ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ACTUALIZACIÓN DE LA BIBLIOTECA DEL SISTEMA LABCON (LABORATORIO REMOTO DE CONTROL AUTOMÁTICO) USADO EN LA FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN (FIEC)"

EXAMEN DE GRADO (COMPLEXIVO)

Previo a la obtención del grado de:

INGENIERÍA EN CIENCIAS COMPUTACIONALES ORIENTACIÓN SISTEMAS MULTIMEDIA

Presentado por:

CRISTHIAN RÓMULO ARROBA RIVERA

GUAYAQUIL – ECUADOR

2015

AGRADECIMIENTO

Principalmente al Ing. Juan Del Pozo, director de tesis, así mismo al Ing. Franklin Kuonqui por su ayuda y colaboración en la realización de este trabajo. A todas las personas que me brindaron su apoyo y su ayuda incondicional, ya que sin sus palabras de aliento no hubiese podido salir adelante y sobre todo a aquellas personas que supieron soportarme en los momentos más difíciles.

DEDICATORIA

Dedico este trabajo a Dios y a nuestros padres, ya que sin su apoyo incondicional no hubiera podido culminar con éxito este trabajo.

TRIBUNAL DE SUSTENTACIÓN

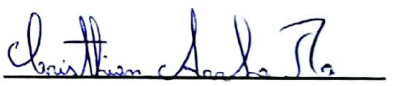
M.Sc. Dennys Cortez Alvarez
EVALUADOR

M.Sc. Juan del Pozo Lemos
EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Informe, corresponde exclusivamente a mi persona; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral".

(Reglamento de exámenes y títulos profesionales de la ESPOL)



Cristhian Rómulo Arroba Rivera

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

RESUMEN

La gran expectativa puesta sobre el sistema LabCon usado dentro del laboratorio de Control Automático de la Facultad de Ingeniería en Electricidad y Computación (FIEC), impulsan la necesidad de perfeccionar la librería de bloques, elemento clave donde se albergan todos y cada uno de los bloques (bloques de simulink y bloques de plantas), de modo que el objetivo de este proyecto está dirigido a la actualización de la librería para que se extienda la vida útil del sistema con un proceso sencillo y claro que permita realizar dicha actualización teniendo como consecuencia que los estudiantes puedan incluir nuevas plantas y bloques simulink que permitan realizar modelos más complejos. Adicionalmente, dentro de este trabajo se utiliza el término “planta” para hacer referencia a un sistema electrónico ya implementado con anterioridad (como un motor electrico o un tanque de presion), que son comúnmente utilizados para realizar las prácticas dentro del Laboratorio de Control Automático. En la tesis de la primera fase del sistema LabCon, realizado por Cristian Idrovo y Humberto Aguilar se usó el término “maquinaria” que será reemplazado por la palabra “planta”.

INTRODUCCIÓN

Anteriormente para poder realizar alguna práctica con alguna planta dentro del Laboratorio de Control Automático, era necesario que el estudiante pueda usar un computador con acceso directo al Compact Field Point (CFP), ahora una de las ventajas de su uso es que debido a que posee diversos módulos para el uso de tarjetas y dispositivos permitiendo el control de más de una planta. Sin embargo permitir que un estudiante manipule directamente este dispositivo debido a su inexperiencia existe una posibilidad de error y desconfigure los valores dentro del propio Compact Field Point poniendo en riesgo la planta. Esta fue uno de las principales motivos para realizar la implementación del sistema LabCon para que los estudiantes realicen las prácticas con las plantas sin tener la limitante del acceso al Compact Field Point y con la ventaja de poder acceder desde cualquier pc que tenga acceso al sistema LabCon, con esto se elimina el inconveniente antes mencionado relacionado al Compact Field Point. Sin embargo, con la constante adquisición de nuevas plantas dentro del laboratorio se vuelve una necesidad que el sistema LabCon se mantenga actualizado acorde a las nuevos equipos usados para esto es necesario general algún documento tipo tutorial en el cual se especifiquen los pasos de forma detallada que permitan incluir las nuevas plantas dentro del sistema LabCon. Además es necesario un documento tipo tutorial especificando los pasos necesarios para la inclusión de nuevos bloques de tipo simulink que se requieran dentro del sistema LabCon. Una vez que se tenga un proceso adecuadamente documentado que permita actualizar la librería del sistema LabCon, se podrán realizar una mayor cantidad de modelos y prácticas que incluyan nuevas plantas y la complejidad de los mismos aumentaría de forma significativa en beneficio de los estudiantes del Laboratorio de Control Automático. (2)

CAPÍTULO 1

1. ANTECEDENTES

El sistema LabCon básicamente esta constituído por dos tipos de bloques: simulink y planta. Ahora debido a que la principal razón para la creación e implementación del sistema LabCon es permitir el control remoto de alguna planta mediante una simulación en tiempo real, de modo que existe la necesidad de continuar agregando una mayor cantidad de bloques de ambos tipos para mejorar la usabilidad, complejidad y experiencia dentro de una simulación del sistema LabCon.

1.1. Identificación de la problemática

Una vez que se realizó la implementación de la primer versión del sistema LabCon, como trabajo de graduación de los estudiantes Cristian Idrovo y Humberto Aguilar, para ser utilizado en el Laboratorio de Control Automático y que permite la ejecución de forma remota de prácticas de laboratorio con bancos de prueba reales, como por ejemplo el control de velocidad de un motor eléctrico y del control de nivel de un tanque de agua, se optó por iniciar una nueva versión del sistema que permita una mejor administración del sistema y el establecimiento de un procedimiento tipo tutorial para la ampliación de las aplicaciones de LabCon con nuevos experimentos.

La primera parte de la actualización del sistema: es decir su administración, está siendo desarrollada por otro equipo de estudiantes como trabajo de graduación. La segunda parte, referente al tutorial, motivo de este trabajo de graduación, permitirá definir el procedimiento para ampliar la librería de bloques operativos de Simulink a ser utilizados por LabCon (de acuerdo a las necesidades) de una manera sencilla e intuitiva, así como también los bloques que representarán las nuevas plantas de experimentación que se irán incorporando en el Laboratorio de Control Automático de la FIEC.

1.2. Justificación del proyecto de Graduación

Actualmente se requiere un procedimiento claro y sencillo para poder agregar un nuevo bloque así como los pasos necesarios para poder agregar una nueva práctica al sistema LabCon, debido a que de forma continua se requiere ampliar la cantidad de prácticas (el total actual es cuatro) así como de nuevos bloques mediante un manual o tutorial. Ahora también es importante tomar en cuenta que se mejoró la funcionalidad del sistema agregando nuevas opciones (Fuzzy Logic Controller) y eliminación de otras opciones (se eliminó el botón de pausa).

1.3. Solución Propuesta

Como primer paso es necesario documentar cada una de las estructuras utilizadas dentro del sistema Labcon (librerías, páginas, etc). Una vez que tenemos identificados cada elemento del sistema, debemos tomar los requerimientos o mejoras que se desean agregar al sistema junto con su respectiva documentación de soporte para los mismos. Estos requerimientos son la base para poder realizar la implementación de los cambios al sistema, en donde una vez realiza-

dos dichas modificaciones se deben realizar varias pruebas para confirmar que el funcionamiento arroja los resultados esperados. Luego de que el resultado de las pruebas es satisfactorio se debe realizar la instalación del sistema dentro del pc servidor. Ahora es importante tener en cuenta que se necesita documentación especializada para realizar el correcto mantenimiento del sistema LabCon una vez que se haya instalado dentro del pc servidor.

1.4. Objetivos

- Se debe incrementar el número actual de bloques para la librería del sistema LabCon.
- Creación del bloque especial “Display” que permita la visualización de los datos de salida (resultados de la ejecución del experimento).
- Modificación del comportamiento e interacción con el usuario para varios bloques especiales (Ejemplo: bloque SimOut).
- Creación de manual para la creación e implementación de nuevas plantas dentro del sistema LabCon.
- Implementación del nuevo bloque especial “Fuzzy Logic Controller” dentro de la librería del sistema LabCon.

1.5 Metodología

Debido a que el proyecto fue desarrollado contando con un solo recurso para su implementación se usará el modelo incremental dividiendo el proyecto en varias fases y para cada una de ellas se obtuvieron requerimientos para lo cual se crearon una lista de tareas junto a sus correspondientes entregables.

1.6. Alcance y Limitaciones

Dentro del proyecto se pretende el aumento y creación de nuevos bloques especiales para el sistema LabCon. Sin embargo existen bloques que poseen un mayor grado de dificultad debido a la gran cantidad de parámetros que poseen y el sistema LabCon al momento tiene un numero limitado de parámetros por bloque y de sus correspondientes tipo de dato. Con respecto al manejo de la simulación en tiempo real se manejan los eventos de inicio y fin cuando se trabaje con una práctica, sin embargo también es necesario implementar un módulo que permita controlar las sesiones para cada una de las diferentes prácticas que trabajen directamente con algún bloque tipo planta (como por ejemplo el bloque de tres tanques) de modo que el sistema LabCon pueda validar que un usuario pueda utilizar una práctica a la vez y justamente para esto también es necesario poder manejar diferentes horarios para cada usuario de modo que se pueda manejar un tiempo máximo para trabajar dentro del sistema LabCon.

Siempre que se ejecute una simulación , se realizará con un tiempo de muestreo de 0.01 segundos (valor predeterminado dentro del sistema LabCon). Al momento de trabajar con el bloque “Fuzzy Logic Controller”, el tiempo de muestreo debe ser compatible con el valor antes mencionado debido a que el valor

del tiempo de muestreo dentro del sistema LabCon no puede ser modificado. Cualquier variación en el tiempo de muestreo afectará directamente al bloque “WebScope”, el cual fue calibrado para poder trabajar con el tiempo de muestreo de 0.01 segundos.

CAPÍTULO 2

2. MEJORAMIENTO DEL SISTEMA LABCON

Dentro de este capítulo se enlistan las modificaciones realizadas al sistema LabCon.

2.1. Introducción

Cuando se empezó con el uso de la primera fase del sistema LabCon, se empezaron a encontrar inconvenientes principalmente a la limitada cantidad de bloques de la librería, de modo que fue necesario empezar a agregar nuevos bloques de plantas y bloques simulink al sistema, sin embargo, se encontraron otras limitantes respecto a la seguridad en la ejecución de una simulación mientras se trabaja con una planta. La seguridad es un elemento clave al momento de trabajar con plantas dentro de una simulación en tiempo real para evitar daños en las mismas, por lo que se deben establecer medidas que permitan salvaguardar su integridad. Así mismo para poder agregar un nuevo bloque simulink se encontró que no existía un procedimiento claro para realizar la implementación y configuración del mismo para la cual fue necesario crear un nuevo procedimiento que se encargue de integrar un nuevo bloque simulink dentro del sistema LabCon.

2.2. Nuevas Características

1. Modificación del comportamiento del bloque SimOut. Anteriormente, sólo se permitía grabar los datos de una simulación si existía el bloque WebScope presente en la misma; ahora el bloque SimOut funciona de manera independiente, es decir, al momento se puede grabar la información sin importar si el bloque WebScope se encuentre presente en la simulación.

2. Se eliminó el botón Pause, de modo que ahora para controlar la simulación solo existirán dos botones (estados de la simulación) para la misma: Play y Stop.

3. Se agregó una nueva práctica (Bloque de 3 Tanques), además, se implementó su correspondiente bloque StopPlanta3T para esta práctica que se activa durante escenarios especiales a modo de contingencia.

4. Se creó un nuevo bloque Display, que permite visualizar hasta 5 señales diferentes mostrando sus valores numéricos de forma independiente.

5. Se pudo identificar y corregir el comportamiento de la simulación mientras se trabaja con alguna práctica en tiempo real (se controla la planta de la práctica de forma remota por medio del Sistema LabCon) de modo que se active el correspondiente bloque Stop para la práctica que se encuentre activa, en estos casos particulares:

    - Al presionar el botón Stop mientras se ejecuta la simulación que incluya una práctica activa.
    - Cuando el tiempo de ejecución (es un valor que puede ser modificado por el usuario) de la simulación ha llegado a su límite máximo.
    - En el momento de salir de presionar el botón LogOut (salir de sesión),

esto puede ocurrir incluso mientras se encuentra trabajando con la práctica.

Para todos los casos mencionados anteriormente, se procederá a activar el correspondiente bloque Stop dependiendo de la práctica que el usuario escogió para trabajar. Este bloque Stop es un archivo de extensión mdl que realiza un procedimiento interno que permite que la planta real detenga su funcionamiento y luego se apague de forma segura para evitar cualquier daño de la misma.

6. La simulación ahora puede ser ejecutada sin necesidad que se incluya un bloque WebScope o un bloque Display; anteriormente esto no era posible de modo que la simulación siempre necesitaba que se incluyera un bloque WebScope dentro de la misma.

7. Se agregaron las nuevas imágenes para los respectivos nuevos bloques que se crearon, existen algunos nuevos bloques que requieren más de una imagen como por ejemplo: el bloque Math Function, en donde dependiendo del valor dentro del combobox que se elija la imagen se modificará acorde al valor escogido.

8. Anteriormente para los bloques: Discrete Transfer Fcn, Discrete Zero-Pole, Transfer fcn y Zero Pole; se necesitó cambiar la forma en que presentaba los datos, debido a que anteriormente solo presentaba el valor del numerador o denominador según sea el caso, sin tomar en cuenta que se debía presentar por medio de una fórmula; esto fue corregido para que se presente la fórmula completa con los valores y signos (+/-) que les corresponde.

9. Aumento en el número de bloques del archivo librería.mdl. En total tenemos un aumento de 25 nuevos bloques de simulink para la librería del sistema

LabCon. A continuación el detalle de nuevos bloques simulink :

- Planta 3T (Bloque especial, corresponde a una nueva práctica).
- Planta de tanque de presión (Bloque especial, corresponde a una nueva práctica).
- Display (Bloque especial).
- Trigonometric Function.
- Subtract.
- Math Function.
- MinMax.
- Abs.
- Sine Wave Function.
- Sign.
- Complex to Real-Imag.
- Complex to Magnitude-Angle.
- Product.
- Divide.
- Product of Elements.
- Backlash.
- Relay
- Rate Limiter.
- Variable Transport Delay.
- Variable Time Delay.
- Transport Delay.
- Quantizer.

- Hit Crossing.
- Dead Zone.
- Fuzzy Logic Controller.

2.2.1. Listado de bloques Simulink

Figura 2.1: Bloque Función Trigonométrica

Bloque Función Trigonométrica (En inglés: Trigonometric Function):
Permite realizar una serie de funciones trigonométricas. (1)

Figura 2.2: Bloque Display

Bloque Display:
Visualiza los datos resultantes de las señales de entrada de los bloques conectados a este bloque “Display”. (1)

Figura 2.3: Bloque Subtract

Bloque Subtract:
Este bloque se puede sumar o restar escalares, vectores o entradas de matrices que se encuentre dentro de una seãl de entrada. Ahora se deben identificar

las operaciones a realizar por el bloque usando una lista de seãles (+) o (-) y el símbolo espacio (—) para especificar el tipo de operación a realizar para cada señal de entrada. (1)

Figura 2.4: Bloque Deadzone

Bloque Deadzone:

Genera una salida cero dentro de una región determinada, llamada la zona muerta (deadzone). Se especifica el límite inferior (LI) y el límite superior (LS) de la zona muerta como el inicio y fin como parámetros de la zona muerta, respectivamente. (1)

Figura 2.5: Bloque Hit Crossing

Bloque Hit Crossing:

El bloque Crossing Hit permite detectar cuando la señal de entrada alcanza el valor de parámetro offset dependiendo del parámetro de dirección con que se encuentre configurado el bloque Hit Crossing. (1)

Figura 2.6: Bloque Quantizer

Bloque Quantizer:

El bloque cuantificador pasa su señal de entrada a través de una función escalonada de manera que muchos puntos vecinos en el eje de entrada se asignan a un punto en el eje de salida. (1)

Figura 2.7: Bloque Transport Delay

Bloque Transport Delay:

El bloque retraso de transporte (Transport delay) y retrasa la entrada por un período de tiempo específico. Es posible usar este bloque para simular un tiempo de retraso. La entrada de este bloque debe ser una señal continua. (1)

Figura 2.8: Bloque Variable Transport Delay

Bloque Variable Transport Delay:

En este modo, la salida del bloque en el paso de tiempo actual es igual al valor de sus datos de entrada en un paso de tiempo anterior igual a la hora actual menos el retardo del transporte. (1)

Figura 2.9: Bloque Variable Time Delay

Bloque Variable Time Delay:

En este modo, el bloque tiene una entrada de datos, una entrada de retardo de tiempo, y una salida de datos. La salida en el paso de tiempo actual es igual al valor de su entrada de datos en un paso de tiempo anterior. Este paso de tiempo es el tiempo de simulación actual menos un tiempo de retardo especificado por la entrada de retardo de tiempo. (1)

Figura 2.10: Bloque Rate Limiter

Bloque Rate Limiter:

Permite limitar la primera derivada de la señal que pasa a través de él. La señal de salida cambia no más rápido que el límite especificado. (1)

Figura 2.11: Bloque Relay

Bloque Relay:

Permite cambiar la señal de salida entre dos valores específicos. Cuando el relay está encendido, permanece encendida hasta que cae la entrada por debajo del valor del parámetro de punto apagado (switch off). Cuando el relay está apagado, permanecerá apagado hasta que la entrada supera el valor del parámetro encendido (switch on). El bloque acepta una entrada y genera una salida. Es importante tomar en cuenta que cuando la entrada inicial se encuentra entre el punto de apagado y el punto de encendido, la salida inicial es el valor cuando el relay está apagado. (1)

Figura 2.12: Bloque Backlash

Bloque Backlash:

Este bloque implementa un sistema en el que un cambio en la entrada causando un cambio igual en la salida. Sin embargo, cuando la entrada cambia de dirección, este cambio inicial en la entrada no tiene ningún efecto inmediato en la salida. (1)

Figura 2.13: Bloque Product

Bloque Product:

Multiplica y divide señales escalares y no escalares así como matrices. Por defecto, este bloque produce como resultado la multiplicación de dos entradas: dos escalares, un escalar y una no escalar, o dos no escalares pero que tengan ambas las mismas dimensiones. (1)

Figura 2.14: Bloque Divide

Bloque Divide:

El bloque divide el valor de la señal de la primera entrada para la segunda. (1)

Figura 2.15: Bloque Complex to Magnitude-Angle

Bloque Complex to Magnitude-Angle:

El bloque toma la señal de entrada de valor complejo y devuelve como salida la magnitud y el ángulo respectivamente. (1)

Figura 2.16: Bloque Complex to Real-Imag

Bloque Complex to Real-Imag:

El bloque toma la señal de entrada de valor complejo y devuelve como salida la parte real y / o imaginaria respectivamente de la señal de entrada. (1)

Figura 2.17: Bloque Sine Wave Function

Bloque Sine Wave Function:

Permite generar una señal de salida como onda sinusoidal, mediante señal externa de entrada como fuente de tiempo. (1)

Figura 2.18: Bloque Abs

Bloque Abs:

Obtiene como resultado el valor absoluto de la señal de entrada. (1)

Figura 2.19: Bloque MinMax

Bloque MinMax:

Permite calcular el valor máximo o mínimo de la señal de entrada. (1)

Figura 2.20: Bloque Math Function

Bloque Math Function:

Este bloque permite usar varias funciones matemáticas. (1)

Figura 2.21: Bloque Math Sign

Bloque Sign:

Permite tomar la señal de entrada y devuelve una señal de salida de acuerdo al cuadro detallado a continuación:

Bloque Fuzzy Logic Controller:

Este es un bloque particular de simulink que se configura mediante un archivo

Figura 2.22: Bloque Fuzzy Logic Controller

de configuración (extensión .fis), este archivo de configuración depende directamente de la simulación que se va a realizar debido que está compuesto por una serie de reglas específicas de comportamiento para algún elemento o elementos de la simulación, en nuestro caso será utilizado directamente con alguna de las plantas dependiendo del caso.

2.2.2. Listado de bloques de plantas

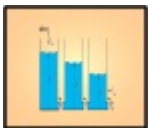


Figura 2.23: Bloque Planta de tres tanques

Bloque Planta 3T (Planta de tres tanques):

La etapa hidráulica del sistema de tres vasos comunicantes que se analiza está constituida por un reservorio desde el cual una bomba hidráulica envía agua hacia el primero de los tres tanques interconectados, de ahí el agua pasa al segundo tanque y de éste al tercero, el cual descarga nuevamente en el reservorio original. Adicionalmente, en la conexión entre los tanques existen válvulas de bola que permiten cambiar el área útil de la tubería entre los tanques. Además, para variar el caudal impulsado por la bomba se conecta la alimentación eléctrica a través de un variador de frecuencia. Este variador de frecuencia recibe una señal de voltaje y entrega energía eléctrica a la bomba con una frecuencia proporcional a la señal de voltaje. Al analizar el sistema se

asumirá que la variación del nivel del agua en el reservorio es despreciable, así como, el tiempo de estabilización de la bomba y del variador de frecuencia, en consecuencia el sistema a analizar queda resumido a una bomba cuyo caudal depende de la señal de voltaje que recibe el variador de frecuencia, y los tanques interconectados.

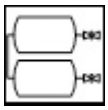


Figura 2.24: Bloque Planta de tanque de presión

Bloque Planta de Regulación de Presión(Planta de tanques de presión): Con dos depósitos de presión en serie que se analiza está constituida por un compresor de aire que envía aire hacia el primero de los dos tanques interconectados, de ahí, el airé pasa al segundo tanque para finalmente ser expulsado hacia el ambiente. Adicionalmente en la conexión entre los dos tanques, existen válvulas que permiten cambiar el área útil de la tubería entre los tanques, también dispone de una válvula a la entrada y una válvula a la salida del sistema con las mismas funciones que se conectan al PLC el cual envía una señal de 4-20ma que controla la apertura de cada una de las válvulas antes mencionadas. Al analizar el sistema se asumirá que la presión de aire de ingreso es constante y la válvula de interconexión entre los dos tanques es fija en un valor constante.

CAPÍTULO 3

3. IMPLEMENTACIÓN

En este capítulo se incluye el detalle de la implementación de cada cambio realizado en el sistema LabCon.

3.1. Listado de cambios realizados del sistema LabCon

A continuación procederemos a enlistar cada una de las implementaciones realizadas para mejorar el sistema LabCon.

3.1.1. Modificación del comportamiento del bloque SimOut

Dentro de la simulación existía un inconveniente respecto a este bloque en particular, esto se debe a que dentro de la primera versión del sistema LabCon, se creó una restricción la cual especificaba que para poder utilizar el bloque SimOut dentro de una simulación, debía existir en la misma simulación el bloque WebScope (osciloscopio), de modo que en caso de que no se incluyera este bloque WebScope dentro de la simulación aparecía un mensaje de alerta informando al usuario que se debía incluir este bloque dentro de la simulación para poder ejecutar la misma, esto como podemos observar creo una relación

de dependencia del bloque SimOut con el bloque WebScope y esta relación no es necesaria para poder utilizar el bloque SimOut y por lo tanto fue necesario eliminar esta restricción. Ahora para poder realizar este cambio se requirió modificar el archivo fuente "panel.fla" (dentro del servidor del sistema LabCon, se encuentra dentro de la ruta `C:\MATLABServerPages\webapps\ROOT\Panel`, ahora para poder editar al código fuente del mismo es necesario usar el programa Adobe Flash CS3 o Adobe Flash CS4 (mínimo se debe usar el Adobe Flash CS3 debido a que el archivo fuente fue desarrollado usando este programa en concreto).

Para poder realizar esta implementación fue necesario modificar parte del código fuente del archivo "panel.fla", escena MenuBar; específicamente se modificaron los métodos agregarMenuBar(). Para este método, la porción de código que se agrego fue la siguiente:

```
//Metodo agregarMenuBar() - linea 106 hasta linea 118
else if( item == menu.mv_workSpace ) //Cuando se cumpla este caso
{
   ocultarGraficador();



   //Se agrego para que siempre exista el bloque To WorkSpace antes de  poder
   //grabar los datos del workspace
   if( hayBloque( "To Workspace" ) )
   {
   Alert.show("Do you want to save Work Space?", "Save Work Space", Alert.OK
   |   Alert.CANCEL, null, myCH_mv_workSpace, "testIcon", Alert.OK);
   }
       else
   {
   Alert.show("Please, first add block (SimOut), then you can save.");
   }
}
```

3.1.2. Eliminación de la opción de pausa

Inicialmente, dentro de la primera versión del sistema LabCon, existían tres controles principales para el manejo de la simulación: iniciar (play), detener

(stop) y pausa (pause). Obviamente al tratarse de una simulación en donde se involucra una planta real es necesario tener solamente dos controles (iniciar y detener), debido a que la simulación tiene un inicio y un fin pero experimentalmente no se requiere y así mismo las plantas usadas dentro del sistema LabCon no poseen ninguna funcionalidad de pausa (poseen encendido y apagado) de modo que este control en particular se vuelve ineficaz al no poseer mayor uso dentro de las simulaciones en general especialmente dentro de las simulaciones que involucran una planta. Para llevar a cabo este cambio se debió modificar el archivo fuente “panel.fla” (dentro del servidor del sistema LabCon, se encuentra dentro de la ruta `C:\MATLABServerPages\webapps\ROOT\Panel`, así mismo para poder editar este archivo se recomienda usar Adobe Flash CS3 o Adobe Flash CS4 (al menos se debe usar Adobe Flash CS3 debido a que el archivo fuente original fue desarrollado usando este programa en concreto).

A continuación detalle del código que tuvo que modificarse en el archivo “panel.fla”, en la escena MenuBar:

```
//Linea 8 hasta linea 9
/*Esto es para que no se muestre el boton de stop dentro de la simulacion
apenas carga el flash en la pagina web.*/
   _root.boton_stop._visible = false;
   _root.boton_stop.enabled = false;
   /*Linea 25 hasta linea 30
   Como ya no se necesita que exista pausa dentro de la simulacion
   sencillamente se reemplaza
   por la llamada al metodo simulation_stop */
   _root.boton_pause.onPress = function()
{
   //simulation_pause(); //Eliminado la llamada a este metodo
   simulation_stop();
}
```

Por último fue necesario modificar el método onRollOver del botón “boton_pause” para que aparezca el texto “Stop”, el detalle a continuación:

```
_root.boton_pause.onRollOver = function() {
//Cambio mayo 11 En vez d mostrar texto pause ahora muestra el texto stop
tooltip_Show("Stop");
};
```

3.1.3. Mejora en la ejecución de la simulación

Al momento de ejecutar una simulación correspondiente a una práctica que involucre una planta (para poder controlar la planta de forma remota en tiempo real por medio del sistema LabCon), estas mejoras están relacionadas directamente con el control detener (stop) mientras se está ejecutando la práctica dentro de los siguientes escenarios:

- Al presionar el botón Stop mientras se ejecuta la simulación que incluya una práctica activa.
- Cuando el tiempo de ejecución (es un valor que puede ser modificado por el usuario) de la simulación ha llegado a su límite máximo.
- En el momento de salir de presionar el botón “LogOut”, esto puede ocurrir incluso mientras se encuentra trabajando con la práctica.

Es importante tener en cuenta que cada uno de los escenarios anteriormente detallados, el sistema LabCon debe incluir dentro de su librería un bloque Stop correspondiente a la planta que se está ejecutando dentro de la simulación (Ejemplo: para el bloque Tanque3T debe existir un correspondiente bloque stopPlanta3T dentro de la librería del sistema LabCon), además se debe modificar el archivo “generarModelo.jsp” (este archivo se encuentra dentro del servidor en la ruta `C:\MATLABServerPages\webapps\ROOT\Panel\funciones` para poder agregar el nuevo bloque correspondiente a la nueva planta que se

desea agregar al sistema LabCon.

Nota: Para mayor detalles relacionados a la implementación de una nueva planta ver la opción “PROCESO PARA AGREGAR UNA NUEVA PLANTA A LA LIBRERÍA DEL SISTEMA LABCON”.

3.1.4. Mejora en la presentación de las fórmulas en la simulación

Anteriormente para los bloques: Discrete Transfer Fcn, Discrete Zero-Pole, Transfer fcn y Zero Pole; se necesito cambiar la forma en que presentaba los datos, debido a que anteriormente solo presentaba el valor del numerador o denominador según sea el caso, sin tomar en cuenta que se debía presentar por medio de una fórmula; esto fue corregido para que se presente la fórmula completa con los valores y signos (+/-) que les corresponde. A continuación se presentara las imágenes de los parámetros presentados por cada uno de los bloques mencionados:

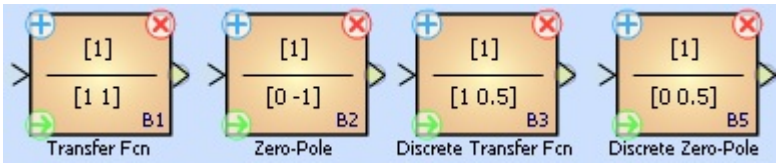


Figura 3.1: Imagénes de los bloques antes de la modificación en la presentación de las fórmulas

Para poder realizar este cambio fue necesario modificar el archivo dentro de la ruta `C:\MATLABServerPages\webapps\ROOT\Panel\panel.fla`, usando un programa Adobe Flash CS3 o Adobe Flash CS4. Luego el cambio fue realizado dentro de la clase “datos”, la función “modificarEP”, el detalle se muestra en la

código de la función desde la **línea 232 hasta la línea 293** que se muestra a continuación:

```
case "Transfer Fcn":
//numerador
    baseMC.textos.txt_varA.text = varPARAM[1].substring(1,varPARAM[1].length-1);
    var loc = varPARAM[2].indexOf("-");
        if(loc > 0){//pregunto si el valor tiene un signo negativo para el denominador
        baseMC.textos.txt_varB.text =
         's'+varPARAM[2].substring(loc,varPARAM[2].length-1);
        }else
        {
        loc = varPARAM[2].indexOf(" ");
        baseMC.textos.txt_varB.text =
        's+'+varPARAM[2].substring(loc+1,varPARAM[2].length-1);
        }
    return;
    break;
case "Zero-Pole":
     var loc = varPARAM[1].indexOf("-");
     if(loc > 0){
         baseMC.textos.txt_varA.text =
         '(s+'+varPARAM[1].substring(loc+1,varPARAM[1].length-1)+')';
     }else
     {
         baseMC.textos.txt_varA.text =
         '(s-'+varPARAM[1].substring(1,varPARAM[1].length-1)+')';
     }



    loc = varPARAM[2].indexOf("-");
    if(loc > 0)
    {//pregunto si el valor tiene un signo negativo
        baseMC.textos.txt_varB.text =
        's(s'+varPARAM[2].substring(loc,varPARAM[2].length-1)+')';
    }else
    {
    //En caso de no encontrar signo entonces busco el espacio en blanco  que separa
    //los 2 valores
        loc = varPARAM[2].indexOf(" ");
        baseMC.textos.txt_varB.text =
        's(s+'+varPARAM[2].substring(loc+1,varPARAM[2].length-1)+')';
    }
    return;
    break;
//Discrete
case "Discrete Zero-Pole" :
    var loc = varPARAM[1].indexOf("-");
    if(loc > 0){
        baseMC.textos.txt_varA.text =
        '(z+'+varPARAM[1].substring(loc+1,varPARAM[1].length-1)+')';
    }else
    {
        baseMC.textos.txt_varA.text =
        '(z-'+varPARAM[1].substring(1,varPARAM[1].length-1)+')';
    }
    loc = varPARAM[2].indexOf("-");
    if(loc > 0)
    {//pregunto si el valor tiene un signo negativo
        baseMC.textos.txt_varB.text =
        'z(z+'+varPARAM[2].substring(loc+1,varPARAM[2].length-1)+')';
    }else
    {
        loc = varPARAM[2].indexOf(" ");
        baseMC.textos.txt_varB.text =
        'z(z-'+varPARAM[2].substring(loc+1,varPARAM[2].length-1)+')';
    }
    return;
    break;
    case "Discrete Transfer Fcn":
        baseMC.textos.txt_varA.text =
        varPARAM[1].substring(1,varPARAM[1].length-1);
        var loc = varPARAM[2].indexOf("-");
        if(loc > 0){
            baseMC.textos.txt_varB.text =
            'z'+varPARAM[2].substring(loc,varPARAM[2].length-1);
        }else{
```

```
        loc = varPARAM[2].indexOf(" ");
baseMC.textos.txt_varB.text =
'z+'+varPARAM[2].substring(loc+1,varPARAM[2].length-1);
}
return;
break;
```

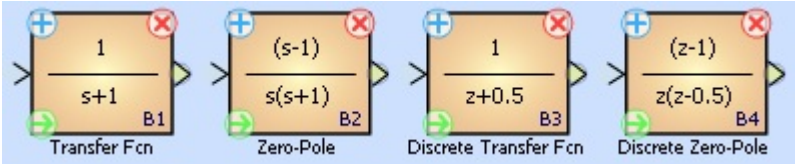


Figura 3.2: Imagénes de los bloques después de la modificación para la presentación en las fórmulas

Como podemos observar en las imágenes después de la modificación que ahora se presentan los valores de los numeradores y denominadores con la forma a la fórmula que le corresponde al bloque, esto sin importar el signo del mismo, por ejemplo, si dentro del bloque **Transfer Fcn** el valor del denominador es -1, la presentación de la fórmula lo mostrará con el signo negativo; así mismo, si para el bloque **Zero Pole** , en el numerador se asigna un valor con signo negativo en la presentación del bloque automáticamente lo mostrara con el signo positivo.

3.1.5. Proceso para agregar un nuevo bloque a la librería del sistema LabCon

Una vez que se ha identificado el nuevo bloque de tipo Simulink que se desea agregar al sistema LabCon se necesitan realizar los siguientes pasos:

Es necesario editar el archivo “grupos_bloques.xml” que se encuentra dentro de la ruta `C:\MATLABServerPages\webapps\ROOT\Panel\datos`, el cual puede ser modificado cualquier editor de texto, debido a que el archivo de extensión “xml” (en inglés las siglas se refieren a Extensible Markup Language) es básicamente un archivo de texto usado para transportar y almacenar datos

mediante un formato predefinido por el sistema que consumirá esta información. La porción de código mostrado a continuación corresponde al archivo “grupos_bloques.xml”:

```
<?xml version="1.0" encoding="utf-8"?>
<groups>
<group name="Continuous">
<block name="State-Space" npIN="1" npOUT="1" sWIN="1.0"
desc="[-BR-]State-space model:[-TAB2-] dx/dt = Ax + Bu [-TAB2-] y = Cx + Du"
enable="1">
<start name="stNameBlock" values="" />
<start name="stAccor" values="vA,|_|,x'= Ax + Bu,|_|,vB,|_|,y = Cx + Du" />
<param name="TextInput" label="A:" varname="A" def="1" />
<param name="TextInput" label="B:" varname="B" def="1" />
<param name="TextInput" label="C:" varname="C" def="1" />
<param name="TextInput" label="D:" varname="D" def="1" />
</block>
```

Una vez que tenemos abierto el archivo “grupos_bloques.xml”, debemos identificar al grupo al que pertenece el nuevo bloque que queremos agregar al sistema LabCon, por ejemplo si queremos agregar un bloque que pertenece al grupo “Discontinuidades” (en inglés “Discontinuities”). Esto se debe a que el nuevo bloque aparecerá en el menú dentro del grupo “Discontinuidades”. Ahora por ejemplo vamos a agregar el bloque “Rate Limiter” y usamos el texto que se puede observar en la porción de código a continuación:

```
<block name="Rate Limiter" npIN="1" npOUT="1" sWIN="0.6" desc="Limit rising
and falling rates of signal." enable="1" >
<start name="stNameBlock" values="" />
<param name="TextInput" label="Rising slew rate:" varname="RisingSlewLimit"
def="1"/>
<param name="TextInput" label="Falling slew rate:" varname="FallingSlewLimit"
 def="-1" />
<param name="ComboBox" label="Sample type mode:" varname="SampleTimeMode"
def="inherited,|_|,continous" validation="inherited,|_|,continous" image=""
ports="" />
<param name="TextInput" label="Initial condition:" varname="InitialCondition"
 def="0"/>
<param name="CheckBox" label="Treat as gain when linearizing"
 varname="LinearizeAsGain" def="on" validation="" image="" ports="" />
</block>
```

Como podemos observar en la porción anterior, tenemos que dentro del tag “block” existen varias propiedades descritas a continuación:

- **Name:** Se refiere al nombre del nuevo bloque, es importante tomar en cuenta que el nombre que se agrega debe ser exactamente igual al

nombre del bloque Simulink de MatLab que se desea agregar (Por ejemplo como nosotros queremos agregar el bloque “Rate Limiter”, exactamente de la misma forma debe estar escrito dentro de esta propiedad).

- **npIN:** El número de señales de entrada del bloque.
- **npOUT:** El número de señales de salida del bloque.
- **sWIN:** Defina la altura que tiene la ventana en donde se muestran las diferentes parámetros propios del bloque.
- **Desc:** Un texto que sirve de descripción para el bloque.
- **Enable:** Si el valor es igual a 1 (enable = “1”) el bloque se encontrara disponible para usar cuando se ingresa a una nueva practica en el sistema LabCon, caso contrario (enable = “0”) no podrá ser usado dentro de una simulación.

Nota: Es muy importante que se tome en cuenta que el bloque debe tener el mismo número de señales de entrada así como señales de salida como el bloque original en MatLab.

Como pudimos ver en el paso anterior, el bloque posee parámetros propios que deben ser agregados dentro del código del nuevo bloque. Como estos parámetros deben ser los mismos que el bloque original en Matlab, necesitamos revisar desde Matlab el bloque en cuestión que deseamos agregar tenemos que ingresar al Simulink Library Browser como se detalla en las dos imágenes a continuación :

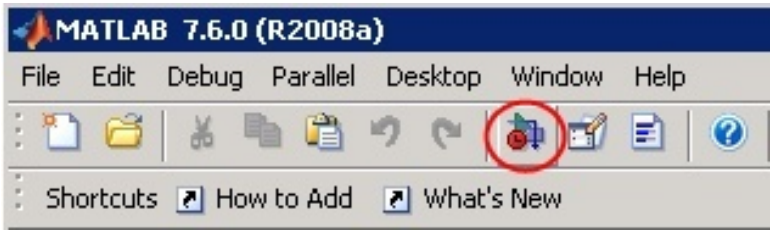


Figura 3.3: Botón de Matlab para invocar al Simulink Library Browser

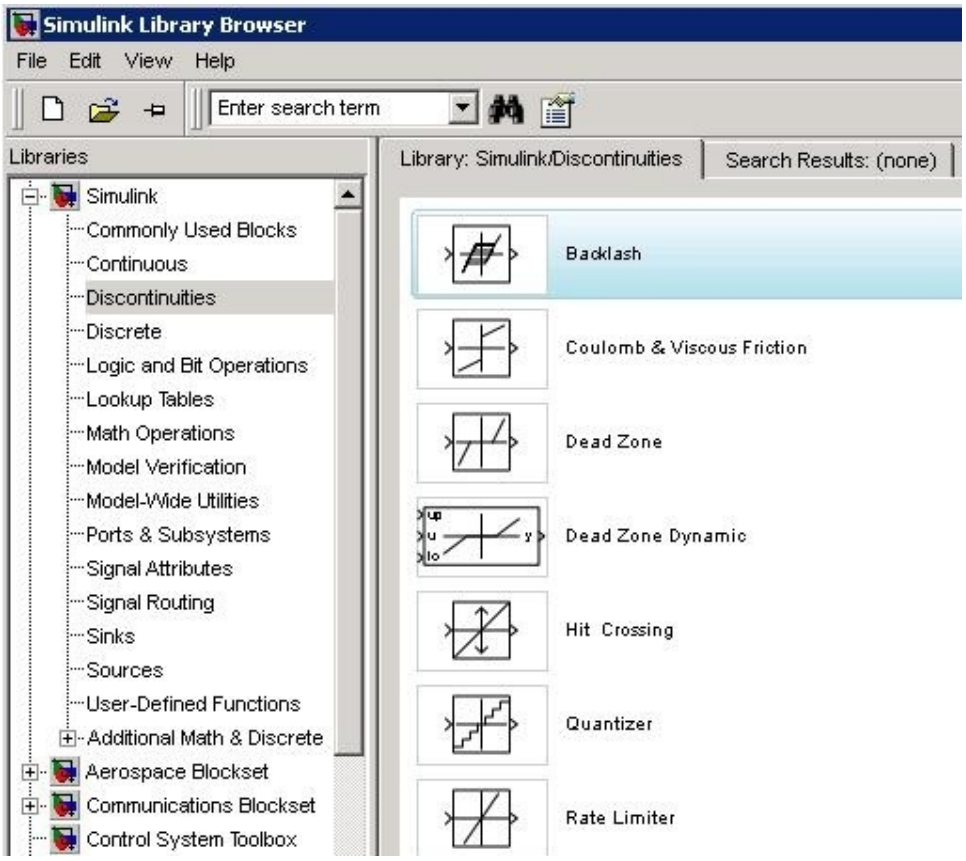


Figura 3.4: Simulink Library Browser

Al escoger dentro del Simulink Library Browser el bloque “Rate Limiter”, luego de dar click derecho se selecciona la opción “Block parameters” de modo que nos muestre la ventana de detalle de los parámetros propios del bloque.

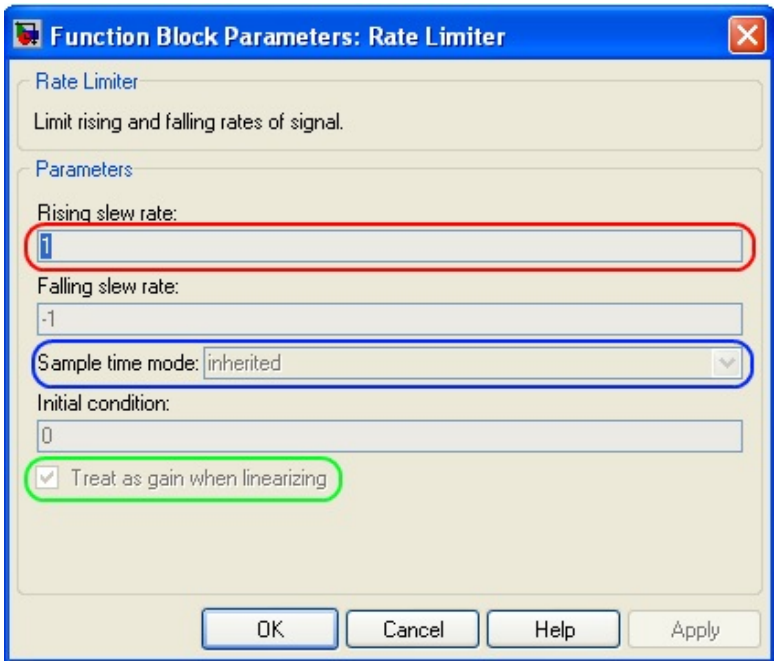


Figura 3.5: Propiedades del bloque “Rate Limiter”

En el caso del bloque “Rate Limiter” que estamos usando como ejemplo, podemos observar que existen cuatro parámetros puntuales que describiremos a continuación:

- **Param name:** Se refiere al tipo de parámetro de entrada (TextInput es una caja de texto, ComboBox para elegir varias opciones, CheckBox para elegir si se selecciona o no).
- **Label:** Es el nombre propio que aparece en la ventana de parámetros.
- **Varname:** Es el nombre de la variable con que se identificará dentro del sistema es muy importante que este nombre sea exactamente el mismo al momento de ingresarlo dentro del archivo “grupos_bloques.xml”.
- **Def:** Se refiere a los valores prederminados que deberá tener un parámetro determinado. Para un parámetro de tipo ComboBox se debe in-

gresar cada valor separado por estos caracteres “,|_|,” como este ejemplo: Valor1,|_|,Valor2, |_|,Valor3... etc.

- **Validation:** Se refiere a los valores que requieren validar esto es solo necesario para los parámetros tipo ComboBox y se usa exactamente el mismo formato para los valores.

  Ejemplo: validation = “Valor1,|_|,Valor2,|_|,Valor3”.

Una vez entendido la mecánica de la estructura de cada parámetro procedemos a agregar todos y cada uno de los parámetros que se requieren dentro de nuestro nuevo bloque “Rate Limiter” que deseamos agregar al sistema LabCon. El código final a agregar dentro del archivo “grupos_bloques.xml” debe ser como el texto que se muestra debajo:

```
<block name="Rate Limiter" npIN="1" npOUT="1" sWIN="0.6" desc="Limit rising
 and falling rates of signal." enable="1" >
<start name="stNameBlock" values="" />
<param name="TextInput" label="Rising slew rate:" varname="RisingSlewLimit"
def="1"/>
<param name="TextInput" label="Falling slew rate:" varname="FallingSlewLimit"
 def="-1"/>
<param name="ComboBox" label="Sample type mode:" varname="SampleTimeMode"
 def="inherited,|_|,continous" validation="inherited,|_|,continous" image=""
 ports=""/>
<param name="TextInput" label="Initial condition:" varname="InitialCondition"
def="0" />
<param name="CheckBox" label="Treat as gain when linearizing"
varname="LinearizeAsGain" def="on" validation="" image="" ports="" />
</block>
```

Una vez modificado el archivo “grupo_bloques.xml” con la porción de código correspondiente al nuevo bloque “Rate Limiter” procedemos a grabar y cerrar el archivo. Anteriormente mencionamos que es importante que los nombres de las variables deben ser exactamente los mismos de modo que es necesario verificar esto comparando con los nombres de las variables de los bloques originales en Matlab. Para realizar esto debemos crear un modelo sencillo que incluya únicamente al bloque que se desea agregar (Para nuestro ejemplo es el bloque “Rate Limiter”) como vemos en la siguiente imagen:

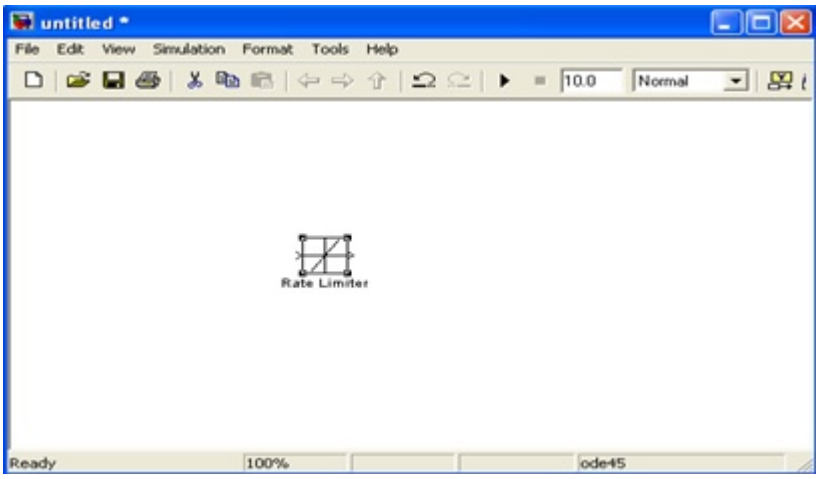


Figura 3.6: Modelo que incluye exclusivamente el bloque “Rate Limiter”

Luego de esto procedemos a grabar el modelo con el nombre que se desee dentro de una ruta de fácil acceso. Nos dirigimos a la ruta en que se grabó el modelo reciente y procedemos a abrirlo usando notepad de modo que nos muestre el código del modelo como vemos a continuación:

```
Model {
  Name   "Rate_Limiter"
  Version   7.1
  MdlSubVersion   0
  GraphicalInterface {
    NumRootInports     0
    NumRootOutports     0
    ParameterArgumentNames  ""
    ComputedModelVersion    "1.1"
    NumModelReferences     0
    NumTestPointedSignals   0
  }

    BlockParameterDefaults {
    Block {
      BlockType       RateLimiter
      RisingSlewLimit       "1"
      FallingSlewLimit       "-1"
      SampleTimeMode       "continuous"
      InitialCondition       "0"
      LinearizeAsGain       on
    }
  }
```

Luego realizamos una búsqueda con la palabra clave “BlockParameterDefaults” y nos mostrará la sección de código que incluye los nombres de las variables del bloque, estos nombres deben coincidir con los parámetros que se agregaron anteriormente en el archivo “grupos_bloques.xml” :

```
BlockParameterDefaults {
  Block {
    BlockType                RateLimiter
1   RisingSlewLimit          "1"
2   FallingSlewLimit         "-1"
3   SampleTimeMode           "continuous"
4   InitialCondition         "0"
5   LinearizeAsGain          on
  }
}
```

Figura 3.7: Los nombres de las variables con sus respectivos valores predeterminados del bloque "Rate Limiter"

Estos nombres dentro del código son los que debemos comparar para revisar si se encuentran agregados dentro del archivo “grupos_bloques.xml” correspondiente al bloque “Rate Limiter”. Dentro del archivo “grupos_bloques .xml”, el código final debe estar así:

```
1 <param name="TextInput" label="Rising slew rate:" varname="RisingSlewLimit" def="1" />
2 <param name="TextInput" label="Falling slew rate:" varname="FallingSlewLimit" def="-1" />
3 <param name="ComboBox" label="Sample type mode:" varname="SampleTimeMode" def="inherited,|_|,continous"
validation="inherited,|_|,continous" image="" ports="" />

4 <param name="TextInput" label="Initial condition:" varname="InitialCondition" def="0" />
5 <param name="CheckBox" label="Treat as gain when linearizing" varname="LinearizeAsGain" def="on" validation=""
image="" ports="" />
```

Figura 3.8: Los nombres de las variables (resaltados en color rojo) correspondientes al bloque "Rate Limiter"

Si realizamos una comparación entre los nombres de los parámetros del archivo “grupos_bloques.xml” y el modelo del bloque “Rate Limiter” como se muestra a continuación:

BlockParameterDefaults		Variables en grupos_bloques.xml
Número 1	RisingSlewLimit	varname="RisingSlewLimit"
Número 2	FallingSlewLimit	varname="FallingSlewLimit"
Número 3	SampleTimeMode	varname="SampleTimeMode"
Número 4	InitialCondition	varname="InitialCondition"
Número 5	LinearizeAsGain	varname=" LinearizeAsGain"

Figura 3.9: Párametros del bloque en Matlab (a la izquierda). Los nombres en el archivo "grupos_bloques.xml" (a la derecha)

Una vez que comparamos los nombres y se pudo confirmar que son los mismos, debemos acceder a la librería del sistema LabCon, que se encuentra en la ruta del servidor `C:\MATLABServerPages\webapps\ROOT\Code\Library`, luego procedemos a editar el archivo "libreria.mdl" usando Matlab (se recomienda realizar esto usando la versión R2008a) y veremos cada uno de los bloques que posee el sistema LabCon.

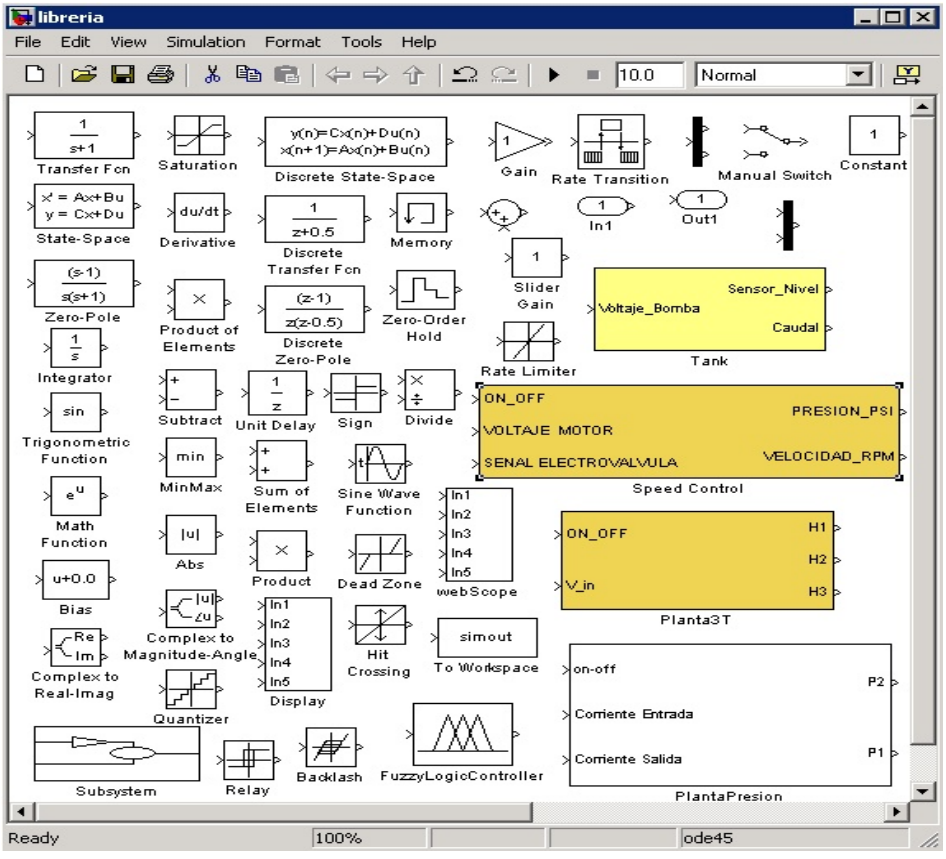


Figura 3.10: Vista del archivo "libreria.mdl" donde se muestran varios bloques que posee el sistema LabCon.

Continuando con el ejemplo dentro de nuestra librería requerimos agregar el bloque “Rate Limiter” (un bloque propio de Simulink), de modo que procedemos a abrir la opción “Simulink” y buscamos el grupo “**Discontinuities**” para poder escoger y arrastrar el bloque “Rate Limiter” correspondiente cuyo resultado debe ser como se ve en la imagen a continuación:

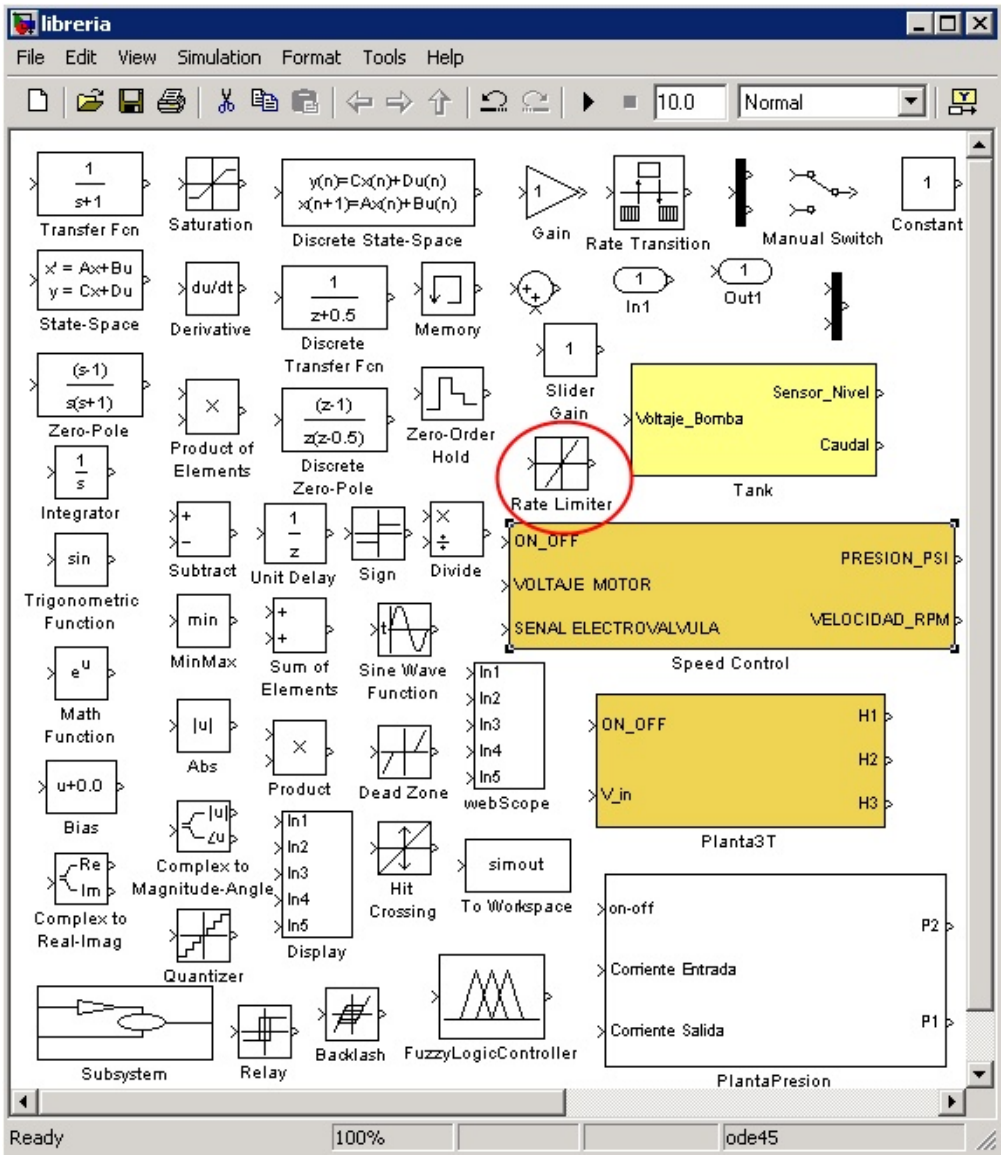


Figura 3.11: El bloque "Rate Limiter" (resaltado en color rojo) que fue agregado recientemente

Una vez que fue agregado el nuevo bloque al archivo “libreria.mdl” procedemos a grabar y cerrar el archivo. Luego de esto necesitamos configurar la imagen que se mostrará dentro del menú para el nuevo bloque, de modo que dependiendo en que grupo del menú se desee mostrar tenemos que ubicar la imagen en el correspondiente grupo a mostrar. Para esto tenemos que ubicarnos en la ruta `C:\MATLABServerPages\webapps\ROOT\Panel\images_panel\Simulink`, para nuestro caso concreto (bloque “Rate Limiter”), debemos escoger la carpeta “Discontinuities” debido a que el grupo original en Matlab precisamente es “Discontinuities”.

Al realizar esto deberíamos estar ubicados dentro del servidor en la ruta `C:\MATLABServerPages\webapps\ROOT\Panel\images\_panel\Simulink\Discontinuities`, en donde tenemos que guardar el archivo con la imagen que queremos mostrar para nuestro nuevo bloque y que debe tener el mismo nombre “Rate Limiter”.

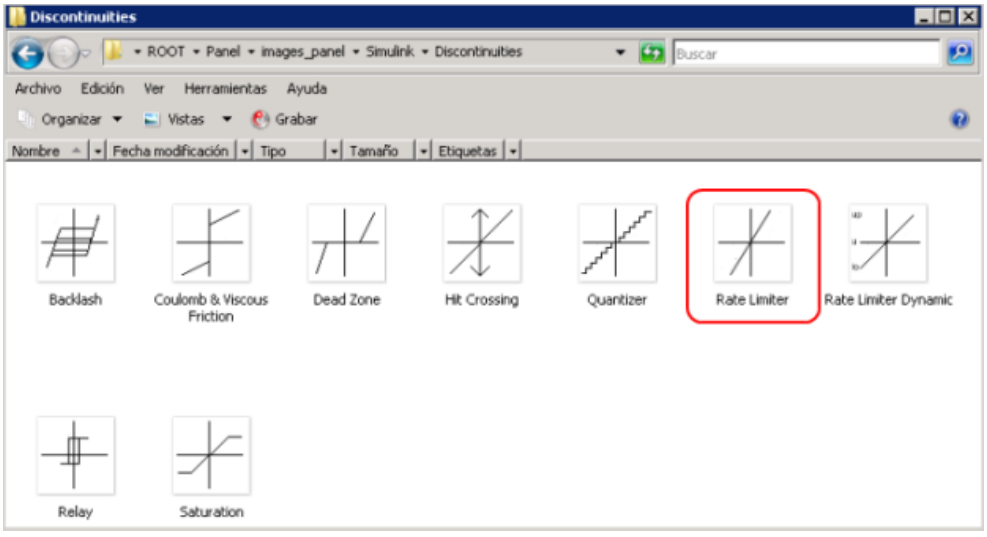


Figura 3.12: Archivo con extensión .png del nuevo bloque “Rate Limiter”

Nota: Es importante tomar en cuenta que el archivo tiene un máximo de 55 x

55 pixeles y solo puede tener formato PNG. Para confirmar que toda la configuración se realizó correctamente ingresamos al sistema LabCon y podemos observar que en el menú dentro del grupo “Discontinuities”, se muestra el nuevo bloque “Rate Limiter”.

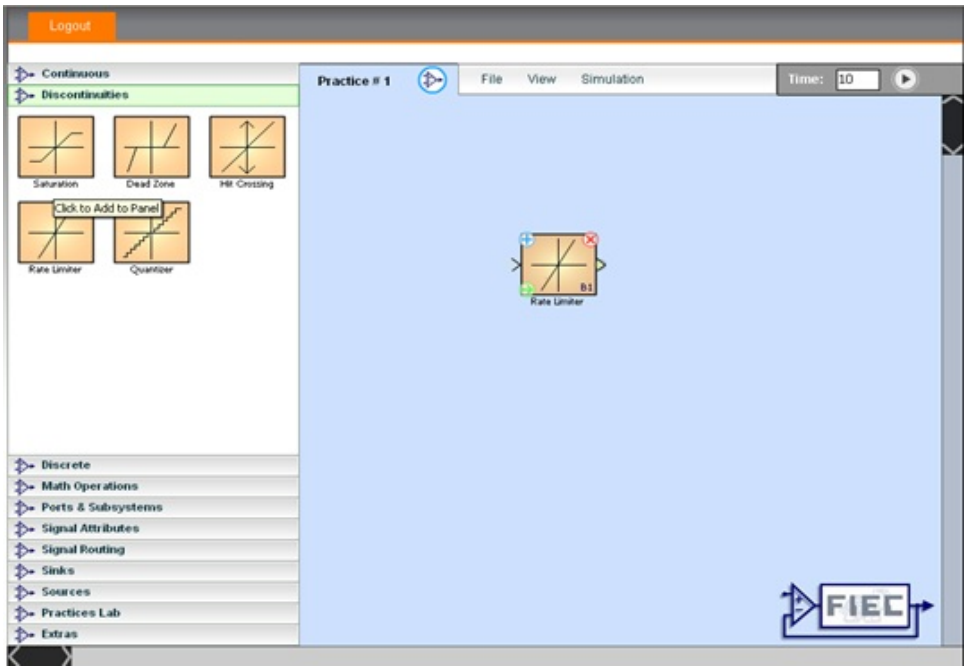


Figura 3.13:Bloque "Rate Limiter" dentro de una simulación usando el sistema LabCon

Existe un caso muy particular en el cual un bloque puede necesitar de varias imágenes como por ejemplo para el bloque “Integrator”. Para esto debemos crear una carpeta con el nombre del bloque (para nuestro ejemplo “Integrator”), de modo que la ruta en donde se deben agregar las imágenes debe ser la siguiente:

```
C:\MATLABServerPages\webapps\ROOT\Panel\images_panel

\Simulink\Integrator
```

Debido a que el bloque “Integrator” posee un parámetro “External reset” el cual

tiene varios valores a escoger como podemos ver dentro de los parámetros del bloque en la imagen a continuación:

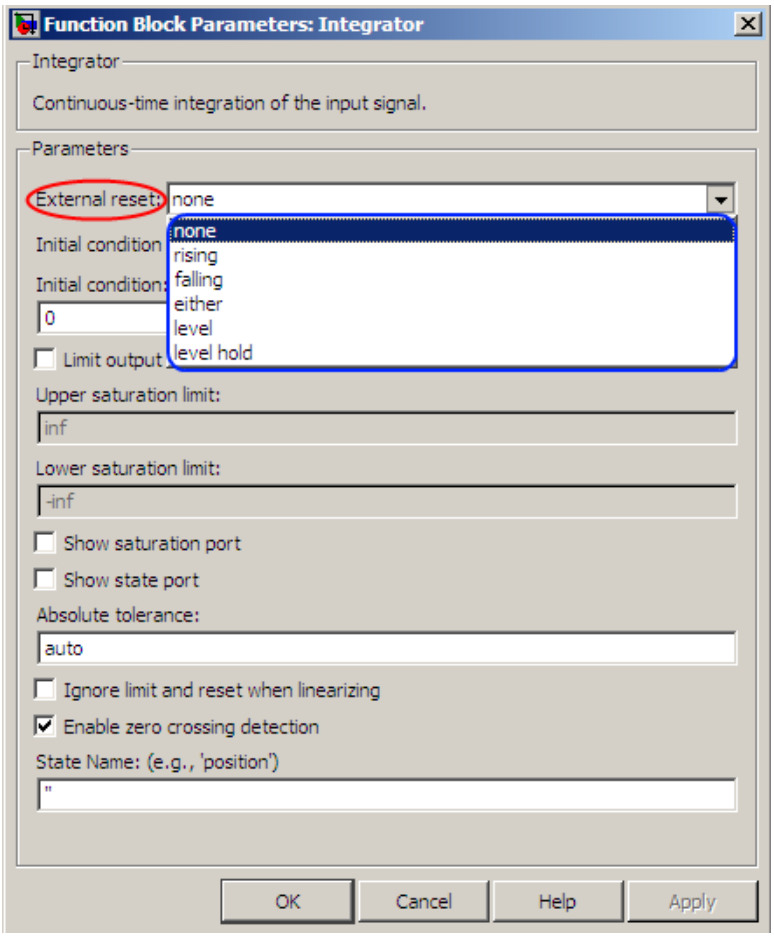


Figura 3.14: Parámetro "External reset" del bloque "Integrator" (dentro del círculo color rojo) junto a sus diferentes valores que se pueden escoger.

Como pudimos observar el parámetro “External reset” tiene varios diferentes valores (marcados dentro del circulo en azul en la imagen anterior) que vemos a continuación:

- rising.
- falling.

- either.
- level.
- levelhold.

Ahora para cada uno de estos valores es necesario crear una imagen diferente en donde se debe respetar una estructura definida, primero debe ir el nombre del parámetro separado por el símbolo guión "-" y luego el valor (Elnombredelparametro-valor), ahora si tenemos los valores anteriores deben existir los siguientes nombres para sus correspondientes imágenes para el parámetro "External Reset":

- ExternalReset-rising.png
- ExternalReset-falling.png
- ExternalReset-either.png
- ExternalReset-level.png
- ExternalReset-levelhold.png

Nota: Cuando se elige el valor "none" para el parámetro "External reset", esta imagen debe estar fuera de la carpeta origen del bloque (la carpeta en donde se deben almacenar todas las imágenes dependiendo de los valores del parámetro) para nuestro ejemplo del bloque "Integrator", la imagen para el valor "none" debe estar almacenada dentro de esta ruta `C:\MATLABServerPages\webapps\ROOT\Panel\images_panel\Simulink\`.

Debido a que el bloque "Integrator" cambia de imagen dependiendo de estos tres parámetros:

- External reset
- Initial condition source
- Limit output

Esto quiere decir que podemos tener combinaciones de entre estos tres parámetros por ejemplo:

1. Combinación con un solo parámetro.-
   - Con External reset and initial condition source.
     Ejemplo: ExternalReset-either.png
     or InitialConditionSource-external.png

     or LimitOutput-on.png.
2. Combinación usando dos parámetros.-
   - Con External reset y initial condition source.
     Ejemplo:

     ExternalReset-either_InitialConditionSource-external.png
   - Con External reset y limit output:

     Ejemplo: ExternalReset-either_LimitOutput-on.png
   - Con Initial condition source and limit output.

     Ejemplo: InitialConditionSource-external_LimitOutput-on.png
3. Combinación usando tres parámetros:
   - Con External reset y initial condition source y limit output.
     Ejemplo:

     ExternalReset-either_InitialConditionSource-external_LimitOut- puton.png

Una vez realizadas las imágenes combinando todos los parámetros entre si deberiamos obtener un total de 23 imágenes para el bloque “integrator”. A

continuación un ejemplo de las diferentes combinaciones:

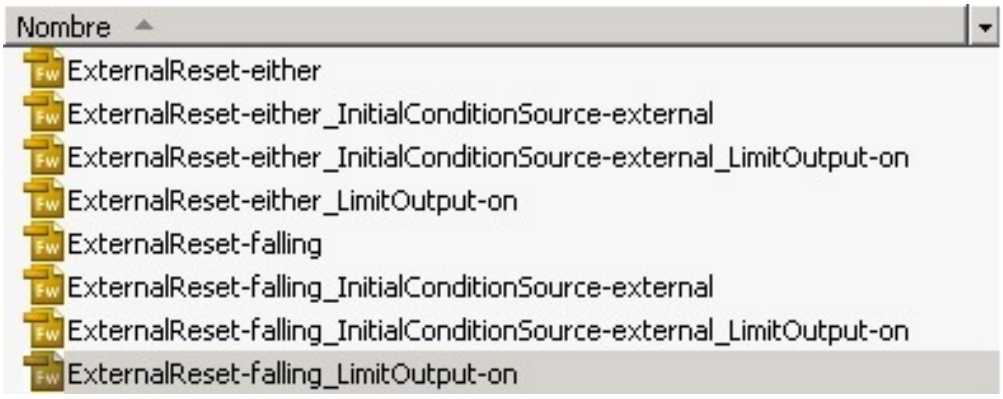


Figura 3.15: Listado cada una de las diferentes imágenes correspondientes al bloque "Integrator"

3.1.6. Proceso para agregar una nueva planta a la librería del sistema LabCon

Para agregar un bloque correspondiente a una planta existen ciertos pasos similares, sin embargo, también es necesario realizar cambios adicionales que requieren un conocimiento a nivel de programación como detallaremos a continuación:

El bloque correspondiente a una planta debe ser un modelo creado y probado previamente por el equipo encargado de su desarrollo e implementación, una vez que tenemos el archivo .mdl del bloque, es necesario también que se cree otro archivo .mdl del mismo bloque pero con la funcionalidad especifica que permita apagar la planta (se entiende que los valores para proceder a realizar el proceso de apagado de la planta). Debemos editar el archivo “grupos_bloques.xml” que se encuentra dentro del servidor en la siguiente ruta `C:\MATLABServerPages\webapps\ROOT\Panel\datos`, luego de esto procede-

mos a abrir el mencionado archivo usando algún tipo de editor de texto (podemos usar Notepad o WordPad que son programas que ya se encuentran instalados dentro del propio servidor). Una vez hecho realizamos una búsqueda usando la palabra clave “Practices Lab” como podemos observar en la imagen a continuación:

```
<group name="Practices Lab" >

    <!-- Variables de los tres tanques con agua (practica 3)-->
    <block name="Planta3T" npIN="2" npOUT="3" sWIN="1.0" desc="Bloque de Prueba"
    enable="0" >
        <start name="stNameBlock" values="" />
    </block>

    <!-- Variables del tanque del tanque con agua (practica 2)-->
    <block name="Tank" npIN="1" npOUT="1" sWIN="0" desc="[-BR-]Practice Lab Control"
    enable="0" >
        <start name="stNameBlock" values="" />
    </block>

</group>
```

Figura 3.16:Vista del archivo grupos_bloques.xml editado usando el programa Notepad ++

Así como los bloques de Simulink tenemos que las propiedades correspondientes al bloque que se pueden observar dentro del ejemplo que se encuentra a continuación:

```
<!-- Variables del tanque de presion (practica 4)-->
<block name="PlantaPresion" npIN="3" npOUT="2" sWIN="1.0" desc="Bloque de Planta de presion" enable="0" >
        <start name="stNameBlock" values="" />
        <param name="TextInput" label="ON/OFF (ON=0, OFF=1):" varname="on-off" def="0" />
        <param name="TextInput" label="Corriente Entrada: ( 0 - 9 VDC )" varname="Corriente Entrada" def="4" />
        <param name="TextInput" label="Corriente Salida: ( 0.004 - 0.02 mA )" varname="Corriente Salida" def="4" />
</block>
```

Figura 3.17: Código del bloque “PlantaPresion”

Estas son las propiedades correspondientes a la etiqueta “block” detalladas a continuación:

- **name:** Se refiere al nombre del nuevo bloque, es importante tomar en cuen-

ta que el nombre que se agrega debe ser exactamente igual al nombre del bloque de planta que se desea agregar.

- **npIN:** El número de señales de entrada del bloque.
- **npOUT:** El número de señales de salida del bloque.
- **sWIN:** Defina la altura que tiene la ventana en donde se muestran las diferentes parámetros propios del bloque.
- **desc:** Un texto que sirve de descripción para el bloque.
- **enable:** Si el valor es igual a 1 el bloque se encontrara disponible para usar cuando se ingresa a cualquier práctica en el sistema LabCon, caso contrario (enable = "0") no podrá ser usado dentro de una simulación a menos que se agregue la validación para que cuando se ingrese a su correspondiente práctica.

De modo que para agregar un nuevo bloque tenemos que saber exactamente la cantidad de señales de entrada así como la cantidad de señales de salida y en caso de requerir los parámetros que se ingresan dentro del bloque (como pudimos observar al agregar un bloque tipo Simulink).

Una vez que se encuentra listo el código correspondiente al bloque de la planta dentro del archivo "grupos_bloques.xml", procedemos a grabar el archivo.

Luego de esto es necesario dirigirnos a esta ruta dentro del servidor `C:\MATLAB ServerPages\webapps\ROOT\Code\Library`, abrimos el archivo "Libreria.mdl" usando Matlab para agregar el nuevo bloque de la planta. Para agregarlo lo único que tenemos que hacer es desde el archivo original de extensión mdl del bloque de la planta se copia hacia el archivo "Libreria.mdl" y procedemos a grabar.

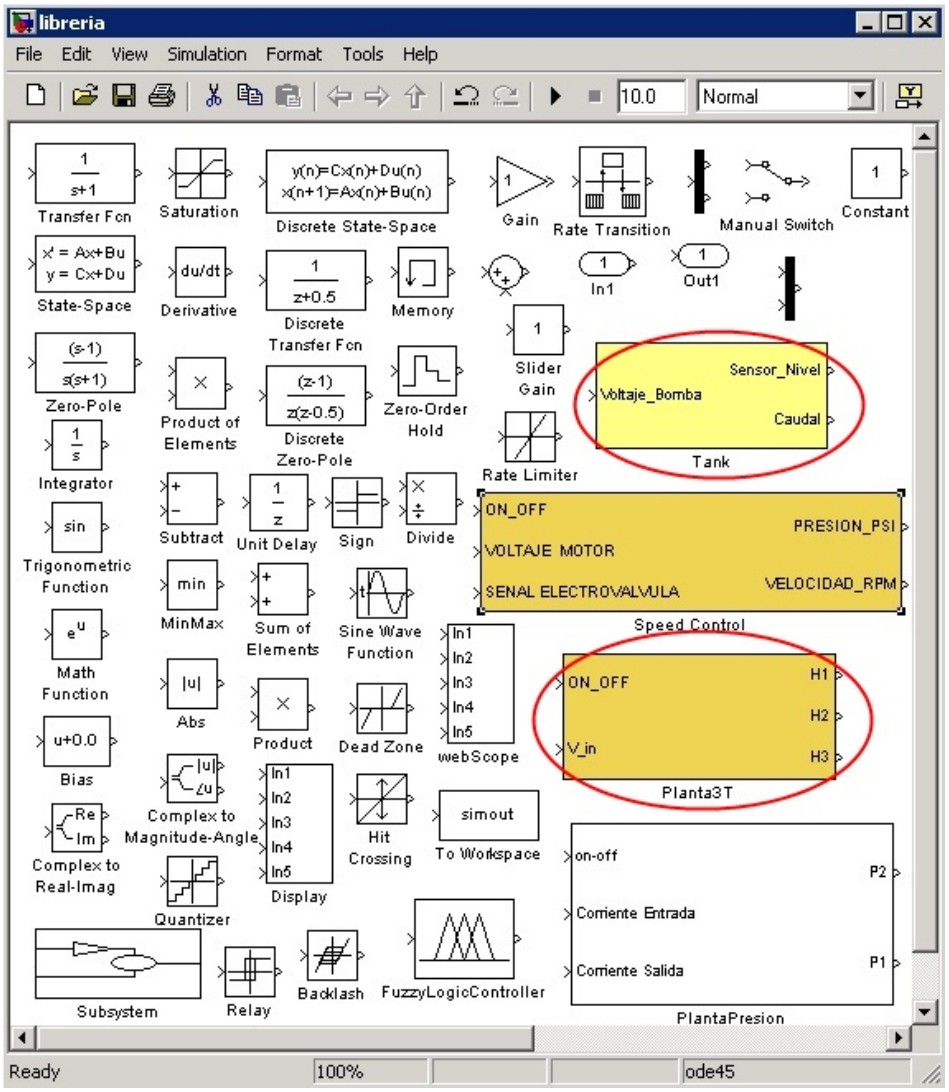


Figura 3.18: Archivo "Libreria.mdl" (los bloques marcados con círculos rojos corresponden a bloques de plantas)

Ahora recordemos que para agregar un nuevo bloque de una planta al sistema LabCon se requieren de dos bloques en total, el bloque de planta original y el mismo bloque de la planta pero con los valores necesarios para proceder a apagar la planta o que básicamente la obligue a apagarse. Este bloque de apagado también debe ser proporcionado por el mismo equipo que desarrolló el bloque original de la planta o en su defecto por una persona con los

conocimientos suficientes para poder configurar el bloque con los valores para realizar el proceso de apagado de la planta. Para esto nos ubicamos en la misma ruta del archivo “libreria.mdl” (`C:\MATLABServerPages\webapps\ROOT\Code\Library`), y procedemos a grabar el archivo de apagado de preferencia con la siguiente estructura, stop seguido del nombre del bloque por ejemplo así: stopPlanta3T.mdl; de modo que dentro de la misma ruta se ubique este archivo del bloque para realizar el apagado de la planta.

Una vez que tenemos listo los bloques de funcionamiento y apagado de la planta, es necesario realizar los cambios que permitirán al sistema LabCon invocar estos bloques dependiendo del caso. Para esto es necesario ubicarnos en la ruta del servidor:

`C:\MATLABServerPages\webapps\ROOT\Panel\funciones`

Una vez que nos encontremos en la dirección antes mencionada, procedemos a editar el archivo “procesarSesionPanel.jsp” (podemos usar cualquier editor de texto para abrir este archivo). Una vez abierto nos ubicamos en la opción 11 (línea de código 108) que corresponde al escenario cuando el usuario presiona el botón “LogOut” para salir de la simulación, esto justamente invoca el bloque de apagado (dependiendo de la práctica invoca un bloque de apagado particular relacionado con esa práctica en cuestión). Dentro del código podemos observar comandos condicional “If”, en el cual tenemos que dependiendo del número de la práctica se ejecuta un particular bloque stop (cada práctica se encuentra asociada a una planta en particular). Esto quiere decir que si deseamos agregar una nueva planta esta debería estar asociada a una práctica en particular (de una práctica nueva que no se encuentre asociada a alguna otra planta ya perteneciente al sistema LabCon). De modo que si deseamos agregar una nueva planta obviamente tendríamos que asociar esa planta a una nueva práctica, definido esto agregamos la porción de código co-

rrespondiente debajo de la última práctica (actualmente la última práctica es la número cuatro correspondiente a la planta del tanque de presión). En todo caso el código que debe agregarse se muestra a continuación:

```
if(numeroPractica.equals("numero correspondiente a la practica que se
desea agregar"))
{
%>
<matlab:Command cmd="<%= "open_system('Library/stopNombredelBloque.mdl')" %>"/>
    <matlab:Command cmd="<%= "set_param('stopNombredelBloque',
             'SimulationCommand','Start')" %>"/>
<%
}
```

Como vemos dentro del código se están ejecutando comandos propios de matlab, entre ellos tenemos:

- Open_System: que permite abrir un archivo de extensión “mdl” y como parámetro recibe la ruta en donde se encuentra el modelo.
- Set_param: permite ejecutar un comando que permite establecer un valor para un bloque simulink específico.

  Ejemplo: set_param('stopPlanta3T', 'SimulationCommand','Start') para iniciar el bloque stopPlanta3T.

Nota: Para mayor detalle acerca de la instrucción “set_param” dirigirse a la bibliografía.

```
if(numeroPractica.equals("4"))
{
%>
<matlab:Command cmd="<%= "open_system('Library/stopPlantaPresion.mdl')" %>"/>
<matlab:Command cmd="<%= "set_param('stopPlantaPresion','SimulationCommand',
'Start')" %>"/>
<%
}
```

Una vez que se agregó la porción de código correspondiente al nuevo bloque como se pudo apreciar en la imagen anterior, procedemos a grabar el archivo. Luego de esto debemos configurar la relación entre el nuevo bloque de la

planta con la práctica (como vimos anteriormente que debemos relacionar un bloque de planta con una práctica) esto se hace dentro del mismo archivo “procesarSesionPanel.jsp”, en la línea de código 210 dentro de la siguiente porción de código:

```
case 20: //Retorna el bloque de practica que debe ser activado (lo  busca en
el archivo Panel\Datos\grupos_bloques.xml)
switch( Integer.parseInt( request.getParameter("numPRACT") ) )
{
case 4:
   out.print( "&practice=PlantaPresion&" );
   break;
case 3:
   out.print( "&practice=Planta3T&" );
   break;
case 2:
   out.print( "&practice=Tank&" );
   break;
case 1:
   out.print( "&practice=Speed Control&" );
   break;
default:
   out.print( "&practice=ERROR&" );
   break;
}
break;
default:
   out.print( "&estado=ERROR_CASE&" );
   break;
```

Dentro del código anterior podemos observar (marcado en amarillo) que dependiendo del parámetro “numPRACT” corresponde al número de la práctica y dependiendo de este valor se invocará dentro menú “Practices Lab” del sistema LabCon debería estar disponible solo el correspondiente bloque de planta (si es la práctica uno dentro del menú “Practices Lab” se podrá escoger únicamente el bloque “Speed Control” relacionado a esa práctica). Una vez que se agregó la porción de código correspondiente a la práctica junto con los comandos para invocar al nuevo bloque de la planta (el nombre de la planta debe ser el mismo al nombre del bloque que se agregó en la librería del sistema LabCon), debemos proceder a grabar el archivo “procesarSesionPanel.jsp” con los cambios de la nueva práctica y damos por terminado las modificaciones a realizar con este archivo.

Existe otro archivo “generarModelo.jsp” que se ejecuta al momento de iniciar una simulación dentro del sistema LabCon, es necesario realizar una modifica-

ción dentro del mismo debido a que este invoca al bloque de stop dependiendo de la práctica en la que se está trabajando. El archivo se encuentra dentro de la misma ruta que el archivo “procesarSesionPanel.jsp”, procedemos a editar el archivo “generarModelo.jsp” usando algún editor de texto que se encuentre disponible dirigimos a la línea de código 165 donde podemos ver la siguiente porción de código a continuación:

```
case 3:
%>
<!-- DETENER -->
   <matlab:SessionGet name="<%= modeloNombre %>"/>
   <matlab:Command cmd="<%= "set_param('" + modeloNombre + "','SimulationCommand',
   'Stop')" %>"/>
   <matlab:Command cmd="<%= "pause(2)" %>"/>
<%
//Abre el modelo stop motor para que apague la planta si pone stop sin apagar.....
//dependiendo del numero de practica en este caso la 2 para el motor y
la 3 para el tanque
//
if(numeroPractica.equals("1")){
%>
       <matlab:Command cmd="<%= "open_system('Library/stopMotor.mdl')" %>"/>
       <matlab:Command cmd="<%= "set_param('stopMotor','SimulationCommand',
       'Start')" %>"/>
<%
}
if(numeroPractica.equals("2")){
%>
       <matlab:Command cmd="<%= "open_system('Library/stopTanq.mdl')" %>"/>
       <matlab:Command cmd="<%= "set_param('stopTanq','SimulationCommand','Start')" %>"/>
<%
}
if(numeroPractica.equals("3")){
%>
       <matlab:Command cmd="<%= "open_system('Library/stopPlanta3T.mdl')" %>"/>
       <matlab:Command cmd="<%= "set_param('stopPlanta3T','SimulationCommand',
       'Start')" %>"/>
<%



}
if(numeroPractica.equals("4")){
%>
        <matlab:Command cmd="<%= "open_system('Library/stopPlantaPresion.mdl')" %>"/>
        <matlab:Command cmd="<%= "set_param('stopPlantaPresion','SimulationCommand',
        'Start')" %>"/>
<%
}
out.print( "&estado=Stopped&" );
%>
<%
break;
```

Como podemos observar dentro de la variable “numeroPractica”, se encuentra el número de la práctica que se envía a este archivo justo cuando se inicia una simulación, luego dependiendo del número de la práctica, al momento en que se ejecuta la simulación y se presiona el botón de detener (Stop), inmediatamente se envía el número de la práctica a este archivo que dependiendo del

valor de la práctica procederá a invocar su correspondiente bloque de apagado. Para poder agregar una nueva planta debemos agregar un nuevo elemento condicional if con este formato de ejemplo:

```
if(numeroPractica.equals("Numerodelapractica")){
%>
       <matlab:Command cmd="<%= "open_system('Library/NombredelBloque.mdl')" %>"/>
       <matlab:Command cmd="<%= "set_param('NombredelBloque','SimulationCommand',
       'Start')" %>"/>
<%
```

Lo que vemos dentro del comando “numeroPractica” corresponde a los valores a modificar dependiendo de la práctica asociada al nuevo bloque de planta, de modo que los valores marcados son los que deberíamos editar y luego agregar dentro de la instrucción select: case y cuyo resultado debería ser similar al código siguiente:

```
%>
<!-- DETENER -->
<matlab:SessionGet name="<%= modeloNombre %>"/>
<matlab:Command cmd="<%= "set_param('" + modeloNombre + "','SimulationCommand',
'Stop')" %>"/>
    <matlab:Command cmd="<%= "pause(2)" %>"/>
<%
//Abre el modelo stop motor para que apague la planta si pone stop sin apagar.....
//dependiendo del numero de practica en este caso la 2 para el motor y la 3 para el
tanque
if(numeroPractica.equals("1")){
%>
        <matlab:Command cmd="<%= "open_system('Library/stopMotor.mdl')" %>"/>
        <matlab:Command cmd="<%= "set_param('stopMotor','SimulationCommand',
        'Start')" %>"/>
<%
}


if(numeroPractica.equals("2")){
%>
        <matlab:Command cmd="<%= "open_system('Library/stopTanq.mdl')" %>"/>
        <matlab:Command cmd="<%= "set_param('stopTanq','SimulationCommand',
        'Start')" %>"/>
<%
}
if(numeroPractica.equals("3")){
%>
        <matlab:Command cmd="<%= "open_system('Library/stopPlanta3T.mdl')" %>"/>
        <matlab:Command cmd="<%= "set_param('stopPlanta3T','SimulationCommand'
        ,'Start')" %>"/>
<%
}
if(numeroPractica.equals("Numero de la nueva practica")){
%>
        <matlab:Command cmd="<%= "open_system('Library/NombredelnuevoBloque.mdl')" %>"/>
        <matlab:Command cmd="<%= "set_param(NombredelnuevoBloque,'SimulationCommand',
        'Start')" %>"/>
<%
}
out.print( "&estado=Stopped&" );
%>
<%
break;
```

Toda la porción de código final que se debe agregar corresponde al bloque donde el comando “numeroPractica” tiene como valor “Numero de la nueva practica”. Luego de realizados estos cambios procedemos a grabar el archivo “generarModelo.jsp”.

Como último paso debemos agregar una imagen para el nuevo bloque de planta, para esto necesitamos ubicarnos dentro del servidor en la ruta `C:\MATLABServerPages\webapps\ROOT\Panel\images_panel\Simulink\PracticesLab\`, que es justamente donde se almacenan las imágenes para los bloques correspondientes a las plantas.

Colocamos el archivo que debe tener un máximo de 55 x 55 pixeles y solo con formato PNG, dentro de esta ruta con el mismo nombre del bloque (esto es importante debido a que si tiene un nombre diferente no podrá ser identificado por el sistema LabCon y por lo tanto al ingresar al sistema dentro del menú no podrá invocarse la imagen del bloque). Una vez realizado esto al momento de ingresar para poder confirmar que la imagen puede ser visualizada en el nuevo bloque de la planta, ingresamos al sistema LabCon y escogemos la práctica asignada al nuevo bloque y dentro del menú “Practices Lab” debe estar el nuevo bloque con la imagen que se agregó anteriormente.

3.1.7. Creación del nuevo bloque Fuzzy Logic Controller

Debido a la necesidad de realizar prácticas con un grado mayor de complejidad se tuvo que crear el bloque “Fuzzy Logic Controller”, como ya se ha visto anteriormente se han seguido los mismos pasos para agregar un nuevo bloque al sistema LabCon, sin embargo, para el correcto funcionamiento existe un parámetro llamado “fis” en donde se especifica el nombre del archivo tipo

fis que se requiere para la configuración del respectivo bloque. Para agregar el bloque Simulink “Fuzzy Logic Controller” al sistema LabCon debemos seguir los pasos especificados en el punto 3.1.5 PROCESO PARA AGREGAR UN NUEVO BLOQUE A LA LIBRERÍA DEL SISTEMA LABCON.

El proceso de carga del archivo de configuración tipo fis, se hace mediante una ventana emergente (tipo popup) dentro de la opción “Fuzzy” que aparecerá en el menú, justo debajo del listado de prácticas como se podrá observar en la siguiente imagen:

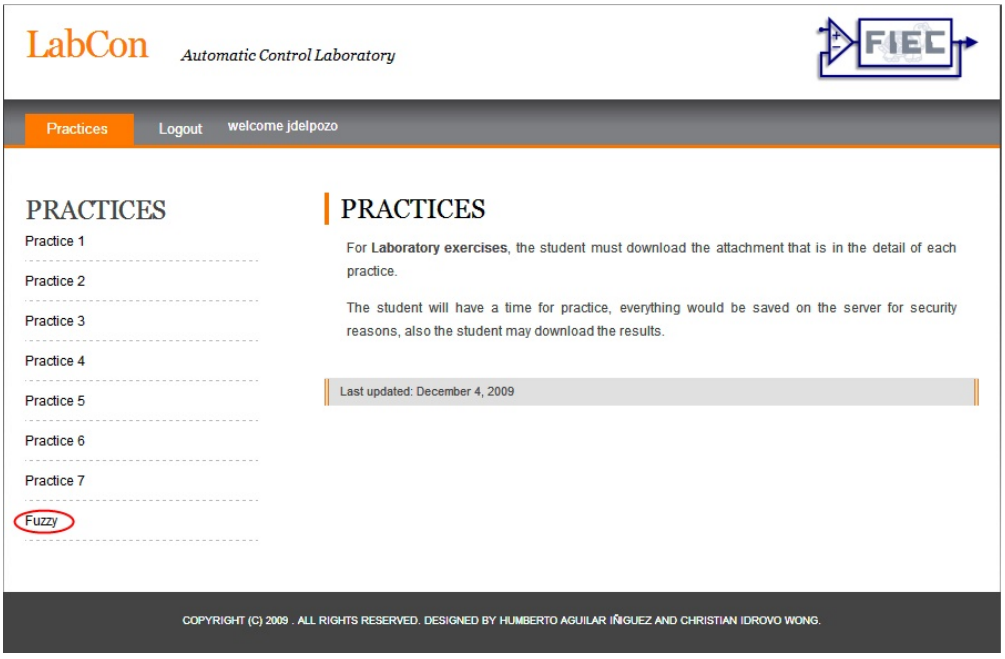


Figura 3.19: Opción en menú "Fuzzy" que se debe elegir para poder subir el archivo de control con extensión "fis".

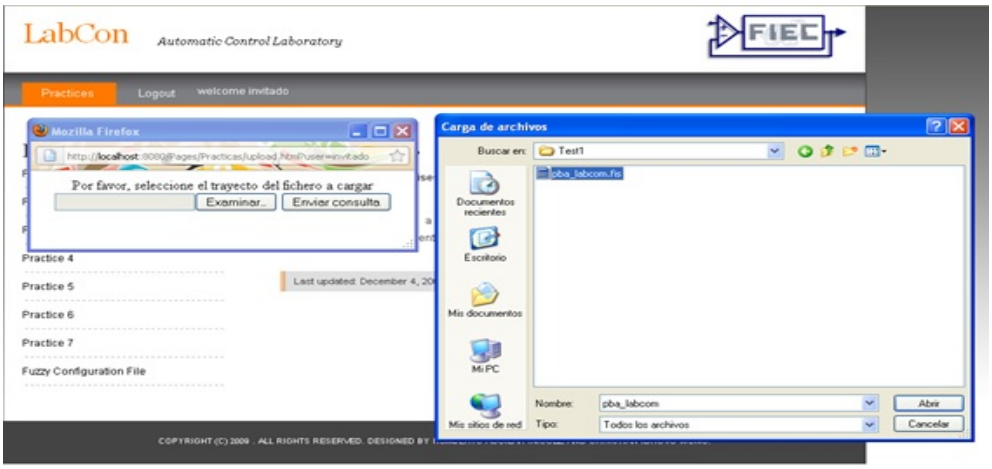


Figura 3.20: Ventana de búsqueda para escoger la ruta en donde se encuentra almacenado el archivo de extensión .fis que se requiere subir al sistema LabCon.

Luego de escoger el archivo, debe presentar un mensaje de confirmación detallando si la operación fue exitosa como vemos a continuación:

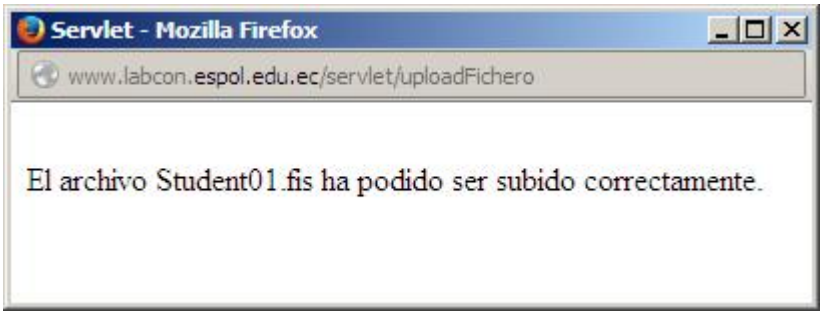


Figura 3.21: Mensaje que nos indica que el archivo .fis fue ingresado correctamente al sistema.

Nota: El archivo del ejemplo originalmente se llama pba_labcom.fis pero al subirlo al sistema LabCon será renombrado con el siguiente nombre “nombredelusuario.fis” dentro de la ruta `C:\MATLABServerPa ges\webapps\ROOT\Panel\archivos_config\`, debido a que al momento de crear el modelo en matlab es necesario saber que archivo tipo fis se debe cargar en el bloque Fuzzy Logic Controller de modo que se relaciona el modelo con el usuario.

Para poder subir este archivo al servidor se creó un servlet llamado uploadFichero, que debe ser declarado dentro del archivo web.xml en la carpeta WEB-INF del sistema LabCon. Para mayor detalle estas fueron las líneas de código que se tuvieron que incluir para la inclusión de este servlet **uploadFichero** (4):

```
<servlet>
        <servlet-name>uploadFichero</servlet-name>
        <servlet-class>uploadFichero</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>uploadFichero</servlet-name>
        <url-pattern>/servlet/uploadFichero</url-pattern>
</servlet-mapping>
```

Luego fue necesario la creación del archivo uploadFichero.class, a continuación se presentará el código fuente del mismo:

```
/* uploadFichero.java
 *
 * Created on August 4 2003, 22:26
 * Modified on September 15 2011, 11:45
 */
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.apache.commons.fileupload.*;
import java.util.*;
/**
 *
 * @author  Roberto Canales
 * @Modified by Cristhian Arroba Rivera
 */
public class uploadFichero extends HttpServlet {
public void doGet(HttpServletRequest request,HttpServletResponse response)
    throws ServletException, IOException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
    void depura(String cadena)
    {
        System.out.println("El error es " + cadena);
    }
    public boolean procesaFicheros(HttpServletRequest req, PrintWriter out )
    {
        try {
            // construimos el objeto que es capaz de parsear la pericion
            DiskFileUpload fu = new DiskFileUpload();
            // maximo numero de bytes
            fu.setSizeMax(1024*512); // 512 K
            depura("Ponemos el tamanio maximo");
```

```
            // tamanio por encima del cual los ficheros son escritos en disco
            fu.setSizeThreshold(4096);
            // directorio en el que se escribiran los ficheros con tamanio superior al
            //soportado en memoria
            fu.setRepositoryPath("/tmp");
            // ordenamos procesar los ficheros
            List fileItems = fu.parseRequest(req);
            if(fileItems == null)
            {
                depura("La lista es nula");
                return false;
            }
String usuario="";
// Process the uploaded items
Iterator iter = fileItems.iterator();
//Para obtener el usuario
while (iter.hasNext())
{
     FileItem item = (FileItem) iter.next();
     if (item.isFormField())
{
         // get the name of the field
String fieldName = item.getFieldName();
if(fieldName.equals("user"))
 depura("el usuario es: "+item.getString());
usuario = item.getString()+".fis";
     }
}
            // Iteramos por cada fichero
Iterator i = fileItems.iterator();
            FileItem actual = null;
            depura("estamos en la iteracion");
String archivo;
            while (i.hasNext())
            {
//esta linea es solo de prueba para saber que hay en cada elemento
             //depura("elemento"+ i.next());
                actual = (FileItem)i.next();
                String fileName = actual.getName();
                // construimos un objeto file para recuperar el trayecto completo
                File fichero = new File(fileName);
                depura("El nombre del fichero es " + fichero.getName());
                archivo = fichero.getName();
                depura("el archivo es: "+archivo);
                if (archivo.indexOf(".fis") > 0)
   {
   //La extension del archivo a subir debe ser tipo fis
out.println("<br>El archivo " + usuario + " ha podido ser subido correctamente.");
// nos quedamos solo con el nombre y descartamos el path
fichero = new  File("c:\\MATLABServerPages\\webapps\\ROOT\\Panel\\archivos_config\\"
+ usuario);
                 // escribimos el fichero colgando del nuevo path
                 actual.write(fichero);
                }
                else{
                 out.println("<br> El archivo "+ fileName + " no es de tipo fis");
                 depura("El archivo no es de tipo fis");
                 return false;
                }
            }
        }
        catch(Exception e) {
            depura("Error de Aplicacion " + e.getMessage());
            return false;
        }
        return true;
    }
    /** Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {
        //processRequest(request, response);
String parameter = "user";
        String user = request.getParameter(parameter);
        PrintWriter out = response.getWriter();
```

```
        //out.println("El user es "+user);
     //out.println("Hello World");
        response.setContentType("text/html");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        System.out.println("Comenzamos procesamiento ficheros1");
        procesaFicheros(request,out);
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
}
```

Finalmente se necesita enviar el nombre del archivo del panel.swf (desde flash hacia matlab) por lo que se modificó el código fuente del mismo:

Clase Exportar: Línea 84 hasta línea 91

```
//Para modificar el parametro fis del bloque Fuzzy Logic Controller
if(tmpEP.VAR_NOMBRE[j] == "fis")//Recordemos que aqui se busca de un listado de
                                //parametros (letra j) no de bloques
{
//_root.ACTIVE_username + "_p" + _root.ACTIVE_numPractice
tmpEP.VAR_VALORFINAL[j] ="''" + _root.ACTIVE_username + ".fis''";
}
```

Como se detalla en los comentarios el parámetro fis se encuentra dentro de una lista de parámetros para todos los bloques del modelo, lo que hago es que en el momento de encontrar el parámetro fis lo agrego el correspondiente nombre del archivo compuesto por el nombre del usuario y extensión fis (nombredelusuario.fis), de modo que al crear el modelo automáticamente el bloque Fuzzy Logic Controller ya tenga cargado el valor del parámetro fis con el correspondiente nombre de archivo de configuración tipo fis.

3.2. Listado de nuevos bloques

Cada uno de los bloques de este listado fue implementado usando los procedimientos anteriormente mencionados en este documento, este es el listado de nuevo bloques (28 en total) clasificados por tipo:

1. **Continuous.- (4 en total)**
    - Variable Transport Delay.
    - Variable Time Delay.
    - Transport Delay.
    - Derivate.
2. **Discontinuities.- (8 en total)**
    - Backlash.
    - Relay
    - Rate Limiter.
    - Rate Limiter Dynamic.
    - Quantizer.
    - Hit Crossing.
    - Dead Zone.
    - Coulomb & Viscous Friction.
3. **Discrete.- (2 en total)**
    - Difference.
    - Memory.
4. **Math Operations.- (9 en total)**
    - Bias.
    - Abs.
    - Complex to Real-Imag.
    - Complex to Magnitude-Angle.
    - Math Function.

- Sine Wave Function.
- Sign.
- Product.
- Trigonometric Function.

5. **Signal Routing.- (1 en total)**
   - Demux.
6. **Sinks.- (1 en total)**
   - Display (Bloque especial).
7. **Practices Lab.- (2 en total)**
   - Planta 3T (Bloque especial, corresponde a una nueva práctica).
   - Planta Presion (Bloque especial, corresponde a una nueva práctica).
8. **Extras.- (1 en total)**
   - Fuzzy Logic Controller.

CAPÍTULO 4

4. PRUEBAS Y ANÁLISIS DE RESULTADOS

Durante este capítulo se explican cada una de las diferentes pruebas realizadas para comprobar el funcionamiento del bloque “Fuzzy Logic Controller”.

4.1. Pruebas Unitarias

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código dentro de un sistema en particular. Esto sirve para asegurar que cada uno de los diferentes módulos funcione correctamente por separado dentro del sistema.

Sin embargo, luego es necesario realizar una prueba integral (con todos los módulos o componentes del sistema) estas pruebas se las conoce como Pruebas de Integración, y dependiendo de su resultado se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión. Para la realización de las pruebas unitarias en cada bloque se creó un modelo sencillo usando una fuente de señal senosoidal y para verificar los resultados un osciloscopio, como se puede apreciar en la imagen a continuación:

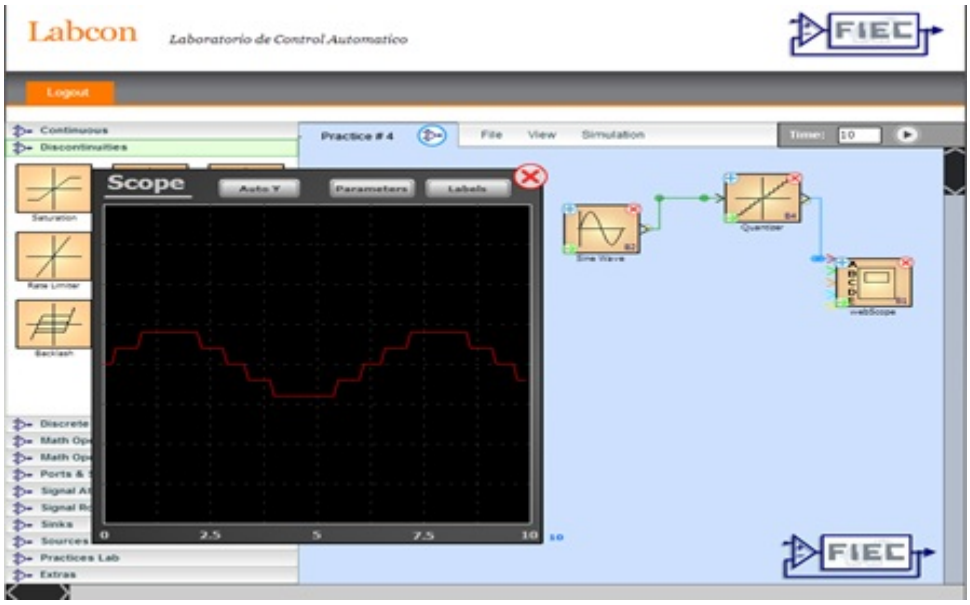


Figura 4.1: Señal de entrada de la onda senosoidal después de pasar por el bloque Quantizer.

Cada nuevo bloque fue probado usando este esquema obteniendo resultados satisfactorios, sin embargo es muy importante anotar que existió un tipo de inconveniente muy puntual (no se recibía señal de salida) y al revisar detenidamente se debió a que se encontró una falla en la configuración inicial del bloque (ver detalles de creación y configuración de un nuevo bloque al sistema LabCon), como por ejemplo que el nombre de un parámetro del bloque se encontraba mal escrito (siempre el nombre de cada parámetro de un bloque deben ser exactamente los mismos comparados con los nombres de los parámetros de los bloques en MatLab), caso contrario el sistema LabCon no ejecuta de forma correcta la simulación con el bloque en cuestión.

4.2. Pruebas de integración con MatLab Server Pages (MSP)

Pruebas integrales o mejor conocidas como pruebas de integración son aquellas que se realizan después vez que se han realizado las pruebas unitarias y los resultados de las mismas son satisfactorios. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un sistema.

Estas pruebas fueron usadas para realizar las pruebas de los bloques correspondientes a las plantas, debido a que estos bloques especiales requieren de una simulación con una mayor complejidad a los bloques tipo simulink. Ahora es importante tener en cuenta que para cada una de las simulaciones se deben probar estos escenarios:

1. Mientras la simulación se está ejecutando y el usuario se desconecta del sistema LabCon (presiona la opción Logout), automáticamente el sistema debe apagar la planta, esto lo hace al ejecutar un bloque especial de apagado (este bloque es básicamente el mismo bloque inicial de la planta pero con los valores de voltaje igual a 0 de modo que la planta se apague)
2. En medio de la ejecución de una simulación y el usuario presiona el botón de detener (stop), el sistema debe detener la planta mediante el bloque especial de apagado.
3. Finalmente, cuando la simulación alcance el valor máximo de tiempo de ejecución , el sistema debe invocar de forma automática el correspondiente bloque de apagado para la planta.

Nota: Es importante tener en cuenta que cada planta debe tener su correspondiente bloque de apagado. Así mismo para cada una de las pruebas los bloques

de las plantas (tanto como el bloque principal como el de apagado deben estar correctamente configurados por sus creadores) y se debe tener en cuenta que cada planta debe estar conectada con CFP (Compact Field Point) que es el componente encargado de la comunicación entre la planta y el sistema LabCon.

4.3. Pruebas de usuario y estabilización

Aunque los usuarios en general comparten algunas características comunes, (tales como memoria o tiempo de reacción), éstos poseen un amplio conocimiento de capacidades físicas, necesidades, expectativas a futuro. Las pruebas de usabilidad (también conocidas como pruebas de usuario) es una técnica usada en el diseño de interacciones centrado en el usuario para evaluar un producto mediante pruebas con los usuarios mismos. Esto puede ser visto como una práctica de usabilidad irreemplazable, dado que entrega información directa de como los usuarios reales utilizan el sistema. Las pruebas de usabilidad se enfocan en medir la capacidad de un producto (en nuestro caso de un software) en cumplir el propósito para el cual fue creado. Ejemplos de productos que normalmente se benefician de pruebas de usabilidad son comidas, productos de consumo, sitios web o aplicaciones web, interfaces de usuario, documentos y dispositivos. Las pruebas de usabilidad miden la usabilidad, o facilidad de uso, de un objeto específico o un conjunto de objetos, mientras que los estudios de interacción persona-computador intentan formular los principios generales. Las pruebas de usabilidad consisten en seleccionar a un grupo de usuarios de una aplicación y solicitarles que lleven a cabo las tareas para las cuales fue diseñada, en tanto el equipo de diseño, desarrollo y otros involucrados toman nota de la interacción, particularmente de los errores y dificultades con las que se encuentren los usuarios. No es necesario que se trate de una aplicación o sistema completamente terminada, pudiendo tratarse de un prototipo.

Estas pruebas se llevaron a cabo para probar directamente el bloque “Fuzzy Logic Controller”, esto se debe a que este bloque en particular requiere de un archivo de configuración (de extensión .fis) específico para el bloque Planta 3T (correspondiente a la planta de los tres tanques de agua). El usuario que se utilizó para esta prueba fue “jdelpozo”, de modo que usando este usuario se ingresa al sistema LabCon, luego de esto de entre las practicas se debe escoger la opción de “Fuzzy Logic Controller”, esto se debe que al ingresar esta opción se puede realizar la carga al sistema LabCon del correspondiente archivo de configuración como se muestra en la imagen:

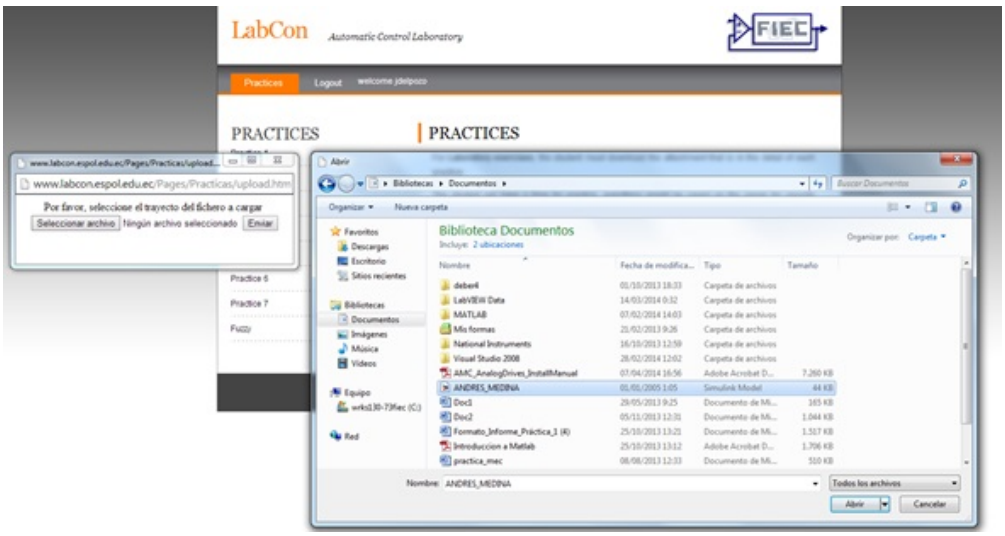


Figura 4.2: Pantalla para cargar el archivo con extensión .fis al sistema LabCon.

Nota: Es importante tomar en cuenta que solo se puede cargar un archivo .fis por usuario dentro del sistema LabCon.

Luego de que se procedió a cargar el archivo de configuración al sistema, se elige la práctica número 3 correspondiente a los tres tanque de agua, y se inició la práctica, para esto ya se contaba con una sesión de la simulación anteriormente usada para probar el bloque Planta 3T, luego de cargar la sesión se agrega a la

simulación el bloque “Fuzzy Logic Controller”, luego de esto al iniciar la práctica, el sistema automáticamente carga el archivo de configuración dentro del bloque “Fuzzy Logic Controller”, sin embargo para poder juzgar el correcto funcionamiento es importante que una persona con el conocimiento necesario en la creación y uso de este tipo de bloques analice los resultados de las señales de salida obtenidas después antes, durante y después de la ejecución de la simulación. Para nuestra prueba en particular contamos con la ayuda del Ing. Franklin Kuonqui y al Ing. Juan del Pozo, a continuación tenemos una imagen de los elementos usados para la simulación:

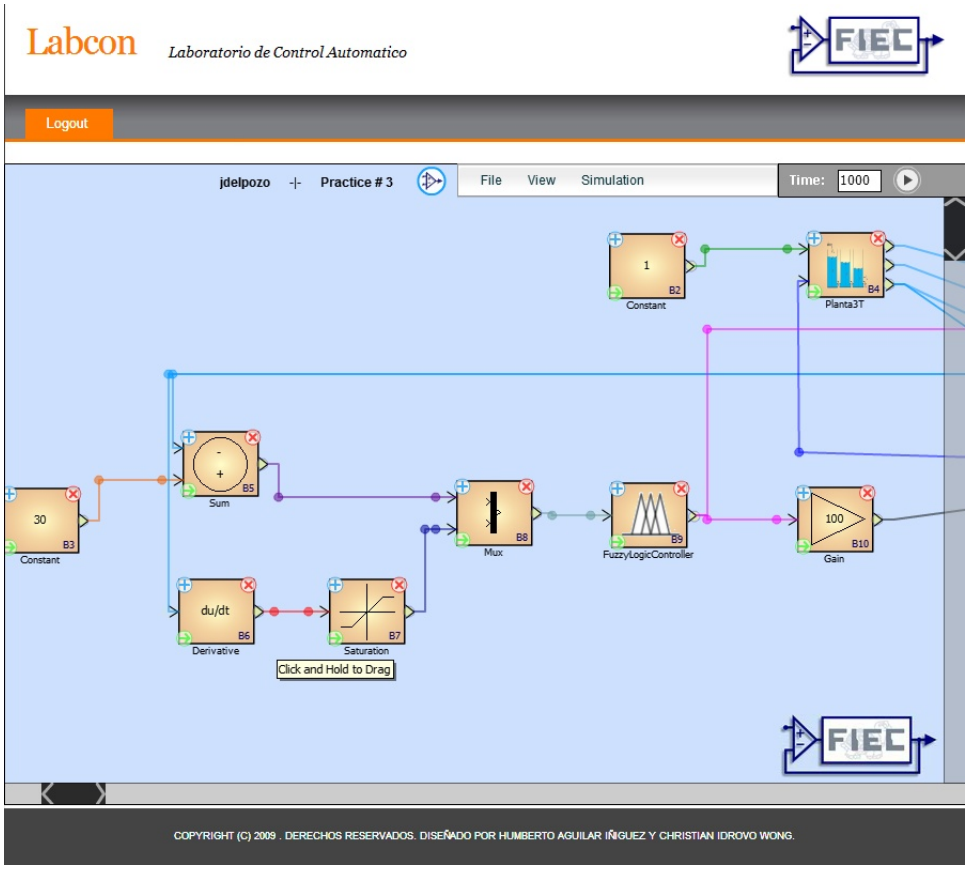


Figura 4.3: Interfase de trabajo del sistema LabCon incluido el bloque "Fuzzy Logic Controller".

Ahora antes de empezar la ejecución de la simulación, lo primero que se debería

confirmar es que el archivo de configuración (archivo .fis) se encuentre cargado dentro de la ruta `C:\MATLABServe rPages\webapps\ROOT\Panel\archivos_config\`, esto es muy importante debido a que el sistema LabCon debe encontrar el archivo con extensión ".fis" para que al momento de iniciar la simulación, de forma automática carga el archivo de configuración respectivo dentro del bloque "fuzzy logic controller" (el sistema busca el archivo dentro de la ruta antes mencionada con el nombre del usuario actual dentro del sistema LabCon. Por ejemplo: si el usuario conectado es jdelpozo el nombre del archivo debe tener el nombre "jdelpozo.fis").

Una vez cargado el archivo controlador con extensión ".fis" dentro del bloque "fuzzy logic controller" procedemos a realizar las pruebas con una simulación en tiempo real en la que se debe incluir la planta de los tres tanques de agua representada en la librería del sistema LabCon por el bloque con el nombre de "Planta3T", a continuación podemos observar la simulación en tiempo real creada usando la interfase del sistema LabCon:

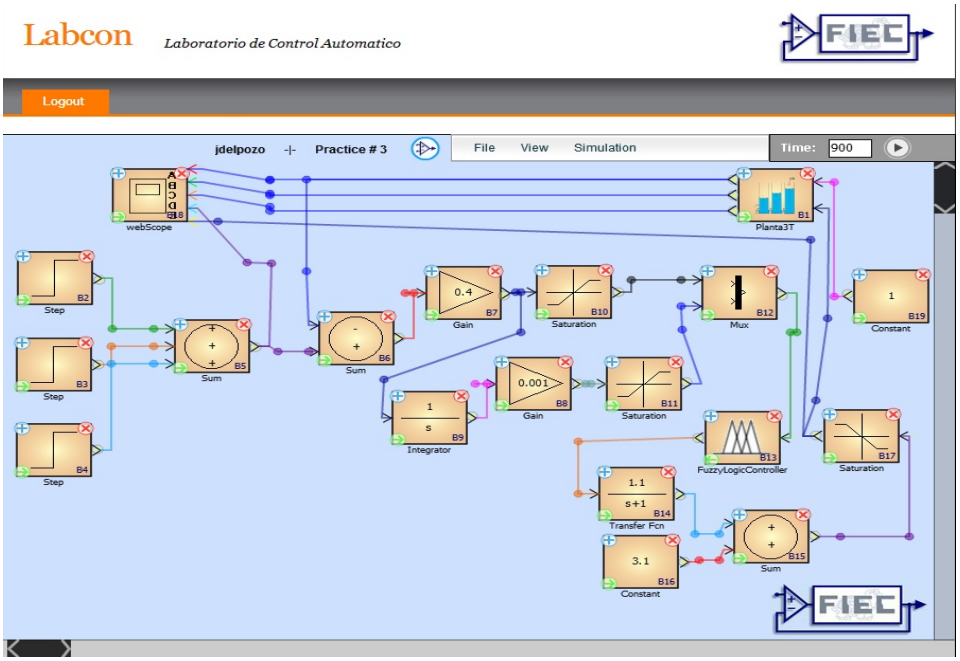


Figura 4.4: Modelo de la simulación usada para las pruebas con el bloque "Fuzzy Logic Controller".

Ahora procederemos a describir de forma general como esta compuesto el archivo de control donde se encuentran las reglas de control. Básicamente se encuentra implementado por dos señales de error como señales de entrada y solo una señal de salida, es importante mencionar que para cada señal existe su correspondiente función de membresia (5) como podemos observar en la siguiente imagen a continuación:

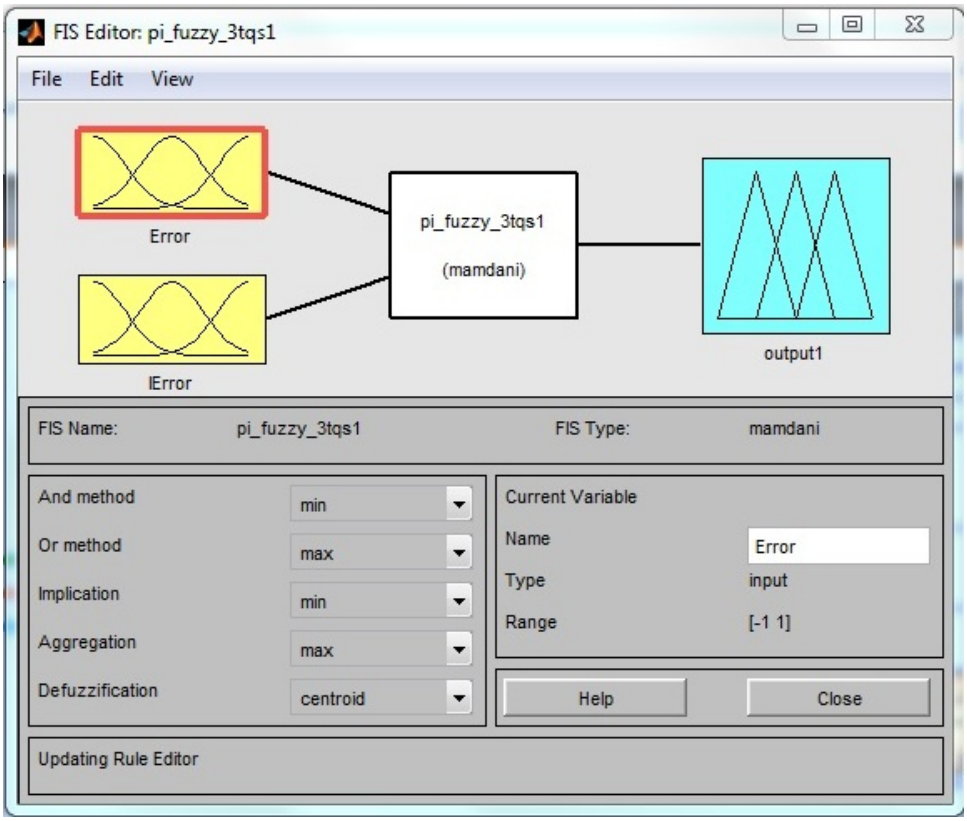


Figura 4.5: Archivo de control visto usando MatLab.

Como se mencionó anteriormente cada señal posee sus correspondientes funciones de membresia (6) como se pueden observar en el siguiente gráfico para las dos señales de entrada (señal del error y la señal integral de error(7)) y la señal de salida detalladas en la imagen a continuación:

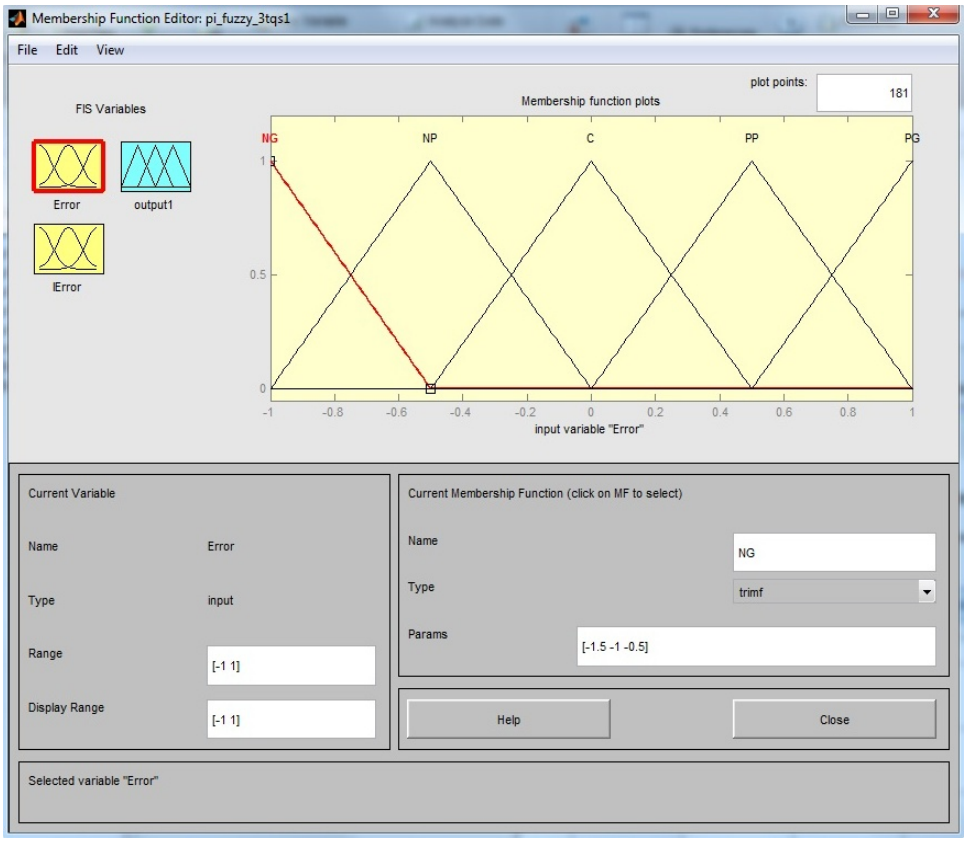


Figura 4.6: Detalle de cada una de las funciones de membresia.

Listado de los nombres de las funciones de membresia:

- NG: Negativo grande.
- NP: Negativo pequeño.
- C: Cero.
- PG: Positivo grande.
- PP: Positivo pequeño.

Dentro de este archivo de control tenemos las siguientes reglas:

```
1. If (Error is NG) and (Error is NG) then (output1 is NG)(1).
2. If (Error is NG) and (Error is NP) then (output1 is NG)(1).
3. If (Error is NG) and (Error is C) then (output1 is NG)(1).
4. If (Error is NG) and (Error is PP) then (output1 is NP)(1).
5. If (Error is NG) and (Error is PG) then (output1 is C)(1).
6. If (Error is NP) and (Error is NG) then (output1 is NG)(1).
7. If (Error is NP) and (Error is NP) then (output1 is NG)(1).
8. If (Error is NP) and (Error is C) then (output1 is NP)(1).
9. If (Error is NP) and (Error is PP) then (output1 is C)(1).
10. If (Error is NP) and (Error is PG) then (output1 is PP)(1).
11. If (Error is C) and (Error is NG) then (output1 is NG)(1).
12. If (Error is C) and (Error is NP) then (output1 is NP)(1).
13. If (Error is C) and (Error is C) then (output1 is C)(1).
14. If (Error is C) and (Error is PP) then (output1 is PP)(1).
15. If (Error is C) and (Error is PG) then (output1 is PG)(1).
16. If (Error is PP) and (Error is NG) then (output1 is NP)(1).
17. If (Error is PP) and (Error is NP) then (output1 is C)(1).
18. If (Error is PP) and (Error is C) then (output1 is PP)(1).
19. If (Error is PP) and (Error is PP) then (output1 is PG)(1).
20. If (Error is PP) and (Error is PG) then (output1 is PG)(1).
21. If (Error is PG) and (Error is NG) then (output1 is C)(1).
22. If (Error is PG) and (Error is NP) then (output1 is PP)(1).
23. If (Error is PG) and (Error is C) then (output1 is PG)(1).
24. If (Error is PG) and (Error is PP) then (output1 is PG)(1).
25. If (Error is PG) and (Error is PG) then (output1 is PG)(1).
```

Una vez que conocemos de forma general la estructura del archivo de control procedemos a ejecutar una simulación que fue creada usando la interfase de LabCon como se pudo observar anteriormente para que pueda realizar los siguientes pasos para controlar la altura en el primer tanque con un tiempo total de 900 segundos para nuestra simulación en tiempo real, detallados a continuación:

```
1. De 0 segundos a 299 segundos la altura debe alcanzar y permanecer en 40
centimetros.
2. De 300 segundos a 599 segundos la altura debe alcanzar y permanecer en 15
centimetros.
3. De 600 segundos a 900 segundos la altura debe alcanzar y permanecer en 30
centimetros.
```

Luego de ejecutar la simulación vemos dentro de la siguiente gráfica el resultado usando el propio bloque “WebScope” (Osciloscopio) del sistema LabCon:

Figura 4.7: Resultados de la simulación desde el osciloscopio.

Como podemos observar dentro del osciloscopio existen cinco señales diferentes que describiremos a continuación:

1. La señal de color rojo corresponde a la altura del primer tanque.
2. La señal de color verde corresponde a la altura del segundo tanque.
3. La señal de color naranja corresponde a la altura del tercer tanque.
4. La señal de color celeste corresponde a la señal de referencia de bloque de control.
5. La señal de color amarilla corresponde a la altura del segundo tanque.

Finalmente como se puede apreciar en la gráfica con respecto a la señal del primer tanque podemos notar que existe un desfase respecto a cada una de las

alturas que debería tener el primer tanque (40 centímetros durante los primeros 300 segundos, luego 15 centimetros y luego 30 centimetros), cuya explicación se refiere a la falta de calibración de la planta, sin embargo debido a que el propósito de este trabajo es verificar la posibilidad del uso del bloque “Fuzzy Logic Controller” para controlar de forma remota la planta de los tres tanques de agua mediante el sistema LabCon, y como podemos observar los datos que se encuentran en la gráfica podemos confirmar el uso del bloque “Fuzzy Logic Controller” de forma correcta dentro del sistema LabCon.

CONCLUSIONES Y RECOMENDACIONES

1. Para poder agregar un nuevo bloque (categoría simulink o categoría planta) al sistema LabCon se documentaron los pasos a seguir. Sin embargo, se recomienda que antes de realizar algún cambio dentro del sistema LabCon se realice una copia de seguridad de todo el sistema para que al momento de presentarse un error se pueda regresar a la última versión funcional del sistema.

2. Si el tiempo de procesamiento para la señal de control es mayor al tiempo de muestreo del sistema LabCon (0.01 segundos) se generaba un error durante la simulación causando que no se pueda apagar la planta de forma automática.

3. Se recomienda que el valor del tiempo de muestreo del sistema LabCon pueda ser modificado como un parámetro adicional dentro de una simulación. Sin embargo se debe tomar en cuenta que cada usuario puede trabajar con tiempos de muestreo distintos de modo que al momento de realizar una modificación de este tiempo este factor debe tomarse en cuenta dentro del análisis que se debe realizar para poder implementar esta mejora.

4. Durante las pruebas al momento de terminar el tiempo de ejecución de forma automática se llama a un bloque Stop (este bloque tiene como tarea enviar las señales pertinentes para el correcto apagado de una planta), sin embargo

esto no ocurre si el bloque “WebScope” (osciloscopio) no se encuentra presente dentro del modelo construido para la simulación. Se recomienda como mejora modificar esta dependencia que actualmente existe con el bloque “WebScope” mientras se trabaja con un bloque de planta.

5. Para poder agregar un bloque especial como es el caso del “Fuzzy Logic Controller”, es necesario que lo haga una persona calificada en el área de computación con experiencia en manejo de Java, debido a que este proceso requiere la modificación de los archivos de configuración (webconfig) de todo el sistema LabCon, y el uso de un servlet (un programa implementado en Java que se ejecuta dentro del servidor del sistema LabCon) para realizar la carga del archivo de configuración al servidor en una ruta preestablecida (revisar el punto 3.1.7 creación del nuevo bloque “Fuzzy Logic Controller”) asi como para cualquier bloque que requiera algún tipo de funcionalidad adicional.

6. Se recomienda crear un programa que permita monitoriar la comunicación entre las plantas y el sistema LabCon y emita una alerta detallada al usuario administrador cuando se pierda la comunicación entre ellos.

7. Es aconsejable agregar un módulo que permita crear un horario de trabajo con las diferentes plantas y así evitar que varios usuarios puedan trabajar con una misma planta a la vez.

8. Para una mayor usabilidad en la interfase del bloque “WebScope” se recomienda poder modificar los nombres de las diferentes señales y aumentar la cantidad de bloques “WebScope” que se pueden incluir dentro de una simulación.

9. Se sugiere asignar a una persona administradora del sistema quien se encargará de realizar un monitoreo constante, organice los horarios de trabajo

para las diferentes plantas.

BIBLIOGRAFÍA

[1] **The Mathworks Inc.**, Simulink Documentation, http://www.mathworks.com/help/simulink/, fecha de consulta julio 2014

[2] **Idrovo Cristhian, Aguilar Humberto**, “Laboratorios Remotos: Comunicación cliente servidor y ejecución remota para las prácticas del laboratorio de control automático de la facultad de Ingeniería en Electricidad y Computación (FIEC)” - Tesis de grado.

[3] **Wikimedia Project**, WikiBook-Latex, http://en.wikibooks.org/wiki/LaTeX , fecha de consulta marzo 2015

[4] **Alegsa Leandro**, Alegsa.com.ar, http://www.alegsa.com.ar/Dic/s ervlet.php , fecha de consulta abril 2015

[5] **Passino M. Kevin y Yurkovich Stephen**, Fuzzy Control, Addison-Wesley , fecha de consulta abril 2015

[6] **Arredondo Vidal Tomás**, Introducción a la lógica difusa, http://profesores.elo.utfsm.cl/ tarredondo/info/soft-comp/Introduccion %20a %20la %20Logica %20 Difusa.pdf, fecha de consulta abril 2015

[7] **Ribeiro Paulo y Klingenberg Bryan**, Getting Started with Fuzzy Logic, https://www.calvin.edu/ pribeiro/othrlnks/Fuzzy/home.htm, fecha de consulta abril 2015