UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

TRABAJO FIN DE GRADO

Control avanzado de una planta virtualizada con LabVIEW a través de un servidor OPC

Alumno: **Guillermo Suárez Martínez**
Director: Adolfo Hilario Caballero

GRADO EN INGENIERÍA ELÉCTRICA

Convocatoria de defensa: **JULIO 2018**

Resumen

Se controlará, utilizando diferentes técnicas, una planta virtualizada en un computador mediante el programa LabVIEW, siendo el controlador un PLC S7-1214C. Se compararán diversos tipos de controles valorando los límites que el autómata presenta. Se mostrará cómo realizar un sistema OPC con el que compartir la comunicación de distintos procesos, programas y lenguajes para poder mejorar el control. El objetivo es crear una nueva puerta para controles más avanzados, pudiendo comprobar nuestros sistemas sin tenerlos delante y mejorando así las posibilidades para unir procesos controlados independientemente. Se gestionará mediante el servidor OPC la posibilidad de mejorar tareas administrativas pudiendo tener en tiempo real datos de nuestro sistema.

Se realizará un estudio técnico-económico de cómo controlar el proceso valorando las limitaciones y puntos fuertes de las distintas tecnologías. Se realizan distintas comparaciones de medios de control avanzado para la planta, observando las mejoras que unos presentan sobre otros y argumentando la solución mas favorable para nuestro proceso.

Se genera un documento excel maestro con el que observar datos en tiempo real del proceso, pudiendo utilizar estos datos dentro de la hoja como un número más.Esta posibilidad, permite incluir dichos números en formulas, para obtener así tanto datos de rendimientos del proceso como temperaturas actuales con las que se podrían generar informes de calidad, informes de rendimiento, organización de medios productivos y aprovisionamiento de materiales, reduciendo así tiempos muertos e incluso optimizando espacio de almacenaje.

Índice general

Capítulo 1

Introducción

En el presente capitulo se mostrarán las directrices que se seguirán para la creación del proyecto, los antecedentes, situación y objetivos que éste intentará cumplir, así como las herramientas con las que contamos para desempeñar esta tarea. El presente punto sirve como argumentación para las decisiones tomadas en los siguientes capítulos, los cuales, tratarán únicamente la solución y se centrarán en cómo llegar a ella. Se mostraran las herramientas utilizadas para el desempeño del proceso y el enfoque con el que se usará cada una de ellas.

1.1 Antecedentes

En la situación actual en la que nos encontramos, la sociedad esta sufriendo una nueva revolución tecnológica. Encontramos en los mercados elementos cotidianos de uso diario con un grado elevado de tecnología, que hace unos pocos años era impensable. Tenemos relojes inteligentes que nos indican cuánto hemos andado, frigoríficos que hacen la lista de la compra por nosotros, televisores con los que poder ver los programas que queramos en el momento que nos interese... La industria no se ha quedado atrás y estamos viviendo una nueva revolución industrial, en la que la industrialización de procesos supone la perpetuidad en el mercado de ciertos productos, o el declive de grandes compañías. Todas las empresas están apostando por la automatización de procesos para generar productos de mejor calidad y a un coste mucho menor.

En este momento las inversiones en mejoras de procesos productivos están en todos los presupuestos de las empresas que quieren asegurar una cuota del mercado. Es así como nacen nuevas tecnologías que al poco tiempo se acaban implantando en nuestra vida cotidiana y en los productos que las propias empresas generaban. Las empresas apuestan por la automatización de procesos para mejorar los costes de producción, la gestión de los recursos, mejorando la competitividad de los mismos y destacándose sobre otras empresas del sector.

Debido a este marco social en el que nos encontramos actualmente y a la gran demanda que tiene en el mercado la automatización de sistemas cada vez más sofisticados y precisos, se pretende desarrollar un método de comunicación con el que conocer el estado en tiempo real de nuestro sistema desde cualquier ordenador de la red empresarial, ya sea una empresa de ámbito local con una única sede, o una empresa multinacional con numerosas factorías. Además, se generaría un proceso por el cual podremos conocer los limites admisibles a los que podemos

someter nuestros equipos y la respuesta que tendrán mediante pruebas reales sobre la planta virtualizada.

1.2 Motivación

La principal motivación para desarrollar y ejecutar este proyecto es ver la cantidad de aplicaciones reales que puede tener la comunicación OPC para la industria en general. Este tipo de comunicación puede añadir valor a distintos campos de la industria, pudiendo someter a situaciones reales a equipos sin la necesidad de recrearlas, unicamente con las lecturas de los sensores. Un ejemplo claro del alcance que puede tener este proyecto puede ser la de someter a los equipos que componen un avión a las lecturas registradas en las cajas negras que se recogen tras un accidente para conocer realmente si alguno de estos equipos esta mal diseñado, o acotar el fallo que género el accidente a alguno de estos, ya que bajo determinadas circunstancias pueda actuar de una manera inesperada (ruido en las señales, acople de señales, resonancias...). Otro ejemplo sería la de someter estos equipos a lecturas que jamás podríamos encontrarnos en situaciones normales de funcionamiento, sabiendo si el grado de seguridad aplicado es el correcto o si estos equipos podrían funcionar en ciertas circunstancias excepcionales como accidentes.

Otro ejemplo de la potencia de este protocolo de comunicación puede ser la de enlazar elementos comunes de la automatización como autómatas, controladores, sensores, pantallas... de distintas marcas que se comportan y funcionan de maneras tan diferentes en una sola maquina, mejorando así las prestaciones y el precio de estas. Actualmente en el mercado encontramos diversas empresas que fabrican componentes para la industria, estando especializadas en alguna de las citadas anteriormente. Con este protocolo de comunicación podríamos enlazar elementos excepcionales de distintas marcas en una sola maquina mejorando así las prestaciones de las mismas, sin importar el protocolo de comunicación que sigan, o el lenguaje de programación que tengan. Por otro lado se podría abaratar el precio de maquinas competitivas, seleccionando elementos lo mas baratos posibles de distintas marcas pudiendo enlazarlos, mejorando el precio que una empresa de automatización puede ofertar ante otras. Un ejemplo claro era la de las pantallas HMI con las que controlar las maquinas mediante un SCADA, ya que estas van asociadas a un autómata, o varios, que controlan unas funciones del segmento que se muestra en la pantalla. Antes desviamos priorizar si preferíamos la comodidad de cierta marca de pantallas por sus prestaciones o por su economía, o la calidad algunas marcas de autómatas. Ademas de todo esto y teniendo en cuenta que las empresas conocían estas limitaciones e incrementaban el precio de equipos cuando los requisitos de ciertas industrias desembocaban en la compra de estos sin competencia alguna podremos mejorar el mercado de la industria aplicando la comunicación OPC.

Por ultimo, la posibilidad de generar un ”hardware in the loop” con el que demostrar el funcionamiento de un autómata y realizar un estudio de automatización si la necesidad de parar un segundo la cadena de producción.

1.3 Objetivos

En primer lugar, este proyecto tiene como objetivo principal adquirir las destrezas y capacidades para aportar soluciones al mundo de la industria, tanto en programación de PLC, como en comunicación industrial, centrándose sobre todo en los servidores OPC, que, tal y como se cito en la sección 1.2, tiene numerosas funcionalidades. Así pues, lo que se ha pretendido con este trabajo es comunicar el autómata con programas que todo el mundo tenga, en este caso con el Excel. Gracias a ello,se acerca la automatización y la gestión a cualquier parte de la industria para mejorar los recursos que las empresas malgastan. De este modo, no solamente se optimizarían los procesos, como hasta ahora, sino que conocerían el estado real de lo que se fabrica, adquiriendo así la capacidad de aprovisionar y dar salida a estos recursos en el momento adecuado, ahorrando espacio de almacenaje, tiempos muertos por falta de material en los distintos procesos, o ganando fiabilidad ante los clientes conociendo el estado del producto que se va a vender. Por otro lado, esta herramienta puede generar certificados de calidad, puesto que el producto puede estar controlado en todo momento, aportando pruebas así de que los procesos de fabricación han estado bajo norma.

Por otro lado, se ha aplicado el control a un sistema real.ºbservando las limitaciones que el autómata pueda presentar y que se estén pasando por alto en la generación del proyecto teórico. Todo ello, permite obtener los datos de los sobrecostes de ingeniería que un proyecto de esta magnitud supone para las empresas que los llevan a cabo.

Por ultimo, tal y como se citó en la sección 1.1, las empresas están demandando ingenieros totalmente cualificados y capaces de generar y realizar proyectos que mejoren y automaticen los procesos productivos. Se les exige ser capaces de simplificar las tareas y/o mejorar la precisión de las mismas para que estas se realicen de forma mas rápida y económica. De esta modo, con la generación de este proyecto pretendo acercarme al futuro mundo laboral, dejando a un lado el mundo de las aulas.

1.4 Factores a considerar

En este punto se va a intentar destacar o enumerar los puntos que se han tenido en cuenta para la realización de este proyecto; desde normativa aplicable (local, regional, estatal...) hasta las propias limitaciones del proceso que la automatización pueda presentar.

1.4.1 Normativa aplicable

Previamente a la realización del proyecto, se ha procedido a la consulta de la normativa aplicable. Esta será de obligado cumplimiento para el diseño, instalación, puesta en marcha y mantenimiento de la misma.

- REBT/: reglamento electrotécnico de baja tensión (RD 842/2002)
- Guía técnica de aplicación de febrero de 2017
- IEC 61131-3 de febrero de 2013 con el estándar de lenguajes de programación.
- EN ISO 13849-1 revisión de 2016 sobre la seguridad de las maquinas.
- UNE-EN 61000-6-2: compatibilidad electromagnética.

- IEC 62541: Especificaciones de la comunicación OPC
- IEC 60870-5-101: definición TCP/IP

1.4.2 Limitaciones de diseño

Para intentar asemejar este proyecto docente a uno real, se han preimpuesto una serie de limitaciones de obligado cumplimiento que se corresponderían a las que un cliente podría proponer para esta clase de proyectos. Algunas de estas limitaciones podrían ser:

- Automatizar el proceso sin necesidad de tener operarios ante la máquina para el funcionamiento normal del proceso, únicamente para la puesta en marcha y finalización del mismo.
- Crear un servidor OPC por el que se gobierne o se monitorice el proceso desde cualquier punto de la industria, incluyendo factorías externas que se encuentran en otro país y que puedan formar parte del producto final.
- Aportar información en tiempo real del estado y de la situación del proceso productivo.

1.4.3 Limitaciones técnicas

Otro factor a considerar son las limitaciones técnicas del proceso productivo, ya que solo se va a automatizar parte de la cadena de producción de este y no el total de la misma. Por ello, deberemos mantener el sistema actual de trabajo, no modificando ni el comienzo ni el final, para que el funcionamiento de la planta permanezca idéntico y no se altere el proceso ni el flujo productivo. Otra limitación importante a tener en cuenta es el espacio físico, ya que no se deberá cambiar el flujo de trabajo del mismo. Si este se modificara, se tendrían que desplazar los semiprocesos anteriores o posteriores para mantener la unión de estos (cintas de transporte, tuberías, brazos robóticos...). Por ello, se pretende realizar una automatización integral con la menor modificación posible, tanto en las obras como en el cambio del proceso.

1.4.4 Prestaciones que debe cumplir el proyecto

Se deben cumplir las prestaciones que actualmente ya tiene el proceso al menos (velocidad de producción, tiempos de respuesta, residuos del proceso...) o mejorarlas, además de incluir las nuevas prestaciones que la automatización implementa, entre ellas:

- Controlar la temperatura y ajustarla hasta la referencia deseada.
- Poder observar el estado de la temperatura, o el nivel del tanque en tiempo real desde cualquier parte de la fábrica.
- Gobernar el proceso desde la propia máquina favoreciendo así el mantenimiento de la misma.
- Controlar a distancia, desde otras sedes, la máquina para acelerar procesos productivos en periodos de vacaciones.

1.5 Descripción del proceso

En este punto se va a pasar a describir el proceso productivo que actualmente tiene lugar en la empresa, y cómo se pretende mejorarlo gracias a las distintas implementaciones a realizar. Posteriormente, se detallará el estudio realizado para obtener el mejor control posible del proceso.

1.5.1 Proceso actual

En estos momentos el proceso cuenta con un tanque que se llena mediante una válvula de accionamiento manual (a partir de ahora válvula A). El operario debe llenar el tanque de un líquido (a partir de ahora líquido A) hasta que un nivel le indica que esta completamente lleno. En ese momento debe cerrar la válvula y encender una caldera que calienta otro líquido (a partir de ahora líquido B) que circula alrededor del tanque, por la camisa del mismo, cediendo el calor al líquido A. Para monitorizar la temperatura, el operario cuenta con un termómetro que está introducido en el líquido A. Con una segunda válvula (válvula B) se regula el caudal del líquido B que circula por la camisa del tanque. El liquido A debe permanecer a una temperatura de referencia hasta que sea requerido por el proceso productivo para otras funciones.

Una vez que el líquido ya está atemperado, y es requerido para continuar con el proceso, el operario cerrará por completo la válvula que permite que circule el líquido que cede calor (válvula B), y activará otra válvula de salida del tanque (válvula C). Esta tercera válvula vaciará el tanque por completo y el operario la cerrará cuando en nivel de llenado indique que está totalmente vacío.

Un esquema del sistema se muestra en la figura 1.1 donde se representa: el nivel del tanque con un rectángulo que contiene la inscripción ”NIVEL”, el termómetro en forma de triángulo con la inicial ”T” y las válvulas manuales con el símbolo normalizado de estas.

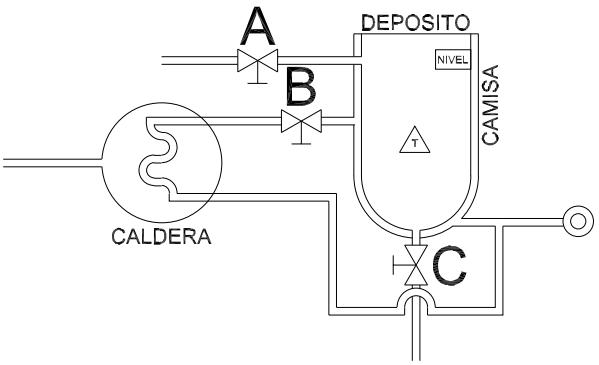

Figura 1.1: Esquema de la planta a controlar

1.5.2 Proceso de mejora

La mejora que se pretende instalar es la automatización y regulación de todo el proceso, de tal manera que sea completamente autónomo, semiautomático o manual, mejorando así las labores de mantenimiento tanto preventivo como correctivo. El primer sistema de mejora que se pretende implantar es la sustitución de las válvulas manuales de llenado (válvula A) y vaciado (válvula C) por dos electroválvulas que se accionen por medio de dos pulsadores. El sensor de nivel se sustituirá por un sensor electrónico de lógica digital. Esto implica que cuando el tanque esté completamente lleno, el sensor se activará. Para controlar el vaciado, se ha de instalar un nuevo sensor, de nivel mínimo, también electrónico de lógica digital, que activará una salida cuando el líquido esté por encima de este. El termómetro se sustituirá por una sonda PT100. La válvula de regulación de caudal del líquido circulante por la camisa del tanque (válvula B) se sustituirá por una electroválvula regulada en tensión entre 0 y $10V$, modificando el grado de apertura.

De esta manera, obtendremos un esquema de la instalación como el que se muestra en la figura 1.2, donde tenemos representado la sonda PT100 con un triangulo con las iniciales "PT", los sensores de nivel máximo y mínimo con dos rectángulos con las iniciales "N.MAX" y "N.MIN" respectivamente y las electroválvulas con el símbolo normalizado de estas.

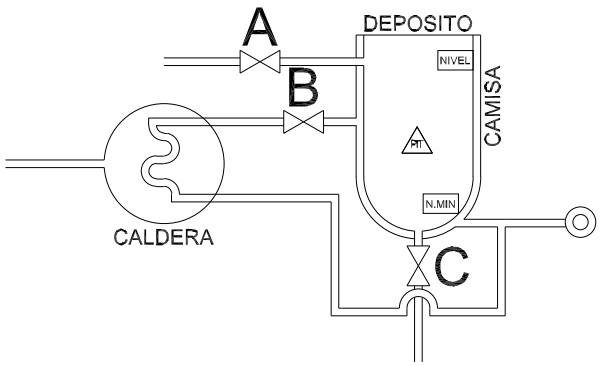

Figura 1.2: Esquema de la planta a controlar con las mejoras implementadas

A continuación, se procederá a la exposición de los distintos modos de funcionamiento que se pretenden poner en marcha en la instalación descrita:

- **Modo manual:** En este modo se cambiará el proceso de apertura de las válvulas A y C por dos pulsadores que abran y cierren cada una de estas válvulas. La válvula B se regulara mediante un potenciómetro que controle el grado de apertura variando la tensión aplicada entre 0 y $10V$. El sensor de llenado activará una luminaria que indicará al operario el estado del tanque lleno, debiendo activar el pulsador de paro de la válvula A en ese momento. También se instalará otra luminaria que indique que el líquido está por debajo del nivel mínimo para que el operario cierre la válvula de vaciado (válvula C). Para controlar la temperatura de la sonda PT100 se instalará un pequeño diplay digital que muestre la temperatura real del líquido. Mediante esta mejora, hemos conseguido centralizar la maniobra de las válvulas a un único panel, reduciendo tiempos en el proceso y haciendo que el operario pueda estar en un único sitio sin tener que desplazarse para manipular las válvulas. Se a mejorado el tiempo de vaciado del tanque puesto que ahora cuenta con una señal luminosa que indica que el líquido A está por debajo del nivel, mientras que antes el trabajador debía estimarlo.

- **Modo semiautomático:** En modo semiautomático las válvulas A y B se abrirán manualmente, pero se cerraran solas una vez el sensor de nivel de la orden. Es decir, el

operario abrirá la válvula A activando el pulsador de apertura de esta válvula, pero se cerrará automáticamente cuando el sensor de nivel indique que está el tanque lleno. El operario accionará la salida del liquido A mediante el pulsador de apertura de la válvula C, pero se cerrara automáticamente cuando el sensor indique que el tanque está vacío. El control de la temperatura quedará bajo la supervisión del operario, debiendo manejar el potenciómetro de regulación de la válvula B. Con este sistema se ha mejorado la respuesta del tanque, ya que en este modo de funcionamiento el operario no debe estar presente para cerrar las válvulas, sino que podrá avanzar al proceso siguiente una vez indique el vaciado del tanque.

- **Modo automático:** En este modo, el operario indicará cuando ha de llenarse el tanque, mediante el pulsador de apertura de la válvula A, y cuando es necesario que el líquido se incoorpore a la cadena de producción, activando el pulsador de la electroválvula C, el cual vaciará el tanque de forma automática. Para seleccionar la temperatura a la que se desea el líquido se introducirá manualmente la referencia desde el panel de operador y un regulador controlará la válvula B externamente. De este modo, el proceso se vuelve automático completamente y permite liberar al operario de esta parte del proceso productivo.

1.6 Soluciones alternativas y justificación de la solución adoptada

Para implementar las mejoras descritas en la subsección 1.5.2 existen diversas tecnologías, las cuales cuentan con ventajas e inconvenientes que pasaré a detallar:

1.6.1 Lógica

Para automatizar este proceso se puede optar por un tipo cableado, con contactores, una automatización mediante microprocesadores, teniendo en cuenta que se necesitarán elementos intermedios que unan los elementos de potencia con los elementos electrónicos, o una automatización mediante un controlador lógico programable. A continuación pasaremos a enumerar las ventajas e inconvenientes de todos ellos:

1. Cableado:
   - Ventajas:
     - Precio más económico, ya que se necesitan pocos elementos para la automatización y el coste de estos no es elevado al tratarse de elementos de poca potencia como electroválvulas.
     - Fácilmente variable para adaptarse a modificaciones del modelo productivo.
     - Conste energético nulo en standby.
     - Alta resistencia a las perturbaciones electromagnéticas.
   - Inconvenientes:
     - El mantenimiento de estos elementos es mayor, ya que tienen un numero de maniobras determinado. El coste de mantenimiento por lo tanto es mayor.
     - Si el modelo productivo cambia habrá que contar con mucho más material, incrementando el coste de modificaciones, aunque este no debería ser caro.

- Imposibilita la mejora de la gestión, punto clave descrito en subsección 1.5.2

2. Microprocesador

   - Ventajas:

     - Reducido espacio.
     - Coste muy bajo de los elementos.
     - Mantenimiento alto, pero a coste muy bajo. La electrónica tiene vida útil pero el coste es menor que en el caso del cableado.

   - Inconvenientes:

     - Alto nivel de especialización del equipo de mantenimiento.
     - Coste elevado de modificaciones, debido al alto nivel de programación necesario.
     - Gran numero de elementos a montar.
     - Coste de standby mayor que en cableado.
     - Se ve afectado en mayor grado por las perturbaciones electromagnéticas y esta clase de tecnología puede degradar la calidad de nuestra red. THDi, THDv, FDP, PF...

3. PLC:

   - Ventajas:

     - Facilidad de modificación.
     - Reducción del espacio del cuadro eléctrico.
     - Bajo mantenimiento.
     - Reducción del coste de mano de obra por mantenimiento.
     - Estable ante las perturbaciones electromagnéticas.

   - Inconvenientes:

     - Coste elevado inicial.
     - Mayor consumo que el microcontrolador en standby.

4. Tarjetas de adquisición de datos:

   - Ventajas:

     - Capaces de convertir datos D/A y A/D.
     - Capaces de tratar magnitudes físicas.
     - Gran almacenamiento de datos.
     - Gran velocidad de procesamiento.

   - Inconvenientes:

     - Requieren de un ordenador para su funcionamiento.

- Precisan gran espacio para el montaje, ya que necesitan instalar el ordenador próximo.
- Coste elevado para un funcionamiento óptimo, ya que se debe mantener no solo el sistema de la tarjeta de adquisición de datos sino los sistemas propios del ordenador.

Actualmente las industrias buscan un término medio de flexibilidad a las modificaciones y un coste reducido en el mantenimiento. Es decir, un sistema robusto con gran capacidad de modificaciones debido al cambio constante en los mercados y a la necesidad de actualizar el producto para fidelizar a su clientela.

Por ese motivo, se descarta la lógica cableada, ya que es poco útil en los modelos actuales de mercado, además no permite la conexión a distancia con el resto de sedes, la conexión a de ser local.

El principal motivo por el que se a descartado la implementación por medio de microcontroladores es por no poder gestionar ni mostrar datos en tiempo real del proceso. Tampoco permite la conexión remota de la máquina sin añadir elementos que permitan una conexión a red o un servidor web.

La tarjeta de adquisición de datos se descarta automáticamente debido a que esta actuaría solo ante la válvula, pero el resto de elementos necesitaríamos controlarlos con otra tecnología. La tecnología más barata sería la cableada con contactares. La conexión remota sí podría darse desde cualquier parte de la factoría a través del PC, pero esta solución es bastante cara.

Por ultimo se ha seleccionado la implementación por PLC debido a su fácil mantenimiento, el coste reducido de modificación de la instalación, pudiendo realizarse unicamente con la programación del autómata sin añadir elementos externos, y la facilidad de conexión remota desde distintas sedes en diferentes áreas geográficas.

1.6.2 Puesta en marcha

La puesta en marcha de este proceso fue descrito en la subsección 1.5.2, pero en este punto se va a pasar a explicar el panel de operador diseñado para este proyecto. Este cumple con el código de colores recogido el artículo 53 de la normativa aplicable a las normas técnicas de prevención. La información se recoge en la tabla 1.1, en la que podemos deducir que los pulsadores verdes son de marcha y los rojos son de paro. El interruptor con enclavamiento mecánico normalmente cerrado constituirá el paro de emergencia o "seta", representado en la figura con el botón ”STOP”. Por otro lado, las luminarias de nivel máximo y mínimo se representan mediante dos rectángulos ”NIVEL MAX” y ”NIVEL MIN” respectivamente y serán de color azul. Una tercera luminaria de color blanco indicará que la máquina tiene tensión. Esta información queda recogida de manera esquemática en la figura 1.3

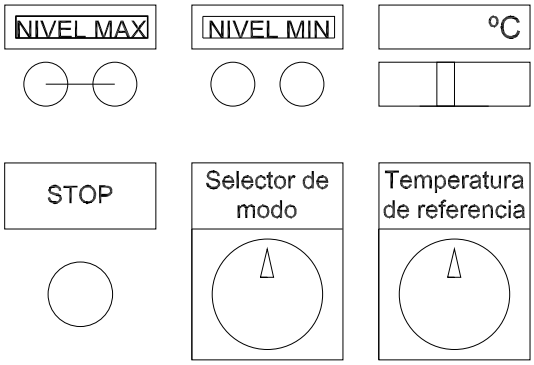

Figura 1.3: Panel de operador diseñado para esta planta

1.7 Material necesario

El material ha sido cedido por el departamento de automatización del campus de Alcoy, perteneciente a la Universidad Politécnica de Valencia, y por la empresa Airbus Helicopters España, donde he realizado las practicas de empresa. La universidad ha cedido todo el material a excepción de la licencia del Tia Portal 14 adquirida mediante licencias de estudiante o convenios con las empresas de los programas.

1.7.1 Software

En el presente punto se pasara a mostrar las principales características de los software instalados y necesitados para el desarrollo del proyecto. Estos software son los necesarios para realizar el estudio de suficiencia del autómata, pero no son los que la empresa tendría que tener instalados necesariamente.

TIA Portal V14

Este software es el necesario para conectarse al autómata y modificar la programación del mismo. Será imprescindible la licencia para la empresa si consideran modificar el programa para implementar futuras mejoras del sistema, siempre que la modificación se realizase de forma interna. Los requisitos mínimos para la instalación del programa son los mostrados en la tabla 1.2. Existen ordenadores servidos por la empresa Siemens que cuentan con todas las licencias y características extras, imposibles de adquirir con las licencias comerciales o profesionales que ellos venden.

Como se puede observar en la tabla 1.2 el sistema operativo Windows 10, no esta contemplado ya que este programa no puede instalarse en él. Es cierto que existe una actualización SP1 que permite la instalación de este programa en el sistema operativo, pero esta licencia no es la facilitada por Airbus Helicopters para la realización del programa. Esto puede parecer un inconveniente del proyecto, ya que hoy en día todos trabajamos con este sistema operativo o estamos actualizándolos para ello, pero no es crítico ya que solo necesitaríamos este sistema operativo para la programación del autómata y no para el funcionamiento. Si precisan más información sobre este programa pueden consultarlo en: enlace de aquí.

LabVIEW 2017

Este software de la casa National Instrument es el escogido para la nacionalización de la planta, para realizar los ensayos del autómata y comparar así las distintas soluciones para el sistema sin generar parones o bajadas de producción en el proceso. Además de simular con un grado alto de fiabilidad los procesos con el autómata, podremos simular las modificaciones que la empresa quiera instalar realizando un estudio del aumento de producción y los costes, aportando así mayor viabilidad y mejores técnicas para la industria 3.0. Los requisitos mínimos para la instalación del software son los mostrados en la tabla 1.3, pudiendo aumentar la información en el enlace de aquí. Nuevamente se indica que el programa no es necesario para el correcto funcionamiento del autómata.

NI OPC Server

Este programa es el encargado de comunicar nuestra planta virtualizada con nuestro autómata físico, pudiendo realizar el estudio de los distintos métodos de control avanzados para alcanzar la temperatura de manera adecuada y en el menor tiempo posible. Este programa tampoco será necesario para el funcionamiento de la planta una vez se instale el autómata, ya que la lectura de los sensores se realizará directamente, sin tener que virtualizar la señal. No obstante, será recomendable tenerlo instalado en el PC en el que se quiera almacenar los datos del sistema en tiempo real, para poder modificar o añadir más datos en cualquier momento. No será obligatorio si el ordenador tiene una base Windows ya que el protocolo es nativo del mismo. Tampoco ha de ser el programa NI OPC Server, sino que en el mercado existen numerosos gestores de servidores OPC de licencia abierta o semiabierta, como el KEPServerEX, el MATRICON OPC... entre otros.

Los requisitos mínimos de este programa son ínfimos, aunque quedan recogidos en la tabla 1.4

Matlab 2017A

Este programa no es imprescindible para la realización del proyecto, pero tiene una serie de ventajas añadidas, son las siguientes: se hace necesario para acortar los tiempos en el diseño del controlador, permite optimizar los resultados y facilita la comprobación correcta de la automatización de la planta virtualizada, pudiendo comparar los resultados del Matlab con los resultados obtenidos con el OPC mediante LabVIEW Serán necesarios los siguientes toolbox como mínimo:

- Control System Toolbox: para realizar lazos cerrados y bloques de control con los que representar la planta, los sensores, el autómata.
- Simulink: virtualización del proceso de control.
- Simulink Real-Time: generar tiempo para el control
- Parallel Computing Toolbox: Cálculo computacional en paralelo para la discretización del autómata

Se recomienda tener todos los toolbox instalados, y AQUÍ podrás ver los disponibles. Para una instalación completa del programa los requisitos mínimos son los mostrados en la tabla 1.5

Office

No será necesario el paquete ofimático al completo, incluso podría emplearse el OpenOfice de distribución libre. Para este proyecto se a utilizado la hoja de calculo Excel, que favorece la comunicación DDE, como posteriormente se explicará, ya que es de Windows. No será necesaria ninguna versión en especial pero si habrá que tener claro el idioma de instalación, ya que en función de este la configuración del OPC para la visualización en tiempo real del proceso cambiará. Algunos de los requisitos mínimos de la versión de Office que se a utilizado para este programa son los mostrados en tabla 1.6

Windows

No es imprescindible la base de Windows para la realización del proyecto, pero si es recomendable ya que el protocolo de comunicación OPC es nativo de Windows y mejorará las prestaciones de velocidad de comunicación en caso de no tener un ordenador muy potente. En cuanto a la versión de Windows recomendada para los programas citados anteriormente se va a realizar un estudio comparativo para ver el que mejor se adapta a las necesidades del estudio.

Como podemos observar en la tabla 1.7 los tres sistemas compatibles a todos los programas son Windows 7 service pack 1, Windows 8 y Windows 8.1. Por motivos de seguridad se descarta el sistema Windows 7 SP1 ya que este dejo de mantener las actualizaciones de seguridad hace algunos años. Se seleccionará Windows 8.1 por mantenerse más tiempo con actualizaciones, retrasando así la obsolescencia. Si se pudiese obtener la licencia de Tia Portal V14.1, que ya es compatible con el Windows 10, se hubiese optado por esta opción, ya el funcionamiento de Windows 8 no es el mas adecuado para este estudio.

1.7.2 Hardware

En el presente punto se va a explicar el hardware necesario para la realización del estudio de suficiencia del autómata, del servidor OPC para la gestión en tiempo real y la virtualización de la planta. Este hardware será el necesario para modificar la planta final y para realizar el estudio. Parte de este hardware no será necesario para el cliente, ya que gran parte de lo citado es exclusivo del estudio.

Autómata

El autómata fue seleccionado por la gran potencia que presenta, por la motivación de conocer un nuevo PLC con numerosas funciones antes de terminar los estudios, por introducirme en un nuevo entorno de programación con infinidad de ajustes y por llegar hasta la implementación de un regulador directamente en el autómata. Hasta ahora, se había estudiado como realizar los controles pero no como implementarlos, exceptuando los controles para motores de corriente continua. Las principales características del autómata se muestran en la tabla 1.8, y como se cito en la subsección 1.7.1 es de la casa Siemens, modelo S7-1200.

Para más información sobre el autómata consulten el catálogo del fabricante en la sección 1.1 haciendo ”click” aqui.

Tarjeta de red

Es una obviedad indicar que necesitamos una tarjeta de red, de hecho la configuración dada al servidor OPC estará ligada a una tarjeta de red. Por ese motivo, se proponen dos soluciones para que el cliente pueda almacenar, en Excel maestro, los datos del proceso en tiempo.

- Configurar el servidor OPC en un ordenador de la red empresarial: Esta opción tiene como ventaja la fácil modificación del servidor para añadir numerosas funcionalidades. Una desventaja a tener en cuenta sería que precisa de potencia suficiente para instalar todos los programas, citados en la subsección 1.7.1, para la realización del estudio de suficiencia del autómata, aunque más adelante el ordenador no necesite tanta potencia.

- Utilizar una tarjeta de red USB: esta opción tiene una gran versatilidad, ya que se podrá realizar el estudio de suficiencia en un ordenador especifico con la potencia suficiente para todos los programas citados en la subsección 1.7.1, y luego ceder la tarjeta de red a modo de "mochila" o licencia para que el Excel maestro funcione en el ordenador que se desee. Tiene por inconveniente la facilidad de extraviarlo, ya que hoy en día en el mercado existen numerosos modelos de pequeño tamaño. Otro inconveniente es la necesidad de una red no solo cableada para el autómata, sino wifi para este receptor.

- Tarjeta de red compatible: existe la opción de programarlo todo con una tarjeta de red compatible con el ordenador del cliente, ya que así la red cableada tiene mayor velocidad, y existe menor probabilidad de perderla. Por otro lado tiene como inconveniente el ser solo compatible para el ordenador que tenga la tarjeta.

Para la realización de este proyecto se ha seleccionado la configuración del servidor OPC con una tarjeta de red cableada y común a la mayoría de conectores a la placa base de ordenadores del mercado. Pudiendo conectar esta, a su vez, a la mayoría de torres. La marca de la tarjeta de red es Realtek con conexión a la placa base PCIe de la familia GBE Family Controller. La versión del controlador es 8.38.15.2015 tal y como se puede observar en la figura 1.4

Switch

Este elemento no es imprescindible para el funcionamiento del servidor, pero si queremos que el ordenador se conecte a la red y a su vez al autómata necesitaremos un "ladrón" que interconecte todo. Lo estándar es que sean de 4 puertos, sobrando uno para futuras ampliaciones. Cualquier switch puede ser empleado, pero si no queremos que no sea limitante para la velocidad de la red, este deberá ser al menos de la misma velocidad que la tarjeta de red descrita en subsección 1.7.2. En este caso de 10/100/1000.

Monitor

Tras analizar los programas necesarios para la realización de este proyecto y viendo la resolución mínima de pantalla que algunos presentan, necesitaremos un monitor que al menos pueda tener 1200x800. La conexión del monitor no será relevante siempre que la tarjeta gráfica del ordenador tenga el puerto necesario. De no cumplir con este requisito, a la hora de programar en el Tia Portal V14 se quedarán ocultos numerosos menús que dificultarán la labor del ingeniero. Esta limitación genera un grado más de dificultad a este proyecto debido a que el software indica donde existe el fallo, pero puede quedar oculto en el menú y sin la posibilidad de modificarlo. Para solucionar este problema, por la imposibilidad de obtener un nuevo monitor con la resolución adecuada, se cambió la resolución y se empleó la conexión de escritorio

Figura 1.4: Datos de configuración de la tarjeta de RED

remoto a otro ordenador que, aunque no cumplía con las limitaciones para los programas, tenia la resolución adecuada para poder trabajar.

Tabla 1.1: Código de colores para creación del panel de operador

Pulsadores		
Color	Función	Ejemplo de utilización
Rojo	Parada o Parada de emergencia	Parada de un motor, parada de elementos de la máquina, parada del ciclo (si el operario acciona el botón mientras el ciclo esta en curso, la máquina parará una vez el ciclo haya finalizado)
Amarillo	Marcha de un retroceso fuera del proceso normal de trabajo o marcha de un movimiento para la eliminación de una condición. Peligro.	Retroceso de elementos de máquina al punto de partida del ciclo, en caso de que aún no estuviese este acabado. El accionamiento del pulsador amarillo puede retirar de vigencia otra función anteriormente seleccionada.
Blanco o azul claro	Toda función que no se engloba en ninguno de los colores anteriores.	Mando de funciones auxiliares que no dependen directamente del ciclo de trabajo. Desenclavamiento o reposición de relés de contactores.
Luminarias		
Rojo	Condiciones que necesitan una acción inmediata del operario.	Parar la máquina inmediatamente o indicación de que la máquina ha parado por un elemento de protección.
Amarillo	Advertencia o atención.	Alguna magnitud se aproxima al valor límite permitido o máquina en ciclo automático.
Verde	Máquina dispuesta.	Máquina dispuesta para funcionar: todas las funciones auxiliares en marchan (posición de partida y presión, tensión corriente...) en los puntos iniciales y límites normales de funcionamiento. Fin de ciclo y maquina lista para ponerse en marcha.
Blanco (claro)	Circuito con tensión.	Interrupción principal en posición cerrado. Elección de velocidad o sentido de giro.
Azul	Cualquier significado no previsto en los colores anteriores.	Selector en posición ajuste. Una unidad adelantada de su posición de partida. Avance lento de un carro o una unidad.

Tabla 1.2: Requisitos mínimos del ordenador para el programa Tia Portal V14

Procesador	Intel core i5-3320M $3,3GHz$ o superior
RAM	$6GB$ o más
Disco duro	$300GB$ o más
Resolución de pantalla	1920 x 1080
Sistema operativo	Windows 7 SP1 (64 bit) o Windows 8 o 8.1 (64 bit)

Tabla 1.3: Requisitos mínimos para el programa LabVIEW 2017

Windows	
Procesador	Pentium 4M o equivalente para 32 bit o Pentium 4 G1 o equivalente para 64 bit
RAM	$1GB$ o mas
Resolución de pantalla	1024 x 768
Sistema operativo	Windows 10/8.1/8/7 SP1 (32- and 64-bit) o Windows Server 2012 R2 (64-bit) o Windows Server 2008 R2 SP1 (64-bit)
Espacio de disco	$5GB$
MAC OS X	
Procesador	Intel-based procesador
RAM	$2GB$ o mas
Resolución de pantalla	1024 x 768
Sistema operativo	OS X 10.11 o 10.12
Espacio de disco	$1{,}4GB$ sin drivers
Linux	
Procesador	o Pentium 4M o equivalente para 32 bit o Pentium 4 G1 o equivalente para 64 bit
RAM	$1GB$ o mas
Resolucion de pantalla	1024 x 768
Sistema operativo	Red Hat Enterprise Linux Desktop + Workstation 6.5 or later, open SUSE LEAP 42.1, open SUSE Leap 42.2, Scientific Linux 6.5 or later, CentOS 7
Espacio de disco	$2{,}2GB$ sin drivers

Tabla 1.4: Requisitos mínimos del ordenador para el programa NI OPC Server

Procesador	Intel Pentium 4 Processor
RAM	$512MB$ o mas
Disco duro	$32MB$ o mas
Resolución de pantalla	Super VGA 800×600
Sistema operativo	sistemas de 32 bit o superior

Tabla 1.5: Requisitos mínimos del ordenador para la instalación del Matlab 2017A

Procesador	Intel o AMD x86-64 Recomendado AVX2
RAM	$4GB$, recomendado $8GB$
Disco duro	$4 - 6GB$ para instalación mínima, $22GB$ para instalación completa
Resolución de pantalla	No necesaria, recomendado para aceleración $1GB$ GPU
Sistema operativo	Cualquier sistema superior al W7 SP1

Tabla 1.6: Requisitos mínimos del ordenador para la instalación del paquete ofimático Office

Procesador	32-64 bit a $1GHz$
RAM	$4GB$, recomendado $2GB$
Disco duro	$3GB$ para instalación completa
Resolución de pantalla	1280x800
Sistema operativo	Windows 7 SP1 o superior

Tabla 1.7: Versiones de Windows compatibles con los programas necesarios

	VERSIONES DE WINDOWS					
Programas	Wxp	W7	W7 SP1	W8	W8.1	W10
TIA Portal V14			X	X	X	
LabVIEW 2017			X	X	X	X
NI OPC Server	X	X	X	X	X	X
Matlab 2017A			X	X	X	X
Office			X	X	X	X

Tabla 1.8: Características principales del autómata S7-1200 de Siemens

Dimensiones físicas	$110x100x78$
Memoria de trabajo	$50Kb$
Memoria de carga	$2Mb$
Memoria remanente	$2Kb$
E/S Digitales	14 entradas y 10 salidas
E/S Analógicas	2 entradas
Ampliación con módulos de señal	8
Signal Board	1
Módulos de comunicación	3 ampliables al lado izquierdo
contactores rápidos	6
salidas de impulsos	2
Memory card	SIMATIC Memory Card (opcional)
Tiempo de respaldo del reloj	de 10 a 6 días
PROFINET	1 puerto de comunicación ethernet
Velocidad de ejecución de funciones matemáticas con números reales	$18\mu s/instruccin$

Capítulo 2

Modelado de la planta

En el presente capítulo se modelará la planta de diferentes maneras utilizando técnicas de diversos autores. Se estudiará el proceso desde un punto de vista físico para obtener las distintas ecuaciones que definan el proceso y las diferentes formas de representar estas ecuaciones. La carga matemática de este capítulo será importante ya que se realizarán cambios y se hablarán de técnicas en las que no entraremos en detalle, pero que son necesarias para seguir el hilo del proceso.

*El objetivo principal de este capítulo es obtener las ecuaciones necesarias para la planta virtualizada con la que realizar ensayos y diseños para obtener un proceso automático. Cada modelado se utilizará para una función distinta y en un programa distinto de los ya citados en la subsección 1.7.1. Para el diseño de esta planta hemos tomado de referencia el trabajo de fin de grado (***tfgencamisado***) de un alumno.*

2.1 Planta a virtualizar

Si analizamos la planta de la cual hablamos en la sección 1.5, encontramos dos grandes bloques a tener en cuenta, uno sera la propia planta formada por la caldera, el tanque y la camisa, es decir, **el tanque encamisado** y otro sera los elementos que añadamos para el control, es decir, **sensores y actuadores**. El primero de ellos se realiza mediante ecuaciones termodinámicas con las que estudiaremos el intercambio de masa y energía. El otro gran punto es la elección de los sensores y actuadores, de los que necesitaremos conocer la ganancia estática, las constantes de tiempo... Con todos estos datos seremos capaces de realizar unos modelos matemáticos que representen la planta contra la que ejecutaremos los algoritmos de control, y la planta contra la que diseñaremos los reguladores, utilizando diversas técnicas para finalmente instalar el que mejor prestaciones de, o el que mejor se adapte a las tecnologías que el cliente presenta.

2.1.1 Tanque encamisado

Tal y como se describió en la sección 1.5 la planta esta formada por un tanque que necesitamos calentar a una temperatura constante para, posteriormente, añadir este líquido al proceso. Ahora analizaremos la planta desde un punto de vista mas físico intentando modelizar las ecuaciones matemáticas que definan el funcionamiento del mismo. Para ello, necesitaremos tener en cuenta algunas concesiones y aproximaciones:

- Los volúmenes son constantes, tanto el del tanque (V) como el de la camisa (V_c)
- El caudal del líquido B circulante por la camisa (Q_c) será una variable del sistema
- La temperatura del líquido B, o temperatura de la camisa (T_c) será otra variable del sistema
- El fluido de la camisa será agua presurizada, por lo que el rango de temperaturas que circulen por la camisa será mayor sin tener un cambio de estado
- La mezcla perfecta de los volúmenes de control manteniendo la temperatura constante en entrada y salida
- Rango de temperaturas lo suficientemente pequeño como para suponer la densidad (ρ) y la capacidad calorífica (C_p) constantes
- UA constante para el intercambio de calor

En régimen normal de funcionamiento o régimen permanente las variables tendrán un valor fijo, determinado por la naturaleza de los componentes. Estos valores son:

- $\rho = 1000 kg/m^3$
- $UA = 4180 kJ/s^oC$
- $C_p = 4{,}18 kJ/kg^oC$
- $V_c = 2m^3$
- $V = 5m^3$
- $Q_f = 0{,}25^{m^3}/_s$
- $Q_c = 0{,}05^{m^3}/_s$
- $T_p = 50^oC$
- $T_c = 55^oC$
- $T_f = 30^oC$
- $T_{ce} = 155^oC$

Ahora, pasaremos al estudio de los volúmenes de control por separado, estudiando tanto el balance de materia como de energía y obteniendo así las ecuaciones que definan el proceso.

Distinguiremos entre volumen de control 1 como el del tanque, y el volumen de control 2 como el de la camisa.

Volumen de control 1

Con estas consideraciones citadas anteriormente (subsección 2.1.1) realizaremos un **balance de materia** entre el flujo másico de la entrada ($\dot{m}_f$), y el flujo másico de producto ($\dot{m}_p$). Obtendremos la siguiente ecuación:

$$\dot{m}_f - \dot{m}_p = 0 \tag{2.1}$$

Teniendo en cuenta la definición de flujo másico como el producto del caudal volumétrico por densidad (ρQ) y sustituyendo en la ecuación (2.1) obtendremos la siguiente expresión:

$$\rho Q_f - \rho Q_p = 0$$

$$Q_f = Q_p \tag{2.2}$$

Si ahora realizamos el **balance de energía** sabiendo que es donde el intercambio de calor se lleva acabo, obtendremos la siguiente expresión:

$$\dot{m}_f \dot{h_f} - \dot{m}_p \dot{h_p} + UA(T_{cs} - T_p) = \frac{dU_{vc1}}{dt} \tag{2.3}$$

Sabiendo que $U = \dot{m}u$ y sustituyendo en la ecuación anterior (2.3) obtendremos:

$$\dot{m}_f \dot{h_f} - \dot{m}_p \dot{h_p} + UA(T_{cs} - T_p) = \frac{d(C_p \rho V T_p)}{dt}$$

$$\dot{m}_f \dot{h_f} - \dot{m}_p \dot{h_p} + UA(T_{cs} - Tp) = \overline{C_p} \rho \overline{V} \frac{dT_p}{dt} \tag{2.4}$$

Si aplicamos la definición de h en la ecuación (2.4) obtendremos:

$$h = h^o + C_p(T - T^o)$$

$$\rho Q_f \overline{C_p}(T_{f1} - T_p) + UA(T_{cs} - T_p) = \overline{C_p} \rho \overline{V} \frac{dT_p}{dt} \tag{2.5}$$

Volumen de control 2

Seguidamente realizamos el **balance de materia** al volumen de control 2 y obtendremos, nuevamente, los mismos resultados que en el balance de control 1. Siendo la expresión idéntica a la reflejada en la ecuación (2.2). Si realizamos ahora el **balance de energía** en el volumen de control 2 nos daremos cuenta, al plantear la ecuación, que el signo del calor intercambiado ha de ser el contrario. Siguiendo el criterio de signos aplicados en la asignatura de motores térmicos, este ha de ser negativo ya que es el fluido el que cede el calor. De este modo la ecuación final quedará parecida a la ecuación (2.5) con las siguientes modificaciones:

$$\rho Q_{ce}\bar{C_p}(T_{ce} - T_{cs}) - UA(T_{cs} - T_p) = \bar{C_p}\rho\bar{V_c}\frac{dT_s}{dt} \tag{2.6}$$

2.1.2 Modelo de sensores y actuadores

Con el fin de no complicar el modelo matemático se buscaran sensores y actuadores de primer orden. Sería conveniente ajustar la ganancia de los mismos a lo representado en las siguientes ecuaciones:

- Actuadores: En este sistema las válvulas serán los actuadores, pero en especial la válvula que regula el caudal de la camisa del tanque (válvula B) que es la que tendremos en cuenta para el control y para la virtualización de la planta. Este elemento ha de quedar definido por la siguiente ecuación:

$$-Q_c + K_c U = \tau_c \dot{Q_c}$$

- Sensores: En este sistema existirán dos sensores de temperatura, uno del tanque y otro de la camisa. Ambos los consideraremos de primer orden y tendrán que estar definidos por las siguientes ecuaciones respectivamente:

$$K_{em}T_p - T_{pm} = \tau_m \dot{T_{pm}}$$

$$K_{em2}T_c - T_{cm} = \tau_{m2} \dot{T_{cm}}$$

En estas ecuaciones están representadas las ganancias estáticas como K y las constantes de tiempo como τ, independientemente del subíndice que acompañe. Será necesario conocer los valores, ya que esto cambiará las futuras constantes del regulador, así como el diseño de los mismos. Si no se conociera el valor de estas constantes habría que realizar un ensayo sobre estas. Es muy importante a la hora de simplificar el diseño del regulador que los elementos sean de primer orden, ya que añadir mas polos o zeros al sistema complicará mucho el diseño y la implementación del regulador sobre la planta.

Como se puede observar, estamos contando ya con un sensor que hasta ahora no habíamos tenido en consideración, se trata del sensor de temperatura de la camisa del tanque. La implementación de este será importante si deseamos controles avanzados del tipo cascada o controles en los que contemos con lazos independientes.

2.2 Modelo en el espacio de estados

Algunos autores recomiendan, incluso cuando no se va a realizar el control en el espacio de estados, implementar los modelos de esta forma ya que el indice de error es menor. Es cierto que la complejidad al tratar con este modelado de la planta es mayor, pero si lo utilizamos como la planta contra la que realizar los ensayos tendremos unos resultados más exactos. Para ello debemos comenzar con las ecuaciones obtenidas del balance de energía (Ecuación 2.5 y Ecuación 2.6) y los sensores y actuadores descritos en la subsección 2.1.2 obteniendo las siguientes ecuaciones:

- Balance de energía del volumen de control 1:

$$K_A Q_f (T_f - T_p) + K_B (T_c - T_p) = \tau_A \dot{T_p} \tag{2.7}$$

- Balance de energía del volumen de control 2:

$$K_A Q_c (T_{ce} - T_c) + K_B (T_c - T_p) = \tau_B \dot{T_c} \tag{2.8}$$

- Modelado de la válvula B:

$$-Q_c + K_c U = \tau_c \dot{Q_c} \tag{2.9}$$

- Modelado del sensor del tanque:

$$K_{em} T_p = \tau_m \dot{T_{pm}} + T_{pm} \tag{2.10}$$

- Modelado del sensor de la camisa:

$$K_{em2} T_c = \tau_{m2} \dot{T_{cm}} + T_{cm} \tag{2.11}$$

- Donde:
  - $K_A = \rho C_p$
  - $K_B = UA$
  - $\tau_A = C_p V \rho$
  - $\tau_B = C_p V_c \rho$

Si, a continuación, reescribimos las ecuaciones (Ecuación 2.7, Ecuación 2.8, Ecuación 2.9, Ecuación 2.10 y Ecuación 2.11) para obtener unas expresiones mas cómodas para el espacio de estados, obtendremos las siguientes:

$$\dot{x_1} = \frac{1}{\tau_A}(K_B + K_A u_2)x_1 + K_{21}x_2 + K_{11}u_2u_3$$

$$\dot{x_2} = K_{22}x_1 - K_{22}x_2 - K_{32}x_2x_3 + K_{32}x_3u_4$$

$$\dot{x_3} = -\frac{1}{\tau_C}x_3 + K_{43}u_1$$

$$\dot{T_{pm}} = \frac{K_{em}}{\tau_m}x_1 - \frac{1}{\tau_m}T_{pm}$$

$$\dot{T_{cm}} = \frac{K_{em2}}{\tau_{m2}}x_2 - \frac{1}{\tau_{m2}}T_{cm}$$

Si montamos en simulink el diagrama de bloques, excluyendo los sensores, obtendremos el resultado mostrado en la figura 2.1

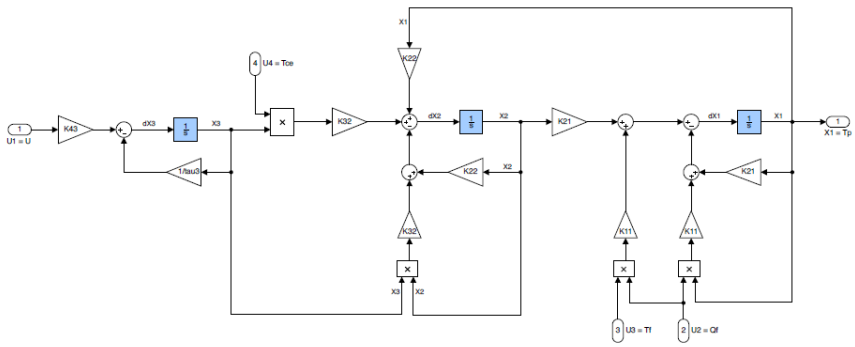

Figura 2.1: Modelo en lazo abierto del sistema no lineal

2.3 Discretización del modelo no lineal

En la presente sección se va a definir el método seguido para la discretización del sistema debido a que el programa utilizado para la virtualización de la planta así lo requiere. Es un punto problemático más, puesto que el algoritmo que utilizamos se ejecuta un número determinado de veces en intervalos fijos de tiempo, generando así una señal discreta. Estas señales son mas difíciles de controlar, pero este modelado no es completamente real, ya que la planta real sobre la que finalmente se montara el sistema es continua, y el regulador así la verá, pero para la virtualización necesitaremos tener las ecuaciones que deduciremos a continuación.

Además de todo esto, deberemos tener en cuenta que el servidor OPC también trabaja con señales discretas, por lo que es conveniente que la aproximación sea lo mas cercana posible, y obtener un numero representativo de puntos. Otra característica que debemos conocer para la futura implementación, y que sera clave en el servidor OPC, será los tiempos de muestreo, tanto de la planta virtual como del servidor, que no han de coincidir como veremos mas adelante, y que serán los que determinen la velocidad final de nuestro sistema.

2.3.1 Método genérico de discretización, base matemática

Para poder implementar el algoritmo necesitamos ecuaciones discretas para el sistema del modelo no lineal que nos permitirá tomar valores estimados de las señales. Si aplicamos la aproximación de derivada según Euler obtendremos aproximaciones discretas. Esta aproximación la podemos observar en la siguiente ecuación:

$$x'(t) = \frac{dx(t)}{dt} = \lim_{\Delta_x \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} \tag{2.12}$$

Si aplicamos el intervalo de tiempo (t) al periodo de muestreo e integramos por el método de Euler la Ecuación 2.12 obtendremos la siguiente expresión:

$$x'(k) \simeq \frac{x(k+1) - x(k)}{T} \tag{2.13}$$

Si despejamos de la Ecuación 2.13 la expresión $x(k+1)$ obtendremos:

$$x(k+1) = x(k) + Tx'(k) \tag{2.14}$$

Si recurrimos ahora a la definición de un sistema de primer orden en el espacio de estados obtendremos la siguiente expresión genérica:

$$\dot{x} = ax + u \tag{2.15}$$

Si extraemos de la ecuación (2.13) los términos de la ecuación (2.15)

$$\frac{x(k+1) - x(k)}{T} = ax(k) + u(k) \tag{2.16}$$

En esta ecuación (2.16) queda representado el instante de x en el instante posterior al actual. Claro está que se trata de una estimación, pero que se ajusta con bastante fiabilidad al proceso real que queremos virtualizar. Si reajustamos la ecuación nuevamente obtenemos:

$$x(k+1) = x(k)T(ax(k) + u(k)) \tag{2.17}$$

Si observamos el proceso seguido, nos damos cuenta de que con este método podemos representar en tiempo discreto cualquier función continua cuya trayectoria o trazado pueda representarse mediante la ecuación (2.15), siempre que conozcamos el estado inicial ($x(0)$) y la entrada al sistema ($u(k)$). Uno de los principales problemas de este método es la elección de la variable del tiempo de muestreo (T), ya que de no ser representativa la función distara bastante de la realidad.

2.3.2 Implementación de la discretización sobre nuestro sistema

Si aplicamos los conceptos de discretización descritos en la subsección 2.3.1 obtendremos para nuestro sistema las siguientes ecuaciones:

$$x_1(k+1) = x_1(k) + T\left(\frac{-1}{\tau_a}\Big(K_B + K_A u_2(k)x_1(k) + K_{21}x_2(k) + K_{11}u_2(k)u_3(k)\Big)\right)$$

$$x_2(k+1) = x_2(k) + T\Big(K_{22}x_1(k) - K_{22}x_2(k) - K_{32}x_2(k)x_3(k) + K_{32}x_3(k)u_4(k)\Big)$$

$$x_3(k+1) = x_3(k) + T\left(\frac{-1}{\tau_c}x_3(k) + K_{43}u_1(k)\right)$$

$$T_{pm}(k+1) = T_{pm}(k) + T\left(\frac{K_{em}}{\tau_m}x_1(k) - \frac{1}{\tau_m}T_{pm}(k)\right)$$

$$T_{cm}(k+1) = T_{cm}(k) + T\left(\frac{Kem2}{\tau_m 2}x_2(k) - \frac{1}{\tau_m 2}T_{cm}(k)\right)$$

Como ya citamos en la subsección 2.1.1 las variables del estado son x_1 correspondiente a la temperatura del tanque (T_p), x_2 correspondiente a la temperatura de la camisa del tanque (T_c) y x_3 correspondiente al caudal de la camisa (Q_c). La entrada de control es u_1 correspondiente a la tensión (U), y las perturbaciones que modificarán el comportamiento del controlador son u_2 correspondiente a las modificaciones de caudal de salida del tanque (Q_f), u_3 correspondiente a las modificaciones de temperatura (T_f) de igual modo que u_4 corresponde a las modificaciones de temperatura de la caldera (T_{ce}). De igual modo implementaremos en el programa Simulink la planta, tal y como se hizo en la figura 2.1, pero esta vez de modo discreto. Esto lo podemos observar en la figura 2.2

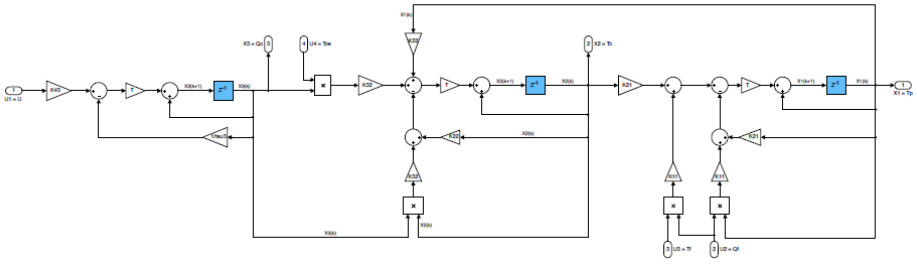

Figura 2.2: Modelo en lazo abierto del sistema no lineal discretizado

2.4 Linealización del modelo entorno al punto de funcionamiento

Puesto que la mayoría de técnicas de control se basan en modelos lineales, a pesar de que la semejanza con la realidad es menor, vamos a pasar a modelizar en torno al punto de funcionamiento el modelo del tanque, del cual podremos extraer futuras técnicas de control para implementarlas en el autómata. Con este diseño lineal, podremos emplear infinidad de técnicas de control, aunque estas se apliquen luego sobre la planta virtual discreta, pero que simplifiquen así el diseño.

Otro punto a favor para la linealización del sistema entorno al punto de funcionamiento es la mejora de la velocidad de computación del programa de diseño, ya que trata el sistema como uno continuo en lugar de uno discontinuo con discontinuidades de salto finito. Es cierto que se asemejará menos al modelo descrito en **??**, pero el regulador que finalmente se implante en el autómata trabajará sobre un sistema lineal real. Por ello es también conveniente realizar el diseño sobre el sistema real y posteriormente testarlo sobre la planta virtual que más se asemeje a la real.

Para esta linealización es necesario conocer el polinomio de Taylor, que es una seria infinita, y conocer que cuanto mayor sea el grado del polinomio más se aproximará con la ecuación real. En nuestro caso se ha elegido el segundo grado de aproximación. Podemos ver gráficamente la aproximación de una función exponencial en la figura 2.3, o la serie en forma genérica del polinomio en la siguiente ecuación:

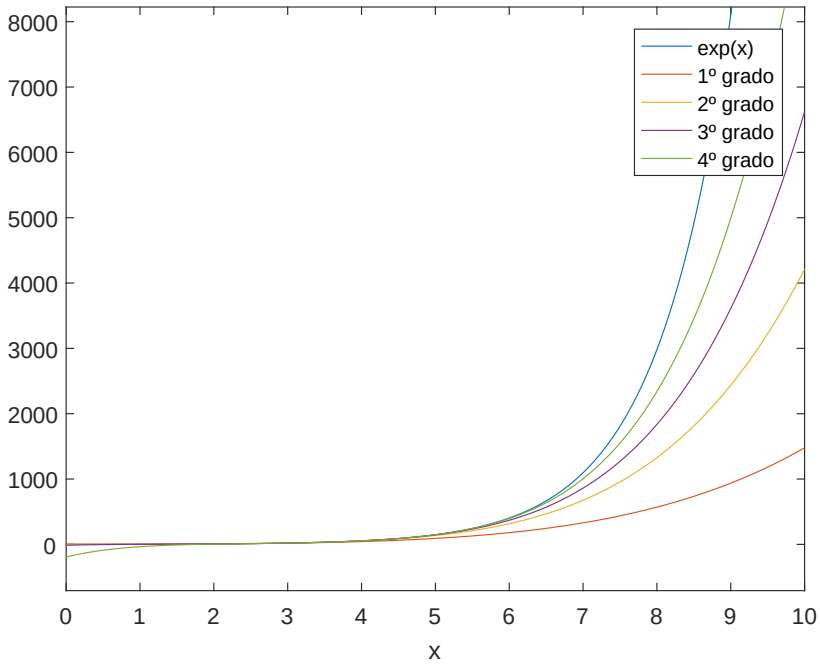

Figura 2.3: Aproximación del cuarto grado de la serie de Taylor a una curva exponencial

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + ... + \frac{f^k(a)}{k!}(x-a)^k + h_k(x)(x-a)^k$$

Como se puede observar, si centramos el punto de funcionamiento entorno a los valores de x comprendidos entre 1 y 5 de la figura 2.3, la aproximación es bastante fiable. Sin embargo, por los extremos difiere demasiado como para considerarlo aceptable. Por ello, se usará la variante de Pade, es decir, igualándolo a zero.

2.4.1 Modelo lineal

Si aplicamos el polinomio de Taylor a las ecuaciones discretas podremos linealizar los modelos y, aplicando las constantes del punto de funcionamiento, obtener el valor en régimen nominal. De este modo se obtienen los sistemas que necesitamos para el diseño del regulador. Claro está que este sistema no se asemejará con tanta precisión al descrito en la sección 2.3 o sección 2.2, pero nos permitirá tener funciones continuas con las que aplicar técnicas de control fácilmente.

Tal y como se explicó en la subsección 2.1.1 y subsección 2.1.1, linealizaremos por volúmenes de control, para luego aplicar el álgebra de bloques y crear un sistema más fácil de implementar y de tratar.

Volumen de control 1

Como el balance ya es lineal, debemos aplicar únicamente la desviación de cada término, quedando la siguiente ecuación:

$$Q_f - \overline{Q_f} = Q_p - \overline{Q_p}$$

$$Q'_f = Q'_p \tag{2.18}$$

Si ahora analizamos el **balance de energía** vemos que no es lineal, ya que existen productos de distintas variables. Si aplicamos el polinomio de Taylor se obtendrá la siguiente ecuación:

$$\rho\overline{C_pQ_f}(T_{f1}-\overline{T_{fi}}+\rho\overline{C_pQ_f}(T_{f1}-\overline{T_p})-\rho\overline{C_pQ_f}(T_p-\overline{T_p})+\rho\overline{C_p}(Q_f-\overline{Q_f})(\overline{T_{f1}}-\overline{T_p})+UA(T_{cs}-T_p)=\rho\overline{VC_p}\frac{dT_p}{dt} \tag{2.19}$$

Si a la ecuación (2.19) le restamos las entalpías, sustituimos el **balance de energía** y simplificamos, obtendremos el siguiente resultado:

$$\rho Q'_f\overline{C_p}(\overline{T_{f1}}-\overline{T_p})+\rho\overline{Q_fC_p}(T'_{f1}-T'_p)+UA(T'_{cs}-T'_p)=\rho\overline{VC_p}\frac{dT'_p}{dt} \tag{2.20}$$

Volumen de control 2

De igual modo que en volumen de control anterior, pasamos a linealizar el segundo volumen de control. Partimos de las ecuaciones de la subsección 2.1.1 y continuando con el mismo proceso citado en la subsección 2.4.1 obtendremos las siguientes ecuaciones:

El primer paso es linealizar el **balance de materia**:

$$Q'_{ce} = Q'_{cs} \tag{2.21}$$

El segundo paso, al igual que en el apartado anterior, es obtener la linealización del **balance de energía**:

$$\rho Q'_c \overline{C_p}(\overline{T_{ce}} - \overline{T_{cs}}) + \rho\overline{Q_{ce}C_p}(T'_{ce} - T'_{cs}) - UA(T'_{cs} - T'_p) = \rho\overline{V_c C_p}\frac{dT'_{cs}}{dt} \tag{2.22}$$

2.4.2 Eliminación de las ecuaciones integro-diferenciales

Ahora que tenemos expresadas las funciones de la planta mediente un metodo de modelado lineal, y según lo descrito en la subsección 2.4.1, pasaremos a eliminar las funciones diferenciales aplicando la transformada de Plaplace. Esto nos permitirá simplificar las ecuaciones más aun y mejorar la futura implementación en el programa de cálculo Matlab, con el fin de diseñar los controladores. Para ello, es conveniente recordar las siguientes transformaciones:

- **Cambio de variable:** $X(t)' = X(s)$
- **Cambio de derivada:** ${}^{dX(t)'}/_{dt} = SX(s)$

Volumen de control 1

Si aplicamos las transformadas citadas en la subsección 2.4.2 a las ecuaciones de la subsección 2.4.1 obtendremos los siguientes resultados:

- Aplicando la transformada a la ecuación (2.18) obtendremos:

$$Q_f(s) = Q_p(s) \tag{2.23}$$

- Amplificamos la transformada de Laplace a la ecuación (2.20)

$$\rho Q_f(s)\overline{C_p}(\overline{T_{f1}} - \overline{T_p}) + \rho\overline{Q_f C_p}(T_{f1}(s) - T_p(s)) + UA(T_{cs}(s) - T_p(s)) = \rho\overline{VC_p}sT_p(s) \tag{2.24}$$

Como la ecuación (2.24) sigue siendo difícil de tratar se van a agrupar términos en función de cada variable, intentando despejar así la salida ($T_p(s)$. De este modo obtenemos:

$$T_p(s) = \frac{[\rho\overline{C_p}(\overline{T_{f1}} - \overline{T_p})]Q_f(s)}{\rho\overline{VC_p}s + UA + \rho\overline{Q_f C_p}} + \frac{[\rho\overline{C_p}(\overline{Q_f}]T_{f1}(s)}{\rho\overline{VC_p}s + UA + \rho\overline{Q_f C_p}} + \frac{[UA]T_c(s)}{\rho\overline{VC_p}s + UA + \rho\overline{Q_f C_p}} \tag{2.25}$$

Si expresamos las ecuaciones anteriores como ecuaciones de transferencia de primer orden, obtendremos las siguientes ecuaciones:

$$T_p(s) = \frac{\frac{\rho\overline{C_p}(T_{f1}-\overline{T_p})}{UA+\rho\overline{C_pQ_f}}}{\frac{\rho\overline{VC_p}s+UA+\rho\overline{Q_fC_p}}{UA+\rho\overline{C_pQ_f}}}Q_f(s) + \frac{\frac{\rho\overline{C_pQ_f}}{UA+\rho\overline{C_pQ_f}}}{\frac{\rho\overline{VC_p}s+UA+\rho\overline{Q_fC_p}}{UA+\rho\overline{C_pQ_f}}}T_{f1}(s) + \frac{\frac{UA}{UA+\rho\overline{C_pQ_f}}}{\frac{\rho\overline{VC_p}s+UA+\rho\overline{Q_fC_p}}{UA+\rho\overline{C_pQ_f}}}T_{cs}(s) \quad (2.26)$$

Si separamos los términos como funciones de transferencia distintas y asemejarlas a funciones de primer orden donde tengamos una entrada y una salida distinta, podemos obtener:

- $GQ_f(s)$: Función de transferencia del caudal:

$$GQ_f(s) = \frac{\frac{\rho\overline{C_p}(T_{f1}-\overline{T_p})}{UA+\rho\overline{C_pQ_f}}}{\frac{\rho\overline{VC_p}s+UA+\rho\overline{Q_fC_p}}{UA+\rho\overline{C_pQ_f}}}$$

- $GT_{f1}(s)$: Función de transferencia de la temperatura de la caldera:

$$GT_{f1}(s) = \frac{\frac{\rho\overline{C_pQ_f}}{UA+\rho\overline{C_pQ_f}}}{\frac{\rho\overline{VC_p}s+UA+\rho\overline{Q_fC_p}}{UA+\rho\overline{C_pQ_f}}}$$

- $GT_{cs}(s)$Función de transferencia de la temperatura de salida de la camisa del tanque:

$$GT_{cs}(s) = \frac{\frac{UA}{UA+\rho\overline{C_pQ_f}}}{\frac{\rho\overline{VC_p}s+UA+\rho\overline{Q_fC_p}}{UA+\rho\overline{C_pQ_f}}}$$

Volumen de control 2

Realizando el mismo proceso que en la subsección 2.4.2, realizaremos los balances de control de energía y materia.

Balance de materia:

$$Q_{ce}(s) = Q_{cs}(s) \quad (2.27)$$

Balance de energía:

$$T_{cs}(s) = \frac{\rho\overline{C_p}(\overline{T_{ce}}-\overline{T_{cs}})}{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}Q_{ce}(s) + \frac{\rho\overline{Q_{ce}C_p}}{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}T_{ce}(s) + \frac{UA}{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}T_p(s) \quad (2.28)$$

Si extraemos las funciones de transferencia de primer orden obtenemos:

$$T_{cs}(s) = \frac{\frac{\rho\overline{C_p}(\overline{T_{ce}}-\overline{T_{cs}})}{\rho\overline{Q_{ce}C_p}+UA}}{\frac{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}{\rho\overline{Q_{ce}C_p}+UA}}Q_{ce}(s)+\frac{\frac{\rho\overline{Q_{ce}C_p}}{\rho\overline{Q_{ce}C_p}+UA}}{\frac{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}{\rho\overline{Q_{ce}C_p}+UA}}T_{ce}(s)+\frac{\frac{UA}{\rho\overline{Q_{ce}C_p}+UA}}{\frac{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}{\rho\overline{Q_{ce}C_p}+UA}}Q_{ce}(s) \quad (2.29)$$

Descomponiendo en funciones de transferencia más pequeñas obtenemos:

- Función de transferencia del caudal de la camisa del tanque:

$$\frac{\frac{\rho\overline{C_p}(\overline{T_{ce}}-\overline{T_{cs}})}{\rho\overline{Q_{ce}C_p}+UA}}{\frac{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}{\rho\overline{Q_{ce}C_p}+UA}}$$

- Función de transferencia de la temperatura de la camisa del tanque:

$$\frac{\frac{\rho\overline{Q_{ce}C_p}}{\rho\overline{Q_{ce}C_p}+UA}}{\frac{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}{\rho\overline{Q_{ce}C_p}+UA}}$$

- Función de transferencia de la temperatura producto:

$$\frac{\frac{UA}{\rho\overline{Q_{ce}C_p}+UA}}{\frac{\rho\overline{V_cC_p}s+\rho\overline{Q_{ce}C_p}+UA}{\rho\overline{Q_{ce}C_p}+UA}}$$

2.4.3 Diagrama de bloques

Una vez tenemos linealizado las funciones de transferencia, podemos crear el diagrama de bloques ordenado. Este diagrama será el que implementaremos en Simulink con el fin de diseñar los controladores mediante Matlab. En la figura 2.4

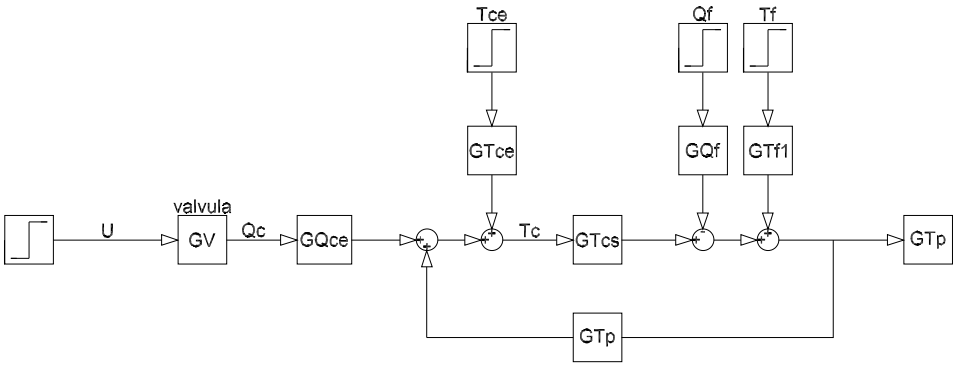

Figura 2.4: Modelo de la planta linealizada

Capítulo 3

Diseño del regulador y preparación para la virtualización

En el presente capítulo se tratará el diseño de los distintos reguladores mediante el programa Matlab, del cual hablamos en la subsección 1.7.1. Se obtendrán las condiciones de contorno para obtener la planta virtualizada y poder realizar así la simulación sobre esta con el autómata físico. Se extraerán distintos archivos que posteriormente se introducirán en el programa LabVIEW (subsección 1.7.1) para poder virtualizar la planta e incluso para exportar los archivos del regulador al programa. Por ultimo, se cargaran los distintos modelos en archivos de Simulink para la posterior simulación del regulador.

3.1 Metodología

En esta sección se va a explicar todos los pasos seguidos para la posterior implementación de la planta virtual en LabVIEW y el diseño del regulador. Se emplea este método de programación para poder tener así una planta virtual genérica en LabVIEW. En ella, únicamente debemos cambiar los parámetros iniciales citados en la subsección 2.1.1 y generar al instante tanto el regulador a implementar como la planta virtual para comprobarlo. Este método sólo será susceptible a aquellas plantas cuyo proceso pueda quedar definido conforme a lo explicado en el Capítulo 2. Estos pasos de programación facilitarán futuros estudios para la empresa y para el autor del proyecto.

Separaremos la programación en varios sub-programas, con el fin de mejorar la computación y ejecutar sólo aquellos que necesitemos en cada caso. Otro aspecto positivo es que solo deberemos modificar uno de ellos para obtener las distintas plantas virtualizadas y los distintos reguladores que deseemos implementar, ya que de este obtendremos los distintos valores nominales.

3.2 Fundamentos teóricos, matemáticos y de programación

Todos los cálculos se van a llevar a cabo con el programa Matlab, del cual hablamos en la subsección 1.7.1. Este programa tiene infinidad de posibilidades de cálculos, formas de programación, paquetes de diseño... En esta sección nos centraremos en explicar la base matemática que debemos conocer para obtener las mejores funciones y algunos términos o conceptos de programación para contar con las mejores configuraciones para el diseño de los reguladores, y la mejor forma de extraer los datos necesarios para la virtualización de la planta.

Sera conveniente no solo conocer o tener la capacidad para entender esta sección, sino ademas haber tratado con reguladores y estar familiarizado con los términos para continuar con este capitulo.

3.2.1 Método de integración

Es imprescindible configurar de una manera adecuada el método de integración para obtener un diseño y simulación adecuados del regulador. Es crucial configurar esto desde un inicio bien, puesto que un error en la integración puede falsear los datos que muestre nuestro regulador e incluso ofrecer un diseño erróneo. Es aun mas importante a tener en cuenta si la planta contra la que simulamos el controlador, o el mismo controlador son de carácter discreto, ya que la aproximaremos la derivada como una integración numérica.

Como ya se ha citado en el Capítulo 2 la planta contra la que se testara el autómata real sera una planta discreta virtualizada, es por ese motivo por lo que las simulaciones de Matlab se realizaran contra una planta discreta. El método con el que calculemos el error y la señal de control serán relevantes para una buena virtualización o un ensayo pésimo Para este proyecto se han considerado tres métodos, los cuales pasaremos a exponer ahora y mostrar como configurar en simulink.

Método forward

El método forward o método hacia adelante, parte de la definición de derivada propuesta por Euler, la cual encontraremos en la siguiente ecuación:

$$x'(t) = \frac{dx(t)}{dt} = \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t}$$

Si ahora aproximamos Δt como el periodo de muestreo T obtendremos el siguiente desarrollo matemático:

$$x'(kT) \simeq \frac{x((k+1)T) - x(kT)}{T} = \frac{x(k+1) - x(k)}{T} \tag{3.1}$$

Si aplicamos ahora la definición de integral de la derivada (en la condición de contorno de imagen de 0 igual a 0) y la ecuación (3.1) obtendremos que la aproximación de la integral corresponde a:

$$x(t) = \int_0^t x'(t)$$

$$x(k) = x(k-1) + Tx'(k-1) \tag{3.2}$$

Mediante este método crearemos rectángulos de base igual a T y una altura igual a $x'(k-1)$

Método Backward

Para este método también comenzaremos con la definición de la derivada de Euler mostrada en la subsección 3.2.1, pero aproximaremos la derivada como:

$$x'(kT) \simeq \frac{x(kT) - x((k-1)T)}{T} = \frac{x(k) - x(k-1)}{T} \tag{3.3}$$

Mediante este método generamos rectángulos donde la base sigue siendo T, y la altura $x'(K)$, pero serán menores que en el caso anterior.

3.2.2 Integración Trapezoidal

Este método es la forma mas común de aproximar la derivada de un sistema discreto. Se trata de un método denominado "trapezoidal" o "de Tustin". Esta aproximación tiene como base los métodos descritos anteriormente en esta misma sección y realizara una media arigmética de ambas. La ecuación que lo define viene dada por:

$$x(k) = x(k-1) + T\frac{x'(k) + x'(k-1)}{2} \tag{3.4}$$

Mediante este método si seleccionamos adecuadamente el tiempo de muestreo podremos obtener la derivada casi perfecta en un tiempo discreto para funciones no-continuas. Si comparamos los tres métodos mostrados en la figura 3.1 observaremos la diferencia de las áreas generadas con estos métodos.

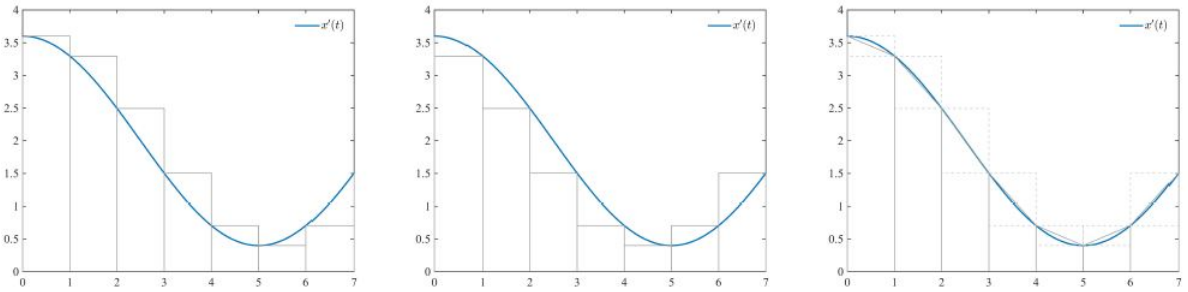

Figura 3.1: (1) método fordward, (2) método backward y (3) método trapezoidal

Implementación de los métodos en el diseño del regulador

En este punto del proyecto enseñaremos a seleccionar alguno de los métodos explicados anteriormente en esta misma sección para el calculo del diseñador. Vamos a explicar paso por paso como hemos logrado seleccionar cualquiera de los tres métodos en el programa simulink y cual a sido el que mejor resultados nos a proporcionado:

1. Seleccionamos el modelo de simulink donde queramos cambiar el tipo de aproximación de la derivada

2. Daremos doble click sobre el controlador donde queramos cambiar el método Tras esto debe aparecer una pestaña como la mostrada en figura 3.2, donde podemos observar en la parte derecha los métodos de integración

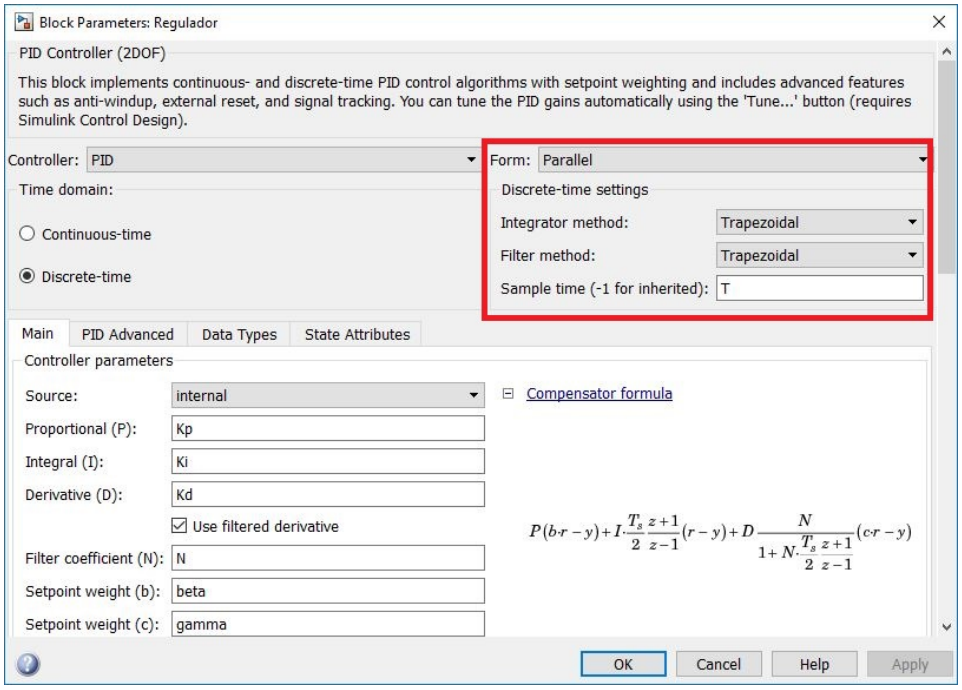

Figura 3.2: Ventana de configuración del PID

3. Seleccionamos el método deseado, en este caso recomendamos el método trapezoidal.

3.2.3 Anti-windup

El efecto windup es un comportamiento no deseado del regulador en una serie de circunstancias claves. Este comportamiento puede realizar que el regulador no funcione adecuadamente, es por ese motivo por el cual es imprescindible en aquellas situaciones en las que se observe comportamientos inadecuados de la señal de control implementar algún algoritmo que lo mejore.

Muchos autores recomiendan utilizar este tipo de algoritmos en reguladores donde contemos con las siguientes características:

- El controlador cuenta con acción integral

- La acción de control cuenta con valores limite para la regulación, es decir, el controlador satura.

- El error se mantiene en valores demasiado altos durante mucho tiempo. Este punto se cumplirá ya que los cambios de temperatura que el proceso necesita varían en gran medida.

Gracias a implementar las mejoras de diseño por medio del anti-windup se conseguirán transiciones mas suaves y un mejor comportamiento del regulador, mejorando la acción de control e impidiendo que la saturación o los picos sean elevados. Con el fin de entender mejor que problema supondría no implementar el anti-windup se muestra la figura 3.3, en la que observamos como mejora el sistema implementando el algoritmo. El pico de sobrepasamiento se reduce, el tiempo de establecimiento también, el error acumulado es menor y las transiciones de la tensión también.

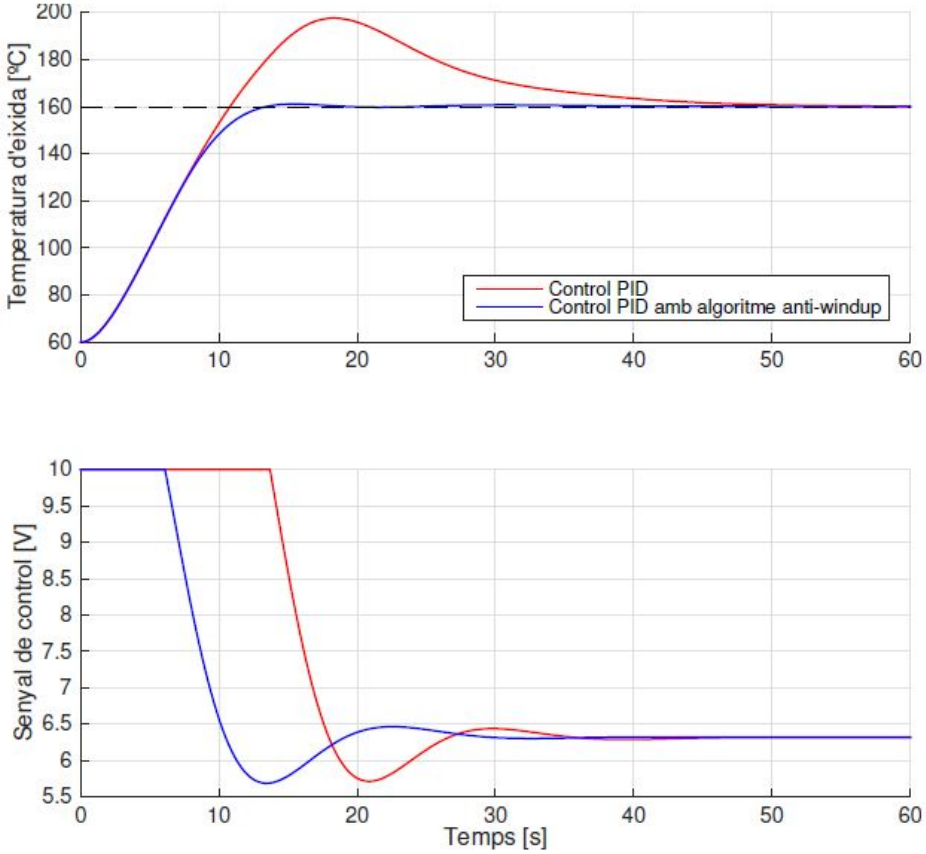

Figura 3.3: Comparación de un sistema con y sin anti-windup

Para este proyecto se van a implementar 2 algoritmos, la elección de uno u otro dependerá del ingeniero que diseñe el regulador. Los métodos que pasaremos a explicar a continuación son los siguientes:

- Back calculation
- Clamping

A continuación nos centraremos en explicar los distintos métodos que en este trabajo se han implementado. Hemos de tener claro que estos métodos actuaran en reguladores discretos y sobre la acción de control. Por eso creo conveniente recordar que el autómata tratara el regulador PID como un control discreto, ejecutando los algoritmos cíclicamente.

Algoritmo back calculation

Este método se va a actuar sobre la acción de control del integrador la cual afectara a la acción de control total del regulador como se puede observar a continuación:

$$u(t) = u_k(t) + u_i(t) + u_d(t)$$

$$u_i(t) = \int_0^t \left(\frac{K}{T_i} e(t) - \frac{1}{T_t} \big(v(t) - u(t) \big) \right) \tag{3.5}$$

En esta ecuación T_t es un parámetro del diseño del regulador cuyo valor es fijo y vale la mitad que el tiempo de integración. Si aplicamos la misma relación de la K_i y la T_i podremos decir que:

$$u_i(t) = \int_0^t \left(\frac{K}{T_i} e(t) - K_b \big(v(t) - u(t) \big) \right) \tag{3.6}$$

Este algoritmo es fácil de utilizar, y si analizamos lo que realiza nos damos cuenta que:

- Cuando $u(t)$ no satura ($u(t) = v(t)$) el integrador funcionara normalmente
- Cuando la acción de control satura ($v(t) > u(t)$) la señal del integrador sera la clásica de un PID por un lado, y la diferencia entre la acción de control no saturada y la saturada.

Clamping

El algoritmo clamping o condicional integrador esta basado en la acumulación de error en la saturación de la acción de control. La acción integral sera igual en todos los valores para los cuales $u(t)$ sea igual a $v(t)$. Y sera 0 para todos los valores en los que la acción sature. Esto se traduce en mantener el valor de la acción de control de la parte integral $ui(t)$ del instante anterior evitando así la acumulación del error.

Una de las acciones de mejora que se pude implementar para este algoritmo es variar el valor del error de la integral. Si definimos la acción de control integral como:

$$u_i(t) = K_i \int_0^t e_i(t) dt \tag{3.7}$$

Mantendremos el valor de $e(t)$ cuando el valor de $u(t)$ no se encuentre saturado ($u(t) = v(t)$) y cuando el producto del error ($e(t)$) por la acción de control ($u(t)$) sea menor que 0. Si no se cumple ninguno de estos dos casos el valor del error de la integral lo igualaremos a 0. Con esta mejora lo que conseguimos es que el integrador ayude a frenar al sistema cuando la lectura del sensor es mayor que la referencia.

Mediante esta implementación podremos ajustar mejor los picos de sobrepasamiento.

3.3 Función mapapz

Se trata de una función auxiliar a la que acudiremos con el fin de modificar algunas gráficas añadiendo información extra. Esta función se podrá activar siempre que el archivo "mapapz.m" este guardado en la misma carpeta que el código desde el que se llama. Si atendemos al código mostrado en Listing 3.1 veremos una pequeña descripción de la función realizada por el autor y tutor de este proyecto Adolfo Hilario Caballero. Si analizamos las propiedades, ha de ser una variable de tipo "struc" o estructura de datos, en la que deben estar contemplados los parámetros descritos en ella (color, grandaria, grossor, noMesPols). Tal y como cita el autor, esta función se puede superponer a la gráfica generada por la función de matlab rlocus.

Listing 3.1: Función auxiliar mapapz parte1

```
function mapapz(sistema,propietats)
%mapapz((miSistema,propietats)
%
%Dibuixa el mapa de pols i zeros de sistema
%amb les propietats:
%
%     propietats.color
%     propietats.grandaria
%     propietats.grossor
%     propietats.noMesPols (boolean)
%
%Es pot superpoar a un rlocus()
%
%Adolfo Hilario
%Febrer de 2014
```

Si ahora analizamos la función en sí, mostrada en Listing 3.2, observamos una serie de condicionales y bucles que modificaran la forma de visualización de estos polos. Los condicionales serán los encargados de modificar la forma de visualización, mientras que los bucles son los encargados de buscar todos los polos o zeros que tenga nuestro sistema. Atendiendo a las variables de esta función (propietat_p y propietat_c) observamos que se asigna el valor tipo char de "kx" y "ko" respectivamente. Estas asignaciones harán referencia a la manera de representar los polos (propietat_p) con una "X" o los zeros (propietat_c) con una "O". Como algo novedoso, se incorpora la función negación en los condicionales con el símbolo " ", la isfield a modo de validación de variables e isreal a modo de comparación numérico.

Listing 3.2: Función auxiliar mapapz parte 2

```
if ~exist('propietats','var')
        propietats.noMesPols = false;
end
if ~isfield(propietats,'nomesPols')
        propietats.nomesPols = false;
end

propietat_p = 'kx';
propietat_c = 'ko';

[pols,zeros] = pzmap(sistema);

for i=1:length(pols),
        if isreal(pols(i)),
                q = plot(pols(i),0,propietat_p);
        else
                q = plot([pols(i),pols(i)'],
                   propietat_p);
        end
        if isfield(propietats,'grandaria') set(q,'
           MarkerSize',propietats.grandaria); end
        if isfield(propietats,'grossor') set(q,'
           LineWidth',propietats.grossor); end
        if isfield(propietats,'color') set(q,'Color',
            propietats.color); end
end

if ~propietats.nomesPols
        for i=1:length(zeros),
                if isreal(zeros(i)),
                        q = plot(zeros(i),0,
                           propietat_c);
                else
                        q = plot([zeros(i),zeros(i)'],
                           propietat_c);
                end
                if isfield(propietats,'grandaria') set
                   (q,'MarkerSize',propietats.
                   grandaria); end
                if isfield(propietats,'grossor') set(q
                   ,'LineWidth',propietats.grossor);
                    end
                if isfield(propietats,'color') set(q,
                   'Color',propietats.color); end
        end
end
```

3.4 Parámetros físicos

Comenzamos con el primer sub-programa generado en Matlab, al que hemos llamado "parametros_fisicos". Este será la base de todos los sub-programas, ya que de él obtendremos los parámetros necesarios tanto para la virtualización de la planta (K_1, K_2, K_3, K_4, τ_A, τ_B, τ_C, K_{11}, K_{21}, K_{22}, K_{32}, K_{43}, K_{em}, K_{em2}, $taum$, $taum2$ y T), como para el diseño del regulador.

A continuación comenzaremos mostrando la primera parte del código de matlab de este sub-programa, en el que cargaremos las constantes descritas en la subsección 2.1.1

Listing 3.3: Parámetros físicos parte 1

```
%% Parametros_fisicos.m

%% Datos

rho = 1000;                 % kg/m3
Cp = 4.18;                  % kJ/kgºC
UA = 4180;                  % kJ/sºC
V = 5;                      % m3
Vc = 2;                     % m3

QfRE = 0.25;        % m3/s
QcRE = 0.05;        % m3/s

URE = 5;            % V
TpRE = 50;
TcRE = 55;

%% Válvula

Kev = QcRE/URE;
tauv = 1;           % s
Gv = tf([Kev],[tauv,1]);

%% Cálculo constantes

K1 = rho*Cp;
K2 = UA;
K3 = rho*Cp;
K4 = Kev;
tau1 = rho*V*Cp;
tau2 = rho*Cp*Vc;
tau3 = tauv;

K11 = K1/tau1;
K21 = K2/tau1;
K22 = K2/tau2;
K32 = K3/tau2;
K43 = K4/tau3;

para_LabVIEW = [K1,K2,K3,K4,tau1,tau2,tau3,K11,K21,K22
    ,K32,K43];
```

Como se puede observar en Listing 3.3, la última linea genera una matriz llamada ”para_labVIEW”. Estos serán los datos que necesitaremos implementar en nuestra planta a virtualizar de LabVIEW. Estos datos son los citados en la sección 2.1. Es cierto que este código no esta completo, ya que no hemos generado la matriz en un formato compatible para exportar a LabVIEW. Es importante destacar, que para virtualizar una u otra planta habría que hacer modificaciones hasta el código mostrado en el Listing 3.3, ya que este sub-programa será la base de los siguien-

tes. Otra razón por la que se ha generado la matriz aquí, es para recordar al ingeniero el orden de los datos que se volcarán.

A continuación, seguimos con el mismo código del programa pero ahora generando las ecuaciones necesarias para posteriormente obtener los bloques y tener así la planta en Simulink.

Listing 3.4: Parámetros físicos parte 2

```
%%

TfRE = (K1*QfRE*TpRE-K2*(TcRE-TpRE))/(K1*QfRE)
TceRE = (K3*QcRE*TcRE+K2*(TcRE-TpRE))/(K3*QcRE)

%% Equivalencia de las variables de estado

X1RE = TpRE;
X2RE = TcRE;
X3RE = QcRE;

U1RE = URE;
U2RE = QfRE;
U3RE = TfRE;
U4RE = TceRE;

%% Calculo de constantes para el sistema linealizado

K1 = (rho*Cp*(TpRE-TfRE))/(rho*QfRE*Cp+UA);    % s*ºC/m3
K2 = (rho*Cp*QfRE)/(rho*Cp*QfRE+UA);           % ADIMENSIONAL
K3 = (UA)/(rho*Cp*QfRE+UA);                    % ADIMENSIONAL
tau1 = (rho*V*Cp)/(rho*Cp*QfRE+UA);            % s
K4 = (rho*Cp*(TceRE-TcRE))/(rho*Cp*QcRE+UA);   % s*ºC/m3
K5 = (rho*Cp*QcRE)/(rho*Cp*QcRE+UA);           % ADIMENSIONAL
K6 = (UA)/(rho*Cp*QcRE+UA);                    % ADIMENSIONAL
tau2 = (rho*Cp*Vc)/(rho*Cp*QcRE+UA);           % s


%% Funciones de transferencia

Gv = tf([Kev],[tauv,1]);
G1 = tf([K4],[tau2,1]);
G2 = tf([K3],[tau1,1]);
GTp = tf([K6],[tau2,1]);
Gd1 = tf([K2],[tau1,1]);
Gd2 = tf([K1],[tau1,1]);
Gd3 = tf([K5],[tau2,1]);
```

En el Listing 3.4, observamos como hemos generado las funciones de transferencia según el proceso descrito en las secciones anteriores. Este es el primer sub-programa de Matlab, el cual no muestra información al ingeniero que lo ejecute, pero si queda recogido en un solo archivo todos los datos que necesitan modificación para generar nuevas plantas y reguladores.

3.5 Diseño de reguladores y tratamiento de datos

3.5.1 Regulador PID estándar

En este punto partimos del código generado en la sección 3.4, y explicaremos paso a paso todo el código utilizado para la generación de las matrices, que posteriormente formaran los archivos CSV necesarios para la virtualización de la planta. Este mismo código también servirá para la generación un regulador PID estándar con ponderación de la referencia y filtro de la derivada.

Listing 3.5: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 1

```
%% Simulación del sistema no lineal

clc; clearvars; close all;

s = tf('s');

parámetros_físicos;

%% Configuración de la simulación

dibuixa_rlocus = false;

dibuixa_Tp = true;

diagrama_de_bloques = 'PID_modelo_no_lineal_2015b';
bloque_PID = '/Regulador';

stop_time = 250;
step_size_continu = 10e-3; %T (periode de muestreo
    del modelo)
```

En Listing 3.5 comenzamos borrando todas las variables, la Comand Windows y cerrando todas las ventanas. Esto se realiza por seguridad a la hora de diseñar, ya que si el programa tiene algunas variables guardadas falseará los datos y provocaría errores en el diseño y la virtualización. En la segunda línea creamos la variable s de Laplace, posteriormente nos servirá para implementar las funciones de transferencia.

En la tercera línea cargamos todas las variables generadas con el código explicado en la sección 3.4 (parametros_fisicos). Posteriormente creamos unas variables booleanas ("dibuixa_rlocus" y "dibuixa_TP"). Estas variables serán las encargadas de mostrar (o no) una serie de gráficas que explicaremos más adelante. Se ha decidido posicionar las variables aquí ya que no son necesarias para la función de este sub-programa, sino que mostrarán información adicional.

A continuación cargamos el diagrama de bloques de simulink "PID_modelo_no_lineal_2015b", el cual observaremos en la figura 3.4. Además de esto, se crea una variable de tipo Char, que se llama "bloque_PID", generamos el límite de la simulación en los $250s$, tomando como periodo de muestreo los $10ms$.

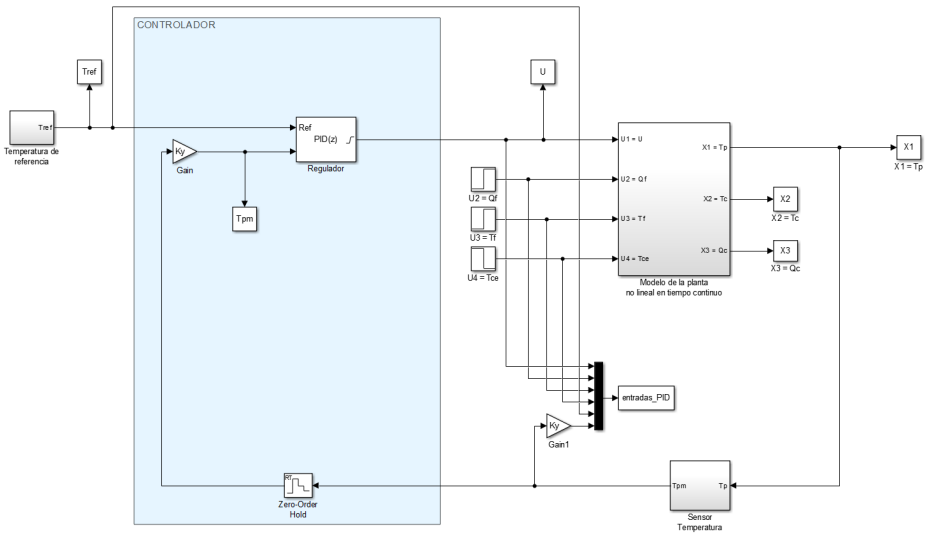

Figura 3.4: Bloque de simulink PID_modelo_no_lineal_2015b

Listing 3.6: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 2

```
%% Configuracion escalones en perturbaciones y en la
   referencia

step_Tf = 100;              % tiempo de inicio de escalón
start_Tf = TfRE;            % valor inicial escalón
% incr_Tf = 5;              % incremento escalón
incr_Tf = 0;                % incremento escalón

step_Qf = 80;               % tiempo de inicio de escalón
start_Qf = QfRE;            % valor inicial escalón
% incr_Qf = 0.1;            % incremento escalón
incr_Qf = 0;                % incremento escalón

step_Tce = 160;             % tiempo de inicio de escalón
start_Tce = TceRE;          % valor inicial escalón
% incr_Tce = 3;             % incremento escalón
incr_Tce = -25;             % incremento escalón
```

Analizando el código mostrado en Listing 3.6, se generan las variables de tiempos, escalones e incrementos para la simulación de las perturbaciones y la referencia. Estos escalones están diseñados para nuestra planta, por lo que necesitarían ser modificados para otro tipo de plantas si se reutiliza el código mostrado.

Listing 3.7: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 3

```
% Temperatura de referencia:

step_Tref1 = 0;          % tiempo de inicio de escalón
start_Tref1 = TfRE; % valor inicial escalón
incr_Tref1 = 10;         % incremento escalón

step_Tref2 = 80;         % tiempo de inicio de escalón
incr_Tref2 = 10;         % incremento escalón

Umax = 10;
Umin = 0;

SAT_MAX = Umax;
SAT_MIN = Umin;

% Vàlvula de control totalmente cerrada
X1RE = TfRE;
X2RE = TfRE;
X3RE = 0;

tipus_regulador = 'PID';

T = 1;   % Ts (periodo de muestreo del PID)
```

Si continuamos con lo mostrado en Listing 3.7, realizamos el incremento de temperatura y configuramos los valores máximos de saturación de la válvula. De esta parte del código lo más importante es el tiempo de muestreo del regulador PID, ya que si queremos realizar la comparación en las mismas condiciones deberemos configurar este a 1s también, que es el tiempo que por defecto tiene el autómata seleccionado. Por otra parte se genera nuevamente una variable de tipo ”char” llamada ”tipus_regulador” en la que guardaremos el valor del regulador seleccionado. ”PID”.

Listing 3.8: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 4

```
%% Modelo del sensor y adaptación a la referencia

Ym_min = 0; Ym_max = 10;
Ymin = 0; Ymax = 100;
Kem = (Ym_max-Ym_min)/(Ymax-Ymin);
taum = 1;
Gm = Kem/(taum*s+1);
Ky = 1/Kem;

Ym_min2 = 0; Ym_max2 = 10;
Ymin2 = 0; Ymax2 = 200;
Kem2 = (Ym_max2-Ym_min2)/(Ymax2-Ymin2);
taum2 = 1;
```

En el Listing 3.8 mostraremos la parte de código con la que obtenemos los sensores que miden la temperatura. Estos sensores, tal y como dijimos en la subsección 2.1.2, son ideales. No introducirán más zeros en el sistema y únicamente generarán un polo. Como esta información no la hemos cargado con los parámetros de la planta en el sub-programa parametros_fisicos.m, la generamos aquí, debiendo cambiarse en función de los sensores obtenidos al final.

Listing 3.9: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 5

```
%% Diseño del controlador

Gp = minreal(feedback(G2,-GTp)*G1*Gv);

L = Gp*Gm*Ky;

ts = 50;
Mp = 0.1;

sigma = 4/ts;
wd = -sigma*pi/log(Mp);
s1 = -sigma + j*wd;

fc = angle_compensacio(L/s,s1);
fprintf('\n\nEl ángulo de compensación es %0.2fº \n\n',
    fc*180/pi)

alfa_c = fc/2; % Porque es un zero doble
c = sigma + wd/tan(alfa_c);

Gc = (s+c)^2/s;

Kd = rlocfind(Gc*L,s1);
Gc = zpk(Kd*Gc)

Gpid = pid(Gc)
Kp = Gpid.Kp;
Ki = Gpid.Ki;
Kd = Gpid.Kd;
%Kd = 0;

Td = Kd/Kp;
Ti = Kp/Ki;

alfa = 1/5; N = 1/(alfa*Td); %N = 100;
beta = 0.2;
gamma = 0;

if strcmpi(tipus_regulador,'PID')
        Kb = 1/sqrt(Ti*Td);
else
        Kb = 1/(0.5*Ti);
end
```

En el Listing 3.9, vamos a mostrar el código encargado del diseño del PID estándar, diseñado por el criterio del zero doble. Este código es el más importante, ya que directamente nos dará los valores necesarios para implementar un PID funcional en nuestro autómata. Esten regulador lo testaremos contra la virtualización de la planta pudiendo comprobar si el diseño es estable

o necesita alguna mejora. La primera línea crea el sistema en lazo abierto junto a la válvula (sabiendo que este sistema tiene un lazo interno), y la segunda crea el lazo abierto con el sensor y la adaptación de la referencia. De este modo, tenemos todo el sistema en las mismas unidades, para que el regulador sea capaz de ajustar la referencia a lo deseado. A continuación, creamos unas variables auxiliares con las que obtener el angulo de compensación. Este dato es mostrado en la "Command Windows" como dato para el ingeniero, y para la localización del futuro zero doble, optimizando así el funcionamiento. Una vez se a asignado el zero doble en la posición óptima se asigna el integrador en el origen y se cierra el lazo con el regulador. Del regulador, mediante la función "pid", se obtendrán las constantes k_p k_i k_d las constantes de tiempo, con las que se ajustarán α, β y γ mejorando el funcionamiento. Estos datos son los necesarios para implementar en el autómata un bloque PID

Listing 3.10: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 6

```
        %% Lugar de las raíces

if dibuixa_rlocus

        figure;
        set(gcf, 'Units', 'Normalized',...
                 'Position',[0.1,0,0.33,1],...
                 'PaperPositionMode', 'Auto'...
                )
        sub_rlocus1 = subplot(2,1,1); rlocus(L/s);
            hold on;
        h = plot([s1,s1'], 'kx'); set(h, 'LineWidth',3,'
            MarkerSize',9);
        sub_rlocus2 = subplot(2,1,2); rlocus(Gc*L);
            hold on;
        h = plot([s1,s1'], 'kx'); set(h, 'LineWidth',3,'
            MarkerSize',9);
        axis equal;
        v = axis; subplot(sub_rlocus1); axis(v);
end
```

Como se cita en la explicación del Listing 3.5, se creó una variable llamada "dibuixa_rlocus". Esta variable booleana se utiliza ahora, en caso de ser afirmativa, en un condicional para mostrar la figura del lugar geométrico de las raíces de la planta sin compensar, y la planta con el regulador, o no mostrarlo y saltarse el condicional mostrado en Listing 3.10 en caso de ser falsa.

Se generará una ventana nueva en la que se mostrarán dos gráficas, una encima de otra y siendo la primera la planta sin compensar donde podemos observar que los "brazos" están en el semiplano derecho, un sistema inestable, y como tras instalar el regulador con el zero doble se convierte en un sistema mucho mas estable (figura 3.5)

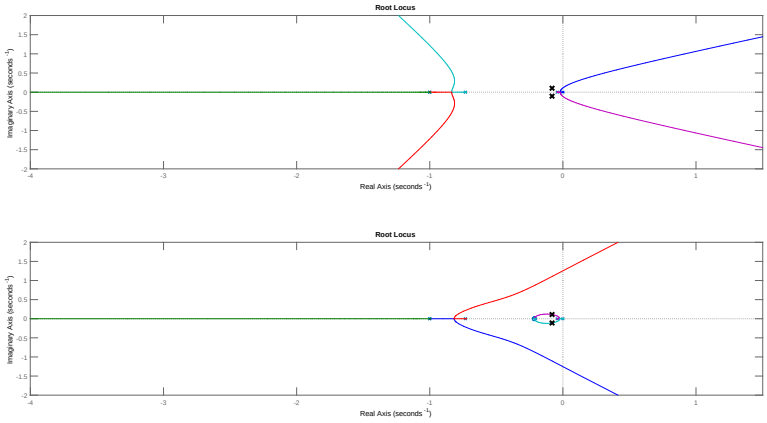

Figura 3.5: Lugar de las raíces de la planta sin compensar y compensada mediante un PID diseñado por el método del zero doble

Listing 3.11: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 7

```
%% Función de transferencia de lazo cerrado

if dibuixa_rlocus

        % Propiedades para la representación gráfica
            de los polos de lazo cerrado
        propiedades_polos_lc.grandaria = 16;
        propiedades_polos_lc.grossor = 1;
        propiedades_polos_lc.nomesPols = true;
        propiedades_polos_lc.color = 'r';

        M = minreal(feedback(Gc*L,1));
        subplot(sub_rlocus2); mapapz(M,
            propiedades_polos_lc);

end
```

Con el fin de mejorar la representación de los polos y zeros mostrados en la gráfica inferior, se ha llamado a la función "mapapz" descrita en la sección 3.3 para mostrar el estado de los polos superpuestos que incluirá el regulador.

Listing 3.12: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 8

```
%% Simulación

awms = {'back-calculation'}; awm = 1;  % 0 None 1 BC 2
    Clamping
% awms = {'clamping'}; awm = 2;  % 0 None 1 BC 2
    Clamping


% Configuración del regulador en el diagrama de
    bloques de Simulink
load_system(diagrama_de_bloques);
set_param([diagrama_de_bloques,bloque_PID],'Controller
    ','PID')
set_param([diagrama_de_bloques,bloque_PID],'Kb',
    num2str(Kb))

Tps = [];
Tpms = [];
Us = [];

for i = 1 : numel(awms)

        set_param([diagrama_de_bloques,bloque_PID],'
            AntiWindupMode',awms{i})

        opciones = simset(...
                'Solver','ode8',...
                'FixedStep', step_size_continu...
                );

        sim(diagrama_de_bloques,stop_time,opciones);

        Tpms = horzcat(Tpms,Tpm.signals.values);
        Tps = horzcat(Tps,X1.signals.values);
        Us = horzcat(Us,U.signals.values);

end
```

A continuación, pasaremos a explicar la parte del código mostrado en el Listing 3.12, mediante el cual configuraremos la simulación que aportará los datos necesarios para comparar por un lado la planta virtualizada con el autómata real, y por el otro el regulador teórico y la planta linealizada.

Este código actuará contra el modelo de Simulink que se mostró en la figura 3.4. En la primera linea, creamos unas variables auxiliares que se utilizarán posteriormente. Después cargamos con la variable "diagrama_de_bloques" que generamos en el Listing 3.5 el modelo de simulink. Configuramos una serie de parámetros para la simulación y luego generamos unas matrices vacías, las cuales, se escribirán en el bucle for.

Dentro de este bucle se configura la resolución de ecuaciones integro-diferenciales por la iteración "ode8", pudiendo cambiarse a "ode16" o cualquier otro método deseado, y escribimos las matrices, las cuales tendrán tantos valores como veces se ejecute el bucles, en este caso 25000. El número de repeticiones del bucle esta ligado a la variable auxiliar "awms" que acabamos de generar, y su valor dependerá de las iteraciones necesarias para estabilizar la planta. Esto puede ser peligroso si la planta fuese amortiguada, ya que las iteraciones tenderían a infinito.

Listing 3.13: Código de simula_PID_sobre_no_lineal_LabVIEW parte 9

```
        %% Representación gráfica

figure;
        set(gcf,'Units','Normalized',...
        'Position',[0.5,0,0.33,1],...
        'PaperPositionMode','Auto'...
        )
sub_Tp = subplot(2,1,1); grid on; hold on;
stairs(Tpm.time,Tpms);
if dibuixa_Tp plot(X1.time,Tps); end;
h = plot(Tref.time,Tref.signals.values); set(h,'
    LineStyle','--','Color',0.4*[1,1,1])
title('Variable controlada: Temperatura del proceso','
    FontSize',14);
ylabel('Temperatura [ºC]');
xlabel('Tiempo [s]');

%h = plot([160,160],[60,58],'--k');
%text(130,57,sprintf('Perturbación en\n la
    temperatura\n de entrada (Tf)'))

if dibuixa_Tp
        h = legend(...
                'Tpm - PID con anti-windup',...
                'Tp - PID con anti-windup',...

                'Referencia',...
                'location','southeast');
else
        h = legend('PI estándar',...
                'PI con anti-windup',...
                'Referencia',...
                'location','southeast');
end
set(h,'FontSize',13)

sub_U = subplot(2,1,2); grid on; hold on;
stairs(U.time,Us);
axis([0,stop_time,Umin,Umax]);
title('Variable manipulada: Tensión aplicada a la
    electroválvula','FontSize',14);
ylabel('Tensión eléctrica [V]');
xlabel('Tiempo [s]');

linkaxes([sub_Tp,sub_U],'x')

%print(gcf,'-depsc','lazo_cerrado.eps')
%print(gcf,'-dpdf','lazo_cerrado.pdf')
```

A continuación, pasaremos a representar gráficamente los resultados del regulador con el código mostrado en Listing 3.13. Comenzamos creando una nueva ventana de figura, con el fin de no sobrescribir la del lugar de las raíces, en el caso que se halla decidido mostrar activando la variable auxiliar "dibuixa_rlocus". Después generamos dos subfiguras, en las que mostraremos posteriormente dos gráficas distintas. En la figura superior se creará la rejilla, y se mostrará la matriz que generamos en el Listing 3.12, con los puntos de color azul del diseño del regulador discreto con anti-windup (matriz Tpms). Después se representa sobre la misma gráfica, pero en color rojo, los valores generados del regulador continuo con anti-windup (matriz Tps). Por último, se representan los saltos de referencia en línea discontinua, la leyenda y se añadirán los títulos de la gráfica y los ejes.

Seguidamente, pasamos a representar la segunda gráfica, en la que mostraremos la tensión de salida del regulador, es decir, la tensión que aplicaremos a la electroválvula B. De igual modo que en la gráfica anterior activaremos primero la rejilla. Después, se representa la tensión de la válvula y se ajustan los ejes a la tensión aplicable, que tal y como se indicó en la subsección 1.5.2 fluctuara entre 0 y $10V$. Activaremos los títulos de los ejes y de la gráfica. Las dos ultimas lineas del código que están comentadas servirán para guardar la gráfica en formato EPS, o PDF. Tras la creación de la representación obtenemos la figura 3.6, en la que observamos el diseño del regulador.

Listing 3.14: Codigo de simula_PID_sobre_no_lineal_LabVIEW parte 10

```
%% Creamos ficheros CSV

U = entradas_PID.signals.values(:,1);
Qf = entradas_PID.signals.values(:,2);
Tf = entradas_PID.signals.values(:,3);
Tce = entradas_PID.signals.values(:,4);
Tref = entradas_PID.signals.values(:,5);
Tpm = entradas_PID.signals.values(:,6);

dlmwrite('../CSV/Tref_i_Pertorbacions.csv',[U,Tce,Qf,
    Tf,Tpm,Tref],';');

%para_LabVIEW = [K1,K2,K3,K4,tau1,tau2,tau3,K11,K21,
    K22,K32,K43]; (12)
dlmwrite('../CSV/parametres_model.csv',[para_LabVIEW,
    step_size_continu,X1RE,X2RE,X3RE,Kem,Kem2,taum,
    taum2],';');

dlmwrite('../CSV/param_PID.csv',[Kp,Ti,Td,alfa,beta,
    gamma,awm,Kb,T],';');
```

Una vez hemos llegado a este punto en el que el regulador y el ensayo están realizados, falta extraer los datos necesarios para virtualizar la planta (K_1, K_2, K_3, K_4, τ_A, τ_B, τ_C, K_{11}, K_{21}, K_{22}, K_{32}, K_{43}), replicar el controlador (K_p, T_i, T_d, α, β, γ, awm, K_b, T) y las condiciones del ensayo (U, T_{ce}, Q_f, T_f, T_{pm}, T_{ref}) con el fin de compararlos. Para ello, necesitamos extraer los datos en un formato compatible para ambos, y se ha seleccionado el CSV. Si comenzamos analizando el código mostrado en el Listing 3.14, observamos como se crean los vectores ne-

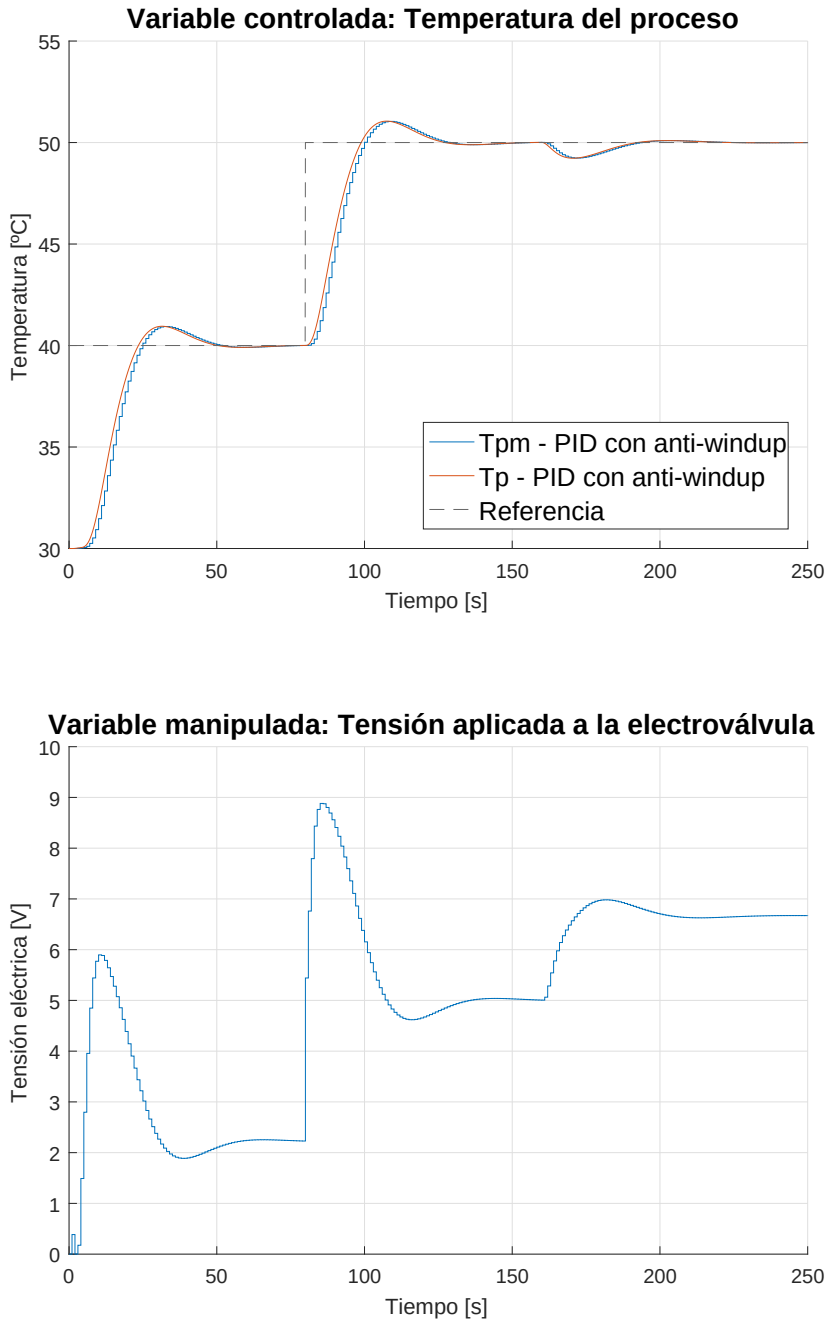

Figura 3.6: Ensayo teórico del comportamiento de la planta con el regulador diseñado

cesarios para replicar las condiciones del ensayo. Estos vectores son guardados todos juntos en forma de matriz y generados en formato csv con el nombre de "Tref_i_Pertorbacions". La siguiente linea del código sirve para extraer los datos para la virtualización de la planta, según el orden indicado en el comentario superior, y serán guardados como "parametres_model". Por ultimo, se genera otra matriz que guarda los datos de configuración del PID con el nombre de "param_PID". Todos serán guardados en la carpeta superior a donde también guardemos los archivos de Matlab en una carpeta que se llame "CSV". Es muy importante que esta carpeta sea creada con antelación a la ejecución del código, de lo contrario puede generar errores.

3.5.2 Regulador PID en cascada

En esta sección realizaremos algunas modificaciones del código desarrollado anteriormente en la subsección 3.5.1, y generaremos un nuevo sub-programa para añadir ciertos parámetros que necesitamos para el diseño del nuevo controlador. Indicaremos al lector que sería conveniente generar dos nuevo sub-programa y ejecutar uno u otro en función del controlador que deseemos. Se explicará el código completo pero centrándonos en las partes nuevas del mismo, pasando por encima de lo ya descrito en la subsección 3.5.1.

Modelo_tanque

Como hemos citado en subsección 3.5.2 se ha de generar un nuevo sub-programa que añada alguna característica adicional al programa. Esto es debido, a que ahora tendremos un nuevo sensor de temperatura, el cual medirá la camisa, y una realimentación negativa para cerrar el tanque adecuadamente. Este sub-programa lo llamaremos "modelo_tanque", parte de su código se muestra a continuación:

Listing 3.15: Codigo de modelo_tanque parte 1

```
%% Modelo de la planta

s = tf('s');

%  Kev = (Tmax1/(K4*(K3/(1-K3*K6))))/(10-0);
%  tauv = 1/4*tau1;

Gv = Kev/(tauv*s + 1);

G1 = K4/(tau2*s + 1);

G2 = K3/(tau1*s + 1);

GTp = K6/(tau2*s + 1); % Reciclo positivo

GTf1 = K2/(tau1*s + 1); % Perturbación en la
    temperatura de entrada del tanque

GQf = K1/(tau1*s + 1); % Perturbación en el caudal de
    entrada al tanque

GTce = K5/(tau2*s + 1); % Perturbación en la
    temperatura de entrada de la camisa

%% Model dels sensors

Kem = (10-0)/(100 - 0);
taum = 1;
Gm1 = Kem/(taum*s + 1); % Sensor primario

Kem2 = (10-0)/(200 - 0);
taum2 = 1;
Gm2=Kem2/(taum2*s+1); % Sensor secundario
```

Atendiendo a lo mostrado en el Listing 3.15, pasaremos a explicar nuevamente línea por línea las funciones que hemos realizado y el por qué de estas. En la primera línea podemos observar como generamos la variable de Laplace, esto nos servirá para crear nuevas funciones de transferencia. Las siguientes líneas volvemos a reproducir las mismas funciones de transferencia que en el código descrito en Listing 3.4, pero añadimos alguna otra que no hacia falta con anterioridad. Esto puede parecer redundante pero, de este modo, sobrescribiremos las funciones generadas con anterioridad y prepararemos las que cerrarán el lazo de realimentación positiva con el signo adecuado. Posteriormente, generamos los sensores ideales, los cuales tendrán la ganancia estática ideal para los rangos de temperatura que manejamos. Antes habíamos creado los mismos sensores, pero ahora lo hacemos centrando en este sub-programa todos los posteriores bloques de la planta (a excepción del regulador que queremos diseñar).

Listing 3.16: Codigo de modelo_tanque parte 2

```
%% Adaptación a la referencia

Ky = 1/Kem;
% Ky2 = 1/(Kev*Kem2*K4);
Ky2 = 1/Kem2;
% Ky2 = 1;

%% Punto de funcionamento y saturación del controlador

Tp0 = TpRE;
Tcs0 = TcRE;
U0 = QcRE/Kev;

Umax = 10;
Umin = 0;

Umax1 = 71.67;
Umin1 = 30;

UPPER_SAT = Umax - U0;
LOWER_SAT = -(U0 - Umin);
```

Si analizamos lo mostrado en el Listing 3.16, generamos la adaptación de la referencia. Para esto, realizaremos la inversa de la ganancia estática de los sensores. Como una adaptación no será necesaria (K_{y2}), se igualará a 1 simulando así un cable. Si encontrásemos que el sensor no fuese ideal, habría que descomentar la línea correspondiente y comentar la que iguala a 1. Por último, se describe el punto de funcionamiento configurado y la saturación del controlador.

Este sub-programa será un código auxiliar que llamaremos en el diseño del regulador en cascada, debe estar guardado en la misma ruta que los demás códigos, pero no será necesaria su ejecución, ya que no aportará información relevante.

Programa del regulador en cascada

Este programa es similar al creado en la subsección 3.5.1, ya que volverá a generar los archivos para la virtualización de la planta (idénticos a los anteriores ya que la esta no varía), diseño del regulador y condiciones de la simulación, pero ahora con un regulador PID en cascada. Será conveniente, tal y como ya se citó en la subsección 3.5.2, generar dos programas y ejecutar solo aquel regulador que deseemos obtener. Pasando nuevamente a la explicación del código:

Listing 3.17: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 1

```
%% Simulación del sistema no lineal

clc; clearvars; close all; %bdclose all;

s = tf('s');

parametros_fisicos;
modelo_tanque;

%% Configuración de la simulación

dibuixa_rlocus = false;

diagrama_de_bloques = 'cascada_modelo_no_lineal_2015b'
    ;
bloque_PID = '/Regulador␣PID␣master';
bloque_slave = '/Regulador␣PI␣slave';

stop_time = 400;
step_size_continu = 10e-3;

T = 0.1;

simula_des_de_zero = true;

Tswitch = 0;
```

Analizando el Listing 3.17, vemos que es parecido al Listing 3.5. Vemos que al principio realizamos las mismas acciones (clc; clearvars; close all) y generamos la función de transferencia s de Laplace, aunque no sería necesario ya que esta será cargada con el código descrito en la subsección 3.5.2. Cargamos los sub-programas "parametros_fisicos" y "modelo_tanque" y volvemos a generar las variables booleanas para crear nuevamente los dibujos. Indicamos un nuevo modelo de simulink ya que ahora tenemos dos lazos y creamos las variables tipo char "bloque_PID" y "bloque_slave", que servirán para la creación del regulador. Se crean los tiempos para la simulación, esta vez se alargará hasta los $400s$ con un tiempo de muestreo de $0{,}1s$. A continuación, mostraremos otra parte del código:

Listing 3.18: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 2

```
%% Configuracion escalones en perturbaciones y en la
   referencia

step_Tf = 260;              % tiempo de inicio de escalon
start_Tf = TfRE;            % valor inicial escalon
% incr_Tf = 5;              % incremento escalon
incr_Tf = 5;                % incremento escalon

step_Qf = 200;              % tiempo de inicio de escalon
start_Qf = QfRE;            % valor inicial escalon
% incr_Qf = 0.1;            % incremento escalon
incr_Qf = 0;                % incremento escalon

step_Tce = 200;             % tiempo de inicio de escalon
start_Tce = TceRE;          % valor inicial escalon
% incr_Tce = 3;             % incremento escalon
incr_Tce = -35;             % incremento escalon
```

En el Listing 3.18, configuramos los tiempos de de la simulación del autómata sobre el simulink. El esquema de simulink lo encontramos en la figura 3.7. Si continuamos con el el sub-programa, veremos que proseguimos con la configuración de escalones en perturbaciones y en la referencia.

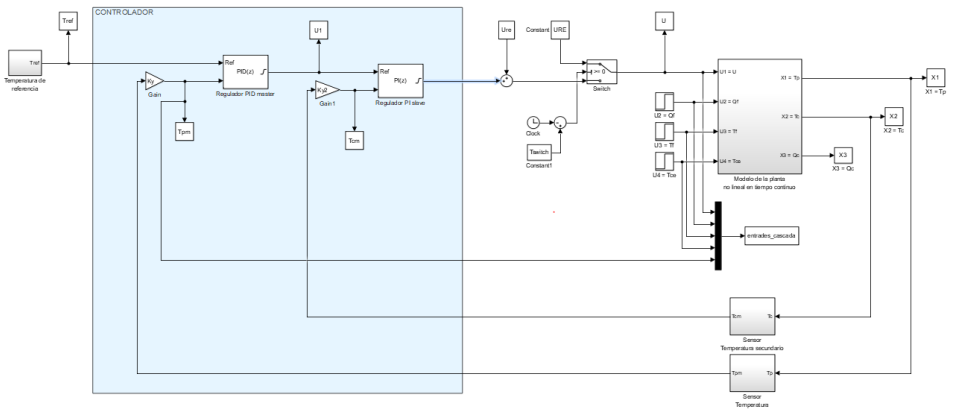

Figura 3.7: Ensayo teórico del comportamiento de la planta con el regulador diseñado

Listing 3.19: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 3

```
if simula_des_de_zero

        % Temperatura de referencia:
        step_Tref1 = 0;          % tiempo de inicio de
            escalon
        start_Tref1 = 0;         % valor inicial
            escalon
        incr_Tref1 = 40;         % incremento escalon

        step_Tref2 = 300;        % tiempo de inicio de
            escalon
        incr_Tref2 = 0;          % incremento escalon


        % Vàlvula de control totalmente cerrada
        X1RE = TfRE;
        X2RE = TfRE;
        X3RE = 0;
        Ure = 0;

else

        % Temperatura de referencia:
        step_Tref1 = 0;          % tiempo de inicio de
            escalon
        start_Tref1 = TpRE;  % valor inicial escalon
        incr_Tref1 = 0;          % incremento escalon


        step_Tref2 = 300;        % tiempo de inicio de
            escalon
        incr_Tref2 = -10;        % incremento escalon

        % Punto de funcionamento
        X1RE = TpRE;
        X2RE = TcRE;
        X3RE = URE;
        Ure = URE;

end
```

Si analizamos el código mostrado en el Listing 3.19, vemos que es un gran condicional con el que configuraremos los tiempos y escalones para la simulación. Este código esta en función de la variable booleana "simula_des_de_zero".

Listing 3.20: Código de Simula_cascada_sobre_no_lineal_LabVIEW parte 4

```
%% Especificaciones de diseño y diseño reguladores

% Lazo secundario

ts2 = 4/0.4;                % Tiempo de establecimiento
    del lazo secundario
Mp2 = 0.001;                % Sobrepasamiento del lazo
    secundario

sigma2=4/ts2;
wd2=-sigma2*pi/log(Mp2);
s2=-sigma2+1j*wd2;
L2 = zpk(Gv*G1*Gm2*Ky2)

metodo2 = 2;          % 1=compensación ; 2=cancelación


if metodo2 == 1

        polos2=pole(L2/s);
        zeros2=zero(L2/s);
        alfa_c2=-pi + sum(angle(s2-polos2)) - sum(
            angle(s2-zeros2));
        c2=sigma2+wd2/tan(alfa_c2);
        Gc2=(s+c2)/s;
        kd2=rlocfind(Gc2*L2,s2);
        Gc2 = zpk(Gc2*kd2)                  % Controlador
            secundario


elseif metodo2 == 2

        polos2=pole(L2/s);
        zeros2=zero(L2/s);
        polos2 = polos2(imag(polos2)==0)
        polos2=sort(polos2(polos2<0),'descend')
        c2 = abs(polos2(1));
        Gc2=(s+c2)/s;
        kd=rlocfind(Gc2*L2,s2);
        Gc2= zpk(kd*Gc2)

end
```

Atendiendo al Listing 3.20, realizaremos las especificaciones del diseño del regulador, comenzando por el secundario, es decir, el rápido. Marcaremos tanto el tiempo máximo de establecimiento como el pico de sobrepasamiento del mismo. Calculamos una serie de variables como σ y ω a la vez que preparamos el lazo abierto de la planta L para el estudio de los polos y zeros. A continuación, generamos una variable para el posterior condicional, con la que elegiremos el método

de diseño. Si a esta variable le asignamos el valor de 1 el método será el de compensación, si por el contrario, elegimos el valor de 2 el método será el de cancelación.

Tras la elección del método independientemente de este, se posicionarán los polos y zeros del autómata para que cumpla con las especificaciones dadas en las primeas partes del código. Se calculan los ángulos de posición y compensación para la corrección según las especificaciones.

Listing 3.21: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 5

```
Gpid2 = pid(Gc2)

K_s = Gpid2.Kp;
Ki_s = Gpid2.Ki;

Ti_s = K_s/Ki_s;
Td_s = 0;

alfa2 = 1/3;
beta2 = 1;
gamma2 = 0;

awm2 = 1; % 0 None 1 BC 2 Clamping
Kb2 = 0.7*1/(0.5*Ti_s);


M2 = minreal(feedback(Gc2*Gv*G1*minreal(feedback(G2,-
    GTp)),(1/G2)*Gm2*Ky2)); %
    Función de transferencia de lazo cerrado del lazo
    secundario

figure;
rlocus(L2/s); v = axis;
figure;
rlocus(minreal(Gc2*L2)); axis(v)
figure;
pzmap(M2)

% return

figure;
rlocus(M2/s)
title('M2/s')
```

Analizando el Listing 3.21, obtenemos el regulador G_{pi_2} del que extraeremos las constantes proporcional (K_s) e integral (K_i_s). No obtenemos la constante derivativa ya que este regulador normalmente es un PI, no un PID. Posteriormente, añadimos la ponderación de la referencia. Por ese mismo motivo, la constante γ esta igualada a 0, ya que así conseguimos eliminar la acción derivada.

Seguidamente, se genera una serie de gráficas que dan información al ingeniero sobre la primera compensación de la planta. La primera de ellas es la figura de la planta en lazo abierto (figura 3.5), la segunda de ellas es el estado de la primera compensación del regulador en lazo abierto (figura 3.8), la tercera es el mapa de polos y zeros (figura 3.9) y la última de todas es el lazo rápido cerrado en la que observamos una planta sin compensar, pero que posteriormente entrará en otro regulador (figura 3.10).

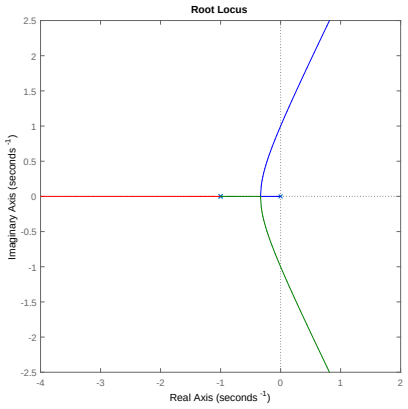

Figura 3.8: Primera compensación del PI rápido sobre la planta abierta

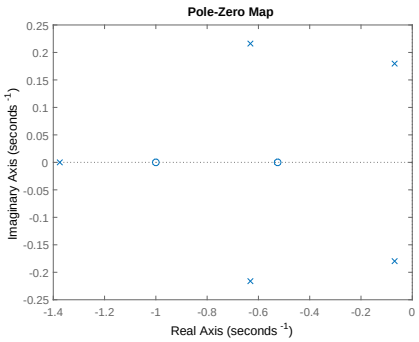

Figura 3.9: Lugar geométrico de las raíces del sistema

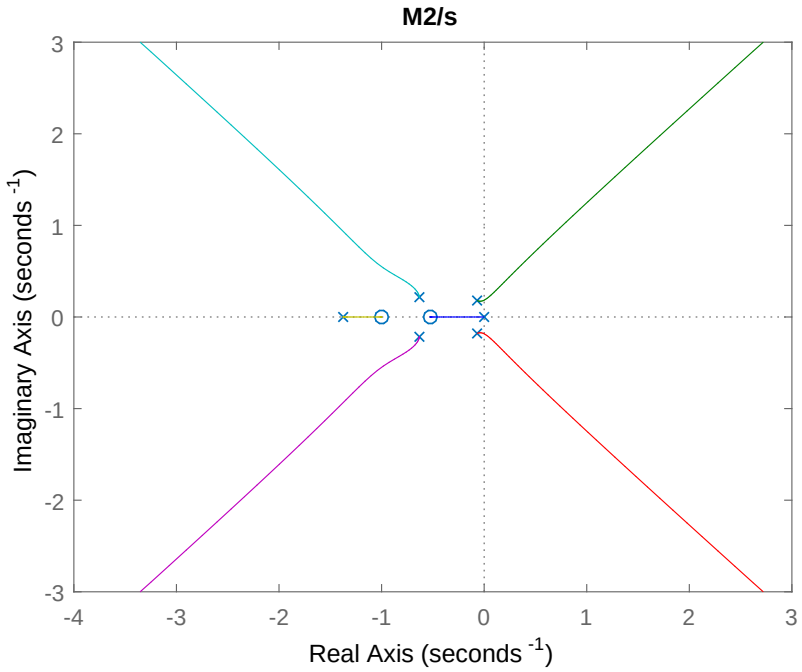

Figura 3.10: Primera compensación del PI rápido sobre la planta cerrada

Listing 3.22: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 6

```
% Lazo primario

ts = 18;          % Tiempo de establecimiento del lazo
Mp = 0.2;         % Sobrepasamiento del lazo

metodo = 1; % 1: zero doble, 2: Cancelación

sigma = 4/ts;
wd = -sigma*pi/log(Mp);
s1 = -sigma + 1j*wd;      % Punto del plano complejo
    para el diseño del lazo
L = zpk(minreal(M2*Gm1*Ky)); % Función de
    transferencia de lazo abierto


if metodo == 1 % zero doble

        polos2=pole(L/s);
        zeros=zero(L/s);
        alfa_c=(1/2)*(-pi + sum(angle(s1-polos2)) -
            sum(angle(s1-zeros)));
        c=sigma+wd/tan(alfa_c);
        Gc = (s+c)^2/s;           % Controlador primario
        Kd=rlocfind(Gc*L,s1);
        Gc=zpk(Gc*Kd)

elseif metodo == 2 % Cancelación

        polos2=pole(L);
        polos2=sort(polos2(polos2<0),'descend');
        c1= abs(polos2(1));
        c2= abs(polos2(2));
        Gc=((s+c1)*(s+c2))/s;
        Kd=rlocfind(Gc*L,s1);
        Gc= zpk(Kd*Gc)

end
```

En el Listing 3.22, analizaremos la manera de diseño del regulador primario. Como en el Listing 3.20, comenzamos asignando el pico de sobrepasamiento y tiempo de establecimiento. Lo siguiente que hacemos es seleccionar, mediante la asignación del valor de la variable "metodo", el tipo de diseño, ya que si el valor de esta se iguala a 1, seleccionaremos el método del zero doble. Si por el contrario, asignamos el valor de 2, seleccionaremos el método de cancelación. Posteriormente, y tal y como se realizó para el lazo rápido, crearemos las variables ω_d y S_1, que ayudarán al diseño del regulador. Para finalizar, generaremos la planta en lazo abierto y calcularemos el regulador.

Listing 3.23: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 7

```
Gpid = pid(Gc)
K = Gpid.Kp;
Ki = Gpid.Ki;
Kd = Gpid.Kd;

Td = Kd/K;
Ti = K/Ki;

alfa = 1/10; N = 1/(alfa*Td);
beta = 1;
gamma = 0;

awm = 1; % 0 None 1 BC 2 Clamping
Kb = 1/sqrt(Ti*Td); % PID
```

Una vez se ha diseñado el regulador primario (Listing 3.22), pasaremos a extraer los datos del regulador, constante proporcional (K), integral (K_i) y derivativa (K_d), aunque lo que realmente necesitaremos para la implementación del regulador en el autómata serán las constantes de tiempo integrativo (T_i) y derivativo (T_d). Luego asignamos valores a los filtros de la derivada y medida y a la ponderación de la referencia.

Una vez hemos realizado el diseño del regulador, volveremos a realizar un estudio con las figuras que mostramos en la subsección 3.5.1, y a generar archivos auxiliares en formato CSV para la posterior virtualización de la planta. Aunque estos no cambiarán, ya que solo exportan archivos de la propia planta, tenemos archivos auxiliares que extraen los datos del regulador.

Listing 3.24: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 8

```
%% Lugar de las raíces

if dibuixa_rlocus

        figure;
        set(gcf,'Units','Normalized',...
                'Position',[0.1,0,0.33,1],...
                'PaperPositionMode','Auto'...
                )
        sub_rlocus1 = subplot(2,1,1); rlocus(L/s);
            hold on;
        h = plot([s1,s1'],'kx'); set(h,'LineWidth',3,'
            MarkerSize',9);
        sub_rlocus2 = subplot(2,1,2); rlocus(Gc*L);
            hold on;
        h = plot([s1,s1'],'kx'); set(h,'LineWidth',3,'
            MarkerSize',9);
        axis equal;
        v = axis; subplot(sub_rlocus1); axis(v);
end
```

Como se observa en el Listing 3.24, se trata de un condicional en función de la variable "dibuixa_rlocus" con el que generaremos una gráfica doble. La parte superior será una representación del lugar geométrico de las raíces del lazo abierto del sistema primario. La parte secundaria mostrará el lugar de las raíces de la planta en lazo abierto y del controlador. Estas gráficas las podemos observar en la figura 3.11

Listing 3.25: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 9

```
%% Función de transferencia de lazo cerrado

if dibuixa_rlocus

        % Propiedades para la representación gráfica
            de los polos de lazo cerrado
        propiedades_polos_lc.grandaria = 16;
        propiedades_polos_lc.grossor = 1;
        propiedades_polos_lc.nomesPols = true;
        propiedades_polos_lc.color = 'r';

        M = minreal(feedback(Gc*L,1));
        subplot(sub_rlocus2); mapapz(M,
            propiedades_polos_lc);

end
```

Si analizamos lo mostrado en Listing 3.25, observaremos que se trata nuevamente de un condicional que esta vinculado a la misma variable que el condicional mostrado en Listing 3.24. Realizamos dos condicionales vinculados sobre la misma variable, para poder comentar el segundo en caso de no necesitar la información que nos muestra. Esta parte del código modificará la segunda gráfica en la que se mostrarán la posición de los polos y zeros de los nuevos reguladores. Si comparamos las dos imágenes podemos observar que en la figura 3.11 en la parte superior no aparecen marcadas las cruces rojas, mientras que en la figura 3.12 sí.

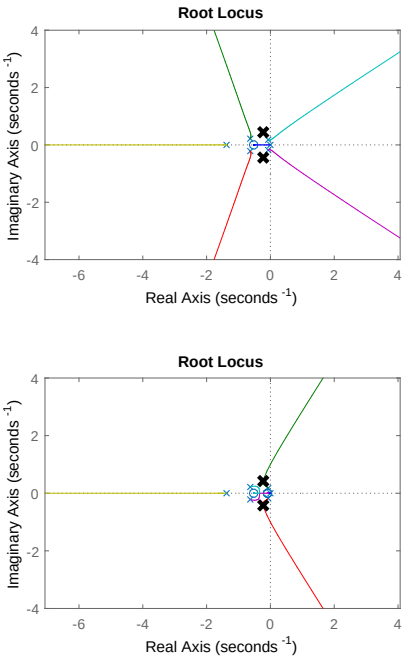

Figura 3.11: Figura sin aplicar la función mapapz

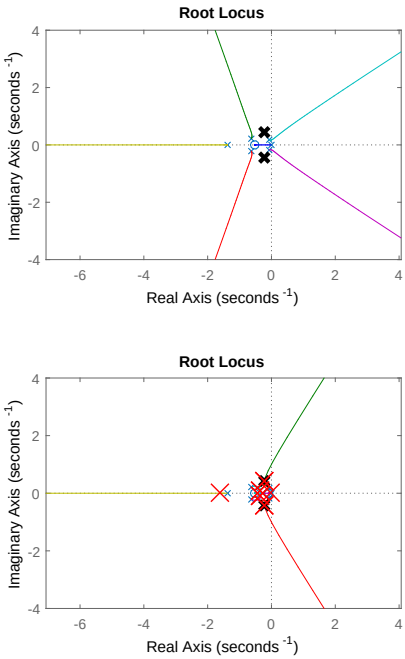

Figura 3.12: Figura aplicando la función mapapz

Listing 3.26: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 10

```
%% Representación gráfica

% % Configuración del regulador principal en el
    diagrama de bloques de Simulink
load_system(diagrama_de_bloques);
set_param([diagrama_de_bloques,bloque_PID],'
    AntiWindupMode','back-calculation')
set_param([diagrama_de_bloques,bloque_PID],'Kb',
    num2str(Kb))

% % Configuración del regulador secundario en el
    diagrama de bloques de Simulink
load_system(diagrama_de_bloques);
set_param([diagrama_de_bloques,bloque_slave],'
    AntiWindupMode','back-calculation')
set_param([diagrama_de_bloques,bloque_slave],'Kb',
    num2str(Kb2))

opciones = simset(...
        'Solver','ode8',...
        'FixedStep', step_size_continu...
        );

sim(diagrama_de_bloques,stop_time,opciones);
```

Si analizamos lo mostrado en el Listing 3.26, observaremos como configuramos la simulación sobre la planta de simulink mostrada en la figura 3.7 y los parametros auxiliares para ello. Estas variables estarán ligadas principalmente al calculo del anti-windup del que tratamos en la subsección 3.2.3

Listing 3.27: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 11

```
% Plot variables

figure;
set(gcf,'Units','normalized','Position'
    ,[0.6,0.2,0.3,0.7]);

sub_Tp = subplot(3,1,1);
plot(Tpm.time,Tpm.signals.values)
grid on; hold on
h = plot(Tref.time,Tref.signals.values);
set(h,'LineStyle','--','Color',0.6*[1,1,1])
xlabel('t(s)')
ylabel('Tp (ºC)')
legend('Tpm (ºC)','Tref')

sub_Tc = subplot(3,1,2);
plot(Tcm.time,Tcm.signals.values)
grid on; hold on
h = stairs(U1.time,U1.signals.values);
set(h,'LineStyle','-','Color',0.6*[1,1,1])
xlabel('t(s)')
ylabel('Tc (ºC)' )
legend('Tcm (ºC)', 'U1 = Tcref (ºC)')

sub_U = subplot(3,1,3);
grid on; hold on;
stairs(U.time,U.signals.values)
xlabel('t(s)')
ylabel('U (V)')
legend('U (V)')

linkaxes([sub_Tp,sub_Tc,sub_U],'x');
```

Si ahora pasamos a analizar el código mostrado en Listing 3.27, observamos como generaremos una imagen con tres figuras dentro. La primera de ellas sera la simulación contra la planta teórica, es decir contra la planta de simulink, la segunda de ellas sera la lectura de los sensores, donde podremos observar las señales que se introducir al regulador tanto al primario como al secundario. Por ultimo, la tercera imagen sea la señal de control que saldrá del regulador y se introductora a la válvula B.

Listing 3.28: Codigo de Simula_cascada_sobre_no_lineal_LabVIEW parte 12

```
%% Creamos CSV

U = entrades_cascada.signals.values(:,1);
Qf = entrades_cascada.signals.values(:,2);
Tf = entrades_cascada.signals.values(:,3);
Tce = entrades_cascada.signals.values(:,4);
Tref = Tref.signals.values;
Tpm = entrades_cascada.signals.values(:,5);

dlmwrite('entrades_cascada.csv',[U,Tce,Qf,Tf,Tpm,Tref
    ],';')

%para_LabVIEW = [K1,K2,K3,K4,tau1,tau2,tau3,K11,K21,
    K22,K32,K43]; (12)

dlmwrite('data_in_cascada.csv',[para_LabVIEW,
    step_size_continu,X1RE,X2RE,X3RE,Kem,Kem2,taum,
    taum2],';');

dlmwrite('param_PID_master.csv',[K,Ti,Td,alfa,beta,
    gamma,awm,Kb,T],';');
dlmwrite('param_PID_slave.csv',[K_s,Ti_s,Td_s,alfa2,
    beta2,gamma2,awm2,Kb2,T],';');
```

Analizando lo mostrado en el Listing 3.28, observamos como generamos los archivos CSV, para la posterior generación de la planta virtual en posteriores capítulos Si observamos detenidamente vemos como la matriz para_LabVIEW está comentada, puesto que los parámetros que generan la planta virtual no cambiaran, y los tendremos de la ejecución del código descrito en la subsección 3.5.1. Si el ingeniero decidiera ir directamente a un regulador en cascada debería descomentar esta matriz para obtener los archivos.

Capítulo 4

Elección del regulador teórico

Tras el diseño de los reguladores mediante el programa de calculo Matlab descritos en el Capítulo 3 pasaremos a realizar un estudio teórico-económico de cada uno de ellos con el fin de comparar cual seria mas conveniente para nuestro sistema. Este estudio tendrá en cuenta los costes que necesitamos tanto para la inversión como para el mantenimiento, aunque dará una especial relevancia al funcionamiento de la planta. En este estudio dejamos a un lado los costes que al ingeniero le costaría diseñar e implementar uno u otro ya que este proyecto de fin de grado necesita de ambos para realizarse.

4.1 Lugar geométrico de las raíces

En esta sección analizaremos el lugar geométrico de las raíces de los sistemas antes de la compensación y después de la misma. Realizaremos un estudio de los reguladores de lazo abierto, donde veremos el tipo de planta a compensar, y el lugar de las raíces tras cerrar el lazo realimentado con los distintos sensores. Compararemos los distintos reguladores diseñados en el Capítulo 3

4.1.1 PID estándar

Tras el diseño del regulador en la subsección 3.5.1 obtenemos una seria de valores que deberemos implementar para obtener un regulador PID estándar con una respuesta aceptable. El diseño tomado para este regulador a sido el del zero doble, el cual consiste en asignar dos zeros en la misma posición con el fin de atraer al mayor numero de ramas generadas por los polos y/o curvarlas para evitar en la medida de lo posible el semiplano real. Los criterios tomados para el diseño han sido un tiempo de establecimiento de $50s$ y un pico de sobrepasamiento inferior al $10\,\%$.

Este método se ha seleccionado tras observar el comportamiento de los brazos en la gráfica del lugar geométrico de las raíces, la cual podemos observar en la figura 4.1. Esta imagen nos da un comportamiento bastante inestable con una curvatura pronunciada que fuerza que las ramas entren en el semiplano real, dificultando el control. Si a esto le sumamos un nuevo polo en el origen debido al integrador, con el fin de evitar así el error estacionario, empeoraremos aun mas la situación, quedando una planta como la que se observa en la figura 4.2. Aplicando el método

del zero doble se han asignado en la posición c de $-2{,}144$ con una constante derivativa de Kd igual a 2,955.

Para realizar este método se ha seguido la bibliografía de Ogata (2002, Capítulo 10, pág. 681), calculando el criterio del argumento y del módulo. Tras el calculo del regulador obtenemos la siguiente función:

$$G_c = K_d\frac{(s+c)^2}{s} = 2{,}9555\frac{(s+0{,}2144)^2}{s} \tag{4.1}$$

Si desglosamos el regulador, ecuación (4.1), en las constantes a implementar en el autómata obtenemos unos valores de:

- K_p: 1,2675
- K_i: 0,1359
- K_d: 2,955
- T_i: 9,3271
- T_d: 2,3318

Partiendo de estos valores debemos implementar el regulador que responda a la siguiente ecuación:

$$u = K_p e(t) + K_i\int_0^t e(t)dt - K_d\frac{de(t)}{dt} = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt - K_p T_d\frac{de(t)}{dt} \tag{4.2}$$

Si partimos de la ecuación (4.2) y desglosamos el error (e(t)) es resultado de la diferencia de la referencia o consigna (r) - la lectura del sensor (y(t)) expresado en las mismas unidades, obtenemos la siguiente función:

$$u = K_p\big(r - y(t)\big) + K_i\int_0^t \big(r - y(t)\big)dt - K_d\left(\frac{dr}{dt} - \frac{y(t)}{dt}\right) \tag{4.3}$$

Iniciando desde la ecuación (4.3) y mejoramos el funcionamiento de este regulador aplicando la ponderación de la derivada (γ) y de la referencia (β) obtendremos ademas un regulador que responde a la siguiente ecuación:

$$u = K_p\big(\beta r - y(t)\big) + K_i\int_0^t \big(r - y(t)\big)dt - K_d\left(\gamma\frac{dr}{dt} - \frac{y(t)}{dt}\right) \tag{4.4}$$

Si a esta expresión anterior (4.4) le aplicamos el filtro de la derivada (α) podemos obtener la siguiente expresión:

$$K_p\big(\beta r - y\big) + K_i\int_0^t e(t)dt - \alpha K_d\frac{dy(t)}{dt} \tag{4.5}$$

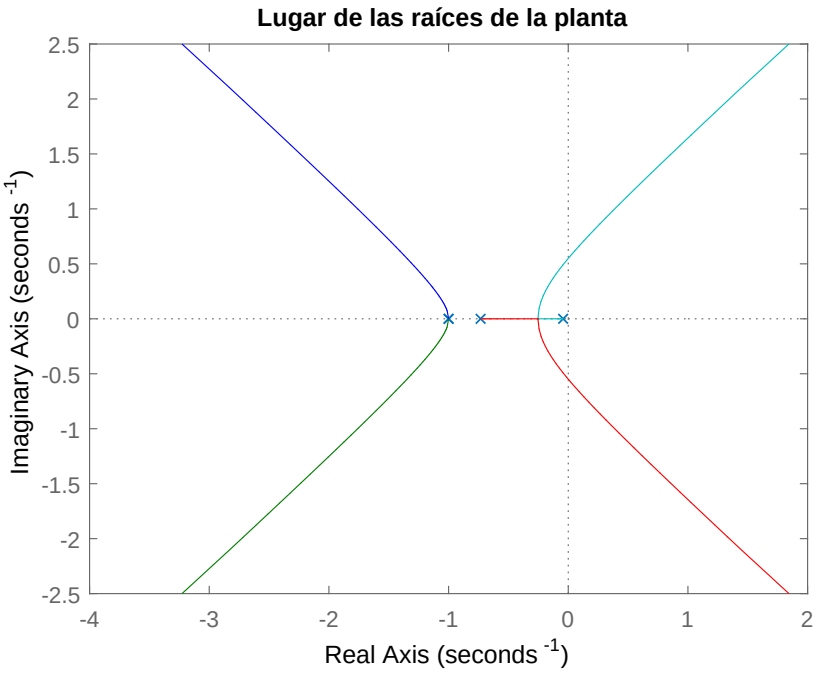

Figura 4.1: Lugar de las raíces de la planta

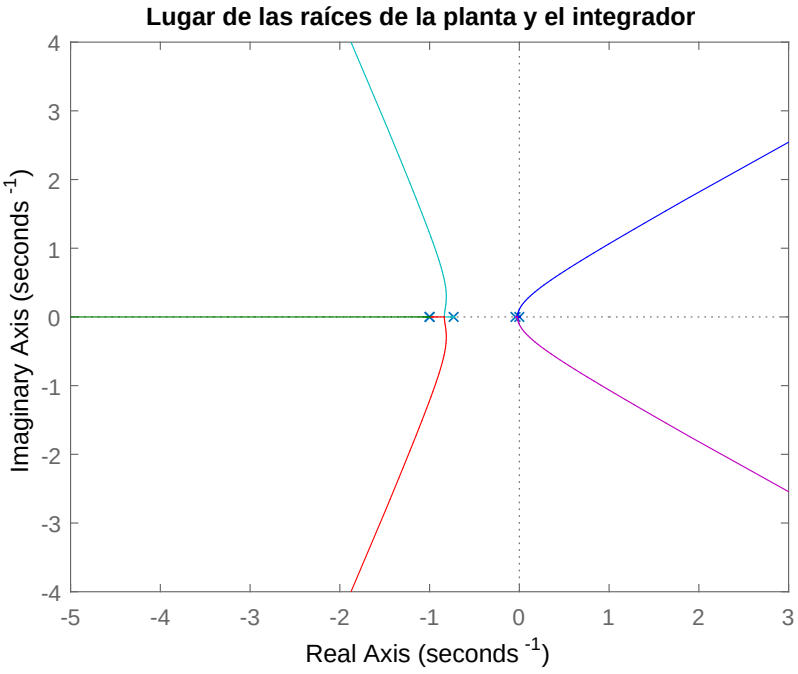

Figura 4.2: Lugar de las raíces de la planta y el integrador

Si estudiamos los valores típicos de diseño nos damos cuenta que la variación de todos ellos oscila entre 0 y 1. Este filtro de la derivada amortiguara el ruido de la señal de la acción, rediciendo los posibles errores surgidos de señales externas de baja frecuencia, que el autómata podría interpretar como una perturbación y reaccionar ante ellos. Si analizamos las ecuaciones (4.4) y (4.5) sera necesaria solo una de los dos factores de la derivada (γ o α), y en este diseño se ha optado por el filtro de la derivada (α). Los valores que hemos de implementar son los siguientes:

- α: 0,2
- β: 0,2
- γ: 0

4.1.2 PID cascada

Este diseño se trato en la subsección 3.5.2, con el que implementamos un segundo lazo de control interno intentando mejorar la respuesta de la planta ante la perturbación de la modificación de la temperatura de la caldera. Este lazo interno intenta corregir las perturbaciones antes de que afecten a la temperatura del tanque, mejorando así el proceso a controlar. Para este tipo de controles avanzados tenemos "dos reguladores" conectados en serie, el primero de ellos o master es el que lee la variable final a controlar, en este caso la temperatura del liquido A del tanque, mientras que el secundario o slave lee la variable rápida que genera perturbaciones, en este caso la temperatura de la caldera que calienta el liquido B. El regulador primario es el que seleccionara la consigan del secundario el cual estará realimentado con el sensor de la caldera, pero su acción de control si actuara sobre la propia planta, en este caso sobre la válvula que controla el caudal de la camisa (válvula B). En la figura 4.3 encontramos un esquema genérico de un regulador en cascada.

Se decide realizar un control en cascada ya que cumplimos con una serie de condiciones que favorecen este control. Las condiciones principales que enumeran la mayoría de autores que favorecen los controles en cascada sobre otro tipo de controles en cascada son:

- El control de un solo lazo no es satisfactorio ante ciertas perturbaciones
- Se dispone de la medida de una variable secundaria que indica una perturbación importante
- Hay relación causal entre la acción de control y la variable secundaria medida
- La dinámica de la variable secundaria es mas rápida que la dinámica de la variable principal

Para este regulador se han tomado unos criterios distintos a los elegidos en la subsección 4.1.1, ya que ahora se ha seleccionado un tiempo de establecimiento de $18s$ y un pico de sobrepasamiento máximo del 20 % para el lazo principal, y un tiempo de establecimiento de $10s$ y un pico de sobrepasamiento máximo del 0,1 % para el lazo secundario. Estos valores son mucho mas restrictivos que los del PID estándar ya que ahora tendremos una compensación extra ante las perturbaciones de la caldera.

A la hora del diseño del regulador secundario se barajaron distintas posibilidades, una de ellas seria la cancelación de los polos, diseño que consiste en ajustar el zero en la misma posición que los polos mas predominantes, intentando evitar así que estos entren en el semiplano real. La

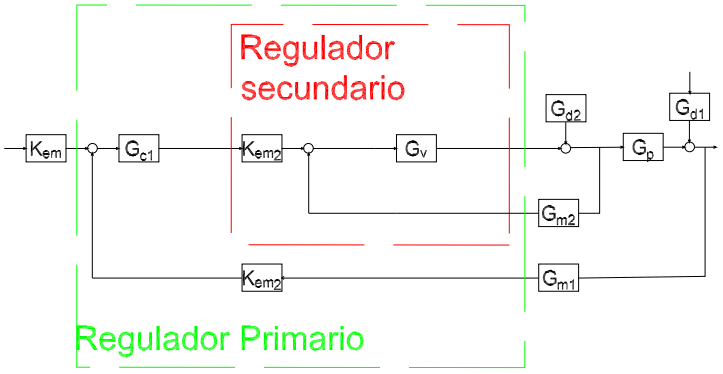

Figura 4.3: Diagrama de bloques de una planta con un controlador en cascada

otra posibilidad es la de compensación situando los zeros en lugares estratégicos para curvar las ramas favoreciendo el control. Si analizamos el lugar de las raíces en lazo abierto de la planta que ha de compensar este regulador observamos que no implica a todos los bloques, sino solo a aquellos sobre los que actúa. Es por ese motivo por el cual en la figura 4.4 tenemos representado una gráfica del lugar del lugar geométrico de las raíces en lazo abierto distinta a la mostrada en la figura 4.1. A esta gráfica abría que añadirle el polo en el origen ya que se trata de un regulador PI, e intentar compensar todos los polos del sistema a excepción de este.

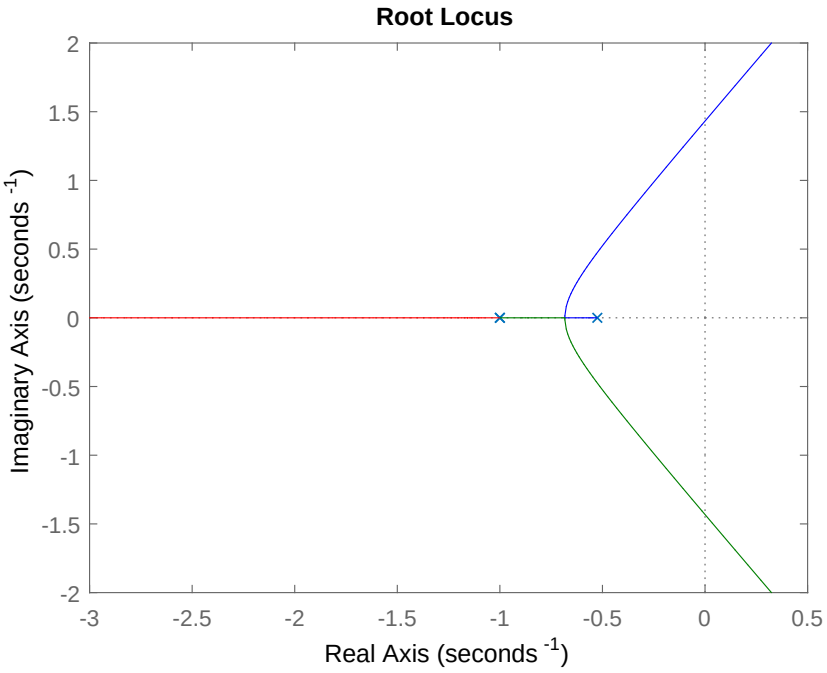

Figura 4.4: Lugar geométrico de las raíces del lazo secundario

Si pasamos a comparar los diseños de este regulador secundario, uno realizado por el método de cancelación y otro por el de compensación, podemos observar las distintas respuestas de la planta tras la compensación. En la figura 4.5 observamos la planta compensada en lazo cerrado con el regulador diseñado por el método de cancelación. En la figura 4.6 se observa el la planta en lazo cerrado compensada por el segundo método Ambas parecen idénticas pero si atendemos a las ecuaciones que definen estos reguladores observaremos que no lo son.

Observando la Ecuación 4.6, realizada por el método de cancelación, podemos ver que los coeficientes de esta ecuación son algo distintos a los de la Ecuación 4.7, regulador diseñado por el método de compensación. Debido a la situación de los polos en lazo abierto (figura 4.4) en el que solo encontramos dos, se ha decidido utilizar el método de cancelación.

$$G_{c2} = \frac{0{,}34547(s + 0{,}525)}{s} \tag{4.6}$$

$$G_{c2} = \frac{0{,}37922(s + 0{,}4857)}{s} \tag{4.7}$$

Si ahora pasamos a realizar una mejora en este regulador secundario de manera análoga a la realizada en la subsección 4.1.1 aplicaremos la ponderación de la referencia (α) igual a $^1/_3$

Una vez contamos con el diseño del regulador esclavo, pasaremos a diseñar el regulador primario. Debemos tener en cuenta que el controlador secundario o slave forma parte del lazo abierto que deberá compensar el regulador maestro, tal y como se muestra en la figura 4.7. Para el regulador primario este lazo de control formado por el sensor y el regulador secundario sera un bloque mas a compensar, por ese motivo debemos aplicar el álgebra de bloques al lazo de control secundario (no confundir con regulador secundario) tal y como se observa en la siguiente ecuación:

$$M_2 = K_{em2} \frac{G_{c2}}{1 + G_{c2} G_{m2}} \tag{4.8}$$

Una vez hemos transformado el lazo secundario de control en un único bloque podemos decir que el lazo abierto del regulador primario sera el producto de la válvula (G_v), el lazo de control secundario ($M2$), adaptación de la referencia (K_{m1}) y sensor primario (G_{m1}). Si atendemos al lugar geométrico de las raíces de este sistema (figura 4.8) tendremos uno mucho mas complejo que el el regulador PID estándar descrito en la subsección 3.5.1, puesto que ahora contamos con polos complejos conjugados, pero también con zeros en el sistema que ayudan a la compensación.

Continuando con el método normal de diseño deberemos añadir un polo en el origen para implementar un regulador con parte integral, obtendremos un sistema cuyo lugar geométrico de las raíces es el que se muestra en la figura 4.9.

Tras observar el lugar geométrico de las raíces del sistema que debemos compensar hemos optado por dos posibles métodos de diseño, el primero de ellos el zero doble, explicado anteriormente en la subsección 4.1.1, o el método de cancelación, que consiste en implantar los zeros en la posición de los polos mas dominantes (obviando el integrador) eliminando así la rama con curvatura hacia el semiplano real o curvando las ramas en la medida de lo posible alejándolas de este.

Si realizamos ambas compensaciones del sistema con el fin de comparar los lugares geométricos de las raíces podemos observar que en la figura 4.10 quedaran unos polos complejos conjugados en el semiplano derecho, creando un sistema sobreamortuguado que oscilara de forma creciente. Este tipo de diseño queda descartado con las limitaciones impuesta, pudiendo realizarse este tipo de control si suavizaremos los tiempos de establecimiento de ambos reguladores, y sobre todo el pico de sobrepasamiento. Otra característica a tener en cuenta es que el zero del regulador secundario elimina el integrador, por ese motivo el regulador ahora es inestable si no se añade un polo para evitarlo.

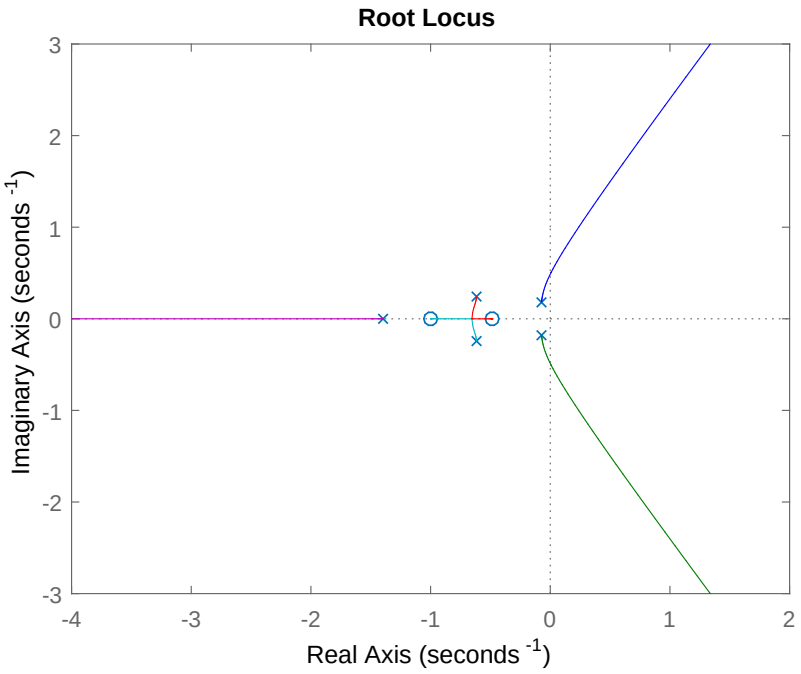

Figura 4.5: Lugar cancelación

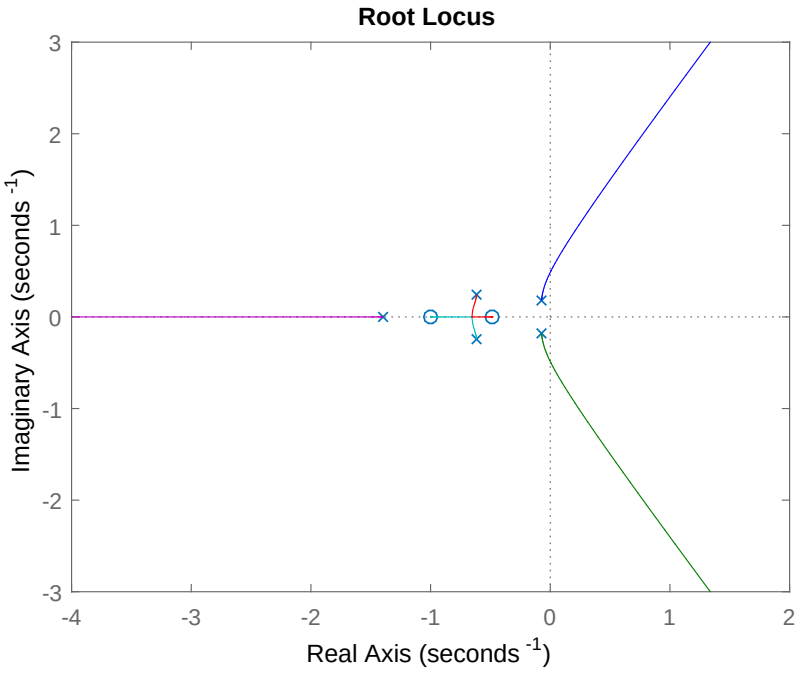

Figura 4.6: Lugar compensación

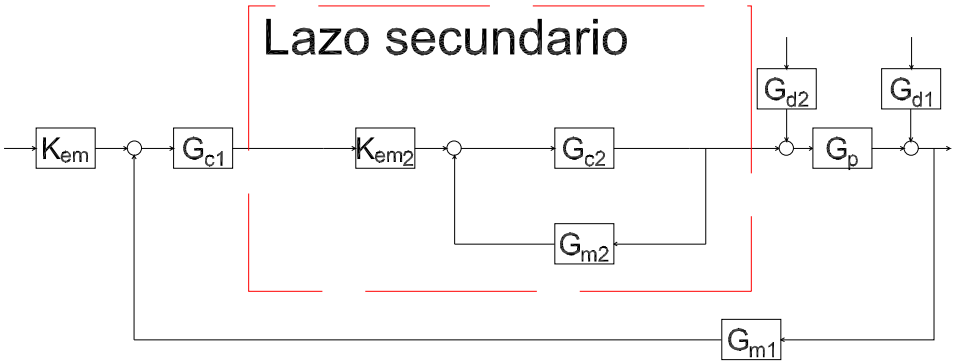

Figura 4.7: Lazo de control secundario

Por otro lado tenemos la figura 4.11 en la que podemos observar un sistema compensado sin polos en el semiplano derecho. Intuimos que es rápido ya el corte con el eje es próximo al origen. Por este motivo se realizara el diseño del regulador primario por el método del zero doble.

Si extraemos la ecuación del regulador primario podemos observar que el cero doble se a posicionado en la ordenada real 0,0911:

$$G_c = \frac{10{,}677(s + 0{,}0911)^2}{s} \tag{4.9}$$

De manera análoga al regulador secundario se añadirán una serie de valores para mejorar el funcionamiento del mismo:

- α: 0,1
- β: 1
- γ: 0

Partiendo de la Ecuación 4.9, podremos extraer los valores del regulador en cascada:

- K_p: 1,9454
- K_i: 0,0886
- K_d: 10,6774
- T_i: 21,9537
- T_d: 5,4884

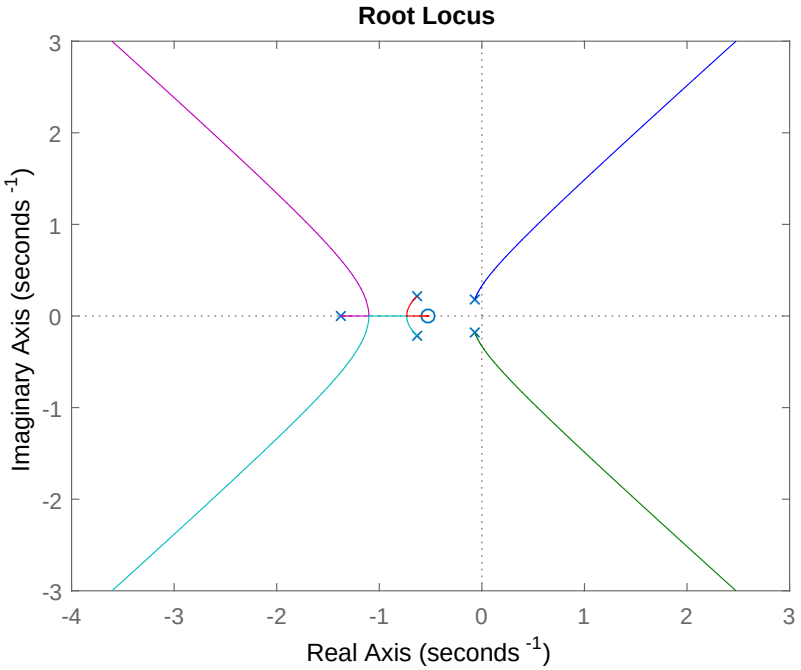

Figura 4.8: Lugar geométrico de las raíces a compensar por el regulador primario

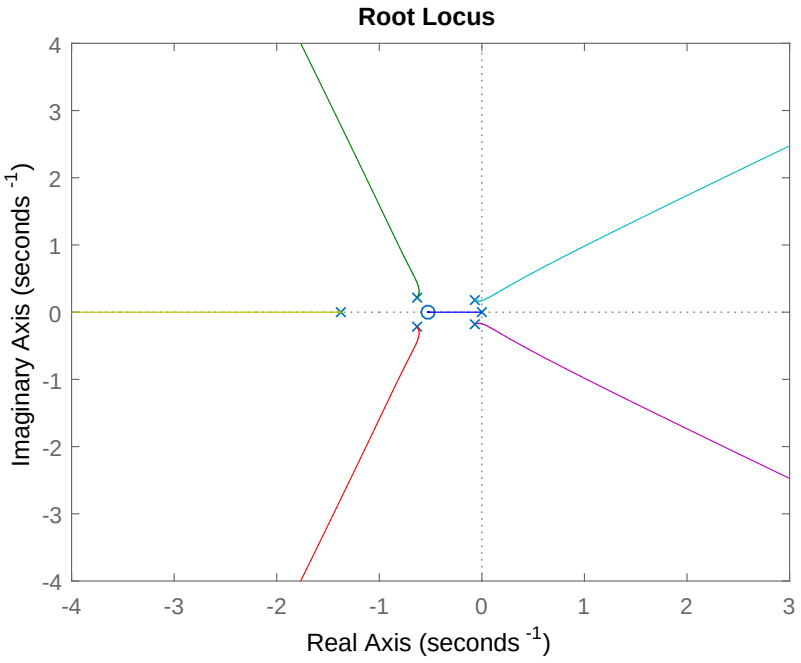

Figura 4.9: Lugar geométrico de las raíces a compensar por el regulador primario con el integrador en el origen

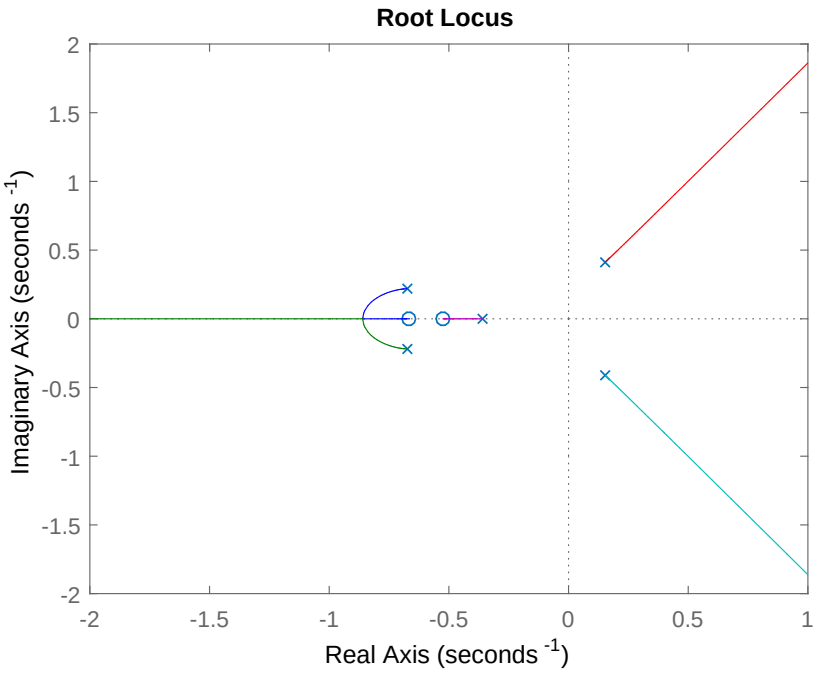

Figura 4.10: Sistema compensado mediante el método de cancelación

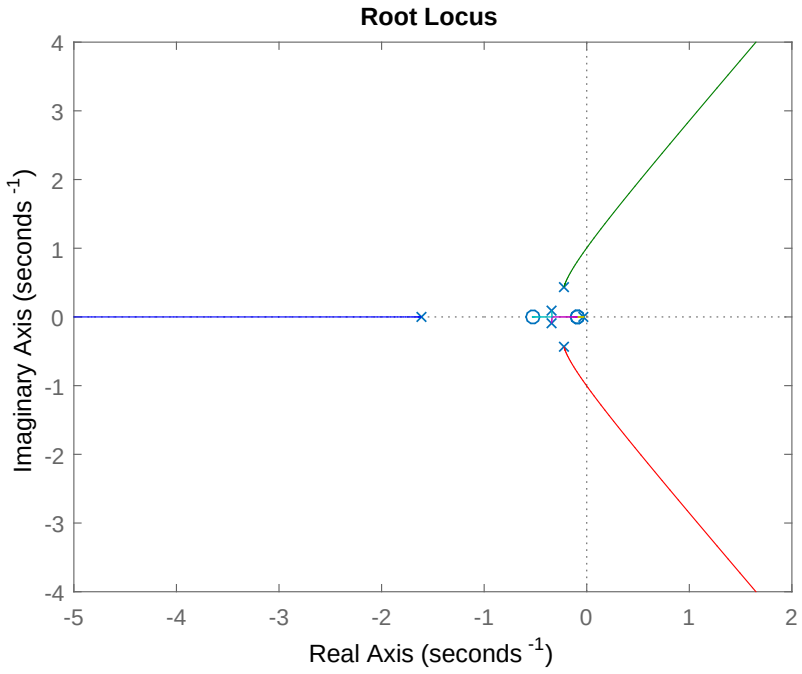

Figura 4.11: Sistema compensado mediante el método del zero doble

4.2 Comportamiento de la simulación

Una vez hemos extraído el diseño de dos reguladores pasaremos a ver el comportamiento teórico que estos tendrán ante la planta.El fin de esta sección es conocer los puntos a favor y en contra que ambos reguladores presentan. Para ello se realizaran simulaciones (ya configuradas en el Capítulo 3) bajo las mismas condiciones intentando ver así el comportamiento que tendrán.

Es muy importante que ambos cumplan con los requisitos preestablecidos de diseño, como son el tiempo de establecimiento y el pico de sobrepasamiento.

4.2.1 PID estándar

Analizando la simulación mostrada en la figura 4.12, podemos observar como el regulador cumple las funciones especificadas:

- Tiempo de establecimiento: $50s$
- Pico de sobrepasamientos: $10\,\%$

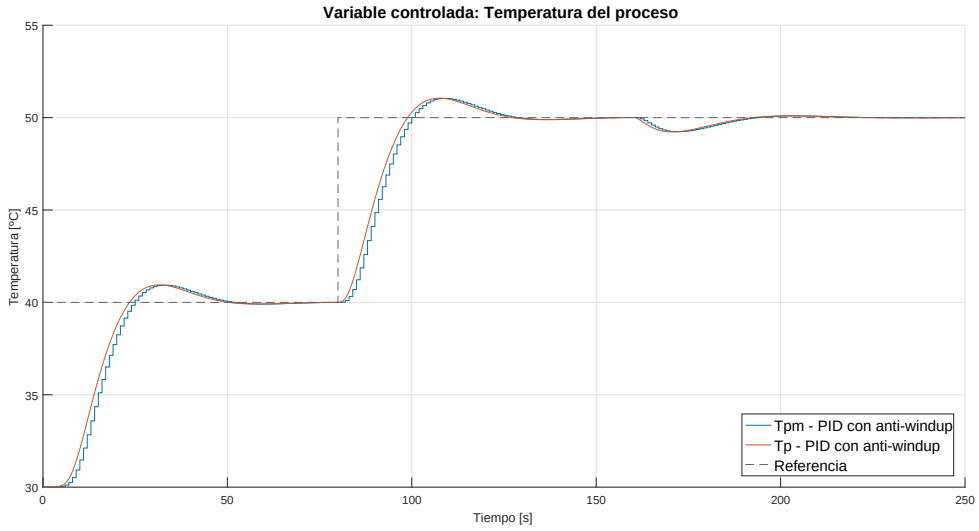

Figura 4.12: Simulación de un regulador PID estándar ante la planta teórica

Podemos observar que el pico de sobrepasamiento queda bastante por debajo del limite teórico exigido ya que en el primero escalón de referencia este queda por debajo de los 44^oC marcados como limite. Por otro lado vemos que la respuesta ante las perturbaciones no es tan bueno, ya que ante una perturbación configurada menor al $17\,\%$ en la temperatura de la caldera se reduce 25^oC tarda en recuperarse y cuenta con un pico de sobrepasamiento grande.

Por otro, casi no es apreciable la perturbación en la temperatura medida por el sensor primario y la respuesta de recuperación es excelente.

Si comparamos la gráfica observamos un sistema continuo y otro discreto. Esta gráfica esta mostrando la simulación realizada en color rojo, y los datos que enviaremos al programa de la planta virtualizada para poder comparar los datos.

4.2.2 PID cascada

Ahora realizaremos la comparación de las distintas configuraciones del regulador en cascada, extrayendo los resultados de las simulaciones configuradas en los programas descritos en la subsección 3.5.2. Las distintas configuraciones serán:

- secundario diseñado por el método de compensación y primario por el de cero doble
- secundario diseñado por el método de compensación y primario por el de cancelación
- secundario diseñado por el método de cancelación y primario por el de cero doble
- Ambos reguladores diseñados por el método de cancelación

Para todas estas configuraciones tendremos las mismas restricciones de diseño:

- Regulador secundario:
  - Tiempo de establecimiento: $10s$
  - Pico de sobrepasamientos: 1 %
- Regulador primario:
  - Tiempo de establecimiento: $18s$
  - Pico de sobrepasamientos: 20 %

Compensación - zero doble

En este punto veremos la respuesta del controlador ante una serie de perturbaciones, tanto de la temperatura de la camisa como de la caldera y analizaremos la respuesta de la simulación ante los distintos cambios de referencia de tipo escalón.

Si observamos la figura 4.13, podemos ver que cumplimos de sobra con las restricciones marcadas para el pico de sobrepasamiento, ya que en el primer cambio de referencia la temperatura, no superamos la referencia por lo que el pico de sobrepasamiento seria negativo y cumpliríamos la limitación. Por otro lado cuando el regulador se encuentra ante una perturbación vemos que si sobrepasarían la referencia, pero en ningún caso llegamos a los 41^oC, sobrepasamiento que se encuentra por debajo del 20 % limite para el primario (figura 4.14).

Si atendemos al criterio del tiempo de establecimiento podemos observar que en los primeros $20s$ de ciclo la referencia alcanza un nivel de $39{,}35^oC$ (figura 4.15), nivel mas que aceptable para un cambio de referencia tan brusco. Si analizamos estadisticamente ese error, considerando que un error de 10^oC (cambio de temperatura del escalón) es un error del 100 %, nos encontramos en un error del 6,5 %, consideraremos que ha llegado a la referencia en el tiempo acordado.

Por todos estos motivos consideramos que este regulador cumple su función adecuadamente, y se tendrá en cuenta a la hora de la elección del mismo.

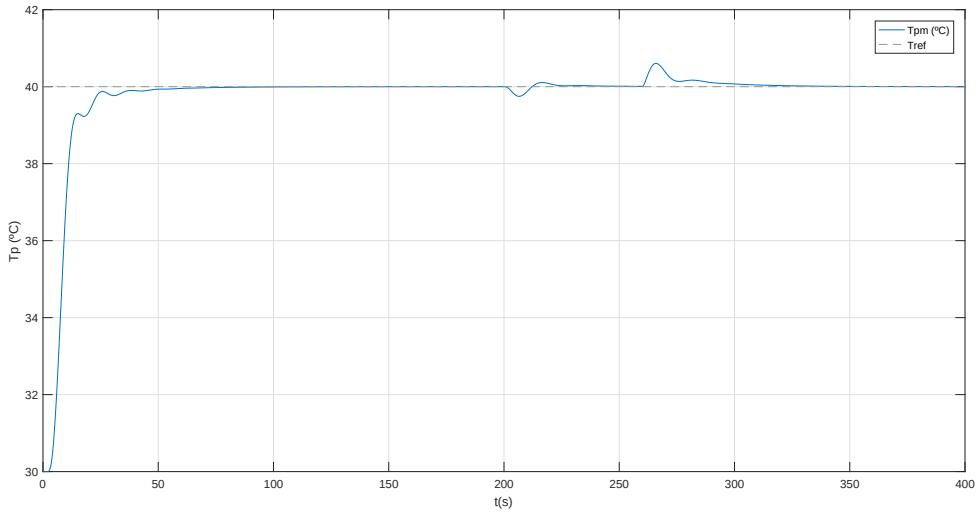

Figura 4.13: Simulación de un regulador PID cascada ante la planta teórica. Método compensación - zero doble

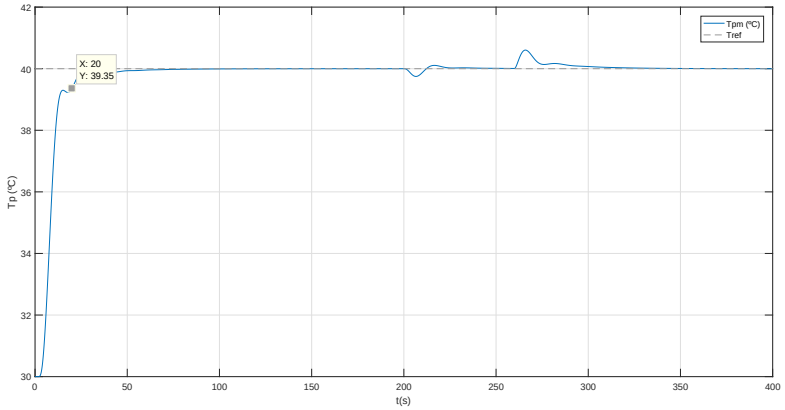

Figura 4.14: Tiempo de establecimiento

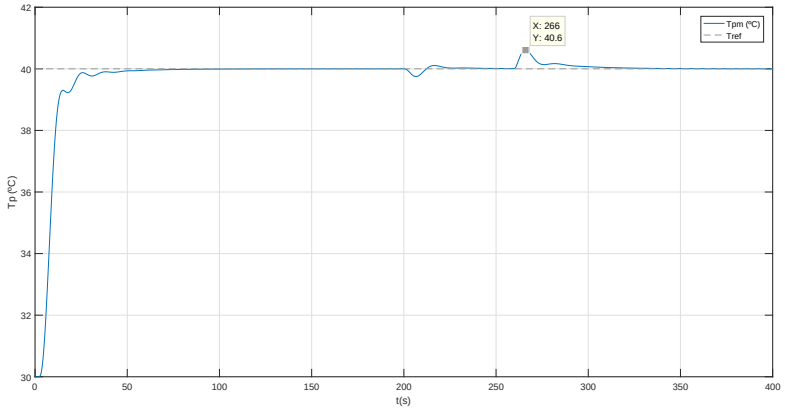

Figura 4.15: Pico de sobrepasamiento

Compensación - cancelación

En este punto analizaremos la respuesta de un regulador en cascada diseñado por el método de compensación para el secundario y cancelación para el primario. La respuesta de este controlador la podemos observar en la figura 4.16, donde observamos tal y como se pronostico en la sección 4.1, una amortiguación con la que nunca llegaremos a la referencia. En este regulador jamas llegaremos a la referencia puesto la amortiguación va creciendo. Como podemos observar no cumplimos con el tiempo de establecimiento establecido, y en este caso la perturbación en lugar de empeorar la planta la mejora.

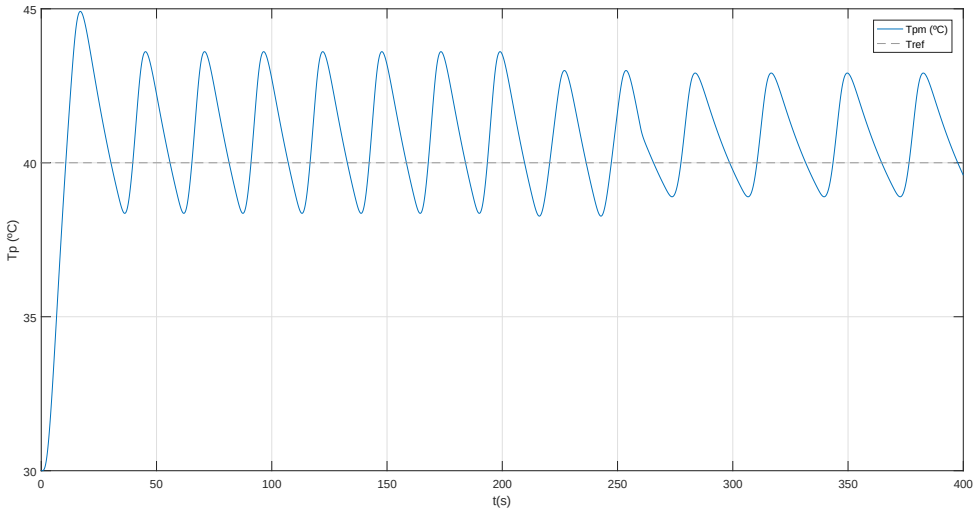

Figura 4.16: Simulación de un regulador PID cascada ante la planta teórica. Método compensación - cancelación

Cancelación - zero doble

En este caso volvemos a analizar un regulador en cascada en el que el secundario lo hemos diseñado por cancelación, y el primario por el método del zero doble. Analizando la figura 4.17, observamos que el regulador, esta vez si, llegaría a la referencia, al contrario que en el controlador diseñado en la subsección 4.2.2.

Si analizamos el pico de sobrepasamiento nos encontramos en un regulador muy parecido al diseñado en la subsección 4.2.2, en el que cumplimos con los criterios establecidos para el regulador primario, tal y como se observa en la figura 4.19. Si analizamos ahora el tiempo de establecimiento nos damos cuenta que en este caso, aunque llega a la referencia, no cumple con el criterio ya que en el tiempo marcado de $20s$ se encuentra en una temperatura de $37{,}43^{o}C$, tal y como observamos en la figura 4.18, equivaliendo a un error del $25{,}7\,\%$.

Con todo ello podremos considerar este regulador para una futura implantación, ya que el global del funcionamiento es relativamente aceptable, y cumple con todas las especificaciones marcadas a excepción del tiempo de establecimiento, sobrepasando los limites. Es cierto que esta planta tiene una inercia elevada, por lo que entenderíamos que el regulador fuese aceptado por un hipotético cliente ya que los tiempos de la planta tenderían a ser mucho mayores, pudiendo suavizar este criterio y cumpliendo con lo establecido.

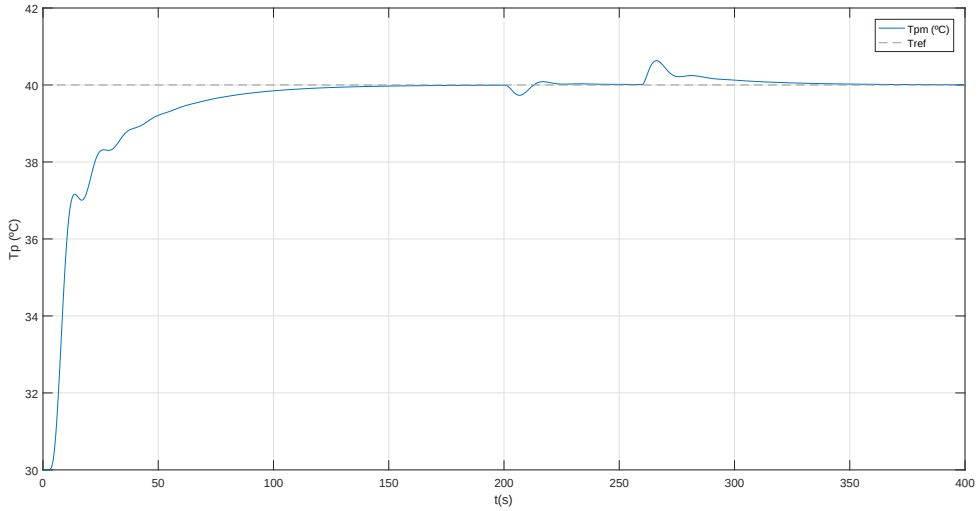

Figura 4.17: Simulación de un regulador PID cascada ante la planta teórica. Método cancelación - cero doble

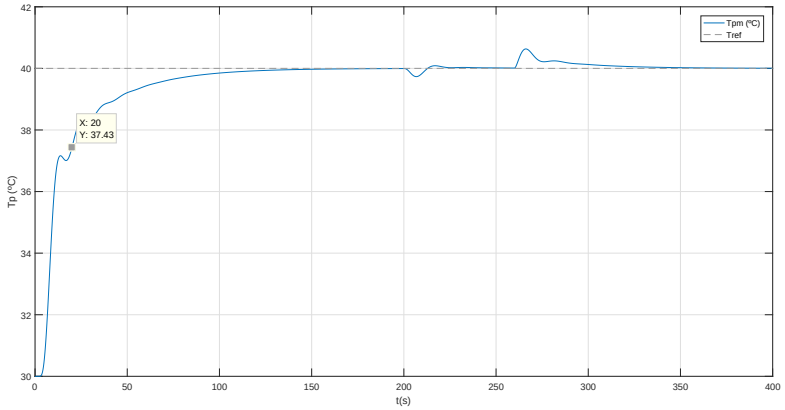

Figura 4.18: Tiempo de establecimiento

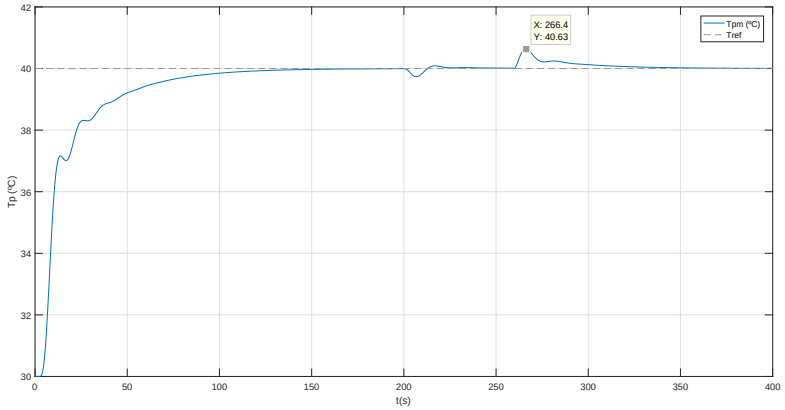

Figura 4.19: Pico de sobrepasamiento

Cancelación - cancelación

Por ultimo, analizaremos los resultados obtenidos tras la simulación del regulador e cascada tomando como diseño el método de cancelación tanto en primario como en secundario. Tal y como se cito en subsección 4.2.2, cuando el regulador primario esta diseñado por el método de cancelación se generan una serie de polos en el semiplano real que hacen que el sistema se vuelva amortiguado. Nuevamente descartamos este regulador ya que no funciona, ni lleva a la referencia a la planta, ni cumple los criterios de sobrepasamiento. Para cumplir con este método de diseño necesitaríamos rebajar los criterios de diseño, evitando así la aparición de polos en el semiplano derecho.

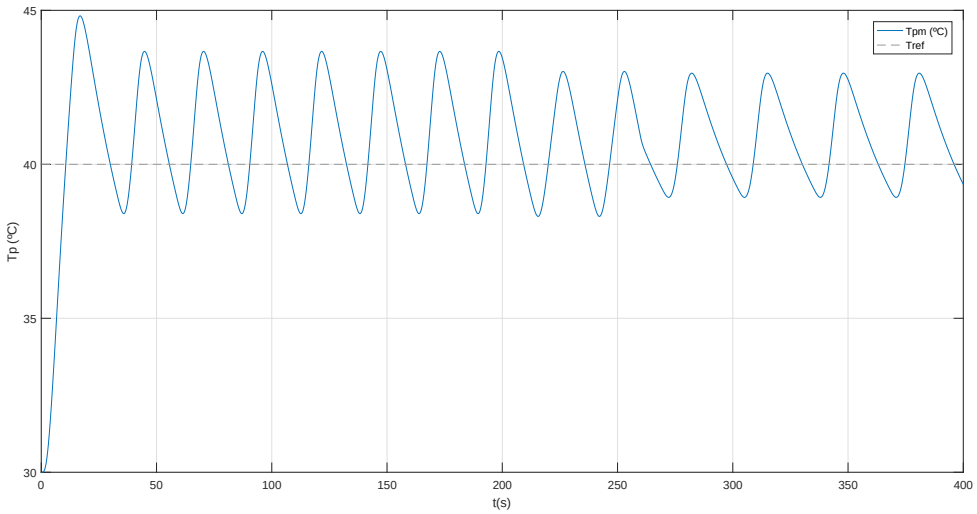

Figura 4.20: Simulación de un regulador PID cascada ante la planta teórica. Método cancelación - cancelación

4.3 Señal de control

En este punto analizaremos la señal de control que el regulador dará a nuestra válvula para controlar el sistema. Es de vital importancia analizar las señales que mandaremos debido a que los picos de tensión, o los cambios bruscos de la misma pueden producir un envejecimiento prematuro de la válvula, acortando así su vida útil. Sera de especial relevancia analizar la señal de control ante los cambios de referencia y ante las perturbaciones, ya que estas serán las encargadas de producir las aculturaciones del controlador, que se traducirán en los picos de tensión que aplicaremos a la válvula.

4.3.1 PID estándar

En este punto analizaremos la señal de control de la válvula que se genera con un regulador estándar del tipo PID, diseñado en la subsección 3.5.1 y analizado el diseño en la subsección 4.1.1. El diseño de la señal de control cuenta con unos limites marcados. Estos limites, definidos en la subsección 1.5.2, corresponden a la tensión máxima de entrada $10V$ y la mínima de salida $0V$, variando en ese grado la apertura de la válvula.

En al figura 4.21, observa la señal teórica que la válvula recibirá, observando que cumple con el criterio de tensión máxima y mínima. Si observamos la señal, podemos fijarnos que esta es

de tipo discreto, es decir, la señal va variando de tensión a saltos discretos en el tiempo con un periodo constante. Este dato sera de especial relevancia ya que el motor de la válvula que instalemos ha de ser capaz de soportar este tipo de saltos. Normalmente estas válvulas cuentan con un servo, o un motor paso a paso los cuales deben ser capaces de soportar los escalones.

Si analizamos ahora la variación mas alta de un punto a otro de tensión, en valor absoluto, podemos asegurar que es de 3,2128V. Este análisis se ha llevado acabo con un bucle FOR diseñado tras la ejecución del programa de Matlab que diseñamos en la subsección 3.5.1. Este bucle lo mostraremos en el Listing 4.1 y no se añadirá al programa principal, ya que se ejecutara posteriormente en el caso de que fuese necesario.

Listing 4.1: Código de simula_PID_sobre_no_lineal_LabVIEW parte 1

```
UsMAX = 0;

for i = 1 : numel(Us)

        if i == 1
                Uscomp = Us(i,1);
        else
                Uscomp = abs(Us((i),1) - Us((i-1),1));
        end

        if UsMAX < Uscomp
                UsMAX = Uscomp;
        end

end

disp('La variación de tensión máxima en la válvula es
    de: ')
disp(UsMAX)
```

Tras este análisis concluimos que las variaciones de tensión ejecutadas a la válvula son aceptables, dando por valido el diseño del regulador y la tensión aplicada al componente, pudiendo ser este el limitante. Por otro lado quedara comprobar si el autómata en el que se compile el regulador es capaz de variar la tensión de este modo, o por el contrario estas variaciones son demasiado bruscas para el.

4.3.2 PID cascada

Tras analizar los distintos diseños de los reguladores pasaremos a analizar las señales de control que nos dan. De este modo tendremos toda la información necesaria para decidir que tipo de diseño del regulador, tanto primario como secundario, nos interesaría testar. En esta sección a diferencia de la sección anterior (sección 4.1), los criterios de diseño son los mismos, es decir la tensión máxima de salida del regulador ha de ser inferior a $10V$ y la mínima superior a $0V$. Dividiremos, tal y como en el caso anterior, los reguladores en función de la naturaleza del diseño de los mismos, obteniendo nuevamente la siguiente clasificación:

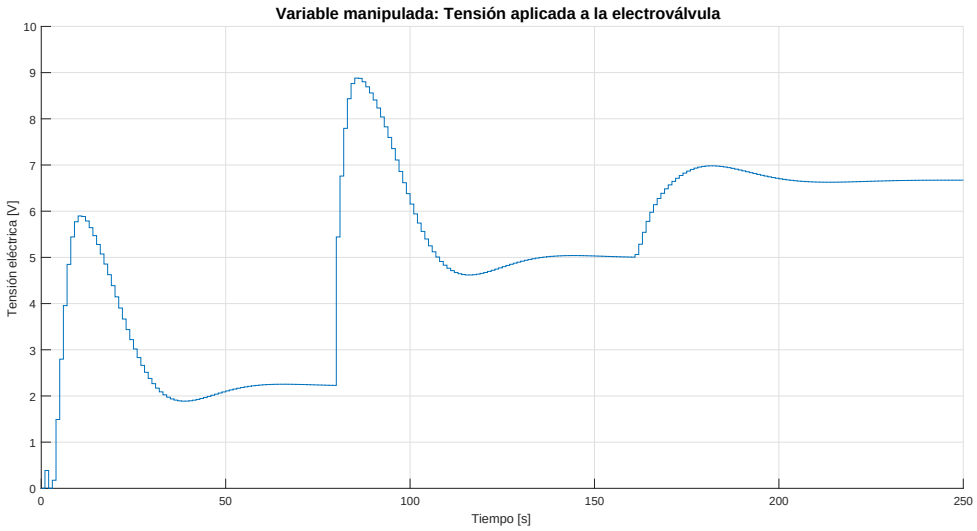

Figura 4.21: Simulación de un regulador PID estándar ante la planta teórica. Señal de control

- secundario diseñado por el método de compensación y primario por el de cero doble
- secundario diseñado por el método de compensación y primario por el de cancelación
- secundario diseñado por el método de cancelación y primario por el de cero doble
- Ambos reguladores diseñados por el método de cancelación

Compensación - zero doble

En esta sección pasaremos a analizar la figura 4.22, donde se muestra la variación de la tensión de salida del regulador. En un principio vemos que la señal satura pero que no excede de los limites teóricos, ni por arriba ni por debajo, por lo que podemos asumir que los resultados serán aceptables.

En cuanto a las variaciones máximas de tensión en valor absoluto modificaremos el código anterior (Listing 4.1), para que muestre los resultados de este regulador. En este caso la variación máxima sera de $5V$, por lo que necesitaremos que el regulador pueda soportar cambios de un 50 % de tensión. En este caso la variación se observa mas suave debido a que se a implementado el algoritmo back-calculation para evitar el windup de la señal. Este algoritmo fue descrito en la subsección 3.2.3.

Compensación - cancelación

En este punto del proyecto analizaremos las variaciones de tensión del regulador diseñado en cascada por el método de compensación, cancelación, para el secundario y primario respectivamente. Para este diseño vimos que el comportamiento era bastante inestable, sufriendo una amortiguación que impedía alcanzar la referencia.

Si analizamos la figura 4.23, observamos que esa amortización oscilante se traduce en una variación de la tensión muy abrupta, saturando tanto por encima como por debajo. Observamos que

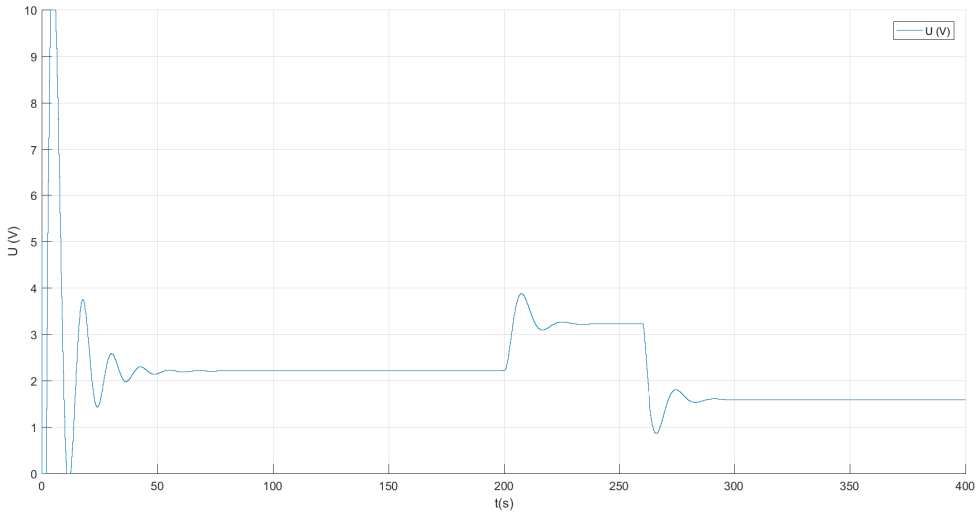

Figura 4.22: Simulación de un regulador PID en cascada ante la planta teórica. Señal de control. Método compensación - zero doble

al menos el diseño de la saturación se cumple no excediendo los límites teóricos marcados para este tipo de regulador, y analizando mediante el código generado obtenemos que la variación máxima en valor absoluto es de $10V$. En el caso de que éste regulador fuese el seleccionado para la implantación en el autómata, este debe ser capaz de variar la tensión un 100 % en periodos iguales al de muestreo, es decir, variar la tensión $10V$ en un tiempo igual a $1s$.

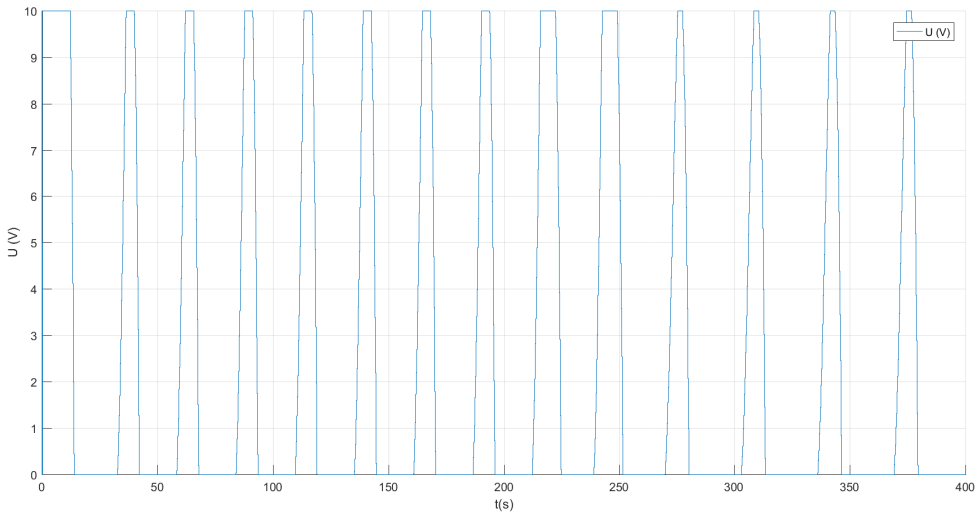

Figura 4.23: Simulación de un regulador PID en cascada ante la planta teórica. Señal de control. Método compensación - cancelación

Cancelación - zero doble

Si analizamos la tensión de este regulador, observaremos nuevamente que se trata de una señal muy parecida a la descrita en la subsección 4.3.2, de igual manera que las secciones subsección 4.2.2 y subsección 4.2.2 donde analizamos el comportamiento de la simulación de ambos.

Si pasamos a analizar la figura 4.24 observamos que sera algo mas suave que la mostrada en la figura 4.22, presentando unas pendientes algo menores en los cambios de tensión. Por otro lado vemos que la variación máxima de tensión sigue siendo de $5V$ referente al primer cambio de la referencia, pero el resto es un poco mas suave produciendo una menor pendiente. Se aprecia en menor medida que el caso de la simulación pero estadisticamente las variaciones y las pendientes son mas suaves.

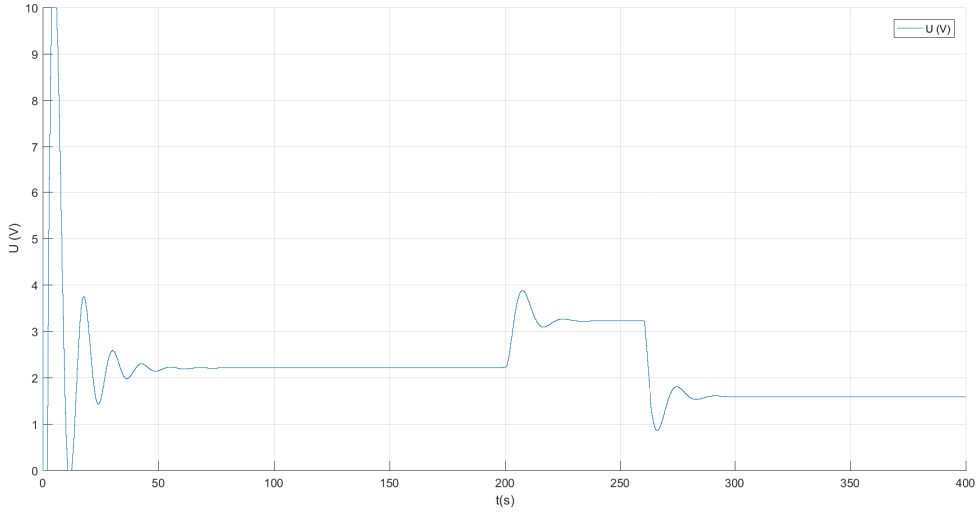

Figura 4.24: Simulación de un regulador PID en cascada ante la planta teórica. Señal de control. Método cancelación - zero doble

Cancelación - cancelación

Si analizamos la salida de tensión del regulador en cascada diseñado por el método de cancelación doble podemos asegurar que este regulador se comportara de una manera muy similar al regulador analizado en la subsección 4.3.2. Tendremos una variación máxima de $10V$ en un inicio.

4.4 Costes

Si pasamos a analizar los costes que llevaría la implementación en el autómata de los distintos reguladores obtendremos una serie de diferencias claras. La primera de ellas, seria la necesidad de la instalación de un segundo sensor de temperatura para controlar la temperatura de la caldera, evitando así que las perturbaciones en este punto afecten a la referencia final. Si decidiéramos instalar este segundo sensor debemos tener en cuenta que la mano de obra subirá, ya que en la caldera actualmente no teníamos ningún sensor y debemos instalarlo desde cero,

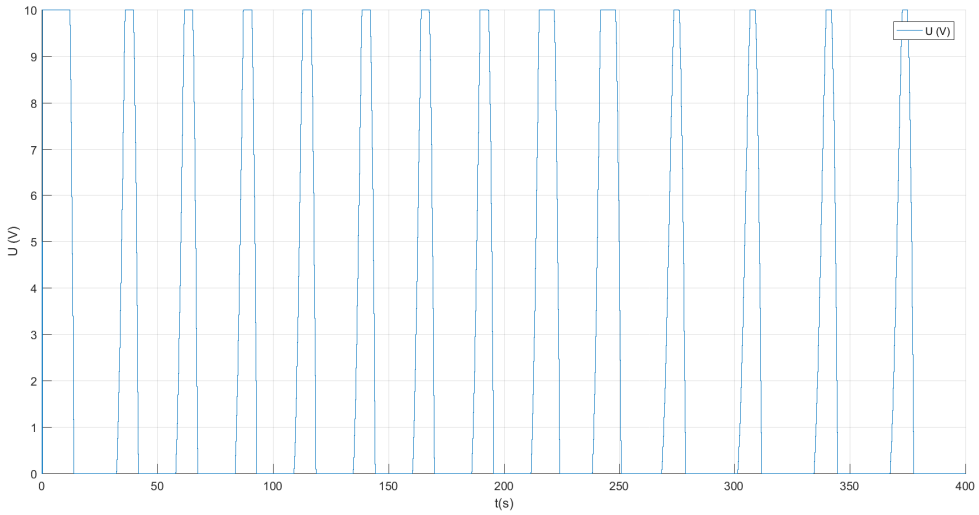

Figura 4.25: Simulación de un regulador PID en cascada ante la planta teórica. Señal de control. Método cancelación - cancelación

con los costes que ello conlleve. Deberemos adaptar la posición del antiguo termómetro por el nuevo, el cual siga controlando la temperatura de la caldera.

Si pensamos ahora en la sustitución de las válvulas, estas estarán presentes en todos los reguladores, por lo que no incrementara ninguno de los reguladores. Por el contrario la instalación del segundo sensor conllevara la instalación de un segundo display que permita al operario conocer la temperatura de la caldera en todo momento. Esta variable no estaba contemplada en la subsección 1.6.2, debiendo adaptar también este punto.

Si analizamos la instalación desde el punto de vista del mantenimiento hemos observado en la sección 4.3 que los reguladores de tipo cascada tienen un comportamiento mas agresivo, es decir, unos picos de tensión mas grandes pudiendo realizar un envejecimiento prematuro tanto del autómata como de la válvula. Deberemos tener instalado un segundo sensor que mejore la respuesta de las perturbaciones, aumentando los costes de los mantenimientos de este sensor.

Por otro lado, desde el punto de vista de la rapidez con la que un regulador estándar y en cascada alcanza la referencia resulta mucho más atractivo la implantación del segundo de estos, puesto que reducimos en mas de la mitad el tiempo de establecimiento con un menor pico de sobrepasamiento, aunque solo en algunos diseños. Este ultimo punto es de especial relevancia debido a que la rapidez con la que alcancemos la temperatura, y el tiempo que podamos mantenerla sera lo que mas beneficios reporte a la empresa.

4.5 elección del regulador

Con todo lo citado hasta ahora, y analizando las simulaciones y señales de control podemos descartar directamente los reguladores en cascada sintonizamos mediante el método de cancelación en el regulador maestro. Esta drástica acción es tomada debido a las siguientes razones:

- Tal y como se cito en la subsección 4.2.2 y subsección 4.2.2, la sintonización por este método genera polos en el semiplano real que transforman la planta en una amortiguada.

- La acción de control del regulador genera picos muy bruscos para que el regulador sea capaz de ejecutarlos. Aunque así fuera, estos picos degenerarían la vida de la válvula y el regulador.

- El sobre coste que genera la instalación de un segundo sensor de temperatura no es equivalente a los beneficios que este reporta con este tipo de regulador, puesto que no se traduce en una aceleración de la señal de control, sino que se traduce en unos inaceptables picos de sobrepasamiento por encima del valor teórico.

Si pasamos ahora a comprar los diseños del regulador en cascada, los cuales el regulador primario de estos se ha sintonizado por el método del zero doble, podemos observar los siguientes puntos:

- El regulador en cascada, el cual el secundario se sintonizó por el método de compensación, presenta un tiempo de establecimiento mucho menor con un pico de sobrepasamiento que no excede los limites marcados.

- El regulador con el secundario diseñado por el método de cancelación, presenta unas transiciones mas suaves, aunque los picos en valor absoluto sean idénticos.

- En ambos casos si compensara la instalación de un segundo sensor de temperatura puesto que mejoramos la velocidad de la planta, aumentando los beneficios de la factoría.

Si comparamos todos los reguladores en cascada, con los argumentos dados nos quedaremos con un único diseño que destaca por encima de los demás, el cual sera el sintonizado por compensación - zero doble. Si ahora analizamos este con el regulador PID estándar obtendremos los siguientes resultados:

- El regulador PID estándar cumple con los requisitos para los que se diseño, pero estos requisitos son menos restrictivos que los del diseño del regulador en cascada.

- El regulador en cascada necesita una mayor inversión, tanto de mantenimiento de los elementos como de instalación de los mismos tal y como se definió en la sección 4.4

- De igual modo este regulador podrá reportar a la empresa un mayor ingreso ya que con el podremos, no solo liberar a un operario poniendo en modo manual la maquina, sino que atemperaremos de una manera mas adecuada el liquido del tanque.

Por todos estos motivos se ha decidido que el regulador en cascada es la mejor opción para la empresa, aunque este suponga un mayor coste inicial y de mantenimiento. Por otro lado y al tratarse de un proyecto docente en futuras secciones se compararan los reguladores PID estándar y PID cascada sintonizado por el método de compensación - zero doble.

Capítulo 5

Comunicación mediante servidor OPC

Este capitulo hablara del protocolo de comunicación que enlazara el autómata con la planta virtual, con el fin de ver así la suficiencia del elemento físico antes de montarlo en la fabrica, asegurando así que todo sera acorde e impactando lo menos posible en el funcionamiento de la misma. En este capitulo partiremos de las variables que necesitemos que entren en el autómata, las cuales si el autómata pasa el test, se introducirán de forma cableada. Y las señales de salida, las cuales deberán entrar en la planta virtual, o posteriormente en el autómata.

5.1 Introducción

El protocolo de comunicación OPC es nativo de Windows, por lo que sera complicado vincular a sistemas de base Linux, o MAC. Existen diversos servidores OPC, entre los que encontramos:

- **OPC-DA**: Data acces. Típica para sistemas SCADA en la que la pantalla del ordenador es la propia pantalla del sistema SCADA.
- **OPC-HDA**: History data acces. Este tipo de sistemas OPC es el mas utilizado para controles de calidad, en los que se va generando una base de datos con los valores que configuremos y podemos consultarlos en cualquier momento.
- **OPC - A&E** Alarms and events. Este tipo de comunicación es la ideal para industrias donde se centralice en un ordenador el control de alarmas o eventos del proceso. Solo servirá si desde el autómata se pueden generar las alarmas, y configurar estas como eventos de menor importancia. Muchos automatismos no son capaces de generar estas alarmas por lo que este tipo de servidor no servirá.
- **OPC-UA**: Unified architecture. Es la suma de todos los servidores anteriores.

El tipo de servidor que en este proyecto se va a generar no esta definido en los servidores estándar, ya que el servidor que se va a configurar permite la lectura y escritura por parte de los clientes en tiempo real. De este modo descartamos los de tipo A&E, por no tratar los datos como alarmas, sino que el autómata los tratara como variables, las cuales se podrán mantener en sistema pudiendo generar archivos de registros de calidad en tiempo real. Esta característica, en tiempo real, impide que la configuración que realicemos sea del modo HDA,

ya que las variables que tratamos han de ser monitorizadas en todo momento y por todos los clientes del servidor. Descartamos los servidores Data Acces, ya que el tipo de servidor solo permite la escritura por parte del autómata, pero en nuestro caso particular necesitamos que escriban todos los clientes para la correcta virtualización de la planta.

Tras analizar los tipos de servidores, no se ha encontrado el cual defina el proceso que se ha seguido para realizar este proyecto, lo cual no indica que sea un nuevo tipo de servidor, sino que este servidor que se a configurado no es el usual. Las configuraciones de este servidor se han tratado en el Capítulo 11, mostrando una pequeña guía de como se ha configurado paso a paso.

5.2 Topología de los servidores

Los protocolos de comunicación clásicos hasta ahora necesitaban de unos ”drivers” comunes para enlazar un dispositivo con otro. Estas topologías eran normalmente de tipo cableadas y punto a punto, es decir, de un dispositivo a otro. Posteriormente se crearon estándares de comunicación en los que se podía interenlazar varios dispositivos a modo de red mayada. Esto fue un gran avance ya que permitía tener un dispositivo maestro con varios esclavos. En la actualidad, la mayor parte de protocolos suelen ser de tipo mallado, o de tipo paralelo en la que desde un mismo ”bus” de comunicación se envíen los distintos mensajes y sea el receptor el encargado de leerlo cuando vaya dirigido a el. De este modo conseguíamos poder tener varios maestros en la misma red o incluso compartir esclavos. El principal inconveniente de este tipo de comunicación es el ”lenguaje” o estándar de la red, ya que no todos los dispositivos ”hablan” el mismo idioma.

Los servidores OPC permiten crear una red con varios buses de comunicación, en los que el servidor recibe todos los mensajes o ordenes, y es el encargado de reenviarlos a los distintos clientes. Esto permite que los clientes no necesiten tener el mismo estándar de comunicación, favoreciendo así la interconexión de elementos de distintas marcas, años, estándares.. en una misma red consiguiendo así abaratar el precio de la automatización de las maquinas, ya que en ocasiones adaptar los drivers para que fuesen comunes era la parte de la automatización mas cara. Otra posibilidad que permite este protocolo de comunicación es enlazar elementos multimarca mejorando el funcionamiento de las maquinas ya que antes debían ser de la misma marca, y estas marcas no estaban especializadas en cierto tipo de elementos que otras empresas si, si ahora podemos enlazar estos elementos podremos generar maquinas de mejor calidad.

La topología básica de los servidores OPC es sencilla, se generaran buses de comunicación, tantos como se necesiten, en los que se enlazaran los distintos elementos. Todos los elementos que cuelguen de un BUS, deben ser comunes al protocolo de comunicación de ese bus. El servidor recibirá los mensajes de todos los buses de comunicación, y el sera el encargado de ”traducirlos” y reenviarlos por el bus adecuado para que el receptor de el mensaje lo ejecute. Dentro de los distintos protocolos, se podrá modificar los parámetro que este permita, pudiendo generar acuses de recibo, bits de paridad, comunicaciones seguras... En la figura 5.1 podemos observar una topología de servidor OPC estándar, en la que encontramos distintos clientes enlazados al mismo servidor, con los distintos lenguajes o estándares de comunicación.

La topología de este proyecto es mas sencilla que la mostrada en la figura 5.1, puesto que unicamente tenemos dos elementos. Uno es el ordenador con la planta virtual, y otro es el autómata físico. En este caso al tratarse de dos únicos elementos podríamos haber testado la suficiencia del autómata con una tarjeta de NI que interconectará la planta virtual con las entradas y salidas analógicas del autómata.

Figura 5.1: Protocolo estándar de comunicación de un servidor OPC

5.3 Drivers de comunicación

En la sección 5.2, hablamos de las distintas topologías de los servidores OPC, y de como se podían interconecte los distintos buses de comunicación a través de el. En esta sección trataremos el tema de los drivers de comunicación que han de enlazar esos buses con el servidor, para que el mensaje llegue a este, o se envié por el mismo. En la actualidad los servidores OPC son ordenadores normalmente con base Windows, a los que les añadimos las tarjetas de comunicación para enlazar los dispositivos. Normalmente estas tarjetas suelen conectarse directamente a la placa madre, por conexiones PCI o PCIe, pero la conexión puede realizarse por cualquier medio "Plug and Play" como un USB. Sera conveniente conocer el COM que el ordenador asigna a los distintos drivers de comunicación, ya que estos deberían mantenerse en el caso de los "Plug and Play" para no modificar los servidores. En cuanto a la comunicación del bus al autómata, suelen venir instaladas y ser las que permiten modificar la comunicación de los mismos

En la época en la que estamos, se esta estandarizando la comunicación con lo autómatas a redes con conexión RJ45, favoreciendo aun mas las comunicaciones OPC. Si todos los elementos que se tienen que conectar al servidor OPC cumplen con esta característica, se podrá diseñar una red que los interconecte en un mismo bus aunque no compartan el estándar de comunicación, configurando el servidor que segmentara los buses para traducir los mensajes y adaptarlos a los estándares del cliente de destino para reenviarlo por la misma red. Si el caso fuese este, se puede generar el servido en un ordenador configurando la tarjeta de red como un USB y pudiendo montar el servidor en cualquier red de la empresa, ya sea un portátil, o una torre.

En nuestro ejemplo se barajo la posibilidad de realizar el servidor en otro ordenador con el fin de demostrar esta teoría, pero por falta de material no se pudo realizar. Nuestro driver de comunicación esta formado por un switch y una tarjeta de red, dando así comunicación OPC como favoreciendo na red y salida a Internet. Con esto se demuestra que la comunicación es efectiva y que no es necesario instalar nuevas tarjetas de red en los servidores OPC sino que la propia tarjeta de los ordenadores es suficiente. Otra posibilidad seria la de conectarse a esta red desde el exterior por medio de VPN y tener localizado el servidor de manera remota. Esto favorecería la comunicación con otras factorías de la red empresarial con las que podrían

ejecutarse estudios o puestas en marcha de las maquinas sin necesidad de mantener gente presente.

5.4 Aplicaciones OPC

Otro punto a favor de la comunicación OPC y de este tipo de servidores son las aplicaciones que se pueden generar. En el Capítulo 11 se a mostrado como vincular las variables del servidor con una hoja de calculo excel. Este tipo de aplicación resulta interesante ya que permite unir programas de labor administrativa con el servidor, ofreciendo así datos en tiempo real con los que poder facilitar las tareas. De este modo podremos mejorar los tiempos de espera y planificación de recursos, puesto que sabemos el estado del proceso. No es lo mismo solicitar un trasporte para una cantidad de productos, que saber la cantidad exacta de estos a la hora de solicitar el transporte.

Otra aplicación que se puede realizar con estos servidores son informes de calidad, ya que podemos ejecutar un simple bucle en el excel con los que guardar las temperaturas de un proceso y conocer así los valores máximos y mínimos, datos que en cierto tipo de productos delimitaran si están o no bajo norma y si cumplirán con ciertas características de diseño.

5.5 Programa de generación de servidores

Para la generación de este proyecto se han barajado numerosos programas con los que generar la base del servidor OPC, muchos de ellos son de licencia gratuita, y todos ellos cuentan con un periodo de prueba para testarlos. Algunos de estos programas son:

- KepSERVER
- Matricon OPC
- NI OPC Server

Todos tienen puntos a favor y en contra, pero finalmente se utilizo el NI OPC Server ya que es de obligado cumplimiento para declarar las variables en el programa. Estas variables serán las que necesitemos para actuar contra la planta. Este servidor es bastante potente, ya que cumple con los mismos estándares de comunicación, pero no hace falta declarar el canal de comunicación para enlazar el LabVIEW, sino solamente generar una librería con un ”NI Server” y mantener el programa abierto.

Capítulo 6

Virtualización de la planta

En el presente capitulo se virtualizara la planta vinculando está a los archivos auxiliares generados en el Capítulo 3. Se explicara la manera de introducir estos archivos, y como vincularlos para modelizar otras plantas a estos programas de LabVIEW. De manera análoga a lo explicado en el Capítulo 3, el método sera la generación de varios sub-programas con los que configurar la planta con la que testar el comportamiento del autómata, mejorando asi el rendimiento de computación ya que solo se ejecutara el programa principal y este llamara a los demás cuando sea necesario. En este caso contamos con las variables del servidor OPC, tanto de entrada como de salida con el fin comprobar el comportamiento del autómata.

6.1 Metodología

En esta sección se va a definir la metodología que se seguirá para la creación de los distintos ensayos en cuanto a la parte de LabVIEW se refiere. En esta sección necesitamos definir con claridad los sub-programas que utilizaremos para el ensayo, las variables o constantes que necesitamos para realizar estas tareas, así como las utilizadas con la virtualización de la planta.

En este apartado se hablara de la creación del servidor, y de la metodología para llevarlo acabo, aunque es de especial relevancia seguir la guía adjunta en el anexo, ya que en ella se definió paso a paso todas las configuraciones necesarias para crearlo. Esta guía se encuentra en el Capítulo 11. Para la creación del servidor es necesario tener el programa de Tia Portal ya que sin este no conoceremos las direcciones de memoria del autómata donde generara las variables.

6.2 Modelo de la planta

En el presente punto se va definir el proceso seguido para la implementación de la planta virtualizada en el programa LabVIEW, del cual hablamos en la subsección 1.7.1. Este software tiene la peculiaridad de ser programación gráfica, pero en este caso y ya que el programa lo permite utilizaremos la programación mixta, parte gráfica y parte escrita. Para ello es muy importante tener instalado el modulo "LabVIEW MathScript RT" que permitirá programar en lenguaje M las ecuaciones descritas en la subsección 2.3.2. La implementación de esas ecuaciones se han escrito de la siguiente manera:

Listing 6.1: Ecuaciones de virtualización de la planta

```
K1 = param(1);
K2 = param(2);
K3 = param(3);
K4 = param(4);
tau1 = param(5);
tau2 = param(6);
tau3 = param (7);
K11 = param(8);
K21 = param(9);
K22 = param(10);
K32 = param(11);
K43 = param(12);
Kem = param(17);
Kem2 = param(18);
taum = param(19);
taum2 = param(20);

T = param(13);

% param 14 = X10
% param 15 = X20
% param 16 = X30

X1_kp1 = X1_k + T*(-1/tau1*(K2+K1*U2_k)*X1_k + K21*
    X2_k + K11* U2_k*U3_k);
X2_kp1 = X2_k + T*(K22*X1_k - K22*X2_k - K32*X2_k*X3_k
     + K32*X3_k*U4_k);
X3_kp1 = X3_k + T*(-1/tau3*X3_k + K43*U1_k);

% Sensors

Tcm_kp1 = Tcm_k + T*(Kem2/taum2*X2_k - 1/taum2*Tcm_k);
Tpm_kp1 = Tpm_k + T*(Kem/taum*X1_k - 1/taum*Tpm_k);
```

Como se observa en el Listing 6.1, los parámetros físicos de las primeras lineas no tienen un valor fijo definido (K_1, K_2, K_3, K_4, $tau1$, $tau2$, $tau3$, K_{11}, K_{21}, K_{22}, K_{32}, K_{43}, K_{em}, K_{em2}, $taum$ y $taum2$). Esto se ha realizado de esta manera para exportar directamente desde un CSV o XLSX los parámetros. Del mismo modo se ha exportado la constante de tiempo T correspondiente al periodo de muestreo. Es importante que este periodo de muestreo sea representativo para que todos los datos quedan representados. Esto permite que el mismo código sirva para cualquier sistema similar, el cual se pueda definir siguiendo el proceso descrito en sección 2.2. En nuestro caso estos parámetros serán obtenidos de un archivo CSV que generara Matlab, y que configuraremos para obtenerlo cuando diseñamos el regulador. Sera necesario que las columnas a exportar sean en el mismo orden que las actuales, para no modificar así el sistema.

Otra parte a tener en cuenta es la de sobrescribir los valores de tiempos futuros $(k+1)$ a los nuevos valores guardadoes en memoria tras cada iteración (k), para el siguiente ciclo del código. Esto se ha realizado mediante programación gráfica. Para esta resignación se ha cerrado el bucle con un "feedback node" que sobrescribe los valores, ademas usando este método podemos asignar el valor inicial de la primera iteración, que también obtendremos del CSV.

La planta virtualizada por completo queda representada en la figura 6.1, el cual sera un "subVI" que ejecutaremos en el ciclo del LabVIEW que conecta con el servidor OPC.

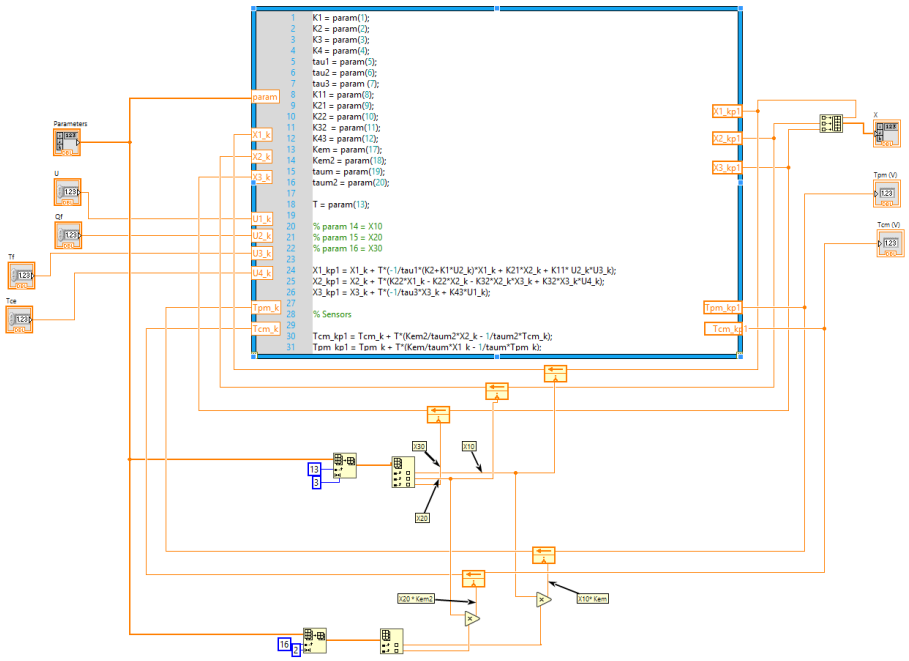

Figura 6.1: Programación para virtualizar la planta en LabVIEW

Una vez tenemos este sub-programa deberemos incluirlo en la librería del trabajo de LabVIEW para finalizar.

6.3 Configuracioón del servidor OPC

En esta sección se va a tratar el tema de la generación del servidor OPC. Se trataran las configuraciones basicas que hay que tener en cuenta para poder configurar adecuandamente el proyecto, aunque en la subsección 11.2.2, encontrarás definido punto a punto los pasos seguidos para la generación de un servidor completo.

Para esta parte del proyecto debemos tener instalado y activado el programa que se definió en la subsección 1.7.1, y como primera acción generar el bus de comunicación, proceso definido en el Párrafo 11.2.2. Es importante a la hora de seleccionar de la lista desplegable el "Device drivers" seleccionar el protocolo de Siemens TCP/IP Ethertent, ya que si se selecciona el protocolo master o slave el autómata funcionara correctamente, pero si se desease conectar otro automata los mensajes de uno a otro no podrían enviarse, ya que estarán configurados como esclavos.

Otro punto relevante es conectarse adecuadamente a la tarjeta de comunicación, ya que en una de las ocasiones en las que se realizo una prueba en el laboratorio de automática, el ordenador contaba con dos tarjetas de red, y debido a una mala eleccion de la tarjeta la comunicacion fue imposible de realizar. Por otro lado, esta tarjeta a de tener una direccón IPv4 dentro del

ancho de banda del automata. Esto es muy importante ya que podremos seleccionar y configurar adecuadamente la comunicación por medio del servidor OPC, pero esta no podra comunicar por estar fuera de banda. Para ello es recomendable realizar un pequeño ensayo desde la ventana de comandos de Windows, con la orden "ipconfig", pudiendo realizar PING a todos los elementos conectados a la red, averiguando así las direcciones y consultando si estan en el mismo ancho de banda.

Por ultimo, uno de los fallos principales de la configuracion del servidor OPC, y que mas errores puede causar es la configuración del tipo de variable. En los estandares de automatización encontramos con un gran numero de variables, definidas por unos nombres comunes que en los estándares de LabVIEW no concuerdan o no se llaman de ese modo. Los estandares de LabView se encuentran bajo la norma de programación informática, de este modo debemos conocer que tipo de dato queremos enviar, y la manera de enviarlo ya sea desde el autómata al programa o viceversa. Es muy importante conocer el numero de bits a enviar, el signo y la coma flotante. De este modo se han tenido que configurar las variables en el automata como "REAL" (16 bit con 4 numeros decimales de resolución), en el servifor OPC como "FLOAT" ya que tiene la misma configuración, y en el LabVIEW como "Single" que tiene la misma resolución.

6.4 Generación de la librería

Una vez tenemos configurado el programa del servidor OPC, debemos generar una librería para luego contar con las variables que declaremos del servidor. El proceso por el cual generamos esta librería, esta definido en la subsección 11.2.3, teniendo que prestar especial atención a incluir fuera de la librería los programas. Es de especial relevancia que los programas y subprogramas queden fuera de esta ya que si no la manera de usarlos es distinta, generando errores e impidiendo el funcionamiento. Si observamos la figura 6.2, veremos como las variables generadas y el servidor OPC estan dentro de la librería "prueba con adolfo.lvlib", mientras que los programas están fuera de la librería, compartiendo la misma estructura.

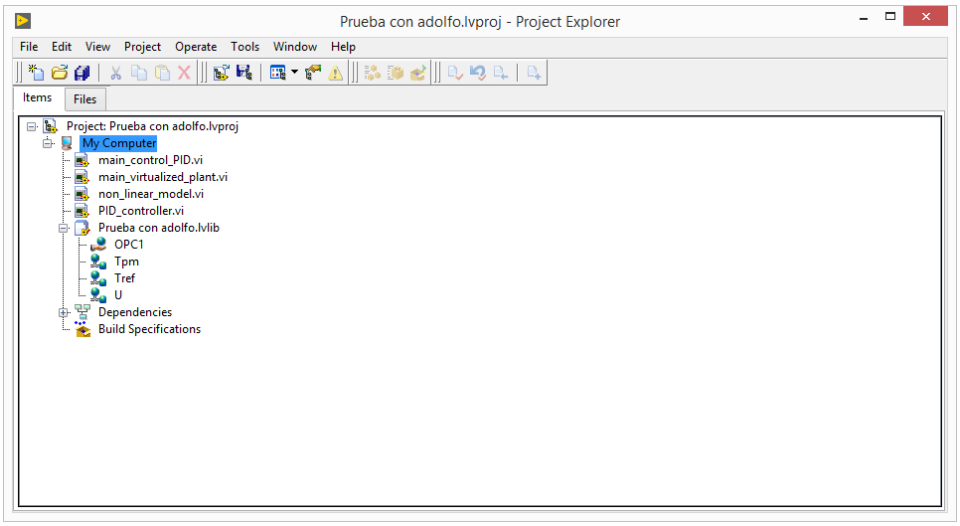

Figura 6.2: Estructura del programa de LabVIEW para su correcto funcionamiento

Si analizamos nuevamente la figura 6.2, observamos que tenemos definido en la estructura el sub-programa "non_lineal_model" que definimos en la sección 6.2. De esta estructura colgaran todos los programas y sub-programas que necesitemos tener para la virtualización de la planta. En este caso contamos con 4 programas ya que se trata de la estructura con la que realizamos numerosas pruebas, pero para virtualizar la planta solo necesitaremos tener definidos dos:

- non_lineal_model: Modelo de la planta virtualizada
- main_virtualized_plant: Programa principal de comunicación con el autómata

6.5 Programa de comunicación

En este programa de LabVIEW mostraremos como se ha generado el bucle que permite la comunicación con el autómata, la lectura de los archivos auxiliares, y la generación de unos archivos tras en ensayo con los que poder comparar los resultados teóricos y prácticos. Para este programa se a utilizado íntegramente la programación grafica que el programa tiene, es decir, no se analizaran las funciones utilizadas como en el caso del Capítulo 3, sino que se mostraran partes del programa, comentando qué se ha programado con esas imágenes.

Si analizamos la figura 6.3, observamos dos funciones conectadas en serie que son idénticas, una encima de la otra. La primeara de las dos series es la función "Build Path", y la segunda de ellas es la "Read From Spreadsheet File.VI". La unión de ambas permite leer los archivos en formato CSV que hemos generado, tanto para comparar la simulación (Tref_i_Pertotbacions.csv) como para introducir los parámetros del modelo (parametres_model.csv). Del segundo archivo se esterara el periodo de muestreo, el cual se puede observar que se introduce en el cuadro del bucle como dt, extrayendo el parámetro 12 con la función "Index Array". Se puede observar que se multiplica por un valor de 1000 unidades, esto es debido a que el bucle del ordenador implementado con el programa LabVIEW tiene los timer en mili-segundos, mientras que el periodo de muestreo del diseño del autómata (Capítulo 3) esta espesado en segundos. En la parte de abajo, y con la misma función que antes, se a extraido el valor de la adaptación de la referencia.

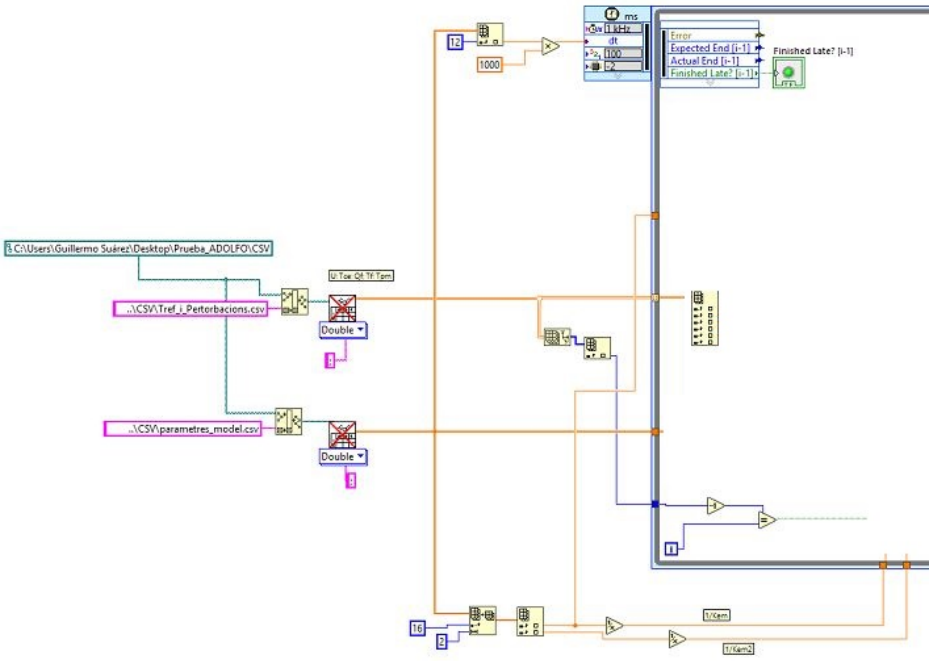

Figura 6.3: Parte de la programación encargada de la lectura de los archivos CSV

Si pasamos ahora a analizar la figura 6.4, podemos observar como están las variables que necesitamos declarar en el servidor, es decir, la salida del autómata o entrada a la planta, la

salida de la lectura del sensor, o entrada para el regulador, y la referencia que esta vinculada a un archivo CSV con el que realizamos las simulaciones bajo las mismas condiciones. En este caso podemos observar que se trata de un PID estándar, puesto que la lectura del sensor rápido no se a declarado en el servidor.

Por otro lado vemos que la matriz que se ha introducido procedente del archivo CSV con el nombre de Tref_i_pertorbacions se ha descompuesto en los distintos vectores de los que hablamos en la sección 3.4, y en el mismo orden. Para esta descomposición se ha utilizado la función gráfica "Index Array" de igual modo que en la parte de la lectura, pero esta vez se a estirado mostrando todos los vectores que la conforman.

En el centro, y con la simbolista característica de este programa, observamos como encontramos en el centro un archivo sub-VI. Este archivo es nuestra planta virtualizada, la cual generamos en el sección 6.2. A este subVI necesitamos introducirle los parámetros que vienen directamente del archivo CSV parametres_models. Luego se introducirán el resto de variables como se aprecia en la figura 6.5

Por ultimo y a modo de análisis se han generado una gráfica en la que se compararan los datos, en la primera de ellas se comparan la lectura del sensor de la caldera, tanto teórico como practico, con la referencia para ver si el funcionamiento es el correcto. Por otro lado se comparan las señales de control, tanto la teórica como la real. En este puto del proyecto podremos realizar la primera prueba para conocer el estado del proyecto, ya que si se ejecuta existirá comunicación con el servidor. Esto es una gran noticia, ya que el servidor estará configurado adecuadamente y los valores de la gráfica han de moverse. Puede ocurrir que el servidor comunique, pero que el autómata no varié los valores, eso querrá decir que el fallo esta en la programación del autómata, pero no en la programación gráfica.

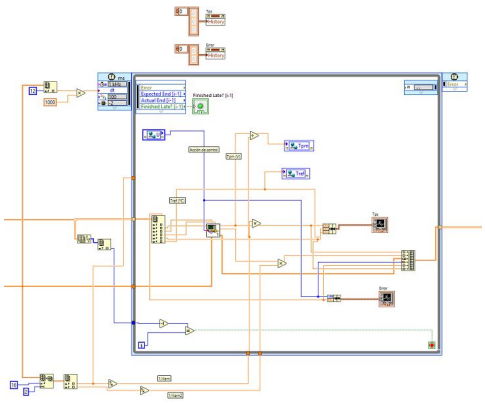

Figura 6.4: Parte de la programación encargada de la comunicación OPC

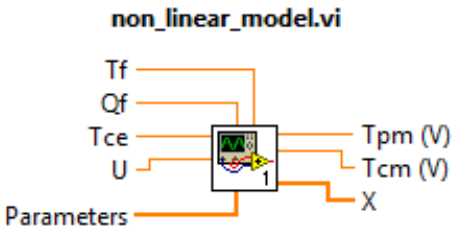

Figura 6.5: Ventana de ayuda de nuestro subVI

Si pasamos a analizar ahora la ultima parte del proyecto, el cual se muestra en la figura 6.6, configuraremos un archivo CSV para poder extraer la información de los ensayos. Esto nos permitirá comparar los distintos ensayos tranquilamente y analizarlos estadisticamente, ya que se podran introducir en el Matlab, realizando programas estadisticos que comparen los resultados objetivamente.

Con el fin de no sobreescribir uno sobre otro cuando se ejecuten, se ha decidido poner en el nombre la fecha completa, la hora y el minuto. D este modo mantendremos todos los ensayos realizados hasta la fecha. Esto se ha realizado mediante la función "Get Date/Time In Seconds" y la función "Write To Spreadsheet File".

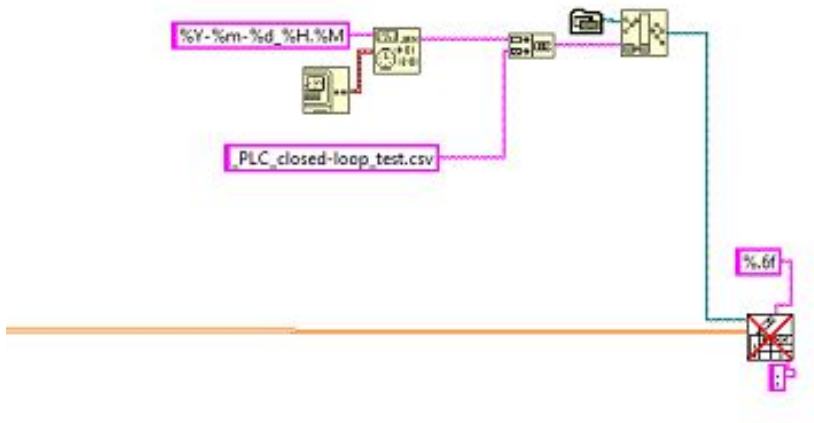

Figura 6.6: Parte de la programación encargada de la comunicación OPC

6.6 Variables a definir

Para este proyecto se barajan una serie de controladores, para los que habrá que tener en cuenta la declaración de unas u otras variables. Es por ello que en esta sección se hablara de las distintas variables que se necesita declarar en el servidor para ejecutar un análisis u otro. Estas variables van íntegramente ligadas a la programación del autómata, por lo que sera imposible declararlas sin haber cambiado o modificado el programa del autómata. En función del regulador a implementar deberemos declarar:

- PID estándar en bloque:
  - Tref: Referencia del ensayo
  - Tpm: Lectura del sensor
  - U: Señal de control
- PID estandar discreto:
  - R_k: Referencia del ensayo
  - Y_k: Lectura del sensor
  - U_k: Señal de control
- PID cascada en bloque:
  - Tref: Referencia del ensayo
  - Tpm: Lectura del sensor lento

- Tmc: Lectura del sensor rápida
- U: Señal de control

- PID cascada discreto:
  - R_k: Referencia del ensayo
  - Y_k: Lectura del sensor lento
  - Y_2k: Lectura del sensor rápida
  - U_k: Señal de control

Capítulo 7

Implementación de los programas en Tia Portal

En el presente capitulo se hablara de la programación del autómata desde el programa Tia Portal, del cual hablamos en la subsección 1.7.1. Nos centraremos en la programación en si ya que las configuraciones necesarias para el servidor han sido declaradas en Párrafo 11.2.2.

Mostraremos como se diseña un regulador por bloques, y uno discreto, aunque en el Capítulo 8, compararemos mas reguladores. Se ha decidido mostrar la programación de un PID por bloques estándar, y un PID por bloques discreto, aunque para el ensayo se ha programado también las variantes por bloque y discreto del regulador en cascada. Los reguladores a implementar son dos, el regulador discreto del cual hablamos en la subsección 4.1.1. y el regulador en cascada del cual hablamos en la subsección 4.2.2.

7.1 Estructura del programa

En esta sección hablaremos de la estructura que el programa del autómata debe tener en memoria, para desempeñar las tareas de control y de automatización las cuales se describieron en la sección 1.5. La parte del control ha de implementarse en un " "cyclic interrupt" o traducido al castellano, interrupción cíclica. La parte de automatización ha de estar programada en el "Main" o cuerpo principal del programa. Este cuerpo principal sera idéntico en todas las pruebas que realicemos, por lo que se mostrara una única vez y se copiara en el resto de programas. Para la realización de los ensayos no es necesario tener programado la parte de automatización, sino solo la interrupción cíclica para que se ejecute el algoritmo de control. En la figura 7.1, podemos observar como contamos con los dos bloques de programa.

Por otro lado, y siguiendo con lo mostrado en la figura 7.1, observamos que tenemos los llamados "Objetos tecnológicos", esto o encontraremos cuando implementemos el regulador PID mediante la opción de bloques. Si por el contrario se realiza de forma discreta esta opción no deberá aparecer. Lo siguiente que observamos son las variables del PLC, este desplegable mostrara todas las variables utilizadas para el sistema, ya sean de tipo interno o sean las declaradas en el servidor OPC. Las demás paletas de opciones no son de especial importancia para la realización de la programación, o para la realización de los ensayos.

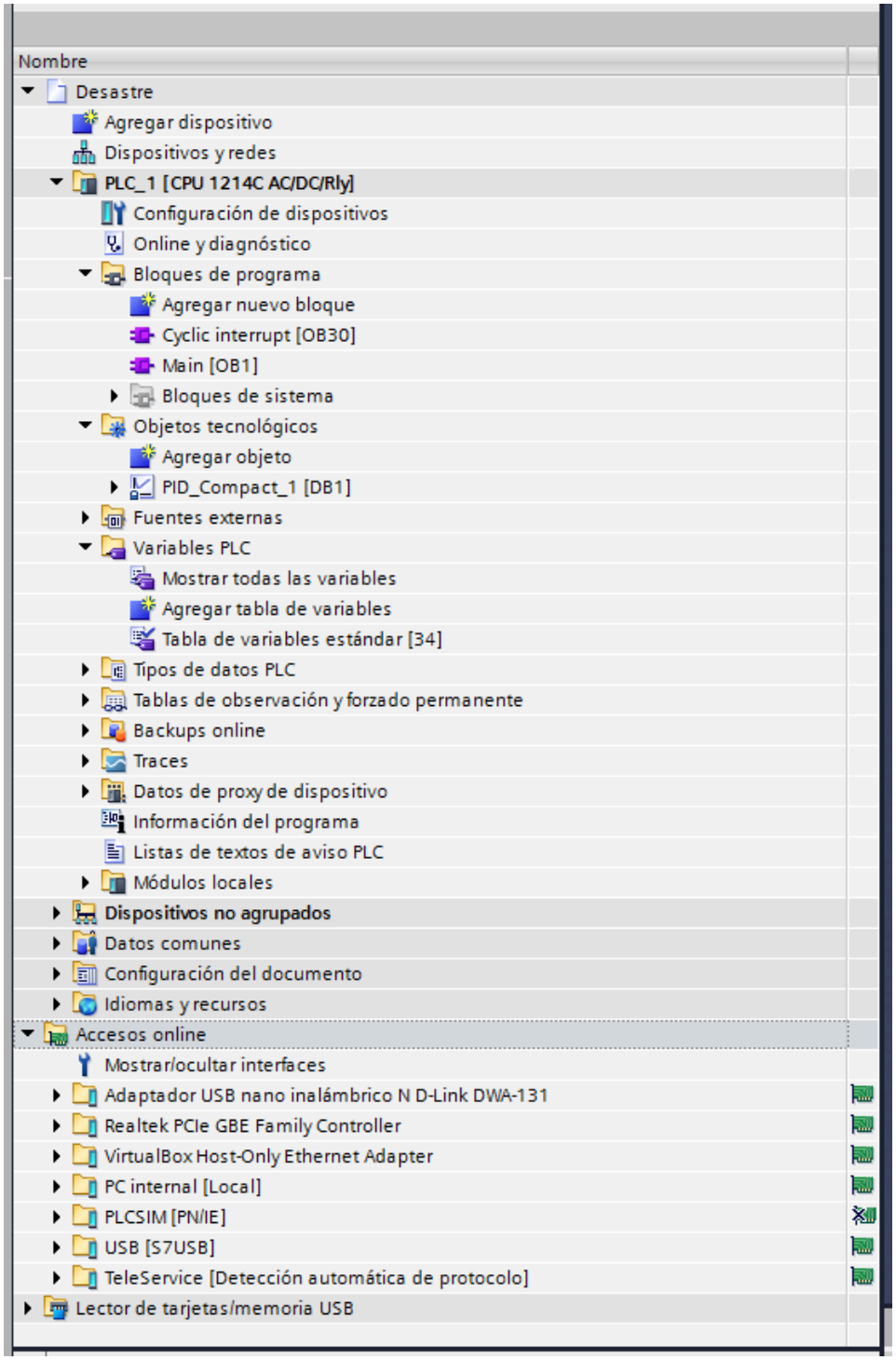

Figura 7.1: Estructura del programa de Tia Portal

Por ultimo, podemos encontrar las tarjetas de comunicación con las que cuenta el ordenador, y desde las cuales podremos configurar la comunicación con el autómata.

7.2 Lenguajes de programación

Este autómata programable cuenta con tres lenguajes básicos de programación, los cuales están recogidos en la normativa descrita en la subsección 1.4.1. Estos lenguajes son:

- KOP: Lenguaje típico de puertas lógicas, muy utilizado en autómatas sencillos de Siemens como el Logo.
- FUP: Lenguaje por defecto, se trata del lenguaje "escalera" o de contactos. Su programación es gráfica y de uso muy sencillo
- SCL: Lenguaje estructurado. Se trata del lenguaje mas parecido a los estándares de programación como M, C++, o Pascal

Para este proyecto se han utilizado solo los dos últimos lenguajes de programación, uno cuando se utilice el sistema de bloques, y otro cuando se realice el estudio de los reguladores discretos.

7.3 Programación de automatización

Como se ha citado en la sección 7.1, el proceso de automatización se realizara en el "Main" del programa. La automatización que se va a llevar a cabo sera la descrita en la sección 1.5.

Contaremos con tres modos de funcionamiento, manual, semiautomático, y automático. Para la elección de cualquiera de los métodos contamos con un selector de tres posiciones. Para ello debemos generar un condicional que en función de la entrada que se excite del autómata, sepamos que estamos en un modo de funcionamiento u otro. Con el fin de ahorrar una de las entradas digitales, el modo automático sera cuando ninguno de los otros dos modos este conectado, de este modo al autómata le entraran unicamente dos señales. Esto lo podemos observar en la figura 7.2, donde observamos la lógica para llevarlo acabo.

Para el control de la válvula A contamos con dos pulsadores, un pulsador A, el cual sera la marcha de la electroválvula que llenara el tanque. Y un pulsador B, que accionara la parada de la electroválvula. Al tratarse de electroválvulas, los encorvamientos se han realizado con biestables de tipo R y S programables. De este modo si se excita la función S esta permanecerá activa hasta que se accione el biestable R. Esto lo podemos observar en la figura 7.3 en el que mostramos el control total de la válvula.

Del mismo modo que en el segmento 2 se ha de controlar la electroválvula C, sustituyendo ahora los pulsadores A y B por el C y D respectivamente. Esto lo encontramos en la figura 7.4.

Para el control de la válvula B, tenemos que contar no solo con el método de funcionamiento sino con el propio regulador PID. Para aclarar el funcionamiento de uno u otro método, se mostrara en la figura 7.5 el regulador PID, pero como ya se cito en esta misma sección, este regulador ha de ir en una interrupción cíclica.

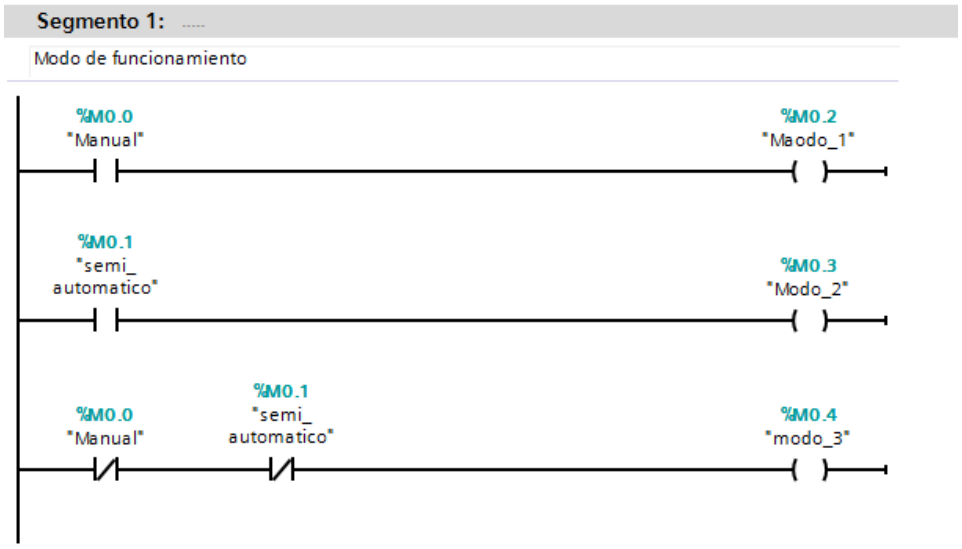

Figura 7.2: Segmento 1, modo de funcionamiento

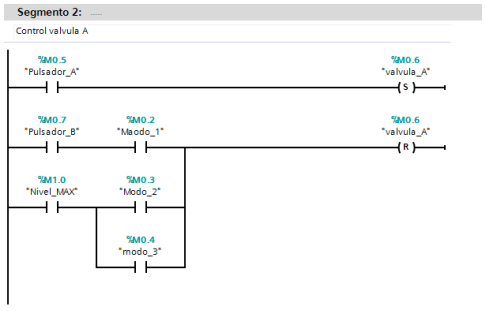

Figura 7.3: Segmento 2, Control de la válvula A

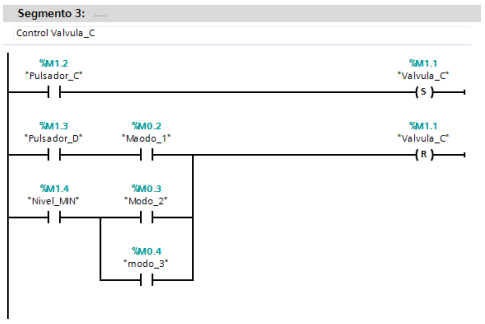

Figura 7.4: Segmento 3, Control de la válvula C

Para solucionar la entrada en funcionamiento del regulador, o la vinculación del potenciómetro a la salida analógica se ha creado una especie de condicional con las funciones ”ENABLE” de ambos. En este segmento no se a normalizado ni escalado la referencia ya que lo único que se pretende es mostrar la entrada de funcionamiento del regulador si se activa el método automático.

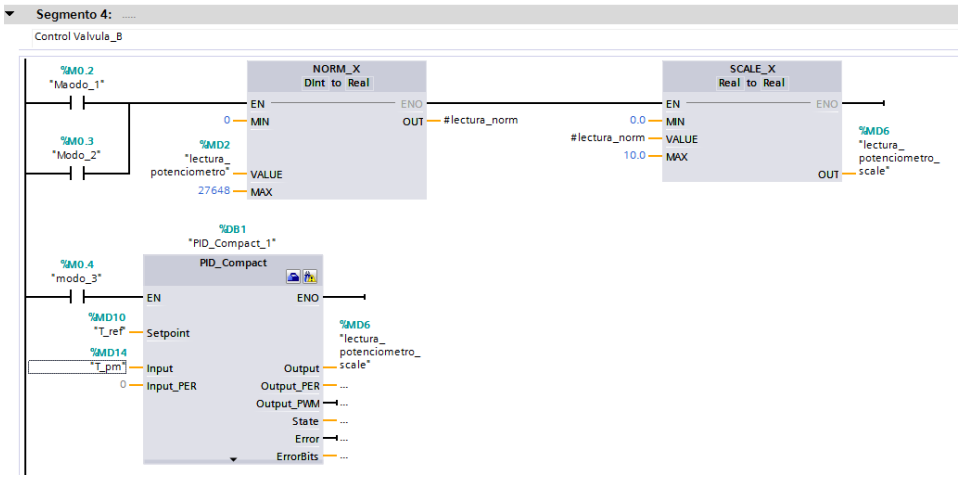

Figura 7.5: Segmento 4, Control de la válvula B

Tras este proceso de automatización unicamente faltarían los elementos de seguridad, como son las luces y el paro de emergencia. Para automatizar este paro, hay que recordar la normativa de seguridad de los paneles de operador, la cual quedo definida en la tabla 1.1. tanto en el REBT, como en las normas técnicas de prevención queda recogido que el paro de emergencia ha de ser un contacto normalmente cerrado, y con enclavamiento mecánico. Es por ese motivo por el cual el paro puede realizarse de forma programada, tal y como se observa en figura 7.6, manteniendo la función predominante ”RESET” activa, impidiendo que entre en funcionamiento cualquier orden del código anterior. De igual modo se podrá encender la luminaria de paro sin que el autómata conmute ningún contacto, impidiendo así un fallo interno.

Para activar el testigo de tensión, unicamente se puentea una salida del autómata. Las luces de nivel máximo y nivel mínimo de los sensores, no deben salir del automatismo, se puenteara la entrada del autómata con la de la luz correspondiente, de este modo si parasemos el funcionamiento del autómata, conoceríamos el estado del tanque evitando así la manipulación del mismo si se encontrase lleno.

7.4 Programación del regulador

7.4.1 Mediante bloques

En esta sección vamos a pasar a mostrar la programación y configuración de un regulador de tipo PID mediante los bloques internos del autómata. En esta sección no vincularemos a la parte de automatización mostrada en la sección 7.3, aunque en dicha sección definimos como hacerlo. Por el contrario, en esta sección nos centramos unicamente en la parte de regulación, y generaremos unicamente el control PID.

Para comenzar con la regulación debemos generar la tabla de variables, y definir en esta las variables que necesitamos declarar en el servidor, y el tipo de la misma. De este modo el

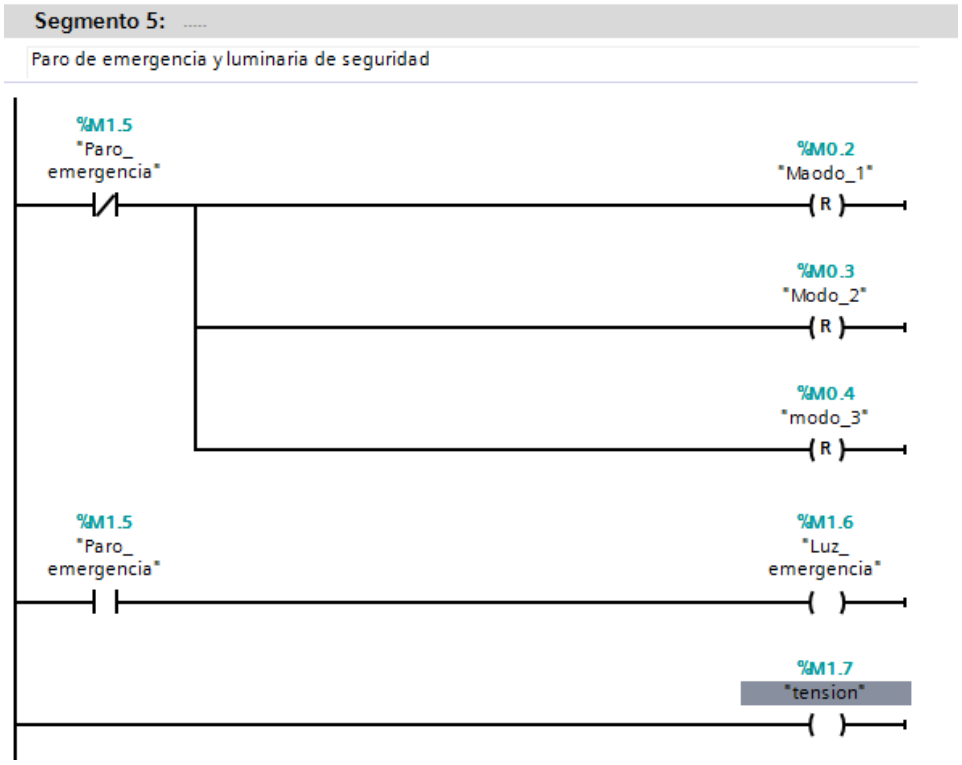

Figura 7.6: Segmento 5, Control de la seguridad

autómata asignara una dirección de memoria, con la que realizaremos la declaración de variables en el servidor OPC. Esta metodología se encuentra definida en la subsección 11.2.2. Para el caso que nos atañe, unicamente definiremos tres variables, las cuales fueron citadas con anterioridad en la sección 6.6.

El segundo paso para la generación del test, es crear el espacio de trabajo donde se montara todo, es decir, la interrupción cíclica. Esta interrupción se generara desde los bloques de programa, (imagen mostrada en la figura 7.1), donde deberemos presta especial atención al ciclo de la interrupción Cuando se selecciona el bloque ”cyclic interrupt” en la opción de generar un nuevo bloque debemos configurar el tiempo de la interrupción igual al periodo de muestreo con el que se diseño el autómata, en este caso $1s$. Esto lo podemos observar en la figura 7.7.

Una vez contamos con la comunicación de las variables del servidor debemos conocer entre qué rango de valores se enviaran estas variables. Esto dependerá de la configuración de la salida del LabVIEW, y si se ha realizado una transformación que simule el paso por el conversor interno A/D, o se envíe la señal en ^oC o V. Si se selecciona la simulación del conversor, hay que conocer que este conversor transforma la entrada en datos que van de 0 a 27648. Este dato es importante ya que todos los datos que comuniquen con el autómata han de normalizarse y escalarse para que entren al regulador conforme se ha definido. Por ello el primer paso es ese, la normalización y escalado de los datos. Esto lo podemos observar en la figura 7.8.

Una vez contamos con el escalado de la señal debemos configurar el regulador PID. Para ello deberemos hacer click sobre ”objetos tecnológicos” y volver a pulsar sobre ”añadir objeto”. En este momento se abrirá una nueva ventana como la mostrada en figura 7.9, donde debemos seleccionar el tipo de regulador que queremos implementar. En este caso generaremos un ”PID_Compact”. Tras seleccionar esa opción se abrirá una ventana de configuración como la mostrada en figura 7.10, en la que seleccionaremos las funciones del regulador.

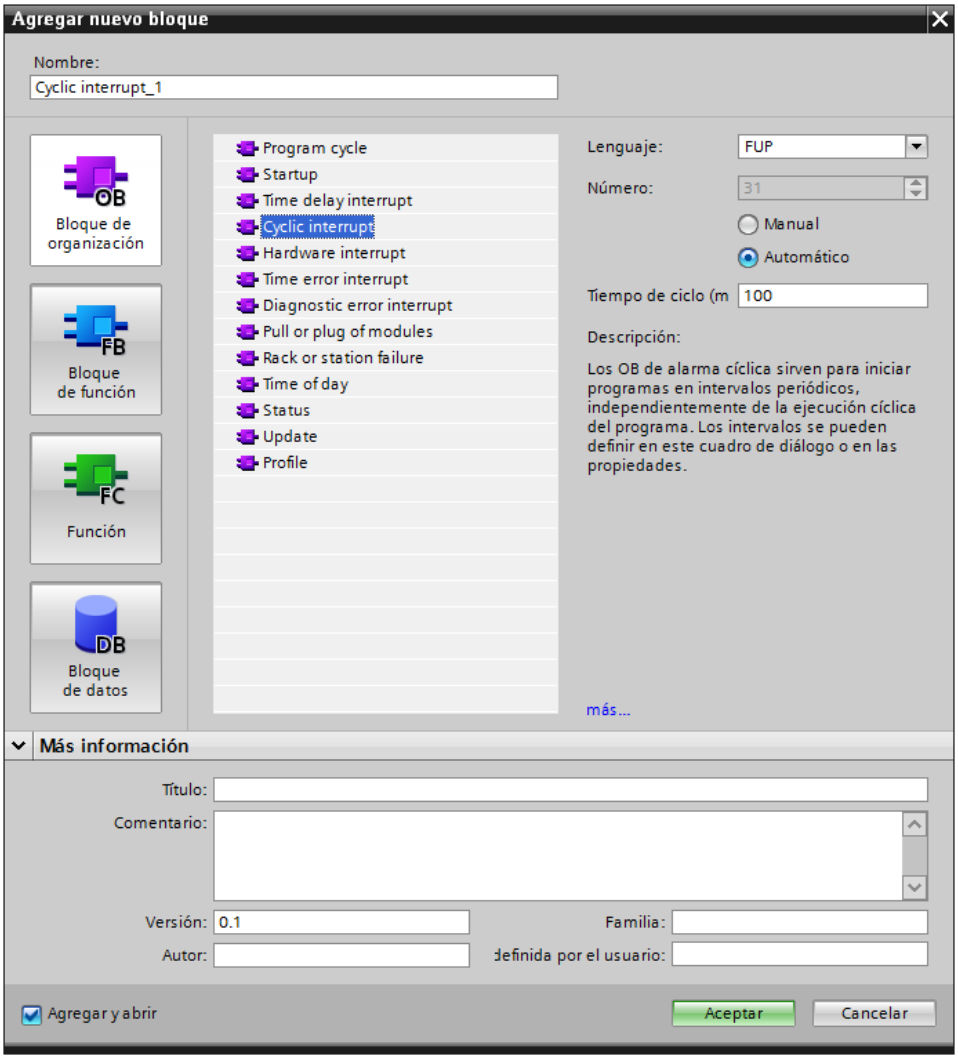

Figura 7.7: Generación de la interrupción

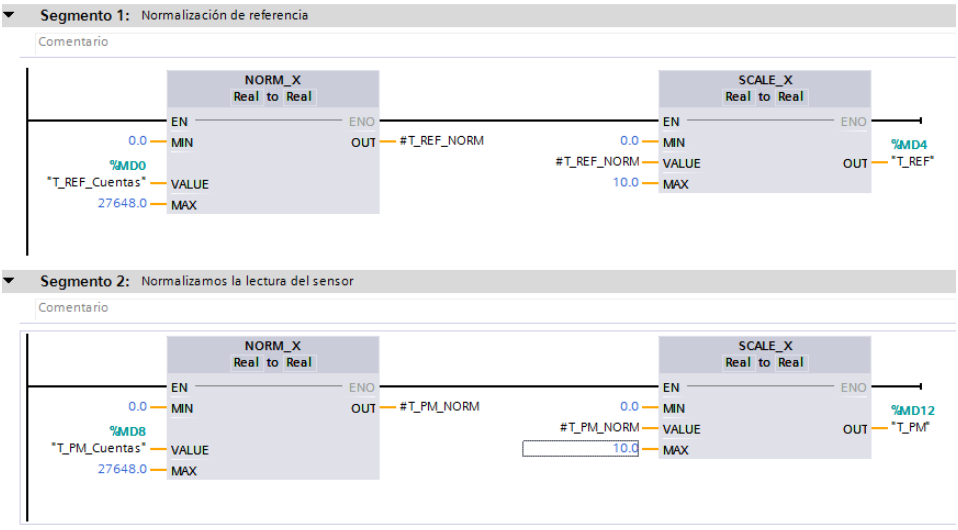

Figura 7.8: Regulador PID en bloque, normalizado de la señal

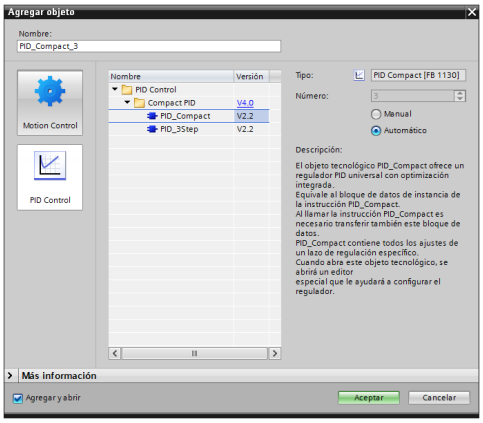

Figura 7.9: Regulador PID en bloque, normalizado de la señal

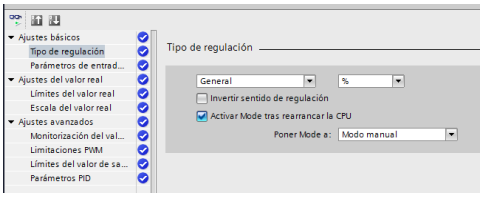

Figura 7.10: Ventana de configuración del regulador PID

En la primera pestaña nos pregunta sobre que tipo de regulador sera, en este caso habrá que indicar que se trata de un regulador de temperatura, y que su unidad básica es el ^oC. Una de las opciones mas importantes que debemos seleccionar es la de "Arrancar mode tras rearrancar la CPU" y seleccionar la activación en "Modo automático". Esta configuración la podemos observar en la figura 7.11.

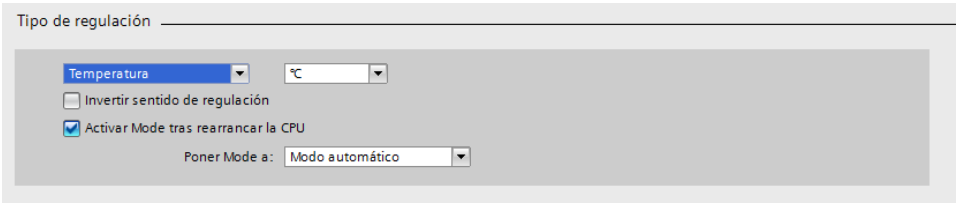

Figura 7.11: Selección del tipo de variable a controlar

Si continuamos navegando por los menús que este regulador en bloque presenta encontramos la opción de "Parámetros de entrada". En esta opción deberemos seleccionar cual sera la entrada y salida del bloque. Al tratarse de una señal OPC se selecciono la entrada "Input", en lugar de la "InputPER" que es de tipo analógico, pero si el autómata pasase el estudio de suficiencia y finalmente se montase en la planta, la señal adecuada para la entrada del bloque seria la de "InputPER", ahorrando así el paso de normalización y escalado anterior. En cuanto a la salida se refiere, tenemos tres opciones, "Output", "OutputPer" y "Output PWM". Del mismo modo se ha descartado la salida de señal analógica, pero si se montase en la planta esta seria la adecuada. La salida PWM se descarga porque la válvula que instalaremos se regula en tensión y no por pulso de ancho de banda. Esta opción la podemos observar en la figura 7.12

La siguiente opción con la que nos encontramos es la de "Ajuste del valor real" tal y como podemos observar en la figura 7.13. En esta opción configuraremos los valores máximos de la planta. Según lo definido en el sensor, este valor ha de oscilar entre 0 y 100^oC. La segunda parte de la ventana correspondería a la entrada "InputPer", la cual no permite la modificación si se encuentra apagada.

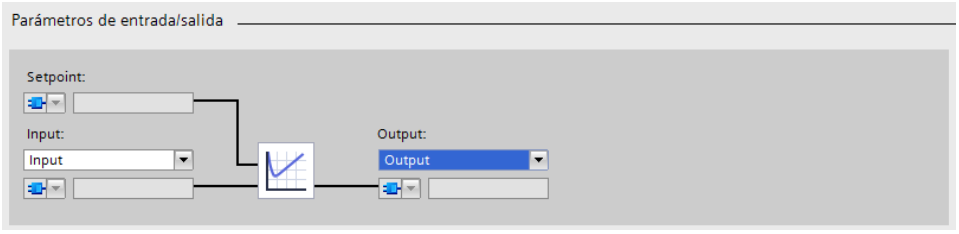

Figura 7.12: Selección del tipo de entrada y salida

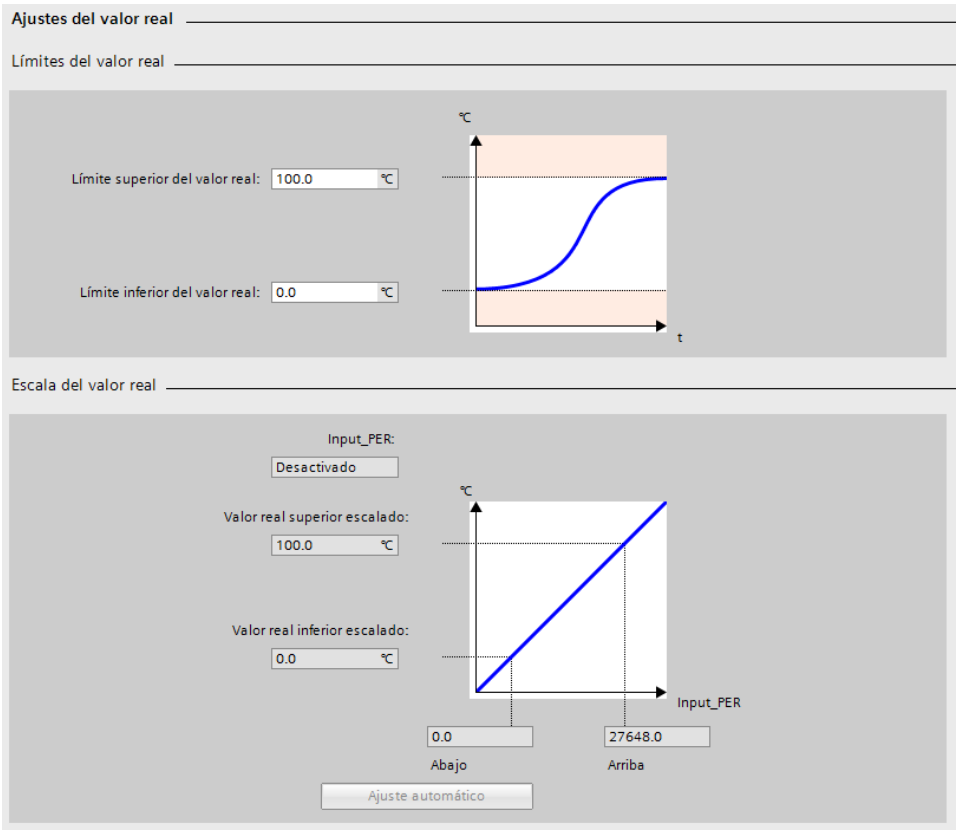

Figura 7.13: Selección del ajuste del valor real

Por ultimo, y tal y como se muestra en figura 7.14, encontramos los ajustes avanzados del regulador. En esta opción configuraremos el limite real del valor de saturación de salida, por ello debemos escribir el valor de 10 % para la saturación a nivel alto, y el 0 % para la saturación a nivel bajo. En el ultimo nivel de las opciones de ajuste avanzado encontramos los parámetros del PID, los cuales deberán de ser idénticos a los obtenidos en el Capítulo 3. Tras esto tendremos nuestro bloque PID listo para ser ejecutado, unicamente quedara la asignación de las variables generadas en las posiciones indicadas (”Input” y ”Output”), para que funcione adecuadamente. Este bloque se ha de añadir en la interrupción cíclica y siempre un segmento por detrás del escalado y normalizado de la señal.

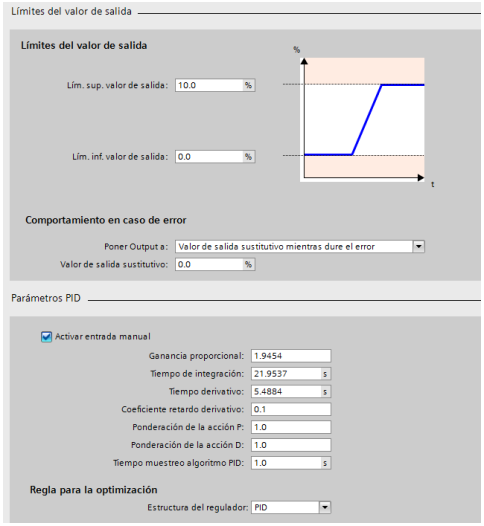

Figura 7.14: Ajustes avanzados del regulador

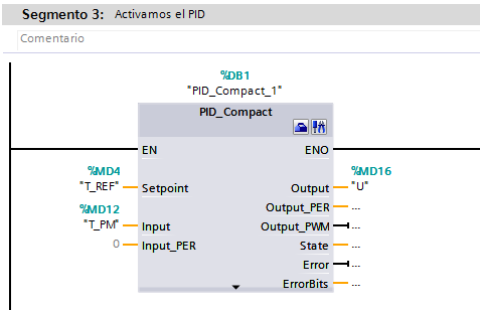

Figura 7.15: PID bloque con entradas y salidas configuradas

7.4.2 Discretizado

Para la implementación de un regulador discretizado, deberemos realizar una serie de configuraciones previas a las citadas en la subsección 7.4.1. En este caso al crear la interrupción cíclica, debemos seleccionar el lenguaje de programación estructurado (SCL). En cuanto a la creación de las variables debemos generar las que citamos en la sección 6.6.

Listing 7.1: Regulador PID discreto parte 1

```
// PID para el PLC

// ------------------------------------------
// Variables a definir en el PLC

"Kp" := 1.2675;
"Ki" := 0.136;
"Kd" := 2.96;
"N" := 2.1443;
"a" := 0.2;
"b" := 1;
"c" := 0;
"awm" := 1; //1 sin WD; 2 = Back Calculation; 3 =
    CLAMPING

"T" := 1;
```

Una vez contemos con las configuraciones descritas pasaremos a programar de forma estructurada el regulador discreto. Tal y como se muestra en el Listing 7.1, comenzaremos con la declaración de de las constantes del regulador PID, en este caso las constantes proporcional (Kp), derivativa (Kd) e integral (Ki), así como la ponderación de la referencia (b) y la derivada (c), y filtro de la derivada (a). Es de especial relevancia haber configurado la interrupción cíclica igual al tiempo de muestreo para el cual se diseño el regulador. En este caso observamos que el periodo de muestreo (T) corresponde a la configuración típica del tiempo de la interrupción

Listing 7.2: Regulador PID discreto parte 2

```
// ------------------------------------------
// Variables de l'algoritme

"K" := "Kp";
"Ti" := "K" / "Ki";
"Td" := "Kd" / "K";
"N" := 1/"a";
```

Posteriormente, tal y como se observa en el Listing 7.2, extraeremos unas variables que harán falta para el calculo de la acción de control, como son el tiempo derivativo (Td), el tiempo integral (Ti) y la variable N.

Listing 7.3: Regulador PID discreto parte 3

```
// ____________________________________________
// Cálculo de l'acción de control

"up_k" :=  "Kp" * ("b"*"r_k" - "y_k");

"ud_k" := ("Td"/("N""T"+"Td"))"ud_k_1" + ("N""Kd"/("N
    ""T"+"Td")("c"("r_k" - "r_k_1") - ("y_k" - "y_k_1")
    ));

"v_k" := "up_k" + "ud_k" + "ui_k";


IF ("v_k" > 10) THEN
        "up_k" := 10;
ELSE_IF ("v_k" < 0) THEN
        "up_k" := 0;
ELSE
        "up_k" := "v_k";
END_IF;
```

Posteriormente y tal y como se observa en Listing 7.3, calcularemos la acción de control proporcional y derivativa. Lo siguiente que realizaremos sera la suma de las acciones de control proporcional, derivativa, las cuales acabamos de calcular, e integral la cual calcularemos mas adelante. Esto puede llegar a crear error, ya que en la primera iteración la variable de la acción de control integral carecerá de valor. Para ello debemos de generar un código que se ejecute una única vez en el que asignemos un valor a la acción integral de 0, y a todas las variables de tiempo futuro. Esto se realiza mediante un ”Startup” tal y como podemos ver en la figura 7.7.

Listing 7.4: Regulador PID discreto parte 4

```
// Actualización de la acción integral con algoritmo
   anti-windup

"e_k" := ("r_k" - "y_k");

CASE "awm" OF
        1:  // sin anti-windupp
              "ui_k" := "ui_k" + "Ki" * "T" * "e_k";
        2:  // Back Calculation
              "ui_k" := "ui_k" + "Ki" * "T" * "e_k"
                 + "Kb" * "T" ("u_k"-"v_k");
        3:  // Improved Clamping
              IF ("u_k" == "v_k") OR ("e_k" * "u_k"
                 < 0)
                    "ui_k" := "ui_k" +  "Ki" * "T"
                       * "e_k";
              ELSE
                    "ui_k" := "ui_k"
              END_IF;
END_CASE;
```

Si pasamos ahora a analizar el Listing 7.4, podemos observar como esta vez si daremos un valor a la acción integral. Este valor se ha mejorado mediante los algoritmos de anti-windup, los cuales definimos en la subsección 3.2.3. Esta acción de control integral sera la que utilicemos en la siguiente iteración de la acción de control.

Listing 7.5: Regulador PID discreto parte 5

```
// ------------------------------------------
// Actualización de variables

"r_k_1" := "r_k";
"y_k_1" := "y_k";
"ud_k_1" := "ud_k";

// ------------------------------------------
// Fi
```

Por ultimo, y tal y como se muestra en el **??**, actualizaremos las variables de tiempo futuro con la nueva lectura de los sensores. Esto permitirá que el valor de las variables se sobrescriba para la siguiente interrupción cíclica.

Capítulo 8

Comparación de reguladores teoricos con reales

En el siguiente capitulo se realizara la comparación de los reguladores teóricos, simulados bajo unas características extremas, con los reguladores implementados en el autómata físicos actuando contra la planta virtualizada en LabVIEW. Este sistema de "hardware in the loop" podrá decirnos si el regulador cumplirá la función para la cual se ha diseñado, sin necesidad de realizar las prubeas sobre la plata real.

8.1 Introducción

En el Capítulo 3, comenzamos el estudio diseñando los reguladores que mejor se ajustasen a la planta de estudio. En los capitulos posteriores se realizo una red virtual con un servidor OPC para "engañar" al autómata y que este pensase que estaba actuando sobre la planta real. Ahora pasaremos a comparar el comportamiento de los reguladores para conocer si estos tienen la suficiente potencia de calculo como para implantar el nuevo sistema en la red empresarial aportando datos que del comportamiento real del automata y de la planta pudiendo abalar o desmentir nuestro estudio.

8.2 Comparación de reguladores estándar

En la presenten sección compararemos los dos métodos definidos en el Capítulo 7 para la implementación de los reguladores PID estándar. Se comparara desde un punto de vista estadístico, calculando el error entre ambos. Se mostraran tres gráficas en las que se observara la temperatura del sistema, y como el regulador a ido variando esta temperatura para alcanzarla. Por otro lado se mostrara la acción de control, y por ultimo el error. En todas las graficas se mostrara la señal teórica y la real, actuando el automatismo contra la planta virtual en un bucle de tipo "hardware in the loop".

Tras los análisis por separado de ambos reguladores, compararemos los resultados entre ambos decidiendo así que tipo de implementación es la mas adecuada para el sistema.

8.2.1 PID estándar mediante bloques

En esta sección vamos a comparar el regulador PID estándar implementado mediante bloques en el automata (subsección 7.4.1), con el PID teórico (subsección 3.5.1). Extrayendo los datos y realizando un análisis estadístico obtenemos los resultados que se muestran en figura 8.1.

Podemos observar que el ajuste es casi nulo mediante la implementación por bloques, teniendo un ajuste total del del 2,30 %. Este error se ha obtenido mediante el calculo de desviaciones de los valores medios.

Analizando la señal de la temperatura, la gráfica de arriba de la figura 8.1, observamos que la forma es casi idéntica. Con el fin de ver ambas se ha desfasado la del regulador real unas unidades hacia arriba. Vemos que las pendientes no son idénticas, de ahí que la gráfica cuente con tan solo un ajuste del 2,30 %. Este error lo podemos achacar al cambio de pendiente que se aprecia.En la figura 8.2 podemos observar esta desviación, donde mostramos en la parte superior la temperatura del sistema, con las fluctuaciones producidas por el regulador y en la parte de abajo la señal de control.

Si pasamos ahora a analizar la señal de control, observamos que son nuevamente casi idénticas. Otra vez podemos observar que la pendiente es la que varia. Si ahora pasamos a observar la señal de control de la figura 8.2, veremos que el regulador real satura en los cambios bruscos de pendiente, mientras que el regulador teórico no satura.

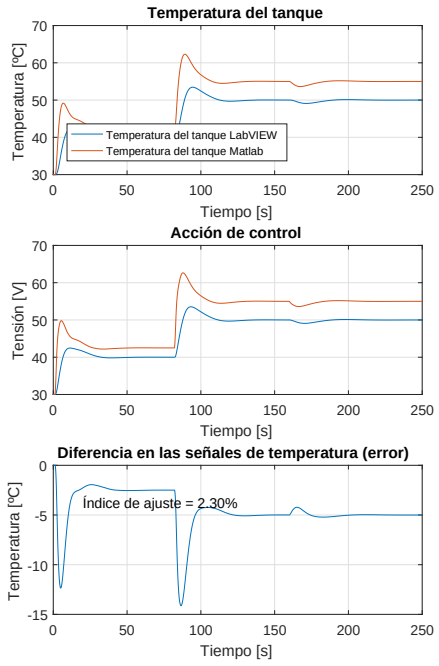

Figura 8.1: Comparación del regulador estándar implementado mediante bloques

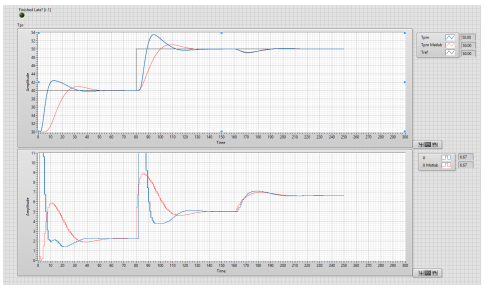

Figura 8.2: Comparación del regulador estándar implementado mediante bloques

Tras este análisis podemos asegurar que el comportamiento del regular es acorde al diseño, aunque presente un ajuste tan bajo. Si analizamos el comportamiento ante la planta vemos que cumple con los requisitos de diseño, considerándolo una buena opción para la implementación en la planta real.

8.2.2 PID estándar discreto

En esta sección compararemos el PID estándar discreto, el cual mostramos como implementar en subsección 7.4.2, con el regulador teórico.

Si analizamos estadisticamente el ajuste este regulador, ajuste que se observa en la **??**, vemos que difiere bastante del ideal.

Si analizamos nuevamente las gráficas mostradas en figura 8.3, observamos que el regulador teórico cuenta con mayor pendiente, tanto de subida como de bajada, en los puntos donde la referencia cambia bruscamente, este echo se traducirá en una subida más rápida. Si analizamos esta señal con la mostrada en figura 8.4, observamos como se cumple este echo aunque las señales sean parecidas. En esta gráfica podemos observar como las señales son muy parecidas, exceptuando los picos de sobrepasamiento.

Si analizamos la señal de control en lafigura 8.3, podemos observar como la forma de las ondas es casi idéntica. Si ahora la observamos en figura 8.4, podemos afirmarlo, aunque en ambas se observa como los picos de sobrepasamiento son algo superiores.

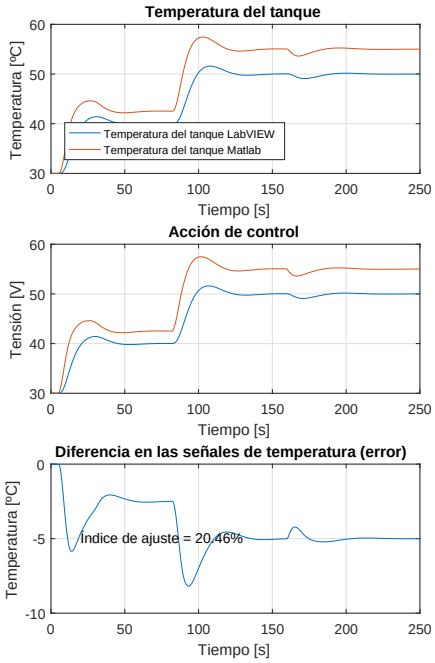

Figura 8.3: Comparación del regulador estándar discreto

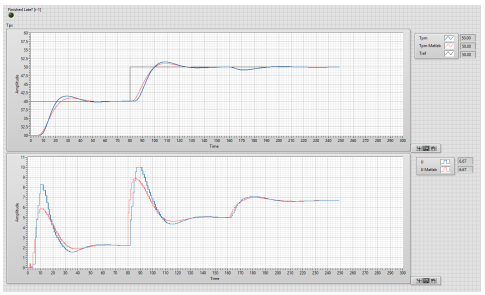

Figura 8.4: Comparación del regulador estándar discreto

Con todo lo analizado hasta ahora podemos concluir con la afirmación de que este regulador es completamente funcional para la implementación en la factoría.

8.2.3 Comparación de implementaciones

Tras analizar estadísticamente el regulador observamos que el comportamiento de los reguladores discretos es notablemente superior. Este echo también lo podemos observar en las simulaciones ante la planta virtual en un "hardware in the loop".

Como era de esperar el regulador teórico es bastante mas preciso que los reguladores que podamos implementar. Otro echo que se observa es la mejora del regulador implementando los mismos parámetros de forma discreta, ya que de este modo se implementa unicamente el algoritmo depurado del regulador ideal dejando a un lado las características extras que el regulador en bloque presenta.

Para concluir con este capitulo habría que implementar el regulador en cascada de forma discreta, ya que como se ha visto y podemos intuir, este funcionara mejor. Esta implementación no se considera necesaria ya que la planta funciona de manera adecuada con un regulador estándar.

Capítulo 9

Conclusiones

El objetivo fundamental del proyecto consiste en aunar los conocimientos adquiridos en los distintos bloques curriculares del grado en un único proyecto, conocer la programación de autómatas que no se han visto en la carrera, e investigar sobre diversas soluciones que el proyecto va presentado a su paso adquiriendo así autonomía y liderazgo ante futuros retos. Estos echos creo que quedan demostrados habiendo sido capaz de montar un sistema que reúna todas estas características.

Por otro lado se intentaba demostrar las funciones que un servidor OPC puede aportar a un sistema automático y a la industria en general. En este proyecto nos centramos en realizar un estudio técnico económico de las mejoras que supondrían implementar un sistema más automatizado, y con un regulador que ahorrase tiempo y mejorase la respuesta del sistema. Para ello nos basamos en la planta virtualizada de un proyecto docente (**tfgencamisado**) de esta misma escuela, e intentamos realizar una virtualización completa del sistema contra el que testar un autómata, decidiendo así si el comportamiento es el adecuado o por el contrario habría que utilizar técnicas mas avanzadas de control, o otro tipo de reguladores con mejores características. Mediante este sistema conseguiríamos demostrar a la empresa el funcionamiento que tendrían en la planta sin haber parado un solo segundo está, ahorrando así los coste que ello supone. Este sistema "hardware in the loop" es una técnica que se esta empezando a utilizar en las nuevas fases de industrialización de las grandes empresas.

Se a realizado la automatización de la planta completa, mejorando así la rapidez de la misma, y ofreciendo al cliente la posibilidad de implementar esta con diversas tecnologías. Para la realización de esta automatización se ha investigado sobre la normativa actual vigente de los organismos tanto nacionales como internacionales.

Otra característica de este proyecto ha sido la de unificar aplicaciones al servidor OPC. Se ha mostrado como configurar un servidor y como vincular parte de las variables definidas en este a una hoja de calculo. Esta característica es de especial relevancia ya que integra en una misma tarea de automatización la gestión y la calidad de los productos que mejoramos, ya que de este modo tendremos datos en tiempo real del proceso. Este echo permitirá abastecer y dar salida a los productos ahorrando tiempo de almacenaje. Permitirá tener controlada, mediante la lectura de los sensores, el estado de la planta en todo momento, generando así informes de calidad

que permitan a las empresas demostrar que los productos ses han fabricado bajo la normativa vigente.

Otra característica que se ha intentado demostrar a sido la unión de los diversos métodos de programación que se han impartido en el grado, en un único proyecto que demuestre la versatilidad que este grado ofrece.

Capítulo 10

Bibliografía

@bookdorf2005sistemas, title=Sistemas de control moderno, author=Dorf, R.C. and Bishop, R.H. and Canto, S.D. and Canto, R.D., isbn=9788420544014, series=Fuera de colección Out of series, url=https://books.google.es/books?id=NPRGAAAACAAJ, year=2005, publisher=Pearson Educación

@BookWarwick1988, title = Industrial Digital Control Systems, publisher = Peter Peregrinus, year = 1988, author = Warwick, K. and Rees, D., series = Control, Robotics and Sensors Series, isbn = 9780863411373, lccn = 88205622, url = https://books.google.es/books?id=4dURB2NTstAC,

@BookOgata2002, title = Modern Control Engineering, publisher = Prentice Hall, year = 2002, author = Ogata, K., series = graph. Darst, isbn = 9780130609076, lccn = 20021683, url = https://books.google.es/books?id=PaseAQAAIAAJ,

@BookJarauta2000, title = Análisis matemático de una variable: fundamentos y aplicaciones, publisher = Edicions UPC, year = 2000, author = Jarauta, E. and Bragulat, E.J., series = Matemticas y Estad¡stica, isbn = 9788483014103, url = https://books.google.es/books?id=1DFumclsLWEC,

@BookAstroem1988, title = Heuristics for Assessement of PID Control with Ziegler-Nichols Tuning, year = 1988, author = Åström, K.J. and Persson, P. and Hang, C.C., series = Institutionen för reglerteknik, Lunds tekniska högskola, url = https://books.google.es/books?id=G4LsMQAACAAJ,

@MastersThesistfgencamisado, author = Jos? Llacer Espa?a, title = Control en cascada de la temperatura de un reactor continuo de tanque agitado encamisado, school = EPSA, year = 2018, url = https://riunet.upv.es/bitstream/handle/10251/101509/LL

Capítulo 11

Anexos

En este presente capitulo se cierra el proyecto con una guia paso a paso de como se genera el servidor OPC, las configuraciones que hay que adoptar en el programa del autómata para que estas variables sean accesibles. Las distintas configuraciones que se pueden realizar con a la hora de configurar este automatismo, y un ejemplo practico paso a paso de como declarar variables.

Por otro lado se explica como generar un excel maestro con la declaración de variables del servidor, este excel no se ha generado en el proyecto por la naturaleza del mismo. La red OPC que se explica en el proyecto unicamente se utiliza para testar la suficiencia del autómata, no para implantar una mejora en la empresa con la que obtener datos "on time" de la producción, aunque con esta guía cualquier lector sera capaz de realizarlo. Se desglosara en varios puntos, tantos como programas, y se analizaran paso a paso con fotografías las distintas opciones.

11.1 Introducción

Para la conexión de este servidor OPC se necesita contar con los siguientes programas:

- TIAPorta V.14
- NI OPC Server
- LABView

11.2 Guia

11.2.1 TIAPorta V.14

Para la conexión del servidor OPC en este ordenador se necesita tener instalado el programa, cuyos requisitos mínimos son los citados en la tabla 11.1.

Procesador	i5-3320M 3,3 GHz o superior
RAM	8 GB o más
Disco duro	300 GB o más
Pantalla	Wide Screen de 15.6" (1920x1080)
Sistema operativo	64 bits

Tabla 11.1: Requisitos mínimos del programa TIAPorta V.14

Necesitaremos conocer las características de nuestro autómata, y tenerlo conectado al ordenador por medio de un cable de red, ya sea directamente o mediante de un switch que los interconecte. En este caso el autómata es el S7-1200. Algunos de los requisitos que debemos conocer previamente son los citados en la tabla 11.2

Autómata	
Modelo	S7-1200
CPU	1214C
Alimentación	AC/DC/Rly
Referencia	6ES7 214-1BG40-0XB0
Versión	V4.0
Complemento	
Signal Board	AQ
Versión	AQ1
Modelo	6ES7 232-4HA30-0XB0
Versión	V1.0

Tabla 11.2: Características del autómata

Pasos a seguir

A continuación se detallarán los pasos a seguir para generar una variable en un servidor OPC, que controlará una salida física del autómata. Posteriormente, a través del servidor OPC y mediante otro programa informático se conmutará una salida física del autómataesta:

1. **Creamos el proyecto:** lo generamos en una ruta conocida para posteriormente guardar allí las rutas creadas del servidor y otros archivos. Esto lo podemos observar en la figura 11.1.

2. **Agregamos el dispositivo:** Tras generar el nuevo proyecto en la columna de la izquierda, añadiremos el autómata citado en la tabla 11.2, prestando especial atención a la versión del autómata, ya que en caso de error, éste comunicará pero no creará el servidor correctamente. Así se observa en la figura 11.2.

3. **Conectamos la signal board:** Tras seleccionar el autómata debemos conectar la también citada en la tabla 11.2, signal board. Ésta se encuentra en la tabla de la derecha y debemos arrastrarla hasta su posición en el autómata. Esto lo podemos observar en la figura 11.3.

4. **Confiramos la IP de la máquina:** seleccionamos el autómata al que queremos cambiar la IP, pulsamos con el botón derecho sobre éste y seleccionamos la ultima opción de propiedades. Tras pulsar esta opción iremos a la pestaña de Interface Profinet [x1] y tal como se ve en la figura 11.4. Tras esto debemos comprobar que la conexión y la comunicación con el autómata son correctas.

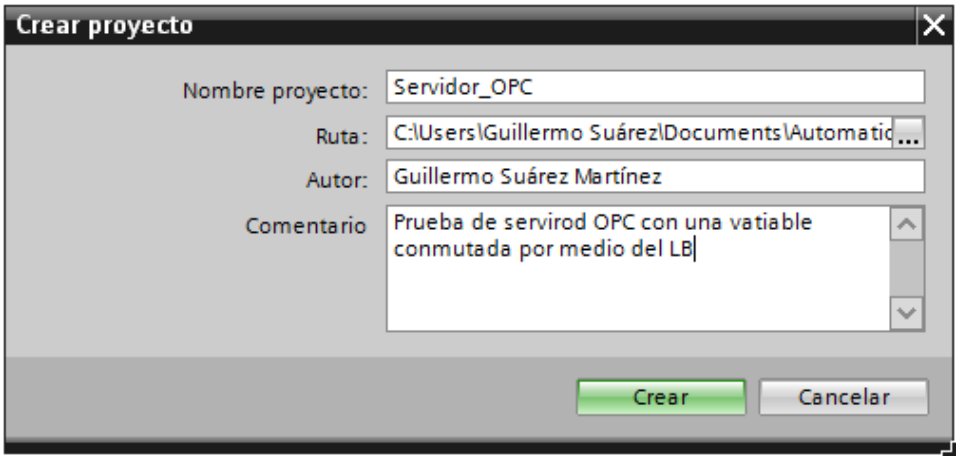

Figura 11.1: Ventana de creación para nuevo proyecto

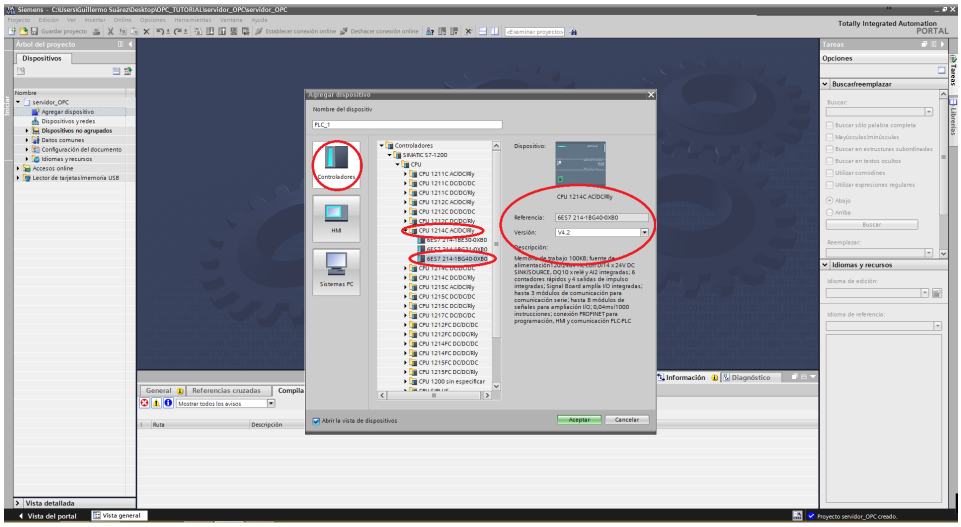

Figura 11.2: Ventana para añadir el autómata

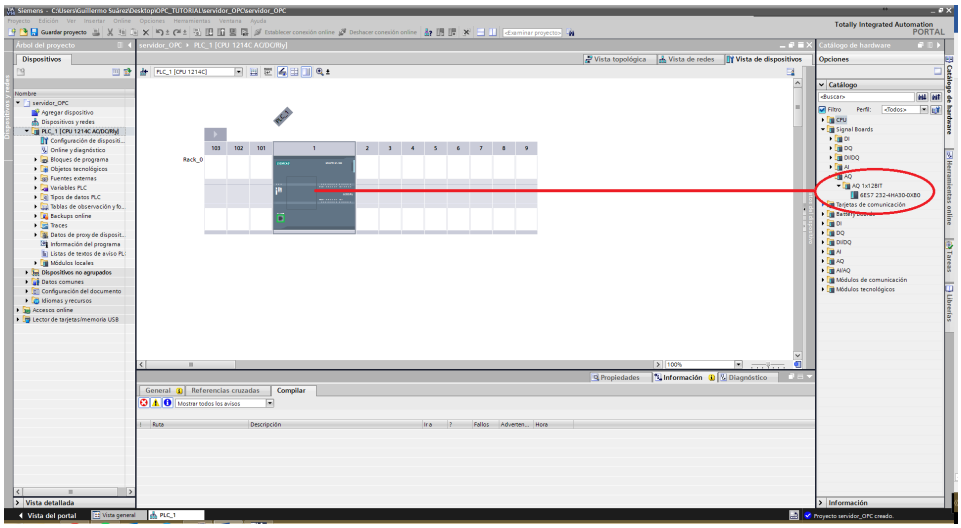

Figura 11.3: Ventana para añadir la signal board

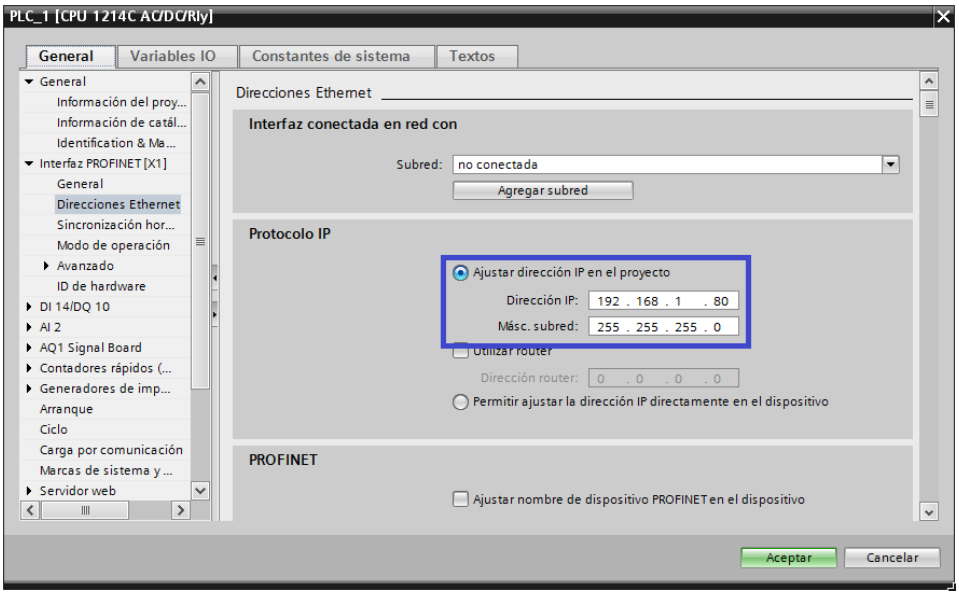

Figura 11.4: Ventana de Interface Profinet [X1]

5. **Comprobación de comunicación:** Para comprobar que la comunicación es efectiva debemos conectarnos al autómata, borrar los programas que tiene en memoria para posteriormente poder sobrescribir. Esto se puede ver en la figura 11.5. Después se abrirá una nueva ventana en la que comprobaremos que la IP de la máquina coincide con la de esta ventana. Debemos tener claro qué tarjeta de memoria es la que está conectada al autómata, tal y como se dijo en el subsección 11.2.1. Tras comprobarlo, pulsaremos ”Iniciar búsqueda”, y aparecerán todos los autómatas conectados a la red. Debemos seleccionar el nuestro y pulsar ”parpadear LED”, como se observa en la figura 11.6

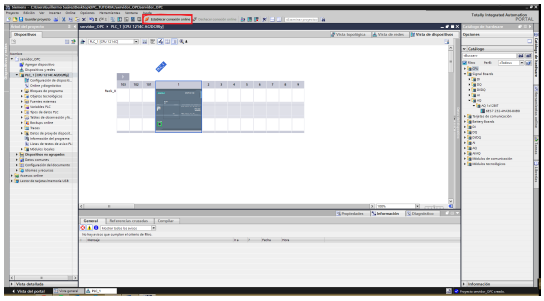

Figura 11.5: Pantalla principal con detalle de conexión

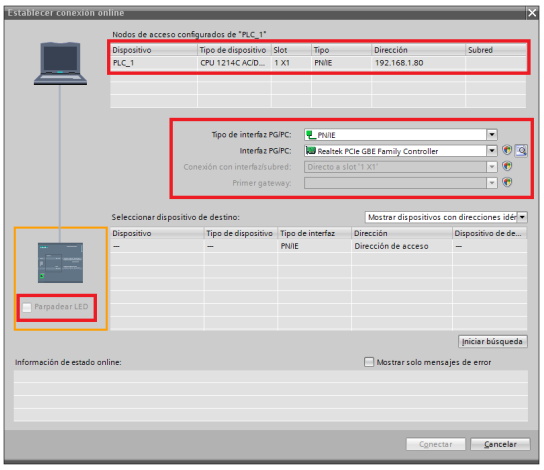

Figura 11.6: Ventana de conexión a autómatas

6. **Borramos la información del autómata:** comenzamos por deshacer la conexión. Para ello se pulsa el botón que está al lado de ”Establecer conexión”. Volvemos a darle con el botón derecho sobre del autómata, seleccionando ”cargar en dispositivo” y posteriormente en ”hardware y software”. Esto lo podemos ver en la figura 11.7.

7. **Creamos un bloque de datos:** tras el borrado del autómata, crearemos el bloque de datos. Para ello, en el panel de la derecha añadimos un bloque de programas, seleccionando

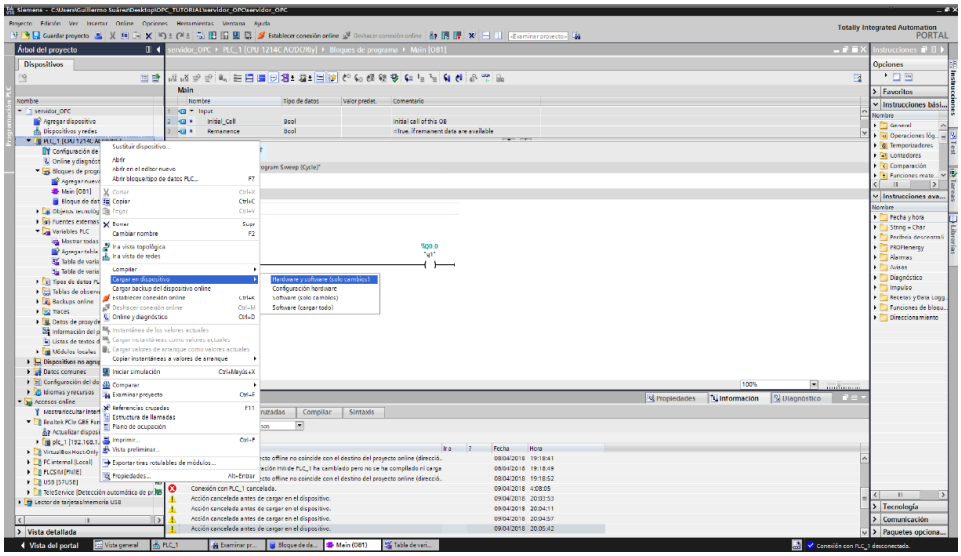

Figura 11.7: compilar software y hardware

”bloque de datos DB global” como se observa en la figura 11.8. Luego modificamos las propiedades del bloque de datos desmarcando en la pestaña ”atributos” la opción ”acceso a optimizado al bloque”. Esto lo podemos observar en figura 11.9

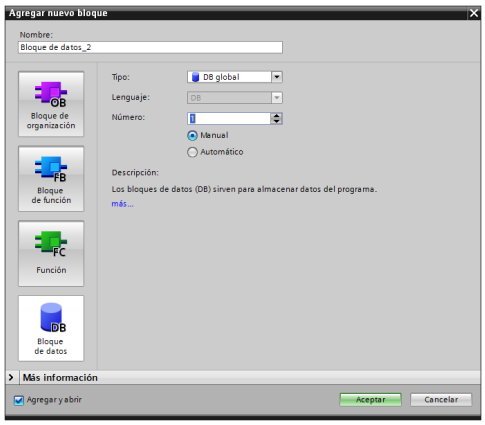

Figura 11.8: Agregar bloques de funciones

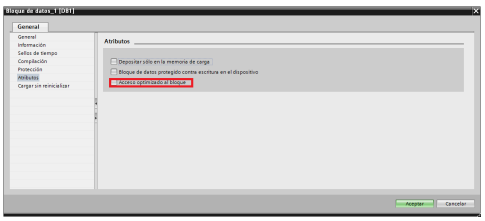

Figura 11.9: Modificar atributos del bloque de datos

8. **Modificamos propiedades del PLC:** para poder acceder posteriormente a los datos desde el servidor OPC, debemos modificar las propiedades. Para accede a las propiedades citadas, en la pestaña ”protección” activaremos la opción de ”acceso a la vía de comunicación” tal y como se ve en figura 11.10

9. **Creamos las variables a compartir:** tras tener configurado el programa, a falta de escribir todos los cambios en el autómata, debemos crear las variables a compartir dentro de nuestro DB1. Como se trata de una guía, crearemos una única variable que conmutara una salida física del autómata, tal y como explicamos anteriormente. En este caso crearemos la variable ”Salida_Q1”, como puede verse en la figura 11.11. Esta variable será accesible para la programación normal del autómata y para comprobar que funciona la conectaremos a la salida del autómata Q1, como podemos observar en la figura 11.12.

10. **Reescribimos el autómata:** repetimos el paso anterior ”borramos la información del autómata” sabiendo que ahora se escribirá el programa y la configuración anterior.

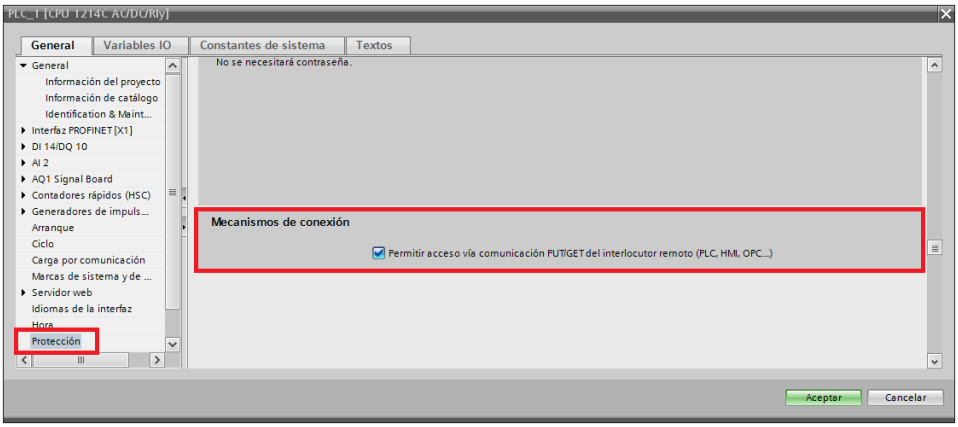

Figura 11.10: Activamos el acceso para comunicación OPC

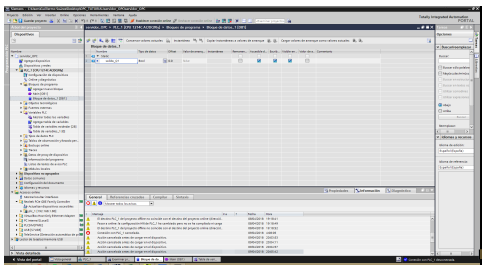

Figura 11.11: Agregar bloques de funciones

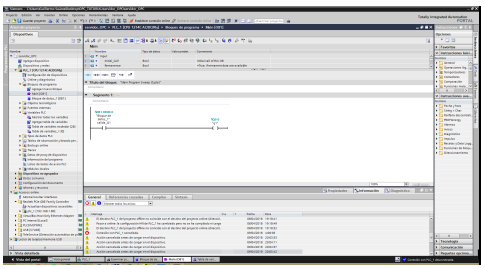

Figura 11.12: Conectamos la variable a la salida física

11.2.2 NI OPC Server

Para configurar el servidor OPC en nuestro ordenador, necesitamos cumplir los requisitos mínimos del programa NI OPC Server que se concretan en la tabla 11.3. Hay que tener en cuenta que este ordenador no debe ser el mismo que el usado para la configuración del TIAPorta V.14. Sin embargo, ambos ordenadores han de estar conectados en red, ya sea directamente o mediante un elemento intermedio como un switch o router. Este ordenador será el servidor OPC, al que se conectaran los clientes OPC, en este caso el autómata y LabVIEW

Procesador	2 GHz o superior
RAM	1 GB o más
Disco duro	180 Mb o más
Pantalla	Super VGA (800x600) o superior
Sistema operativo	32/64 bits

Tabla 11.3: Requisitos mínimos del programa NI OPC Server

Para más información podéis consultar la página del fabricante

Pasos a seguir

A continuación se indicarán los pasos para configurar un servidor OPC con la variable creada en la subsección 11.2.1, que posteriormente controlaremos mediante el LabVIEW. Es necesario distinguir tres partes:

1. Canal de comunicación: Al programador le compete conocer la tarjeta de red o medio seleccionado donde están conectados los clientes OPC. Esto lo trataremos en el Párrafo 11.2.2

2. Elemento: En este caso será el propio LabVIEW o el autómata, como cliente del servidor, quienes compartan el canal de comunicación ya sea para leer o escribir tareas. Esto lo trataremos en el Párrafo 11.2.2

3. Variable: serán los distintos registros de la base de datos los que configuraremos para que sean de lectura, escritura, tiempos de respuesta... Esto lo trataremos en el **??**

Canal de comunicación

- **Creamos un nuevo canal de comunicación:** Para ello daremos doble clic en "click to add a chanel", tal y como se muestra en la figura 11.13

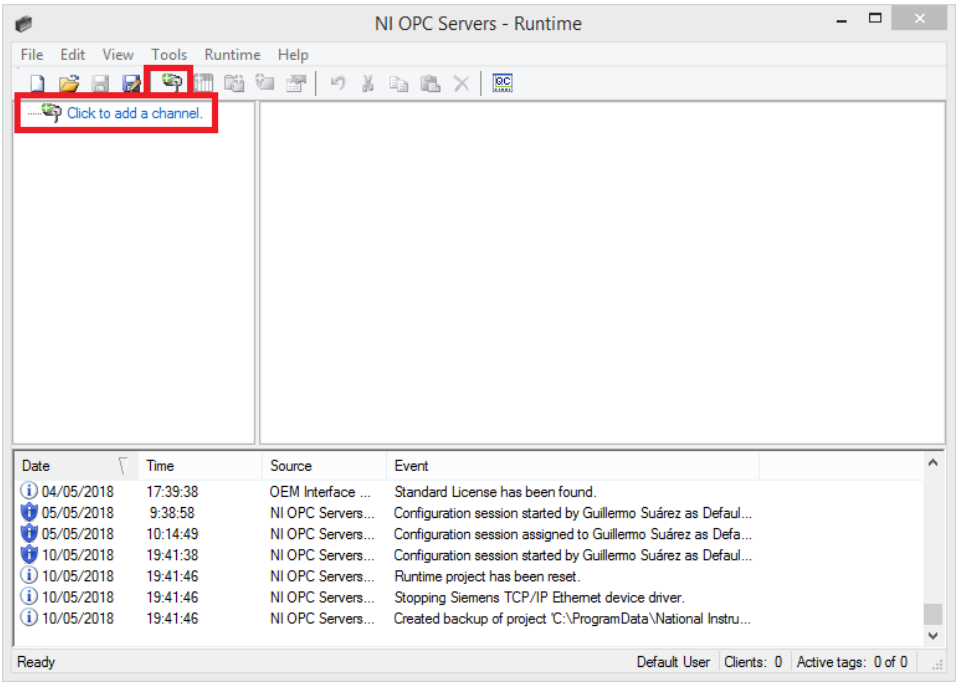

Figura 11.13: Creamos nuevo canal de comunicación para servidor OPC

- **Nombramos el canal con un nombre representativo:** este punto es importante debido a que en un mismo servidor pueden existir distintos canales, tantos como naturalezas de las variables que necesitemos. En este caso lo nombraremos como SIEMENS. Esto lo podemos observar en la figura 11.14.

- **Seleccionamos el Driver de origen de la variable:** elegimos el origen de la variable que queremos declarar en el servidor. En este caso el autómata se comunica con el protocolo TCP/IP de Siemens. Por eso seleccionaremos "Siemens TCP/IP Ethernet", tal y como se

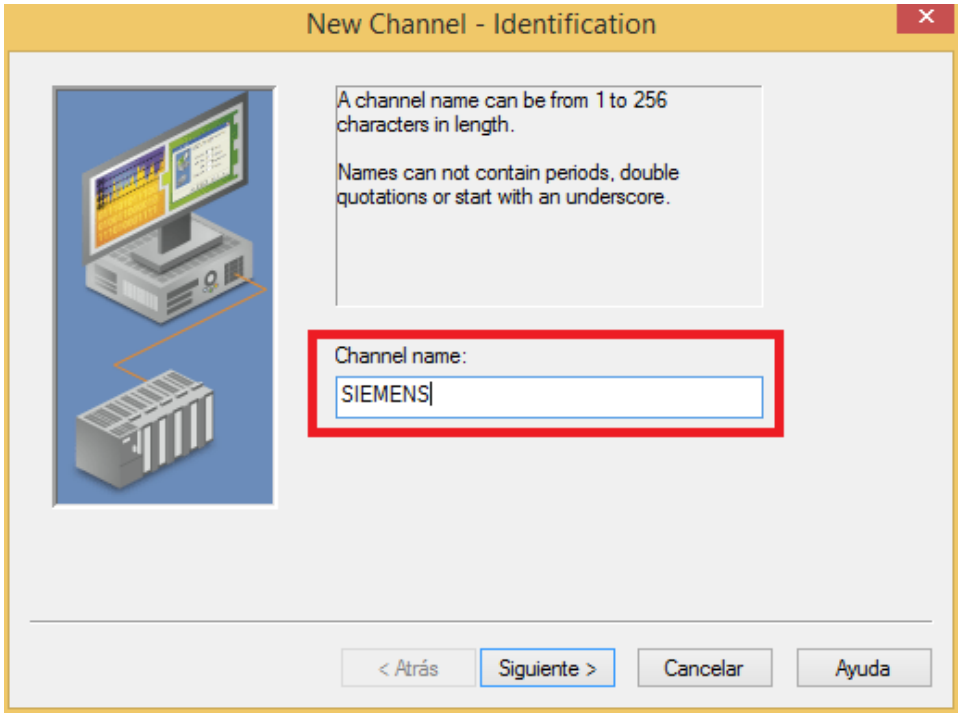

Figura 11.14: Ponemos nombre al canal de comunicación

muestra en la figura 11.15. Es un error común confundir ”Siemens TCP/IP Ethernet” por ”Siemens TCP/IP Slave Ethernet”. Aunque funcionaría de igual modo en este caso, si creamos más canales de comunicación de Siemens para comunicar entre si más autómatas de la misma marca, no podremos utilizar las variables definidas en el servidor, ya que solo el maestro lo podrá hacer.

- **Seleccionamos la tarjeta de red:** Seleccionaremos la tarjeta de red que esté interconectada con el autómata, tal y como se citó en la subsección 11.2.1. Esto lo podemos observar en la figura 11.16. Como para este ejemplo estamos utilizando el mismo PC, debería coincidir con la configurada en la figura 11.6.

- **Optimizamos los parámetros de escritura:** Estos parámetros son importantes debido a que puede ser determinantes para el muestreo de señales. Debemos tener claro la velocidad de la planta o sistema para contar con que los datos sean suficientes representativos. Hay que evitar tener un elevado numero de datos sin variación en el caso de que nuestro sistema sea demasiado lento, ya que podríamos llegar a saturar el servidor. En este caso hemos seleccionado una escritura cada milisegundo, como podemos observar en la figura 11.17. Por otro lado se deben normalizar los valores cuando estos no sean accesibles. En este caso, queremos que la variable sea nula cuando no sea accesible para lo que debemos dejar la configuración como en la figura 11.18.

- **Comprobamos que los datos son correctos:** Los datos deben coincidir con los citados en los puntos anteriores. En este caso, tal y como se observa en la figura 11.19 son correctos.

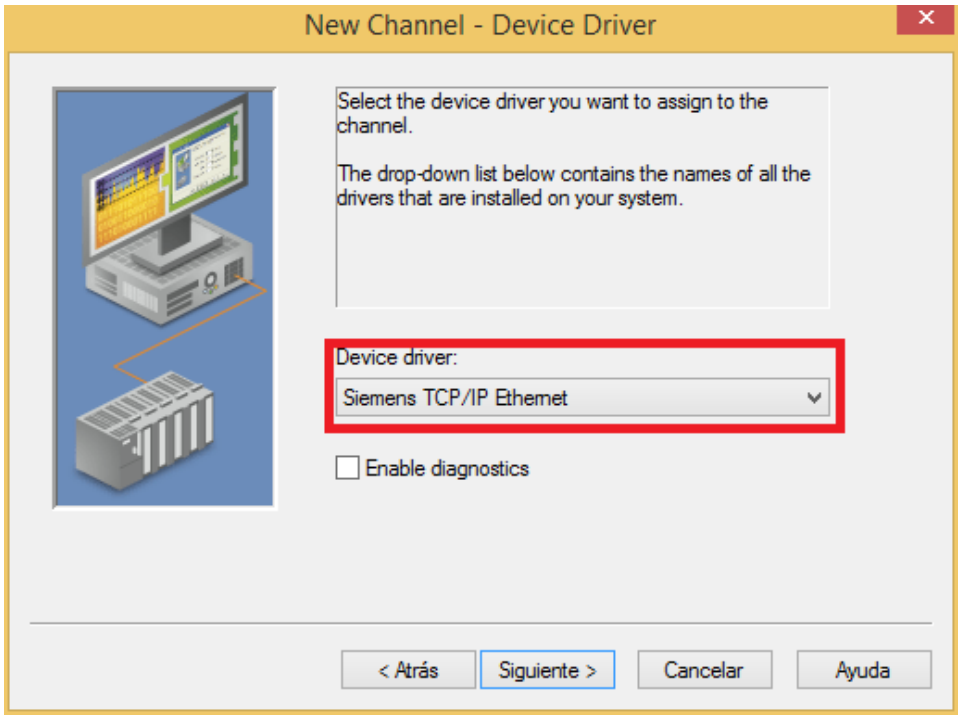

Figura 11.15: Seleccionamos el protocolo de comunicación

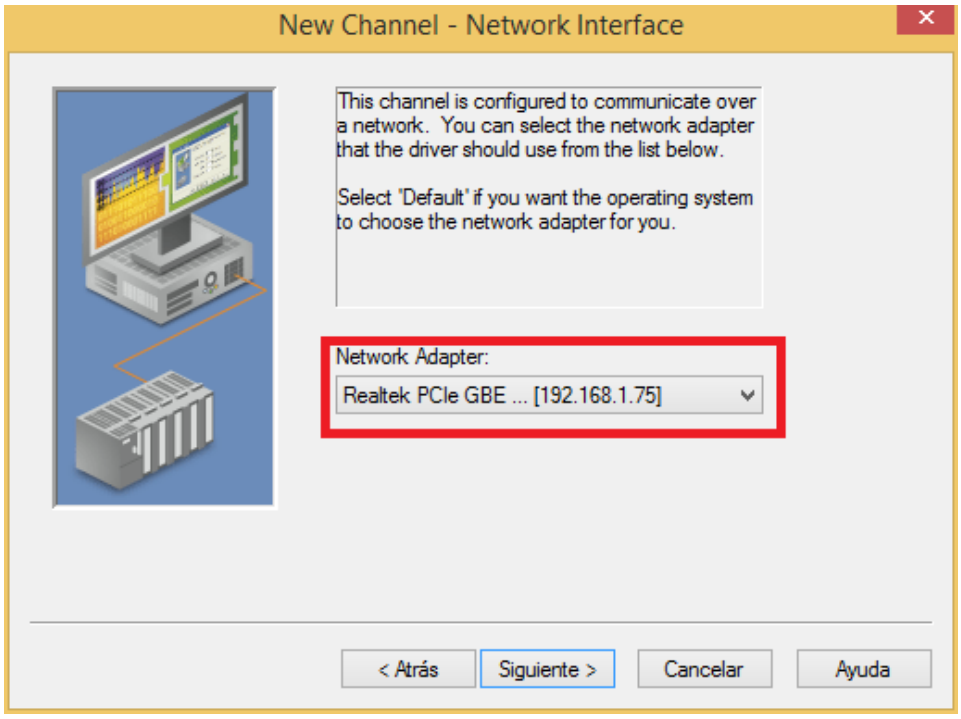

Figura 11.16: Seleccionamos la tarjeta de red

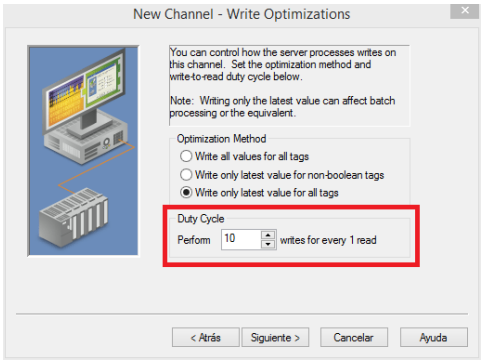

Figura 11.17: Seleccionamos la velocidad de escritura

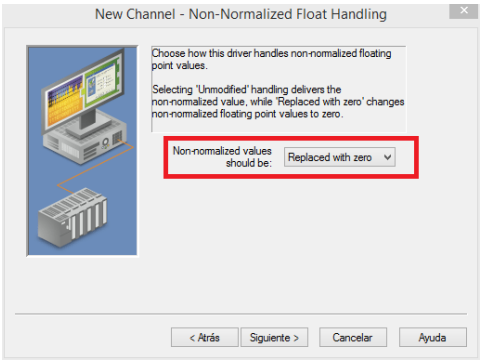

Figura 11.18: Valores cuando la variable no sea accesible

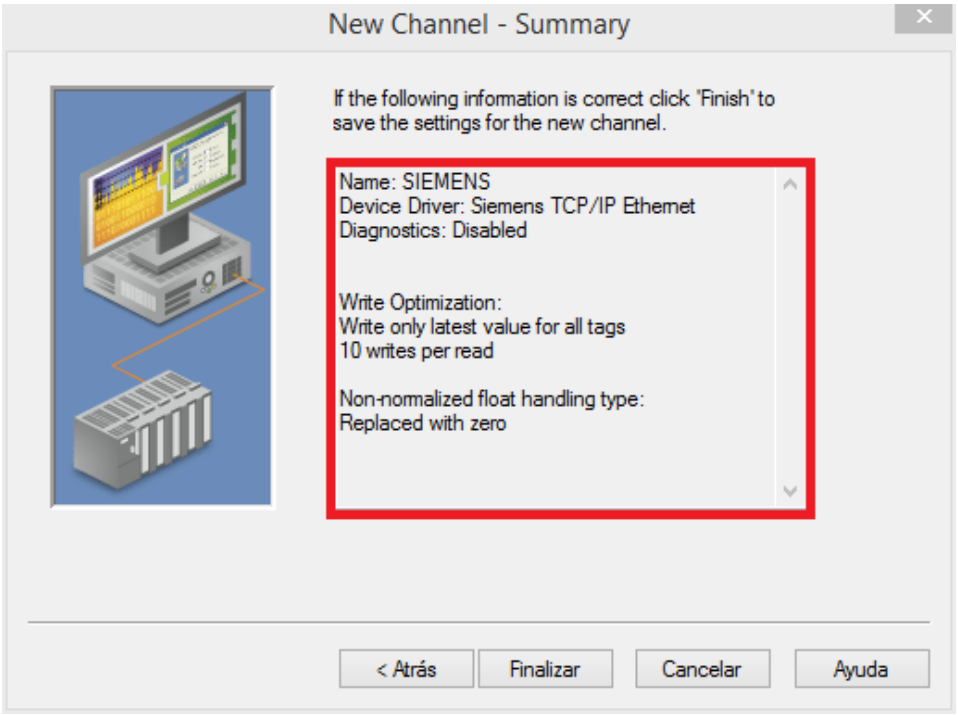

Figura 11.19: Comprobamos los parámetros

Automata

- **Añadimos el autómata:** Tras haber configurado el canal de comunicación daremos de alta al autómata pulsando con doble clic en el botón "Click to add device", tal y como podemos observar en la figura 11.20

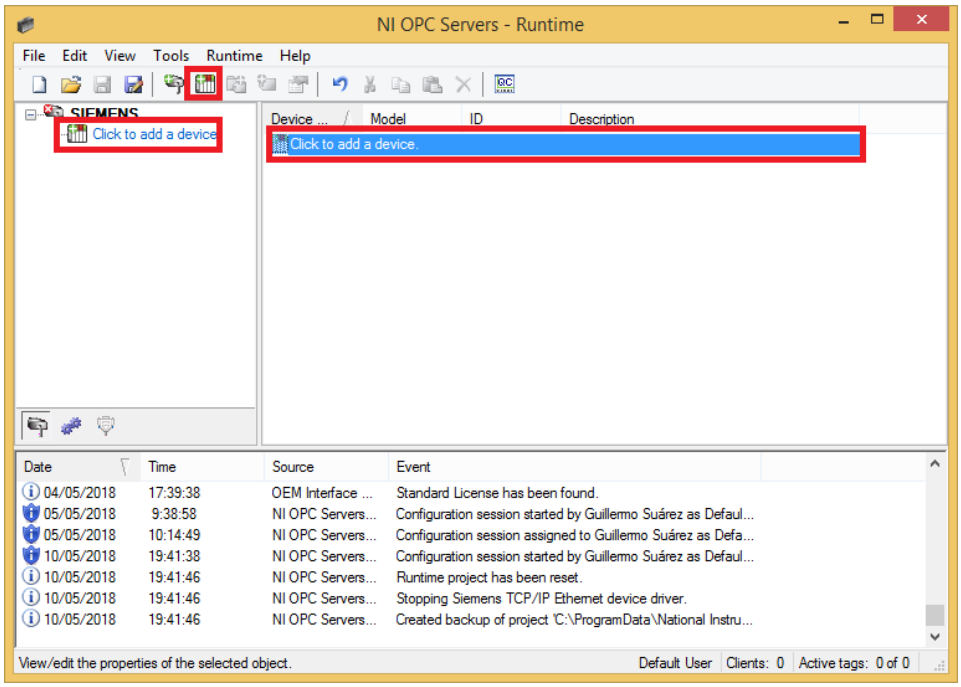

Figura 11.20: Añadimos un nuevo autómata

- **Nombramos el dispositivo a conectar:** Pondremos un nombre representativo tal y como hicimos nombrando el canal de comunicación en el Párrafo 11.2.2. En este caso lo llamaremos S-1200. As´se puede comprobar en figura 11.21

- **Seleccionamos el dispositivo:** Seleccionaremos el modelo del dispositivo para que el programa conozca el protocolo de comunicación. Este modelo ha de ser el mismo que el nombrado en la tabla 11.2, en este caso S7-1200. Eso lo podemos observar en figura 11.22.

- **Configuramos la IP:** Configuraremos la IP para conocer qué autómata tiene la variable, ya que podemos tener varios autómatas iguales pero sólo uno contendrá las variables que deseamos compartir. Esta IP ha de coincidir con la IP asignada al autómata en la figura 11.4. Podemos observarlo en la figura 11.23

- **Seleccionamos el modo de escaneo:** El medio de escaneo es una de las configuraciones más importantes del autómata, ya que si mantenemos el modo de escaneo especifico conseguiremos que sólo consulte el estado al autómata configurado anteriormente mediante la IP de la máquina, evitando así el error de consultar por nombre las variables. Por otro lado, si se borrara la variable a compartir y el autómata diese la dirección de memoria a otra variable no deseada (configuración típica de este tipo de autómatas) el servidor mostraría esta variable, aunque el nombre no coincida. Debemos tener en cuenta este aspecto porque si nuestra planta es susceptible de cambios, este método de configuración nos reportará un trabajo extra. Si por el contrario, la planta es duradera este método nos permite una mayor velocidad ya que no compara todos los requisitos de la variable. Esto lo podemos observar en la figura 11.24

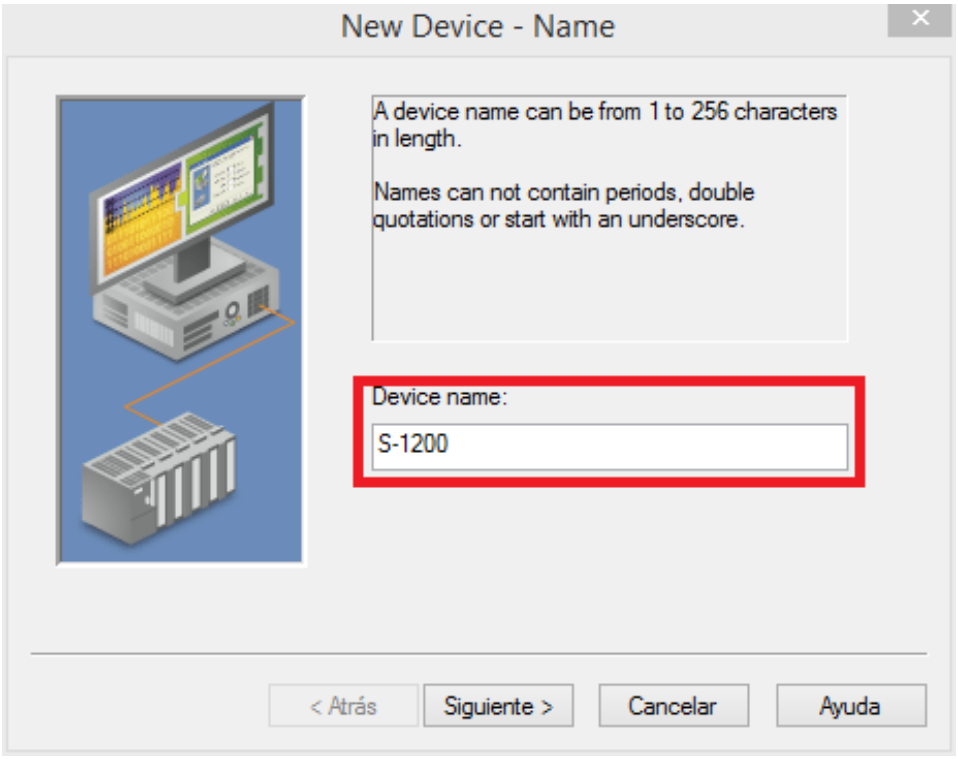

Figura 11.21: Nombre asignado al dispositivo del que queramos compartir las variables

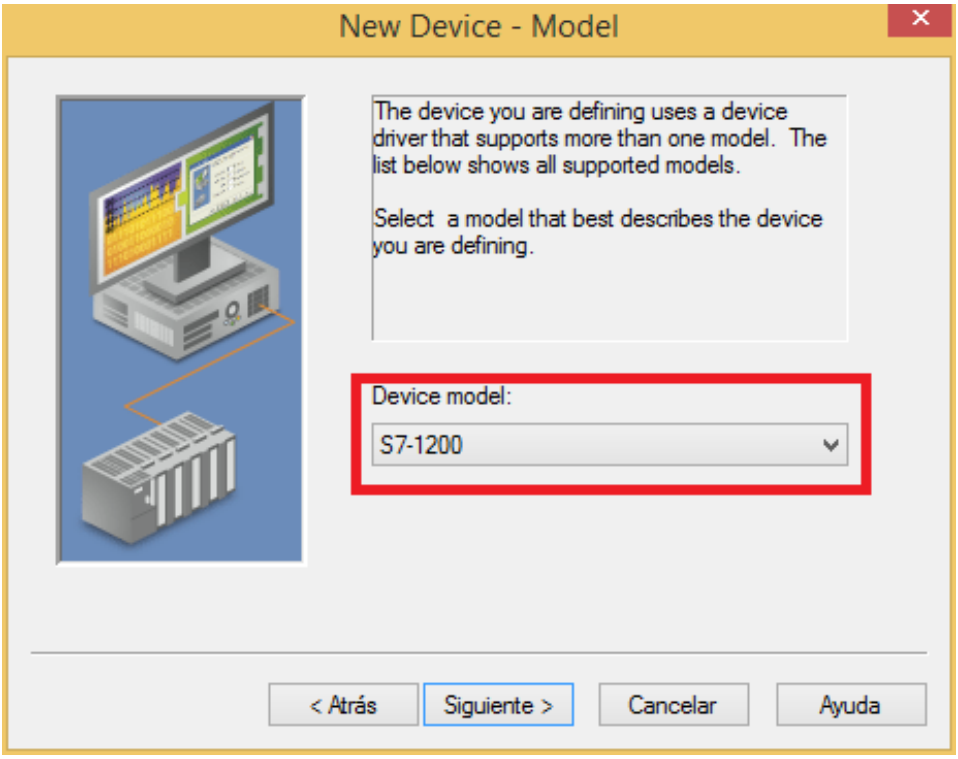

Figura 11.22: Seleccionamos el dispositivo

Figura 11.23: Configuramos la IP del dispositivo

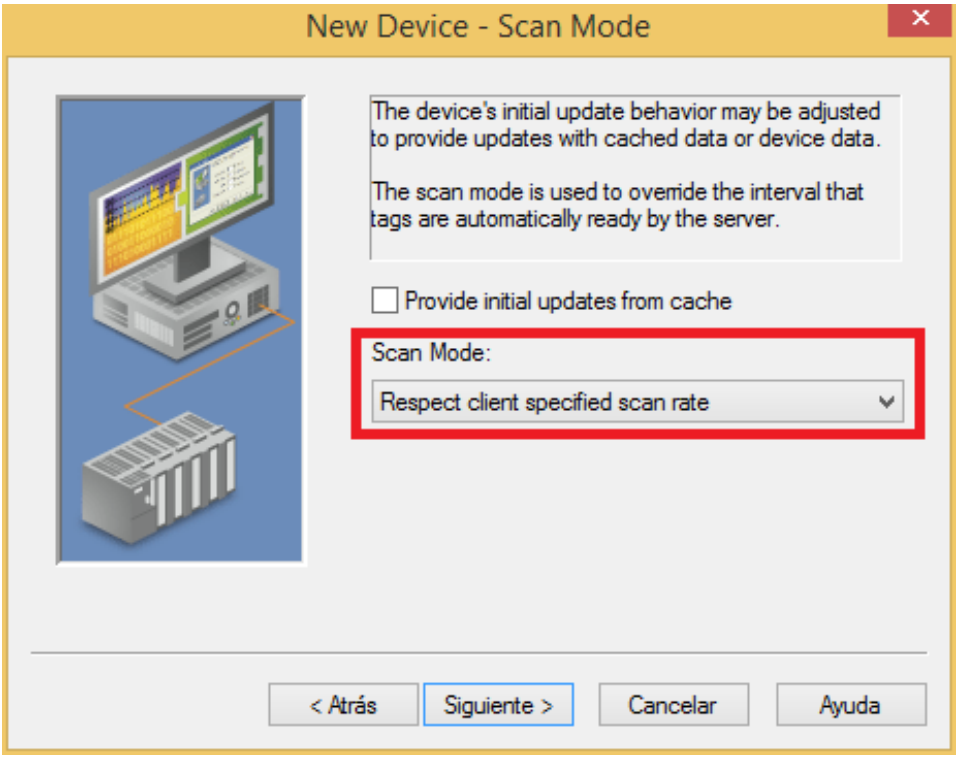

Figura 11.24: Configuramos el método de escaneo

- **Configuración de tiempo:** En este menú podemos configurar el tiempo de consulta al autómata, sabiendo que este dato, junto con el ya configurado del canal de comunicación, serán los que marquen futuras configuraciones para el controlador, evitando así errores en el control y malas interpretaciones. En este caso se dejarán los valores por defecto mostrados en la figura 11.25, ya que lo único que pretendemos es comunicar.

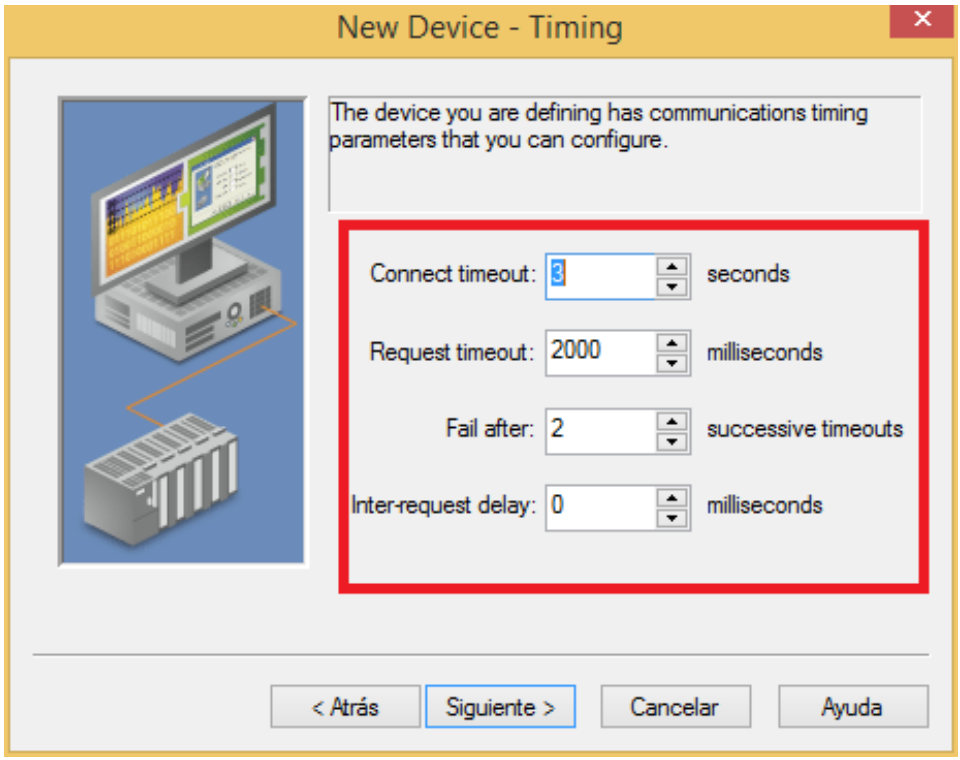

Figura 11.25: Configuración del tiempo de consulta

- **Configuramos los tiempos muertos:** Tras fallo de comunicación, podemos configurar el tiempo muerto que queramos hasta volver a lanzar la consulta. Esto puede ser importante para reducir los hilos del procesador del autómata, liberando la consulta siempre que sepamos que la variable no puede cambiar. Esto lo podemos ver en la figura 11.26

- **Configuración de la base de datos:** Podemos configurar qué hacer tras arrancar el proceso: generar una nueva base de datos, continuar con la anterior... Esta decisión se tomará en función de nuestra planta ya que si continuamos con la base anterior tendremos en memoria el ultimo estado de las variables, pudiendo enfrentar dos estados de un mismo proceso que de manera normal seria imposible enfrentarlos. Pero por otro lado, podemos pausar la máquina y continuar por el ultimo estado conocido. En este caso se borrará la base de datos. Esta configuración sera la mostrada en la figura 11.27

- **Configuramos el puerto de comunicación:** Debemos tener claro que el puerto esté libre consultando el administrador de dispositivos. Por defecto, el servidor seleccionará el puerto 102, como se puede observar en la figura 11.28

- **Configuramos la posición física:** Con el fin de obtener mas datos diremos la posición física del autómata dentro de nuestra organización, en este caso solo contamos con el autómata y el servidor, por ese motivo al no contar con mas autómatas indicar la posición del crack sera innecesario. Por otro lado sera conveniente indicar la posición del autómata si hay varios para ayudar a futuras modificaciones. Otra configuración importante sera indicar el tipo de ordenador que configuro el automara, pudiendo distinguirse tres opciones:

Figura 11.26: Configuración de tiempos muertos

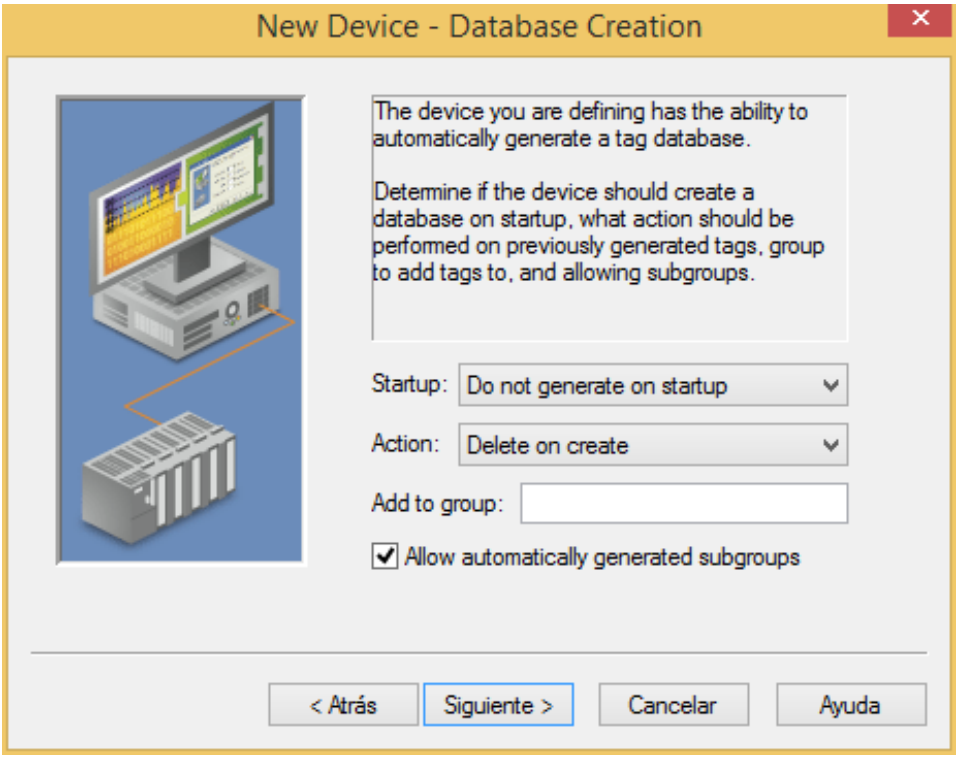

Figura 11.27: Configuración de la base de datos

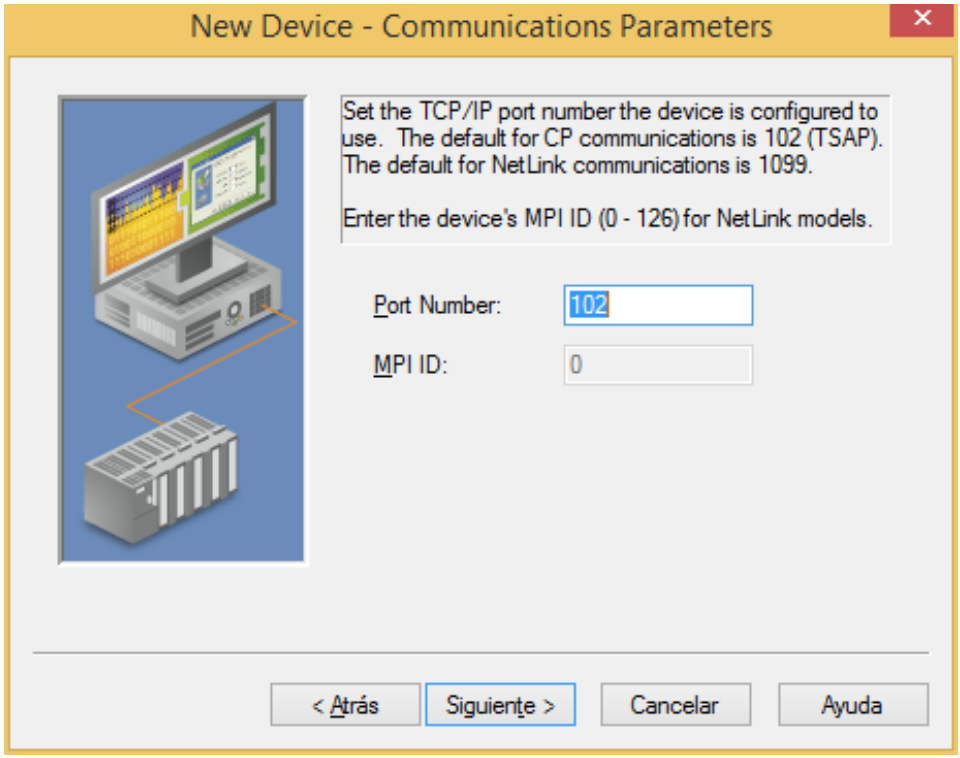

Figura 11.28: Configuración de el puerto de comunicación

- PC: Un ordenador normal con el programa TIAPorta instalado. El caso que aplica
- PG: Un ordenador de siemens que graba la tarjeta directamente, mejorando asi la velocidad del autómata y las prestaciones
- OP: Pantalla de programador, graba similar al PC pero con una preconfiguración. Es interesante para sistemas que se modifiquen con frecuencia dejando esta pantalla conectada y cargando un programa u otro

En este caso y tal y como se muestra en la figura 11.29 seleccionamos la versión PC, que es la que utilizamos para configurar el autómata.

- **Configuración de parámetros del autómata:** Esta configuración ayuda a la comunicación siempre y cuando coincida con la configurada en la figura 11.5, de este modo sabra en que slot esta conectado el automata. Esta configuración la podemos observar en la figura 11.30.
- **Configuramos la dirección de operaciones:** por defecto el protocolo de comunicación de siemens son cadenas de 16 a 32 bit, adicionalmente se puede modificar para acelerar el proceso. Por defecto se deberá dejar la configuración que se muestra en la figura 11.31
- **Comprobación del autómata:** si la configuración es la correcta deberían coincidir los datos del sumario con los puestos en los puntos anteriores. Esto lo podemos observar en la figura 11.32

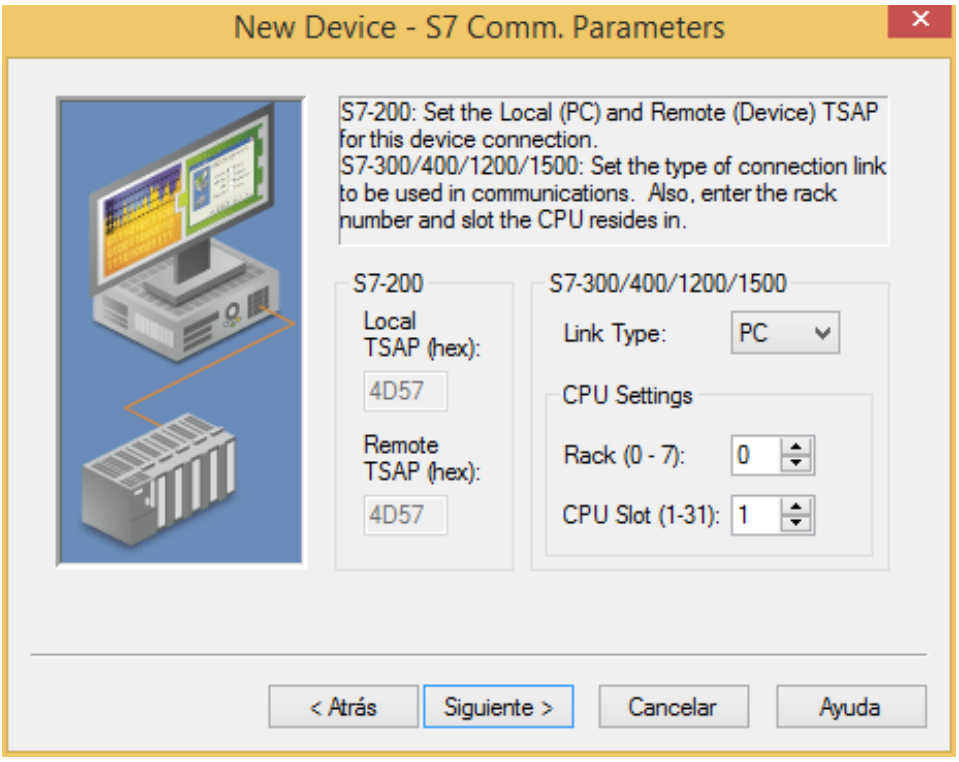

Figura 11.29: Configuración de situación física del autómata y método de configuración

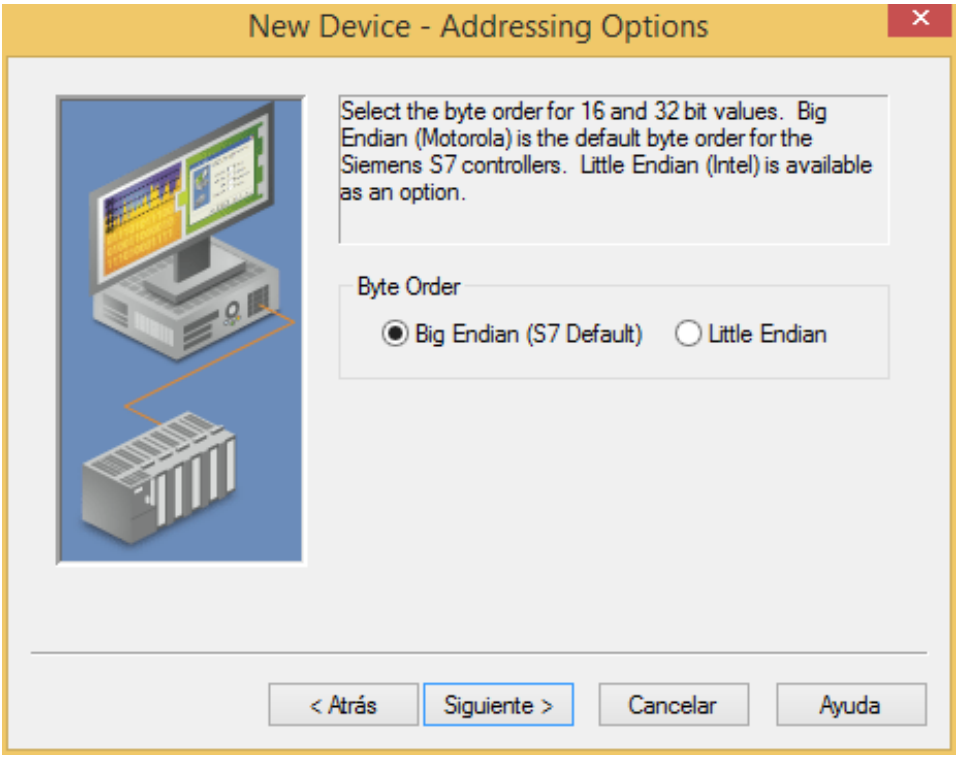

Figura 11.30: Configuración de situacion física del autómata

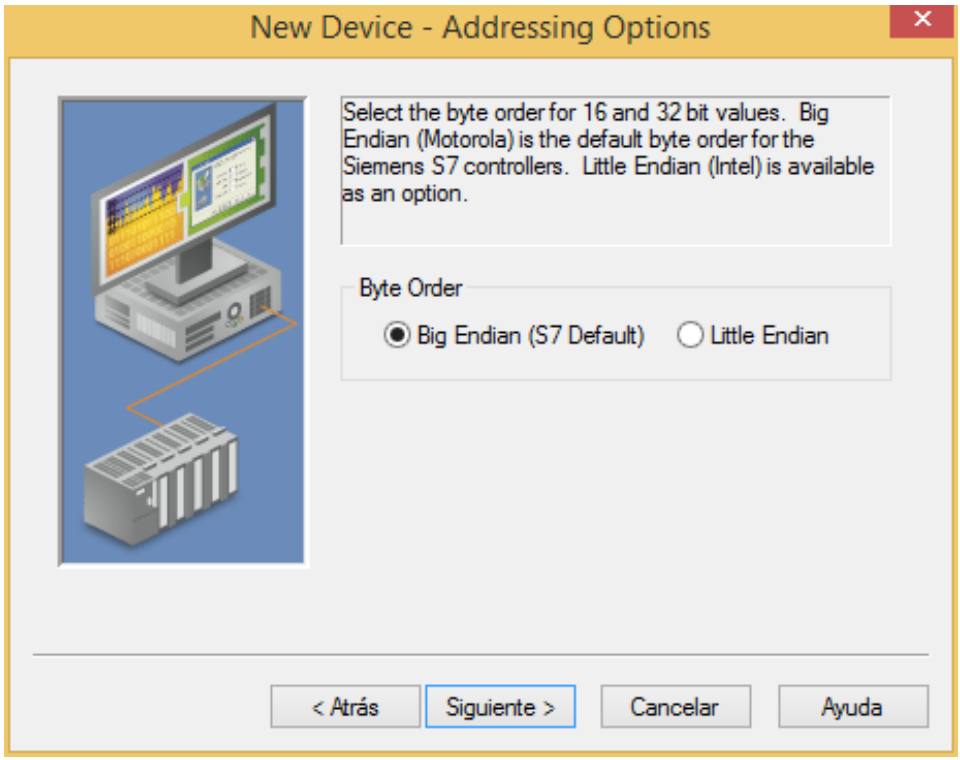

Figura 11.31: Configuración de la dirección de operaciones

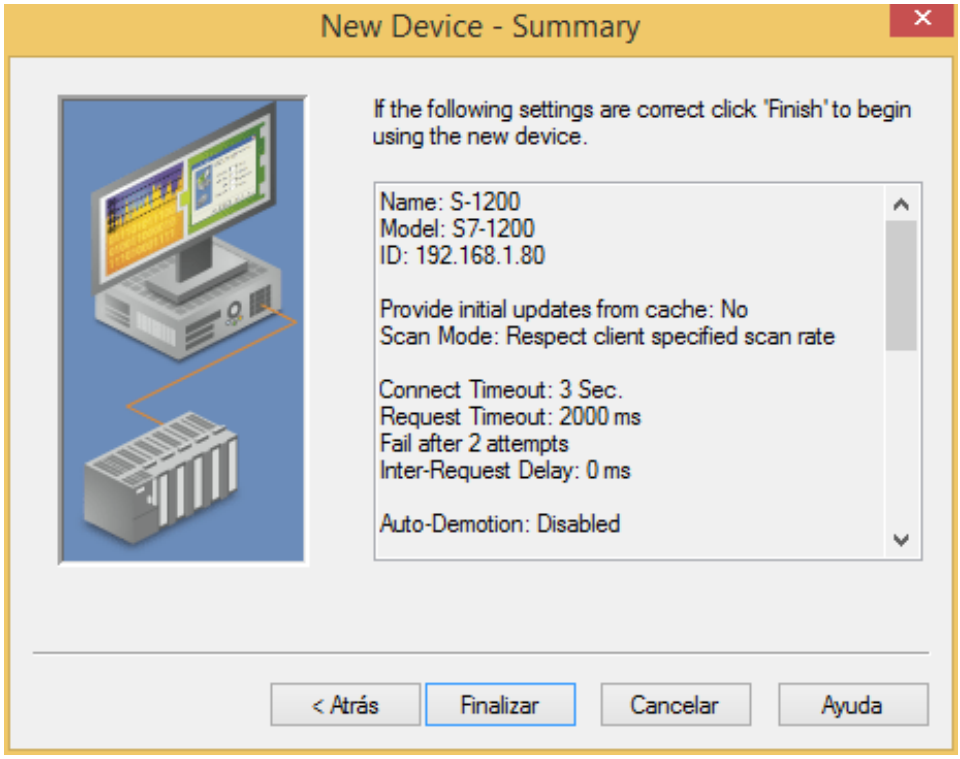

Figura 11.32: Sumario de configuración del autómata

Variables

En este punto describiremos el método para dar de alta variables en el servidor. Se pueden declarar variables de forma masiva pero aquí unicamente describiremos el método para declarar una, la que describimos en subsección 11.2.1.

- **comenzamos el proceso:** Como se ha citado anteriormente daremos doble click en NEW TAG, esto lo podemos observar en la figura 11.33

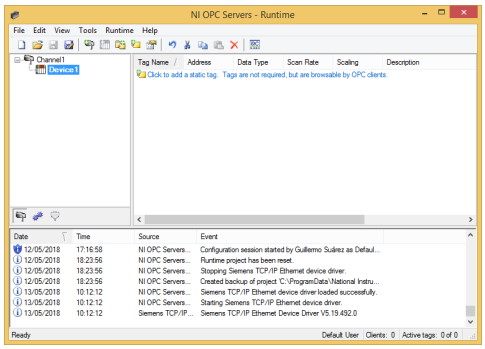

Figura 11.33: Añadimos un nuevo TAG

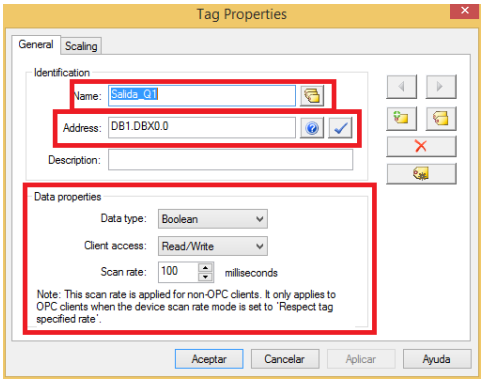

Figura 11.34: Valores que debe tener el TAG

- **Creamos la variable:** Este punto tiene varios pasos a rellenar, los cuales pasaremos a describir a continuación:

  - **Damos nombre a la variable:** Este nombre sera el que tenga la variable en el servidor, y no tiene por que coincidir con la ya dada en la subsección 11.2.1, aunque lo mas representativo sera que coincida. En nuestro caso lo realizaremos así.

  - **Nombramos la dirección física:** Este punto es el mas importante hasta ahora, ya que si escribimos otra dirección física existirá comunicación con el autómata pero éste no podrá leer o escribir la variables que nosotros deseamos en el servidor. Este dato lo obtendremos del cuerpo del programa principal del Automata. Como podemos observar en la figura 11.12, en este caso la direccion que ocupa es %DB1.DBX0.0.

  - **Descripción:** Es completamente opcional, pero recomendable ya que en este caso de ejemplo no tiene interes pero si lo necesitaremos en el caso de un servidor con numerosas variables, o un servidor gestionado por mas de una persona.

  - **Tipo de dato:** Deberemos indicar el tipo de variable que es, es cierto que si dejamos por defecto el solo reconocerá la variable que es pero esto ralentizara el sistema.

  - **Acceso de los clientes:** Por defecto los clientes del servidor tendrán acceso a lectura y escritura. Es muy importante configurar adecuadamente este permiso ya que otros programas podrían cambiar el valor produciendo errores desconocidos forzando situaciones que en la realidad seria imposible que se dieran. Por otro lado puede ser de interés que esta variable sea manipularle pudiendo por ejemplo generar un paro de varias maquinas enlazadas entre si con cualquier paro de ellas.

  - **Comprobación de la generación de variable:** Esta configuración la podemos comprobar en la figura 11.34

- **Comprobamos la comunicación:** Para comprobar la comunicación del autómata deberemos pulsar en el observador del servidor, el QUICK CLIENT, esto lo podemos observar en la figura 11.35, deberemos pulsar sobre el canal deseado SIEMENS S-1200, en citando anteriormente en la Párrafo 11.2.2. Aquí aparecerá un GOOD si la comunicación es correcta. En este punto podemos comprobar si el es todo correcto forzando salidas del autómata, viendo así como cambia el QUICK CLIENT, como se puede observar en figura 11.36

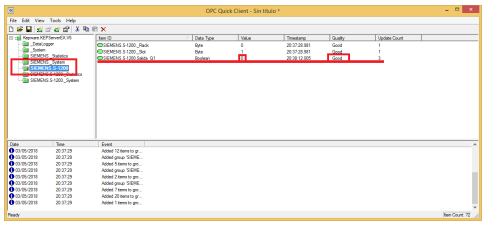

Figura 11.35: Monitorización de las variables

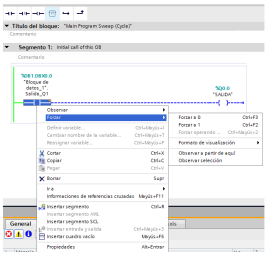

Figura 11.36: Forzar valores de la variable

Pasos a seguir para clientes de Windows

El protocolo de comunicación OPC es nativo del sistema operativo en base windows, pero no exclusivo de este, por eso muchos de sus programas son susceptibles de conectarse con el servidor, siempre que nosotros lo autoricemos. En el siguiente apartado vamos a enseñar a crear el canal de comunicación para posteriormente declarar variables. habra que tener en cuenta que las variables no son documentos enteros, sino que como pasaba en la subsección 11.2.2 seran partes de estos.

Canal de comunicación

- **Cambiamos la configuración del OPC:** En la barra de tareas, buscaremos el icono del NI OPC SERVER, y pulsaremos sobre el con el botón derecho. Se desplegara una lista de opciones en la que deberemos buscar ”Setting” para abrirla. Una vez abierta encontraremos numerosas pestañas en la parte superior, deberemos ir a la que pone ”Runtime process” para cambiar el modo a interactivo (”select mode interactive”) Esto lo podemos observar en la figura 11.37.

- **Creamos un canal de comunicación:** Tal y como se explico en el Párrafo 11.2.2 se ha de crear un canal que enlace el servidor con el elemento que deseamos, por ello pulsaremos sobre ”CREATE NEW COMMUNICATION CHANNEL” tal y como se mostró en la figura 11.13 y le daremos un nombre representativo, tal y como se mostró en la figura 11.14, pero esta vez la llamaremos ”Windows”. Esto lo podemos observar en la figura 11.38

- **Seleccionamos el protocolo de comunicación:** de manera análoga al apartado anterior, debemos seleccionar la naturaleza de las variables, en esta ocasión se seleccionara ”DDE Client”, que es el protocolo interno de OPC. Esto lo podemos observar en la figura 11.39.

- **Optimizamos los parámetros de escritura:** Tal y como se realizo con el canal de comunicacion del automata seleccionaremos los parametros de escritura, para esta prueba los dejaremos del mismo modo que en la figura 11.17

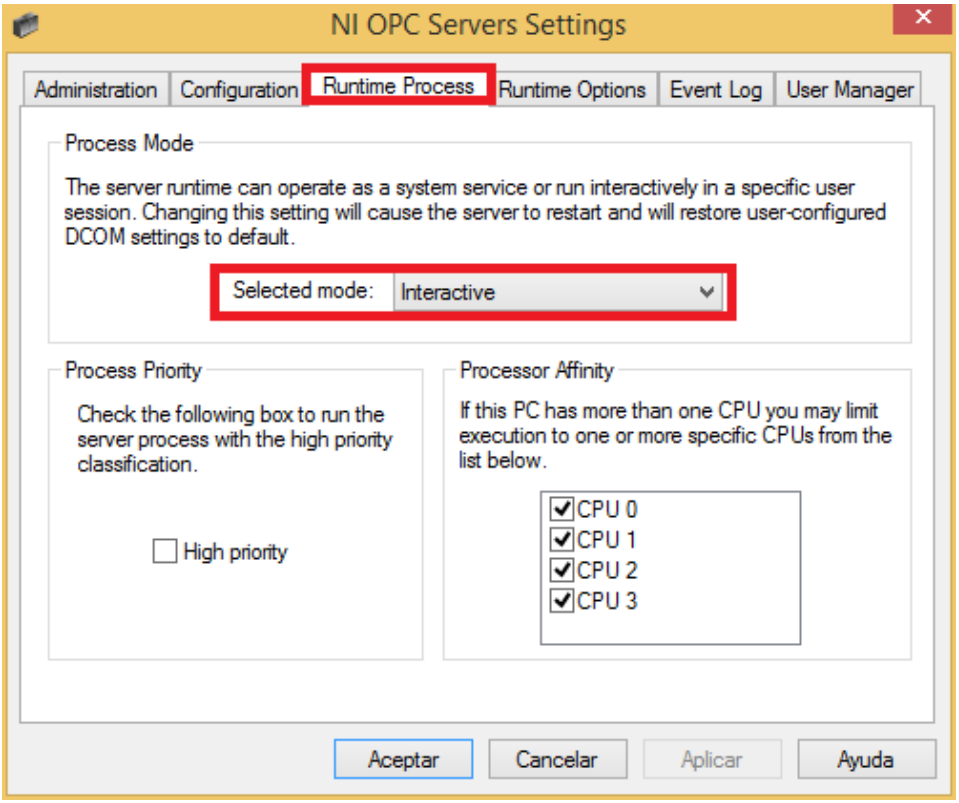

Figura 11.37: Cambiamos a modo interactivo el programa

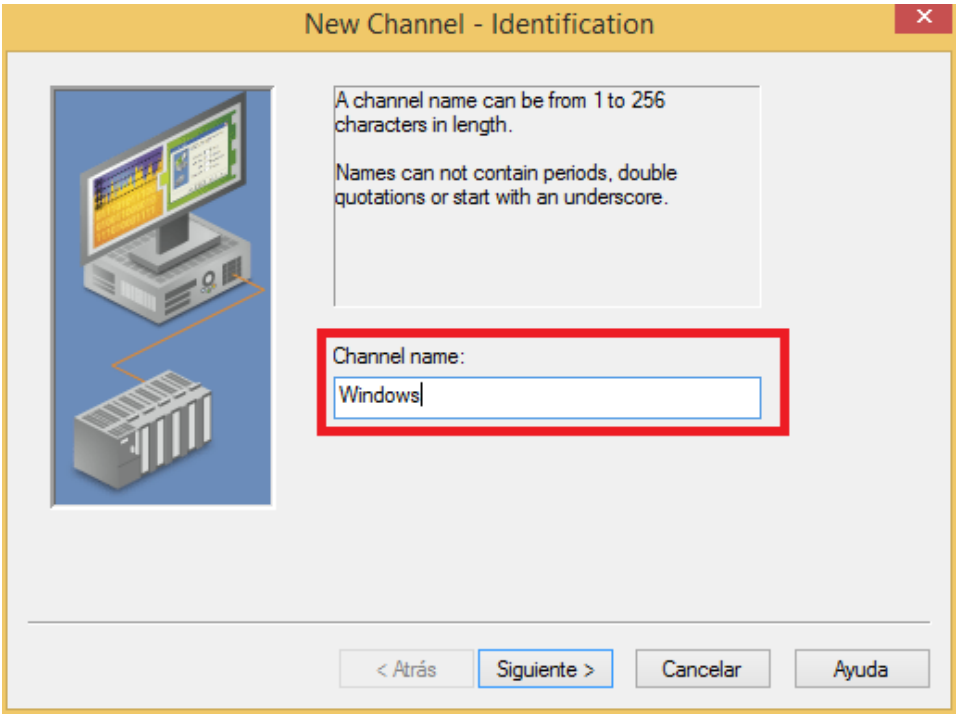

Figura 11.38: Nombramos el canal de comunicación para conectar con Windows

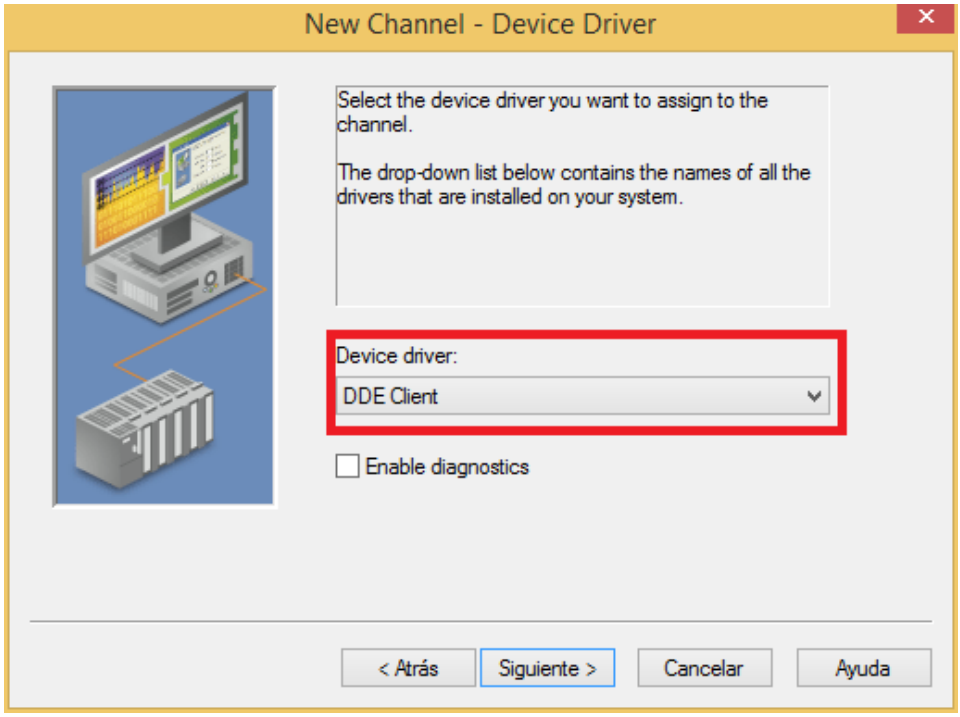

Figura 11.39: Selección del protocolo de comunicación

- **Normalización:** Elegimos los valores para normalizar los valores, por defecto esta configurado el no remplazar los elementos, pero para este ejemplo seleccionaremos remplazar por cero. Tal y como se pudo observar en figura 11.18

- **Comprobamos que los datos son correctos:** Como en el apartado anterior, saldra una ventana con la configuración seleccionada para aceptar la configuracion o declinarla. Esto lo podemos observar en la figura 11.40

Añadir driver

- **Creamos el driver:** de manera análoga a la descrita en Párrafo 11.2.2, deberemos hacer click sobre alguna de las dos opciones que se muestran en la figura 11.41, observando que estamos en el canal de comunicación adecuado, y no en el erróneo.

- **Denominamos el driver:** Como en el apartado anterior, habrá que poner el nombre representativo. En este caso como solo conectaremos con el excel, lo llamaremos de ese modo. Esto lo podemos observar en figura 11.42

- **Configuramos el modo de escaneo:** tras seguir los pasos hasta este punto debemos configurar el modo de escaneo del cliente OPC, en este caso dejaremos el que viene por defecto, el modo de escaneo especifico del cliente "Respect client specified scan rate'. Esta configuración la podemos observar en la figura 11.43

- **Configuramos los tiempos:** En este apartado se configuraran los tiempos de respuesta de los clientes, como podemos observar con lo declarado en Párrafo 11.2.2 los tiempos que propone el servidor son mucho mas rapido, ya que los clientes DDE son comunes al ordenador del servidor. Por ese motivo no ha habido que configurar la tarjeta de red.

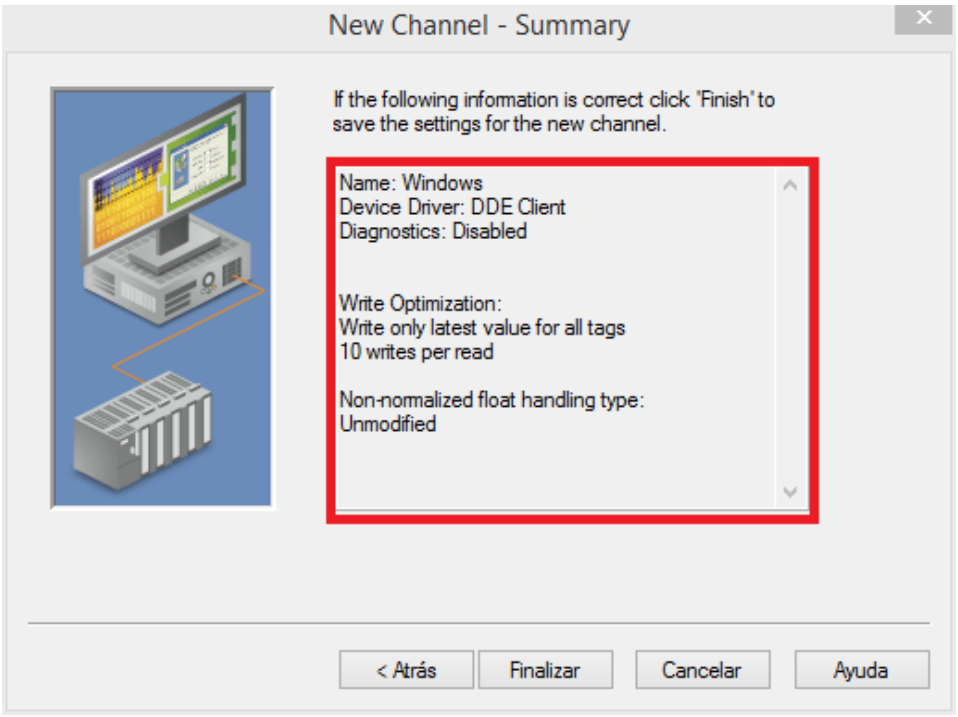

Figura 11.40: Selección del protocolo de comunicación

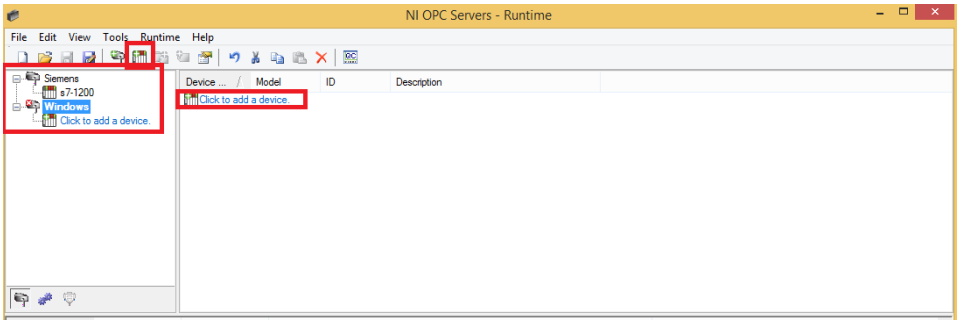

Figura 11.41: Selección del protocolo de comunicación

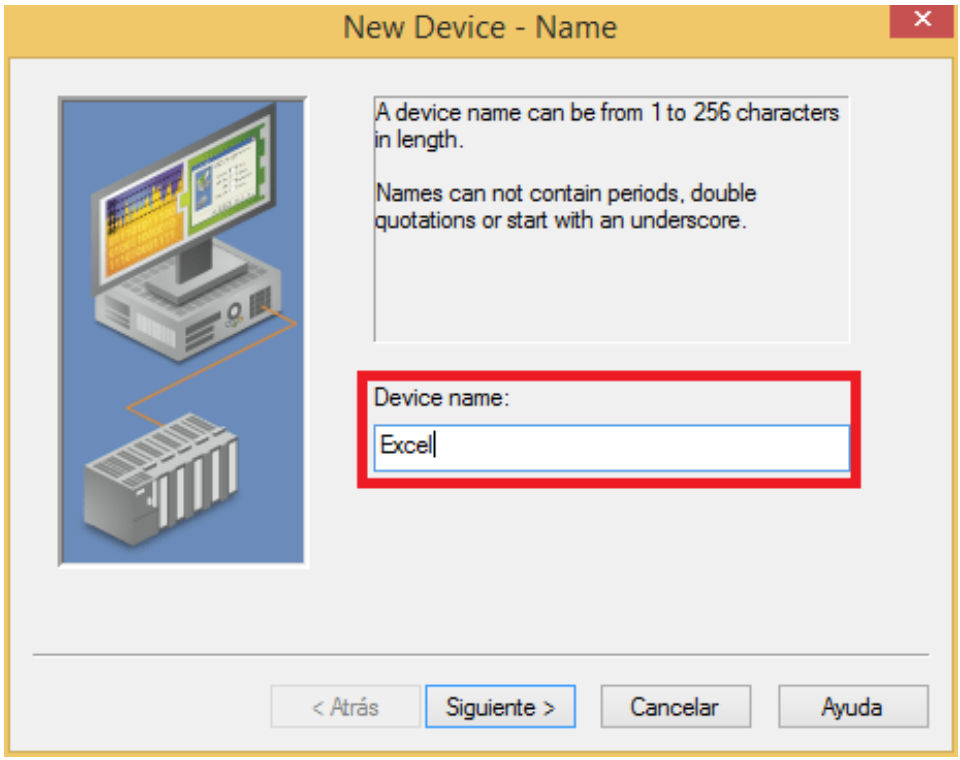

Figura 11.42: Selección del protocolo de comunicación

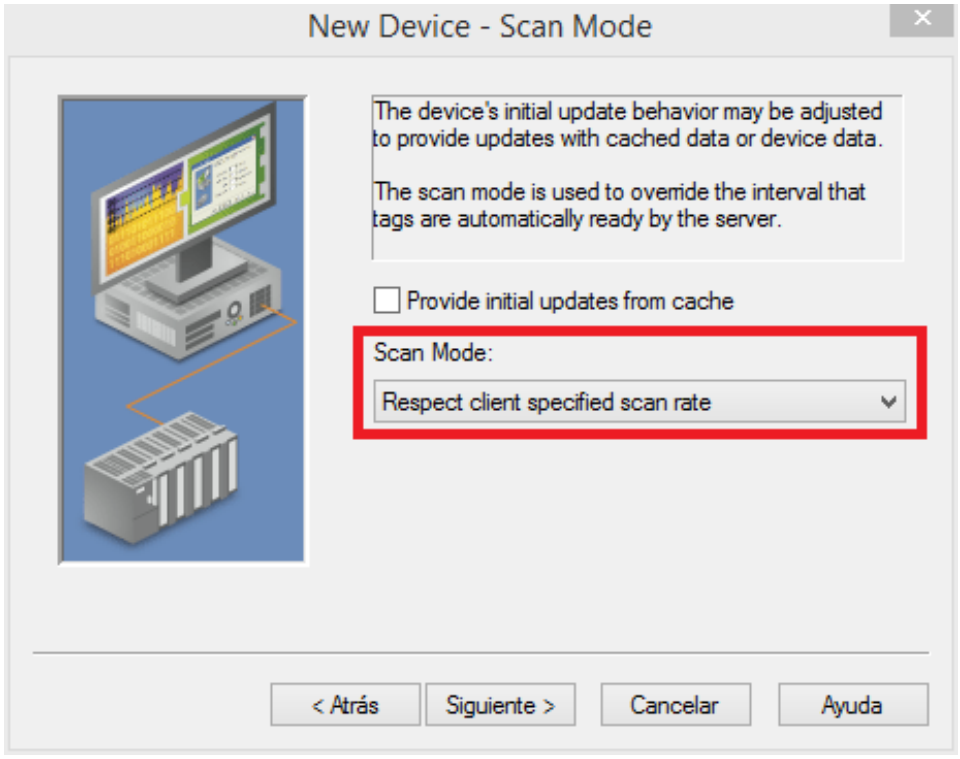

Figura 11.43: Selección del protocolo de comunicación

Para esta ocasión mantendremos los tiempos propuestos. Esto lo podemos observar en figura 11.44

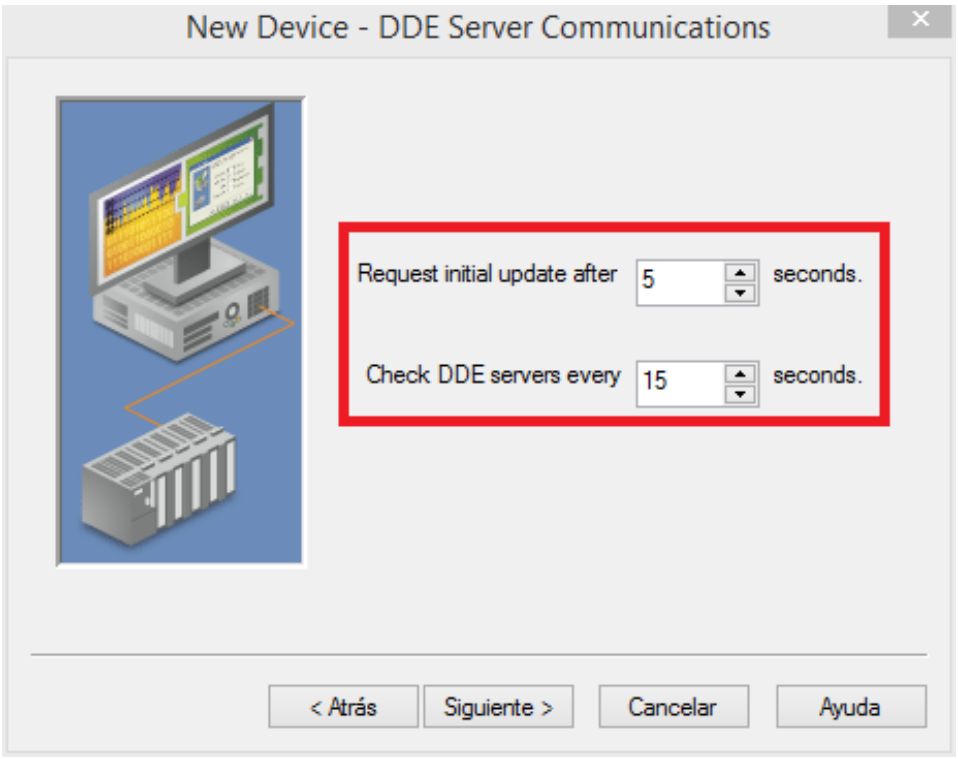

Figura 11.44: Selección del protocolo de comunicación

- **Comprobamos la configuracioó:** Si todo ha ido bien deberás tener un resumen de la configuración identico al mostrado en la figura 11.45

Variables

Si todo ha ido bien hasta este punto deberás encontrarte con que la pequeña cruz que salia alado del nombre del canal de comunicación en la figura 11.41 ha desaparecido, y que donde antes teníamos ”Click to add a driver” ahora tenemos el nombre de excel. Para añadir variables debemos seguir el siguiente proceso:

- **Comenzamos el proceso:** Como en el Párrafo 11.2.2 daremos click sobre cualquiera de las dos opciones para añadir una nueva variable, pero en este caso en el canal de comunicacion que hemos creado. Esto lo podemos observar en figura 11.46

- **Denominamos la variable:** En este caso no tendran nombre del programa del que vienen, por eso recomiendo utilizar las coordenadas del excel de su posicion concreta. En este caso vamos a declarar los valores de la celda A2. Esto lo podemos obesrbar en la figura 11.47

- **Dirección:** En este caso para añadir cualquier tipo de variable pulsaremos sobre el símbolo de ayuda del cuadro de dirección, desplegándose una ventana con una única opción, pulsaremos sobre ella y aparecerá escrito en el recuadro ”DDE Service Name|Topic Name!Item Name”. Esta es la ruta genérica para todos los DDE client de windows. Debemos sustituir ”DDE Service Name” por ”Excel”, sustituiremos ”Topic Name” por el nombre de

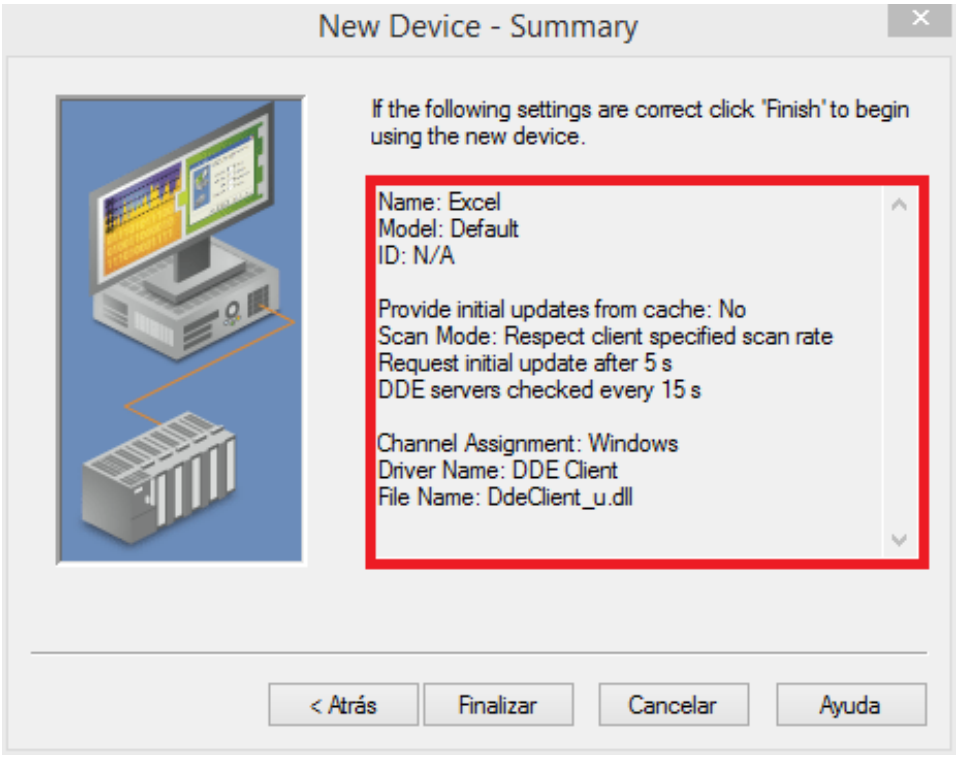

Figura 11.45: Selección del protocolo de comunicación

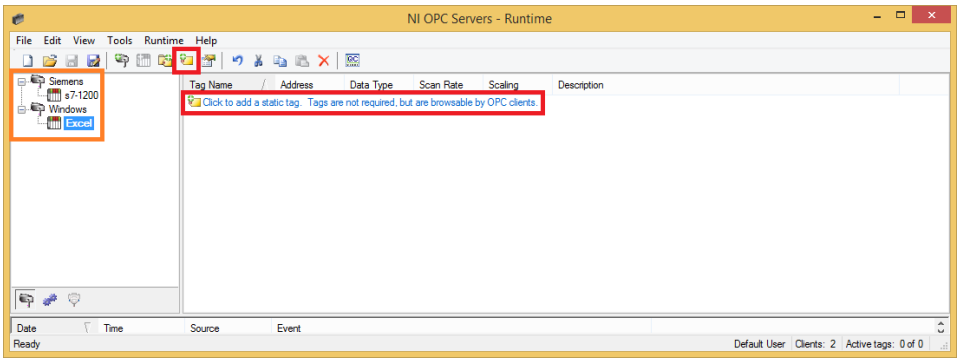

Figura 11.46: Selección del protocolo de comunicación

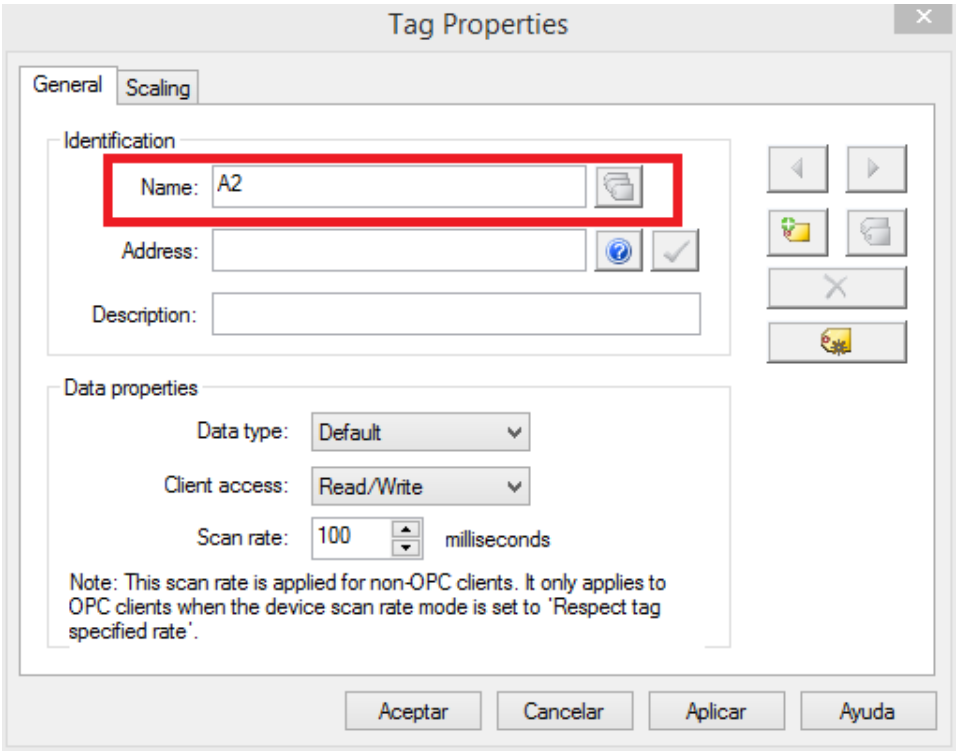

Figura 11.47: Selección del protocolo de comunicación

la hoja de excel que queramos la celda, en este caso ”Hoja1” siempre que sea el idioma principal en castellano. y sustituiremos ”Item Name” por ”F1C2” siendo ”F” la fila y ”C” la columna. Se deberán mantener los símbolos como se observa en la figura 11.48

- **Tipo de dato:** Tras escribir la dirección debemos configurar el tipo de dato, por defecto al pulsar sobre el ”check” se configura Dword como predeterminado. En función de lo que queramos en nuestro servidor se puede cambiar el tipo de dato.

- **Comprobación:** Abriremos un Excel nuevo y en la celda A2 escribimos un valor, numérico comprendido entre 0 y 4294967295, mediante el OPC Quick Client y seleccionando el canal adecuado veremos como cambia este numero. Esta celda puede cambiar mediante una formula o mediante una escritura manual, ya que el servidor leerá el valor independientemente de como se obtenga. Esto lo podemos observar en la figura 11.49

Cosas a tener en cuenta:

Unas de las cosas mas importantes a tener en cuenta es el tipo de dato que se va a enviar, ya que se trataran de forma distinta. Las opciones que tenemos para declarar en el servidor son limitadas, aunque suficientes para todos los datos. Deberemos distinguir entre los siguientes tipos de datos:

- Boolean: 1 bit de posibilidades, 1 ó 0

- Char: Un valor con signo de 8 bit

- Byte: Un valor sin signo de 8 bit

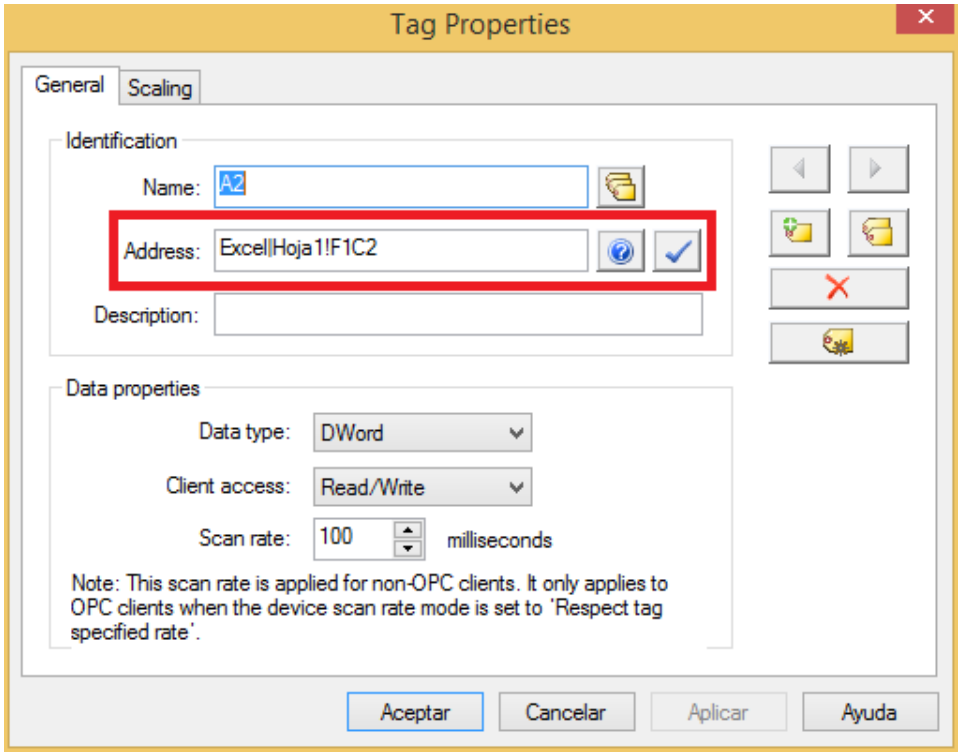

Figura 11.48: Selección del protocolo de comunicación

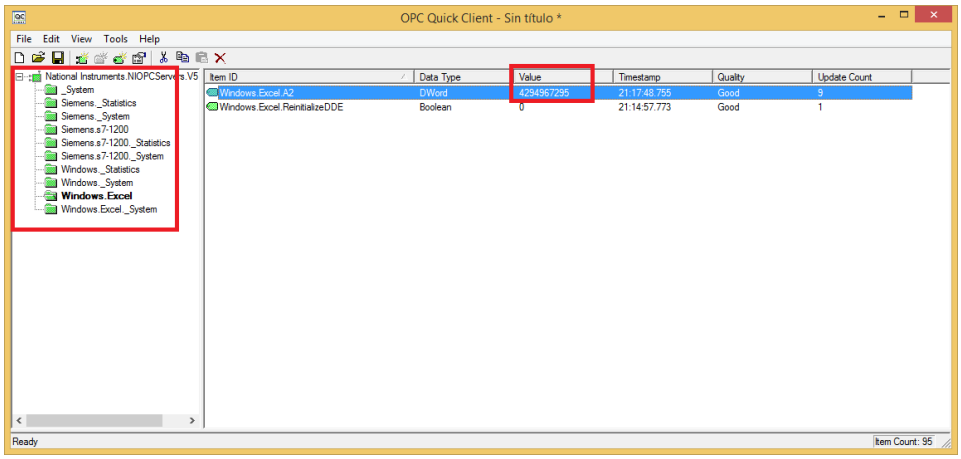

Figura 11.49: Selección del protocolo de comunicación

- Short: un valor con signo de 16 bit, el bit 0 es el bajo, el bit 14 el alto, y el bit 15 el signo
- Word: un valor sin signo de 16 bit, el bit 0 es el bajo, y el bit 15 el alto
- Long: un valor con signo de 32 bit, el bit 0 es el bajo, el bit 30 el alto, y el bit 31 el signo
- DWord: un valor sin signo de 32 bit, el bit 0 es el bajo, y el bit 15 el alto
- LLong: No aceptado en S7-1200
- Qword: No aceptado en S7-1200
- Float: No aceptado en S7-1200
- Double: No aceptado en S7-1200
- String: Un valor de la tabla ASCII
- BCD: Un paquete de 2 byte. Comprende valores entre 0 - 9999
- LBCD: Un paquete de 4 byte. Comprende valores entre 0 - 99999999
- Date: 64 bit con coma flotante

En función de las direcciones físicas que tienen nuestros tipos de datos del autómata podremos darles un tratamiento u otro, teniendo que tener clara la nomenclatura mostrada en la tabla 11.4

I o E	Input	Cualquier tipo de dato
Q o A	Outputs	Cualquier tipo de dato
PI o PE	peripheral Inputs	Cualquier tipo de dato
PQ o PA	peripheral Outputs	Cualquier tipo de dato
M o F	Flag Memory	Cualquier tipo de dato
DB	Data Bloks	Cualquier tipo de dato
T	Timers (T0 - T65535)	Dword o Long
C o Z	Counters (C0 - C65535 o Z0 - Z65535)	Word Short

Tabla 11.4: Tipo de dato según la dirección física del autómata

11.2.3 LabVIEW

En este punto vamos a pasar a explicar como podemos configurar el cliente del servidor OPC, para ello necesitaremos contar con el programa LabVIEW, en nuestro caso la versión 2017, la cual necesita unos requisitos mínimos como los mostrados en la tabla 11.5. Como podemos observar los requisitos son tan bajos que podria instalarese en casi cualquier ordenador, pero esto no incluye los paquetes necesarios para el servidor OPC. Necesitaremos tener instalados el Paquete DSC (Data Loging and supervisory control) para poder configurar servidores OPC Como se ha citado anteriormente. Los requisitos recomendados desde NI son los que aparecen en la tabla 11.5.

Para conocer mas requisitos puede recurrir a la página oficial del fabricante. A continuación pasaremos a describir los pasos a seguir para controlar la variable del servidor.

Requisitos	Minimos	Recomendados
Procesador	Pentium 4M/Celeron 866 o superior	Pentium 4G1
RAM	256MB o más	1 GB
Disco duro	620 Mb o más	5 GB
Pantalla	1024 x 768 pixels	
Sistema operativo	32 bit	

Tabla 11.5: Requisitos del programa LabVIEW 2017

Pasos a seguir

En esta sección pasaremos a describir los pasos seguidos para conmutar mediante el programa la salida física del autómata:

- **Creamos un nuevo proyecto:** Para ello iniciaremos el programa y clicaremos en en boton de ”Create Project tal y como se muestra en la figura 11.50

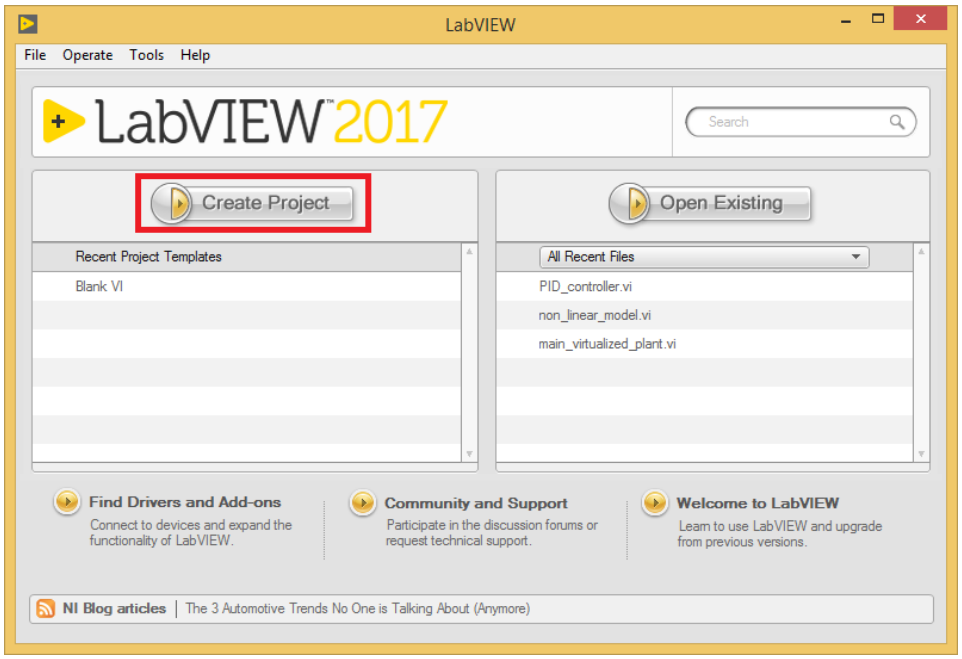

Figura 11.50: Crear nuevo proyecto en el programa LabVIEW 2017

- **Seleccionamos el tipo de proyecto:** Existen diversos tipos de proyectos en función de lo que queramos realizar con el programa, en este acaso como se trata de una simple prueba vamos a seleccionar un proyecto en blanco. En este caso se abrirá una ventana, la cual podemos observar en la figura 11.51.

- **Creamos el IO SERVER:** tras haber abierto la configuración de un nuevo proyecto en blanco, debemos hacer clic con el botón secundario en ”My Computer y seleccionaremos NEW, y posteriormente ”IO SERVER. Esto lo podemos observar en la figura 11.52

- **Cliente OPC:** se abrirar un cuadro de dialogo en el que seleccionaremos ”OPC Client. Esto lo podemos observar en la figura 11.53

- **Seleccionamos el servidor OPC:** En el cuadro de dialogo que aparece debemos seleccionar cual es el servidor OPC, este servidor es el configurado en la subsección 11.2.2, y

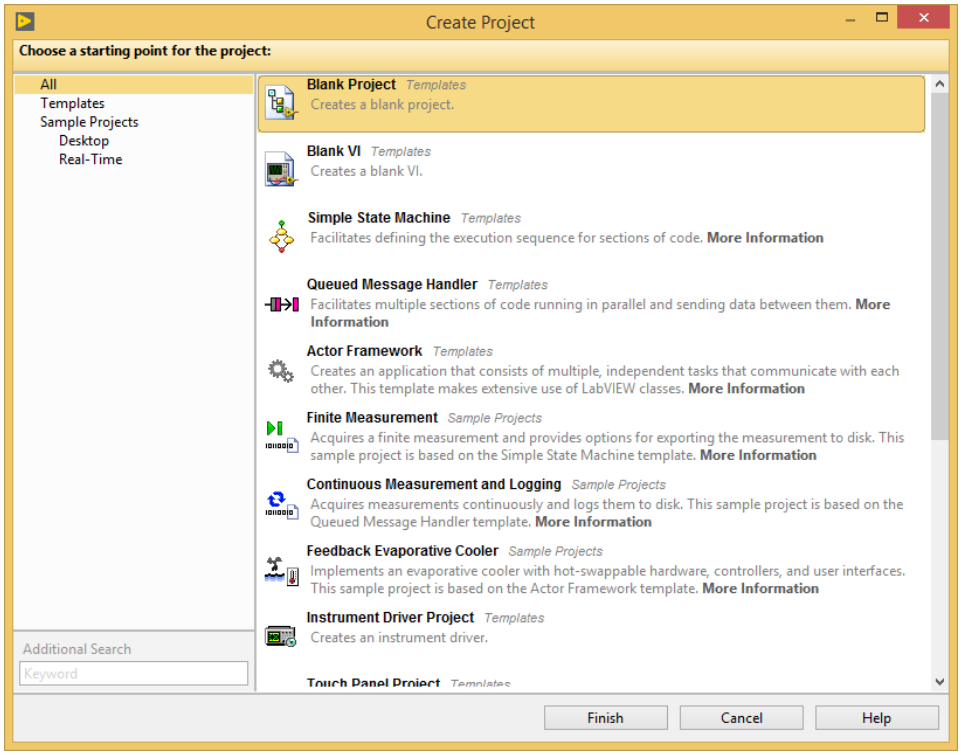

Figura 11.51: Crear un proyecto en blanco

por eso seleccionamos la opción ”National Instruments.NIOPCServers.V5, como podemos observar en la figura 11.54, tras esto saldra un aviso que aceptaremos. Este aviso se puede ver en figura 11.55

- **Agregamos la variable:** En este punto deberíamos encontrarnos con el servidor dentro de la librería que hemos creado, y sera en este punto donde creemos la Variable. ANtes de este momento deberemos tener todo como se muestra en la figura 11.56. Tras esto pulsaremos sobre la librería con el botón derecho y haremos click en ”NEW, para pulsar posteriormente en variable, de este modo se abrirá una nueva ventana tal y como la que podemos observar en figura 11.57.

- **Seleccion de variable:** en este punto se ha debido abrir una ventana como la que se muestra en la figura 11.58, en la que deberemos activar la opción indicada en esta para posteriormente seleccionar ”BROWSE y que se abra una nueva ventana como la que se muestra en figura 11.59. En esta encontraremos todos los canales creados en el servidor, con los autómatas, o programas que tengan las variables, y las variables que se hallan declarado hasta este momento. En este punto seleccionaremos la variable que deseamos controlar, la que declaramos en subsección 11.2.2 y regresaremos al punto anterior.

- **Configuramos parámetros de la variable para LabVIEW:** en este punto nos encontraremos con la misma ventana mostrada en la figura 11.58, pero sin el error que aparece en la parte inferior, debido a que en este momento conoce la variable que debe crear. Del mismo modo a cambiado el tipo de variable de ”DOOBLE a ”BOOLEAN. Tras esto solo quedaria configurar el nombre con el que queramos que se muestre, y si es de lectura y escritura o solo de lectura. En este caso queremos que el programa cambie el valor y conmute la salida fisica del autómata, por eso le otorgaremos lectura y escritura. La configuración ha de quedar tal y como se muestra en la figura 11.60. En este momento ya tenemos nuestra variable creada y lo único que debemos hacer es vincularla a un interruptor.

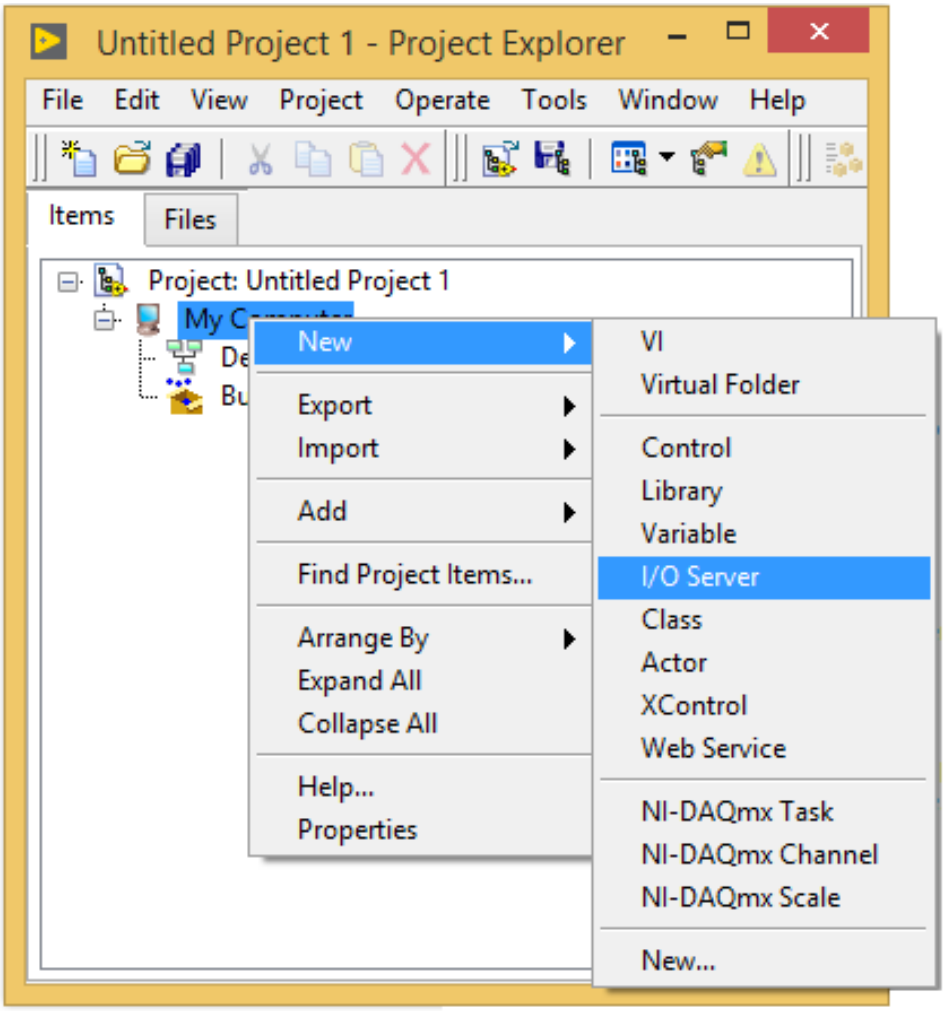

Figura 11.52: Crear un IO SERVER

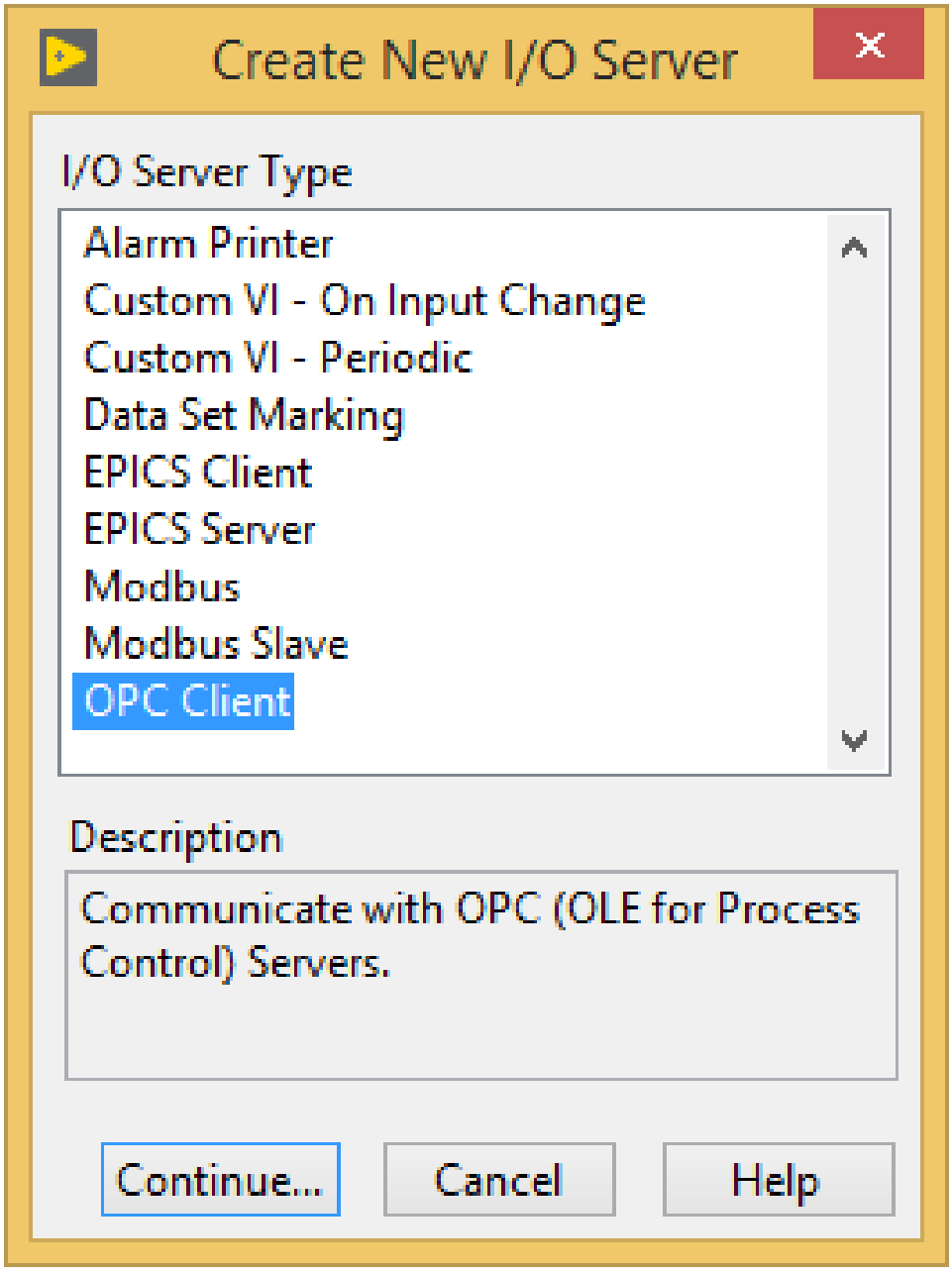

Figura 11.53: Seleccionamos OPC Client

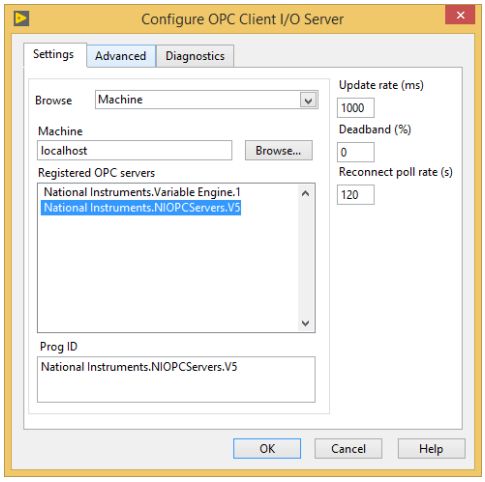

Figura 11.54: Selección del gestor del servidor OPC

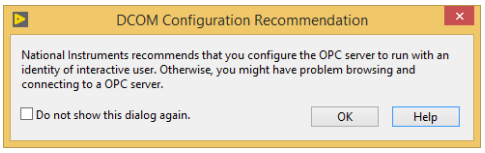

Figura 11.55: Aviso tras crear el servidor OPC client

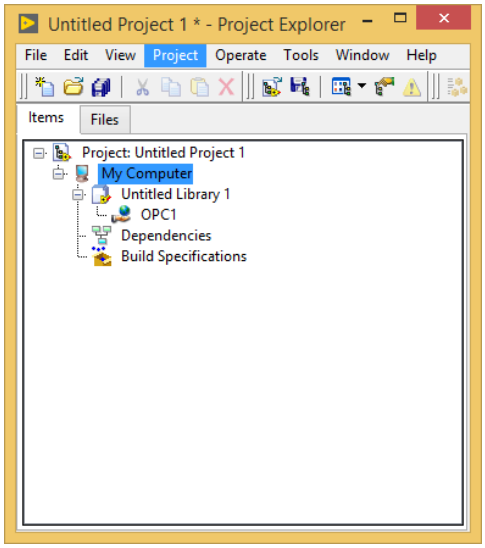

Figura 11.56: Servidor OPC dentro de la variable

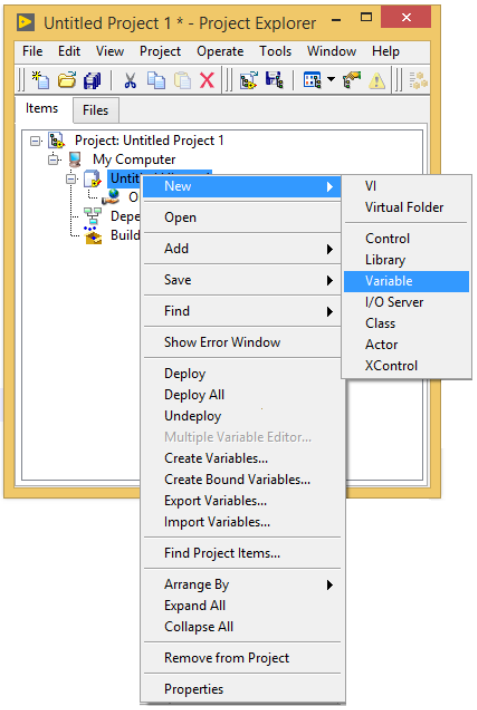

Figura 11.57: Ventana para dar de alta variables

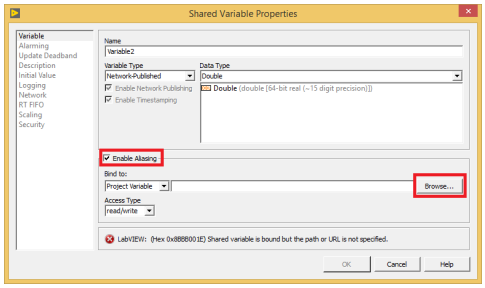

Figura 11.58: Ventana para declarar variables

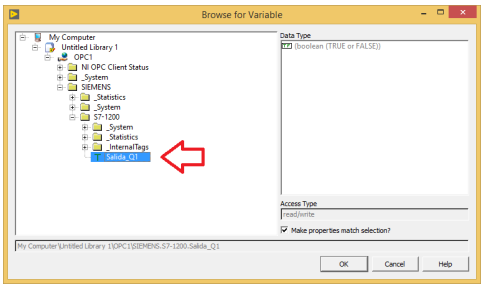

Figura 11.59: Seleccion de la ruta de la variable

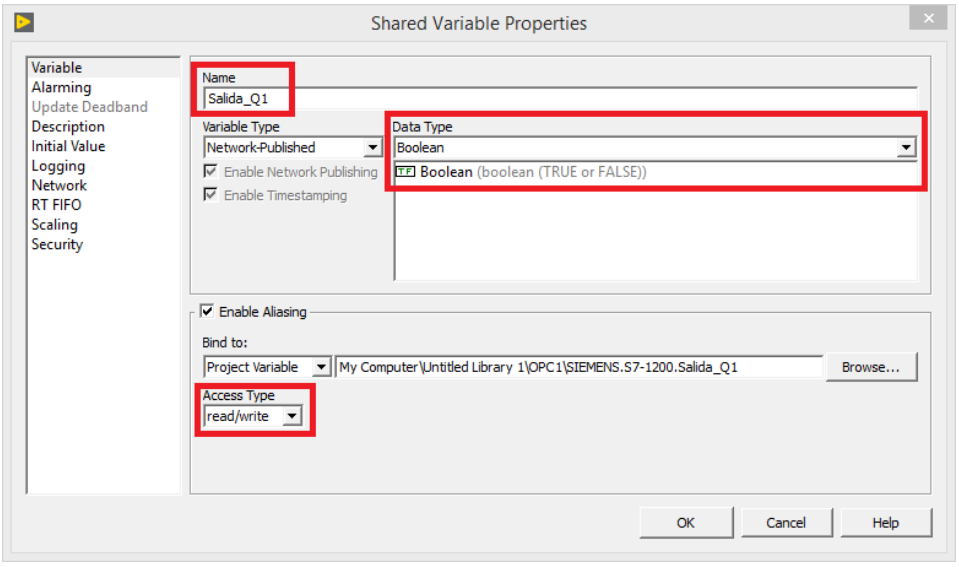

Figura 11.60: Configuración de la variable del servidor para lectura y escritura

- **Creamos un nuevo VI:** Debemos crear el espacio de trabajo del VI para asignar la variable al elemento que queramos, en este caso a un interruptor. Para ello es importante que el VI de trabajo se encuentre fuera de la librería, ya que si no estariamos creando funciones. Para ello realizaremos los mismo pasos que en la figura 11.52, pero en lugar de hacer clic sobre I/O Server, lo haremos sobre VI, obteniendo un resultado como el que se muestra en la figura 11.61

- **Creación de un botón:** Tras crear el VI, se han abierto dos nuevas ventanas, uno sera el panel y lo detectaremos por ser de color grisáceo, y otro es el simbólico de color blanco. En el panel pulsaremos con el botón derecho y en el cuadro de ”BOOLEAN encontraremos ”PUSH BUTTON, esto lo podemos observar en la figura 11.62. Tras realizar esto veremos que aparecen dos nuevos símbolos en el sistema, uno en cada ventana. Esto lo podemos observar en la figura 11.63

- **Vinculamos el boton con la variable:** Para realizar esta tarea pasamos a pulsar con el botón derecho sobre el botón en el panel simbólico, tras esto pulsaremos sobre ”Properties, e iremos a la pestaña ”Data Binding, tras esto en el desplegable seleccionaremos ”Shared Variable Enfine (NI-PSP), eligioendo el tipo de acceso de lectura y escritura y pulsando sobre ”Browse” y seleccionaremos del arbol la variable deseada. Esto lo podemos observar en la figura 11.64, en este momento el icono del interruptor habra cambiado teniendo un pequeño triangulo debajo del nombre, indicando que es esterna la variable. En estos momentos estan vinvuladas, pudiendo conmutar la variable siempre y cuando la CPU del auntomata y el NI esten en modo RUN, y el servidor abierto. Para comprobarlo realizaremos la puesta en marcha y desde el ”Quick CLIENT del servidor observaremos como conmuta el estado de la variable, de igual modo que se escuchara como conmuta la salida del automata.

- **Introducción de variables de escritura:** Con el método anterior podremos vincular elementos de de Labview como interruptores o indicadores, pero si los datos que queremos necesitan un tratamiento el método anterior no sirve. Si queremos ademas tener mas claras las variables y poder verlas en nuestro VI podemos arrastrarlas de la biblioteca a la pantalla directamente. En ese momento se configuraran como una entrada para VI. Aparecerá una imagen como la mostrada en la figura 11.65

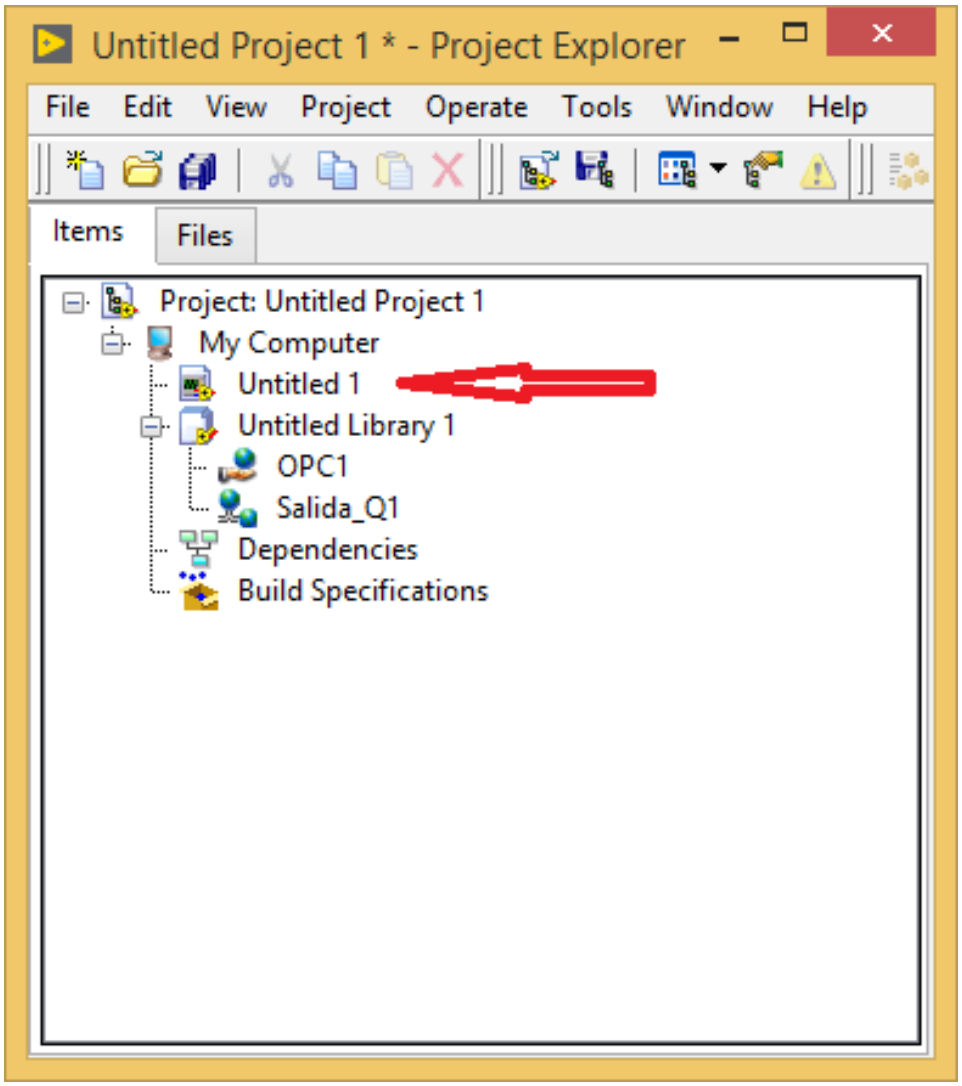

Figura 11.61: Creación de in VI fuera de la biblioteca

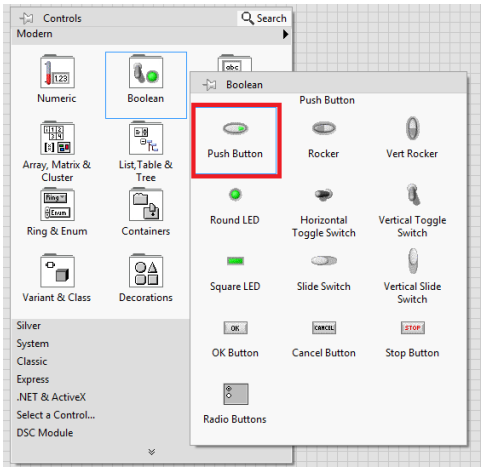

Figura 11.62: Crear un nuevo botón

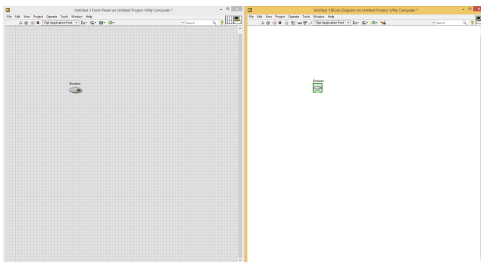

Figura 11.63: Botón creado

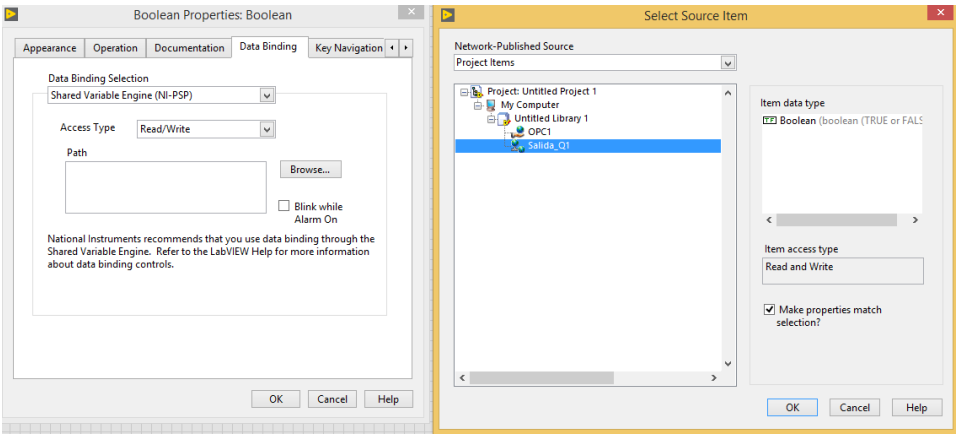

Figura 11.64: Vinculación de la variable

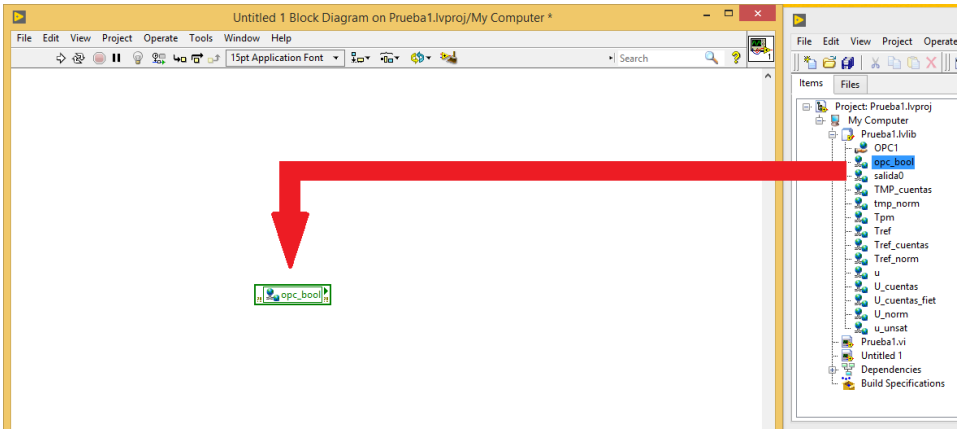

Figura 11.65: Vinculación de la variable

- **Introducir variables de lectura:** Si ahora necesitamos leer la variable para modificar en LabVIEW los datos, deberemos ir a un cable, pulsar con el botón derecho sobre "Select Variable", después sobre "My Computer", se desplegara una paleta, daremos click sobre nuestra librería, y sobre la variable que deseemos insertar. Esta tendrá el aspecto similar a la descrita en la figura 11.65 pero el cableado ira en sentido contrario. Todo esto lo podemos observar en la figura 11.66 obteniendo como resultado lo mostrado en la figura 11.67

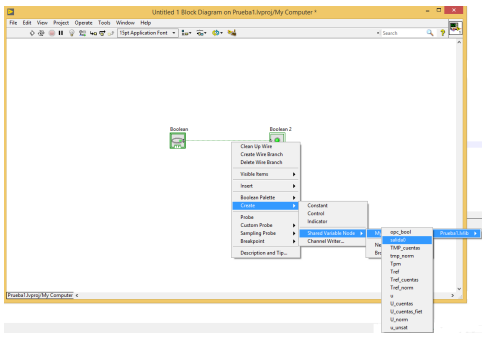

Figura 11.66: Vinculación de variable para lectura

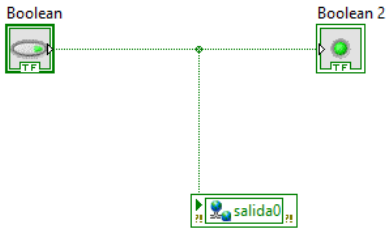

Figura 11.67: Icono de variable para lectura en un VI de LabVIEW

11.2.4 Excel

El Excel es el programa ofimático de hoja de calculo mas común en el mundo. Viene incluido en el paquete ofimático OFFICE, de Windows. Este programa es una potente hoja de calculo con la que muchas empresas gestionan las ventas, cuentas de la empresa, datos de los clientes, pedidos... con la que gracias a los servidores OPC, podremos obtener datos reales del estado de la producción, pudiendo dar datos reales a nuestros clientes en todo momento, e incluso tratar estos datos de manera normal a la hoja de calculo pudiendo realizar estudios y mejorar nuestra competitividad ante otras empresas. Gracias al servidor podremos conocer el estado de las maquinas de la industria, e incluso poder realizar marchas y paros desde la misma. Este programa no consume grandes recursos informáticos, esto lo podemos observar en la tabla 11.6

Procesador	32 o 64 bits a 1 GHz
RAM	2 GB o más
Disco duro	3 GB o más
Pantalla	Resolución de 1024x678
Sistema operativo	Windows 7 SP1 o superior

Tabla 11.6: Requisitos mínimos del office 2016