CONTROL EN CASCADA EN TIEMPO REAL DE UN PROCESO INDUSTRIAL CASO DE ESTUDIO: PLANTA DE TANQUES INTERACTUATES

ANEXOS

ERMILSO DIAZ BENACHI

YONNY FERNANDO CABEZAS ORTIZ

Director

I. E. Juan Fernando Flórez Marulanda

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL

POPAYÁN

2008

TABLA DE CONTENIDO

LISTA DE FIGURAS

PÁG.

LISTA DE TABLAS

PÁG.

PREFACIO

El documento presentado a continuación, contiene varios apartados que lejos de ser una producción autónoma e inédita, es una recopilación de datos experimentales relizados por el equipo de trabajo de este proyecto. Tambien se exponen conceptos de diferentes autores, algunos sin mayores modificaciones. Sin embargo se han adicionado algunos conceptos y sugerencias como resultado de conocimientos adquiridos en el desarrollo de este trabajo de grado.

Los Autores.

ANEXO A. EXPERIMENTO BIAS (Uo) ADICIONADO A LA ESTRATEGIA DE CONTROL PID

A.1. DISEÑO DEL EXPERIMENTO

La planta usada para la observación de la implicación del parámetro *bias* en el controlador PID esta representado por la ecuación 1.

$$G_{lc}(s) = \frac{K_p e^{-t_m s}}{\tau s + 1} \quad (1)$$

Donde:

t_m Es el retardo en segundos y tiene un valor de 1.2 segundos.

τ Es la constante de tiempo del sistema, tiene un valor de 1.2 segundos.

K_p Es la ganancia de la planta, con un valor de 1.

De acuerdo a la literatura existen diferentes métodos de sintonización, para este caso se preseleccionan los métodos basados en criterios integrales para el funcionamiento del PID como servomecanismo y como regulador los cuales se explican a continuación.

A.1.1 Métodos de sintonización PID como regulador

Método de López, Miller, Smith y Murril. El primer método basado en criterios integrales que presentó ecuaciones para el cálculo de los parámetros del controlador es conocido como el método de López. Definiendo una función de costo como la ecuación 2.

$$\Phi = \int_0^\infty F(e(t), t)dt \quad (2)$$

Donde F es una función del error y del tiempo, se obtiene un valor que caracteriza la respuesta del sistema. Entre menor sea el valor de Φ, mejor será el desempeño del

sistema de control, por ello, un desempeño óptimo se obtiene cuando Φ es mínimo. Las ecuaciones obtenidas se presentan en (3) y los parametros se aprecian en la tabla 1.

$$K_c k_p = a\left(\frac{t_m}{\tau}\right)^b \quad \frac{T_i}{\tau} = \frac{1}{c}\left(\frac{t_m}{\tau}\right)^{-d} \quad \frac{T_d}{\tau} = e\left(\frac{t_m}{\tau}\right)^f \tag{3}$$

Donde Kp es la ganancia proporcional, Ti el tiempo iintegral y Td el tiempo integral.

Tabla 1. Parámetros de las ecuaciones del método de López.

	a	***b***	***c***	***d***	***e***	***f***
IAE	**1.435**	**-0.921**	**0.878**	**-0.749**	**0.482**	**1.137**
ITAE	1.357	-0.947	0.842	-0.738	0.381	0.995
ISE	1.495	-0.945	1.101	-0.771	0.560	1.006

Los criterios de desempeño utilizados por López fueron: Integral del error absoluto (IAE), Integral del error absoluto por el tiempo (ITAE) e Integral del error cuadrático (ISE). Se utilizaron en lazos de control que funcionan como reguladores con un controlador PID-Ideal.

Método de Kaya y Sheib. Kaya y Sheib realizaron lo mismo que López, pero para controladores que denominaron PID- Clásico (PID-Serie), *PID No Interactuante* (una variación del *PID-Paralelo*) y *PID-Industrial*. Los valores de las constantes obtenidas se presentanen la tabla 2.:

Tabla 2. Constantes para el método de Kaya y Sheib.

Controlador clásico - Regulador						
	a	***b***	***c***	***d***	***e***	***f***
IAE	0.98089	-0.76167	0.91032	-1.05211	0.59974	0.89819
ITAE	0.77902	-1.06401	1.14311	-0.70949	0.57137	1.03826
ISE	1.11907	-0.89711	0.79870	-0.95480	0.54766	0.87798
Controlador no interactuante - Regulador						
IAE	1.31509	-0.88260	1.25870	-1.37560	0.56550	0.45760
ITAE	1.31760	-0.79370	1.12499	-1.42603	0.49547	0.41932

ISE	1.34660	-0.93080	1.65850	-1.25738	0.79715	0.41941
Controlador industrial - Regulador						
IAE	0.91000	-0.79380	1.01495	-1.00403	0.54140	0.78480
ITAE	0.70580	-0.88720	1.03326	-0.99138	0.60006	0.97100
ISE	1.11470	-0.89920	0.93240	-0.87530	0.56508	0.91107

A.1.2. Métodos de sintonización PID como servomecanismo

Método de *Rovira, Murrill* y *Smith.* Se basaron en el método de López para controladores PID que operan como reguladores. Conocido como el método de *Rovira*, consiste en definir una función de coste (ecuación 2), por medio de dicha función se obtiene un valor que caracteriza la respuesta del sistema. Entre menor sea el valor de Φ, mejor será el desempeño del sistema de control por lo que un desempeño optimo se obtiene cuando Φ sea mínimo. Los criterios de desempeño utilizados por Rovira fueron: Integral del error absoluto (IAE) y la Integral del error absoluto por el tiempo (ITAE), para lazos de control que funcionan como servomecanismos, el controlador usado es un PID Ideal. En (4) se presentan las ecuaciones obtenidas y en la tabla 3 se presentan los valores de los parámetros para este método.

$$K_c k_p = a\left(\frac{t_m}{\tau}\right)^b \quad \frac{T_i}{\tau} = \frac{1}{c + d\left(t_m/\tau\right)} \quad \frac{T_d}{\tau} = e\left(\frac{t_m}{\tau}\right)^f \tag{4}$$

Tabla 3. Parámetros método de Rovira

	a	***b***	***c***	***d***	***e***	***f***
IAE	1.435	-0.921	0.878	-0.749	0.482	1.137
ITAE	1.357	-0.947	0.842	-0.738	0.381	0.995
ISE	1.495	-0.945	1.101	-0.771	0.560	1.006

Método de *Kaya and Sheib.* Utiliza los mismos criterios de análisis que el método de *Rovira*, las diferencias están en que este método se aplicó a controladores de estructura serie (clásico), PID no interactuante (standar) y el PID industrial, además los parámetros de las ecuaciones son diferentes. Los parámetros de las ecuaciones de este método se observan en tabla 9.

Tabla 4. Parámetros método de Kaya y Sheib

Controlador clásico - Servomecanismo						
	a	***b***	***c***	***d***	***e***	***f***
IAE	0.65000	-1.04432	0.98950	0.09539	0.50814	1.08433
ITAE	1.12762	-0.80368	0.99783	0.02860	0.42844	1.00810
ISE	0.71959	-1.03092	1.12666	-0.18145	0.54568	0.86411
Controlador no interactuante - Servomecanismo						
IAE	1.13031	-0.81314	5.75270	-5.72410	0.32175	0.17707
ITAE	0.98384	-0.49851	2.71348	-2.29778	0.21443	0.16768
ISE	1.26239	-0.83880	6.03560	-6.01910	0.47617	0.24572
Controlador industrial - Servomecanismo						
IAE	0.81699	-1.00400	1.09112	-0.22387	0.44278	0.97186
ITAE	0.83260	-0.76070	1.00268	0.00854	0.44243	1.11499
ISE	1.14270	-0.93650	0.99223	-0.35269	0.35308	0.78088

A.1.3. Sintonización como Reguladores

Con la planta antes descrita y con los métodos de sintonización mencionados se hizo una simulación que cosistio en introducir una perturbación al sistema en t=40s correpondiente a un aumento en el VM de 5 unidades, esto con el fin de observar la variable controlada (VC) y el esfuerzo de control (EC). La simulación se realizó en la herramienta software Matlab-7.2a.

En la figura 1 se observa la VC en donde el sobre pico es igual para la mayoría de métodos, sin embargo la oscilación sucesiva siguiente a este sobre impulso es menor en elmetodo ITAE, así como el tiempo de establecimiento, de igual manera en la figura 2 para el EC. Para el PID serie la como se precia la VC y el EC en la figura 3 y 4 respectivamente, el metodo que mayor porcentaje de rechazo presenta y menores osciaciones es el método ITAE. En la figura 4 se observa un menor esfuerzo de control con el método ITAE, también presenta un tiempo de establecimiento menor que en los con los otros métodos de sintonización.

Figura 1. VC PID paralelo industrial sintonizado como regulador

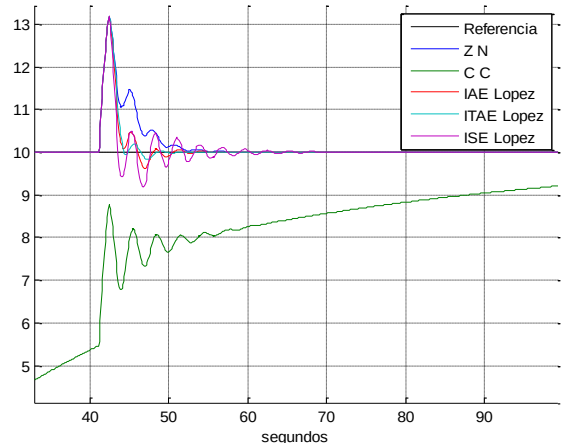


Figura 2. EC PID paralelo industrial sintonizado como regulador

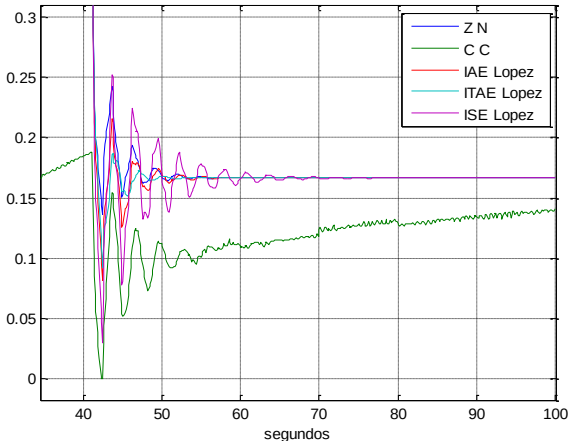


Figura 3. VC PID serie industrial sintonizado como regulador

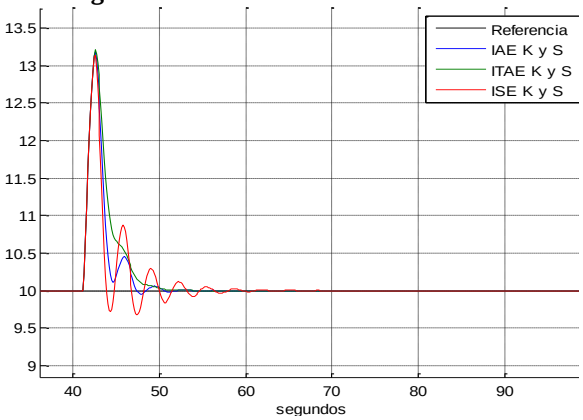
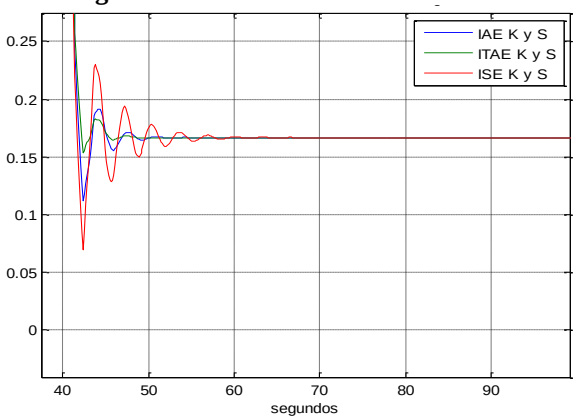


Figura 4. EC PID serie industrial sintonizado como regulador



A.1.4. Sintonización como Servomecanismos

La simulacion consiste en efectuar un cambio de el valor de cosigna de -10 unidades, estando el sistema estable en un valor de 20. Como servomecanismos, en el PID paralelo se observa un mejor comportamiento del método ITAE ante los otros métodos en cuanto a la respuesta de la VC, figura 5, igualmente mejor respuesta del EC, figura 6.

Para el caso del pid serie se observa también un mejor comportamiento en la respuesta de la VC del método ITAE ante cambios en el valor de consigna, figura 7, igualmente el EC de la figura 8, la sintonizacion con dicho metodo evita la presencia de oscilaciones y posee un menor tiempo de establecimiento.

Figura 5. VC PID paralelo industrial sintonizado como servomecanismo

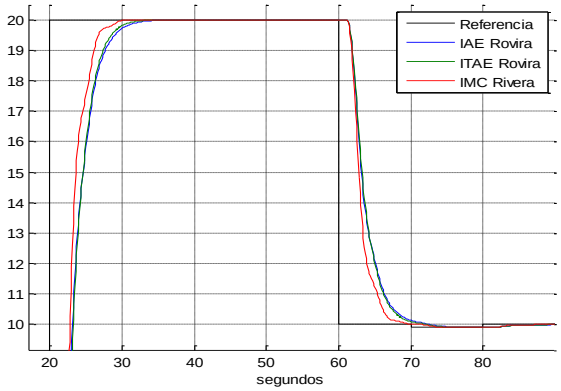


Figura 6. EC PID paralelo industrial sintonizado como servomecanismo

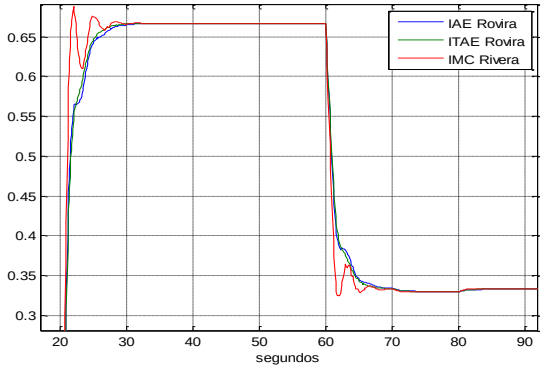


Figura 7. VC PID serie industrial sintonizado como servomecanismo

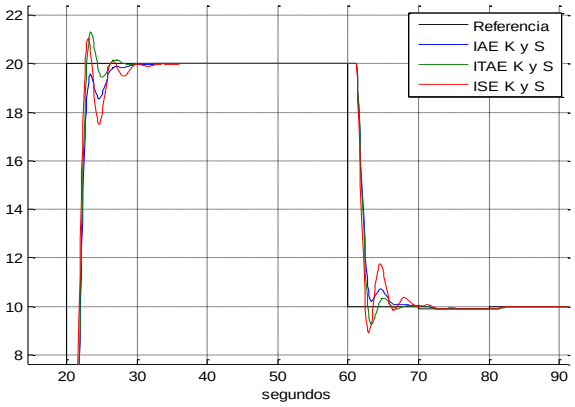
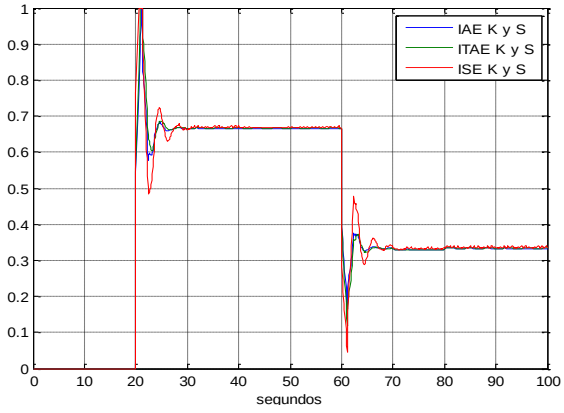


Figura 8. EC PID serie industrial sintonizado como servomecanismo



Una vez simulado el efecto de cada método de sintonización son selecciona uno de los métodos que ofrecen mayor seguimiento ante cambios en el valor de consigna y rechazo a perturbaciones y un permiten un mejor esfuerzo de control. El método seleccionado para el análisis del adicionamiento del parámetro bias (Uo) es el método del criterio de ITAE, teniendo en cuenta los siguientes criterios.

- De los métodos de sintonización usados, es uno de los que mejor funcionamiento tiene tanto como regulador y como servomecanismo.
- De los métodos integrales usados el método ITAE tiene un buen desempeño, y permite hacer una diferenciación clara como regulador y como servomecanismo tanto en controladores estándar, serie, paralelos, lo cual es muy apropiado para la experiencia.

A.2. SIMULACION PARÁMETRO U_O PID PARALELO Y SERIE

El siguiente paso es realizar la simulación, usando el método ITAE tanto como regulador y como servomecanismo, para ver la implicación del U_o, en la dinámica del sistema y en los esfuerzos de control.

A.2.1. PID paralelo y serie como servomecanismos

El comportamiento de los controladores serie y paralelo es muy similar ante cambios en la consigna, ver figura 9, la única diferencia es que el PID paralelo presenta un pequeño sobreimpulso de 0.3031 cms, en cambio el PID serie no produjo sobreimpulso.

Como se observa la adición del término Uo en la figura 10, adiciona a las respuestas del sistema un sobreimpulso en la entrada de 3.9402 para el paralelo y 2.6968 para el serie, además atenúa el sobreimpulso de la señal de salida inicial del controlador paralelo (disminuye de 0.3031 a 0.0838). En la figura 11 se aprecia la saturación del PID serie y paralelo ante el mayor cambio del set point, con la adición del parámetro Uo, se atenuan los esfuerzos de control "bruscos", figura 12, evitando la saturación del actuador.

Figura 9. VC PID paralelo y serie como servomecanismos sin Uo

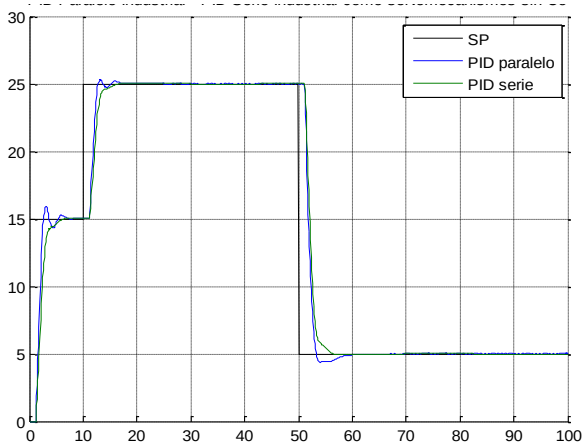


Figura 10. VC PID paralelo y serie como servomecanismos con Uo

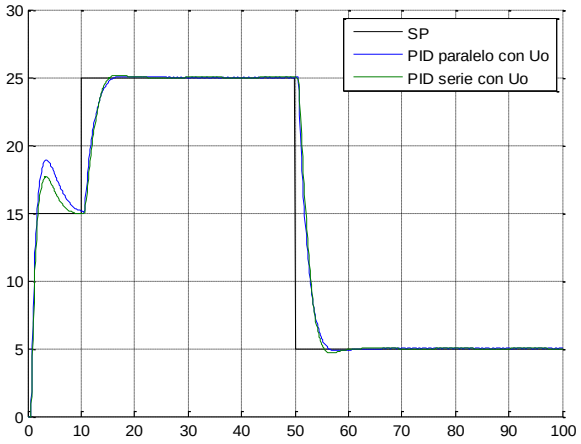


Figura 11. EC PID serie y Paralelo sin Uo

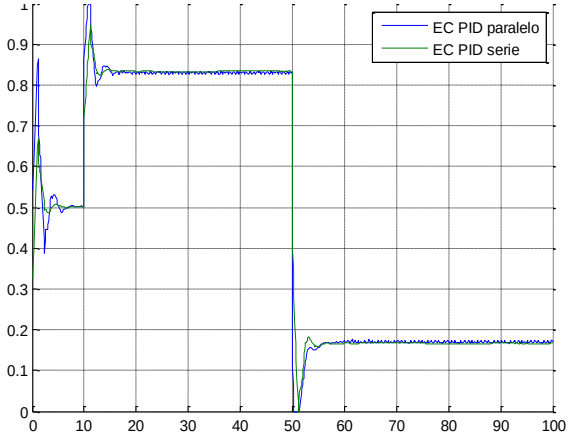
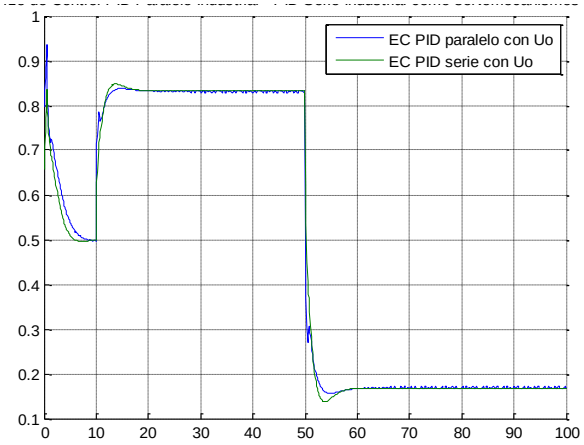


Figura 12. EC PID serie y Paralelo con Uo



A.2.2. PID paralelo y serie como reguladores

La adición del parámetro Uo, beneficia el comportamiento del PID como regulador y serie, ya que aumenta en gran proporción la capacidad de rechazo de las perturbaciones. En el PID paralelo las oscilaciones presentadas inicialmente, ver figura 13, se transforman en oscilaciones suaves pero con sobreimpulso mayor, figura 14. En el PID serie el rechazo de la perturbación se realiza con mayor rapidez, pero esto a su vez genera un sobreimpulso, figura 14. En cuanto a los esfuerzos de control, en el control pid paralelo y serie se presentan transiciones con oscilaciones mas suaves, figura 16, en comparación con las presentadas sin la adición del parametro, figura 15.

Figura 13. VC PID paralelo y serie como Reguladores sin Uo

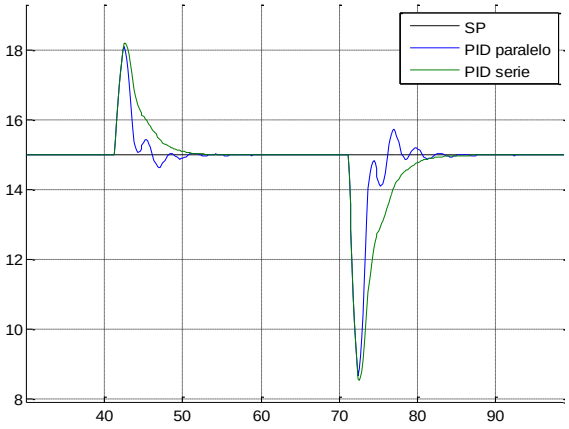


Figura 14. VC PID paralelo y serie como Reguladores con Uo

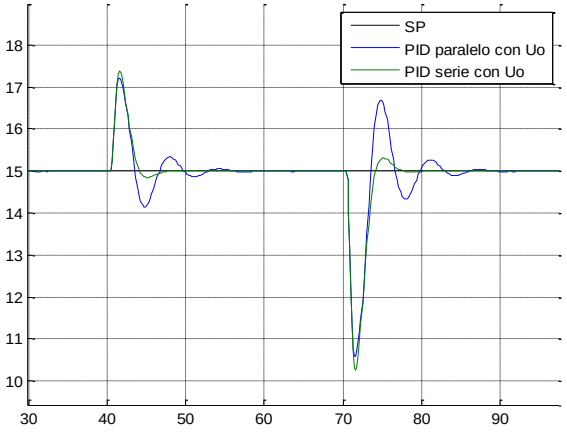


Figura 15. EC PID serie y Paralelo sin Uo

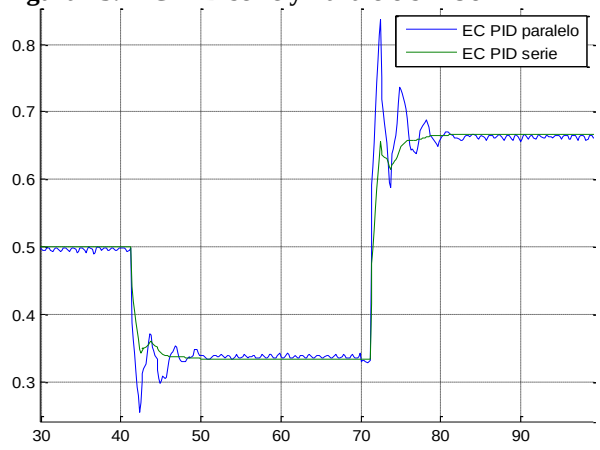
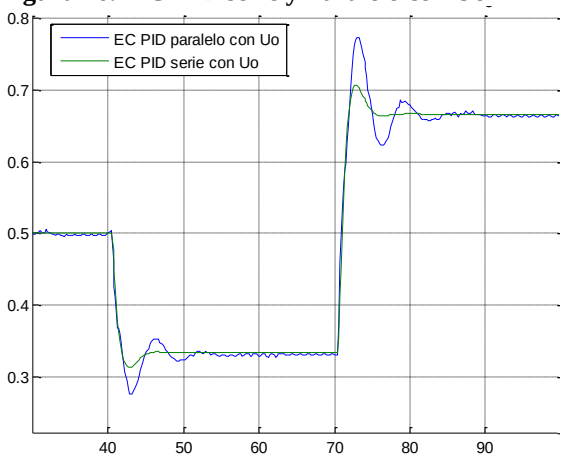


Figura 16. EC PID serie y Paralelo con Uo



A.3. CONCLUSIONES

A nivel de resultados de la simulacion se notan diferencias en la adición del parametro bias (Uo) dentro de la estrategia de control PID tanto serie como paralelo. Los parametros que se ven beneficiados por su adicion son las oscilaciones en el transitorio, y los tiempos de establecimiento.

Una de las ventajas con la adición de este parametro es la disminucion del esfuerzo de control, al disminuir la cantidad de oscilaciones de la señal de control.

El PID paralelo es la estructura que más se beneficia con la adición del parámetro Uo, ya que presenta algunos inconvenientes en su respuesta ante cambios en el valor de consigna y rechazo a perturbaciones lo cual hace que su esfuerzo de control sea mayor, ver figura 15. El Uo limita la cantidad de oscilaciones, permitiendo transiciones suaves, ver figura 16.

De lo anterior se concluye que se debe tener en cuenta cuando se implementa un sistema de control PID, la adición del parametro Uo, ya que este mejora las transiciones en la respuesta del sistema.

ANEXO B EXPERIMENTO MÉTODOS DE SINTONIZACIÓN PID SERIE Y PARALELO

Este experimento consiste seleccionar los mejores métodos de sintonización existentes para controladores PID de estructura serie y paralela cuando el lazo de control funciona como servomecanismo (cambios en el valor de consigna) y como regulador (perturbaciones en la carga).

Inicialmente en la sección B1 se presenta la descripción de los métodos de sintonización existentes mas utilizados con base en la investigación de diferentes autores, seguidamente en la sección B2 se hace una selección de los métodos usados y diseñados para cada estructura del PID y en la sección B3 se realiza la simulación de los metodos seleccionados y análisis de los resultados.

B.1. MÉTODOS DE SINTONIZACIÓN DE CONTROLADORES PID

La sintonización de los controladores PID se realiza diferentes métodos y técnicas generalmente empíricas, los cuales se basan principalmente de la respuesta temporal del proceso o con base en la función de transferencia dada en (1).

$$G(s) = \frac{Ke^{-Ls}}{Ts+a} \quad \text{Con a>0} \tag{1}$$

Donde K es la ganancia de la planta, T la constante de tiempo y L el retardo

Para sintonizar un controlador tipo PID partiendo de la dinámica de la planta, se deben realizar generalmente las siguientes etapas.

- Realizar un experimento para determinar los parámetros dinamicos: K, T y L.
- Calcular con formulas apropiadas, la ganancia proporcional Kp, el tiempo integral Ti y el tiempo derivativo Td del controlador a partir de los parámetros hallados.

B.1.1. Ajuste de los parámetros del PID

En la literatura existen variados métodos de sintonización algunos definidos para determinadas estructuras de controladores PID y para diferentes aplicaciones en cuanto a mejorar la robustez por rechazo al ruido y perturbaciones y seguimiento del

valor de consigna. La forma principal empleada en la teoría para representar dichos controladores es conocida como la forma Ideal o estandar. También se le conoce como no interactuante porque el tiempo integral Ti no influye en la parte derivativa, así como el tiempo derivativo Td no influye con la parte integral. La representación en el dominio de la frecuencia se aprecia enla ecuación 2.

$$G_{cPIDIdeal}(s) = K_c\left(1+\frac{1}{sT_i}+sT_d\right) \tag{2}$$

Donde Kc es la componente proporcional, Ti es el tiempo integral y Td es el tiempo derivativo. Su representación en el dominio del tiempo esta dado por (3)

$$u(t) = K_c\left[e(t)+\frac{1}{T_i}\int_0^t e(t)+T_d\frac{de(t)}{dt}\right] \tag{3}$$

Donde e(t) es el error.

Existen otros dos tipos de configuraciones básicas que son la representación Serie (ecuación 4) y Paralela (ecuación 5).

$$G_{cPIDSerie}(s) = K_c^{'}\left(1+\frac{1}{T_i^{'}}\right)\left(1+T_d^{'}s\right) \tag{4}$$

$$G_{cPIDParalelo}(s) = \left(K_p+\frac{K_i}{s}+K_d s\right) \tag{5}$$

El controlador PID paralelo posee una ganancia independiente para la acción proporcional Kp, para la integral Ki y para la diferencial Kd. Los parámetros están relacionados con la forma ideal a través de la siguiente conversión:

$$\begin{aligned} K_p &= K_c \\ K_i &= \frac{K_c}{T_i} \\ K_d &= K_c T_d \end{aligned} \tag{6}$$

El controlador interactuante se puede representar siempre como no interactuante, por medio de la conversión de parámetros que se presenta en (7)

$$K_c = K_c^{'} \frac{T_i^{'} + T_d^{'}}{T_i^{'}}$$
$$T_i = T_i^{'} + T_d^{'} \qquad (7)$$
$$T_d = \frac{T_i^{'} \cdot T_d^{'}}{T_i^{'} + T_d^{'}}$$

El controlador interactuante que corresponde al no interactuante puede ser encontrado solo si:

$$T_i \geq 4T_d \qquad (8)$$

Luego,

$$K_c^{'} = \frac{K_c}{2}\left(1 + \sqrt{1 - \frac{4T_d}{T_i}}\right)$$
$$T_i^{'} = \frac{T_i}{2}\left(1 + \sqrt{1 - \frac{4T_d}{T_i}}\right) \qquad (9)$$
$$T_d^{'} = \frac{2T_d}{1 + \sqrt{1 - \frac{4T_d}{T_i}}}$$

B.2. MÉTODOS DE SINTONIZACIÓN CONTROLADORES PID

La mayoría de los métodos encontrados en la literatura parten del método realizado por *Zieger-Nichols* mediante la respuesta temporal del sistema.

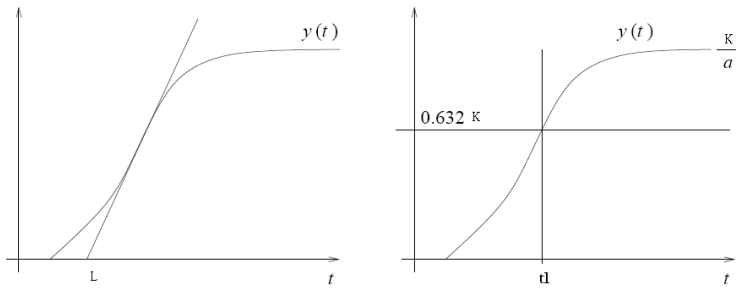
B.2.1. Métodos de Respuesta Temporal

Si se supone que G(s) es una función de transferencia valida para una planta, el método de respuesta temporal consiste en aplicar un escalón unitario a la planta y medir su respuesta temporal y(t) para, a partir de ella, calcular los parámetros K, T y L de su supuesto modelo.

Como se aprecia en la figura 17 se ha medido la respuesta al escalón y(t) de la planta, primero para calcular T, una opción es trazar una línea tangente a la curva y(t) en el punto de máxima pendiente y hallando su intersección con el eje t, se obtiene el valor del retardo L del modelo G(s) de la planta. Si la planta real siguiera exactamente el modelo supuesto la máxima pendiente de y(t) se daría en el punto de intersección de la curva con el eje t. Pero, como no va a ser así, se hace una aproximación que consiste en sustituir la parte baja de la curva por la tangente de pendiente máxima. Para hallar K y T, se sabe que el valor de la respuesta $y_0(t)$ a un escalón unitario de la

planta con retardo nulo en el instante τ= 1/a (constante de tiempo) es $y_o(\tau) = K(1 - e^{-at})|_{t=1/a} = 0.632K$

Figura 17. Parámetros K, T y L de una planta de primer orden.



Por tanto, la respuesta al escalón unitario de la planta con retardo T en el instante L+T, valdrá también y(L+T) = 0.632A. Así que para determinar la constante de tiempo T se traza una recta horizontal de ordenada igual a 0.632K, hallamos el punto de intersección de la recta con la curva y(t) y entonces la vertical por dicho punto marca el valor t1 = L+T. Por tanto,

$$T = t1 - L \tag{10}$$

Por ser la entrada un escalón unitario, el valor final de la respuesta o valor en estado estacionario es igual a la ganancia estática de la planta.

$$y_{ss} = \lim_{t\to\infty} y(t) = \frac{K}{a} \tag{11}$$
$$K = ay_{ss}$$

El valor

$$R_r = \frac{K}{T}$$

Rr es la tasa de reacción (*reaction rate*), y es igual a la pendiente máxima de la respuesta. El principal inconveniente del método de lazo abierto es que el experimento que da la respuesta temporal de la planta exige controlar en lazo abierto la planta durante un tiempo suficiente para que la respuesta llegue al estado estacionario y esto puede ser exagerado en ciertos procesos industriales en funcionamiento.

Entre los métodos de lazo cerrado se encuentra el realizado por primera vez por Zieger-Nichols, utilizando un controlador solamente proporcional y mediante un proceso reiterado se aumenta la ganancia hasta que la respuesta y(t) sea oscilante, con ciclos de amplitud mantenida, es decir, de aspecto sinusoidal. Entonces los valores de Ku y Tu son la ganancia y el período últimos de oscilación. Aunque este método es más factible que el anterior, también puede resultar a veces inseguro

debido a que hay procesos que no toleran oscilaciones mantenidas por mucho tiempo y hay otros, muy lentos, que aunque las admitan, pueden exigir un tiempo de experimento demasiado largo. Hay otros métodos, también de lazo cerrado, que se basan en el funcionamiento normal del controlador para hacer los cálculos y, además, si se ponen en modo automático son capaces de ponerse ellos mismos los valores mas adecuados a las circunstancias. Son los controladores inteligentes o auto sintonizables.

Los métodos que se han desarrollado partiendo de la respuesta de reacción del proceso están.

El método de Ziegler-Nichols, Shinskey y Cohen Coon. Suponen a=0 en el modelo G(s) de la planta, y el de Cohen-Coon que supone a≠ 0 (Ziegler and Nichols, 1943; Cohen and Coon, 1953). Enla tabla 5 se presentan las ecuaciones de sintonización para cada método.

Tabla 5. Expresiones para calcular los parámetros de un PID

Tipo		Zeigler-Nichols (lazo cerrado)	*Shinskey* (lazo cerrado)	Zeigler-Nichols (lazo abierto)	Cohen-Coon (lazo abierto)
PID	K_p	$0.6K_u$	$0.5K_u$	$\frac{1.2}{R_r \mathrm{L}}$	$\frac{\tau}{K\mathrm{L}}(1.35+0.27\frac{\mathrm{L}}{\tau})$
	T_d	$0.5K_u$	$0.34K_u$	$2\mathrm{L}$	$(\frac{2.5+0.5\mathrm{L}/\tau}{1+0.6\mathrm{L}/\tau})$
	T_i	$0.125K_u$	$0.08K_u$	$0.5\mathrm{L}$	$(\frac{0.37}{1+0.2\mathrm{L}/\tau})$

Métodos Chien *et al (Hrones-Rewick).* Estos autores ofrecen los parámetros que consiguen sobre-impulsos del 20% o del 0 %, tanto ante entradas referencia, como ante entrada perturbación (Chien at al, 1952). Este método se basó en la curva de reacción del proceso y se diseñó para un controlador PID Ideal, las ecuaciones del método se encuentran en la tabla 6.

Tabla 6. Ajuste de Chien-Hrones-Rewick.

Regla	Kp	Ti	Td	Comentarios
Regulador	$\frac{0.95\mathrm{T_m}}{\mathrm{K_m}\tau_\mathrm{m}}$	$2.38\tau_\mathrm{m}$	$0.42\tau_m$	0% sobreimpulso; $0.11<\frac{\tau_\mathrm{m}}{\mathrm{T_m}}<1$
	$\frac{1.2\mathrm{T_m}}{\mathrm{K_m}\tau_\mathrm{m}}$	$2\tau_\mathrm{m}$	$0.42\tau_\mathrm{m}$	20% sobreimpulso; $0.11<\frac{\tau_\mathrm{m}}{\mathrm{T_m}}<1$
Servomecanismo	$\frac{0.6\mathrm{T_m}}{\mathrm{K_m}\tau_\mathrm{m}}$	$\mathrm{T_m}$	$0.5\tau_\mathrm{m}$	0% sobreimpulso; $0.11<\frac{\tau_\mathrm{m}}{\mathrm{T_m}}<1$

	$\frac{0.95T_m}{K_m\tau_m}$	$1.36T_m$	$0.47\tau_m$	20% sobreimpulso; $0.11 < \frac{\tau_m}{T_m} < 1$

Método de Dahlin Lazo Abierto Estándar. Se basa en el método de la curva de reacción del proceso para obtener la función de transferencia del mismo, en donde T es la constante de tiempo de la respuesta del proceso, *L* es el tiempo muerto, y *K* es la ganancia del proceso; partiendo de ella se pueden obtener las constantes del controlador PID. En la tabla 7, Dahlin propone unos parámetros de ajuste de controladores de acuerdo al tipo de proceso al cual se le introducirá el controlador (Camacho, 2005).

Tabla 7. Parámetros de sintonización de un controlador por el Método de Dahlin.

Controlador	Parámetro de ajuste	Ecuación
PID	Kp	$\frac{1.2}{2K}\left(\frac{L}{T}\right)^{-1}$
	Ti	T
	Td	$\frac{L}{2}$

Método Kappa-Tau. Tomando como base un conjunto amplio de plantas de prueba, utilizaron la sensibilidad máxima como parámetro de diseño empleando técnicas basadas en la localización de los polos dominantes, para determinar los valores de los parámetros de controladores PI y PID de dos grados de libertad (Gómez de La Riva, 2006) [4]. Este método esta considerado dentro de los basados en márgenes de robustez.

Con este método se puede optar entre robustez y eficacia de regulación por medio del parámetro Ms.

Ms = 1,4 Mayor robustez, Ms = 2 Mayor rapidez en la respuesta.

La estructura del PID utilizado por Aström y Hägglund (autores) es:

$$u(s) = K_c \times \left(b\, r(s) - y(s) + \frac{1}{T_i\, s} \times \big(r(s) - y(s)\big) - T_d s\, y(s) \right) \qquad (12)$$

Donde: KC es la componente proporcional, Ti es el tiempo integral, TD es el tiempo derivativo. r(s) es el punto de consigna, y(s) la variable controlada, u(s) la salida del regulador, b es el peso de la consigna. Si b = 0 no hay acción proporcional en la consigna. Si b = 1 no se modifica la acción proporcional; éste parámetro es muy

interesante en reguladores esclavos en una cascada. La forma de sintonizar los parámetros del PID se obtiene a partir de una serie de tablas siguiendo los siguientes pasos:

Se definen las variables α = K·L/T y τ = L/(L+T), donde los valores de K, L y T se conocen del modelo de función de transferencia obtenido previamente.
Se toman de una serie de tablas los valores α ·KC, Ti/L, TD/L y b. Estos valores vienen dados por funciones del tipo $f(\tau)=a_0\times e^{(a_1\times\tau+a_2\times\tau^2)}$

Los coeficientes a0, a1 y a2 de la expresión anterior para un regulador PID se muestran en la tabla 8.

Tabla 8. Cálculo de parámetros de un PID según el método KAPPA-TAU.

	M_s = 1,4			M_s = 2		
	a_0	a_1	a_2	a_0	a_1	a_2
$\alpha\cdot K_c$	3,8	-8,4	7,3	8,4	-9,6	9,8
T_i / L	5,2	-2,5	-1,4	3,2	-1,5	-0,93
T_d / L	0,89	-0,37	-4,1	0,86	-1,9	-0,44
b	0,4	0,18	2,8	0,22	0,65	0,051

B.2.2. Métodos como regulador

Cuando el control PID opera como regulador se dice que actúa como un control que trata de minimizar la sensibilidad ante las perturbaciones que afectan al proceso. Entre los métodos de lazo abierto en donde usualmente se aplica a una planta de primer orden más tiempo muerto (POMTM). Así como se explica en (Alfaro, 2002) los más utilizados son los siguientes:

Método de *Ziegler* y *Nichols.* El criterio de desempeño que seleccionaron fue el de un decaimiento de *1/4,* o sea que el error decae en la cuarta parte de un periodo de oscilación. Las ecuaciones mostradas en la tabla 13, fueron determinadas de forma empírica a partir de pruebas realizadas en laboratorio con diferentes procesos, y están basadas en un modelo de primer orden más tiempo muerto (POMTM) identificado por el método de la tangente, para un funcionamiento del lazo de control como regulador con un controlador PID-Ideal (Ziegler and Nichols, 1943).

$$K_c = 1.2\frac{\tau}{k_p t_m} \quad \text{a} \quad 2.0\frac{\tau}{k_p t_m}$$
$$T_i = 2t_m \qquad (13)$$
$$T_d = 0.5t_m$$

Método de *Cohen – Coon.* Cohen y Coon introdujeron un índice de auto regulación definido como μ = *tm*/τ y plantearon nuevas ecuaciones de sintonización. Estas se basan el criterio de desempeño de decaimiento de *1/4* con sobrepaso mínimo, y con mínima área bajo la curva de respuesta, se aplicó en un controlador PID-Ideal (Cohen and Coon, 1953). Las ecuaciones obtenidas se presentan en (14).

$$K_c = \frac{\tau}{k_p t_m}\left(\frac{4}{3} + \frac{t_m}{4\tau}\right) \qquad T_i = t_m\left(\frac{32 + 6t_m/\tau}{13 + 8t_m/\tau}\right) \qquad T_d = t_m\left(\frac{4}{11 + 2t_m/\tau}\right) \qquad (14)$$

Método de *López, Miller, Smith* y *Murril.* El primer método basado en criterios integrales que presentó ecuaciones para el cálculo de los parámetros del controlador es conocido como el método de López (Lopez et al, 1967). Definiendo una función de costo de la forma

$$\Phi = \int_0^\infty F(e(t), t)dt \qquad (15)$$

Donde F es una función del error y del tiempo, se obtiene un valor que caracteriza la respuesta del sistema. Entre menor sea el valor de Φ, mejor será el desempeño del sistema de control, por ello, un desempeño óptimo se obtiene cuando Φ es mínimo.

$$K_c k_p = a\left(\frac{t_m}{\tau}\right)^b \qquad \frac{T_i}{\tau} = \frac{1}{c}\left(\frac{t_m}{\tau}\right)^{-d} \qquad \frac{T_d}{\tau} = e\left(\frac{t_m}{\tau}\right)^f \qquad (16)$$

Los para metros de sintonización para del método de López se presentan en la tabla 9.

Tabla 9. Parámetros de las ecuaciones del método de López.

	a	b	c	d	e	f
IAE	1.435	-0.921	0.878	-0.749	0.482	1.137
ITAE	1.357	-0.947	0.842	-0.738	0.381	0.995
ISE	1.495	-0.945	1.101	-0.771	0.560	1.006

Los criterios de desempeño utilizados por López fueron: Integral del error absoluto (IAE), Integral del error absoluto por el tiempo (ITAE) e Integral del error cuadrático (ISE). Se utilizaron en lazos de control que funcionan como reguladores con un controlador PID-Ideal.

Método de *Kaya* and *Sheib.* Kaya and Sheib realizaron lo mismo que López, pero para controladores que denominaron PID-Clásico (PID-Serie), PID No Interactuante y PID-Industrial (Kaya and Sheib, 1988). Los valores de las constantes son:

Tabla 10. Constantes para el método de Kaya and Sheib.

Controlador clásico - Regulador						
	a	***b***	***c***	***d***	***e***	***f***
IAE	0.98089	-0.76167	0.91032	-1.05211	0.59974	0.89819
ITAE	0.77902	-1.06401	1.14311	-0.70949	0.57137	1.03826
ISE	1.11907	-0.89711	0.79870	-0.95480	0.54766	0.87798
Controlador no interactuante - Regulador						
IAE	1.31509	-0.88260	1.25870	-1.37560	0.56550	0.45760
ITAE	1.31760	-0.79370	1.12499	-1.42603	0.49547	0.41932
ISE	1.34660	-0.93080	1.65850	-1.25738	0.79715	0.41941
Controlador industrial - Regulador						
IAE	0.91000	-0.79380	1.01495	-1.00403	0.54140	0.78480
ITAE	0.70580	-0.88720	1.03326	-0.99138	0.60006	0.97100
ISE	1.11470	-0.89920	0.93240	-0.87530	0.56508	0.91107

Método de *Sung, O, Lee, Lee* y *Yi.* Sung, basó su procedimiento de sintonización en un modelo de segundo orden más tiempo muerto (SOMTM), identificado mediante la realización de una prueba con realimentación por relé, seguido por una constante de control *P*. El criterio de desempeño corresponde a la minimización de la ITAE y el controlador un PID-Ideal. El ajuste de las ecuaciones la realizaron para 0,05 ≤ *tm*/τ ≤ 2,0 (Sung, 1996). Las ecuaciones obtenidas se presentan en (17).

$$
\begin{aligned}
K_c k_p &= -0.67 + 0.297\left(\frac{t_m}{\tau}\right)^{-2.001} + 2.189\left(\frac{t_m}{\tau}\right)^{-0.766}\varsigma && , \ \frac{t_m}{\tau} \le 0.9 \\
K_c k_p &= -0.365 + 0.260\left(\frac{t_m}{\tau} - 1.4\right)^{2.0} + 2.189\left(\frac{t_m}{\tau}\right)^{-0.766}\varsigma && , \ \frac{t_m}{\tau} \ge 0.9 \\
\frac{T_i}{\tau} &= 2.212\left(\frac{t_m}{\tau}\right)^{0.520} - 0.3 && , \ \frac{t_m}{\tau} < 0.4 \\
\frac{T_i}{\tau} &= -90.975 + 0.91\left(\frac{t_m}{\tau} - 1.845\right)^2 + \left[1 - EXP\left(\frac{-\varsigma}{0.15 + 0.33(t_m/\tau)}\right)\right]\left[5.25 - 0.88\left(\frac{t_m}{\tau} - 2.8\right)^2\right] && , \ \frac{t_m}{\tau} \ge 0.4 \\
\frac{\tau}{T_d} &= -1.9 + 1.576\left(\frac{t_m}{\tau}\right)^{-0530} + \left[1 - EXP\left(\frac{-\varsigma}{-0-15 + 0.939(t_m/\tau)^{-1.121}}\right)\right]\left[1.45 + 0.969\left(\frac{t_m}{\tau}\right)^{-1.171}\right]
\end{aligned}
\qquad (17)
$$

Método de *Zhuang and Atherton.* Este método se basa en el concepto de la integral del cuadrado del tiempo por el cuadrado del error y en la integral del cuadrado del error y esta diseñado para controladores PID ideales que trabajan como reguladores (Zhuang and Atherton, 1993).

Tabla 11. Ecuaciones para el método de Zhuang and Atherton.

Regla	Kp	Ti	Td	Comentarios
ISTSE	$\frac{1.468}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.970}$	$\frac{T_m}{0.942}\left(\frac{T_m}{\tau_m}\right)^{0.725}$	$0.443T_m\left(\frac{\tau_m}{T_m}\right)^{0.939}$	$0.1 \le \frac{\tau_m}{T_m} \le 1.0$
	$\frac{1.515}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.730}$	$\frac{T_m}{0.957}\left(\frac{T_m}{\tau_m}\right)^{0.598}$	$0.444T_m\left(\frac{\tau_m}{T_m}\right)^{0.847}$	$1.1 \le \frac{\tau_m}{T_m} \le 2.0$
	$\frac{1.531}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.960}$	$\frac{T_m}{0.971}\left(\frac{T_m}{\tau_m}\right)^{0.746}$	$0.413T_m\left(\frac{\tau_m}{T_m}\right)^{0.933}$	$0.1 \le \frac{\tau_m}{T_m} \le 1.0$
	$\frac{1.592}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.705}$	$\frac{T_m}{0.957}\left(\frac{T_m}{\tau_m}\right)^{0.597}$	$0.414T_m\left(\frac{\tau_m}{T_m}\right)^{0.850}$	$1.1 \le \frac{\tau_m}{T_m} \le 2.0$
ISE	$\frac{1.473}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.970}$	$\frac{T_m}{1.115}\left(\frac{T_m}{\tau_m}\right)^{0.753}$	$0.55T_m\left(\frac{\tau_m}{T_m}\right)^{0.948}$	$0.1 \le \frac{\tau_m}{T_m} \le 1.0$
	$\frac{1.524}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.735}$	$\frac{T_m}{1.130}\left(\frac{T_m}{\tau_m}\right)^{0.641}$	$0.552T_m\left(\frac{\tau_m}{T_m}\right)^{0.851}$	$1.1 \le \frac{\tau_m}{T_m} \le 2.0$

Entre los métodos de lazo cerrado en donde el controlador opera automáticamente produciendo un cambio en el valor deseado y se obtiene información del comportamiento dinámico del sistema para identificar un modelo de orden reducido para el proceso, o de las características de la oscilación sostenida del mismo, para utilizarla en el cálculo de los parámetros del controlador, se encuentran los siguientes

El primer método de sintonización a lazo cerrado fue propuesto por *Zieger-Nichols* los cuales presentaron ambos métodos en la misma publicación (Ziegler and Nichols, 1943); utilizaron un controlador puramente proporcional y mediante un proceso iterativo, el procedimiento requiere aumentar paulatinamente la ganancia del mismo hasta lograr que el sistema entre en una oscilación sostenida ante un cambio del escalón en el valor deseado. La ganancia en este punto es la ganancia última Kcu y el periodo de la oscilación, el periodo último Tu. Para el ajuste proporcional seleccionaron, el decaimiento de 1/4 como un compromiso entre el error permanente y el decaimiento, y encontraron que la ganancia proporcional para un controlador *P* debería ser la mitad de la ganancia última. Las ecuaciones obtenidas son.

$$K_c = 0,6K_{cu} \text{ a } 1,0K_{cu}$$
$$T_i = 0,5\ T_u \qquad (18)$$
$$T_d = 0,125T_u$$

Existen métodos adicionales de sintonización a lazo cerrado que no son muy utilizados y se basan en el método anterior y no solo por ser el primero sino porque aún se emplea en su forma original. Entre lo mas destacados estan: (Shinskey, 1971) ajusta las constantes de las ecuaciones de sintonización de *Ziegler-Nichols* para minimizar el criterio *IAE*; (Chidambara, 1970) utiliza un procedimiento iterativo basado en las mismas ecuaciones de *Ziegler-Nichols*, pero evitando tener que llevar el sistema al límite de la estabilidad; mientras que (Aström and Hägglund, 1988) lo adaptan para la sintonización de controladores PI y PID de dos grados de libertad.

B.2.3. Métodos como servomecanismo

Al igual que los métodos como regulador, la sintonización del PID como servomecanismo parte del método propuesto por *Ziegler-Nichols.* Los procedimientos de sintonización de lazo abierto utilizan un modelo de POMTM o SOMTM, obtenido generalmente a partir de la curva de reacción del proceso, así como se explica en (Alfaro, 2003), los métodos en lazo abierto más utilizados se presentan a continuación.

Método de *Rovira, Murrill* y *Smith.* Se basaron en el método de López para controladores PID que operan como reguladores. Conocido como el método de *Rovira*, consiste en definir una función de coste (ecuación 15), por medio de dicha función se obtiene un valor que caracteriza la respuesta del sistema. Entre menor sea el valor de Φ, mejor será el desempeño del sistema de control por lo que un desempeño optimo se obtiene cuando Φ sea mínimo (Rovira et al, 1969). Los criterios de desempeño utilizados por Rovira fueron IAE y e ITAE, para lazos de control que funcionan como servomecanismos, el controlador usado es un PID Ideal.

$$K_c k_p = a\left(\frac{t_m}{\tau}\right)^b \quad \frac{T_i}{\tau} = \frac{1}{c + d(t_m/\tau)} \quad \frac{T_d}{\tau} = e\left(\frac{t_m}{\tau}\right)^f \qquad (19)$$

Donde,Kp es la ganancia, t_m es el tiempo muerto, τ es la constante de tiempo, todas constantes dinamicas del proceso.

Los parámetros para este método se aprecian en la tabla 12.

Tabla 12. Parámetros método de Rovira

	a	***b***	***c***	***d***	***e***	***f***
IAE	1.435	-0.921	0.878	-0.749	0.482	1.137
ITAE	1.357	-0.947	0.842	-0.738	0.381	0.995
ISE	1.495	-0.945	1.101	-0.771	0.560	1.006

Método de *Kaya and Scheib.* Utiliza los mismos criterios de análisis que el método de Rovira, las diferencias están en que este método se aplicó a controladores de estructura serie (clásico), PID no interactuante (standar) y el PID industrial, además los parámetros de las ecuaciones son diferentes (Kaya and Scheib, 1988). Los parámetros de las ecuaciones de este método se observan en tabla 13.

Tabla 13. Parámetros método de Kaya and Scheib

Controlador clásico - Servomecanismo						
	a	***b***	***c***	***d***	***e***	***f***
IAE	0.65000	-1.04432	0.98950	0.09539	0.50814	1.08433
ITAE	1.12762	-0.80368	0.99783	0.02860	0.42844	1.00810
ISE	0.71959	-1.03092	1.12666	-0.18145	0.54568	0.86411
Controlador no interactuante - Servomecanismo						
IAE	1.13031	-0.81314	5.75270	-5.72410	0.32175	0.17707
ITAE	0.98384	-0.49851	2.71348	-2.29778	0.21443	0.16768
ISE	1.26239	-0.83880	6.03560	-6.01910	0.47617	0.24572
Controlador industrial - Servomecanismo						
IAE	0.81699	-1.00400	1.09112	-0.22387	0.44278	0.97186
ITAE	0.83260	-0.76070	1.00268	0.00854	0.44243	1.11499
ISE	1.14270	-0.93650	0.99223	-0.35269	0.35308	0.78088

Método de *Sung*, *O*, *Lee*, *Lee* y *Yi.* Los autores de este método se basaron en la sintonización de un modelo de SOMTM, identificado mediante la realización de una prueba con realimentación por relé seguida por una connstante de control P. El criterio de desempeño corresponde a la minimización de la ITAE y el controlador usado es un PID Ideal. El ajuste de las ecuaciones la realizaron para $0.005 \leq t_m/T \leq 2.0$ (Sung et al, 1996). Las ecuaciones obtenidas se aprecian en (20)

$$K_c k_p = -0{,}04 + \left[0{,}333 + 0{,}949\left(\frac{t_m}{\tau}\right)^{-0.983}\right]\varsigma \quad ,\varsigma \le 0{,}9$$

$$K_c k_p = -0{,}544 + 0{,}308\left(\frac{t_m}{\tau}\right) + 1{,}408\left(\frac{t_m}{\tau}\right)^{-0.832}\varsigma \quad ,\varsigma > 0{,}9$$

$$\frac{T_i}{\tau} = \left[2{,}055 + 0{,}072\left(\frac{t_m}{\tau}\right)\right]\varsigma \quad ,\frac{t_m}{\tau} \le 1 \qquad (20)$$

$$\frac{T_i}{\tau} = \left[1{,}768 + 0{,}329\left(\frac{t_m}{\tau}\right)\right]\varsigma \quad ,\frac{t_m}{\tau} > 1$$

$$\frac{\tau}{T_d} = \left[1 - EXP\left(\frac{-(t_m/\tau)^{0.060}\varsigma}{0{,}870}\right)\right]\left[0{,}55 + 1{,}683\left(\frac{t_m}{\tau}\right)^{-1{,}090}\right]$$

Método de *Wang et al.* Ese método se basa en el mismo criterio de *Kaya and Scheib* y es aplicado a un controlador ideal actuando como servomecanismo (Wang et al, 1995). En la tabla 14 se aprecian las formulas de sintonización obtenidas.

Tabla 14. Parámetros método de Wang et al.

Regla	Kp	Ti	Td	Comentario
IAE	$\frac{\left(0.7645 + \frac{0.6032}{\tau_m/T_m}\right)(T_m + 0.5\tau_m)}{K_m(T_m + \tau_m)}$	$T_m + 0.5\tau_m$	$\frac{0.5T_m\tau_m}{T_m + 0.5\tau_m}$	$0.05 < \frac{\tau_m}{T_m} < 6$
ISE	$\frac{\left(0.9155 + \frac{0.7524}{\tau_m/T_m}\right)(T_m + 0.5\tau_m)}{K_m(T_m + \tau_m)}$	$T_m + 0.5\tau_m$	$\frac{0.5T_m\tau_m}{T_m + 0.5\tau_m}$	$0.05 < \frac{\tau_m}{T_m} < 6$
ITAE	$\frac{\left(0.7303 + \frac{0.5307}{\tau_m/T_m}\right)(T_m + 0.5\tau_m)}{K_m(T_m + \tau_m)}$	$T_m + 0.5\tau_m$	$\frac{0.5T_m\tau_m}{T_m + 0.5\tau_m}$	$0.05 < \frac{\tau_m}{T_m} < 6$

Método de *Zhuang and Atherton.* Basado en la integral del cuadrado del error (ISE) y en la integral del cuadrado del tiempo por el cuadrado del error (ISTSE). Este método es aplicado a un controlador Ideal (Zhuang and Atherton, 1993).

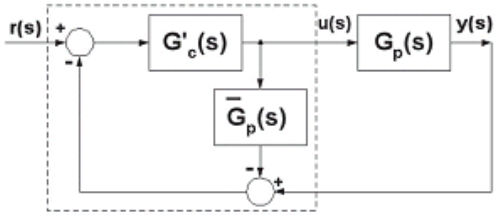
Tabla 15. Parámetros método de Zhuang and Atherton

Regla	Kp	Ti	Td	Comentario
Mínimo ISE	$\frac{1.048}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.897}$	$\frac{T_m}{1.195 - 0.368\frac{\tau_m}{T_m}}$	$0.489T_m\left(\frac{\tau_m}{T_m}\right)^{0.888}$	$0.1 \le \frac{\tau_m}{T_m} \le 1.0$
	$\frac{1.154}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.567}$	$\frac{T_m}{1.047 - 0.220\frac{\tau_m}{T_m}}$	$0.490T_m\left(\frac{\tau_m}{T_m}\right)^{0.708}$	$1.1 \le \frac{\tau_m}{T_m} \le 2.0$
Mínimo ISTSE	$\frac{1.042}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.897}$	$\frac{T_m}{0.987 - 0.238\frac{\tau_m}{T_m}}$	$0.385T_m\left(\frac{\tau_m}{T_m}\right)^{0.906}$	$0.1 \le \frac{\tau_m}{T_m} \le 1.0$
	$\frac{1.142}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.579}$	$\frac{T_m}{0.919 - 0.172\frac{\tau_m}{T_m}}$	$0.384T_m\left(\frac{\tau_m}{T_m}\right)^{0.839}$	$1.1 \le \frac{\tau_m}{T_m} \le 2.0$

Mínimo ISTES	$\frac{0.968}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.904}$	$\frac{T_m}{0.977 - 0.253\frac{\tau_m}{T_m}}$	$0.316T_m\left(\frac{\tau_m}{T_m}\right)^{0.892}$	$0.1 \le \frac{\tau_m}{T_m} \le 1.0$
	$\frac{1.061}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.583}$	$\frac{T_m}{0.892 - 0.165\frac{\tau_m}{T_m}}$	$0.315T_m\left(\frac{\tau_m}{T_m}\right)^{0.832}$	$1.1 \le \frac{\tau_m}{T_m} \le 2.0$

Métodos basados en Control por Modelo Interno (IMC). Las técnicas de control basadas en modelos incorporan dentro del controlador un modelo del proceso. Ese tipo de control se conoce como el Control por Modelo Interno (IMC por sus siglas en inglés). En la figura 18 se muestra el diagrama de bloques básico del sistema de control basado en modelo, en donde $G'_p(s)$ es un modelo de la planta, $G_p(s)$ y $G'_c(s)$ es un modelo del controlador *IMC.*

Figura 18. Modelo básico control IMC



Fuente: Tomado de (Rivera et al, 1986)l

El modelo equivalente para el controlador IMC de acuerdo a la figura 18 es:

$$G(s) = \frac{G_c'(s)}{1 - G_c'(s)\tilde{G}_p(s)} \tag{21}$$

Los mayores exponentes de la aplicación de este tipo de control son:

Método de *Rivera*, *Morari* y *Stogestad.* Partieron de la estructura general IMC y desarrollaron un procedimiento para obtener los controladores y lograr cierto desempeño deseado, demostrando que para modelos simples esta estructura conduce a controladores del tipo PID. Su aporte consistió en redefinir el controlador IMC como la ecuación 22

$$G_c'(s) = \tilde{G}_p^{-1}(s)F(s) \tag{22}$$

Donde *F(s)* es un filtro pasa bajo, el cual debe seleccionarse de manera que se garantice que la función de transferencia del controlador IMC sea propia, el filtro es de la forma.

$$F(s) = \frac{1}{(\lambda s + 1)^n} \tag{23}$$

Los resultados de su investigación conducen a la obtención de parámetros de sintonización de controladores PID-IMC para modelos sin tiempo muerto y con tiempo muerto. Para la obtención de estos últimos Rivera utilizó una aproximación de Padé de orden cero para el controlador PI y una de primer orden para el PID (Rivera et al, 1986). Las ecuaciones obtenidas se presentan en las tablas 16 y 17.

Tabla 16. Ecuaciones método Rivera para plantas sin tiempo muerto

Modelo $\tilde{G}_p(s)$	$\frac{y(s)}{r(s)}$	*Controlador* $G_c(s)$	*Parámetros* K_c	T_i	T_d
$\frac{k_p}{\tau s+1}$	$\frac{1}{\lambda s+1}$	$\frac{\tau s+1}{\lambda k_k s}$	$\frac{\tau}{\lambda k_p}$	τ	-
$\frac{k_p}{(\tau_1 s+1)(\tau_2 s+1)}$	$\frac{1}{\lambda s+1}$	$\frac{(\tau_1 s+1)(\tau_2 s+1)}{\lambda k_p s}$	$\frac{\tau_1+\tau_2}{\lambda k_p}$	$\tau_1+\tau_2$	$\frac{\tau_1\tau_2}{\tau_1+\tau_2}$
$\frac{k_p}{\tau^2 s^2+2\delta s+1}$	$\frac{1}{\lambda s+1}$	$\frac{\tau^2 s^2+2\delta s+1}{\lambda k_p s}$	$\frac{2\varsigma\tau}{\lambda k_p}$	$2\varsigma\tau$	$\frac{\tau}{2\varsigma}$
$\frac{k_p}{s}$	$\frac{1}{\lambda s+1}$	$\frac{1}{\lambda k_p}$	$\frac{1}{\lambda k_p}$	-	-
$\frac{k_p}{s(\tau s+1)}$	$\frac{1}{\lambda s+1}$	$\frac{2\lambda s+1}{\lambda^2 k_p s}$	$\frac{1}{\lambda k_p}$	-	τ

Tabla 17. Parámetros método Rivera para plantas con tiempo muerto

Controlador	K_c	T_i	T_d	λ/t_m recomendado $(<0.1\tau/t_m)$
PID	$\frac{2\tau+t_m}{k_p(2\lambda+t_m)}$	$\tau+\frac{t_m}{2}$	$\frac{\tau t_m}{2\tau+t_m}$	>0.8
PI	$\frac{\tau}{\lambda k_p}$	τ	-	>1.7
PI método mejorado (*)	$\frac{2\tau/t_m}{2\lambda k_p}$	$\tau+\frac{t_m}{2}$	-	>1.7

(*) El tiempo muerto de la planta no fue considerado en el método original para la sintonización del PI, pero si en el método mejorado.

Método de *Brosilow.* *En* este método, *Brosilow* utilizó un procedimiento basado en la aproximación de la función de transferencia del controlador por una serie de *Maclaurin* para derivar las ecuaciones en el cálculo de los parámetros del controlador *PID-IMC* (Brosilow, 1992). Si la planta es de POMTM, como:

$$G_{lc}(s) = \frac{K_p e^{-t_m s}}{\tau s + 1} \tag{24}$$

Y la función de transferencia de lazo cerrada deseada es

$$G_{lc}(s) = \frac{e^{-t_m s}}{\lambda s + 1} \tag{25}$$

Es decir, debe ser con ganancia unitaria (error permanente cero) y un parámetro ajustable que permita variar la velocidad de respuesta del sistema, los parámetros de ajuste del control PID IMC son:

$$\begin{aligned} Kc &= \frac{\tau + t_m^2/[2(\lambda + t_m)]}{k_p(\lambda + t_m)} = \frac{T_i}{k_p(\lambda + t_m)} \\ Ti &= \tau + \frac{t_m^2}{2(\lambda + t_m)} \\ Td &= \frac{t_m^2}{2(\lambda + t_m)}\left(1 - \frac{t_m}{3T_i}\right) \end{aligned} \tag{26}$$

Otros métodos que basados en los modelos anteriores pero menos utilizados son: el método de *Chien* y *Fruehauz* (Chien and Fruehauf, 1990), quienes utilizaron el método de *Rivera* pero para controladores PID serie. También esta el método de *Brambilla* (Brambilla et al, 1990) quienes lo extienden a plantas de SOMTM. Y el método de *Lee* (Lee at al, 1990) aplica el procedimiento empleado por *Brosilow* a plantas de segundo orden sobreamortiguadas y subamortiguadas con tiempo muerto. En la tabla 18 se presentan las ecuaciones propuestas por dichos autores.

Tabla 18. Parámetros métodos Brambilla, Lee et al y Fruehauf et al.

Regla	Kp	Ti	Td	Comentario
Chien and Fruehauf	$\frac{5T_m}{9\tau_m K_m}$	$5\tau_m$	$\leq 0.5\tau_m$	$\frac{\tau_m}{T_m} < 0.33$
	$\frac{T_m}{2\tau_m K_m}$	T_m	$\leq 0.5\tau_m$	$\frac{\tau_m}{T_m} \geq 0.33$
Brambilla *et al.*	$\frac{1}{K_m}\left(\frac{T_m + 0.5\tau_m}{\lambda\tau_m}\right)$	$T_m + 0.5\tau_m$	$\frac{T_m\tau_m}{2T_m + \tau_m}$	$0.1 \leq \frac{\tau_m}{T_m} \leq 10$; $\lambda \approx 0.35$
Lee et al.	$\frac{T_i}{K_m(\lambda + \tau_m)}$	$T_m + \frac{\tau_m^2}{2(\lambda + \tau_m)}$	$\frac{\tau_m^2}{2(\lambda + \tau_m)}\left(1 - \frac{\tau_m}{3T_i}\right)$	$\lambda = \frac{\tau_m}{3}$

A continuacion se presentan de manera general otros tipos de métodos sintonizacion, basandose en la investigacion realiza en (Alfaro, 2004) en donde se recopila resulatados de investigaciones de diferentes autores.

B.2.4. Métodos basados en cancelación de polos

El principio básico de los métodos dentro de esta categoría, es el colocar los ceros del controlador sobre los polos dominantes de la planta de manera que se cancelen entre si. El primero de estos métodos fue propuesto por *Haalman* (Haalman, 1965), el cual utilizó un criterio de desempeño ISE y realizó pruebas con el control PI para un proceso de tiempo muerto puro ante cambios en la perturbación, encontrando que el mínimo de la función de costo se alcanzaba cuando *Ti*=1.5 *tm*. Conocido este valor, dedujo los ajustes del controlador para otros procesos de manera que dieran la misma función de transferencia de lazo abierto. Fue aplicado a un controlador serie.

Wang. Utiliza un controlador PID-Paralelo y una planta SOMTM (Wang et al, 1969)

Vitecková, Vitecek y Smutny**.** Desarrollaron un procedimiento de sintonización que denominaron método de inversión dinámica general. En este, los polos de la planta son cancelados con los ceros del controlador y la ganancia del controlador se varía con base a un parámetro de sintonización, el cual seleccionaron como el porcentaje de sobrepaso máximo permitido (Vitecková, 2000a; 2000b)

Estos metodos basados en la cancelacion de polos conlleva a que una planta de primer orden se controle con un controlador *PI* y una de segundo orden con un *PID*; de ahí que este método no será utilizado.

B.2.5. Métodos basados en La localización de pólos

Los métodos de sintonización de controladores basados en cancelación de polos de la planta con los ceros del controlador, conducen a respuestas de lazo cerrado sobreamortiguadas, sin permitir la localización arbitraria de los polos de lazo cerrado.

Aström y Hägglund**.** Presentan un procedimiento para lograr una respuesta subamortiguada con un decaimiento y frecuencia natural determinados, fijando la localización de los polos de lazo cerrado, para servomecanismos con controladores PI y PID (Aström y Hägglund, 1988).

Shen**.** Extiende el procedimiento de localización de los polos dominantes, a la sintonización de controladores PID para controlar plantas subamortiguadas. La

localización de los polos dominantes se hace para minimizar la integral del error y que en forma simultánea se logre una sensibilidad determinada (Shen, 2000).

Las ecuaciones para el cálculo de los parámetros del controlador obtenidas, son funciones no lineales que involucran doce términos dependientes de los parámetros del modelo subamortiguado de la planta, es por ello que estas funciones que resultan no lineales no se tomaran en cuentan debido a la complejidad de las mismas. Adicionalmente la información de los principales exponentes del método es comercial.

B.2.6. Métodos basados en márgenes de robustez

El márgenes de ganancia (MG) y margen de fase (MF) de un sistema de control, son indicadores de la robustez y grado de estabilidad relativa del mismo, esto es, indican la lejanía del punto de operación del sistema de la condición de inestabilidad. Buenos márgenes de magnitud y fase garantizan la operación estable del sistema ante cambios en las características estáticas y dinámicas de la planta. En adición a estos, la sensibilidad máxima (Ms) es también otra medida de robustez. Como la robustez tiene que ver con la estabilidad del lazo de control, es independiente del funcionamiento el mismo, esto es, si opera como servomecanismo o como regulador. Se han desarrollado entonces varios métodos de sintonización de controladores que aseguren un determinado grado de estabilidad.

***Aström y Hägglund*.** Desarrollaron el procedimiento original para la identificación de la ganancia y el periodo de oscilación último utilizando un control con relé. A partir de esta información, establecieron métodos de sintonización de controladores basados en el margen de ganancia y el margen de fase como una variante del método de sintonización de Ziegler y Nichols que garantice mayor robustez (Aström and Hägglund, 1988). Los autores recomiendan un margen de ganancia de por lo menos 2 y uno de fase de al menos 45º (Aström and Hägglund, 1984).

Ho, Hang ,Cao y J. Zhou . Utilizaron un controlador PI para controlar una planta de POMTM y empleando una aproximación de la función tangente inversa, obtuvieron los parámetros de sintonización en función de los márgenes de ganancia y de fase deseados. Además del método anterior, estos autores también desarrollaron un método aplicable a plantas de segundo orden subamortiguadas más tiempo muerto (Ho et al, 1995) y (Ho at al, 1997).

Panagopoulos. Estableció un problema de optimización con restricciones con base en el conocimiento del modelo completo del proceso (Aström et al, 1998). Este procedimiento de optimización se aplicó a un controlador *PI* (Panagopoulos et al, 1999) y luego se extendió al controlador *PID* (Panagopoulos, 2000). Sin embargo, la extensión del procedimiento a la obtención de los parámetros del controlador *PID* no pudo ser directa, requirió de la inclusión de otras restricciones para garantizar la convergencia del método de optimización (Panagopoulos et al, 2000).

La aplicación del procedimiento de optimización de la integral del error con restricciones en la sensibilidad máxima desarrollado por Panagopoulos, a un conjunto particular de plantas de prueba, llevó a *Hägglund y Aström* a establecer un método de diseño que denominan MIGO (*"Ms – constrained Integral Gain Optimization"*), con el que obtienen las ecuaciones para un procedimiento de diseño de reguladores robustos llamado AMIGOs ("Approximate MIGO based step response data"). Para el caso particular de plantas de primer orden más tiempo muerto, establecen ecuaciones simples en función de los parámetros del proceso para lograr una sensibilidad máxima Ms=1.4 (Hägglund and Aström, 2002).

El método basado en criterios de robustez tampoco es tenido en cuenta ya que algunos de los autores como Aström y Hägglund lo usaron a partir de un control con relé, otros como Ho, Hang y Cao lo usaron a partir de un control PI con plantas de SOMTM.

B.2.7. Métodos basados en un criterio múltiple

Es posible establecer también criterios de desempeño que consideren más de una característica deseada para el lazo de control, denominados de criterio múltiple

Harris y Mellichamp. Consideraron la resonancia pico Mp, el margen de fase Φm y la frecuencia de resonancia pico ωp (Harris and Mellichamp, 1985), y establecieron un índice de desempeño de la forma.

$$ID_{hm} = k_m \frac{|M_p - M_{pd}|}{M_{pd}} + \frac{k_f}{\omega_p} + \frac{|\phi - \phi_d|}{\phi_d} \quad (27)$$

Ho et al. Estos autores indican que el método de López y el de Rovira, aunque son óptimos respecto a la función de costo minimizada, no producen sistemas robustos. Mientras que se recomienda que el margen de ganancia esté entre 2 y 5 y que el margen de fase esté entre 30º y 60º, el método de López y el de Rovira dan 1.5 y 30º respectivamente, razón por la que dedujeron ecuaciones para reguladores y servomecanismos que minimicen el criterio ISE y permitan seleccionar el margen de fase y de ganancia del sistema. A este método lo denominaron ISE-GPM ("optimización del criterio ISE con márgenes de ganancia y fase dados"), para plantas FOPDT y controladores PID operando como reguladores o como servomecanismos (Ho et al, 1999).

Khan and Lehman. Desarrollaron un método de sintonización basado en un criterio múltiple para plantas de primer orden más tiempo muerto y controladores PI que incluye: tiempo de establecimiento mínimo, sobrepaso mínimo, error permanente cero, e insensibilidad a los cambios de los parámetros de la planta (Kaya, 2003).

Alfano and Embirucu**.** Definen una función de costo como una suma ponderada de los criterios *IAE*, *ISE*, *ITAE*, *ITSE*, sobrepaso máximo y cambios en el esfuerzo de control. El problema de optimización lo plantean con restricciones en el sobrepaso máximo, el esfuerzo de control y el amortiguamiento de la respuesta (Alfano and Embirucu, 2000).

Kristiansson. *D*esarrolló recientemente un procedimiento de sintonización de controladores *PI* y *PID* basado la optimización de la sensibilidad máxima con restricciones en cuatro ámbitos del ancho de banda del sistema. Estas restricciones incluyen: integral del error mínima, esfuerzo de control bajo, robustez de alta frecuencia e insensibilidad al ruido. El proceso está caracterizado por su ganancia a baja frecuencia, a la frecuencia de corte de fase o a otras frecuencias. Las ecuaciones generales derivadas no le resultaron aplicables a modelos simples como el de primer orden más tiempo muerto, por lo que dedujo ecuaciones de sintonización específicas para este modelo. En ese caso el criterio de desempeño establecido es fijo (Kristiansson, 2003).

Estos métodos se dejan como información ya que la mayoría no obtiene ecuaciones para aplicar en plantas FOPDT con controladores PID, adicionalmente usan funciones de costo y por ello se necesitan mayores conocimientos en el tema de la optimización, lo que conllevaría a una mayor investigación e inversión de tiempo.

Por ser los métodos más utilizados dentro del ambiente de la investigación e implementación de sistemas de control basados en controladores PID se tendrán en cuanta los métodos basados en la respuesta temporal del sistema y los métodos basados en la integral del error para controladores PID que trabajan como servomecanismos y reguladores. Además se cuenta con la mayoría de la información de cada método, sus formulas de sintonización y constantes para cada formula.

B.3. JUSTIFICACIÓN SELECCIÓN DE LOS MÉTODOS DE SINTONIZACIÓN

Como se mencionaron anteriormente, existen diversos métodos de sintonización de controladores PID, en donde los autores analizan esta estrategia de control desde diferentes puntos de vista (O'Dwyer, 1999). Una de las principales consideraciones para la selección del método de sintonización es que sea para controlar plantas POMTM y que además sea aplicable a diferentes estructuras de PID como serie (interactuante), Ideal (no interactuante) o industriales.

Los metodos deben permitir la comparación de del PID esta configurado como servomecanismo o reguladores. A continuación en la tablas 19 y 20 se presenta un resumen de los métodos y la selección de los métodos para la simulación de la estrategia PID.

Tabla 19. Resumen métodos sintonización PID servomecanismo

SERVOMECANISMO			
Método	**Ideal**	**Serie**	**Serie industrial**
Rovira, Murrill y Smith (IAE-ITAE-ISE)	*	-	-
Kaya y Sheib (IAE-ITAE-ISE)	*	*	*
Sung, O, Lee, Lee y Yi	*	-	-
Wang et al (IAE-ISE-ITAE) ***	*	-	-
Zhuang and Atherton (ISE-ISTSE)	*	-	-
Rivera, Morari y Stogestad (IMC)	*	-	-
Brosilow (planta 2º orden)	-	-	-
Chien y Fruehauz	-	*	-
Brambilla et al (planta 2º orden)	-	-	-
Lee (planta 2º orden)	-	-	-

Tabla 20. Resumen métodos sintonización regulador

REGULADOR			
Método	**Ideal**	**Serie**	**Serie industrial**
Ziegler y Nichols	*	-	-
Cohen – Coon	*	-	-

López, Miller, Smith y Murril (IAE-ITAE-ISE)	*	-	-
Kaya y Sheib (IAE-ITAE-ISE)	*	*	*
Sung, O, Lee, Lee y Yi (planta 2º orden)	*	-	-
Zhuang and Atherton (ISE-ISTSE)	*	-	-
Shinskey	-	*	-
Chidambara (No definido)	-	-	-
Aström et al (PID 2 DOF)	-	-	-
Chien et al (Reacción del proceso)	*	-	-

El total de métodos encontrados en la literatura y el total de métodos de sintonicaion de PIDs como servomecanismo y regulador se encuentran en las tablas 21 y 22 respectivamente.

Tabla 21. Total de Métodos

Base del método	Cantidad
Reacción del proceso	5
Integral del error (Servo y regulador)	20
Cancelación de polos	3
Localización de polos	2
Márgenes de robustez	4
Criterio Múltiple	5
TOTAL	39

Tabla 22. Total de métodos como servo y regulador

	Servomecanismo	Regulador
TOTAL	10	10

Los principales motivos de selección y rechazo fueron los siguientes:

Seleccionados como reguladores

Ziegler y Nichols: se aplica porque a partir de esta investigación, parten los demás métodos de sintonización que se basan en la reacción del proceso.

Cohen – Coon: seleccionado porque introduce el índice de desempeño tiempo-muerto/Constante de tiempo del proceso.

López, Miller, Smith y Murril: seleccionado por basarse en los criterios integrales del error (IAE-ITAE-ISE).

Kaya and Sheib: Seleccionado porque se aplica con base en López et al, y se emplea en controladores PID Serie, Ideal y Serie Industrial.

No seleccionados como Reguladores

Sung, O, Lee, Lee y Yi: no seleccionado debido a que se emplea con plantas de SOMTM.

Zhuang and Atherton: este método no se usa debido a que se aplica solamente a controladores PID ideales con base en solo dos criterios del error (ISE-ISTSE).

Shinskey se basa en el método de IAE y solo es aplicado para un controlador serie.
Chidambara: se basa en Ziegler y Nichols pero sin llevar al sistema a la inestabilidad.
Aström et al: aplica el método a controladores de dos grados de libertad.

Seleccionados como Servomecanismo

Wang et al: este método se selecciona porque se basa en los tres criterios IAE, ITAE e ISE, permitiendo su comparación en un PID Ideal como servo.
Kaya and Sheib: usan los tres criterios de integral del error más comunes y se aplican al PID ideal, PID serie y PID serie industrial como servomecanismos.
Rovira, Murrill and Smith: usan los tres criterios de integral del error más comunes y se aplican al PID ideal como servomecanismo.

No seleccionados Servomecanismo

Zhuang and Atherton: Este método no se usa debido a que se aplica solamente a controladores PID ideales con base en solo dos criterios del error ISE e ISTSE.
Rivera et al. No se selecciona este método debido a que es aplicado para analizar la robustez del sistema. Adicionalmente el método se aplica solamente a un tipo de controlador.
Brosilow usa un procedimiento a partir de la función de transferencia estimada, de ahí que no aplica para este experimento.
Brambilla, aplican sus estudios en un controlador PID ideal pero con plantas de segundo orden.
Lee, aplican sus investigaciones en un controlador PID ideal pero con plantas de segundo orden sobreamortiguadas y subamortiguadas con tiempo muerto.
Chien and Fruehauz se basa en Brosilow para controladores serie, pero sin definir si el controlador trabaja como servomecanismo o regulador.
Sung, O, Lee, Lee y Yi: no se usa porque se usa un modelo de segundo orden más tiempo muerto.

Las ecuaciones de los métodos de sintonización seleccionados se resumen en las tablas 23,24 y 25.

Tabla 23. Ecuaciones sintonización de controladores PID paralelo

PID Estandar o Ideal $G_c(s) = K_c\left(1 + \frac{1}{T_i s} + T_d s\right)$ **Como Servomecanismo**				
Método	Kc	Ti	Td	Comentario
Minimum IAE - Rovira et al.	$\frac{1.086}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.869}$	$\frac{T_m}{0.740 - 0.13\frac{\tau_m}{T_m}}$	$0.348T_m\left(\frac{\tau_m}{T_m}\right)^{0.914}$	$0.1 < \frac{\tau_m}{T_m} \le 1$
Minimum ITAE - Rovira et al.	$\frac{0.965}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.85}$	$\frac{T_m}{0.796 - 0.1465\frac{\tau_m}{T_m}}$	$0.308T_m\left(\frac{\tau_m}{T_m}\right)^{0.929}$	$0.1 \le \frac{\tau_m}{T_m} \le 1$
Rivera et al. IMC	$\frac{1}{K_m}\left(\frac{T_m + 0.5\tau_m}{\lambda + 0.5\tau_m}\right)$	$T_m + 0.5\tau_m$	$\frac{T_m\tau_m}{2T_m + \tau_m}$	$\lambda > 0.1T_m$, $\lambda \ge 0.8\tau_m$.
Minimum IAE Wang et al	$\frac{\left(0.7645 + \frac{0.6032}{\tau_m/T_m}\right)(T_m + 0.5\tau_m)}{K_m(T_m + \tau_m)}$	$T_m + 0.5\tau_m$	$\frac{0.5T_m\tau_m}{T_m + 0.5\tau_m}$	$0.05 < \frac{\tau_m}{T_m} < 6$
Minimum ISE Wang et al	$\frac{\left(0.9155 + \frac{0.7524}{\tau_m/T_m}\right)(T_m + 0.5\tau_m)}{K_m(T_m + \tau_m)}$	$T_m + 0.5\tau_m$	$\frac{0.5T_m\tau_m}{T_m + 0.5\tau_m}$	$0.05 < \frac{\tau_m}{T_m} < 6$
Minimum ITAE Wang et al	$\frac{\left(0.7303 + \frac{0.5307}{\tau_m/T_m}\right)(T_m + 0.5\tau_m)}{K_m(T_m + \tau_m)}$	$T_m + 0.5\tau_m$	$\frac{0.5T_m\tau_m}{T_m + 0.5\tau_m}$	$0.05 < \frac{\tau_m}{T_m} < 6$
PID Estandar o Ideal $G_c(s) = K_c\left(1 + \frac{1}{T_i s} + T_d s\right)$ **Como Regulador**				
Método	Kc	Ti	Td	Comentario
Ziegler and Nichols	$\left[\frac{1.2T_m}{K_m\tau_m}, \frac{2T_m}{K_m\tau_m}\right]$	$2\tau_m$	$0.5\tau_m$	Decaimiento de 1/4
Cohen and Coon	$\frac{1}{K_m}\left(1.35\frac{T_m}{\tau_m} + 0.25\right)$	$T_m\left(\frac{2.5\frac{\tau_m}{T_m} + 0.46\left(\frac{\tau_m}{T_m}\right)^2}{1 + 0.61\frac{\tau_m}{T_m}}\right)$	$\frac{0.37\tau_m}{1 + 0.2\frac{\tau_m}{T_m}}$	Decaimiento de 1/4
Minimum IAE - Lopez-Murrill	$\frac{1.435}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.921}$	$\frac{T_m}{0.878}\left(\frac{T_m}{\tau_m}\right)^{0.749}$	$0.482T_m\left(\frac{\tau_m}{T_m}\right)^{1.137}$	$0.1 < \frac{\tau_m}{T_m} \le 1$
Minimum ITAE - Lopez-Murrill	$\frac{1.357}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.947}$	$\frac{T_m}{0.842}\left(\frac{T_m}{\tau_m}\right)^{0.738}$	$0.381T_m\left(\frac{\tau_m}{T_m}\right)^{0.995}$	$0.1 < \frac{\tau_m}{T_m} \le 1$
Minimum ISE – Lopez-Murrill	$\frac{1.495}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.945}$	$\frac{T_m}{1.101}\left(\frac{T_m}{\tau_m}\right)^{0.771}$	$0.56T_m\left(\frac{\tau_m}{T_m}\right)^{1.006}$	$0.1 < \frac{\tau_m}{T_m} \le 1$

Tabla 24. Ecuaciones sintonización de controladores PID Serie industrial

PID Serie Industrial $U(s) = K_c\left(1+\frac{1}{T_i s}\right)\left(R(s) - \frac{1+T_d s}{1+\frac{T_d s}{N}}Y(s)\right)$ **Servomecanismo**				
Método	Kc	Ti	Td	Comentario
Minimum IAE Kaya and Scheib	$\frac{0.81699}{K_m}\left(\frac{T_m}{\tau_m}\right)^{1.004}$	$\frac{T_m}{1.09112 - 0.22387\frac{\tau_m}{T_m}}$	$0.44278T_m\left(\frac{\tau_m}{T_m}\right)^{0.97186}$	$0 < \frac{\tau_m}{T_m} \le 1$; N=10
Minimum ITAE Kaya and Scheib	$\frac{0.8326}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.7607}$	$\frac{T_m}{0.99223 - 0.35269\frac{\tau_m}{T_m}}$	$0.35308T_m\left(\frac{\tau_m}{T_m}\right)^{0.78088}$	$0 < \frac{\tau_m}{T_m} \le 1$; N=10
Minimum ISE Kaya and Scheib	$\frac{0.8326}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.7607}$	$\frac{T_m}{1.00268 + 0.00854\frac{\tau_m}{T_m}}$	$0.35308T_m\left(\frac{\tau_m}{T_m}\right)^{0.78088}$	$0 < \frac{\tau_m}{T_m} \le 1$; N=10
PID Serie Industrial $U(s) = K_c\left(1+\frac{1}{T_i s}\right)\left(R(s) - \frac{1+T_d s}{1+\frac{T_d s}{N}}Y(s)\right)$ **Regulador**				
Método	Kc	Ti	Td	Comentario
Minimum IAE - Kaya and Scheib	$\frac{0.91}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.7938}$	$\frac{T_m}{1.01495}\left(\frac{T_m}{\tau_m}\right)^{1.00403}$	$0.5414T_m\left(\frac{\tau_m}{T_m}\right)^{0.7848}$	$0 < \frac{\tau_m}{T_m} \le 1$; N=10
Minimum ITAE - Kaya and Scheib	$\frac{0.7058}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.8872}$	$\frac{T_m}{1.03326}\left(\frac{T_m}{\tau_m}\right)^{0.99138}$	$0.60006T_m\left(\frac{\tau_m}{T_m}\right)^{0.971}$	$0 < \frac{\tau_m}{T_m} \le 1$; N=10
Minimum ISE - Kaya and Scheib	$\frac{1.1147}{K_m}\left(\frac{T_m}{\tau_m}\right)^{0.8992}$	$\frac{T_m}{0.9324}\left(\frac{T_m}{\tau_m}\right)^{0.8753}$	$0.56508T_m\left(\frac{\tau_m}{T_m}\right)^{0.91107}$	$0 < \frac{\tau_m}{T_m} \le 1$; N=10

B.4. SIMULACIÓN Y ANÁLISIS DE LOS MÉTODOS DE SINTONIZACIÓN

Para la simulación del algoritmo PID en sus estructuras serie y paralela funcionando como servomecanismo y como regulador se utilizó la herramienta Simulink de Matlab-7.2a, los métodos utilizados en la simulación son los mencionados en las tablas 11 y 12 debido a que se usan en los controladores PID industriales funcionando como servomecanismo y regulador en sus dos estructuras. A continuación se presentan las salidas y esfuerzos de control del PID paralelo como servomecanismo y regulador, luego se muestran las salidas y esfuerzos de control del PID serie como servomecanismo y regulador. Cabe aclarar que el controlador PID cuenta con el parámetro Uo.

El experimento de simulación consiste en analizar la respuesta del PID ante cambios en el valor de consigna (como servomecanismo), se introducirán al sistema dos escalones de ±10 centímetros a los 20 y 60 segundos de la simulación. Y analizar la

respuesta del sistema ante presencia de perutrbaciones (como regulador), se introducirá al sistema dos perturbaciones de ±5 gpm en la carga a los 40 y 60 segundos respectivamente del tiempo de simulación, este último corresponde a un total de 100 segundos.

Para realizar una comparación cuantitativa se realizará el análisis de los valores del tiempo de subida, tiempo de establecimiento, máximo sobreimpulso y el tiempo en alcanzar la banda del 2% equivalente a 1.5 cms (+0.75,-0.75 del set point).

Para la sintonización del PID Industrial Paralelo como servomecanismo se utilizaron los metodos de Rovira, Rivera y Wang, para el PID Industrial paralelo como regulador se utilizaron los métodos de Zieger-Nichols, Cohen-Coon y López.

Para la sintonización del PID Serie Industrial como Servomecanismo se seleccionó el método de Kaya and Sheib porque es el único método disponible en la literatura para un controlador de estas características. Y para este mismo controlador cuando funciona como regulador se seleccionó el mismo método mencionado para este tipo de estructura.

B.4.1. PID paralelo como servomecanismo

La respuesta del sistema muestra un comportamiento un poco lento de ahí que las respuestas no presentan sobre impulso. En la figura 19 se aprecia la respuesta de la VC y en la figura 20 la respuesta del EC.

Figura 19. PID Paralelo Industrial Servomecanismo

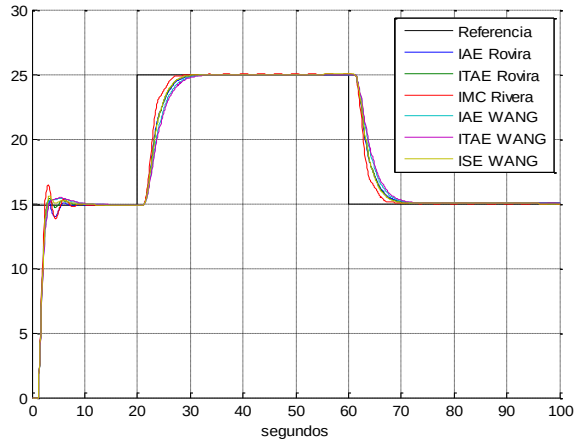
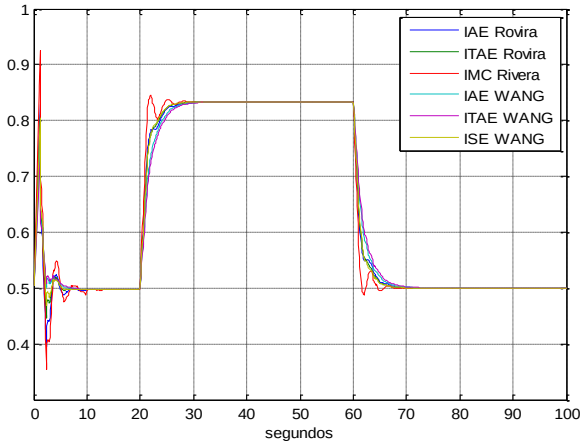


Figura 20. Esfuerzo de Control PID paralelo Industrial como Servomecanismo



Los EC son suaves ante cambios en el valor de consigna, con la diferencia que el método IMC Rivera presenta oscilaciones en la señal de control al momento de alcanzar la referencia.

B.4.2. PID paralelo como regulador

El rechazo a las perturbaciones se hace de manera rápida presentando oscilaciones para alcanzar el estado estable, el método de Cohen-Coon no es apto para este tipo de estructuras. Se observa que los métodos que mejor se comportan ante la presencia y rechazo de las perturbaciones son IAE y el ITAE de Lopez, ver figura 21.

En la fgura 22 se muestra el comportamiento del EC, el cual presenta respuestas rápidas y con oscilaciones suaves. Los métodos de sintonización que mejor respuesta presentan rechazando las perturbaciones son el IAE e ITAE López.

Figura 21. PID Paralelo Industrial Regulador

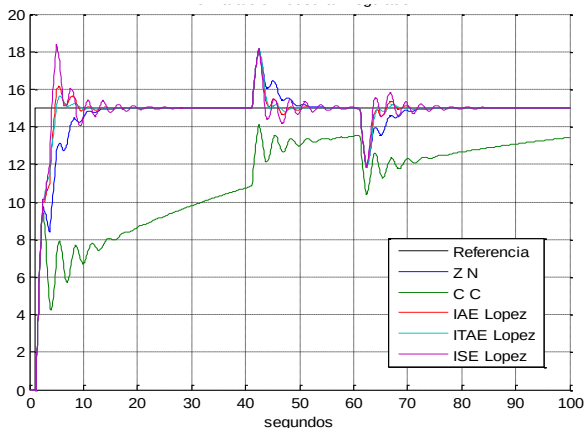
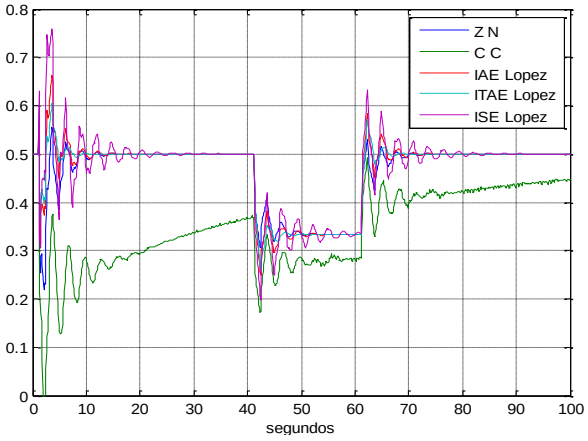


Figura 22. EC PID Paralelo Industrial Regulador



B.4.3. PID serie como servomecanismo

Las respuestas de la VC de la figura 23 como servomecanismo son rápidas, lo que implica que se presenten pequeños sobre impulsos, los métodos seleccionados brindan una buena respuesta del proceso ante cambios en el valor de consigna.

La señal de control para el PID serie, figura 24 ante los cambios en el valor de consigna son oscilantes y presenta un máximo pico al aumentar la referencia de manera que tiende a la saturación del actuador, el único método de sintonización que no conlleva a la saturación es el método IAE de *Kaya y Sheib* el cual presenta el mejor comportamiento ante aumento o disminución de la referencia.

Figura 23. PID Serie Industrial Servomecanismo

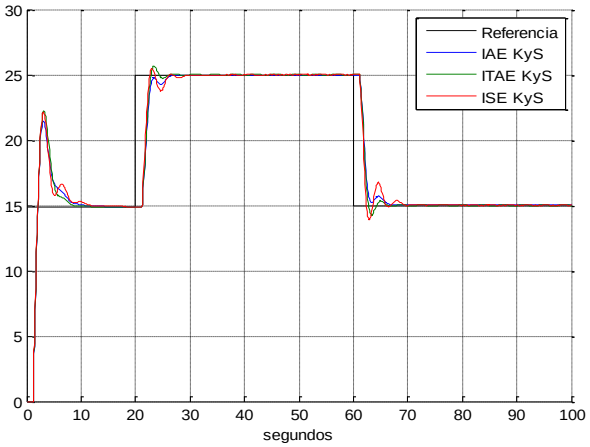
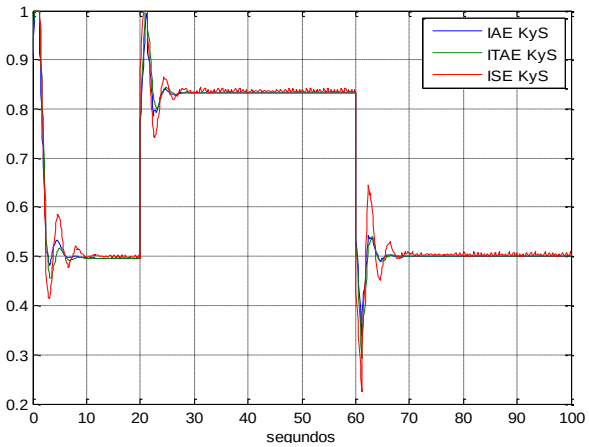


Figura 24. EC PID Serie Industrial Servomecanismo



B.4.4. PID serie como regulador

En cuanto al funcionamiento como regulador, el rechazo a las perturbaciones se realiza de forma rápida, presentando respuestas suaves y mínimas oscilaciones suaves ante perturbaciones positivas o negativas, ver figura 25. Adicionalmente, el comportamiento del EC de cada metodo es suave y no presentan grandes sobre impulsos ante las perturbaciones, el método que mayor esfuerzo de control presenta es el ISE López, ver figura 26.

Figura 25. PID Serie Industrial Regulador

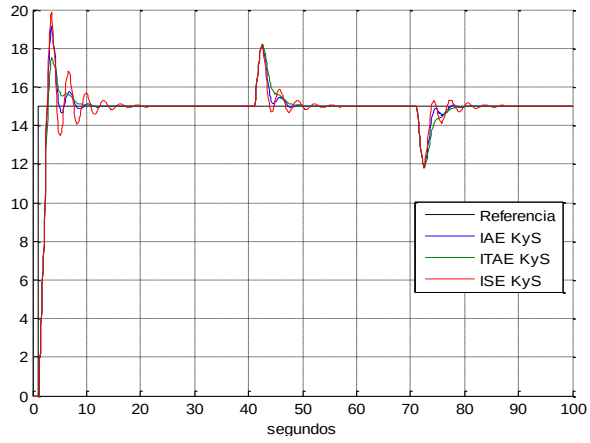
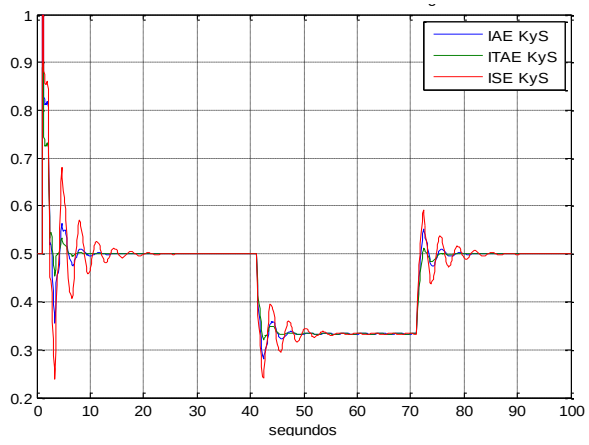


Figura 26. EC PID Serie Industrial Regulador



B.5. MEDICIÓN CUANTITATIVA DE PARÁMETROS PID PARALELO

En las siguientes tablas se consolidan los valores de los parámetros que identifican la respuesta del sistema como son: máximo sobreimpulso, tiempo en llegar a la banda del 2%, tiempo de subida o tiempo de elevación, este ultimo considerado como el tiempo que tarda la respuesta en ir del 10% al 90%. Se debe tener en cuenta que los cambios en el valor de consigna equivalen a ±10 cms y las perturbaciones son de ±5 gpm.

Tabla 25. Valores cuantitativos VC PID Paralelo como servomecanismo y regulador.

PID PARALELO INDUSTRIAL – VC																						
	Servomecanismo												Regulador									
	IAE-Ro		ITAE- Ro		IMC- Ri		IAE-W		ITAE-W		ISE-W		ZN		CC		IAE-Lo		ITAE-Lo		ISE-Lo	
	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓
Ms	0	0	0	0	0.03% 0.003	0.03% 0.003	0	0	0	0	0	0	37.2% 3.17	36.6% 3.169	--	--	37% 3.15	36.7% 3.165	36.8% 3.16	36.7% 3.166	37.2% 3.14	36.3% 3.162
Ts 2%	9.49	9.46	8.79	8.76	7.31	7.28	10.15	10.1	10.77	10.75	8.63	8.6	13.82	10.86	--	--	10.37	10.35	7.05	7.06	18.79	18.73
Tsub seg	4.45	4.47	4.29	4.3	3.55	3.57	4.94	4.96	5.23	5.26	4.22	4.23	na	na	na	na	na	na	na	na	na	na

Tsub=Tiempo de subida, na= no aplica. El porcentaje como Regulador equivale al porcentaje de rechazo de la perturbación.

Tabla 26. Valores cuantitativos EC PID Paralelo como servomecanismo y regulador.

PID PARALELO INDUSTRIAL – Esfuerzo de Control																						
	Servomecanismo												Regulador									
	IAE-Ro		ITAE- Ro		IMC- Ri		IAE-W		ITAE-W		ISE-W		ZN		CC		IAE-Lo		ITAE-Lo		ISE-Lo	
	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓
Ms (%)	0	0	0	0	.6% 0.012	3.9% 0.0133	0	0	0	0	0	0	0.0283	0.0309	-	-	0.0831	0.0866	0.0722	0.0762	0.1354	0.1397
Ts (seg)	7.69	7.71	6.89	7.01	7.05	7.04	8.51	8.42	9.08	9.08	6.62	6.91	12.17	9,75	--	--	9.95	11,08	7.62	7,44	21.08	22.21
Tsub (seg)	3.45	3.52	3.125	2.873	1.13	1.17	3.76	3.83	4.09	4.23	2.90	3.03	0.619	0.614	--	--	0.41	0.08	0.572	0.56	1.14	1.14

B.6. MEDICIÓN CUANTITATIVA DE PARÁMETROS PID SERIE

En las siguientes tablas se consolidan los valores de los parámetros que identifican la respuesta del sistema como son: tiempo de establecimiento (Ts), Máximo sobreimpulso (Ms), tiempo en llegar a la banda del 5%, tiempo de subida (Tsub). Se debe tener en cuenta que los cambios en el valor de consigna equivalen a ±10 cms y las perturbaciones son de ±5 gpm.

Tabla 27. Valores cuantitativos VC PID Serie como servomecanismo y regulador.

PID SERIE INDUSTRIAL – VC												
	Servomecanismo						Regulador					
	IAE		ITAE		ISE		IAE		ITAE		ISE	
	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓
Ms %	0	0	6.53% 0.6535	6.53% 0.6530	4.96% 0.4961	10.9% 1.096	36.2% 3.19	36% 3.2	35.6% 3.22	35.2% 3.22	36.8% 3.16	36.6% 3.17
Ts 2%	5.23	5.28	4.65	4.71	7.94	7.96	9.72	9.75	9.47	9.48	14.3	14.28
Tsub seg	1.39	1.39	1.21	1.18	1.07	0.96	na	na	na	na	na	na

Tsub=Tiempo de subida, na= no aplica. El porcentaje como Regulador equivale al porcentaje de rechazo de la perturbación.

Tabla 28. Valores cuantitativos EC PID Serie como servomecanismo y regulador

PID SERIE INDUSTRIAL – Esfuerzo de Control												
	Servomecanismo						Regulador					
	IAE		ITAE		ISE		IAE		ITAE		ISE	
	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓
Ms	52.1% 0.1595	47.8% 0.1738	S	32.94% 0.2233	S	13.21% 0.289	36.2% 3.19	36% 3.2	35.6% 3.22	35.6% 3.22	36.8% 3.16	36.6% 3.17
Ts 2%	6.58	6.49	5.68	5.67	N2	N2	9.42	9.57	7.57	6	17.3	17.14
Tsub (seg)	0.14	0.14	0.1	0.1	0	0	0.65	0.63	0.9	0.89	0.44	0.42

S=saturado, N2=no entra a la banda del 2%

B.7. CONCLUSIONES

El PID paralelo industrial como servomecanismo tiene un buen comportamiento ante cambios en la consigna al ser sintonizado con los métodos propuestos por Rovira y Wang, el método IMC-Rovira presenta un pequeño sobre impulso debido la rapidez de respuesta, es decir es uno de los métodos mas rápidos de los seleccionados en llevar a la variable de control al estado deseado, pero la presencia de esta rapidez, le implica al sistema un sobreimpulso. Para el controlador PID serie como servomecanismo, presenta una buena respuesta ante los cambios en la consigna por medio de la sintonización con los métodos IAE e ITAE de *Kaya and Scheib*. A pesar de que el ITAE presenta un sobre impulso por debajo de 7%, implica que sea rápido y presenta oscilaciones suaves. Por medio de la señal de control se observa que el mejor método es el IAE, debido a que no alcanza a saturar el actuador.

En cuanto al esfuerzo de control para el mismo PID, solo representan oscilaciones en la salida del controlador al ser sintonizado con el método IMC-Rovira, la rapidez mencionada anteriormente ocasiona un mayor esfuerzo de control en el actuador. Los métodos no presentan oscilaciones y sus transiciones son suaves aptas para una señal de control.
De los métodos usados par sintonizar el controlador PID paralelo industrial, los que brindan mayor rechazo a las perturbaciones son IAE y el ITA de López, los cuales presentan los mejores tiempos de establecimiento, como se puede ver en la Tabla 25.

En el PID serie como regulador se presenta el mismo caso del serie como servomecanismo, en donde los métodos de sintonización más sobresalientes son el IAE y el ITAE de *Kaya and Scheib*, pero para reguladores, aunque según los resultados cuantitativos presentan los menores porcentajes de rechazo a la perturbación, presentan los mejores tiempos de establecimiento, ver tabla 28; adicionalmente de acuerdo a la figura 26, se observa que el esfuerzo de control es suave en especial para el caso del método ITAE que no presenta oscilaciones.
Finalmente, de acuerdo a los resultados del experimento se puede decir que los métodos que mejor sintonizan al controlador PID industrial serie y paralelo son:

PID paralelo servomecanismo:	Rovira (IAE ,ITAE), Wang (IAE, ITAE, ISE)
PID paralelo regulador.	López (IAE ,ITAE)
PID serie servomecanismo:	Kaya y Sheib (IAE ,ITAE)
PID paralelo regulador:	Kaya y Sheib (IAE ,ITAE).

B.8. REFERENCIAS

ALFANO, C., y M. EMBIRUCU. Tuning of PID Controllers: an Optimization-based Method. IFAC Digital Control: Past, Present and future of PID Control, Terrassa, España, 2000.

ALFARO, Víctor. Métodos de sintonización de controladores PID que operan como reguladores. Articulo de divulgación, Universidad de Costa Rica, San José, Costa Rica, pp 21-36, 2002.

ALFARO, Víctor. Métodos de sintonización de controladores PID que operan como servomecanismos. Artículo de divulgación, Universidad de Costa Rica, San José, Costa Rica, pp 13-29, 2003.

ALFARO, Víctor**.** Evolución y Tendencias en el Desarrollo de los Métodos de Sintonización de Controladores PID. Articulo de divulgación. Departamento de Automática, Universidad de Costa Rica., pp 1-85, Diciembre 2004.

ASTRÖM, K.J. Y T. HÄGGLUND. Automatic Tuning of Simple regulators with specifications on Phase and Amplitude Margin. Automatica, vol. 20 nº 5, 1984.

ASTRÖM, K.J. Y T. HÄGGLUND. Automatic Tuning of PID Controllers. Research Triangle Park, NC. EUA, Instrument Society of America, 1988.

ASTRÖM, K.J., H. PANAGOPOULOS and T. HÄGGLUND. Design of PI Controllers based on Non-Convex Optimization. Automatica, Vol. 34 Nº 5, pp 585-601, 1998.

BRAMBILLA A., S. CHEN and C. SCALL. Robust tuning of conventional controllers. Hydrocarbon Processing, November 1990.

BROSILOW, C. PI/PID Controller parameters from IMC design. Department of Chemical Engineering, Case West Reserve University. 1992.

CAMACHO, O.E.. Sintonización de Controladores. Artículo de divulgación, Universidad de los Andes, Venezuela, pp 1-21, Julio 2005.

CHIDAMBARA, M.R. Chemical Process Control. A new technique for adaptive tuning controller, International Journal of Control (UK), Vol. 12 Nº 6, pp. 1057 – 1074, 1970.

CHIEN, K.L., HRONES, J.A. and RESWICK, J.B. On the automatic control of generalised passive systems. Transactions of the ASME, February 1952, pp. 175-185.

CHIEN I.L and P.S FRUEHAUF. Consider IMC tuning to improve controller performance. Chemical Engineering Progress, October 1990.

COHEN, G.H and COON, G.A. Theoretical considerations of retarded control. Transactions of the ASME, May 1953, pp. 827-834.

GÓMEZ DE LA RIVA, D. Identificación y sintonización de PID de una regulación de temperatura en un proceso Industrial. Artículo de divulgación, pp 1-9, 2006.

HÄGGLUND, T. and K.J. ASTRÖM. Revisiting the Ziegler-Nichols tuning rules for PI control. Asian Journal of Control, Vol. 4 Nº 4, pp 364- 380, Dic. 2002.

HAALMAN, A. Adjusting Controllers for a Dead Time Process. Control Engineering, July. 1965.

HARRIS, S.L. and D.A. MELLICHAMP. Controller Tuning Using optimization to Meet Multiple Close-Loop Criteria", ALCHE Journal, Vol. 31 Nº 3, Mar. 1985.

HO, W.K., C.C. HANG and L.S. CAO. Tuning of PID Controllers Based on Gain and Phase Margin Specification. Automatica, Vol. 31 Nº 3, 1995

HO, W.K., C.C. HANG and J. ZHOU. Self tuning PID Control of a Plant with under-Damped Response with Specifications on Gain and Phase Margins. IEEE Transactions on Control Systems Technology, Vol. 5, Nº 4, Jul. 1997.

HO, W.K., K.W. LIM, C.C. HANG and L.Y. NI. Getting more phase margin and performance out of PID controllers. Automatica, Vol. 35, 1999.

KAYA, A. and T.J. SCHEIB; Tuning of PID Controllers of Different Structures, Control Engineering (EUA), pp. 62 – 65, Dic. 1988.

KAYA I. A new Smith predictor and controller for control of processes with long dead time. ISA Transactions, Vol. 42 Nº 1. 2003.

KRISTIANSSON, B. PID Controllers, Design and Evaluation. PhD. Thesis, Control and Automation Laboratory, Department of Signals and Systems, Chalmers University of Technology, Göteborg, Suecia, 2003.

LEE, T., S. PARK, M. LEE AND C. BROSILOW, “PID Controller tuning to obtain desired closed loop response for single input, single output”, Dept. of Chemical engineering, Korean Advanced Institute of Science and Technology Taejon, Korea. 1996.

LÓPEZ, A.M., J.A. Miller, C.L. Smith y P.W. Murril; Tuning Controllers with Error-Integral Criteria, Instrumentation Technology (EUA), Nov. 1967.

O’DWYER, A. PI and PID controller tuning rules for time delay processes: a summary. Part 2: PID controller tuning rules. Proceedings of the Irish Signals and Systems Conference, National University of Ireland, Galway, Ireland, pp 339-347, June 1999.

PANAGOPOULOS, H, K.J. ASTRÖM and T. HÄGGLUND. Design of PID controllers based on constrain optimization. Proceedings of the American Control Conference, San Diego, CA, June. 1999.

PANAGOPOULOS, H. PID Control, Design, Extension, Application. PhD. Thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Suecia, 2000.

PANAGOPOULOS, H, K.J. ASTRÖM and T. HÄGGLUND. Supplement and Errata to Design of PID controllers based on constrain optimization. Departamento de Control Automático, Instituto de Tecnología de Lund, Suecia, 2000.

RIVERA, D.E., M. MORARI and S. STROGESTAD. Internal Model Control. 4. PID Controller Design. Ind. Eng. Chem. Process Des. Dev, Vol 25, No 1, pp 252-265, 1986.

ROVIRA, A., P.W. Murrill and C.L. Smith. Tuning controllers for set point changes. Instrumentation and Control Systems, pp 67-69, December 1969.

SHEN, J-C. New tuning method for PID control of a plant with under-damped response. Asian Journal of Control, Vol. 2 Nº 1, pp 31-41, Mar. 2000.

SHINSKEY, F.G.; Process Control Systems,Segunda Edición, New York, NY, UA, McGraw-Hill Book Co., 1979.

SUNG, S.W., J. O, I.B. LEE, J. LEE y S.H. YY. Automatic Tuning of PID Controller using Second-Order plus Time delay Model. Journal of Chemical Engineering of Japan, Vol. 29 Nº 6, pp. 990 – 999, Japan, 1996.

VITECKOVÁ, M., A. VITECEK Y L. SMUTNY. Controller Tuning for Controlled Plants with Time Delay. IFAC Workshop on Digital Control: Past, Present and Future of PID Control, Terrassa, España, pp 5-7, April 2000a.

VITECKOVÁ, M., A. VITECEK Y L. SMUTNY. Simple PI and PID Controllers Tuning for Monotone Self Regulating plants. IFAC Workshop on Digital Control: Past, Present and Future of PID Control, Terrassa, España, pp 5-7 April 2000b.

WANG, F.-S., JUANG, W.-S. and CHAN, C.-T.: 'Optimal tuning of PID controllers for single and cascade control loops', Chemical Engineering Communications, 1995, pp. 15-34.

WANG, Q.G., T.H. LEE, H.W. FUNG, Q. BI and Y. ZHANG. PID Tuning for Improved Performance. IEEE Transactions on control Systems Technology, Vol. 7 Nº 4, Jul. 1999.

ZHUANG, M. and D.P ATHERTON. Automatic tuning of optimum PID controllers. IEEE Proceedings, Part D, 1993, 140(3), pp. 216-224.

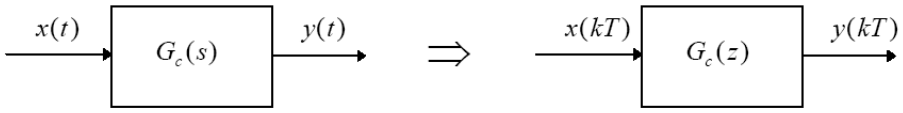
ZIEGLER, J.B. and NICHOLS N.B. Optimum Settings for Automatic Controls. ASME Transactions (EUA), Vol. 64, pp. 759-768, 1942.

ANEXO C DISCRETIZACIÓN PID

C.1. MÉTODOS BASADOS EN LA APLICACIÓN DE MÉTODOS NUMÉRICOS

Transforman la ecuación diferencial que define al controlador continuo Gc(s),en una ecuación en diferencias que define Gc(z).

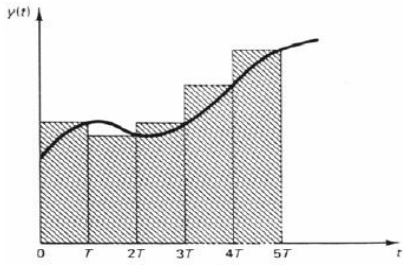
Figura 27. Transformación de ecuación diferencial a ecuaciones en diferencias



C.1.1. Método de Euler hacia atrás.

El método aproxima la integral bajo la curva por la suma de rectángulos hacia atrás.

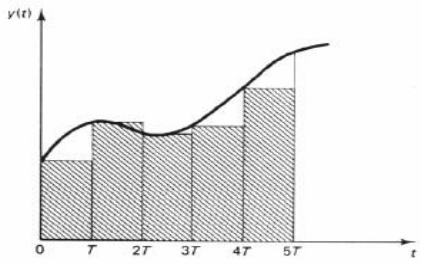
Figura 28. Aproximación de Euler hacia Atras



C.1.2. Método de Euler hacia adelante

El método aproxima la integral bajo la curva por la suma de rectángulos hacia adelante.

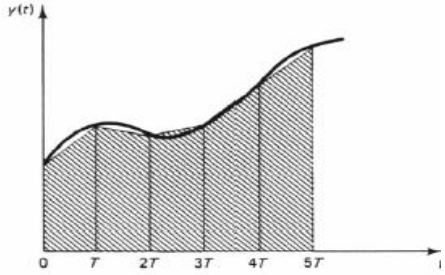
Figura 29. Aproximación de Euler hacia Adelante



C.1.3. Método trapezoidal

Se hace la aproximación de la integración por el área del trapecio.

Figura 30. Aproximación Trapezoidal



C.2. MÉTODOS BASADOS EN LA APLICACIÓN DE LA TRANSFORMADA Z.

C.2.1. Método invariante impulsional

Se trata de preservar la respuesta impulso, tal que definiendo las respuestas impulso continua y discreta.

$$g_c(t) = L^{-1}\{G_c(s)\},\ g_d(kT) = Z^{-1}\{G_c(z)\}$$

Se cumple:

$$g_d(kT) = T g_c(t)|_{t=kT}$$
$$G_d(z) = TZ\{g_c(t)\} = TG_c(z)$$

C.2.2. Método invariante del escalón

Se tratara de preservar la respuesta al escalón, donde:

$$Z^{-1} = \left\{ G_d(z)\frac{1}{1-z^{-1}} \right\} = L^{-1}\left\{ G_c(s)\frac{1}{s} \right\}_{t=kT}$$
$$G_d(z) = Z\left\{ \frac{1-e^{-Ts}}{s} G_c(s) \right\}$$

Por tanto, el $G_d(z)$ equivale a $G_c(s)$ precedido por un muestreador y HOLD0 ficticios, es decir, no existen como circuitos físicos.

C.2.3. Emparejamiento polos-ceros

Se consideran por separado el numerador y denominador de Gc(s), y se mapean los polos y los ceros de Gc(s) en los de Gc(z).

Polos finitos

$$G_c(s)|_{s_p=-a} \rightarrow G_c(z)|_{z_p=e^{-aT}}$$

Ceros finitos

$$G_c(s)|_{s_c=-b} \rightarrow G_c(z)|_{z_c=e^{-bT}}$$

Ceros infinitos se mapean en Zc=-1, y los polos infinitos en Zp=-1

Se ajusta la ganancia de Gc(z) para que iguale a la de Gc(s) en w=wc

C.3. DISCRETIZACION PID PARALELO SIN ANTIWINDUP

Se parte de la ecuación característica de un PID paralelo, ecc (1):

$$U_d(s) = Kp\left(bR(s) - Y(s) + \frac{1}{T_i s}\, E(s) - \frac{T_d s}{T_d s/_N + 1} Y(s) \right) \qquad (1)$$

C.3.1. Discretización parte derivativa

Para discretizar la componente derivativa es necesario obtener una realización mínima de función de transferencia y transformarla a variables en espacio de estados así.

Sea

$$u_d(s) = -\frac{T_d}{T_d/_N s+1} Y(s)$$

Se obtiene una realización mínima de la función de transferencia:

$$U_d(s) = \left(\frac{N^2/_{T_d}}{s + N/_{T_d}} - N \right) Y(s) \qquad (2)$$

Ahora se define una variable auxiliar:

$$X(z) = \left(\frac{1}{s + {}^{N}/_{T_d}}\right) Y(s) \tag{3}$$

Sustituyendo (3) en (2), aplicando Transformada de Laplace se obtiene el sistema en espacio de estados del componente derivativo así:

$$\begin{aligned} \dot{x} &= \left(-\frac{N}{T_d}\right) x + y \\ U_d &= \left(\frac{N^2}{T_d}\right) x - Ny \end{aligned} \tag{4}$$

Ahora al discretizar un sistema de tiempo continuo se debe cambiar el integrador de tiempo continuo por uno de tiempo discreto. Dicho integrador puede ser:

Euler en atraso (5a)

Euler en adelanto (5b)

Aproximación de Euler (5c)

$$\underset{(a)}{I_F(z) = \frac{hz}{z-1}} \qquad \underset{(b)}{I_B(z) = \frac{h}{z-1}} \qquad \underset{(c)}{I_T(z) = \frac{h}{2}\left(\frac{z+1}{z-1}\right)} \tag{5}$$

Para poder realizar la discretización se utiliza la aproximación poligonal de Tustín $(ecc\ 5c)$, porque este aporta la mayor precisión de los métodos numéricos, en la aproximación h representa el tiempo de muestreo del sistema de control. Luego de seleccionar el integrador se debe transformar el diagrama de bloques a uno de la forma:

$$\begin{aligned} x(kh + h) &= \Phi x(kh) + \Gamma y(kh) \\ u_d &= C_d x(kh) + D_d y(kh) \end{aligned} \tag{6}$$

Para convertir el sistema discretizado en un sistema basado en el integrador de Tustín , sustituyendo (5) en (4) , los valores de Φ, Γ, C_d y D_d se encuentran (7).

$$\begin{aligned} \Phi = 1 + hA\left(1 - \frac{h}{2}A\right)^{-1} &= \frac{2T_d - hN}{2T_d + hN} \\ \Gamma = B + hA\left(1 - \frac{h}{2}A\right)^{-1} B &= \frac{2T_d}{2T_d + hN} \\ C_d = hC\left(1 - \frac{h}{2}A\right)^{-1} &= \frac{2hN^2}{2T_d + hN} \\ D_d = \frac{h}{2}C\left(1 - \frac{h}{2}A\right)^{-1} B + D &= -\frac{2T_d N}{2T_d + hN} \end{aligned} \tag{7}$$

La ecuaciones (4) y (7) corresponden a la implementación en espacio de estados en tiempo discreto de la componente derivativa, ecuación (2).

C.3.2. Discretización parte integral

Para la discretización de la componente integral se define la variable:

$$U_i(s) = \frac{1}{T_i s} E_i(s) \tag{8}$$

$$E_i(s) = R(s) - Y(s) \tag{9}$$

Para la discretización de $E_i(s)$ se toman *kh* muestras a R(s) y Y(s), así el sistema discreto para (9) es (10)

$$e_i(kh) = r(kh) - y(kh) \tag{10}$$

El componente integral de $U_i(s)$, $1/T_i s$ es discretizado por medio de la transformada de Tustín (5c), así el tiempo discreto de la componente integral esta dado por (11) y se obtiene así:

$$U_i(kh) = \frac{1}{T_i}\frac{h}{2}\left(\frac{q+1}{q-1}\right) e_i(kh)$$

$$U_i(kh) = \frac{h}{2T_i}\left(1 + \frac{2}{q-1}\right) e_i(kh)$$

$$U_i(kh) = \frac{h}{2T_i} e_i(kh) + \frac{h}{T_i}\frac{1}{q-1} e_i(kh)$$

Sea z(kh) es una variable auxiliar de estado definida por:

$$z(kh) = \frac{1}{q-1} e_i(kh) \tag{11}$$

Y pasando a espacio de estados se obtiene (12)

$$z(kh+h) = z(kh) - e(kh)$$
$$u_i(kh) = \frac{h}{T_i} z(kh) + \frac{h}{2T_i} e(kh) \tag{12}$$

Luego, la ecuación (12) es la representación en espacio de estados del componente integral.

C.3.3. Discretización parte proporcional

Como se puede observar en (13), la parte proporcional no presenta componentes dinámicos de ahí que para su discretización se realiza lo siguiente:
Se define la variable:

$$U_p(s) = bR(s) - Y(s) \tag{13}$$

La discretización del componente proporcional, se realiza tomando *kh* muestras a R(s) y Y(s) así:

$$u_p(kh) = br(kh) - y(kh) \tag{14}$$

Finalmente las ecuaciones (6), (11) y (14) corresponden al sistema discretizado del controlador PID paralelo sin antiwindup.

C.4. DISCRETIZACION PID PARALELO CON ANTIWINDUP

La discretización del controlador PID paralelo de la ecuación (1), se realiza igual que la discretización del PID paralelo sin antiwindup en sus componentes proporcional y derivativo, los cambios surgen en la parte integral porque se le debe adicionar las entradas del modo *antiwindup* y de su ganancia de compensación.

$$U_i(s) = \frac{1}{T_i s}\left(E(s) - \left(\frac{\mathrm{U(s) - U_r(s)}}{\mathrm{K_c}}\right)\right) \tag{15}$$

Donde $E_i(s)=R(s)-Y(s)$ es el error del sistema, U(s) es la señal a la entrada del actuador, $U_r(s)$ es la señal a la salida del actuador, Kc es la ganancia de compensación del antiwindup.
Ahora con base en la aproximación de Tustín para discretizar el término integral se obtiene una representación espacio de estados discreta como en (11), con diferencia que:

$$e_i(kh) = r(kh) - y(kh) - \left(\frac{u(kh-h) - u_r(kh-h)}{K_c}\right) \tag{16}$$

Como se puede observar las entradas del modo antiwindup, cambian de U(s)- $U_r(s)$ a u(kh-h)-u_r(kh-h) principalmente porque el tiempo de muestreo es un número finito de siempre mayor que cero y por lo tanto las señales usadas para calcular las siguiente señales de control son las anteriores señales de control.

C.5. DISCRETIZACION PID SERIE SIN ANTIWINDUP

Se parte de la ecuación característica de un PID serie, ecuación (17)

$$U_{serie}(s) = K_c\left(bR_{(s)} - Y_{(s)}D_{(s)}\right) + K_c\left(R_{(s)} - Y_{(s)}D_{(s)}\right)\ I_{(s)} \qquad (17)$$

C.5.1. Discretización parte derivativa

Para discretizar la componente derivativa es necesario obtener una realización mínima de función de transferencia y transformarla a variables en espacio de estados así.
Sea

$$U_d(s) = \frac{sT_d + 1}{{sT_d}/{N} + 1} Y(s)$$

Se obtiene una realización mínima de la función de transferencia:

$$U_d(s) = \left(N - \frac{{N(N-1)}/{T_d}}{s + {N}/{T_d}} \right) Y(s) \qquad (18)$$

Ahora se define una variable auxiliar:

$$X(s) = \left(\frac{1}{s + {N}/{T_d}} \right) Y(s) \qquad (19)$$

Sustituyendo (19) en (18), aplicando Anti-Transformada de Laplace se obtiene el sistema en espacio de estados del componente derivativo así:

$$\dot{x} = \left(-\frac{N}{T_d}\right)x + y$$
$$U_d = \left(-\frac{N(N-1)}{T_d}\right)x + Ny \qquad (20)$$

Ahora al discretizar un sistema de tiempo continuo se debe cambiar el integrador de tiempo continuo por uno de tiempo discreto. Dicho integrador puede ser:

Euler en atraso (21a)

Euler en adelanto (21b)

Aproximación de Euler (21c)

$$I_F(z) = \frac{hz}{z-1} \quad (a) \qquad I_B(z) = \frac{h}{z-1} \quad (b) \qquad I_T(z) = \frac{h}{2}\left(\frac{z+1}{z-1}\right) \quad (c) \qquad (21)$$

Para poder realizar la discretización se utiliza la aproximación poligonal de Tustín (ecuación 21c), porque este aporta la mayor precisión de los métodos numéricos, en la aproximación, h representa el tiempo de muestreo del sistema de control. Luego de seleccionar el integrador se debe transformar el diagrama de bloques a uno de la forma:

$$x(kh+h) = \Phi x(kh) + \Gamma y(kh)$$
$$u_d = C_d x(kh) + D_d y(kh) \qquad (22)$$

Para convertir el sistema discretizado en un sistema basado en el integrador de Tustín, sustituyendo (21c) en (20), los valores de Φ, Γ, C_d y D_d se encuentran de la siguiente forma:

$$\Phi = 1 + hA\left(1 - \frac{h}{2}A\right)^{-1} = \frac{2T_d - hN}{2T_d + hN}$$
$$\Gamma = B + hA\left(1 - \frac{h}{2}A\right)^{-1}B = \frac{2T_d}{2T_d + hN}$$
$$C_d = hC\left(1 - \frac{h}{2}A\right)^{-1} = \frac{2Nh(1-N)}{2T_d + hN} \qquad (23)$$
$$D_d = \frac{h}{2}C\left(1 - \frac{h}{2}A\right)^{-1}B + D = -\frac{N(2T_d + h)}{2T_d + hN}$$

La ecuaciones (20) y (23) corresponden a la implementación en espacio de estados en tiempo discreto de la componente derivativa, ecuación (18).

C.5.2. Discretización parte integral

Para la discretización de la componente integral se define las variables en (24) y (25).

$$U_i(s) = \frac{1}{T_i s} E_i(s) \tag{24}$$

$$E_i(s) = R(s) - (U_d s) \tag{25}$$

Para la discretización de $E_i(s)$ se toman *kh* muestras a R(s) y Y(s), así el sistema discreto para (25) es (26).

$$e_i(kh) = r(kh) - U_d(kh) \tag{26}$$

El componente integral de $U_i(s)$, $1/T_i s$ es discretizado por medio de la transformada de Tustín (21c), así el tiempo discreto de la componente integral esta dado por (11) y se obtiene así:

$$U_i(kh) = \frac{1}{T_i}\frac{h}{2}\left(\frac{q+1}{q-1}\right) e_i(kh)$$

$$U_i(kh) = \frac{h}{2T_i}\left(1 + \frac{2}{q-1}\right) e_i(kh)$$

$$U_i(kh) = \frac{h}{2T_i} e_i(kh) + \frac{h}{T_i}\frac{1}{q-1} e_i(kh)$$

Sea z(kh), una variable auxiliar de estado definida por (27)

$$z(kh) = \frac{1}{q-1} e_i(kh) \tag{27}$$

Y pasando a espacio de estados se obtiene (28)

$$\begin{aligned} z(kh+h) &= z(kh) - e(kh) \\ u_i(kh) &= \frac{h}{T_i} z(kh) + \frac{h}{2T_i} e(kh) \end{aligned} \tag{28}$$

Luego la ecuación (28) es la representación en espacio de estados del componente integral.

C.5.3. Discretización parte proporcional

Como se puede observar en (29), la parte proporcional no presenta componentes dinámicos de ahí que para su discretización se define la variable.

$$U_p(s) = bR(s) - U_d(s) \tag{29}$$

La discretización del componente proporcional, se realiza tomando *kh* muestras a R(s) y Y(s) así,

$$u_p(kh) = br(kh) - U_d(kh) \tag{30}$$

Finalmente las ecuaciones (22), (28) y (30) corresponden al sistema discretizado del controlador PID serie sin anti-windup.

C.5.4. Discretización PID serie con *antiwindup*

La discretización del controlador PID serie con antiwindup, se realiza igual que la discretización del PID serie de la ecuación (17), la única diferencia es en la discretización de su componente integral porque se deben adicionar las entradas del modo *antiwindup* y de su ganancia de compensación.

$$U_i(s) = \frac{1}{T_i s}\left(E(s) - \left(\frac{\mathrm{U(s) - U_r(s)}}{\mathrm{K_c}}\right)\right) \tag{31}$$

Donde: $E_i(s)=R(s)-Y(s)$ es el error del sistema, U(s) la señal a la entrada del actuador, $U_r(s)$ la señal a la salida del actuador y Kc la ganancia de compensación del anti-windup. Ahora con base en la aproximación de Tustín para discretizar el término integral se obtiene una representación espacio de estados discreta.

$$e_i(kh) = r(kh) - U_d(kh) - \left(\frac{u(kh-h) - u_r(kh-h)}{K_c}\right) \tag{32}$$

Como se puede observar las entradas del modo anti-windup, cambian de $U(s)-U_r(s)$ a $u(kh-h)-u_r(kh-h)$ principalmente porque el tiempo de muestreo es un número finito siempre mayor que cero y por lo tanto las señales usadas para calcular las siguiente señales de control son las anteriores señales de control.

ANEXO D ESTRATEGIAS AWBT

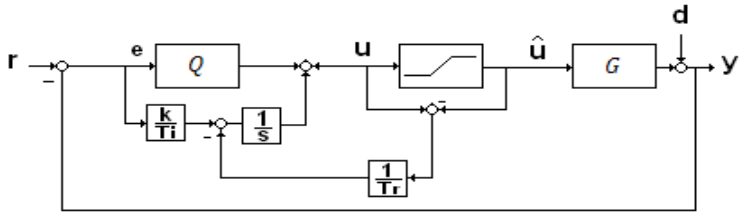
Dentro de la investigación en el área de estrategias de control existen técnicas para eliminar efectos indeseados en la respuesta de un proceso; a continuación se describen algunas estrategias descritas en (Kothare et al, 1993), usadas para formar esquemas AWBT, algunos de estos esquemas fueron propuestos en su inicio únicamente tomando en cuenta la saturación del actuador, mientras que otras permiten la consideración de las no linealidades que presentes en este.

D1. TECNICAS ANTIWINDUP

D.1.1. Anti-reset Windup

Consiste en una realimentación extra en el controlador midiendo la salida $\hat{u}$ del actuador y formando una señal de error como la diferencia entre la salida **u** del controlador y la salida $\hat{u}$ del actuador. Esta señal de error es alimentada a la entrada del integrador a través de una ganancia 1/Tr, como se muestra en la figura 31.

Figura 31. Estrategia Anti-Reset windup



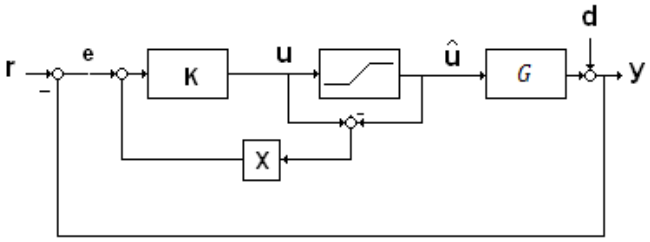
Fuente: Tomado de (Kothare et al, 1993).

Cuando el actuador se satura la señal de realimentación $\mathbf{u}$-$\hat{u}$ intenta manejar el valor de esta señal a 0 por un re-calculo de la integral, tal que la salida del controlador esta exactamente al límite de la saturación.

D.1.2. *Conventional anti-windup (CAW)*

Esta técnica adopta una estrategia similar al *Anti-reset windup*, en este sentido CAW puede considerarse como una extensión de este, la compensación es provista por las alimentación de la diferencia $\mathbf{u}$-$\hat{\mathbf{u}}$ a través de una matriz X de alta ganancia a la entrada del controlador e. La colocación de la matriz puede apreciarse en la figura 32.

Figura 32. Estrategia *Conventional Anti-windup*



Fuente: Tomado de (Kothare et al, 1993).

Típicamente X=αI, donde α>>1 es un escalar, y I es la matriz identidad.

D.1.3. HANUS controlador condicionado

La técnica condicionante fue inicialmente formulada como una extensión del método *back calculation*. En esta técnica el *windup* se considera como una falta de consistencia entre los estados internos del controlador y la entrada a la planta. La consistencia es restaurada modificando las entradas al controlador de tal manera que esas entradas modificadas llamadas referencias realizables habiendo sido aplicadas al controlador, la salida no debería ser diferente de la entrada de control a la planta.

D.1.4. Observador basado en *anti-windup*

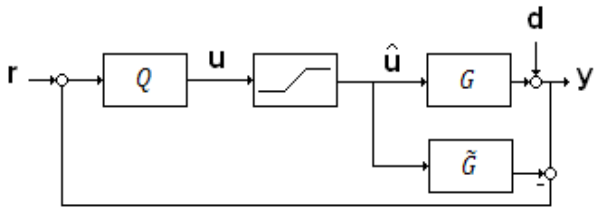
Partiendo de la interpretación de que el problema *windup* es que los estados del controlador no corresponden a la señal de control siendo esta alimentada a la planta. Esta imprecisión en el vector de estados del controlador es debido a la falta de estimadores correctos de los estados del controlador en presencia de la no linealidad del actuador.

Para obtener estados estimados correctos y evitar el *windup*, *Äström* sugiere que un observador sea introducido dentro del controlador. En lugar de tener un controlador y un observador por separado, ambos están integrados dentro de un esquema de la forma de compensador AWBT. De esta manera el observador viene dentro de la estructura del controlador únicamente en presencia de la no linealidad del actuador.

D.1.5. Control por Modelo Interno (IMC)

Este tipo de estructura nunca se pretendió usar como esquema *Anti-windup*, sin embargo puede ser aplicada para dar solución al problema del windup en el caso en donde el sistema es estable a lazo abierto. La aplicación de IMC es estudiado en (Cohen et al, 1995). La figura 33 muestra la estructura IMC con un actuador no lineal. Si el controlador es implementado en configuración IMC y la limitación del actuador no causa problemas de inestabilidad con tal que la señal de control limitada sea enviada a la planta y al modelo. Bajo la suposición que no hay mal emparejamiento de la planta $G = \tilde{G}$, se puede mostrar que la estructura IMC continua siendo eficaz a lazo abierto y la estabilidad es garantizada por la estabilidad de la planta G y del controlador IMC (Q).

Figura 33. Estructura IMC



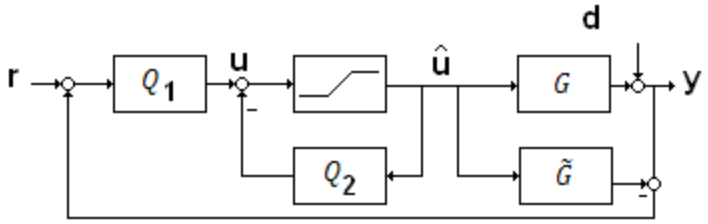
Fuente: Tomado de (Cohen at al, 1995).

El controlador IMC ofrece la posibilidad de implementar algoritmos complejos sin generar resultados complejos de estabilidad solo si no hay desajustes de planta-modelo.

Diseño de antiwindup para IMC. En el método anterior se muestra algunas ventajas del IMC en especial que no causa problemas de estabilidad, desafortunadamente el costo de dicha estabilidad está en que el desempeño el cual es un poco lento. Esto es porque la salida del controlador es independiente de la salida de la planta en régimen lineal y no lineal. Mientras esto no opaque en el régimen lineal, su aplicación en el régimen no lineal

esta en que el controlador no es “conciente” del efecto de esta acción en la salida, resultando en algunos lentitudes. Este efecto afecta más cuando el control IMC tienes rápida dinámica donde son alcanzados por la saturación. Sin embargo, si el IMC es diseñado para optimizar el rendimiento no lineal, esto puede dar rendimiento satisfactorio para la saturación del sistema. La implementación *anti-windup* del control IMC se muestra en la figura 34, en donde $Q1$ y $Q2$ se pueden considerar como funciones de transferencia estables (Zheng et al, 1994).

Figura 34. Implementación Anti-windup de IMC



Fuente: Tomado de (Zheng at al, 1994).

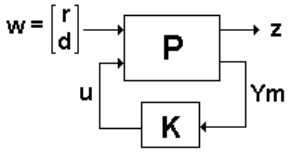
D.1.6. El filtro de Kalman extendido

Este esquema es una implementación AWBT aplicable a compensadores que se basan en observadores, esta implementación es desarrollada para mantener la estimación de los estados validos en el observador independiente de algunas no linealidades entre la salida del controlador y la entrada de la planta.

Considerando el sistema línea idealizado de la figura 35, con w=[r d]’, ym=[r y_1]’, P y K so asumidos como sistemas LTI de dimensión finita, la planta P(s) con el estado x es descrita por las ecuaciones de estado en (1).

$$\begin{aligned} \dot{x} &= Ax + B_1 r + B_2 w + B_3 u \\ z &= C_1 x + D_{11} r + D_{12} w + D_{13} u \\ y_1 &= C_3 x + D_{31} r + D_{32} w \end{aligned} \qquad (1)$$

Figura 35. Diseño Lineal Idealizado



Fuente: Tomado de (Kothare et al, 1993).

En la realización el componente r, se sume que esta disponible para K(s) sin ruido, K(s) es el controlador observador estándar de estado realimentado con estado x, y sus ecuaciones de estado pueden ser escritas como:

$$\begin{aligned} \dot{x} &= A\hat{x} + B_1 r + B_3 u + L(y_1 - C_3\hat{x} - D_{31} r) \\ u &= -F\hat{x} \end{aligned} \tag{2}$$

Donde L es la ganancia del observador y F es la ganancia de realimentación de estado. En presencia de la no linealidad N entre la salida del controlador y la entrada de la planta, se tiene:

$$\hat{u} = N(u) \neq u = -F\hat{x} \tag{3}$$

Así, el observador puede ser estimado por el estado real de la planta. Esto es porque la ecuación de salida del espacio de estados del observador anterior asume que $\hat{u} = u = -F\hat{x}$, que no es el caso en la presencia de la no linealidad *N*, esto puede resultar en windup. Para proveer compensación *anti-windup*, las ecuaciones del observador deben ser modificadas para que el estimador del estado se base en la entrada actual $\hat{u}$ a la planta. Así el compensador observador de estado realimentado modificado esa dado por (4).

$$\begin{aligned} \dot{x} &= A\hat{x} + B_1 r + B_3\hat{u} + L(y_1 - C_3\hat{x} - D_{31} r) \\ &= (A - LC_3)\hat{x} + (B - LD_{31})r + B_3\hat{u} + Ly_1 \\ u &= -F\hat{x} \end{aligned} \tag{4}$$

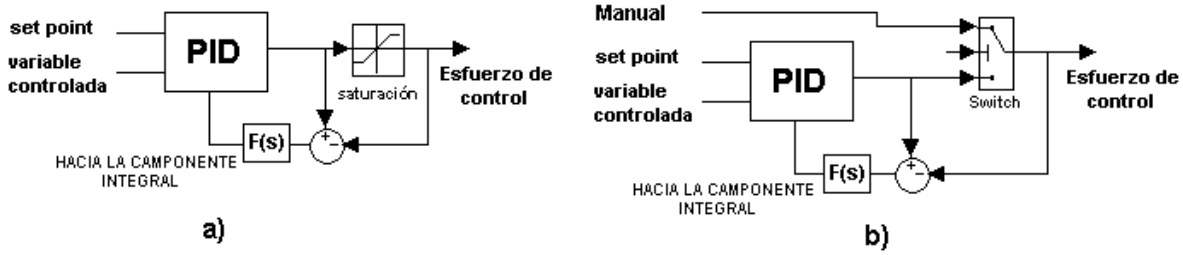
D.2. IMPLEMENTACIÓN DE LA TÉCNICA AWBT

La implementación de las técnicas antiwindup se realizan con base en la saturación del actuador o en las no-linealidades del mismo; para la realización de los experimentos se tendrá en cuenta que la técnica utilizada se base en la saturación del actuador, debido a esto se ha seleccionado la Técnica condicionante para la implementación del AWBT. Es importante resaltar que usualmente técnicas *antiwindup* también son implementadas como técnicas *bumpless transfer*, de ahí que la Técnica Condicionante usada como técnica *anti-windup* es una solución para la técnica de Transferencia Condicionante con la

que se logra eliminar el efecto *bumper transfer*, ya que la esencia de ambos métodos es la misma: hacer que señales manual y automática se aproximen lo más cercano posible una a la otra (Peng et al, 1996). Aunque las técnicas tiene la misma filosofía, se podría llegar a la confusión y pensar que los fenómenos son los mismos, de debe tener en cuenta que *Windup o* reajuste excesivo *es* un *f*enómeno de sobre saturación de la componente integral del PID que puede causar sobre-picos en la respuesta temporal del sistema generalmente ante cambios en el valor de consigna o ante perturbaciones y el *Bump Transfer* es un fenómeno que produce cambios bruscos en la variable controlada y sobreesfuerzos en el actuador, debido al cambio de operación del controlador de un estado manual a automático.

La implementación de la Técnica condicionante y/o Transferencia condicionante se basa en la realimentación hacia la componente integral del PID, de la resta de la señal de entrada del controlador con la señal de salida del actuador. Para la aplicación de dichas técnicas realmente se debe utilizar un sensor que mida la señal con el fin de obtener los valores actuales y de esta manera poder realizar la realimentación para asi hacer los ajustes que evitaran la presencia de fenómenos no deseados; la desventaja de esta solucion es que implica costos en la adquisición de los instrumentos. Debido a este inconveniente y con el propósito de realizar el experimento de una manera mas cercana a la real, la implementación de las técnicas se realizarán por medio de la realimentación de la resta de la señal a la entrada y a la salida de un bloque saturador que simula las restricciones del actuador, para la técnica condicionada, ver figura 36a, y para la transferencia condicionada se tomara la señal de la resta de la señal de salida del controlador y a señal de entrada al actuador ver figura 36b. Dichas señales son señales normalizadas (entre 0 y 1).

Figura 36. Técnica AWBT.



a) Técnica Condicionante **b) Transferencia Condicionante**

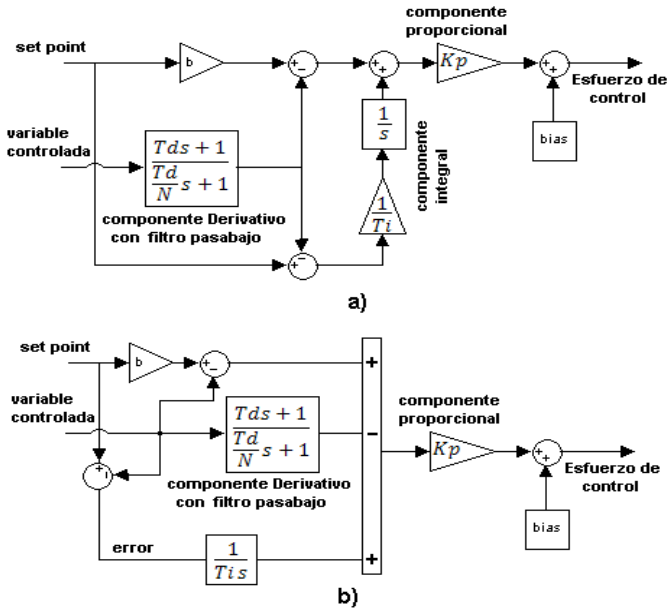
D.3. EXPERIMENTO EFECTO WINDUP EN LA RESPUESTA DEL PROCESO DE CONTROL DE NIVEL

El experimento de simulación del efecto *windup* consiste en la presencia de una perturbación constante en el flujo de entrada o variable manipulada VM durante un periodo de tiempo determinado y de esta manera analizar la reacción del sistema ante grandes perturbaciones.

El proceso de nivel esta representado por la ecuación 5, denominada planta de POMTM. Para la simulación se utilizará la estrategia de control PID Industrial anto en suestructura serie (figura 37a) y en su estrutura Paralela (figura 37b), sintonizados como reguladores sintonizados por medio el método IAE de *Kaya* y *Scheib* tanto en su forma continua como discreta.

$$G(s) = \frac{Ke^{-Ls}}{Ts + a} \quad \text{con a>0} \tag{5}$$

Figura 37. Estructura de control PID.



La constante N (ganancia en alta frecuencia) del filtro de la componente derivativa se encuentra en un valor de 10 para el controlador serie y de 3 para el paralelo, el pesaje de la referencia *b* se ha fijado en 1 debido a que la referencia no cambia con el tiempo. Además se le agregó el parámetro Uo, a la salida del controlador con un valor de 0,5. En la tabla 29 se pueden observar las constantes de sintonización para el PID.

Tabla 29. Constantes sintonización estructura PID

Estructura PID	Kc	Ti	Td
Serie continuo y discreto	0.91	1.1823	0.6497
Paralelo continuo y discreto	1.3570	1.3667	0.5784

Para la realización del experimento se considera un flujo a la entrada del actuador en condiciones de operación estables de 30 gpm, en t=20 segundos, aparece la perturbación ocasionando una disminución en el flujo de entrada a válvula de 10 gpm, esto a su vez genera una caída estrepitosa en el nivel del tanque, ver figura 38; el control PID mediante la señal de control le indica a la válvula que se abra al máximo para alcanzar nuevamente la referencia deseada. La perturbación estará presente durante 25 segundos, la válvula esta totalmente abierta, alcanzando su máximo desempeño y la variable controlada se encuentra lejos de alcanzar el valor en estado estable; el error aun existe y debido a su presencia el control PID intenta corregirlo mediante un incremento en su valor de salida, llegará un tiempo en que ya no se podrá incrementar la salida del controlador debido a que se encuentra saturado, pero mientras el error esté presente el PID lo seguirá integrando, ver figura 39. Ahora, en t=45 segundos, el flujo en la entrada se incrementa y vuelve a su valor de estado estable, la perturbación desaparece y el nivel en el tanque comienza a incrementar hasta alcanzar y sobrepasar la referencia, en estos momentos apenas empieza el control PID a encontrar valores negativos en el valor del error (integrando hacia abajo) para que compense la integración inicial y después de un tiempo se alcancen los valores de estado estable.

Para resolver ese inconveniente se implementó la estrategia *anti-windup* denominada **Técnica Condicionante**, como se explicó anteriormente, consiste en generar una señal de realimentación de la diferencia entre la señal de control saturada y la no saturada y

esta señal se usa para reducir la entrada al integrador, cuando la salida del controlador excede los límites del actuador.

En las graficas siguientes la respuesta de la VC y del EC para el sistena continuo se presenta en color azul y para el sistema en tiempo discreto la VC se presenta en color verde y el EC encolor rosado.

Figura 38. VC PID Serie ante una gran perturbaciòn – Fenómeno windup

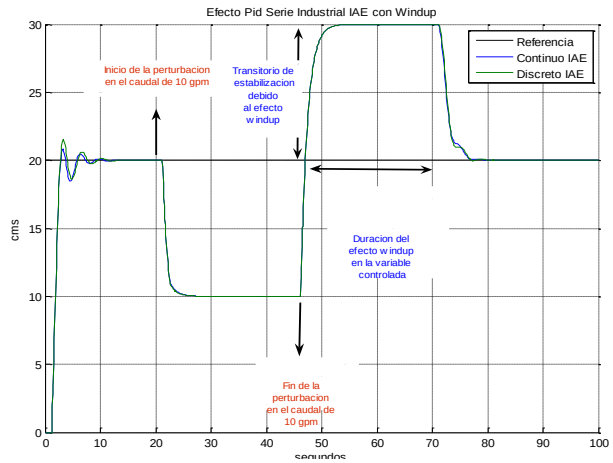


Figura 39. EC PID Serie – Fenómeno Windup

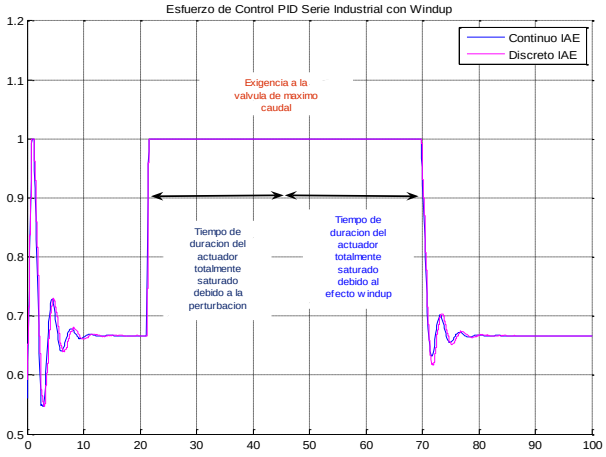


Figura 40. VC PID Serie con Estrategia AWBT.

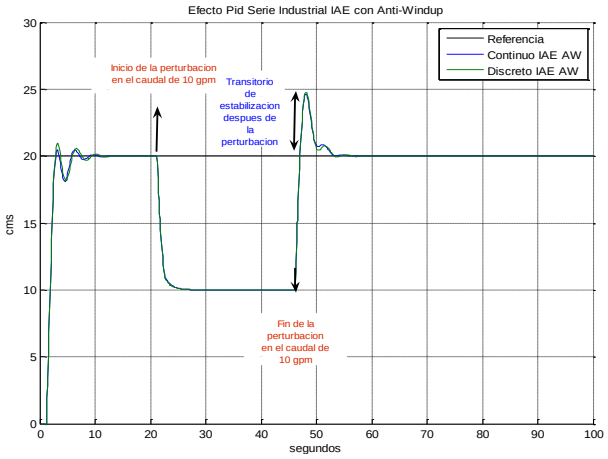
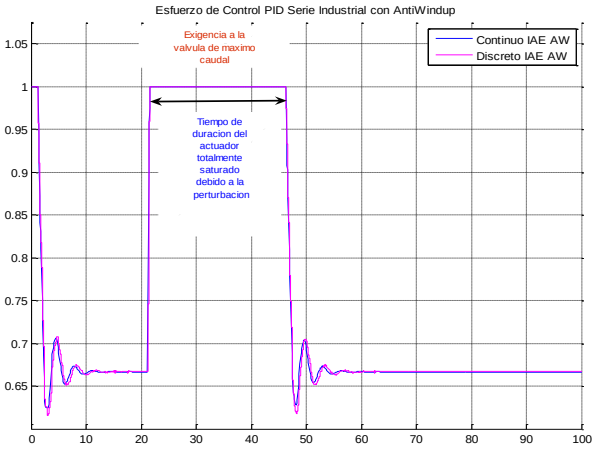


Figura 41. EC PID Serie con Estrategia AWBT



En la figura 40 se aprecia como la VC no es capaz de seguir la referencia ante la gran perturbacion y solo alcanza el valor de consigna una vez la perturbacion desaparece, el

efecto de la pertubacion es solamente un corto transiente hasta estabilizarce en el valor de consigna, en cuanto al EC de la figura 41, el actuador se satura solamente mientras la perturbación esta presente en el proceso.

En cuanto a la VC, con el PID paralelo de igual manera que en el PID serie, cuando la perturbación desaparece, produce un salto en la VC, figura 42, por la continua integración del error hecha por la compnente integral, originando la saturacion del control PID, figura 43.

Figura 42. VC PID Paralelo ante una gran perturbacion – Fenómeno windup

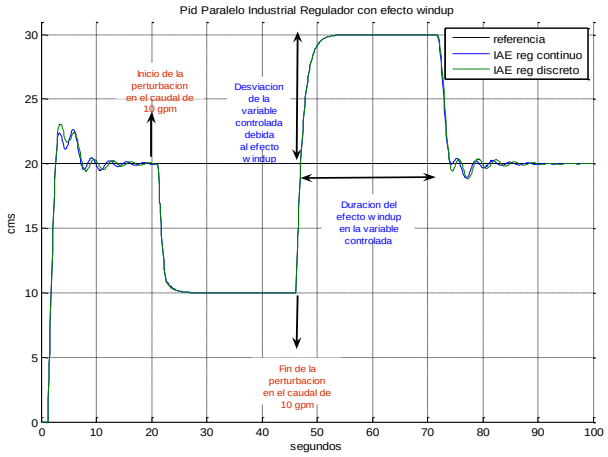


Figura 43. EC PID Paralelo – Fenómeno Windup

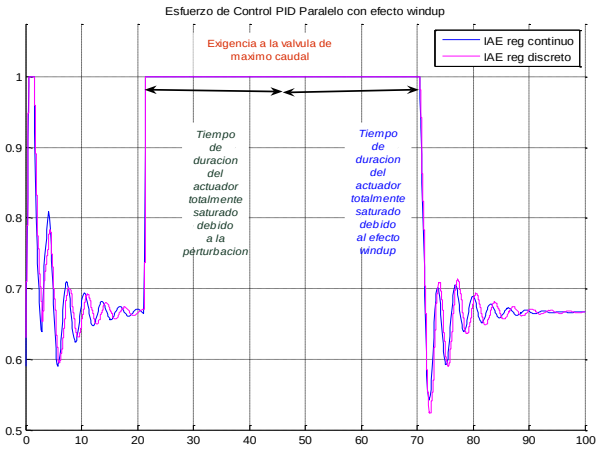


Figura 44. VC PID Paralelo con Estrategia AWBT

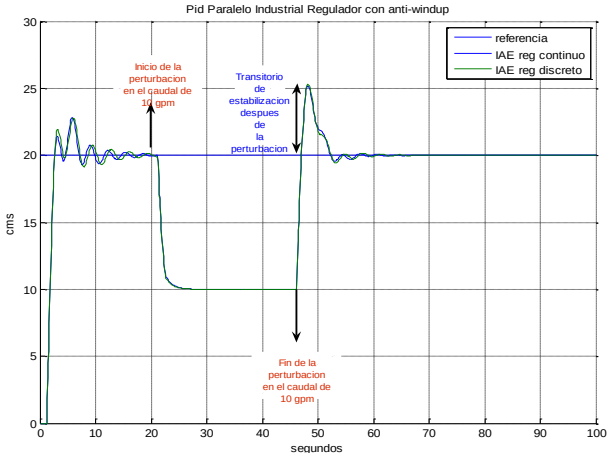
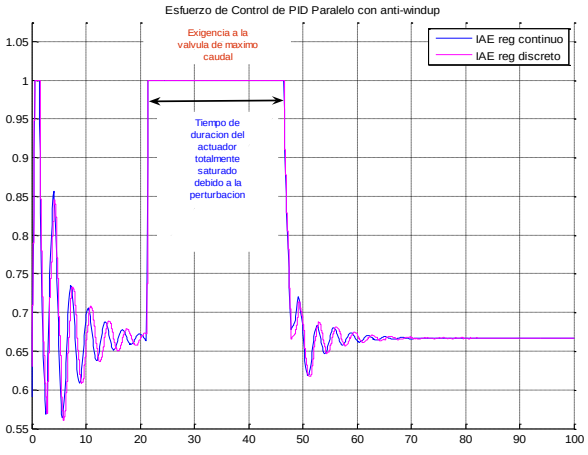


Figura 45. EC PID Paralelo con AWBT



Con la tecnica AWBT activada, en la figura 44 se aprecia como la VC igual que en el caso del PID serie tampoco es capaz de seguir la referencia ante la gran perturbacion y solo alcanza el valor de consigna una vez la perturbacion desaparece, el efecto de la pertubacion es solamente un corto transiente hasta estabilizarce en el valor de consigna, en cuanto al EC de la figura 45, el actuador se satura solamente mientras la perturbación esta presente en el proceso, tal como sucede en el PID serie.

A continuacion se presentan dos tablas comparativas de los resultados dinamicos, en la tabla 30 se realiza la medición de la respuesta de la VC y en la tabla 31 se mide cuantitativamente la respuesta del EC.

Tabla 30. Tabla comparativa de aspectos relevantes de la VC.

CONTROLADOR PID			**DESVIACION**	**TIEMPO DE DESVIACION POR WINDUP 1%**	**TIEMPO DE PICO**
Serie	continuo	Sin Anti-windup	100%	31.59 seg.	Indefinido
		Con Anti-windup	64.2%	5.87 seg	1.12 seg
	discreto	Sin Anti-windup	100%	32.19 seg.	Indefinido
		Con Anti-windup	65%	5.97 seg.	1.12 seg
Paralelo	continuo	Sin Anti-windup	100%	30.71 seg	Indefinido
		Con Anti-windup	67.73%	10.89 seg	1.24 seg
	discreto	Sin Anti-windup	100%	30.29 seg	Indefinido
		Con Anti-windup	68.73%	12.86 seg	1.24 seg

Tabla 31. Tabla comparativa de tiempo de máximo EC.

CONTROLADOR PID			**TIEMPO DE SATURACIÓN POR WINDUP 1%**
Serie	continuo	Sin Anti-windup	25 seg.
		Con Anti-windup	0 seg
	discreto	Sin Anti-windup	25 seg.
		Con Anti-windup	0 seg.
Paralelo	continuo	Sin Anti-windup	25 seg
		Con Anti-windup	0 seg
	discreto	Sin Anti-windup	25 seg
		Con Anti-windup	0 seg

D.4. EXPERIMENTO EFECTO BUMP TRANSFER EN LA RESPUESTA DEL PROCESO DE LA PLANTA DE NIVEL

Este experimento consiste en la simulación del proceso de nivel en un tanque y del efecto *Bump Transfer* el cual se presenta en dicho sistema ante los cambios en la referencia entre los modos manual y automático, la simulación tendrá una duración de 100 segundos y para realizar los cambios de manual-automático y automático-manual se cuenta con un *switch*. El control del sistema esta a cargo de un controlador PID de estructura paralela y serie como regulador tanto en sus representaciones continua y digital, sintonizados por medio del método ITAE correspondiente a cada estructura, los valores de las constantes de sintonización de cada PID se pueden observar en la tabla 32. El proceso estará representado por una planta POMTM, ver (5).

Tabla 32. Constantes estructuras PID

Estructura PID	Kc	Ti	Td
Serie continuo y discreto	0.8326	1.1867	0.5309
Paralelo continuo y discreto	0.9650	1.8476	0.4176

En t=0 el proceso inicia con una señal de referencia automática representada por un valor de consigna fijo de 15 cms hasta que en t=20 se cambia el *switch* de automático a manual, la señal manual esta compuesta de 4 cambios en escalón.

La señal de control manual tendrá acción directa al proceso durante 50 segundos, es decir en t=70 se cambia el *switch* de manual a automático.

Como se puede ver en la figura 46a, inicialmente la señal de referencia es el modo automático hasta que en t=20 segundos hay un cambio de automático-manual de ahí en adelante la señal de control que actual sobre el proceso es la señal manual, mientras que el PID automático sigue generando su señal de control, debido a que el error esta presente en el proceso la acción integral que genera crece, como resultados la señal del PID alcanza valores diferentes a la señal de control manual. En t=70 segundos se presenta el cambio de manual-automático y como el sistema de control esta sintonizado para seguir cambios en el valor de consigna la señal de salida debería seguir la señal automática, pero debido a que la componente integral del PID ha estado integrando durante todo el tiempo anterior al cambio, generando una señal de error aunque no sea la

que actué sobre la planta, conlleva a que en ese instante se produzca un fuerte salto y la imposibilidad de seguir la referencia y a la saturación del actuador; solo seguirá la señal de referencia hasta que encuentre valores negativos del error y compense el error almacenado por la integración del PID. En las figuras 46 y 48 se puede apreciar el efecto *Bump Tansfer* por el cambio manual-automático en la VC para el PID serie y paralelo respectivamente y en las figuras 47 y 49 el efecto en el EC para el PID serie y paralelo rspectivamente.

Figura 46. VC PID Serie ante cambios manual automatico– Fenómeno Bump Transfer

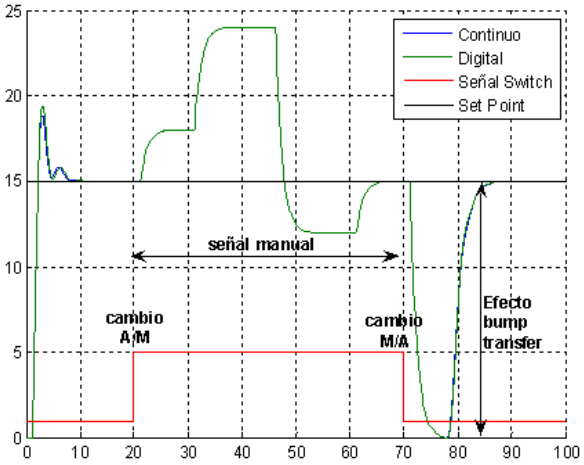


Figura 47. EC PID Serie – Bump Transfer

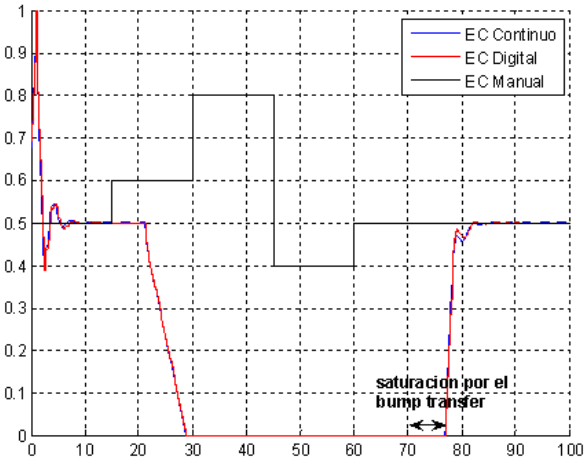


Figura 48. VC PID Paralelo ante cambios manual automatico– Fenómeno Bump Transfer

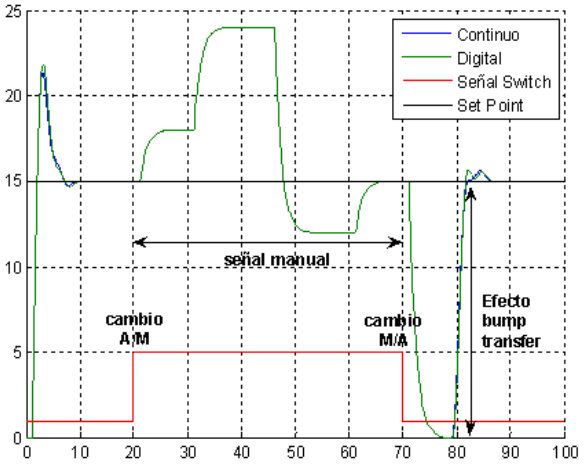
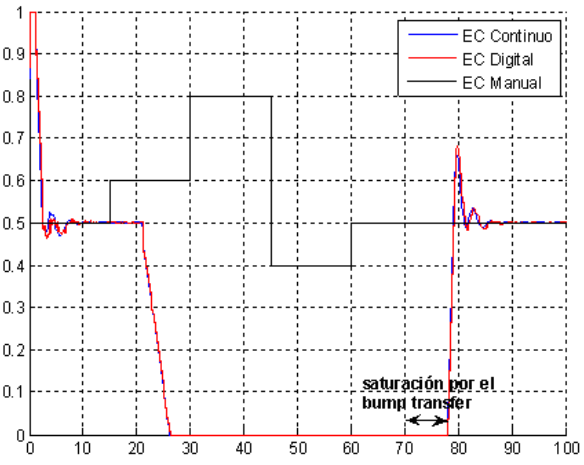


Figura 49. EC PID Paralelo – Bump Transfer

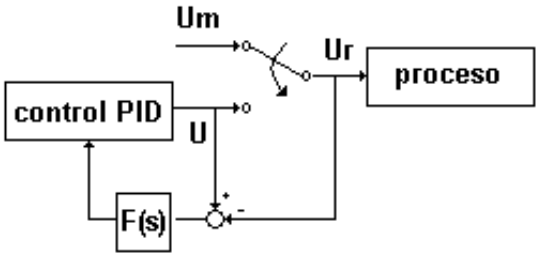


D.5. TRANSFERENCIA CONDICIONANTE PARA ELIMINAR EL FENÓMENO BUMP TRANSFER

Para solucionar el efecto *bump transfer* presentado por la conmutación manual-automático se aplicará la técnica Transferencia Condicionante, que es equivalente a la Técnica Condicionante para eliminar el efecto windup del integrador.

Esta técnica consiste en la generación de un lazo extra con una señal generada por la diferencia de señales saturada y no saturada del actuador (a la salida del controlador y a la entrada de la planta) hacia el componente integral del control PID amplificado por una función F(s) que para el experimento equivale a F(s)=1/Kc donde Kc es la ganancia proporcional del controlador. El objetivo de aplicar esta señal realimentada a la componente integral es que las señales manual y automática sean lo mas cercanas posibles. Durante el modo manual *(Ur=Um),* como el controlador no está conectado a *Um*, la señal de control generada por el controlador suele ser diferente de *Ur*. En este caso, tras el cambio, un salto será producido en la entrada de la planta, para eliminar el salto, la salida U del controlador PID deberán hacerse lo más cerca posible de *Um* durante el modo manual, ver figura 50. Entonces el salto en el instante de conmutación será mínimo. Este modo de conmutación es llamado ***Bumpless Transfer***.

Figura 50. Técnica Transferencia Condicionante



Si la salida del controlador *U* es ajustada para que después de cambiar de control manual a un control automático; la salida de la planta sigue la referencia con la misma dinámica como en la respuesta al escalón en lazo cerrado, entonces este modo conmutación se llama **Transferencia Condicionante**. Es decir, después de la conmutación la

transferencia condicionante puede asegurar un buen seguimiento de la referencia. Generalmente el salto es pequeño pero no es eliminado totalmente.

Los resultados obtenidos se pueden apreciar en la figura 51 para la VC y 52 para el EC del PID serie y la figuras 53 para le VC y en la figura 54 el EC para el PID paralelo, en donde el salto al cambiar de modo manual a automático se disminuye considerablemente permitiendo el seguimiento de la consigna por parte de la variable controlada.

Figura 51. VC PID Serie ante cambios manual automatico– Técnica AWBT

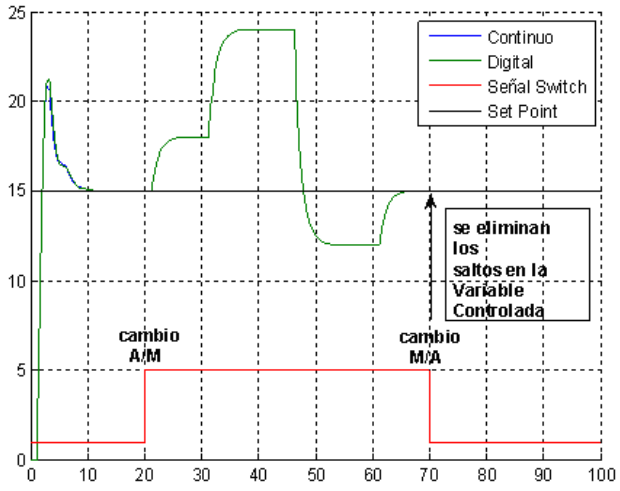


Figura 52. EC PID Serie – Técnica AWBT

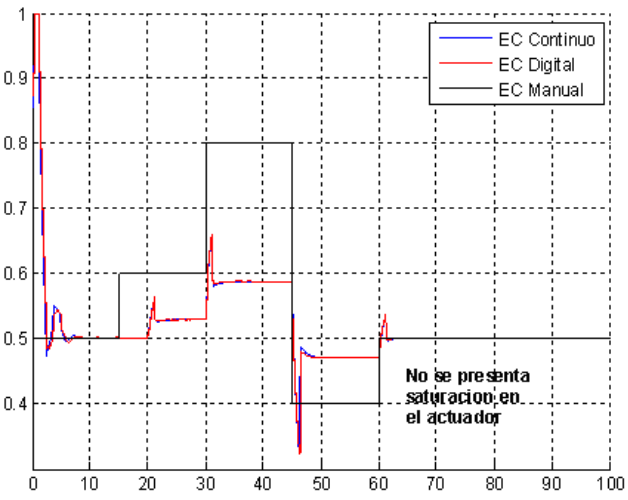


Figura 53. VC PID Paralelo ante cambios manual automatico– Técnica AWBT

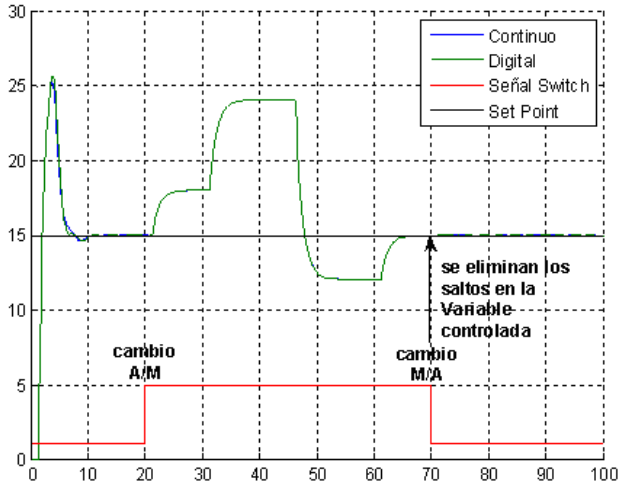
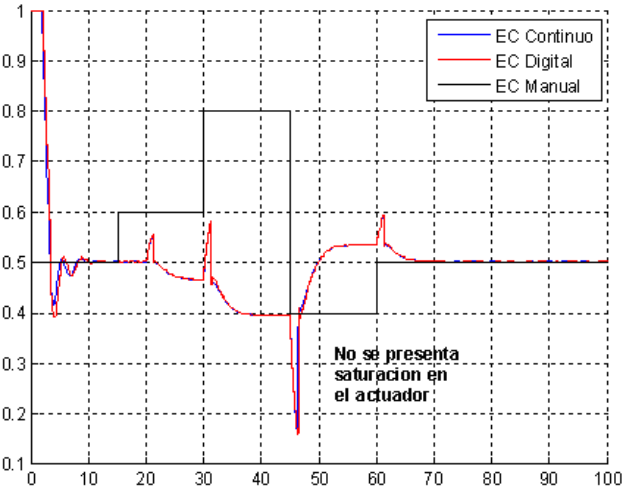


Figura 54. EC PID Paralelo – Técnica AWBT



Ahora se realiza una comparacion cuantitativa entre los controladores PID con y sin tecnica AWBT, mediante parámetros dinamicos como el máximo sobre impulso, el tiempo de saturación del actuador y el tiempo de desviación (tiempo en alcanzar la banda del 2% después del efecto *bump transfer*). Los resultados se presentan en la tabla 34.

Tabla 33. Valores cuantitativos efecto bump transfer y bumpless transfer.

CONTROLADOR PID			SOBREIMPULSO	TIEMPO DE SATURACIÓN	TIEMPO DESVIACION
Serie	continuo	Bump Transfer	50%	7 seg.	16 seg
		Bumpless Transfer	6.5%	NS	NE
	discreto	Bump Transfer	50%	7 seg.	18.44 seg
		Bumpless Transfer	8.3%	NS	NE
Paralelo	continuo	Bump Transfer	50%	8 seg	15.7 seg
		Bumpless Transfer	1.4%	NS	NE
	discreto	Bump Transfer	50%	8 seg	15.91 seg
		Bumpless Transfer	1.4%	NS	NE

NS= No saturación, NE= No existe tiempo de desviación.

D.6. CONCLUSIONES

La presencia de grandes perturbación en un sistema, conlleva a este a presentar exigencias de máximo paso de la variable manipulada, esto se verá reflejado en saturaciones sufridas por el actuador.

Uno de los fenómenos que se presentan ante grandes perturbación es el *windup*, el cual genera grandes sobre-picos en la variable controlada, imposibilitando al sistema de seguir la referencia.

Con base en las técnicas investigadas, la técnica usada para la eliminación del fenómeno windup fue el método de Técnica Condicionante, ya que se basa en la saturación del actuador, y es uno de los métodos que se utilizan dentro de la práctica y la simulación de sistemas; con este método se logran resultados satisfactorios.

Con la Técnica Condicionante se disminuye el valor del sobre-pico, así como el tiempo de establecimiento de la salida del sistema, generados por el reajuste excesivo de la componente integral.

Otro beneficio con la técnica implementada es la disminución del tiempo de saturación del actuador.

La presencia de conmutaciones de modo manual a automático en un sistema de control sin estrategias para eliminar fenómenos como el *bump transfer*, genera en el sistema sobre impulso en la variable controlada, alargamiento del transitorio, mal seguimiento de consigna y la saturación del actuador por largos periodos de tiempo.

La implementación de la técnica Transferencia Condicionante disminuye en gran parte la presencia de saltos abruptos en la variable controlada por la conmutación en la señal de referencia entre sus modos manual y automático, puesto que hace que la salida del control PID sea lo mas cercana posible a la señal de control manual; adicionalmente permite un buen seguimiento de la consigna y disminución de la saturación del actuador.

D.7. BIBLIOGRAFIA

COHEN I., R. HANUS and J.P. VANBERGEN. The impact of the control structure on the closed loop efficiency. Journal A, 26(1):pp 18-25, 1985.

KOTHARE, M.V,. P.J. CAMPO and M. MORARI. A Unified Framework for Study of Anti-Windup Designs. Article of divulgation, Pasadena California, USA, pp 1-27, June 1993.

PENG, Y, D. VRANCIC and R. HANUS. Anti-Windup, Bumpless, and Conditioned transfer Techniques for PID Controllers. IEEE Control Systems, Vol 16, Issue 4, pp 48-57, August 1996.

ZHENG, A, M. KOTHARE and M. MORARI. Anti-windup design for Internal Model Control. International Journal of Control, 1994.

ANEXO E SISTEMAS OPERATIVOS DE TIEMPO REAL

Con el avance de la tecnología cada vez los computadores son de menores tamaños, rápidos y fiables, y el campo de aplicación se extiende a muchas más tareas que antes. Construidos al principio para realizar cálculos matemáticos, en la actualidad se encuentran en áreas como el hogar, el trabajo, entretenimiento, comunicaciones etc.

Una de las áreas de expansión mas rápida de la explotación de computadoras es el control en este caso la función principal no es la de procesar información, pero que necesita de dicho procesamiento de información con el fin de realizar su función principal Por ejemplo un horno microondas controlado por microprocesador, su función principal es calentar la comida, sin embargo dependiendo del tipo de alimento se deben ejecutar programas diferentes. Este tipo de aplicaciones de computador se conocen genéricamente como de tiempo real o embebidas.

De esta manera los computadores facilitan algunas tareas de nuestras vidas; Uno de los elementos más importantes en sistema de cómputo es el sistema operativo, el cual es un programa que actúa como intermediario entre el usuario y el hardware de un computador y su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar programas. El objetivo principal de un sistema operativo es, entonces, lograr que el sistema de computación se use de manera cómoda por parte del usuario, y el otro es permitir la comunicación entre los diferentes dispositivos hardware del computador y que estos se empleen de manera eficiente.

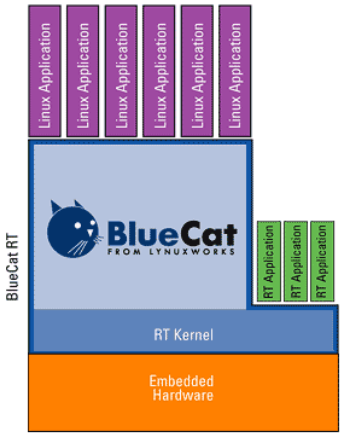
El sistema operativo proporciona los medios para usar adecuadamente los componentes de un sistema de computación (hardware, software y datos) durante el funcionamiento del sistema y sin llevar a cabo ninguna función útil por sí mismo, proporcionando un entorno dentro del cual otros programas pueden realizar trabajo útil. Sin embargo para sistemas embebidos o de tiempo real existe un tipo de sistema operativo que le permite características puntuales como acceso al planificador de tareas, mejor respuesta a interrupciones externas, comunicación y sincronización entre procesos, este tipo de sistemas operativo son llamados como Sistemas operativos de tiempo real. Existe una gran cantidad de este tipo de sistemas operativos tanto libres como comerciales, este

anexo trata de dar una visión general de este tipo de sistemas y dar un panorama de lo que cada uno de ellos puede ofrecer.

E.1. *BLUECAT RT*

BlueCat RT es una versión de Linux para tiempo real y embebido. Es una solución híbrida entre el sistema operativo embebido *BlueCat (LinuxWorks)* y el sistema *RTLinux/Pro (FSMLabs)*. Licencia: Comercial. *Open source*. Su arquitectura se presenta en la figura 55.

Figura 55. Arquitectura BlueCat RT.



Ventajas. Asegura el manejo de interrupciones con tiempo crítico y otras operaciones hardware de bajo nivel para implementar el *kernel* de Linux como un *thread* pequeño y de alta respuesta en un sistema operativo de tiempo real. Este mini sistema operativo de tiempo real captura las interrupciones y otras funcionalidades de bajo nivel para el procesamiento preliminar antes de pasárselas al *thread* que ejecuta el *kernel* de Linux.

Los dos sistemas operativos forman una solución híbrida que ejecuta en un sólo procesador, donde la mayoría de las aplicaciones operan bajo el *kernel* estándar de Linux y un subconjunto hace uso de las características de tiempo real de *RTLinux Pro*, que forma el *kernel* de tiempo real. Esto hace de *BlueCat* una solución ideal para aquellas aplicaciones en las que sus tareas no requieran tiempo real la mayor parte del tiempo.

Incluye un API de tiempo real compatible con POSIX y con IEEE 1003.13, lo que permite que aplicaciones que no requieren una completa funcionalidad de Linux puedan ser ejecutadas en esta capa.

Basado en estándares abiertos y en código abierto, lo que facilita la migración de aplicaciones hacia *BlueCat RT.*

Características.

Estabilidad. Es una distribución estable porque se basa en un sistema operativo embebido con cierto grado de comercialidad y por tanto se diseña para que sea flexible, escalable y con productividad inmediata.

Flexibilidad. Soporte de un completo rango de configuraciones, desde pequeños dispositivos para el consumidor hasta grandes sistemas multiprocesadores

E.2. RED-LINUX

RED-Linux es una versión de Linux para tiempo real y embebido. Proporciona capacidades adicionales al *kernel* estándar de Linux para proporcionar comportamiento de tiempo real. Licencia: Desconocida

E.2.1. Fundamentos

RED-Linux soporta:

Un núcleo pequeño.

Un tiempo de respuesta rápido para las tareas.

Modularidad y Planificador de CPU reemplazable en tiempo de ejecución.

Un marco general de planificación (*GSF-General Sheduling Framework*).

En teoría se puede usar bajo cualquier distribución de Linux, pero recomiendan usar Debian.

E.3. REDHAWK

Es una versión de tiempo real del código abierto de Linux desarrollada por la empresa *Concurrent Computer Corporation's RedHawk* bajo los estándares industriales y POSIX. *RedHawk* es compatible con la distribución *Red Hat Linux* proporcionando alto rendimiento de entrada/salida, tiempo de respuesta garantizado a evento externos y comunicación entre procesos optimizada. *RedHawk* es ideal para ambientes con complejas aplicaciones de tiempo real como pueden ser la simulación, la adquisición de datos ó los sistemas de control industrial. Licencia: Comercial

E.3.1 Arquitectura

El sistema de administración, las utilidades y los comandos del nivel de usuario son los estándares proporcionados por *Red Hat*, en cambio, para lograr rendimiento de tiempo real, reemplaza el *kernel* de *Red Hat* por uno que es *multithread,* con un *kernel* completamente apropiable, es decir capaz de interrumpir a tareas con menor prioridad, y de baja latencia. *RedKawk* proporciona soporte al multiprocesamiento simétrico, que incluye un control de carga y protección de CPU para maximizar el determinismo y el rendimiento de tiempo real en soluciones críticas.

Características.

Ambiente completo de desarrollo. Ofrece un conjunto completo de herramientas de desarrollo, desde compiladores a *debuggers* y herramientas que muestran la traza de ejecución.

SMP escalable y protección de procesador.

Multithread y *kernel* prioritario.

Planificador basado en la frecuencia.

Reloj de tiempo real y módulo de interrupciones.

Herramientas de desarrollo de aplicaciones de tiempo real.

Depurador de código fuente.

Analizador de eventos mediante traza de código.

Simulador de un planificador.

Monitor de datos de prueba

E.4. ART LINUX

ART Linux es una extensión de tiempo real sobre Linux basado en *RTLinux* y desarrollado por *Youichi Ishiwata*. Licencia: ART Linux. Es un parche sobre el *kernel* estándar de Linux pero que actualmente no se distribuye bajo la licencia GPL ya que contiene código desarrollado por *Ishiwata* en exclusiva. El que no se distribuya bajo la licencia GPL es debido a que *Ishiwata* trabaja para el gobierno y existe una ley de propiedad intelectual del gobierno.

E.4.1. Arquitectura

La arquitectura es similar a la de *RTLinux* por tanto tienen arquitectura de *kernel dual*.

Ventajas

ART Linux permite reusar los drivers de dispositivos existentes en el Linux estándar así como las aplicaciones.

Compatibilidad a nivel de fuente de los drivers con el Linux estándar. Es decir si recompilas el driver, puede seguir siendo usado por las tareas de tiempo real.

No inversión de prioridades. Se resuelven automáticamente los problemas de inversión de prioridades.

Compatibilidad a nivel binario de los programas de usuario con el Linux estándar. No es necesario recompilar los programas con el objetivo de que sean usados por las tareas de tiempo real. Evita indeterminismo en la ejecución de tiempo real debido a las interrupciones ya que trata las interrupciones con una estrategia periódica.

Las tareas de tiempo real ejecutan en modo usuario pero privilegiado, de forma que se beneficia de la protección de memoria y por tanto las tareas de tiempo real son seguras y no pueden provocar fallos en el *kernel.*

E.5. KURT (Kansas University Real Time)

KURT junto con UTIME es una extensión al *kernel* estándar de Linux para proporcionar, bajo demanda, resolución de microsegundos en los relojes y planificación de tiempo real.

Licencia: Desconocida. *Open source*

E.5.1. Arquitectura

UTIME. El *kernel* estándar de linux ofrece 10ms de resolución en la sincronización. Para la mayoría de las aplicaciones este nivel de granularidad es suficiente, pero no para las aplicaciones de tiempo real. UTIME es una modificación al *kernel* de Linux para proporcionar una resolución entorno a los 10 microsegundos. Para llevarlo a cabo usa varios métodos:

Llamada al sistema "*nanosleep".* Mediante esta llamada se permite que los procesos duerman con una resolución de microsegundos.

Temporizadores. Cada proceso tiene acceso a 3 temporizadores. Cuando vencen se envía una señal al proceso. Gracia a UTIME es posible configurar los intervalos de tiempo del temporizador con resolución de microsegundos.

Llamada al sistema *"select*". Esta llamada espera a que varios descriptores de ficheros cambien de estado y en el k*ernel* estándar esta limitada esa espera a una resolución de 10 ms. Gracias a UTIME, permite configurar con resolución de microsegundos.

Llamada al sistema "*poll".* Similar a la anterior, permite configurar con resolución de microsegundos.

Módulo por defecto del *kernel.* Linux permite que nosotros mismo definamos nuestros propios módulos para el *kernel*, con lo que podemos añadir nuestros propios relojes y sus manejadores.

KURT. Proporciona un manejador de eventos con la facilidad de un planificador de tiempo real con resolución de microsegundos gracias a UTIME.

El *kernel* estándar de Linux proporciona 3 políticas de planificación diferentes, que son: SHED_FIFO, SHED_RR y SHED_OTHER, que son suficientes para la mayoría de aplicaciones que no requieren una granularidad de resolución muy fina. Un valor alto de prioridad no garantiza a un proceso su planificación, por eso KURT proporciona un mecanismo de planificación bajo demanda para garantizar a los proceso el acceso a los recursos que necesite y en un orden explicito.

Todas las interacciones con el subsistema KURT se realizan a través de un psudodispositivo (/dev/kurt), que únicamente proporciona 3 operaciones: *open, close* y *ioctl.* Cada vez que se quiera usar KURT simplemente hay que abrir una instancia del dispositivo y cuando se termina, simplemente se cierra.

Para facilitar la programación de aplicaciones KURT dispone de un API que proporciona las siguientes 3 categorías de rutinas:

Operaciones generales y de utilidad. Un ejemplo de utilidad es la función "*kurt_open()",* que abre una instancia del psudodispositivo.
Inicialización, registro y control de procesos. Un ejemplo es la función *"rt_suspend()",* que suspende un proceso por un determinado plazo de tiempo. Antes de registrar los procesos de tiempo real se debe determinar que tipo de planificación van a soportar, puesto que KURT soporta un amplio abanico de modos y niveles de planificación y combinación entre ellos.
Planificación. Un ejemplo es la función "*set_sheduling_task()",* que registra los procesos actuales en el planificador de tareas de KURT.
Para usar este API en los programas de usuario simplemente hay que incluir el fichero de cabecera linux/rt.h del kernel de KURT en el archivo. Más información en el manual de usuario.

UTIME/KURT frente a RTLinux. KURT es capaz de planificar cada tarea en un nivel distinto del planificador según que requerimientos de tiempo real necesite, frente a *RTLinux/RTAI* que planifica todas las tareas de tiempo real en un planificador ejecutivo
E.6. Linux/RK (Linux/Resource Kernel).

Linux/RK significa *Linux/Resource Kernel* y consiste en incorporar extensiones de tiempo real a Linux mediante una abstracción llamada recurso *kernel.* Licencia: Desconocida. *Open source*

Las aplicaciones que requieran acceder a alguno de los recursos se comunican con el recurso *kernel* y realizan una reserva. Sí es aceptada, obtendrá el recurso cuando lo haya solicitado. Un recurso *kernel* es un *kernel* de tiempo real que proporciona el oportuno y garantizado acceso a los recursos del sistema para las aplicaciones que lo requieran.

Características
Actualmente las actividades que soporta Linux/RK son:
Reserva de ancho de banda en acceso a los discos.

Reserva de ancho de banda de la red.
Co-planificación de varios recursos.
Integración con Java para tiempo real.
Lista de control de recursos

E.7. LINUX SRT

Linux-SRT es una extensión al *kernel* de Linux que inició el camino hacia la ejecución de aplicaciones de tiempo real. El *kernel* estándar de Linux no era un sistema operativo multimedia, no garantizaba que el audio, el vídeo u otro proceso critico ejecutará con una transferencia fijada, por eso nació este proyecto con el objetivo de crear un sistema operativo multimedia que proporcionara una calidad de servicio determinada, pero no estricta, de ahí el nombre, *Linux-SRT* (*soft real time*). No es adecuado para sistemas críticos. Este proyecto se ha abandonado actualmente debido al laborioso proceso de modificación del *kernel.* Licencia: GNU/GPL

Arquitectura. La arquitectura de Linux-SRT es similar a la de KURT, es decir, añadir al kernel estándar requerimientos de tiempo real mediante la inclusión de librerías conformes al estándar POSIX 1003.13.

Características.

- Planificación. El paquete correspondiente al planificador permite especificar términos de calidad de servicio, como por ejemplo el uso de un determinado porcentaje máximo a cada tarea de tiempo real. Esto es útil por ejemplo para grabar cds o escuchar mp3's sin interrupción.
- Gráficos. La asignación de CPU a una tarea no es el único factor que afecta a la velocidad de las aplicaciones, también lo es el servidor X, por eso se modifica para priorizar la renderización de gráficos en función de los parámetros de planificación de cada cliente X.
- Interfaz de usuario. Permite la configuración de parámetros de planificación a nivel de usuario

E.7. QLINUX

Qlinux es un *kernel* de Linux que proporciona calidad de servicio garantizada para requerimientos de tiempo real flexible. Por tanto no es un sistema operativo específico de tiempo real, sino que esta orientado hacia aplicaciones multimedia que requieren una determinada calidad de servicio. Licencia: GNU/GPL

Arquitectura. Usa una arquitectura precursora a la de "*kernel preemptable*", pero en este caso no solo aplicada al acceso de la CPU sino que también aplicada al acceso a la red y al disco.

Características. La última versión de *Qlinux* se basa en el *kernel* 2.4.4 e incluye las siguientes características:
H-SFQ (*Hierarchical Start Time Fair Queuing*) para el planificador de la CPU. El planificador activa una planificación jerárquica de forma que asigna ancho de banda de la CPU de forma justa entre las aplicaciones ó clases de aplicaciones.
H-SFQ para el planificador de paquetes de red. De manera similar a antes, pero en este caso, proporciona transferencias garantizadas y una asignación de ancho de banda para los paquetes de flujos individuales ó clases de flujos.
Algoritmo de planificación del disco (*Cell disk sheduler*). Soporta múltiples clases de aplicaciones tales como interactivas, *best-effort*, transferencias intensivas, tiempo real flexible, etc. [No estable]

Cuando se activa *Qlinux* estas características reemplazan a las disponibles en el *kernel* estándar de Linux. Las actuales versiones proporcionan flexibilidad permitiendo la combinación de estas características según se necesiten.

E.8. REDICE Linux

REDICE-Linux es un *kernel* de tiempo real que combina la planificación de tiempo real estricto y un *kerne*l prioritario con la baja latencia de un completo *kernel* de tiempo real estricto como es el de RTAI. Es decir, combina ambas tecnologías de diseño de sistemas operativos de tiempo real.

REDICE-Linux es un sistema operativo embebido ideal para aplicaciones de internet, dispositivos de redes, dispositivos médicos, industriales, adquisición de datos, control de maquinas y otros donde el tiempo es crítico, además de aplicaciones que requieran una determinada y garantizada calidad de servicio. Licencia: *Libre de Royalties. Open Source*. Soporte es de pago

Arquitectura. Está diseñado con una arquitectura dual, por una parte utiliza la arquitectura de *microkernel,* es decir tiene un segundo *kernel,* que es el de RTAI, y por la otra, modifica el *kernel* estándar de Linux para que se convierta en "*kernel preemptable*".

Ventajas.

- Ahorro de dinero: Usando los recursos y el sistema de control de presupuesto de los componentes de REDICE, los usuarios pueden construir distintas funcionalidades a bajo coste.
- Fiabilidad: Todas las tareas tienen su propio control de los recursos y son independientes las unas de las otras, como resultado de esto, el comportamiento inadecuado de algunas de las tareas no afecta a la estabilidad del sistema.
- Pronosticable: REDICE-Linux proporciona un comportamiento más predecible mediante el uso de una guía de tiempos integrada y un paradigma de planificación compartido.
- Sensibilidad: Las tareas no pueden ser bloqueadas por otros trabajos menos críticos por un periodo de tiempo indefinido.

Características.

REDICE-Linux proporciona un completo software de desarrollo, y de herramientas para el rápido desarrollo de sistemas de tiempo real embebidos.

Reloj con alta precisión del orden de microsegundos.
Sistema de planificación potente y predecible.
Mecanismo para garantizar rendimiento a las tareas.
Potencia y fiabilidad del código abierto de Linux.
Sistema de tiempo real con capacidad de garantizar a las aplicaciones rendimiento y ejecución de tareas con latencias del rango de los microsegundos

E.9. RTAI (Real Time Application Interface)

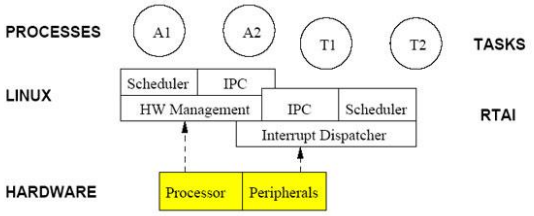
RTAI es una implementación de Linux para tiempo real basada en *RTLinux*. Añade un pequeño *kernel* de tiempo real bajo el *kernel* estándar de *linux* y trata al *kernel* de *linux* como una tarea con la menor prioridad. RTAI además proporciona una amplia selección de mecanismos de comunicación entre procesos y otros servicios de tiempo real. Adicionalmente, RTAI proporciona un módulo llamado LXRT para facilitar el desarrollo de aplicaciones de tiempo real en el espacio de usuario. Licencia: GNU/GPL

E.9.1. Arquitectura

RTAI tiene una arquitectura similar a *RTLinux.* Al igual que *RTLinux*, RTAI trata el *kernel* estándar de Linux como una tarea de tiempo real con la menor prioridad, lo que hace posible que se ejecute cuando no haya ninguna tarea con mayor prioridad ejecutándose. Las operaciones básicas de las tareas de tiempo real son implementadas como módulos del *kernel* al igual que *RTLinux.* RTAI maneja las interrupciones de periféricos y son atendidas por el *kernel* de *linux* después de las posibles acciones de tiempo real que hayan podido ser lanzadas por efecto de la interrupción. La figura 56 muestra la arquitectura básica de RTAI, que es muy similar a la de *RTLinux.* Las interrupciones se originan en el procesador y en los periféricos. Las originadas en el procesador (principalmente señales de error como división por cero), son manejadas por el *kernel* estándar, pero las interrupciones de los periféricos (como los relojes) son manejadas por RTAI *Interrupt Dispatcher*. RTAI envía las interrupciones a los manejadores del *kernel* estándar de *linux* cuando no hay tareas de tiempo real activas. Las instrucciones de activar/desactivar las interrupciones del *kernel* estándar son reemplazadas por macros

que se enlazan con las instrucciones de RTAI. Cuando las interrupciones están desactivadas en el *kernel* estándar, RTAI encola las interrupciones para ser repartidas después de que el *kernel* estándar haya activado las interrupciones de nuevo. Adicionalmente, se puede ver en la figura el mecanismo de comunicación entre procesos (IPC), que esta implementado de forma separado por Linux y por RTAI. También existe un planificador (*sheduler*) distinto para Linux y para RTAI.

Figura 56. Arquitectura RTAI



E.9.2. HAL - *Hardware Abstraction Layer*

Los desarrolladores de RTAI introducen el concepto de *Real Time Hardware Abstraction Layer* (RTHAL) que es usado para interceptar las interrupciones hardware y procesarlas después. RTHAL es una estructura instalada en el *kernel* de Linux que reúne los punteros a los datos internos del hardware relacionados en el *kernel* y las funciones necesarias por RTAI para operar. El objetivo de RTHAL es minimizar el número de cambios necesarios sobre el código del *kernel* y por tanto mejorar el mantenimiento de RTAI y del código del *kernel* de Linux. Con RTHAL, las diferentes operaciones (ej. manejar interrupciones) son más fáciles de cambiar ó modificar sin tener que interferir con la implementación de Linux. Por ejemplo, la estructura de RTHAL contiene la tabla de manejadores de interrupción, la cual es un lista de las funciones que son llamadas para manejar las diferentes interrupciones. El cambio consiste únicamente en modificar unas 20 líneas del código del *kernel* de Linux y añadir unas 50 nuevas líneas. Cuando RTHAL es instalado en Linux, las funciones y las estructuras de datos relacionada con la interacción con el hardware son reemplazadas por punteros a la estructura de RTHAL.

E.9.3. Planificación

La unidad de planificación de RTAI es la tarea. Siempre hay al menos una tarea, llamada *kernel* de linux, que ejecuta como la tarea de menor prioridad. Cuando las tareas de tiempo real son añadidas, el planificador da entonces mayor prioridad a éstas sobre la tarea del *kernel* de Linux. El planificador proporciona servicios tales como *suspend, resume*, *yield, make periodic, wait until*, que son usadas en varios sistemas operativos de tiempo real.
El planificador es implementado como un módulo del *kernel* dedicado (contrario a RTLinux) lo que facilita la implementación de planificadores alternativos si es necesario. Actualmente hay tres tipos de planificadores dependiendo del tipo de máquina:

Uniprocesador (UP). Unico procesador
Multiprocesador simétrico (SMP). Esta diseñado para maquinas SMP y proporciona un interfaz para las aplicaciones de forma que es posible seleccionar el procesador ó procesadores que deben ejecutar una tarea. Si el usuario no especifica un procesador para la tarea, SMP selecciona el procesador en función de la carga de trabajo.
Multi-Uniprocesador (MUP). Puede ser usado con ambos, pero al contrario que SMP, a las tareas se les debe especificar el procesador que deben usar. Viéndolo por el lado positivo, el planificador MUP permite un mecanismo de tiempo más flexibles para las tareas que los planificadores SMP ó UP.

Características. Con el objetivo de hacer el desarrollo de aplicaciones más fáciles y flexibles posibles, los desarrolladores de RTAI han introducido diferentes mecanismos para la comunicación entre procesos (IPC), entre las tareas de tiempo real y los procesos en el espacio de usuarios. Como añadido al mecanismo IPC, RTAI proporciona servicios de manejo de memoria y *threads* compatibles con *Posix.*

E.9.4. Comunicación entre procesos (IPC - Inter-process comunication)

RTAI proporciona una variedad de mecanismos para la comunicación entre procesos. Aunque los sistemas Unix proporcionan mecanismos similares a IPC para los procesos en el espacio de usuario, RTAI necesita proporcionar una implementación propia para que

las tareas de tiempo real puedan usar este mecanismo y no usen el estándar del *kernel* de Linux. Los diferentes mecanismos de IPC están incluidos como módulos de *kernel*, lo que facilita la carga cuando son necesarios. Como ventaja adicional el uso de módulos para los servicios, IPC facilita el mantenimiento y la expansión.

El antiguo mecanismo básico de comunicación de RTAI eran los FIFOs. FIFO es un asíncrono y no bloqueante canal de comunicación entre los procesos de Linux y las tareas de tiempo real. La implementación de RTAI de FIFO esta basado en la implementación de *RTLinux*, pero RTAI proporciona algunas características que no son posibles en *RTLinux.* Primeramente RTAI puede lanzar señales cuando hay eventos en el FIFO (escritura de nuevos datos). Los procesos en el espacio de usuario pueden entonces crear un manejador para la señal por los mecanismos estándar de Unix. Sin embargo, este mecanismo no es necesario para los procesos de usuario que quieran leer ó escribir del FIFO. Adicionalmente, puede haber múltiples lectores y escritores en el FIFO, cosa que no es posible en la versión de *RTLinux*. Finalmente, los identificadores FIFO pueden ser dinámicamente localizados con un nombre simbólico. Antes era necesario establecer un identificador global para el FIFO, lo que causaba problemas cuando múltiples e independientes procesos y tareas lo usaban.

Los semáforos es otra herramienta básica de sincronización entre procesos usada en los sistemas operativos. RTAI proporciona un API para usar semáforos, aunque cada semáforo esta técnicamente asociado a un FIFO, por tanto cada semáforo usa una entrada global del FIFO. Como añadido al servicio básico de semáforos, un semáforo puede estar asociado con un reloj, el cual puede ser usado para despertar un proceso encolado en un semáforo, incluso cuando el semáforo aun esta cerrado.

La compartición de memoria proporciona una alternativa a IPC y al paradigma FIFO cuando un modelo de comunicación diferente es requerido. La memoria compartida es un bloque común de memoria que puede ser leído ó escrito por un proceso y una tarea en el sistema. Como los diferentes procesos pueden operar de forma asíncrona en la región de memoria, es necesario un diseño para asegurar que los datos no sean sobreescritos de forma intencionada.

Finalmente, el método más flexible de IPC quizás sean los *mailboxes*. Cualquier número de procesos pueden enviar y recibir mensaje de y desde un *mailbox*. Un *mailbox* almacena mensajes hasta un límite que se defina, y contrario a los FIFOs, *mailbox* preserva los mensajes que están en el límite. Puede haber un número arbitrario de

mailbox activos en el sistema simultáneamente. RTAI también facilita la comunicación entre procesos mediante RPC.

E.9.5. Gestión de memoria.

En las primeras versiones de RTAI la memoria tenía que ser asignada estáticamente y no era posible las asignación en tiempo real. Sin embargo en las actuales versiones se incluye un módulo gestor de memoria que permite la asignación dinámica de memoria por parte de las tareas de tiempo real usando un interfaz basado en una librería estándar de C. RTAI preasigna trozos de memoria antes de la ejecución de tiempo real. Cuando la tarea de tiempo real llama a la función *rt_malloc(),* la respuesta que obtiene es el trozo preasignado. Antes de que el espacio se agote, RTAI reserva nuevos trozos de memoria (preasigna) para futuras llamadas. De manera similar ocurre con la función *rt_free(),* en este caso, se libera la memoria preasignada a la espera de futuras reservas. Cuando la suma de memoria liberada es mayor que un valor para una marca, se ordena su liberación.

Threads Posix. RTAI tiene módulos que proporcionan la implementación de *threads* de acuerdo al estándar POSIX 1003.1c. Usando las operaciones especificadas en el estándar, el usuario puede manejar de manera similar a como lo hace con los t*hreads posix* convencionales, excepto en cuanto a los conceptos de *joining* y *detaching*.
Los *threads* de un mismo proceso comparten el espacio de memoria, por tanto es fácil el intercambio de información entre ellos, sin embargo, el uso de áreas memoria compartida son necesarias para la sincronización. También se proporcionan los mecanismos de *mutex* y de variables de condición.

E.9.6. LXRT: User-space Interface to RTAI

LXRT es un API para RTAI que hace posible el desarrollo de aplicaciones de tiempo real en el espacio de usuario sin tener que crear módulos para el *kernel*. Esto es útil en primer lugar porque el espacio de memoria destinado al *kernel* no esta protegido de accesos inválidos, lo que puede provocar la corrupción de datos y el mal funcionamiento del *kernel*

de Linux. En segundo lugar, si el *kernel* es actualizado, los módulos necesitan ser recompilados lo que puede provocar que sean incompatibles con la nueva versión.

Mientras se desarrolla una aplicación de tiempo real en el espacio de usuario el programador puede usar las herramientas estándar de depuración hasta que ya no contienen errores, pero en este caso se trata de procesos de tiempo real flexibles (*soft*). Además, los servicios proporcionados por las llamadas al sistema de Linux están disponibles para las tareas. La ventaja de esto es que cuando la aplicación esté libre de errores puede convertirse a un módulo en el espacio del *kernel,* por lo que ya tendremos una tarea de tiempo real estricto. Sin embargo, al hacer esto, las llamadas al sistema utilizadas no sirven y deberán ser cambiadas por las proporcionadas por RTAI.

El cambio de la aplicación de procesos del espacio de usuario a tareas de tiempo real es fácil porque LXRT proporciona un API simétrico para la comunicación entre procesos y otros servicios de RTAI, lo que significa que el mismo API puede ser usado por las tareas de tiempo real y por los procesos del espacio de usuario. El mismo API de LXRT puede ser también usado cuando 2 procesos del espacio de usuario ó 2 tareas de tiempo real se comunican entre si, esto implica que varios relojes y mensajes del sistema proporcionados por LXRT puedan ser usados incluso cuando la aplicación no requiera tiempo real.

LXRT permite a las aplicaciones el intercambio dinámico entre tiempo real flexible y estricto mediante el uso de una simple llamada en el espacio de usuario. Cuando una aplicación esta en el modo de tiempo real flexible, usa el planificador de Linux, pero requiere el uso de la política de planificación FIFO. Sin embargo la planificación de procesos FIFO puede provocar la perdida de ranuras de tiempo por varias razones, por ejemplo, porque se esta ejecutando un manejador de interrupciones ó el planificador de tareas de RTAI.

Con el objetivo de facilitar el paso a tiempo real estricto, LXRT crea un agente en el espacio del kernel para cada proceso del espacio de usuario con requerimientos de tiempo real. Cuando el proceso entra en modo de tiempo real estricto, el agente desactiva las interrupciones, y mueve al proceso fuera del planificador de Linux y lo añade a la cola del planificador de RTAI. Ahora el proceso no es interrumpido por las interrupciones de otro proceso Linux. Para poder realizar este cambio el proceso debe asegurar que la memoria usada por el proceso este en la memoria RAM y debe desactivar la paginación usando la función *mlockall().* Esta llamada no debe ser usada en modo tiempo real

estricto. Aunque los desarrolladores de RTAI ven bien el desarrollo de aplicaciones de tiempo real estricto usando LXRT, el tiempo de respuesta no es tan bueno como la ejecución de las tareas como módulos del *kernel*.

E.10. RTLINUX

RTLinux es un sistema operativo de tiempo real que ejecuta Linux como un *thread* de menos prioridad que las tareas de tiempo real. Con este diseño, las tareas de tiempo real y los manejadores de interrupciones nunca se ven retrasados por operaciones que no son de tiempo real

La primera versión de *RTLinux* estaba diseñada para ejecutarse en la plataforma x86 y proporcionaba una pequeña API y un pequeño entorno de programación. La versión 2, que fue totalmente reescrita, fue diseñada para el soporte de multiprocesamiento simétrico (SMP) y para ser ejecutada en una amplia variedad de arquitecturas

RTLinux proporciona la capacidad de ejecutar tareas de tiempo real y manejadores de interrupciones en la misma máquina que el Linux estándar. Estas tareas y los manejadores ejecutan cuando se necesitan en detrimento de lo que estuviera ejecutando Linux. El peor caso de tiempo es entre que se detecta la interrupción hardware y el procesador ejecuta la primera instrucción del manejador de la interrupción. Este tiempo es del orden de los 10 microsegundos en la plataforma x86. Licencia: Actualmente hay 2 versiones de *RTLinux*:

***RTLinux/Open*.** Disponible bajo la licencia GPL, pero que ya no se trabaja en ella desde 2001.

***RTLinux/Pro*.** Distribución comercial.

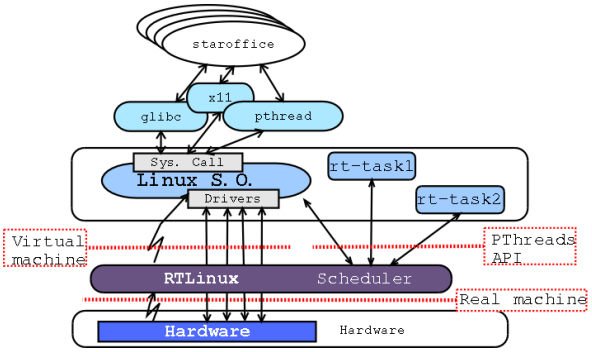
Ambas versiones son distribuidas y su arquitectura es patentada por la empresa *FSMLabs*. A partir de aquí la descripción que se hace es sobre *RTLinux/Open* o *RTLinux/GPL.*

E.10.1. Arquitectura

RTLinux es un pequeño y rápido sistema operativo que sigue el estándar POSIX 1003.13: sistema operativo de tiempo real mínimo. *RTLinux* añade una capa de abstracción hardware entre el *kernel* estándar de Linux y el hardware de la máquina. Esta es la arquitectura de *micro-kernel*.

Hay 3 modificaciones principales en el kernel de Linux con el objetivo de que RTLinux tenga el control del hardware de la máquina:
Control directo de las interrupciones hardware.
Control del reloj hardware e implementación de un reloj virtual para Linux.
El control de las interrupciones por parte de Linux es reemplazado por 2 funciones que permiten activar ó desactivar las interrupciones, pero las virtuales.
Estas modificaciones del kernel de Linux son complejas aunque no requieren excesivo código, pero si más que RTAI.

Figura 57. Arquitectura de RT-Linux



RTLinux proporciona un entorno de ejecución bajo el *kernel* de Linux, como consecuencia de esto, las tareas de tiempo real no pueden usar los servicios de Linux. Para disminuir este problema, el sistema de tiempo real se ha divido en dos partes: la capa de tiempo real estricto que ejecuta encima de *RTLinux*, y la capa de tiempo real flexible que ejecuta como un proceso normal de Linux. Para la comunicación de ambas capas se pueden usar varios mecanismos (FIFO, memoria compartida).

La propuesta de 2 capas es un método útil para proporcionar tiempo real estricto mientras se mantienen las características de escritorio de un sistema operativo. La separación del mecanismo del *kernel* de tiempo real del mecanismo del *kernel* de Linux de propósito general permite optimizar ambos sistemas de forma independiente.

Características.

Soporte de múltiples arquitecturas y válida para arquitecturas multiprocesador.

Gestión de procesos: Planificación, soporte de *threads* periódicos, amplio rango de prioridades, creación y borrado de *threads,* etc.

Gestión de memoria: No protección de memoria en el *kernel* y no asignación de memoria de forma dinámica.

Comunicación entre procesos: Semáforos, *Mutex,* control de inversión de prioridades, memoria compartida y FIFO's.

Tiempo y relojes: Resolución de nanosegundos. No relojes de usuario. Facilidades para añadir nuevos relojes hardware.

Programación de drivers: Se proporcionan funciones de acceso a dispositivos.

No proporciona herramientas de calidad de servicio

E.11. TIMESYS LINUX

TimeSys Linux es más que una distribución de un *kernel* de tiempo real, es un conjunto completo de herramientas para el desarrollo de aplicaciones de tiempo real de forma cómoda y rápida. Es una distribución para sistemas embebidos

Licencia: Comercial, menos algunos elementos que se distribuyen bajo la licencia GNU/GPL

Arquitectura. Se basa en la arquitectura de "*kernel preemptable*" para proporcionar rendimiento predecible y latencia limitada.

Características.

En el peor de los casos, tiempo de respuesta de 100 microsegundos.

Soporte en el mismo sistema, de tiempo real estricto y tiempo real flexible.

Soporte para aplicaciones Linux estándar y con interfaces POSIX, así como relojes con resolución de microsegundos (dependiendo del hardware).

Ventajas.

Flexibilidad, fiabilidad y fuerza del código abierto de Linux.
Mejor control sobre el rendimiento de las aplicaciones que otras distribuciones de Linux.
Mejora de la predicción para las aplicaciones embebidas actuales.
Soporte empotrado para el desarrollo de aplicaciones tanto en ordenadores Linux como Windows.
Escalabilidad para futuros requerimientos de rendimiento.

E.12. PORTOS

Portos es un sistema operativo de tiempo real (RTOS), orientado a componente, que puede cumplir requisitos críticos de tiempo real, y que permite añadir nuevos componentes software en tiempo de ejecución. Presenta dos novedades con respecto a los RTOS aplicados a los sistemas empotrados empleados en la industria en la actualidad, como son:

La planificación por plazos en lugar de la planificación por prioridades, lo que facilita enormemente a los ingenieros el diseño de aplicaciones por ser la manera mas natural para la planificación temporal de tareas ,y en segundo lugar el empleo de software de componentes, con las ventajas en cuanto a modularidad, escalabilidad, reusabilidad, sencillez y componibilidad que supone.

La Componibilidad, clave del software de componentes, se entiende como la posibilidad de añadir nuevos componentes en tiempo de ejecución. Para los sistemas de tiempo real, significa que la adición de nuevas tareas no perjudica el cumplimiento de plazos de las que ya se están ejecutando.

Portos está diseñado de manera que soporta los estándares de componentes de mayor futuro dentro de los sistemas empotrados, como son COM de Microsoft (actualmente

mejorado con COM+) y *Java* de *Sun.* COM es atractivo por su mínima sobrecarga y por su estándar OLE para el control de procesos estándar (OPC) mientras Java es atractivo por su portabilidad y seguridad.

E.13. QNX

QNX Software Systems Ltd. fue fundada en 1980 por *Gordon Bell* y *Dan Dodge* para desarrollar, mantener y poner en el mercado el sistema operativo de tiempo real QNX que corre bajo procesadores INTEL: 386, 486, *Pentium* y sus clones como AMD, *Nat Semiconductor*, *Cyrix* y SGS *Thompson.* Compañías como *3Com*, *Motorola, Cisco, Matsushita*, IBM, y otros escogen tecnología de QNX para ayudarse a construir aplicaciones fiables en sistemas electrónicos, telecomunicaciones, automotores, instrumentación médica, transporte, puntos de venta, y telefonía.

La idea de este sistema operativo comienza con sus fundadores a principio de los 80 comenzaron a trabajar en un pequeño SO basado en un *kernel* con pasaje de mensajes cuya interfaz de usuario se parecía a la UNIX. Los prototipos eran dos procesadores un 6809 y un 8088. Al principio de los 80 IBM lanzó su IBM PC basada en el 8088 y ahí comenzó el desarrollo. En las últimas etapas de esta historia la compañía cambió su nombre a *QNX Software Systems Limited*, ya que el nombre *Quantum* se confundía con otras compañías de nombres similares y además QSSL quería identificar su nombre más con su producto. Cuando la última versión QNX 4.24 apareció, la cual incluye a *Photon*, se lo comparó con una Ferrari con respecto al lento e inestable Win95. Pero nada dura para siempre en el rápido mercado de la computación. El sucesor de QNX: QNX/Neutrino ya está disponible, más bien dirigida al mercado de los embebidos.

E.14. MaRTE

Es un sistema operativo mínimo de tiempo real para aplicaciones empotradas, que puede ser usado en diferentes plataformas, incluyendo micro-controladores.

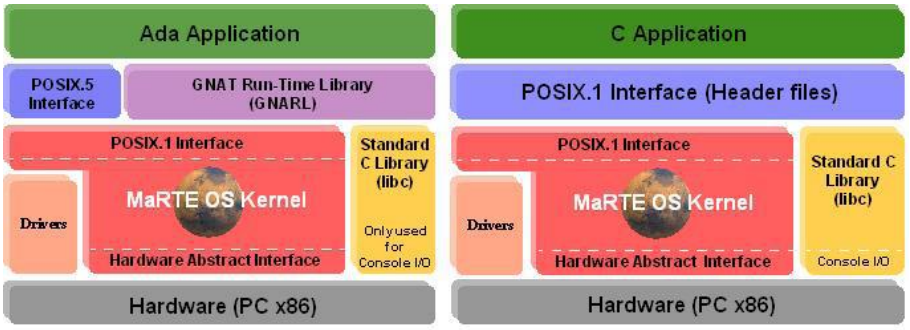
Cumple un subconjunto POSIX.13.

Ofrece los interfaces POSIX para los lenguajes C y ADA.

Permite el desarrollo cruzado de aplicaciones de tiempo real C y ADA.
Permite el desarrollo de aplicaciones C-ADA.

Como se aprecia en la figura 58, la parte central del sistema operativo MaRTE está en su *kernel,* el cual implementa las funcionalidades básicas del sistema. El *kernel* tiene una interfaz abstracta de bajo nivel para el acceso al hardware. Esta interfaz encapsula operaciones tales como: manejo de interrupciones, manejo del reloj y del tiempo y cambios de contexto en los *threads.* El objetivo de esto es facilitar la migración desde una plataforma a otra, ya que lo único que sería necesario cambiar es la capa de abstracción del hardware.

Figura 58. Arquitectura MaRTE



En la parte de arriba del *kernel* hay una interfaz para aplicaciones POSIX.1 (actualmente funciones Ada con la forma de funciones POSIX.1). Esto proporciona dos importantes ventajas:

- Aplicaciones C pueden usar el *kernel* directamente, solo son necesarias un conjunto de cabeceras C.
- La capa de bajo nivel GNARL está por encima de la interfaz POSIX. Esta capa es para adaptarlo al *kernel.*

La distribución GNAT que se ha adaptado para que funcione en el sistema operativo MaRTE es la versión estándar desarrollada para Linux.

Características.

Apropiado para aplicaciones estáticas con el números de tareas y recursos del sistema conocido en tiempo de compilación.

Los servicios tendrán tiempo de respuesta limitados.

No está protegido.

Multiplataforma.

Multilenguaje.

E.15. VxWorks

Es un componente *run-time* de la plataforma de desarrollo de sistemas empotrados Tornado II (Windows y UNIX). Hay dos versiones de *VxWorks*:

- *VxWorks*, que es de la que hablaremos aquí.
- *VxWorks AE*, que está diseñado especialmente para alta disponibilidad con la ayuda distribuida de la mensajería y de la alta tolerancia.

Características.

Herramientas de desarrollo cruzado host-target:

Compilador, Linkeador y Cargador para el sistema target.

Depurador remoto.

Determinación de tiempos de ejecución.

Simulador del sistema sobre el host.

Soporte de un gran número de placas comerciales mediante Board Support Packages (más de 200), así como de placas a medida.

Inicialización del HW.

Configuración de interrupciones y temporizadores

Memory mapping, etc.

Consta de una interfaz con varias APIs (más de 1800 funciones).

Es utilizado en diferentes tipos de aplicaciones: desde automoción (*Antilock Braking Systems*) a aplicaciones espaciales (*Mars PathFinder*), equipamiento de oficina (impresoras, faxes, etc.) y electrodomésticos (microondas, lavavajillas, etc.).
Está adaptado a diferentes arquitecturas de CPU's tales como: *PowerPC, Intel 80x86, Motorola 68K, Intel 80960, ARM, SPARC*, etc.

Arquitectura. La arquitectura está basada en microkernel, el cual lo han llamado *wind* de unos pocos KB que incluye:
Planificador multitarea basado en prioridades y con desplazamiento, el cual consta de 256 prioridades y tiene limitado el número de tareas.
Primitivas de sincronización y comunicación entre tareas.
Soporte a interrupciones.
Gestión de *watchdog timers.*
Gestión de memoria.

Al tener una arquitectura modular, el sistema final puede ser ampliado con diferentes módulos tales como:

Sistema de E/S, sistemas de ficheros soporte para redes, etc. Por lo tanto el sistema final es adaptable a cientos de configuraciones diferentes ya que incluso los diferentes módulos son escalables. Los módulos individuales pueden ser utilizados durante el desarrollo y ser omitidos en el sistema final.
Es compatible con la norma POSIX 1003.1 (llamadas básicas al sistema) y 1003.1b (*real-time extensions*).

E.16. CHIMERA

Es un sistema operativo en tiempo real de multiprocesadores de la generación siguiente de RTOS diseñado especialmente para soportar el desarrollo del software dinámicamente reconfigurable para los sistemas robóticos y de la automatización. La fuerza principal de *Chimera*, radica en la multiplicidad de las características ofrecidas las cuales son

necesarias para el rápido desarrollo de código reconfigurable y reutilizable, pero estos no están disponibles en otros sistemas operativos en tiempo real.

Arquitectura. Es un sistema operativo basado en VMEbus5 el cual soporta procesadores de propósitos múltiples, generales y especiales. Tiene un núcleo en tiempo real completamente equipado el cual soporta programación estática y dinámica, detección y control de errores frecuentes, un sistema completo de utilidades de biblioteca, varios y diferentes multiprocesadores de comunicación y sincronización de primitivas, y un ambiente completamente integrado del sitio de trabajo del host. El núcleo en tiempo real es multitareas por lo que permiten correr múltiples tareas al mismo tiempo.

Por defecto, el núcleo soporta los programas de prioridad fijos y los programas de prioridad dinámicos. Utiliza una separación de mecanismo/policía que permita que el usuario modifique fácilmente cualquier parte para requisitos particulares del núcleo o agregue un planificador nuevo. Las tareas periódicas y aperiódicas todas se programan de la misma manera, y los tiempos de plazo y de ejecución se pueden especificar opcionalmente para el uso, con algoritmos de programación estática y dinámica. El lenguaje C es una interfaz local, bus VME, y se proporcionan las interrupciones del buzón, que permiten el rápido desarrollo de aplicaciones de manejo de interrupciones. El núcleo se configura automáticamente para utilizar los dispositivos incorporados del RTPU (unidad de proceso en tiempo real) para proporcionar estos servicios, permitiendo que el mismo ejecutable binario funcione en varios y diferentes modelos de RTPU, siempre que utilicen la misma familia de procesadores.

Se proporciona un sistema extenso de bibliotecas para uso general, incluyendo las bibliotecas típicas de UNIX (e.g. *strings, math, random, time*, etc.), un estándar concurrente E/S (e.g. *printf, fgets*, etc. con la sincronización de multitrabajos), un paquete de matriz matemático (e.g. *addition, inverse, gaussian elimination, determinant*, etc.), una biblioteca de comandos intérpretes para desarrollar rápidamente interfaces de uso en líneas de comandos, servicios de configuración de archivos de lectura, y una variedad de otros servicios útiles.

Chimera tiene capacidades elaboradas para la detección y gestión de errores. Sus características más prominentes son los mecanismos de gestión de errores globales y de gestión de incumplimientos de plazos. Por defecto se proporciona un manejador de error que envía un mensaje de error y aborta la tarea. El manejador de error predeterminado puede ser reemplazado por cualquier número de manejadores definidos por el usuario. Los manejadores pueden ser asignados a errores específicos, o a cualquier sistema de errores basados en el módulo o espacio donde ocurrió el error. Las excepciones del procesador también generan señales del error, permitiendo que las excepciones del procesador y los errores del software se manejen con un solo mecanismo. El mecanismo puede también detectar varios problemas de la corrupción de la memoria, así ayudan en el rastreo de virus en el sistema en una primera fase.

Chimera es un verdadero sistema operativo en tiempo real de multiprocesos, diferente a la mayoría de los RTOS comerciales, que son los sistemas operativos de un solo procesador que hacen replicas en CPUs múltiple y se comunican con cada réplica usando cierta forma de protocolo de red.
Los núcleos en tiempo real se comunican con los otros mediante *low-overhead* (gastos indirectos bajos), mensaje no bloqueando que pasa por el mecanismo que llamamos correo expreso.

Esta comunicación subyacente del sistema proporciona la base para muchos diversos mecanismos nivel usuario de la comunicación y de la sincronización del multiprocesador en tiempo real, incluyendo memoria global compartida distribuida dinámicamente, semáforos remotos, mensajes pasados en orden de prioridad, tablas de variables de estado global, control de tarea del subsistema del multiprocesador, llamadas remotas de procedimientos, integración del sitio de trabajo del host, eliminación de errores simbólicos remotos, comunicación externa del subsistema del triple-buffer, y el sistema de ficheros extendido.

Chimera proporciona muchas herramientas para el rápido desarrollo de sistemas de control basados en sensores dinámicamente reconfigurables, tales como el mecanismo del control de tareas del subsistema del multiprocesador, la tabla de variable de estado global, controladores de dispositivos reconfigurables, sensor/activo genérico e interfaces

de propósitos especiales del procesador y un mecanismo de configuración de archivos de lectura.

El sistema operativo integra automáticamente los módulos reconfigurables que crean y inicializan tareas en el RTPUs apropiado, establecimiento de las trayectorias de comunicación entre módulos, manejando su tiempo y sincronización, capturando y direccionando las señales las cuales controlan el flujo de una aplicación, y proporcionando información en línea de un subsistema tal como el estado de cada tarea, medido contra frecuencia deseada, plazos perdidos, y de la ejecución y de la utilización de la CPU de cada tarea, a través del uso de una nueva herramienta *built-into* el núcleo el cual llamamos la descripción automática de la tarea.

Chimera proporciona un interfaz de red a *Onika*, de modo que los programadores puedan trabajar con código en tiempo real gráficamente combinando los iconos que representan los módulos en software reutilizable.
El controlador de eventos de aplicaciones de alto nivel y el software de reacción de bajo nivel pueden generar, modificar, eliminar errores, y ejecutar en esta manera.

Chimera ha permitido que todos estos sistemas compartan software y el hardware, reduciendo así los costes de desarrollo y mejorando la flexibilidad de aplicaciones, también ha demostrado ser un sitio de prueba excelente para los grupos de sistemas en tiempo real, porque cada parte del núcleo puede ser reemplazado fácilmente por el código del usuario.

Características. Proporciona la mayoría de las características disponibles en sistemas operativos en tiempo real comerciales, más la ayuda avanzada para el despliegue rápido de los sistemas de control basados en sensores y módulos reconfigurables y reutilizables del software.
Ayuda avanzada para la creación de aplicaciones basados en los módulos en tiempo real reconfigurables y reutilizables del software.
Apoya los propósitos generales múltiples CPUs en una placa madre de *VMEbus*.
Programación estática y dinámica.
El planificador por defecto se puede reemplazar por códigos de encargo.

Gestión y detección de errores globales.

Bibliotecas estándares completamente equipadas (secuencias, matemáticas, tiempo).

Bibliotecas adicionales útiles para crear el software reconfigurable (marco del comando-interprete, utilidad de la configuración de archivo de lectura, funciones de la matriz).

Semáforos locales de alto rendimiento.

Ambiente completamente integrado del sitio de trabajo del host.

Herramientas de desarrollo estándares de GNU.

Ayuda para los procesadores especiales del propósito (es decir DSP’s6, FPP’s).

Interfaz flexible para el acceso del dispositivo de I/O.

ANEXO F INSTALACIÓN DE RTAI/RTAILAB/SCILAB-SCICOS

La instalación completa requiere varios paquetes de software que deben ser descargados como código fuente. RTAI, RTAI-LAB, y COMEDI. Debido a que estas herramientas están experimentando fuerte desarrollo se recomienda no usar paquetes precompilados proporcionados por distribuciones de Linux.

The Mesa 3D graphical library. http://www.mesa3d.org.
The EFLTK graphic widgets library. equinox-project.org.
A new Linux kernel that will be patched with RTAI code.
The Comedilib data acquisition device interface library.
The RTAI real-time modules.
The Comedi data acquisition device modules.
The Scilab/Scicos Computer Aided Control System Design (CACSD) software.

Las siguientes secciones proporcionan instrucciones detalladas para la instalación de los paquetes (es importante seguir el orden propuesto).

F.1. LIBRERIA MESA

Hacer una nueva instalación de Mesa incluso si su distribución ya incluye. Mesa es necesaria para compilar EFLTK. Se Necesitan paquetes de desarrollo tales como xlibs-dev,..., x11proto-xext-dev,... 1.

1. Inicie como root
2. cd /usr/local/src
3. Visite http://www.mesa3d.org or http://sourceforge.net/projects/mesa3d and download MesaLib-6.x.x.tar.bz2
4. tar jxvf MesaLib-7.x.x.tar.bz2
5. cd Mesa-7.x.x
6. make realclean
7. make linux-x86 o make linux-x86-static

Note que linux-dri configuraciones están enumeradas en el subdirectorio configs.
Ver tambien http://dri.freedesktop.org/wiki/

8. make install

Requiere aceptar la instalación predeterminada en los directorios: /usr/local/include y /usr/local/lib. Algunas distribuciones de Linux puede requerir instalación en /usr/X11R6/include y /usr/X11R6/lib.

F.2. LIBRERÍA EFLTK

EFLTK es necesaria para xrtailab. EFLTK es parte del proyecto EDE ede.sourceforge.net. Se pueden requerir algunos paquetes adicionales como gettext, flex y svn.

1. cd /usr/local/src
svn co https://ede.svn.sourceforge.net/svnroot/ede/trunk/efltk
svn co https://ede.svn.sourceforge.net/svnroot/ede/trunk/ede o descargarlo desde:
http://equinox-project.org/

2. Si se descarga un paquete efltk-2.x.x.tar.gz (e.g. efltk-2.0.7.tar.gz) inicialmente se debe descomprimir
tar xzvf efltk-2.x.x.tar.gz
cd efltk o cd efltk-2.x.x
autoconf
./configure --disable-mysql --disable-unixODBC
./emake
./emake install

3. Edite /etc/ld.so.conf y adicione la linea con el path /usr/local/lib.
gedit /etc/ld.so.conf
4. Ejecute en el terminal:
/sbin/ldconfig para actualizar la base de datos de las librerias.

F.3. KERNEL DE LINUX KERNEL Y PARCHE DE RTAI

1. cd /usr/src
2. Descargue RTAI (última versión 3.6) de las opciones:

– Estable (numerada): http://www.rtai.org (e.g. rtai-3.6.tar.bz2)
tar xjvf rtai-3.6.tar.bz2
ln -s /usr/src/rtai-3.6 rtai
– Experimental (tener cuidado): cvs -d:pserver:anonymous@cvs.gna.org:/cvs/rtai co magma
(ver también https://gna.org/cvs/?group=rtai)

2. Compruebe las versiones de parches para el kernel disponibles en el directorio RTAI patches:
ls /usr/src/rtai/base/arch/i386/patches/

y busque una versión de kernel como por ejemplo: hal-linux-<kernel-version>.patch

Lea cuidadosamente /usr/src/rtai/base/arch/i386/patches/README.
Esto puede requerir editar algunos archivos después de configurado el núcleo.

3. Decida que versión de kernel utilizará por ejemplo: 2.4.xx or a 2.6.xx

Nota: Si usa una tarjeta PCMCIA, se recomiendan kernels hasta 2.6.12.6 o 2.6.17. Para kernels superiores de 2.6.13.xx se han experimentado problemas cuando se inserta el modulo para la tarjeta pcmcia (incluyendo kernel 2.6.16.15). Hay también dificultades con algunas versiones de 2.6.17.xx (2.6.17.4) y kernels ≤ 2.6.19.

Cuando compile Comedi, debe agregar la opción -- disable-pcmcia.

Kernel 2.4.xx. La version de RTAI v.3.5 soporta kernels 2.4.30 to 2.4.32 with ”hal-linux-2.4.xx-i386.patch”.
1. Descargue un ”vanilla” linux-2.4.x.tar.bz2 desde http://www.kernel.org
2. Desempaquete el kernel:
tar xjvf linux-2.4.30.tar.bz2
mv /usr/src/linux-2.4.30 /usr/src/linux-2.4.30-rtai
ln -s /usr/src/linux-2.4.30-rtai linux
3. Parche del kernel:
cd /usr/src/linux
patch -p1 < <rtaidir>/base/arch/i386/patches/<kernel-version>.patch
Por ejemplo:
patch -p1 < /usr/src/rtai/base/arch/i386/patches/hal-linux-2.4.30-i386.patch
4. make xconfig or make menuconfig
5. Configure el kernel. Ver detalles en Sección 13.

6. Posiblemente debe editar sugerencias en
/usr/src/rtai/base/arch/i386/patches/README
7. make
8. make modules_install
9. make install

Kernel 2.6.xx. The version de RTAI v.3.3 soporta kernel 2.6.10 y superiores with ”hal-linux-2.6.xx-i386-xxx.patch”
WARNING: Algunas distribuciones Linux incluyen la nueva plataforma ***udev***. Esta puede introducir problemas si los nuevos ***inodes*** no son registrados (ver Sección 15).

El kernel usado para la aplicación es la versión 2.6.23.

1. Descargue un ”vanilla” linux-2.6.xx.xx.tar.bz2 desde http://www.kernel.org
2. Desempaquete el kernel:
tar xjvf linux-2.6.23.tar.bz2
mv /usr/src/linux-2.6.23 /usr/src/linux-2.6.23-rtai
ln -s /usr/src/linux-2.6.23-rtai linux
3. Parche el kernel:
cd /usr/src/linux
patch -p1 < <rtaidir>/base/arch/i386/patches/<kernel-version>.patch
Por ejemplo:
patch -p1 < /usr/src/rtai/base/arch/i386/patches/hal-linux-2.6.23-i386-r12.patch
4. make xconfig o make menuconfig
5. Configure el kernel. Ver detalles en sección 13.
6. Posiblemente debe seguir las sugerencias en
/usr/src/rtai/base/arch/i386/patches/README
7. make
8. make modules_install
9. make install

Actualice su *boot loader* (lilo o grub):

- Lilo: edite /etc/lilo.conf, y ejecute lilo
- Grub: edite /boot/grub/menu.lst (Debian) o /etc/grub.conf (Fedora, ver sección 14).
- Adicione los módulos a /etc/modules (Debian), e.g. pcmcia si se usa una *laptop* con una tarjeta de adquisición de datos y ejecute update-modules (Debian).
- Fedora Core III actualiza el *boot loader* automáticamente.

Reinicie el computador y entre por el nuevo kernel compilado.

F.4. INSTALACION Comedilib

1. cd /usr/local/src
2. cvs -d :pserver:anonymous@cvs.comedi.org:/cvs/comedi login
cvs -d :pserver:anonymous@cvs.comedi.org:/cvs/comedi co comedi
cvs -d :pserver:anonymous@cvs.comedi.org:/cvs/comedi co comedilib
3. cd comedilib
4. Leer requerimientos de software de instalación en README.CVS y verifique los paquetes (automake etc.)
Por ejemplo use el comando: automake --version
5. sh autogen.sh
Nota: Puede ignorer *warnings*
6. ./configure --sysconfdir=/etc
Nota: Ponga atención a *warnings* y posiblemente repare los inconvenientes
7. make
8. make install
Nota: los lugares de instalación de comedi.h y comedilib.h son /usr/local/include.
9. make dev
Este paso crea los *inodes* /dev/comedi[0-3].
10. Herramienta de calibración opcional: comedi_calibrate
Instale paquetes desde su distribución favorita Linux libboost-program-options-dev y libgsl0-dev
Ver también el manual de RTAI sección 4.7-8.

F.5. RTAI (1er paso)

Para este paso tú puedes necesitar paquetes como: libxmu-dev and libxi-dev.
Inicie como root
1. cd /usr/src/rtai)
2. make xconfig o make menuconfig
3. Menu General: acepte los directorios por defecto.
_ Installation directory /usr/realtime (directorio de instalación de RTAI)
_ Kernel source directory /usr/src/linux (directorio de instalación del Kernel)
4. Menu General: seleccione opcionalmente *RTAI Documentation (HTML, PDF,...)*
Para no tener problemas con la instalación es ideal seleccionar únicamente la opción de la documentación HTML.
5. Menu Machine (x86): Ajuste *Number of CPUs* (por defecto = 2)
Cuando se utiliza doble núcleo se debe seleccionar la opción por defecto.

6. Seleccione Exit xconfig/menuconfig y guarde la configuración.
7. make
8. make install
9. **IMPORTANTE:** Adicione la direccion /usr/realtime/bin a la variable PATH en /etc/profile o en su directorio home en el archivo .bashrc.

También puede utilizar la opción en un terminal: #PATH="PATH: /usr/realtime/bin"

F.6. RTAI Paso 1 – Prueba del Sistema.

Cargue los módulos de RTAI, ver sección 12, iniciando con:

insmod modules rtai_hal hasta rtai_fifos.

Esto con el fin de medir la latencia del sistema para la atención de interrupciones en tiempo real

cd /usr/realtime/testsuite/kern/latency
./run

Esto corre una tarea periódica. El periodo por defecto para la ejecución de la tarea es de 100000 ns y esta definido por un DEFAULT_PERIOD en

/usr/src/rtai/testsuite/kern/latency/latency-module.c

La salida de la simulación son los valores de la latencia del sistema en nanosegundos (ns) y *overruns*. Es buena idea cargar la CPU del sistema con aplicaciones multimedia para analizar los cambios de la latencia.

Para parar la tarea en tiempo real, ejecute ctrl-c

El periodo se puede redefinir de la siguiente manera:

Inserte modulos de la seccion 2.13 desde rtai hal.ko hasta rtai fifos.ko, asi:

insmod /usr/realtime/modules/rtai lxrt.ko
insmod latency rt.ko period=1000000
./display

Para parar el programa ejecute:
ctrl-c …y ejecute el comando
rmmod latency rt

Puede comparar los resultados de la latencia del sistema con los publicados en:
http://issaris.org/%7Etakis/projects/rtai/livecd/list.php

F.7. Comedi

Si se presentan dificultades, por favor diríjase al archivo INSTALL en el directorio de comedi y también la información de www.comedi.org.

NOTA IMPORTANTE
Asegurarse de siguir las recomendaciones de
/usr/src/rtai/base/arch/i386/patches/README

Especialmente si se aplica el archivo *.config* ya que ésta configuracion es leida por el archivo script de configuración de comedi.

Ahora siga las siguientes instrucciones:

1. cd /usr/local/src/comedi
2. sh autogen.sh
3. ./configure con las siguentes opciones :

./configure --with-linuxdir=/usr/src/linux --with-rtaidir=/usr/realtime
o: ./configure --with-linuxdir=/usr/src/linux --with-rtaidir=/usr/realtime --enable-kbuild

Con una tarjeta de adquisición de datos PCMCIA use el siguiente comando

./configure --with-linuxdir=/usr/src/linux --with-rtaidir=/usr/realtime --enable-pcmcia

Nota: en la salida de la compilación se debe tener en cuenta que CONFIG IPIPE, CONFIG RTHAL, o CONFIG ADEOS esta seleccionado con “yes”. Esto refleja que es almacenado en /usr/src/linux/.config

4. make
5. make install (instala los módulos comedi para el kernel)
6. make dev

7. cp include/linux/comedi.h include/linux/comedilib.h /usr/include/
   cp include/linux/comedi.h include/linux/comedilib.h /usr/local/include/

8. ln -s /usr/include/comedi.h /usr/include/linux/comedi.h
   ln -s /usr/include/comedilib.h /usr/include/linux/comedilib.h

9. Cargue los módulos de RTAI y de Comedi (Sección 12)
10. Configure Comedi para trabajar con el hardware de adquisición de datos (ver sección 2 de documentación de Comedi en www.comedi.org/doc).

El manual de Comedi y la ayuda de comedi_config nuestra como asociar un particular *driver* y una herramienta *hardware* a uno de las herramientas de archivo /dev/comedi.

Por ejemplo, para configurar la tarjeta de adquisición de datos National Instruments 6024E PCMCIA card, siga los siguientes pasos y ejecute en un terminal:

comedi_config -v /dev/comedi0 ni_mio_cs y la salida es:

configuring driver=ni_mio_cs 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

WARNING: con la nueva plataforma udev existen algunos problemas si los nuevos nodos no son registrados, ver sección 14.

F.8. RTAI Paso 2

1. cd /usr/src/rtai
2. make xconfig o make menuconfig
3. Menu Add-ons:

Select COMEDI support over LXRT
Specify COMEDI installation directory (/usr/local or /usr). Este directorio debe contener: lib/libcomedi.a, include/comedi.h y include/comedilib.h

4. Menu RTAI Lab:

Select RTAI Lab
Adjust EFLTK installation directory (default is /usr/local)

5. Exit xconfig/menuconfig y guarde la configuración
6. make
7. make install

F.9. SCILAB/SCICOS

1. cd /usr/local
2. Descargue la ultima versión del código fuente de Scilab desde www.scilab.org
3. En algunas ocasiones la última versión requiere un *patch* que debe ser descargado. Compruebe esta recomendación en www.scicos.org

4. tar xzvf scilab-4.x.x-src.tar.gz (e.g. tar xzvf scilab-4.1.2-src.tar.gz)
5. Asegúrese que su sistema tenga los siguientes paquetes instalados: g77, sablotron, tcl8.4-dev, tk8.4-dev, xaw3dg, xaw3dg-dev, libpvm3, pvm-dev, y opcionalmente el conjunto de paquetes asociado con gtk-2.0.

En nuestro caso se instaló solamente tcl8.4 y tk8.4

6. Instalación de Tcl y Tk

Tcl8.4	**Tk8.4**
cd /usr/src/	cd /usr/src/
tar xzvf tcl-8.4.xx	tar xzvf tkl-8.4.xx
cd tcl	cd tk-8.4
./configure –enable	./configure –enable
threads	threads
make	make
make install	make install

7. cd scilab-4.x.x
8. Depending on your Linux distribution you may try either:
./configure
Debian:
./configure --without-java --with-tcl-library=/usr/lib --with-tcl-include=/usr/include/tcl8.4
Fedora Core:
./configure --without-java
9. make all
Nota: no ejecute make install

10. ln -s /usr/local/scilab-4.x.x/bin/scilab /usr/local/bin/Scilab

11. En un terminal como usuario normal ejecute: scilab. Ejecute quit en el prompt de Scilab para cerrar el programa. Si no inicia ver el final de la sección 11.

F.10. ARCHIVOS ADICIONALES PARA SCILAB

Si se utiliza la última versión: scilab-4.1.2-src.tar.gz, existe el problema de compatibilidad cuando se genera el código para aplicaciones de tiempo real con RTAI.
Para solucionar estos inconvenientes se deben instalar los archivos adicionales scilab-4.1.2-rtailab.tgz los cuales se puede descargar desde http://web.dti.supsi.ch/~bucher/.

Siguiendo las siguientes instrucciones se pueden compilar los archivos adicionales

Iniciar como root
Descargar el archivo scilab-4.1.2-rtailab.tgz. en cd /usr/src/
tar xzvf scilab-4.1.2-rtailab.tgz
cd /usr/src/ scilab-4.1.2-rtailab
cd /usr/src/ scilab-4.1.2-rtailab/macros
make install (como superusuario o root)
su usuario_rtai (se debe haber creado un usuario para este caso el usuario se llama usuario_rtai)
exit
make user (como usuario normal)

F.10.1 RTAI-Lab add-ons a Scilab

1. cd /usr/src/rtai/rtai-lab/scilab/macros
2. Edite la version de Scilab y el direcorio de instalcion en el archive: Makefile, e.g.:
SCILAB VERSION = 4.0
3. make install (este comando copia archivos a /usr/local/scilab-4.0/macros/RTAI/)

F.11. CONFIGURACIÓN DE USUARIO

Cada usuario de RTAI-Lab debe editar ${HOME}/.Scilab/scilab-4.x.x/.scilab with:

```
load('SCI/macros/RTAI/lib')
%scicos_menu($+1)=['RTAI','RTAI CodeGen','Set Target']
scicos_pal($+1,:)=['RTAI-Lib','SCI/macros/RTAI/RTAI-Lib.cosf']
```

Con estas lineas se adiciona:

- Menu RTAI al diagrama editor de Scicos con los items !CodeGen y !Set Target
- La paleta de bloques RTAI-Lib al menu Edit!Palettes

Para adicionar a Scilab como un path ejecutable se debe realizar cualquiera de los commandos:

cd /usr/local/bin
ln -s /usr/local/scilab-4.x.x/bin/scilab Scilab
o
Adicionar /usr/local/src/scilab-4.x.x/bin a la variable PATH en el perfil .bash y/o .bashrc

F.12. CARGAR LOS MÓDULOS

Para compilar y ejecutar programas de RTAI-Lab se deberá arrancar el sistema por el núcleo de Linux-RTAI. Si todo va bien su adquisición hardware será detectado (ejecute el comando ***cardctl info*** para kernels menores de 2.6.13 o ***pccardctl info*** en otro caso.

Cargue entonces los módulos de RTAI y modulos relacionados con el hardware. Dicha carga de los módulos depende de la configuración del núcleo de RTAI.

Se recomienda cargar inicialmente los módulos manualmente, y posiblemente más tarde configurar su sistema para cargar estos módulos en el momento de de arranque utilizando archivos de sistema o de comandos (dependiendo de su distribución de Linux y hardware) como por ejemplo desde: /etc/modules, update-modules, /etc/modutils/..., /etc/hotplug/ o /etc/udev y/o manualmente ejecutando un *script.*

Para cargar manualmente los módulos, se necesita iniciar como ***root*** o usar el comando ***sudo.*** Se puede comprobar el proceso usando el comando ***dsmesg*** o ***tail -f /var/log/messages***

Por ejemplo para una *laptop* con la tarjeta *DAQ 6024E PCMCIA* de la *National Instruments* (ver comentarios específicos indicados con # siguientes), se debe realizar las siguientes instrucciones para cargar los módulos.

```
modprobe rsrc_nonstatic                    # PCMCIA-related
modprobe i82092                            # PCMCIA-related
modprobe yenta_socket                      # PCMCIA-related
insmod /usr/realtime/modules/rtai_hal.ko
insmod /usr/realtime/modules/rtai_up.ko    # or rtai_lxrt.ko
insmod /usr/realtime/modules/rtai_fifos.ko
insmod /usr/realtime/modules/rtai_sem.ko
insmod /usr/realtime/modules/rtai_mbx.ko
insmod /usr/realtime/modules/rtai_msg.ko
insmod /usr/realtime/modules/rtai_netrpc.ko ThisNode="127.0.0.1"
```

```
insmod /usr/realtime/modules/rtai_shm.ko
insmod /usr/realtime/modules/rtai_signal.ko
insmod /usr/realtime/modules/rtai_tasklets.ko
modprobe comedi
modprobe kcomedilib
modprobe comedi_fc
modprobe 8255                                   # acq. card hardware-specific
modprobe ni_mio_cs                              # acq. card hardware-specific
insmod /usr/realtime/modules/rtai_comedi.ko
/etc/init.d/pcmciautils restart                 # from pcmciautils package
# or pcmcia restart (kernel < 2.6.13)
```

Se pueden cargar más módulos en ***/usr/realtime/modules/*** dependiendo de lo que se use en la aplicación. Es muy útil escribir un ***script*** de ***shell*** para cargar estos módulos luego de arrancar RTAI.

F.13. Configuración del núcleo de Linux

Esta sección muestra cómo configurar un nuevo núcleo RTAI que usted arrancará como una alternativa a su núcleo Linux. En otras palabras, no debería usar RTAI como el único núcleo en su sistema. Debe usar el menú de arranque (Lilo o GRUB) para seleccionar su actual núcleo de Linux, núcleo RTAI-Linux, u otro sistema operativo.

Las siguientes opciones son para un núcleo 2.6.x.

make xconfig or make menuconfig

General setup: seleccione ”Prompt for development...”
General setup: coloque en “Local version” a –rtai
Enable loadable module support: seleccione “Enable module support”, “Module unloading”, y “Automatic module loading”. Deseleccione “Module versioning support”; los modulos de RTAI no son versiones dependientes.

Processor type and features:
Seleccione su Subarquitectura *(PC-Compatible)* y *Processor family*.
Seleccione *”Preemption Model (Preemptible kernel (Low-Latency Desktop))”*.
Puedes necesitar “High Memory Support (4GB)” si se usa una tarjeta PCMCIA para adquisición de datos.
Deseleccione ”Use register arguments*(EXPERIMENTAL)”*.
Possiblemente deseleccione “Local APIC support on uniprocessors”.

Power Management options: Seleccione “ACPI Support” y las caracteristicas relevantes de si hardaware.
Deseleccione APM.
Deseleccione “*CPU Frequency scaling”*

Warning: ACPI support puede ser un problema en portátiles que utilizan botones para "screen closed" o para poner la computadora a “dormir” o en modo *“standby”*

Bus options: deje por defecto.
Compruebe el soporte de su hardware. “Laptops” necesitan soporte para PCCARD (PCMCIA/CardBus) y PC-card bridges. e.g. ***CardBus yenta-compatible bridge support.***

Device Drivers:
Generic driver options: dejar por defecto.
Memory Technology Devices (MTD): no necesita.
Parallel port support: deseleccione “*Parallel port support”*.
El puerto paralelo es un instrumento útil para depuración y experimentación en tiempo real. Debe dejar desactivada a fin de que controladores de Comedi puedan acceder directamente al puerto.
Plug and Play support: dejar por defecto.
Block devices: seleccione sus dispositivos. Fedora Core III necesita “*Ram Disk Support”* e “*Initial RAM Disk (initrd)”* para el boot.
ATA/ATAPI/MFM/RLL Support: seleccione el principal ítem *”ATA/ATAPI/MFM/RLL support”* y todos los ítems relevantes de su sistema.
SCSI device support: seleccione *“SCSI device support”* y deje por defecto las selecciones de acuerdo a los dispositivos SCSI del computador.
Multi-device support (RAID and LVM): solo necesita en casos especiales. Para Fedora Core III por defecto de la instalación se necesita esta opción activarla también con *“LVM (Device Mapper)”.*
Network device support: deje por defecto. Explore el menu de dispositivos hasta que encuentre su interfase de red *“network”.*
El comando ***lspci*** puede revelar el nombre del controlador Ethernet.
Amateur Radio, Irda, Bluetooth, ISDN subsystem, y Telephony support: Deje sedabilitado. Se puede activar despues.
Input device support: Garantice que el *mouse* es seleccionado.
Character devices: Garantice que *“/dev/agpart AGP Support”* y *“Direct Rendering Manager (XFree 4.1.0 and higher DRI support)”* son seleccionados.
I2C support: deje deseleccionado; hay reportes de dificultades cuando se usa con RTAI.

Multimedia devices: deje deseleccionado. Se puede activar después.
Graphics support: deje deseleccionado, Se puede activar después.
Con esta opción se puede usar características avanzadas de la tarjeta de video, pero algunas veces crea problemas de compatibilidad.
Sound: preferentemente actívelo como módulo y seleccione adecuados *drivers*
USB Support: preferentemente actívelo como módulo.

File Systems:
Second extended fs support: seleccionelo.
Ext3 journalling file system support: seleccionar con *"Ext3 extended attributes"*
Reiserfs support: la distribución Suse lo necesita, Talvez la distribución Linux usada no lo necesita. *Fedora Core III* hasta hora no lo necesita.
CD/ROM-DVD Filesystems: seleccione *"ISO 9660."* y sub-items.
DOS/FAT/NT Filesystems: seleccione si lo necesita. Es usado para compatibilidad de archivos en caso de tenerlos dentro de su sistema.

Deje las demás opciones por defecto

F.14. CONFIGURANDO EL ARCHIVO GRUB BOOT

Esta sección da un ejemplo de cómo configurar el archivo de configuración del GRUB boot usualmente ubicado en /etc/grub.conf (Fedora Core III) o en /etc/grub/grub.conf (distribución Debian). Ver también www.gnu.org /software/grub. Por defecto la instalación lo hace automáticamente.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,5)
#          kernel /boot/vmlinuz-version ro root=/dev/hda6
#          initrd /boot/initrd-version.img
#boot=/dev/hda
default=1
timeout=5
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
hiddenmenu
title Fedora Core (2.6.23-rtai)
          root (hd0,5)
          kernel /boot/vmlinuz-2.6.23-rtai ro root=LABEL=/ rhgb quiet enforcing=0
```

```
        initrd /boot/initrd-2.6.23-rtai.img
title Fedora Core (2.6.9-1.667)
        root (hd0,5)
        kernel /boot/vmlinuz-2.6.9-1.667 ro root=LABEL=/ rhgb quiet
        initrd /boot/initrd-2.6.9-1.667.img
title WINDOWS XP
        rootnoverify (hd0,0)
        chainloader +1
```

F.15. UDEV Y RTAI

udev es una dinámico esquema de detección de dispositivos (ver www.kernel.org /pub/linux/utils/kernel/hotplug/udev. html) que es utilizado por varios distribuciones de Linux y trabaja con kernel 2,6 de Linux. udev crea automáticamente dispositivo "*inodes"* en el directorio/dev sobre detección de dispositivos. Es desoues borra los *inodes* si los dispositivos no son usados.

RTAI usa inodes en /dev para FIFOs, para pasar mensajes entre el usuario y el espacio del núcleo *(kernel)*, y para interactuar con el hardware de adquisición de datos. Por tanto es necesario crear inores persistentes. Puede por ejemplo ejecutar el siguiente un *script* antes de cargar RTAI y Comedi módulos (se hace ejecutable con el comando chmod 777 rtai-inode).

```
# rtai-inode: RTAI inode creation for UDEV systems, creates /dev/rtf(n)

rm   -f  /dev/comedi* /dev/rtf* /dev/rtai_shm

for n in 'seq 0 9'; \
do \
rm -f /dev/rtf$n; \
mknod -m 666 /dev/rtf$n c 150 $$n; \
done ; \

# create shared memory inode
mknod -m 666 /dev/rtai_shm c 10 254

# create Comedi inodes
for i in `seq 0 15`;
do \
rm -f /dev/comedi$i; \
mknod -m 666 /dev/comedi$i c 98 $i ; \
done;
```

Básicamente, udev tiene un *"script init"* que reinstala todos los dispositivos en /dev. Un usuario puede crear los dispositivos en /lib/udev/devices y el *"script init"* se copia en /dev al momento del arranque. La scripts pueden ser modificados para crear los dispositivos bajo /lib/udev/devices.

F.16. Generación de tareas de tiempo real

Una vez construido el diagrama de bloques, para realizar su compilación y generar la tarea de tiempo real asociada al diagrama, se deben seguir los siguientes pasos.

Se selecciona en el menú *Scicos* → *Diagram* → *Region to Super Block*, esta función permite dibujar un marco "elástico" alrededor del diagrama de bloques, pero se debe excluir el evento de reloj, de esta manera se ha creado un super-bloque asociado al diseño en diagrama de bloques de la estrategia de control. Opcionalmente se da *click* en el menú *RTAI* → *RTAI Set Target, se debe* seleccionar el súper bloque generado y aparece la ventana mostrada en la ilustración 54a, en donde el usuario puede modificar los siguientes parámetros.

Target: Este es el *Makefile* que se pasa como primer parámetro en la compilación.
Ode function: estas son las funciones de las ecuaciones diferencial ordinarias disponibles en *scilab.* El valor por defecto es *ode4* y los posibles valores son:

- **ode1.** Usa el método de *Euler*.
- **ode2.** Usa el método de *Heun*, conocido como el método mejorado de Euler.
- **ode4.** Usa la formula de 4º orden de Runge-Kutta.

Step between sampling. Indica el número de puntos del sub muestreo computacional usados para varias funciones como las funciones ODE. El valor por defecto es 10.

Luego se selecciona en el menú *RTAI* → *RTAI CodeGen* y se da *click* en el superbloque generado en el paso anterior desde la barra de tareas. En este punto *scilab* convierte el diagrama de bloques en código C. Al compilar cada bloque dentro del superbloque se producen dos líneas en el *prompt* de *scilab*. Estas líneas son:

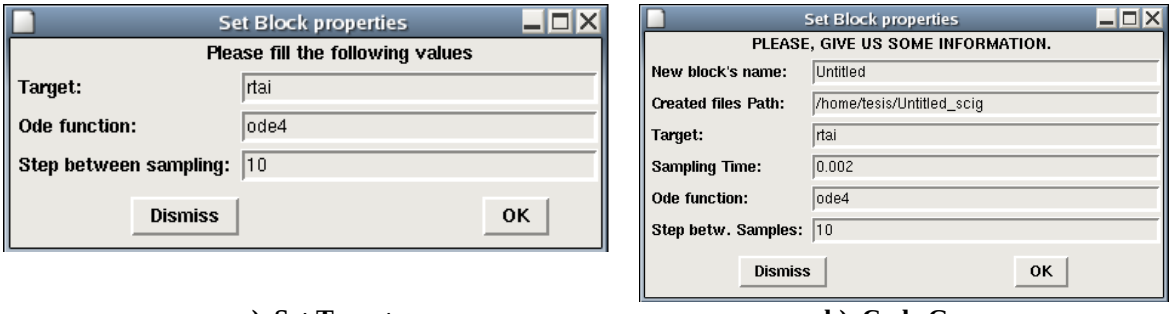
- *Shared archive loaded.*
- *Link done.*

Una ventana de dialogo se abre, ver ilustración 54b, donde el usuario puede modificar.

New block's name. Este será el nombre del ejecutable final. Se guardará en donde Scilab cree es su directorio actual, por defecto el nombre del superbloque es *Untitled*.
Created files Path. Es el directorio donde los archivos generados en código C son guardados junto con un *Makefile*, en este directorio es donde la compilación ocurre.
Target. Es el nombre base del *Makefile*. Este será copiado a los directorios de archivo generados y usados para la compilación. Por defecto RTAI evalúa el archivo que corresponde a la ruta *$SCILAB/macros/RTAI/RT templates/rtai.mak*
Sampling Time. Corresponde al valor del periodo seleccionado en el parámetro del bloque del evento de reloj (bloque rojo) y puede ser ajustado nuevamente por el usuario en esta parte.

Luego se da *click* en *OK* y la compilación inicia, los pasos de la compilación se pueden ver en el *prompt* de *Scilab.* En caso de que la compilación falle, se puede aun iniciar la compilación manualmente escribiendo "*make*" en el directorio donde los archivos C fueron generados.

Figura 59. Opciones para generación de código tarea de tiempo real



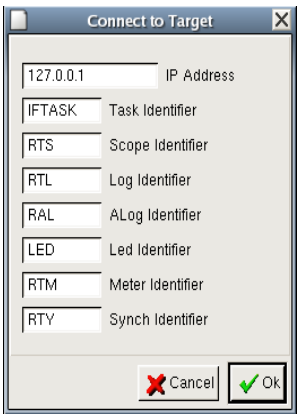
a) Set Target **b) Code Gen**

F.17 Ejecución de tareas de tiempo real

Para ejecutar una tarea de tiempo real es indispensable utilizar la herramienta "Terminal" provista por la distribución Linux, para ello se da inicio a dos terminales o se pueden ver las dos ventanas en un solo terminal. En el primer terminal se digita el nombre del ejecutable generado, para el caso general es "***./****Untitled –u"* o de acuerdo al nombre dado por el diseñador en la opción "*New block's name"* de la ilustración 54b; la opción *"–u"* provee instrucciones de uso, como opciones de comando de línea. Para iniciar el ejecutable de tiempo real con tipo de salida "*verbose*", entonces se digita "***./****Untitled –v"*, esta opción muestra las características de la tarea de tiempo real en el terminal. Para ejecutar el osciloscopio *xrtailab*, se debe tener en cuenta que únicamente se ejecuta cuando un *kernel* RTAI Linux esta corriendo. *Xrtailab* genera una falla si inicia sobre un *kernel* estándar de Linux. En la segunda terminal se digita "*xrtailab -h"* para llamar al osciloscopio de RTAI-Lab.

Ahora se debe conectar la tarea de tiempo real en espacio de *kernel* con el osciloscopio del espacio de usuario. Para ello, en la ventana principal de *xrtailab* se selecciona el menú *File ->Connect* o con *Alt+C*, aparece la ventana que se aprecia en la figura 60. Se selecciona *OK* y *xrtailab* desde el espacio de usuario se conecta con la tarea de tiempo real en espacio del *kernel.*

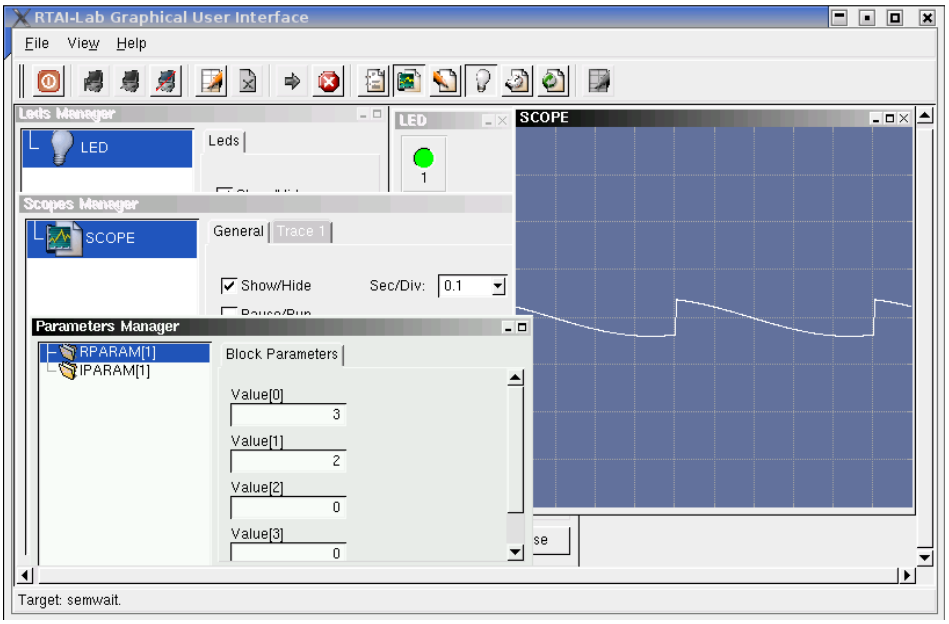
Figura 60. Ventana de Comunicación xrtailab con una tarea de tiempo real.



Dependiendo del tipo de bloques con los que se diseña el diagrama, en la barra de herramientas que provee *xrtailab* se pueden activar *scopes* en donde se pueden ver las

trazas o señales adquiridas y enviadas al proceso, también se pueden activar *leds* que indican circunstancias o eventos del proceso como por ejemplo alarmas, o activar *meters* que son indicadores de variables o señales; también se puede activar el administrador de parámetros (*Parameters Manager*) que permite modificar constantes y valores de los parámetros en tiempo real; dichas opciones se aprecian en la figura 61.

Figura 61. Ventana Xrtailab con sus opciones.

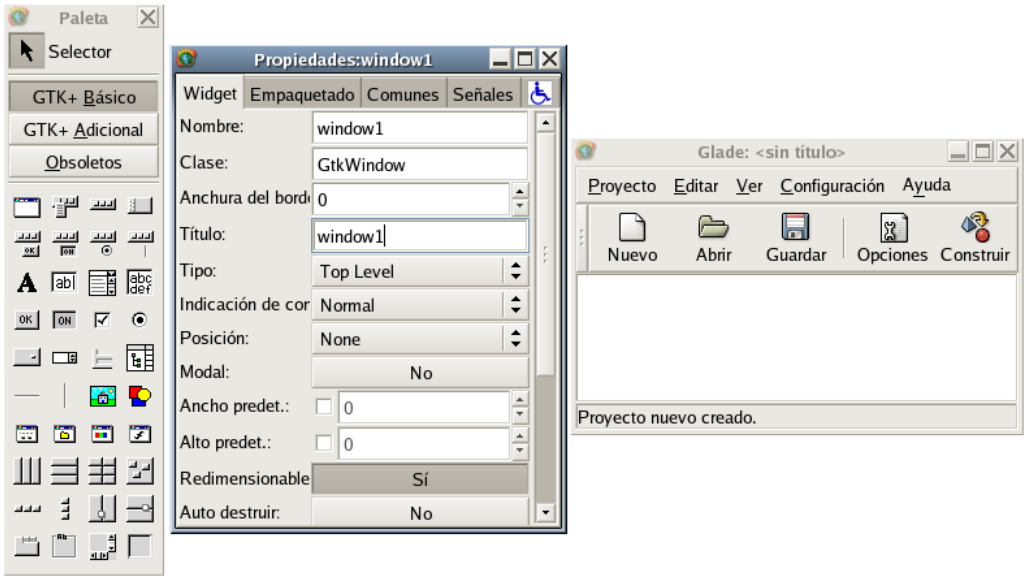


F18. Interfaz de usuario diseñada con Glade

El software utilizado para el desarrollo de dicha interfaz es Glade-2.6.0 incorporado en la distribución de Linux Fedora Core 3, dicho software permite desarrollar interfaces en forma de proyectos y permite a la vez generar el código en lenguaje C, C++ y Ada y modificarlo de acuerdo a las necesidades del proyecto. El arranque del software se realiza desde el menú *gnome* o si se prefiere desde un terminal tecleando *glade-2*; el sistema mostrará las tres ventanas que se aparecen en la figura 62. En dicha ilustración, la ventana *"Paleta"* permite añadir objetos gráficos o *widgets* a la aplicación, la ventana de *"Propiedades"* permite la modificación de las propiedades del *widget* elegido y la definición de sus señales o funcionalidades y la ventana *"Glade"* es la encargada del control del proyecto.

Para crear una interfaz se hace clic en el primer icono de la ventana *"Paleta"*, con esto aparece una ventana "vacia" lista para ser modificada, esta será la base de la interfaz, el programador selecciona y adiciona con *click* sostenido desde la ventana "*Paleta"* hacia la ventana creada, los botones, entradas de texto de acuerdo a los requerimientos de la aplicación y según el diseño deseado.

Figura 62. Ventanas software de diseño Glade-2.6.0



Cuando se ha realizado todo el diseño de la interfaz se debe guardar el proyecto por medio de la opción "*Guardar"* de la ventana *"Glade",* al seleccionar esta opción aparece una ventana con opciones acerca del lenguaje en el que se quieren crear los archivos, la ubicación y otras opciones menos relevantes, se deben seleccionar las opciones de acuerdo a las necesidades, el directorio por defecto en donde se almacena el código generado es en /root/Proyectos. Luego se debe generar el código del proyecto por medio de la opción *"Construir"* de la ventana *"Glade",* con esta opción se genera el código asociado a cada ventana, *widget* y señales asociadas a los mismos.

Posteriormente se debe compilar el código generado y crear el ejecutable del proyecto, para ello desde un terminal se va al directorio del proyecto y se teclea *./autogen.sh*, al teclear esta opción se crean los *makefiles* y se verifica el sistema, en seguida se digita *make*, con esta opción en la carpeta *src* se genera el ejecutable del proyecto. Por último para correr el proyecto se digita desde el directorio en mención *./nombre_del_proyecto*.

F.19 Comunicación interfaz de usuario con tareas de tiempo real

La comunicación de la interfaz diseñada con las tareas de tiempo real se realiza por medio de FIFOs *("First In- First Out")*, las cuales permiten escribir y leer desde y hacia la tarea de tiempo real. Para implementar el código que escribe la FIFO en el proyecto, se abre con el editor de texto en la carpeta *src* del proyecto, el archivo *callbacks.c* este archivo contiene las funciones que debe ejecutar la aplicación dependiendo de la retrollamada o *callback* sobre el widget que escribe en la FIFO, dentro de la función generada correspondiente el botón se digita el siguiente código.

```
void
on_button1_clicked              (GtkButton       *button,
                                gpointer        user_data)
{
        GtkWidget * entry = lookup_widget(GTK_WIDGET(button), "entry1");
        int fifo_write;
        double dato;
                if ((fifo_write = open("/dev/rtf0", O_WRONLY)) < 0) {
                        fprintf(stderr, "Error opening /dev/rtf0\n");
                        exit(1);}
        dato = atof(gtk_entry_get_text(GTK_ENTRY(entry)));
        write(fifo_write, &dato, sizeof(dato));
        printf("El dato escrito es:  %f\n",dato);
}
```

Con la implementación de esta función cuando se produce la retrollamada, primero se abre la FIFO con la función *open*, el primer argumento indica el nombre del FIFO y el segundo la forma en la que se va a acceder a ella. Los posibles estados son los siguientes:

O_RDONLY: se abre sólo para realizar operaciones de lectura.
O_WRONLY: se abre sólo para realizar operaciones de escritura.
O_RDWR: se abre para realizar operaciones de lectura y escritura.

Si la operación de apertura no se realiza, se ejecuta el mensaje de error "Error opening /dev/rtf0\n" y la aplicación termina, en cambio, si la FIFO se abre, se toma el dato escrito

en el *widget* de entrada de texto que es de tipo cadena, con la función de *gtk_entry_get_text*, con la función *atof* de la librería <stdlib.h> se convierte a double la cadena, luego este dato se escribe en la FIFO 0 mediante la función *write*; el primer argumento de dicha función representa el descriptor de archivo de la FIFO, el segundo argumento especifica el buffer de usuario donde se encuentran los datos que se van a escribir a la FIFO y el último argumento indica el número de bytes a escribir. Con esto ya se puede escribir a la FIFO para que desde RTAI-Lab sea leida.

La escritura y lectura desde Glade con la tarea de tiempo real se realiza por medio de los bloques FIFOint o FIFOout respectivamente, adicionados en los diagramas de bloques correspondientes a las tareas de tiempo real. Estos bloques se encuentran en la paleta RTAI-Lib de Scilab/Scicos.

Para la lectura de la FIFO en el espacio de kernel, desde la interfaz de usuario creada con Glade, se usará un manejador que permite desplegar este valor en widgets o usarlo de acuerdo al diseño de la interfaz. Antes de leer la FIFO se abre igual que para escribirla solo cambia el comando. Para acceder al manejador se usa la función *gdk_input_add* que es llamada desde la función principal o *main*, ubicado en el archivo *main.c* de la carpeta *src* del proyecto cread, por medio del siguiente código.

```
if ((fifo_read = open("/dev/rtf1", O_RDONLY)) < 0) {
        fprintf(stderr, "Error opening /dev/rtf1\n");
        exit(1);}
gdk_input_add(fifo_read,GDK_INPUT_READ,mi_manejador,NULL);
```

Y la función que ejecuta el manejador llamado “mi manejador”, es la siguiente.

```
static void mi_manejador()
{
   int valor, err,i;
   err=read(fifo_read,&val,sizeof(val));      */Lectura de la FIFO  abierta previamente.
        cadena= double_to_char (val.u[0]);  */Conversión de Double a caracter
        printf("%f\t%f\n", val.t, val.u[0]);
        gtk_label_set_text(GTK_LABEL(label1),cadena);  //se imprime el valor de la fifo en
label1.
}
```

Para hacer uso de la asignación de texto a la etiqueta *label*, se requiere la instrucción *gtk_label_set_text*, sin embargo esta asignación solo es posible debido a que anteriormente se ha creado un puntero global al *Widget label*, definido en el archivo *interface.h*, por lo tanto este puntero se convierte en global y mediante la línea *#include "interface.h",* la etiqueta label puede ser usada por *interface.c* y *main.c*.

ANEXO G CONFIGURACIÓN TARJETA PCI 6024·NI

Una vez instalado todos el software necesario para la instalación de todo el sistema de tiempo real se procede a hacer la detección de la tarjeta que se va a utilizar en primer lugar se deben cargar los siguientes módulos de RTAI y comedi:

```
insmod /usr/realtime/modules/rtai_hal.ko
insmod /usr/realtime/modules/rtai_up.ko              # or rtai_lxrt.ko
insmod /usr/realtime/modules/rtai_fifos.ko
insmod /usr/realtime/modules/rtai_sem.ko
insmod /usr/realtime/modules/rtai_mbx.ko
insmod /usr/realtime/modules/rtai_msg.ko
insmod /usr/realtime/modules/rtai_netrpc.ko          ThisNode="127.0.0.1"
insmod /usr/realtime/modules/rtai_shm.ko
insmod /usr/realtime/modules/rtai_signal.ko
insmod /usr/realtime/modules/rtai_tasklets.ko
modprobe comedi
modprobe kcomedilib
modprobe comedi_fc
modprobe 8255                                        # acq. card hardware-specific
modprobe ni_pcimio                          # acq. card hardware-specific
insmod /usr/realtime/modules/rtai_comedi.ko
```

Luego para Configurar Comedi para trabajar con el hardware de adquisición de datos (ver sección 2 de documentación de Comedi en www.comedi.org/doc). El manual de Comedi y la ayuda de comedi_config nuestra como asociar un particular driver y una herramienta hardware a uno de las herramientas de archivo /dev/comedi.

Para configurar la tarjeta de adquisición de datos National Instruments 6024E PCI card, se ejecuta en un terminal las siguiente instrucción:

```
comedi_config  -v  /dev/comedi0  ni_pcimio
```

y la salida debe ser:

```
configuring driver=ni_mio_cs
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

Este resultado indica que la tarjeta ha sido reconocida.

Si en el momento de ejecutar la instrucción anterior sale un mensaje de error que indica que archivo comedi0 no existe, diríjase al directorio donde esta instalado comedi y digite:

make dev

y vuelva a digitar:

comedi_config -v /dev/comedi0 ni_pcimio

Para correr los ejemplos del demo diríjase al directorio de comedilib y allí busque el directorio demo, una vez ubicado allí, para correr cualquiera de los demos digite:

./nombre del demo

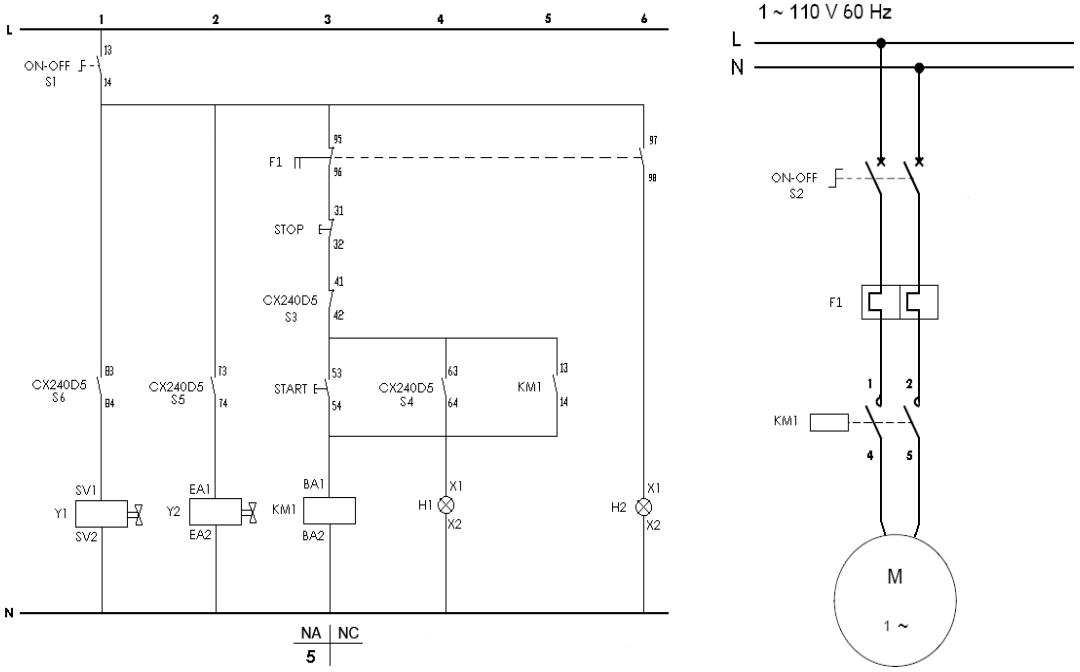
Por ejemplo para correr el ejemplo de la señal senoidal se debe digitar la instrucción:

./ao_waveform

ANEXO H DIAGRAMAS DE CONEXIONES DE LA PLANTA DE TANQUES INTERACTUANTES.

A continuación se presentan los diagramas electricos utilizados para la adecuacion de la planta de tanques interactuantes.

Figura 63. Diagrama de potencia de control de potencia.



S1 y S2:	Pulsadores ON-OFF.
S3, S4, S5, S6:	Contactos que dependen de los reles de estado solido CX240D5.
Y1:	Servo válvula
Y2:	Electro válvula
H1:	Piloto de encendido de la bomba (Verde).
H2:	Piloto de falla de la bomba (Rojo).
F1:	Rele térmico.
M:	Motor de la bomba.
START, STOP:	Pulsadores de marcha y paro desde el panel.

Figura 64. Diagrama de conexiones de SOTR/Tarjeta PCI 6024E NI/Planta de tanques interactuantes

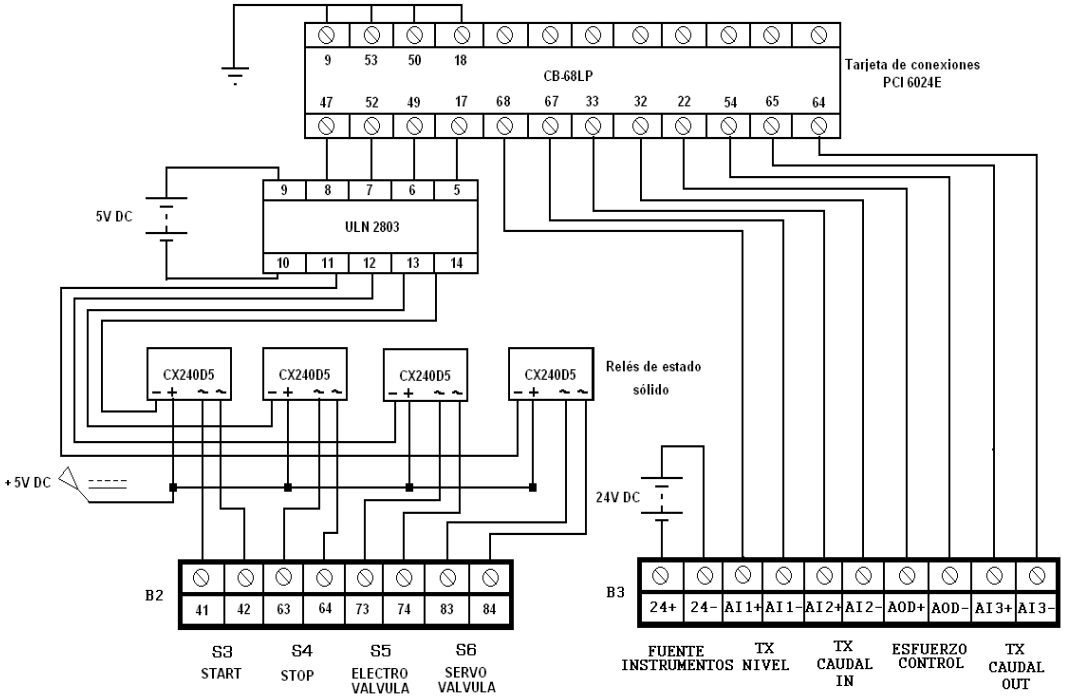
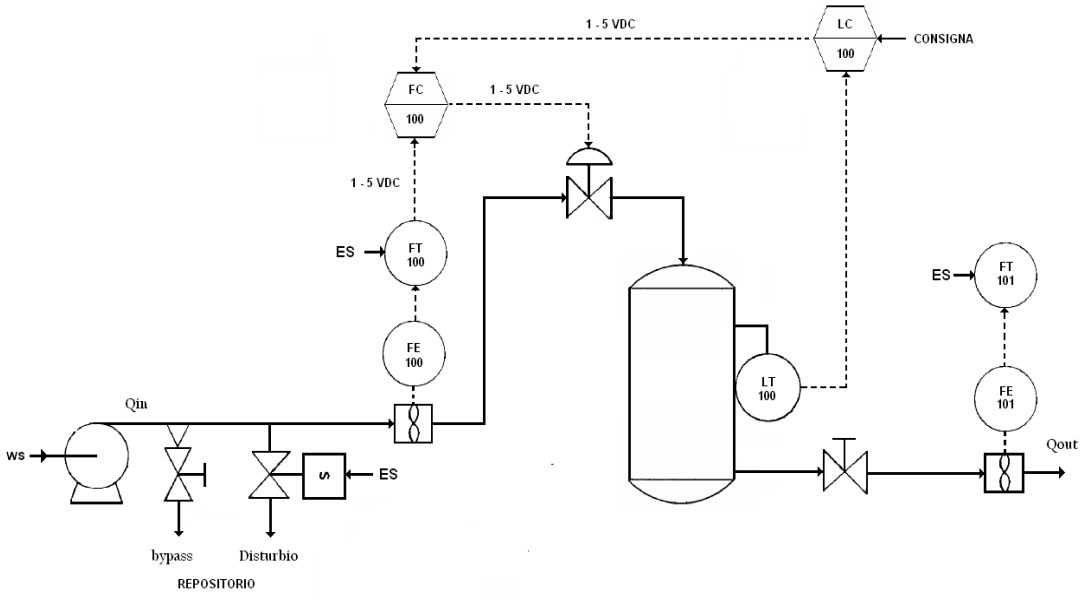


Figura 65. Diagrama de la instrumentación de la planta de tanques interactuantes



ANEXO I GUIA DE LABORATORIO DE LA PLANTA DE TANQUES INTERACTUANTES

GUIA DE LABORATORIO
PLANTA DE NIVEL Y FLUJO DE AGUA

La planta de nivel y flujo de agua está compuesta por un circuito hidráulico y la instrumentación necesaria para efectuar el control de flujo de agua por el circuito o el control de nivel de agua en uno de los tanques o en dos tanques interactuantes. Consta de un tanque plástico grande para almacenamiento, 3 tanques plásticos pequeños, una bomba centrífuga QB 128 que entrega un caudal máximo de 38 L/min, una válvula de control de flujo W.E. Anderson ABV111 con un actuador AMC-100A serie 4078, el cual puede ser usado para un rango de entrada de 1-5 V o 4-20 mA, un sensor de flujo Metalex 525 +GF+SIGNET con su respectivo transmisor de flujo +GF+SIGNET 8550-1 con salida de 4-20 mA, un transmisor de nivel implementado por medio del transmisor de presión diferencial YOKOGAWA EAJ110 con salida de 4-20 mA, tubería PVC de ¾", válvulas de bola de accionamiento manual y un *switch* que aplica alimentación de 120 VAC al sistema.

El funcionamiento de la planta es hacer circular el agua del tanque de almacenamiento hacia los tanques pequeños y volviendo al tanque de almacenamiento, bajo impulso de la bomba y la gravedad terrestre. Se puede hacer que el agua circule por uno o más tanques dependiendo de que válvulas manuales están abiertas o cerradas. De esta manera es posible configurar procesos de control de nivel de primero o segundo orden, según el número de tanques que intervengan en el proceso.

Se debe tener en cuenta que la válvula de control esta configurada para ser controlada por una tensión en el rango de 1-5 Vdc.

Ejecución de tareas de tiempo real en XRTAILAB

Para ejecutar una tarea de tiempo real es indispensable utilizar la herramienta "Terminal" provista por la distribución Linux, para ello se da inicio a un terminal desde el icono en la barra de tareas. En el terminal y ubicado en el directorio de origen, se digita el nombre del ejecutable generado, para el caso de la primera practica es "***./****histerisis_valvula_final –v"* (en la tabla 1 se presentan las tareas de tiempo real asociadas a cada practica), la opción *"–v"* muestra los características de la tarea de tiempo real y los cambios realizados por el usuario. Ahora desde el escritorio por medio de un acceso directo se da doble *click* en el icono XRTAILAB para llamar al osciloscopio de RTAI-Lab. Ahora se debe conectar la tarea de tiempo real en espacio de *kernel* con el osciloscopio del espacio de usuario. Para ello, en la ventana principal de *xrtailab* se selecciona el menú *File ->Connect,* o con *Alt+C*, o desde la interfaz usando el botón ***"Conectar",*** aparece la ventana que se aprecia en la ilustración 1. Se selecciona *OK* y *xrtailab* desde el espacio de usuario se conecta con la tarea de tiempo real en el espacio del *kernel.*

Ilustración 1. Ventana de Comunicación xrtailab con una tarea de tiempo real.

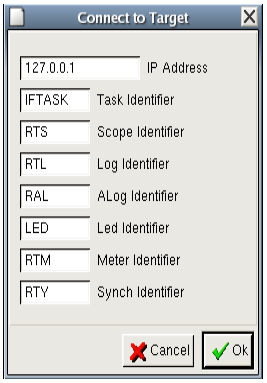


Tabla 1. Prácticas y su correspondiente tarea de tiempo real.

Practica		Tarea Tiempo Real
1	Histéresis válvula	histeresis_valvula_final
2	Identificación de la Planta	identificacion_planta_final
3	PID Paralelo Industrial AWBT	pid_paralelo_final
	PID Serie Industrial AWBT	pid_serie_final
4	Control en cascada	Identificación_lazo_interno_final
		Identificación_lazo_externo_final
		Monitor_pi_final
		control_cascada

PRÁCTICA 1. HISTERESIS DE LA VÁLVULA

Las válvulas cuyo comportamiento no es absolutamente lineal, presentan, frecuentemente, fenómenos de histéresis, que consisten en la diferencia entre los valores de caudal obtenidos para los mismos valores de posición de actuador (y por tanto, de apertura de la válvula) cuando se realizan barridos de estos valores en sentido creciente y decreciente. Sin embargo, algunas ocasiones esta histéresis es bastante sensible al intervalo de presiones o caudales de la corriente que la válvula manipula, como en general lo es la característica de la válvula. Así, la histéresis que presentan la mayoría de las válvulas empleadas en la industria es relativamente pequeña o, al menos, admisible, siempre que dicha válvula se utilice dentro del rango de presión para los que fue diseñado, por lo que su selección debe hacerse cuidadosamente para que la calidad del control sea la esperada.

Esta práctica consiste en identificar si la válvula de control presenta el fenómeno de histéresis, modificando el voltaje en la válvula de manera descendente y ascendente dentro de su rango de trabajo y analizar la respuesta del flujo entregado por la misma con la posición del actuador. Se debe tener en cuenta que la válvula de control de flujo se controla con un voltaje en el rango de 1-5 Vdc.

Los pasos a seguir en la práctica son los siguientes.

1. Ajuste la valvula de bypass para que permita un caudal de 6 gpm en el caudal de entrada.

2. Desde el escritorio arranque la interfaz de usuario, dando doble *click* en el icono “LABORATORIO CONTROL DE NIVEL”. Igualmente, por medio de un acceso directo en el escritorio, se da doble *click* en el icono “XRTAILAB” para llamar al osciloscopio de RTAI-Lab. En un terminal, (usado para correr todas las tareas durante las prácticas), se corre la tarea de tiempo real, digitando:

```
cd /home/usuario
su usuario
./histeresis_valvula_final –v
```

Ahora desde al interfaz *xrtailab* conecte la tarea de tiempo real.

3. Por medio del botón ***"Encender",*** energize los instrumentos, luego pulse el botón ***"ON***", para encender la bomba con la entrada numerica ***"Valor Manual"*** se selecciona el valor de la consigna manual al actuador, y con el botón ***"Iniciar"*** se envía el valor seleccionado, ver ilustración 2 . En la ventana se puede apreciar una tabla llamada ***"Datos"*** que permite guardar, modificar y/o eliminar los datos. Por medio del botón "***Guardar"*** se almacenan los datos actuales del valor manual y del caudal. Capture el dato en la tabla de la interfaz y en paralelo llene la tabla 2 con los resultados obtenidos empezando en el voltaje de control en su máximo valor 5 V y bajándolo a 1 V. No es necesario esperar a que el nivel se estabilice. Solo escriba el valor del flujo, capturado desde el botón ***"Valor Flujo",*** que se presenta la tabla ***"Datos".***

Ilustración 2. Interfaz de definicion de la Histeresis de la valvula.

Tabla 2. Datos voltaje actuador y flujo a la salida de la válvula.

Voltaje de Control, V	Flujo Bajando GPM	Flujo Subiendo GPM	Diferencia entre flujos
5			
4.6			
4.2			
3.8			
3.4			
3.0			
2.6			
2.2			
1.8			
1.4			
1.0			

4. Grafique las dos columnas de flujo de la tabla anterior contra el voltaje de control. Si las dos graficas no coinciden, se esta en presencia de histéresis en la válvula de control.

Explique a que se debe la presencia del fenómeno.

Apague la bomba con el botón ***"OFF"*** luego pulse el botón ***"Encender"***, para apagar la instrumentación. Ahora desconecte la tarea de tiempo real desde *xrtailab* por medio del botón ***"Desconectar"*** y pase a la practica 2.

PRACTICA 2. IDENTIFICACIÓN DE LA PLANTA Y SINTONIZACIÓN DEL CONTROL PID.

La puesta a punto de un sistema de control industrial requiere de la correcta sintonización del controlador, es decir de la selección adecuada de sus parámetros. Para poder sintonizar el controlador de un lazo de control, es necesario identificar primero la dinámica del proceso que se va a controlar, para luego obtener los parámetros del controlador, empleando el método de sintonización seleccionado. El proceso de sintonización del controlador consta así de dos etapas: identificación y sintonización.

La obtención de la información dinámica del proceso requiere que éste sea excitado de alguna forma y que tanto la entrada aplicada así como la respuesta del proceso, sean registradas. Por estas razones resulta necesario realizar una prueba experimental que permita identificar un modelo dinámico para el proceso. Los métodos a lazo abierto mas utilizados son los basados en la curva de reacción del proceso; el controlador puede o no estar instalado y si lo está operará de modo manual durante la prueba.

El objetivo de la práctica es realizar la identificación del sistema de nivel y sintonización de la estrategia de control PID, partiendo de que el sistema se presenta como una caja negra y obtener su respuesta en el dominio del tiempo ante una entrada escalón. Una entrada escalón se puede describirse como un cambio en la entrada desde cero a un valor finito en el tiempo t=0. La identificación debe arrojar la obtención de los parámetros dinámicos

que representan a una planta de primer orden más tiempo muerto (POMTM), que son la ganancia, el retardo y la constante de tiempo del sistema.

El método aquí utilizado es basado en el método de la tangente de Zieger-Nichols y en el método modificado de Miller. El procedimiento propuesto por Miller es una variación del método de Ziegler y Nichols (Método de la tangente). La variación propuesta por Miller radica en el cálculo de la constante de tiempo del modelo, ésta se calcula como el tiempo requerido para que la respuesta alcance el 63.2% del cambio total a partir del tiempo muerto.

Los pasos a seguir en la práctica son los siguientes.

1. Ajuste la valvula de bypass para que permita un caudal de 6 gpm en el caudal de entrada.

2. En el mismo terminal de ejecución de tareas de tiempo real se corre la tarea de tiempo real y desde el mismo directorio (cd /home/usuario), se corre la tarea para identificar la planta, digitando.

./identificacion_planta_final -v

Ahora desde la interfaz *xrtailab* conecte la tarea de tiempo real y diríjase a la sección de la guía *Obtención de parámetros dinámicos de la respuesta de un proceso POMTM*, con el fin de analizar y seleccionar las opciones para configurar de los *scopes* y las opciones para la obtención de los parámetros dinámicos del proceso (Ganancia, retardo, constante de tiempo). Debe realizar los dos métodos propuestos.

3. Por medio del botón ***"Encender",*** energize los instrumentos, luego pulse el botón ***"ON***", para encender la bomba. Ahora se procede a generar el escalón hacia la entrada de la planta. Por medio del de la entrada numerica ***"Escalón en GPM"***, se selecciona el valor de la consigna manual a la entrada de la planta, y con el botón ***"Enviar"*** se envía el valor seleccionado, ver ilustración 3. Por defecto el valor inicial es de 3 gpm espere que el sistema este en estado estable, y luego de esto cambie a un nuevo valor de caudal.

Observe el comportamiento del proceso y aplique los métodos manual y automático para obtener los parámetros dinámicos. Con el mismo cambio de escalón puede aplicar los dos métodos. Una vez obtenidos los parámetros, deben ser almacenados en la tabla 3.

Valor en estado estable _____.

Ilustración 3. Práctica Identificación de la planta.

Tabla 3. Parámetros dinámicos identificación planta.

Método	K	Retardo	Constante de tiempo
Manual			
Automático			

4. Compare los valores obtenidos y seleccione cualquiera de los resultados. Almacénelos manualmente en las casillas correspondientes a cada parámetro en la interfaz de usuario y por medio del botón ***"Guardar"***, guarde los parámetros.

Pulse ***"OFF"*** para apagar la bomba, apague la instrumentacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Adelante"*** para realizar la sintonización del control PID.

AJUSTE DE UN CONTROL FEEDBACK TIPO PID INDUSTRIAL AWBT

Una vez se ha identificado el proceso se procede a elegir y sintonizar la estrategia de control. La estrategia de control utilizada es un control *Feedback*, dentro esta estrategia

de control se encuentra el clásico control PID (Proporcional-Integral-Derivativo). Este control es ciertamente la estrategia de control más usada hoy en día; se calcula que más del 95% de lazos de control en los procesos emplean control PID, esto es principalmente debido a la facilidad de programación y alto desempeño de dichos controladores. A nivel académico, y en una buena parte de los artículos donde se comparan nuevas leyes de control versus el control PID, se hace un extensivo uso de la forma estándar idealizada de estructura paralela del PID, ver ecuación 1. Sin embargo esta forma no es realmente empleada en los controladores comerciales, los cuales implementan una variante del PID denominada industrial que adicionalmente incluye alguna técnica de *antiwindup* (AW) y *bumpless transfer* (BT). El controlador PID ha tomado diferentes formas según las investigaciones y desarrollos que el entorno industrial ha exigido con el fin de aumentar su robustez, garantizando favorables características de respuesta en frecuencia, dominio temporal y fácil calibración, incluyendo técnicas *antiwindup* y *bumpless transfer.*

$$\frac{\tilde{U}(s)}{\tilde{E}(s)} = K_c(1 + \frac{1}{sT_i} + sT_d) \qquad (1)$$

PID Industrial de Estructura Paralela

El PID Industrial Paralelo es la versión robusta y confiable de un PID, básicamente es un PID estándar con mejoras. Las principales modificaciones realizadas al PID estándar involucran.

Filtro lineal pasa-bajo. La primera mejora realizada al algoritmo PID paralelo industrial es el filtrado de la componente derivativa, esto en atención a:

a) Un diferenciador puro es un sistema no propio.
b) En un ambiente industrial el ruido es un componente inherente y siempre esta presente en la medición de la variable controlada. De no eliminarse, cuando se aplica la componente derivativa, se generan cambios bruscos en el esfuerzo de control.

Teniendo en cuenta lo anterior se ha propuesto limitar la ganancia de la parte derivativa, esto se hace aproximando la expresión "*sTd*" de la ecuación 1 del PID estándar por:

$$sT_d \cong \frac{sT_d}{1+\frac{sT_d}{N}} \qquad (2)$$

La expresión (2) aproxima bien el derivador a bajas frecuencias, pero limita su ganancia a altas frecuencias por un factor de *N*, este es un parámetro constante que típicamente esta en el rango de 3 a 33.

Realimentación de la velocidad. Normalmente los valores de consigna que se ingresan a un controlador PID industrial se presentan en forma de escalón, los cuales generan componentes de frecuencia infinita en el instante de compararlos con la variable controlada, lo que de nuevo genera cambios bruscos, por parte de la componente derivativa, en el esfuerzo de control. Por esta razón se propuso no aplicar la derivada a la señal error sino a la variable controlada. El esquema resultante de la segunda mejora aplicado a la ecuación (1) del PID estándar se ilustra en (3).

$$\tilde{U}_{(s)} = Kc\left(\tilde{E}_{(s)} + \frac{1}{sTi}\tilde{E}_{(s)} - \frac{sTd}{1+\frac{sTd}{N}}Y_{(s)}\right) \qquad (3)$$

Donde *Ẽ(s)* se ha ubicado dentro de la ecuación, pero dado que en la componente derivativa no se tiene en cuenta el valor de consigna *R(s),* tan solo queda menos la variable controlada *-Y(s)*, a esta modificación se le conoce como realimentación de la velocidad.

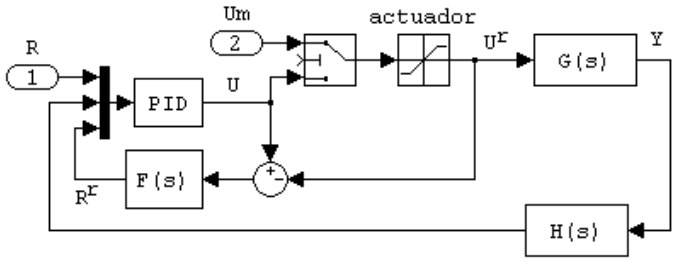
Filtro del valor de consigna. Para reducir la sensibilidad ante cambios en los valores del punto de consigna y evitar sobre impulsos, se adopta un filtro en el valor de consigna con el fin de mejorar la respuesta transitoria. Este es implementado en la acción proporcional, haciendo que la señal de consigna *R(s)* sea pesada por un factor b<1. El algoritmo PID así resultante esta dado por (4).

$$\tilde{U}_{(s)} = Kc\left(bR_{(s)} - Y_{(s)} + \frac{1}{sTi}\tilde{E}_{(s)} - \frac{sTd}{1+\frac{sTd}{N}}Y_{(s)}\right) \qquad (4)$$

Técnica AWBT. A pesar de que se obtuvo un PID paralelo industrial mejorado, el aún puede experimentar reajuste excesivo de la componente integral y en el caso de requerir que el controlador conmute entre manual y automático, situación común en los procesos industriales, esto provocará saltos en el actuador por lo que este PID industrial debe ser reforzado con alguna técnica AWBT.

La Técnica Condicionante es una de las tantas técnicas AW que tiene como ventaja que implementa AW y BT al mismo tiempo, donde la estrategia *antiwindup* es usualmente implementada como una técnica *bumpless transfer*, ver ilustración 4. En dicha ilustración se muestra el mecanismo básico de cualquier técnica AWBT que requiere medir el valor real U^r de entrada a la planta G(s) y compararlo con el esfuerzo de control U dado por el algoritmo PID, para ponderarlo por F(s) y realimentarlo de nuevo al PID.

Ilustración 4. Modulo anti windup y bumpless transfer



La Técnica Condicionante realimenta una señal llamada referencia realizable R^r, esta es tal que si se aplica al controlador en lugar de la referencia R, la salida de control U habría sido igual a la entrada real de planta U^r obtenida con la referencia R y bajo esta situación el actuador no esta saturado. Esto puede lograrse mediante una compensación adicional que realimenta $U-U^r$ al componente integral del PID mediante un compensador F(s), generando la señal referencia realizable dada por (5).

$$R^r(s) = \big(U(s) - U^r(s)\big)F(s) \qquad (5)$$

Si R^r es aplicada al componente integral del controlador PID industrial dado por (4), la entrada al integrador esta dada por (6).

$$E(s) - \big(U(s) - U^r(s)\big)F(s) \qquad (6)$$

Definiendo I(s) = 1/s, entonces la salida del integrador esta dada por (7):

$$\Big(E(s) - \big(U(s) - U^r(s)\big)F(s)\Big)I(s) \qquad (7)$$

Una vez descritas las características del controlador PID industrial, para poder realizar la sintonización de los controladores, primero debe identificarse la dinámica del proceso, realizada en la práctica anterior. Ahora a partir de dichos parámetros se determinan los parámetros del controlador utilizando métodos de sintonización.

PID INDUSTRIAL DE ESTRUCTURA SERIE

Este tipo de controlador a diferencia del estándar en donde sus tres acciones de control están en paralelo, presenta las componentes una detrás de la otra o en serie. En la ecuación (8) se aprecia la ecuación de este controlador.

$$\frac{\tilde{U}(s)}{\tilde{E}(s)} = Kc'\left(1 + \frac{1}{sTi'}\right)(1 + sTd') \qquad (8)$$

Donde *Kc'* es el termino proporcional, *Ti'* es la constante de tiempo integral y *Td'* es la constante de tiempo derivativa con restricción *Ti'≥4Td'.*

El PID serie presenta las mismas variantes realizadas al PID paralelo, siendo la primera, la estructura más usada en controladores industriales, las diferencias radican en el uso del filtro derivativo, tal como se aprecia en la ecuación (9).

$$\frac{\tilde{U}(s)}{\tilde{E}(s)} = Kc'\left(1 + \frac{1}{sTi'}\right)\left(\frac{1 + sTd'}{1 + s\,{}^{Td'}/_{N}}\right) \qquad (9)$$

FUNCIONAMIENTO DEL CONTROL PID

Dentro de los tipos de funcionamiento de la estrategia de control PID, que pueden presentarse en un lazo de control realimentado, en el cual hay dos entradas el valor deseado y la perturbación y una salida, la señal realimentada, se deben considerar, dos posibles condiciones de operación del sistema de control.

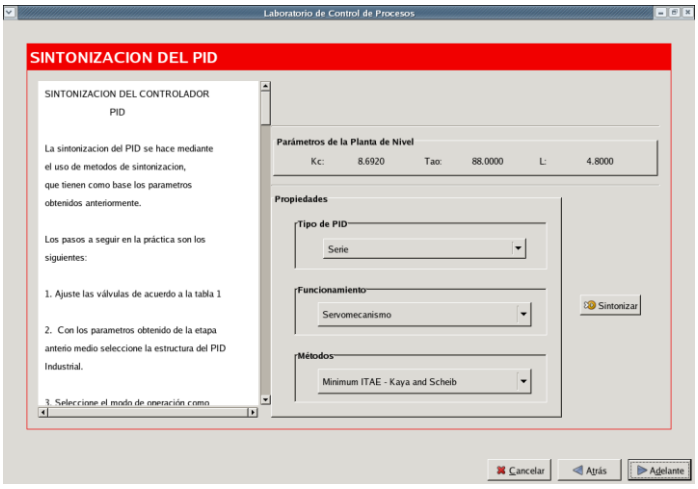
• **Servomecanismo**. Bajo esta condición el PID requiere un buen seguimiento ante cambios en el valor deseado. Aquí no se tiene en cuenta la presencia de perturbaciones.
• **Regulador.** Esta característica implica que la estrategia de control tenga una respuesta adecuada ante la presencia de las perturbaciones, estando el sistema en un valor estable, es decir sin cambios en la consigna.

Con base en la explicación anterior se procede a realizar la sintonización de un controlador PID industrial con base en los parámetros dinámicos de la planta obtenidos en la primera parte de esta práctica. Los pasos a seguir en la práctica son los siguientes.

5. Una vez se han guardado los parámetros obtenidos en el punto 4 de la primera parte de esta practica, automáticamente se cargan en la ventana de sintonización del control PID, ilustración 5. Ahora seleccione las siguientes opciones:

Tipo de PID ->Paralelo.
Funcionamiento -> Servomecanismo
Métodos -> *Seleccione un método de sintonización.*

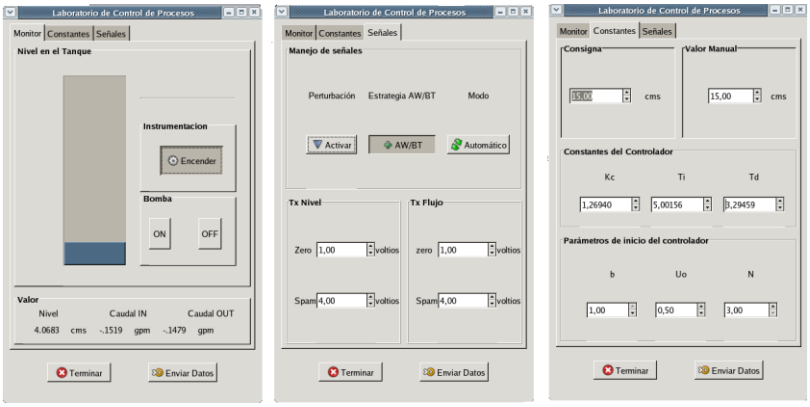
Ilustración 5. Opciones de sintonización estrategia de control PID



Click en el botón ***"Sintonizar"*** y saldrá la ventana de la ilustración 6 en donde aparecen los valores obtenidos automáticamente de las contantes de sintonización del controlador

seleccionado, los botones para activar y desactivar la tecnica AWBT, la perturbación y los , los modos de operación manual o automatico; los botones para encender la instrumentación y la bomba y el estado del nivel en el tanque.

Ilustración 6. Sintonización del control PID Industrial AWBT.



6. Ahora corra la tarea de tiempo real desde el terminal donde ha corrido las tareas anteriores. Asegúrese de correr la tarea adecuada, digitando.

./ pid_paralelo_final –v (si selecciona al PID Paralelo Industrial)
./ pid_serie_final –v (si selecciona al PID Serie Industrial)

7. Ahora desde la interfaz *xrtailab* conecte la tarea de tiempo real y seleccione la opción *Open/close scope manager* y configure las escalas *de cada scope* para observar las tendencias en el tiempo de cada variable.
Encienda la bomba con el botón ***"ON"***. Los parámetros b, N y Uo aparecen inicialmente por defecto, pero pueden ser configurados por el usuario. Por medio del botón ***"Consigna"***, se selecciona el valor de la consigna a la entrada de la planta, y se presiona el botón ***"Enviar Datos"*** para enviar todos los valores seleccionados, hacia la tarea de tiempo real. Ahora espere que el sistema este en estado estable. En estos momentos el controlador debe esta en modo automático.

8. Ahora envié un cambio positivo de 5 cms por medio del la entrada numerica ***"Consigna"***, y luego uno negativo de 5 cms. Observe el comportamiento del proceso y

del esfuerzo de control. Congele la grafica en cada cambio, mida y almacene los valores de la respuesta del sistema en la tabla 4.

9. **Fenómeno *Bump Transfer***. Ahora cambie el controlador de Modo automático a manual presionando el botón Modo-> ***"Automatico"*** y de *click* en ***Enviar Datos*** (el sistema pasa a modo manual). Introduzca al sistema cambios en la consigna manual con el botón ***"Valor Manual"*** (rango 0-30 cms) durante un tiempo t fijo. La estrategia AWBT debe estar desactivada, es decir el botón ***"AWBT"*** no debe estar presionado.
Ahora cambie de modo manual a automático y observe la reacción del proceso hasta que se llegue al estado estable. Congele la imagen de la salida del sistema, tome datos de tiempos de saturación del esfuerzo de control y la variable controlada y amplitudes de las mismas. Llene la tabla 5. Analice y concluya.

10. **Adición técnica *Bumpless Transfer***. Con el controlador en modo automático, espere que el sistema se estabilice. Ahora active la técnica AWBT presionando el botón ***"AWBT"*** y en seguida cambie el controlador a manual presionando el botón Modo-> ***"Automatico"***. Introduzca los mismos cambios hechos al sistema anteriormente por medio de la consigna manual ***"Valor Manual"*** durante el mismo tiempo t de la prueba anterior.

Ahora cambie nuevamente de manual a automático, observe la reacción del proceso y espere que el sistema se estabilice. Congele la imagen de la salida de la variable controlada y del esfuerzo de control, tome datos de tiempos de saturación del esfuerzo de control y la variable controlada y amplitudes de las mismas. Llene la tabla 5. Analice y concluya.

Compare los resultados obtenidos almacenados en la tabla 5, con la estrategia AWBT en modo activo y sin ella, analice los esfuerzos de control, la variable controlada y concluya.

Pulse ***"OFF"*** para apagar la bomba, apague la instrumentacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Terminar"***.

11. Ahora se debe sintonizar el PID seleccionado como regulador, en la ventana actual de sintonización del control PID, ilustración 5, seleccione las siguientes opciones:

Tipo de PID ->Paralelo.

Funcionamiento -> Regulador

Métodos -> *Seleccione un método de sintonización.*

Click en el botón ***"Sintonizar"*** y saldrá nuevamente la ventana de la ilustración 6, en donde aparecen los valores obtenidos de las contantes de sintonización del controlador con base en las opciones seleccionadas.

12. Realice nuevamente los pasos 6 y 7.

13. Ahora configure la válvula de perturbación en 180 grados de cerrada a abierta, luego active la perturbación presionando el botón ***"Activar"***. Presione el botón ***"Enviar Datos"*** Observe el comportamiento del proceso y del esfuerzo de control y espere que el sistema se estabilice, desactive la perturbacion. Congele la grafica, mida y almacene los valores de la respuesta del sistema en la tabla 4.

14. **Fenómeno *Windup*.** Ahora con el controlador en automático y el sistema en estado estable, abra completamente la válvula de perturbación de tal manera que se genere una gran perturbación al sistema, active la perturbación con el botón ***"Activar"***, espere que el esfuerzo de control se sature y espere otro tiempo t luego desactive la perturbación. La estrategia AWBT debe estar desactivada.

Observe la reacción del proceso y espere que el sistema se estabilice. Congele la imagen de la salida de la variable controlada y del esfuerzo de control, tome datos de tiempos de saturación del esfuerzo de control y la variable controlada y amplitudes de las mismas. Llene la tabla 5. Analice y concluya.

15. **Adición técnica *Antiwindup*.** Con el controlador en modo automático, espere que el sistema se estabilice. Ahora active la técnica AWBT presionando el botón ***"AWBT"*** y en seguida introduzca al sistema una gran perturbación por medio del botón ***"Activar"***,

espere que el esfuerzo de control se sature y mida el mismo tiempo t de la prueba anterior luego desactive la perturbación.

Observe la reacción del proceso y espere que el sistema se estabilice. Congele la imagen de la salida de la variable controlada y del esfuerzo de control, tome datos de tiempos de saturación del esfuerzo de control y la variable controlada y amplitudes de las mismas. Llene la tabla 5. Analice y concluya.

Compare los resultados obtenidos, los esfuerzos de control, analice y concluya.

Pulse ***"OFF"*** para apagar la bomba, apague la instrumentacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Adelante"*** para realizar la sintonización del control PID.

REPITA TODO EL PROCEDIMIENTO DESDE EL PASO 2 HASTA EL PASO 15, AHORA SELECCIONADO LA ESTRATEGIA DE CONTROL PID SERIE INDUSTRIAL.

Una vez completas las tablas 4 y 5, compre los resultados, al igual que los esfuerzos de control obtenidos y concluya.

Pase a la practica 3.

Tabla 4. Medición cuantitativa respuesta del sistema

Estructura Utilizada	**Modo Operación**	**Valor Estado estable (cms)**	**Valor Cambio**	**MS (%)**	**TR (s)**	**TS (s)**
PID Paralelo Industrial	Servomecanismo					
PID Serie Industrial	Servomecanismo					
Estructura Utilizada	**Modo Operación**	**Valor Estado estable (cms)**	**Valor Perturbación**	**% Rechazo**	**TS (s)**	
PID Paralelo Industrial	Regulador					
PID Serie industrial	Regulador					

MS = Máximo sobreimpulso.
TR = Tiempo subida correspondiente al tiempo entre el 10% y el 90% de la respuesta del proceso
TS =tiempo establecimiento correspondiente al tiempo en entrar en la banda 2% del total del cambio.
% Rechazo= Porcentaje de rechazo de 100% de la perturbación.

Tabla 5. Medición cuantitativa técnica AWBT.

Estructura Utilizada	Modo Operación	AWBT	VEE (cms)	Valor Δs Manual	MS (%)	Tsat (s)	TS (s)
PID Paralelo Industrial	Servomecanismo	S					
		C					
PID Serie Industrial	Servomecanismo	S					
		C					
Estructura Utilizada	**Modo Operación**	**AWBT**	**VEE (cms)**	**VP %**	**MS (%)**	**TS (s)**	
PID Paralelo Industrial	Regulador	S					
		C					
PID Serie industrial	Regulador	S					
		C					

VEE= Valor estado estable. VP= Valor perturbación. S= sin AWBT. C= con AWBT.
MS = Máximo sobreimpulso.
Tsat = Tiempo total de saturacion del actuador
TS =tiempo establecimiento correspondiente al tiempo en entrar en la banda 2% del total del cambio.
% Rechazo= Porcentaje de rechazo de 100% de la perturbación.

PRACTICA 3. CONTROL EN CASCADA

Uno de los métodos más utilizados para reducir la maximo perturbaciones que entran en un proceso es el control en cascada. Dicho control puede acelerar también la respuesta del sistema de control, reduciendo la constante de tiempo de la función de transferencia del proceso que relaciona la variable manipulada con la salida del mismo. El control en cascada se define como la configuración donde la salida de un controlador de realimentación es el punto de ajuste para otro controlador de realimentación, por lo menos. Más exactamente, el control de cascada involucra sistemas de control de realimentación o circuitos que estén ordenados uno dentro del otro. Comúnmente, hay tres características principales presentes en el control en cascada para que sea eficaz. La constante de tiempo del circuito cerrado del circuito secundario o interno debe ser menor que un tercio de la constante de tiempo del circuito primario o externo, el circuito secundario debe incluir una fuente de perturbación de proceso importante, y la variable de proceso que se regula debe ser capaz de desplazar a la variable controlada primaria a su valor deseado.

Algunas de las ventajas del control en cascada son.

- Las perturbaciones en el lazo interno o secundario son corregidas por el controlador secundario, antes de que ellas puedan afectar a la variable primaria.

- Cualquier variación en la ganancia estática de la parte secundaria del proceso es compensada por su propio lazo.
- Las constantes de tiempo asociadas al proceso secundario son reducidas drásticamente por el lazo secundario.
- El controlador primario recibe ayuda del controlador secundario para lograr una gran reducción en la variación de la variable primaria.

La selección de la variable controlada secundaria es tan importante en un sistema de control en cascada que es muy útil formalizar algunas reglas que ayuden a la selección.

Regla 1. Diseñar el lazo secundario de manera que contenga las perturbaciones más influyentes. Estas perturbaciones, las cuales entran en el lazo secundario son las únicas para las cuales el sistema de cascada mostrará mejoría sobre el control de retroalimentación convencional.

Regla 2. Seleccionar una variable secundaria cuyos valores estén definidamente y fácilmente relacionados a los valores de la variable primaria. Durante una operación no perturbada la relación entre la variable primaria y la variable secundaria debe estar representada por una sola línea y si esta es una línea recta, la sintonización de los controles es mucho más simple.

Regla 3. Incluir en el lazo secundario tantas perturbaciones como sea posible, manteniéndolo al mismo tiempo, un control relativamente rápido.

La estrategia mas general para ajustar los controladores en cascada teniendo en cuenta que se realiza de manera secuencial es la siguiente.

El controlador secundario es ajustado primero porque el lazo secundario afecta la dinámica a lazo abierto del lazo primario. Se debe poner el controlador esclavo en automático y sintonizar el controlador secundario como un lazo PID normal. Durante este experimento (por ejemplo curva de reacción), el controlador primario no esta en operación (debe estar en manual).

El lazo externo se sintoniza (con el lazo interno en automatico) usando cualquiera de los métodos de sintonización disponibles. Se puede sintonizar como un lazo PID normal.

La práctica 3 consiste en la identificación del sistema y sintonización de un control en cascada correspondiente a un control PI como controlador del lazo interno y un PID industrial AWBT como controlador del lazo externo. El método de sintonización utilizado en el control en cascada es el método convencional mencionado de manera general anteriormente.

Inicialmente el usuario se debe ubicar en la primera interfaz de la practica, ilustración 7. Los pasos a seguir en la práctica son los siguientes.

1. Ajuste la valvula de bypass para que permita un caudal de 6 gpm en el caudal de entrada.

2. Identificación del proceso secundario. En el mismo terminal de ejecución de tareas de tiempo real se corre la tarea de tiempo real y desde el mismo directorio (cd /home/usuario), se corre la tarea para identificar la dinámica del proceso secundario que para este caso esta representado por el actuador, digitando.

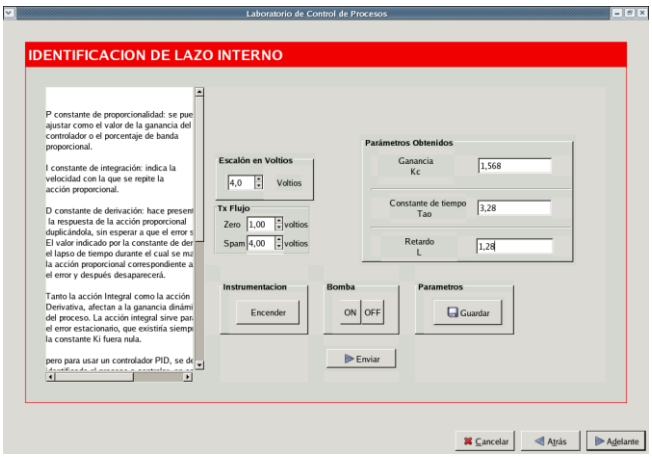
./ Identificación_lazo_interno_final -v

Ahora desde la interfaz *xrtailab* conecte la tarea de tiempo real y diríjase a la sección de la guía *Obtención de parámetros dinámicos de la respuesta de un proceso POMTM*, con el fin de analizar y seleccionar las opciones para configurar de los *scopes* y las opciones para la obtención de los parámetros dinámicos del proceso (Ganancia, retardo, constante de tiempo). Debe realizar los dos métodos propuestos.

Ahora se procede a generar el escalón hacia la entrada del actuador. Por medio del botón ***"Escalón en Voltios"***, se selecciona el valor de la consigna manual a la entrada del actuador, y con el botón ***"Enviar"*** se envía el valor seleccionado, ver ilustración 7. Por defecto el valor inicial es de 3 voltios espere que el sistema este en estado estable, y luego de esto inserte un escalón de 1 voltio. Observe el comportamiento del proceso y

aplique los métodos manual y automático para obtener los parámetros dinámicos. Con el mismo cambio de escalón puede aplicar los dos métodos. Una vez obtenidos los parámetros, deben ser almacenados en la tabla 6.

Ilustración 7. Práctica Identificación lazo interno



Valor en estado estable _____.

Tabla 6. Parámetros dinámicos Lazo secundario.

Método	K	Retardo	Constante de tiempo
Manual			
Automático			

Compare los valores obtenidos y seleccione cualquiera de los resultados. Almacénelos manualmente en las casillas correspondientes a cada parámetro en la interfaz de usuario y por medio del botón ***"Guardar"***, guarde los parámetros.

Pulse ***"OFF"*** para apagar la bomba, apague la instruementacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Adelante"*** para realizar la sintonización del control PI.

3. Sintonización control PI lazo secundario. Una vez se han guardado los parámetros obtenidos en el punto anterior, automáticamente se cargan en la ventana de sintonización del control PID. Seleccione un método de sintonización con la opción ***Métodos***, ver ilustración 8. *Click* en el botón ***"Sintonizar"*** y aparece la ventana de la ilustración 9, en donde aparecen los valores obtenidos automáticamente de las contantes de sintonización

del controlador PI, los botones para activar y desactivar la tecnica AWBT, la perturbación y los modos de operación manual o automatico; los botones para encender la instrumentación y la bomba y el estado del nivel en el tanque.

Ilustración 8. Selección métodos de sintonización control PI.

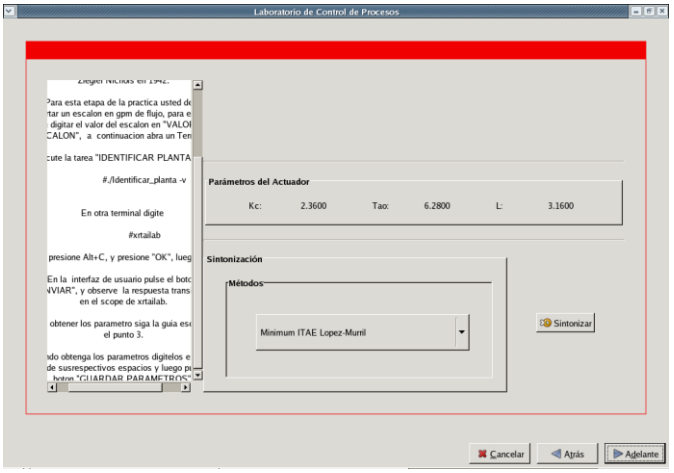
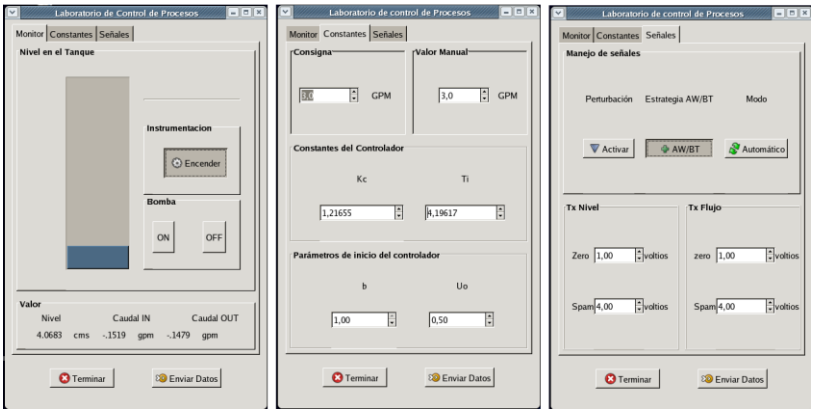


Ilustración 9. Sintonización Lazo interno controlador PI.



Corra la tarea de tiempo real desde el terminal donde ha corrido las tareas anteriores, digitando.

./ Monitor_pi_final –v

Desde la interfaz *xrtailab* conecte la tarea de tiempo real y seleccione la opción *Open/close scope manager* y configure las escalas *de cada scope* para observar las tendencias en el tiempo de cada variable.

Encienda la bomba con el botón ***"ON"*** y encienda la instrumentación con el botón ***"Encender"***. Los parámetros b y Uo aparecen inicialmente por defecto, pero pueden ser configurados por el usuario. Por medio del botón ***"Consigna"***, se selecciona el valor de la consigna a la entrada del actuador, y se presiona el botón ***"Enviar datos"*** para enviar todos los valores seleccionados, hacia la tarea de tiempo real. Por defecto el valor inicial es de 3 gpm. Ahora espere que el sistema este en estado estable. En estos momentos el controlador debe esta en modo automático.

Ahora envié un cambio positivo en el valor de consigna y posteriormente active la perturbación. Observe el comportamiento del proceso y del esfuerzo de control. Congele la grafica, mida y almacene los valores de la respuesta del sistema en la tabla 7.

Pulse ***"OFF"*** para apagar la bomba, apague la instruementacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Adelante"*** para realizar la identificación del lazo externo.

Tabla 7. Parámetros dinámicos lazo interno sintonizado.

Estrategia	Ms	Ts	Retardo	% Rechazo
PI				

4. Identificación lazo primario. En el mismo terminal de ejecución de tareas de tiempo real se corre la tarea de tiempo real y desde el mismo directorio (cd /home/usuario), se corre la tarea para identificar la dinámica del proceso primario correspondiente al nivel, para este caso esta representado por el control PI+Actuador+Planta, digitando.

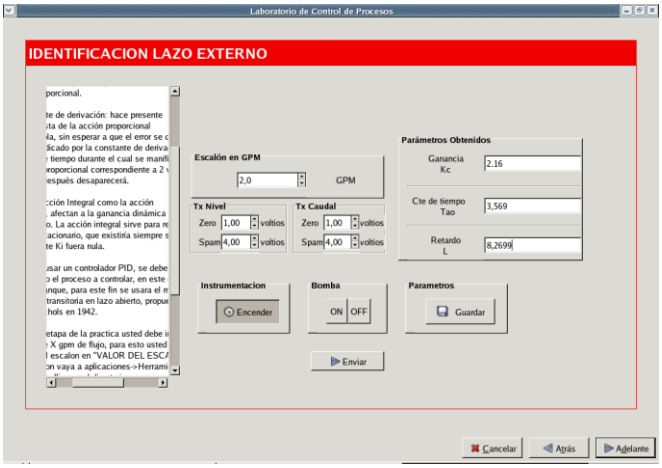
./ Identificación_lazo_externo_final -v

Desde la interfaz *xrtailab* conecte la tarea de tiempo real y diríjase a la sección de la guía *Obtención de parámetros dinámicos de la respuesta de un proceso POMTM*, con el fin de analizar y seleccionar las opciones para configurar de los *scopes* y las opciones para la

obtención de los parámetros dinámicos del proceso (Ganancia, retardo, constante de tiempo). Debe realizar los dos métodos propuestos.

El usuario se debe encontrar en la interfaz mostrada en la ilustración 10. Ahora se procede a generar el escalón hacia la entrada del actuador. Por medio del botón ***"Escalón en GPM"*** se selecciona el valor de la consigna a la entrada del actuador, y con el botón ***"Enviar"*** se envía el valor seleccionado. Por defecto el valor inicial es 3 gpm, espere que el sistema este en estado estable, y luego de esto cambie la consigna a 4 gpm. Observe el comportamiento del proceso, aplique los métodos manual y automático para obtener los parámetros dinámicos, congele la grafica para el método manual. Con el mismo cambio de escalón puede aplicar los dos métodos. Una vez obtenidos los parámetros, deben ser almacenados en la tabla 8.

Ilustración 10. Identificación lazo externo (Proceso Nivel)



Valor en estado estable _____.

Tabla 8. Parámetros dinámicos lazo primario.

Método	K	Retardo	Constante de tiempo
Manual			
Automático			

Compare los valores obtenidos y seleccione cualquiera de los resultados. Almacénelos manualmente en las casillas correspondientes a cada parámetro en la interfaz de usuario y por medio del botón ***"Guardar"***, guarde los parámetros.

Pulse ***"OFF"*** para apagar la bomba, apague la instrumentacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Adelante"*** para realizar la sintonización del control PID.

5. Sintonización control PID lazo primario. Una vez se han guardado los parámetros obtenidos en el punto anterior, automáticamente se cargan en la ventana de sintonización del control PID, ver ilustración 11. Seleccione las siguientes opciones.

Tipo de PID ->Paralelo

Métodos -> *Seleccione un método de sintonización.*

Para el proceso primario el control PID funciona como servomecanismo, es decir se optimiza para realizar buen seguimiento ante cambios en el valor de consigna.

Click en el botón ***Sintonizar*** y saldrá la ventana de la ilustración 12, en donde aparecen los valores obtenidos automáticamente de las contantes de sintonización para el controlador PID, los botones para activar y desactivar la tecnica AWBT, la perturbación y los modos de operación manual o automatico; los botones para encender la instrumentación y la bomba y el estado del nivel en el tanque.

Ilustración 11. Selección métodos de sintonización control PID – Lazo primario.

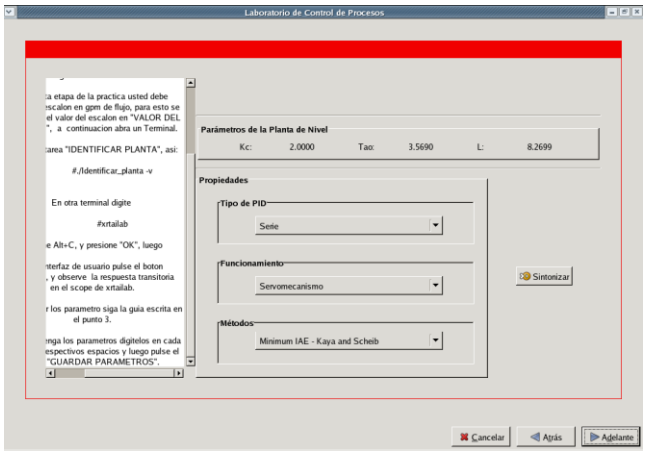
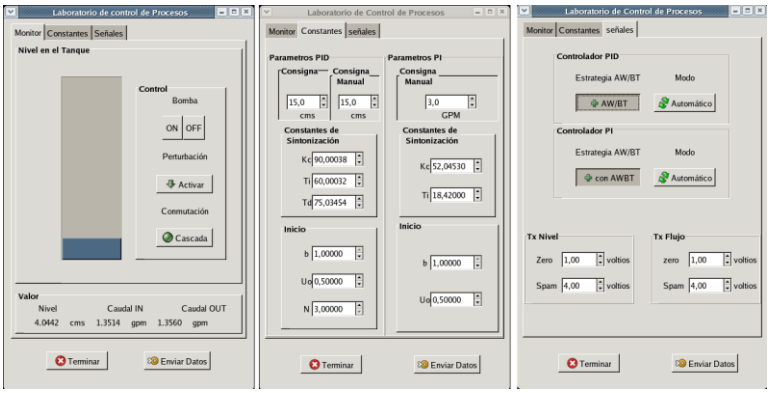


Ilustración 12. Sintonización Lazo primario controlador PID – Control Cascada.



Corra la tarea de tiempo real desde el terminal donde ha corrido las tareas anteriores, digitando.

./control_cascada –v

Desde la interfaz *xrtailab* conecte la tarea de tiempo real y seleccione la opción *Open/close scope manager* y configure las escalas *de cada scope* para observar las tendencias en el tiempo de cada variable.

Encienda la bomba con el botón ***"ON"***. Los parámetros b, N y Uo del PID aparecen inicialmente por defecto, al igual que los del control PI, pero pueden ser configurados por el usuario. Por medio del botón ***"Consigna"***, se selecciona el valor de la consigna a la entrada de la planta, y se presiona el botón ***"Enviar"*** para enviar todos los valores seleccionados, hacia la tarea de tiempo real. Por defecto el valor inicial es de 15 cms Ahora espere que el sistema se estabilice. En estos momentos el controlador PI debe estar en modo automático al igual que el controlador PID, es decir el control se ha implementado el control en cascada.

Ahora cambie a 20 cms el valor de consigna, observe el comportamiento del proceso y del esfuerzo de control. Congele la grafica, mida y almacene los valores de la respuesta del sistema en la tabla 9.

Ahora active la perturbación con el botón ***"Activar"***, permita que la perturbación este presente en el proceso durante u tirmpo t fijo y desactivela, espere a que el sistema se estabilice, congele la señal de salida y obtenga el valor de valor del porcentaje de rechazo.

Analice la respuesta del sistema ante la presencia de la perturbación y concluya.

Tabla 9. Parámetros dinámicos lazo externo sintonizado.

Estrategia	Ms	Ts	Retardo	% Rechazo
CASCADA*				

* La estrategia de control es en cascada, el control PID y el control PI se encuentran en modo automático.

Compare los resultados obtenidos de los porcentajes de rechazo entre el PID en sus dos estructuras y el control en cascada.

Concluya.

Pulse ***"OFF"*** para apagar la bomba, apague la instruementacion por medio del botón ***"Encender"*** desconecte la tarea de tiempo real por medio del botón ***"Desconectar"*** en la interfaz *xrtailab* y de *click* en ***"Terminar"*** para finalizar la practica.

OBTENCIÓN DE PARÁMETROS DINÁMICOS DE LA RESPUESTA DE UN PROCESO DE POMTM

La obtención de los parámetros dinámico de un proceso el usuario los podrá realizar de dos formas: Manual con *Xrtailab* y automática con *Xrtailab/Scilab*

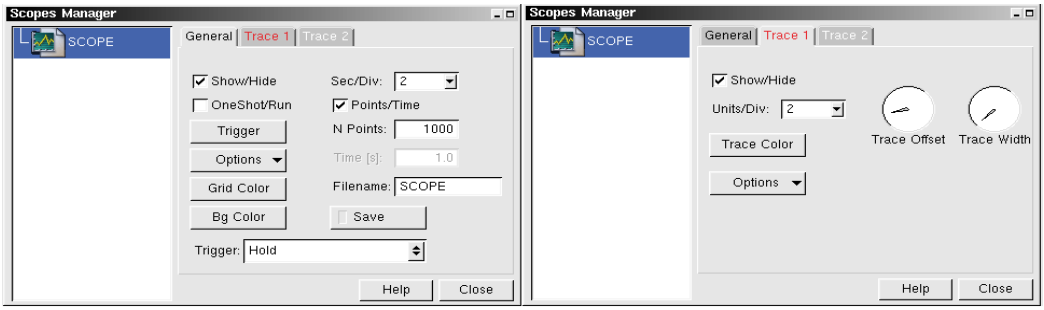
METODO MANUAL CON XRTAILAB

Inicialmente se debe correr la tarea de tiempo real y hacer el llamado desde un terminal al osciloscopio xrtailab con el siguiente comando:

xrtailab –v

Una vez aparece la ventana de *xrtailab*, se debe conectar con la tarea de tiempo real y seleccionar en la barra del menú, la opción correspondiente a los *scopes* llamada *Open/close scope manager* una vez se abre la ventana, tal como aparece en la ilustración 13 , se selecciona la señal por medio del respectivo *scope* que aparece en la columna derecha del *Scope Manager.* En la ventana aparece el menú *General* y *Traces.*

Ilustración 13. Ventana Scopes Manager.



General: esta ventana presenta las siguientes opciones:

Show/Hide: activa o desactiva el scope actual.

Sec/Div: se configura el valor de la escala de tiempo en que se mostraran las señales.

Trigger: activa o desactiva el modo de mostrar las señales.

Options:

Show grid: activa o desactiva la cuadricula del *scope.*

Show Tickmarks:

Show Cursors: activa/desactiva los cursores y el valor del tiempo actual.

Horizontal bars: activa/desactiva ejes horizontales en los cursores.

Vertical bars: activa/desactiva ejes verticales en los cursores.

Grid Color: opción para cambiar el color a la cuadricula del *scope.*

Bg Color: opción para cambiar el color del fondo del *scope.*

Points/Time: permite seleccionar la cantidad de almacenamiento de datos de las señales del *scope.*

N Points: cantidad de puntos a almacenar.

Time (s): si no se selecciona Points/Time se puede configurar con esta opción el espacio de tiempo para almacenar los datos.

Filename: nombre del archivo resultante del almacenamiento de los datos. Por defecto usa el nombre del *scope*, pero el usuario lo puede modificar.

Save: opción para guardar las señales en el archivo de la opción Filename. Depende la configuración de puntos o tiempo seleccionada anteriormente por el usuario.

Trace: el identificador numérico hace referencia al orden de ingreso de las señales configuradas desde el diagrama de bloques, y presenta las siguientes opciones:

Show/Hide: muestra/oculta la señal correspondiente.

Units/Div: se configura el valor de la escala de las unidades en el eje *Y* en que se mostraran las señales.

Trace Offset: configura la posición dela señal dentro del *scope.*

Trace Width: configura el grosor de la señal.

Trace color; selecciona el color de la señal.

Options:

Draws Zero Axis: muestra el eje cero de la señal

Draws Axis Label: muestra la etiqueta de la señal, por defecto Trace.

Show Units/div: muestra el valor de las unidades de la señal en el eje Y.

Show min: muestra el valor mínimo actual de la señal.

Show max: muestra el valor máximo actual de la señal.

Show Peak to Peak: muestra el valor pico a pico de la señal.

Show average: muestra el valor actual.

Show RMS: muestra le valor efectivo de la señal.

Show value y1: muestra el valor del primer cursor en el eje Y.

Show value y2: muestra el valor del segundo cursor en el eje Y.

Show value dy: muestra la diferencia de los dos anteriores.

Show Value dy/dt: muestra la derivada con respecto al tiempo de dy.

AC Coupling: elimina el componente de DC en la señal.

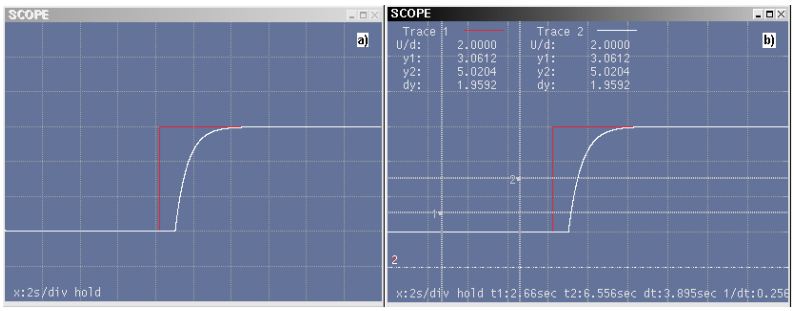
Cada ventana posee las opciones *Help* para la ayuda y *Close* para cerrar la ventana.

Una vez conocidas las opciones de los menús, se procede a correr la tarea de tiempo real, para realizar la identificación de los parámetros de la dinámica del proceso con base en la respuesta ante un escalón. Para entender mejor se utilizará como ejemplo la planta de POMTM mostrada en al ecuación 1 y se introducirá a la entrada de la misma un cambio de escalón de 6 unidades de la variable manipulada, estando la variable controlada en un valor estable de 2 unidades.

$$G(s) = \frac{e^{-0.8s}}{0.66s + 1} \tag{1}$$

Con el sistema en un valor de estado estable se le aplica un escalón al sistema de manera que se pueda observar la respuesta del sistema en el *scope* de *xrtailab,* tal como se muestra en la figura 2a, en ella se muestran la señal del valor de consigna (en rojo) y la respuesta del proceso (en blanco), en donde se ha aplicado al sistema un cambio del escalón de un valor de 6 unidades, partiendo de un valor estable de 2 unidades. En este momento se selecciona la opción *General/Trigger/Hold*, la cual permite congelar la imagen. También se deben seleccionar todas las opciones del menú *General/Options* por medio de *click* sostenido. Una vez seleccionadas se da doble *click* dentro del *scope* y aparece un cursor, con un segundo doble *click* aparece el segundo cursor. Si se ubica encima de cualquier cursor, con *click* sostenido se puede desplazar el cursor por toda la pantalla del *scope*. Luego se seleccionan las opciones Draws Zero Axis, Draws Axis Label, Show Units/div, Show value y1, Show value y2, Show value dy del menú *Trace x/Options* para la señal que se desea medir, donde x equivale al número de la señal. Con esta ultima elección se podrán ver en el *scope* los valores que marcan los cursores tanto en el eje y como en el eje “X” correspondiente al tiempo. En la figura 78 se aprecian las opciones seleccionadas anteriormente.

Ilustración 14. Ventana Scopes Manager.

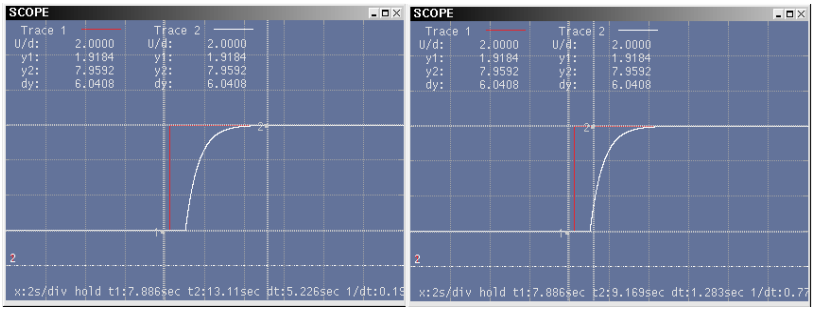


Obtención de la ganancia del sistema

Se debe ubicar el cursor 1 en el valor mínimo de la señal que corresponde a la respuesta del proceso (en blanco), ver ilustración 15, el cursor 2 se ubica en el valor máximo de la misma señal, en el *scope* aparece el titulo "*dy"* que equivale a 6.0408, este valor equivale al valor del cambio ante un cambio en el valor de consigna. De igual manera se para la señal del valor de consigna (en rojo), se observa el titulo "*dy*" para la señal roja y su valor es 6.0408. Luego la ganancia es la división de los valores obtenidos de la respuesta del total del cambio del sistema entre el valor del cambio de la señal de consigna.

$$K = \frac{dy\,proceso}{dy\,consigna} = \frac{6.0408}{6.0408} = 1.0000 \qquad (2)$$

Ilustración 15. Medición cambio consigna y cambio respuesta planta POMTM

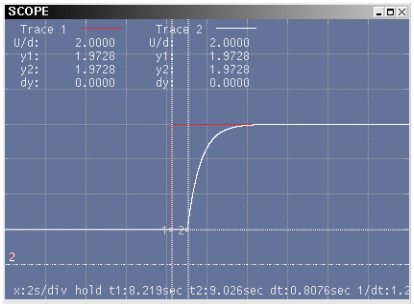


Obtención del retardo de tiempo

Tal como se aprecia en la ilustración 16, se ubica el cursor 1 en el instante de cambio del valor de consigna y el cursor 2 en el instante cambio de la señal del proceso. El valor del retardo será el valor mostrado en el titulo *"dt"* en la parte inferior del *scope*,

correspondiente a la diferencia de tiempo en los puntos en donde están ubicados los cursores 2 y 1. Para el ejemplo el valor de *dt*= 0.8076 segundos.

Ilustración 16. Medición retardo de tiempo respuesta planta POMTM.

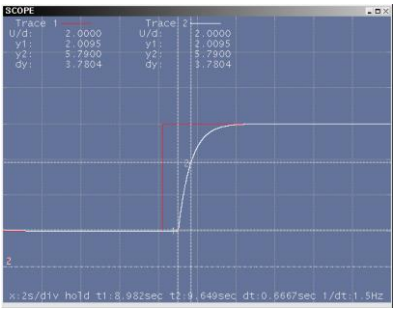


Obtención de la constante de tiempo

Para encontrar el valor de la constante de tiempo se debe saber el valor de la señal en el 63.2% del cambio que corresponde al Tao del sistema, ver ilustración 17. Con base en dicho valor y con el cursor 2 se busca en la señal del proceso dicho valor, una vez hallado, se ubica el cursor 1 en el punto de la reacción del sistema ante el escalón luego del retardo. Ahora se observa el valor del titulo "dt" en la parte baja del scope que corresponde al valor de la constante de tiempo del sistema. Para el ejemplo el valor del 63.2% se obtiene por medio de la ecuación 3 y se obtiene el valor del Tao =dt=0.6667.

$$Valor_{tao} = Valor\ inicial + \ dy * 0.632 = 2 + 6 * 0.632 = 5.792 \qquad (3)$$

Ilustración 17. Medición constante de tiempo respuesta planta POMTM.



MÉTODO AUTOMÁTICO CON XRTAILAB/SCILAB

Automáticamente se pueden obtener los parámetros dinámicos por medio de los datos generados por *xrtailab* de cada una de las señales que se muestran en el *scope*. *Xrtailab* posee la propiedad de generar un archivo de texto plano con los valores muestreados de cada una de las señales mostradas y también del tiempo de la tarea de tiempo real, dichos de cada señal se presentan en forma de vectores, cada vector correspondiente a cada señal Los datos se pueden almacenar de dos formas: la primera de acuerdo a la cantidad de datos definidos por el usuario, por medio de la selección de la opción General/ *Points/Time* y en la opción *N points* el usuario introduce la cantidad de puntos que desea almacenar en el archivo. La segunda de acuerdo a un intervalo de tiempo definido por el usuario con la opción *General/Time,* se debe deseleccionar *Points/Time* y en la opción *Time* se introduce el tiempo en segundo durante el cual se desea almacenar los datos.

Una vez conocido las opciones de almacenamiento de los datos, el usuario debe correr la tarea de tiempo real y configurar el *scope* de acuerdo a la figura 77 y se procede a realizar el cambio de escalón en la entrada del sistema con el sistema en estado estable. Para este caso se utiliza la misma la tarea de tiempo utilizada en el caso de obtención de parámetros en forma manual. La opción de almacenamiento será la segunda opción, es decir usando un intervalo de tiempo de 20 segundos.
Cuando el sistema esta en estado estable, para este caso 2 unidades, se debe usar la opción *General/Save* para que el sistema inicie el almacenamiento de los datos, luego se introduce el cambio de escalón al sistema y se espera a transcurra el tiempo de almacenamiento. Una vez finalizado el almacenamiento se comprueba la existencia del archivo generado en el directorio actual. Para el ejemplo, el nombre del archivo es SCOPE, que se muestra en la opción *General/Filename* de la figura 77, el nombre del archivo puede ser modificado a conveniencia del usuario.

Una vez se crea el archivo, en un *terminal* y desde el mismo directorio donde se corre la tarea*, se* escribe *scilab*, para llamar el programa que procesará la información del archivo por medio de una función de código llamada *calculo_parametros.sci*. Una vez el

programa abre (aparece un *prompt* como el de *Matlab*), en el prompt se escriben los siguientes comandos:

```
--> exec('calculo_parametros.sci');   // Carga la función que calcula los parámetros
--> x=read('SCOPE',-1,3);             // Lee los datos almacenados en el archivo SCOPE
--> [K,Tao,Ret]=calculo_parametros(x) // Llamado a la función para calcular parámetros
```

Y la salida es

```
--> Ret
        0.80187
-->Tao
        0.6579
--> K
        1.
```

Como se puede observar la función devuelve los parámetros dinámicos de la respuesta de la planta de POMTM con base en los datos del archivo generado.

Luego los parámetros dinámicos de la planta obtenidos con ambos métodos se almacenan en la tabla 10.

Tabla 10. Parámetros dinámicos obtenidos por métodos manual y automático

	K	Tao	Retardo
Planta utilizada	1	0.66	0.8
Xrtailab (Método manual)	1	0.6667	0.8076
Xrtailab/Scilab (Método Automático)	1	0.6579	0.80187

A continuación se describe el código utilizado en la función *calculo_parametros.sci*

[k,tao,ret]=calculo_parametros(x)

```
sp_min=min(x(:,2));          // Se obtiene el valor mínimo del vector del escalón.
sp_max=max(x(:,2));          // Se obtiene el valor máximo del vector del escalón.
delta_sp=sp_max-sp_min;      // se obtiene el valor del cambio del escalón.

vc_min=min(x(:,3));          // Se obtiene el valor mínimo del vector de respuesta del sistema.
vc_max=max(x(:,3));          // Se obtiene el valor máximo del vector de respuesta del sistema.
delta_vc=vc_max-vc_min;      //Se obtiene el valor del cambio de la respuesta del sistema ante el
                             // escalón.
```

```
//obtención de la ganancia del sistema

k=delta_vc/delta_sp;          // División entre el valor del cambio de a respuesta del sistema
                              // entre el valor del cambio del escalón.

// Obtención del retardo del sistema

indices_sp=find(x(:,2)>sp_min);          //Obtención de índices de los valores finales del escalón.
primer_valor_sp=indices_sp(1);           //Se obtiene el primer valor del vector anterior.
valor_cambio=x(primer_valor_sp,2);       //Se obtiene el valor del cambio del escalón.
valor_t_cambio_sp=x(primer_valor_sp,1);   // Se obtiene el valor del tiempo que se presenta
                                          // escalón en el vector de tiempo.

indices_vc=find(x(:,3)>vc_min);          //Obtención de índices en el vector de respuesta sistema.
primer_valor_vc=indices_vc(1);           //Se obtiene el primer valor del vector.
valor_t_reaccion=x(primer_valor_vc,1); // Se obtiene el valor en el momento que reacciona el
                                       // sistema.

ret=valor_t_reaccion-valor_t_cambio_sp;      //Resta del tiempo de reacción del proceso menos
                                             //el tiempo de cambio del escalón.

// Obtención de la constante de tiempo del sistema

valor_tao=vc_min + (delta_vc*0.632);         //Se obtiene el valor en el 63.2% en el vector de
                                             //respuesta del sistema.

indices_tao=find(x(:,3)>valor_tao);      // Obtención vector de índices en respuesta del sistema
primer_valor_tao=indices_tao(1);         // Se obtiene el primer valor del vector.
valor_t_tao=x(primer_valor_tao,1);       // Se identifica el valor en el vector de tiempo que
                                         // corresponda con el valor en el 63.2%

tao=valor_t_tao-valor_t_reaccion;        //Resta del tiempo en el 63.2% menos el tiempo en el
//momento de cambio del escalón
```