es una suspensión temporal de la ejecución normal de un programa para que el CPU pueda atender a las condiciones que registran los periféricos. En el caso del HC11 podemos hablar de dos tipos:

- Enmascarables: generadas por los sistemas periféricos del circuito integrado, se reconocen solamente si el bit global de interrupciones mascarables I en el CCR (*Condition Code Register*) es cero. Las interrupciones se pueden habilitar o deshabilitar mediante los bits de máscara (X e I en el CCR) y los registros de control de los periféricos.
- No enmasacarables: Son fuentes importantes de interrupción que siempre están habilitadas. Se hacen utilizando el bit X del CCR y la terminal $\overline{\text{XIRQ}}$

Por omisión, al atender una llamada de interrupción, éstas son deshabilitadas, de tal manera que no se puedan anidar a menos que el usuario expresamente lo indique[13].

Debido a la necesidad de tener control sobre el periodo de muestreo, así como la de conocerlo dentro del microcontrolador para cálculos en las rutinas de integración, predicción y adaptación; generamos interrupciones periódicas utilizando el temporizador principal y los comparadores de entrada y salida.

El temporizador principal es un acumulador de pulsos de dieciséis bits. El periodo de los pulsos se puede variar a manera de múltiplos del reloj del sistema. Cuando llega al final ocurre un sobrepaso y vuelve a comenzar.

Los comparadores de entrada guardan en una localidad de memoria (llamada registro de captura de entrada) el valor que tiene el temporizador principal cuando ocurre algún suceso específico en su terminal asociada. Por el contrario, los de salida ejecutan una acción en su terminal asociada en cuanto el temporizador alcanza determinado número.

Tanto el sobrepaso en el temporizador, como los comparadores de entrada y salida pueden, o bien generar interrupciones enmascarables, o bien afectar solamente las banderas de un registro. En nuestro caso seleccionamos, mediante el registro TMSK1, el generar una interrupción cada vez que se presentara la condición de igualación entre el segundo registro de comparación de salida y el temporizado principal.

Lo primero que se hace al entrar en la subrutina de interrupción es definir el siguiente límite de cuenta mediante la suma al registro de comparación de salida del número de pulsos que queremos transcurran antes de la próxima interrupción. Al final de la subrutina se borra la bandera de interrupción, se regresa al programa principal y se espera la próxima igualación entre el temporizador y el registro. Desde luego se debe garantizar que la subrutina acabe de correr antes de que llegue la nueva interrupción.

El rango de periodos de muestreo deseados, por diseño, se fijó entre 50 y 500 ms. Para abarcarlo, fueron puestos en alto los bits PR1 y PR0 del TMSK2. Esto permitiría un periodo de muestreo mínimo de 8μs y otro máximo de 524.3 ms[14].

Debido a que el microcontrolador sólo permite alterar el registro TMSK2 durante los primeros 60 ciclos de reloj después de reposición, nos vimos en la necesidad de modificar la EEPROM que contiene al BUFFALO.

Como es distinto expresar el periodo de muestreo en segundos que en ciclos de reloj, fue necesario hacer una subrutina que fuera la interfaz entre el usuario, que piensa y hace cálculos en segundos, y el microcontrolador, que solamente cuenta pulsos antes de volver a interrumpir. Esta interfaz es la subrutina ESCALA.A11. El tiempo entre interrupciones se define en segundos en la localidad de memoria "Ts", la subrutina toma este valor y lo pasa a número de ciclos de reloj que guarda en "periodo".

4. Resultados

En esta sección mostramos los resultados de los experimentos llevados a cabo con el fin de analizar comparativamente el desempeño de los controles tipo PI, PIP y PIPA aplicados a un sistema con tiempo muerto largo.

El análisis va enfocado al comportamiento para cada controlador ante tres condiciones distintas de la planta, fruto de la variación en la ventana a: 25°, 45° y 90°.

La actuación del controlador puede observarse no solo en las gráficas de seguimiento, sino también bajo otro enfoque en aquellas de error y error acumulativo. Otro punto de interés es la magnitud del esfuerzo de control que es necesario desarrollar en cada caso.

4.1 Condiciones de los experimientos

El graficar la respuesta en malla abierta del sistema con la ventana en 45° permitió calcular sus características:

$$K=0.97$$
$$L=0.35\ s$$
$$\tau=0.615\ s$$

Siguiendo el método Ziegler-Nichols[15] para sintonización de un controlador PI obtuvimos las constantes del controlador:

$$k_p=0.1187$$
$$k_i=0.1130$$

Con base en las características experimentales obtenidas y trabajando con un periodo de muestreo h de 0.125 s, parametrizamos el sistema utilizando las ecuaciones (2.3) y (2.4):

$$a=-0.8161$$
$$b=0.2901$$

Sin embargo, en la práctica encontramos valores mejores tanto para los parámetros del controlador como para los de la planta. Estos datos conforman la Tabla 4.1

Parámetro	Valor Decimal	Valor Hexagesimal
k_p	0.5	0080h
k_i	1.5	0180h
a	-0.98	FF05h
b	0.0159	0004h

Tabla 4.1. Parámetros encontrados mediante ensayo y error para la planta y controlador, ventana en 45°

4.2 Gráficas

Con la ventana en 45° los tres tipos de controlador probados se comportaron como muestran las ilustraciones Ilustración 4.1, Ilustración 4.2 e Ilustración 4.3.

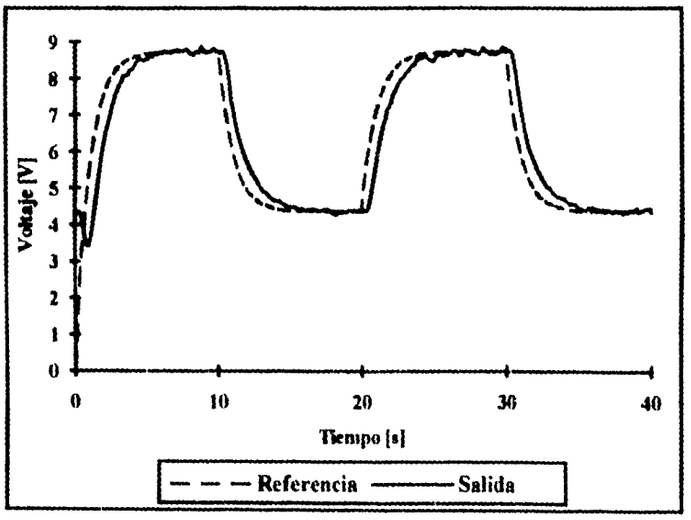


Ilustración 4.1. Seguimiento con control PI, ventana en 45°

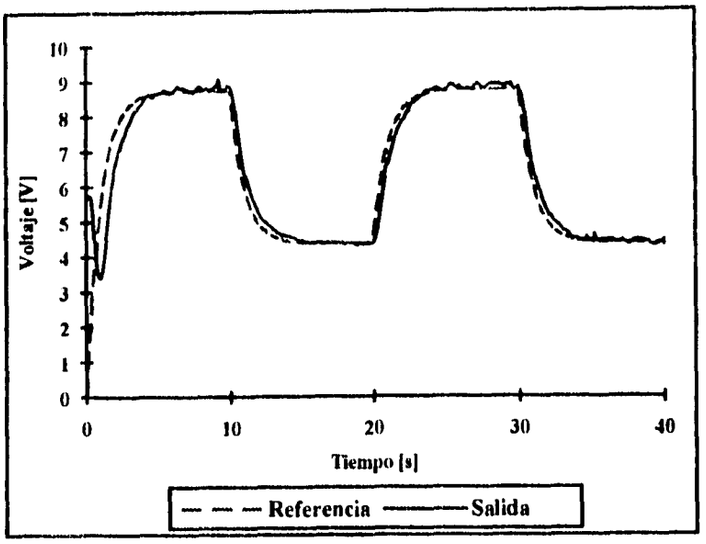


Ilustración 4.2. Seguimiento con control PIP, ventana en 45°

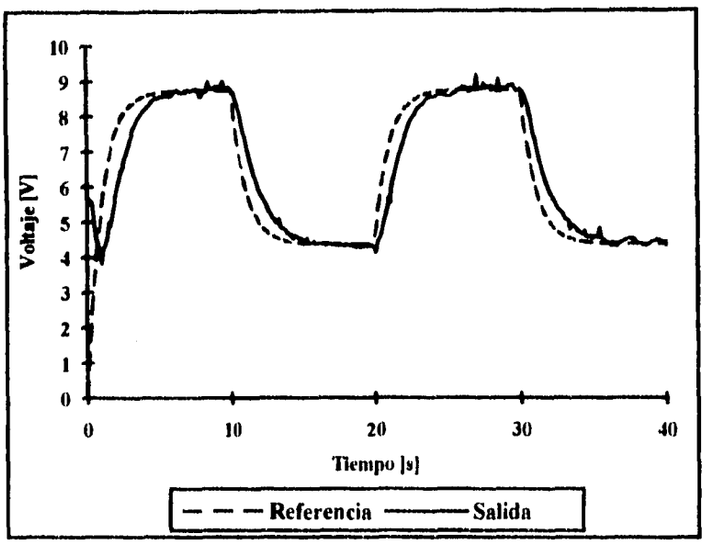


Ilustración 4.3. Seguimiento con control PIPA, ventana en 45°

Cerrando la ventana a 25° el sistema se comporto como se observa en las Ilustración 4.4, Ilustración 4.5 e Ilustración 4.6.

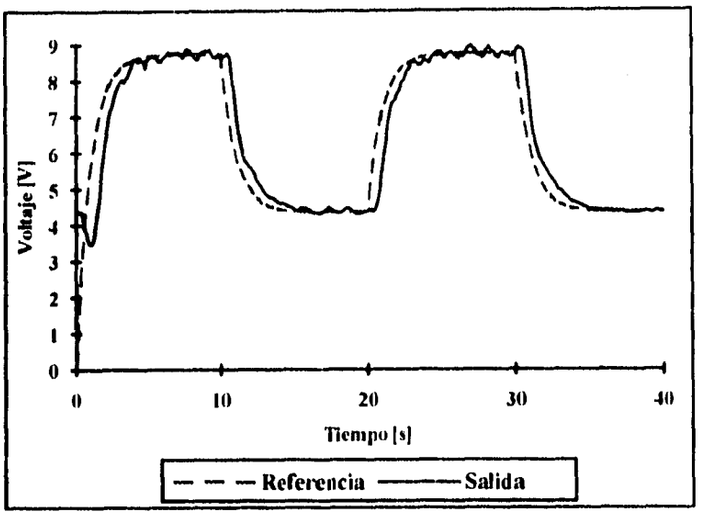


Ilustración 4.4. Seguimiento con control PI, ventana en 25°

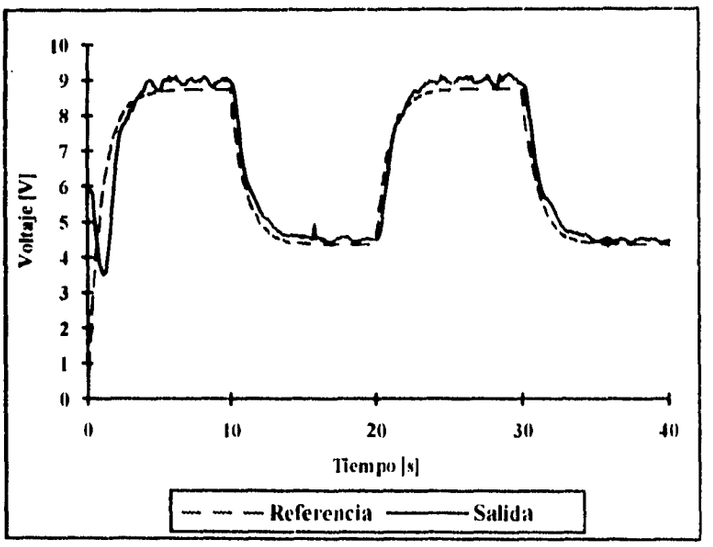


Ilustración 4.5. Seguimiento con control PIP, ventana en 25°

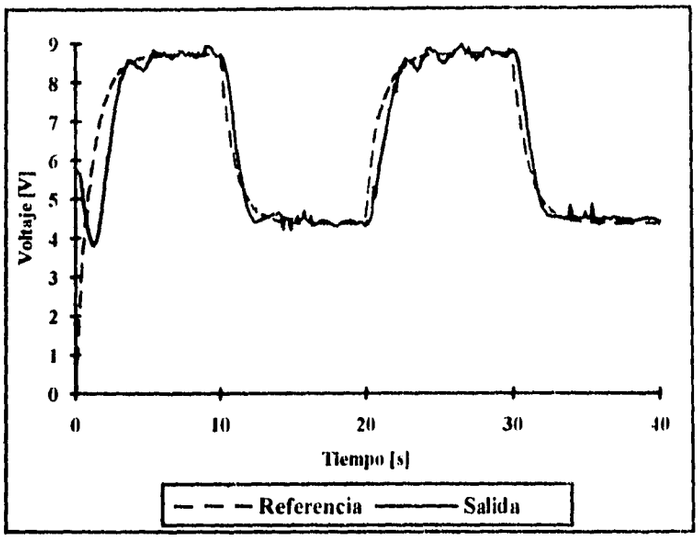


Ilustración 4.6. Seguimiento con control PIPA, ventana en 25°

Con la ventana en 90° obtuvimos las ilustraciones Ilustración 4.7, Ilustración 4.8 e Ilustración 4.9

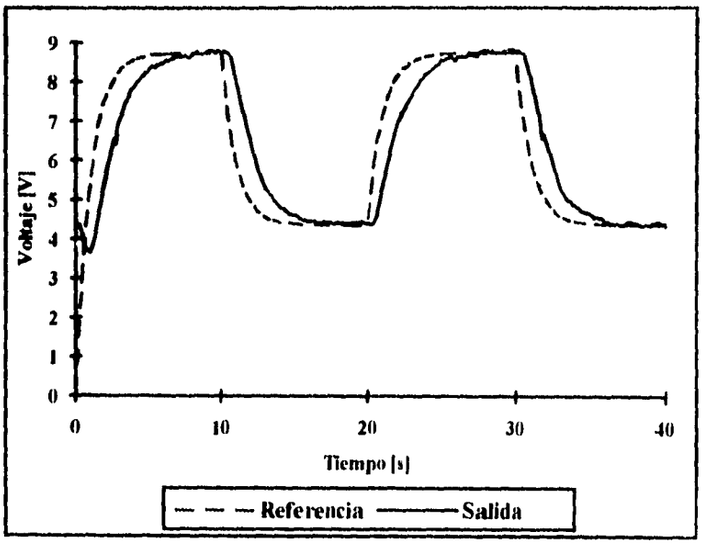


Ilustración 4.7. Seguimiento con control PI, ventana en 90°

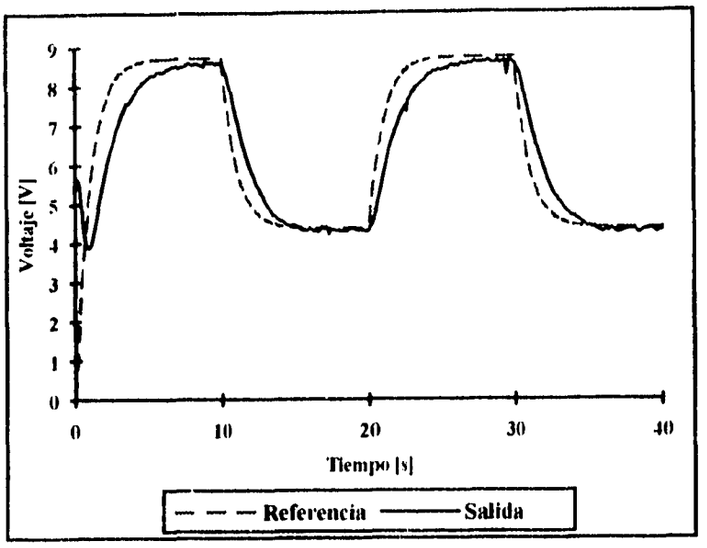


Ilustración 4.8. Seguimiento con control PIP, ventana en 90°

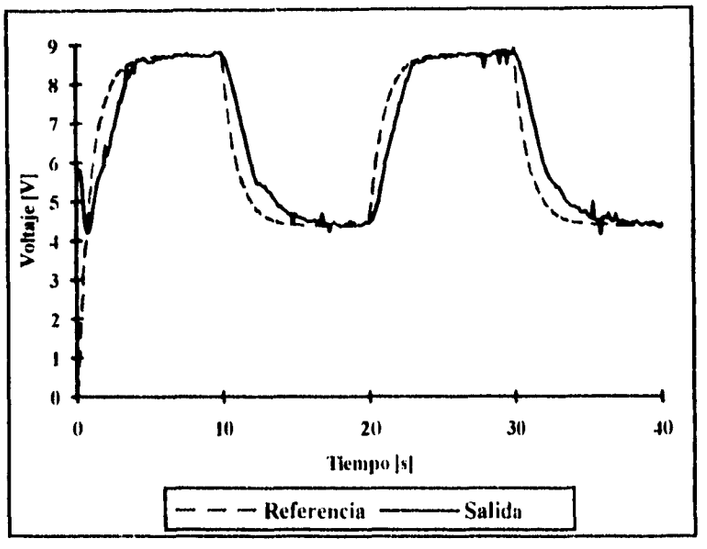


Ilustración 4.9. Seguimiento con control PIPA, ventana en 90°

Las señales que muestran el esfuerzo de control están plasmadas en las ilustraciones Ilustración 4.10, Ilustración 4.11e Ilustración 4.12.

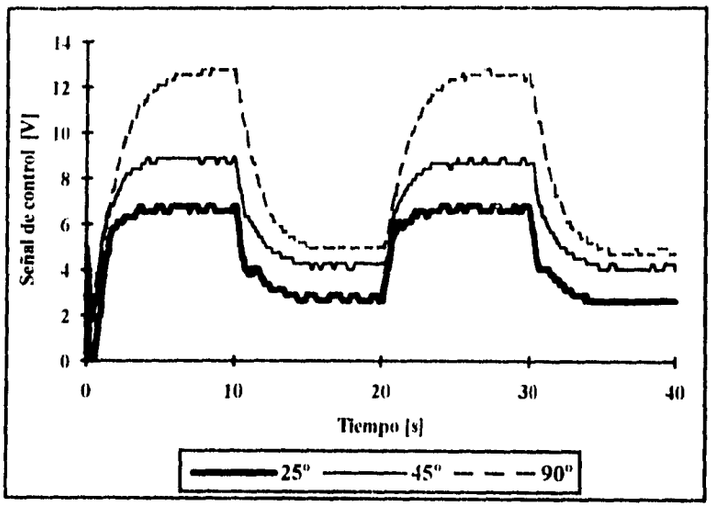


Ilustración 4.10. Señal de control *u(t)*, controlador PI

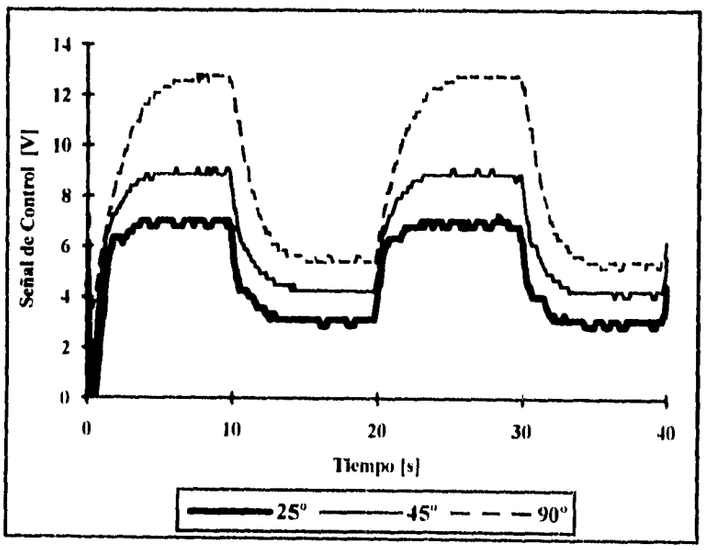


Ilustración 4.11. Señal de control *u(t)*, controlador PIP

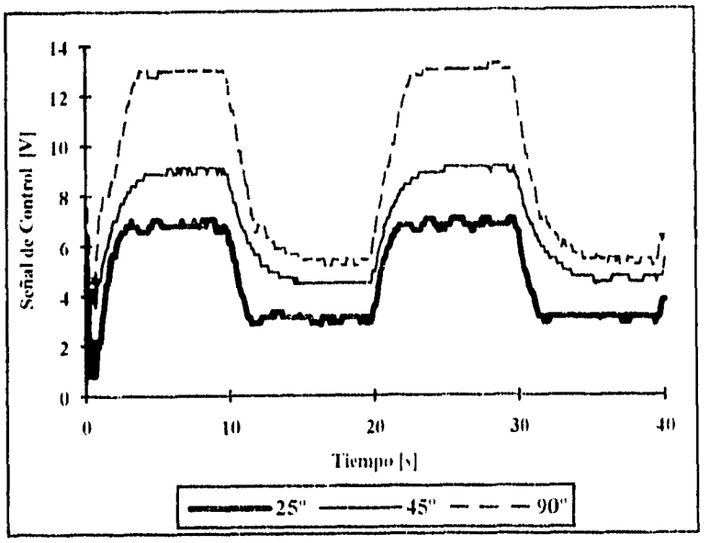


Ilustración 4.12. Señal de control *u(t)*, controlador PIPA

Las diferencias de la respuesta de los distintos controladores actuando sobre la misma planta se aprecian en las ilustraciones Ilustración 4.13, Ilustración 4.14 e Ilustración 4.15.

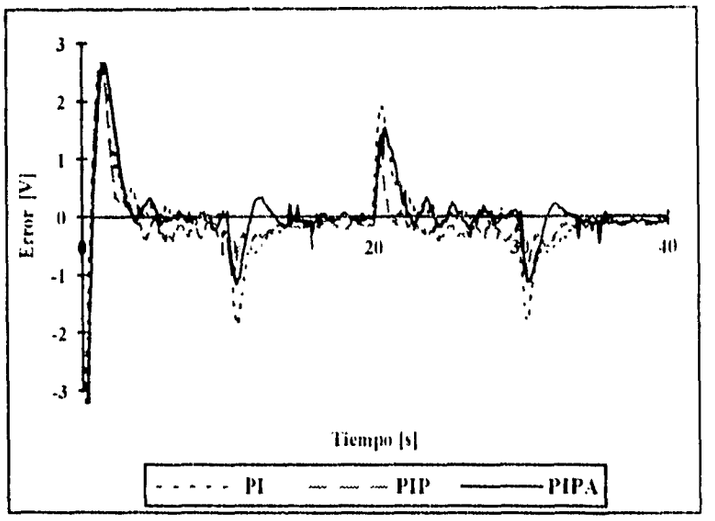


Ilustración 4.13. Error para los tres controladores, ventana en 25°

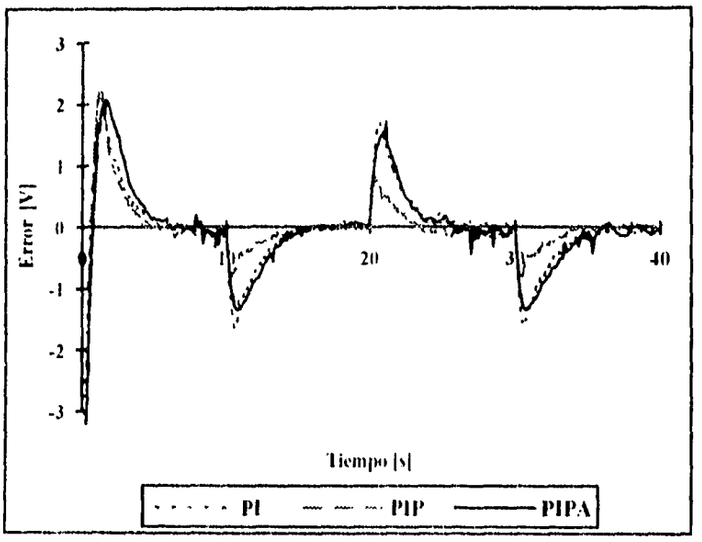


Ilustración 4.14. Error para los tres controladores, ventana en 45°

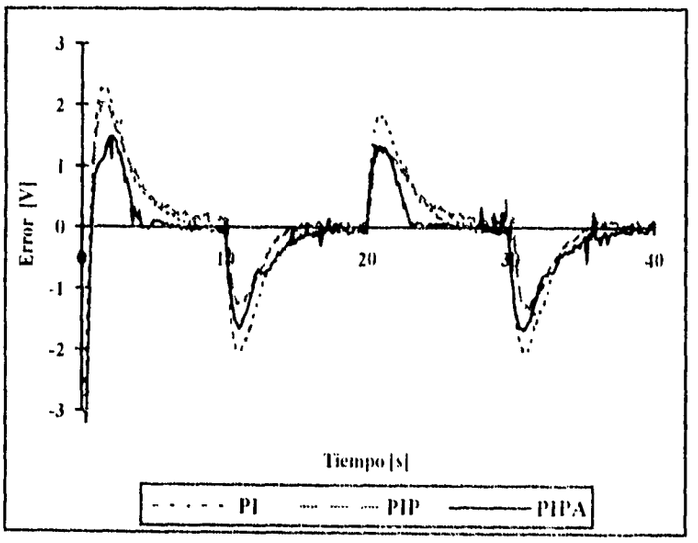


Ilustración 4.15. Error para los tres controladores, ventana en 90°

La sumatoria del valor absoluto del error, error acumulativo, se aprecia en las ilustraciones Ilustración 4.16, Ilustración 4.17 e Ilustración 4.18.

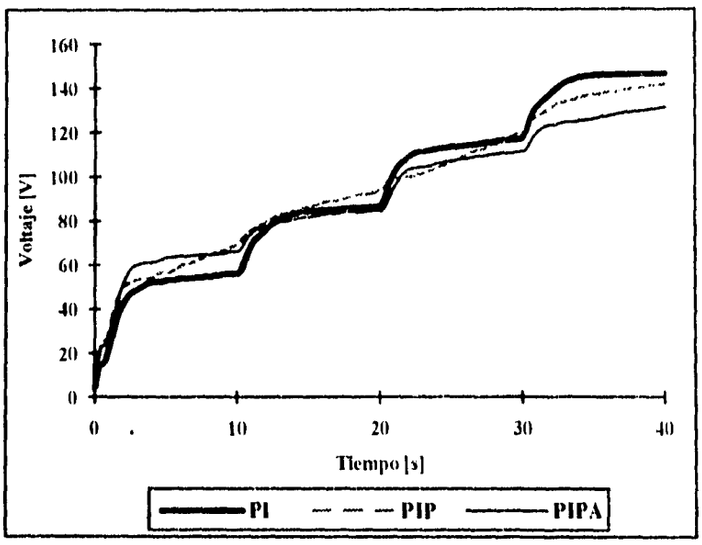


Ilustración 4.16. Error acumulativo, ventana en 25°

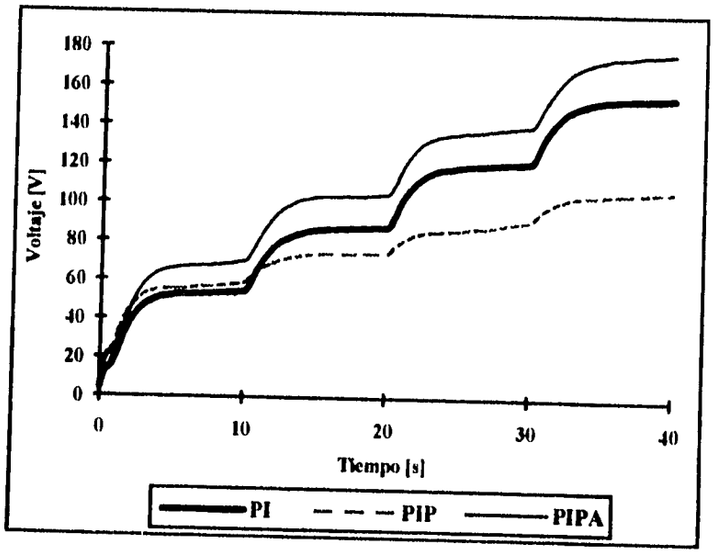


Ilustración 4.17. Error acumulativo, ventana en 45°

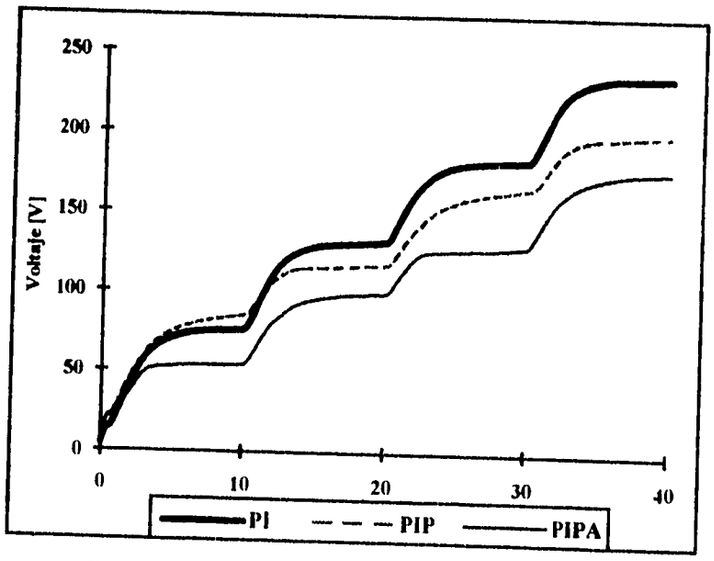


Ilustración 4.18. Error acumulativo, ventana en 90°

5. Discusión de resultados

En la Ilustración 4.2 y en la Ilustración 4.14 podemos ver las ventajas del control PIP cuando la selección de los parámetros a y b es correcta. Esta observación se confirma en la Ilustración 4.17, al transcurrir el tiempo el controlador PIP incurre en un error acumulativo menor. Esto se debe al correcto funcionamiento de la parte predictiva, que anula prácticamente el retraso de la planta.

La situación cambia cuando los parámetros de la planta son modificados mediante la variación de la ventana, la predicción simple -al estar basada en unos parámetros ahora no exactos- arroja datos incorrectos que llevan a que el error no converja en cero. En la Ilustración 4.5 y la Ilustración 4.13 vemos que para 25° el control PIP nunca tiene error cero, esto ocasiona que la pendiente de su error acumulativo (Ilustración 4.17) sea siempre pronunciada.

Las ventajas de la predicción adaptable se observan tanto en el caso anterior (Ilustración 4.6) como cuando la ventana se abre a 90°: la respuesta de los controles PI y PIP se vuelve muy lenta (Ilustración 4.7 e Ilustración 4.8), y éste último no alcanza nunca el valor de referencia; mientras que el tipo PIPA reacciona rápida y exactamente (Ilustración 4.9). Las ilustraciones Ilustración 4.16 e Ilustración 4.18 nos permiten analizar el fenómeno de una manera más evidente.

Comparando las señales de control de las ilustraciones Ilustración 4.10, Ilustración 4.11 e Ilustración 4.12 podemos ver que si bien el esfuerzo de $u(t)$ es mayor para PIPA, nunca es excesivo y disminuye conforme mejora la adaptación al transcurrir el tiempo.

El desempeño del controlador PIPA mejora si los parámetros k_p y k_i del controlador son sintonizados a partir de él y no a partir del control tipo PI como hicimos en nuestra serie de experimentos.

6. Conclusiones

Este trabajo muestra la comparación entre tres tipos de control aplicados a un mismo sistema con tiempo muerto largo en distintas circunstancias. Todos los algoritmos fueron implantados en un microcontrolador de bajo costo y aplicados a una planta real.

El análisis hecho en el capítulo anterior demuestra la conveniencia de introducir un controlador tipo PIP en plantas con retraso largo y características constantes en el tiempo.

Sin embargo, cuando es factible que los parámetros de la planta cambien con el transcurso del tiempo es necesario añadir adaptabilidad al algoritmo de predicción.

Tanto el algoritmo de predicción adaptable como la implantación llevada a cabo en microprocesador son factibles de ser utilizados en la industria, pues el costo es bajo y su manejo es -pasada una primera y única etapa de caracterización- prácticamente igual al de un PI convencional.

El desempeño del controlador PIPA mejoraría drásticamente si contara además con un mecanismo de autosintonización de los parámetros del controlador k_p y k_i. Es decir, si la adaptación no fuera sólo para la predicción sino también para el controlador.

Bibliografía

[1] Bahill, A. Terry "A simple adaptive Smith Predictor for controlling Time-Delay Systems" IEEE Control Systems Magazine Mayo 1983

[2] Shinskey, F. G. "Process-Control Systems" McGraw-Hill EEUU (1979) pp. 8-9

[3] St.Clair, David W. "Improving Control Loop Performance-Without The Math" Control Engineering Octubre 1991

[4] Tang Yu, Okie Lorenzo. "Controlador PI con predicción Adaptiva para Procesos con Tiempo Muerto" Reportes internos de la División de Estudios de Posgrado de la Facultad de Ingeniería, UNAM México (1994)

[5] Hägglund, Tore. "A predictive PI controller por Processes with Long Dead Times". IEEE Control Systems Magazine, Febrero 1992.

[6] Åström, Karl J. & Witenmark, Björn "Computer-Controlled Systems, Theory and Design" Prentice Hall EEUU (1990) pp. 350-351

[7] Goodwin, Graham C. & Sin, Kwai S. "Adaptive Filtering Prediction and Control" Prentice Hall EEUU (1984) pp. 106-108

[8] Åström, Karl J. & Witenmark, Björn "Adaptive Control" Addison-Wesley EEUU (1989) pp. 1-14

[9] Sastry, Shankar & Bodson, Marc "Adaptive Control: Stability, Convergence, and Robustness" Prentice Hall EEUU (1989) pp.6-8

[10] Goodwin, Graham C. & Sin, Kwai S. "Adaptive Filtering Predition and Control" Prentice Hall EEUU (1984) pp. 50-52

[11] Varios autores "Process Trainer PT326" Feedback Instruments Ltd. England.

[12] Varios autores "M68HC11EVB Evaluation Board User's Manual" Motorola Inc. EEUU (1986)

[13] Varios autores "M68HC11 Reference Manual" Motorola Inc. EEUU (1991) capítulo 5

[14] Varios autores "M68HC11 Reference Manual" Motorola Inc. EEUU (1991) capítulo 10

[15] Åstrom, Karl J. & Wittenmark, Björn "Computer-Controlled Systems, Theory and Design" Prentice Hall EEUU (1990) pp. 186, 187

PAGINACION VARIA

COMPLETA LA INFORMACION

A. Apéndice A.

A.1 Programas para el HC11

PRINCIPA

```
; Programa principal para generación de interrupciones que hace uso
; del temporizador principal del HC11.

; Algoritmo:
; Después de una etapa de inicialización, el programa espera a que llegue
; el instante to, que está guardado en TOC2 (Ya que utilizamos el 2o.
; registro de comparación de salida). En ese momento se lleva a cabo una
; interrupción que ejecuta el programa de control (rutina). Al finalizar
; éste, se renueva to sumándole el periodo T y se vuelve a esperar el
; acontecimiento de una interrupción.

; Mediante la alteración del BUFFALO para cambiar el registro TMSK2 se ha
; seleccionado un escalamiento de 16, de tal manera que las características
; del reloj son las siguientes:
                        ; Resolución: 0.008 ms, T: 524.3 ms
; Notas:
;       El vector alfa tiene solo un término.
;       Por cuestiones de programación, es necesario repetir el vector Beta
;       Los valores de kp, ki y Td no se definen aquí, puesto que al
;       sintonizar es más práctico no tener que volver a compilar todo.

; Elaborado por Ignacio Arcelus y Eric López
; 1o. de noviembre de 1994

; Subrutinas
rutina      equ $C400
escala      equ $C1A6
mete        equ $C1C9
divide      equ $C0EA
negar       equ $C000
mult        equ $C05E

; Variables
kp          equ $DFF2       ;Constante del control proporcional
ki          equ $DFF0       ;Constante del control integral
Ts          equ $DFE6       ;Periodo de muestreo en segundos
```

```
error       equ $DFEE     ;Error integral acumulado
lim_inf     equ $DFF4     ;Límite del ciclo
L           equ $DFEC     ;Tiempo muerto en segundos
A           equ $DFEA     ;Constante del modelo
B           equ $DFE8     ;Constante del modelo
d           equ $DFE3     ;Número de pasos de predicción

; Registros
DDRC        equ $1007     ;de dirección de datos del puerto C
option      equ $1039     ;de control
TOC2        equ $1018     ;de comparación, 2a. salida del temporizador
TCTL1       equ $1020
TMSK1       equ $1022
TFLG1       equ $1023     ;Registros de control del temporizador
vector      equ $00DC     ;Vector de interrupciones del TOC2

; Vectores
alfa        equ $DFDF     ;de parámetros de predicción
beta        equ $DB00     ;de parámetros de predicción
control     equ $DBFE     ;de salidas de control

            org $D000

;inicialización de variables:

            ldy #$0000          ;la pila
            ldd #$005C
            std L               ;L=0.36 s
            ldd #$0000
            std error

; inicialización exclusiva para pip y pipa:

            ldd L
            jsr mete
            ldd Ts
            jsr mete
            jsr divide
            ldd 0,y
            staa d
            tba
            beq exacto
              inc d             ;d=int(L/Ts)+1 si L/Ts>int(L/Ts)
exacto      dey
            dey                 ;determina los pasos de adelanto
```

```
                ldd #$0004
                std B                   ;B=0.0159
                ldd #$FF05
                std A                   ;A=-0.98

                ldx #beta
               ldab #$01
               stab lim_inf             ;el limite superior será d
ciclo             clr 0FF,x             ;Llena el vector de control con ceros
                  inx
                  clr 0FF,x
                  inx
                  clr 0FF,x
                  inx
                  clr 0FF,x
                  inx
               ldaa d
                inc lim_inf
               suba lim_inf
                bpl ciclo
```

; Obtención de los coeficientes de alfa(z^-1):

```
                ldd #$0100
                jsr mete
               ldab #$01
               stab lim_inf             ;El limite superior será d
ciclo0            ldd A
                  jsr mete
                  jsr negar
                  jsr mult
               ldaa d
                inc lim_inf
               suba lim_inf
                bpl ciclo0
                ldd 0,y
                dey
                dey
                std alfa
```

; Cálculo de los coeficientes de beta(z^-1)

```
                ldx #beta
                ldd B
                std 0,x                 ;Beta1 = b
```

```
                jsr mete                ;queda como valor inicial de la
                                        ;multiplicación en la pila ldab #$02

                stab lim_inf            ;el limite superior sera d
ciclo1          ldd A
                jsr mete
                jsr negar
                jsr mult
                ldd 0,y
                inx
                inx
                std 0,x
               ldaa d
                inc lim_inf
               suba lim_inf
                bpl ciclo1
                dey
                dey
```

; Repetición del vector beta.

```
                pshy
                ldy #beta
               ldab #$01
               stab lim_inf             ;el limite superior será d
ciclo2          iny
                iny
                ldd 0,y
                inx
                inx
                std 0,x
               ldaa d
                inc lim_inf
               suba lim_inf
                bpl ciclo2
               puly
```

; Carga en el vector de interrupciones la dirección en la que comienza.

```
                ldaa #$7E               ;código de la operación "jmp"
               staa vector
                ldd #rutina             ;lugar de inicio del programa de interés
                std vector+1
```

; Prepara el puerto C para actuar como salida digital.

```
                ldx #DDRC
               bset 0,x,$FF

; Prepara el puerto E para recibir las señales analogicas

                ldx #option
               bset 0,x,$80          ;Habilita el convertidor
               bclr 0,x,$40          ;Selecciona el reloj E

; Prepara al 2o. Comparador de salida del temporizador (TOC2):

                jsr escala           ;Transfiere el periodo en s a pulsos de reloj
               ldaa #$40
               staa TCTL1            ;Cambiará de salida en cada igualación
               staa TFLG1            ;Borra banderas de interrupciones previas
               staa TMSK1            ;Habilita interrupciones del TOC2

; Configura el sistema para poder recibir la interrupción enmascarable.

                cli                  ;Coloca el bit I del CCR listo para
;                                     interrumpir

; Una espera de interrupciones.

espera          wai
                bra espera

;                          PI
;                          ==
; Programa para pruebas con control proporcional-integral del PT.

; Algoritmo:
; Lee la señal de referencia procedente de la tarjeta D/A de simnon por PE5
; mediante la subrutina lee.
; El error se obtiene restando la salida a la referencia (es decir PE5-PE4)
; La señal de control está dada por la siguiente regla:
;                  u(t)=Kp*e(t)+Ki*sumatoria(e(t))
; La sumatoria del error se obtiene sumando al error acumulado el producto
; del periodo de muestreo por el error instantáneo.
; La señal de control sale por el puerto B utilizando la subrutina escribe.

; Elaborado por Ignacio Arcelus y Eric López
; 19 de octubre de 1994
```

```
;Subrutinas
negar       equ $C000
suma        equ $C021
mult        equ $C05E
lee         equ $C14E
escribe equ $C183
dup         equ $C19B
mete        equ $C1C9

;Variables
kp          equ $DFF2
ki          equ $DFF0
error       equ $DFEE   ;Localidad para guardar el error acumulado
Ts          equ $DFE6   ;Periodo de muestreo.
periodo     equ $DFE4   ;Periodo de muestreo expresado en ciclos de reloj
temporal    equ $DFF6

;Registros
TOC2        equ $1018    ;2o. comparador de salida del temporizador
TFLG1       equ $1023    ;Registro de banderas de los comparadores de salida

            org $C400
            ldd periodo
          addd TOC2
            std TOC2     ;Definición de la próxima interrupción
            jsr lee
            jsr negar
            jsr suma     ;Obtiene el error

            jsr dup
            ldd Ts
            jsr mete
            jsr mult     ;Obtiene el error integral instantáneo.

            ldd error
            jsr mete
            jsr suma
            ldd 0,y
            std error    ;Actualiza el error acumulado

            ldd kp
            std 0,y
            jsr mult     ;Obtiene el primer término de la ley de control

            ldd error
```

```
        jsr mete
        ldd Ki
        jsr mete
        jsr mult        ;Obtiene el segundo término de la ley de control

        jsr suma        ;Obtiene la ley de control final
        jsr escribe
        dey
        dey

        ldx #TFLG1
        bclr 0,x,$BF
        rti
```

PIP.A11

```
; Subrutina que hace el control proporcional-integral-predictivo

; Subrutinas
lee         equ $C14E
controla    equ $C1D1   ;Hace el control PI
predice     equ $C211   ;Hace la predicción

; Variables
lim_inf equ $DFF4   ;Límite del ciclo
d           equ $DFE3   ;Número de pasos de predicción
periodo     equ $DFE4   ;Periodo de muestreo en ciclos de reloj

; Registros
TOC2        equ $1018   ;2o. comparador de salida del temporizador
TFLG1       equ $1023   ;Registro de banderas de los comparadores de salida.
puertoC     equ $1003

; Vectores
control     equ $DBFD   ;de salidas de control
respuesta   equ $DD00   ;de salidas del sistema

            org $C400
            ldd periodo
          addd TOC2
            std TOC2            ,Definición de la próxima interrupción

; Actualización del vector respuesta
```

```
                jsr lee
                std respuesta
                dey
                dey
                jsr predice             ;Obtiene la salida predicha
                jsr controla            ;Hace el control pi

; Actualiza el vector de señales de control

                ldx #control
                ldd 0,y                 ;Coloca la señal de control nueva un lugar
                std 0,x                 ;antes del vector
                ldab d
                abx
                abx
                dex                     ;Coloca el apuntador del vector en la parte
                dex                     ;alta del penúltimo argumento

        ldab #$01
        stab lim_inf    ;el limite superior será d
ciclo3          ldd 0,x
                std 2,x
                dex
                dex
        ldaa d
                inc lim_inf
        suba lim_inf
                bpl ciclo3
                dey
                dey

        ldx #TFLG1              ;Limpia la bandera para poder volver a
        bclr 0,x,$BF            ;interrumpir
        rti

;                       PIPA.A11
;                       ========
;
; Subrutina que hace el control proporcional-integral-predictivo-adaptable,
; que va a estimar los parámetros desconocidos y variantes en el tiempo
; de la planta con la ayuda de la ecuación de gradiente normalizada mientras
; que el error de estimación sea diferente de cero.
; La estimación se lleva a cabo en la subrutina estima.
; La predicción se lleva a cabo en la subrutina predice.
; El control PI a partir de r(t+d) y y(t+d) se hace en la subrutina controla
```

```
; Elaborada por Ignacio Arcelus y Eric Lopez
; 30 de noviembre de 1994

; Subrutinas
negar       equ $C000
suma        equ $C021
mult        equ $C05E
divide      equ $C0EA
lee         equ $C14E
escribe     equ $C183
dup         equ $C19B
mete        equ $C1C9
controla    equ $C1D1
predice     equ $C211
grafica     equ $C249
estima      equ $C261

;Variables
lim_inf     equ $DFF4   ;Límite del ciclo
A           equ $DFEA   ;Constante del modelo
B           equ $DFE8   ;Constante del modelo
d           equ $DFE3   ;Número de pasos de predicción
alfa        equ $DFDF   ;Constante de la predición
gamma       equ $DFDD   ;Razón de adaptación
const       equ $DFDB   ;Constante intermedia
kp          equ $DFF2   ;Constante proporcional
ki          equ $DFF0   ;Constante integral
error       equ $DFEE   ;Contiene el error acumulado
Ts          equ $DFE6   ;Periodo de muestreo
periodo     equ $DFE4   ;Periodo de muestreo en ciclos de reloj
temporal    equ $DFF6   ;Variable temporal para el redondeo

;Registros
TOC2        equ $1018   ;2o. comparador de salida del temporizador
TFLG1       equ $1023   ;Registro de banderas de los comparadores de salida
puertoC     equ $1003   ;Registro del puerto C

;Vectores
beta        equ $DB00   ;Vector de constantes intermedias
control     equ $DBFD   ;Vector de salidas de control
respuesta   equ $DD00   ;Vector de salidas de la planta

            org $C400
```

```
            ldd periodo
           addd TOC2
            std TOC2          ;Definicion de la próxima interrupcion

            jsr lee

; Actualiza el vector de señales de salida del sistema

            ldx #respuesta
            dex
            dex
            ldd 0,y           ;Coloca la señal de salida nueva un lugar
            std 0,x           ;antes del vector
            ldab d
            abx               ;Coloca el apuntador del vector en la parte
            abx               ;alta del penultimo argumento

      ldab #$00
      stab lim_inf            ;el ciclo se hará d+1 veces
ciclo       ldd 0,x
            std 2,x
            dex
            dex
      ldaa d
      inc lim_inf
      suba lim_inf
      bpl ciclo

; Prepara el cálculo del error de estimación:

            jsr negar

; Calcula la salida estimada del sistema:
            ldx #respuesta
            ldab d
            abx
            abx
            ldd 0,x
            jsr mete          ;Coloca la y(t-d) en la pila
            ldd alfa
            jsr mete
            jsr mult
            ldx #beta         ;Coloca el apuntador en u(t-d-1)
            ldab d
            abx
```

```
                abx

        ldab #$01
        stab lim_inf    ;El limite superior será d
ciclo1                  ldd 0,x ;Toma una beta
                        jsr mete
                        ldd OFF,x       ;Toma una señal de control
                        jsr mete
                        jsr mult
                        jsr suma
                        inx
                        inx
        ldaa d
        inc lim_inf
        suba lim_inf
        bpl ciclo1

                jsr grafica

; Calcula el error de estimación
                jsr suma

; Estima los parámetros estimados si el error de estimación no es nulo

                beq nulo
                bra no_nulo
nulo                    dey
                        dey             ;Quita de la pila el error de estimacion
                bra fin
no_nulo                 jsr estima
fin             jsr predice
                jsr controla

; Actualiza el vector de señales de control

                ldx #control
                ldd 0,y                 ;Coloca la señal de control nueva un lugar
                std 0,x                 ;antes del vector
                ldab d
                abx
                abx
                abx
                abx
                dex                     ;Coloca el apuntador del vector en la parte
                dex                     ;alta del penúltimo argumento
```

```
                ldab #$01
                stab lim_inf            ;el limite superior será d
ciclo5          ldd 0,x
                std 2,x
                dex
                dex
                ldd 0,x
                std 2,x
                dex
                dex
                ldaa d
                inc lim_inf
                suba lim_inf
                bpl ciclo5

                dey
                dey                     ;Limpia la pila

                ldx #TFLG1              ;Limpia la bandera para poder volver a
                bclr 0,x,$BF            ;interrumpir
                rti
```

OPERA.A11

```
; Compendio de las subrutinas que operan números signados de un par de bytes
; y de aquellas que son de uso común en los algoritmos de control.

; Elaborado por Ignacio Arcelus y Eric López
; 2o. semestre de 1994

;                               MAPA DE MEMORIA

; Registros
option          equ $1039       ;de opciones del microprocesador
adctl           equ $1030       ;de control del A/D
adr1            equ $1031       ;salida 1 del A/D relacionada a y(t), PE0 y PE4
adr2            equ $1032       ;salida 2 del A/D relacionada a r(t), PE1 y PE5
adr3            equ $1033       ;salida 3 del A/D relacionada a PE2 y PE6
adr4            equ $1034       ;salida 4 del A/D relacionada a PE3 y PE7
puertoB         equ $1004       ;Dirección del puerto B
puertoC         equ $1003       ;Direccion del puerto C

; Subrutinas
negar           equ $C000
```

```
cambia      equ $C012
suma        equ $C021
mult        equ $C05E
divide      equ $C0EA
lee         equ $C14E
escribe     equ $C183
dup         equ $C19B
mete        equ $C1C9
grafica     equ $C278

; Variables
bandera     equ $DFFF     ;Registro de banderas
sumando1    equ $DFFD
sumando2    equ $DFFB
factor1     equ $DFF9
factor2     equ $DFF7
divisor     equ $DFF9     ;(Denominador)
dividendo   equ $DFF7
temporal    equ $DFF6     ;Variable temporal
kp          equ $DFF2     ;Ganancia proporcional del controlador
ki          equ $DFF0     ;Ganancia integral del controlador
error       equ $DFEE     ;Localidad para guardar el error acumulado
Ts          equ $DFE6     ;Periodo de muestreo [s]
periodo     equ $DFE4     ;Periodo de muestreo expresado en ciclos de reloj
lim_inf     equ $DFF4     ;Límite inferior de ciclo
d           equ $DFE3     ;Número de pasos de predicción
A           equ $DFEA     ;Constante (parámetro) del modelo de la planta
B           equ $DFE8     ;Constante (parámetro) del modelo de la planta
ganma       equ $DFDD     ;Razón de adaptación
const       equ $DFDB     ;Constante intermedia

; Vectores
alfa        equ $DFDF     ;de parámetros de predicción
beta        equ $DB00     ;de parámetros de predicción
control     equ $DBFD     ;de salidas de control
respuesta   equ $DD00     ;de salidas del sistema
```

NEGAR A11

```
. Subrutina que obtiene el complemento a dos de un número signado
. de doble precisión (Dos bytes)
. El índice de registro Y debe apuntar a la parte alta del numero
. Entrada  ultimo elemento de la pila (Dos bytes)
```

```
; Salida  pila

                org $C000

                ldd 0,y
                comb            ;complemento a uno del byte bajo.
                coma            ,complemento a uno del byte alto
                std 0,y
                ldd #0001
               addd 0,y         ;complemento a dos de todo el número

                std 0,y
                rts

                        CAMBIA.A11
                        ==========

; Subrutina que obtiene el complemento a dos de un número signado
; de doble precisión (Dos bytes). Además cambia la bandera que indica
; número negativo.

; El indice de registro Y debe apuntar a la parte alta del número.

; Entrada: último elemento de la pila (Dos bytes)
; Salida: acumulador D
;         registro de banderas

                jsr negar

               ldaa #$01
               eora bandera
               staa bandera     ,actualiza la bandera
                ldd 0,y
                rts

;                       SUMA.A11
;                       ========

; Subrutina para sumar dos números signados de doble precision.

; Entrada: Dos números en la pila.
; Salida:  Un número en la pila.

; Algoritmo:
;       -Realiza la suma.
;       -La probabilidad de sobrepaso se presenta si tienen signo igual.
;       - Para números positivos hay sobrepaso si el MSB es 1.
```

```
;      - Para numeros negativos hay sobrepaso si hay acarreo.

; 9 de agosto de 1994

                ldd 0,y
                std sumando1
                dey
                dey
                ldd 0,y
                std sumando2
               addd sumando1
                std 0,y              ;efectúa la suma y la guarda en la pila.
               ldaa sumando1
               eora sumando2         ;decide si hay probabilidad de sobrepaso
                bpl prob_sobrepaso
                bra fin10

prob_sobrepaso
                ldd sumando1  ;define el tipo de sobrepaso probable
                bpl positivos
                bra negativos

positivos       ldd 0,y ;sobrepaso de números positivos
                bpl fin12
                  ldd #7FFF
fin12           bra fin11
negativos       ldd 0,y          ;sobrepaso de numeros negativos
                bmi fin13
                ldd #8001

fin13
fin11           std 0,y

fin10           rts

;                               MULT A11
;                               ========

; Subrutina que multiplica dos numeros signados de doble precisión.
; Entrada: factor1 en el penúltimo número de la pila
;          factor2 en el último número de la pila

; Salida:  producto en la pila

; Algoritmo:

; Sea la parte alta de los dos números a multiplicar a1 y b1 respectivamente
```

```
; Sea la parte baja de los dos números a multiplicar a2 y b2 respectivamente.
; El HC11 puede multiplicar dos números de 8 bits, dando como resultado uno
; de 16

; Sean los siguientes subproductos:

;       sp1 = LSBy (a1*b1)
;       sp2 = MSBy (a2*b2)
;       sp3 = a2*b1
;       sp4 = a1*b2

;       Producto = factor1*factor2 = sp1sp2 + sp3 + sp4

; El último bit es redondeado.
; Es necesario que todos los subproductos sean positivos.

; 9 de agosto de 1994

                pshx                    ;Evita que se modifique X con la subrutina

                ldx #bandera
                bclr 0,x,$01            ;inicializa el registro de banderas

                ldd 0,y
                bpl fin21
                   jsr cambia           ;el número es negativo

fin21           std factor2
                dey
                dey
                ldd 0,y
                bpl fin22
                   jsr cambia           ;el número es negativo.
fin22           std factor1             ;inicializa las variables
                ldaa factor1
                ldab factor2
                mul
                stab 0,y                ; Realiza sp1 y lo guarda en pila

                tab
                beq correcto
                bra sobrepaso

correcto          ldaa factor1+1
                  ldab factor2+1
                    mul                 ; Realiza sp2
                    iny
```

```
                staa 0,y
                 dey                ,Forma sp1 sp2

                stab temporal
                ldab #$80
                andb temporal       , Verifica si se debe de redondear hacia arriba
                beq se_queda        , o hacia abajo

                    ldd #$0001
                    jsr mete
                    jsr suma

se_queda          ldd 0,y
                  bmi sobrepaso
                ldaa factor1
                ldab factor2
                  mul
                  bita #$80         , Si está encendido quiere decir que el resultado
                  bmi sobrepaso     ; es negativo, y eso no puede ser.
                  jsr mete
                  jsr suma          ; Realiza sp3 y lo suma a sp1sp2

                ldaa factor1
                ldab factor2+1
                  mul
                bita #$80           ;Si está encendido el resultado
                bmi sobrepaso       ;es negativo, y eso no puede ser
                  jsr mete
                  jsr suma          ;Realiza sp4 y lo suma a (sp1sp2+sp3).
                  bra fin20

sobrepaso         ldd #$7FFF
                  std 0,y           ;Guarda el resultado en la pila.

fin20           ldx #bandera
              brclr 0,x,$01,fin23

                  jsr negar

fin23           pulx
                  rts
```

; DIVIDE A11

; Subrutina que divide dos números signados de doble precision

```
; Entrada  dividendo en el penúltimo numero de la pila
;                divisor en el ultimo número de la pila

; Salida:  cociente en la pila

; Algoritmo: Se hace primero la división no signada.

; Sea la parte alta del denominador b1 y b2 la parte baja
; Sea la parte alta del numerador a1 y a2 la parte baja.

; La división de dos números de 16 bits da como resultado uno de 16

; Existen dos tipos de divisiones en el microprocesador:
; -Para números con el punto decimal en el mismo sitio (IDIV).
; -Para cuando el denominador es 2^16 menor al numerador (FDIV).

; La división IDIV de a1a2 entre b1b2 nos da un cociente q1q2
; más un residuo r1r2.
; La división FDIV de r1r2 entre b1b2 nos da un cociente q3q4
; más un residuo r3r4.
; El resultado es LSBy(q1q2)•MSBy(q3q4) mas el redondeo relativo al LSBy(q3q4).

; Habrá sobrepaso si el MSBy(q1q2) es mayor a cero o si LSBy(q1q2) > 7F.
; En este caso el resultado será 7FFF u 8001, dependiendo de los signos

; Elaborada por Ignacio Arcelus y Eric López.
; 11 de agosto de 1994

            pshx

            ldx #bandera
                bclr 0,x,$01    ;inicializa el registro de banderas.

            ldd 0,y
            bne opera
                ldd #0001

opera       bpl fin30

                jsr cambia      ,el denominador es negativo

fin30       std divisor
            dey
            dey
            ldd 0,y
            bpl fin31           ;el numerador es positivo

              jsr cambia        ;el numerador es negativo
```

```
fin31       ldx divisor
            idiv                ; cociente en x, residuo en d
             std dividendo
            xgdx
            stab 0,y            ; guarda en la pila la parte alta del resultado final

            bmi sobrepaso1      ;si la parte baja del cociente > 7F hay sobrepaso

            tab
             beq correcto1
                bra sobrepaso1  ;si la parte alta del cociente > 0 hay sobrepaso

correcto1       ldd dividendo   ;no hubo sobrepaso
                ldx divisor
               fdiv             ; cociente en x, residuo en d
               xgdx
                iny
               staa 0,y         ; guarda la parte baja del resultado final en la pila
                dey
                tba
                bpl se_queda1
                   ldd #$01
                   jsr mete
                   jsr suma

se_queda1       bra fin32

sobrepaso1      ldd #$7FFF
                   std 0,y      ;Guarda el resultado en la pila
fin32

              ldx #bandera
            brclr 0,x,$01,fin33

               jsr negar

fin33         pulx
               rts

;                           LEE A11
;                           =======

; Subrutina de lectura de datos

; Entrada: Señales analógicas en el puerto E (PE4 y PE5)
; Salida: Los datos leidos y normalizados son colocados en la pila
; La normalizacion consiste en pasar de números no signados de 1 byte a
```

; números signados de 2 bytes

; Algoritmo. Se repite la parte alta en la baja y luego se suma $8000 sin
; fijarse en el acarreo

; Elaborada por Ignacio Arcelus y Eric López
; 11 de octubre de 1994.

```
            ldaa #14        ;selecciona conversión múltiple y
            staa adctl      ;discontinua de PE4-PE7

            ldaa #80
ciclo       bita adctl
                bpl  ciclo      ;espera a que termine las conversiones
```

;etapa de normalización:

```
              ldd #8000
              std temporal

            ldaa adr2
              tab
              bne fin40
                incb            ;evita la formación de $0000
fin40       addd temporal
              iny
              iny
              std 0,y

            ldaa adr1
              tab
              bne fin41
                incb            ;evita la formación de $0000
fin41       addd temporal
              iny
              iny
              std 0,y

              rts
```

ESCRIBE A11

; Subrutina para escribir en el puerto B los números normalizados de doble
; precisión signados en formato de números no signados de un solo byte

```
; Entrada: Un número signado de doble precisión en la pila
; Salida: Un número no signado de un solo byte en el puerto B

, Algoritmo: Se le suma $8000 sin considerar acarreo a la entrada
;                    Se redondea el primer byte de acuerdo al segundo
;                    El número de salida se queda en la pila

; Elaborada por Ignacio Arcelus y Eric López
; 12 de octubre de 1994

                ldd #$8000
                std temporal
                ldd 0,y
               addd temporal

;sección de redondeo:

                comb        ;Genera los cambios de bandera necesarios
                  bmi se_queda2
                    inca    ; hace el redondeo
                    bne no_es_FF2
                      ldaa #$FF
no_es_FF2
se_queda2

              staa puertoB
                rts

;                              DUP A11
;                              =======

; Subrutina para duplicar un número en la pila

; Entrada: último número de la la pila
, salida. nueva última posición en la pila

; Elaborada por Ignacio Arcelus y Eric López
; 19 de octubre de 1994

                ldd 0,y
                iny
                iny
                std 0,y

                rts

                              ESCALA A11
```

; Subrutina para escalar el tiempo del temporizador principal al resto
; del algoritmo de control, especificamente al periodo de muestreo.

; Algoritmo:

; Para sacar el equivalente en pulsos del temporizador del periodo de
; muestreo en segundos es necesario multiplicarlo por un número tal que el
; resultado sea el número de pulsos que el reloj debe contar para que
; transcurra dicho lapso.

; Un periodo de muestreo de FFFF pulsos en el temporizador equivale a
; 0.5243 s, es decir a $0086.

; Multiplicando el periodo por el número #$01E7, y repitiendo la parte baja
; del resultado en la parte alta encontramos el número proporcional de pulsos
; buscado.

; NOTA:Para que haya congruencia física, el periodo siempre debe estar dentro
; del rango entre #$000D y #$0086.

```
ldd Ts
iny
iny
std 0,y

ldd #$01E7          ;Factor de escala
iny
iny
std 0,y

jsr mult

ldd 0,y
tba
std periodo

dey
dey
rts
```

METE.A11

; Subrutina para meter en la pila el dato que este cargado en el acumulador D.

```
; Entrada: Acumulador D
; Salida: Ultima posicion de la pila.

; Elaborada por Ignacio Arcelus y Eric López
; 10 de noviembre de 1994

                iny
                iny
                std 0,y

                rts

;                       CONTROLA
;                       ========

; Subrutina que hace el control PI.

; Entrada:
;       - Referencia en el penultimo número de la pila
;       - Salida en el último número de la pila
;       - En memoria:
;               - Kp, Ki, error acumulado y Ts.

; Salida:
;       - Señal de control u(t) en la pila y el puerto B

; Algoritmo:
; El error instantáneo se obtiene restando los últimos dos numeros de la pila.
; La señal de control está dada por la siguiente regla:
;               u(t)=Kp*e(t)+Ki*sumatoria(e(t))
; La sumatoria del error se obtiene sumando el error acumulado al producto
; del error instantáneo por el periodo de muestreo.
; La señal de control sale por el puerto B utilizando la subrutina escribe

; Elaborada por Ignacio Arcelus y Eric López
; 1o de diciembre de 1994.

                jsr negar
                jsr suma        ;obtiene el error instantáneo

                jsr dup
                ldd Ts
                jsr mete
                jsr mult        ;obtiene el error integral instantaneo

                ldd error
                jsr mete
```

```
jsr suma
ldd 0,y
std error       ;actualiza el error acumulado

ldd kp
std 0,y
jsr mult        ;obtiene el primer término de la ley de control

ldd error
jsr mete
ldd ki
jsr mete

jsr mult    ;obtiene el segundo término de la ley de control

jsr suma    ;obtiene la ley de control final

jsr escribe

rts
```

PREDICE.A11

```
; Subrutina que obtiene la salida predicha y(t+d).

; Entrada:
;       - El vector de salidas anteriores: "resultado" en memoria
;       - El vector de señales de control: "control" en memoria
;       - Los vectores de parámetros: "alfa" y "beta" en memoria
;       - El número de pasos de predicción d.

; Salida:
;       - La salida predicha y(t+d) en la pila y en el puerto C.

; Algoritmo:
; Se basa principalmente en la utilización de las ecuaciones de Bezout,
; que en una forma particular son:

;               A(z^-1)F(z^-1)+z^-dG(z^-1) = 1

;               siendo que,
;                       deg(F) = d-1
;                       deg(G) = n-1

;                           donde: (n) Es el orden del sistema
;                                  (d) Es el número de pasos de
;                                      adelanto
```

```
; Asi obtenemos

;                   y(t+d) = beta(z^-1)u(t) +alfa(z^-1)y(z^-1)

;                   siendo que,
;                       beta(z^-1) = F(z^-1)B(z^-1)
;                       alfa(z^-1) = G(z^-1)

;                           donde  (B(z^-1)) Es el numerador de la
;                                   función de transferencia
;                                   de nuestro modelo.

; Los vectores alfa y beta son obtenidos en el programa principal junto con d

; Elaborada por Ignacio Arcelus y Eric López
; 1o. de diciembre de 1994.

; Obtención del producto de alfa por y(t)
                ldd respuesta
                jsr mete
                ldd alfa
                jsr mete
                jsr mult

; Obtención de la predicción de y(t+d):

                ldx #beta

               ldab #$01
               stab lim_inf        ;El limite superior será d
ciclo2            ldd 0,x          ;Toma una beta.
                  jsr mete
                  ldd 0FF,x        ;Toma una señal de control
                  jsr mete
                  jsr mult
                  jsr suma
                  inx
                  inx
                 ldaa d
                  inc lim_inf
                 suba lim_inf
                  bpl ciclo2

                 jsr grafica

                nop
                nop                ; Para no mover el mapa de memoria al omitir
```

```
                nop                 ; la graficación

                rts

;                               GRAFICA A11
;                               ===========

; Procedimiento que saca por el puerto C la informacion en la pila para que
; sea graficada en Simnon.

; Entrada:
;               - En la pila: el valor a ser representado gráficamente

; Salida:
;               - Dicho valor en el puerto C
;               - El número queda en la pila.

; Elaborado por Ignacio Arcelus y Eric López
; 6 de diciembre de 1994.

                ldd #$8000
                std temporal
                ldd 0,y
               addd temporal

;sección de redondeo:

                comb            ,Genera los cambios de bandera necesarios
                bmi se_queda4
                  inca          ,Hace el redondeo
                    bne no_es_FF4
                      ldaa #$FF
no_es_FF4
se_queda4
               staa puertoC

                rts

;                               ESTIMA A11
;                               ==========
;
; Subrutina que emula un controlador proporcional-integral-predictivo-adaptable
; que va a estimar los parametros desconocidos y variantes en el tiempo
; de la planta con la ayuda de la ecuación de gradiente normalizada.

; Entrada
;               - El error de estimación 'ee' en la pila
```

```
;                   - Los vectores en memoria
;                         alfa, beta, respuesta y control.
;
; Salidas:
;                   - Los nuevos vectores alfa y beta en memoria
;
; Algoritmo:
; Se estiman los nuevos parámetros con base en:
;
;     teta(t+1) = teta(t) - fi(t-d)·(ee·gamma/1+fiT(t-d)·fi(t-d))
;     teta(t+1) = teta(t) + fi(t-d)·K
;
;                   1. Producto ee·gamma
;                   2. Obtención de fiT(t-d)·fi(t-d)
;                   3. Suma de lo anterior con uno
;                   4. Obtención de K
;                   5. Obtención de K·fi(t-d)
;                   6. Obtención de teta (t+1) (betas y alfas)
;
; NOTA: Para evitar sobrepasos, el denominador del segundo término está
;       dividido entre 100. En el numerador esta división se realiza
;       dando en memoria una constante gamma 100 veces menor que la real
;
; Elaborada por Ignacio Arcelus y Eric Lopez
; 2 de diciembre de 1994

                ldd gamma
                jsr mete
                jsr mult            ;producto ee·gamma

                ldx #respuesta
               ldab d
                abx
                abx
                ldd 0,x
                jsr mete            ;Coloca la y(t-d) en la pila
                ldd #$0A00
                jsr mete
                jsr divide
                jsr dup
                jsr mult

                ldx #beta           ;Coloca el apuntador en u(t-d-1)
               ldab d
                abx
                abx
```

```
                ldab #$01
                stab lim_inf            ;El limite superior será d
ciclo3          ldd 0FF,x               ; Toma una señal de control
                jsr mete
                ldd #$0A00
                jsr mete
                jsr divide
                jsr dup
                jsr mult
                jsr suma
                inx
                inx
               ldaa d
                inc lim_inf
               suba lim_inf
                bpl ciclo3

                ldd #$0002              ;aproximadamente 0.01d
                jsr mete
                jsr suma
                jsr divide
                jsr negar
                ldd 0,y
                std const               ;Calcula la constante

                ldx #respuesta
               ldab d
                abx
                abx
                ldd 0,x
                jsr mete                ;Coloca y(t-d) en la pila
                ldd #$0A00
                jsr mete
                jsr divide
                jsr mult
                ldd alfa
                jsr mete
                jsr suma
                ldd 0,y
                std alfa                ;Calcula la nueva alfa
                dey
                dey

                ldx #beta               ;Coloca el apuntador en u(t-d-1)
               ldab d
                abx
```

```
                abx

; Calcula las nuevas betas y además las deja en la pila para llenar la parte
; baja del vector

                ldab #$01
                stab lim_inf            ;El limite superior será d
ciclo4          ldd 0FF,x               ;Toma una señal de control
                jsr mete
                ldd #$6400
                jsr mete
                jsr divide
                ldd const
                jsr mete
                jsr mult
                ldd 0,x                 ;Toma la beta anterior
                jsr mete
                jsr suma
                ldd 0,y
                std 0,x
                inx
                inx
                ldaa d
                inc lim_inf
                suba lim_inf
                bpl ciclo4


                ldx #beta               ;Coloca el apuntador en u(t-d-1)
                ldab d
                abx
                abx

; Llena la parte baja del vector de las betas

                ldab #$01
                stab lim_inf            ;El limite superior será d
ciclo5          ldd 0,y
                dey
                dey
                dex
                dex
                std 0,x
                ldaa d
                inc lim_inf
                suba lim_inf
```

```
bpl ciclo5

rts
```

A.2 Programas para Simnon

```
"                                   FILTRADO.T
"                                   ----------

"Archivo que recibe las entradas analógicas y genera una
"señal escalón de magnitud tal, que provoca en el PT un cambio de
"referencia entre 30 y 50 grados centigrados.
"Para facilitar el seguimiento, se filtra la señal escalón.

"Elaborado por Ignacio Arcelus y Eric López
"18 de octubre de 1994

CONTINUOUS SYSTEM filtrado
TIME t
STATE filtro filpred
DER   dfiltro dfilpred

      dfiltro=a*(ra-filtro) dfilpred=a*(rb-filpred)
      cuadrada=sqw(t*f)
      predice=sqw((t+L)*f)
      d=if L/Ts>int(L/Ts) then int(L/Ts)+1 else L/Ts
      ra=if cuadrada>0 then lim_sup/2 else lim_inf/2
      rb=if predice>0 then lim_sup/2 else lim_inf/2 control=10*14.55/10*ADIN(u,t)
      ref=10*14.55/5*DAOUT(r,filtro) refpred=DAOUT(r,filpred)
      salida=10*14.55/5*ADIN(y,t) auxiliar=10*14.55/10*suma
      error=ref-salida

"Cálculo del valor del puerto C, que se lee individualmente en las
"entradas digitales 0 a 7

      suma = (D0+D1+D2+D3+D4+D5+D6+D7)/256
      D0 = DIGIN(0,t)
      D1 = DIGIN(1,t)*2
      D2 = DIGIN(2,t)*4
      D3 = DIGIN(3,t)*8
      D4 = DIGIN(4,t)*16
      D5 = DIGIN(5,t)*32
      D6 = DIGIN(6,t)*64
      D7 = DIGIN(7,t)*128
```

```
lim_sup:0.600          "Estos valores se seleccionaron de acuerdo con las
lim_inf:0.300          "temperaturas de salida que puede alcanzar el actua-
                       "dor sin saturarse, con el fin de poder observar me
                       "jor las caracteristicas de la señal de salida.

    a:1                "Constante del filtro de la referencia
    f:0.050            "frecuencia del tren de escalones [Hz]
    L:0.36             "Tiempo muerto [s] (0.36)
    Ts:0.125           "período de muestreo [s]
    r:3                "canal 3 (sense) tarjeta D/A
    u:3                "canal 3 (13H) tarjeta A/D
    y:4                "canal 4 (14H) tarjeta A/D
"   d                  "Número de pasos de adelanto de prediccion

END

"                              MUESTRA2.T
"                              ==========

"Conjunto de instrucciones que toma las señales analógicas del sistema
"y permite hacer uso de ellas dentro de la computadora

"Elaborado por Ignacio Arcelus y Eric López
"18 de octubre de 1994


"Funciones a Graficar

"control           Es la salida de nuestra tarjeta con una magnitud
"                  entre 0-10 V y la entrada al PT en el punto A.
"                  Está normalizada para graficación.
"salida            Es la salida del PT, marcada con la letra Y. Entra
"                  a nuestra tarjeta por "Entrada analógica (0-14.5 V).
"                  A la tarjeta de simnon entra tomada de PE4 y es nor-
"                  malizada y filtrada en filtrado.t.
"rf                Es la señal de referencia producida mediante simnon.
"                  Su magnitud es escalable entre 0 y 5 V de acuerdo a
"                  los parámetros lim_inf y lim_sup. Entra al HC11
"                  directamente por PE5.
"ref               Es la señal de referencia normalizada para la
"                  graficación.
"error             Es la diferencia entre ref y salida
"                  NOTA: Es necesario escalarlas, puesto que el rango
"                  de salida es tres veces el de ref.
"auxiliar          Es la varible asociada al puerto C del microcontro-
"                  lador. Se utiliza para graficar señales durante el
```

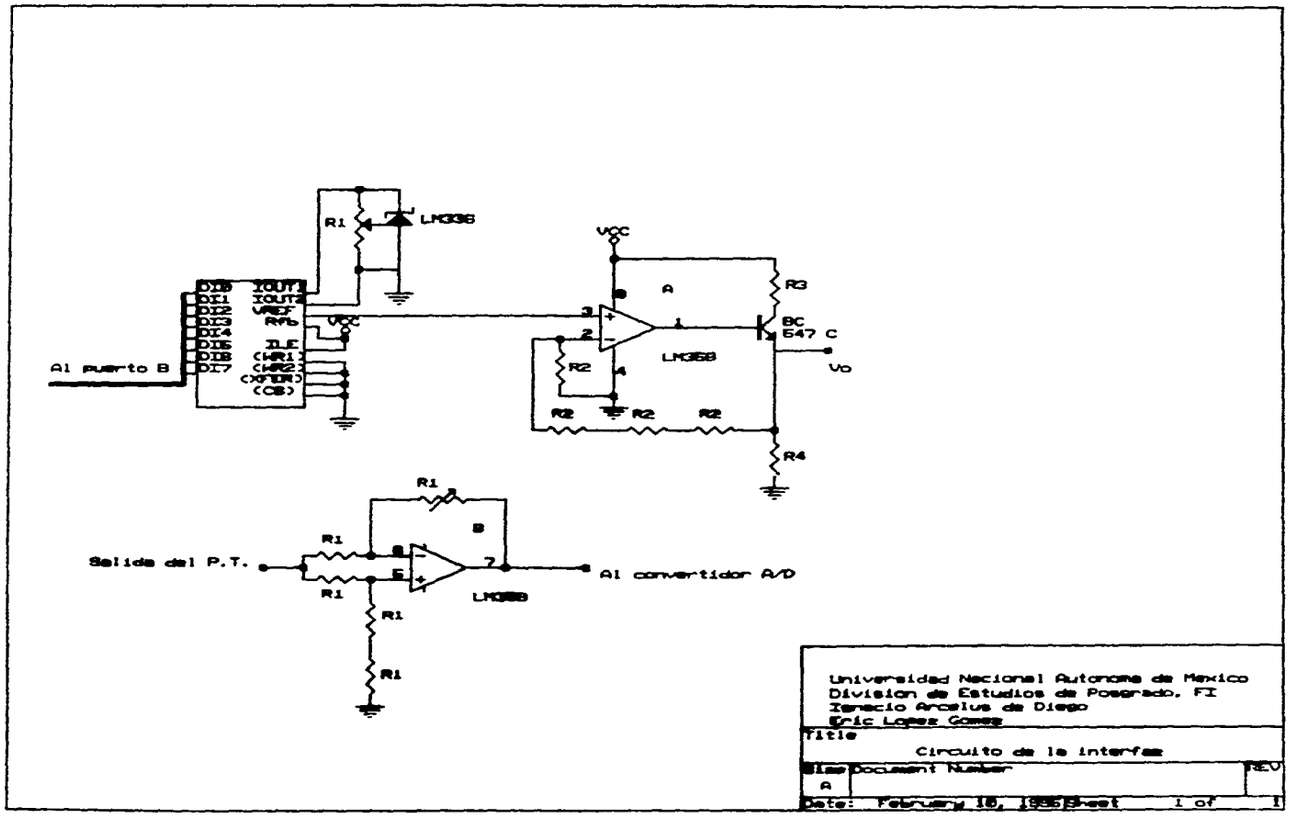
desarrollo, tales como el error integral y la salida predicha.

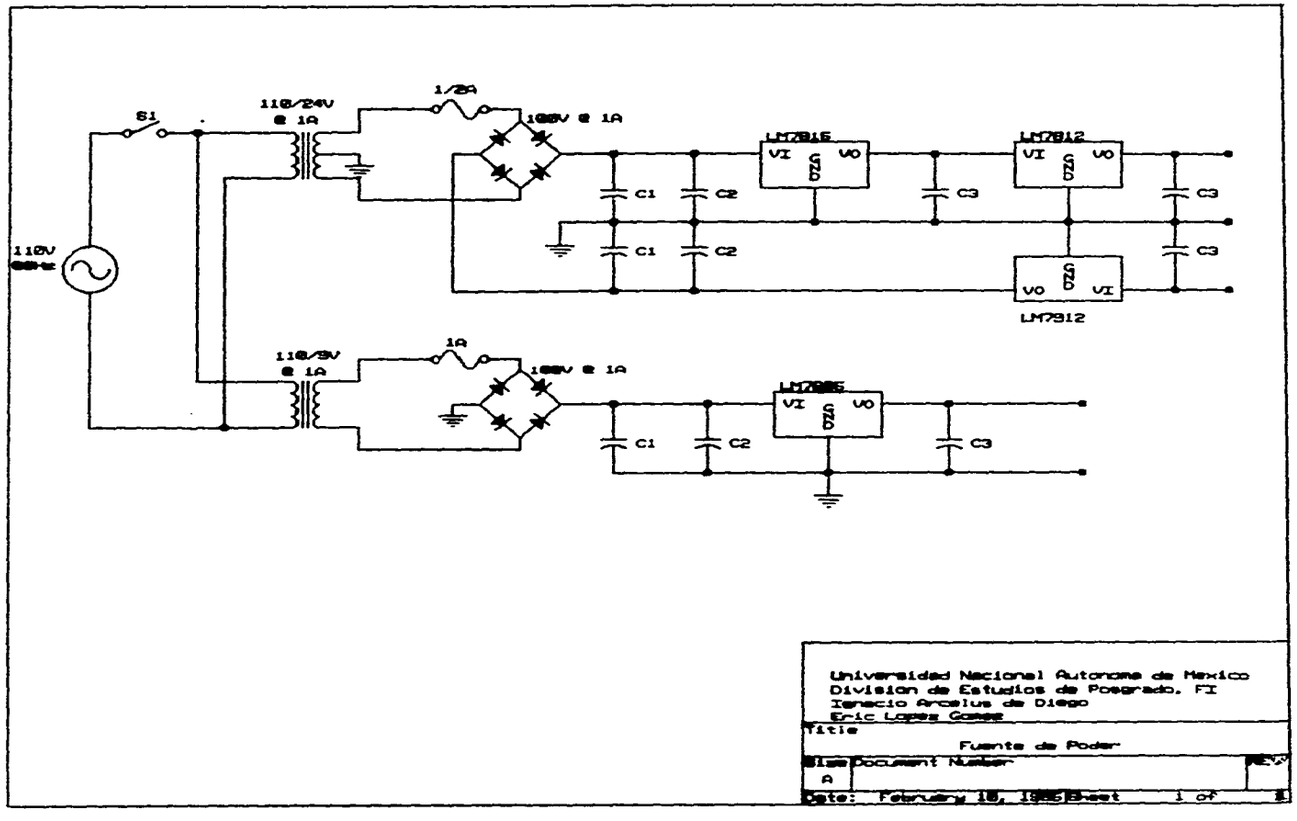
Macro muestra2

```
syst filtrado
axes h 0 40 v 0 15
switch color blue red green white
plot salida ref "control auxiliar
store ref salida error control "auxiliar
simu 0 40 0.1
end
```

Apéndice B.

Circuitos de Interfaz


R1
LM336
VCC
A
R3
BC
547 C
Vo
LM358
R2
R4
Al puerto B
B
Salida del P.T.
Al convertidor A/D
LM358
Universidad Nacional Autonoma de Mexico
Division de Estudios de Posgrado, FI
Ignacio Arcelus de Diego
Eric Lopez Gomez
Title
Circuito de la interfaz
Size
A
Document Number
REV
Date: February 18, 1995
Sheet 1 of 1


S1
110/24V
@ 1A
1/2A
1A
110/9V
@ 1A
110V
C1
C2
C3
LM7815
LM7812
LM7912
VI
VO
GND
Universidad Nacional Autonoma de Mexico
Division de Estudios de Posgrado, FI
Ignacio Arcelus de Diego
Eric Lopez Gomez
Title
Fuente de Poder
Size
A
Document Number
REV
Date:
Sheet
of

Valores de los elementos

Resistencias		
R1	10	kΩ
R2	12	kΩ
R3	1.2	kΩ
R4	82	Ω

Capacitores		
C1	4700	μF
C2	0.01	μF
C3	1	μF