SISTEMA DE PROTOTIPADO RÁPIDO DE CONTROL PARA EL MÓDULO DE PRACTICAS MIC955 DE LA EMPRESA FEEDBACK

ANEXOS

Oscar Fernando Jaramillo Chamorro
José Ricardo Galíndez López

Director
Mg. Juan Fernando Flórez Marulanda

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Abril de 2011

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE CUADROS

PREFACIO

Este documento contiene varios apartados que lejos de ser una producción autónoma e inédita, son 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: GUIA DE INSTALACIÓN DEL SISTEMA OPERATIVO GNU LINUX FEDORA 7

En este anexo se detallan las instrucciones de instalación del sistema operativo GNU Linux Fedora 7 para la arquitectura 32-bit PC (i386). Los requisitos mínimos para instalar GNU Linux Fedora 7 son los siguientes:

Procesador: Intel Pentium 2-4, Celeron, AMD Duron, Athlon, Semprom u Opteron

Memoria RAM: mínima (128 MB), recomendada (256 MB)

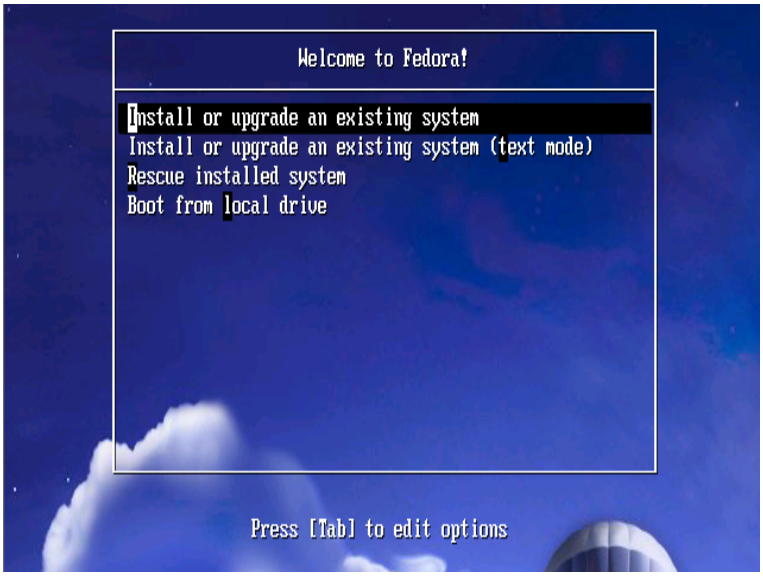
Espacio en disco duro: mínimo (500 MB), recomendado (3 GB)

1. Inserte el DVD de instalación de Fedora 7 en la bandeja de la unidad de DVD – ROM.

   Nota: Si al encender el PC no reconoce el DVD, se debe configurar la BIOS para arrancar desde la Unidad de DVD-ROM, para ello se ingresa en la BIOS, normalmente presionando la tecla esc, F2 o supr, esto depende de cada PC; seguidamente se cambia los parámetros de arranque en Boot Sequence. Activar en primer lugar que el PC arranque desde CD-ROM, guardar los cambios, salir y reiniciar el PC.

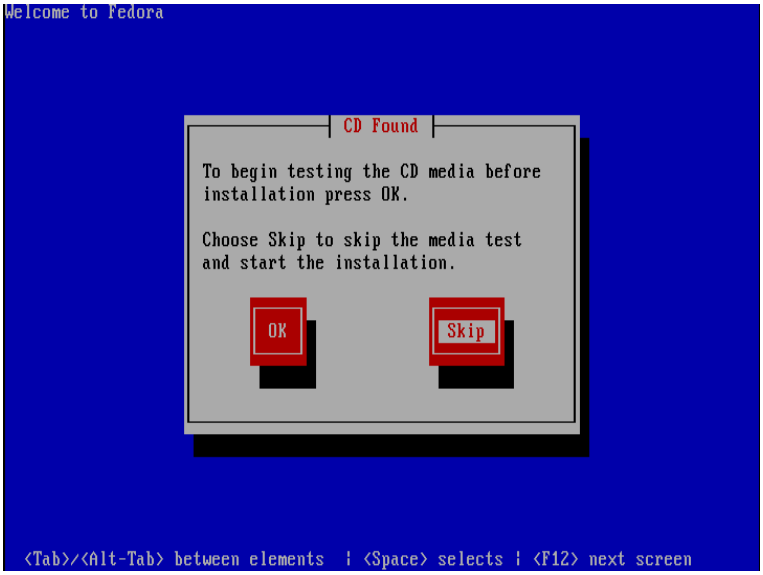
2. Al reiniciar el pc, deberá observarse la pantalla de bienvenida que se muestra en la figura 1. Seleccione "**Install or upgrade an existing system**". Presione la tecla Enter para continuar.

Figura 1: Ventana de bienvenida a la instalación de GNU Linux Fedora 7



3. En la siguiente ventana (figura 2) ubíquese en “Skip” utilizando la tecla “Tab”, con el fin de evitar que se haga una prueba del DVD para reconocer errores en el medio. Presione Enter para continuar.

Figura 2: Prueba del medio de instalación del sistema operativo GNU Linux Fedora 7

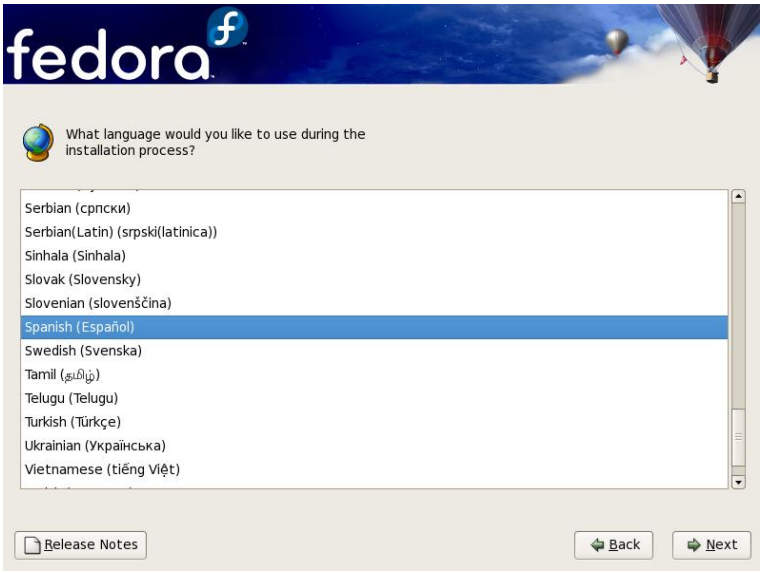


4. La pantalla de Bienvenida que se muestra en la figura 3, no pide ninguna información. Haga clic en “Next”.

Figura 3: Pantalla de inicio de Fedora 7

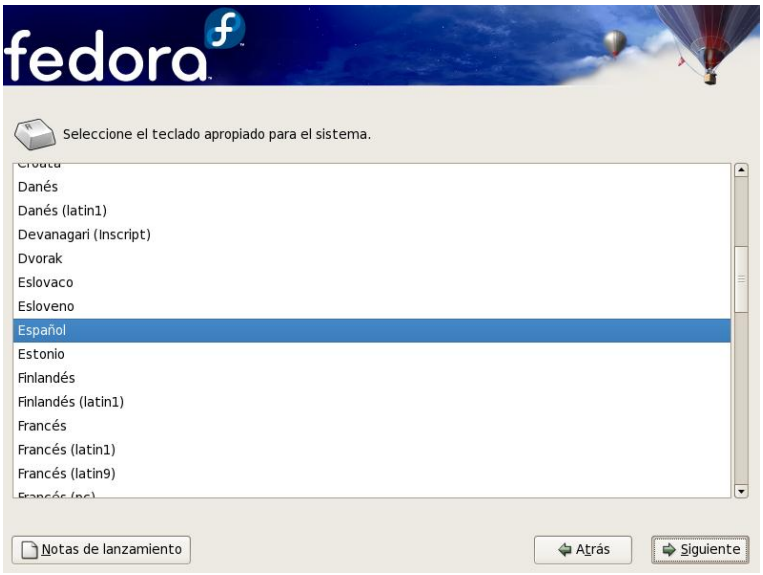
5. La ventana de selección de lenguaje que muestra en la figura 4, permite elegir el idioma que se usará durante el proceso de instalación del GNU Linux Fedora 7. Seleccione el idioma a utilizar durante la instalación. Haga clic en “Next”

Figura 4: Ventana de selección de Idioma para el proceso de instalación de GNU Linux Fedora 7



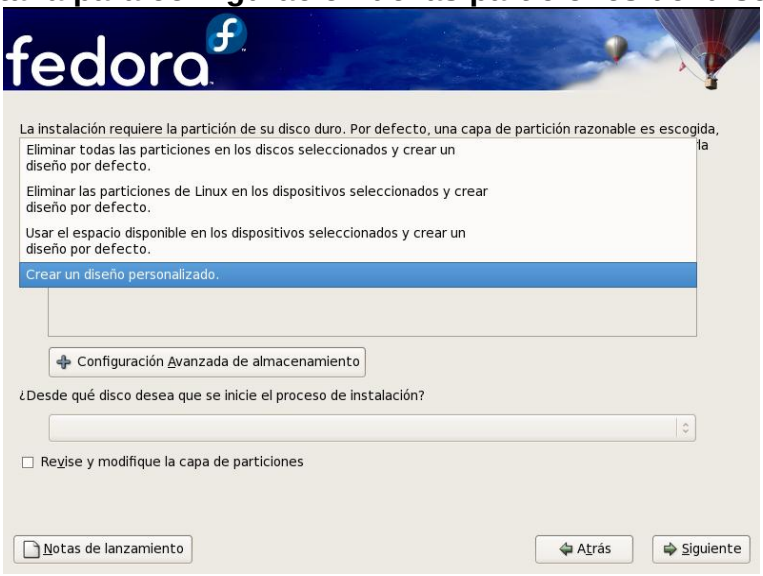
6. La ventana que se muestra en la figura 5, permite la configuración del teclado apropiado para el sistema. Seleccione el tipo de teclado que se utilizará durante el proceso de instalación y como teclado predeterminado del sistema. Haga clic en “Siguiente”.

Figura 5: Ventana de selección de Idioma para el proceso de instalación de GNU Linux Fedora 7



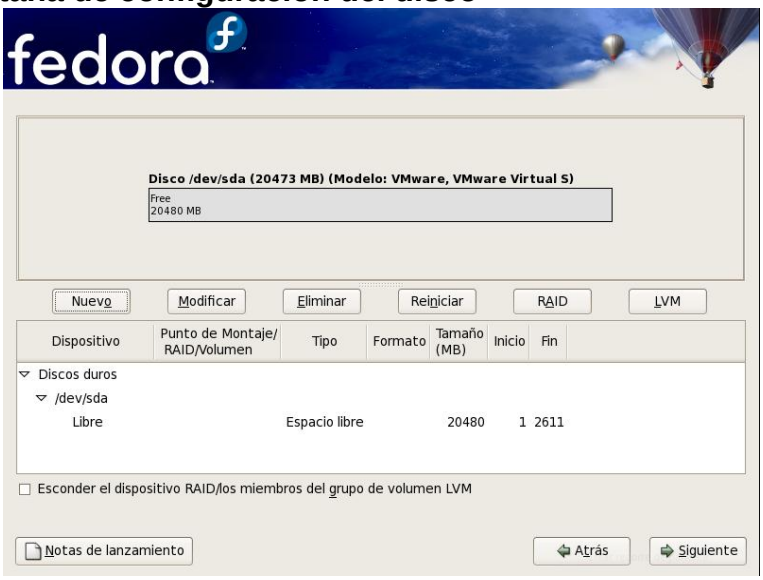
7. Si el programa de instalación no encuentra particiones legibles en los discos existentes, pregunta si debe inicializar el disco duro. Esta operación provoca que cualquier dato que se encuentre en el disco sea ilegible. Si su sistema tiene un nuevo disco duro sin ningún sistema operativo instalado, o si ha removido todas las particiones, responda Si.
8. La ventana que se muestra en la figura 6, permite el ingreso para realizar la configuración del particionamiento del disco. Seleccione “Crear un diseño personalizado”. Haga clic en “Siguiente”.

Figura 6: Ventana para configuración de las particiones del disco



9. En la ventana de configuración del disco (figura 7). Haga clic en “Nuevo”.

Figura 7: Ventana de configuración del disco



10. Configure las opciones del disco tal como se muestra en la figura 8.

    Nota: cuando establezca el tamaño de la partición en MB, DEBE dejar por lo menos 2000 MB libres, este espacio será utilizado como partición SWAP.

Figura 8: Opciones para partición ext3

Haga clic en “Aceptar”.

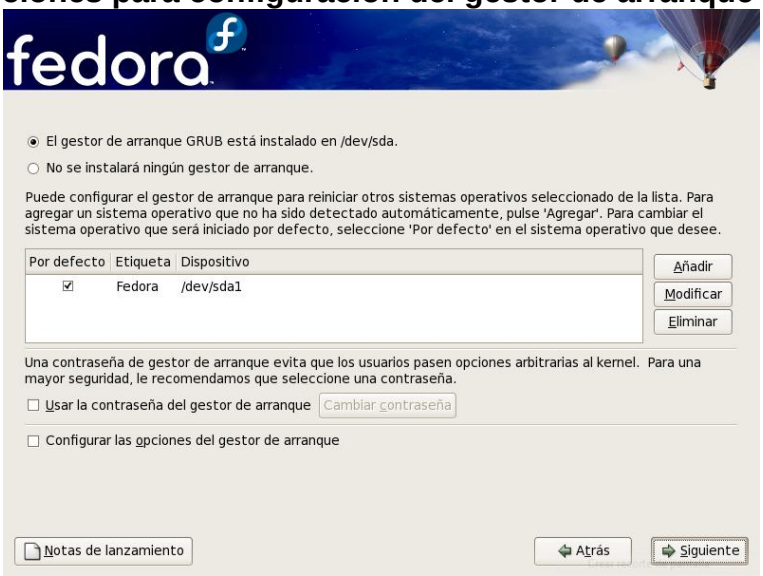
11. Seleccione nuevamente “Espacio libre”. Haga clic en “Nuevo”. Configure como se muestra en la figura 9.

Figura 9: Opciones para partición swap

Haga clic en “Aceptar”, luego en “Siguiente”.

12. En la ventana de configuración del gestor de arranque (figura 10), deje las opciones por defecto, haga clic en “Siguiente”.

Figura 10: Opciones para configuración del gestor de arranque



13. En la ventana de configuración de la red (figura11), deje las opciones por defecto. Haga clic en “Siguiente”.

Figura 11: Ventana de configuración de la red

14. En la configuración de zona horaria (figura 12). Seleccione en mapa la ciudad más cercana a su zona. Haga clic en “Siguiente”.

Figura 12: Ventana de configuración de la zona horaria

15. La siguiente ventana (figura 13), permite crear la contraseña del Administrador (*root*). Haga clic en “Siguiente”

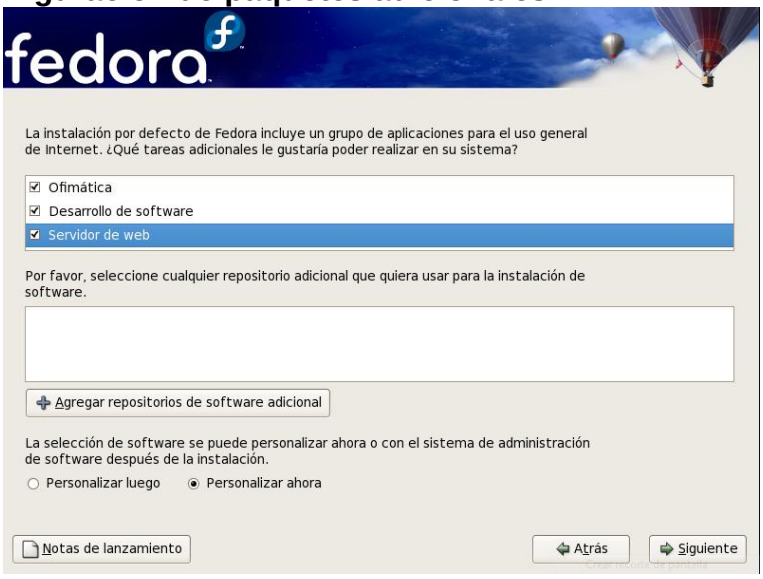
Figura 13: Creación de contraseña del Administrador (root)

16. La ventana de configuración de paquetes permite la instalación de paquetes adicionales. Seleccione:

- Ofimática
- Desarrollo de software
- Servidor de web

Seleccione “Personalizar ahora”. Haga clic en siguente.

Figura 14: configuración de paquetes adicionales



17. Seleccione cuidadosamente los paquetes adicionales tal como se muestra en las figuras 15 y 16:

Figura 15: Selección de paquetes adicionales Entornos de Escritorio

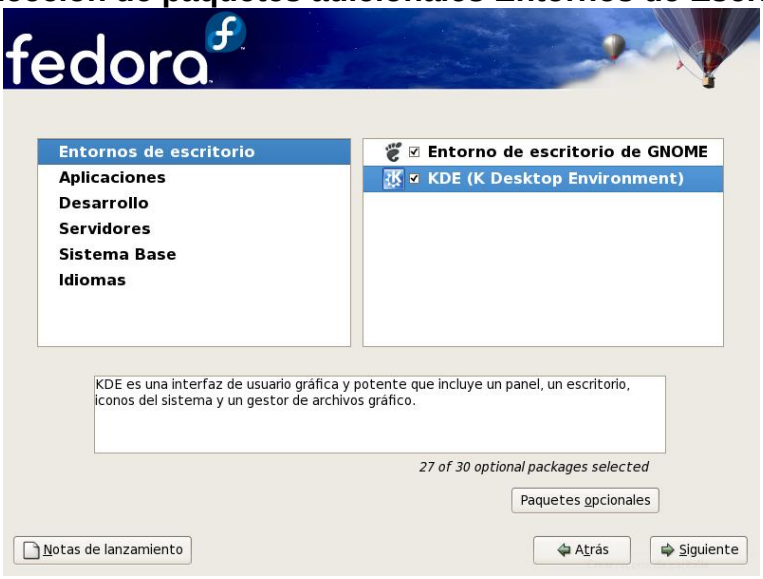


Figura 16: Selección de paquetes adiconales Sistema Base

Haga clic en “Siguiente”

18. En la ventana que se muestra en la figura 17, haga clic en Siguiente para comenzar la instalación.

Figura 17: Ventana de inicio de instalación de GNU Linux Fedora 7

Espere unos minutos mientras la instalación finaliza.

19. Cuando la instalación finazila, aparece la ventana que se muestra en la figura 18. Haga clir en reiniciar y remueva el DVD de instalación de Fedora 7

Figura 18: Ventana de finalización de la instalación de GNU Linux Fedora 7

20. La pantalla de Bienvenida (figura 19) no pide ninguna información, haga clic en siguiente.

Figura 19: Ventana de Bienvenido a Fedora

21. En las siguientes ventanas, podrá configurar su sistema GNU Linux. Se le pedirán algunos datos.
    Información de Licencia (figura 20). Haga clic en siguiente.

Figura 20: Ventana de información de Licencia

Cortafuegos: Dejar por defecto. Haga clic en siguiente

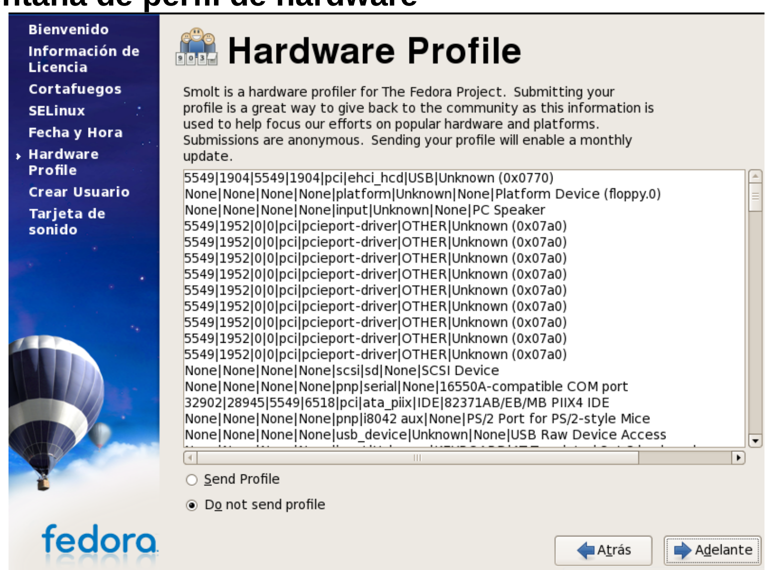
SELinux: Dejar por defecto. Haga clic en siguiente

Fecha y Hora.(figura 21) Establezca Fecha y Hora, haga clic en siguiente.

Figura 21: Ventana de configuración de fecha y hora

22. En la ventana de perfil de Hardware (figura 22), seleccionar “Do not send profile”. Haga clic en siguiente.

Figura 22: Ventana de perfil de hardware

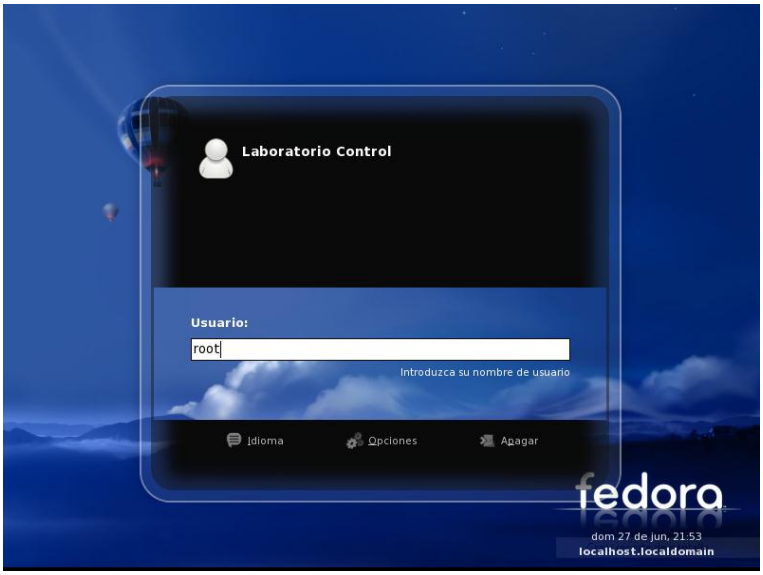


23. La ventana Crear Usuario (figura 23), permite la creación de una cuenta de usuario personal, rellene los datos de usuario.

Figura 23: Ventana de creación de usuario

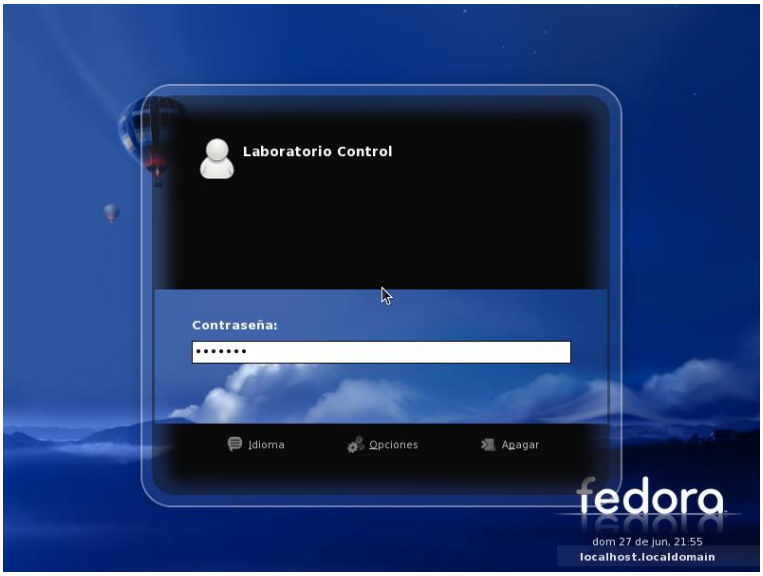
24. Tarjeta de sonido: Dejar por defecto. Haga clic en “Finalizar”
25. En la ventana de inicio (figura 24), digite “root” en el campo usuario y presione Enter.

Figura 24: Ventana de inicio de sesión



26. Digite la contraseña para root (figura 15)

Figura 25: contraseña de inicio de sesión



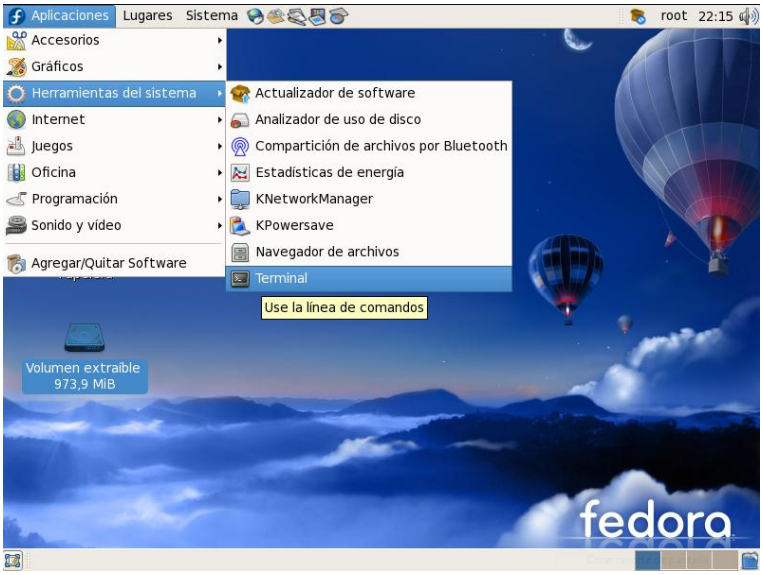
ANEXO B. GUIA DE INSTALACIÓN DE RTAI-LAB

En este anexo se detallan las instrucciones de instalación de RTAI-Lab y los paquetes necesarios para su correcto funcionamiento. Esta guía de instalación tiene como propósito ayudar a una instalación existosa de RTAI-Lab en el sistema operativo GNU Linux Fedora 7, para la instalación en otras distribuciones se debe consultar el manual de usuario de la distribución. Esta guía no pretende ser un compendio para usar y administrar un sistema operativo GNU Linux, por tal razón la información sobre los comandos que aquí se usan no es extensiva. El procedimiento de instalación comprende los siguientes pasos:

1. Iniciar Fedora 7 e ingresar como root.
2. Digitar la contraseña para root.
3. Copiar el contenido de la carpeta Paquetes, dentro de la carpeta de inicio de root. (ATENCIÓN: Sólo el contenido, NO la carpeta)
4. **Pre-Instalación RTAI-Lab**

Abrir un Terminal: Para ello ubíquese en el menú aplicaciones, Herramientas del sitema, Terminal, tal como se muestra en la figura 26.

Figura 26: Ejecución de un Terminal



Agregar los directorios /usr/local/lib/pkgconfig y /usr/local/lib a las variables de entorno PKG_CONFIG_PATH y LD_LYBRARY_PATH respectivamente. Para hacer esto digite los siguientes comandos en el Terminal:

NOTA: Despues de digitar cada comando, presione la tecla enter.

cd /root

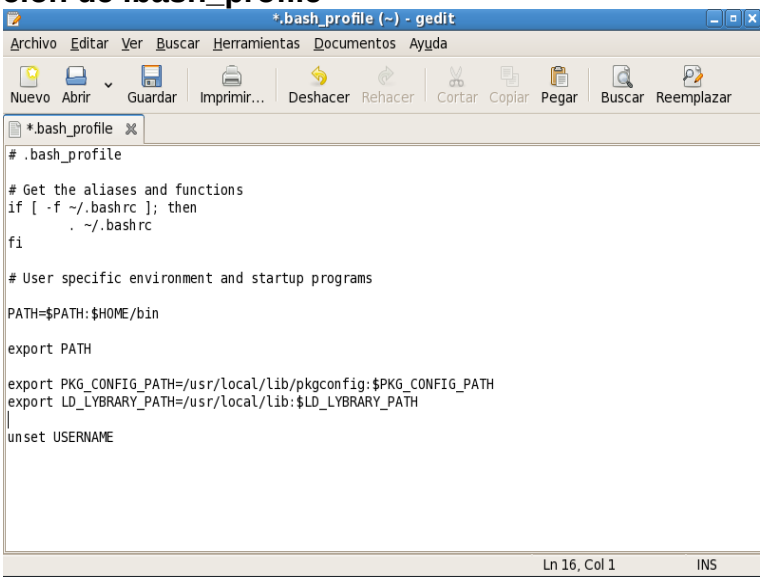
gedit .bash_profile

Agregue las siguientes líneas al archivo .bash_profile, tal como muestra la figura 27.

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH

export LD_LYBRARY_PATH=/usr/local/lib:$LD_LYBRARY_PATH

Figura 27: Edición de .bash_profile



27. **Instalación tcl8.5.9:** Digite los siguientes comandos en el terminal.

cd /opt

mv /root/tcl8.5.9.tar.gz /opt

tar xzvf tcl8.5.9.tar.gz

cd tcl8.5.9/unix

./configure

make

make install

28. **Instalación tk8.5.9:** Digite los siguientes comandos.

cd /opt

mv /root/tk8.5.9.tar.gz /opt

tar xzvf /root/tk8.5.9.tar.gz

cd tk8.5.9/unix

./configure

make

make install

29. **Instalación libgtkhtml-2.11.1:** Digite los siguientes comandos.

cd /opt

mv /root/libgtkhtml-2.11.1 /opt

tar xzvf /root/libgtkhtml-2.11.1

cd libgtkhtml-2.11.1

./configure

make

make install

30. **Instalación libzvt-2.0.1:** Digite los siguientes comandos

cd /opt

mv /root/libzvt-2.0.1.tar.gz /opt

tar xzvf libzvt-2.0.1.tar.gz

cd libzvt-2.0.1

./configure

make

make install

31. **Instalación MesaLib:** Digite los siguientes comandos

cd /usr/local/src

mv /root/MesaLib-7.7.1.tar.bz2 /usr/local/src

tar jxvf MesaLib-7.7.1.tar.bz2

cd Mesa-7.7.1/

make realclean

make linux-x86-static

make install

32. **Instalación eFLTK:** Digite los siguientes comandos.

cd /usr/local/src

mv /root/efltk-2.0.8.tar.gz /usr/local/src

tar xzvf efltk-2.0.8.tar.gz

cd efltk

autoconf

./configure --disable-mysql --disable-unixODBC

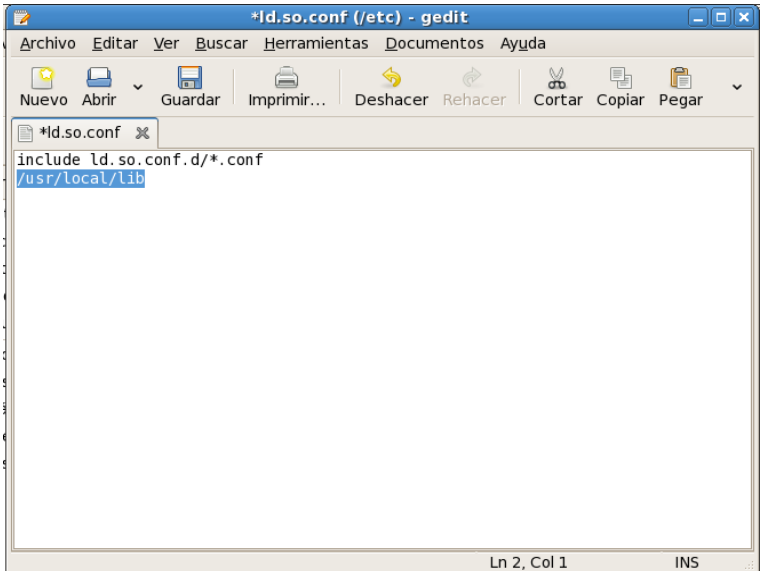
./emake

./emake install

gedit /etc/ld.so.conf

Agregar la línea “/usr/local/lib” (sin comillas) al archivo /etc/ld.so.conf, como se observa en la figura 28:

Figura 28: Edición de .ld.so.conf



Guardar y Salir.

/sbin/ldconfig

33. **Instalación de RTAI:** Digite los siguientes comandos.

cd /usr/src

mv /root/rtai-3.7.1.tar.bz2 /usr/src

tar xjvf rtai-3.7.1.tar.bz2

ln -s rtai-3.7.1 rtai

34. Parchar el Kernel de Linux

cd /usr/src

mv /root/linux-2.6.23.tar.bz2 /usr/src/

tar xjvf linux-2.6.23.tar.bz2

mv linux-2.6.23 linux-2.6.23-rtai

ln -s /usr/src/linux-2.6.23-rtai linux

cd /usr/src/linux

patch -p1 < /usr/src/rtai/base/arch/i386/patches/hal-linux-2.6.23-i386-1.12-03.patch

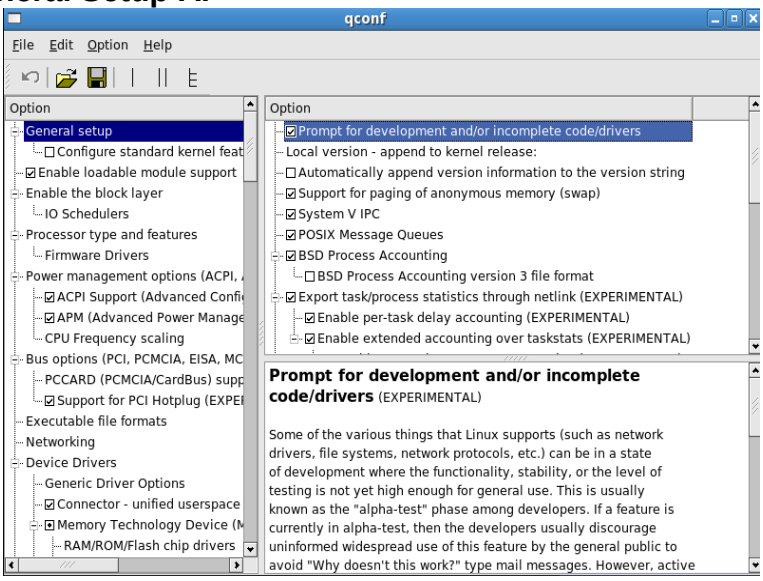
cp /boot/config-`uname -r` .config

make xconfig

Configurar el *kernel* como se observa en las figuras 29 a 43, verificando que estén seleccionadas las casillas resaltadas.

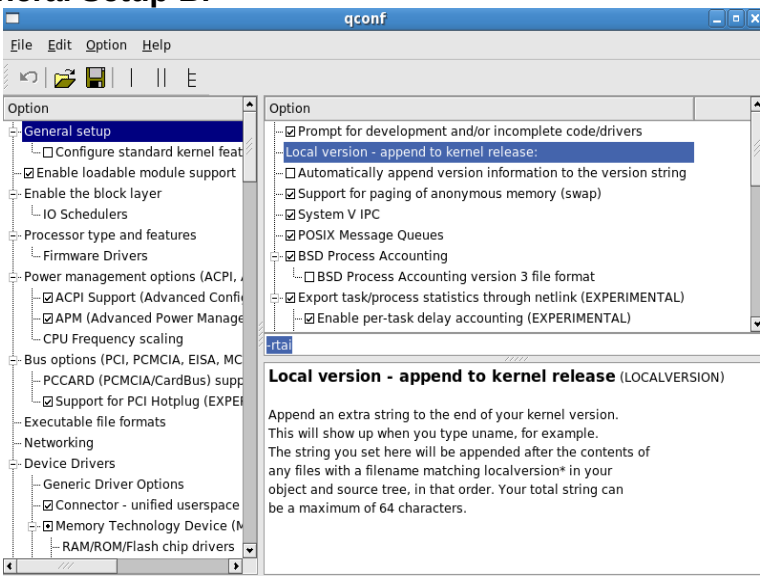
- **General setup:** Seleccionar “Prompt for development and/or incomplete code/drivers”

Figura 29: General Setup A.



Establecer “Local version – append to Kernel release: -rtai”

Figura 30: General Setup B.

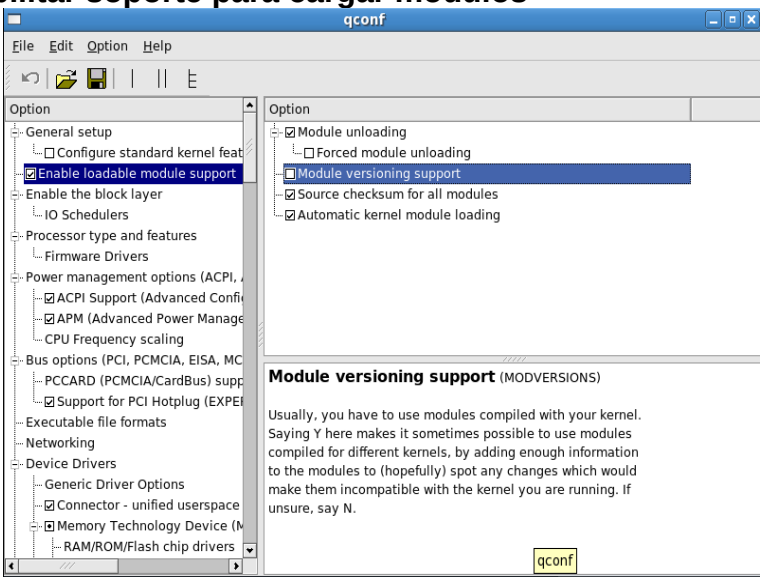


Presionar Enter.

- **Enable loadable module support:**

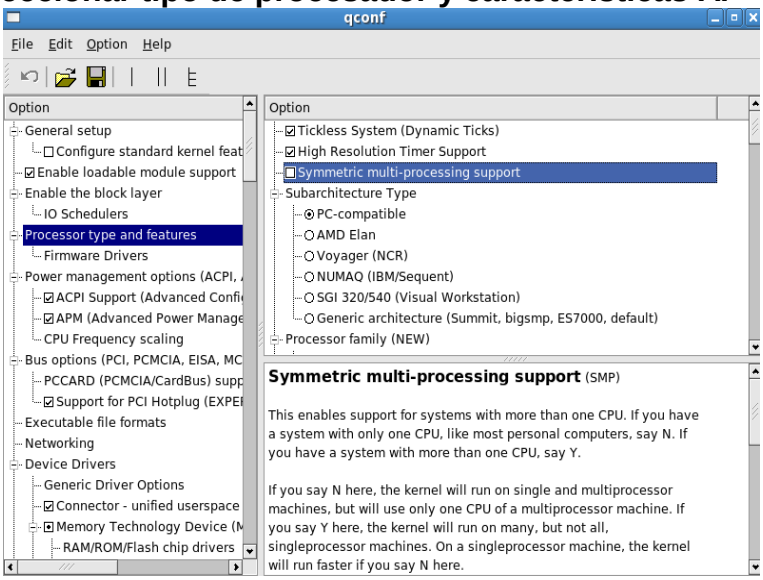
Seleccionar “Module unloading”, “Source cheksum for all modules”, and “Automatic kernel module loading”. Desmarcar “Module versioning support”.

Figura 31: habilitar soporte para cargar modulos



- **Processor type and features:** Si el PC, no posee un procesador de dos o más núcleos, desmarcar la opción “Symmetric multi-processing support” y seleccionar el tipo de Sub-arquitectura (PC-Compatible).

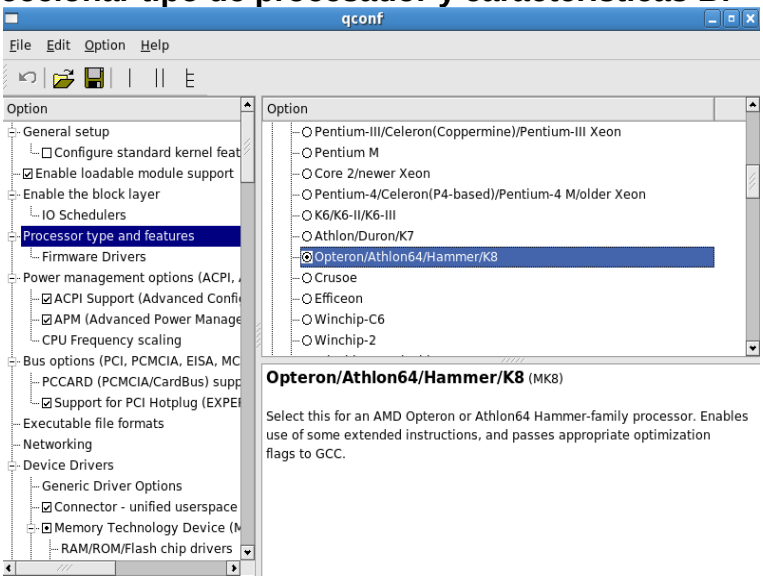
Figura 32: Seleccionar tipo de procesador y características A.



Seleccionar la familia del procesador, (en este caso era un AMD Athlon), para saber cual es la familia de procesadores del PC, digitar en un terminal el siguiente comando:

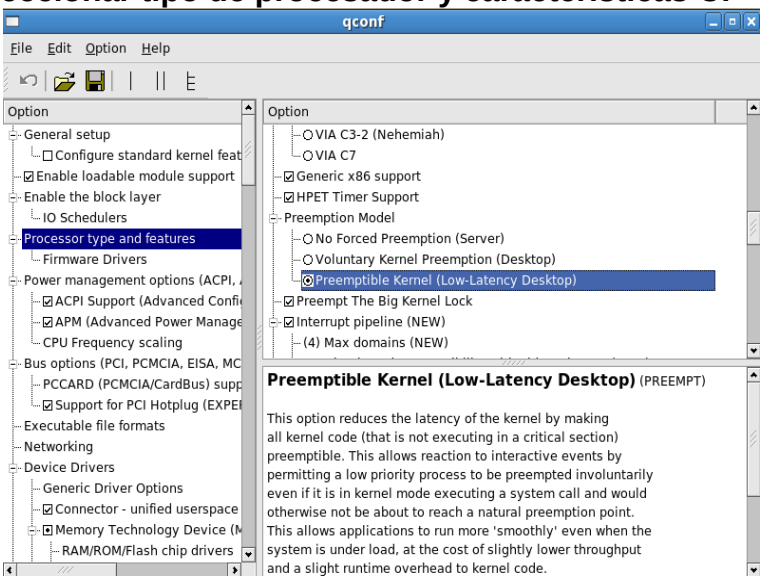
cat /proc/cpuinfo

Figura 33: Seleccionar tipo de procesador y características B.



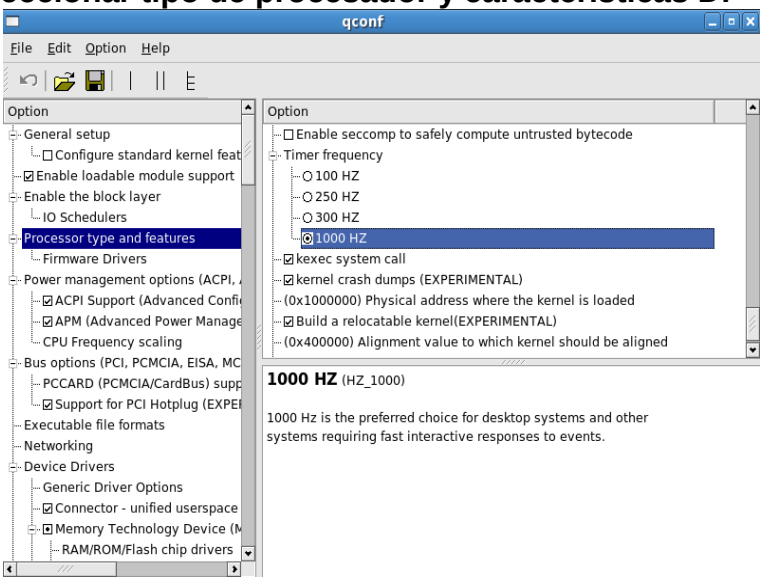
Seleccionar “Preemption Model” (Preemptible kernel (Low-Latency Desktop)

Figura 34: Seleccionar tipo de procesador y características C.



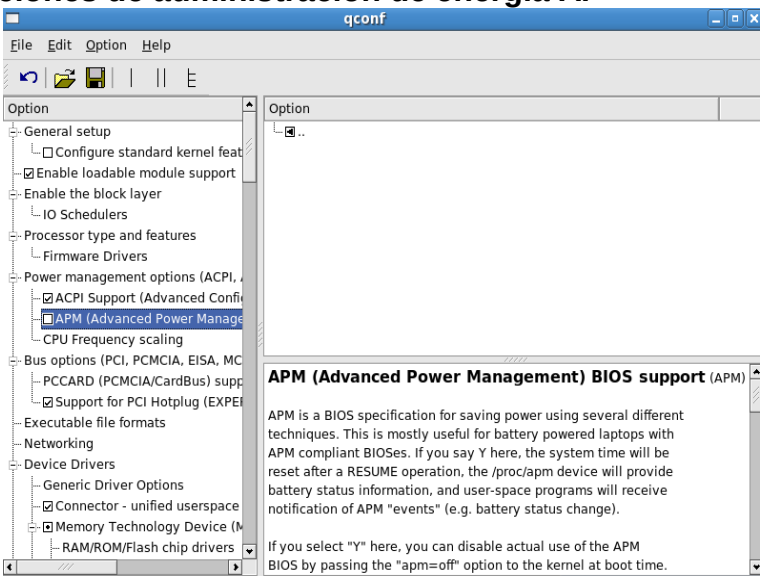
Seleccionar en “Timer frequency” 1000 Hz

Figura 35: Seleccionar tipo de procesador y características D.



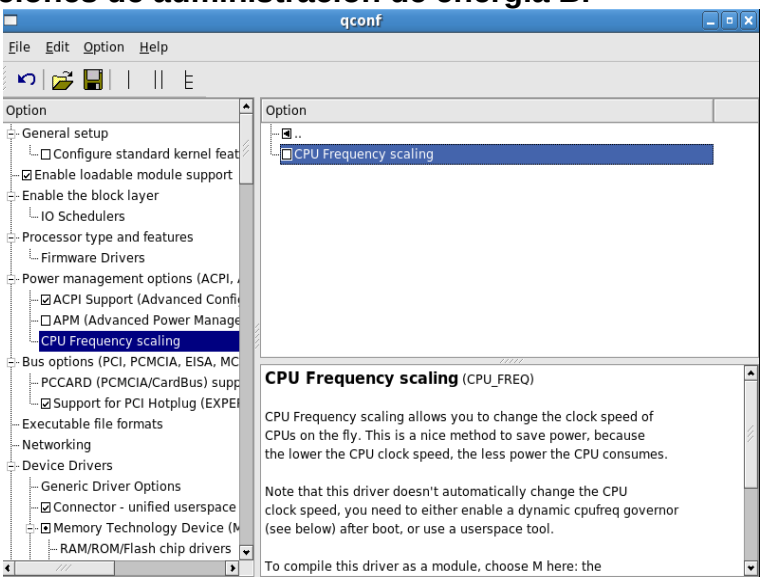
- **Power Management options:** Seleccionar ACPI Support (Advanced Configuration and Power Interface), Desmarcar APM.

Figura 36: Opciones de administración de energía A.



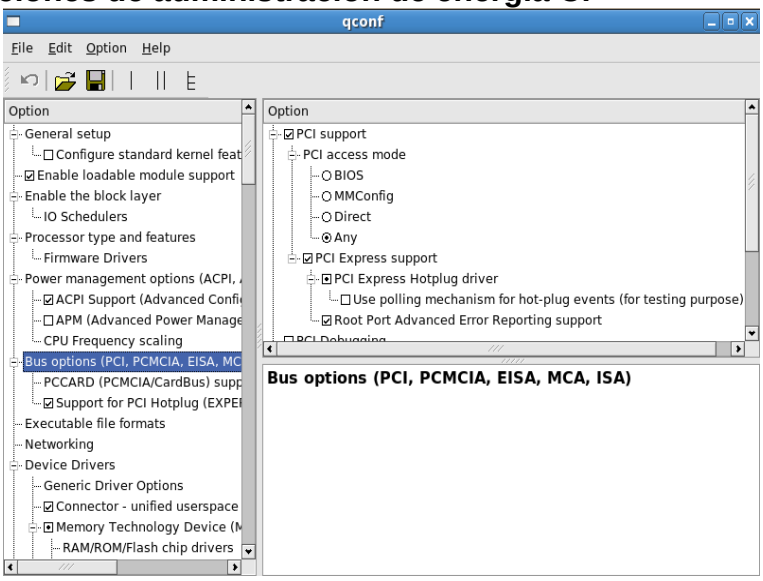
Desmarcar “CPU Frecuency scaling”

Figura 37: Opciones de administración de energía B.



- **Bus options:** por defecto

Figura 38: Opciones de administración de energía C.



- **Device Drivers:**

Generic driver options: por defecto

Memory Technology Devices (MTD): no es necesario.

Parallel port support: desmarquelo.

Figura 39: Opciones de Bus

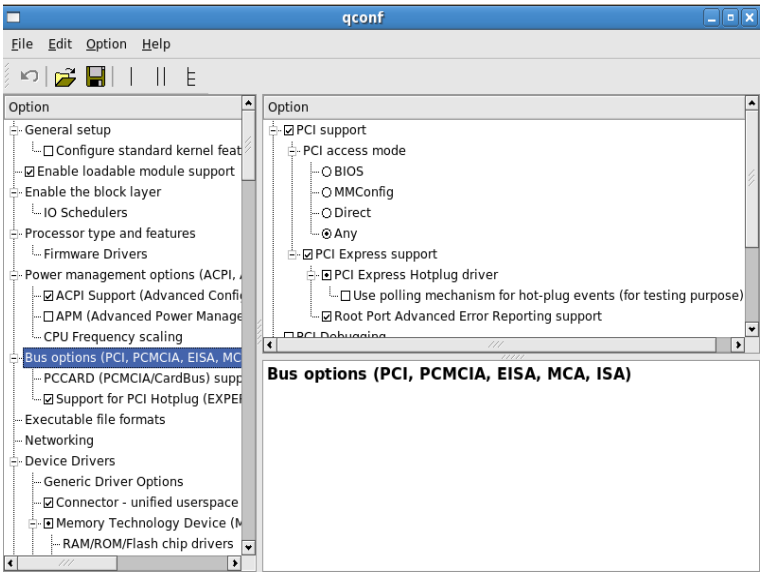
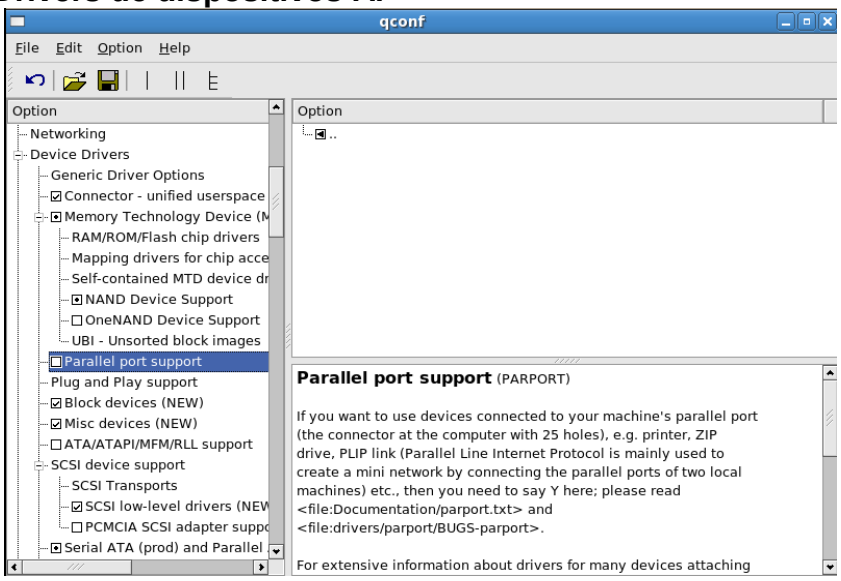


Figura 40: Drivers de dispositivos A.



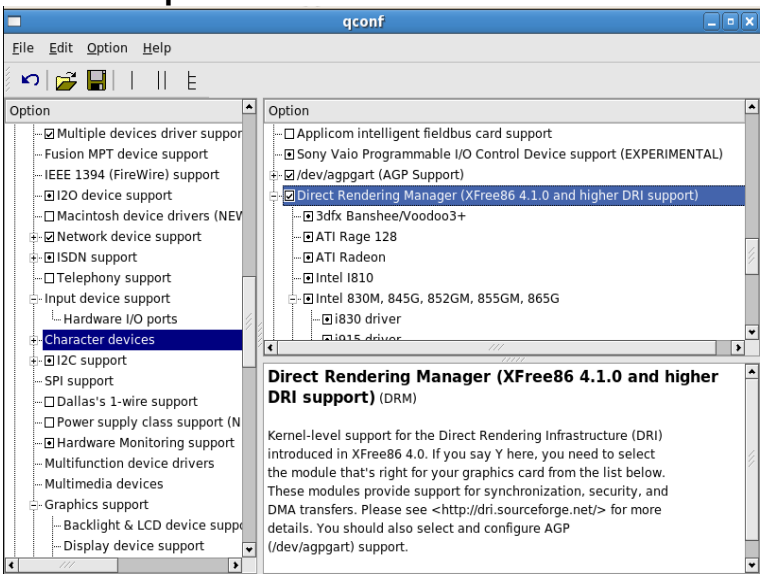
Plug and Play support: por defecto

Network device support: por defecto.

Input device support: por defecto.

Character devices: Asegurarse que "/dev/agpart (AGP Support)" y "Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)" están seleccionados

Figura 41: Drivers de dispositivos B.



I2C support: mantener deshabilitado.

Multimedia devices: mantener deshabilitado.

Graphics support: por defecto.

Sound: por defecto

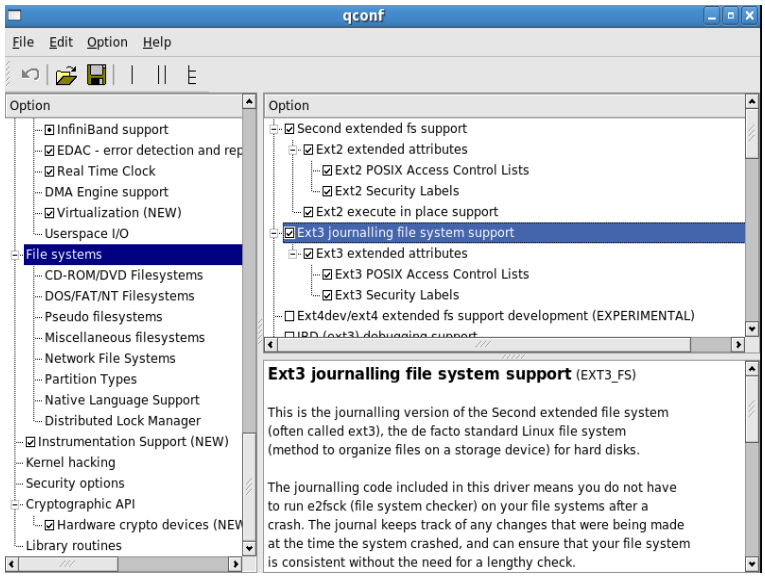
USB Support: por defecto

- **File Systems:**

Second extended fs support: habilitar

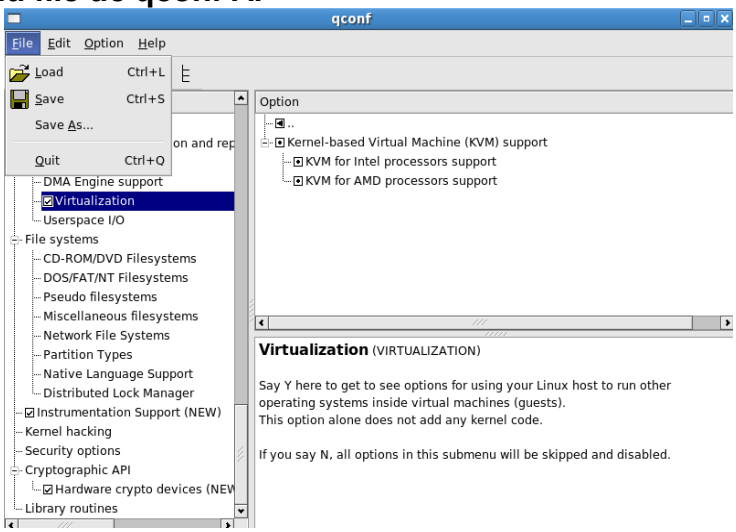
Ext3 journalling file system support: seleccionar éste y “Ext3 extended attributes”

Figura 42: Sistema de archivos



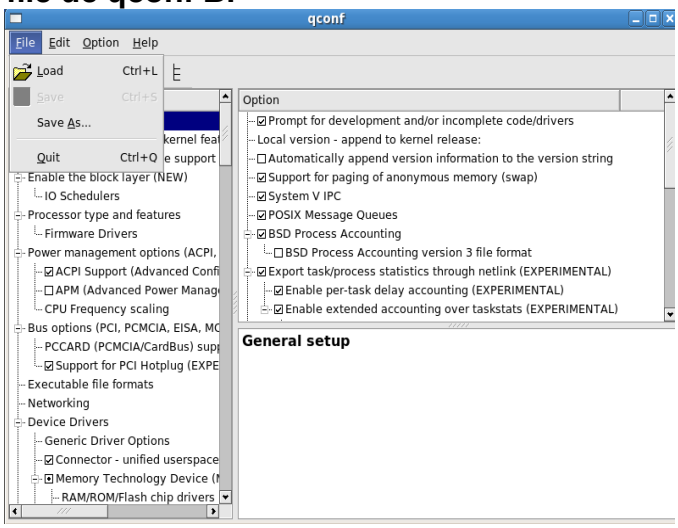
Guardar: File → Save

Figura 43: Menú file de qconf A.



Salir: File → Quit

Figura 44: Menú file de qconf B.



make

make modules_install

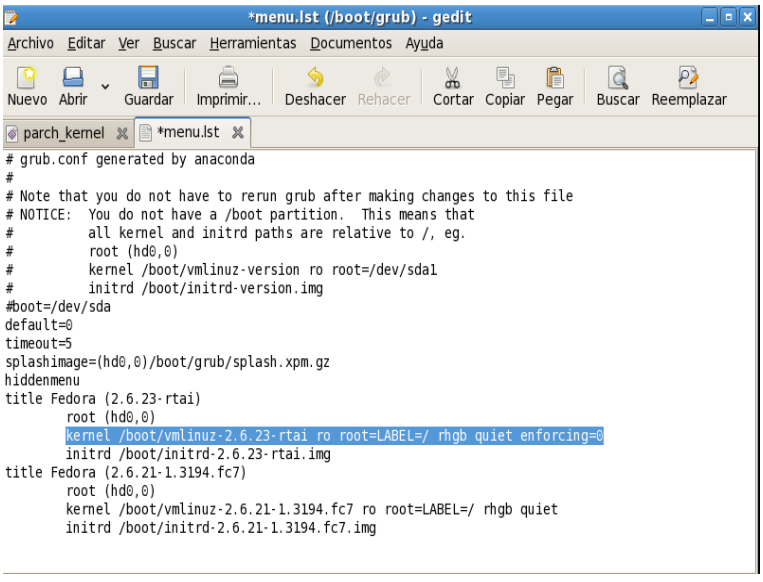
make install

gedit /boot/grub/menu.lst

Editar el archivo menu.lst para que el sistema inicie por defecto con el núcleo parchado. Cambiar la línea default=0

Adicionar enforcing=0 a kernel /boot/vmlinuz-2.6.23-rtai root=LABEL=/ rghb quiet

Figura 45: archivo menú.lst



reboot

35. **Instalación Comedilib:** Abrir un terminal y digitar los siguientes comandos:

cd /usr/local/src/

mv /root/comedilib-0.8.1.tar.gz /usr/local/src/

tar xzf comedilib-0.8.1.tar.gz

cd comedilib-0.8.1

sh autogen.sh

./configure --sysconfdir=/etc

make

make install

make dev

36. **Instalación RTAI 1st Paso:** Digitar los siguientes commandos:

cd /usr/src/rtai

make xconfig

Guardar y cerrar

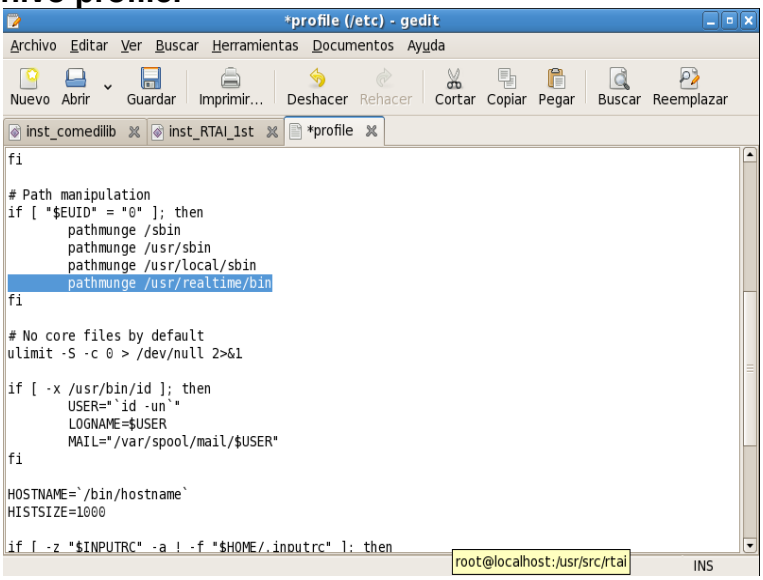
make

make install

gedit /etc/profile

Agregar el PATH /usr/realtime/bin a etc/profile como muestra la siguiente figura:

Figura 46: archivo profile.



37. **RTAI Test:** Digitar los siguientes comandos:

cd /usr/realtime/testsuite/kern/latency/

insmod /usr/realtime/modules/rtai_hal.ko

insmod /usr/realtime/modules/rtai_lxrt.ko

insmod /usr/realtime/modules/rtai_fifos.ko

insmod /usr/realtime/modules/latency_rt.ko period=1000000

./display

Para detener presionar Control + C

rmmod latency_rt

rmmod rtai_fifos

rmmod rtai_lxrt

rmmod rtai_hal

38. **Instalación Comedi:** digitar los siguientes comandos

cd /usr/local/src/

mv /root/comedi-0.7.76.tar.gz /usr/local/src/

tar xzf comedi-0.7.76.tar.gz

cd comedi-0.7.76

sh autogen.sh

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

make

make install

make dev

cp /usr/local/src/comedi/include/linux/comedi.h /usr/local/include/

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

mkdir /usr/local/include/linux

ln -s /usr/local/include/comedi.h /usr/local/include/linux/comedi.h

ln -s /usr/local/include/comedilib.h /usr/local/include/linux/comedilib.h

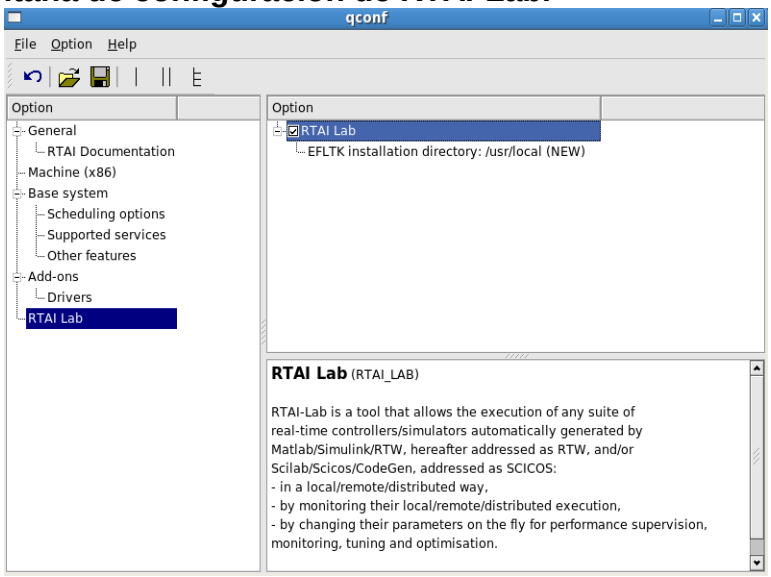
39. **Instalación RTAI 2° Paso:** Digitar los siguientes commandos:

cd /usr/src/rtai

make xconfig

Menu Add-ons: Seleccione “Real Time COMEDI support in user space”

Figura 47: ventana de configuración de RTAI-Lab.



Menu RTAI Lab: Seleccione RTAI Lab

Guardar y salir

make

make install

reboot

40. **Instalación Scilab/Scicos:** Digite los siguientes comandos:

cd /usr/local

mv /root/scilab-4.1.2-src.tar.gz /usr/local/

tar xzvf scilab-4.1.2-src.tar.gz

cd scilab-4.1.2

./configure --without-java --with-tk --with-gtk2

make all

ln -s /usr/local/scilab-4.1.2/bin/scilab /usr/local/bin/scilab

41. **instalación Add-ons RTAI-Lab:** Digite los siguientes comandos:

cd /usr/local

mv /root/scilab-4.1.2-rtailab.tgz /usr/local

tar xzvf scilab-4.1.2-rtailab.tgz

cd scilab-4.1.2-rtailab/macros

make install

make user

42. **Post-Instalación RTAI-Lab:** Digite los siguientes comandos:

cd /bin

mv /root/load /bin

mv /root/load_rtailab /bin

edite el archive load_rtailab, según sus necesidades.

gedit load_rtailab

gedit /root/.bash_profile

Agregue la línea load_rtailab al archivo bash_profile.

reboot

ANEXO C: CREACIÓN DE BLOQUES EN RTAI-LAB CON SCILAB/SCICOS 4.1.2

Un bloque es el módulo básico con el cual se construyen diagramas de bloques en Scicos. Los bloques corresponden a una operación y con la interconexión de los mismos se logra implementar un algoritmo. Normalmente, las paletas estándar de Scicos permiten la programación de diversas funciones, a través de bloques elementales, que permiten la creación de cualquier tarea. Sin embargo es posible crear nuevas funciones mediante la programación de funciones.

Cada bloque de Scicos está definido por dos funciones:

- Una "función de interfaz" (*Interfacing Function*) escrita en lenguaje Scilab, la cual maneja la interacción con el editor. Esta función específica la geometría del bloque, el número de entradas y salidas, el tipo de bloque y además permite al usuario definir el valor de los parámetros iniciales.
- La segunda función llamada "función computacional", escrita normalmente en lenguaje C, define el comportamiento de la función durante la simulación.

Antes de empezar el estudio de estas funciones es necesario tener unos conceptos básicos de programación en Scilab.

B.1. CONCEPTOS BÁSICOS DE PROGRAMACIÓN EN SCILAB

Un programa de Scilab es un conjunto de instrucciones ejecutadas en un orden específico. Estas pueden ser escritas en la ventana de Scilab o en un archivo ASCII, el cual es utilizado al ejecutar el comando de Scilab: *exec*. Este archivo se conoce con el nombre de *script* y puede contener funciones definidas. Por convención los nombres de los *script* terminan con el sufijo ".sce", pero si el *script* sólo contiene funciones definidas entonces se usa el sufijo ".sci".

Los ciclos o iteraciones y las instrucciones de ramificación son una parte fundamental de la programación en Scilab y se explican a continuación:

B.1.1. Ramificaciones

Las instrucciones de ramificación, se usan para ejecutar un conjunto de códigos dependiendo de condiciones booleanas. La forma más simple de ramificación en Scilab tiene la siguiente forma:

if <condition> then <instructions> end

El conjunto de instrucciones <*instructions*> se ejecuta si la condición <*condition*> es evaluada como un booleano de valor verdadero. Debido a la estructura matricial de Scilab, la evaluación de condiciones puede hacerse a través de una matriz booleana o una matriz escalar. Si se trata de una matriz, la condición se

considera verdadera sólo si todas las entradas de la matriz son valores booleanos verdaderos o si todas las entradas de la matriz son escalares no nulos.

En caso de ser necesario el uso de dos ramas se utiliza la siguiente sintaxis:

if <condition> then <instructions> else <instructions> end

El primer conjunto de instrucciones se ejecuta si la condición es verdadera, en caso contrario se ejecuta el segundo conjunto de instrucciones. Si es necesario ingresar o generar más ramificaciones se realiza utilizando sucesivamente la instrucción *else if.*

También es posible generar ramificaciones múltiples por medio de la ejecución de la instrucción *select,* la cual puede ser usada cuando la ejecución depende de un conjunto de valores predefinidos:

select <expr>
case <expr1> then <instructions>
case <expr2> then <instructions>
...
else <instructions>
end

En éste caso el valor de <expr> se compara con el valor de <expr1>, <expr2>… Si las expresiones evaluadas son iguales, entonces se ejecuta el conjunto de instrucciones, en caso de no coincidir ningún valor, serán ejecutadas las intrusiones dadas por *else* en el caso que este exista.

B.1.1. Ciclos o Iteraciones

Existen dos tipos de ciclos en Scilab: los lazos *for* y *while*. Para mejorar la eficiencia de la ejecución estos no son utilizados si existe una operación matricial que pueda reemplazarles, ya que estas se ejecutan con mayor eficiencia.

El lazo *for* tiene la siguiente sintaxis:

for <name>=<expr>
<instructions>
end

La expresión <expr> sólo se evalúa una vez. El conjunto de instrucciones internas se ejecutan y en cada ciclo del lazo la variable <*name*> tendrá un nuevo valor. Si la evaluación de <expr> da como resultado una matriz, el número de ciclos es igual al número columnas y la variable <*name*> obtendrá como valores los datos sucesivos de la columna en la matriz.

La instrucción *break* puede ser utilizada dentro del conjunto de instrucciones <*instructions*>, si esta se ejecuta, el ciclo se detiene inmediatamente.

El lazo *while* es utilizado cuando no es posible saber de antemano el número de ciclos que son necesarios para la ejecución. Su sintaxis es la siguiente:

while <condition>
<instructions>
end

El conjunto de instrucciones <*instructions*> se ejecutan mientras el valor booleano de <*condition*> es verdadero. Ya que la condición se evalúa constantemente no es necesario el uso del la instrucción *break*, sin embargo si la condición crea un lazo infinito sólo con la ejecución del *break* se dará fin al ciclo

B.1.1. Funciones en Scilab

Es posible definir funciones en Scilab. La función, a diferencia de los *scripts*, tiene un entorno local que se comunica con el exterior a través de argumentos de entrada y salida.

Una función se define usando la instrucción *function,* seguida por la sintaxis de llamado, un conjunto de instrucciones de Scilab y la instrucción *endfunction*.

function [<name1>,<name2>,...]=<name-of-function>(<arg1>,<arg2>,...)
<instructions>
endfunction

Cuando se define una función es necesario darle un nombre <*name-of-function*>, la lista de argumentos de llamada (<arg1>,<arg2>,...), y la lista de las variables que son usadas para retornar los valores (<*name1*>,<*name2*>,...). La función puede retornar más de un valor.

El llamado de una función se realiza de una de las siguientes dos maneras:

<name-of-function>(<expr1>,<expr2>,...)
[<v1>,<v2>,...,<vp>]=<name-of-function>(<expr1>,<expr2>,...)

En el primer caso el valor retornado corresponde al valor retornado por el primer argumento de la función evaluada. En el segundo caso los valores retornados por la función evaluada serán copiados en las variables [<v1>,<v2>,...,<vp>].

Cuando se llama una función, las expresiones dadas como argumentos son evaluadas primero y sus valores son enviados a la función. Scilab usa un mecanismo de llamado, el cual envía todos los tipos de argumentos. Si un argumento no se cambia durante la ejecución de la función, entonces no se copia durante la llamada. Una variable que no es llamada por la función, puede tener un valor asignado dentro del entorno local, sin que cambie su valor al final de la llamada.

La función normalmente se detiene cuando todas sus instrucciones internas son ejecutadas o cuando el flujo de instrucciones llega a una sentencia de *return*. Cuando ésta se detiene se retornan los valores <*name1*>, <*name2*>… que tienen adquiridos en el momento del fin de la ejecución de la función.

B.2 FUNCIÓN DE INTERFACE

Esta función es usada sólo en el editor de Scicos y especifica el nombre de la función computacional que será usada en el simulador. Maneja la siguiente sintaxis:

function [x,y,typ]=block(job,arg1,arg2)

Donde x, y, typ son los valores que serán retornados, *block* es el nombre de la función (es recomendable que coincida con el nombre del archivo) y *job*, *arg1*, *arg2* son los argumentos recibidos en la llamada.

Los valores que la función de interface puede devolver, dependen del parámetro *job*, el cual por medio de un *select* case puede devolver los siguientes valores:

***'*plot'**: La función dibuja el bloque y su etiqueta. 'arg1' es la estructura de datos del bloque. 'arg2', 'x', 'y', 'typ' no son usados. En general se usa la función *standard draw* para dibujar un bloque cuadrado.

'getinputs': Retorna la posición y el tipo de puertos de entrada (corrientes y de eventos). 'arg1' es la estructura de datos del bloque y 'arg2' no es usado. 'x' es el vector de las coordenadas "x" de los puertos de entrada, 'y' es el vector de las coordenadas "y" de los puertos de entrada, y 'typ' es el vector del tipo de puertos de entrada. En general se usa la función *standard input*.

'getoutputs': Retorna la posición y el tipo de los puertos de salida (corrientes y de eventos), 'arg1' es la estructura de datos del bloque, 'arg2' no es usado, 'x' es el vector de coordenadas "x" de los puertos de salida, 'y' es el vector de las coordenadas "y" de los puertos de salida, y 'typ' es el vector del tipo de puertos de salida. En general se usa la función *standard output*. Las funciones *standard input* y *standard output* colocan los puertos de entrada y salida regulares a los lados del bloque y los puertos de eventos de activación en la parte superior e inferior del bloque.

'getorigin': Retorna las coordenadas del punto inferior izquierdo del rectángulo que contiene la forma del bloque, 'arg1' es la estructura del bloque, 'arg2' no es usado, 'x' es el vector de coordenadas "x" de los puertos de salida, 'y' es el vector de las coordenadas "y" de los puertos de salida, 'typ' no es usado. Generalmente es usada la función *standard origin*.

'set': Esta función abre una caja de dialogo para la adquisición de parámetros del bloque, 'arg1' es la estructura de datos del bloque, 'arg2' no es usado, 'x' es la nueva estructura de datos del bloque, 'y' y 'typ' no son usados.

'define': Define los valores por defecto del bloque. Este establece su tipo, número de entradas y de salidas, etc. 'arg1' y 'arg2' no son usadas, 'x' es la estructura de datos del bloque, 'y' y 'typ' no son usadas.

En el caso 'set' es el único lugar donde se establece la interfaz con el usuario, esto se realiza a través de la función *getvalue* que se usa para crear una caja de

dialogo. Sin embargo, para ello es necesario conocer más a fondo la estructura de los datos de Scicos, la cual incluye los objetos *graphics*, *model*, *gui, doc* que se describen a continuación:

graphics: Es el objeto tipo gráfico de Scilab que incluye lo concerniente a la información grafica del bloque, sus componentes son:

- orig: Vector [xo,yo], donde 'xo' es la coordenada en "x" y 'yo' es la coordenada en "y" del origen del bloque.
- sz: Vector [w,h], donde 'w' es el ancho del bloque y h es su altura.
- flip: Booleano. Indica la orientación del bloque.
- exprs: Un vector de cadenas de caracteres y de expresiones formales (por lo general con los números y nombres de variables) utilizado en el cuadro de diálogo del bloque.
- pin: Vector. Indica el número del punto de enlace conectado en el puerto de entradas corrientes.
- pout: Vector. Indica el número del punto de enlace conectado en el puerto de salidas corrientes.
- pein: Vector. Indica el número del punto de enlace conectado al puerto de entrada de eventos.
- peout: Vector. Indica el número del punto de enlace conectado al puerto de salida de eventos.
- gr i: Vector de cadenas de caracteres incluido en las expresiones graficas de Scilab para dibujar el icono del bloque.
- id: Una cadena de caracteres que identifica el bloque. Esta es colocada por debajo del bloque en el diagrama.
- in implicit: Un vector de cadenas de caracteres que indica si la entrada del puerto es implícita o explicita.
- out implicit: Un vector de cadenas de caracteres que indica si la salida del puerto es implícita o explicita.

model: Es un objeto tipo modelo de Scilab. Sus componentes son:

- sim: Es una lista que contiene dos elementos, el primero es una cadena de caracteres que contiene el nombre de la función computacional y el segundo es un entero que especifica el tipo de la función computacional. En la actualidad se utilizan los tipos 4 y 5.
- in: Es un vector que especifica el número y el tamaño de las entradas corrientes.
- out: Es un vector que especifica el número y el tamaño de las salidas corrientes.
- evtin: Es un vector que especifica el número y el tamaño de las entradas de activación de eventos.
- evtout: Es un vector que especifica el número y el tamaño de las salidas de activación de eventos
- state: Vector que contiene el estado inicial de tiempo continuo.

- dstate: Vector que contiene el estado inicial de tiempo discreto.
- rpar: Vector de parámetros reales que son trasferidos a la función computacional asociada.
- ipar: Vector de parámetros enteros que son trasferidos a la función computacional asociada.
- blocktype: Puede ser configurado indiferentemente como c o d para bloques regulares. Es configurado como x si es necesario forzar el llamado de la función computacional durante la fase de simulación, si esta no contribuye con el cómputo del resultado.
- firing: Vector de las condiciones iniciales del tiempo disparo de activación de eventos, con tamaño igual al número de puertos.
- dep ut: vector booleano de dos componentes [timedep udep]:
    1. timedep boolean: Verdadero si el bloque siempre está activo.
    2. udep boolean: Verdadero si el bloque tiene conexión directa entre la menor de sus salidas que dependa directamente de una de sus entradas (valido con flag=1). Es decir cuando el valor de una entrada se usa para computar una salida.

- label: Una cadena de caracteres. Es usado para identificar el bloque acensándole y modificando sus parámetros.
- nzcross: Número de cruces de la señal por cero que pueden surgir.
- nmode: Número de modos, dado por el tamaño del vector *mode.*
- equations: Usado en caso de bloques implícitos.
- **gui**: El nombre de la función Scilab GUI asociada con el bloque.
- **doc:** Usada para la documentación del bloque.

B.3. FUNCIÓN COMPUTACIONAL

La función computacional es llamada de varias formas por la aplicación. La forma en que el bloque es llamado (secuencia de llamado) se caracteriza por el tipo de la función. Es muy recomendado que los tipos de función computacional 4 y 5 sean usados. El tipo 4 es usado por programas escritos en C y el 5 en lenguaje de Scilab.

En el tipo 4 las funciones reciben dos argumentos: una estructura que contiene la información del bloque y una bandera (*flag*). Por ejemplo:

```
#include "scicos_block.h"
#include <math.h>
void Mi bloque (scicos block *block,int flag)
{
…//Instrucciones del bloque//…
}
```

B.3.1. Banderas

Las banderas indican el trabajo que la función computacional puede realizar. Normalmente consisten en la actualización de algunos campos de la estructura del bloque. Estas son Inicialización, Actualización de salidas, Actualización de estado, Llamada de integración, Modo y crucé por cero, Evento programado y Finalización:

Inicialización: Se activa cuando la bandera *flag*=4, Los eventos continuos y discretos son inicializados o reinicializados. Las salidas del bloque también son inicializadas. También puede ser usado cuando se necesita almacenamiento dinámico de memoria dando la ubicación del archivo bajo esta bandera. Cada bloque es llamado sólo al inicio de la ejecución.

Actualización de salidas: *flag*=1. La ejecución solicita las salidas de los bloques. La función computacional puede necesitar información de la estructura del bloque (entradas, estados, etc) para procesar el valor de las salidas y colocar el resultado en la dirección dada por la estructura del bloque.

Actualización de estado: *flag* =2. Mientras que un evento haya activado el bloque, o exista un cruce por cero, la ejecución permite actualizar (en caso de ser debido) los valores de las variables de tiempo continúo y tiempo discreto, a la vez que indica en que dirección la variable medida dio el cruce por cero.

Llamada de integración: flag=0. Durante la integración, permite llamar la función que procesa los valores de la derivada de la variable de tiempo continuo y enviar este valor a una dirección indicada en la estructura del bloque.

Modo y cruce por cero: flag= 9. Computa los valores de las variables que han sido afectadas por eventos de estados de tiempo continuo, que pueden afectar el funcionamiento del sistema, cambiando su modo de operación. Un ejemplo de ello es un bloque de valor absoluto, donde el bloque tiene dos modos, uno en que la salida es igual a la entrada, y otro en que la salida es igual a la entrada negada, cuando la variable cruza por cero el modo de operación del bloque debe ser cambiado dependiendo de la dirección del cruce.

Evento programado: flag =3. Permite la programación de retrasos en la tabla de eventos.

Finalización: flag=5. Sólo se produce una vez durante el final de la ejecución, y es usado para finalizar los valores de las diferentes variables y cerrar los archivos que fueron abiertos al principio de la ejecución.

B.3.2. Funciones definidas por Scicos

La función computacional recibe otras entradas del bloque de estructura a través de las siguientes funciones:

double get scicos time(): Retorna el valor actual del tiempo t.

int get phase simulation(): Retorna la fase de ejecución (1 o 2).

int get block number(): Retorna el numero del bloque en la estructura %cpr que contiene toda la información necesaria que necesita el simulador.

void set block error(int): Es usado por el bloque para señalar un error en la ejecución.

void do cold restart(): Es usado para forzar un reinicio de los valores numéricos, es usado automáticamente por Scicos.

void set pointer xproperty(int* pointer): Es usado solo internamente por bloques implícitos para diferenciar los estados algebraicos de los diferenciales.

void *scicos malloc(size t): Es usado para almacenar memoria en el espacio de trabajo si es necesario.

void scicos free(void *p): Usado para almacenamiento gratis de memoria.

B.3.3. Estructura del bloque

La estructura del bloque es codificada en C y define:

int nevprt: Código binario de activación de entradas, toma el valor de -1 si son internamente activadas.

voidg funpt: Puntero de la función computacional.

int type: Tipo de la función interface (en este caso tiene el valor de 4)

int nz: Tamaño del estado de tiempo discreto.

double *z: Vector de tamaño nz de tiempo discreto

int nx: Tamaño del estado de tiempo continuo.

double *x: Vector de tamaño nx de tiempo continuo.

double *xd: Vector de tamaño nx correspondiente a la derivada del estado de tiempo continuo.

double *res: Es un vector de tamaño nx usado internamente por bloques implícitos.

int nin: Número de entradas.

int *insz: Tamaño de las entradas.

double **inptr: Tabla de punteros de entradas.

int nout: Número de salidas.

int *outsz: Tamaño de las salidas.

double **outptr: Tabla de punteros de salidas.

int nevout: Número de salidas del Puerto de activación de eventos.

double *evout: Retardo de tiempo de la activación de las salidas.

int nrpar: Número de los parámetros reales.

double *rpar: Vector de parámetros reales de tamaño nrpar.

int nipar: Número de parámetros enteros.

int *ipar: Vector de parámetros enteros de tamaño nipar.

int ng: Número de cruces por cero que pueden aparecer.

double *g: Aparición de cruce por cero.

int ztyp: Boleano que es solo positivo si el bloque puede tener cruces por cero.

int *jroot: Vector de tamaño ng indicando la presencia y la dirección de un cruce por cero.

char *label: Etiqueta del bloque.

void **work: Puntero del espacio de trabajo si el bloque ubica su localización.

int nmode: Número de modos.

int *mode: Vector modo de tamaño nmode.

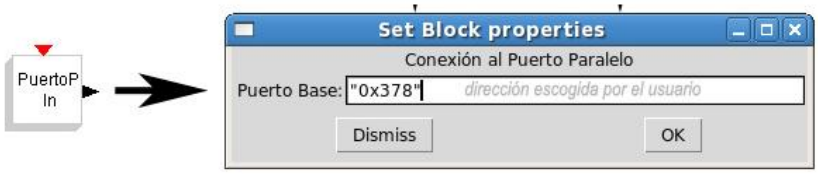
B.4. BLOQUES CREADOS PARA EL SISTEMA DE PROTOTIPADO RÁPIDO DE CONTROL PARA EL MÓDULO MIC955 DE LA EMRPESA FEEDBACK

En el sistema RCP para el módulo MIC955, primero se crearon dos bloques, los cuales permiten la entrada y salida de datos a través del puerto paralelo, con el fin de hacer pruebas experimentales que permitieran verificar el estado físico de cada uno de los pines necesarios, así como garantizar el comportamiento bidireccional de la configuración escogida. Seguidamente se crearon los bloques de comunicación con el MIC 955, tanto de entrada como de salida de datos, de acuerdo a las especificaciones exigidas por su protocolo de comunicación. Y finalmente se creó un bloque que permite enviar un señal adecuada para ejercer el esfuerzo de control. Estos bloques son: *PuertoPIn*, *PuertoPOut*, *Mic955Sensor*, *Mic955Actuador* y *PWM* .

B.4.1. *PuertoPIn*, Bloque de Entrada de datos por el Puerto Paralelo

Su función es la de leer datos escritos en cualquiera de los registros del Puerto paralelo del PC, este lee sólo el registro configurado por el usuario al indicar la dirección de puerto exacta del sistema, siendo 0x378h para Datos, 0x379h para Estado y 0x37Ah para Control. Sin embargo el bloque puede leer otros puertos establecidos del sistema. En la figura 48 se muestra el bloque y su interfaz de comunicación. La dirección debe escribirse en hexadecimal y aunque por defecto es mostrada entre comilla dobles (“) no deben ser usadas al ingresar el dato.

Figura 48: Bloque PuertoPIn y parámetros



El bloque tiene una sola salida que envía un número entero entre 0 y 255 (en decimal) para el caso del Puerto Paralelo. Esta depende del número de bits manejado por el registro configurado. En la Tabla B.1 están escritos los códigos necesarios para el montaje del bloque.

Tabla 1: Función de Interfaz y computacional del bloque PuertoPIn

puertoin.sci

```
function [x,y,typ] = puertoin(job,arg1,arg2)
  x=[];y=[];typ=[];
  select job
  case 'plot' then
    exprs=arg1.graphics.exprs;
    standard_draw(arg1)
  case 'getinputs' then
    [x,y,typ]=standard_inputs(arg1)
  case 'getoutputs' then
    [x,y,typ]=standard_outputs(arg1)
  case 'getorigin' then
    [x,y]=standard_origin(arg1)
  case 'set' then
    x=arg1
    model=arg1.model;graphics=arg1.graphics;
    exprs=graphics.exprs;
    while %t do
      [ok,dp,exprs]=..
      getvalue('Conexión al Puerto Paralelo',..
      ['Puerto Base:'],..
      list('str',-1),exprs)
      if ~ok then break,end
      in=[],
      if exists('outport') then out=ones(outport,1), else out=1, end
      [model,graphics,ok]=check_io(model,graphics,in,out,1,[])
      if ok then
        graphics.exprs=exprs;
        model.rpar=[];
        model.ipar=[hex2dec(strsubst(dp,'0x',''))];
        model.dstate=[1];
        x.graphics=graphics;x.model=model
        break
      end
    end
  case 'define' then
    dp='0x378'
    model=scicos_model()
    model.sim=list('rt_puertoin',4)
    model.in=[]
    if exists('outport') then model.out=ones(outport,1), else model.out=1, end
    model.evtin=1
    model.rpar=[]
    model.ipar=[hex2dec(strsubst(dp,'0x',''))]
    model.dstate=[1];
    model.blocktype='d'
    model.dep_ut=[%t %f]
    exprs=[sci2exp(dp)]
    gr_i=['xstringb(orig(1),orig(2),["PuertoP" ; "In"],sz(1),sz(2),"fill");']
    x=standard_define([3 2],model,exprs,gr_i)
  end
endfunction
```

rtai_puertoin.c

```
/*
implementación del puerto paralelo por RTAI-lab para el modulo MIC 955
de FEEDBACK
*/

#include <machine.h>
#include <scicos_block4.h>
#include <stdio.h>
#include <sys/io.h>

static void init(scicos_block *block)
{
  double * y;
  y = block->outptr[0];
  y[0]=0.0;
  int par=block->ipar[0];
}

static void inout(scicos_block *block)
{
  double * y;
  int par = block->ipar[0];
  y = block->outptr[0];
  y[0] = inb(par);
}

static void end(scicos_block *block)
{
  double * y;
  y = block->outptr[0];
  y[0]=0.0;
  int par=block->ipar[0];
}

void rt_puertoin(scicos_block *block,int flag)
{
  if (flag==1){          /* set output */
    inout(block);
  }
  else if (flag==5){     /* termination */
    end(block);
  }
  else if (flag ==4){    /* initialisation */
    init(block);
  }
}
```

B.4.1.1 Explicación detallada del ciclo While utilizado:

En *case 'set'* se utiliza un ciclo While para realizar la comunicación con el usuario por medio de la caja de dialogo que se muestra en la figura 48. A continuación se explica con detalle algunas de sus instrucciones contenidas para entender mejor su funcionamiento:

- **while %t do**:Abre el ciclo por tiempo indefinido.
- **getvalue**: Función encargada de desplegar la caja de dialogo donde:

1. [ok,v1, v2,…vn,exprs]: Variables de salida, donde v1, v2,…vn son introducidas por el usuario. ok y exprs son variables utilizadas por Scilab y se ubican al principio y al final de la sentencia.
2. ('Titulo',['et1';'et2';…'etn'],list('vec',-1,'vec',-1,…'str',-1),exprs**)**: Muestra el título de la caja de dialogo, define las etiquetas y el tipo de valores entregados a las variables, siendo 'vec' valores numéricos y 'str' cadenas de caracteres.

B.4.2. PuertoPOut, Bloque de Salida de datos por el Puerto Paralelo

Su función es la de escribir datos en cualquiera de los registros del Puerto paralelo del PC, este escribe sólo en el registro configurado por el usuario al indicar la dirección de puerto exacta del sistema, siendo 0x378h para Datos, 0x379h para Estado y 0x37Ah para Control. Sin embargo el bloque puede escribir en otros puertos establecidos del sistema. En la figura 49 se muestra el bloque y su interfaz de comunicación. La dirección debe escribirse en hexadecimal y aunque por defecto se muestra entre comilla dobles (“), estas no deben ser usadas al ingresar la dirección.

Figura 49: Bloque PuertoPOut y parámetros

El bloque tiene una sola entrada y recibe un número entero entre 0 y 255 (en decimal) para el caso del Puerto Paralelo. En la Tabla 2 están escritos los códigos necesarios para el montaje del bloque.

Tabla 2: Función de Interfaz y computacional del bloque PuertoPOut

puertoout.sci

```
function [x,y,typ] = puertoout(job,arg1,arg2)
  x=[];y=[];typ=[];
  select job
  case 'plot' then
    exprs=arg1.graphics.exprs;
    standard_draw(arg1)
  case 'getinputs' then
    [x,y,typ]=standard_inputs(arg1)
  case 'getoutputs' then
    [x,y,typ]=standard_outputs(arg1)
  case 'getorigin' then
    [x,y]=standard_origin(arg1)
  case 'set' then
    x=arg1
    model=arg1.model;graphics=arg1.graphics;
    exprs=graphics.exprs;
    while %t do
      [ok,dp,exprs]=..
      getvalue('Conexión al Puerto Paralelo',..
      ['Puerto Base:'],..
      list('str',-1),exprs)
      if ~ok then break,end
      if exists('inport') then in=ones(inport,1), else in=1, end
      out=[]
      [model,graphics,ok]=check_io(model,graphics,in,out,1,[])
      if ok then
        graphics.exprs=exprs;
        model.rpar=[];
        model.ipar=[hex2dec(strsubst(dp,'0x',''))];
        model.dstate=[1];
        x.graphics=graphics;x.model=model
        break
      end
    end
  case 'define' then
    dp='0x378'
    model=scicos_model()
    model.sim=list('rt_puertoout',4)
    if exists('inport') then model.in=ones(inport,1), else model.in=1, end
    model.out=[]
    model.evtin=1
    model.rpar=[]
    model.ipar=[hex2dec(strsubst(dp,'0x',''))]
    model.dstate=[1];
    model.blocktype='d'
    model.dep_ut=[%t %f]
    exprs=[sci2exp(dp)]
    gr_i=['xstringb(orig(1),orig(2),["PuertoP" ; "Out"],sz(1),sz(2),"fill");']
    x=standard_define([3 2],model,exprs,gr_i)
  end
endfunction
```

rtai_puertoout.c

```
/*
implementaciÃ³n del puerto paralelo por RTAI-lab para el modulos MIC 955
de FEEDBACK
*/

#include <machine.h>
#include <scicos_block4.h>
#include <stdio.h>
#include <sys/io.h>

static void init(scicos_block *block)
{
  outb(0x00, block->ipar[0]);
}

static void inout(scicos_block *block)
{
  int p=0;
  double *u;
  int ntraces=block->nin;
  struct {
    float t;
    float u[ntraces];
  } data;
  int i;

  data.t=(float) get_scicos_time();
  for (i = 0; i < ntraces; i++) {
    u=block->inptr[i];
    data.u[i] = (float) u[0];
  }
  p=data.u[0];
  int par=block->ipar[0];
  if (p < 255){
    if(p < 0){
                outb(0x00,(par));
    }
    else{
                outb(p,(par));
    }
  }
  else{
    outb(255,(par));
  }
}

static void end(scicos_block *block)
{
  outb(0x00, block->ipar[0]);
}


void rt_puertoout(scicos_block *block,int flag)
{
  if (flag==1){          /* set output */
    inout(block);
  }
  else if (flag==5){     /* termination */
    end(block);
  }
  else if (flag ==4){    /* initialisation */
    init(block);
  }
}
```

B.4.3. Mic955 Sensor.

Se encarga de la comunicación con el ADC ZN427 del MIC955. Primero envía la señal de activación de conversión a través del pin 16 del Puerto Paralelo y posteriormente recibe los datos enviados por el conversor ADC al registro de datos configurado para lectura. En la figura 50 se muestra el bloque y su interfaz de comunicación. Donde debe ser configurada la dirección base del puerto a trabajar, en este caso 0x378h.

Figura 50: Bloque Mic955 Sensor y parámetros

El bloque tiene una sola salida que envía un valor entre 0 y 255 equivalente a la temperatura convertida por el ADC. En la Tabla 3 están escritos los códigos necesarios para el montaje del bloque.

Tabla 3: Función de Interfase y computacional del bloque Mic955 sensor

mic_sensor.sci

```
function [x,y,typ] = mic_sensor(job,arg1,arg2)
  x=[];y=[];typ=[];
  select job
  case 'plot' then
    exprs=arg1.graphics.exprs;
    pb=exprs(1)
    standard_draw(arg1)
  case 'getinputs' then
    [x,y,typ]=standard_inputs(arg1)
  case 'getoutputs' then
    [x,y,typ]=standard_outputs(arg1)
  case 'getorigin' then
    [x,y]=standard_origin(arg1)
  case 'set' then
    x=arg1
    model=arg1.model;graphics=arg1.graphics;
    exprs=graphics.exprs;
    while %t do
      [ok,pb,exprs]=..
      getvalue('ConexiÃ³n al Puerto Paralelo',..
      ['Puerto Base:'],..
      list('str',-1),exprs)
      if ~ok then break,end
      in=[],
      if exists('outport') then out=ones(inport,1), else out=1, end
      [model,graphics,ok]=check_io(model,graphics,in,out,1,[])
      if ok then
        graphics.exprs=exprs;
        model.rpar=[];
                model.ipar=[hex2dec(strsubst(pb,'0x',''))];
        model.dstate=[1];
        x.graphics=graphics;x.model=model
        break
      end
    end
  case 'define' then
    pb='0x378'
    model=scicos_model()
    model.sim=list('mrt_sensor',4)
    model.in=[]
    if exists('outport') then model.out=ones(outport,1), else model.out=1, end
    model.evtin=1
    model.rpar=[]
    model.ipar=[hex2dec(strsubst(pb,'0x',''))]
    model.dstate=[1];
    model.blocktype='d'
    model.dep_ut=[%t %f]
    exprs=[sci2exp(pb)]
    gr_i=['xstringb(orig(1),orig(2),["Mic955 Sensor" ; pb ],sz(1),sz(2),"fill");']
    x=standard_define([3 2],model,exprs,gr_i)
  end
endfunction
```

mic_sensor.c

```
/*
implementación del sensor por RTAI-lab para el modulos MIC 955 de
FEEDBACK
*/
#include <machine.h>
#include <scicos_block4.h>
#include <stdio.h>
#include <sys/io.h>
#include <stdio.h>
#include <stdlib.h>

static void init(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara al principio de la Tarea en
tiempo real
  double * y;                    //Establece la variable de entrada del bloque
  y = block->outptr[0];
  y[0]=0.0;
  int pb=block->ipar[0];
  int s0 = 219;          //asegura que la configuracion del puerto sera
modificada para nuestras necesidades
  int s1=4;              //Coloca en alto el bit de conversion
  int s2=32;                     //configura el puerto paralelo como entrada de
datos
  int s3=inb(pb+2);              //Leemos el estado actual del puerto
  int st=0;
  s3= s3 & s0;
  st = s3+s2+s1;                 //Modifica la palabra para configurar el puerto
  outb(st,pb+2);         //Escribi en el puerto
}

static void inout(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara con cada paso del reloj  int
st=0;
  int s0 = 251;          //asegura que solo modificarÃ¡ el bit 16
  int s1=0;
  int s3=0;
  int sr=0;
  double * y;            //Variable de Salida del Bloque
  y= block->outptr[0];   //relaciona la variable con el bloque
  int pb=block->ipar[0]; //Toma el valor del puerto
  s3= inb(pb+2);         //Mira la lectura actual del puerto en control
  sr= s3&s0;                     //Modifica la palabra de control para que se
envie el bit de activaciÃ³n
  s1=0;
  st= sr+s1;
  outb(st,pb+2);         //Envia un cero al pin 16
  s1=4;
  st= sr+s1;
  outb(st,pb+2);         //Envia un uno al pin 16
  usleep(50);                    //Espera la conversion del ZN427
  y[0]= inb(pb);         //Lee el puerto y lo envia a la salida del bloque
}

static void end(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara al final de la Tarea en
tiempo real
  double * y;
```

	y = block->outptr[0];
	y[0]=0.0;int pb=block->ipar[0];
	int s0 = 219; //asegura que la configuracion del puerto sera modificada para nuestras necesidades
	int s1=0; //Coloca en bajo el bit de conversion (*opcional)
	int s2=0; //Configura el puerto paralelo como salida de datos (*opcional)
	int s3=inb(pb+2); //Mira la lectura actual del puerto en control
	int st=0;
	s3= s3 & s0;
	st = s3+s2+s1;
	outb(st,pb+2); //Establece la configuracion normal del puerto
	}
	
	void mrt_sensor(scicos_block *block,int flag)
	{
	if (flag==1){ /* set output */
	inout(block);
	}
	else if (flag==5){ /* termination */
	end(block);
	}
	else if (flag ==4){ /* initialization */
	init(block);
	}
	}

B.4.4. Mic955 actuador

Se encarga de la comunicación con los actuadores físicos del módulo MIC955 a través del registro de control del puerto paralelo, por medio de los pines 14 (para el ventilador) y 17 (para el calentador). En la figura 51 se muestra el bloque y su interfaz de comunicación. Donde debe ser configurada la dirección base del puerto a trabajar, en este caso 0x378h.

Figura 51: Bloque Mic955 Actuador y parámetros

El bloque tiene dos entradas las cuales pueden variar entre 0 y 1, cualquier valor por encima de 0.5 será tomado como 1 y cualquier valor por debajo será tomado como 0. La primera entrada activa el calentador en 1 y lo desactiva en 0, la segunda entrada cumple la misma función con el ventilador del MIC955. En la Tabla B.4 están escritos los códigos necesarios para el montaje del bloque.

Tabla 4: Función de Interfase y computacional del bloque Mic955 actuador

mic_actuador.sci	mic_actuador.c

unction [x,y,typ] = mic_actuador(job,arg1,arg2)
x=[];y=[];typ=[];
select job
case 'plot' then
exprs=arg1.graphics.exprs;
pb=exprs(1)
standard_draw(arg1)
case 'getinputs' then
[x,y,typ]=standard_inputs(arg1)
case 'getoutputs' then
	/*
	implementación del actuador por RTAI-lab para el modulos MIC 955 de FEEDBACK
	*/
	
	#include <machine.h>
	#include <scicos_block4.h>
	#include <stdio.h>
	#include <sys/io.h>
	#include <stdio.h>

```
    [x,y,typ]=standard_outputs(arg1)
  case 'getorigin' then
    [x,y]=standard_origin(arg1)
  case 'set' then
    x=arg1
    model=arg1.model;graphics=arg1.graphics;
    exprs=graphics.exprs;
    while %t do
      [ok,pb,exprs]=..
      getvalue('ConexiÃ³n al Puerto',..
      ['Puerto Base:'],..
      list('str',-1),exprs)
      if ~ok then break,end
      if exists('inport') then in=ones(inport,1), else in=1, end
      out=[]
      [model,graphics,ok]=check_io(model,graphics,in,out,1,[])
      if ok then
        graphics.exprs=exprs;
        model.rpar=[];
                  model.ipar=[hex2dec(strsubst(pb,'0x',''))];
        model.dstate=[1];
        x.graphics=graphics;x.model=model
        break
      end
    end
  case 'define' then
    inport=2
    pb='0x378'
    model=scicos_model()
    model.sim=list('mrt_actuador',4)
    if exists('inport') then model.in=ones(inport,1), else model.in=1, end
    model.out=[]
    model.evtin=1
    model.rpar=[]
    model.ipar=[hex2dec(strsubst(pb,'0x',''))]
    model.dstate=[1];
    model.blocktype='d'
    model.dep_ut=[%t %f]
    exprs=[sci2exp(pb)]
    gr_i=['xstringb(orig(1),orig(2),["Mic955 Actuador" ; pb ],sz(1),sz(2),"fill");']
    x=standard_define([3 2],model,exprs,gr_i)
  end
endfunction
```

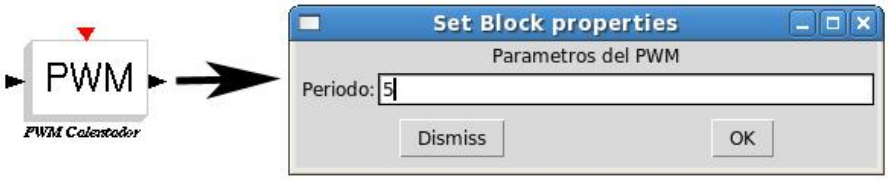
```
#include <stdlib.h>

static void init(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara al principio de la Tarea en
tiempo real
  int pb=block->ipar[0];        //Asigna el puerto base
  int s0 = 245;                 //Asegura que la configuracion del puerto de
control no sera modificada
  int s1=8;                     //Asegura que el ventilador y el calentador no
se encenderan en un principio
  int s2=2;                     //colocando sus bits en 1 debido a su logica
negativa
  int s3=inb(pb+2);             //Lee el estado actual del puerto
  int st=0;
  s3= s3 & s0;
  st = s3 +s2 +s1;
  outb(st,pb+2);                //Escribe la configuracion del puerto
}

static void inout(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara con cada paso del reloj de
la Tarea en tiempo real
  int pb=block->ipar[0];  //Asigna el puerto base a pb
  int temp=0;                   //Variable de temperatura
  int vent=0;           //Variable de ventilaciÃ³n
  int s0 = 245;          //Esta variable simboliza los pines que queremos
controlar con ceros 11110101
  int s1=0;
  int s2=0;
  int s3=0;
  int st=0;
  double *u;                    //Variables de salida del bloque
  int ntraces=block->nin; //Lee el nÃºmero de entradas del bloque
  struct {                      //Establece un arreglo para las entradas del
bloque
    float t;
    float u[ntraces];
  } data;

  int i;                        //Toma los valores de entrada del bloque
  data.t=(float) get_scicos_time();  //Tiempo
  for (i = 0; i < ntraces; i++) {    //Temperatura y Ventilador
    u=block->inptr[i];
    data.u[i] = (float) u[0];
  }
  temp=data.u[0];               //Asigna a temp la entrada de temperatura
(Valor normalizado de 0 a 1)
  if (temp >= 1) s1=0;    //para el pin 17 alto es 0
  else s1=8;
  vent=data.u[1];               //Asigna a vent la entrada de ventilador (Valor
normalizado de 0 a 1)
  if (vent >=1)s2=0;      //para el pin 14 alto es 0
  else s2=2;
  s3= inb(pb+2);                //lee el estado actual del puerto
  s3= s3 & s0;          //coloca en cero los bits a controlar para un facil
manejo matematico
  st= s1+s2+s3;
  outb(st,pb+2);                //escribe en el puerto
}

static void end(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara al final de la Tarea en
tiempo real
  int pb=block->ipar[0];  //Asigna el puerto base
  int s0 = 245;                 //Asegura que la configuracion del puerto de
control no sera modificada
  int s1=8;                     //Asegura que el ventilador y el calentador no
se encenderan en al final
  int s2=2;
  int s3=inb(pb+2);       //Lee el estado actual del puerto
  int st=0;
  s3= s3 & s0;                                  //Asegura no modificar la
configuracion del puerto
  st = s3 +s2 +s1;
  outb(st,pb+2);          //Escribe el estado final de los bit del puerto
}



void mrt_actuador(scicos_block *block,int flag)
{
  if (flag==1){         /* set output */
    inout(block);
  }
  else if (flag==5){    /* termination */
    end(block);
```

	}
	else if (flag ==4){ /* initialization */
	init(block);
	}
	}

B.4.5. Bloque PWM

Este bloque generar una señal de control para los actuadores a través de modulación por ancho de pulsos. En la figura 52 se muestra el bloque y su interfaz de comunicación. Donde debe ser configurado el periodo de PWM en segundos.

Figura 52: Bloque PWM y parámetros



El bloque tiene una entrada la cual puede variar de 0 a 1, cualquier valor mayor a 1 será tomado como 1 y cualquier valor menor a 0 será tomado como 0. Su salida es un tren de pulsos con el periodo igual al configurado por el usuario y donde el tiempo en alto de la señal dependerá del valor de la entrada (1 = 100% y 0 =0% de tiempo en alto. La resolución de la señal esta definida por la formula: (resolución= PeriodoPWM/Temporización). La temporización es definida por el reloj de eventos de Scicos que se conecte al bloque. En la Tabla 5 están escritos los códigos necesarios para el montaje del bloque.

Tabla 5: Función de Interfase y computacional del bloque PWM

mic_pwm.sci	mic_pwm.c

function [x,y,typ] = mic_pwm(job,arg1,arg2)
x=[];y=[];typ=[];
select job
case 'plot' then
exprs=arg1.graphics.exprs;
standard_draw(arg1)
case 'getinputs' then
[x,y,typ]=standard_inputs(arg1)
case 'getoutputs' then
[x,y,typ]=standard_outputs(arg1)
case 'getorigin' then
[x,y]=standard_origin(arg1)
case 'set' then
x=arg1
model=arg1.model;graphics=arg1.graphics;
exprs=graphics.exprs;
while %t do
[ok,periodo,exprs]=..
getvalue('Parametros del PWM',..
['Periodo:'],..
list('vec',-1),exprs)
if ~ok then break,end
if exists('inport') then in=ones(inport,1), else in=1, end
if exists('outport') then out=ones(outport,1), else out=1, end
[model,graphics,ok]=check_io(model,graphics,in,out,1,[])
if ok then
graphics.exprs=exprs;
model.rpar=[periodo];
model.ipar=[];
model.dstate=[1];
	/*
	implementación del pwm por RTAI-lab para el modulos MIC 955 de FEEDBACK
	*/
	
	#include <machine.h>
	#include <scicos_block4.h>
	#include <stdio.h>
	#include <sys/io.h>
	
	static void init(scicos_block *block)
	{
	// aqui se establece el codigo que se ejecutara al principio de la Tarea en tiempo real
	double *y = GetRealOutPortPtrs(block,1);
	y[0] = 0.0;
	//Colocamos la salida del bloque en cero
	}
	
	static void inout(scicos_block *block)
	{
	// aqui se establece el codigo que se ejecutara con cada paso del reloj de la Tarea en tiempo real
	double p; //Periodo
	double v;
	//Porcentaje del periodo
	double * rpar = GetRparPtrs(block); //Asigna los parametros reales del bloque
	double *y = GetRealOutPortPtrs(block,1); //Asigna a y las entradas del bloque

```
      x.graphics=graphics;x.model=model
      break
    end
  end
  case 'define' then
   periodo=1
   model=scicos_model()
   model.sim=list('mrt_pwm',4)
   if exists('inport') then model.in=ones(inport,1), else model.in=1, end
   if exists('outport') then model.out=ones(outport,1), else model.out=1, end
   model.evtin=1
   model.rpar=[periodo]
   model.ipar=[]
   model.dstate=[1]
   model.blocktype='d'
   model.dep_ut=[%t %f]
   exprs=[sci2exp(periodo)]
   gr_i=['xstringb(orig(1),orig(2),["PWM"],sz(1),sz(2),"fill");']
   x=standard_define([3 2],model,exprs,gr_i)
  end
endfunction
```

```
  double *u = block->inptr[0];                                    //Asigna
a u las salidas del bloque
  double t = get_scicos_time();                                   //Asigna
a t el tiempo real de la tarea

  if(u[0]<0) p=0.0;
            //Establece limites de validaciÃ³n
  else {
    if(u[0]>1) p=1.0;
    else       p=u[0];                                            //p toma
un valor porcentual
  }
  v=t/rpar[0];
  v=(v - (int) v);                                                //v
establece la posiciÃ³n porcentual del tiempo en cada periodo
  if(v > p) y[0] = 0.0;
              //comparamos y asignamos el valor de la salida
  else      y[0] = 1.0;
}

static void end(scicos_block *block)
{
  // aqui se establece el codigo que se ejecutara al final de la Tarea en
tiempo real
  double *y = GetRealOutPortPtrs(block,1);
  y[0] = 0.0;
            //Colocamos la salida del bloque en cero
}



void mrt_pwm(scicos_block *block,int flag)
{
  if (flag==1){          /* set output */
    inout(block);
  }
  else if (flag==5){     /* termination */
    end(block);
  }
  else if (flag ==4){    /* initialisation */
    init(block);
  }
}
```

B.5. ADICIÓN DE LOS NUEVOS BLOQUES A SCICOS

Para compilar los nuevos bloques las funciones de interfase y computacional deben ser ubicadas en los correspondientes directorios de RTAI, estos son:

- Funcion de interfase: /usr/src/rtai/rtai-lab/macros/device

- Funcion Computacional: /usr/src/usr/src/rtai/rtai-lab/macros/

Una vez ubicados se procede a recompilar el RTAI-lab ubicándose en el directorio: /usr/src/usr/src/rtai/rtai-lab/macros/ y ejecutando el comando make install desde un terminal.

Para adicionar los bloques a Scicos se ingresa al menú: Edit – Add new Block y en el cuadro de dialogo se digita el nombre de la función creada, por ejemplo mic_pwm. Estos bloques pueden ser guardados en forma de una paleta dirigiéndose al menú: Pallete- Save As Pallete. Si abrimos Scilab desde la ubicación de la paleta esta se cargará automáticamente si no puede ser llamada nuevamente desde el menú: Pallete- Load As Pallete.

ANEXO D: LINEALIZACIÓN, FILTRADO Y NORMALIZACIÓN DEL TRANSMISOR DE TEMPERATURA Y EXPERIMENTOS ADJUNTOS

En este anexo se describe la serie de experimentos que se realizaron para la creación de los diferentes bloques de Scicos, integrados en el transmisor diseñado para el módulo MIC 955 de la empresa Feedback: linealización, filtrado y normalización, así como la asignación de sus parámetros.

D.1. LINEALIZACIÓN:

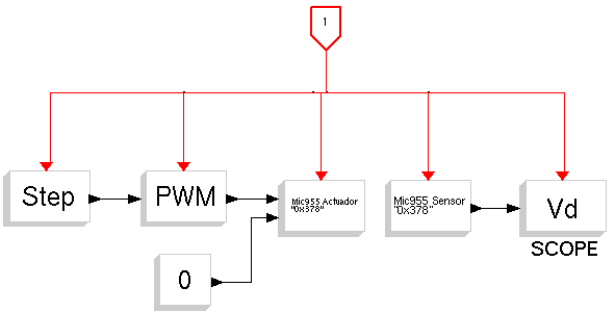
La etapa de linealización en el transmisor se realizo mediante el uso del bloque de Scilab *Mathematical Expression,* por medio del cual se implementó un polinomio que permite convertir el valor entregado por el ADC a unidades de ingeniería (ºC).

Para lograr este objetivo se realizo el siguiente experimento:

D.1.1. Toma de Datos

Para realizar la toma de datos de la planta de temperatura se implementó el diagrama de bloques que se muestra en la figura 53. El *Step* tiene la función de enviar un valor de 0 a 1, equivalente al cambio de 0 al 100% de energía entregada al calentador del MIC955. El bloque *PWM* se encarga de entregar un tren de pulsos que permita realizar el control propuesto. El bloque *Mic955 Actuador* envía la señal de control al módulo MIC955. El bloque *Mic955 Sensor* lee la salida del conversor ADC y la envía al PC. El *Scope* permite monitorear la salida del conversor desde el Xrtailab.

Figura 53: Experimento de Linealización



El experimento consistió en los siguientes pasos:

1. Hacer variaciones equivalentes a 0.5% de la energía entregada al actuador, por medio del valor generado por el *Step,* hasta el punto que produjera un cambio en la respuesta del conversor ADC.
2. Medir el valor de temperatura en la planta con un termómetro.
3. Medir el voltaje de entrada del conversor ADC.
4. Repetir los dos pasos anteriores hasta que la energía entregada al actuador de la planta sea del 100%

En la tabla 6 se muestran los valores obtenidos durante la experimentación, en la columna ADC se consiga el valor digital entregado por el conversor, en la columna °C se consiga la temperatura registrada en el momento del cambio del valor digital y en la columna Vol, se registra el voltaje de entrada al conversor en el momento del cambio:

Tabla 6: Datos obtenidos en el experimento de Linealización.

ADC	ºC	Vol.
47	23.5	0.482
51	25.6	0.523
52	26	0.527
53	26.5	0.544
54	26.8	0.549
55	27	0.555
56	27.5	0.56
57	28	0.575
58	28.4	0.587
59	29	0.600
60	29.5	0.610
61	30	0.622
62	30.2	0.629
63	30.9	0.640
64	31.2	0.650
65	31.6	0.664
66	31.9	0.673
67	32.5	0.685
68	33	0.695
69	33.3	0.705
70	33.9	0.713
71	34.3	0.724
72	34.8	0.733
73	35.2	0.744
74	35.6	0.752
75	36	0.765
76	36.4	0.772
77	36.9	0.782
78	37.2	0.792
79	37.6	0.804
80	38.2	0.814
81	38.6	0.822
82	39	0.833
83	39.4	0.844
84	40	0.852
85	40.3	0.862
86	40.7	0.872
87	41.3	0.883

ADC	ºC	Vol.
88	41.7	0.892
89	42	0.902
90	42.5	0.910
91	42.9	0.921
92	43.5	0.932
93	44	0.945
94	44.2	0.951
95	44.5	0.962
96	45	0.972
97	45.3	0.981
98	45.9	0.991
99	46.3	1.000
100	46.5	1.011
101	46.9	1.020
102	47.3	1.030
103	47.9	1.040
104	48.3	1.050
105	48.7	1.059
106	49	1.069
107	49.4	1.080
108	50	1.090
109	50.3	1.099
110	51	1.110
111	51.3	1.120
112	51.7	1.130
113	52	1.140
114	52.3	1.150
115	53	1.160
116	53.3	1.170
117	53.8	1.179
118	54.2	1.190
119	54.8	1.200
120	55	1.209
121	55.8	1.219
122	56	1.229
123	56.4	1.238
124	56.9	1.249
125	57.3	1.258

ADC	ºC	Vol.
126	57.9	1.268
127	58.1	1.278
128	58.8	1.286
129	59.4	1.297
130	60	1.309
131	60.5	1.319
132	61	1.327
133	61.4	1.337
134	61.9	1.348
135	62.2	1.358
136	62.5	1.369
137	63	1.379
138	63.5	1.387
139	64	1.397
140	64.4	1.409
141	65.2	1.419
142	65.5	1.428
143	66.1	1.437
144	66.5	1.448
145	67	1.458
146	67.5	1.467
147	68	1.477
148	68.5	1.488
149	69	1.496
150	69.5	1.508
151	70	1.517
152	70.3	1.526
153	70.8	1.536
154	71.3	1.545
155	71.5	1.506
156	72	1.565
157	72.4	1.575
158	73	1.584
159	73.5	1.595
160	74	1.605
161	74.3	1.615
162	74.9	1.624
163	75.5	1.634
164	76	1.644
165	76.3	1.655
166	77	1.664
167	77.5	1.676
168	77.9	1.684
169	78.5	1.696
170	78.9	1.705

ADC	ºC	Vol.
171	79.3	1.717
172	79.7	1.723
173	80.1	1.733
174	80.5	1.743
175	81.1	1.753
176	81.5	1.763
177	82.1	1.774
178	82.7	1.783
179	83.5	1.794
180	83.7	1.803
181	84.1	1.813
182	84.5	1.822
183	84.8	1.829
184	85.1	1.839
185	85.5	1.847
186	86.1	1.859
187	86.7	1867

Con los anteriores datos se procedió a encontrar una ecuación que permita relacionar los valores digitales generados por el conversor, vistos en la columna ADC de la tabla 6, con los valores de temperatura medidos con el termómetro en el módulo MIC955. Esto se realizó por medio de la técnica de aproximación de mínimos cuadrados.

D.1.2. Aproximación por Mínimos Cuadrados

Con los datos de la tabla 5 se procedió a calcular raíces de polinomios de diferentes grados a partir de la técnica de aproximación de mínimos cuadrados, para luego comparar los resultados de las funciones obtenidas con los datos reales de la planta por medio de observación grafica y por métodos cuantitativos de índices de desempeño.

Los polinomios obtenidos fueron los siguientes:

Primer Orden: $2.0190614 + 0.4491786x$

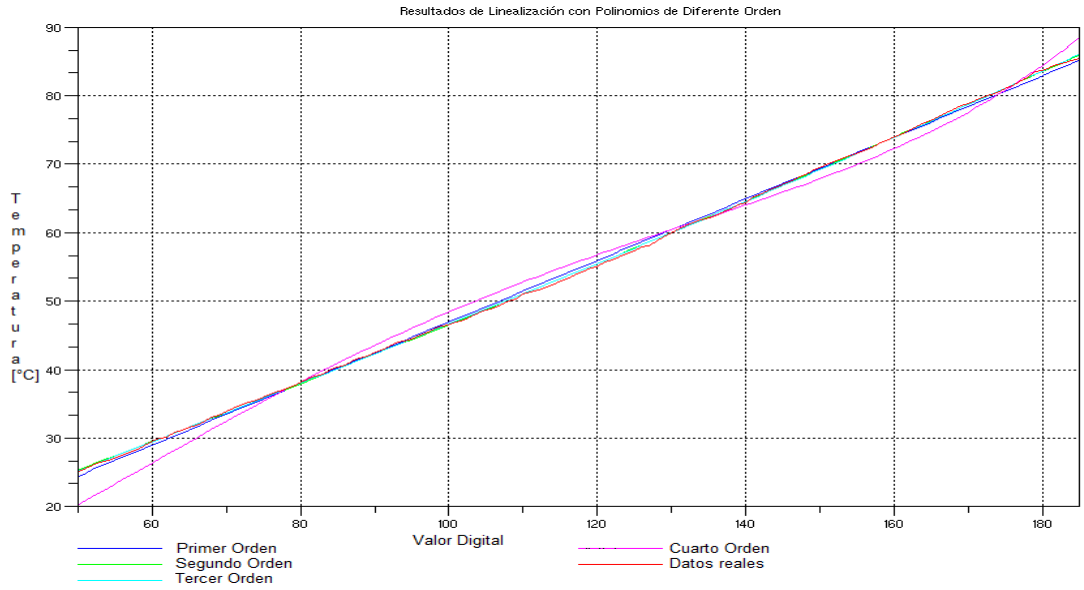
Segundo Orden $5.5994837 + 0.3809689x + 0.0002880x^2$

Tercer Orden: $4.1903175 + 0.4228497x - 0.0000920x^2 + 0.0000011x^3$

Cuarto Orden: $0.0127671x^2 - 0.0001077x^3 + 0.0001077x^4$

El comportamiento de las funciones se muestra en la figura 54, donde los datos reales se grafican en color rojo. La similitud entre los polinomios de primero, segundo, y tercer orden hacen que sea totalmente necesario utilizar la valoración cuantitativa de los índices de desempeño.

Figura 54: Comparación grafica de los funciones



D.1.2. Índices de Desempeño

La Tabla 7 muestra los índices de desempeño IEA, que representa el área diferencial entre la respuesta real y la del modelo, y ISE que da mayor peso a las deviaciones grandes permitiendo medir la variación de los valores reales con respecto a los del modelo, la cual es recomendada para elegir la función en este caso.

Tabla 7: Índices de desempeño de los polinomios de linealización

Orden del Polinomio	IEA	ISE
1	55.087593	30.521731
2	26.295282	7.2312616
3	25.491321	6.8296344
4	208.29677	460.78879

Los valores consignados en la Tabla 7 muestran los resultados de los índices de desempeño, los cuales sugieren un mejor resultado entre menor es el valor del mismo, siendo 0 una respuesta optima. Como resultado de este análisis se escogió el polinomio de tercer orden para realizar la linealización de la planta. Los cálculos fueron evaluados en Scilab mediante el siguiente programa:

```
x= [47…187]';
y= [23.5 … 86.7]';
m = size(x,1);
t = (x(1):0.01:x(m))';

// Polinomio de orden 1
n = 1;
A = zeros(m, n+1);
for i=0:n
A(:,i+1) = x.^i;
end
cf = A\y;
p1 = poly(cf, 'x', 'c');

// Polinomio de orden 2
n = 2;
A = zeros(m, n+1);
for i=0:n
A(:,i+1) = x.^i;
end
cf = A\y;
```

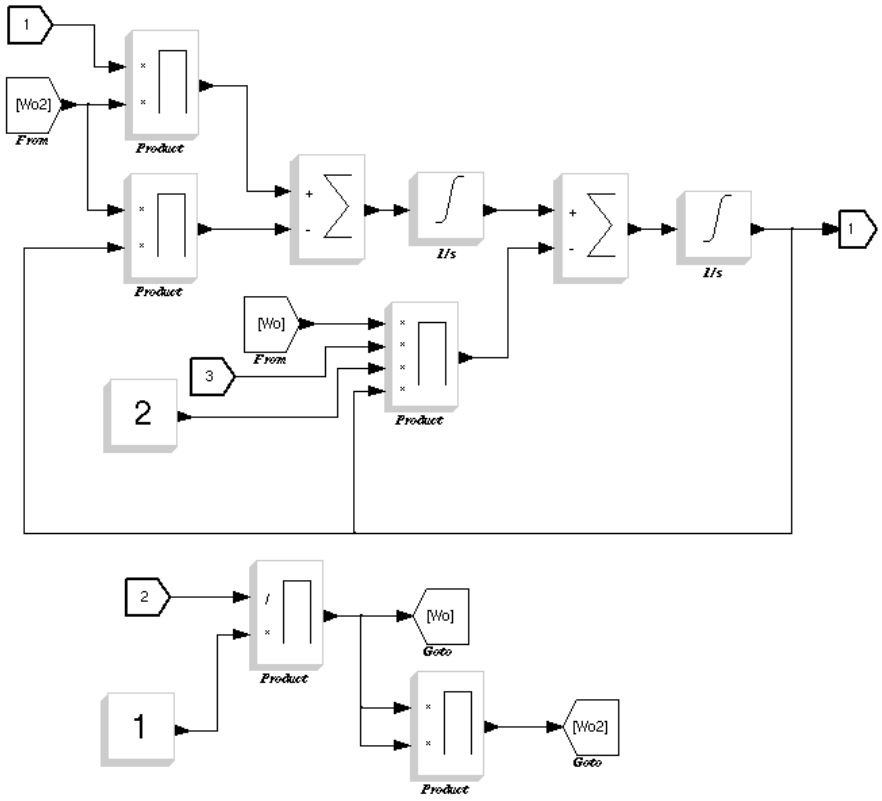
```
p2 = poly(cf, 'x', 'c');

// Polinomio de orden 3
n = 3;
A = zeros(m, n+1);
for i=0:n
A(:,i+1) = x.^i;
end
cf = A\y;
p3 = poly(cf, 'x', 'c');

// Polinomio de orden 4
n = 4;
A = zeros(m, n+1);
for i=0:n
A(:,i+1) = x.^i;
end
cf = A\y;
p4 = poly(cf, 'x', 'c');

// Validación del módelo

y1= horner(p1,x);
y2= horner(p2,x);
y3= horner(p3,x);
y4= horner(p4,x);
// Integral absoluta del error
iea1= norm(y1-y,1);
iea2= norm(y2-y,1);
iea3= norm(y3-y,1);
iea4= norm(y4-y,1);
//Integral cuadrada del error
iec1= sum ((y1-y)^2);
iec2= sum ((y2-y)^2);
iec3= sum ((y3-y)^2);
iec4= sum ((y4-y)^2);

//Graficando
xbasc()
plot2d(x,[y1,y2,y3,y4,y], [2 3 4 6 5] , rect = [50,20,185,90] , leg = " P=1@ P=2@
P=3@ P=4@ Datos" )
xgrid()
xtitle("Resultados de Linealización con Polinomios de Diferente Orden","Valor
digital","Temperatura")
```

D.2 FILTRADO

Éste se realizó mediante el diseño del filtro de segundo orden que se ve en la ecuación 1, a través de la técnica de implementación en variables de estado tal como se puede ver figura 55. :

$$\frac{Y(s)}{U(s)} = \frac{W_0^2}{s^2 + 2W_0\zeta s + W_0^2} \tag{1}$$

Figura 55: Filtro Pasa Bajos de Segundo Orden



El puerto 1 es la señal de entrada del filtro, los puertos 2 y 3 corresponden al Damping, introducido por el usuario en segundos, y al factor de amortiguamiento de la señal, el cual define la forma de la curva de frecuencia del filtro.

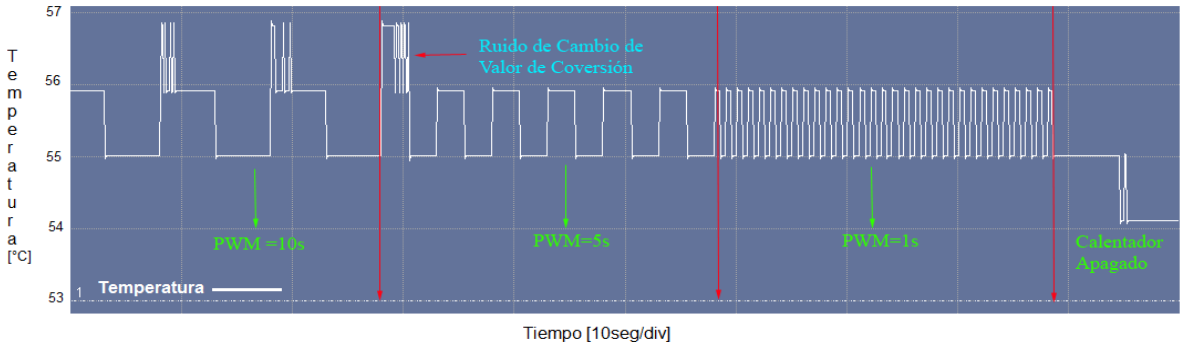
D.2.1. Ruido De PWM Indeseado en la Variable Temperatura

Un fenómeno particular e indeseado del módulo MIC955 del laboratorio de Instrumentación Industrial, se produce cuando se enciende el calentador del módulo MIC955, ya que éste genera un cambio instantáneo indeseado en la lectura del conversor ADC, la cual puede variar entre 1 o 2 valores digitales, equivalentes a aproximadamente 0.5 y 1 grado de temperatura. Este ruido puede

ser medido en la entrada del conversor ADC, siendo su variación de alrededor de 25mv.

En la figura 56 se observa la variable temperatura, sin filtrar, oscilando alrededor de 55°C, esta oscilación es de ±0.5°C. La señal de PWM que genera los 55°C es de 50%, la lenta dinámica de la planta térmica debería filtrar las oscilaciones del calentador y generar una señal de temperatura constante, esto es sin oscilaciones. Sin embargo la figura muestra tres cambios de periodo de PWM, equivalentes a 10, 5, y 1 segundos, y sin embargo se mantiene esta oscilación asociada al periodo de encendido y apagado del calentador. Como puede observarse la temperatura medida cambia instantáneamente con cada cambio de la señal de PWM aplicada al calentador, lo que físicamente no es posible por las condiciones y comportamiento dinámico de la planta.

Figura 56: Variable Temperatura sin filtro



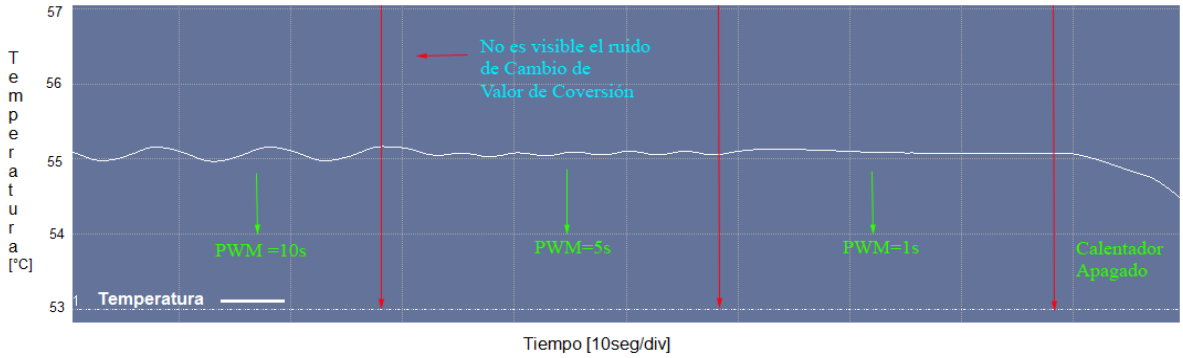
Adicionalmente puede verse como al apagar el calentador, el ruido oscilatorio desaparece instantáneamente y la temperatura empieza a descender. Esto conduce a concluir que al interior del módulo MIC955, se presenta un ruido indeseable, que sigue las variaciones de encendido y apagado del calentador, y que se manifiesta en la señal de voltaje, generada por el puente de Wheatstone DC conectado al sensor, que ingresa al conversor ADC, por lo que la salida digitalizada hereda este mismo ruido.

La Ilustración 57 muestra la señal de temperatura bajo el efecto del filtro de segundo orden cuyos valores son: Damping = 4 segundos y factor de amortiguamiento = 0.707.

Aunque la respuesta de la variable no presenta oscilaciones con el periodo de PWM = 1 segundo, el periodo escogido fue de 5 segundos, esto debido a que el tiempo de muestreo de 100 milisegundos escogido para la planta, origina una resolución de 10 partes por periodo en éste caso, en comparación con las 50 partes por periodo en PWM = 5 segundos. Es posible reducir el tiempo de muestreo del trasmisor, pero no es recomendable, ya que la dinámica de la planta es lenta, y por ende, son requeridos largos periodos de tiempo para monitorear cambios en el sistema. El xrtailab requiere de un mayor uso de recursos computaciones a medida que son graficados más puntos en el Scope. El aumento

excesivo de los periodos de tiempo, así como del número de eventos graficados por periodo, pueden ocasionar el bloqueo de aplicaciones del sistema operativo.

Figura 57: Variable Temperatura con Filtro



D.3. NORMALIZACIÓN:

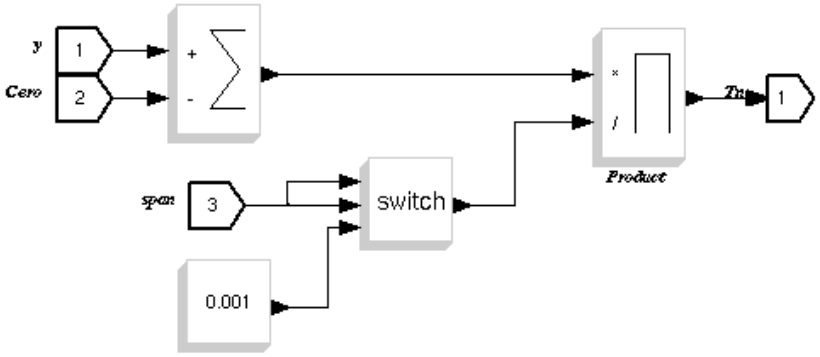
La normalización es la etapa final del transmisor y se realiza mediante el cálculo de la ecuación 2, donde "y" es la señal de temperatura en grados centígrados, "cero" el límite inferior de la señal medida y "*span*" el rango de medición.

$$TN = \frac{y_{(t)} - cero}{span} \qquad (2)$$

El cero es el límite inferior del rango de medición de temperatura con la cual el usuario trabajará, el *span* es la diferencia algebraica entre el límite superior y el límite inferior del rango de medición. En el caso del bloque Transmisor de temperatura los máximos límites son Temperatura ambiente y 86.7 grados centígrados. Los valores por defecto son cero de 45 grados y *span* de 20 grados.

La figura 58 muestra la implementación de la ecuación 2 en bloques de Scicos donde el puerto de entrada 1, "y", es la temperatura filtrada y medida en grados centígrados, los puertos de entrada 2 y 3 son los parámetro cero y *span* configurados por el usuario en las propiedades del bloque Transmisor de Temperatura. El puerto de entrada 3 está conectado a un *switch* que evita realizar una posible división por cero (0) en caso de que el usuario ingrese erróneamente este valor en el parámetro *Span*. El puerto de salida equivale a la temperatura normaliza entregada por el bloque planta MIC955.

Figura 58: Bloque de Normalización



ANEXO E: IMPLEMENTACIÓN DEL COMPONENTE DERIVATIVO

En este anexo, se presenta dos formas con las cuales se intentó solucionar el problema de la implementación del componente derivativo.

E.1. SOLUCIÓN MEDIANTE VARIABLES DE ESTADO:

La función de transferencia del filtro derivativo para un controlador PID de estructura paralela es la siguiente:

$$\frac{Y(s)}{U(s)} = \frac{T_d s}{\frac{T_d s}{N} + 1} \tag{3}$$

Multiplicando por $\frac{N}{T_d}$ al denominador y numerador, tenemos:

$$\frac{Y(s)}{U(s)} = \frac{Ns}{s + \frac{N}{T_d}} \tag{4}$$

Aplicando división de polinomios:

$$\begin{array}{rl} Ns & \left| \overline{s + N/T_d} \right. \\ \cline{2-2} -Ns - N^2/T_d & N \\ \hline -N^2/T_d & \end{array} \tag{5}$$

Se puede expresar la ecuación (5) de la siguiente manera:

$$\frac{Y(s)}{U(s)} = N - \frac{\frac{N^2}{T_d}}{s + \frac{N}{T_d}} \tag{6}$$

La ecuación (6) puede modificarse de la siguiente manera:

$$Y(s) = NU(s) - \hat{Y}(s) \tag{7}$$

En donde:

$$\hat{Y}(s) = \frac{\frac{N^2}{T_d}}{s + \frac{N}{T_d}} U(s) \tag{8}$$

La ecuación (8) puede escribirse de la forma:

$$\frac{\hat{Y}(s)}{\frac{N^2}{T_d}} = \frac{U(s)}{s + \frac{N}{T_d}} = Q(s) \tag{9}$$

A partir de la ecuación (9) se obtienen las ecuaciones (10) y (11)

$$sQ(s) = U(s) - \frac{N}{T_d} Q(s) \tag{10}$$

$$\hat{Y}(s) = \frac{N^2}{T_d} Q(s) \tag{11}$$

Se definen las variables de estado:

$$X_1(s) = Q(s) \tag{12}$$

$$X_2(s) = sQ(s) \tag{13}$$

Por lo tanto es evidente que:

$$sX_1(s) = X_2(s) \tag{14}$$

Lo cual puede escribirse como:

$$\dot{x}_1 = x_2 \tag{15}$$

La ecuación (10) puede reescribirse como:

$$X_2(s) = U(s) - \frac{N}{T_d} X_1(s) \tag{16}$$

O bien:

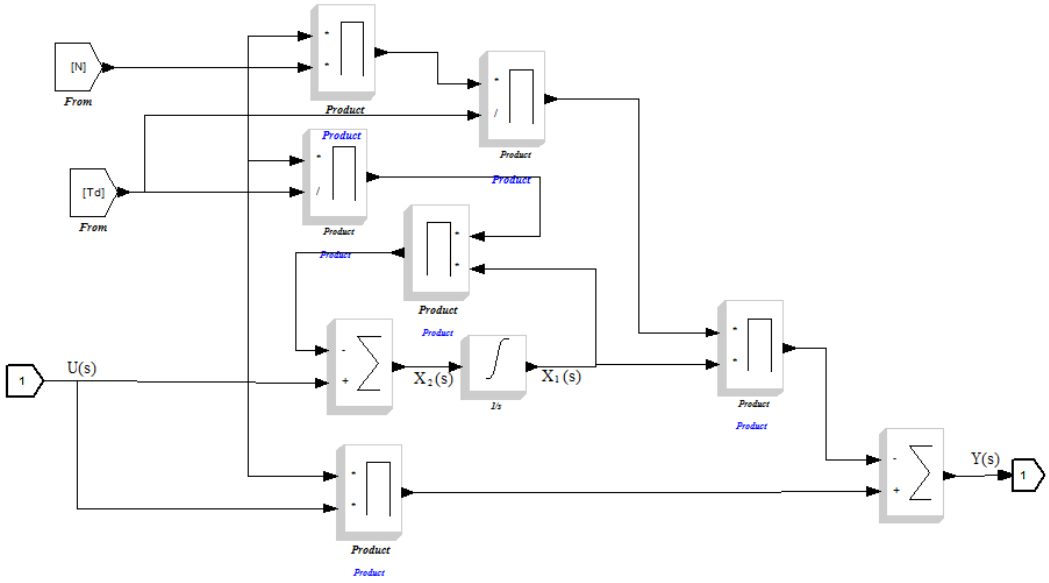
$$x_2 = u - \frac{N}{T_d} x_1 \tag{17}$$

Así mismo de las ecuaciones 7 y 11 se obtiene:

$$Y(s) = NU(s) - \frac{N^2}{T_d} X_1(s) \tag{18}$$

La figura 59 muestra la representación en diagrama de bloques del sistema definido mediante las ecuaciones (16) y (18).

Figura 59: Representación en diagrama de bloques del filtro derivador para el controlador PID de estructura paralela (forma canónica controlable)



La función de transferencia del filtro derivativo para un controlador PID de estructura serie es la siguiente:

$$\frac{Y(s)}{U(s)} = \frac{T_d s + 1}{\frac{T_d s}{N} + 1} \tag{19}$$

Multiplicando por $\frac{N}{T_d}$ al denominador y numerador, se obtiene:

$$\frac{Y(s)}{U(s)} = \frac{Ns + \frac{N}{T_d}}{s + \frac{N}{T_d}} \tag{20}$$

Aplicando división de polinomios:

$$\begin{array}{l} Ns + N/T_d \quad \big| \, s + N/T_d \\ \underline{-Ns - N^2/T_d \quad\quad N} \\ \quad\quad N/T_d - N^2/T_d \end{array} \tag{21}$$

Se puede expresar la ecuación (21) de la siguiente manera:

$$\frac{Y(s)}{U(s)} = N + \frac{\frac{N}{T_d} - \frac{N^2}{T_d}}{s + \frac{N}{T_d}} \tag{22}$$

La ecuación (22) puede modificarse de la siguiente manera:

$$Y(s) = NU(s) + \hat{Y}(s) \tag{23}$$

En donde:

$$\hat{Y}(s) = \frac{\left(\frac{N}{T_d} - \frac{N^2}{T_d}\right)}{s + \frac{N}{T_d}} U(s) \tag{24}$$

La ecuación (24) puede escribirse de la forma:

$$\frac{\hat{Y}(s)}{\left(\frac{N}{T_d} - \frac{N^2}{T_d}\right)} = \frac{U(s)}{s + \frac{N}{T_d}} = Q(s) \tag{25}$$

A partir de la ecuación (25) se obtiene las ecuaciones (26) y (27

$$sQ(s) = U(s) - \frac{N}{T_d} Q(s) \tag{26}$$

$$\hat{Y}(s) = \left(\frac{N}{T_d} - \frac{N^2}{T_d}\right) Q(s) \tag{27}$$

Se definen las variables de estado:

$$X_1(s) = Q(s) \tag{28}$$

$$X_2(s) = sQ(s) \tag{29}$$

Por lo tanto, es evidente que:

$$sX_1(s) = X_2(s) \tag{30}$$

Lo cual puede escribirse como:

$$\dot{x}_1 = x_2 \tag{31}$$

La ecuación (26) puede reescribirse como:

$$X_2(s) = U(s) - \frac{N}{T_d} X_1(s) \tag{32}$$

O bien:

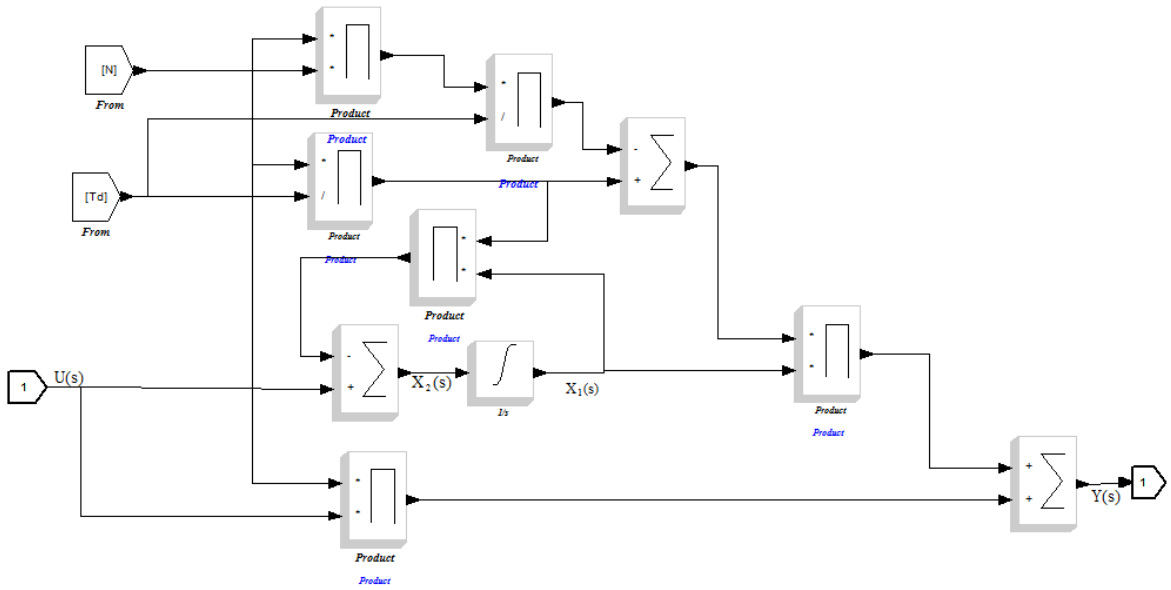
$$x_2 = u - \frac{N}{T_d} x_1 \tag{33}$$

Así mismo, de las ecuaciones (23) y (27) se obtiene:

$$Y(s) = NU(s) + \left(\frac{N}{T_d} - \frac{N^2}{T_d} \right) X_1(s) \tag{34}$$

La figura 60 muestra la representación en diagrama de bloques del sistema definido mediante las ecuaciones (32) y (34).

Figura 60: Representación en diagrama de bloques del filtro derivador para el controlador PID de estructura Serie (forma canónica controlable).



E.2. REPRESENTACIÓN ALTERNATIVA DEL FILTRO DERIVATIVO.

Partiendo de la función de transferencia del filtro derivativo para un controlador PID de estructura paralela (ver ecuación 3):

Se multiplica por N al denominador como al numerador, se obtiene:

$$\frac{Y(s)}{U(s)} = \frac{NT_d s}{T_d s + N} \tag{35}$$

La ecuación (35) puede modificarse de la siguiente manera:

$$\left(\frac{T_d s + N}{T_d s}\right) Y(s) = NU(s) \tag{36}$$

La ecuación (36) puede representarse de la siguiente manera:

$$\left(1 + \frac{N}{T_d s}\right) Y(s) = NU(s) \tag{37}$$

Distribuyendo $Y(s)$:

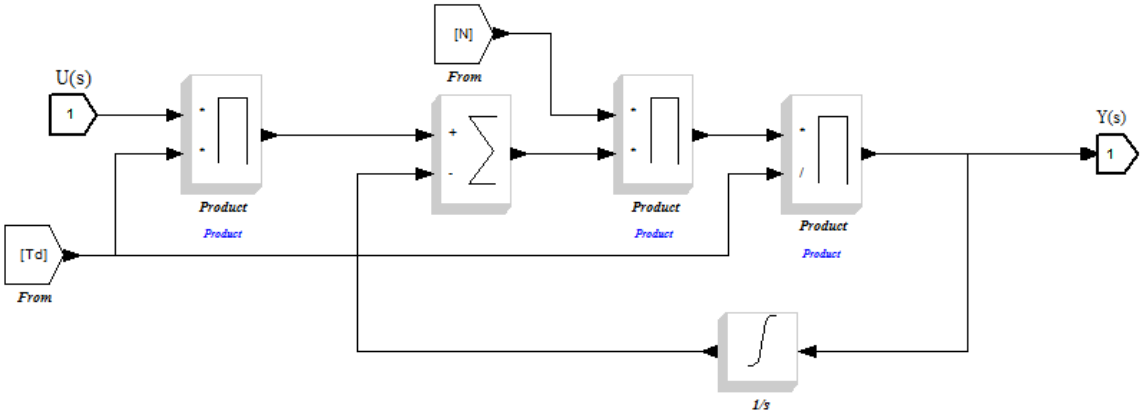
$$Y(s) = NU(s) - \frac{N}{T_d s} Y(s) \tag{38}$$

Por último la ecuación (38) puede escribirse de la forma:

$$Y(s) = \frac{N}{T_d}\left(T_d U(s) - \frac{1}{s} Y(s)\right) \tag{39}$$

La figura 61 muestra la representación en diagrama de bloques del filtro derivativo para el controlador PID de estructura paralela:

Figura 61: Diagrama de bloques filtro derivativo para un controlador PID de estructura paralela



Partiendo de la función de transferencia del filtro derivativo para un controlador PID de estructura Serie (ver ecuación 19):

Multiplicando por N al denominador como al numerador, se obtiene:

$$\frac{Y(s)}{U(s)} = \frac{NT_d s + N}{T_d s + N} \tag{40}$$

La ecuación (1.32) puede expresarse de la siguiente manera:

$$(T_d s + N) Y(s) = (NT_d s + N) U(s) \tag{41}$$

La ecuación (1.33) puede representarse de la siguiente manera:

$$\left(\frac{T_d s + N}{T_d s}\right) Y(s) = \left(N + \frac{N}{T_d s}\right) U(s) \tag{42}$$

La ecuación (1.34) puede modificarse de la siguiente manera:

$$\left(1 + \frac{N}{T_d s}\right) Y(s) = \left(N + \frac{N}{T_d s}\right) U(s) \tag{43}$$

Distribuyendo $Y(s)$:

$$Y(s) = \left(N + \frac{N}{T_d s}\right) U(s) - \frac{N}{T_d s} Y(s) \tag{44}$$

La ecuación (1.36) puede modificarse de la forma:

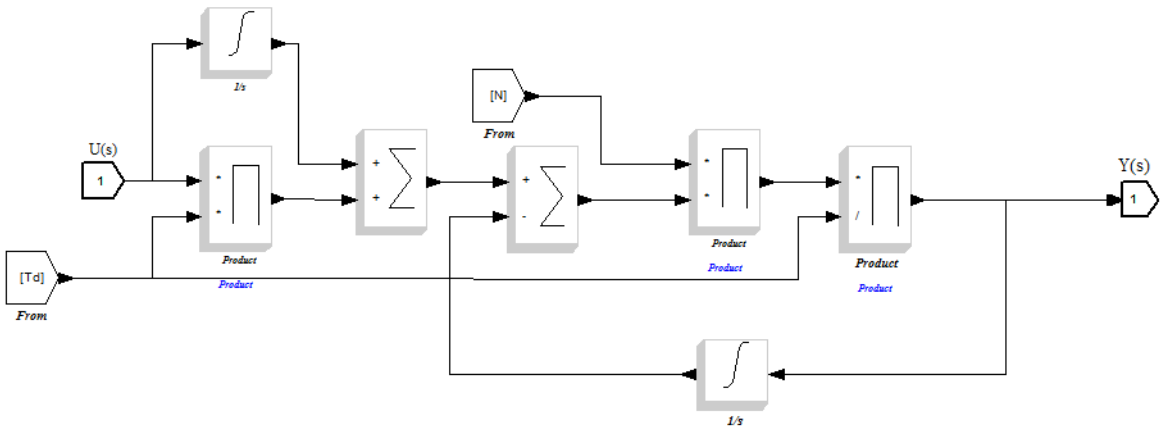
$$Y(s) = NU(s) + \frac{N}{T_d s} U(s) - \frac{N}{T_d s} Y(s) \tag{45}$$

Por último la ecuación (1.37) puede expresarse:

$$Y(s) = \frac{N}{T_d}\left(T_d U(s) + \frac{1}{s} U(s) - \frac{1}{s} Y(s)\right) \tag{46}$$

La figura 62 muestra la representación en diagrama de bloques del filtro derivativo para el controlador PID de estructura serie:

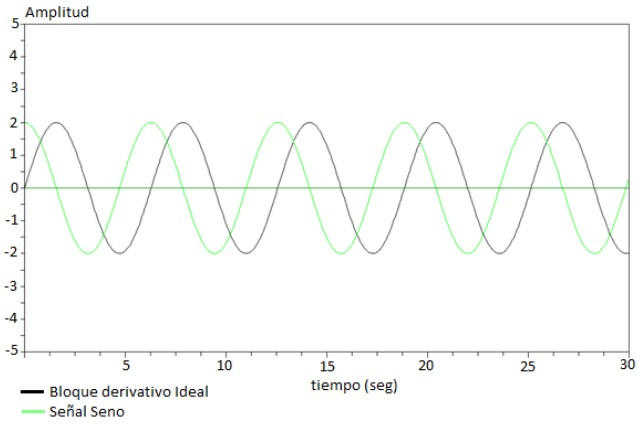
Figura 62: Diagrama de bloques filtro derivativo para un controlador PID de estructura serie



E.3. EXPERIMENTOS REALIZADOS CON LOS DIFERENTES BLOQUES.

Se realizó una prueba experimental para validar el componente derivativo. Esta prueba consistía en someter las diferentes representaciones a una onda seno de amplitud 2 y compararlas con la respuesta de un derivador ideal. Se dio a N un valor de 10 y el tiempo derivativo igual a 1 segundo. La figura 63 muestra la respuesta del bloque derivativo ideal (traza negra) ante una entrada seno de amplitud 2 (traza verde)

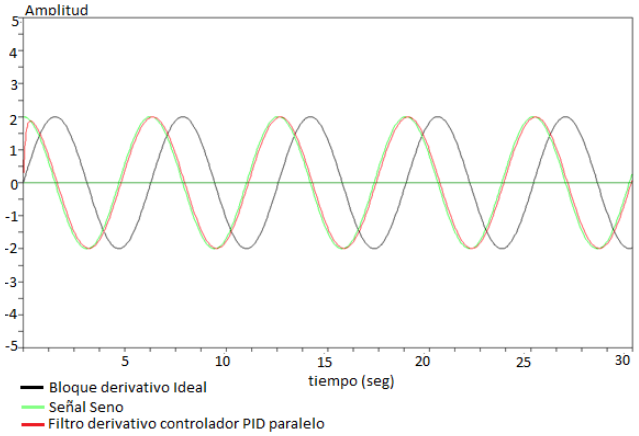
Figura 63: Respuesta bloque derivativo ideal ante una señal seno de amplitud 2



N tiene un valor de 10 y el tiempo derivativo es 1.

En la figura 64 se puede observar la respuesta del filtro derivativo implementado mediante variables de estado (ver figura 59).La traza negra representa la onda seno, la traza verde la respuesta del bloque derivativo ideal y la traza roja la respuesta del filtro derivativo.

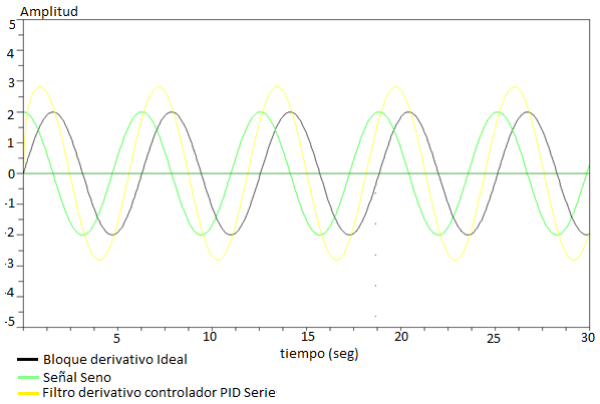
Figura 64: Comparación Respuesta filtro derivativo controlador PID paralelo (ver figura 59.) – Bloque derivativo ideal



Como puede observarse la respuesta del filtro derivativo se asemeja mucho a la respuesta del bloque derivador ideal.

En la figura 65 se puede observar la respuesta del filtro derivativo para controlador PID de estructura serie implementado mediante variables de estado (ver figura 60).La traza negra representa la onda seno, la traza verde la respuesta del bloque derivativo ideal y la traza amarilla la respuesta del filtro derivativo.

Figura 65: Respuesta filtro derivativo controlador PID serie (ver figura 2.)



Como puede observarse la respuesta del filtro derivativo difiere un poco de la respuesta del bloque derivador ideal, sin embargo esto se debe a la implementación del filtro derivativo.

ANEXO F: MÉTODOS DE IDENTIFICACIÓN TEORÍA Y PRACTICA

En la puesta en marcha de un sistema de control industrial, primero se requiere sintonizar correctamente los controladores del lazo de control retroalimentado. Para ello, es necesario contar con información del comportamiento dinámico del proceso. La mayoría de métodos de sintonización de controladores se basan en los parámetros identificados de un modelo de orden reducido, que permite representar sistemas dinámicos de orden alto, por esta razón los más empleados son los modelos de primer o segundo orden más tiempo muerto, por ende, el proceso de sintonización del controlador consta así de dos etapas: identificación y sintonización.

Teóricamente, para llegar a obtener un modelo podrían adoptarse dos enfoques diferentes (Arafet et al, 2008):

- **Por vía analítica (*a priori*)**: Se determinan las ecuaciones y parámetros que intervienen siguiendo exclusivamente las leyes generales de la física. En este enfoque es preciso tener en cuenta que normalmente es extremadamente difícil considerar todas las leyes físicas que intervienen en el proceso y que por lo general resultan en un modelo complejo, difícilmente manejable por las técnicas de sintonización de sistemas de control.
- **Por vía experimental (*a posteriori*)**: Se considera el sistema como una "caja negra", con entradas y salidas determinadas. En esta situación se realizaría un conjunto de experimentos que proporcionan pares de medidas de entradas y salidas durante la evolución del sistema hacia el estado estacionario, a partir de los cuales se trata de determinar el modelo del sistema.

En la práctica se combinan los dos enfoques, teniendo en cuenta primero las leyes físicas y las condiciones particulares de trabajo, para establecer hipótesis sobre la estructura y propiedades del modelo que se pretende identificar. Y segundo, adoptando las hipótesis establecidas anteriormente, teniendo en cuenta las mediciones para determinar el modelo.

Algunos factores para tener en cuenta en el momento de realizar el modelo son:

- Un sistema puede ser no lineal, sin embargo, puede adoptarse un modelo lineal alrededor de un punto de operación, que presente variaciones pequeñas que no sobrepasen la zona de trabajo.
- En sistemas con múltiples entradas, es posible aplicar el principio de superposición, considerando la salida, como la suma de salidas elementales correspondientes a una sola entrada.
- Pueden existir parámetros que varíen en razón de perturbaciones lentas o aparecer no linealidades, que no fueron manifiestas en el transitorio alrededor del punto de operación.

Los métodos de identificación pueden ser clasificados según sus características en dos grandes grupos: métodos de identificación no paramétrica y métodos de identificación paramétrica.

E.1. MÉTODOS DE IDENTIFICACIÓN NO PARAMÉTRICA

Estos métodos se caracterizan por que los parámetros de los modelos resultantes son obtenidos de respuestas temporales o frecuenciales. Son conocidos también como métodos indirectos. Entre estos métodos se encuentran:

- El análisis transitorio: Se observa la salida del sistema frente a una entrada escalón, desde la respuesta obtenida se derivan los parámetros.
- El análisis de frecuencia: La entrada es una onda sinusoidal conocida, el cambio en la amplitud y fase de la onda seno de salida puede ser analizadas a través de los diagramas de Bode, Nyquist, Nichols.
- Análisis de correlación: Se usan señales periódicas o estocásticas como entrada del sistema.

E.1.1. Métodos de Identificación de Lazo Abierto

Los métodos de identificación de lazo abierto son métodos de análisis transitorio generalmente realizado por curva de reacción del proceso, donde el controlador no está presente o se encuentra en modo manual. La identificación de los parámetros de estos modelos (ganancia, tiempo muerto y constantes de tiempo), puede hacerse a partir de la respuesta del proceso a un cambio escalón en la entrada, denominada curva de reacción del proceso, la cual se registra desde el instante en que es aplicado el cambio hasta el momento en que el sistema alcance un nuevo punto de operación estable si es un proceso auto-regulado.

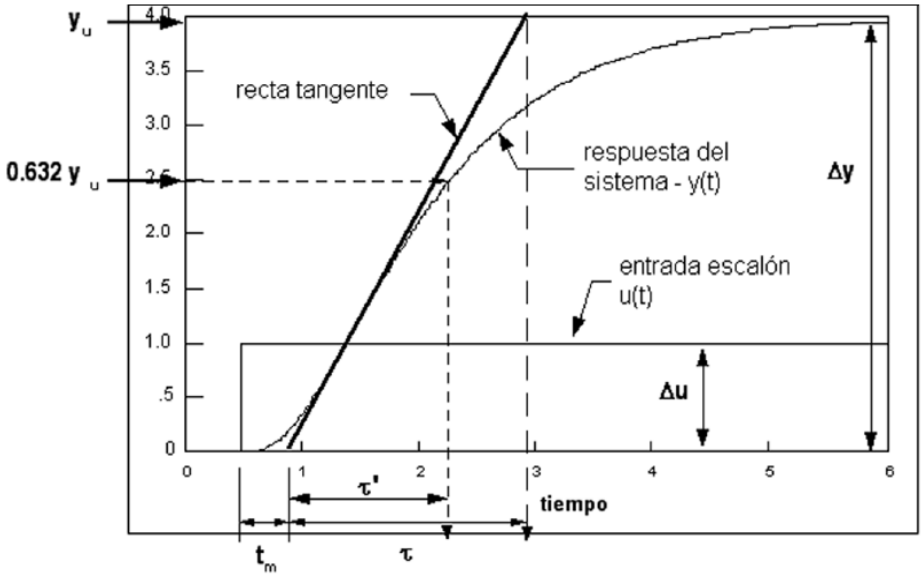
El escalón es considerado válido, si la constante de tiempo de la señal real es menor que la décima parte de la menor constante de tiempo que se quiere determinar en la identificación. El uso de esta señal tiene la ventaja de la sencillez en su generación y que el tiempo de experimentación es corto. Como desventaja se puede mencionar la introducción de una alteración relativamente grande en el comportamiento del sistema, lo cual no siempre es permisible (Arafet et al, 2008).

Entre estos métodos se encuentran: Los método de la recta tangente, los métodos de dos puntos, los métodos de tres puntos, los métodos de Strejc, y el método de las áreas características, los cuales son vista a continuación.

E.1.1.1. Métodos de la Recta Tangente

Requieren el trazo de una recta tangente en el punto de inflexión de la curva como se observa en la figura 66, la determinación del instante en que este punto ocurre, la determinación de los tiempos para que la respuesta alcance dos o tres porcentajes determinados del cambio total de la respuesta, o el cálculo de áreas definidas por las curvas de las señales de entrada y salida. Estos métodos tienen el inconveniente que este punto usualmente es difícil de determinar en forma precisa (Alfaro, 2006).

Figura 66: Método de la recta tangente



Fuente: Modificado de (Alfaro, 2006)

Entre estos métodos se encuentran el método de la tangente de Ziegler – Nichols y el método de la tangente modificado de Miller, los cuales se describen a continuación:

Método de la tangente de Ziegler Y Nichols

El procedimiento de identificación puede utilizarse para obtener un modelo de primer orden más tiempo muerto indicado en la Ecuación 47, donde primero se debe identificar la ganancia K_p, la constante de tiempo τ y el tiempo muerto aparente del sistema t_m.

$$G_p(s) = \frac{K_p e^{-t_m s}}{\tau s + 1} \tag{47}$$

La ganancia se expresa en términos de la variación de la salida dividida por la variación en la entrada, tal como se observa en la ecuación 48.

$$k_p = \Delta y / \Delta u \tag{48}$$

El tiempo transcurrido entre la aplicación del escalón y el punto en que la recta tangente corta el eje del tiempo, es el tiempo muerto aparente del sistema, y el tiempo transcurrido entre el instante y el tiempo en que la tangente corta el valor final de la salida y_u es la constante de tiempo.

Método De La Tangente Modificado De Miller

El procedimiento propuesto por Miller es una variación del método de Ziegler y Nichol, donde la ganancia y el tiempo muerto se calculan de la misma forma. La

variación radica en el cálculo de la constante de tiempo requerido para que la respuesta alcance el 63.2% del cambio total a partir del tiempo muerto. Esta variación hace que la respuesta del modelo y la del sistema real coincidan por lo menos en el punto de tiempo dado por la ecuación 49.

$$t = t_m + \tau \tag{49}$$

E.1.1.2 Métodos de Dos Puntos

Para identificar los parámetros, constante de tiempo y tiempo muerto, se puede establecer dos ecuaciones con dos incógnitas utilizando dos puntos sobre la curva de reacción. Esto garantiza que la respuesta del modelo coincida con la del sistema real en mínimo dos puntos. A continuación se describen el método de Smith y algunos métodos de dos puntos en general.

Método de Smith

Los instantes seleccionados por el autor fueron los tiempos requeridos para que la respuesta alcance el 28.3% (t_{28}) y el 63.2% (t_{63}) del valor final. Estos tiempos se corresponden a las ecuaciones, y corresponden a las ecuaciones 50 y 51, respectivamente.

$$t_{28} = t_m + \tau / 3 \tag{50}$$

$$t_{63} = t_m + \tau \tag{51}$$

Este sistema de ecuaciones se resuelve para t_m y τ t_m. Obteniendo las ecuaciones 52 y 53:

$$\tau = 1.5(t_{63} - t_{28}) \tag{52}$$

$$t_m = t_{63} - \tau \tag{53}$$

La ganancia del modelo se calcula a través de la ecuación 54:

$$K_p = \frac{\left(\frac{\Delta y(t)}{span}\right)}{\Delta u(t)} \tag{54}$$

Método de Dos Puntos General

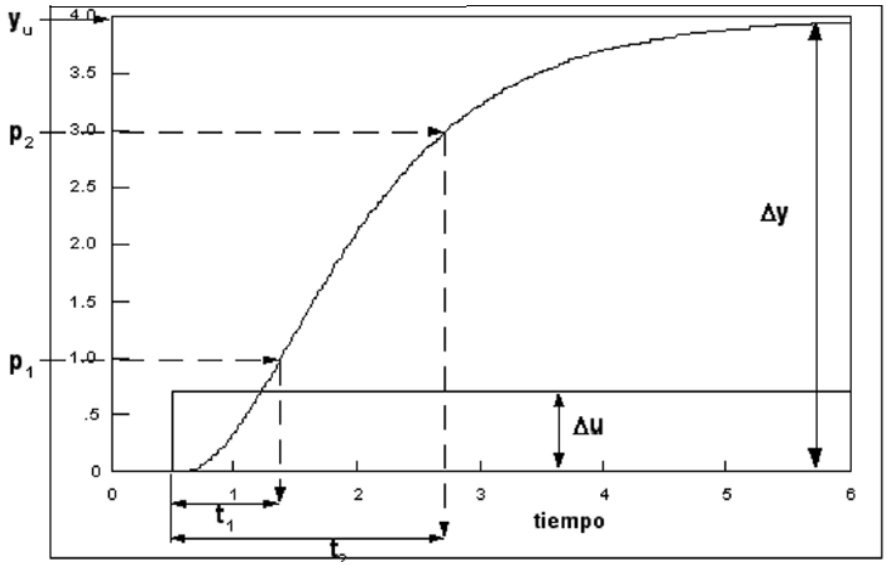
Posterior a la presentación del método de dos puntos de Smith se han desarrollado otros métodos basados en el mismo procedimiento, diferenciándose únicamente en la selección de los dos instantes donde la respuesta del modelo coincide con la del proceso real. Para modelos de primer orden más tiempo muerto, si tomamos a p_1 y p_2 como valores porcentuales del cambio en la respuesta del sistema a un cambio escalón en la entrada y t_1 y t_2 como los tiempos requeridos para alcanzar estos valores, como se muestra en la figura 67, entonces

los parámetros del modelo se pueden obtener de las ecuaciones 55 y 56. La ganancia se calcula con la ecuación 54.

$$\tau = at_1 + bt_2 \tag{55}$$

$$t_m = ct_1 + dt_2 \tag{56}$$

Figura 67: Método de los dos puntos



Fuente: Modificado de (Alfaro, 2006)

Los porcentajes del cambio en la respuesta para la determinación de los dos tiempos requeridos por el procedimiento de identificación, así como los valores de las constantes *a, b, c* y *d* para los métodos de Alfaro, Bröida, Chen y Yan, Ho *et al*, Smith y Vitecková *et al* se indican en la Tabla E.1.

Tabla 8: Constantes para métodos de dos puntos. Modificado de Alfaro 2006.

Método	$\%p_1(t_1)$	$\%p_2(t_2)$	a	b	c	d
Alfaro	**25.0**	**75.0**	**-0.910**	**0.910**	**1.262**	**-0262**
Bröda	**28.0**	**40.0**	**-5.500**	**5.500**	**2.800**	**-1.800**
Chen y Yang	**33.0**	**67.0**	**-1.400**	**1.400**	**1.540**	**-0.540**
Ho *et al*	**35.0**	**85.0**	**-0.670**	**0.670**	**1.300**	**-0.290**
Smith	**28.3**	**63.2**	**-1.500**	**1.500**	**1.500**	**-0.500**
Vitecková *et al*	**33.0**	**70.0**	**-1.225**	**1.245**	**1.498**	**-0.498**

Para obtener modelos de segundo orden con un polo doble, indicado en la ecuación (57), pueden utilizarse los métodos de Ho *et al* y el de Viteckotá *et al* que se basan en dos puntos de la curva de reacción del proceso.

$$G_p(s) = \frac{k_p e^{-t_m s}}{(\tau s + 1)^2} \tag{57}$$

Tabla 9: Constantes de plantas de segundo orden. Modificado de Alfaro 2006.

Método	$\%p_1(t_1)$	$\%p_2(t_2)$	a	b	c	d
Ho *et al*	35.0	85.0	-0.463	0.463	1.574	-0.574
Vitecková *et al*	33.0	70.0	-0.749	0.749	1.937	-0.937

E.1.1.3. Métodos de Tres Puntos

Se utiliza en modelos de segundo sobreamortiguado orden más tiempo muerto dado por la ecuación 60, y en modelos se segundo orden subamortiguado más tiempo muerto dado por la ecuación 61, que tienen tres parámetros en adición a la ganancia, por lo que requiere de tres puntos sobre la curva de reacción para poder ser identificado. A continuación se muestran el método de Stark y el método de Jahanmiri y Fallahi.

$$G_p(s) = \frac{k_p e^{-t_m s}}{(\tau_1 s + 1)(\tau_2 s + 1)} \tag{58}$$

$$G_p(s) = \frac{k_p e^{-t_m s}}{(\tau_1 s + 1)(\tau_2 s + 1)} \tag{59}$$

$$G_p(s) = \frac{k_p e^{-t_m s}}{(\tau_1 s + 1)(\tau_2 s + 1)} \tag{60}$$

$$G_p(s) = \frac{\omega_n^2 k_p e^{-t_m s}}{s^2 + 2\varsigma\omega_n s + \omega_n^2} = \frac{k_p e^{-t_m s}}{\tau^2 s^2 + 2\varsigma\tau s + 1} \tag{61}$$

Método de Stark

Los instantes seleccionados por este método son los tiempos requeridos para que la respuesta alcance el 15% (t_{15}), el 45% (t_{45}) y el 75% (t_{75}) del valor final. Su procedimiento de identificación esta dado por:

Sea:

$$x = \frac{t_{45} - t_{15}}{t_{75} - t_{15}} \tag{62}$$

$$\varsigma = \frac{0.0805 - 5.547(0.475 - x)^2}{x - 0.356} \tag{63}$$

$$f_2(\varsigma) = 0.708(2.811)^2 \tag{64}$$

$$f_2(\varsigma) = 2.6\varsigma - 0.60w \tag{65}$$

$$\omega_n = \frac{f_2(\varsigma)}{t_{75} - t_{15}} \tag{66}$$

$$f_3(\varsigma) = 0.922(1.66)^{\varsigma} \tag{67}$$

$$t_m = t_{45} - \frac{f_3(\varsigma)}{\omega_n} \tag{68}$$

$$\tau_1 = \frac{\varsigma + \sqrt{\varsigma^2 - 1}}{\omega_n},\ para\ \varsigma\ \geq\ 1 \tag{69}$$

$$\tau_2 = \frac{\varsigma - \sqrt{\varsigma^2 - 1}}{\omega_n} w,\ para\ \varsigma\ \geq\ 1 \tag{70}$$

El valor de la ganancia es determinado con la Ecuación 54.

Método de Jahanmiri y Fallahi

Se basa en los tiempos para alcanzar el 2% (t_2) o el 5% (t_5), el 70% (t_{70}) y el 90 (t_{90}) del valor final. Las ecuaciones para identificar el modelo son:

$$t_m = t_2\ o\ t_5 \tag{71}$$

$$\eta = \frac{t_{90} - t_{70}}{t_{90} - t_m} \tag{72}$$

$$\varsigma = \sqrt{\frac{0.4844651 - 0.75323499\eta}{1 - 2.0946444\eta}} \qquad \eta \leq 0.4771 \tag{73}$$

$$\varsigma = 13.9352 \qquad \eta \leq 0.4771 \tag{74}$$

$$\tau = \frac{t_{90} - t_m}{0.424301 + 4.62533\varsigma - 2.65412e^{-\varsigma}} \tag{75}$$

La ganancia se calcula con la Ecuación 54.

E.1.1.4 Métodos de Strejc

Puede utilizarse este método para identificar un modelo de polos múltiples dado por las Ecuación (76) y(77). En el caso particular de que las constantes de tiempo del sistema sean aproximadamente iguales.

$$G_p(s) = \frac{k_p}{(\tau s + 1)^n} \tag{76}$$

$$G_p(s) = \frac{k_p e^{-t_m s}}{(\tau s + 1)^n} \quad (77)$$

Se requiere trazar una recta tangente en el punto de inflexión de la curva de reacción del proceso y obtener los valores T_u y T_a, que corresponden al tiempo muerto t_m y a la contante de tiempo t del método de la tangente. La ganancia de los dos modelos se obtiene con la Ecuación 54.

El orden del modelo dado en la ecuación 76 se obtiene redondeando al número entero inferior de n dado en la Ecuación 78 y la constante de tiempo de polo múltiple es dado por la Ecuación 54.

$$n = 10\frac{T_u}{T_a} + 1 \quad (78)$$

$$\tau = \frac{T_a (n-1)^{n-1}}{(n-1)!} e^{-(n-1)} \quad (79)$$

Para identificar los parámetros del modelo POMTM, una vez obtenidos T_u y T_a de la curva de respuesta, se calcula T_u / T_a y se determina el orden del modelo n de la Tabla E.3, como el valor de n correspondiente al valor T_u/T_a inmediatamente inferior al calculado. Con los valores obtenidos de T_a/t y T_u/t de correspondientes a n, se procede a calcular t. La valides del modelo puede medirse con la proximidad de los dos valores de t.

Tabla 10: Metodo de Strejc

N	T_u/T_a	Tu/t	T_a/t
1	0.000	0.000	0.000
2	0.104	0.282	2.718
3	0.218	0.805	3.695
4	0.319	1.425	4.465
5	0.410	2.100	5.119
6	0.493	2.811	5.699
7	0.570	3.549	6.226
8	0.642	4.307	6.711
9	0.000	5.081	7.164
10	0.773	5.869	7.590

Se calcula el tiempo muerto aparente del sistema con los valores de T_u/T_a de la Tabla E.3 y el orden del sistema n, encontrando el valor de T_u que será denominado T_{ut} . El tiempo muerto corresponde a la diferencia entre éste valor y el valor de T_u determinado de forma grafica posteriormente. Ver Ecuación 80.

$$t_m = T_{ur} - T_{rt} \quad (80)$$

E.1.1.5. Método de las Áreas Características

Consiste en la identificación de un modelo de orden alto más tiempo muerto dado por la Ecuación 81 mediante el cálculo de las áreas determinadas por la curva de reacción del sistema.

$$G_p(s) = \frac{k_k e^{-t_m s}}{\prod_{j=1}^{n}(\tau_j s + 1)} \tag{81}$$

Define las áreas dadas por las Ecuaciones 82 y 83.

$$S_y = \int_0^{\infty} |\Delta y - y(\tau)|\, d\tau \tag{82}$$

$$S(t) = \int_0^{t} y(\tau) d\tau \tag{83}$$

La ganancia k_k del modelo es dada por la ecuación 48. El área S_y es igual a la suma de las contantes de tiempo más el tiempo muerto del sistema, como lo muestra la Ecuación 84

$$\tau_T = \sum_{j=1}^{n} \tau_j + t_m = \frac{S_y}{\Delta y} \tag{84}$$

E.1.2. Métodos de Identificación en Lazo Cerrado

Los métodos de identificación en lazo cerrado deben de obtener información del comportamiento del sistema bajo una condición de control retroalimentado, y mediante esta información estimar los parámetros del modelo de la planta. A continuación de mencionaran métodos que utilizan la información última y métodos de control P

E.1.2.1. Métodos para la Obtención de la Información "Última"

Antes de poder identificar un modelo a partir de la información última de un proceso es necesario conocer algunos métodos que nos permitan conocer dicha información. Se denomina información última o critica a la que se obtiene de la operación del sistema de control en el límite de estabilidad, obteniendo la ganancia K_{cu} del controlador proporcional que lleva al sistema al límite de su estabilidad (oscilación mantenida) y el periodo T_u en el cual oscila el sistema. A continuación se describen dos métodos para la obtención de la información última, estos son: El método de oscilación mantenida y el método de realimentación con relé.

Método de Oscilación sostenida

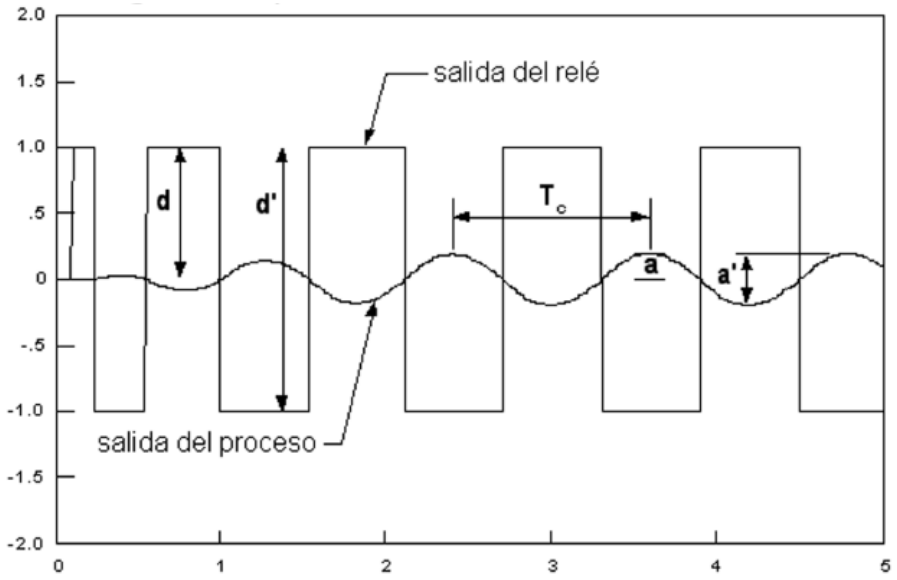
Desarrollado por Ziegler y Nichols. Se determinan los parámetros K_{cu} y T_u a través de un procedimiento iterativo que emplea un controlador proporcional. La ganancia del controlador aumenta paulatinamente hasta lograr que la respuesta del sistema

oscile a un cambio de escalón. En último valor de la ganancia es K_{cu} y el periodo de oscilación de la respuesta es T_u.

Método de Retroalimentación con Relé

Desarrollado por Aströn y Hägglund y se basa en la siguiente propiedad: Un proceso que tenga un retrazo de fase de 180º a altas frecuencias, oscilará con un periodo de oscilación igual al periodo crítico bajo el control de un relé. En la figura 68 se puede observar la salida del sistema con retroalimentación con relé.

Figura 68: Salida del sistema con retroalimentación con relé



Fuente: Modificado de (Alfaro, 2006)

Si la amplitud de la señal pasada a través del relé es *d*, se puede obtener por series de Fourier que la amplitud de la primera armónica de la salida del relé es igual a $4d/\pi$. Si "*a"* es la amplitud de salida del proceso, la ganancia critica estará dada por la Ecuación 85.

$$K_{cu} = \frac{4d}{\pi a} = \frac{4d'}{\pi a'} \tag{85}$$

E.1.2.2. Métodos de Identificación Basados en la Información Última

Los parámetros obtenidos por los métodos de obtención de información última pueden usarse para identificar un modelo por medio del método de Chen, el método de Ho *et al* y el método de Lee y Sung.

Método de Chen

Este método utiliza una prueba de lazo cerrado con un controlador proporcional para determinar la ganancia de la planta, junto con la información última para

identificar un modelo de POMTM. Las Ecuaciones 86, 87 y 88, determinan los valores de K_p,τ, t_m.

$$K_p = \frac{\Delta y}{K_c(\Delta u - \Delta y)} \tag{86}$$

$$\tau = \frac{T_u}{2\pi}\sqrt{K_{cu}^2 kp - 1} \tag{87}$$

$$t_m = \frac{T_u}{2\pi}\left[\pi - \tan^{-1}\left(\frac{2\pi\tau}{T_u}\right)\right] \tag{88}$$

Método de Ho *et al*

Utiliza la información última para identificar un modelo con polo doble más tiempo muerto dado por la Ecuación 89.

$$G_p(s) = \frac{k_p e^{-t_m s}}{(\tau' s + 1)^2} \tag{89}$$

Las Ecuaciones 90 y 91, permiten calcular los valores de τ' y t_m:

$$\tau' = \frac{T_u}{2\pi}\sqrt{K_{cu}k_p - 1} \tag{90}$$

$$t_m = \frac{T_u}{2\pi}\left[\pi - 2\tan^{-1}\left(\frac{2\pi\tau'}{T_u}\right)\right] \tag{91}$$

La ganancia de la planta se obtiene a partir de una prueba en lazo cerrado con un controlador proporcional con la Ecuación 86

Método de Lee y Sung

Combinando los procedimientos de obtención de información última de retroalimentación con relé y de oscilación mantenida, se obtiene la información necesaria para identificar en base a un modelo POMTM.

Se realiza primero la prueba de retroalimentación con relé para obtener la amplitud de la salida denominada "*a*" y luego una prueba con un controlador P con la ganancia dada por la Ecuacion 92.

$$K_c = 0.6\frac{4d}{\pi a} \tag{92}$$

El modelo se identifica a partir de las Ecuaciones 93 y 94.

$$\tau = \frac{T_u / 2}{In\left[\left(1+\frac{a}{k_p d}\right)/\left(1-\frac{a}{k_p d}\right)\right]} \quad (93)$$

$$tm = \tau In\left(\frac{1+e^{Tu/2\tau}}{2}\right) \quad (94)$$

E.1.2.3 Métodos de Control P

Consisten en una única prueba de lazo cerrado, la cual se basa en la identificación del proceso a partir de la curva de respuesta del sistema a un cambio escalón en el valor deseado, cuando este se controla con un controlador proporcional Gc(s)= Kc. A continuación están descritos los métodos deYuwana y Seborg, Jutan y Rodriguez y el método de Lee.

Método deYuwana y Seborg

Fue el primer método procedimiento de identificación de lazo cerrado que empleó un controlador proporcional, sin la necesidad de llevar el sistema al límite de estabilidad. El procedimiento requiere la respuesta del sistema a un cambio escalón en el valor deseado, la cual debe ser subamortiguada. Se modela la misma como la respuesta de un sistema de segundo orden más tiempo muerto, y se identifica para el proceso un modelo de primer orden más tiempo muerto.

La figura 69 muestra los valores y_{p1}, y_{p2}, y_{m1} y Δt que deben obtenerse y corresponden a los valores del primer y segundo pico de la respuesta, el primer mínimo, el valor final y el semiperiodo.

Por medio de una aproximación de Padé de primer orden, para un modelo de segundo orden más tiempo muerto del lazo cerrado, se obtiene la Ecuación 95

$$e^{-t_m s} = \frac{1-0.5t_m s}{1+0.5t_m s} \quad (95)$$

La ganancia del modelo de la planta se identifica usando la Ecuación 48, la constante de tiempo y el tiempo muerto aparente son dados por las Ecuaciones 96 y 97.

$$\frac{\Delta t}{\pi} = \left[\varsigma\sqrt{K+1}+\sqrt{\varsigma^2(K+1)+K}\right]\sqrt{(1-\varsigma^2)(K+1)} \quad (96)$$

$$t_m = \frac{2\Delta t\sqrt{(1-\varsigma^2)(K+1)}}{\pi\left[\zeta\sqrt{K+1}+\sqrt{\zeta^2(K+1)+K}\right]} \quad (97)$$

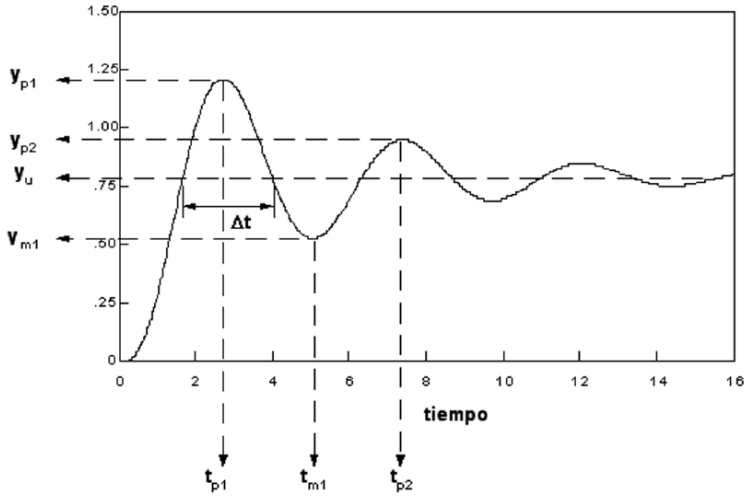
Donde:

$$K = K_c k_p \quad (98)$$

$$t_m = \varsigma = (\varsigma_1 + \varsigma_2)/2 \tag{99}$$

$$\varsigma_1 = \frac{-In[(y_u - y_{m1})/(y_{p1} - y_u)]}{\sqrt{\pi^2 + In^2[(y_u - y_{m1})/(y_{p1} - y_u)]}} \tag{100}$$

Figura 69: Método deYuwana y Seborg



Fuente: Modificado de (Alfaro, 2006)

$$\varsigma_2 = \frac{-In[(y_{p2} - y_u)/(y_{p1} - y_u)]}{\sqrt{4\pi^2 + In^2[(y_{p2} - y_u)/(y_{p1} - y_u)]}} \tag{101}$$

Método de Jutan y Rodríguez

A diferencia de Yawana y Seborg. Propone el uso de la aproximación dada en la Ecuación 102, con γ_1= -0.61453, γ_2= 0,1247 y δ = 0,3866.

$$e^{-t_m s} = \frac{1 + \gamma_1 t_m s + \gamma_2 t_m^2 s^2}{1 + \delta t_m s} \tag{102}$$

La ganancia del modelo se obtiene de la Ecuación 86, y los parámetros de la constante de tiempo y tiempo muerto, de las Ecuaciones 103 y 104.

$$\tau = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{103}$$

$$t_m = \alpha + \beta\tau \tag{104}$$

Donde:

$$a = \beta^2 \gamma_2 K + \beta\delta, \quad b = 2\gamma_2 K\alpha\beta + \alpha\delta, \quad c = \gamma_2 K\alpha^2 - \tau'^2(1+K) \tag{105}$$

$$\alpha = \frac{2\varsigma'\tau'(1+K)}{\delta + \gamma_1 K}, \quad \beta = -(\delta + \gamma_1 K)^{-1} \tag{106}$$

$$K = \frac{\Delta y}{\Delta u - \Delta y}, \quad \varsigma' = \frac{-In\alpha_1}{\sqrt{\pi^2 + In^2\alpha_1}}, \quad \alpha_1 = \frac{y_u - y_{m1}}{y_{p1} - y_u}, \quad \tau' = \frac{\Delta t}{\pi}\sqrt{1-\varsigma'^2} \tag{107}$$

Método de Lee

Este método logra que los polos de la función de transferencia, identificada para la respuesta subamortiguada con control P, coincidan con los de la desarrollada a partir del supuesto que la planta controlada sea de primer orden más tiempo muerto.

La ganancia está dada por la Ecuación 86, y la constante de tiempo y el tiempo muerto se obtiene de las Ecuaciones 108 y 109.

$$e^{-\alpha t_m s} - \frac{K_c kp}{\beta}\left[\alpha sen(\beta t_m) - \beta\cos(\beta t_m)\right] = 0 \tag{108}$$

$$\tau = \frac{1}{\alpha}\left[1 + K_c k_p e^{\alpha t_m}\cos(\beta t_m)\right] \tag{109}$$

Donde:

$$\alpha = \frac{\varsigma}{\tau'} \quad y \quad \beta = \frac{\sqrt{1-\varsigma^2}}{\tau'} \tag{110}$$

Debido a que la ecuación 108 no es lineal, debe resolverse en forma iterativa utilizando el algoritmo dado por la Ecuación 111.

$$t_{m,k+1} = \frac{1}{\beta}\left[\upsilon + \tan^{-1}(\psi)\right], \quad \psi = \frac{\beta e^{-\alpha t_{m,k}}}{K_c k_p\sqrt{\alpha^2+\beta^2}\cos(\beta t_{m,k} - \upsilon)}, \quad \upsilon = \tan^{-1}\left(\frac{\beta}{\alpha}\right) \tag{111}$$

Iniciando con: $t_{m,0} = (\upsilon+\pi/4)/\beta$. Una vez que converja el algoritmo de la Ecuación 108, el valor obtenido del tiempo t_m se utiliza para calcular la contante de tiempo del sistema con la Ecuación 109.

E.2. MÉTODOS DE IDENTIFICACIÓN PARAMÉTRICA

Se pretenden obtener los valores de los coeficientes de las funciones o matrices de transferencia, o los elementos de las matrices de representación en el espacio de estado, mediante el uso de secuencias binarias seudo-aleatorias (PRBS), como señales de prueba y métodos numéricos basados en los mínimos cuadrados para la identificación de los modelos en tiempo continuo o en tiempo discreto. A

continuación se hace referencia a los modelos ARX, ARMAX, OE y BJ (Arafet et al, 2008).

E.2.1 Modelo ARX

De manera explícita el modelo ARX se expresa según la Ecuación 112.

$$y(t)+a_1y(t-1)+...+a_{na}y(t-na)=b_1u(t-nk)+b_2u(t-nk-1)+...+b_{nb}u(t-nk-nb+1) \quad (112)$$

Las incógnitas a y b son los coeficientes de la función de transferencia discreta y se obtienen según:

- Mínimos Cuadrados: Minimiza la suma de los cuadrados de la parte derecha menos la parte izquierda de la Ecuación 112 con respecto a los coeficientes a y b.
- Variable Instrumental: Se determina a y b de manera tal que el error entre las partes derecha e izquierda de la Ecuación 112 no correlaciona con alguna combinación lineal de la entrada.

E.2.2 Modelo ARMAX

La estructura ARMAX (*AutoRegressive Moving Average eXogen)* se introduce el polinomio C(q) al modelo ARX dado por la Ecuación 112, donde C se puede expresar según la Ecuación 114.

$$A(q)y(t)=B(q)u(t-nk)+C(q)e(t) \quad (113)$$

$$C(q)=1+c_1q^{-1}+...+c_{nc}q^{-nc} \quad (114)$$

Las incógnitas son los coeficientes del modelo discreto, cuyas soluciones se obtienen por predicción del error con el método de máxima verosimilitud, que también es utilizado en los modelos posteriores.

E.2.3 Modelo OE

La estructura del Modelo OE (Output-Error) es dada por la Ecuación 115

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+e(t) \quad con \quad F(q)=1+f_1q^{-1}+...+f_{nf}q^{-nf} \quad (115)$$

E.2.3 Modelo BJ

Es llamado la estructura de Box-Jenkins, ésta dada por la Ecuación 116.

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t) \quad con \quad D(q)=1+d_1q^{-1}+...+d_{nd}q^{-nd} \quad (116)$$

Todos los anteriores modelos son casos particulares de la estructura general dada por la Ecuación 117.

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t-nk) + \frac{C(q)}{D(q)}e(t) \quad (117)$$

Donde:

- La estrutura AR corresponde con nb=nf=nc=nd=0 y además u(t) =0.
- La estructura ARX se obtiene haciendo nd=nc=nf=0.
- La estructura ARMAX corresponde a nf=nd=0.
- La estructura OE corresponde a na=nc=nd=0.
- La estructura BJ corresponde a na=0

E.3 IDENTIFICACIÓN DEL MIC955

Por medio del método de identificación propuesto por Smith se procedió a identificar los diferentes sectores de trabajo del Módulo MIC955 con el fin de comprobar la existencia de no linealidades presentes en la planta, lo que por ende, no permite sacar un modelo exacto de orden sencillo de todo el rango de trabajo permitido.

Se procedió a identificar el modelo mediante la respuesta en lazo abierto del sistema, haciendo variaciones en forma de escalón en la entrada equivalentes al 10% del porcentaje de PWM, ya que la variación de grados centígrados entre escalones es menor y facilita su análisis grafico. La energía entregada al actuador esta dada en porcentaje de potencia. En la Tabla 11 se muestran los parámetros obtenidos modelando por medio de un sistema POMTM. Donde la columna Porcentaje de Entrada PWM, se refiere a los cambios en la entrada del sistema dados en porcentajes, la columna Porcentaje de Potencia se refiere a la potencia (variable manipulada) entregada al proceso termico, la columna Ganancia se refiere a los valores de ganancia parcial del sistema dada en °C/%potencia, la columna constante de tiempo se refiere a los valores de Tao(τ) calculados del modelo y la columna Tm muestra los valores de tiempo muerto calculados.

De la Tabla 11 se puede concluir que a medida que la temperatura aumenta la ganancia del sistema disminuye consumiendo más potencia.

Tabla 11: Parámetros obtenidos por el método de Smith para el módulo MIC955

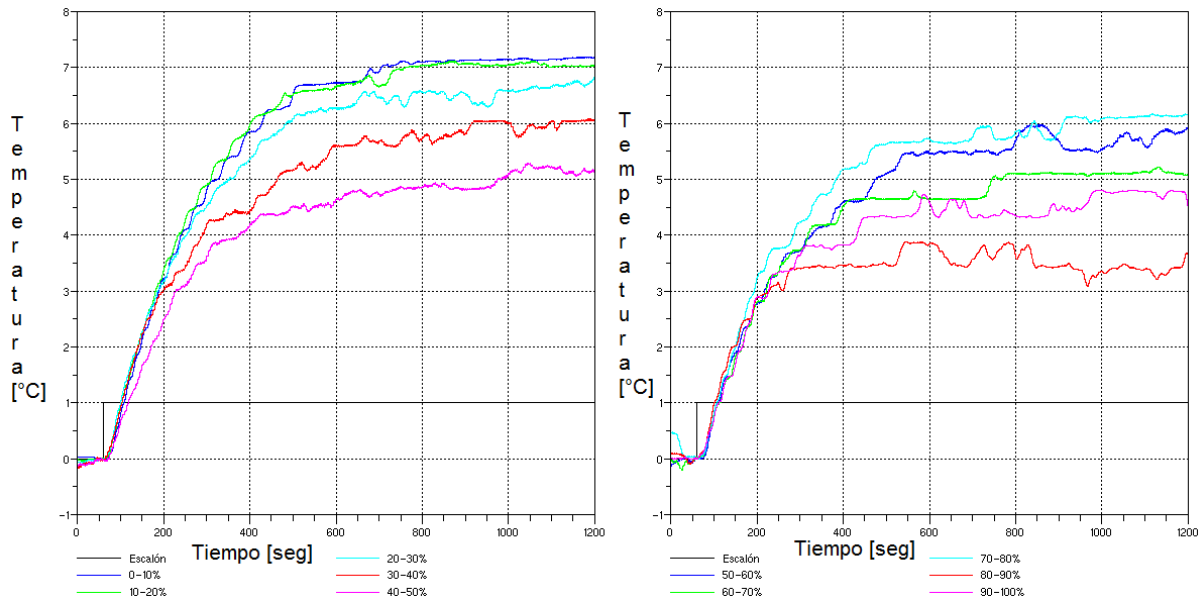
Porcentaje de Entrada PWM (%PWM)	Porcentaje de Potencia (%)	Ganancia (kp) (°C / %Pot)	Contante de Tiempo (τ) (seg)	Tiempo muerto (Tm) (seg)
0-10	0-1	7.1	191.95	23.15
10-20	1-4	2.33	182.40	19.00
20-30	4-9	1.32	196.20	7.80
30-40	9-16	0.85	215.40	7.50
40-50	16-25	0.57	187.35	21.25
50-60	25-36	0.52	198.15	9.65

Tabla 12: Continuación

Porcentaje de Entrada PWM (%PWM)	Porcentaje de Potencia (%)	Ganancia (kp) (°C / %Pot)	Contante de Tiempo (τ) (seg)	Tiempo muerto (Tm) (seg)
60-70	36-49	0.39	142.80	25.00
70-80	49-64	0.40	209.70	7.45
80-90	64-81	0.20	86.40	11.60
90-100	81-100	0.25	154.50	13.30

La figura 70 muestra las respuestas producidas por la planta en lazo abierto con entradas tipo escalón. Para su comparación las gráficas coinciden en t = 60 segundos, tiempo en el cual se produce el cambio escalonado de la señal de entrada de la planta.

Figura 70: Respuesta de lazo abierto del Módulo MIC995 ante un escalón en las diferentes zonas de trabajo



La gráfica está dividida en dos zonas, a la izquierda la señal de entrada varía en pasos de 10% del 0 a 50% de porcentaje PWM entregado al calentador equivalente a un cambio de 0 a 25% de la energía total entregada por el actuador, y a la derecha la señal de entrada varia de igual manera desde el 50 al 100% de porcentaje PWM equivalente al cambio de 25 a 100% de energía entregada al proceso termico.

En figura 70 puede apreciarse como la ganancia del sistema, percibida por la variación de temperatura medida entre el instante de tiempo donde se produce el escalón y el tiempo donde la variable se estabiliza, decrece a medida que el porcentaje de energía entregado al actuador aumenta. Sin embargo, la reducción

en el valor de la ganancia no es constante ni lineal, existiendo regiones donde éste valor es menor a valores encontrados en zonas de mayor porcentaje de energía entregado al actuador.

E.4. VALIDACIÓN DE LOS MODELOS IDENTIFICACIÓN DEL MIC955 EN EL PUNTO DE OPERACIÓN ELEGIDO

Los métodos de sintonización varían de acuerdo a la dinámica del sistema a controlar. Cada autor especifica las condiciones que el comportamiento dinámico de la planta debe cumplir para el buen desempeño del controlador, por ende, muchos de los autores sugieren un método de identificación específico con el fin de evaluar los parámetros funcionales del sistema bajo las condiciones adecuadas.

Haciendo el estudio de diferentes métodos de sintonización y eligiendo de acuerdo a sus principales características y a las restricciones que los mismos imponen, se eliminaron sistemáticamente diferentes métodos de sintonización que a su vez redujeron el número de métodos de identificación útiles para el desarrollo de un sistema de control optimo para el módulo MIC955. En conclusión se escogieron dos métodos de identificación que gracias a su fácil implementación son comunes entre muchos de los autores y permiten desarrollar la mayoría de técnicas de sintonización óptimas para las características lentas del proceso de temperatura del MIC955.

Los métodos elegidos para la identificación de la planta son: El método de Alfaro y el método de Smith, para la identificación de modelos de primer orden más tiempo muerto, a través de curva de reacción del proceso en lazo abierto, con la elección de dos puntos. Los modelos identificados son dados por la Ecuación 118 para Alfaro y la Ecuación 119 para Smith.

$$G_p(s) = \frac{2.75e^{-9.709s}}{164.25s+1} \tag{118}$$

$$G_p(s) = \frac{2.75e^{-12.5s}}{154.5s+1} \tag{119}$$

Una vez identificados lo métodos se procedió a verificar su desempeño bajo los índices de desempeño IEA y IEC comparando las señales de temperatura esperadas, con nuevos datos tomados de la planta real. La Tabla 12 muestra las valores de los índices de desempeño para los métodos escogidos de identificación

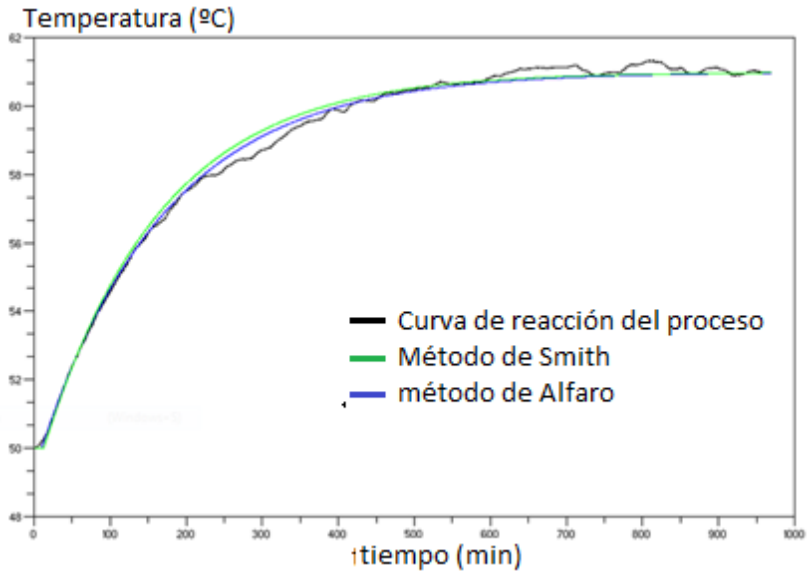
Tabla 13: Evaluación mediante índices de desempeño integral de los modelos obtenidos.

Metodo de identificacion	IEA	ISE
Método de Alfaro	1403.5764	354.1979
Método de Smith	1872.6797	592.4214

De a cuerdo a los valores mostrados en la Tabla 12, el método de Smith tiene un porcentaje de error de IEA = 33.42% y ISE = 67.25% mayor en comparación con el método de Alfaro para este caso de estudio.

La figura 71 muestra las curvas de reacción de los modelos simulados de Alfaro (azul) y Smith (verde) en comparación con los datos reales (negro) tomados del proceso tomados en el punto de operación deseado en el sistema para una entrada de escalón de 20% del total de energía entrada al actuador.

Figura 71: Comparación de los modelos POMTM y Temperatura real



ANEXO G RESULTADOS PRUEBAS OPERACIONALES

En el capítulo 5 se presentó el procedimiento para realizar RCP que se desarrolló en el presente caso de estudio, en este procedimiento se propone una serie de pruebas operacionales de los prototipos de control para lazos de control regulatorio y servocontrolado. En este anexo se presentan las respuestas y el análisis de los resultados obtenidos en las pruebas operacionales para el lazo de control regulatorio.

G.1. PRUEBAS OPERACIONALES DEL CONTROLADOR PID CLÁSICO OPERADO EN LAZOS DE CONTROL REGULATORIO.

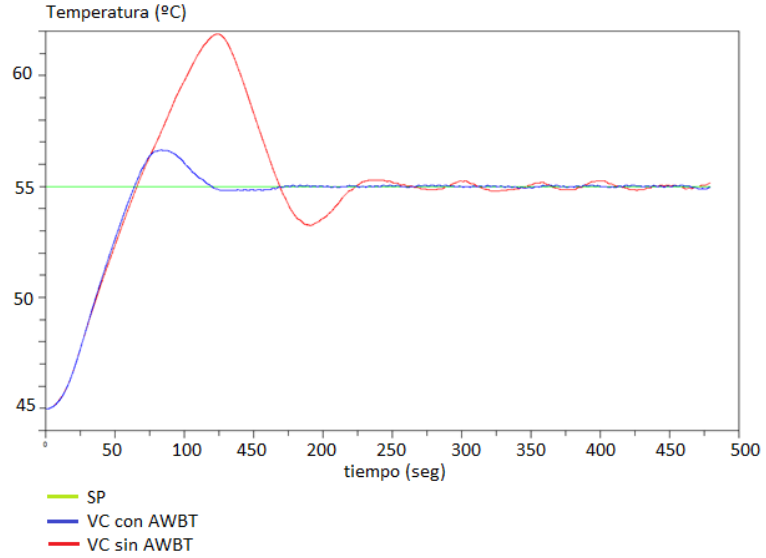
En el cuadro 1, se presenta el método de sintonización Minimun IAE Huang – Chao, para el controlador PID clásico y el valor de las constantes de sintonización calculadas para el modelo POMTM que se obtuvo mediante el método de identificación de Smith.

Cuadro 1: Método de sintonización *Minimum* IAE Huang – Chao (1982) para lazos de control regulatorio. Fuente (O'Dwyer, 2009)

MINIMUN IAE HUANG – CHAO (1982)			
Modo de operación controlador PID:		Regulador	
Criterio de desempeño:		*minimun* IAE	
Método de identificación:		Método de Smith	
Modelo del proceso:		$G_p(s)=\frac{2.75e^{-12.5s}}{1+154.5s}$	
Validez:		N/A	
Ecuaciones de sintonización y estimación de parámetros del controlador:			
$K_c=\frac{0.817}{K_m}\left(\frac{\tau}{t_m}\right)^{0.982}$	$K_c=3.5$	$T_i=0.903\tau\left(\frac{t_m}{\tau}\right)^{0.780}$	$T_i=19.62$
$T_d=0.602\tau\left(\frac{t_m}{\tau}\right)^{0.954}$	$N=8\ T_d=8.44$	$T_t=\sqrt{T_iT_d}$	$T_t=12.86$

La figura 72 muestra la respuesta del controlador PID Clásico sintonizado mediante el método minimun IAE de Huang – Chao (1982), para un SP de 55 ºC (traza verde). La traza roja corresponde VC cuando la protección AWBT se encuentra desactivada y la traza azul corresponde a la VC cuando se activa la protección AWBT.

Figura 72: Respuesta del controlador PID Clásico sintonizado mediante el método IAE de Huang – Chao, 1982.



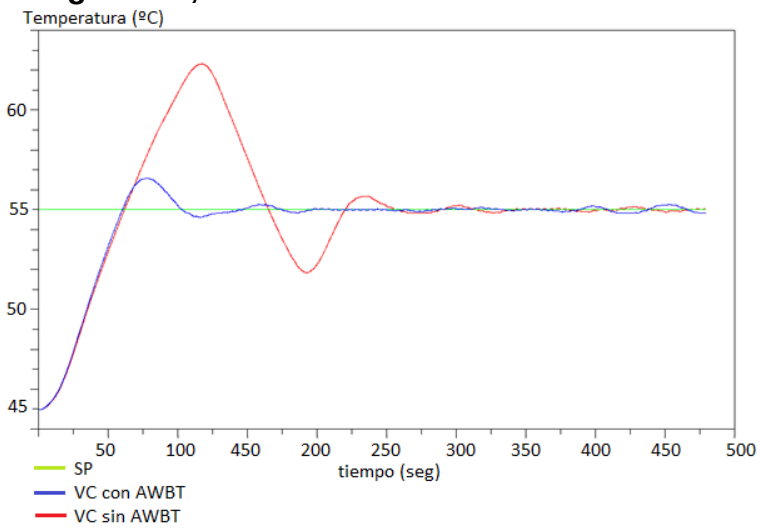
En el cuadro 2, se presenta el método de sintonización Minimun IAE de Edgar et al (1997) para el controlador PID clásico y el valor de las constantes de sintonización calculadas para el modelo POMTM que se obtuvo mediante el método de identificación de Smith.

Cuadro 2: Método de sintonización *Minimum* IAE Edgar et al (1997) para lazos de control regulatorio. Fuente (O'Dwyer, 2009).

MINIMUN IAE - EDGAR ET AL (1997)			
Modo de operación controlador PID:		**Modo de operación controlador PID:**	
Criterio de desempeño:		**Criterio de desempeño:**	
Método de identificación:		**Método de identificación:**	
Modelo del proceso:		**Modelo del proceso:**	
Validez:		**Validez:**	
Ecuaciones de sintonización y estimación de parámetros del controlador:			
$K_c = \frac{0.94\tau}{K_m t_m}$	$K_c = 4.28$	$T_i = 1.5t_m$	$T_i = 18.75$
$T_d = 0.55t_m$	$T_d = 6.875$ $N = 10$	$T_t = \sqrt{T_i T_d}$	$T_t = 11.35$

La figura 73 muestra la respuesta del controlador PID Clásico sintonizado mediante el método minimun IAE de Edgar et al, (1997) – Chao (1982), para un SP de 55 ºC (traza verde). La traza roja corresponde VC cuando la protección AWBT se encuentra desactivada y la traza azul corresponde a la VC cuando se activa la protección AWBT.

Figura 73: Respuesta del controlador PID Clásico sintonizado mediante el método IAE de Edgar et al, 1997.



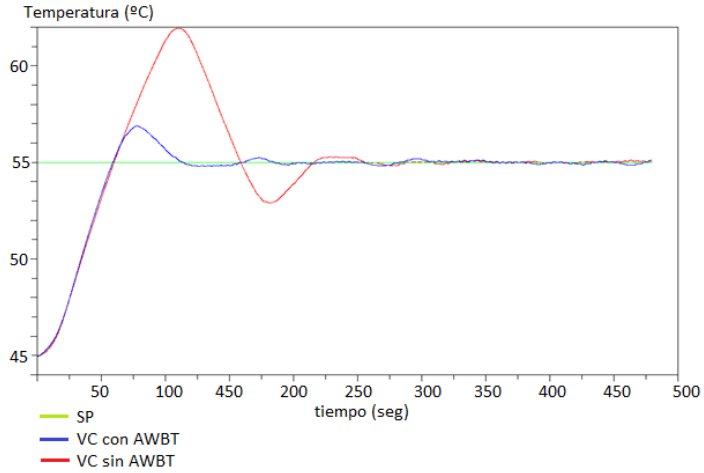
En el cuadro 3, se presenta el método de sintonización Minimun ISE de Huang – Chao, (1982) para el controlador PID clásico y el valor de las constantes de sintonización calculadas para el modelo POMTM que se obtuvo mediante el método de identificación de Smith.

Cuadro 3: Método de sintonización *Minimum* ISE Huang – Chao (1982) para lazos de control regulatorio. Fuente (O'Dwyer, 2009).

MINIMUN ISE HUANG – CHAO (1982)			
Modo de operación controlador PID		Regulador	
Criterio de desempeño:		*minimun* ISE	
Método de identificación:		Método de Smith	
Modelo del proceso:		$G_p(s)=\frac{2.75e^{-12.35s}}{1+153.15s}$	
Validez:		N/A	
Ecuaciones de sintonización y estimación de parámetros del controlador:			
$K_c=\frac{1.101}{K_m}\left(\frac{\tau}{t_m}\right)^{0.881}$	$K_c=3.66$	$T_i=1.134\tau\left(\frac{t_m}{\tau}\right)^{0.883}$	$T_i=19.02$
$T_d=0.563\tau\left(\frac{t_m}{\tau}\right)^{0.881}$	$T_d=9.5$ $N=8$	$T_t=\sqrt{T_iT_d}$	$T_t=13.43$

La figura 74 muestra la respuesta del controlador PID Clásico sintonizado mediante el método minimun ISE de Huang – Chao (1982), para un SP de 55 ºC (traza verde). La traza roja corresponde VC cuando la protección AWBT se encuentra desactivada y la traza azul corresponde a la VC cuando se activa la protección AWBT.

Figura 74: Respuesta del controlador PID Clásico sintonizado mediante el método ISE de Huang – Chao, 1982.



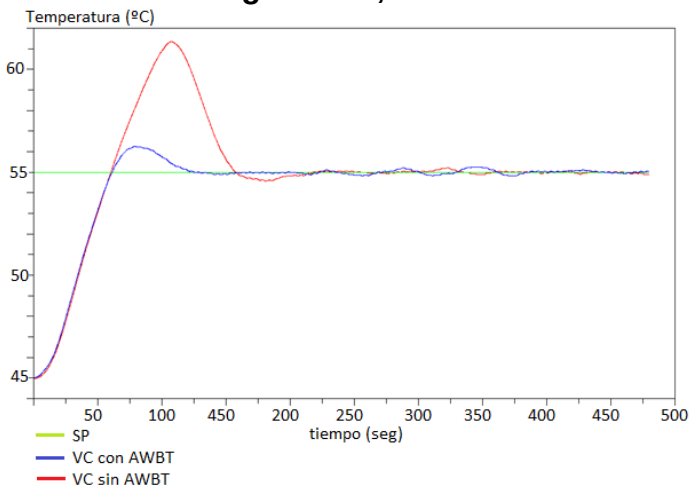
En el cuadro 4, se presenta el método de sintonización Minimun ITAE de Huang – Chao, (1982) para el controlador PID clásico y el valor de las constantes de sintonización calculadas para el modelo POMTM que se obtuvo mediante el método de identificación de Smith.

Cuadro 4: Método de sintonización *Minimum* ITAE Huang – Chao (1982) para lazos de control regulatorio. Fuente (O'Dwyer, 2009).

<table>
<tr><th colspan="4">MINIMUN ITAE HUANG – CHAO (1982)</th></tr>
<tr><td colspan="2">Modo de operación controlador PID</td><td colspan="2">Regulador</td></tr>
<tr><td colspan="2">Criterio de desempeño:</td><td colspan="2">minimun ITAE</td></tr>
<tr><td colspan="2">Método de identificación:</td><td colspan="2">Método de Smith</td></tr>
<tr><td colspan="2">Modelo del proceso:</td><td colspan="2">$G_p(s)=\frac{2.75e^{-12.5s}}{1+154.5s}$</td></tr>
<tr><td colspan="2">Validez:</td><td colspan="2">N/A</td></tr>
<tr><td colspan="4">Ecuaciones de sintonización y estimación de parámetros del controlador:</td></tr>
<tr><td>$K_c=\frac{0.745}{K_m}\left(\frac{\tau}{t_m}\right)^{1.036}$</td><td>$K_c=3.66$</td><td>$T_i=0.771\tau\left(\frac{t_m}{\tau}\right)^{0.595}$</td><td>$T_i=26.68$</td></tr>
<tr><td>$T_d=0.597\tau\left(\frac{t_m}{\tau}\right)^{1.006}$</td><td>$T_d=7.35$
$N=8$</td><td>$T_t=\sqrt{T_iT_d}$</td><td>$T_t=14$</td></tr>
</table>

La figura 75 muestra la respuesta del controlador PID Clásico sintonizado mediante el método minimun ITAE de Huang – Chao (1982), para un SP de 55 ºC (traza verde). La traza roja corresponde VC cuando la protección AWBT se encuentra desactivada y la traza azul corresponde a la VC cuando se activa la protección AWBT.

Figura 75: Respuesta del controlador PID Clásico sintonizado mediante el método mimimun ITAE de Huang – Chao, 1982.



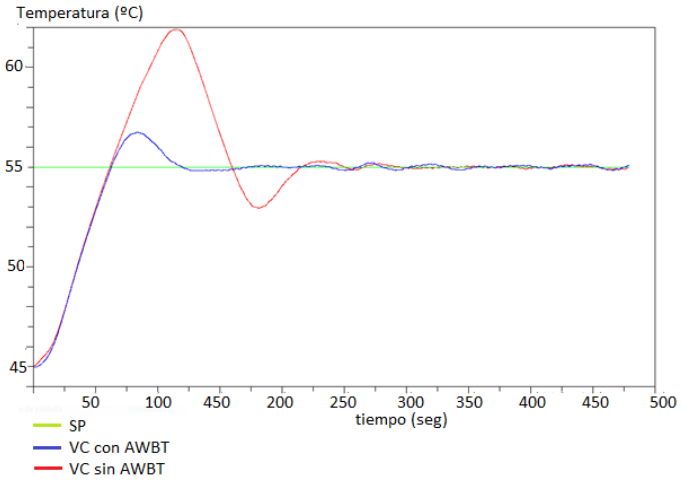
En el cuadro 4, se presenta el método de sintonización Minimun ITSE de Huang – Chao, (1982) para el controlador PID clásico y el valor de las constantes de sintonización calculadas para el modelo POMTM que se obtuvo mediante el método de identificación de Smith.

Cuadro 5: Método de sintonización *Minimum* ITSE Huang – Chao (1982) para lazos de control regulatorio. Fuente (O'Dwyer, 2009).

MINIMUN ITSE HUANG – CHAO (1982)			
Modo de operación controlador PID		Regulador	
Criterio de desempeño:		*minimun* ITSE	
Método de identificación:		Método de Smith	
Modelo del proceso:		$G_p(s)=\frac{2.75e^{-12.5s}}{1+154.5s}$	
Validez:		N/A	
Ecuaciones de sintonización y estimación de parámetros del controlador:			
$K_c=\frac{0.994}{K_m}\left(\frac{\tau}{t_m}\right)^{0.907}$	$K_c=3.53$	$T_i=1.032\tau\left(\frac{t_m}{\tau}\right)^{0.869}$	$T_i=17.93$
$T_d=0.570\tau\left(\frac{t_m}{\tau}\right)^{0.884}$	$T_d=9.54$ $N=8$	$T_t=\sqrt{T_iT_d}$	$T_t=13.07$

La figura 75 muestra la respuesta del controlador PID Clásico sintonizado mediante el método minimun ITAE de Huang – Chao (1982), para un SP de 55 ºC (traza verde). La traza roja corresponde VC cuando la protección AWBT se encuentra desactivada y la traza azul corresponde a la VC cuando se activa la protección AWBT.

Figura 76: Respuesta del controlador PID Clásico sintonizado mediante el método mimimun ITSE de Huang – Chao, 1982.



La tabla 13 muestra los resultados de la evaluación del desempeño del controlador PID clásico sintonizado con los diferentes métodos. Esta evaluación se lleva a cabo mediante criterios temporales como el tiempo de subida (TR), el tiempo de establecimiento (TS), el máximo sobreimpulso (MP) y criterios integrales (ver sección 4.2.2) como la integral del error absoluto (IAE), la integral del cuadrado del error (ISE), la integral del error absoluto multiplicado por el tiempo (ITAE) y la integral del cuadrado del error multiplicada por el tiempo (ITSE).

Tabla 14: Evaluación del desempeño del controlador PID clásico, sintonizado con los diferentes métodos.

CONTROLADOR	Controlador Clásico			**MODO DE OPERACIÓN**			Control Regulatorio	
METODO DE SINTONIZACIÓN	**AWBT**	**TR (seg)**	**TS (seg)**	**MP(%)**	**IAE**	**ISE**	**ITAE**	**ITSE**
Minimun IAE – Huang and Chao (1982)	NO	65.8	206.5	12.51	429.6	122.3	38494	7691
Minimun IAE – Huang and Chao (1982)	SI	63.9	99.3	2.98	217.3	70.7	8285.6	1309.5
Minimun IAE – Edgar et al (19997)	NO	61.6	211.9	13.3	457.5	130.97	42310	8905.4
Minimun IAE – Edgar et al (19997)	SI	60.2	88.8	2.86	210.3	66.66	10008	1181.5
Minimun ISE – Huang and Chao (1982)	NO	59.8	200.5	12.61	413.4	118.9	34126	7070.9
Minimun ISE – Huang and Chao (1982)	SI	58.8	93.8	3.445	211.52	66.357	9282.4	1188.3
Minimun ITAE – Huang and Chao (1982)	NO	60.1	145.5	11.53	355.77	106.15	24650.	5265.3

Tabla 13: Continuación

CONTROLADOR	Controlador Clásico			**MODO DE OPERACIÓN**			Control Regulatorio	
METODO DE SINTONIZACIÓN	**AWBT**	**TR (seg)**	**TS (seg)**	**MP(%)**	**IAE**	**ISE**	**ITAE**	**ITSE**
Minimun ITAE – Huang and Chao (1982)	SI	61.0	90.4	2.27	207.02	65.90	9356.2	1143.2
Minimun ITSE – Huang and Chao (1982)	NO	61.4	198.2	12.54	412.09	118.19	34294	7177.8
Minimun ITSE – Huang and Chao (1982)	SI	62.6	98.8	3.16	220.77	69.65	10150	1288.0

Como se puede observar en la tabla 13, el método mínimum ITAE de Huang – Chaos, provee los valores mínimos para MP, IAE e ISE. El tiempo de subida no es un factor determinante en esta evaluación, pues la diferencia entre los métodos nunca supera los 3 segundos y debido a que el proceso térmico es lento, no constuye una falla grave en el sistema de control. Tambien es posible observar que aunque el método miminum IAE de Huang and Chao provee el menor ITAE, presenta un sobreimpulso mayor que el método mínimum ITAE de Huang – Chaos. Es importante notar que la protección AWBT mejora notablemente la corrección generada por el controlador.

ANEXO H: GUIAS DE PRÁCTICA PARA CONTROL DE TEMPERATURA EN EL MÓDULO MIC955

H.1. DESCRIPCIÓN FÍSICA DEL MÓDULO MIC 955 DE LA FEEDBACK PARA PRÁCTICAS DE CONTROL DE TEMPERATURA:

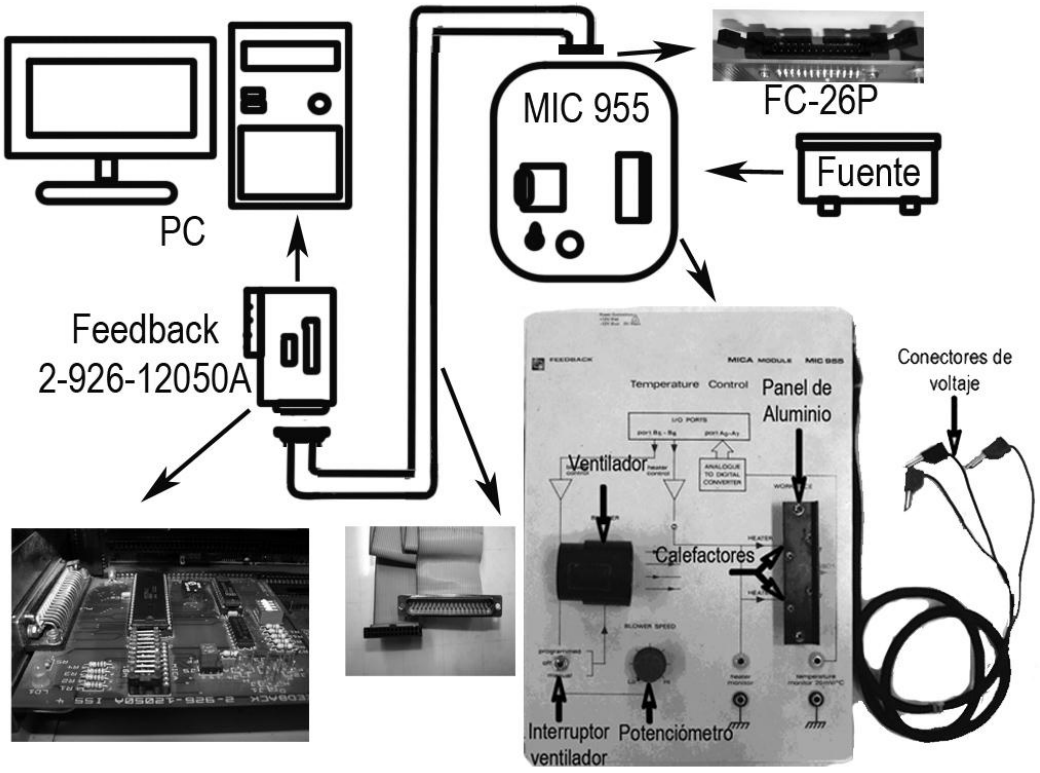
Básicamente el módulo MIC 955 está compuesto por dos unidades, tal como se muestra en la figura 77:

- Un panel compuesto por una tira de aluminio negro anodizado (*Workpiece*), dos resistencias que sirven de calefactores (*Heater*), las cuales pueden ser activadas desde un PC y una resistencia de banda de platino al aguafuerte que es utilizada como sensor de temperatura.
- Un ventilador refrigerador (*Blower*) que tiene tres modos de configuración (programado, desactivado y manual), los cuales pueden activarse mediante un interruptor (*switch*). En modo manual la velocidad del ventilador varía con el movimiento de un potenciómetro (*Blower Speed*). En modo programado el ventilador se activado desde el PC, a través de un bit conectado a un amplificador, entregando mayor potencia que en el modo manual (Feedback, 1983).

El sensor de temperatura está conectado a un circuito puente de Wheatstone DC. La función de este sensor es convertir las variaciones de resistencia a variaciones de voltaje, las cuales que serán amplificadas y posteriormente equilibradas a 0ºC, obteniendo una sensibilidad de 20mV/ºC. Ésta sensibilidad puede ser monitoreada con un osciloscopio desde el panel frontal del módulo. La señal analógica resultante está conectada a un conversor analógico digital (ADC) ZN427, el cual posee 8 bits de salida, los cuales se transmiten al PC al recibir una orden. Las señales de activación del ventilador y de los calefactores están acopladas a un circuito amplificador de potencia. El módulo se alimenta mediante una fuente de voltaje externa a través de tres conectores:

- +12v (rojo).
- -12v (azul).
- Tierra (negro).

Figura 77: Módulo MIC 955.



Fuente: Propia

El módulo MIC 955 cuenta con dos puertos de E/S:

- El puerto A envía el valor de la temperatura entregado por el conversor ADC en una palabra de 8 bits.
- El puerto B recibe los comandos de activación de las señales para los calentadores, el ventilador en modo programado y del conversor ADC.

La Tabla 14 muestra las asignaciones de puerto y de bits para realizar las comunicaciones entre el PC y el conector FC-26P macho del módulo.

Adicionalmente el módulo MIC 955 tiene dos salidas a tierra en los pines 13, 26 y una señal de fin de conversión generada por el ADC en el pin 23 del conector FC26-P.

Para realizar la acción de conversión de un dato, el bit 7 del puerto B debe recibir un impulso positivo generado por una secuencia binaria 1-0-1, el conversor devuelve un valor digital entre 0 y 255 que RTAI-Lab toma y convierte a un valor en grados centígrados mediante un método de linealización, el cual es ejecutado dentro del bloque *MIC955*.

Tabla 15: Configuración de puertos del módulo MIC 955.

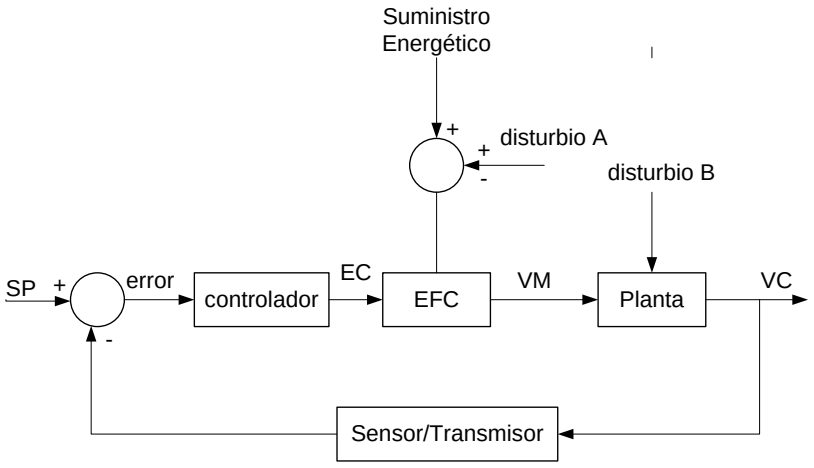
Nº de bit	Puerto A (Entrada al PC)	Pin	Puerto B (Salida del PC)	Pin
0 (LSB)	ADC salida del conversor analógico-digital, bit 0	21		
1	ADC salida del conversor analógico-digital, bit 1	20		
2	ADC salida del conversor analógico-digital, bit 2	19		
3	ADC salida del conversor analógico-digital, bit 3	18		
4	ADC salida del conversor analógico-digital, bit 4	17		
5	ADC salida del conversor analógico-digital, bit 5	16	Señal de ventilador (activado = 1)	3
6	ADC salida del conversor analógico-digital, bit 6	15	Señal de calentador (activado = 1)	2
7(MSB)	ADC salida del conversor analógico-digital, bit 7	14	Inicio de conversión de ADC	1

H.2. DIAGRAMAS DE BLOQUES

Los sistemas de control a menudo se representan mediante diagramas de bloques como se muestra en la figura 78, los cuales indican las interrelaciones existentes entre los diferentes componentes del sistema. En un diagrama de bloques las señales del sistema se enlazan mediante bloques funcionales. Un bloque funcional o bloque, es un símbolo de la operación matemática que el sistema produce a la salida sobre la señal de entrada. Una flecha hacia adentro indica la entrada y la que se aleja indica la salida. La magnitud de la señal de salida del bloque será la señal de entrada multiplicada por la función de transferencia del bloque.

Un elemento importante de los diagramas de bloques es el punto de suma. Los símbolos “+” o “–” indican si esa entrada se suma o se resta.

Figura 78: Diagrama de bloques de un sistema de control realimentado



Fuente: Notas de clase, curso Instrumentación Industrial, Ingeniero Juan Fernando Flórez, Ingeniería en Automática Industrial, Universidad del Cauca, 2010

H.3. CONTROL ON/OFF

La selección del controlador para una determinada aplicación depende de la complejidad de las tareas de control para dicha aplicación, por ejemplo para aplicaciones sencillas se puede utilizar un control on – off; este tipo de controlador es la forma más simple de control realimentado. Se caracteriza por que el esfuerzo de control sólo puede cambiar entre dos valores 100 % On (Activado) o 100 % Off (Desactivado). En términos matemáticos la ley de control ON/OFF está representada por la siguiente ecuación:

$$U_{(t)} = \begin{cases} U_{\max} & \forall\ e_{(t)} > 0 \\ U_{\min} & \forall\ e_{(t)} < 0 \end{cases} \qquad (120)$$

La histéresis en este tipo de controladores es una propiedad que previene que la salida no conmute demasiado rápido, ayudando de esta forma a alargar la vida útil de los elementos finales de control. La histéresis se define como la diferencia entre el tiempo de encendido y el tiempo de apagado.

Esta práctica consiste en aplicar un control ON/OFF al módulo de control de temperatura MIC955 de la empresa Feedback, con el objetivo de identificar las características de este algoritmo de control y analizar la respuesta de este tipo de controladores en presencia de histéresis.

NOTA: Se recomienda prestar especial atención a las sugerencias, notas Y ejemplos.

5. Encienda el ordenador y espere a que cargue la ventana de inicio.
6. Digite “root” en la casilla usuario. Presione “Enter”.
7. Digite “feedback” en la casilla contraseña. Presione “Enter”.
8. Conecte el cable de conexión del módulo MIC955 a la fuente de alimentación.

IMPORTANTE: Conecte primero el conector negro (tierra) para evitar cualquier tipo de daño por energía almacenada dentro de la planta.

9. Conecte el enchufe de la fuente de poder al tomacorriente más cercano.
10. Encienda la fuente de poder.
11. En el escritorio haga doble clic sobre el icono de “Equipo”
12. Haga doble clic en el disco “Sistema de archivos”.
13. Haga doble clic en la carpeta “practicas”.
14. En el menú “Archivo” seleccione la opción “crear una carpeta”.

Nota: Esta acción también puede realizarse haciendo clic derecho y seleccionando la opción “crear una carpeta”.

15. Asígnele el nombre de “grupxx”.

NOTA: “xx” representa el número de su grupo. **Ejemplo:** “grupo00”

16. Copie los archivos “controlAWBT.cos”, “controlONOFF.cos”, “controlPID.cos” a la carpeta que acabó de crear.

IMPORTANTE: ¡copie! no mueva.

17. Haga clic sobre el icono del “Terminal”, ubicado en la barra de accesos directos que se observa en la figura 79.

Figura 79: Icono del Terminal en la barra de accesos directos de GNU Linux Fedora 7

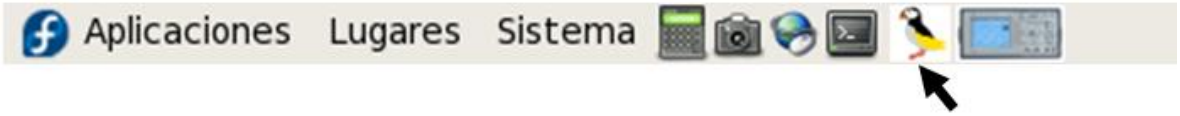
18. En la ventana del Terminal digite el siguiente comando:

cd /practicas/grupxx

Presione “Enter”.

19. Haga clic sobre el icono de “scilab”, ubicado en la barra de accesos directos.

Figura 80: Icono de Scilab en la barra de accesos directos de GNU Linux Fedora 7



Nota: Maximice la ventana de “scilab”.

20. En la ventana de “scilab” digite los siguientes comando:

cd /practicas/grupxx

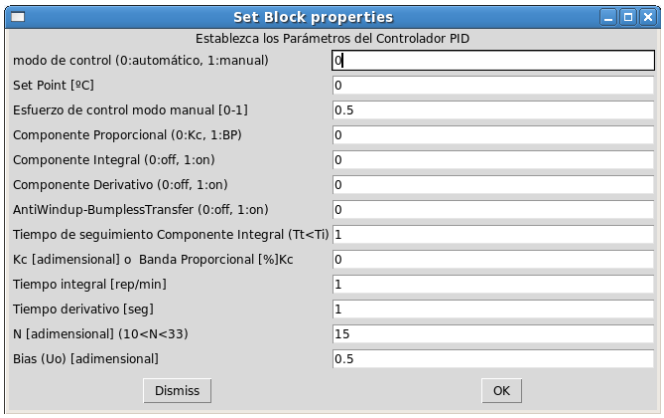
Presione “Enter”

scicos controlONOFF.cos

Presione “Enter”

El comando anterior abrirá la siguiente ventana:

Figura 81: Ventana “controlONOFF”

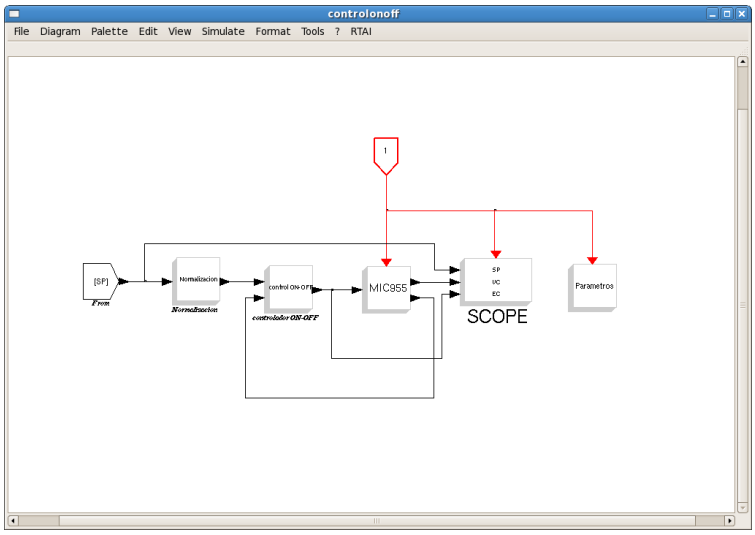


21. Haga doble clic sobre el súper bloque, podrá observar el diagrama de bloques del sistema de control on-off que se muestra en la figura 82, el cual está constituido por los siguientes bloques:

- **Bloque de comunicación real con la planta (MIC955).** Este bloque internamente trae los bloques de entrada y salida para el módulo: un bloque para los actuadores (ventilador y calefactor) y un bloque transmisor(cuyos valores por defecto son: **cero**: 45 ºC, **span**: 20 ºC)
- **Bloque de normalización de señales (normalización):** este bloque se encarga de transformar las señales de entrada a valores normalizados entre 0 y 1.
- **Bloque controlador On-Off (control ON-OFF):** este bloque contiene el algoritmo de control ON-OFF
- **Bloque de configuración sistema de control (Parametros):** permite configurar los parámetros del sistema tales como set point, habilitar y deshabilitar modos de control, selección de modo de control (automático-manual), etc.
- **Osciloscopio (SCOPE):** permite el monitoreo de las señales a través de xrtailab.

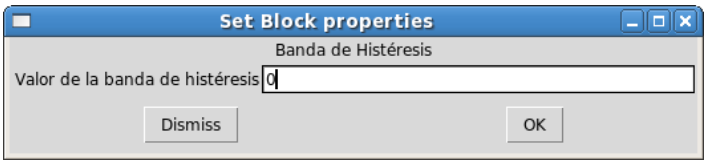
Nota: esta acción también puede realizarse haciendo clic derecho sobre el bloque y escogiendo la opción “Open/Set”.

Figura 82: Ventana “controlONOFF”



22. Haga doble clic sobre el bloque “control ON-OFF” y establezca el valor de histéresis en 0. Este valor será modificado en línea para observar el efecto de la histéresis.

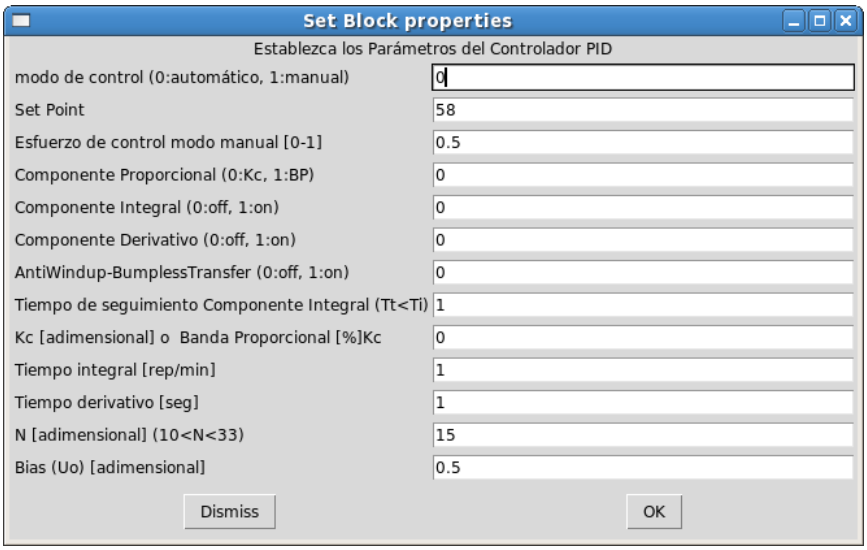
Figura 83: Configuración bloque “control ON-OFF”



Haga clic en “OK”.

23. Haga doble clic sobre el bloque “Parámetros” y establezca el SP en el valor intermedio del rango del transmisor, es decir 55 grados.

Figura 84: configuración bloque "Parámetros"



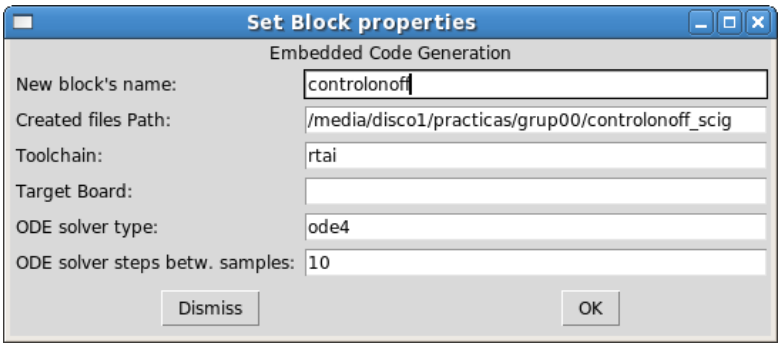
Haga clic sobre "OK".

Cierre la ventana "controlonoff".

24. En la ventana "controlONOFF", ubíquese en el menú "RTAI".

Haga clic sobre la opción "RTAI CodeGen". Aparecerá la siguiente ventana:

Figura 85: Generador de código para la tarea de control ON-OFF



Deje los parámetros por defecto.

Haga clic en "OK".

25. Para asegurarse que el ejecutable ha sido creado, revise la ventana de "scilab". Debería observar las ventana que se muestra en la figura 86.

Figura 86: Creación exitosa de la tarea en tiempo real

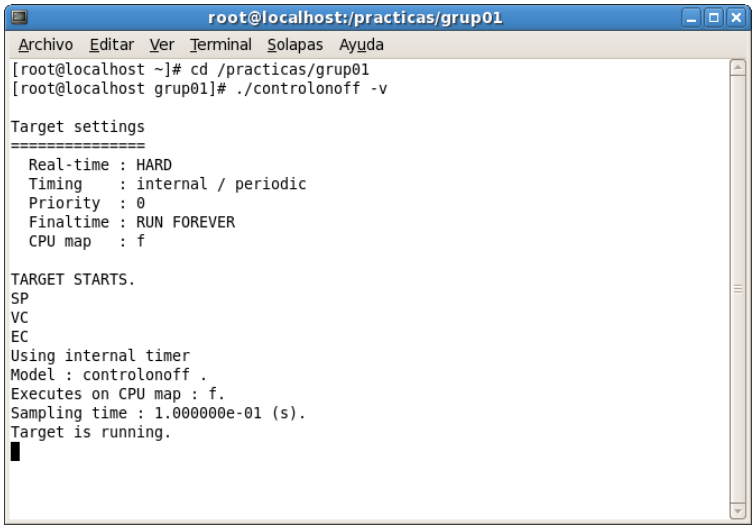
26. En la ventana del “Terminal” digite:

./controlonoff –v

Presione “Enter”

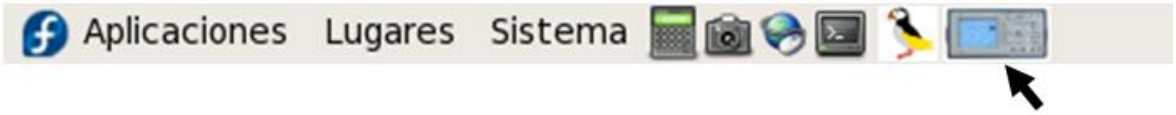
Si el ejecutable fue creado correctamente debería observarse la siguiente ventana.

Figura 87: Ejecución tarea de control ON-OFF



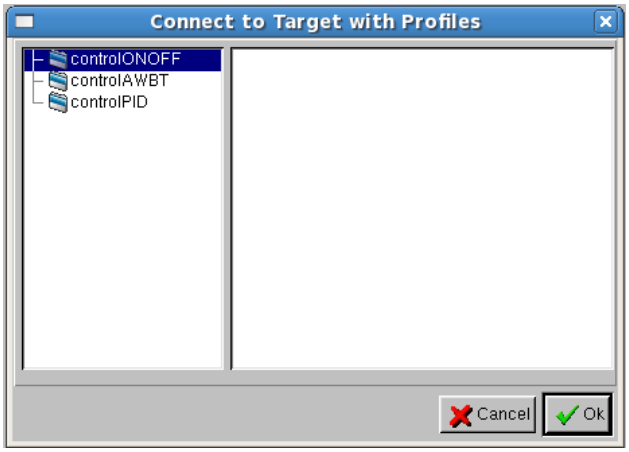
27. Haga clic sobre el icono de “RTAI-Lab Graphical User Interface” ubicado en la barra de accesos directos.

Figura 88: Icono de RTAI-Lab Graphical User Interface en la barra de accesos directos de GNU Linux Fedora 7



28. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “controlONOFF”. Debe abrir la ventana que se muestra en la figura 89.

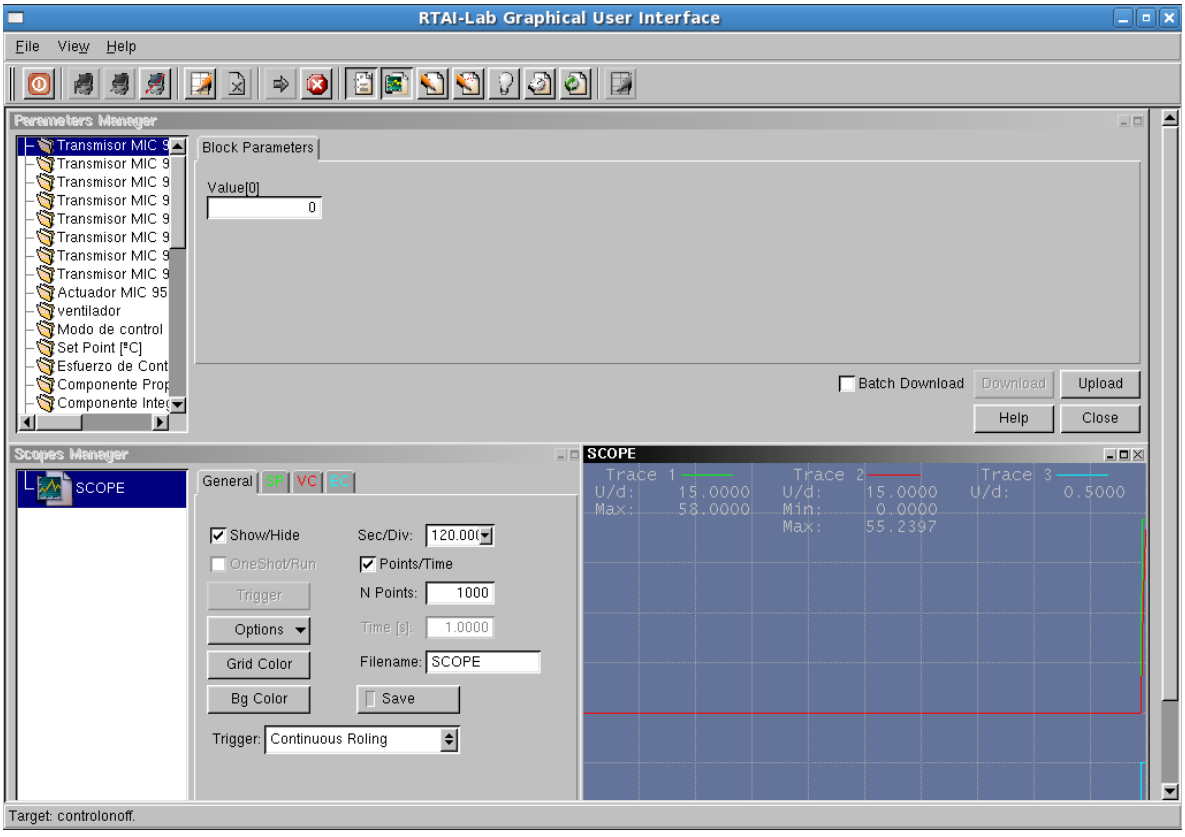
Figura 89: Selector de perfiles



Haga clic en “OK”

Debería abrir la ventana que se muestra en la figura 90:

Figura 90: Perfil Control ON-OFF



Note que ahora en la parte superior del área de trabajo de la ventana “RTAI-Lab Graphical User Inferface” se encuentra el “Parameters Manager” o “Administrador

de Parametros", el cual le permite modificar en línea los parámetros del sistema de control.

En la parte inferior izquierda se encuentra el "Scope Manager" o "Administrador de Osciloscopios" el cual le permite modificar los parámetros de los osciloscopios:

- En la pestaña "General" se encuentran las opciones para la monitorización de las señales en el osciloscopio virtual.
- La casilla "Show/Hide" muestra o desactiva el osciloscopio virtual.
- El divisor de tiempo "Sec/Div" permite cambiar el tiempo de división.
- El selector "Options" permite el manejo de los cursores en el osciloscopio virtual.
- El selector "Trigger" permite elegir el modo en el que funcionará el osciloscopio virtual (Continous Roling, Continous Overwrite,Hold, etc).

En la parte inferior derecha se encuentra el "Scope", el cual le permite monitorear las diferentes señales de interés. Muevas las barras vertical y horizontal de la ventana "RTAI-Lab Graphical User Interface".

La traza verde representa el set point, establecido en un valor de 55 ºC, unidades por división de 15 grados ºC. La traza roja representa la temperatura controlada, sus valores máximos, mínimo y el valor de los punteros Y1 y Y2. La traza azul claro es el esfuerzo de control ON-OFF a la salida del controlador.

Espere hasta que el sistema empiece a oscilar alrededor del set point. (Duración recomendada 240 segundos).

Anote el valor máximo de la traza roja (VC): ____

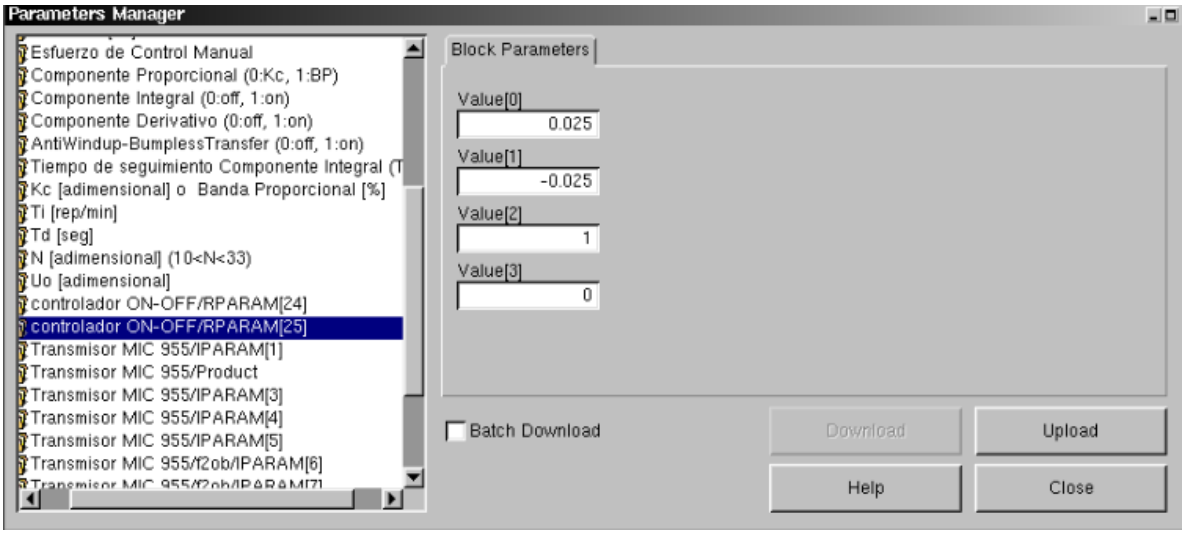
29. Cambie el valor de la histéresis a 2.5 %.

Diríjase al "Administrador de Parámetros", encuentre la opción "controlador ON-OFF/RPARAM[25]".

Establezca el valor de 0.025 a "Value[0]", presione "Enter". Tal como se muestra en la figura

Establezca el valor de -0.025 a "Value[1]", presione "Enter"

Figura 91: Cambiar el valor de la banda de histéresis



Esto cambia los límites superior e inferior de la banda de histéresis.

Espere a que la variable controlada oscile alrededor del set point durante 240 segundos.

Anote el valor máximo de la traza roja(VC).

Valor VC: ____

30. Repita el paso anterior para un valor de 5 %. Espere a que la variable controlada oscile alrededor del set point durante 240 segundos.

Anote el valor máximo de la traza roja(VC)

Valor VC: ____

31. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
32. Maximice la ventana del osciloscopio.
33. Ubique el puntero 2 del osciloscopio hasta el momento que la variable controlada (traza roja) alcance al set point (traza verde).
34. Ubique el puntero 1 en la parte inicial de la respuesta. La diferencia “dt” entre los tiempos que marcan el puntero 1 (t1) y el puntero 2 (t2) corresponde al tiempo de subida.

Tiempo de subida: _____.

35. Guarde una imagen de la respuesta del sistema bajo el nombre de “controlonoff”.

En la barra de accesos directos, haga clic sobre el icono de la cámara, seleccione “grab the current window” y luego haga clic sobre “take screenshot”. Luego escriba el nombre la imagen y guárdela por defecto en el escritorio.

36. Detenga la tarea de de control mediante el botón “stop real time code” de la ventana “RTAI-Lab Graphical User Interface”. Presione “Yes”.

Figura 92: “Boton Stop real time code”

37. Cierre la ventana “RTAI-Lab Graphical User Interface”
38. Cierre la ventana “controlONOFF”.
39. En la ventana de “scilab” presione la tecla “N”.
40. Abra la ventana “grupxx”
41. Cree una carpeta llamada “controlONOFF”
42. Mueva la imagen “controlonoff” que se encuentra en el escritorio a la carpeta “controlONOFF”
43. Responda las siguientes preguntas:

- ¿Cómo se comporta el proceso con histéresis de 0, 2.5 y 5 % del valor del set point?
- ¿Qué efecto produce la histerésis?
- ¿Cómo se comporta mejor el proceso con más o menos histéresis?
- Cuáles son las ventajas del controlador ON/OFF.
- Cuáles son las desventajas del controlador ON/OFF

H.4. PRÁCTICA 3: CONTROL P, PI y PID

El objetivo de esta práctica es familiarizarse y profundizar en el conocimiento de la operación de los controladores P, PI y PID, identificando las características de cada controlador a partir de la respuesta del proceso.

PARTE A. CONTROL PROPORCIONAL

1. En la ventana de “scilab” digite:

scicos controlPID.cos

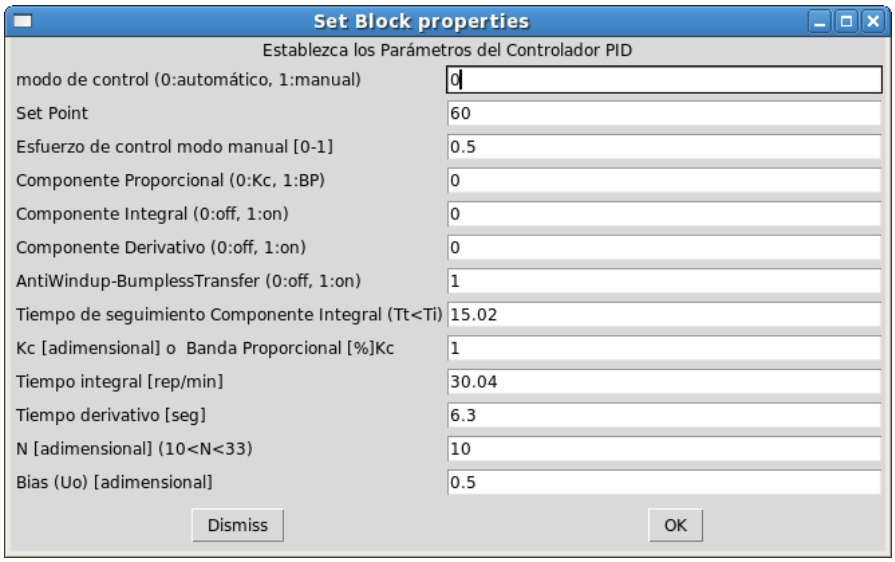
Presione “Enter”

2. En la ventana “controlPID”, Haga doble clic sobre el súper bloque, podrá observar el diagrama de bloques del sistema de control on-off, el cual está constituido por los siguientes bloques:

- **Bloque de comunicación real con la planta (MIC955).** Este bloque internamente trae los bloques de entrada y salida para el módulo: un bloque para los actuadores (ventilador y calefactor) y un bloque transmisor(cuyos valores por defecto son: **cero**: 28 ºC, **span**:60 ºC)
- **Bloque de normalización de señales (normalización):** este bloque se encarga de transformar las señales de entrada a valores normalizados entre 0 y 1.
- **Bloque controlador PID (control PID Ideal):** este bloque contiene el algoritmo de control PID Ideal.
- **Bloque de configuración sistema de control (Parametros):** permite configurar los parámetros del sistema tales como set point, habilitar y deshabilitar modos de control, selección de modo de control (automático-manual), etc.
- **Osciloscopio (SCOPE):** permite el monitoreo de las señales a través de xrtailab.

3. Configuración de los parámetros. Haga doble clic sobre el bloque “Parámetros” y configure tal como se muestra en la figura 93. Observe que el valor de Kc está establecido en 1 y los componentes Integral y Derivativo están desactivados.

Figura 93: Configuración parámetros para control P



Haga clic sobre “OK”.

Cierre la ventana “controlpid”.

En la ventana “controlPID”, en el menú “RTAI”, haga clic sobre la opción “RTAI CodeGen”. Haga clic sobre el súper bloque.

Deje los parámetros por defecto.

Haga clic en “OK”.

4. Revise la ventana de “scilab”, para asegurarse que el ejecutable ha sido creado.
5. En la ventana del “Terminal” digite:

./controlpid –v

Presione “Enter”

6. Abra una ventana de “RTAI-Lab Graphical User Interface”.
7. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “controlP”.
8. Haga clic en “OK”
9. En la ventana de “RTAI-Lab Graphical User Interface”, la traza verde representa el set point, establecido en un valor de 60 ºC, unidades por división de 15 grados ºC. La traza roja representa la temperatura controlada, sus valores máximos, mínimo y el valor de los punteros Y1 y Y2. La primera traza azul claro es el esfuerzo de control P a la salida del controlador. La segunda traza roja representa la señal de error, unidades por división de 0.25 unidades. La segunda traza verde representa el componente proporcional de la respuesta del controlador PID, unidades por división de 0.25 unidades, la traza purpura representa el componente integral de la respuesta del controlador PID,

unidades por división de 0.25 unidades. La traza amarilla representa el componente derivativo del controlador PID, unidades por división de 0.2 unidades. La segunda traza azul claro representa la respuesta total de todos los términos del controlador PID, unidades por división 0.25 unidades.

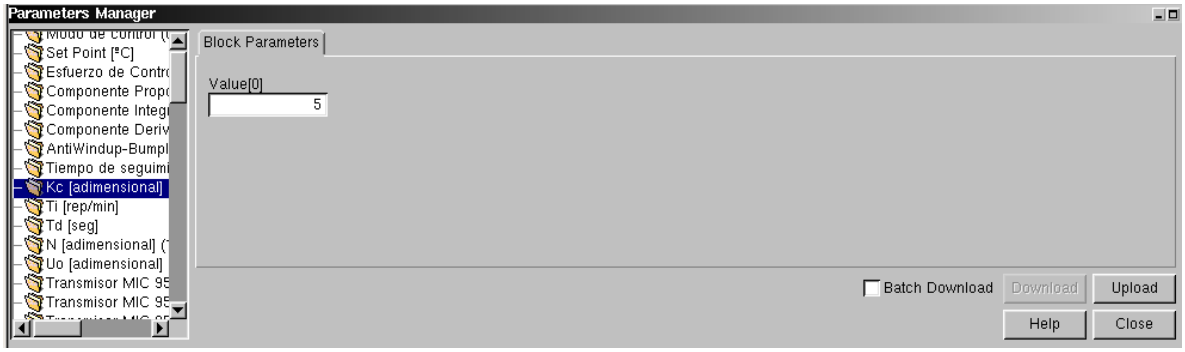
Espere hasta que el sistema se estabilice.

Cuál es el máximo valor de la variable controlada (traza roja) con Kc=1?

Valor: _____

10. En el “Administrador de Parametros” encuentre la opción “Kc [adimensional] o Banda Proporcional [%]” y cambie el valor de “Value[0]” a 5, como se observa en la figura 94.

Figura 94: Cambiar el valor de la ganancia proporcional Kc.



Espere hasta que el sistema se estabilice.

¿Cuál es el máximo valor de la variable controlada con Kc=5?

Valor: _____

11. Repita el paso anterior para un valor de Kc = 50.

Cuál es el máximo valor de la variable controlada con Kc=50?

Valor: _____

12. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
13. Maximice la ventana del osciloscopio.
14. Guarde una imagen del osciloscopio bajo el nombre de “controlP”.
15. Minimice la ventana del osciloscopio.
16. En el “Administrador de osciloscopios”

Seleccione la pestaña “sp” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “vc” y deshabilite la casilla “Show/Hide”.

Seleccione la primera pestaña “ecpid” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “error” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cproporcional” y habilite la casilla “Show/Hide”.

Seleccione la segunda pestaña “ecpid” y habilite la casilla “Show/Hide”.

17. Maximice la ventana del osciloscopio.
18. Guarde una imagen del osciloscopio bajo el nombre de “esfuerzosdecontrolP”.
19. Detenga la tarea de de control mediante el botón “stop real time code” de la ventana “RTAI-Lab Graphical User Interface”.

Presione “Yes”.

20. Cierre la ventana “RTAI-Lab Graphical User Interface”
21. Coloque el interruptor del ventilador en modo manual. Retire la tapa del ventilador.
22. Espere a que la tira de aluminio del módulo MIC955 se enfríe.
23. Abra la ventana “grupxx”
24. Cree una carpeta llamada “control P”
25. Mueva las imagenes “controlP” y “esfuerzosdecontrolP” a la carpeta “control P”
26. Responda las siguientes preguntas:

¿Qué sucede con el error en estado estacionario al aumentar kc?

¿Es posible eliminar completamente el error en estado estacionario con un controlador P?

¿Que sucede cuando Kc es demasiado grande?

PARTE B. CONTROL PROPORCIONAL + INTEGRAL (CONTROL PI)

1. Abra la ventana “controlPID”
2. Haga doble clic sobre el súper bloque.
3. Haga doble clic sobre el bloque “Parámetros”.

Habilite el componente Integral, estableciendo su valor en 1.

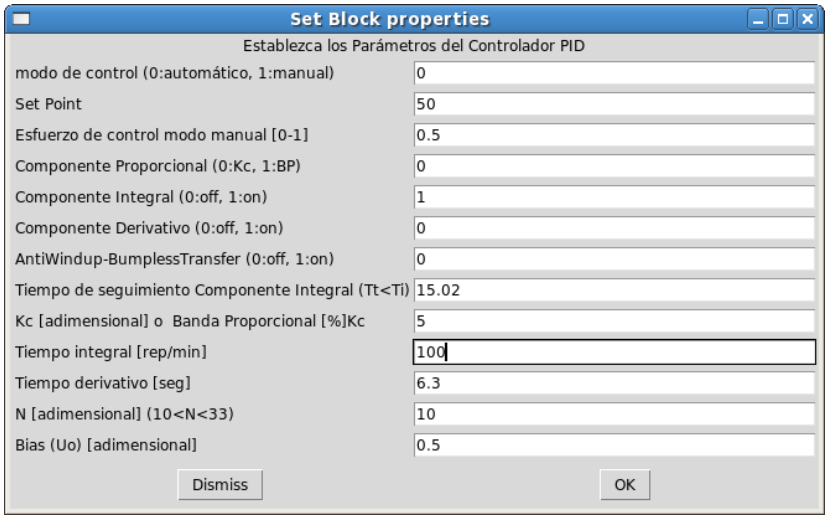
Cambie el valor del set point a 50 grados.

Cambie el valor de Kc a 5.

Cambie el valor de Tiempo integral a 100.

La configuración debería verse como la figura 95:

Figura 95: configuración parámetros para control PI.



Haga clic sobre “OK”.

4. Cierre la ventana “controlpid”.
5. En la ventana “controlPID”, en el menú “RTAI”, haga clic sobre la opción “RTAI CodeGen”.

Deje los parámetros por defecto.

Haga clic en “OK”.

6. Revise la ventana de “scilab”, para asegurarse que el ejecutable ha sido creado.
7. Coloque el interruptor del ventilador en modo off.
8. En la ventana del terminal digite:

./controlpid –v

9. Abra una ventana de “RTAI-Lab Graphical User Interface”.

10. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “controlPID”.

Haga clic en “OK”

11. Espere hasta que el sistema se estabilice.
12. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
13. Maximice la ventana del osciloscopio.

Cuál es el máximo valor de VC con Ti=100? _____

14. Utilizando los punteros tal como se hizo en primera práctica (control ON-OFF), encuentre el valor del tiempo de estabilización.

Tiempo de estabilización: _____.

15. Guarde una imagen del osciloscopio bajo el nombre de “controlPI(Ti100)”.
16. Minimice la ventana del osciloscopio.
17. En el “Administrador de osciloscopios”

Seleccione la pestaña “sp” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “vc” y deshabilite la casilla “Show/Hide”.

Seleccione la primera pestaña “ecpid” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “error” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cproporcional” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cintegral” y habilite la casilla “Show/Hide”.

Seleccione la segunda pestaña “ecpid” y habilite la casilla “Show/Hide”.

18. Maximice la ventana del osciloscopio.
19. Guarde una imagen del osciloscopio bajo el nombre de “esfuerzoscontrolPI(Ti100)”.
20. Detenga la tarea de de control mediante el botón “stop real time code” de la ventana “RTAI-Lab Graphical User Interface”.

Presione “Yes”.

21. Cierre la ventana “RTAI-Lab Graphical User Interface”
22. Coloque el interruptor del ventilador en modo manual.
23. Espere a que la tira de aluminio del módulo MIC955 se enfríe.
24. Abra la ventana “grupxx”
25. Cree una carpeta llamada “control PI”
26. Mueva las imágenes “controlPI(Ti100)” y “esfuerzoscontrolPI(Ti100)” que se encuentran en el escritorio a la carpeta “control PI”
27. Abra la ventana “controlPID”
28. Haga doble clic sobre el súper bloque.

29. Haga doble clic sobre el bloque “Parámetros” y cambie el valor de Tiempo integral a 50.

Haga clic sobre “OK”.

30. Cierre la ventana “controlpid”.
31. En la ventana “controlPID”, en el menú “RTAI”, haga clic sobre la opción “RTAI CodeGen”. Haga clic sobre el súper bloque.

Deje los parámetros por defecto.

Haga clic en “OK”.

32. Revise la ventana de “scilab”, para asegurarse que el ejecutable ha sido creado.
33. Coloque el interruptor del ventilador en modo off.
34. En la ventana del terminal digite:

./controlpid –v

35. Abra una ventana de “RTAI-Lab Graphical User Interface”.
36. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “controlPID”.

Haga clic en “OK”

37. Espere hasta que el sistema se estabilice.
38. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
39. Maximice la ventana del osciloscopio.

Cuál es el máximo valor de VC con Ti=50? _____

Utilizando los punteros tal como se hizo en primera práctica (control ON/OFF), encuentre el valor del tiempo de estabilización.

Tiempo de estabilización: _____.

40. Guarde una imagen del osciloscopio bajo el nombre de “controlPI(Ti50)”.
41. Minimice la ventana del osciloscopio.
42. En el “Administrador de osciloscopios”

Seleccione la pestaña “sp” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “vc” y deshabilite la casilla “Show/Hide”.

Seleccione la primera pestaña “ecpid” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “error” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cproporcional” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cintegral” y habilite la casilla “Show/Hide”.

Seleccione la segunda pestaña “ecpid” y habilite la casilla “Show/Hide”.

43. Maximice la ventana del osciloscopio.
44. Guarde una imagen del osciloscopio bajo el nombre de "esfuerzoscontrolPI(Ti50)".
45. Detenga la tarea de de control mediante el botón "stop real time code" de la ventana "RTAI-Lab Graphical User Interface".

Presione "Yes".

46. Cierre la ventana "RTAI-Lab Graphical User Interface"
47. Coloque el interruptor del ventilador en modo manual.
48. Espere a que la tira de aluminio del módulo MIC955 se enfríe.
49. Abra la ventana "grupxx"
50. Mueva las imágenes "controlPI(Ti50)" y "esfuerzoscontrolPI(Ti50)" que se encuentran en el escritorio a la carpeta "control PI"
51. Abra la ventana "controlPID"
52. Haga doble clic sobre el súper bloque.
53. Haga doble clic sobre el bloque "Parámetros" y cambie el valor de Tiempo integral a 25.

Haga clic sobre "OK".

Cierre la ventana "controlpid".

54. En la ventana "controlPID", en el menú "RTAI", haga clic sobre la opción "RTAI CodeGen".

Deje los parámetros por defecto.

Haga clic en "OK".

55. Revise la ventana de "scilab", para asegurarse que el ejecutable ha sido creado.
56. Coloque el interruptor del ventilador en modo off.
57. En la ventana del terminal digite:

./controlpid –v

Presione "Enter".

58. Abra una ventana de "RTAI-Lab Graphical User Interface".
59. En la ventana de "RTAI-Lab Graphical User Interface", en el menú "File", haga clic sobre "Connect with Profiles" y seleccione "controlPID".

Haga clic en "OK"

60. Espere hasta que el sistema se estabilice.
61. En el "Administrador de osciloscopios", en la pestaña "General" ubíquese en el selector "Trigger" y seleccione la opción "Hold" para congelar la respuesta del sistema.
62. Maximice la ventana del osciloscopio.

Cuál es el máximo valor de VC con Ti=25? _____

Utilizando los punteros tal como se hizo en primera práctica (control ON-OFF), encuentre el valor del tiempo de estabilización.

Tiempo de estabilización: _____

63. Guarde una imagen del osciloscopio bajo el nombre de “controlPI(Ti25)”.
64. Minimice la ventana del osciloscopio.
65. En el “Administrador de osciloscopios”

Seleccione la pestaña “sp” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “vc” y deshabilite la casilla “Show/Hide”.

Seleccione la primera pestaña “ecpid” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “error” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cproporcional” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cintegral” y habilite la casilla “Show/Hide”.

Seleccione la segunda pestaña “ecpid” y habilite la casilla “Show/Hide”.

66. Maximice la ventana del osciloscopio.
67. Guarde una imagen del osciloscopio bajo el nombre de “esfuerzoscontrolPI(Ti25)”.
68. Detenga la tarea de de control mediante el botón “stop real time code” de la ventana “RTAI-Lab Graphical User Interface”.

Presione “Yes”.

69. Cierre la ventana “RTAI-Lab Graphical User Interface”
70. Coloque el interruptor del ventilador en modo manual.
71. Espere a que la tira de aluminio del módulo MIC955 se enfríe.
72. Abra la ventana “grupxx”
73. Mueva las imágenes “controlPI(Ti25)” y “esfuerzoscontrolPI(Ti25)” que se encuentran en el escritorio a la carpeta “control PI”
74. Analice las diferentes imágenes tomadas y con base en su observación explique y concluya:

¿Qué sucede con el error en estado estacionario al habilitar el modo de control integral?

¿El tiempo de respuesta es más rápido o más lento con Ti grande?

¿Que sucede cuando Ti es demasiado grande?

¿Que sucede cuando Ti es demasiado pequeño?

PARTE C: CONTROL PID

1. Abra la ventana “controlPID”
2. Haga doble clic sobre el súper bloque.
3. Haga doble clic sobre el bloque “Parámetros”.

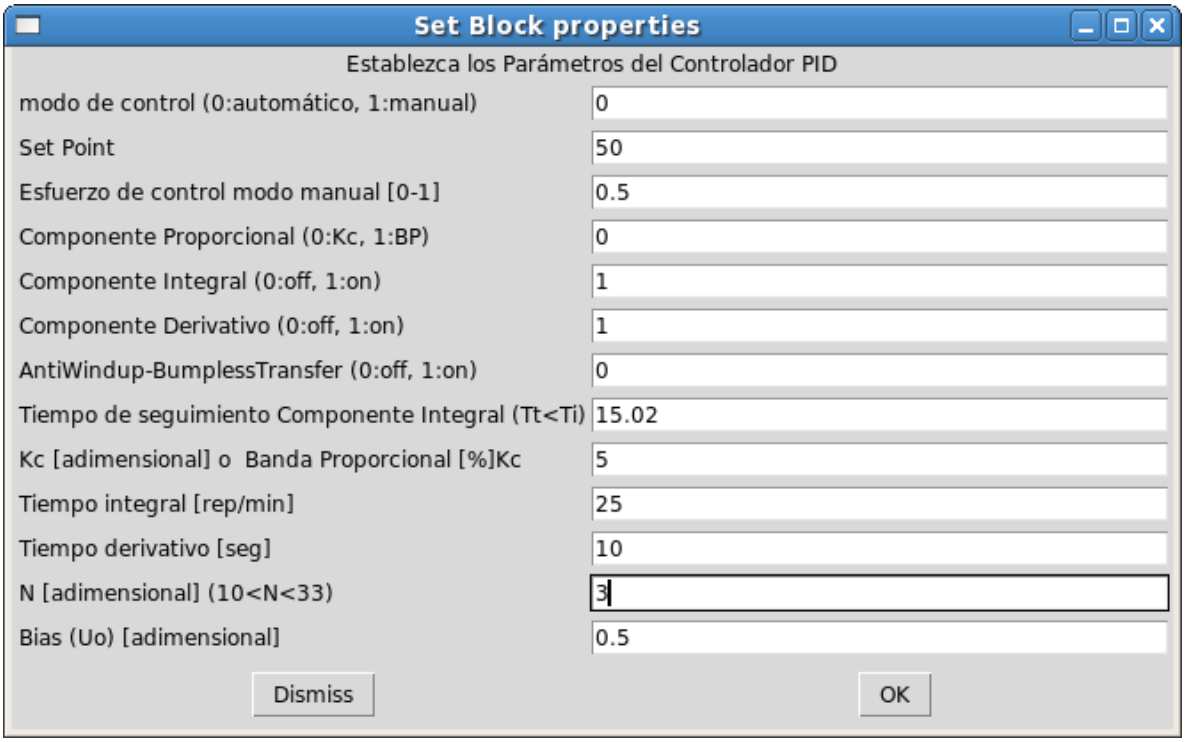
Habilite el componente derivativo estableciendo su valor en 1.

Cambie el valor de Tiempo derivativo a 10.

Cambie el valor de N a 3.

La configuración debería verse como la figura 96:

Figura 96: configuración parámetros para control PI.



Haga clic sobre “OK”.

Cierre la ventana “controlpid”.

4. En la ventana “controlPID”, en el menú “RTAI”, haga clic sobre la opción “RTAI CodeGen”. Haga clic sobre el súper bloque.

Deje los parámetros por defecto.

Haga clic en “OK”.

5. Revise la ventana de “scilab”, para asegurarse que el ejecutable ha sido creado.
6. Coloque el interruptor del ventilador en modo off.

7. En la ventana del terminal digite:

./controlpid –v

8. Abra una ventana de “RTAI-Lab Graphical User Interface”.
9. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “controlPID”.

Haga clic en “OK”

10. Espere hasta que el sistema se estabilice.
11. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
12. Maximice la ventana del osciloscopio.

¿Que sucede al habilitar el modo derivativo?

Utilizando los punteros tal como se hizo en primera práctica (control ON-OFF), encuentre el valor del tiempo de estabilización.

Tiempo de estabilización: _____.

13. Guarde una imagen del osciloscopio bajo el nombre de “controlPID”.
14. Minimice la ventana del osciloscopio.
15. En el “Administrador de osciloscopios”

Seleccione la pestaña “sp” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “vc” y deshabilite la casilla “Show/Hide”.

Seleccione la primera pestaña “ecpid” y deshabilite la casilla “Show/Hide”.

Seleccione la pestaña “error” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cproporcional” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cintegral” y habilite la casilla “Show/Hide”.

Seleccione la pestaña “cderivativa” y habilite la casilla “Show/Hide”.

Seleccione la segunda pestaña “ecpid” y deshabilite la casilla “Show/Hide”.

16. Maximice la ventana del osciloscopio.
17. Guarde una imagen del osciloscopio bajo el nombre de “esfuerzoscontrolPID”.
18. Detenga la tarea de de control mediante el botón “stop real time code” de la ventana “RTAI-Lab Graphical User Interface”.

Presione “Yes”.

19. Cierre la ventana “RTAI-Lab Graphical User Interface”
20. Cierre la ventana “controlPID”.
21. En la ventana de “scilab” presione la tecla “N”.
22. Coloque el interruptor del ventilador en modo manual.
23. Espere a que la tira de aluminio del módulo MIC955 se enfríe.

24. Abra la ventana “grupxx”
25. Cree una carpeta llamada “control PID”
26. Mueva las imágenes “controlPID” y “esfuerzoscontrolPID” que se encuentran en el escritorio a la carpeta “control PID”
27. Responda las siguientes preguntas:

¿Que sucede al adicionar el componente derivativo?

¿Qué sucede al esfuerzo de control?

¿Qué es mejor utilizar un controlador PI o un controlador PID?

H.5. SINTONIZACIÓN DE CONTROLADORES PID

H.5.1. Identificación de los parámetros de un modelo POMTM asociado al módulo MIC955

Para sintonizar correctamente un controlador PID se requiere identificar un modelo lineal asociado a la planta. a través de la información la información dinámica del proceso. El proceso requiere ser excitado registrando tanto la entrada aplicada como la respuesta de la planta.

La mayoría de métodos de sintonización de controladores se basan en los parámetros de un modelo de orden reducido que permita representar sistemas dinámicos de orden alto, por esta razón los más empleados son los de primer o segundo orden más tiempo muerto.

La curva de reacción del proceso se obtiene mediante una prueba de lazo abierto con el controlador manual y el sistema situado en un punto de operación deseado. En estas condiciones se aplica un cambio de escalón en la salida del controlador y se registra esta señal y la de salida del proceso, desde el instante en que se aplicó el escalón de entrada hasta que el sistema alcance un nuevo punto de operación estable, si este es un proceso auto-regulado.

Para esta práctica se utiliza el método basado en dos puntos sobre la curva de reacción propuesto por Smith.

Realizar los siguientes pasos antes de empezar la práctica:

NOTA: Se recomienda prestar especial atención a las sugerencias, notas Y ejemplos.

1. Encienda el ordenador y espere a que cargue la ventana de inicio.
2. Digite “root” en la casilla usuario. Presione “Enter”.
3. Digite “feedback” en la casilla contraseña. Presione “Enter”.
4. Conecte el cable de conexión del módulo MIC955 a la fuente de alimentación.

IMPORTANTE: Conecte primero el conector negro (tierra) para evitar cualquier tipo de daño por energía almacenada dentro de la planta.

5. Conecte el enchufe de la fuente de poder al tomacorriente más cercano.
6. Encienda la fuente de poder.
7. En el escritorio haga doble clic sobre el icono de “Equipo”
8. Haga doble clic en el disco “Sistema de archivos”.
9. Haga doble clic en la carpeta “practicas”.
10. En el menú “Archivo” seleccione la opción “crear una carpeta”.

Nota: Esta acción también puede realizarse haciendo clic derecho y seleccionando la opción “crear una carpeta”.

11. Asígnele el nombre de “grupxx”.

NOTA: “xx” representa el número de su grupo. **Ejemplo:** “grupo00”

Inicio de la Practica de Identificación.

1. Digite los siguientes comandos en el terminal:

cd grup00

cp /practicas/identificacion.cos identificacion.cos

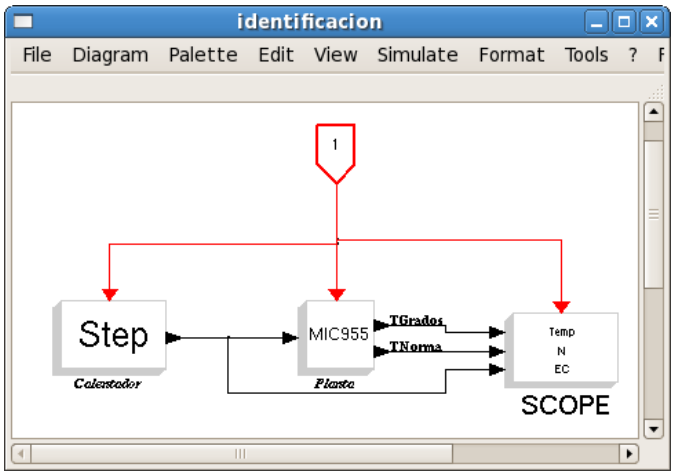
2. En la ventana de scilab, digite el siguiente comando:

scicos identificacion.cos

3. Haga clic sobre el super bloque.

Dentro del súper bloque contenido se encuentra el diagrama que se muestra en la figura 97, el cual costa de un escalón que funciona como entrada a la planta identificado como “Calentador”, el bloque de comunicación real con la planta y un SCOPE el cual permite monitorear el proceso.

Figura 97: configuración parámetros para control PI.



4. En la ventana identificación, seleccione la opción “RTAI” y seleccione “RTAI CodeGen”.
5. Deje por defecto.
6. Haga clic en “OK”
7. En la ventana de Terminal digite:

/.identificacion –v

8. Abra una ventana de “RTAI-Lab Graphical User Interface”.
9. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “identificacion”.

Haga clic en “OK”

10. En la ventana de “RTAI-Lab Graphical User Interface”, la traza verde representa el EC, establecido inicialmente en un valor de 16 % (el actuador entrega el 16 % de la potencia disponible). La traza roja representa la temperatura controlada, sus valores máximos, mínimo y el valor de los punteros Y1 y Y2.
11. Espere a que la respuesta se estabilice
12. Cambie el valor del EC de 16 % 36 %. Para ello ubíquese en el administrador de parámetros y ubíque Calentador, y cambie el valor de 0.16 a 0.36.
13. Espere a que laVC alcance el nuevo estado estacionario.
14. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
15. Maximice el osciloscopio

El método de los dos puntos sobre la curva de reacción fue propuesto por Smith, su procedimiento de identificación puede utilizarse para obtener un modelo POMTM representado por la ecuación 121:

$$G_p(s) = \frac{k_p e^{-t_m s}}{\tau s + 1} \quad (121)$$

Los instantes seleccionados por este autor fueron los tiempos requeridos para que la respuesta alcance el 28.3% ($t_{28.3}$) y el 63.2% ($t_{63.2}$) del valor final, de donde podemos obtener las siguientes ecuaciones:

$$\tau = 1.5(t_{63} - t_{28}) \quad (122)$$

$$t_m = t_{63} - \tau \quad (123)$$

$$K_p = \frac{\left(\frac{\Delta y(t)}{span}\right)}{\Delta u(t)} \quad (124)$$

16. Utilizando los punteros tal como se hizo en primera práctica (control ON-OFF), encuentre el valor del tiempos en los cuales la respuesta alcanza el 28.3% ($t_{28.3}$) y el 63.2% ($t_{63.2}$). Y reemplace en las ecuaciones 122, 123 y 124 para calcular los parámetros del modelo POMTM.
17. Con base en el modelo POMTM obtenido, seleccione la estructura del controlador a utilizar y los métodos de sintonzación válidos para este modelo.
18. Calcule los parámetros del controlador PID.

H.6. FENOMENOS NO LINEALES PRESENTES EN CONTROL DE PROCEOS (WINDUP Y BUMPTRANSFER), TECNICA ANTIWINDUP-BUMPLESSTRANSFER (AWBT)

Windup del Integrador: En aplicaciones industriales los sistemas de control deben operar con restricciones que los sistemas físicos imponen, por ejemplo la capacidad finita de almacenamiento de compresores y tanques o el rango limitado de velocidad que tienen los motores. Un problema muy común tiene que ver con la saturación de los actuadores, por ejemplo una válvula de control solamente puede operar entre el rango de completamente abierta y completamente cerrada.

Uno de los principales efectos indeseables de la saturación en los actuadores, es que el integrador del controlador PID continuará operando aún mientras la entrada se encuentra saturada. Así, el estado del integrador puede alcanzar valores excesivos, que deteriorarán la respuesta transitoria del sistema, generalmente produciendo grandes sobreimpulsos. Este efecto se denomina “windup del integrador”.

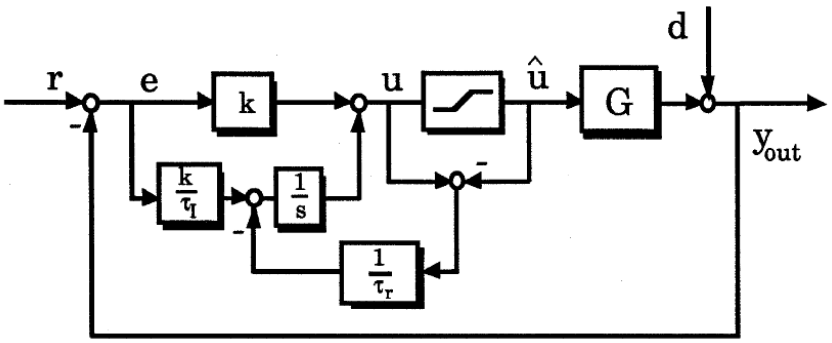
BumpTransfer en la conmutación de modos de control: básicamente un controlador PID puede operar en dos modos de control (manual y automático). Cuando un controlador está operando en modo automático y su modo de operación se conmuta a modo de control manual, el lazo de control se rompe y el controlador intenta ajustar su esfuerzo de control para que la variable controlada permanezca en el valor del set point, debido a esto el esfuerzo de control

automático será diferente al esfuerzo de control manual. Esta diferencia hace que cuando se conmute nuevamente de control manual a control automático se presente un salto indeseable en la respuesta del sistema.

Este fenómeno se conoce como salto de transferencia (*bump transfer)* y se debe a que el término integral del controlador basa su operación en la señal de error a la entrada del controlador y no toma en cuenta ninguna señal introducida manualmente.

Técnica AntiWindup-BumplessTransfer: En esta técnica, la solución para evitar el *windup* es medir la salida del actuador $\hat{u}$ y formar una señal de error que sería la diferencia entre la salida u del controlador y la salida el actuador $\hat{u}$. Esta señal de error se realimenta a la entrada del controlador a través de una ganancia $\frac{1}{T_r}$, como se observa en la figura 97.

Figura 98: Técnica AWBT por seguimiento (tracking) del modo Integral



Fuente: (Kothare,et al, 1993)

Cuando el actuador se satura, la señal de realimentación intenta conducir el error $u-\hat{u}$ a cero, re calculando la acción integral de tal forma que la salida del controlador sea exactamente igual que el límite de saturación.

1. En la ventana de “scilab” digite:

scicos controlAWBT.cos

Presione “Enter”

2. Haga doble clic sobre el súper bloque, podrá observar el diagrama de bloques del sistema de control PID AWBT, el cual está constituido por los siguientes bloques:

- **Bloque de comunicación real con la planta (MIC955).** Este bloque internamente trae los bloques de entrada y salida para el módulo: un bloque

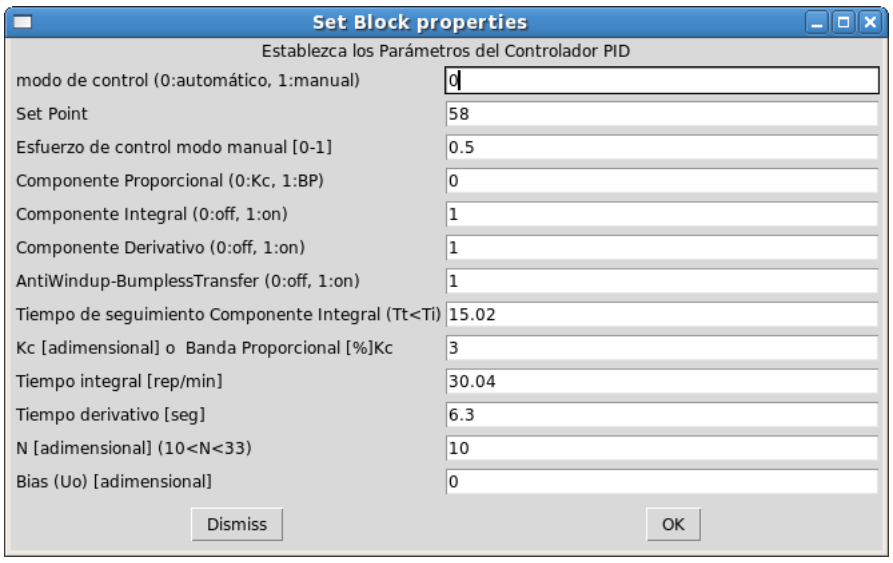
para los actuadores (ventilador y calefactor) y un bloque transmisor(cuyos valores por defecto son: **cero**: 28 ºC, **span**:60 ºC)

- **Bloque de normalización de señales (normalización):** este bloque se encarga de transformar las señales de entrada a valores normalizados entre 0 y 1.
- **Bloque controlador PID (control PID Clásico 1):** este bloque contiene el algoritmo de control de estructura serie.
- **Bloque Control Manual (MCM):** este bloque accesorio para el control PID contiene el algoritmo para control manual.
- **Bloque de configuración sistema de control (Parametros):** permite configurar los parámetros del sistema tales como set point, habilitar y deshabilitar modos de control, selección de modo de control (automático-manual), etc.
- **Osciloscopio (SCOPE):** permite el monitoreo de las señales a través de xrtailab.

3. Configuración de los parámetros. Haga doble clic sobre el bloque “Parámetros” y configure la ventana de la figura 98.

Nota: Ingrese los valores de Kc, Ti y Td que obtuvo en la práctica de identificación.

Figura 99: configuración parámetros para control PID AWBT.



Fuente: (Kothare,et al, 1993)

Haga clic sobre “OK”.

4. Cierre la ventana “controlawbt”.
5. En el diagrama de la figura 1, en el menú “RTAI”, haga clic sobre la opción “RTAI CodeGen”.
6. Deje los parámetros por defecto.

Haga clic en “OK”.

7. Revise la ventana de “scilab”, para asegurarse que el ejecutable ha sido creado.
8. En la ventana del terminal digite:

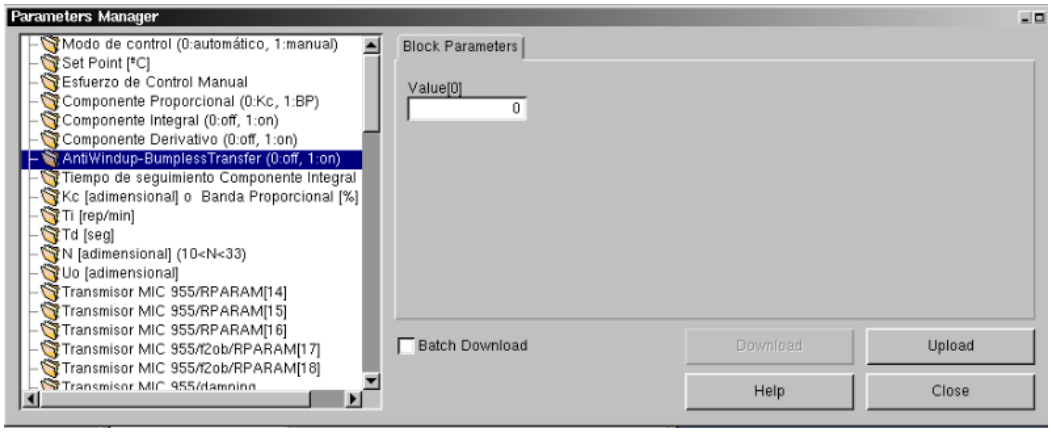
./controlawbt –v

9. Abra una ventana de “RTAI-Lab Graphical User Interface”.
10. En la ventana de “RTAI-Lab Graphical User Interface”, en el menú “File”, haga clic sobre “Connect with Profiles” y seleccione “controlAWBT”.

Haga clic en “OK”

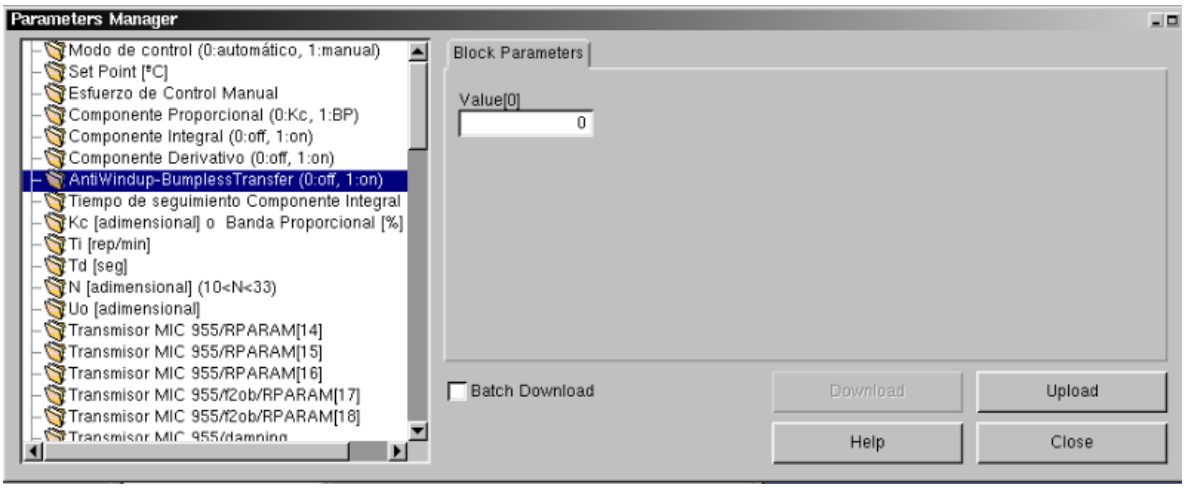
11. En la ventana de “RTAI-Lab Graphical User Interface”, la traza verde representa el set point, establecido en un valor de 58 ºC, unidades por división de 15 grados ºC. La traza roja representa la temperatura controlada, sus valores máximos, mínimo y el valor de los punteros Y1 y Y2. La traza azul claro representa el esfuerzo de control en modo automático, unidades por división de 0.5 unidades. La traza purpura representa el esfuerzo de control en modo manual, unidades por división de 0.5 unidades. Las trazas amarillas la presencia de disturbio y momento de conmutación modos de control, unidades por división 1 unidad.
12. Espere a que la respuesta se estabilice
13. Seleccione el osciloscopio y ubique el puntero 2 en la parte derecha del osciloscopio hasta t2 marque 1700 segundos
14. Deshabilite la protección AntiWindup – BumplessTransfer: En el administrador de parámetros encuentre la opción “Antiwindup-BumplessTransfer (0:off; 1:on)” y establezca el parámetro “Value[0]” en 0, tal como se muestra en la figura 99.

Figura 100: Deshabilitar AntiWindup-BumplessTransfer



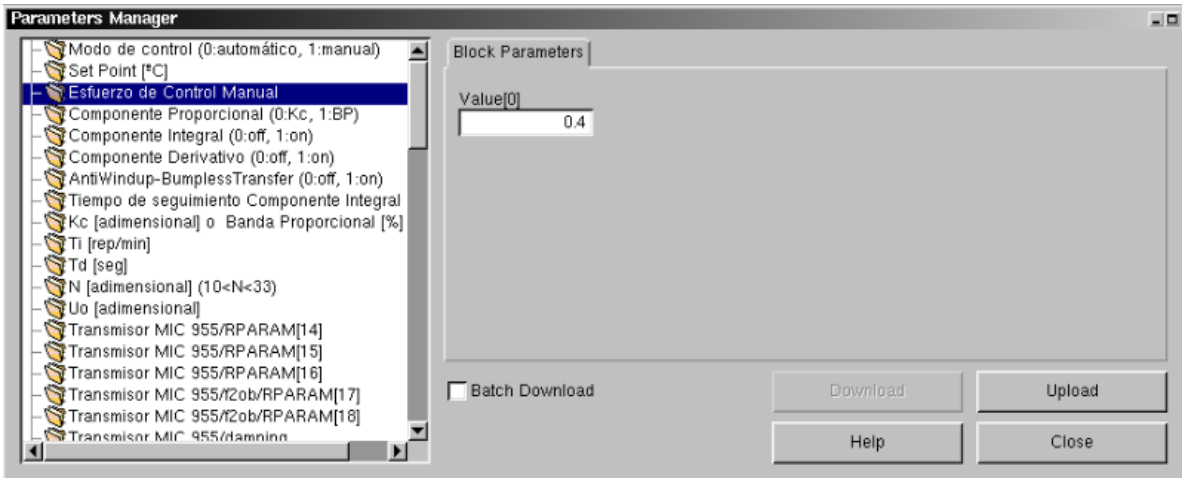
15. Cambie el modo de control de automático a manual. En el administrador de parámetros ubíquese en “Modo de control (0:automatico; 1:manual)” y establezca el parámetro “Value[0]” en 1, tal como se muestra en la figura 100

Figura 101: conmutación modos de control automático a manual



16. Seleccione el osciloscopio y ubique el puntero 1 en la parte en una posición tal que la diferencia “dt” entre los tiempos que marcan el puntero 1 (t1) y el puntero 2 (t2) sea igual a 60 segundos.
17. En el administrador de parámetros ubíquese en “Esfuerzo de control manual” y establezca el parámetro “Value[0]” en 0.4, tal como se muestra en la figura 101:

Figura 102: Cambiar valor esfuerzo de control Manual



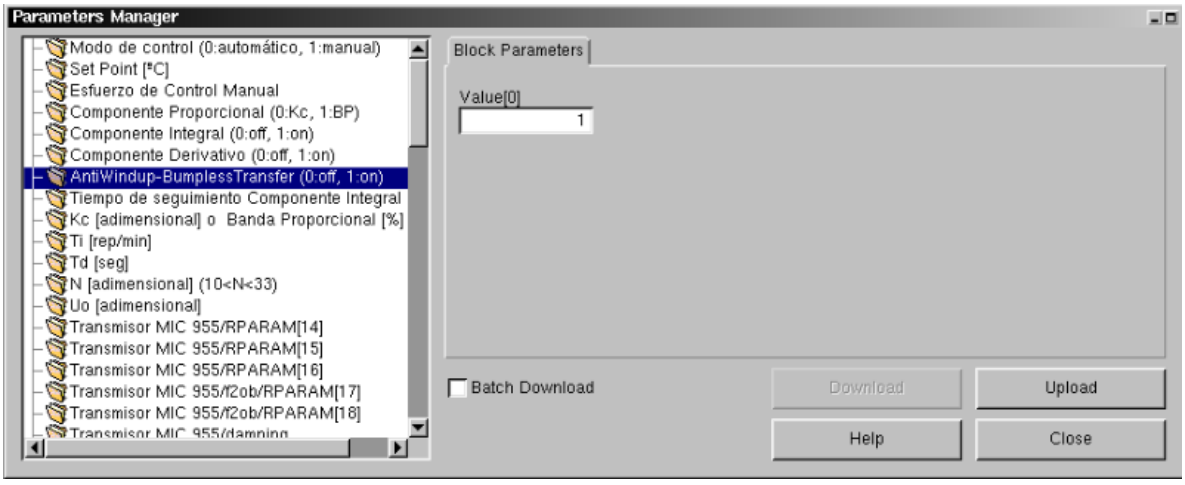
18. Cuando la traza amarilla llegue al punto marcado en el paso anterior, presione “Enter”.
19. Seleccione el osciloscopio y ubique el puntero 1 en la parte en una posición tal que la diferencia “dt” entre los tiempos que marcan el puntero 1 (t1) y el puntero 2 (t2) sea igual a 240 segundos.
20. En el administrador de parámetros ubíquese en “Esfuerzo de control manual” y establezca el parámetro “Value[0]” en 0.6.

21. Cuando la traza amarilla llegue al punto marcado en el paso anterior, presione “Enter”.
22. Seleccione el osciloscopio y ubique el puntero 1 en la parte en una posición tal que la diferencia “dt” entre los tiempos que marcan el puntero 1 (t1) y el puntero 2 (t2) sea igual a 360 segundos.
23. En el administrador de parámetros ubíquese en “Esfuerzo de control manual” y establezca el parámetro “Value[0]” en 0.5.
24. Cuando la traza amarilla llegue al punto marcado en el paso anterior, presione “Enter”.
25. Seleccione el osciloscopio y ubique el puntero 1 en la parte en una posición tal que la diferencia “dt” entre los tiempos que marcan el puntero 1 (t1) y el puntero 2 (t2) sea igual a 420 segundos.
26. En el administrador de parámetros ubíquese en “Modo de control (0:automatico; 1:manual)” y establezca el parámetro “Value[0]” en 0.

Cuando la traza amarilla llegue al punto marcado en el paso anterior, presione “Enter”.

27. Espere a que la respuesta del sistema se estabilice.
28. Habilite la protección Antiwindup-BumplessTransfer. En el administrador de parámetros ubíquese en “Antiwindup-BumplessTransfer (0:off; 1:on)” y establezca el parámetro “Value[0]” en 1, tal como se muestra en la figura 102:

Figura 103: Habilitar AntiWindup-BumplessTransfer



Presione “Enter”

29. Repita los pasos 171 a 180.
30. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
31. Maximice el osciloscopio

32. Guarde una imagen de la respuesta del sistema bajo el nombre de “bumplesstransfer”.
33. Abra la ventana “grupxx”
34. Cree una carpeta llamada “control AWBT”
35. Mueva la imagen “bumplesstransfer” que se encuentran en el escritorio a la carpeta “control AWBT”

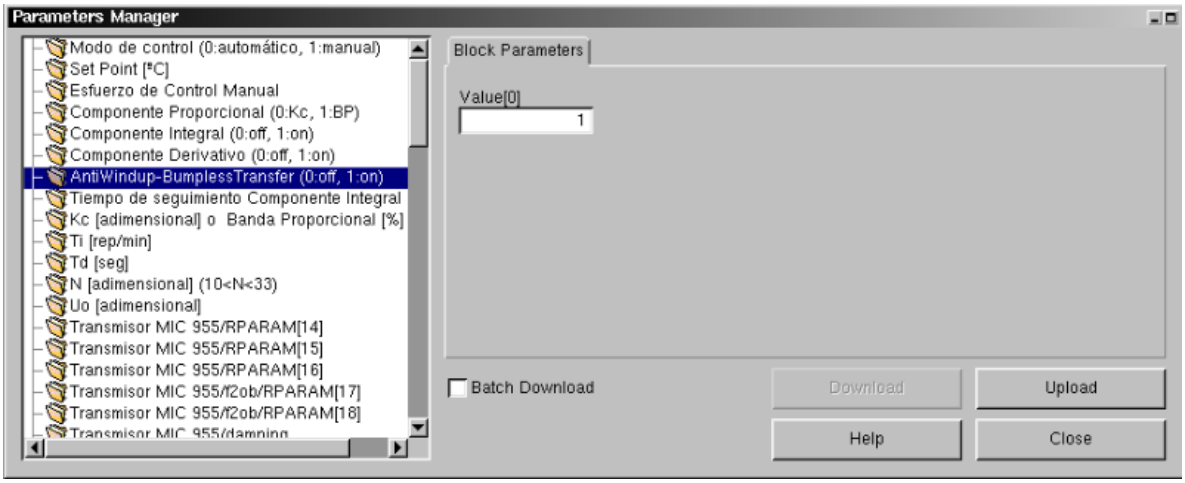
Efecto Windup:

36. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Continous Roling”.
37. Seleccione la pestaña “mcontrol” y deshabilite la casilla “Show/Hide”.
38. Seleccione la pestaña “disturbio” y habilite la casilla “Show/Hide”.
39. En el administrador de Parámetros deshabilite la protección AntiWindup – BumplessTransfer, tal como se ve en la figura 99.
40. Coloque el interruptor del ventilador en “Programmed”.
41. Coloque la tapa del ventilador.

Importante: Asegúrese que la tapa este cerrada completamente.

42. En el administrador de parámetros ubíquese en “Ventilador” y en “Value[0]” establezca el valor de 1, tal como se muestra en la figura.

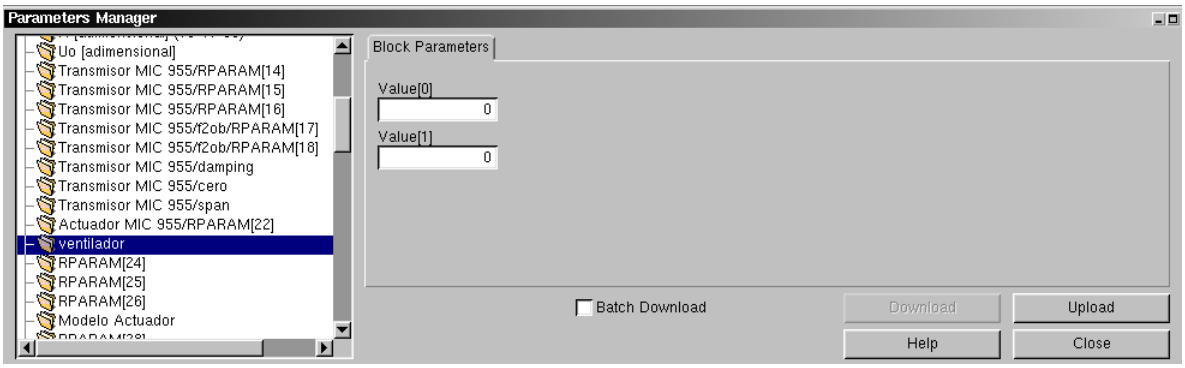
Figura 104: Habilitar disturbio



Presione “Enter”

43. Seleccione el osciloscopio y ubique el puntero 1 en la parte en una posición tal que la diferencia “dt” entre los tiempos que marcan el puntero 1 (t1) y el puntero 2 (t2) sea igual a 180 segundos.
44. En el administrador de parámetros ubíquese en “Ventilador” y en “Value[0]” establezca el valor de 0.

Figura 105: Encender ventilador



45. Cuando la traza amarilla llegue al punto marcado en el paso anterior, presione “Enter”.
46. Espere a que la respuesta del sistema se estabilice.
47. En el administrador de Párametros habilite la protección AntiWindup – BumplessTransfer, tal como se hizo en la figura 23.
48. Repita los pasos 41 a 46.
49. En el “Administrador de osciloscopios”, en la pestaña “General” ubíquese en el selector “Trigger” y seleccione la opción “Hold” para congelar la respuesta del sistema.
50. Maximice el osciloscopio
51. Guarde una imagen de la respuesta del sistema bajo el nombre de “antiwindup”.
52. Mueva la imagen “bumplesstransfer” que se encuentra en el escritorio a la carpeta “control AWBT”
53. Detenga la tarea de de control mediante el botón “stop real time code” de la ventana “RTAI-Lab Graphical User Interface”.

Presione “Yes”.

54. Apague la fuente de alimentación.
55. Desconecte el enchufe del tomacorrientes.
56. Desconecte los cables de conexión del módulo MIC 955 de la fuente de alimentación.
57. Guarde el cable de conexión en la caja del Módulo MIC 955.
58. Guarde la carpeta “grupxx”en una memoria USB.
59. Apague el computador.
60. Responda las siguientes preguntas:

a) En la figura “bumplesstransfer”, ubíquese en la primera parte.

¿Qué sucede con el esfuerzo de control automático (traza azul claro) cuando se conmuta de modo de control automático a manual?

¿Qué sucede la variable controlada (traza roja cuando se conmuta de modo de control manual a automático?

¿Por qué sucede esto?

Cuando se habilita la protección AWBT. ¿Qué sucede con el esfuerzo de control automático (traza azul claro) cuando se conmuta de modo de control automático a manual?

¿Qué sucede la variable controlada (traza roja cuando se conmuta de modo de control manual a automático?

¿Por qué sucede esto?

b) En la figura “antiwindup”, ubíquese en la primera parte.

¿Qué sucede con el esfuerzo de control automático (traza azul claro) en presencia del disturbio?

¿Qué sucede con la variable controlada cuando el disturbio deja de existir?

¿Por qué sucede esto?

Cuando se habilita la protección AWBT.

¿Qué sucede con el esfuerzo de control automático (traza azul claro) en presencia del disturbio?

¿Qué sucede con la variable controlada cuando el disturbio deja de existir?

¿Por qué sucede esto?