Dan Beninson

Instituto de Tecnología Nuclear

“Desarrollo de controladores software en Sistemas de Control Distribuido y su validación usando modelos simulados de planta”

CARRERA: ESPECIALIZACIÓN EN REACTORES NUCLEARES Y SU CICLO DE COMBUSTIBLE

Alumno: **Pablo Gustavo Juárez del Valle**
Director: **Juan Carlos Dezzutti**
Co-director : **Luis María Pizarro**

CNEA

UNSAM
UNIVERSIDAD NACIONAL DE SAN MARTÍN

Índice:

Bibliografía.

1. ***0758-6000-0ICKI-860-10, "Análisis preliminar del lazo de control de presión del circuito primario a potencia nominal." - Etchepareborda, Andrés; Flury, Celso.***
2. ***Modelo dinamico de un reactor tipo CAREM - San Sebastían, Gustavo.***
3. ***Informe Preliminar de Seguridad CAREM - Sistema de Adquisición, Visualización, Instrumentación y Control - Juárez del Valle, Pablo.***
4. (s.f.). ***www.wikipedia.org*****. Obtenido de DCS - distributed control system.**
5. ***"Simple functions for the fast approximation of light water thermodynamic properties", Gar-land & Hand, Nuclear Engineering and Design 113 (1989).***
6. ***IN- CAREM25 I-5-r0 - "Sistema de Control Distribuido 800xA - ABB".San Sebastían, Gustavo; Juárez del Valle, Pablo; Dupont, Alfredo.***
7. ***MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b.***
8. ***3BSE034679R5021 - Industrial IT 800xA - System - Automated Installation.***

9. ***3BSE046579 - Technical Description Third Party HW Products verified for Industrial IT System 800xA.***
10. ***3BSE035982R5021 ABB - Control and I/O, Communication, Protocol and Design.***
11. ***Technical Reports Series No. 387. Modern Instrumentation and Control for Nuclear Power Plants: A Guidebook.***
10. ***3BSE035982R5021 ABB - Control and I/O, Communication, Protocol and Design.***
11. ***Technical Reports Series No. 387. Modern Instrumentation and Control for Nuclear Power Plants: A Guidebook.***
12. ***3BSE028809R101Rev B - Control AC 800 M/C - Analog Process Control - Object and Design.***
13. ***Comparison of PID Control Algorithms - http://www.expertune.com.***

Glosario.

CASE	*(Computer Aided Software Engineering, Ingeniería de software asistida por computadora). Son herramientas destinadas a aumentar la productividad en el desarrollo de software.*
Coils	*En el protocolo MODBUS es lo que se conoce como salida binaria.*
Gateway	***Un gateway (puerta de enlace) es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación.***
Holding Register	*En el protocolo MODBUS es lo que se conoce como salida analógica (16 bits).*
Hot stand by:	*El término hot stand by se emplea cuando la redundancia de un elemento implica que la redundancia sigue el funcionamiento del primero, para detectar la falla del mismo y así tomar el control y evitar la pérdida de funcionalidad.*
Hot swap	*Este término implica en intercambio de componentes electrónicos durante el funcionamiento, evitando que el dispositivo tenga que apagarse.*
Input Dicrete	*En el protocolo MODBUS es lo que se conoce como entrada binaria.*
Input Register	*En el protocolo MODBUS es lo que se conoce como entrada analógica (16 bits).*
NIC	*Interfaz de comunicación Ethernet para PC .*
Process image	*En el protocolo MODBUS es el área de memoria que contiene a los Inputs Discretes, las Coils, los Inputs Register y los Holding Registers. En el process image se direccionan los entradas y salidas del dispositivo.*
Thread	*Hilo de ejecución de un software. El thread posee sus propios bloques de datos. Comparte el CPU y la memoria con otros threads.*
VLAN	*Redes virtuales. Permite la división virtual de una misma LAN física.*

1. Introducción

1.1 Introducción.

La mayoría de los sistemas de I&C en las NPPs actuales están basadas en la tecnología de las décadas del 50 y 60. Sin embargo a medida que se producían avances tecnológicos en la electrónica, la industria nuclear empezó a implementar nuevas tecnologías. El advenimiento del microprocesador revolucionó los sistemas de instrumentación y control, debido a su flexibilidad en la programación y su capacidad para implementar complejas funciones.

Al principio las centrales nucleares implementaron sistemas digitales de control centralizado. Como ejemplo de esto podemos citar a las centrales de tipo CANDU que utilizan una computadora central, la cual recibe todas las señales de la planta, e implementa los algoritmos de control. Esta computadora esta redundada por otra que ejecuta los mismos algoritmos de control y verifica el estado de la computadora principal.

Posteriormente, los DCS (Distributed Control Ssytem) fueron ganando terreno en centrales nucleares aplicándose a sistemas no relacionados con la seguridad.

Hoy el uso de los DCS se extiende a todo el sistema de control del reactor. Se pueden citar como antecedentes: el reactor de experimentación Opal construido para ANSTO en Australia, por nuestro país y la modernización del sistema de control de la central de potencia Atucha II. Siguiendo esta tendencia, se ha propuesto la utilización de un DCS para el control del reactor CAREM.

El proceso de desarrollo de software que se propone presentar, es un proceso que se integra a una cadena de producción que comienza en la elaboración de requerimientos para el sistema de control, usando modelos de simulación y finaliza en la validación del control implementado.

Los modelos de simulación dan un comportamiento tiempo real de la planta, sus entradas y salidas se encuentran disponibles, usando protocolos de red adecuados, para que el controlador del DCS pueda leerlas y/o escribirlas. Por otra parte se cuenta con los algoritmos de control diseñados a ser implementados en el DCS. Se mostrará el proceso de desarrollo que implementa el control de la planta en el DCS y se validará su comportamiento comparando su respuesta contra el modelo de simulación de planta y controlador. De esta manera se tendrán, para comparar las respuestas de dos sistemas, la del controlador en DCS controlando la planta simulada, contra la simulación de la planta y del controlador en la plataforma de simulación.

Para el presente trabajo se cuenta con un DCS, un modelo analítico de un sistema del reactor CAREM, y su controlador implementados en MATLAB-SIMULINK (2). También se cuenta con un dispositivo de entrada/salida que, usando protocolo de comunicaciones Modbus RTU, vincula las señales generadas por el modelo de planta con el DCS.

Para este trabajo, también se resolverá el problema de interfaces entre el modelo de simulación dinámica de planta y el dispositivo de entrada/salida Modbus RTU completando el código de un software específico.

Una vez resueltas las interfaces, se procederá al desarrollo del software para implementar el control definido en el DCS.

1.2 Antecedentes.

Como antecedente del trabajo se puede mencionar a (1), el cuál realiza un análisis preliminar del lazo de control de presión del circuito primario a potencia nominal tomando la configuración de "Reactor sigue a Turbina", esto es con el lazo: presión del primario controlado mediante la inserción o extracción de las barras de control; para un reactor tipo CAREM. Por otro lado se debe mencionar a **(2), el cuál realiza un modelo en la plataforma de simulación MATLAB-SIMULINK a partir de** (1) **.Este modelo MATLAB se utilizo en este trabajo para implementar el simulador, el cual será controlado con un sistema DCS. En el trabajo original el controlador también fue implementado en MATLAB.**

Como antecedente del trabajo se puede mencionar a (1), el cuál realiza un análisis preliminar del lazo de control de presión del circuito primario a potencia nominal tomando la configuración de "Reactor sigue a Turbina", esto es con el lazo: presión del primario controlado mediante la inserción o extracción de las barras de control; para un reactor tipo CAREM. Por otro lado se debe mencionar a (2), el cuál realiza un modelo en la plataforma de simulación MATLAB-SIMULINK a partir de (1). Este modelo MATLAB se utilizo en este trabajo para implementar el simulador, el cual será controlado con un sistema DCS. En el trabajo original el controlador también fue implementado en MATLAB.

1.3 Objetivo.

El presente trabajo tiene como finalidad desarrollar una arquitectura para la integración de un DCS de última generación y un modelo simulado de reactor nuclear en un determinado punto de operación.

El desarrollo del trabajo se hará siguiendo las pautas de programación requeridas para un sistema real resuelto por un sistema DCS, a modo de ensayo y para sentar las bases sobre las que se podrían resolver, formalmente, futuros desarrollos basado en software de PLC´s para el control de reactores nucleares.

El desarrollo del trabajo se hará siguiendo las pautas de programación requeridas para un sistema real resuelto por un software DCS, a modo de ensayo y para sentar las bases sobre las que se deberán encarar formalmente futuros desarrollos basado en software de PLC´s

Se toma como ejemplo de aplicación la implementación del controlador lineal definido en el modelo de planta. Se compara la respuesta del sistema ante el control del DCS y el control ideal del modelo. Se comparan ambos resultados para validar el sistema de integración.

Se presentarán las características del sistema DCS comercial y su configuración, el desarrollo de software de comunicaciones entre la plataforma de simulación MATLAB-SIMULINK y el dispositivo entrada/salida a través del protocolo Modbus-RTU, y la presentación de la planta a controlar y su modelo matemático.

Por otro lado se analizan los requerimientos del controlador para su implementación en la plataforma DCS, utilizando el lenguaje de programación FBD (function block diagram) y ST (structured text).

La **Figura 1.** Esquema de Planta Virtual y su controlador en MATLAB – SIMULINK. muestra un esquema del modelo de la planta (P) y el controlador implementado en MATLAB-SIMULINK.

La **Figura 2** muestra un esquema donde se reemplaza el controlador (C) por un controlador implementado en un DCS comercial, y las interfaces desarrolladas para lograr la comunicación entre la planta virtual (MATLAB-SIMULINK) y el DCS (Sistema 800xA).

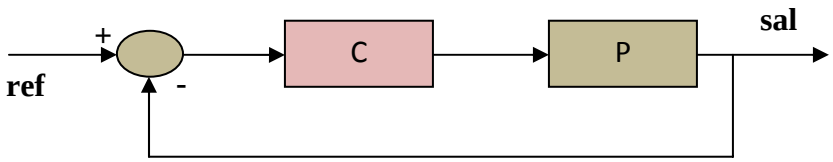

Figura 1. Esquema de Planta Virtual y su controlador en MATLAB – SIMULINK.

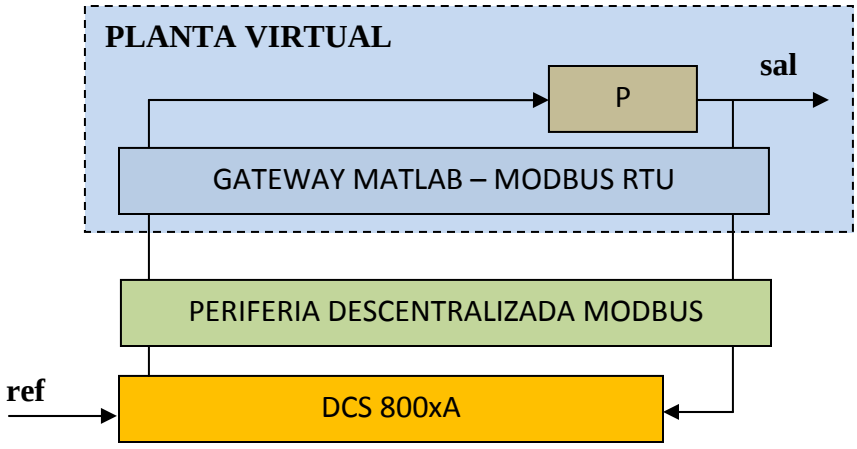

Figura 2. Modelo de sistema a implementar integrando una planta virtual y un DCS.

1.4 Descripción general de un sistema de Control.

Un sistema de control distribuido (DCS – Distributed Control System), es aquel en el cual, los elementos controladores no se encuentran en una ubicación central, sino están distribuidos en distintos puntos del sistema, alejados de la sala de control y cada componente o subsistema está controlado por uno o más controladores. Los controladores están conectados entre sí mediante una red de comunicación digital de datos.

Actualmente los DCS son utilizados en muchas industrias, incluso en la industria nuclear. Podemos citar a Atucha II como ejemplo en la cual la modernización de su sistema de control se hizo utilizando un sistema de control distribuido (SPPA-T2000 Siemens). Otro ejemplo, conocido por nosotros, es el sistema de control del reactor Opal de Ansto en Australia, el cuál fue implementado íntegramente con un sistema de control distribuido (IA-Serie Invensys).

Generalmente los DCS utilizan microprocesadores como controladores y utilizan tanto interconexiones propietarias como protocolos estándares para la comunicación. Los módulos de Entrada / Salida (E/S) pueden conectarse directamente al controlador, o bien, conectarse a través de dispositivos que hacen de gateway entre los controladores y los módulos de E/S. Esta configuración se conoce como *periferia descentralizada*. Los microprocesadores reciben la información del estado de la planta a través de los módulos de entrada, procesan algoritmos de control y/o enclavamientos y envían información hacia los módulos de salida para modificar el comportamiento de la planta. Los módulos de entrada reciben información desde instrumentos de entrada (sensores). Los módulos de salida transmiten instrucciones a instrumentos de salida (actuadores). Los sensores y actuadores sirven de interfaz entre el sistema de control y el proceso de la planta a controlar. Cuando los módulos de E/S se conectan directamente a los microprocesadores lo hacen generalmente en forma analógica (4-20 mA), digital (0-10 V) ; mientras que cuando se utiliza una configuración de periferia descentraliza lo hacen a través de protocolos de comunicación digital, denominados buses de campo. Podemos mencionar como ejemplo de buses de campo ampliamente difundidos a Profibus, Modbus, Foundation Fieldbus, HART. Por otro lado los controladores pueden utilizar buses con protocolos propietarios para comunicarse con estos módulos.

Las redes de comunicación permiten conectar (además de los procesadores de control) las consolas de supervisión donde se encuentran las consolas de operación de la planta.

1.5 Arquitectura de un sistema de control distribuido.

En la arquitectura de un sistema de control distribuido generalmente se puede distinguir los siguientes niveles horizontales:

- Nivel de supervisión
- Nivel de control
- Nivel de campo

- Nivel de adquisición y actuación.

En cada nivel se ejecutan diferentes funciones. Las capas inferiores sirven a las capas superiores con información, aunque la comunicación es siempre bidireccional. Para que el nivel de control ejecute los algoritmos de control, requiere que tanto el nivel de campo, como el nivel de adquisición y actuación trabajen adecuadamente. El nivel de supervisión depende del nivel de control para que le llegue la información de monitoreo. Cada nivel procesa la información que recibe del nivel inferior y la envía al nivel superior, para que este le agregue valor.

Se presenta la arquitectura típica de un sistema de control distribuido en la **Figura 3**.

1.5.1. Nivel de supervisión

Este nivel sirve de interfaz entre el sistema de control y el operador. Las principales funciones en este nivel son: presentación de los valores de las variables de la planta, presentación de alarmas, información sobre anormalidades, comandos de operación para el accionamiento manual de equipos (bombas, válvulas, y otros actuadores).

En este nivel se encuentran las consolas de operación, servidores de almacenamiento de registros históricos de la planta, administradores de eventos y alarmas. En algunos casos las consolas de operación pueden configurarse, permitiendo que el operador acceda a la visualización del estado de la planta desde cualquier consola de operación. Generalmente el operador debe autenticarse ante el sistema para acceder al mismo. Las consolas de operación se conectan a través de una red de alta velocidad, en la mayoría de los casos esta tecnología es Ethernet.

En este nivel además se ubican las funciones de administración general de toda la información que maneja el sistema de control de la planta (análisis de comportamiento, análisis de tendencia, reportes, etc.).

1.5.2. Nivel de control

En este nivel se llevan a cabo las funciones de control automático de la planta, además de la adquisición de señales y su validación.

El componente principal en este nivel es el procesador de control el que, en que en algunos casos, se encuentra redundado con características de *hot-stand-by* y *hot-swap* lo que permite su reemplazo en caso de necesidad sin deterioro de la funcionalidad de control.

Como se mencionó previamente una de las características importantes de los DCS es la distribución de las funciones de control de procesos en forma dedicada en distintas unidades de control. De este modo el nivel de control está integrado por unidades físicamente distribuidas, que se encargan de ejecutar los distintos algoritmos de control.

Cada unidad de control se encuentra conectada a las otras unidades de control que conforman la totalidad del sistema de control distribuido, a través de una red de comunicación denominada red de control, lo cual permite, en caso de requerirse, que un procesador de control acceda a las variables de E/S adquiridas por otra unidad de control, cualquiera sea.

Generalmente estas unidades mantienen un archivo temporal almacenando las variables que gestiona cada una, para que, en caso de una eventualidad pueda recuperarse la información.

1.5.3. Nivel de campo

En este nivel se llevan a cabo las tareas de acondicionamiento de las señales enviadas por los sensores o hacia los actuadores, escalamientos, linearizaciones y validaciones locales.

Las unidades de campo son los elementos principales en este nivel. La información elaborada por las unidades de campo es enviada a los procesadores de control para ejecutar el lazo de control.

Las funciones de las unidades de campo son: adquirir las variables requeridas por el sistema para poder ejecutar las rutinas o lazos de control y conformar las variables de modo que el operador de la planta pueda conocer el estado de la misma. Para ello realiza las conversiones de las señales adquiridas a unidades de ingeniería. También procesa las excitaciones de los actuadores del sistema.

Para ser más específico, podríamos definir a una unidad de campo como el dispositivo al cuál se conectan los módulos de entrada/salida, logrando desacoplar estos módulos de los controladores, permitiendo alejarlos del mismo. Esto se conoce como periferia descentralizada.

Una unidad de campo, puede manejar varias señales como en el caso de la configuración de periferia descentralizada; o bien, unas pocas señales como en el caso de los transmisores.

La comunicación de las unidades de campo con las unidades de control, sean éstas redundadas o no, se realiza a través de un bus de campo (Profbus, Foundation Fieldbus, Modbus, etc.). Esta comunicación es bidireccional: hacia y desde la unidad de control. Se envía hacia el procesador de control la información del valor de las distintas variables conjuntamente con información de estado y desde los procesadores de control se reciben las órdenes de accionamiento de las actuaciones y valores de calibración de los sensores, generalmente a la variable sensada se le adjunta una marca de tiempo, la cual se denomina *time stamping.* Esta marca de tiempo depende del fabricane; en algunos casos es el controlador quien coloca el *time stamping*.

En la actualidad, existen diversas configuraciones, y pueden encontrarse casos en los que las unidades de campo recolectan la información de los sensores y las envían directamente hacia las consolas de operación.

1.5.4. Nivel de adquisición y actuación

En este nivel se realiza la medición de las magnitudes físicas del proceso mediante sensores y se excitan los actuadores existentes.

Estos dispositivos pueden ser del tipo convencional y conectarse a las unidades de campo, las cuales convierten las mediciones a unidades de ingeniería. En algunos casos se conectan directamente a las unidades de control, a sus respectivos módulos de entrada/salida; evitando así las unidades de campo. También estos dispositivos pueden ser dispositivos inteligentes y conectarse a un bus de campo, e incluso implementar un simple control.

Se presenta la arquitectura típica de un sistema de control distribuido en la **Figura 3.** Arquitectura típica de un sistema de control distribuido.

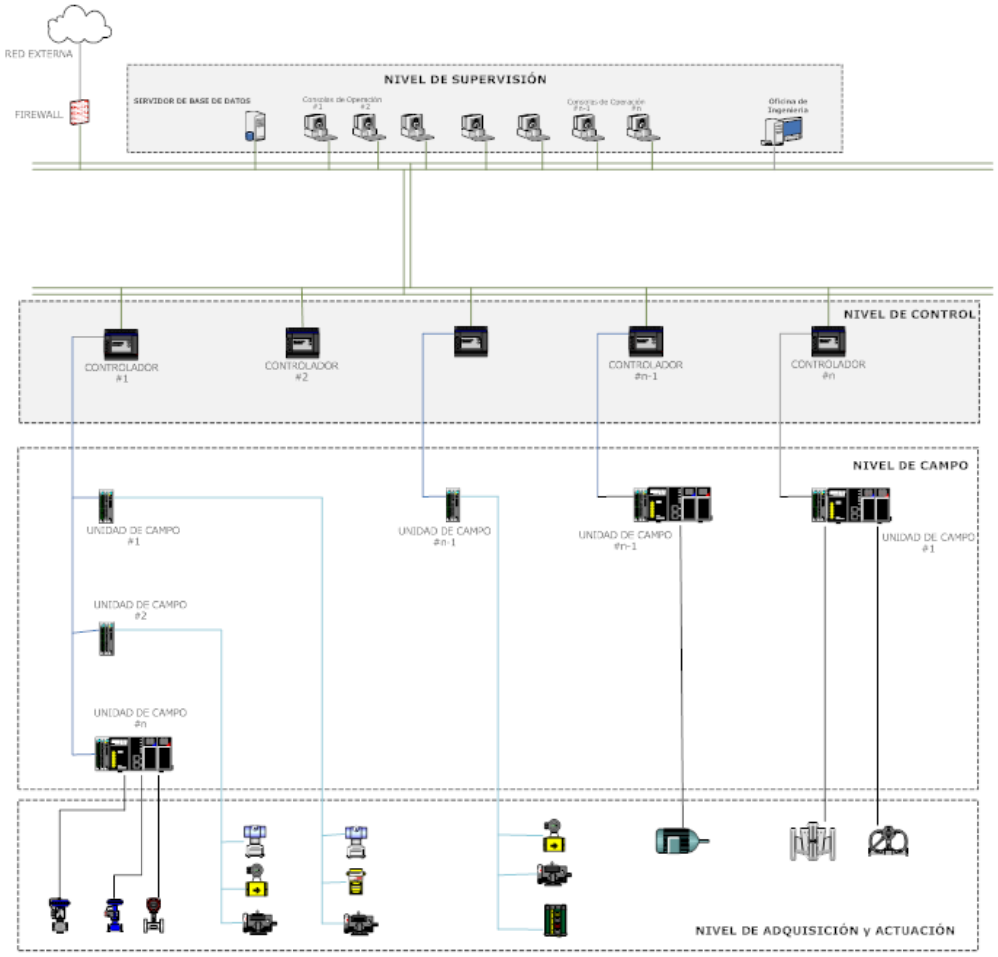

Figura 3. Arquitectura típica de un sistema de control distribuido.

2. Modelo dinámico de planta utilizado.

2.1 Introducción.

El modelo del circuito primario considerado es el presentado en las referencias (1) y (2). Este modelo divide el recipiente de presión en dos zonas: el domo y el circuito. A su vez el circuito se divide en: generador de vapor, downcommer, plenum inferior, núcleo y la chimenea. La **Figura 4** muestra el modelo compartimentado de referencia.

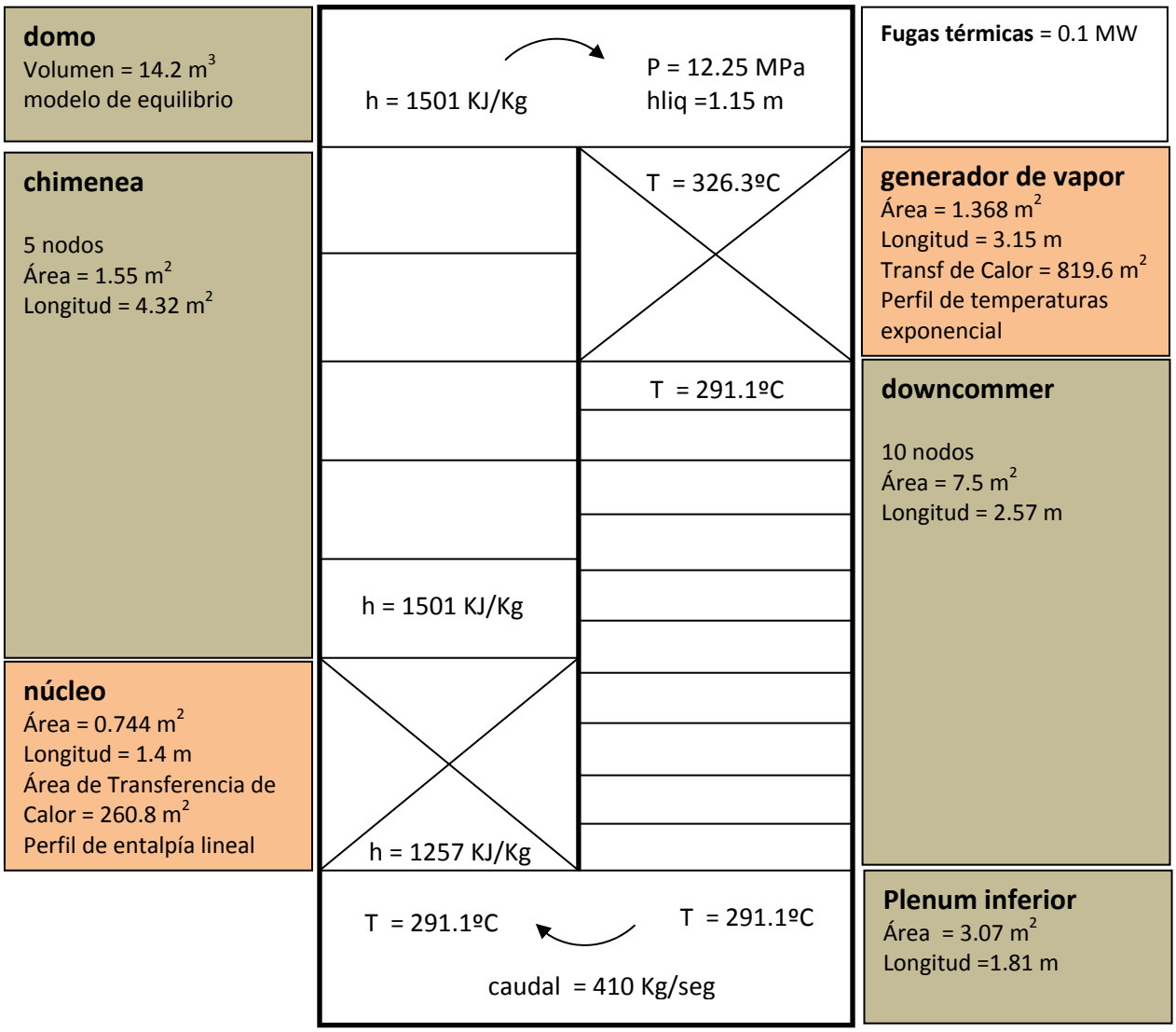

Figura 4. Esquema del modelo utilizado del primario.

El punto trabajo del reactor está descripto por los parámetros:

Pot0 = 100MW. Potencia neutrónica.

Pe = 0.1MW. Fugas térmicas al exterior.

w0 = 410 Kg/seg. Caudal másico del circuito.

P0 = 12.25 Mpa. Presión del RPV.

Tf = 700ºC. Temperatura del combustible.

Ts = 260.14ºC. Temperatura de alimentación al generador de vapor.

Los parámetros definidos fueron obtenidos del estacionario del reactor trabajando a 100% de potencia de (5).

En (2) se presenta el modelo del reactor en MATLAB-SIMULINK, del cual se hace a continuación una breve descripción.

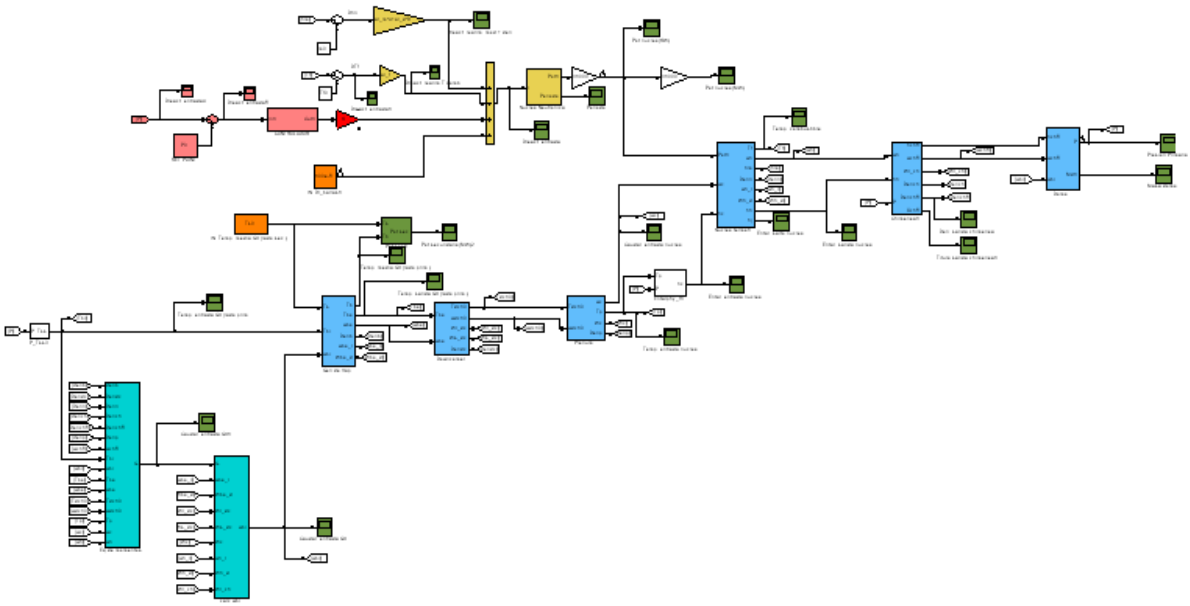

Figura 5. Modelo de planta implementado en MATLAB-SIMULINK

La **Figura 5** muestra el modelo implementado en MATLAB-SIMULINK y en ella pueden observarse diferentes bloques según su color.

2.2 Termohidráulica

El análisis termohidráulico que se considera se basa en las siguientes simplificaciones (1):

- “El problema se resuelve en forma unidimensional. La variable espacial se toma en sentido de circulación del fluido.”
- “Para resolver la zona bifásica de líquido y vapor (núcleo y chimenea) se usa el modelo homogéneo, o sea que se trata a la mezcla como un pseudo-fluido que obedece las ecuaciones de simple fase (no se consideran diferencias de velocidades entre vapor y líquido).”
- “No existe arrastre de burbujas hacia el generador de vapor y se considera que el líquido que ingresa al generador de vapor se encuentra en saturación.”
- “En todo el circuito se considera que la presión es la misma e igual a la del domo, y no se tienen en cuenta diferencias de presión por columna de agua.”
- “Solo se considera intercambio de calor en la zona del núcleo y del generador de vapor (que son variables), y una pérdida de calor constante en el domo. El intercambio de calor en el núcleo es gobernada por la diferencia de temperaturas medias del refrigerante y del secundario (condición de contorno), considerando un perfil de temperaturas exponencial del lado primario y constante del lado secundario.”
- “No se consideran los caudales de by-pass del generador de vapor, ni del núcleo.”

El comportamiento termohidráulico del reactor se describepor medio de los bloques de color celeste. Los mismos se describen de izquierda a derecha.

Básicamente el modelo termohidráulico describe los balances de masa y energía. El balance de energíase expresa mediante una ecuación diferencial que determina en algunos casos la entalpía y en otros la temperatura, del balance de masa se obtiene el caudal de refrigeración.

Por otro lado se considera la conservación de momento en forma integral para cerrar el cálculo del circuito. En el **Anexo I** se citan las ecuaciones diferenciales para cada nodo, obtenidas de (2).

2.3 Cinética Neutrónica

Se utilizó un modelo de cinética puntual a un solo grupo de neutrones retardados sin linealizar. Para definir la realimentación de reactividad se consideró una variación lineal por temperatura del moderador (en este caso también refrigerante), por temperatura del combustible y por densidad del moderador (2).

Las ecuaciones que definen este modelo se pueden encontrar en el **Anexo I**.

La cinética del reactor se describe por medio de bloques de color amarillo.

2.4 Controlador

Se definió un lazo de control de la presión del RPV considerando la inserción o extracción de las barras. La función transferencia del controlador definida en (1) es:

$$\boldsymbol{Gc_2(s)} = \frac{20\ s+1}{s}\ \ 1.05e-4\ \ \frac{partes\_enteras}{MPa}$$

Las siguientes figuras muestran evolución de la potencia neutrónica, la presión y la potencia del secundario **para el transitorio producido por un escalón en la reactividad de +100pcm**; el cual es compensado por el sistema de control mediante el movimiento de barras. Esto es simulado **para los primeros 300 segundos(2).**

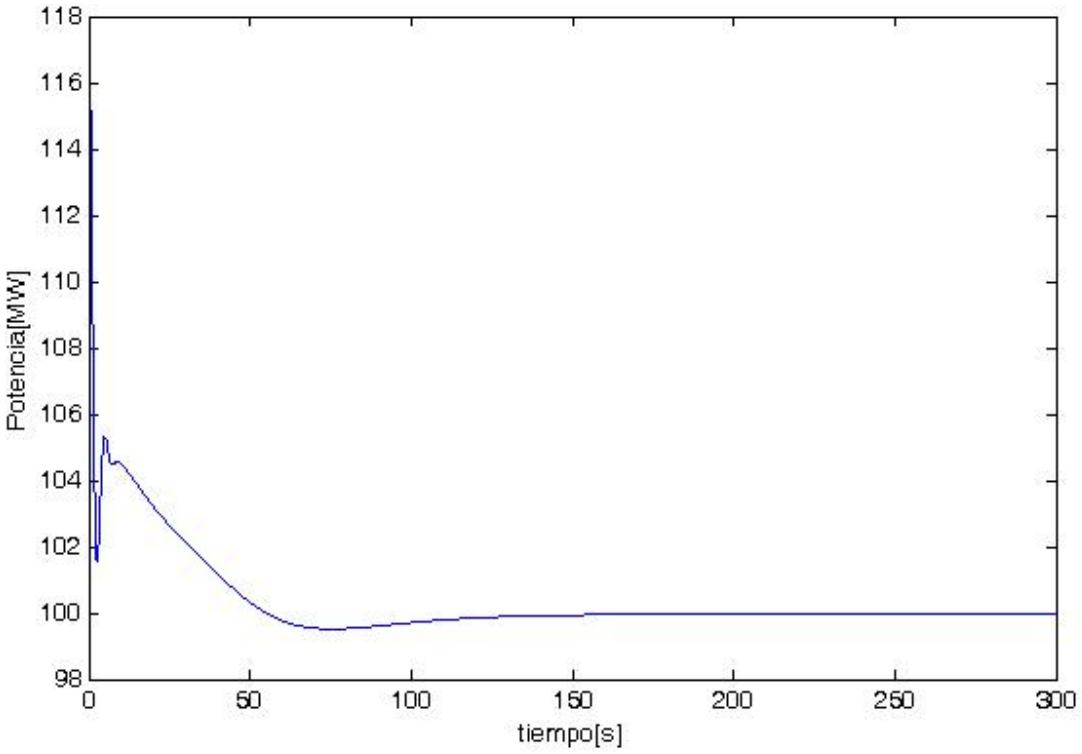

Figura 6. Evolución de la Potencia Neutrónica a lazo cerrado con un escalón de r=100pcm

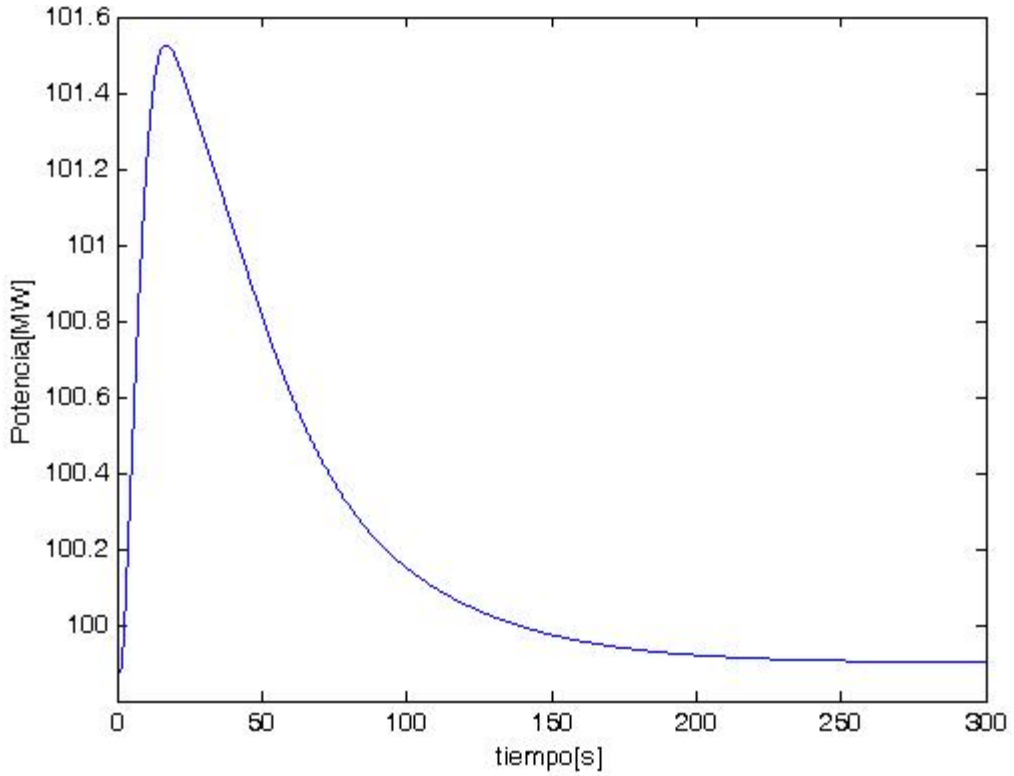

Figura 7. Evolución de la potencia extraída por el GV a lazo cerrado con un escalón de r=100pcm.

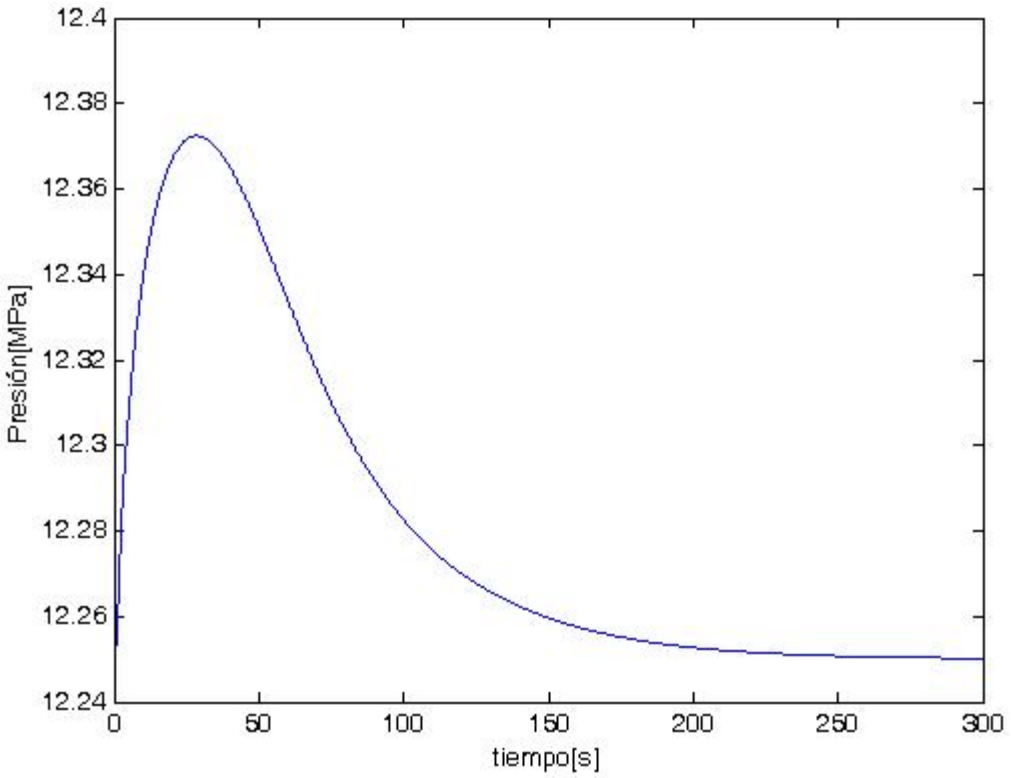

Figura 8. Evolución de la presión en el RPV a lazo cerrado con un escalón de r=100pcm.

El modelo que se utiliza en este trabajo es el modelo en tiempo real de (2), el cuál fue discretizado, reemplazando los integradores continuos por integradores discretos. El modelo en tiempo real toma un tiempo de muestreo de 10^{-3} segundos. A este modelo se le agrega un controlador discreto para obtener las gráficas de tendencia mostradas anteriormente.

Por último debe destacarse que al modelo presentado se le agregan las interfaces UDP para comunicarse con los Gateway Matlab-Modbus RTU.

Se incorporan dos interfaces UDP: la primera envía información al puerto 25000 (61A8); la segunda envía información al puerto 30000 (7530) y recibe información a través del puerto 35000 (88B8). La **Figura 9** y la **Figura 10** muestran la implementación de lo mencionado. Para lograr implementar esto se utilizan dos NIC destinando una a la primera interfaz y otra a la segunda interfaz. La **Figura 11** muestra la incorporación de las interfaces UDP al modelo completo de la planta.

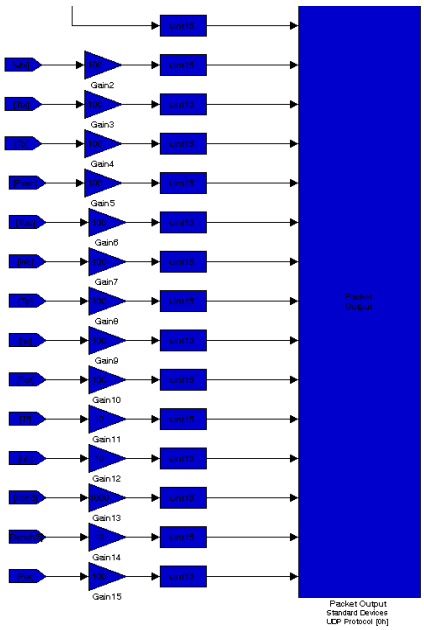

Figura 9. Interfaz UDP, la cual envía datos al puerto 25000.

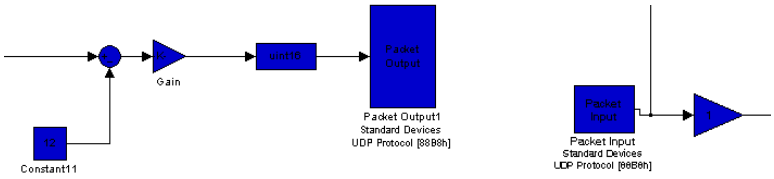

Figura 10. Interfaz UDP, la cual envía datos al puerto 35000 y recibe en el puerto 30000.

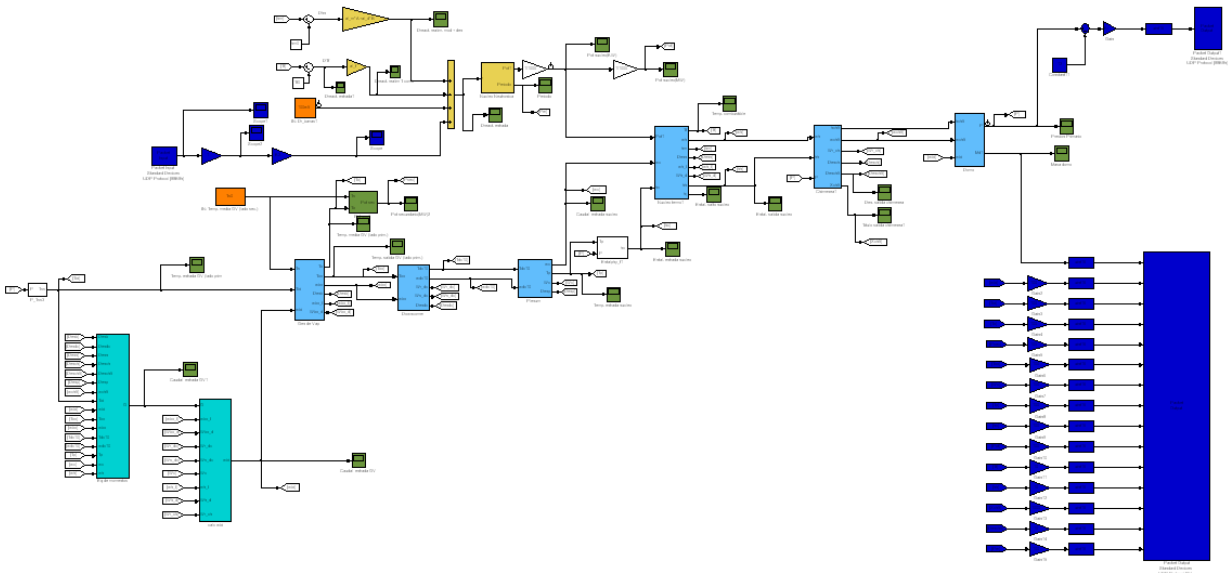

Figura 11. Modelo de planta a la cual se incorporó las Interfaces UDP.

3. Sistema de Control.

3.1 Descripción general del DCS utilizado. Sistema 800xA – ABB (6).

El sistema Industrial IT 800xA de ABB es un sistema para la automatización de procesos industriales. Está basado en el concepto DCS (Distributed Control System), e integra la ingeniería, la operación y el mantenimiento en un mismo entorno de trabajo. El sistema 800xA fue concebido bajo la tecnología Aspect Object lo cual en este caso permite integrar por ejemplo documentos de Word para ver documentación técnica, archivos de Excel para el análisis de tendencias, o bien archivos de SAP o de Máximo para el manejo de la información de Mantenimiento, entre otras cosas.

El sistema soporta una gran cantidad de protocolos industriales, entre ellos Profibus, Modbus y Foundation; pero por otro lado incorpora protocolos propietarios entre los que podríamos citar a RNRP (Redundant Network Routing Protocol), el cual es utilizado para redundar la red de control.

Este sistema tiene una gran funcionalidad y abarca casi todos los aspectos de una red de control, sin embargo la arquitectura y la administración suelen ser muy complejas.

Está, obviamente, integrado con todos los DCS fabricados por ABB, aunque también permite integrar PLCs de otros fabricantes.

3.1.1. Redundancia

El sistema 800xA tiene una arquitectura tolerante a fallas, donde los componentes pueden ser redundados (la redundancia se implementa mediante un switch automático):

- Controlador (CPU redundantes, cables de conexión redundante).
- Módulos de Entrada / Salida.
- Módulos de Comunicaciones (a nivel de Module Bus, a nivel de CEX).
- Servidores de Conectividad (1oo2).
- Servidores de Aspectos (2oo3).
- Redes de Comunicación (Redes de Supervisión, Redes de Control, Redes de Campo).

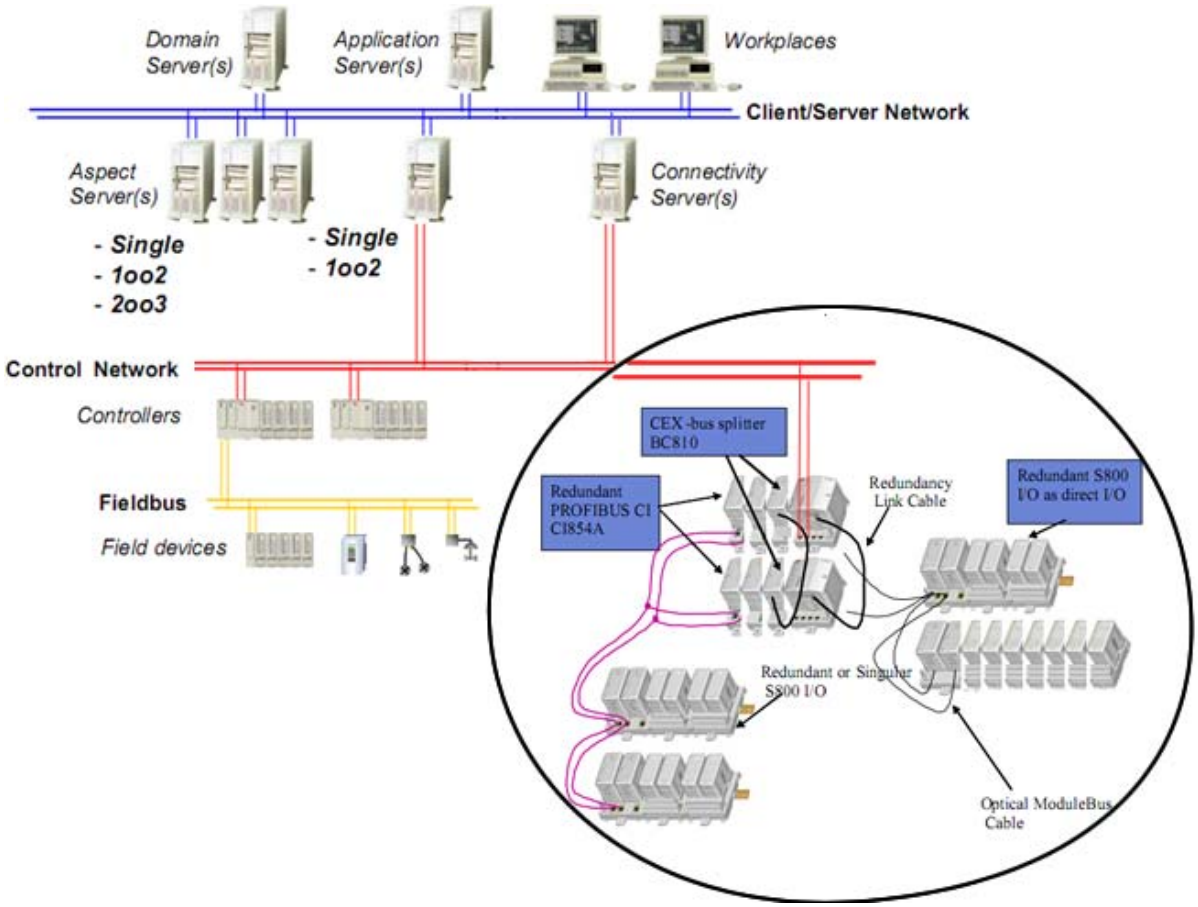

Figura 12. Arquitectura típica en un sistema de control 800xA.

3.2 Arquitectura del sistema de prueba.

La arquitectura utilizada para implementar el sistema de prueba consta de:

- Servidores de Aspectos (Aspect Servers) en configuración 1oo2.
- Servidores de Conectividad (Connectivity Servers) en configuración 1oo2.
- Estación de Ingeniería y Operación.
- Controladores Redundantes, módulos de comunicación y módulos de entrada/salida.
- Estación de Simulación y Gateways Matlab-Modbus RTU.
- Módulo de periferia descentralizada (Modbus RTU).
- Switch Ethernet 3Com Baseline 2226.

La **Figura 14** muestra la interconexión de los componentes del sistema.

A continuación se describe la funcionalidad de cada uno de ellos de acuerdo a (6).

3.2.1. Servidor de Aspectos (Aspect Server).

El servidor de aspectos permite organizar, administrar y tener acceso a la información de los distintos aspectos de una gran cantidad de objetos de la planta o entidades de proceso de manera unificada proveyendo una manera consistente de acceder a toda la información sin depender de la aplicación que la requiere. Este servidor ejecuta el Diccionario de Aspectos junto con otros servicios relacionados para el manejo de los objetos, administración de los nombres y seguridad.

El servidor de aspectos es el corazón del sistema 800xA ya que aquí reside toda la configuración del sistema. El servidor debe ser accesible por todos los nodos y en todo momento; para asegurar esto se puede configurar en forma redundante "1 de 2" o "2 de 3". La configuración 1 de 2 requiere que al menos uno de ellos esté en línea para poder leer o escribir sobre el sistema. La configuración 2 de 3 en cambio requiere que al menos 2 servidores estén activos para escribir pero sólo uno para leer.

La configuración que se aplica, en este trabajo, para el servidor de aspectos es "1 de 2".

3.2.2. Servidores de Conectividad (Connectivity Server).

El servidor de conectividad (SC) provee acceso a los controladores y otras fuentes de datos a través de las redes del sistema. Es el encargado de interconectar la red de supervisión con la red de control, actuando como un gateway entre ambas.

Su existencia es clave en la arquitectura del 800xA, para permitir la visualización de los datos de la planta en la consola de operación.

El SC utiliza el protocolo MMS para comunicarse con los controladores. El SC en este nivel toma el rol de servidor. Del lado de la red de supervisión se utiliza un protocolo tipo cliente-servidor que es el OPC. Aquí también actúa como servidor para los clientes OPC que pueden ser el servidor de aspectos o las distintas estaciones de trabajo del sistema.

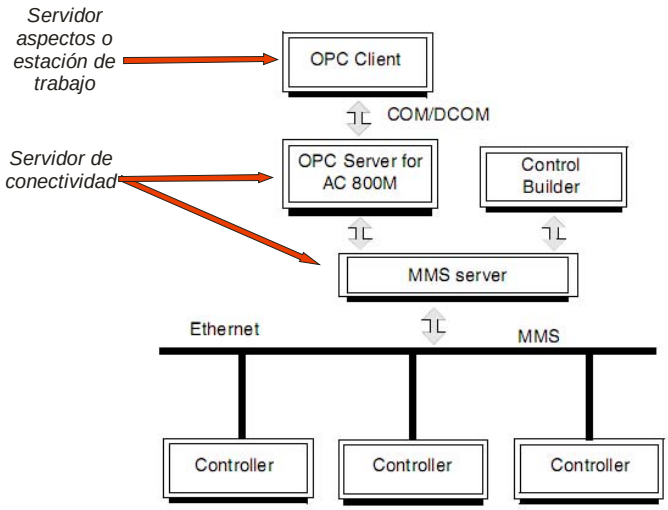

Figura 13. Estructura de comunicación del Servidor de Conectividad.

El SC ejecuta, 2 servicios uno que implementa el servidor MMS y otro para el servidor OPC. En **Figura 13** se muestra un esquema de esta configuración.

Debido a que el servidor de conectividad es un componente crítico del sistema de control ya que de este depende el acceso a los datos de planta se configura la redundancia 1 de 2 pasiva.

3.2.3. Estación de Ingeniería y Operación (Engineering Workplaces, Operator Workplaces)

La estación de ingeniería es la herramienta más importante para la configuración del sistema 800xA. Es un entorno multiusuario apuntando a distribuir la ingeniería de la planta, sin embargo en este proyecto se combina la estación de ingeniería y la de operación en una sola consola. Las tareas que se desarrollan desde la estación de ingeniería son:

- Definir la arquitectura de la planta.
- Definir la arquitectura del sistema de control.
- Programar las lógicas de control.
- Desarrollar la interfaz gráfica de usuario.
- Definir y administrar alarmas.
- Realizar tareas de mantenimiento y generar reportes.

Los programas que se utilizaron para la configuración y desarrollo del sistema son el Process Portal y el Control Builder. El primero se utilizó fundamentalmente para la configuración del sistema, el diseño de la interfaz gráfica. El segundo se utilizó para definir la arquitectura del sistema de control y desarrollar las lógicas.

La estación de operación permite básicamente supervisar y controlar de forma eficiente la planta. La estación de operación del 800xA implementa la interfaz con el operador donde se pueden integrar:

- Displays gráficos
- Paneles de mando de los objetos del proceso
- Manejo y presentación de alarmas y eventos
- Gráficos de tendencias
- Generación de reportes
- Estado general del sistema
- Estado de la topología del sistema
- Visor de procesos secuenciales (SFC)

3.2.4. Controladores.

El sistema, que se implementó para este trabajo, utiliza dos controladores redundantes.

El ***controlador 1*** está constituido por los siguientes módulos:

PM861A: Contiene el procesador, memoria RAM, controladores para las interfaces de comunicación on-board y reloj de tiempo real. El controlador cuenta con dos CPU que trabajan redundadas en modo hot stand-by (unidad primaria y unidad backup). Cada una trabaja insertada en una base (TP830) en la que se encuentran dos conexiones Ethernet y dos puertos RS232, uno de los cuales se utiliza para realizar la configuración del controlador.

Las unidades redundadas están conectadas mediante un cable RCU por el que ambas CPUs intercambian datos e información de estado y control.

BC810: Este módulo permite el recambio de una CPU sin causar interrupciones en el tráfico del bus que comunica el CPU con los módulos de comunicación (bus CEX). El módulo tiene una conexión a la fuente de alimentación.

CI854A: Es la interfaz de comunicación entre el bus CEX y la red Profibus DP de la cual es el maestro. Un solo dispositivo permite conectar dos redes Profibus redundadas. En la configuración planteada existen dos dispositivos para lograr la redundancia del maestro Profibus. Recibe alimentación del mismo bus CEX. El módulo permite hot swap.

CI840A: Permite la conexión del bus ModuleBus eléctrico de un cluster a la red Profibus DP. La base TU847 soporta dos módulos CI840A cada uno de los cuales se conecta a una de las redes Profibus redundadas y posee una entrada de alimentación que provee la tensión de alimentación a todos los dispositivos de E/S conectados al ModuleBus eléctrico.

AI810: Módulo de 8 entradas analógicas de 0 a 20 mA, 4 a 20 mA, 0 a 10 V ó 2 a 10 V. Cada uno de los canales puede ser una entrada de corriente o voltaje. La resolución del dispositivo es de 12 bits.

AO810V2: Módulo de 8 salidas analógicas de 0(4) a 20 mA unipolares. La resolución del dispositivo es de 14 bits.

El ***controlador 2*** destinado al monitoreo está constituido por los siguientes módulos:

PM861A: Este controlador también cuenta con dos CPU que trabajan redundados en modo hot stand-by. La base TP830 sobre la cual se inserta el CPU, como se explico anteriormente posee dos conexiones Ethernet y dos puertos RS232; el primer puerto RS232 soporta una interfaz Modbus RTU/ASCII, solo en la configuración *Maestro*. Esta interfaz Modbus se utiliza para comunicarse con el Gateway ***Matlab - Modbus RTU Slave***, la cual se comunica con el simulador.

3.2.5. Estación de Simulación y Gateways Matlab-Modbus RTU

En esta estación se ejecuta el modelo de MATLAB-SIMULINK desarrollado en (2), y modificado para lograr la interfaz buscada con los Gateways Matlab-Modbus RTU.

Por otro lado en esta estación se ejecutan dos gateways:

Gateway Matlab-Modbus RTU Slave: implementa la interfaz entre el modelo de simulación y el controlador 2. Este Gateway escucha en dos sentidos diferentes. Un proceso escucha desde el

modelo simulado (interfaz Ethernet UDP) para almacenar los datos, y otro proceso escucha peticiones Modbus del maestro implementado en el controlador 2 para enviarle los datos.

Gateway Matlab-Modbus RTU Master: implementa la interfaz entre el modelo de simulación y la periferia descentralizada Modbus RTU. En este caso el Gateway es bidireccional, lee y escribe datos en ambos sentidos.

Este Gateway ejecuta dos procesos:

- El primer proceso recibe los datos enviados por el modelo simulado a través de una segunda interfaz Ethernet UDP en un puerto determinado y almacena los mismos en un *process image* (7).
- Otro proceso realiza las siguientes operaciones dentro de un ciclo de scan:

  1-Lee la entrada de la periferia descentralizada modbus rtu*.* Cabe aclarar que a cada entrada en la periferia descentralizada corresponde una salida del módulo *AO810V2* del controlador 1, y lo escribe en el process image.

  2-Escribe el valor leído de la periferia descentralizada en un puerto UDP para que lo lea el modelo simulado ejecutándose en la plataforma MATLAB-SIMULINK.

  3-Escribe la salida a la periferia descentralizada Modbus RTU con el valor leído por el primer proceso*.* La salida en la periferia descentralizada es conectada al módulo *AI810* del controlador 1.

3.2.6. Periferia descentralizada Slave Modbus RTU.

La periferia descentralizada slave Modbus RTU es un dispositivo desarrollado íntegramente en el sector de Ingeniería de la Unidad Actividades de Reactores y Centrales Nucleares de la *GAEN.* Este dispositivo cuenta con las características definidas a continuación.

Tiene implementado un Esclavo Modbus RTU, con las siguientes funciones:

- **02.** Read Discrete Inputs.
- **04.** Read Inputs Registers.
- **15.** Write Multiple Coils.
- **16.** Write Multiple Registers.

Sus módulos de entrada / salida son:

- **16 Discrete Inputs.** 0V False, 5V True.
- **16 Coils.** 0V False, 5V True.
- **8 Input Registers.** 0-5V. 10bits.
- **2 Holding Registers.** 0-5V. 10bits.

Su configuración es:

- **Slave Address** = 02.

- **Baud Rate** = 38400Kbps.
- **Data Bits** = 8 bits.
- **Parity** = none.
- **Stop Bits** = 2 bits.
- **Flow Control** = none,

Se alimenta con 12V D.C. Posee una interfaz RS-485. Debido a esto se utiliza un conversor ADAM RS-232/RS-485 para conectarla con el nodo de simulación.

3.2.7. Switch 3Com Baseline 2226.

La interconexión del sistema requiere de cuatro subredes Ethernet para conectar los controladores, los servidores de conectividad, los servidores de aspectos y la estación de ingeniería/operación.

- **Red de control A.** esta red conecta el primer puerto de los dos CPU de cada controlador , así como el primer puerto de la red de control de los servidores de conectividad. Por otro lado a esta red también se conecta un puerto a la estación de ingeniería. Se requieren 7 puertos
- **Red de control B.** esta red conecta el segundo puerto los dos CPU de de cada controlador con el segundo puerto de la red de control los servidores de conectividad. Se requieren 6 puertos.
- **Red de supervisión A.** esta red conecta el primer puerto de la red de supervisión de los servidores de conectividad, con el primer puerto de los servidores de aspectos y con el primer puerto de la estación de ingeniería. Se requieren 5 puertos
- **Red de supervisión B.** esta red conecta el segundo puerto de la red de supervisión de los servidores de conectividad, con el segundo puerto de los servidores de aspectos y con el segundo puerto de la estación de ingeniería. Se requieren 5 puertos.

Los puertos necesarios que se requieren para interconectar este sistema son 23 puertos.

Cada CPU de control posee 2 puertos. Se tienen 4 CPU, dos por controlador. Lo que implica un total de 8 puertos para los controladores 4 para la red de control A y 4 para la red de control B.

Los servidores de conectividad poseen 4 puertos, dos para la red de control (A y B) y dos para la red de supervisión (A y B). Los servidores de conectividad suman 8 puertos más. 2 para la red de control A, 2 para la red de control B, 2 para la red de supervisión A y 2 para la red de supervisión B.

Los servidores de aspectos poseen dos puertos. Considerando que se tiene dos servidores de aspectos, suman un total de 4 puertos; 2 para la red de supervisión A y 2 para la red de supervisión B.

La estación de ingeniería y operación posee tres puertos. 1 para la red de supervisión A, 1 para la red de supervisión B y 1 para la red de control A.

	Red de Control A	Red de Control B	Red de Supervisión A	Red de Supervisión B
Controlador 1	**2**	**2**	**-**	**-**
Controlador 2	**2**	**2**	**-**	**-**
Servidor de Aspectos Primario	**-**	**-**	**1**	**1**
Servidor de Aspectos Secundario	**-**	**-**	**1**	**1**
Servidor de Conectividad Primario	**1**	**1**	**1**	**1**
Servidor de Conectividad Secundario	**1**	**1**	**1**	**1**
Estacion de Ingeniería	**1**	**-**	**1**	**1**
	7	**6**	**5**	**5**
Total de puertos:	**23**			

Tabla 1. Puertos Ethernet en el sistema de pruebas.

El switch que se dispone es un switch administrable el cual posee 24 puertos Ethernet 10/100 Mb y 2 puertos 1 Gb. Un switch administrable es un dispositivo que permite una configuración externa por parte de un usuario

Para evitar tráfico innecesario se divide el switch en 4 VLAN que representan las 4 subredes mencionadas. Se dedica el **Anexo V** a la configuración del switch.

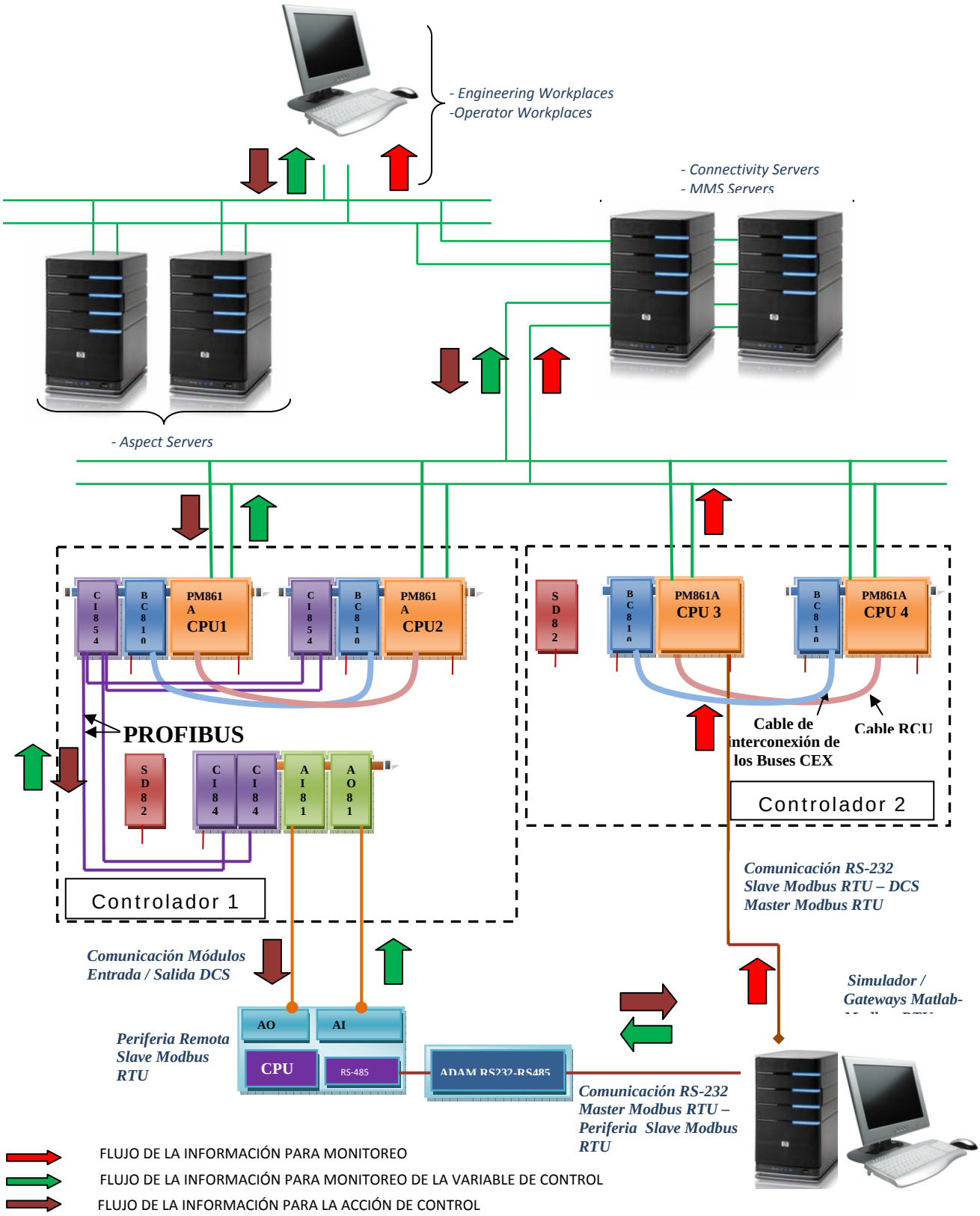

Figura 14. Arquitectura empleada del sistema 800xA para implementar el sistema de pruebas.

4. Arquitectura Software de los Gateway

4.1 Introducción.

En este punto se muestra la arquitectura de software utilizada en el diseño de los Gateway Matlab – Modbus RTU. Se diseñaron dos Gateways:

- Matlab – Modbus RTU Master.
- Matlab – Modbus RTU Slave.

El primer Gateway abre dos puertos UDP a través de la misma NIC para comunicarse con la plataforma MATLAB-SIMULINK. En un puerto monitorea las variables para realizar el control, y en el otro puerto ejecuta las acciones de control sobre el modelo dinámico de planta, el cual se ejecuta sobre la plataforma MATLAB-SIMULINK. Además posee un Maestro Modbus el cual se comunica con la periferia descentralizada Modbus leyendo y escribiendo datos.

El segundo Gateway abre un puerto UDP para comunicarse con la plataforma MATLAB-SIMULINK y monitorear las variables del sistema. Con esas variables crea un image process (7) creando un esclavo modbus, con el cual el DCS se comunica para ver el estado de las variables generadas en el simulador MATLAB-SIMULINK.

4.2 Gateway Matlab – Modbus RTU Master

4.2.1. Funciones del Gateway

- Leer desde MATLAB la/s variable/s a controlar.
- Escribir la/s variables leídas desde MATLAB en la periferia descentralizada Modbus.
- Leer la/s variable/s controladora/s por el DCS desde la periferia descentralizada Modbus.
- Escribir la/s variable/s leídas desde la periferia descentralizada en MATLAB.

4.2.2. Ciclo de Ejecución.

El Gateway ejecuta cíclicamente las siguientes tareas:

- Leer las variables desde la periferia descentralizada.
- Escribir los valores en el modelo simulado de MATLAB.
- Escribe las variables controladas en la periferia descentralizada, la cual genera salidas de tensión que sirven de entrada para los módulos AI810 del sistema DCS.

Este ciclo de ejecución se ejecuta con un periodo definido en el parámetro *Interval.*

4.2.3. Threads del Gateway

Los *threads* que intervienen en el desarrollo del Gateway son:

Scan. Es el thread que ejecuta el scan del Gateway.

winSckUDPMon. Este thread abre un puerto UDP y escucha en el mismo recibiendo los datos que se envían desde la plataforma MATLAB-SIMULINK.

winSckUDPControl. Este thread abre un puerto UDP y escribe los datos leídos de la periferia descentralizada Modbus en la interfaz UDP. La información escrita en la interfaz UDP es leída por el modelo dinámico de planta en la plataforma de simulación MATLAB-SIMULINK.

4.2.4. Diagrama de clases.

El diagrama de clases de la **Figura 15** muestra la arquitectura de clases del Gateway. En el **Anexo VI** se describe la estructura de cada clase.

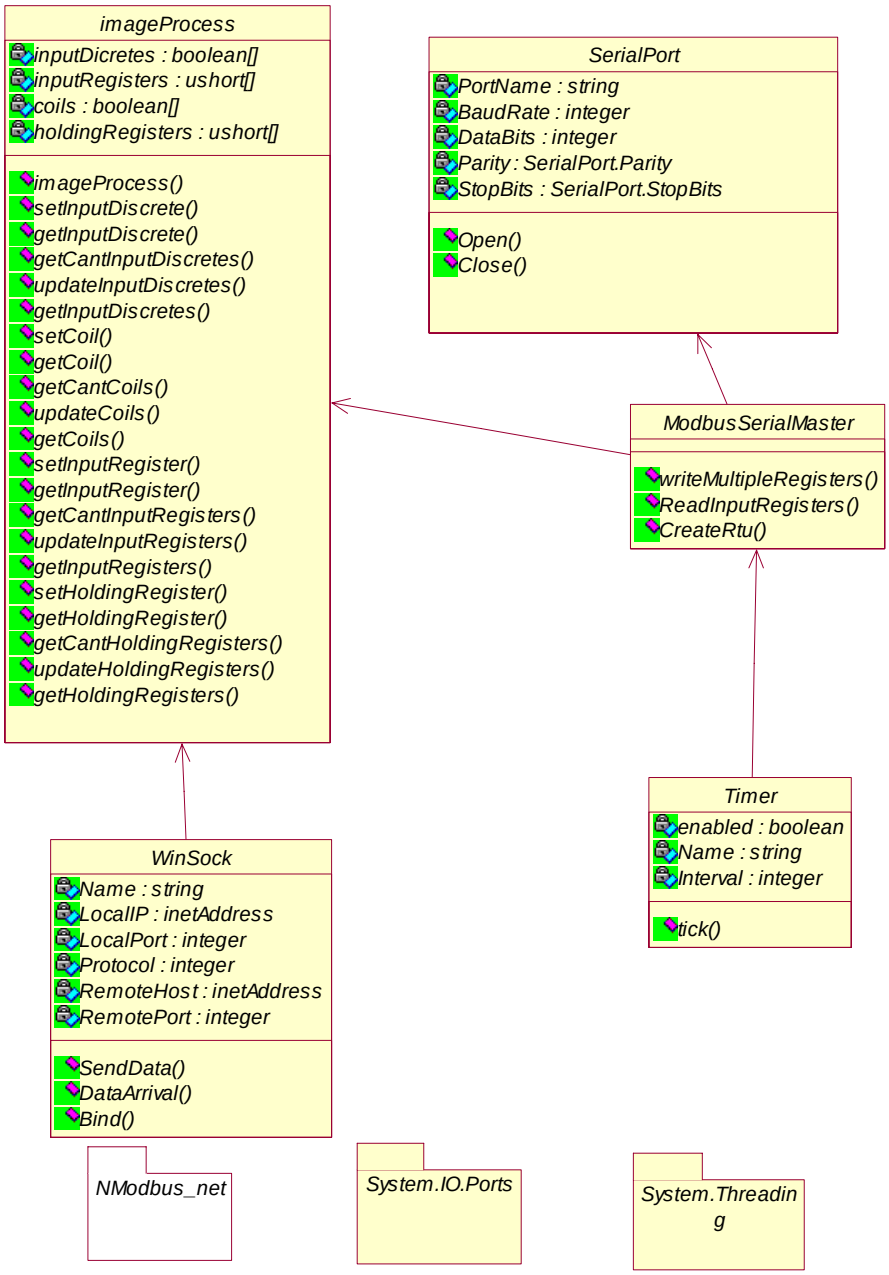

Figura 15. Diagrama de Clases del Gateway Matlab – Modbus Rtu Master.

4.2.5. Interfaz gráfica.

Básicamente la interfaz gráfica está compuesta por una pantalla con tres pestañas: *Monitoreo*, *Control* y *Configuración*.

4.2.5.1. Configuración.

Los parámetros a configurar se dividen en cuatro sectores, como puede observarse en la **Figura 16.** Pantalla de Configuración del Gateway Matlab – Modbus Rtu Master.:

Conexión Modbus Rtu.

Id del esclavo Modbus. Esta dirección se refiere a la dirección de la periferia descentralizada Modbus con la cual se comunicará.

Puerto Serie. Esta lista desplegable muestra un listado con los posibles *puertos serie*, se debe elegir un puerto.

Tasa de Baudios. Es la cantidad de bits transmitidos por segundo. En este caso la transferencia se realiza a una velocidad de 38400 bps, ya que la periferia descentralizada trabaja a esta velocidad.

Bits de Datos. Son el número de bits de una palabra.

Paridad. Indica si se trabaja con paridad ya sea *par* o *impar*. En este caso no se trabaja con paridad.

Bits de parada. Los bits de parada, luego de transmitir una palabra. En este caso es 2.

General.

Cant. de Señales Monitoreo. Este parámetro indica la cantidad de señales que intervienen en el lazo de control. Es decir señales que van hacia la periferia descentralizada.

Cant. de Señales de Control. Este parámetro indica la cantidad de señales de actuación sobre el sistema.

Scan (mseg). Aquí se define la frecuencia con que se ejecutará el scan del mestro modbus.

Conexión UDP – Matlab - Monitoreo.

UDP Monitoreo. Es la dirección IP local a través de la cual leerá los datos enviados por el simulador.

Puerto Monitoreo. Es el puerto local en el cual escribirá el simulador.

Conexión UDP – Matlab - Control.

UDP Control. Es la dirección IP remota en la cual, el Gateway escribirá los datos leídos de la periferia descentralizada. En esta dirección el simulador leerá la acción de control.

Puerto Control. Es el puerto remoto en el cual escribirá el Gateway.

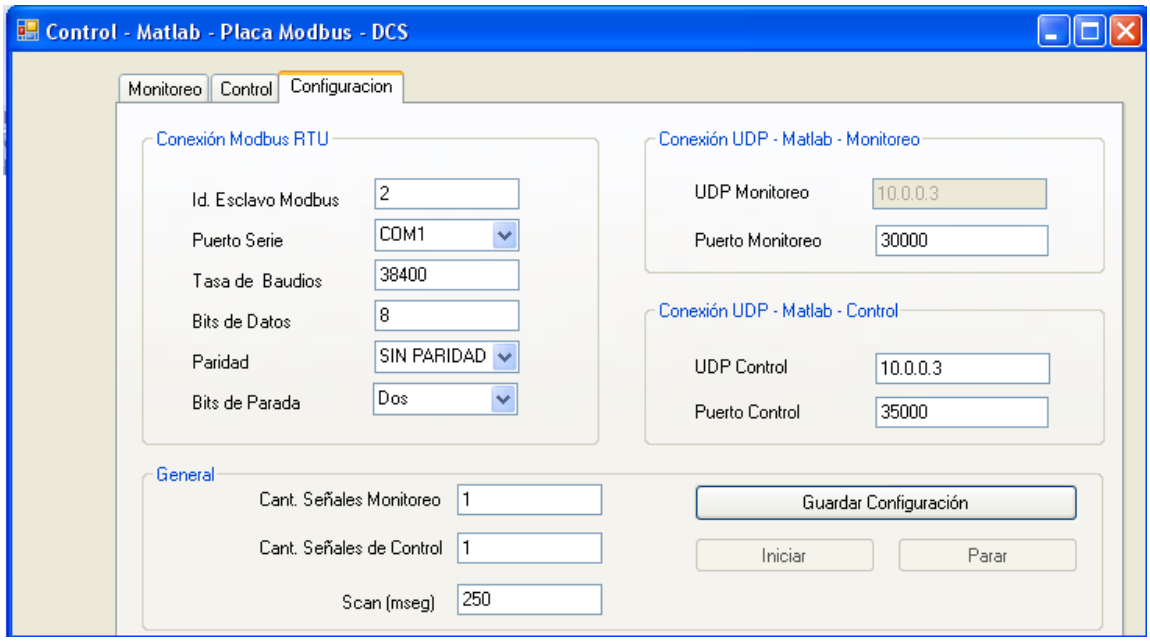

Figura 16. Pantalla de Configuración del Gateway Matlab – Modbus Rtu Master.

4.2.5.2. Monitoreo.

La pestaña *Monitoreo* muestra un campo con el valor transmitido por el simulador MATLAB.

Además en esta pestaña se presenta el estado de la conexión.

4.2.5.3. Control.

La pestaña Control muestra un campo con el valor leído desde la periferia descentralizada Modbus.

Además en esta pestaña se presenta el estado de la conexión con la periferia descentralizada.

4.3 Gateway Matlab – Modbus RTU Slave

4.3.1. Funciones del Gateway

- Leer desde la plataforma MATLAB-SIMULINK la/s variable/s a monitorear.
- Definir un esclavo Modbus RTU para comunicarse con el maestro Modbus RTU del DCS.
- Escribir la/s variable/s leídas desde la periferia descentralizada en la plataforma de simulación MATLAB-SIMULINK.

4.3.2. Threads del Gateway

Los *threads* que intervienen en el desarrollo del Gateway son:

winSckUDPMon. Este thread abre un puerto UDP y escucha en el mismo recibiendo los datos que se envían desde el simulador.

listenSlave. Este thread abre un puerto serie y escucha las peticiones del maestro Modbus RTU del DCS.

4.3.3. Diagrama de clases y artefactos.

El diagrama de clases de la **Figura 17** muestra la arquitectura de clases del Gateway Matlab-Modbus RTU Slave. En el **Anexo VII** se describe la estructura de cada clase.

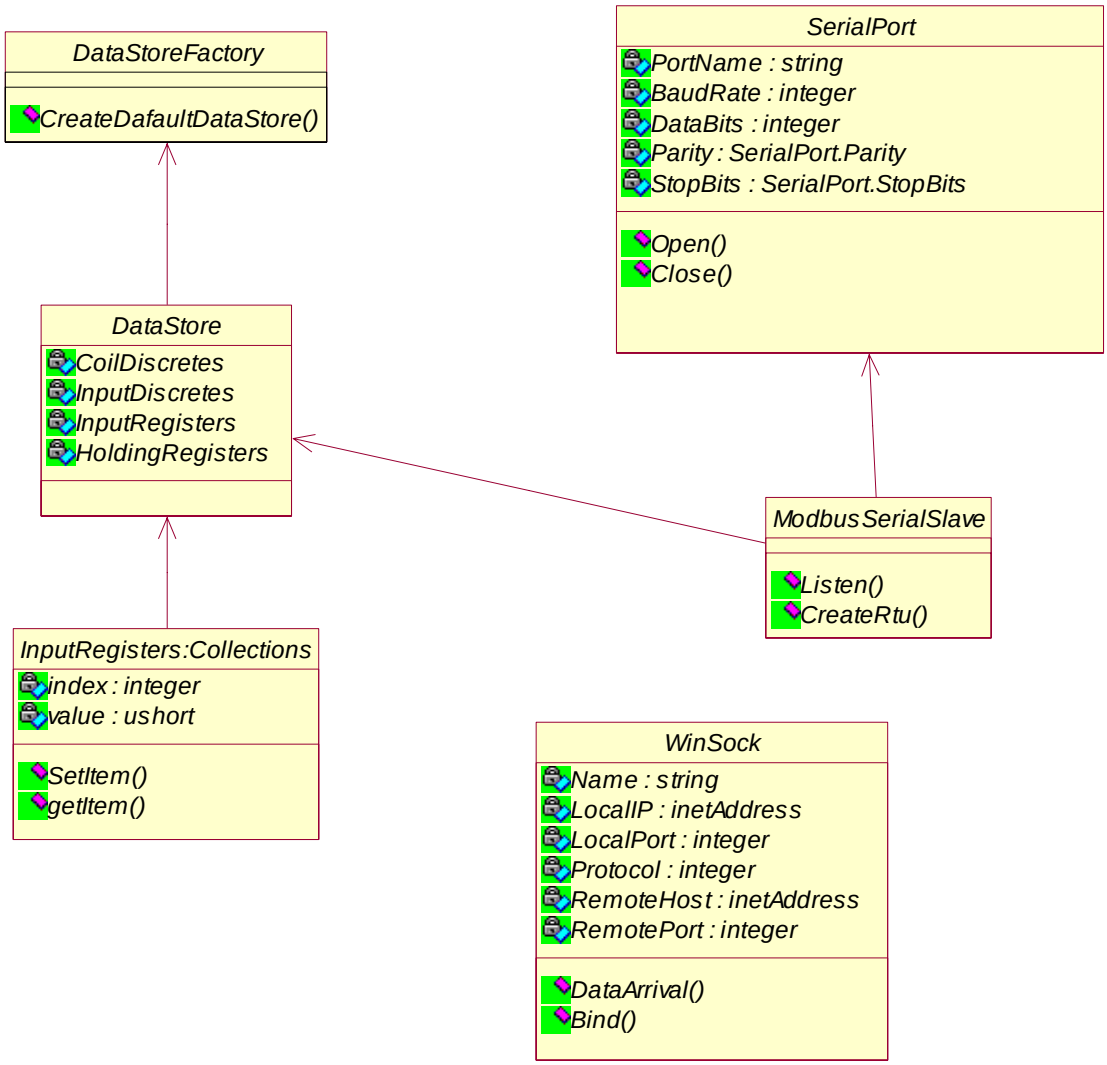

Figura 17. Diagrama de Clases del Gateway Matlab – Modbus Rtu Slave.

4.3.4. Interfaz gráfica.

Básicamente la interfaz gráfica está compuesta por una pantalla con dos pestañas: *Monitoreo* y *Configuración*.

4.3.4.1. Configuración.

Los parámetros a configurar se dividen en cuatro sectores, como puede observarse en la **Figura 18**.

Conexión Modbus RTU.

Id del esclavo Modbus. Esta dirección se refiere a la dirección del esclavo Modbus con el cual se comunicará el maestro Modbus del DCS.

Puerto Serie. Esta lista desplegable muestra un listado con los posibles *puertos serie;* se debe elegir un puerto.

Tasa de Baudios. Es la cantidad de bits transmitidos por segundo. En este caso la transferencia se realiza a una velocidad de 19200 bps, ya que el master del DCS trabaja a esta velocidad.

Bits de Datos. Son el número de bits de una palabra.

Paridad. Indica si se trabaja con paridad ya sea *par* o *impar*. En este caso no se trabaja con paridad.

Bits de parada. Los bits de parada, luego de transmitir una palabra. En este caso es 2.

General.

Cant. de Señales Monitoreo. Este parámetro indica la cantidad de señales que monitorea el sistema desde el modelo simulado.

Conexión UDP – Matlab - Monitoreo.

UDP Monitoreo. Es la dirección IP local a través de la cual se comunicará con el simulador.

Puerto Monitoreo. Es el puerto local en el cual escribirá el simulador.

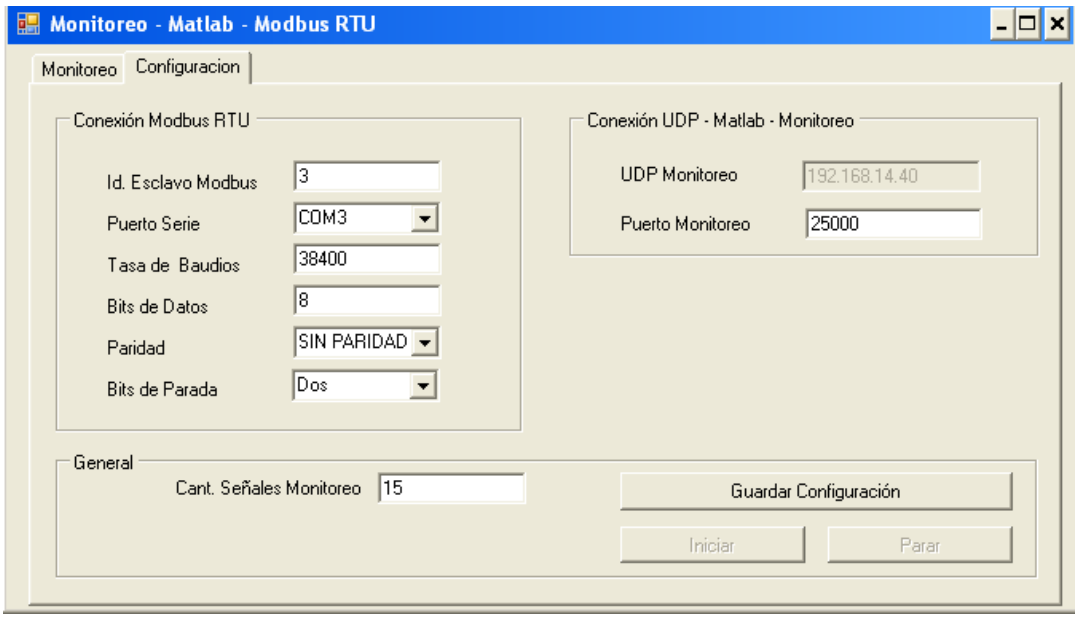

Figura 18. Pantalla de Configuración del Gateway Matlab – Modbus Rtu Slave.

4.3.4.2. Monitoreo.

La pestaña *Monitoreo* muestra los campo con los valores transmitidos por el simulador, para el monitoreo de los mismos a través del DCS.

Además en esta pestaña se presenta el estado de la conexión.

5. Desarrollo de Software en el DCS.

5.1 Introducción.

El proceso de desarrollo de software aplicado en sistemas DCS difiere del proceso de desarrollo tradicional, sobre todo en la fase de codificación, ya que en esta fase además debe definirse la arquitectura de hardware del sistema completa y definitiva, ya que es necesario considerar las comunicaciones y controladores intervinientes en todo el sistema. Las otras fases se asemejan al desarrollo de cualquier sistema software. El ciclo de vida del software que se considera en este trabajo es el mismo que se define en (3). Tal proceso consta de las siguientes fases:

- Conceptual.
- Especificación de Requerimientos.
- Análisis.
- Validación de los Requerimientos.
- Diseño.
- Codificación del Software.
- Integración.
- Validación.

5.2 Desarrollo de Software.

El proceso de desarrollo de este trabajo se centra en las fases de Diseño, Codificación del Software, Integración y Validación. Por otro lado se realiza la *Gestión de la Configuración* en el **Anexo II**.

Generalmente el proceso que se aplica para la especificación y la validación de los requerimientos, se basa en modelos. En el contexto de este trabajo la fase conceptual, la especificación de requerimientos, análisis y su validación fue desarrollada en (1).

En este punto se describen las etapas de diseño, codificación del software y de integración. La etapa de validación no es sencilla debido a que los requerimientos del sistema no están claramente definidos en (1).

Por otro lado se compara la respuesta del sistema utilizando el controlador especificado idealmente contra la respuesta del sistema una vez implementado en el controlador del DCS.

5.2.1. Especificación de Requerimientos.

Principalmente los requerimientos funcionales se pueden dividir en:

- El monitoreo de variables importantes.

- La implementación de un lazo de control.

5.2.1.1. Requerimientos de Monitoreo.

Los requerimientos de monitoreo se obtienen de (2). Los requerimientos de monitoreo no están definidos en cuanto a tiempo de actualización, precisión ni detalles en la presentación; por lo tanto los mismos se definen de acuerdo a lo mostrado en GUI definida en MATLAB.

Las variables a monitorear se muestran definidas en la siguiente tabla:

Tag Variables	Descripción	Unidades de Ingeniería
P	Presión del RPV.	MPa
M	Masa del refrigerante.	Kg
Wbi	Caudal del refrigerante a la entrada al GV.	ºC
Tbi	Temperatura del refrigerante a la entrada a GV.	ºC
Tb	Temperatura media del refrigerante en el GV.	ºC
Psec	Potencia del Secuandario.	MW
Tbo	Temperatura del refrigerante a la salida del GV.	ºC
Wc	Caudal másico del refrigerante a la entrada del núcleo.	Kg/Seg
Tc	Temperatura del refrigerante a la entrada del núcleo.	ºC
Hn	Entalpía del refrigerante a la entrada del núcleo.	KJ/Kg
Pot	Potencia del Reactor.	MW
Tf	Temperatura del combustible.	ºC
Ho	Entalpía a la salida del núcleo.	KJ/Kg
Xch5	Titulo del refrigerante a la salida del 5to nodo de la chimenea.	
Dch5	Densidad del refrigerante a la salida del 5to nodo de la chimenea.	Kg/m^3
Per.	Periodo.	

Tabla 2. Variables Monitoreadas.

Un requerimiento a considerar para el HMI es la pantalla definida en (2), la cual se muestra en la **Figura 19**.

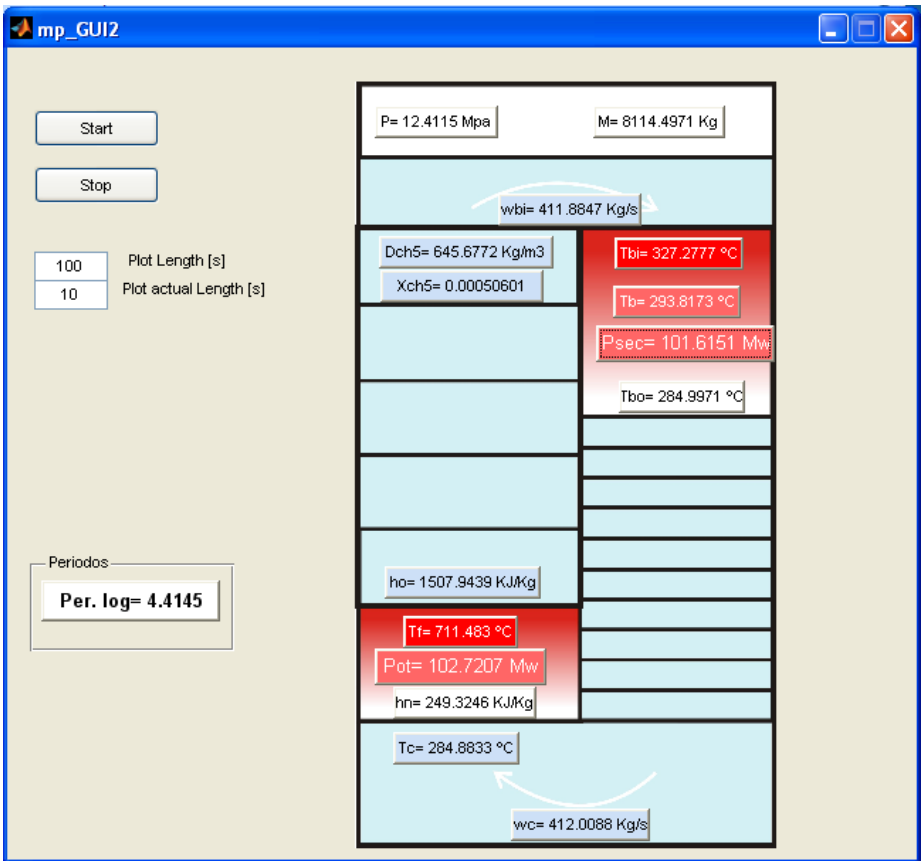

Figura 19. Interfaz definida para el monitoreo de las variables en la interfaz MATLAB.

5.2.1.2. Requerimientos de Control.

Los requerimientos de control están definidos de acuerdo a la siguiente ecuación que representa la función transferencia.

$$\boldsymbol{Gc_2(s)} = \frac{20\,s+1}{s}\;\; 1.05e-4\;\; \frac{partes\_enteras}{MPa}$$

Esta función representa el lazo de control de presión del primario como se encuentra implementado en (2). El control parte de una señal de error de presión en MPa que es la entrada al controlador PID. La salida del controlador es señal para el motor del mecanismo de barras, una vez obtenida la posición de las barras se obtiene la reactividad en dólares.
Los parámetros que se definen para el control se expresan en la **Tabla 3**:

Parámetro	Descripción	Valor
Sp	Set Point. Valor de referencia	12.25
Kp	Ganancia Proporcional	20
Ki	Tiempo de integración	1
Kd	Tiempo de Derivación	0

K	Ganancia del sistema que define el peso de las barras en reactividad.	$1.05e^{-4}$

Tabla 3. Parámetros definidos para el control.

5.2.2. Diseño.

Las consideraciones de diseño que se toman de acuerdo a los requerimientos definidos se pueden clasificar en: *criterios para el monitoreo, criterios para el control.*

5.2.2.1. Criterios de diseño para el monitoreo.

Considerando que para monitorear las variables de la planta simulada se utiliza el *Gateway – Matlab – Modbus RTU Slave (4.3)* el cual utiliza un image process de 15 input registers de 16 bits; de los cuales los más significativos se mantienen constantes, se prefiere correr el punto decimal dos lugares a la derecha, o bien un solo lugar, dependiendo de la variable, para no perder precisión.

Tag Variables	Descripción	Desplazamiento	Rango(*)
M	Masa del refrigerante.	-	8000- 8500
Wbi	Caudal del refrigerante a la entrada al GV.	2 lugares (x100)	350 – 450
Tbi	Temperatura del refrigerante a la entrada a GV.	2 lugares (x100)	300 – 350
Tb	Temperatura media del refrigerante en el GV.	2 lugares (x100)	250 – 350
Psec	Potencia del Secuandario.	2 lugares (x100)	80 – 110
Tbo	Temperatura del refrigerante a la salida del GV.	2 lugares (x100)	250 – 350
Wc	Caudal másico del refrigerante a la entrada del núcleo.	2 lugares (x100)	350 – 480
Tc	Temperatura del refrigerante a la entrada del núcleo.	2 lugares (x100)	200 – 350
hn	Entalpía del refrigerante a la entrada del núcleo.	2 lugares (x100)	200 – 350
Pot	Potencia del Reactor.	2 lugares (x100)	50 – 150
Tf	Temperatura del combustible.	1 lugar (x10)	600 – 800
ho	Entalpía a la salida del núcleo.	1 lugar (x10)	700 – 1600
Xch5	Titulo del refrigerante a la salida del 5to nodo de la chimenea.	5 lugares (x10000)	0 – 1

Dch5	Densidad del refrigerante a la salida del 5to nodo de la chimenea.	1 lugar (x10)	500 – 700
Per.	Periodo.	2 lugares (x100)	2 – 4

Tabla 4. Desplazamiento del punto decimal a la derecha de las variables de monitoreo.

(*) El rango establecido es un valor de referencia obtenido a partir de los valores observados en el sistema a lazo abierto, para justificar el desplazamiento del punto decimal.

Considerando que a los Input Registers puede accederse mediante el protocolo MODBUS, se debe diseñar un maestro Modbus en el controlador del DCS para acceder a los valores del Gateway Matlab – Modbus Rtu Slave.

El maestro Modbus debe implementar la función 04 de Modbus -*READ INPUT REGISTERS.*

Con lo descripto pueden definirse las funciones que debe ejecutar el software desarrollado. En la **Tabla 5** se muestran las funciones a diseñar.

Referencia	Función
1.	El Master se conecta con el esclavo.
2.	El Master envía una solicitud para la lectura de *Input Registers*.
3.	Se muestran los valores recibidos.

Tabla 5. Curso Normal de Eventos sobre el caso de uso Monitoreo.

Considerando que el lenguaje de codificación que se utilizará será *Function Block Diagram*, se diseña a partir de las funciones definidas en la tabla anterior.

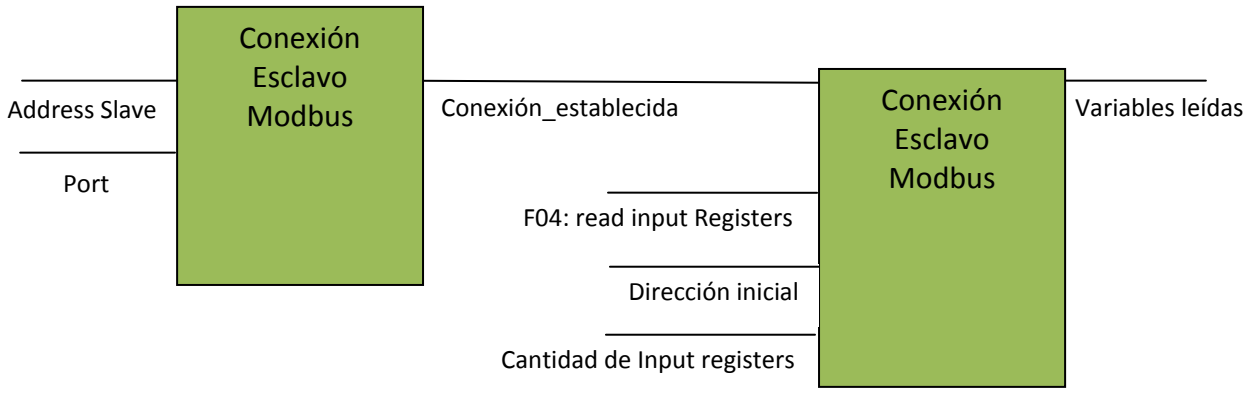

Figura 20. Diseño de las funciones de monitoreo mediante bloques.

5.2.2.2. Criterios de diseño para el Control.

Considerando que el lazo de control especificado solo tiene una variable de entrada y una variable de salida de 10 bits ya que el DCS se conecta con la periferia descentralizada Modbus. Por esto se utiliza solo la parte decimal de la variable de entrada ya que el rango de la misma está entre 12.25 y 12.45 MPa. La variable de salida tiene un rango entre 0 y 100 por lo que se corre el punto decimal un lugar, para enviarlo a través de la periferia descentralizada Modbus.

Tag Variables	Descripción	Desplazamiento	Rango(*)
P	Presión del RPV.	3 Lugares (x1000)	0 – 1000
U	Salida del Controlador	1 lugares (x10)	0 – 1000

Tabla 6. Desplazamiento del punto decimal a la derecha de las variables de Control.

En el diseño del lazo de control se utilizó un bloque FBD donde se configuraron los parámetros definidos en los requerimientos.

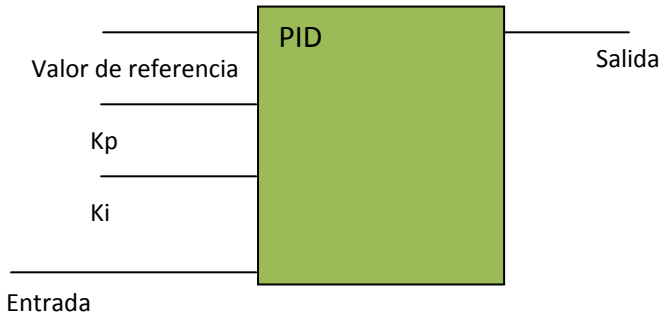

Figura 21. Diseño del Lazo de Control a través de un PID.

5.2.3. Codificación del Software.

La codificación del software en un sistema DCS implica además la configuración de la arquitectura del hardware.

La **Figura 14** muestra un esquema de la arquitectura utilizada en el DCS. A partir de esta se configura la arquitectura en el sistema utilizando la herramienta *Control Builder Professional M* 5.0.2. La **Figura 22** muestra esta arquitectura.

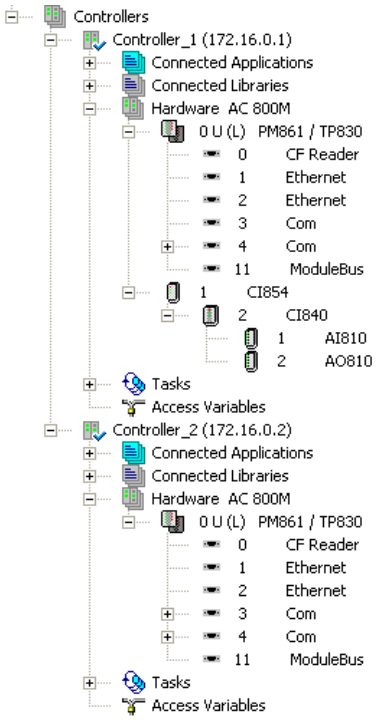

Figura 22. Arquitectura del sistema configurada en el sistema DCS – 800xA

La codificación del sistema se divide en: codificación del control, codificación del monitoreo.

La estructura de codificación se plantea creando dos aplicaciones: Application_1 y Application_2 las cuales se cargaran en el *Controlador 1* y *Controlador 2* respectivamente.

5.2.3.1. Codificación del Monitoreo, Application_2.

La codificación del monitoreo implica en este caso la configuración de un puerto serie en el controlador para programar un maestro Modbus RTU, el cual se comunicará con el Gateway Matlab – Modbus Rtu Slave. El puerto serie del controlador que se utiliza es el puerto 3, ya que el puerto 4 se utiliza solo para configuración del sistema.

Por otro lado se requiere que las consultas Modbus se realicen secuencialmente, para esto se considera utilizar dos temporizadores para generar la periodicidad deseada. Esto podemos observarlo en el siguiente diagrama el cual muestra la periodicidad que se requiere para que los bloques Modbus realicen cíclicamente las consultas.

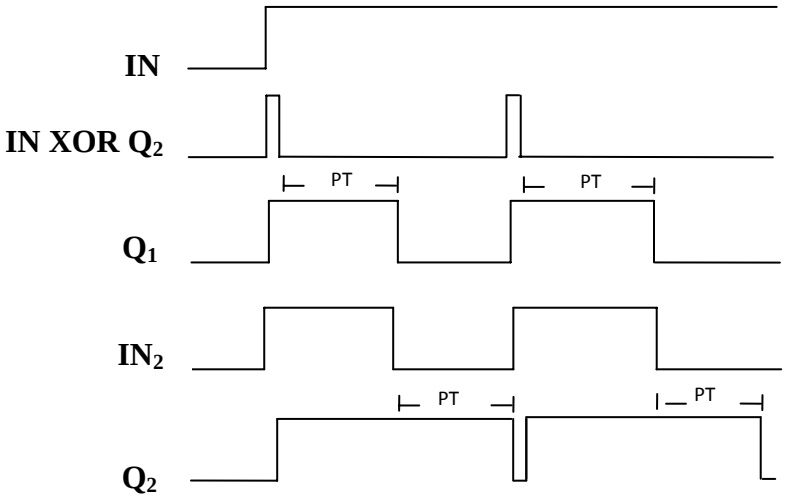

Figura 23. Esquema cíclico para el envió de consultas MODBUS mediante dos temporizadores ToF.

Si bien en la codificación del monitoreo se utiliza el FBD, también se utiliza el lenguaje ST para resolver los desplazamientos de las variables mencionados con anterioridad.

La **Figura 24** muestra la codificación realizada en el DCS.

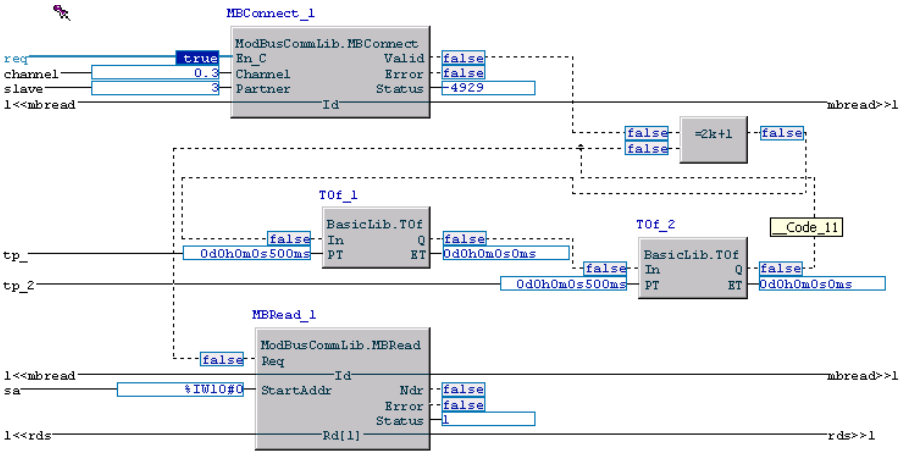

Figura 24. FBD con la comunicación MODBUS.

La Tabla 7 describe brevemente los bloques de función utilizados en la programación para el monitoreo.

Bloque	Descripción
MB_Connect	Este bloque realiza la lectura de registros en un esclavo MODBUS. **En_C**. Habilita el bloque. Variable booleana de entrada. **Channel**. Define el puerto del controlador que se utilizará para

	establecer la comunicación con el esclavo. **Partner**. Define la dirección MODBUS del esclavo. **Id.** Este parámetro debe ser conectado al módulo para configurar la comunicación. **Valid**. Si se establece la conexión esta salida devuelve *TRUE.* **Error**. Devuelve *TRUE* si se genera algún error. **Status**. Devuelve el código de error.
MBRead	Este bloque establece la conexión con el esclavo. **req**. Habilita el bloque. Variable booleana de entrada. Debe tenerse en cuenta que este bloque requiere que se haya establecido previamente la conexión con el esclavo MODBUS. Una vez que el req se establece en *true* se envía una petición al esclavo Modbus, pero solo una petición; es por esto que se utilizan los temporizadores para establecer un ciclo en la variable *req* y así lograr actualizar el valor de las variables periódicamente. El periodo de actualización se define como la suma de PT_1+PT_2, el cual se configuró en 1s (500ms + 500ms). **Id.** Este parámetro debe ser conectado al módulo para configurar la comunicación. **StartAddr**. Este parámetro define la función Modbus que se consultará. Este parámetro debe definir lo siguiente: FNCOD#R, donde **FN**. Indica la función Modbus. Ej IW. Lectura de input registers. **COD.** Define en que código se indicará el registro inicial de la función MODBUS. Ej. 10 decimal. **#R**. Indica el registro inicial. **Rd[1].** Este parámetro recibe como entrada un vector que determina la cantidad de registros que se leerán en el esclavo Modbus. Cada elemento del vector debe ser compatible con la variable leída del esclavo Modbus; ya que los valores de la repuesta del esclavo se escribirán en esta variable. **Nrd**. Si la consulta se realiza correctamente esta salida devuelve *true.* **Error**. Devuelve *true* si se genera algún error. **Status**. Devuelve el código de error.
=2k+1	Este bloque ejecuta la función booleana XOR, entre dos entradas.
Tof	Temporizador de tipo Tof. Este temporizador funciona de acuerdo a la siguiente gráfica.

	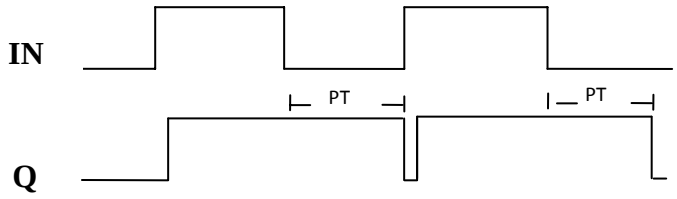 In. Variable booleana de entrada al temporizador. Q. Variable booleana de salida al temporizador. PT. Variable time que define el intervalo temporal que se mantendrá la salida Q, una vez que la entrada In haya pasado a *FALSE*.

Tabla 7. Bloques utilizados para el monitoreo.

La **Tabla 8** muestra el listado de las variables utilizadas en la programación del monitoreo.

Variable	Tipo	Descripción
req	Boolean	Esta variable habilita al módulo MBConnect para establecer la conexión con el esclavo MODBUS.
channel	Integer	Indica que puerto del controlador se utilizará para la comunicación MODBUS.
slave	Int	Indica la dirección del esclavo modbus.
mbread		Esta variable es una variable de configuración para el modulo MBConnect.
tp_	Time	Esta variable define el valor temporal para el PT del temporizador Tof_1.
tp_2	Time	Esta variable define el valor temporal para el PT del temporizador Tof_2.
sa	word	Esta variable define la función modbus que se desea consultar al esclavo modbus; dirección inicial de registros. **IW**. Función 04. Lectura de input registers. **10**. Indica que el registro de inicio se indicará en decimal. **#0**. Indica el registro inicial.
rds		Esta variable es un vector que define la cantidad de registros que se desean leer del esclavo MODBUS. En esta variable se escriben los

		valores leídos.
masa	Real	Masa de agua en el circuito primario.
wbi	Real	Caudal de entrada al generador de vapor.
Tbi	Real	Temperatura de entrada al generador de vapor.
Tb	Real	Temperatura media en el generador de vapor.
Psec	Real	Potencia del secundario.
Tbo	Real	Temperatura de salida del primario del generador de vapor.
wc	Real	Caudal de salida del plenum inferior.
Tp	Real	Temperatura de salida del plenum, la cuál es equivalente a la temperatura de entrada al núcleo.
hc	Real	Entalpía de entrada al núcleo.
Pot	Real	Potencia del primario.
Tf	Real	Temperatura del núcleo.
hh	Real	Entalpía de salida del núcleo.
Xch5	Real	Título del agua a la salida del último nodo de la chimenea.
Dench5	Real	Densidad del agua en el último nodo de la chimenea.
Per	Real	Periodo logarítmico del reactor.

Tabla 8. Variables utilizadas para la programación del monitoreo.

Anteriormente se mencionó que además de programar el maestro MODBUS en FBD, se utilizó el lenguaje *Structured Text* para resolver el desplazamiento efectuado en las variables de monitoreo. A continuación se muestra el código ST (Structured Text) con las variables utilizadas para el monitoreo, partiendo del vector rds.

```
Masa:=rds.register_0;
wbi:=rds.register_1*0.01;
Tbi:=rds.register_2*0.01;
Tb:=rds.register_3*0.01;
Psec:=rds.register_4*0.01;
Tbo:=rds.register_5*0.01;
wc:=rds.register_6*0.01;
```

```
Tp:=rds.register_7*0.01;
hc:=rds.register_8*0.01;
Pot:=rds.register_9*0.01;
Tf:=rds.register_10*0.1;
hh:=rds.register_11*0.1;
Xch5:=rds.register_12*0.0001;
Dench5:=rds.register_13*0.1;
Per:=rds.register_14*0.01;
```

A partir de las variables monitoreadas se realiza una interfaz gráfica para el monitoreo y el control de la aplicación. La interfaz consta de tres partes principales: El Workplace de visualización de variables, el faceplate de control y los trends (presión del primario, potencia del primario, potencia del secundario), para el seguimiento de tendencias.

Las siguientes figuras muestran la interfaz gráfica del sistema.

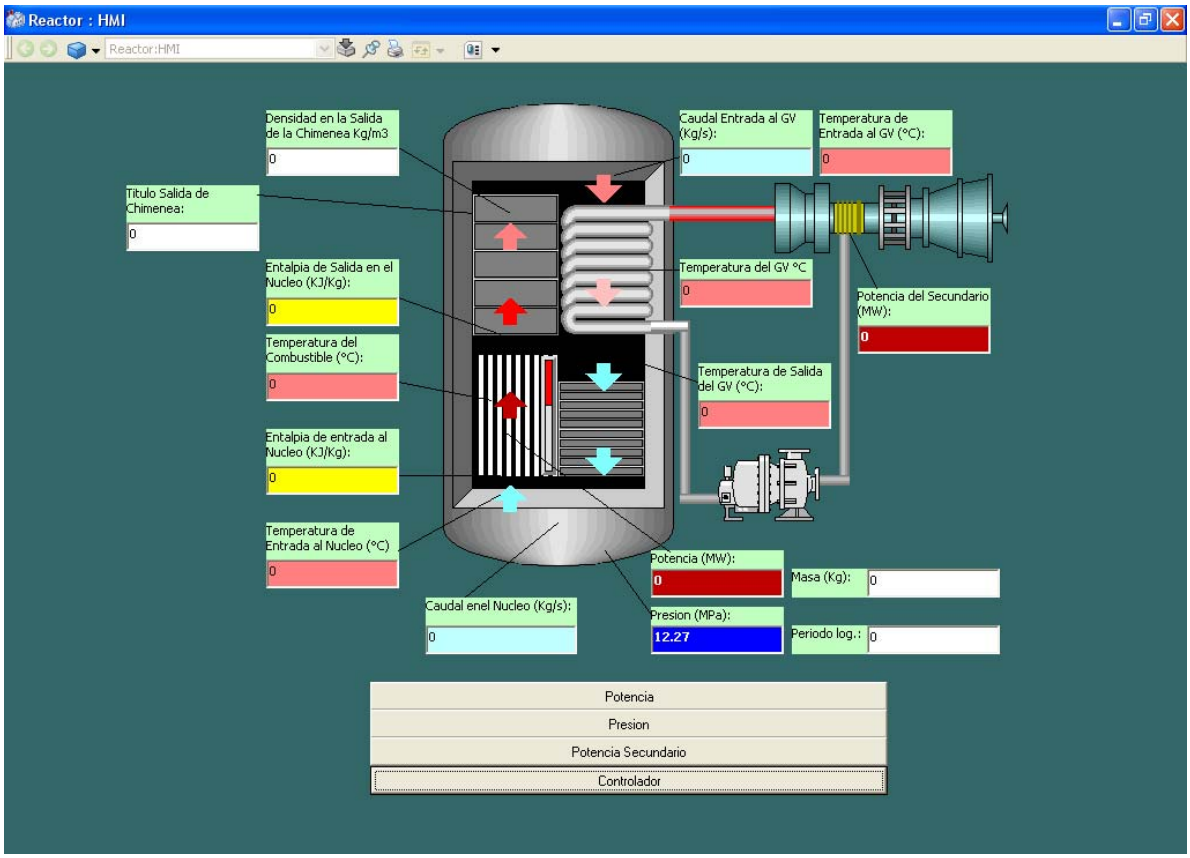

Figura 25. HMI del sistema.

El HMI muestra básicamente la estructura del reactor con la que se trabajó, las variables monitoreadas dentro del reactor. Además en el HMI se muestran tres botones, los cuales despliegan los trends: de la potencia del primario, la potencia del secundario, la presión del primario; así como un faceplate para el control del PID del controlador 1.

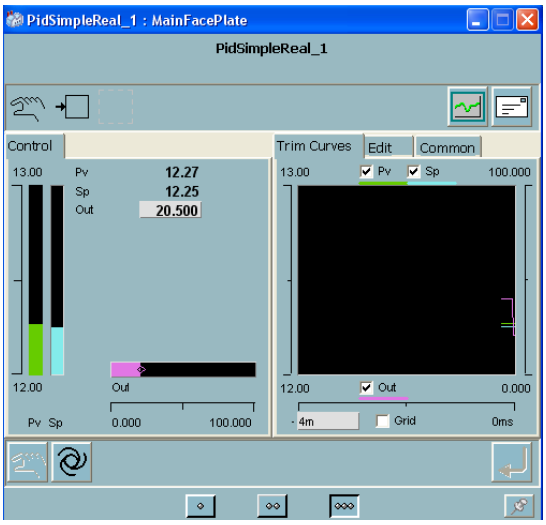

Figura 26. Faceplate para el control del PID que controla el lazo de presión.

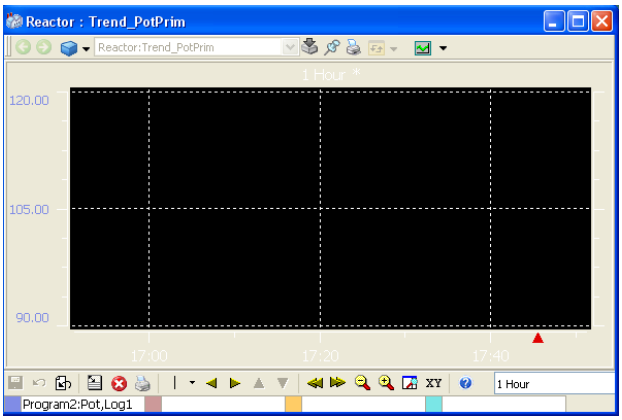

Figura 27. Trend de la potencia del primario.

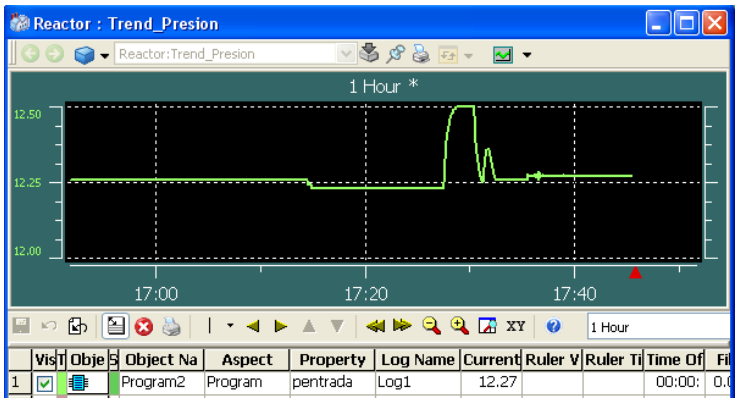

Figura 28. Trend con la presión del primario,

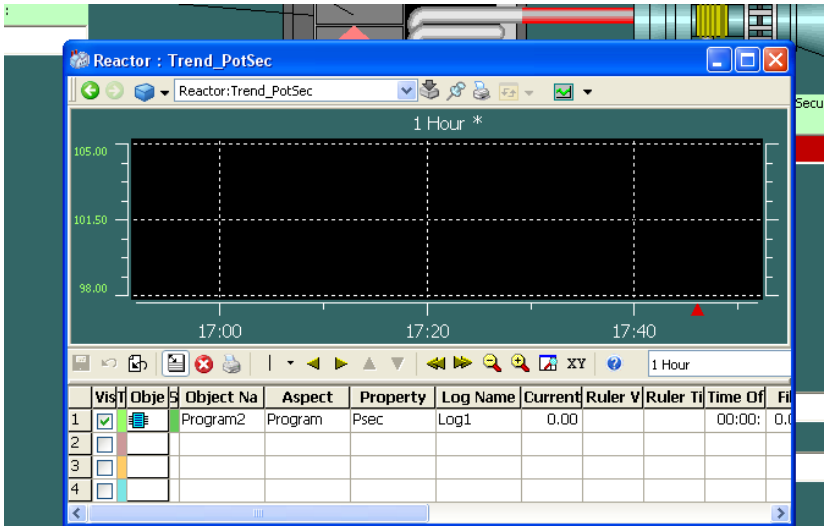

Figura 29. Trend de la potencia del secundario.

5.2.3.2. Codificación del Control, Application_1.

La codificación del control es relativamente sencilla en este caso ya que se trata de un PID, y debido a que los DCS ya tienen bloques de funciones especiales para ello; la codificación se reduce a la configuración de los parámetros. La configuración de los parámetros, si bien es sencilla, hay que tener ciertos cuidados con respecto al tipo de implementación del controlador PID por parte del fabricante. Existen tres tipos principales de algoritmos PID implementados por los fabricantes más importantes:

Algoritmo ideal:

$$\text{out} = K_c\left[e(t) + \frac{1}{I}\int e(t)dt + D\frac{d\,e(t)}{dt}\right]$$

Algoritmo paralelo:

$$\text{out} = K_p e(t) + \frac{1}{I}\int e(t)dt + D\frac{d\,e(t)}{dt}$$

Algoritmo en Series:

$$\text{out} = K_c\left[e(t) + \frac{1}{I}\int e(t)dt\right]\left[1 + D\frac{d\,e(t)}{dt}\right]$$

Si bien las diferencias son sutiles, obviamente los resultados obtenidos son diferentes.

Por lo tanto lo que primero que debe analizarse es la forma en que se implementan los algoritmos de control PID en la plataforma MATLAB-SIMULINK y en los controladores ABB.

La plataforma MATLAB-SIMULINK, por defecto implementa el controlador de tipo paralelo, lo cual fue comprobado en el help del mismo. Por otro lado los controladores de ABB implementan los algoritmos de tipo ideal, esta información fue obtenida de (12).

Para convertir un controlador de tipo paralelo en un controlador de tipo ideal se utilizan las siguientes fórmulas (13):

$$K_c = K_p$$
$$I = K_p I_p$$
$$D = \frac{D_p}{K_p}$$

De este análisis resulta que los parámetros a configurar en el bloque PID del controlador son:

Gain=20

Ti=20

Se utiliza, además, en otra pestaña algunas sentencias en el lenguaje ST (Structured Text) para acondicionar la señal de entrada al bloque PID como la salida enviada al modelo, a través de la periferia descentralizada. Las siguientes figuras muestran la codificación en el DCS.

A continuación se muestra una captura de pantalla con el controlador PID.

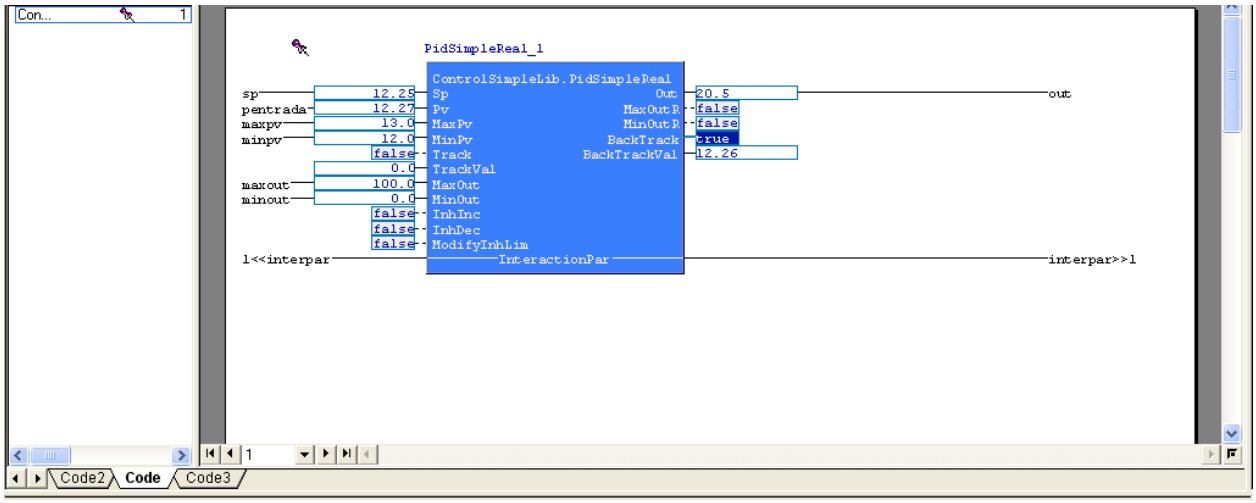

Figura 30. Controlador PID para el lazo de presión del reactor.

El bloque PidSimpleReal corresponde a un controlador PID sencillo incluido en las librerías del software de programación *Control Builder Professional M*. Este bloque tiene la particularidad, a diferencia de otros bloques PID, de que sus entradas y salidas son de tipo *Real*; mientras que los otros bloques PID poseen entradas y salidas *RealIO*. Una variable de tipo *RealIO* implica muchos parámetros más como por ejemplo valor mínimo, valor máximo, estado, valor, etc. Esto hace al bloque PidSimpleReal sencillo de utilizar.

A continuación en la **Tabla 9** se describen los parámetros utilizados por el bloque PID.

Parámetro	Tipo	Descripción
Sp	Real	Este parámetro define el *Set Point* para el controlador.
Pv	Real	Este parámetro define la entrada para el controlador. El parámetro de control.
interactionPar		Este parámetro es obligatorio en el bloque PID ya que permite establecer otros parámetros, para el control. **Gain**. Establece la ganancia del controlador. **Ti.** Establece la constante de integración. **Td**. Establece la constante de derivación. Este parámetro no es utilizado en este controlador. **ControllerType**. Define el tipo de controlador. 1 – Proporcional, 2-Proporcional-Integraltivo, 3-Proporcional-Derivativo, 4-Proporcional-Integral-Derivativo. **AutoMode**. Define si el controlador actúa automáticamente o en forma manual. **OutManValue**. Este parámetro se utiliza para fijar el valor de salida, cuando el parámetro *AutoMode* está fijado en *False.*
Out	real	Salida del controlador.

Tabla 9. Parámetros utilizados del PidSimpleReal.

Las variables utilizadas para programar este controlador se muestran en la siguiente tabla.

Variable	Tipo	Descripción
pentrada	Real	Esta variable se conecta al parámetro Pv del Bloque PID.
pi	RealIO	Esta variable de entrada es la que se conecta directamente al módulo de entrada AI810. Este módulo recibe la corriente generada por la periferia remota slave MODBUS RTU, a través de una resistencia variable, y la convierte a un valor real.
po	RealIO	Esta variable de salida es la que se conecta directamente al módulo de salida AO810. Este módulo convierte un valor real en una corriente de salida, la cual se transforma, a través de otra resistencia variable, en tensión de entrada para la periferia remota slave MODBUS RTU.
aux	Real	Esta variable toma el valor de pi.value y acondiciona ese valor leído para representar el valor decimal de la presión. Este valor decimal se

		suma al valor constante 12, lo cual sirve como entrada para el controlador PID. Se utiliza este artilugio debido a que se conoce que la presión en el modelo, a lazo abierto, tiene un rango 12.25 – 12.45, es por esto que se decidió enviar solo los valores decimales de la presión.
out	Real	Esta variable se conecta al controlador PID y se multiplica por el factor K para incrementar el valor de la salida y reducir el error en la periferia remota slave MODBUS RTU, ya que para valores pequeños la tensión generada por la periferia remota se hace menos estable. Este puede hacerse ya que el rango de la salida del controlador es 0-100.
K	Integer	Esta variable define el factor para incrementar el valor de salida del controlador.

Tabla 10 . variables utilizadas para programar el PID.

Para explicar claramente la relación entre las variables definidas en **Tabla 10** se muestran las líneas de código, escritas en ST (Structured Text), que las relacionan:

```
interpar.ControllerType:=1;
interpar.Gain:=20;
interpar.Ti:=20;
aux:=trunc(pi.value*0.1);
pentrada:=aux*0.01+12;
po.value:=out*K;
```

5.2.4. Integración.

Las pruebas de integración se clasificaron en 4 grupos:

Grupo 1. *Pruebas de comunicación entre Matlab y los Gateways Matlab – Modbus Rtu*

Estas pruebas consistieron en verificar que al ejecutarse el modelo en MATLAB-SIMULINK, se establecía conexión con los Gateway Matlab – Modbus Rtu visualizando el monitoreo de las variables. Para ello se fijo los valores en la plataforma MATLAB-SIMULINK y se observó el monitoreo de los mismos en los Gateways.

Por otro lado se fijo valores en el “Gateway Matlab – Modbus Rtu Master” y se observó con un scope en la plataforma MATLAB-SIMULINK el valor obtenido.

Grupo 2. *Pruebas de comunicación entre el DCS y los Gateways Matlab – Modbus RTU.*

Estas pruebas consistieron en verificar que al fijar valores en el DCS, era posible visualizar los mismos en los *Gateways Matlab – Modbus Rtu* ; y viceversa, al fijar valores en los Gateways debe ser posible visualizar los mismos en el DCS.

Estas pruebas de integración dieron origen a lo desarrollado en el **Anexo III**: **¡Error! No se encuentra el origen de la referencia.**.

Grupo 3. *Pruebas de comunicación entre el DCS y la plataforma MATLAB-SIMULINK.*

Una vez concluidas las pruebas de integración anteriores, se procedió a ejecutar todo el sistema:

1. En principio se configura el PID, a través de su faceplate, en modo manual. Se lleva la salida a 0, para asegurarse que no haya quedado cargado con un valor de prueba anterior.
2. Se ejecutan los gateways. Considerando que ambos trabajan con una interfaz UDP distinta; y que a su vez que esta interfaz es detectada automáticamente por el Gateway según la prioridad de la interfaz para el Sistema Operativo. Se debe setear la prioridad de las mismas para que coincidan con las interfaces definidas en el modelo MATLAB: 192.168.14.38 para el control y 192.168.14.40 para el monitoreo.

   Por lo tanto se debe cambiar la prioridad de la interfaz antes de arrancar los gateways. Para definir la prioridad de la interfaz UDP se logra desde:

   Start->Control Panel->Network Connections, y en el menú Advanced se elige la opción Advanced Setting… Allí aparecerá una ventana con las interfaces de red disponibles.

   Antes de ejecutar el *Gateway Matlab – Modbus Rtu Slave* se debe definir como interfaz primaria la interfaz *Secondary Network Control (192.168.14.40).* Antes de ejecutar el *Gateway Matlab – Modbus Rtu Master* se debe definir como interfaz primaria la interfaz Primary *Network Control (192.168.14.38).*

   Una forma de verificar que las interfaces han sido configuradas correctamente, consiste en mirar el marco *Conexión UDP-Matlab – Monitoreo* de cada Gateway para observar si están bien definidas las interfaces.

   Para el *Gateway Slave* la interfaz UDP debe ser 192.168.14.40 y para el *Gateway Master* la interfaz UDP debe ser 192.168.14.38.

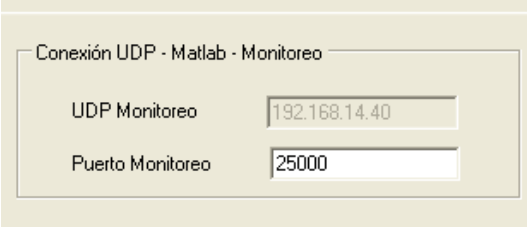

Figura 31. Interfaz UDP para el monitoreo del *Gateway Slave.*

3. Una vez que los Gateways están ejecutándose, se lleva el control en el DCS (faceplate del PID) a modo automático, y se comienza la ejecución del modelo en la plataforma MATLAB-SIMULINK.

Esta prueba de integración consiste en observar que el DCS interactúa con el modelo ejecutándose en la plataforma MATLAB-SIMULINK.

Grupo 4. Verificación de los recursos utilizados.

Considerando que en una PC se ejecuta el modelo de planta y ambos gateways, el procesamiento de esta PC es crítico, ya que las aplicaciones se ejecutan en tiempo real. Para ello se analiza el desempeño de la PC ejecutando estos programas. Con el *Windows Task Manager* se analiza la utilización de los recursos; se observa que el procesamiento llega al 100%, incluso en algunas situaciones las aplicaciones se han bloqueado.

Con esto se definió que el tiempo de scan mas apropiado para el *Gateway Matlab – Modbus Rtu Master* es de 300ms, lo cual de todas formas no reduce la performance de todo el sistema. Este *Gateway* consume mucho procesamiento.

5.2.5. Validación.

Considerando que los requerimientos no fueron definidos en los documentos (1) y (2) la validación se centra en la comparación de la evolución de la presión, la potencia del primario, y la del secundario, producidas por el control ideal simulado en la plataforma MATLAB-SIMULINK y el controlador implementado en el DCS 800xA.

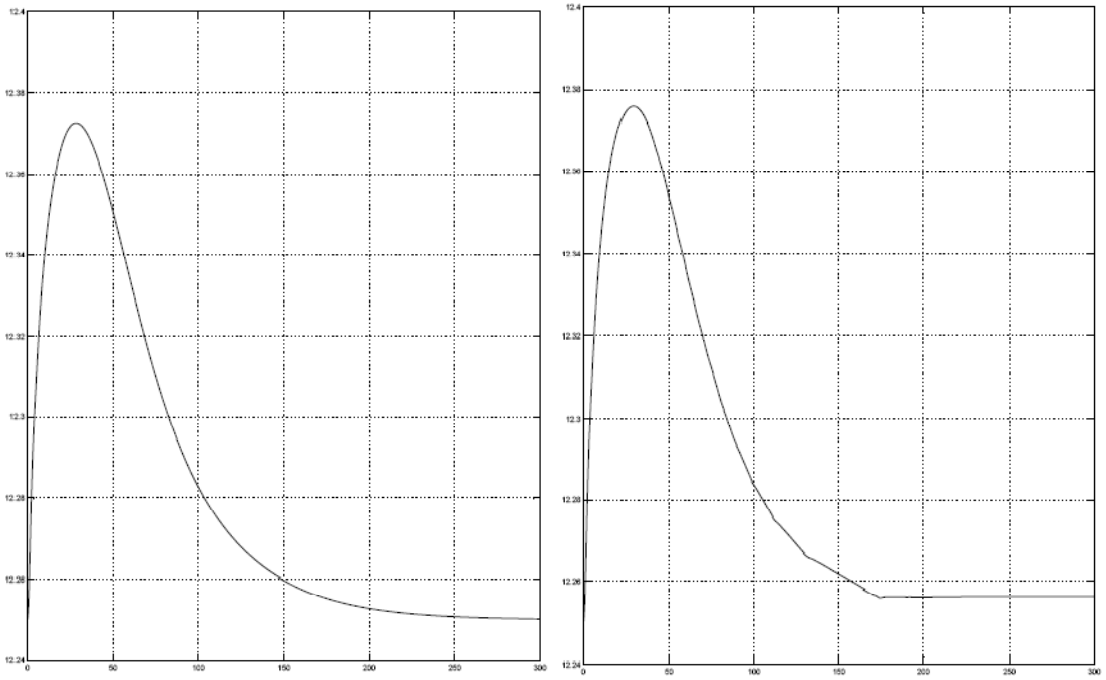

Figura 32. Evolución de la presión utilizando el controlador ideal vs Control DCS.

Si bien la evolución de la presión no son exactamente iguales, la convergencia se da en forma similar. Podría asemejarse modificando los parámetros del PID. Sobre todo disminuyendo la ganancia del controlador.

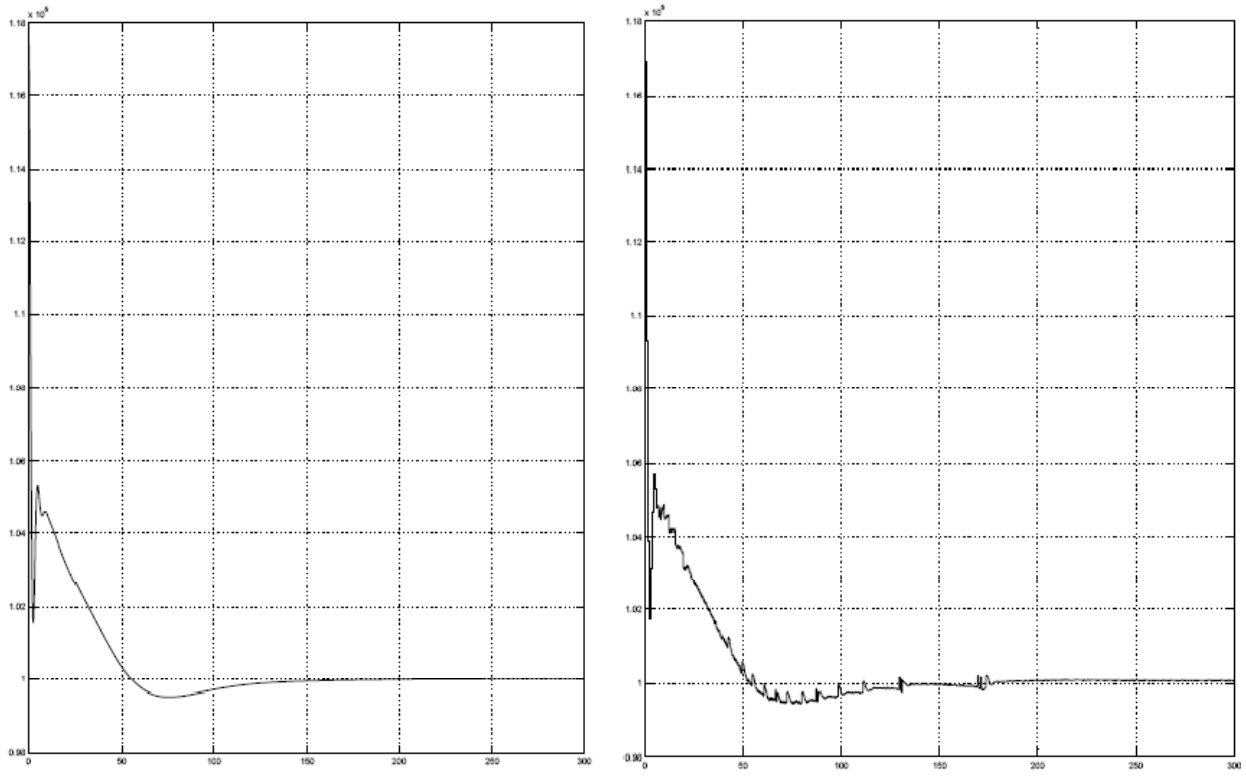

Figura 33. Evolución de la Potencia del Primario utilizando el control ideal vs el control del DCS.

Aquí también se observan variaciones, en este caso las variaciones son más notorias debido a los impulsos generados por la alta ganancia del controlador que el integrador no llega a compensar cuando se produce una disminución importante del error.

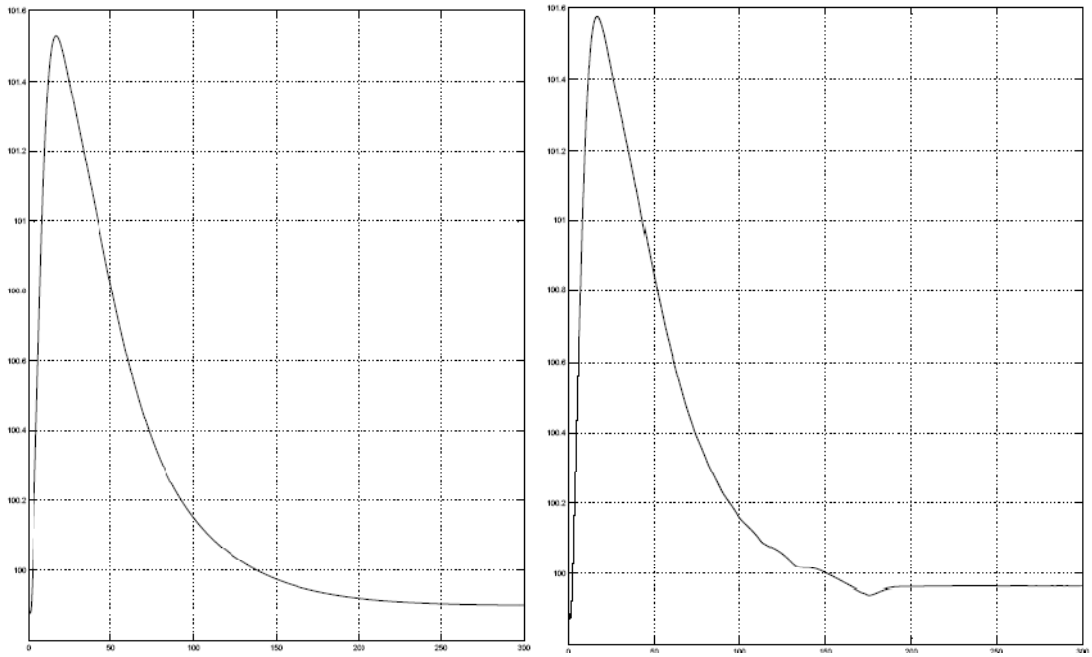

Figura 34. Evolución de la potencia del Secundario con el controlador ideal vs el controlador DCS.

En el caso de la potencia del secundario también hay variaciones. Obviamente la presión influye en las demás variables monitoreadas.

6. Conclusiones.

Considerando las variaciones observadas producidas por ambos controladores podemos inferir lo siguiente:

La primera conclusión que se observa es que la incorporación de redes implica cambios que deben tenerse en cuenta para diseñar un controlador en un DCS. Las variaciones observadas se deben en principio a las diferencias en el tiempo de muestreo: la plataforma de simulación posee un tiempo de muestreo de 1 ms, mientras que el scan del controlador llega a 50ms, sin tener en cuenta los retardos producidos por el scan de los gateways y de la red Profibus.

Obviamente las redes generan retardos que deben tenerse en cuenta, sin embargo en este caso los retardos no son lo suficientemente grandes como para que la convergencia de las variables monitoreadas se aleje de los parámetros esperados.

Por otro lado, es necesario que los requerimientos de control sean claramente especificados para lograr ajustar los parámetros del controlador PID, en caso que sea posible hacerlo.

Otros Estudios.

Este trabajo motiva el estudio de los retardos introducidos por las redes de campo, los sensores y los actuadores.

Otro estudio que motiva este trabajo es el análisis del sistema 800xA ante la falla de alguno de sus componentes. Entre estos podemos citar a:

- El CPU
- Las redes de campo
- El Servidor de Conectividad
- El Servidor de Aspectos.
- Las redes de Supervisión.

Anexo I. Modelo de reactor. Termohidráulica y Cinética.

A continuación se describen las ecuaciones que corresponden a la termohidráulica del modelo del reactor utilizado en este trabajo; además se describen las variables de entrada, salida y de estado que corresponden a la plataforma de simulación Matlab-Simulink (2).

Como se mencionó anteriormente en este trabajo se hacen las siguientes simplificaciones con respecto a la termohidráulica (1):

- "El problema se resuelve en forma unidimensional. La variable espacial se toma en sentido de circulación del fluido."
- "Para resolver la zona bifásica de líquido y vapor (núcleo y chimenea) se usa el modelo homogéneo, o sea que se trata a la mezcla como un pseudo-fluido que obedece las ecuaciones de simple fase (no se consideran diferencias de velocidades entre vapor y líquido)."
- "No existe arrastre de burbujas hacia el generador de vapor y se considera que el líquido que ingresa al generador de vapor se encuentra en saturación."
- "En todo el circuito se considera que la presión es la misma e igual a la del domo, y no se tienen en cuenta diferencias de presión por columna de agua."
- "Solo se considera intercambio de calor en la zona del núcleo y del generador de vapor (que son variables), y una pérdida de calor constante en el domo. El intercambio de calor en el núcleo es gobernada por la diferencia de temperaturas medias del refrigerante y del secundario (condición de contorno), considerando un perfil de temperaturas exponencial del lado primario y constante del lado secundario."
- "No se consideran los caudales de by-pass del generador de vapor, ni del núcleo."

Generador de Vapor

Balance de energía:

$$\frac{\partial T_b}{\partial t} = \frac{1}{A_b Longb\ (a + 2b\ T_b)} \left(w_{bi} T_{bi} - w_{bo} T_{bo} + \frac{K_b}{C_b} A_{bf} (T_s - T_b) \right)$$

Balance de masa y energía:

$$\boldsymbol{w_{bo}} = \underbrace{\frac{\frac{K_b}{C_b}\, A_{bf}\, (T_s - T_b)}{T_{bo} - \frac{a + 2bT_b}{b}}}_{w_{bo_{ind}}} + w_{bi} \underbrace{\frac{T_{bi} - \frac{a + 2bT_b}{b}}{T_{bo} - \frac{a + 2bT_b}{b}}}_{W_{bo_{dep}}}$$

Otras relaciones:

$\boldsymbol{T_{bo}} = T_s + (T_{bi} - T_s) \exp\left(-\frac{(T_{bi} - T_b)}{(T_b - T_s)}\right)$

$\boldsymbol{T_{bi}} = f(Psat), Temperatura\ de\ sat\ en\ domo$

$\boldsymbol{Denb} = a + b\, T_b$

Entradas:

Ts: Temperatura del secundario. En este modelo está considerada como una fuente de perturbación al sistema.

Tbi: Temperatura de entrada al generador de vapor. Esta proviene del equilibrio del domo.

wbi: Caudal de entrada del generador de vapor. Esta proviene del cálculo del balance de momento integral.

Las salidas se obtienen en base a implementar las ecuaciones formuladas previamente.

Salidas:

Tb: Temperatura media del primario del GV. La salida se utiliza solo para medición.

Tbo: Temperatura de salida del primario del GV. Esta salida se utiliza como entrada en el downcommer (DC).

wbo: Caudal de salida del primario del GV. Esta también está involucrada en la dinámica del sistema ya que es otra de las entradas al DC.

Denb: Densidad media del primario del GV. Esta se utiliza en el cálculo de ΔP_{boy} en el balance de momentos

wbo_I: Término independiente de la ecuación algebraica del caudal que se utiliza en el cálculo wbi.

Wbo_d: Término dependiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Variables de estado:

Tb: Temperatura media del GV. En *estado estacionario* (SS) tenemos Tb0=293.25 ºC. Este valor se despeja de la ecuación 3.1.1.3 a partir de Th0, Tc0 y Ts0.

Downcommer

El Downcommer se divide en 10 nodos, las ecuaciones que describen la dinámica de cada nodo son:

Balance de energía:

$$\frac{\partial \boldsymbol{T_{dci}}}{\partial \boldsymbol{t}} = \frac{1}{A_{dc}\frac{Longdc}{10}(a+2b\,T_{dci})}(w_{dci-1}T_{dci-1} - w_{dci}T_{dci}) \qquad i = 1,2,\dots,10$$

Balance de masa y energía:

$$\boldsymbol{w_{dci}} = w_{dci-1}\underbrace{\frac{a+2b\,T_{dci}-b\,T_{dci-1}}{a+b\,T_{dci}}}_{W_{dci\_dep}}$$

Otras relaciones:

$\overline{T_{dci}} = T_{dci} = T_{dci\_o}$

$\boldsymbol{w_{dc0} = w_{bo}}$

$\boldsymbol{T_{dc0} = T_{b0}}$

$\boldsymbol{Dendc} = a + \frac{b}{10}\sum_{i=1}^{10} T_{dci}$

$\widehat{W}_{dc\_dep} = W_{dc1_{dep}}.W_{dc2_{dep}} \dots\dots W_{dc10_{dep}}$

$$\widetilde{\boldsymbol{W}}_{\boldsymbol{dc\_dep}} = 1 + 2\,W_{dc1_{dep}}.W_{dc2_{dep}} + 2\,W_{dc1_{dep}}.W_{dc2_{dep}}.W_{dc3_{dep}} + 2W_{dc1_{dep}}.W_{dc2_{dep}} \dots..W_{dc9_{dep}} + W_{dc1_{dep}}.W_{dc2_{dep}} \dots..W_{dc10_{dep}}$$

Entradas:

Tbo: Temperatura de salida del GV.

wbo: Caudal de salida del GV.

Salidas:

Tdc10: Temperatura de salida del último nodo del Downcommer.

wdc10: Caudal de salida del Downcommer. Este caudal y la temperatura anterior serán las entradas al plenum.

Ws_dc = $\widehat{W}_{dc\_dep}$ Término dependiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Wr_dc = $\widetilde{W}_{dc\_dep}$ Término dependiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Dendc: Densidad media en el Downcommer. Esta se utiliza en el cálculo de ΔP_{boy} en el balance de momentos.

Variables de estado:

Tdc_i : Temperatura media del nodo i del Downcommer. En SS tenemos Tdci0=Tdc0=Tc0=284.53 ºC. En SS la temperatura media de todos los nodos será la misma e igual a la temperatura de la rama fría. Tc0 se obtiene del balance en SS del primario.

Plenum

Balance de energía:

$$\frac{\boldsymbol{\partial T_p}}{\boldsymbol{\partial t}} = \frac{1}{A_p\, Longp\,(a + 2b\,T_p)}\left(w_{dc10}T_{dc10} - w_c T_p\right)$$

Balance de masa y energía:

$$\boldsymbol{w_c} = w_{dc10}\,\underbrace{\frac{a + 2b\,T_p - b\,T_{dc10}}{a + b\,T_p}}_{W_{c\_dep}}$$

Otras relaciones:

$$\overline{T_p} = T_p = T_{p\_o}$$

$$\boldsymbol{Denp} = a + b\,T_p$$

Entradas:

Tdc10: Temperatura de salida del último nodo del Downcommer.

wdc10: Caudal de salida último nodo del Downcommer.

Salidas:

Tp: Temperatura de salida del plenum, esta es la temperatura de entrada al núcleo y se utiliza también para calcular la entalpía de entrada.

wc: Caudal de salida al plenum, que en definitiva es el caudal de entrada al núcleo.

Wc_dep: Término dependiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Dendp: Densidad media en plenum. Esta se utiliza en el cálculo de ΔP_{boy} (fuerza boyante) en el balance de momentos.

Variables de estado:

Tp: Temperatura media del plenum. En SS tenemos Tp0=Tc0=284.53 ºC. Tc0 se obtiene del balance en SS del primario.

Núcleo (modelo termohidráulico)

En el modelo del núcleo se consideró un solo nodo.

Balance de energía:

Refrigerante:

$$\frac{\partial h_m}{\partial t} = \frac{1}{A_n\, Longn\, Denn}\left(K_n\left(T_f - T_n\right) - \left(w_c + w_h\right)\left(h_m - h_c\right)\right)$$

Combustible:

$$\frac{\partial T_f}{\partial t} = \frac{1}{mfuel}\left(\frac{Pot}{1000} - K_n\left(T_f - T_n\right)\right)$$

Balance de masa y energía:

$$\boldsymbol{w_h} = \underbrace{\frac{-K_n\left(T_f - T_n\right)}{(Denn\ B - (hm - hc))}}_{w_{h\_ind}} + w_c \underbrace{\frac{(Denn\ B\ + (hm - hc))}{(Denn\ B\ - (hm - hc))}}_{W_{h_{dep}}}$$

Otras relaciones:

$\boldsymbol{hc} = f(T_c, P)$

$\boldsymbol{hm} = \frac{hh + hc}{2}$

$\boldsymbol{hj} = hh - hc$

$\boldsymbol{Tn} = A\ \Delta hn + Tn_0$

$\boldsymbol{Denn} = B\ \Delta hn + Denn_0$

Entradas:

wc: Temperatura de salida del GV.

Pot: Caudal de salida del GV.

Hc: Entalpía de entrada el núcleo. Se obtiene a partir de la temperatura del plenum con la relación para líquido subsaturado de (5).

Salidas:

Tf: Temperatura del combustible. Se utiliza solo para información.

hm: Entalpía media en el núcleo. Se utiliza para información.

hj: Salto de entalpía en el núcleo. Se utiliza para información.

hh: Entalpía de salida en el núcleo.

wh: Caudal de salida del núcleo. Este junto con la entalpía de salida son entradas a la chimenea.

wh_I: Término independiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Wh_d: Término dependiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Dendn: Densidad media en el núcleo. Esta se utiliza en el cálculo de ΔP_{boy} (fuerza boyante) en el balance de momentos. Se calcula a partir de B que aproxima la relación entre entalpia y densidad.

Variables de estado:

Tf: Temperatura media del combustible. En SS tenemos Tf0=700 ºC. Este valor se saca del documento, según las condiciones de SS planteadas.

hm: Entalpía media en el núcleo. En SS se despeja del balance de energía suponiendo un perfil lineal de entalpía en el núcleo y se obtiene:

$$\boldsymbol{hm0} = hh0 - \frac{hj0}{2} = hh0 - \frac{Pot0}{2\ 1000\ w0} = 1378.82\ KJ/kg$$

Chimenea

La chimenea se divide en 5 nodos fijos. Las ecuaciones de cada nodo se muestran a continuación:

Balance de energía:

$$\boldsymbol{\frac{\partial h_{chi}}{\partial t}} = \frac{1}{A_{ch}\ \frac{Longch}{5}\ Dench_i}\left(\ w_{chi-1}\ (h_{chi-1} - h_{chi})\right)$$

Balance de masa y energía:

$$\boldsymbol{w_{chi}} = \ w_{chi-1}\ \left[1 + \frac{Au2\ (h_{chi-1} - h_{chi})}{(1 + \chi_{chi}\ Au2)(hg - hl)}\right]$$

Otras relaciones:

$$\boldsymbol{Au2} = \frac{\rho_l}{\rho_g} - 1$$

$$\boldsymbol{\chi_{chi}} = \frac{h_{chi} - h_l}{h_g - h_l}$$

$$\boldsymbol{Denchi} = \frac{\rho_l}{1 + Au2\ \chi_{chi}}$$

$$\boldsymbol{\rho_l}, \boldsymbol{\rho_g}, \boldsymbol{h_g}, \boldsymbol{h_l} = f(Psat)\ propiedades\ de\ saturación, sacadas\ de \quad (5)$$

$$\overline{\boldsymbol{h_{chi}}} = h_{chi} = h_{chi\_o}$$

$$\boldsymbol{w_{ch0}} = w_c$$

$$\boldsymbol{h_{ch0}} = h_h$$

$$\boldsymbol{Dench} = \frac{1}{5}\sum_{i=1}^{5} Den_{chi}$$

$$\widetilde{\boldsymbol{W}}_{\boldsymbol{ch\_dep}} = 1 + 2\,W_{ch1_{dep}}.W_{ch2_{dep}} + 2\,W_{ch1_{dep}}.W_{ch2_{dep}}.W_{ch3_{dep}} + 2W_{ch1_{dep}}.W_{ch2_{dep}} \ldots..W_{ch9_{dep}} + W_{ch1_{dep}}.W_{ch2_{dep}} \ldots..W_{ch10_{dep}}$$

Entradas:

hh: Entalpía de salida del núcleo. Se calcula sumando el salto de entalpía en el núcleo a la entalpía de la rama fría. Esta última se obtiene a partir de la temperatura del plenum y la relación con la entalpía para líquido subsaturado.

wh: Caudal de salida del núcleo.

P: Presión en el RPV. La fija el domo y se supone constante en todo el sistema. Se utiliza para calcular las propiedades de saturación.

Salidas:

hch5: Entalpía de salida del último nodo de la chimenea.

wch5: Caudal de salida de la chimenea. Junto con hch5 se utilizan como entradas en el domo.

Wch_r: Término dependiente de la ecuación algebraica del caudal que se utiliza en el cálculo de wbi.

Dench: Densidad media en la chimenea. Esta se utiliza en el cálculo de ΔP_{boy} en el balance de momentos.

Dench5: Densidad del último nodo. Se utiliza para el AP de fricción en el balance de momentos.

Xch5: titulo de salida del último nodo. Informativo.

Variables de estado:

hchi : Entalpía media del nodo i de la chimenea. En SS tenemos hchi0=hh0=1500.77 KJ/kg. Hh0 se calcula a partir del balance general del SS.

Domo

Este modelo supone equilibrio termodinámico en el domo, las ecuaciones son:

Balance de energía:

$$\frac{\partial P}{\partial t} = \frac{w_{ch5}\, h_{ch5} \; - \; w_{bi}\, h_l - f_2 \, \frac{\partial Md}{\partial t} \; - \; qe/1000}{V_d \, \frac{\partial f1}{\partial P} + M_d \, \frac{\partial f2}{\partial P} - Vd}$$

Balance de masa:

$$\frac{\partial Md}{\partial t} \; = \; w_{ch5} - w_{bi}$$

Otras relaciones:

$$f1 = \frac{h_l - h_g}{\frac{1}{\rho_l} - \frac{1}{\rho_g}}$$

$$f2 = \frac{1}{\frac{1}{\rho_l} - \frac{1}{\rho_g}} \left(\frac{h_l}{\rho_l} - \frac{h_g}{\rho_g} \right)$$

$$\frac{\partial f1}{\partial P} = \left(\frac{\partial h_l}{\partial P} - \frac{\partial h_g}{\partial P} \right) \frac{1}{\frac{1}{\rho_l} - \frac{1}{\rho_g}} - \frac{(h_l - h_g) \left(\frac{1}{{\rho_g}^2} \frac{\partial \rho_g}{\partial P} - \frac{1}{{\rho_l}^2} \frac{\partial \rho_l}{\partial P} \right)}{\left(\frac{1}{\rho_l} - \frac{1}{\rho_g} \right)^2}$$

$$\frac{\partial f2}{\partial P} = \left(\frac{1}{\rho_l} \frac{\partial h_g}{\partial P} - \frac{h_g}{{\rho_l}^2} \frac{\partial \rho_l}{\partial P} - \frac{1}{\rho_g} \frac{\partial h_l}{\partial P} + \frac{h_l}{{\rho_g}^2} \frac{\partial \rho_g}{\partial P} \right) \frac{1}{\frac{1}{\rho_l} - \frac{1}{\rho_g}} - \frac{\left(\frac{h_l}{\rho_l} - \frac{h_g}{\rho_g} \right) \left(\frac{1}{{\rho_g}^2} \frac{\partial \rho_g}{\partial P} - \frac{1}{{\rho_l}^2} \frac{\partial \rho_l}{\partial P} \right)}{\left(\frac{1}{\rho_l} - \frac{1}{\rho_g} \right)^2}$$

$$T_d = f(Psat) = Tbi, Temperatura\ de\ sat\ en\ domo.$$

$$\rho_l \, , \rho_g , h_g \, , h_l = f(Psat)\ propiedades\ de\ saturación, sacadas\ de\ (5)$$

Entradas:

hch5: Entalpía de salida de la chimenea.

wch5: Caudal de salida de la chimenea.

wbi: Caudal de entrada al Generador de Vapor. Se supone que entra con X=0.

Salidas:

P: Presión del RPV. Esta es la variable del primario que se desea controlar. Además es entrada para calcular las propiedades de la chimenea y para calcular Tbi.

Md: Masa en el domo. Liquido más vapor. Junto con hch5 se utilizan como entradas en el domo.

Variables de estado:

P: Presion del RPV. En SS tenemos P0=12.25Mpa. Es una condición en SS.

Md: Masa en el domo. Liquido + vapor. En SS =7879,6 kg.

Balance de momento integral

El balance de momento se plantea en forma integral, para todo el circuito en función de un flujo de masa integral G.

$$\frac{\partial G}{\partial t} = \Delta P_{boy} - \Delta P_{fricc}$$

$$G = \sum_{i=1}^{18} \frac{w_{ei} + w_{si}}{2\, A_i}\, Long_i$$

La fuerza boyante se define como:

$$\Delta P_{boy} = g\,(Denb\ Longb + Dendc\ Longdc + Denp\ Longp - Denn\ Longn - Dench\ Longch\ )$$

La pérdida de presión por ficción está definida por las diferencias de presión por diferencias de área a la salida del domo (ΔP_{f_DO}), a la salida del GV (ΔP_{f_GV}), a la salida del downcommer (ΔP_{f_DC}), a la salida del plenum (ΔP_{f_PL}), a la salida del núcleo (ΔP_{f_NU}), a la salida de la chimenea (ΔP_{f_CH}), y por la fricción concentrada a la salida del generador de vapor (ΔP_{f_NI}).

$$\Delta P_{fricc} = \Delta P_{f_{DO}} + \Delta P_{f_{GV}} + \Delta P_{f_{DC}} + \Delta P_{f_{PL}} + \Delta P_{f_{NU}} + \Delta P_{f_{CH}} + \Delta P_{f_{CH}} + \Delta P_{f_{NI}}$$

$$\Delta P_{f\_DO} = 0.42\, \frac{\left(1 - \left(\frac{Ab}{Ad}\right)^2\right) {w_{bi}}^2}{2\, Ab^2 (a + b\, Tbi)} \quad ; \quad \Delta P_{f\_GV} = \frac{\left(1 - \left(\frac{Ab}{Adc}\right)^2\right)^2 {w_{bo}}^2}{2\, Ab^2 (a + b\, Tbo)}$$

$$\Delta P_{f\_DC} = 0.42\, \frac{\left(1 - \left(\frac{Ap}{Adc}\right)^2\right) {w_{dc10}}^2}{2\, Ap^2 (a + b\, Tdc_{10})} \quad ; \quad \Delta P_{f\_PL} = 0.42\, \frac{\left(1 - \left(\frac{An}{Ap}\right)^2\right) {w_c}^2}{2\, An^2 (a + b\, T_p)}$$

$$\Delta \boldsymbol{P}_{\boldsymbol{f\_NU}} = \frac{\left(1 - \left(\frac{An}{Ach}\right)^2\right)^2 {w_h}^2}{2\, Ab^2 Denn} \quad ; \quad \Delta \boldsymbol{P}_{\boldsymbol{f\_CH}} = \frac{\left(1 - \left(\frac{Ach}{Ad}\right)^2\right)^2 {w_{ch5}}^2}{2\, Ach^2 Dench_5}$$

$$\Delta \boldsymbol{P}_{\boldsymbol{f\_NI}} = \frac{K_{fri}\, {w_{bo}}^2}{2\, Ab^2 (a + b\, Tbo)}$$

Entradas:

Denb: Densidad promedio en el Generador de Vapor.

Dendc: Densidad promedio en el Downcommer.

Denn: Densidad promedio en núcleo.

Dench: Densidad promedio en la chimenea.

Denp: Densidad promedio en el plenum. Todas estas densidades se utilizan para el AP_{boy}.

Dench5: Densidad de salida en la chimenea.

wch5: Caudal de salida de la chimenea.

Tbi: Temperatura de entrada al Generador de Vapor

wbi: Caudal de entrada al Generador de Vapor.

Tbo: Temperatura de salida del Generador de Vapor.

wbo: caudal de salida del Generador de Vapor.

Tdc10: Temperatura de salida del Downcommer.

Wdc10: Caudal de salida del Downcommer.

Tp: Temperatura del plenum.

wc: Caudal de entrada al núcleo.

wh: Caudal de salida al núcleo. Todas estas variables se utilizan para calcular AP_{fric} (Pérdida de presión por fricción).

Salidas:

G: Flujo de masa integral. Este sirve luego para despejar el caudal de entrada al Generador de Vapor.

Variables de estado:

G: Flujo integral. En SS tenemos G_0:

$$\boldsymbol{G_0} = w_0 \left(\frac{Longb}{Ab} + \frac{Longdc}{Adc} + \frac{Longp}{Ap} + \frac{Longn}{An} + \frac{Longch}{Ach}\right) \approx 3240.5 \ \frac{Kg}{s\ m}$$

Cinética del reactor.

Debe recordarse que se utilizó un modelo de cinética puntual a un solo grupo de neutrones retardados sin linealizar. Para definir la realimentación de reactividad se consideró una variación lineal por temperatura del moderador, por temperatura del combustible y por densidad del moderador (2).

Las ecuaciones de la cinética del reactor se describen a continuación.

$$\frac{\boldsymbol{\partial Pot}}{\boldsymbol{\partial t}} = \frac{\delta k - \beta}{\Lambda}\ Pot + \ \lambda\ C$$

$$\frac{\boldsymbol{\partial C}}{\boldsymbol{\partial t}} = \frac{\beta}{\Lambda}\ Pot - \ \lambda\ C$$

$$\boldsymbol{\delta k} = \delta k_{barras} + \delta k_{den} + \delta k_{mod} + \delta k_{fuel}$$

$$\boldsymbol{\delta k_{den}} = \alpha_d\ \Delta\mathrm{Denn}$$

$$\boldsymbol{\delta k_f} = \alpha_f \Delta T_f$$

$$\boldsymbol{\delta k_m} = \alpha_m\ \Delta T_m$$

Parámetros:

Los parámetros neutrónicos que se consideran son:

$\boldsymbol{\beta}$ =7.0e-3 partes enteras. Fracción de neutrones retardados.

$\boldsymbol{\Lambda} = 1\mathrm{e} - 5\ \mathrm{seg}^{-1}$. Tiempo de reproducción de neutrones.

$\boldsymbol{\lambda} = 1.4\ seg^{-1}$. Tiempo de decaimiento de los precursores (un grupo).

$\boldsymbol{\alpha_d} = 24\ e - 5\ \frac{partes\ enteras}{(kg/m^3)}$

$\boldsymbol{\alpha_f} = \ -2.3\ e - 5\ \frac{partes\ enteras}{℃}$

$\boldsymbol{\alpha_m} = 2.3\ e - 5\ \frac{partes\ enteras}{℃}$.

Todos parámetros anteriores fueron sacados de (1) y se tomaron los valores medios.

Entradas:

$\boldsymbol{\delta k}$**:** Reactividad de entrada. Este incluye el movimiento de barras y las realimentaciones por temperatura y densidad.

Salidas:

Pot: Potencia neutrónica del núcleo. Esta se usa como entrada al modelo termohidráulico del núcleo, en KW. Para información se expresa en MW.

Periodo: Periodo logarítmico del reactor. Esta variable esta con fines informativos.

Variables de estado:

Pot: Potencia neutrónica. En SS en plena potencia el valor previsto es Pot0=100Mw.

C: Densidad de precursores. $\boldsymbol{C0} = \frac{\beta}{\Lambda\,\lambda} Pot0 \approx 5e10$.

Anexo II. Configuración del sistema- Gestión de la Configuración.

Introducción.

Este anexo describe la instalación y configuración del sistema. Se muestra la instalación del sistema 800xA, la instalación de la estación de simulación, la conexión de la periferia descentralizada con los módulos de entrada / salida del controlador 1.

Instalación sistema 800xA.

Preparación.

Las etapas para preparar el sistema son:

- Determinar el alcance y el objetivo del sistema.
- Planificar la topología de la red.
- Obtención de todos los instaladores.
- Confirmar que los requisitos se Hardware se cumplan.
- Instalación de los Sistemas Operativos.
- Instalación de las aplicaciones necesarias.

Determinar el alcance y el objetivo del sistema.

El objetivo del sistema como se menciono en el punto 1, es servir de base de pruebas para analizar las respuestas del modelo simulado a lazo cerrado, contra las respuestas del modelo simulado cuando el lazo de control es cerrado por un sistema DCS.

Se plantea instalar solo los componentes importantes del sistema para servir de base de pruebas para futuros ensayos.

Planificar la topología de la red.

La planificación de la topología de la red queda explícita a partir de la **¡Error! No se encuentra el origen de la referencia.14**.

Obtención de todos los instaladores.

Los instaladores necesarios para la instalación del sistema total son:

- Windows XP SP2 vs English.
- Industrial^{IT} 800xA System Version 5.0 with SP2.
- Framework .Net 1.1
- Visual Basic 6.0 vs English.
- Service Pack 6 para Visual Basic 6.0 vs English.

- Microsoft Office 2007 vs English.
- Drivers de cada PC.

Requisitos se Hardware.

Con respecto al hardware necesario para la instalación del sistema se sigue a (9). En base a esos requerimientos se detalla en la **Tabla-A i** el hardware de cada estación en el sistema 800xA. Si bien (9) recomienda la utilización de *Servidores y Workstation* de marcas reconocidas en este trabajo se emplearon *PCs* estándares, conocidas en la jerga como *PCs Clones.*

Estaciones	Procesador	Memoria RAM	Disco Rígido	Lectora DVD	Interfaces Ethernet
Servidor de Aspectos Primario	Intel Core 2 Duo 2.66GHz	4GB RAM	160 GB 160 GB	Sí	2 NIC
Servidor de Aspectos Secundario	Intel Core2 Quad 2.4GHz	3GB RAM	250 GB	Sí	2 NIC
Servidores de Conectividad	Intel Core 2 Duo 2.66GHz	4GB RAM	160 GB 160 GB	Sí	4 NIC
Estación de Ingeniería y Operación.	AMD LE-1250 2.21GHz	2GB RAM	320 GB	Sí	3 NIC

Tabla-A i. Hardware para las distintas estaciones.

Instalación de Sistemas Operativos y Aplicaciones necesarias.

Las tareas iniciales de instalación en cada nodo fueron las siguientes:

- Instalación del Sistema Operativo.
- Instalación de Drivers.
- Instalación de Aplicaciones en caso que el nodo lo requiera.

Como se definió anteriormente se utilizaron 5 PC's para la instalación del sistema. El software instalado en las distintas estaciones según lo definido en (8) se detalla en la **Tabla-A ii**.

Estaciones	Sistemas Operativos	Aplicaciones
Servidores de Aspectos	Windows 2003 Server R2, Enterprise Edition, Service Pack 2. Versión en Ingles.	• System Installer(*).
Servidores de Conectividad	Windows XP, Service Pack 2, Versión en Inglés.	• System Installer.
Estación de Ingeniería y Operación.	Windows XP, Service Pack 2, Versión en Inglés.	• System Installer 800xA. • Visual Basic 6.0 – Service Pack 6. Versión en Inglés. • Microsoft Office 2007

Tabla-A ii. Software básico instalado en las distintas estaciones, para iniciar la instalación.

(*) La aplicación System Installer debe instalarse para proseguir con la misma. Se encuentra en DVD 1: 800xA System Installation.

Una vez instalado cada nodo se configuraron los siguientes parámetros:

- Nombre del Grupo de Trabajo: WORKGROUPABB
- Se asigna un nombre a cada Conexión de Red: Network Control Primary, Network Control Secondary, Network Supervision Primary, Network Supervision Secondary, según corresponda.
- Se configura el teclado desde: *Control Panel* > *Regional and Language Options* > *Pestaña Language* **>** Presione el botón *Details* y seleccione *English (United States)***.**
- También desde *Regional and Language Options* se configura el símbolo decimal como punto (.) haciendo clic en *Customize* y seleccionando la pestaña *Numbers* y el formato de la fecha corta a MM/dd/yyyy en la pestaña *Date***.**

Planificación del Sistema.

Siguiendo la recomendación de (8) antes de comenzar la planificación del sistema se definen los siguientes parámetros del sistema partir de una tabla configuración. Cabe aclarar que no se toman los parámetros enunciados en (8), sino solo aquellos que son relevantes para el sistema de prueba de este trabajo. Estos parámetros se pueden observar en **Tabla-A iii**.

Nodo	Parámetro	Valor
Servidor de Aspectos Primario	Dirección IP Primaria	172.16.4.11
	Dirección IP Secundaria	172.17.4.11

Nodo	Parámetro	Valor
	Mascara de Subred	255.255.252.0
	Nombre del Nodo	PriAS1
Servidor de Aspectos Secundario	Dirección IP Primaria	172.16.4.12
	Dirección IP Secundaria	172.17.4.12
	Mascara de Subred	255.255.252.0
	Nombre del Nodo	SecAS1
Servidor de Conectividad Primario	Dirección IP Primaria – Red Control A	172.16.0.21
	Dirección IP Secundaria – Red Control B	172.17.0.21
	Dirección IP Primaria – Red Supervisión A	172.16.4.21
	Dirección IP Secundaria – Red Supervisión B	172.17.4.21
	Mascara de Subred	255.255.252.0
	Nombre del Nodo	PriAC800MCS1
Servidor de Conectividad Primario	Dirección IP Primaria – Red Control A	172.16.0.22
	Dirección IP Secundaria – Red Control B	172.17.0.22
	Dirección IP Primaria – Red Supervisión A	172.16.4.22
	Dirección IP Secundaria – Red Supervisión B	172.17.4.22
	Mascara de Subred	255.255.252.0
	Nombre del Nodo	SecAC800MCS1
Estación de Ingeniería y	Dirección IP Primaria	172.16.4.71

Nodo	Parámetro	Valor
Operación	Dirección IP Secundaria	172.17.4.71
	Mascara de Subred	255.255.252.0
	Nombre del Nodo	EngClient1
AC 800M Controlador 1	Dirección IP Primaria	172.16.0.1
	Dirección IP Secundaria	172.17.0.1
	Dirección IP del puerto PPP	192.168.255.254
	Módulos de Comunicación	CI854 (Master) – CI840 (Slave)
	Módulos de Entrada / Salida	AI810, AO810v2
AC 800M Controlador 2	Dirección IP Primaria	172.16.0.2
	Dirección IP Secundaria	172.17.0.2
	Dirección IP del puerto PPP	192.168.0.1
	Módulos de Comunicación	
	Módulos de Entrada / Salida	
Servidor de Dominio / Grupo de Trabajo	Nombre Grupo de trabajo	WORKGROUPABB
Switch 3Com Baseline 2226	Dirección IP	192.168.14.45

Tabla-A iii. Parámetros para la Configuración e Instalación del sistema 800xA.

System Planner Tool

La herramienta *System Planner Tool* nos permite armar una configuración del sistema, en base a los parámetros relevados, la cual facilita la instalación.

Esta herramienta puede ejecutarse en cualquier nodo del sistema, basta logearse en el sistema operativo con privilegios de *Administrador*.

Para ejecutar *System Planner* ir a *Automated Installation > System Setup Tools > System Planner*.

A continuación se enumeran los pasos que se siguieron:

a) Plan a new System.

b) Se especifica el numero de Workplaces.
Workplaces Operator: 0 (cero)
Workplaces Engineering: 1 (uno). Debe destacarse que un *Workplace Engineering* incluye un Workplace Operator.

c) Se define el número de servidores para clientes remotos.
Number of Servers for Remote Clients: 0 (cero).

d) Se define el número de servidores de conectividad y su redundancia.
AC 800M Connectivity Servers: 1 (uno). Solo se establece conectividad con los controladores AC 800M.
Se define su redundante marcando la opción "U*se Redundant Servers".*

e) Opciones de los Servidores de Conectividad.
Desmarcar la opción "*use Base Software for Soft Control".*

f) Opciones para la administración de dispositivos.
Se marca la opción "*Device Management Profibus and HART".*
No se marca la opción "*Device Management FOUNDATION Fieldbus*".

g) Opciones para el monitoreo.
Se marca la opción "*PC, Network and Software Monitoring runs on Connectivity Server PriAC800MCS1.*

h) Opciones de Administración de Producción.
El sistema prescinde del *Batch Server.* No se marca la opción "*Use Batch Server*".

i) Opciones de la Base de Datos Histórica.
El sistema prescinde del Information Management. No se marca la opción "*Use Information Management*".

j) **Opciones para la optimización de activos.**
El sistema prescinde de "*Asset Optimization Server*" y de "*Use sms and Email Menssaging*". No se marcan ninguna de estas opciones.

k) Opciones de las Estaciones de Ingeniería.
Se elige la opción *"Professional Engineering Tools"* y se tilde la opción "*use Process Engineering Tools*".

l) Opciones sobre la Redundancia del Servidor de Aspectos.
Se selecciona la opción "*Redundant 1 out of 2".*

m) **Opciones del Aspect Server.**
No se marca ninguna opción dentro del marco *"Primary Aspect Server"* ni las opciones del marco "*Secondary Aspect Server*". Esto significa que ningún workplaces se ejecutará en los servidores de aspectos.

n) **Opciones para combinar el Servidor de Conectividad Primario con otros servidores.**
No se marca ninguna opción, solo "*Add a Redundant Server*" (porque hay redundancia). Esto implica que ningún otro servidor se ejecutará conjuntamente con el servidor de conectividad primario.

o) **Opciones para combinar el Servidor de Conectividad Secundario con otros servidores.**
No se marca ninguna opción. Esto implica que ningún otro servidor se ejecutará conjuntamente con el servidor de conectividad secundario.

p) **Opciones de la Red.**
Se elige la opción "*Use a workgroup*".

q) **Opciones de Integración para un multisistema.**
No se marca la opción "*This system is part if a multisystem Integration"* ya que el sistema no es parte de un sistema mayor.

r) **El sistema muestra un resumen con los nodos definidos.**
EngClient
PriAC800MCS
SecAC800MCS
PriAS
SecAS

s) **Se termina de definir el tamaño y las opciones del sistema.**
Workgroup: WORKGROUPABB
System service Account: 800xAService
Marcar la opción *"Allow non secure password".*
Password: admin800xAService
System Name: MySystem

t) **Se definen los usuarios y las password del sistema**

User Name	Full Name	Password	IndustrialITuser	IndustrialITAdmin
800xAInstaller	800xAInstallA	Sv50IIT1useracc		
Apeng	ApplicationEn	Sv50IIT2useracc		
Syseng	SystemEngin	Sv50IIT3useracc		
Operator	Operator	Sv50IIT4useracc		
Operator2	Operator2	Sv50IIT5useracc		

Tabla-A iv. Usuarios del sistema – System Planner

Se marca la opción "*User must change password at first logon*".
Password are not eneryted by this application. The system is not secure until all password have been changed by the users at logon.

u) Nombres de los Nodos del Sistema.

Tipo de Nodo	Nombre
EngClient	EngClient1
PriAC800MCS	PriAC800MCS1
PriAS	PriAS1
SecAC800MCS	SecAC800MCS1
SecAS	SecAS1

Tabla-A v. Nodos del Sistema. System Planner.

v) Direcciones IP de los Nodos.
Se definen las IP de los nodos de acuerdo a la **Tabla-A iii**.

w) Paquete de la Configuración.
Se marca la opción "*Common destination for all Setup Packages (all nodes)*" para crear un solo paquete de configuración para la instalación.

Una vez hecho esto la herramienta *System Planner* creará una carpeta con la configuración definida, la cual nos facilitará la instalación en cada nodo.

Instalación del Sistema.

La instalación se realiza en cada nodo utilizando el paquete de configuración generado anteriormente.

El orden de instalación en este momento es indistinto, sin embargo durante la configuración se debe empezar por Servidor de Aspectos Primario.

Instalación común a cada nodo.

a) Se copia en el nodo el paquete de configuración creado en la etapa anterior.

b) Se inserta el *DVD 1: 800xA System Installation.*

c) Se ejecuta "*Start Install & Setup of this node*", desde Start > All Programs > ABB Industrial IT 800xA > System > System Installer > Start Install & Setup of this node.

d) Aparecerá un cuadro de diálogo explicando que es recomendable la creación de una cuenta para la instalación del sistema (800xA Installation); si la cuenta ya fue creada por el System Installer, es conveniente reiniciar el sistema y logearse con esa cuenta. Caso

contrario se puede continuar con la instalación desde la cuenta actual, para ello es recomendable hacer clic en "*Yes.*"

e) **Installation Type.** Este cuadro de dialogo permite elegir entre los distintos tipos de instalaciones. Se elige la opción "*Select Setup Package manually"* y se indica la dirección del paquete de configuración creado anteriormente, para el nodo que queremos instalar.

La opción "*Select from Predefined Node Type"* es utilizada para instalar un tipo de nodo predefinido.
La opción "*800xA Core System"* instala la Base para el sistema 800xA. A esta opción también puede llegarse eligiendo la opción "*800xA Base"* del combo desplegable de "*Select from Predefined Node Type".*

f) **Configuración de Windows.** Esta herramienta realiza la configuración de cuatro componentes.
Network Adapters. La configuración de las tarjetas de red se basa en el archivo netset.xml, el cual se encuentra en el paquete de configuración.
Windows components and services. La herramienta configura los componentes y servicios necesarios para que se ejecute el sistema en el nodo.
Join Domain. La herramienta verifica si en el nodo está configurado con el dominio o con el workgroup definido en el paquete de configuración.
Windows users and groups. Aquí la herramienta crea las cuentas de usuarios y le asigna los permisos a los mismos.

g) **Verificación del sistema base (Instalación de Software de terceras partes).** El sistema verifica que se cumpla con los requisitos de Hardware y Software. Con respecto al software, la herramienta viene provista con instaladores de ciertos componentes que si no están instalados dan la posibilidad de instalarlos.
Hay requisitos mandatorios, y si no se cumplen la instalación no puede continuar, otros requisitos son recomendaciones y pueden pasarse por alto.

h) El sistema indica un mensaje indicando que es recomendable descargar de la página de *Microsoft* una actualización de seguridad. En este caso no la usamos. *Clic en No.*

i) Aquí el sistema muestra los componentes que se instalaran de acuerdo al nodo y al hacer clic comienza la instalación del nodo.

Configuración del Sistema.

Una vez instalados los nodos, se prosigue con la configuración de cada uno de estos. La configuración requiere que todos los nodos estén encendidos y conectados a las subredes.

Configuración común a cada nodo.

Las siguientes etapas son comunes a todos los nodos.

a) Create system backup: System Software User Settings

b) Clic en el botón "*Configure User Settings"*

c) Ir a Windows\System32\drivers\etc\hosts y agregar información sobre: Servidor de Aspectos Primario, Servidor de Aspectos Secundario, Servidor de Conectividad Primario, Servidor de Conectividad Secundario.

172.16.4.71 EngClient1

172.16.4.11 PriAS1

172.16.4.12 SecAS1

172.16.4.21 PriAc800MCS1

172.16.4.22 SecAc800MCS1

d) Administrative Tools > Local Security Policy. Esta opción solo se aplica a los nodos con Windows XP.

- Local Security Settings
- Local Policies > Security Options
- Network access. Sharing and security model for local accounts.
- Classic-local users authenticate as themselves

e) Configure redundant network in a PC

- Seleccionar *Network Connections* del panel de control
- Menu "*Advanced > Advanced Settings…"*
- *Adapters and Bindings*: Primary Supervision Network, Secondary Sepervision Network, en el caso de los servidores de conectividad también se configura Primary Control Network, Secondary Control Network.

f) Configure NetBIOS for Network Connections

Se habilita NetBIOS sobre TCP/IP para la conexión primaria e inhabilitarlo en la conexión secundaria.

g) Configurar la IP de la segunda conexión.

h) Configure daylight saving and power settings.

i) Clic en el botón "*Configure Power Node".*

j) Inhabilitar *Windows Time Service for 800xA*, para todos los nodos excepto para el Servidor de Conectividad.

Configuración del Servidor de Aspectos Primario.

Las tareas de configuración que se hicieron en el Servidor de Aspectos Primario son las siguientes:

k) Creación de un nuevo sistema.

Clic en el botón *Create System*.

"*Si se produce un error, posiblemente sea porque el usuario con el que se inicio la sesión de Windows no tiene los privilegios necesarios, es conveniente iniciar una sesión con el usuario 800xAService*".

l) Conexión con el Servidor de Conectividad Primario.

Se establece la conexión con el servidor de conectividad primario, eligiendo de una lista desplegable a PriAC800MCS1.

m) Extensiones Adicionales al Sistema.

Se agregan las extensiones adicionales al sistema. Se agrego solo la extensión *"Process Engineering Tool Integration"*

n) Conexión con nodos clientes.

Se conectan nodos clientes. En nuestro caso "*EngClient1".*

o) Servidor de Aspectos Secundario

Se conecta con el *Servidor de Aspectos Secundario* "*SecAS1*".

p) Servidor de Conectividad Secundario

Se conecta con el *Servidor de Conectividad Secundario* (SecAC800MCS1).

q) Usuarios del Sistema y Usuarios de Windows

Se obvia la asociación de usuarios de sistema a grupos de windows ya que solo se dispondrá de un solo Operator Workplaces, y no se utiliza un *Domain Controller*.

r) Configuración de *Time Service*

En el Plant Explorer en *Service Structure > Service > Time, service*

Aspecto: *Service definition > Special Configuration*

- Marcar *"Server Running".*
- Marcar *"Clients allowed to set time".*
- Desmarcar *"Enabled external clock master function".*

s) Servidor de Tiempo

Se agrega como servidor de tiempo al Servidor de Conectividad Primario.

En el Plant Explorer en *Service Structure > Service > Time, service > Basic, Service Group.*

Se agrega:

- ***Time_ServerPriCS1:*** *Current Service y*
- ***Time_ServerSecCS1:*** *Current Stanby*

t) Configuración de Time Client.

En el Plant Explorer en *Node Administration Structure > Node Administration > All nodes, node Group*

Seleccionar un nodo *>* ***Aspecto****: Time Server Client Configuration*

- Marcar *Allowed to set time* (Solo si es deseable)
- Marcar *Time Sync running (Siempre)*
- *Deviation Limit: 1000 msec*

u) Configuración Control Structure.

En el Plant Explorer en *Control Structure > Root Domain >* Clic derecho *"New Object" > SoftPoint Basic Object Types > SoftPoint Generic Control Network.*

En el *Aspecto:* *Generic Control Network Configuration*

Pestaña: Configure

Server Settings*:*

Clic en Botón "Configure" Seleccionar PriAC800MCS1

AlarmandEventSettings:

Clic en botón Configure Seleccionar PriAC800MCS1

"Esta configuración es la representación visual del servidor de conectividad"

v) **Configuración Office 2007***.*

Este paso se obvia ya que no se instaló en el Office en el Servidor de Aspectos.

w) **Configuración DCOM.**

Configuration for HART Multiplexer connect on client node. Esta opción se utiliza cuando se utiliza un nodo servidor para conectar a un multiplexor de HART. En este caso no es necesario.

x) Device Library Wizard settings.

Esta opción no se aplica en el Servidor de Aspectos Primario, ya que esta opción permite conectar con el servidor de librerías el cual generalmente es el Servidor de Aspectos.

y) Install HART Device Types

Esta opción no se aplica en este caso ya que no se disponen de dispositivos HART.

z) Install Profibus Device Types

Se instalaron dos componentes Profibus que corresponden a transmisores de temperatura *Profibus PA*:

ABB_TFx12_V1_2_PA ***y*** *ABB_TFx12_12MB_PNOID_04C4_V1_0_PA.*

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Extract Device Types
- Clic en Next
- Extract device Types via Manual Selection.
- Clic en Next
- Clic en Browse

  En el DVD 2 se busca los dispositivos deseados en la carpeta *Device Library > Profibus.*
- Clic en el botón Next.
- Clic en el botón Finish.

aa) **Configuración del Firewall de Windows***.*

Se utiliza la configuración automática del sistema haciendo clic en *"Configure Firewall"*

Configuración del Servidor de Conectividad Primario.

Las tareas de configuración que se hicieron en el Servidor de Conectividad Primario son las siguientes:

k) **Configuración básica para el monitoreo de la red.**

- Start ABB Industrial IT 800xA > Asset Optimization > PC, Network & Software Monitoring > Basic Computer Monitoring Configuration Tool .
- Start

l) **Extensiones Adicionales al Sistema.**

Se agregan las extensiones adicionales al sistema. Se agrego solo la extensión *"Process Engineering Tool Integration"*

m) ***Configuración Control Structure.***

En el Plant Explorer en *Control Structure > IT Server, IT OPC Server Network.*

En el *Aspecto:* *OPC Data Source Definition* > Clic derecho "*Config View"* > Clic Botón *New* > Clic Botón *Add.*

Se selecciona el OPC Server (PriAC800MCS1).

"Esto indica que en este nodo se ejecutara el OPC Server"

n) **Configuración del OPC data Access.**

- En el Plant Explorer en *Control Structure > Root Domain > Clic derecho "New Object".*

- Se selecciona dentro de *Object Type*: en la pestaña *Common > Control System > AC800M/C Connect > Control Types > Control network.*
- Clic en el botón *Create.*
- Una vez creado el objeto red de control.

  En el aspecto: *OPC Data Source Definition* > pestaña *" Connectivity"*
- Clic en el botón New > Add > Seleccione el Servidor de Conectividad Primario (PriAC800MCS1) y luego se agrega el Servidor de Conectividad Secundario (SecAC800MCS1).

o) Configuración del OPC Server para los Eventos y Alarmas.

- En el Plant Explorer en Service *Structure > Services > Event Collector, Services.* Clic derecho *New Object.*
- Se selecciona en la pestaña *Common* > Service Group y se coloca el nombre AC800SG1

  Clic en el botón *Create.*
- Una vez creado el nuevo *Service Group (AC800SG1)*.

  Clic en el botón derecho *New Object.*
- Se selecciona en la pestaña *Common* > Service Provider y se coloca el nombre AC800SP_PriAC800MCS1
- Clic en el botón *Create.*

  Una vez creado el nuevo *Service Provider (*AC800SP_PriAC800MCS1*)*, en el **Aspecto:** *Service Provider Definition* en la pestaña *Configuration Node* se elige de la lista desplegable a **PriAC800MCS1.**
- Se aplican los cambios. *Apply*
- Nuevamente en *Service Group (AC800SG1)*.

  Clic en el botón derecho *New Object.*
- Se selecciona en la pestaña *Common* > Service Provider y se coloca el nombre AC800SP_SecAC800MCS1
- Clic en el botón *Create.*

  Una vez creado el nuevo *Service Provider (*AC800SP_SecAC800MCS1*)*, en el **Aspecto:** *Service Provider Definition* en la pestaña *Configuration* en *Node* se elige de la lista desplegable a **SecAC800MCS1.**

- Nuevamente en *Service Group (AC800SG1)*, en el **Aspecto:** *Service Group Definition* en la pestaña *Configuration Special,* en el marco OPC A&E Server

- De la lista desplegable para la opción *Alarm Server* se selecciona*: **OPC AE Server for AC 800M.***
- De la lista desplegable para la opción Collection Definition*: **OPC AE Server for AC 800M.***

p) **Configuración Office 2007***.*

Este paso se obvia ya que no se instaló en el Office en el Servidor de Aspectos.

q) **Configuración DCOM.**

Configuration for HART Multiplexer connect on client node. Esta opción se utiliza cuando se utiliza un nodo servidor para conectar a un multiplexor de HART*.*

r) Device Library Wizard settings.

En este caso se conecta con la librería del Servidor de Aspectos.

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Se selecciona: *Device Library Wizard Administration*
- *Clic en Next.*
- *Connect Client…*
- *Clic en Next*
- *Se elige la IP del Servidor de Aspectos Primario*. En nuestro caso **172.16.4.11**
- Clic en *el botón Finish*

s) Install HART Device Types

Esta opción no se aplica en este caso ya que no se disponen de dispositivos HART.

t) Install Profibus Device Types

Se instalaron dos componentes Profibus que corresponden a transmisores de temperatura *Profibus PA*:

*ABB_TFx12_V1_2_PA **y** ABB_TFx12_12MB_PNOID_04C4_V1_0_PA.*

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Extract Device Types
- Clic en Next
- Extract device Types via Manual Selection.
- Clic en Next
- Clic en Browse
- En el DVD 2 se busca los dispositivos deseados en la carpeta *Device Library > Profibus.*
- Clic en el botón Next.

- Clic en el botón Finish.

u) **Configuración del Firewall de Windows***.*

Se utiliza la configuración automática del sistema haciendo clic en "Configure Firewall"

Configuración del Servidor de Conectividad Secundario.

Las tareas de configuración que se hicieron en el Servidor de Conectividad Secundario son las siguientes:

k) **Configuración Office 2007***.*

Este paso se obvia ya que no se instaló en el Office en el Servidor de Aspectos.

l) **Configuración DCOM.**

Configuration for HART Multiplexer connect on client node. Esta opción se utiliza cuando se utiliza un nodo servidor para conectar a un multiplexor de HART*.*

m) **Device Library Wizard settings.**

En este caso se conecta con la librería del Servidor de Aspectos.

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Se selecciona: *Device Library Wizard Administration*
- *Clic en Next*
- *Connect Client… "Si esta opción no está disponible es porque ya fue configurada"*
- *Clic en Next*
- *Se elige la IP del Servidor de Aspectos Primario*. En nuestro caso **172.16.4.11**
- Clic en *el botón Finish*

n) ***Install HART Device Types***

Esta opción no se aplica en este caso ya que no se disponen de dispositivos HART.

o) ***Install Profibus Device Types***

Se instalaron dos componentes Profibus que corresponden a transmisores de temperatura *Profibus PA*:

ABB_TFx12_V1_2_PA ***y*** *ABB_TFx12_12MB_PNOID_04C4_V1_0_PA.*

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Extract Device Types

- Clic en Next
- Extract device Types via Manual Selection.
- Clic en Next
- Clic en Browse
- En el DVD 2 se busca los dispositivos deseados en la carpeta *Device Library > Profibus.*
- Clic en el botón Next.
- Clic en el botón Finish.

p) **Configuración del Firewall de Windows***.*

Se utiliza la configuración automática del sistema haciendo clic en "*Configure Firewall*"

Configuración del Servidor de Aspectos Secundario.

Las tareas de configuración que se hicieron en el Servidor de Aspectos Secundario son las siguientes:

k) Verificar que todos los servidores de conectividad están ejecutándose.

- En el Plant Explorer en Service *Structure > Services > OpcDA_Connector, Service > SG_Control Network, Service Group*
- OPCDA_Provider_PriAC800MCS1.

  Aspecto: Service Provider Definition en la pestaña *Configuration,* verificar que el estado este en *Initialize.*
- OPCDA_Provider_SecAC800MCS1.

  Aspecto: Service Provider Definition en la pestaña *Configuration,* verificar que el estado este en *Initialize.*

l) **Configuración Office 2007***.*

Este paso se obvia ya que no se instaló en el Office en el Servidor de Aspectos.

m) **Configuración DCOM.**

Configuration for HART Multiplexer connect on client node. Esta opción se utiliza cuando se utiliza un nodo servidor para conectar a un multiplexor de HART.

n) Device Library Wizard settings.

En este caso se conecta con la librería del Servidor de Aspectos.

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Se selecciona: *Device Library Wizard Administration*

- *Clic en Next*
- *Connect Client…*
- *Clic en Next*
- *Se elige la IP del Servidor de Aspectos Primario*. En nuestro caso **172.16.4.11**
- Clic en *el botón Finish.*

o) Install HART Device Types

Esta opción no se aplica en este caso ya que no se disponen de dispositivos HART.

p) Install Profibus Device Types

Se instalaron dos componentes Profibus que corresponden a transmisores de temperatura *Profibus PA*:

ABB_TFx12_V1_2_PA ***y*** *ABB_TFx12_12MB_PNOID_04C4_V1_0_PA.*

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Extract Device Types
- Clic en Next
- Extract device Types via Manual Selection.
- Clic en Next
- Clic en Browse
- En el DVD 2 se busca los dispositivos deseados en la carpeta *Device Library > Profibus.*
- Clic en el botón Next.
- Clic en el botón Finish.

q) **Configuración del Firewall de Windows***.*

Se utiliza la configuración automática del sistema haciendo clic en "*Configure Firewall*".

Configuración de la Estación de Ingeniería y Operación.

Las tareas de configuración que se hicieron en la estación de ingeniería y operación son:

k) **Configuración Office 2007***.*

Este paso se obvia ya que no se instaló en el Office en el Servidor de Aspectos.

l) **Configuración DCOM.**

Configuration for HART Multiplexer connect on client node. Esta opción se utiliza cuando se utiliza un nodo servidor para conectar a un multiplexor de HART.

m) Device Library Wizard settings.

En este caso se conecta con la librería del Servidor de Aspectos.

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Se selecciona: *Device Library Wizard Administration*
- *Clic en Next*
- *Connect Client…*
- *Clic en Next*
- *Se elige la IP del Servidor de Aspectos Primario*. En nuestro caso **172.16.4.11**
- Clic en *el botón Finish*

n) Install HART Device Types

Esta opción no se aplica en este caso ya que no se disponen de dispositivos HART.

o) Install Profibus Device Types

Se instalaron dos componentes Profibus que corresponden a transmisores de temperatura *Profibus PA*:

ABB_TFx12_V1_2_PA ***y*** *ABB_TFx12_12MB_PNOID_04C4_V1_0_PA.*

- Start > ABB Industrial IT 800xA > Device Mgmt > ABB Device Library Wizard.
- Extract Device Types
- Clic en Next
- Extract device Types via Manual Selection.
- Clic en Next
- Clic en Browse
- En el DVD 2 se busca los dispositivos deseados en la carpeta *Device Library > Profibus.*
- Clic en el botón Next.
- Clic en el botón Finish.

p) **Configuración del Firewall de Windows***.*

Se utiliza la configuración automática del sistema haciendo clic en “Configure Firewall”.

Instalación del nodo de Simulación.

Los requerimientos del nodo de simulación son los siguientes:

Requisitos de Hardware.

Considerando que el modelo de planta realizado en Matlab-Simulink requiere muchos recursos se utiliza una PC con gran capacidad de cálculo. En la tabla

Estaciones	Procesador	Memoria RAM	Disco Rígido	Lectora DVD	Interfaces Ethernet
Nodo de Simulación	Intel Core2 Quad 2.4GHz	4GB RAM	250 GB	Sí	2 NIC(*)

(*) Se requiere la instalación de dos tarjetas de red para la configuración de dos board Matlab y manejar en una NIC el monitoreo y por la otra el control.

Tabla-A vi. Hardware para el nodo de simulación.

Instalación de Sistemas Operativos y Aplicaciones necesarias.

Las tareas de instalación en el nodo de simulación fueron las siguientes:

- Instalación del Sistema Operativo.
- Instalación de Drivers.
- Instalación de Matlab R2008a.
- Instalación de Framework 3.5 .Net

Anexo III. Conexión Placa Modbus – Módulos I/O

Los módulos con se trabaja son los siguientes: AO810 y AI810.

Configuración AO810.

Tanto la salida con la entrada de la placa modbus es de 0 a 5V, y 10 bits. Teniendo en cuenta que el modulo AO810 tiene una salida en 0-20mA se utilizó una resistencia variable para generar tensión en el rango deseado. Primero se midió la resistencia con un Tester para validar la misma. Luego se verificó que la diferencia de potencial en el cero entre el módulo AO810 y la placa no fuese significativa. El valor observado fue de 0.004 V.

Se calculó que la resistencia necesaria sería de unos 250 ohm. Luego se corrigió la resistencia experimentalmente logrando una configuración de 252 ohm, aunque eso significa realmente unos 250.4 ohm. El rango en el que se configuró este módulo es 0-1024, para garantizar que no se superen los 5V, y así lograr equiparar el valor medido en 10 bits.

Configuración AI810.

Teniendo en cuenta la despreciable diferencia de tensión entre las masas se conectó directamente la salida de la placa al modulo AI810, aunque se configuró en el módulo un valor de rango de 0-2048, para lograr equiparar el valor medido en 10bits. En este módulo se observaba que la salida de la periferia descentralizada cargaba el módulo AI810, reduciendo así el voltaje medido. Esto se lo resolvió fácilmente cambiando el rango en el módulo de entrada; se llevó este rango a 0-3100. Con esto se lograba medir valores correctos considerando que se había decidido trabajar con tres decimales de la presión lo cual implica un rango de operación entre 250 y 450.

Anexo IV. Modificación Librería NModBus.

La librería Nmodbus que se utilizó se modificó para lograr el direccionamiento estándar de Modbus. Esta librería direccionaba el primer elemento del process image en 1 (uno), mientras que el protocolo define que debe direccionarse en 0 (cero) el primer elemento.

Para esto se utilizó la herramienta de desarrollo Visual Studio 2008.

De la librería se modificó la clase DataStore.cs. De esta clase se modificó el método *ReadData* y *WriteData.*

A ambos métodos se modificó la primer línea de código que definía la dirección inicial de consulta dentro del process image.

```
int startIndex = startAddress + 1;
```

La variable *startIndex* se inicializa con el valor del parámetro *startAddress +1.* Esta variable indica la dirección inicial del elemento a leer/escribir dentro del process image de MODBUS.

Esto significa que si pasamos como parámetro a una función MODBUS la dirección inicial *startAddress*=0, el método incrementará nuestra dirección en 1. Por lo tanto bastaría con quitar de la sentencia "*+ 1".* La sentencia corregida quedaría:

```
int startIndex = startAddress;
```

Luego de las modificaciones de código se realiza nuevamente la compilación de la dll, la cuál es utilizada por los *Gateways Matlab – Modbus Rtu.*

Anexo V. Configuración de las VLAN en el Switch.

Para configurar las VLAN dentro de los switches se utilizó la herramienta web del switch.

Para crear VLANs se utiliza el menú Device, la opción VLAN. Una vez dentro de esta opción lo primero que se debe hacer es crear las cuatro VLANs. Esto se hace a través de la pestaña *Setup*.

En esta pestaña se crean las cuatro VLAN necesarias. Una vez creadas las VLANs se procede a asignar los puertos a cada una, esto se realiza desde la pestaña Modify VLAN; aquí se elige la VLAN, y se seleccionan, con el mouse, los puertos que integrarán la misma. Las siguientes figuras muestran la distribución de las VLANs dentro del switch.

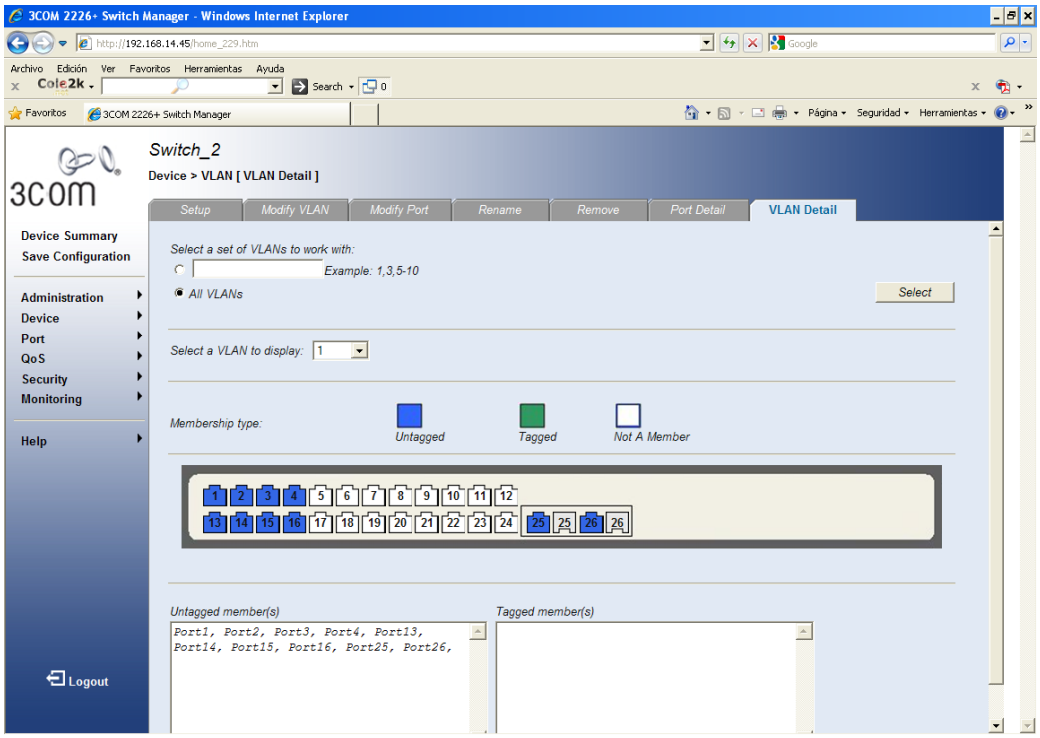

Figura-A. i. Puertos correspondientes a la VLAN2

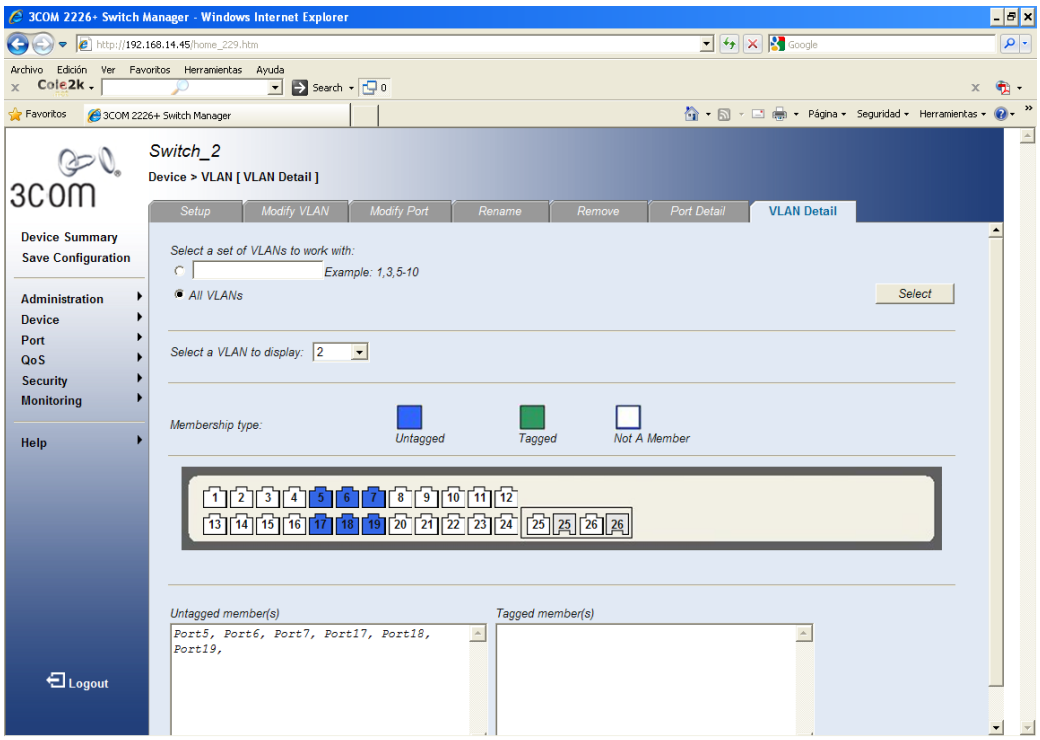

Figura-A. ii. Puertos correspondientes a la VLAN2

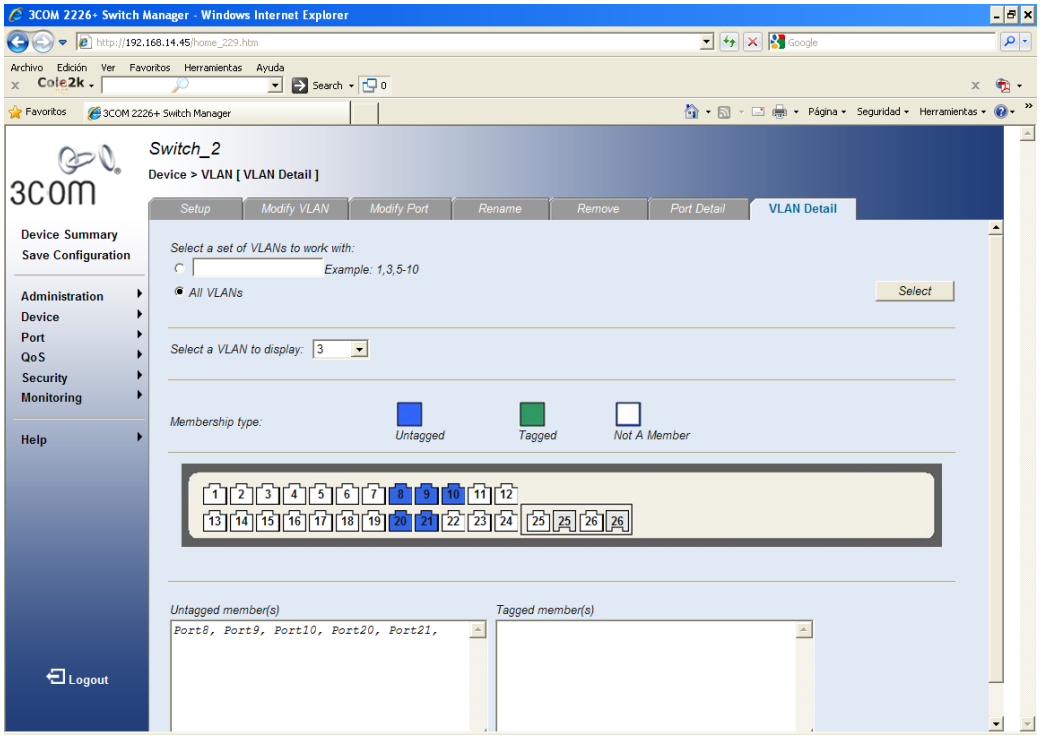

Figura-A. iii. Puertos correspondientes a la VLAN3.

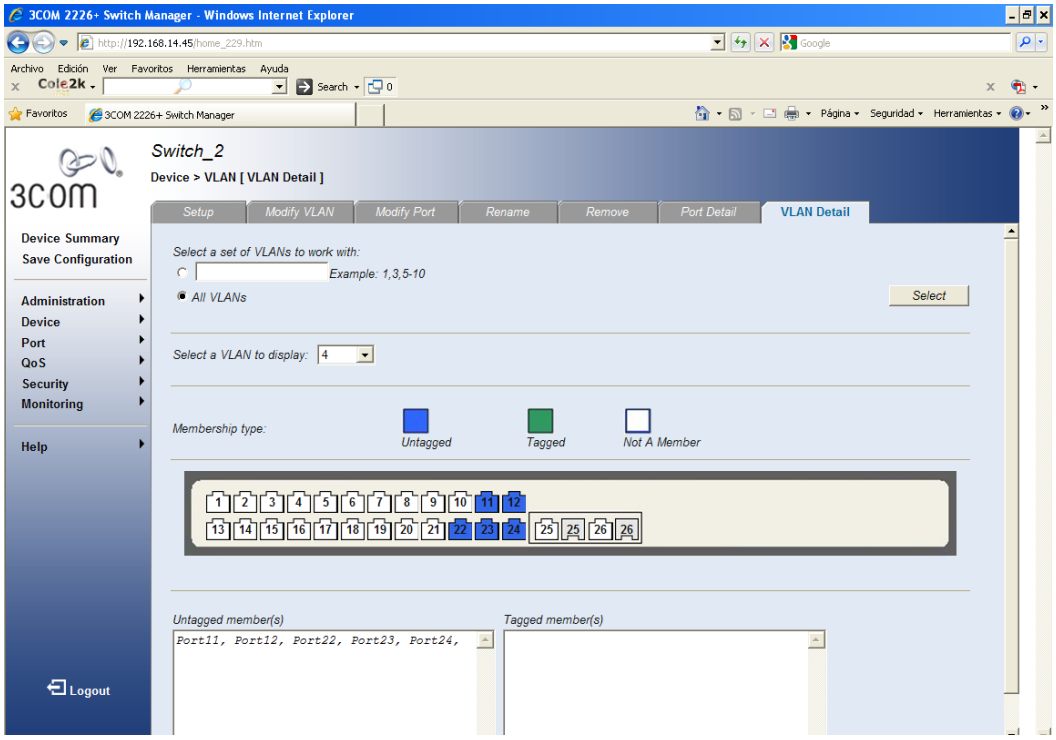

Figura-A. iv. Puertos correspondientes a la VLAN4.

Anexo VI. Estructura del Gateway Matlab - Modbus Rtu Master

Pseudo código generado automáticamente por una herramienta CASE para el diseño de sistemas software a través de una exportación del modelo.

Clase ModbusSerialMater

```
(Class "ModbusSerialMaster"
        (Operation "writeMultipleRegisters"
                (Parameter "SlaveAddress"          type    “byte")
                (Parameter "startAddress"          type    "ushort")
                (Parameter "data"                  type    "ushort[]"))
        (Operation "ReadInputRegisters"
                (Parameter "slaveAddress"          type    "byte")
                (Parameter "startAddress"          type    "ushort")
                (Parameter "numberOfPoints"        type    "ushort"))
        (Operation "CreateRtu"
                (Parameter "port"                  type    "SerialPort")))
```

Esta clase pertenece a la librería NModbus Net. Los métodos que se utilizan de esta clase son:

writeMultipleRegisters: Este método implementa la *Función 16* del protocolo Modbus. Se refiere a la escritura de holding registers. Los parámetros son:

- *SlaveAddress.* Dirección del esclavo Modbus.
- *startAddress.* Dirección que indica a partir de donde se comienza la escritura.
- *Data.* Vector de datos. Los mismos se escribirán en el esclavo a partir de la dirección *StartAddress*.

readInputRegisters: Este método implementa la *Función 02* del protocolo Modbus, es la lectura de input registers. Los parámetros son:

- *SlaveAddress.* Dirección del esclavo Modbus.
- *startAddress.* Dirección que indica a partir de donde se comienza la lectura.
- *numberOfPoints.* Cantidad de registros a leer a partir de la dirección *StartAddress*.

CreateRtu: Este método instancia un maestro modbus Serie Rtu en el Puerto definido por el parámetro *port*.

Clase imageProcess

Esta clase implementa un *process image* en la cual el maestro almacena las variables de la periferia descentralizada y desde el modelo simulado en matlab. Los métodos de esta clase se agrupan en de acuerdo al tipo de entrada / salida.

```
(Class "imageProcess"
        (Attribute "inputDicretes" type  ”boolean[]")
        (Attribute "inputRegisters" type "ushort[]")
        (Attribute "coils" type "boolean[]")
        (Attribute "holdingRegisters" type "ushort[]")
        (Operation "imageProcess"
                (Parameter "countInputDiscretes"    type “integer”)
                (Parameter "countInputRegisters"    type “integer”)
                (Parameter "countCoils" type “integer”)
                (Parameter "countHoldingRegisters" type “integer”))
        (Operation "setInputDiscrete"
                (Parameter "index" type “integer”)
                (Parameter "newValue" type “boolean”))
        (Operation "getInputDiscrete"
                (Parameter "index" type “integer”)
                result          "boolean")
        (Operation "getCantInputDiscretes"
                result          "integer")
        (Operation "updateInputDiscretes"
                (Parameter "initIndex" type “integer”)
                (Parameter "vectorInputDiscretes" type “boolean[]”))
        (Operation "getInputDiscretes"
                (Parameter "initIndex" type “integer”)
                (Parameter "count" type “integer”)
                result          "boolean[])"
        (Operation "setCoil"
                (Parameter "index" type “integer”)
```

```
        (Parameter "newValue" type “boolean”))
(Operation "getCoil"
      (Parameter "index" type “integer”)
        result          "boolean")
(Operation "getCantCoils"
        result          "integer")
(Operation "updateCoils"
        (Parameter "initIndex" type “integer”)
        (Parameter "vectorCoils" type “boolean[]”))
(Operation "getCoils"
        (Parameter "initIndex" type “integer”)
        (Parameter "count" type “integer”)
        result          "boolean[]")
(Operation "setInputRegister"
        (Parameter "index" type “integer”)
        (Parameter "newValue" type “ushort”))
(Operation "getInputRegister"
        (object Parameter "index" type “integer”)
        result          "ushort")
(Operation "getCantInputRegisters"
        result          "integer")
(Operation "updateInputRegisters"
        (Parameter "initIndex" type “integer”)
        (Parameter "vectorInputRegisters" type “ushort[]”))
(Operation "getInputRegisters"
        (Parameter "initIndex" type “integer”)
        (Parameter "count" type “integer”)
        result          "ushort[]")
(Operation "setHoldingRegister"
        (Parameter "index" type “integer”)
        (Parameter "newValue" type “ushort”))
```

(Operation "getHoldingRegister"
(Parameter "index" type “integer”)
result "ushort")
(Operation "getCantHoldingRegisters"
result "integer")
(Operation "updateHoldingRegisters"
(Parameter "initIndex" type “integer”)
(Parameter "vectorHoldingRegisters" type “ushort[]”))
(Operation "getHoldingRegisters"
(Parameter "initIndex" type “integer”)
(Parameter "count" type “integer”)
result "ushort[]"))

Atributos

Los atributos definidos en esta clase son:

inputDiscretes: *Es un vector que contiene a cada una de las input discretes.*
coils: *Es un vector que contiene a cada una de las coils.*
inputRegisters: *Es un vector que contiene a cada uno de los input registers.*
holdingRegisters*: Es un vector que contiene a cada una de los holding registers****.***

imageProcess: Este método es el constructor de la clase inicializando el objeto con la cantidad de inputDiscretes, coils, inputRegistes y holdingRegisters definidos en los parámetros. Se refiere a la escritura de holding registers. Los parámetros son:

- *countInputDiscretes****.*** Cantidad de Input Discretes que se inicializaran.
- *countCoils****.*** Cantidad de Input Coils que se inicializaran.
- *countInputRegisters****.*** Cantidad de Input Registers que se inicializaran.
- *countHoldingRegisters****.*** Cantidad de Holding Registers que se inicializaran.

Input Discretes

setInputDiscrete: Este método actualiza una determinada input discrete. Los parámetros son:

- *index****.*** Índice de la input discrete a actualizar.
- *newValue****.*** Nuevo valor con el cual se actualiza la input dicrete.

getInputDiscrete: Este método devuelve el valor de la input discrete deseada. El parámetro definido es:

- *index.* Índice de la input discrete deseada.

getCantInputDiscretes: Este método devuelve la cantidad de Input Discretes definidas en el Image Process.

updateInputDiscretes: Este método actualiza varias input Discretes, a partir de una dirección inicial y un vector con los valores a actualizar. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual actualizar las input discretes.
- *vectorInputDiscretes.* Vector con los valores a actualizar de las input discretes.

getInputDiscretes: Este método devuelve un vector con los valores de varias input discretes, a partir de una dirección inicial y un valor indicando la cantidad de input discretes deseadas. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se desea obtener las input discretes.
- *count.* Cantidad de input discretes deseados.

Coils.

setCoil: Este método actualiza una determinada coil. Los parámetros son:

- *index.* Índice de la coil a actualizar.
- *newValue.* Nuevo valor con el cual se actualiza la coil.

getCoil: Este método devuelve el valor de la coil deseada. El parámetro definido es:

- *index.* Índice de la coil deseada.

getCantCoil: Este método devuelve la cantidad de coils definidas en el image Process.

updateCoils: Este método actualiza varias coils, a partir de una dirección inicial y un vector con los valores a actualizar. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se actualizaran las coils.
- *vectorCoils.* Vector con los valores a actualizar las coils deseadas.

getCoils: Este método devuelve un vector con los valores de varias Coils, a partir de una dirección inicial y un valor indicando la cantidad de coils deseadas. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se desea obtener las coils.
- *count.* Cantidad de coils deseadas.

Input Registers

setInputRegister: Este método actualiza un determinado input register. Los parámetros son:

- *index.* Índice del input register a actualizar.
- *newValue.* Nuevo valor con el cual se actualiza el input register.

getInputRegister: Este método devuelve el valor del input register deseado. El parámetro definido es:

- *index.* Índice del input register deseado.

getCantInputRegisters: Este método devuelve la cantidad de input registers definidos en el image Process.

updateInputRegisters: Este método actualiza varios input registers, a partir de una dirección inicial y un vector con los valores a actualizar. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se desea actualizar los input register.
- *vectorInput*Register*s.* Vector con los valores a actualizar los input registers, deseados.

getInputRegisters: Este método devuelve un vector con los valores de varios input registers, a partir de una dirección inicial y un valor indicando la cantidad de input register deseados. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se desea obtener los input registers.
- *count.* Cantidad de input register deseados.

Holding Registers

setHoldingRegister: Este método actualiza un determinado holding register. Los parámetros son:

- *index.* Índice del holding register a actualizar.
- *newValue.* Nuevo valor con el cual se actualiza el holding register.

getHoldingRegister: Este método devuelve el valor del holding register deseado. El parámetro definido es:

- *index.* Índice del holding register deseado.

getCantHoldingRegisters: Este método devuelve la cantidad de holding registers definidos en el image Process.

updateHoldingRegisters: Este método actualiza varios holding registers, a partir de una dirección inicial y un vector con los valores a actualizar. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se desea actualizar los holding register.
- *vectorInput*Register*s.* Vector con los valores a actualizar los holding registers, deseados.

getHoldingRegisters: Este método devuelve un vector con los valores de varios holding registers, a partir de una dirección inicial y un valor indicando la cantidad de holding register deseados. Los parámetros son:

- *initIndex.* Dirección inicial a partir de la cual se desea obtener los holding registers.
- *count.* Cantidad de holding register deseados.

Clase Timer

Esta clase pertenece al framework .net 3.5 de Microsoft. Implementa un thread que ejecuta cíclicamente tareas considerando un intervalo definido en milisegundos a través del método sleep. Aquí se muestran los atributos y los métodos empleados.

```
(Class "Timer"
        (Attribute "enabled" type  "boolean")
        (Attribute "Name" type "string")
        (Attribute "Interval" type "integer")
        (Event "tick"))
```

Atributos.

Los atributos que se utilizan de la clase son:

Enabled: este atributo habilita al objeto a ejecutar cíclicamente el evento tick.

Name: este atributo define el nombre del objeto para su identificación.

Interval. este atributo define los milisegundos de cada ciclo del objeto *Timer.*

Eventos.

Evento tick. Este evento se ejecuta cíclicamente cada *Interval* misegundos.

Clase Winsock

Esta clase pertenece a un framework anterior de Microsoft. Este componente permite abrir un sock en una dirección IP particular y en un puerto determinado. Este componente permite enviar y recibir datos mediante el protocolo UDP y TCP, los cuales ya están implementados. Este componente es: *microsoft winsock control, 6.0*. Aquí se muestran los atributos y los métodos empleados.

```
(Class "WinSock"
```

```
(Attribute "Name"   type “string”)
(Attribute "LocalIP"  type “inetAddress”)
(Attribute "LocalPort" type “inetAddress”)
(Attribute "Protocol" type “integer”
        (Options  value “1- sckUDPProtocol”)
        (Options  value “2- sckTCPProtocol”))
(Attribute "RemoteHost" type “inetAddress”
(Attribute "RemotePort" type “integer”)
(Operation "SendData")
(Operation "DataArrival”)
(Operation "Bind”))
```

Atributos.

Los atributos que se utilizan de la clase son:

Name: este atributo define el nombre del objeto para su identificación.

LocalIP: este atributo define la dirección IP local.

LocalPort: este atributo define el puerto local.

Protocol: este atributo define el protocolo de comunicación (TCP, UDP).

RemoteHost: este atributo define la dirección IP del dispositivo remoto con el cual se establecerá la comunicación.

RemotePort: este atributo define el puerto del dispositivo remoto con el cual se establecerá la comunicación.

Eventos y Métodos.

Método SendData. Este método permite enviar los datos al dispositivo remoto a través del puerto remoto definido.

Evento DataArrival. Este evento se produce cuando llegan datos desde otro dispositivo al puerto definido como *LocalPort*.

Método Bind. Este método crea un thread que escucha en el puerto definido como *LocalPort.* Cuando llegan datos al puerto local se genera el evento *DataArrival.*

Clase SerialPort

Esta clase pertenece al framework .net 3.5 de Microsoft. Este componente permite abrir un puerto serie. Este componente permite enviar y recibir datos mediante el protocolo RS232. Aquí se muestran los atributos y los métodos empleados.

(Class "SerialPort"

(Attribute "PortName" type “string”)

(Attribute "BaudRate" type “integer”)

(Attribute "DataBits" type “integer”)

(Attribute "Parity" type “SerialPort.Parity")

(Attribute "StopBits" type "SerialPort.StopBits")

(Operation "Open")

(Operation "Close"))

Atributos.

Los atributos que se utilizan de la clase son:

PortName: este atributo define el nombre del puerto serie a abrir. En el caso de Windows son COM1, COM2, etc.

BaudRate: este atributo define la velocidad de transmisión (bits por segundo).

DataBits: este atributo define los bits de datos, 8 es utilizado ya que periferia descentralizada utiliza ese parámetro.

Parity: este atributo define el tipo de paridad (par, impar, ninguna).

StopBits: este atributo define los bits de parada en la transmisión.

Métodos.

Open. Este método abre el puerto serie para iniciar la transferencia de datos.

Close. Este método cierra el puerto.

Anexo VII. Estructura del Gateway Matlab - Modbus Rtu Slave

Pseudo código generado automáticamente por una herramienta CASE a través de una exportación del modelo.

Clase DataStoreFactory

```
(Class "DataStoreFactory"
  (Operation "CreateDefaultDataStore"
      (Parameter "coilsCount"  type  "ushort")
      (Parameter "inputsCount" type  "ushort")
      (Parameter "holdingRegistersCount"  type  "ushort")
      (Parameter "inputRegistersCount"  type  "ushort"))
```

Esta clase pertenece a la librería NModbus Net. Los métodos que se utilizan de esta clase son:

CreateDefaultDataStore: Este método implementa la instanciación de un *DataStore*. Los parámetros son:

- ***coilsCount.*** Cantidad de Coils que se crearán en el image process.
- ***inputCount.*** Cantidad de input discretes que se crearán en el image process.
- ***holdingRegistersCount.*** Cantidad de holding registers que se crearán en el image process.
- ***inputRegistersCount.*** Cantidad de input Registers que se crearán en el image process.

Clase DataStore

```
(Class "DataStore"
      (Attribute "CoilDiscretes")
      (Attribute "InputDiscretes")
      (Attribute "InputRegisters")
```

(Attribute "HoldingRegisters"))

La clase pertenece a la librería NModbus. DataStore es una clase que representa el image process.

Clase InputRegisters

```
(Class "InputRegisters:Collections"
        (Attribute "index" type  "integer")
        (Attribute "value" type  "ushort")
        (Operation "SetItem"
          (Parameter "index"  type "integer")
          (Parameter "item"   type "ushort"))
        (Operation "getItem"
          (Parameter "index" type "integer")
           result          "ushort")
```

La clase InputRegisters es la implementación de una clase Collections del Framework .Net 3.5, ,aunque así como las demás clases, pertenece a la librería NModbus. En este caso se refiere a la colección de input registers asociados al tipo de dato ushort.

Los atributos de esta clase son:
Index: *Este atributo representa el índice del elemento en la colección.*
Value: *Este atributo representa el valor del elemento de la colección.*

Los métodos de esta clase son:

SetItem: Este método actualiza un determinado input register, es decir un elemento de la colección. Los parámetros son:

- *index.* Índice del elemento de la colección Input Registers.
- *item.* Nuevo valor con el cual se actualiza el elemento de la colección Input Registers.

GetItem: Este método devuelve el valor del elemento de la colección Input Registers. El parámetro definido es:

- *index. Índice del elemento de la colección deseado.*

Clase ModbusSerialSlave

```
(Class "ModbusSerialSlave"
        (Operation "Listen")
        (Operation "CreateRtu"
                (Parameter "unitId"    type    "byte"))

                (Parameter "port"      type    "SerialPort")))
```

Esta clase pertenece a la librería NModbus Net. Los métodos que se utilizan de esta clase son:

CreateRtu: Este método instancia un esclavo modbus Rtu en el Puerto definido por el parámetro *port* y con la dirección definida en *unitId*.

- ***UnitId.*** Parámetro que define la dirección del esclavo dentro de la red modbus.
- ***Port.*** Parámetro que define la instancia hacia un puerto serie.

Clase WinSock y SerialPort

Las clases Winsock y SerialPort fueron definidas en el anexo VI.