Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Electrónica Industrial y Automática

DESARROLLO DE UNA PLATAFORMA PARA LA REGULACIÓN DE PROCESOS MEDIANTE CONTROLADORES INDUSTRIALES

Alumna: Andrea Sophia Sánchez Ramírez

Tutor: Juan Albino Méndez Pérez

Julio 2018

ÍNDICE

Índice de figuras

0. ASPECTOS GENERALES

0.1. Abstract

The main objective of this project is the development of a platform for the monitoring and control of a plant using an industrial controller. To control the level of a tank filling system a PID controller will be implemented.

Once the controller has been implemented and the SCADA system has been designed for the monitoring of the plant, the effect of the PID parameters on the control system will be studied.

On the other hand, the main interest of this assignment is to apply the theoretical and practical knowledge acquired during the degree in the field of automation and industrial control, expanding them by developing a more extensive project to those done during the degree.

0.2. Nomenclatura y Abreviaciones

- SCADA: Supervisión, Control y Adquisición de Datos. Tipo de software para ordenadores que permite el control y la supervisión de procesos industriales a distancia y en tiempo real.

- HMI: Human Machine Interface o Interfaz de usuario. Medio a través del cual el usuario puede comunicarse con una máquina.

- MES: Manufacturing Execution System. Sistema elaborado para la gestión de todos los procesos de producción de una industria de una manera más eficiente. Este sistema conecta un sistema de gestión empresarial (ERP) con la planta de producción.

- ERP: Enterprise Resource Planning o Sistemas de Planificación de Recursos Empresariales. Sistema encargado de la planificación, organización y procesos de una industria, incluyendo la logística, distribución e inventario de los productos, entre otros aspectos.

- PID: Controlador Proporcional-Integral-Derivativo. Mecanismo de control usado en sistemas de control industrial que calcula el error entre un valor medido y un valor deseado.

- SP: Setpoint o consigna. En control, se refiere al objetivo o valor de una variable que se desea alcanzar en un proceso de control.

- OPC: OLE for Process Control. Estándar de comunicación en el campo de control y supervisión de procesos industriales, que ofrece una interfaz común para la comunicación entre componentes de software individual. Esta comunicación se realiza mediante una arquitectura Cliente-servidor.

- OLE: Object Linking and Embedding o Incrustación y enlazado de objetos. Sistema que permite el manejo de documentos compuestos y la transferencia de datos entre aplicaciones diferentes.

- FTP: File Transfer Protocol. Protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

- Arquitectura cliente-servidor: Modelo de diseño software en el que las tareas se reparten entre los servidores y clientes. Un cliente realiza peticiones a otro programa y el servidor es quien le da la respuesta.

- ModBus: Protocolo de comunicaciones basado en la arquitectura maestro-esclavo (RTU) o cliente-servidor (TCP/IP).

- Válvula de control: válvula usada para controlar el flujo de un fluido, comportándose como un orificio de área continuamente variable.

- CPU: Unidad de procesamiento central. Dispositivo hardware situado dentro de un ordenador u otros dispositivos programables, que interpreta las instrucciones de un programa informático.

- Servo válvula: Válvula de tipo neumática o hidráulica que controla presiones o caudales.

- IDE: Entorno de desarrollo integrado. Aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

1. INTRODUCCIÓN

1.1. Antecedentes

Hoy en día la automatización se ha convertido en uno de los aspectos más importantes para las empresas y sus trabajadores, ya que son muchos los beneficios que esta nos ofrece. Por una parte, cabe destacar el aumento de la productividad que aporta este sistema, ya que permite realizar ciclos de producción más rápidos mejorando la calidad del producto. También, el rendimiento de producción mejora considerablemente con la automatización, ya que permite eliminar los errores humanos, aumentando de esta forma la precisión y eficiencia de producción. Por otro lado, dado que la seguridad operativa es uno de los aspectos más importantes en una empresa, se intentan reducir continuamente los riesgos de accidentes laborales con esta tecnología.

Cada sistema automatizado es diferente dependiendo de las necesidades y demandas de cada empresa, pero todos ellos se organizan de la misma manera, mediante la "Pirámide de automatización". Esta pirámide se puede encontrar en cualquier entorno industrial y recoge cinco niveles de tecnologías diferentes, todas ellas comunicadas entre sí.

- Nivel I o "nivel de campo". En este nivel abarca todos los sensores y actuadores presentes en la industria.

- Nivel II o "nivel de control". Este nivel incluye los sistemas de control presentes en la industria, los cuales se fundamenta en dos aspectos diferentes.

    1. El algoritmo de control a implementar para alcanzar un objetivo. Para esto, se modifican las entradas del sistema con el fin de obtener las salidas o estados deseados.
    2. Los dispositivos electrónicos encargados de controlar el sistema estudiado, es decir, los controladores. Estos dispositivos de control se pueden utilizar para diferentes trabajos de medición y regulación, y se componen de una entrada de un sensor, un indicador digital y una salida de regulación.

    Con el fin de alcanzar los objetivos del sistema, todo sistema de control debe ser eficiente y estable ante posibles perturbaciones o errores presentes en el proceso.

- Nivel III o "nivel de supervisión". Este nivel corresponde a los sistemas de supervisión, control y adquisición de datos, SCADA. Estos sistemas software permiten controlar y supervisar en tiempo real cualquier proceso industrial a distancia desde un ordenador. Estos sistemas también se pueden llevar a cabo mediante una interfaz de usuario, HMI, que permite la interacción entre el usuario y una máquina.

- Nivel IV o "nivel de planificación" y nivel V o "nivel de gestión". Estos dos últimos niveles superiores corresponden a los sistemas de información. Un sistema de información es un conjunto de elementos encargados de procesar y administrar datos que posteriormente se utilizarán para alcanzar un objetivo. Este conjunto se compone de personas, procesos y equipos de tecnologías de la información.

  El funcionamiento de estos sistemas es el siguiente. En primer lugar, los elementos hardware procesan y almacenan los datos recopilados. Posteriormente, son extraídos con los sistemas software, y, por último, esta información es compartida entre los distintos dispositivos mediante la red. Existen diversos sistemas de información, dependiendo del campo de trabajo donde se lleve a cabo y los objetivos a alcanzar.

  En el nivel IV se encuentran los sistemas de ejecución de la producción, encargados del mantenimiento y el seguimiento de inventarios, entre otras cosas, MES. Y el nivel V lo componen los sistemas de planificación de recursos empresariales, ERP. Estos son los sistemas de información gerencial, es decir, los encargados de la producción, ventas, compras, logística, etc.

En la actualidad, existe un nuevo concepto de organización industrial que abarca todo el proceso de automatización llamado Industria 4.0 o también La cuarta revolución industrial, que se pretende implantar en las fábricas con el objetivo de digitalizarlas. Este concepto de organización industrial trata de hacer las fábricas cada vez más inteligentes mediante tecnologías de procesamiento de datos, software inteligente y sensores, con el fin de aumentar la capacidad de adaptabilidad a las necesidades y a los procesos de producción, así como hacer un uso más eficiente de los recursos. Esta organización se trabaja desde los proveedores hasta los clientes para poder predecir, controlar, planear y producir de forma inteligente.

El presente trabajo se contextualiza en este entorno industrial y persigue afrontar la implementación de un sistema de control industrial incluyendo diferentes tecnologías ubicadas en los distintos niveles de la pirámide de automatización. Se considerará una aplicación sobre una planta de laboratorio sobre la que se pondrán en práctica muchos de los conceptos revisados en esta sección.

1.2. Interés de la propuesta

La propuesta tiene interés por cuanto permite poner en práctica una solución eficiente y avanzada desplegando muchas de las tecnologías sobre las que actualmente se sustentan los sistemas de control en la industria. En este sentido se abordarán cuestiones relacionadas con el control directo, las comunicaciones industriales, la gestión de la información y los sistemas de monitorización y supervisión.

Desde un punto de vista académico, el principal interés de esta propuesta es aplicar los conocimientos teóricos y prácticos adquiridos durante el grado en el ámbito de la automatización y control industrial, así como ampliarlos elaborando un proyecto más extenso y diferente a los realizados durante la carrera.

Por otro lado, se pretende adquirir un concepto de automatización y control industrial más profesional, aprendiendo nuevos métodos de trabajo y completando de esta forma las competencias obtenidas con la titulación.

1.3. Objetivos

El objetivo principal de este proyecto es el estudio de la solución para la monitorización y control de una planta mediante un controlador industrial. La planta elegida es un sistema de tanques en el que se pretende regular el nivel de un tanque actuando sobre una válvula.

La propuesta se basará en el uso de controlador industrial y contempla el desarrollo de un software SCADA para monitorizar el proceso a través del controlador. Se deberá diseñar una interfaz amigable que permita la configuración básica del controlador, la monitorización y el registro en fichero de las variables de interés. Asimismo, se plantea la configuración de un servidor OPC que haga de interfaz entre el dispositivo y el SCADA.

Otro objetivo planteado es el de realizar el estudio del efecto de los parámetros del PID en el rendimiento del sistema de control.

1.4. Metodología y plan de trabajo

Con el objetivo de controlar el sistema de llenado de tanques mediante un controlador industrial y posteriormente monitorizarlo, se plantea el desarrollo de la plataforma hardware/software de la siguiente manera:

1) Estudiar el funcionamiento del sistema de llenado de tanques para determinar las variables que nos interesan controlar. Posteriormente, analizar y llevar a cabo la instalación del controlador a utilizar.

2) Una vez instalado el controlador, se realiza la configuración determinando las entradas y salidas, las constantes del PID y el SP deseado, entre otros aspectos.

3) Luego, es necesario hacer uso de un estándar de comunicación en el campo del control y supervisión de procesos industriales, en este caso es el OPC. Se configura el OPC y se estructura el sistema de monitorización que se va a desarrollar.

4) Finalmente, se lleva a cabo la programación del sistema de control y monitorización de la planta en Matlab para el desarrollo del SCADA.

1.5. Estructura del informe

En el presente informe figura detalladamente la descripción de los elementos empleados en este proyecto, así como el desarrollo del sistema de control de la planta de nivel de tanques llevado a cabo. Para el desarrollo del sistema de control y monitorización de la planta, se ha distribuido el trabajo en tres partes.

Tal y como se recoge en los apartados *2. La automatización y control en la industria* y *3. Control de una planta de nivel de tanques mediante un controlador industrial* del informe, primero se describe la base de este proyecto y las tecnologías aplicadas para su desarrollo. Por otra parte, se expone una descripción de la planta a controlar, los elementos empleados para el desarrollo del mismo y el sistema de monitorización llevado a cabo.

La segunda parte del trabajo lo componen los apartados *4. Solución adoptada* y *5. Programación del sistema de monitorización y control.* En estos, se explica el trabajo realizado para el desarrollo del sistema de control y monitorización de la planta, especificando la configuración del controlador empleado, la configuración del OPC y la estructura del sistema de monitorización. Además, se expone la programación en Matlab llevada a cabo para la monitorización de la planta.

Por último, se describen los resultados de control obtenidos tras la realización del trabajo, analizando la influencia de los parámetros en el control del sistema, y se estudia el presupuesto del proyecto. Esta última parte del proyecto, se refleja en los apartados *6. Resultados de control* y *7. Presupuesto* del informe.

2. LA AUTOMATIZACIÓN Y CONTROL EN LA INDUSTRIA

2.1. Introducción

En la actualidad, la automatización se ha convertido en uno de los aspectos fundamentales para cualquier tipo de industria, ya que son muchas las aplicaciones que esta ofrece. Para llevar a cabo la automatización de un sistema en un entorno industrial, es necesario aplicar un conjunto de tecnologías para poder realizar el control y monitorización de los procesos de forma automática, reduciendo de esta forma la interacción del operador con el proceso. Estas tecnologías se pueden aplicar en una amplia variedad de industrias tales como las textiles, alimentarias, químicas, automovilísticas, petroleras, etc, y todas ellas las recoge la pirámide de la automatización, tal y como se explicó en el apartado *1.1 Antecedentes.* Esta pirámide se divide en cinco niveles diferentes, como se muestra en la *Figura 1*, donde se recogen las tecnologías utilizadas para la automatización del sistema, dependiendo del nivel al que pertenezcan.

1. Nivel de campo
2. Nivel de control
3. Nivel de supervisión
4. Nivel de planificación
5. Nivel de gestión

Figura 1. Niveles de la pirámide de la automatización.

2.2. Nivel de campo

En el nivel I o "nivel de campo" se encuentran todos los sensores y actuadores que componen el sistema. A continuación, se exponen algunos de los sensores y actuadores más comunes en las industrias.

2.2.1 Sensores más comunes en las industrias

Los dispositivos encargados de detectar una determinada magnitud física con el fin de variar una propiedad son los sensores. Estos, se diferencian según los siguientes aspectos:

- Tipo de señal de salida: Existen tres tipos de sensores según el tipo de señal de salida, sensores analógicos, sensores digitales y sensores todo/nada.

- Magnitud física a medir: Según la magnitud física que miden, existen sensores de posición, de fuerza, caudal, movimiento, de humedad, etc.

2.2.2 Actuadores más comunes en las industrias

Por otra parte, los actuadores son los elementos destinados a realizar funciones específicas con el fin de controlar un proceso. Existe una amplia variedad de actuadores, entre los cuales cabe mencionar los más comunes en la industria dependiendo del tipo de energía que emplean:

- Actuadores eléctricos: Actuadores que actúan sólo con energía eléctrica, por ejemplo, un motor de corriente continua.

- Actuadores neumáticos: Estos elementos funcionan convirtiendo la energía del aire comprimido en trabajo mecánico, por ejemplo, un motor neumático.

- Actuadores hidráulicos: Estos actuadores funcionan mediante fluidos a presión, por ejemplo, un motor hidráulico.

2.3. Nivel de control

Este nivel lo compone los sistemas de control empleados para la automatización del sistema. En primer lugar, se analizan las entradas y salidas del proceso para determinar cuáles de ellas nos interesa controlar para lograr el propósito deseado. Posteriormente, se estudia el proceso que conduce al objetivo planteado, y, por último, se determinan los dispositivos de control que más se ajustan a la planta y a los fines propuestos, para llevar a cabo el sistema de control.

En la actualidad, los sistemas de control industrial pueden estar basados en diferentes tipos de controladores. Por un lado, cabe destacar los controladores lógicos programables, o también conocidos como PLCs, *Figura 2*.

Figura 2. PLC. Fuente: https://upload.wikimedia.org/wikipedia/commons/f/f3/Siemens_Simatic_S7-416-3.jpg

Estos dispositivos se emplean para automatizar procesos industriales y están compuestos por una CPU y los módulos de entrada y salida.

- La CPU está formada por el procesador, el cual se encarga de ejecutar el programa escrito por el usuario que se encuentra almacenado en la memoria.
- El módulo de entrada se ocupa de adaptar las señales provenientes de los sensores para que la CPU pueda interpretar la información.

- El módulo de salida es el encargado de activar los actuadores de campo una vez la CPU ha ejecutado la información recibida.

Actualmente, los PLCs emplean entradas y salidas de alta velocidad, permiten implementar control PID y proporcionan entradas y salidas tanto digitales como analógicas. Algunos de ellos, también son capaces de realizar comunicación entre múltiples PLCs, así como implementar HMIs y SCADAs.

Estos dispositivos de control cuentan con unas numerosas ventajas, entre las cuales cabe destacar su reducido tamaño y el ahorro de tiempo en la elaboración de proyectos. Aunque, frente a todas las ventajas que estos equipos ofrecen, es necesario contar con técnicos calificados específicamente en PLCs para el mantenimiento y seguimiento de su funcionamiento.

Por otro lado, existen los PACs o también llamados controladores de automatización programable, *Figura 3*. Estos controladores se basan en el control automático de sistemas, el diseño de prototipos y la medición. Están compuestos por una CPU, módulos de entradas y salidas, y uno o varios buses de datos que lo interconectan todo.

Los PACs se usan para comunicar, monitorizar y controlar equipos a través de múltiples redes y dispositivos, mediante protocolos estándar y tecnologías de red como Ethernet y OPC.

Figura 3. PAC. Fuente: http://www.machinedesign.com/mechanical/what-s-difference-between-plc-and-pac

Estos dos dispositivos de control son muy similares ya que ambos realizan las mismas funciones principales, pero existen algunos aspectos que los diferencian. La principal diferencia entre PLCs y PACs es su interfaz de programación, ya que los PACs emplean lenguajes como C o C ++, mientras que los PLCs, utilizan lenguajes basados en diagrama de contactos o lenguaje literal, entre otros. Por otro lado, los PLCs son útiles

para procesos simples y pequeños proyectos de automatización, mientras que los PACs proporcionan más flexibilidad en la programación y mayor capacidad de memoria. A continuación, en la *Figura 4*, se expone una tabla con otras diferencias presentes entre estos dos dispositivos de control.

Características	PLC	PAC
Soporta shocks eléctricos y vibración	✓	✓
Seguridad y estabilidad	✓	✓
Rangos de temperatura industriales	✓	✓
Trabajo en tiempo real	✓	✓
Entradas de fuente de poder redundantes	✓	✓
Procesador de punto flotante	✗	✓
Memoria no volátil	✗	✓
Conectividad a Ethernet vía WEB	✗	✓
Capacidad de administración de recursos	✗	✓
Capacidad ilimitada de lazos de control	✗	✓

Figura 4. Comparación entre PLC y PAC. Fuente: http://www.logicelectronic.com/BECKHOFF/Que%20es%20un%20PAC.htm

2.4. Nivel de supervisión

En este nivel se desarrollan los sistemas de supervisión, control y adquisición de datos. Estos sistemas permiten controlar y supervisar en tiempo real cualquier proceso industrial a distancia. Es decir, el operador puede observar el funcionamiento de la planta, leer y/o modificar los valores de los parámetros de esta e incluso arrancar o detener el proceso desde un ordenador.

Para llevar a cabo este sistema de supervisión y control, es necesario establecer una comunicación entre la planta a controlar y el ordenador donde se emplea el SCADA. Para facilitar la comunicación entre los sistemas software se emplea la tecnología OPC y para realizar la comunicación entre el dispositivo de control y el ordenador empleado para la monitorización, se utiliza el protocolo Modbus.

2.4.1 Tecnología OPC

Este estándar de comunicación fue creado por OPC Foundation y tiene múltiples ventajas, tanto para los fabricantes de hardware como para los de software. Por un lado, cabe destacar la simplificación de los programas que los fabricantes de hardware tienen que emplear, ya que con este sistema sólo tienen que llevar a cabo un conjunto de componentes de programa que utilizarán los clientes en sus aplicaciones. Y, por otro lado, la utilidad que ofrece a los fabricantes de software, ya que no es necesario que se adapten los drivers ante cualquier cambio de hardware.

Para entender mejor el concepto, en la *Figura 5* se muestra un ejemplo de una situación donde no se aplica este estándar de comunicación, y en la *Figura 6* se expone la misma situación, pero esta vez empleando el sistema OPC.

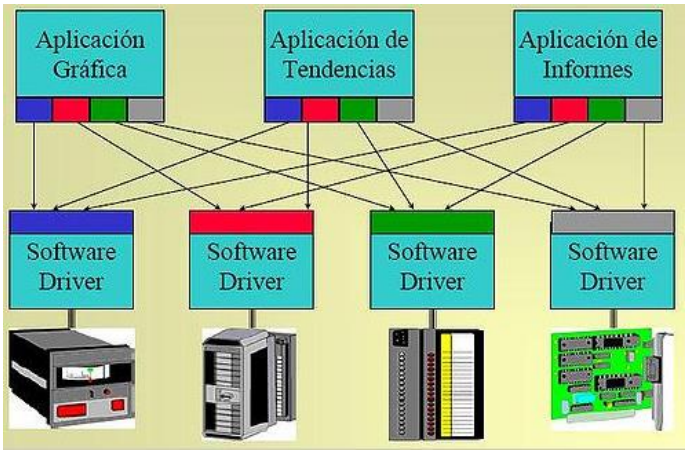

Figura 5. Problema sin tecnología OPC. Fuente*:*
https://es.wikipedia.org/wiki/OPC#/media/File:PROBLEOPC.JPG

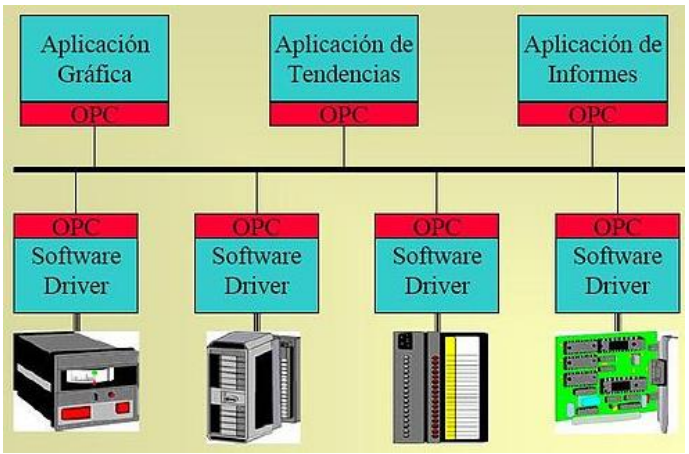

Figura 6. Solución al problema aplicando tecnología OPC. Fuente:
https://es.wikipedia.org/wiki/OPC#/media/File:SOLUCIOPC.JPG

Esta tecnología permite el acceso a datos de cualquier fuente, ya sea un dispositivo hardware o software, y se basa en una arquitectura cliente-servidor. Hoy en día se utiliza OPC como interfaz entre sistemas de automatización que se encuentran en los diferentes niveles de la pirámide de automatización, anteriormente explicada.

Existen diferentes especificaciones OPC, dependiendo de las necesidades de las aplicaciones industriales:

- OPC-DA: Esta tecnología OPC se basa en el acceso a datos de procesos actuales.
- OPC-A&E: OPC donde la información está basada en eventos y reconocimiento de alarmas.
- OPC-HDA: OPC basado en el acceso a datos históricos o archivados.
- OPC-DX: OPC basado en el intercambio de datos y la comunicación entre dos servidores, es decir, sin la necesidad de un cliente OPC intermedio.
- OPC-XML DA: OPC basado en el intercambio de datos que ofrece una interfaz *Simple Object Applicaction Protocol* (SOAP), es decir, permite la programación del cliente en Java y Python, entre otros lenguajes que soporta SOAP.

La versión más reciente de esta tecnología es *OPC UA*, o también OPC Arquitectura Unificada. Esta especificación integra las funcionalidades de las especificaciones anteriores (OPC-DA, OPC-A&E, OPC-HDA, …) y amplía dos funcionalidades más. Por un lado, se abandona la tecnología COM/DCOM y se utilizan los Servicios Web o el protocolo binario TCP para la transmisión de datos. Y, por otro lado, OPC-UA unifica las funcionalidades de las especificaciones OPC anteriores a esta, en un solo espacio de direcciones. Es decir, muestra datos actuales, notifica eventos y expone el historial en un solo espacio de direcciones.

2.4.2 Modbus

Modbus es un protocolo de comunicación que pertenece al nivel 4 del *Modelo TCP/IP*, es decir, al nivel de aplicación, *Figura 7*. Modbus está basado en la arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP), es decir, se puede utilizar tanto con cables serie, como con cables de red.

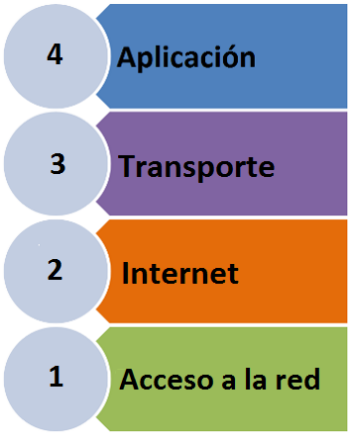

Figura 7. Modelo TCP/IP.

Este protocolo se utiliza con el fin de transmitir información entre distintos dispositivos electrónicos conectados a un mismo bus. Un claro ejemplo donde se aplica este protocolo es para la comunicación entre un PLC y un SCADA, *Figura 8.* Al presentar una arquitectura maestro/esclavo, el maestro es quien inicia las comunicaciones pidiéndole datos a un esclavo, y este es quien le proporciona la información. Cada dispositivo de red posee una única dirección.

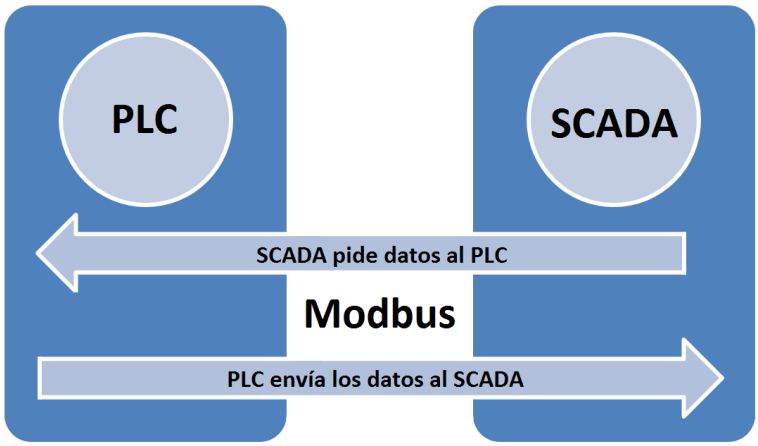

Figura 8. Ejemplo comunicación PLC-SCADA.

3. MONITORIZACIÓN Y CONTROL DE UNA PLANTA DE NIVEL DE TANQUES MEDIANTE UN CONTROLADOR INDUSTRIAL

Tal y como se mencionó en la sección *2.1. Introducción*, para la realización de este proyecto se han empleado las tecnologías que componen el nivel de campo, el nivel de control y el nivel de supervisión de la pirámide de automatización. A continuación, se detalla cada uno de estos niveles empleados para el diseño del sistema de control de la planta de nivel de tanques.

3.1. Elementos del nivel de campo

3.1.1 Descripción de la planta

La planta de nivel de tanques utilizada para este proyecto es la *Basic Process Rig 38-100 Feedback Instruments Limited, Figura 9*. Esta planta permite el estudio de control de procesos, utilizando como variables de medición el nivel de líquido y el caudal volumétrico.

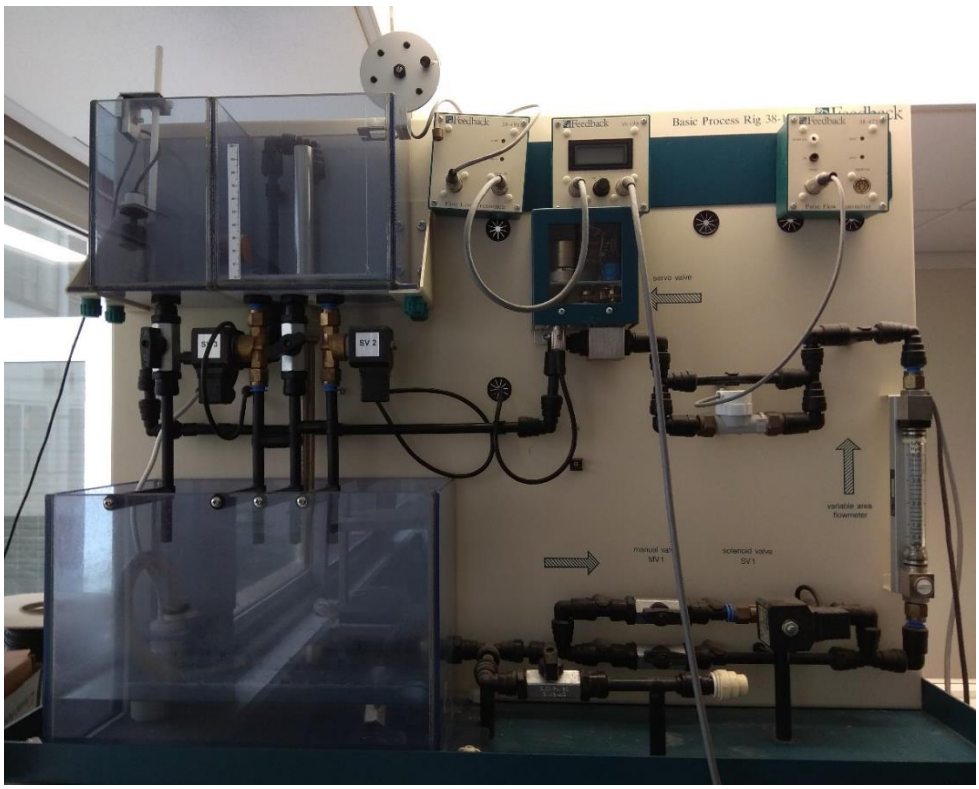

Figura 9. Basic Process Rig 38-100.

La Basic Process Rig 38-100 está formada por los siguientes elementos:

- Un tanque de proceso de doble compartimento donde se mide el nivel de agua situado en la parte superior de la planta.
- Un sensor de nivel encargado de medir el nivel de agua en el tanque.
- Un tanque de sumidero.
- Válvulas manuales o válvulas solenoides que conectan el tanque de proceso con el tanque de sumidero.
- La bomba, encargada de impulsar el agua a través del sistema.
- Un caudalímetro de área variable que mide la cantidad de caudal en el sistema.
- Una servo-válvula encargada de regular la cantidad de agua que llega al tanque de proceso.

Para entender mejor el proceso de llenado de la planta, se expone a continuación un esquema del mismo en la *Figura 10.*

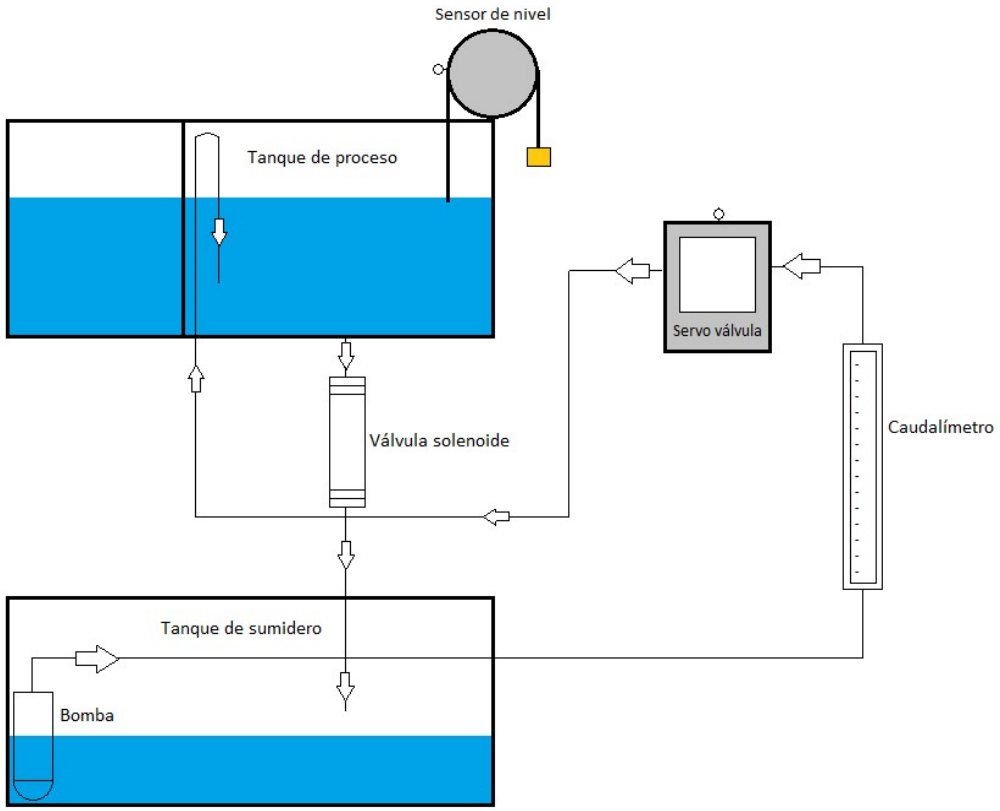

Figura 10. Esquema de la planta de nivel de tanques.

Además, el sistema incluye el *Process Interface 38-200, Figura 11,* que dispone de hasta un máximo de cuatro entradas de 4-20mA e incluye una fuente de intensidad, también de 4-20mA, dos convertidores de corriente a voltaje y un comparador de tensión. Desde esta interfaz se acceden a todos los sensores y actuadores de la planta.

Figura 11. Process Interface 38-200.

3.2. Control del sistema

En este apartado se estudia el controlador que se quiere implementar para alcanzar el objetivo planteado. Para este proyecto, el fin que se pretende obtener es el control del nivel de llenado del tanque de agua, por lo que se determina el dispositivo de control a utilizar y se analiza el proceso que conduce al objetivo deseado.

3.2.1 Registrador/controlador Nanodac

Con el fin de llevar a cabo el sistema de control de la planta, se considera la instalación de un dispositivo de control que se encargue de medir y regular el sistema. Como ya se ha explicado anteriormente, estos dispositivos se componen de una entrada de un sensor, un indicador digital y una salida de regulación, y se pueden utilizar para diferentes trabajos de control. Para el diseño de este proyecto se ha utilizado el registrador/controlador *nanodac de Eurotherm-Schneider Electric, Figura 12.*

El nanodac es un dispositivo de control de diseño compacto que permite comunicaciones por vía Ethernet. Este registrador/controlador ofrece una elevada precisión, y es capaz de adquirir y registrar los datos de un proceso en tiempo real, así como de realizar transferencia de datos mediante FTP a un servidor. Este instrumento

de control, cuenta con dos lazos de control PID que ofrecen un alto rendimiento y fiabilidad a su proceso. Desde la pantalla del *nanodac* se visualiza en la parte superior el nombre del canal con el que se está trabajando, en la pantalla principal los datos del canal y en la parte inferior, en la zona de estados, se muestra el nombre del dispositivo, la fecha y hora e iconos del sistema.

Este registrador/controlador ofrece múltiples funcionalidades, entre las cuales cabe mencionar la incorporación de alarmas del sistema y del canal en caso de fallo, y la representación gráfica de las variables del canal.

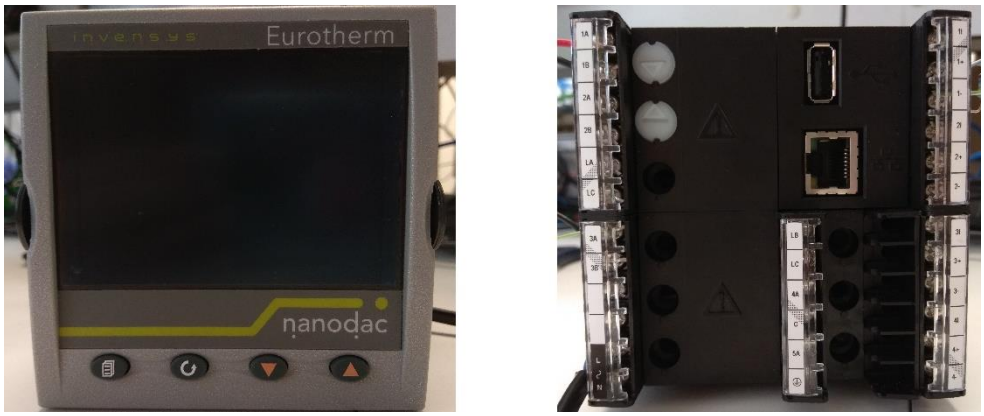

Figura 12. Registrador/controlador nanodac.

3.2.2 Problema de control

Con la intención de alcanzar los objetivos propuestos, se modifican las entradas del sistema para obtener las salidas o estados deseados. El objetivo de este proyecto es el control del nivel de una planta de tanques, por lo que se analizan las entradas del sistema que influyen en este.

El agua es impulsada por la bomba pasando por el caudalímetro de área variable que nos indica la cantidad de agua que se está bombeando. Una vez el agua haya pasado por el caudalímetro, llega a la servo-válvula que está conectada a la salida del dispositivo de control. Este le indica qué cantidad de agua debe dejar pasar al tanque de procesos dependiendo del nivel de agua que haya en el momento. El tanque de proceso dispone del sensor de nivel que va variando dependiendo del nivel del agua en el tanque y está conectado a la entrada del registrador/controlador *nanodac*. De esta forma, el dispositivo de control sabrá en todo momento a qué nivel está el agua del tanque.

Para entender mejor el funcionamiento del sistema de control, podemos ver un esquema del proceso en la Figura 13.

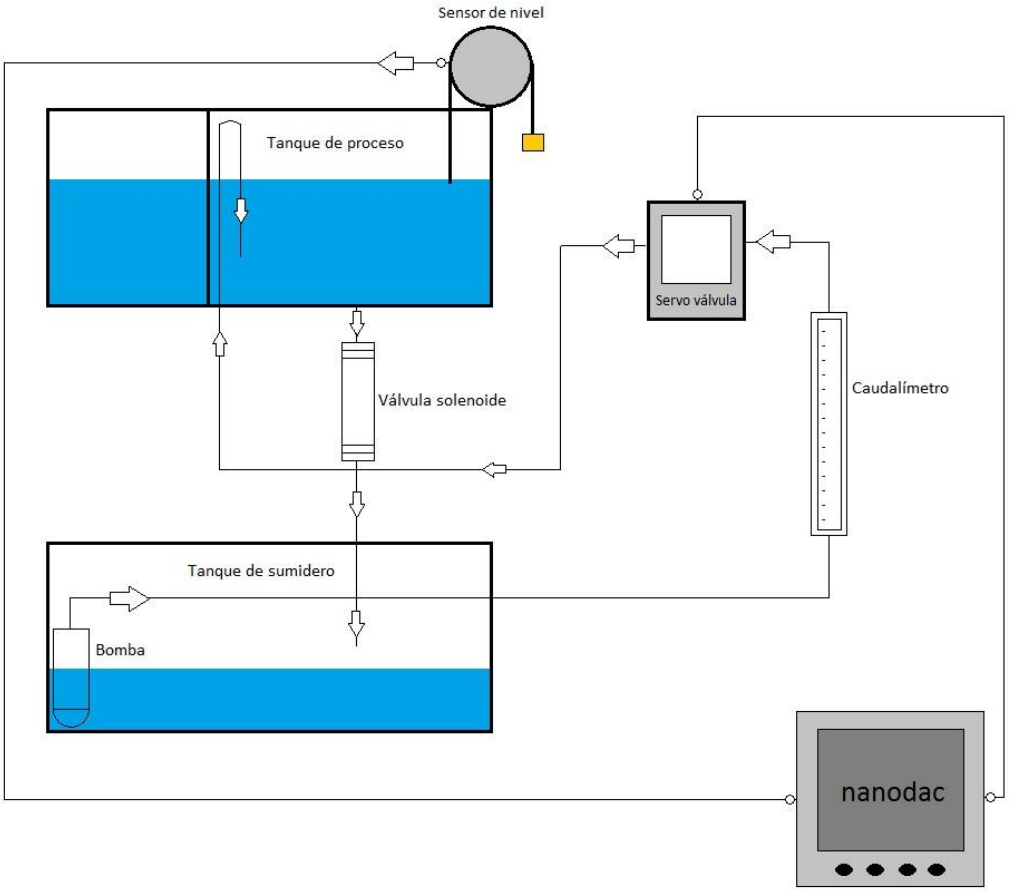

Figura 13. Esquema del sistema de control de la planta de nivel de tanques.

3.3. Supervisión del sistema

En este apartado se detalla el procedimiento a seguir para la implementación del sistema de supervisión.

3.3.1 Monitorización del sistema mediante OPC

Para la monitorización del sistema es necesario hacer uso de un estándar de comunicación que permita la interacción entre los componentes de software y compartan datos. Para la realización de este diseño se ha empleado el sistema OPC.

En este proyecto, el servidor OPC es quien lee los valores de los parámetros del controlador *nanodac* y el cliente OPC es el sistema SCADA que accede a los valores

leídos del servidor OPC para mostrarlos por pantalla y/o modificarlos. A continuación, se muestra un esquema del funcionamiento del sistema OPC empleado, en la *Figura 14*.

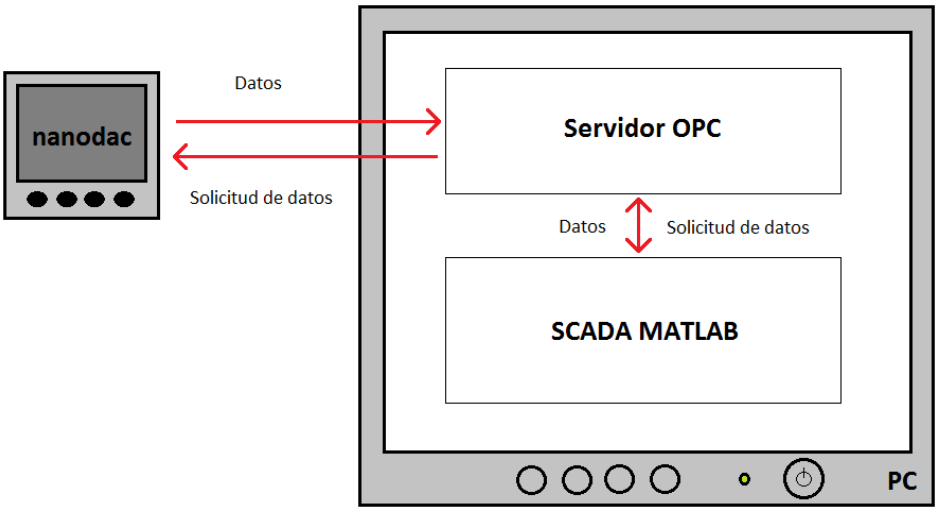

Figura 14. Esquema de la monitorización del sistema mediante OPC.

Los parámetros de la planta recogidos por el controlador son transferidos al servidor OPC mediante FTP. El servidor almacena estos datos para proporcionárselos posteriormente al SCADA, y este los muestre por pantalla. Si se desea cambiar el valor de cualquier parámetro del controlador para modificar el control del sistema, se realiza el mismo procedimiento, pero de forma inversa. Se introduce desde el SCADA el valor del parámetro a cambiar, este lo almacena en el servidor OPC que a su vez los transfiere al controlador mediante FTP actualizando así los valores que se encontraban anteriormente en el dispositivo.

4. IMPLEMENTACIÓN Y CONFIGURACIÓN

Esta sección describe los aspectos del proyecto relacionados con la implementación hardware y software del sistema de control.

A continuación, se expone detalladamente la instalación y la configuración del registrador/controlador *nanodac* y del estándar de comunicación OPC empleado. Además, se desarrolla la estructura diseñada para la realización del sistema de monitorización y control de la planta.

4.1. Configuración del controlador

Para llevar a cabo el sistema de control correctamente, es fundamental el análisis detallado de la planta que se quiere controlar para realizar la instalación eléctrica y la configuración del controlador.

4.1.1 Instalación eléctrica del sistema

Para realizar la comunicación entre el nivel de campo y el nivel de control, se analizan las entradas y salidas de la planta que nos interesan controlar y se lleva a cabo la instalación eléctrica del dispositivo con la planta. La conexión de la alimentación del dispositivo se realiza tal y como se muestra en la *Figura 15,* y posteriormente se conectan las entradas y las salidas del controlador que más se adapten al sistema. En este caso las entradas son las 1+ y 1-, y las salidas son las 3A y 3B, según se muestra en la *Figura 16*.

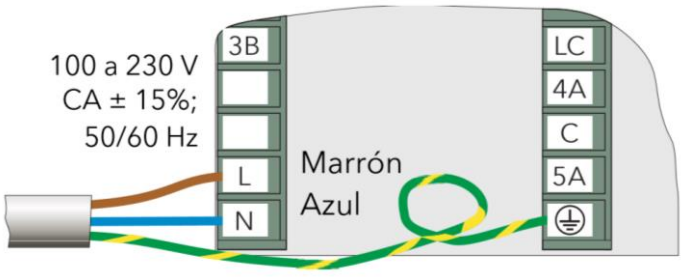

Figura 15. Conexión de alimentación. Fuente: *Guía del usuario* de *nanodac.*

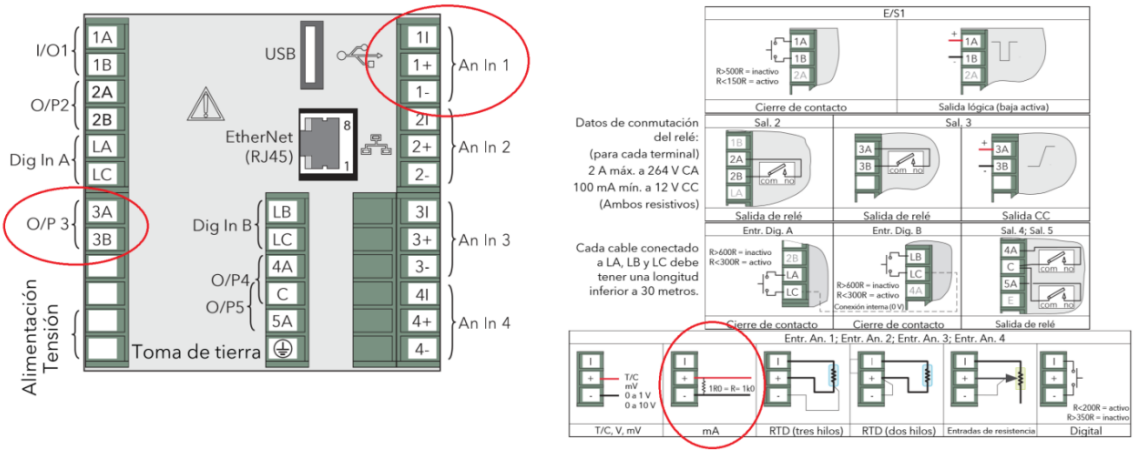

Figura 16. Instalación Eléctrica del Controlador. Fuente: *Guía del usuario* de nanodac.

La conexión entre el dispositivo de control y las variables de la planta, *Figura 17,* se lleva a cabo mediante cables banana banana conectados por un extremo a la planta y por el otro a diferentes cables de hilo unipolar que a su vez están conectados al *nanodac*.

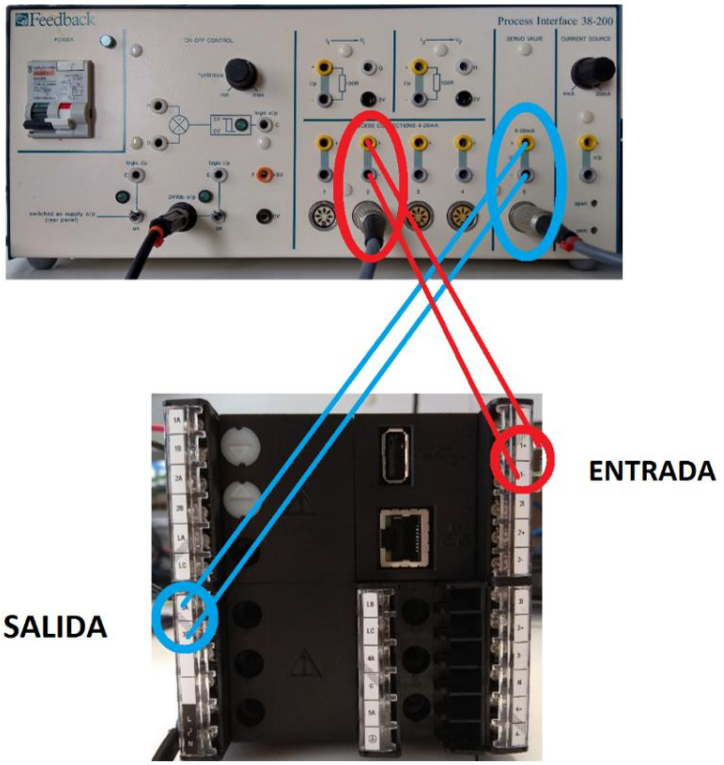

Figura 17. Conexión dispositivo de control-planta.

4.1.2 Configuración del Instrumento

A continuación, se describe la configuración establecida en el controlador, ajustando las entradas y las salidas siguiendo los pasos de la *Guía del usuario* de *nanodac*.

El controlador dispone de un menú principal tal y como se muestra en la *Figura 18*, donde se detalla cada una de las áreas que contiene la opción *Configuración*. En este proyecto sólo de ajusta esta opción del controlador.

Figura 18. Menú principal. Fuente: *Guía del usuario* de nanodac.

4.1.2.1 Ajuste de la entrada

El primer apartado a configurar es el de *Ajuste Entrada* dentro del área de *Instrumento*. En esta sección, se ajusta el canal a utilizar, en este caso, el *Canal 1*. El procedimiento a seguir para el ajuste se expone en la *Figura 19.*

Figura 19. Procedimiento de ajuste del canal 1.

En este apartado se especifica manualmente el valor bajo y alto de entrada del canal 1 y se estabiliza el valor medido por el controlador tal y como se muestra en la figura anterior.

- Valor bajo de entrada: 4mA.
- Valor alto de entrada: 20mA.

4.1.2.2 Ajuste de la salida

El siguiente ajuste de esta área es *Ajuste de salida, Figura 20*. Aquí, se lleva a cabo la calibración del nivel bajo y alto de la salida del controlador.

- Nivel bajo de salida: 4mA.
- Nivel alto de salida: 20mA.

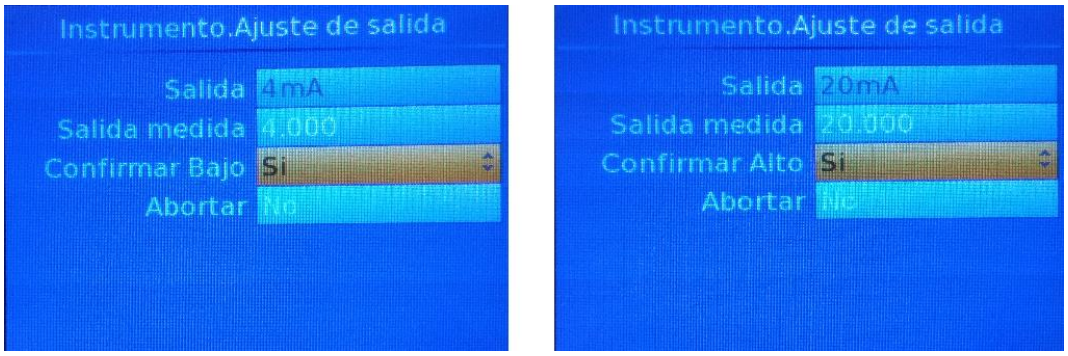

Figura 20. Instrumento. Ajuste de salida.

4.1.3 Configuración de red

Para poder monitorizar posteriormente el sistema, se configura el apartado de red del controlador, *Figura 21. Menú de interfaz de red*, donde se especifican los siguientes aspectos para más adelante poder establecer conexión con el PC.

- Tipo de IP.
- Dirección IP.
- Submáscara.
- Puerta de enlace.

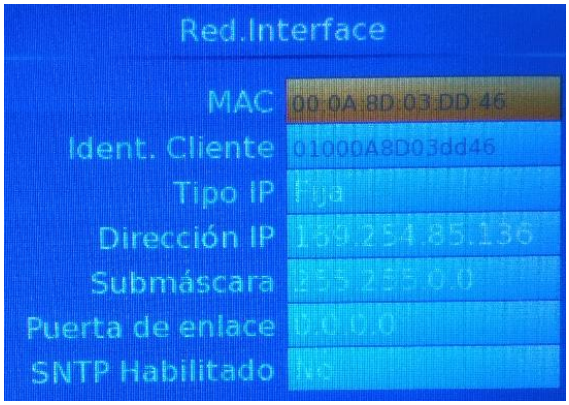

Figura 21. Menú de interfaz de red.

4.1.4 Configuración del canal

El *Menú Canal, Figura 22,* corresponde con los ajustes del canal a utilizar. En este se configuran los siguientes aspectos:

- Tipo de entrada del canal utilizado: mA.
- Valor más bajo de la señal de entrada: 4mA.
- Valor más alto de la señal de entrada: 20mA.
- Escala baja del valor de proceso: 4.
- Escala alta del valor de proceso: 20.

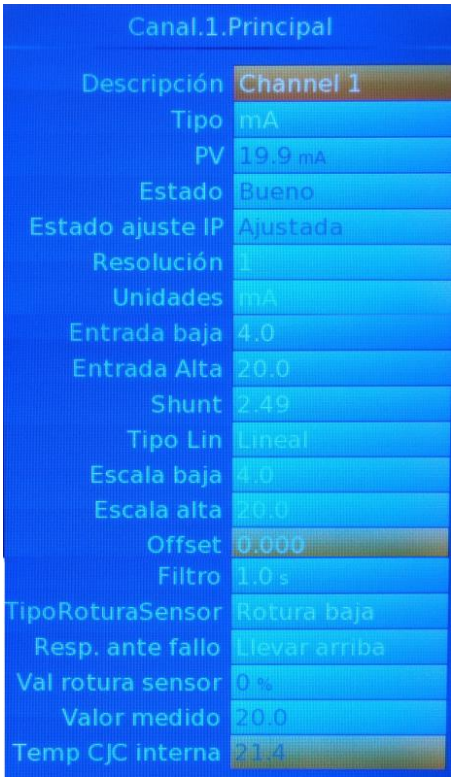

Figura 22. Menú principal del canal.

4.1.5 Configuración del lazo de control

El *Menú lazo* es el que corresponde con la configuración del PID del controlador que lleva a cabo el sistema de control de la planta, en este caso es el PID 1. Dentro de este menú se han ajustado cinco apartados diferentes: *Principal, Configuración, PID, Consigna* y *Salida*.

4.1.5.1 Ajuste Principal

El primer apartado es el *Principal*, *Figura 23*, y en él se establece si el control de la planta se llevará a cabo de forma manual o automática, muestra el PV actual y se especifica el SP deseado.

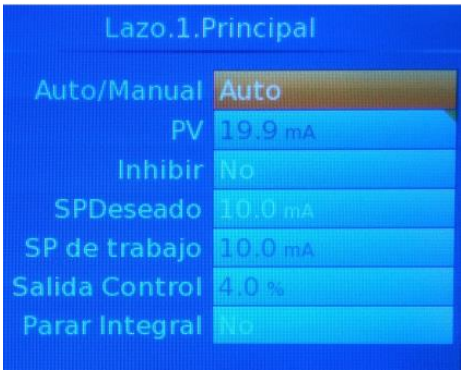

Figura 23. Menú Lazo 1. Principal.

4.1.5.2 Configuración del lazo 1

En el apartado *Configuración*, *Figura 24*, se especifica el tipo de control del lazo, el tipo de acción de control y las unidades de la banda proporcional del controlador, entre otros.

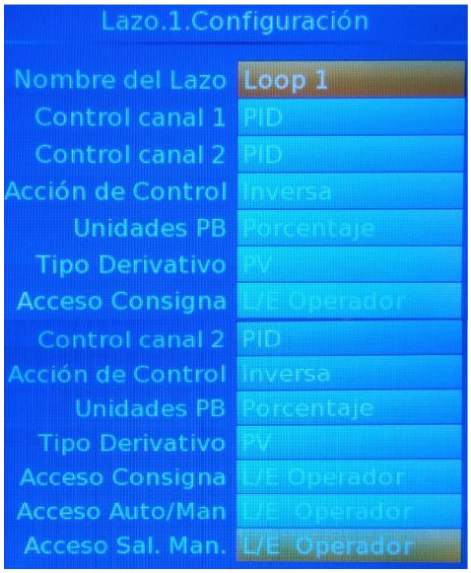

Figura 24. Menú Lazo 1. Configuración.

4.1.5.3 Configuración del PID

En el menú *PID*, *Figura 25,* se especifican los valores de los parámetros del PID, es decir, la banda proporcional del lazo, el tiempo integral y el tiempo derivativo.

Figura 25. Lazo 1. PID.

4.1.5.4 Configuración de la consigna

En la configuración de la *Consigna, Figura 26*, el operador especifica el valor del SP al que se desea llegar, entre otros ajustes disponibles en este apartado.

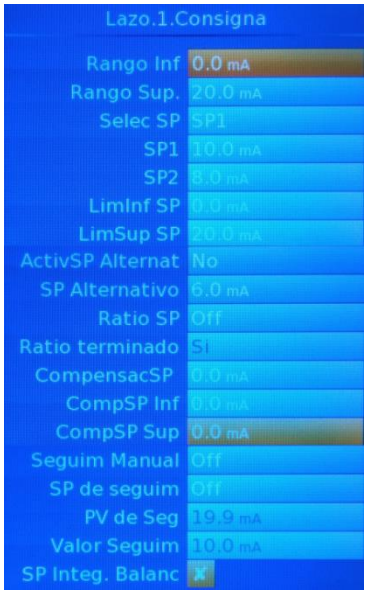

Figura 26. Lazo 1. Consigna.

4.1.5.5 Configuración de la salida del lazo 1

La última sección de esta configuración corresponde con la configuración de la *Salida, Figura 27*. Aquí se definen todos los parámetros que influyen en la salida del controlador:

- Valor de la salida forzada en caso de que el controlador opere en modo manual
- Valor de salida inferior y superior en porcentaje.

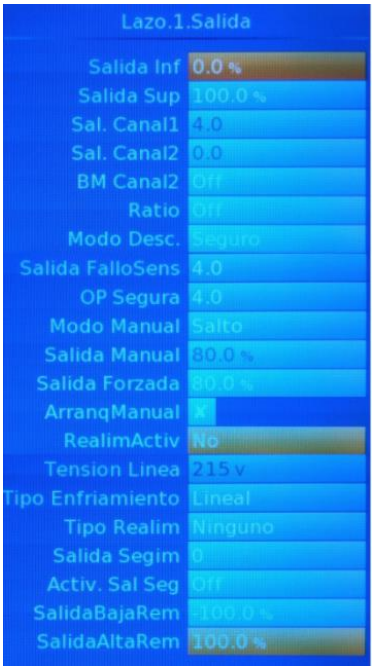

Figura 27. Lazo 1. Salida.

4.2. Comunicación TCP ModBus

En el presente apartado se expone la conexión llevada a cabo entre el nivel de control y el nivel de supervisión, para posteriormente monitorizar el sistema. Para establecer la comunicación se utiliza el protocolo TCP ModBus que permite que la unidad funcione como un dispositivo "esclavo" de uno o más ordenadores conectados a través del conector RJ45 de la parte posterior del registrador/controlador *nanodac*. Para realizar la conexión se conecta un cable de Ethernet estándar entre el conector RJ45 del dispositivo y el ordenador, tal y como se muestra en la *Figura 28.*

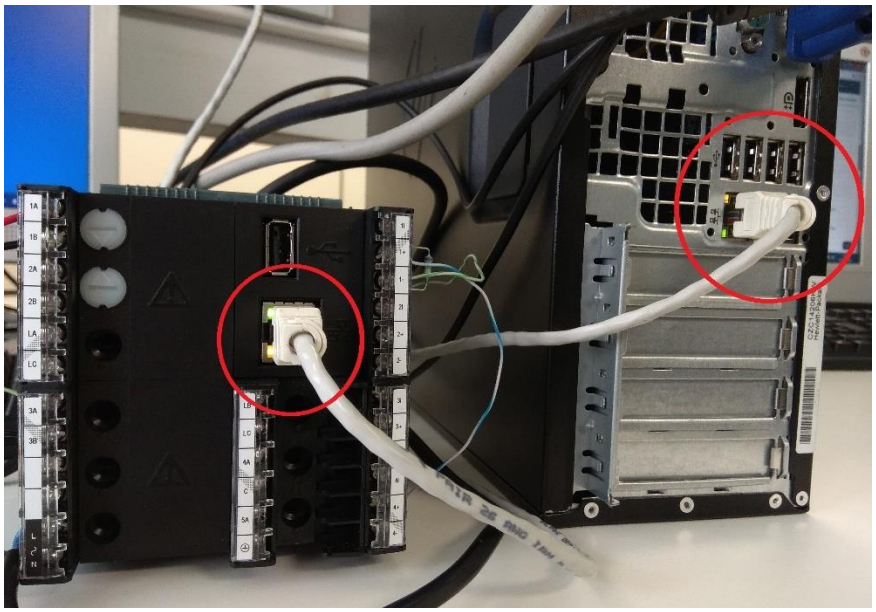

Figura 28. Conexión dispositivo de control-PC.

Para configurar los puertos TCP/IP para ModBus mediante Ethernet desde el PC, se determina el nombre para el puerto y se escribe la dirección IP del controlador para que el ordenador lo reconozca, tal y como se observa en la *Figura 29.*

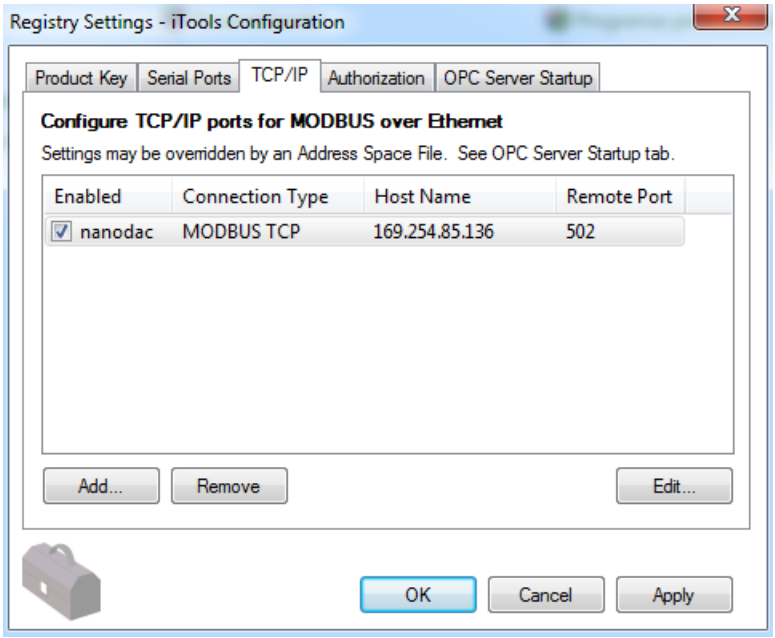

Figura 29. Configuración ModBus TCP.

Para comprobar la conexión, se realiza una búsqueda de dispositivos con la herramienta *iTools Engineering Studio* del *nanodac*. *iTools* es un software para PC que permite al operador acceder de manera rápida y sencilla a la configuración del dispositivo de control, ya que contiene los mismos apartados de configuración que el *nanodac,* por lo que cada vez que se desee modificar un parámetro, se llevará a cabo a través de esta herramienta.

Si la configuración de los puertos TCP/IP se ha realizado correctamente y se establece conexión entre el dispositivo de control y el PC, aparecerá en la parte inferior de *iTools* una imagen del dispositivo de control, como se muestra en la *Figura 30.*

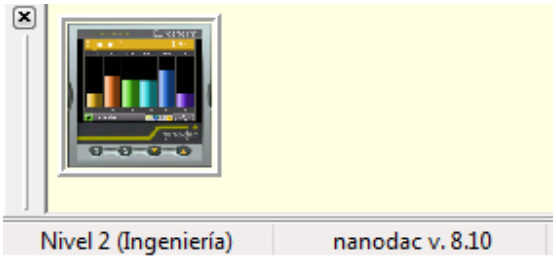

Figura 30. Conexión establecida nanodac-PC.

La herramienta *iTools Engineering Studio* también dispone de un editor gráfico de conexiones, *Figura 31*, que permite conectar parámetros entre sí y acceder a la configuración del dispositivo de control ya sea para ver y/o modificar los valores de los parámetros, *Figura 32.*

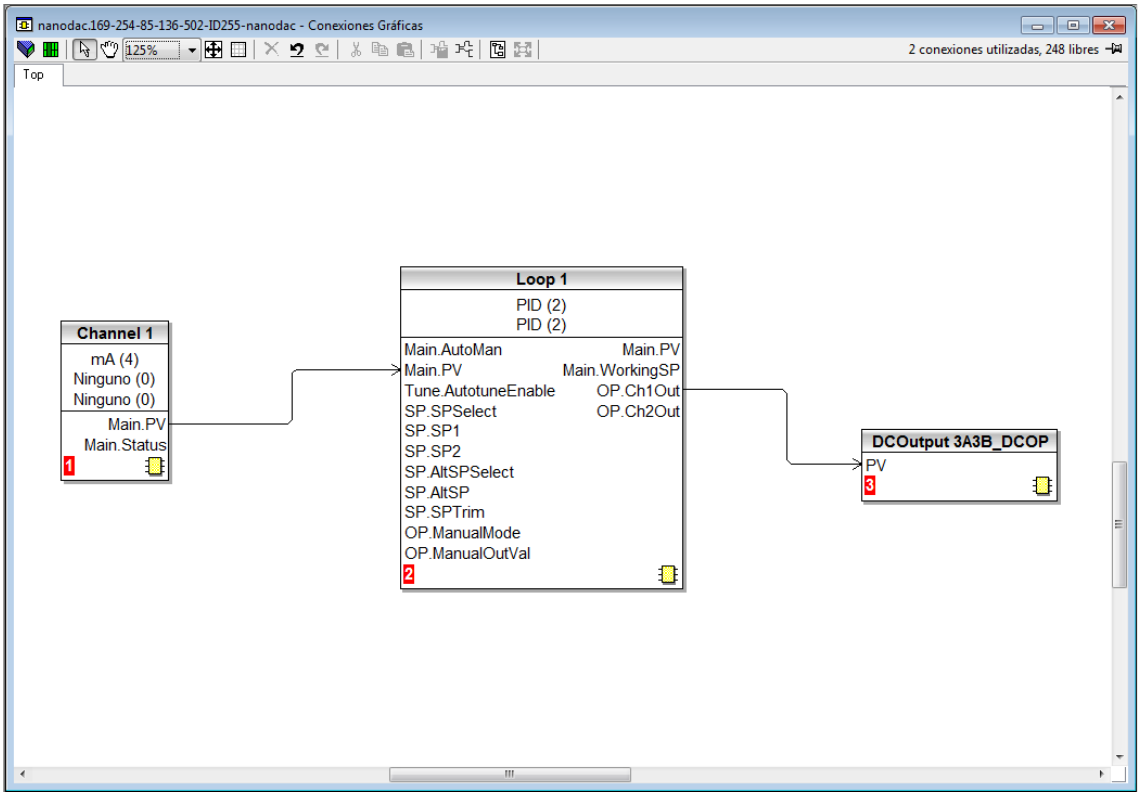

Figura 31. Editor gráfico de iTools.

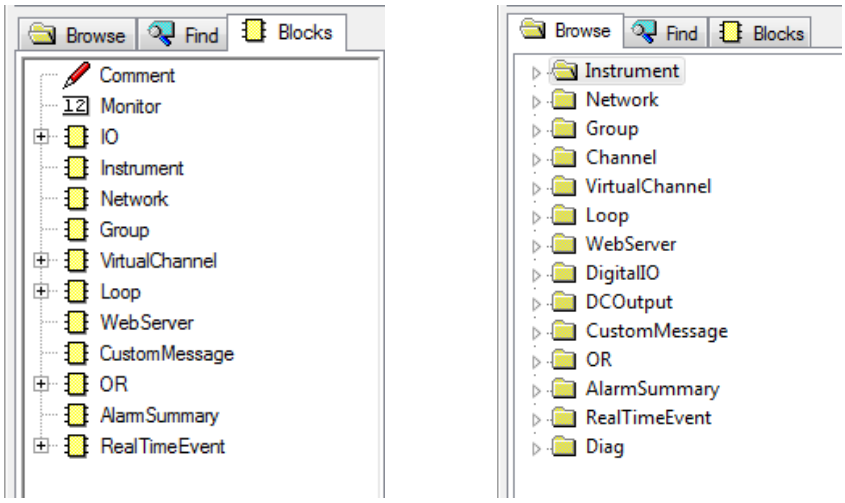

Figura 32. Bloques funcionales y configuración del dispositivo de control en iTools.

4.3. Configuración del OPC

iTools Engineering Studio contiene una interfaz que dispone de un explorador OPC llamado *iTools OPC Scope* y un servidor OPC, *iTools OPC Server.* El explorador permite conectarse a cualquier servidor OPC, y es capaz de analizar tendencias, registrar datos y realizar intercambio dinámico de datos. Tal y como se observa en la *Figura 33*, el explorador permite visualizar el menú completo de la configuración del dispositivo, así como elegir los canales que se desean mostrar por pantalla, en este caso, se configura para representar únicamente los valores del canal 1, ya que es el canal que se utiliza para el sistema de control de este proyecto.

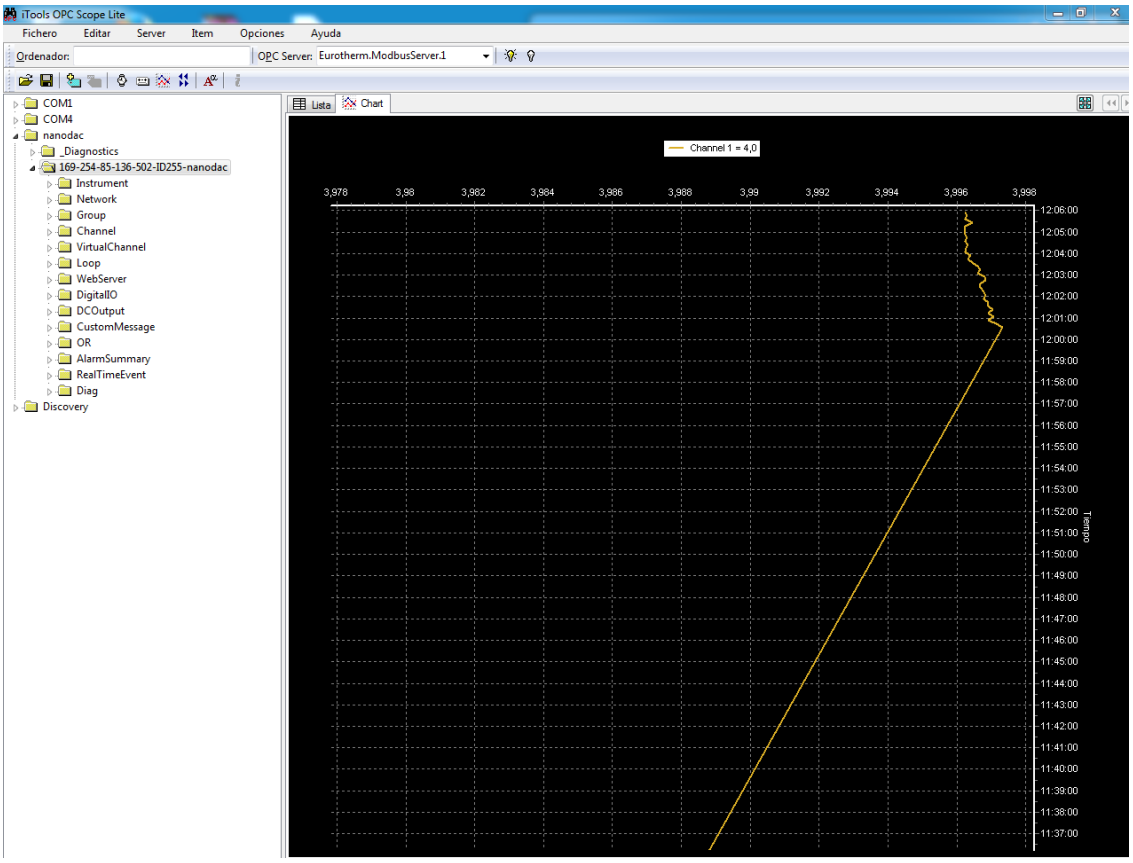

Figura 33. iTools OPC Scope.

Por otro lado, *iTools OPC Server* es el servidor encargado de registrar los valores de los parámetros del controlador para que posteriormente el cliente OPC pueda acceder a ellos. Este software también incorpora la visualización del menú de configuración del controlador, como se observa en la *Figura 34.*

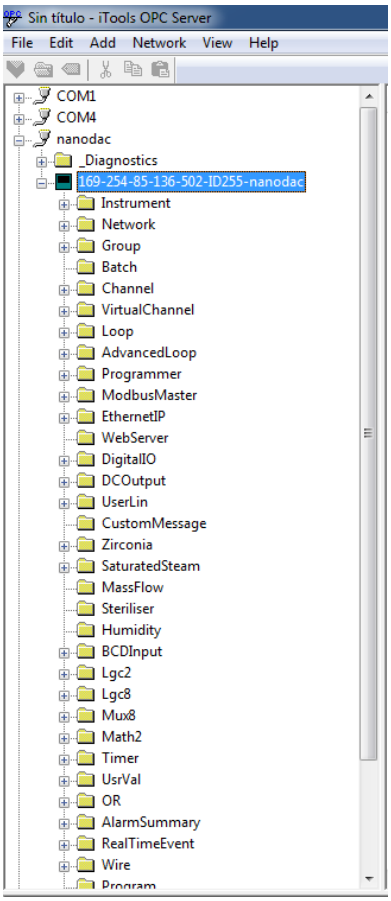

Figura 34. Menú iTools OPC Server.

Además, en la parte inferior del programa se encuentra una ventana, *Figura 35*, que muestra por pantalla diferentes tipos de mensajes, ya sean de error o simplemente mensajes de información, y debajo de esta, se indica si se está escaneando el sistema, la cantidad de clientes conectados al servidor, el número de grupos OPC presentes y los dispositivos conectados.

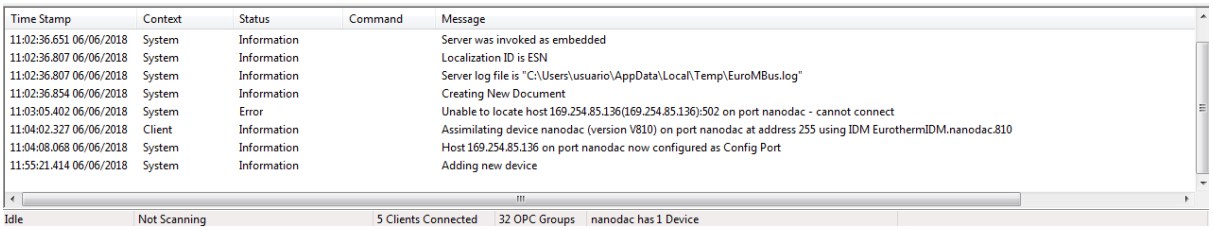

Figura 35. Ventana de mensajes OPC Server.

Tanto el explorador como el servidor OPC son del mismo fabricante que el registrador/controlador *nanodac*, por lo que no ha hecho falta realizar ningún tipo de configuración en estos programas. Una vez el PC detecta el dispositivo de control con el programa *iTools Engineering Studio*, el explorador y el servidor OPC cargan de forma inmediata los valores de cada uno de los parámetros del dispositivo.

Por otra parte, se encuentra el cliente OPC, encargado de acceder a los valores leídos por el servidor para posteriormente monitorizar el sistema. En este proyecto se ha llevado a cabo mediante la herramienta MATLAB, apartado *5. Programación del sistema de monitorización y control,* la cual dispone de una *App* que permite diseñar y desarrollar un sistema SCADA mediante el cual se llevará a cabo la monitorización de la planta.

4.4. Diseño del sistema SCADA

Para la monitorización y control de la planta de nivel de tanques se ha diseñado un sistema SCADA que permite observar el funcionamiento de la planta en tiempo real, leer y modificar los parámetros del controlador y ver gráficamente el nivel del agua y la salida operativa del PID implementado.

El sistema de supervisión, control y adquisición de datos cuenta con un botón *Conectar*, *Figura 36 (a), que* realiza la conexión con el dispositivo de control mediante el OPC Data Access, grupo de estándares cliente-servidor que proporciona las especificaciones necesarias para la comunicación entre el controlador y el SCADA. También cuenta con un botón *Cargar Configuración, Figura 36 (b),* que lleva a cabo la lectura de los parámetros del controlador y los muestra por pantalla.

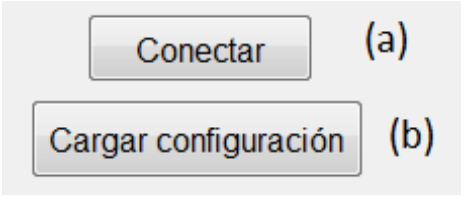

Figura 36. Botón de conexión y cargar configuración SCADA.

Estos valores aparecen en un cuadro de texto no editable de color amarillo. Si el usuario desea modificar estos valores, el sistema dispone de unos cuadros de texto editables situados a la derecha de cada uno de los valores. El valor introducido es leído y se carga directamente al controlador, modificando de esta forma el valor que se

encontraba anteriormente. Al modificar cualquiera de estos valores, se cambia el valor del cuadro de texto amarillo automáticamente por el valor actual introducido. De esta forma, aparecerán siempre por pantalla los valores actuales de los parámetros del controlador.

Los parámetros del controlador que el SCADA muestra por pantalla son los siguientes:

- SP o setpoint: Este es el valor de la consigna a la que se desea llegar.
- PV o process value: Valor actual de proceso.
- BP: Banda proporcional del lazo de control.
- Ti: Tiempo integral del lazo de control.
- Td: Tiempo derivativo del lazo de control.
- Salida operativa: Valor en porcentaje de la salida del controlador, es decir, apertura de la servo-válvula en porcentaje.

Todos estos parámetros aparecen en el SCADA tal y como se muestra en la *Figura 37.*

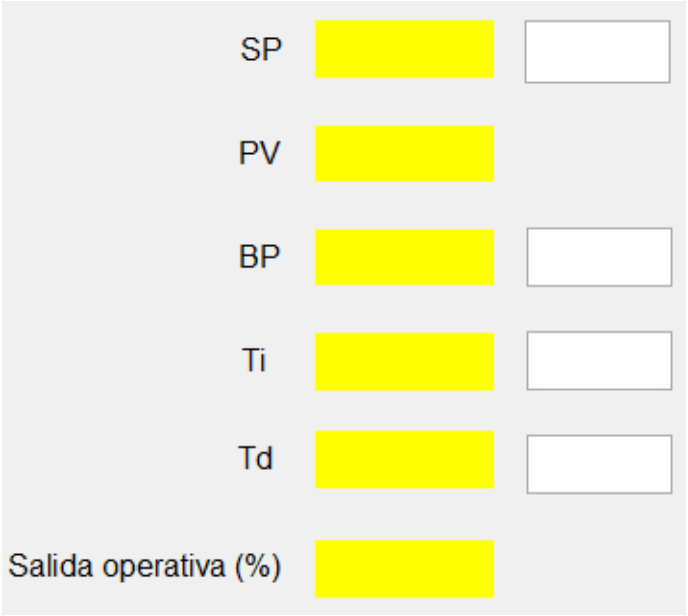

Figura 37. Parámetros del controlador en el SCADA.

Por otro lado, el SCADA cuenta con un botón *Start* y un botón *Stop*, *Figura 38,* para iniciar y finalizar el seguimiento del sistema de control, respectivamente.

Figura 38. Botón Start y Stop SCADA.

También, se dispone de una opción que permite el cambio del modo de control del sistema, es decir, permite seleccionar si se desea un control automático o un control manual de la planta, *Figura 39*. Si la opción elegida es la de *Modo Automático* se controla automáticamente la potencia de salida del controlador en una configuración de lazo cerrado. Si, por el contrario, la opción elegida es la de *Modo Manual*, es el operador quien controla la potencia de salida.

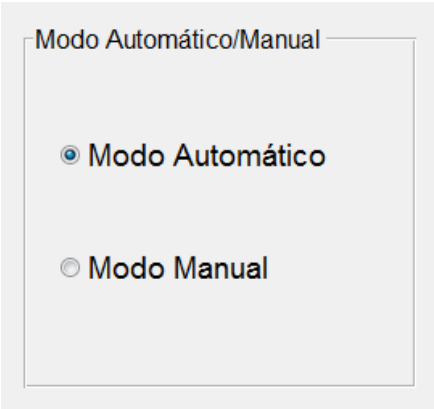

Figura 39. Modo Automático/Manual SCADA.

Para efectuar el seguimiento de forma óptima, se representan gráficamente los tres parámetros que más nos interesan del sistema de control, estos son, el PV, el SP y la salida operativa del controlador. Estas gráficas se pueden observar en la *Figura 40*, donde en la gráfica *Altura* se representa en color verde el valor del SP y en negro el valor del PV. Y en la gráfica *Salida operativa (%)* el valor en porcentaje de la salida del controlador, independientemente del modo de control en el que se encuentre (modo manual o modo automático).

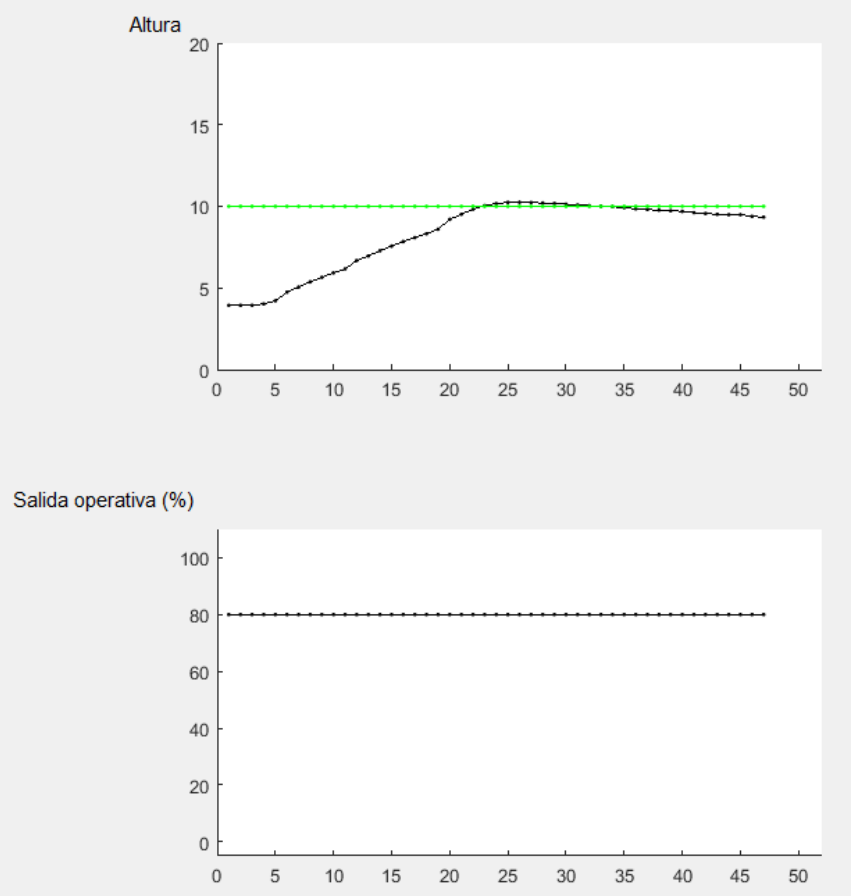

Figura 40. Gráficas de los parámetros del controlador en el SCADA.

Por último, se cuenta con dos botones, *Apagar* y *Borrar*, *Figura 41*, que desconecta el SCADA del controlador y borra el grupo de estándares OPC Data Access, respectivamente. Estos botones se utilizan cuando el seguimiento del sistema de control ha finalizado, y se quiere cerrar el sistema SCADA.

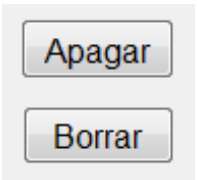

Figura 41. Botón Apagar y Borrar SCADA.

A continuación, se observa en la *Figura 42,* el SCADA finalizado con cada uno de los elementos anteriormente descritos.

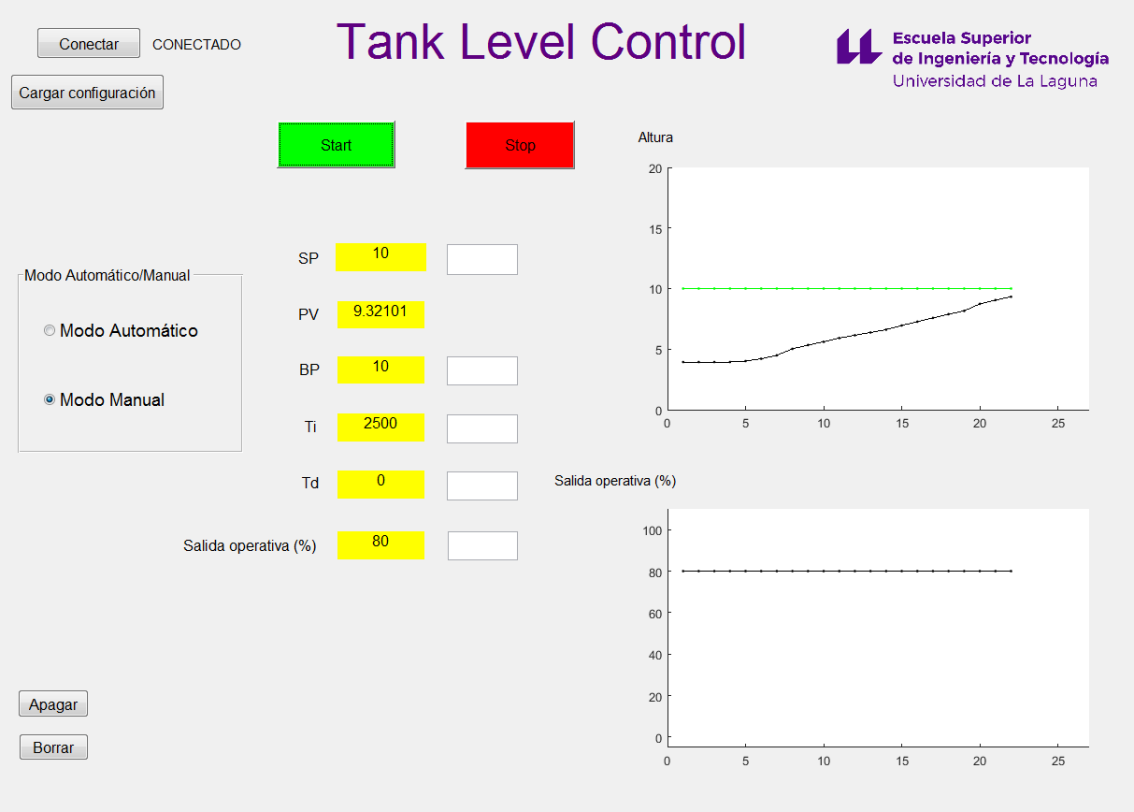

Figura 42. SCADA.

4.4.1 Alarmas del sistema

Para garantizar un buen seguimiento del sistema de control, se ha creado una alarma que se active cuando el valor de PV supere el valor de la consigna definido más un porcentaje de margen de error establecido. Cuando esta se activa, aparece de forma intermitente el símbolo de una 'X' al lado del valor de PV, indicando que hay un fallo en el sistema. Por otro lado, también aparece en la parte inferior del SCADA un mensaje de *Alarm*, como se observa en la *Figura 43.* Para desactivar la alarma y borrar el mensaje, el operador debe pulsar el botón *OK.* Si el valor de PV sigue siendo mayor que el SP, la alarma no se desactivará, pero si, por el contrario, el valor ha disminuido y se encuentra por debajo del valor de consigna, la alarma se desactivará y desaparecerá tanto el símbolo 'X' como el mensaje.

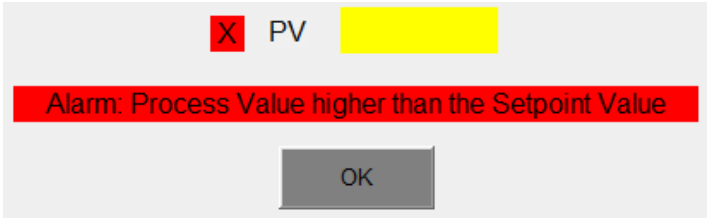

Figura 43. Mensaje de alarma del sistema.

En la *Figura 44,* se muestra el SCADA con el indicador de la alarma activado, mostrando el mensaje de alarma por pantalla.

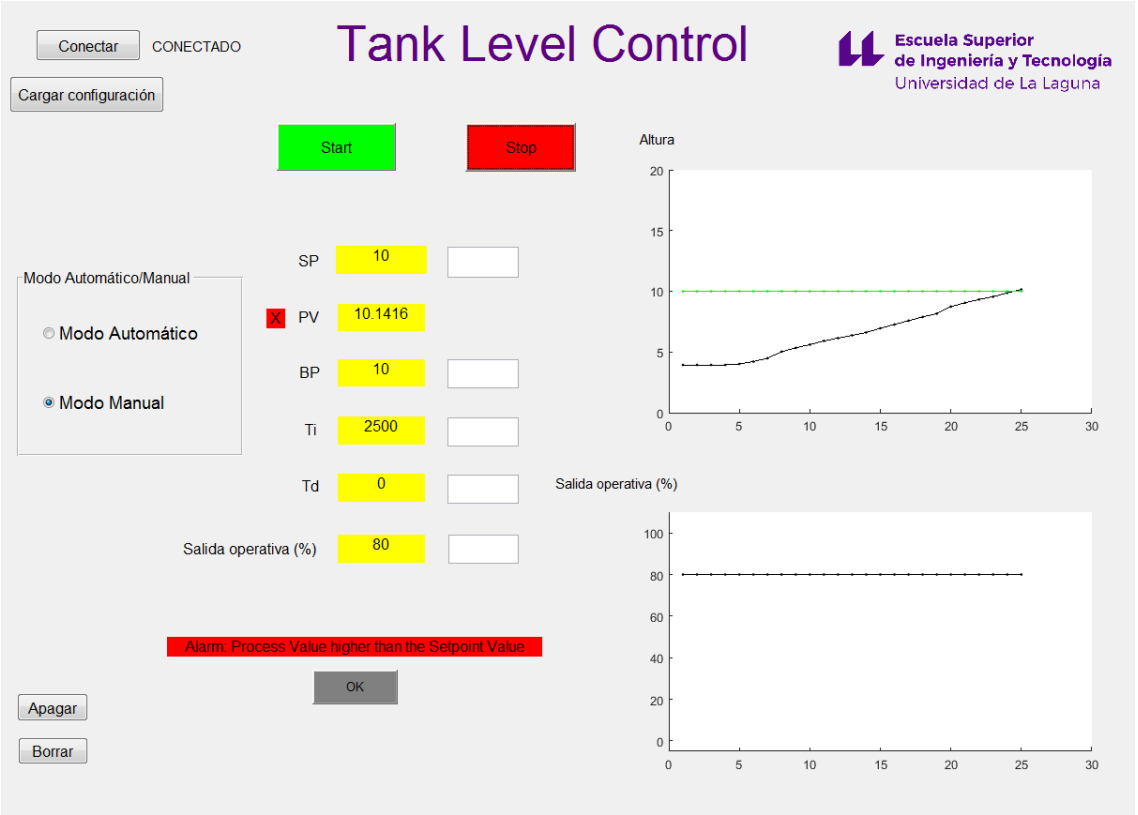

Figura 44. SCADA con alarma.

5. PROGRAMACIÓN DEL SISTEMA DE MONITORIZACIÓN Y CONTROL

Para llevar a cabo el desarrollo del sistema SCADA, se ha empleado la herramienta de software matemático MATLAB. Esta herramienta ofrece un IDE con un lenguaje de programación de alto nivel propio de la herramienta y ampliamente extendido en el campo de la ingeniería.

MATLAB presenta un amplio conjunto de funcionalidades, entre los cuales cabe destacar la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con otros dispositivos hardware, estas dos últimas funcionalidades se emplean en el desarrollo del sistema de monitorización y control de este proyecto.

5.1. Solución en Matlab

El sistema SCADA se ha elaborado mediante la app GUI de MATLAB, que proporciona herramientas para diseñar gráficamente interfaces de usuario. Tal y como se mencionó en el apartado *4.4 Estructura del sistema de monitorización y control*, el SCADA está formado por un botón *Conectar*, uno de *Cargar Configuración*, cuadros de texto editables y no editables para mostrar los parámetros leídos por pantalla y modificarlos, un botón de *Start* y otro de *Stop*, las opciones para elegir el modo de operación, *Modo Automático* o *Modo Manual* y los botones *Apagar* y *Borrar*. Estas funciones se han ido creando de una en una de la siguiente manera.

5.1.1 Función *Conectar*

El botón *Conectar* realiza la conexión entre el cliente y el servidor mediante el OPC Data Access. Esta función carga primero toda la información del PC y del servidor OPC para posteriormente crear el objeto OPC data Access, *da,* y realizar la conexión mediante el comando: *connect(da).* También, se crea con esta función el grupo de ítems *grp,* donde se irá añadiendo cada ítem utilizado para la monitorización y control de la planta. Esta primera parte del código del programa se ve reflejado en la *Figura 45.*

```
function conectar_da_Callback(hObject, eventdata, handles)

IP2 = 'localhost'; %Dirección IP del PC
serverID = 'Eurotherm.ModbusServer.1'  %Nombre del servidor OPC

opcreset; %Borra todos los objetos existentes

hostInfo = opcserverinfo(IP2) %Informacion del servidor OPC
allServers = hostInfo.ServerID
da = opcda(IP2, serverID) %Se crea el OPC data access object

connect(da); %Conecta el cliente OPC con el servidor OPC

handles.da = da
guidata(hObject,handles)

grp = addgroup(handles.da) %Se crea grupo de items

handles.grp = grp
guidata(hObject,handles)
```

Figura 45. Función Conectar.

La segunda parte de la función *Conectar* corresponde con la exploración de cada uno de los ítems del servidor utilizados para el sistema de control y adición de estos al grupo de ítems *grp*. En la *Figura 46,* se expone los comandos empleados para esta parte de la función. Se realiza exactamente lo mismo para el ítem 2, 3, 4, 5, 6 y 8 que corresponden con el parámetro *SP, BP, Ti, Td, salida operativa del controlador* y *salida forzada*, respectivamente.

```
% % % % % % % % % % % % % % % %  ITEM 1:PV  % % % % % % % % % % % % %

%Exploración de los items del servidor:
allitems = serveritems(handles.da, '*');

%Exploración de los items del servidor con el nombre: Loop.1.Main.PV:
variable_1 = serveritems(handles.da, '*Loop.1.Main.PV*');

%Se añade el item encontrado en el grupo de items:
item_1 = additem(handles.grp, variable_1);

handles.item_1 = item_1
guidata(hObject,handles)
```

Figura 46. Función Conectar - ITEM_1.

Para el ítem 7, variable que corresponde con el modo de operación del controlador (modo automático/modo manual), el código es diferente. Esta variable permite conocer el modo de operación dependiendo de su valor.

- Si el valor de la variable es igual a '0', el controlador opera en modo automático.

- Si, por el contrario, el valor es igual a '1', estará en modo manual.

Para conocer el valor del ítem, se añade el comando *read*, que lee el contenido del ítem, y posteriormente el comando *getfield*, que accede al campo *'Value'*, tal y como se muestra en la *Figura 47,* para conocer su contenido, es decir, para saber si el valor de la variable es igual a '1' o a '0'.

```
% % % % % % % % % % % % % ITEM 7: Auto/Manual % % % % % % % % % % % % %

%Exploración de los items del servidor con el nombre: Loop.1.Main.AutoMan:
variable_7 = serveritems(handles.da, '*Loop.1.Main.AutoMan*');

%Se añade el item encontrado en el grupo de items:
item_7 = additem(handles.grp, variable_7);

r_auto_manual = read(item_7) %Se lee el contenido del item
handles.r_auto_manual = r_auto_manual

%Se accede al campo 'Value' del item leido para conocer su contenido
value_auto_manual = getfield(r_auto_manual, 'Value')

handles.value_auto_manual = value_auto_manual

handles.item_7 = item_7
guidata(hObject,handles)
```

Figura 47. Función Conectar - ITEM_7.

Se emplean estos comandos también para el ítem 9 y 10, que corresponden con las variables de *Límite superior salida operativa* y *Valor salida manual*, respectivamente.

5.1.2 Función *Cargar Configuración*

La segunda función creada es la correspondiente al botón *Cargar Configuración.* Esta función lee el contenido de cada uno de los ítems añadidos con la función *Conectar* para posteriormente mostrarlos por pantalla en el SCADA. Para esto, se emplean los comandos *read* y *getfield*, explicados en el apartado *5.1.1 Función Conectar.* Para mostrar los valores leídos por pantalla, se utiliza el comando *set*, como se muestra en la *Figura 48.* Este comando establece el valor leído en su variable correspondiente del SCADA.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CONFIGURAR da %%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in configurar_da.
function configurar_da_Callback(hObject, eventdata, handles)
% % % % % % % % % % % % % % % % % % ITEM 1: PV % % % % % % % % % % % % % % %
%Se lee el contenido de item_1
r_pv = read(handles.item_1)

%Se accede al campo 'Value' del item leido para conocer su contenido
value_pv = getfield(r_pv, 'Value')

handles.value_pv = value_pv;

%Se establece el valor de value_pv en handles.valor_pv_leido_tanque
set(handles.valor_pv_leido_tanque, 'String' , value_pv);
```

Figura 48. Función Cargar Configuración - ITEM_1.

Se realiza lo mismo para el ítem 2, 3, 4 y 5, que corresponden con los parámetros de *SP, BP, Ti, Td,* respectivamente.

Por otro lado, para el ítem 7 el código es diferente, *Figura 49.* En primer lugar, se lee el valor del ítem 7, que corresponde con el modo de operación del controlador (modo automático/modo manual).

- Si el valor del ítem es igual a ‘0’, el botón del SCADA que corresponde con el modo de operación automático se activa.

- Si, por el contrario, el valor es igual a ‘1’, se activará el botón correspondiente al modo de operación manual.

Si el modo de operación está en manual, se habilita un cuadro de texto en el SCADA para que el operador pueda establecer el valor de la salida que se desee. Para esto, se utilizan los siguientes comandos:

- *get(handles.valor_salida_operativa_forzada, 'Visible').* Se emplea para acceder al campo *'Visible'* de la función: *valor_salida_operativa_forzada.*

- *set(handles.valor_salida_operativa_forzada, 'Visible', 'On').* Se emplea para poner el campo *'Visible'* de la función: *valor_salida_operativa_forzada*, en *'On'.*

Cuando el operador introduce el nuevo valor de la salida, este se transfiere al controlador para actualizar el valor que se encontraba anteriormente en el dispositivo. Esta operación se realiza mediante el comando *write*, como se observa en la *Figura 50.*

```
% % % % % % % % % % % % % % ITEM 7: Auto/Manual % % % % % % % % % % % % %

%Se lee el contenido de item_7
r_auto_manual = read(handles.item_7)

%Se accede al campo 'Value' del item leido para conocer su contenido
value_auto_manual = getfield(r_auto_manual, 'Value')

if value_auto_manual == 0 %Si el valor de value_auto_manual es igual a 0:
    disp('esta en modo auto')
    hObject = handles.modo_automatico %El botón de 'Modo Automático' se activa

    if (hObject ~= 1) %Si el valor del botón no está a 1, se accede al valor para ponerlo a 1
        get(hObject, 'Value')
        set(hObject, 'Value', 1)
    end

else %Si no ocurre lo anterior, el valor de value_auto_manual es igual a 1:
   disp('esta en modo manual')
   hObject = handles.modo_manual%El botón de 'Modo Manual' se activa

   if (hObject ~= 1)%Si el valor del botón no está a 1, se accede al valor para ponerlo a 1
        get(hObject, 'Value')
        set(hObject, 'Value', 1)
   end

    %Se accede al campo 'Visible' y se pone a 'On'
    get(handles.valor_salida_operativa_forzada, 'Visible')
    set(handles.valor_salida_operativa_forzada, 'Visible', 'On')

end
```

Figura 49. Función Cargar Configuración - ITEM_7.

```
%Se lee el contenido de item_10
r_valor_salida_manual = read(handles.item_10)
handles.r_valor_salida_manual = r_valor_salida_manual

%Se accede al campo 'Value' del item leido para conocer su contenido
value_salida_manual = getfield(r_valor_salida_manual, 'Value')
handles.value_salida_manual = value_salida_manual

%Se carga el valor de value_salida_forzada en el item_10
write(handles.item_10, value_salida_forzada)

%Se establece el valor de value_salida_manual en handles.salida_operativa:
set(handles.salida_operativa, 'String' , value_salida_manual)
```

Figura 50. Función Cargar Configuración - comando 'write'

5.1.3 Cuadros de texto

El SCADA cuenta con unos cuadros de texto editables y no editables, *Figura 37. Parámetros del controlador en el SCADA* del apartado *4.4 Estructura del sistema de monitorización y control*, para visualizar los valores de los parámetros del controlador, y modificarlos si el operador lo desea. Las funciones de estos cuadros de texto son muy sencillas.

Por un lado, las funciones de los cuadros de texto no editables, es decir, los cuadros de texto que muestran por pantalla el valor de un parámetro determinado, se basan en dos comandos principales.

- *str2double.* Este comando convierte el contenido de la función de *String* a *Double.*

- *handles.* Devuelve en el SCADA el valor de la función, como se observa en la *Figura 51.*

```
%%%%%%%%%%%%%%%%%%%%%%% TEXTO AMARILLO PV LEIDO %%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes during object creation, after setting all properties.
function valor_pv_leido_tanque_CreateFcn(hObject, eventdata, handles)
%Devuelve el contenido de valor_pv_leido_tanque como String
valor_pv_leido_tanque = str2double(get(hObject, 'String'));

%Devuelve en el SCADA el valor de valor_pv_leido_tanque
handles.valor_pv_leido_tanque = valor_pv_leido_tanque;


guidata(hObject,handles)
```

Figura 51. Cuadro de texto no editable PV.

Por otro lado, las funciones de los cuadros de texto editables, es decir, los cuadros de texto en los que se introduce el nuevo valor de la variable que se desea modificar, *Figura 52*, se basan en los comandos *str2double, handles, set y write*, anteriormente explicados.

```
%%%%%%%%%%%%%%%%%% VALOR SP TANQUE DE AGUA INTRODUCIDO %%%%%%%%%%%%%%%%%%

function valor_sp_tanque_Callback(hObject, eventdata, handles)
%Devuelve el contenido de valor_sp_tanque como String
valor_sp_tanque = str2double(get(hObject, 'String'));

%Devuelve en el SCADA el valor de valor_sp_tanque
handles.valor_sp_tanque = valor_sp_tanque;

%Se establece el valor de valor_sp_tanque en handles.valor_sp_leido_tanque:
set(handles.valor_sp_leido_tanque, 'String' , valor_sp_tanque)

%Se carga el valor de valor_sp_tanque en el item_2
write(handles.item_2, valor_sp_tanque)

guidata(hObject,handles)
```

Figura 52. Cuadro de texto editable SP.

5.1.4 Función *Start*

La siguiente función es la del botón *Start.* Este botón es el encargado de iniciar el seguimiento de la planta mediante un temporizador *Timer*, utilizado para ejecutar comandos o funciones una o varias veces. En este caso, se utiliza para ejecutar la función *configurar_da, que c*omo ya se explicó en el apartado *5.1.2 Función Cargar Configuración,* es la función que lee el contenido de cada uno de los ítems para posteriormente mostrarlos por pantalla en el SCADA. Al ejecutarse toda la función de forma cíclica con el *Timer,* se irán actualizando los valores de los parámetros periódicamente hasta que el operador detenga el seguimiento de forma manual.

En la función *start_timer*, *Figura 53,* se utilizan los siguientes comandos:

- *timer.* Crea el temporizador.
- *set.* Asigna propiedades del temporizador, como ejemplo el periodo de ejecución.
- *start.* Iniciar el timer.

```
%%%%%%%%%%%%%%%%%%%%%%%%% START TIMER %%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in start_timer.
function start_timer_Callback(hObject, eventdata, handles)
disp('antes del timer')
a = timer; %Se crea el timer
handles.a = a

%Se define que el timer se ejecute inmediatamente despues iniciar la
%funcion Callback del timer:

set(handles.a, 'ExecutionMode', 'fixedRate')

%Se define el periodo de ejecucion del timer
set(handles.a, 'Period', 1)

%Se define la funcion a ejecutar para el timer
set(handles.a, 'TimerFcn',{@configurar_da_Callback, handles})

disp('start')
start(handles.a) %Se inicia el timer

disp('despues start')
guidata(hObject,handles)
```

Figura 53. Función Start Timer.

5.1.5 Función *Stop*

Una vez iniciado el temporizador *timer,* es el operador quien decide cuándo parar el seguimiento. Para llevar a cabo la finalización del temporizador, se emplea la función *stop, Figura 54.* Esta función corresponde con el botón de *Stop* del SCADA, y una vez se pulse, el timer se detiene mediante el comando *stop* utilizado en la función.

```
%%%%%%%%%%%%%%%%%%%%%%% BOTON STOP TIMER %%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)

stop(handles.a) %Detiene el timer
```

Figura 54. Función Stop Timer.

5.1.6 Función *Modo Automático* y *Modo Manual*

Para el cambio del modo de control del sistema, es decir, para cambiar el modo de control de 'modo manual' a 'modo automático', se emplean dos funciones diferentes.

En primer lugar, si se desea operar en modo automático, es decir, controlar automáticamente la potencia de salida del controlador, se ejecuta la función *modo_automatico, Figura 55.* Esta función corresponde con el botón 'Modo Automático' del SCADA, y al ejecutarse, se caga en el *ítem_7*, que corresponde al modo de operación del controlador, el valor '0' para que el controlador pase a trabajar en modo automático.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% BOTON AUTOMATICO %%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in modo_automatico.
function modo_automatico_Callback(hObject, eventdata, handles)

modo_automatico = get(hObject,'Value')

get(handles.valor_salida_operativa_forzada, 'Visible')
set(handles.valor_salida_operativa_forzada, 'Visible', 'Off')

write(handles.item_7, 0)

r_limite_sup_salida = read(handles.item_9)
handles.r_limite_sup_salida = r_limite_sup_salida

value_limite_sup_salida = getfield(r_limite_sup_salida, 'Value')
handles.value_limite_sup_salida = value_limite_sup_salida

set(handles.salida_operativa, 'String', value_limite_sup_salida);

handles.modo_automatico = modo_automatico

guidata(hObject,handles)
```

Figura 55. Función Modo Automático.

Si, por el contrario, se desea operar en modo manual, es decir, que el operador controle manualmente la potencia de salida del controlador, se ejecuta la función *modo_manual, Figura 56.* Esta función corresponde con el botón 'Modo Manual' del SCADA, y al ejecutarse, se caga en el *ítem_7*, que corresponde al modo de operación del controlador, el valor '1' para que el controlador pase a trabajar en modo manual.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% BOTON MANUAL %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in modo_manual.
function modo_manual_Callback(hObject, eventdata, handles)

modo_manual = get(hObject,'Value')

get(handles.valor_salida_operativa_forzada, 'Visible');
set(handles.valor_salida_operativa_forzada, 'Visible', 'On');

write(handles.item_7, 1)

r_salida_forzada = read(handles.item_8)

value_salida_forzada = getfield(r_salida_forzada, 'Value')

set(handles.salida_operativa, 'String' , value_salida_forzada)

handles.modo_manual = modo_manual

guidata(hObject,handles)
```

Figura 56. Función Modo Manual.

5.1.7 Función *Apagar* y *Borrar*

Cuando se desee finalizar el seguimiento, la monitorización y el control del sistema, se cuenta con dos botones, *Apagar*, que desconecta el SCADA del controlador, y *Borrar*, que borra el grupo de estándares OPC Data Access. Para la programación del botón *Apagar*, se cuenta con la función *apagar, Figura 57*, compuesta por el comando *disconnect*, encargado de realizar la desconexión entre el cliente y el servidor OPC.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% BOTÓN APAGAR %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in apagar.
function apagar_Callback(hObject, eventdata, handles)


disconnect(handles.da) %Se desconecta el cliente OPC del servidor
```

Figura 57. Función apagar.

Por otro lado, para la programación del botón *Borrar*, se cuenta con la función *borrar, Figura 58*, basada en el comando *delete,* encargado de borrar el OPC Data Access.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% BOTÓN BORRAR %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in borrar_datos.
function borrar_datos_Callback(hObject, eventdata, handles)

delete(handles.da) %Se borra da, es decir, el OPC data access object
```

Figura 58. Función borrar.

5.1.8 Alarma

Para crear la alarma del seguimiento del proceso de control, se emplean tres funciones diferentes. En primer lugar, la función *indicador_alarma*, *Figura 59,* encargada de sacar por pantalla el símbolo 'X' de forma intermitente para indicar al operador que hay un fallo en el sistema.

```
%%%%%%%%%%%%%%%%%%%%%%%%%% INDICADOR ROJO ALARMA %%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes during object creation, after setting all properties.
function indicador_alarma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to indicador_alarma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

Figura 59. Función indicador_alarma.

Por otro lado, se cuenta con la función *texto_alarma*, *Figura 60,* encargada de mostrar por pantalla el mensaje de alarma: '*Alarm: Process Value higher than the Setpoint Value'* cuando el valor de PV supera el de SP.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% TEXTO ALARMA %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes during object creation, after setting all properties.
function texto_alarma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to texto_alarma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

Figura 60. Función texto_alarma.

La tercera función implementada es la encargada de desactivar la alarma, *Figura 61*. Tal y como se explicó en la sección *4.4.1 Alarmas del sistema*, del apartado *4.4 Estructura del sistema de monitorización y control*, el SCADA cuenta con un botón

OK que desactiva la alarma y borra tanto el indicador, como el mensaje mostrado por pantalla, una vez el valor de PV esté por debajo del SP.

```
%%%%%%%%%%%%%%%%%%%%%%%%%% BOTON OK ALARMA %%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in boton_ok_alarma.
function boton_ok_alarma_Callback(hObject, eventdata, handles)

%Se define el campo 'Visible' de las siguientes funciones como 'off':
set(handles.texto_alarma, 'Visible', 'off')
set(handles.indicador_alarma, 'Visible', 'off')
set(handles.boton_ok_alarma, 'Visible', 'off')
```

Figura 61. Función botón_ok_alarma.

6. RESULTADOS DE CONTROL

En este apartado se lleva a cabo el análisis de la influencia de los parámetros en el control del sistema, con el fin de encontrar los valores de estos que mejor se adapten a este, y obtener así, la aproximación al control más óptimo de la planta.

6.1. Resultados del análisis de la influencia de los parámetros en el control

Tal y como se ha mencionado anteriormente, para llevar a cabo el control de la planta de nivel de tanques de este proyecto se ha empleado un control PID mediante el registrador/controlador *nanodac.* Para ello, se han analizado los parámetros: BP, Ti y Td que se emplean para este tipo de control, y así obtener los valores de estos parámetros que mejor se adapten al sistema.

- BP – Banda Proporcional. Este parámetro representa el rango en el cual la salida se puede ajustar linealmente de forma continua para lograr que el error en estado estacionario se aproxime a cero.

  - Cuanto más alto sea el valor de la BP, más lenta será la aproximación del PV al SP. Esto se debe a que la acción proporcional empieza antes, evitando las oscilaciones, pero de esta forma el sistema también se vuelve más lento en su reacción ante perturbaciones.

  - Cuanto más bajo sea el valor de la BP, más rápida será la aproximación del PV al SP, ya que la acción proporcional empieza más tarde, pero el sistema tiende oscilar más.

  - Si BP=0, se elimina la acción proporcional y la regulación del sistema se convierte en un control ON-OFF.

- Ti – Tiempo integral. Este parámetro se emplea para disminuir y eliminar el error en estado estacionario provocado por perturbaciones externas al sistema y que no pueden ser rectificados por el control proporcional.

  - Cuanto mayor sea el valor de Ti, el sistema será más oscilante y menos preciso, en este caso, el valor de PV se desvía más del SP sobrepasando el valor de este.
  - Cuanto menor sea el valor de Ti, el valor de PV tardará más en alcanzar el SP.

- Td – Tiempo derivativo. Este parámetro se emplea para evitar que el error del control incremente, corrigiéndolo proporcionalmente con la misma velocidad que se produce, es decir, se hace más rápido o más lento el incremento del valor del PV para evitar que el error incremente.

  - Cuanto mayor sea el valor de Td, más tardará el PV en alcanzar el SP, es decir, el sistema se hace más lento, pero menos oscilante.

  - Cuanto menor sea el valor de Td, el sistema se hace más rápido, pero más oscilante, es decir, el valor de PV oscilará más para alcanzar el SP.

Para el control de la planta de este proyecto se lleva a cabo un control PI, sin aplicar el parámetro de la acción derivativa, ya que se trata de un sistema con una dinámica lenta con leves perturbaciones externas que alteran mínimamente el error del control. Si se tratase de una planta con una dinámica más rápida, se emplearía un control PID donde el parámetro Td actuaría para evitar que el error del control incrementase tal y como se explicó anteriormente.

Para la realización del análisis de la influencia de los parámetros en el control de la planta, se llevan a cabo seis pruebas. En las tres primeras pruebas, se mantiene el valor del parámetro Ti fijo y se varía el valor del parámetro BP para ver cómo influye este en el control. Y en las tres últimas pruebas, se lleva a cabo el procedimiento contrario, se deja el valor BP fijo y se varía el Ti para ver cómo influye este en el control.

Prueba 1

BP	Ti	Td
9	100	0

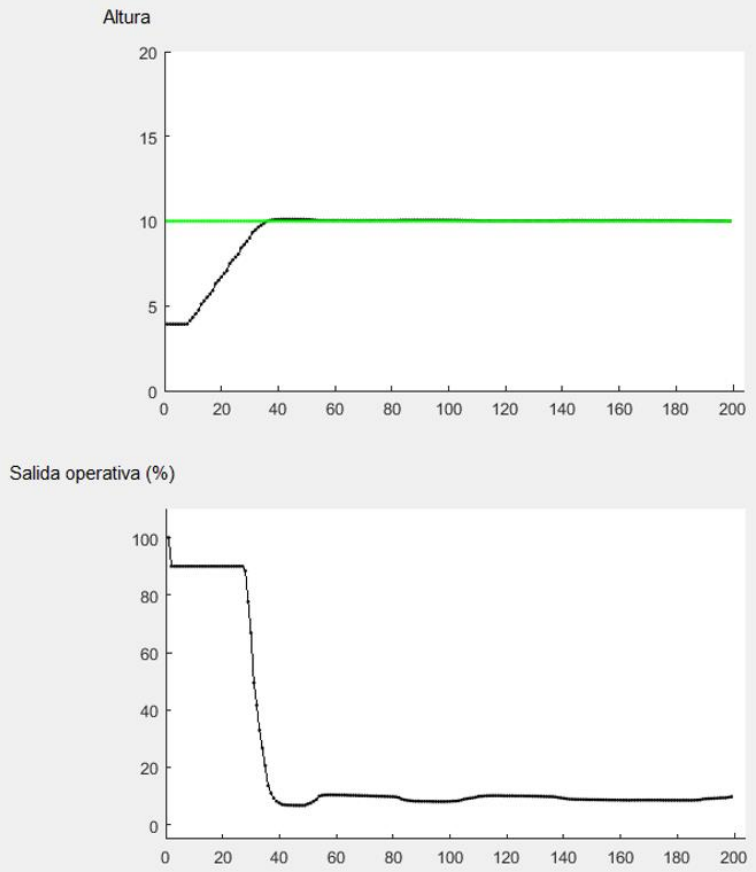

Figura 62. Prueba 1 – Simulación de un PI con BP = 9 y Ti = 100.

En la primera prueba, se ha elegido un valor de BP próximo al valor de SP, en este caso 10. Como se puede observar en la *Figura 62,* el valor de PV llega a la consigna después de una leve oscilación por encima del valor de SP.

Prueba 2

BP	Ti	Td
15	100	0

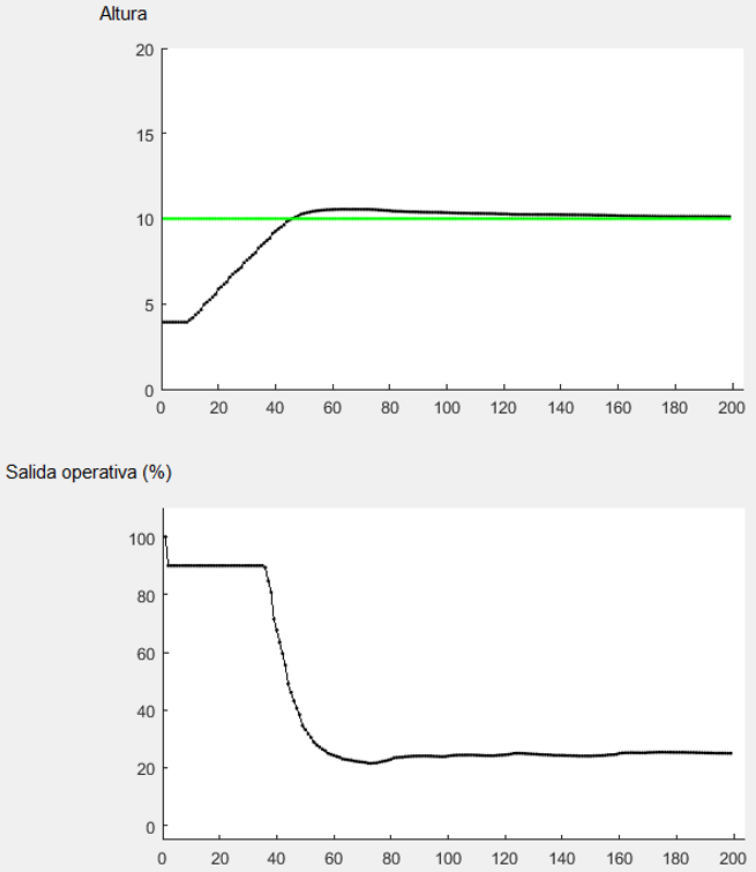

Figura 63. Prueba 2 – Simulación de un PI con BP = 15 y Ti = 100.

En la segunda prueba, se ha elegido un valor de BP por encima del valor de la consigna, en este caso BP = 15. Como se puede observar en la *Figura 63,* el valor de PV sobrepasa la consigna y tarda más en aproximarse al valor deseado.

Prueba 3

BP	Ti	Td
5	100	0

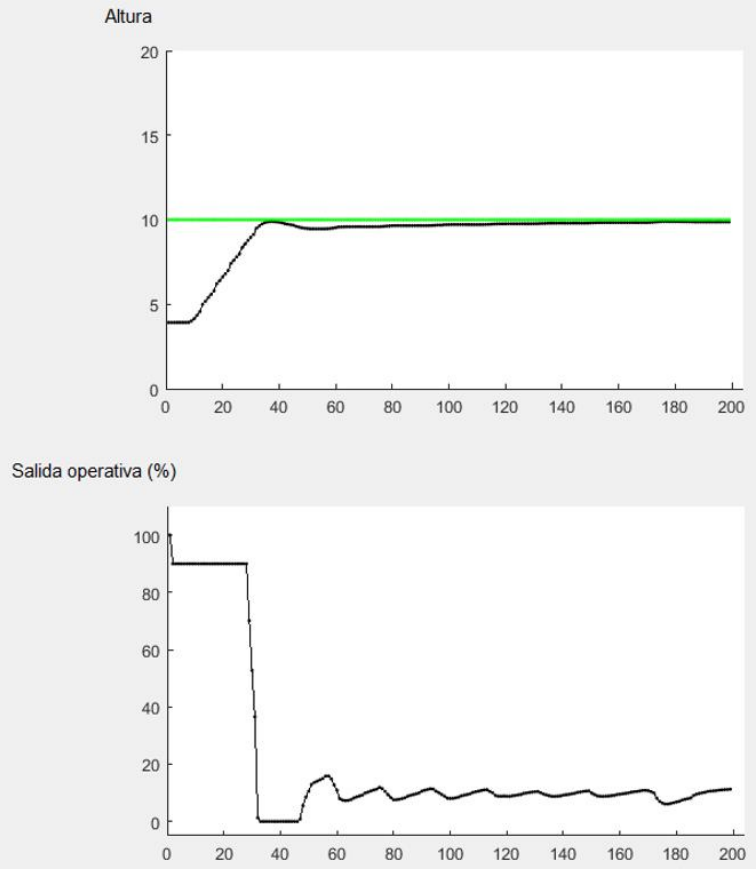

Figura 64. Prueba 3 – Simulación de un PI con BP = 5 y Ti = 100.

En la tercera prueba, se ha elegido un valor de BP muy bajo, pero sin aproximarse a 0, ya que en ese caso se encontraría el sistema en un control del tipo ON-OFF, y no es lo que interesa en este caso. Para una BP = 5, el valor de PV oscila por debajo del valor de SP antes de aproximarse a este, *Figura 64*.

Para analizar mejor las tres primeras pruebas, se representan en una misma gráfica, *Figura 65.*

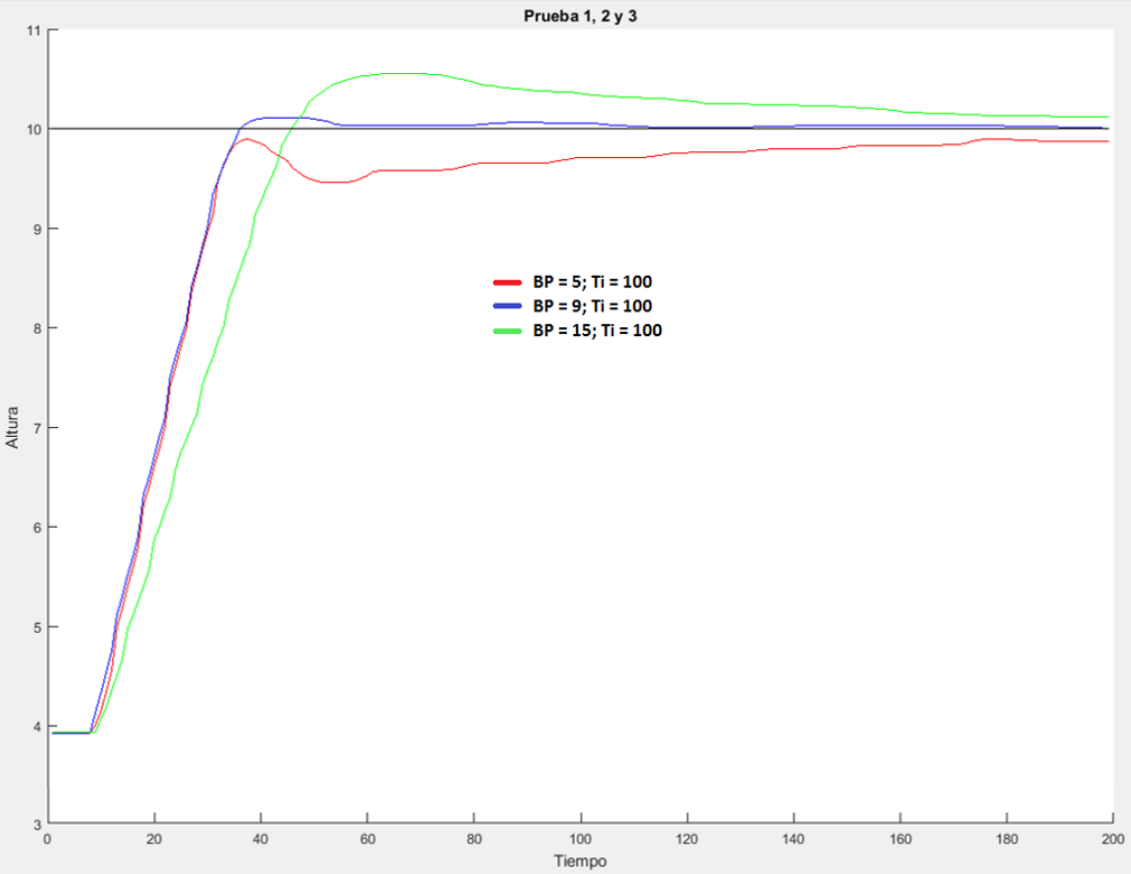

Figura 65. Comparación de pruebas 1, 2 y 3.

De las tres pruebas realizadas, la mejor respuesta es la de color azul, que corresponde con la primera prueba:

BP	Ti	Td
9	100	0

Es la respuesta que alcanza el valor de la consigna de forma más rápida y con menos oscilaciones, es decir, con un error más bajo que el resto de respuestas.

Prueba 4

BP	Ti	Td
9	10	0

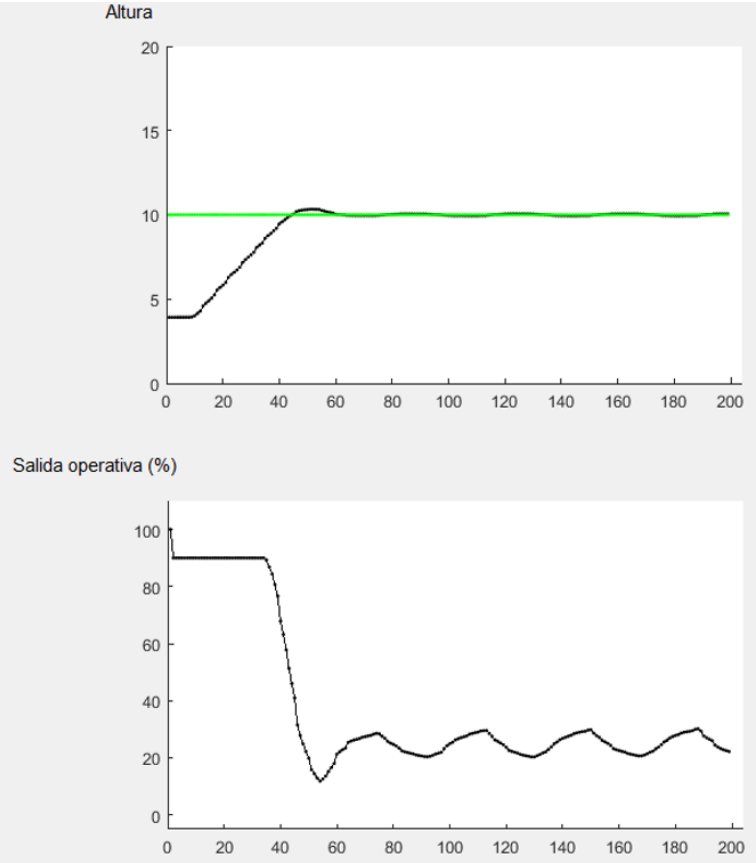

Figura 66. Prueba 4 – Simulación de un PI con BP = 9 y Ti = 10.

En la cuarta prueba, *Figura 66,* se observa que, con un valor muy pequeño de Ti, el valor de PV supera el valor de la consigna y no llega a estabilizarse. Con estos valores, el PV está en continua oscilación en valores próximo al de SP.

Prueba 5

BP	Ti	Td
9	500	0

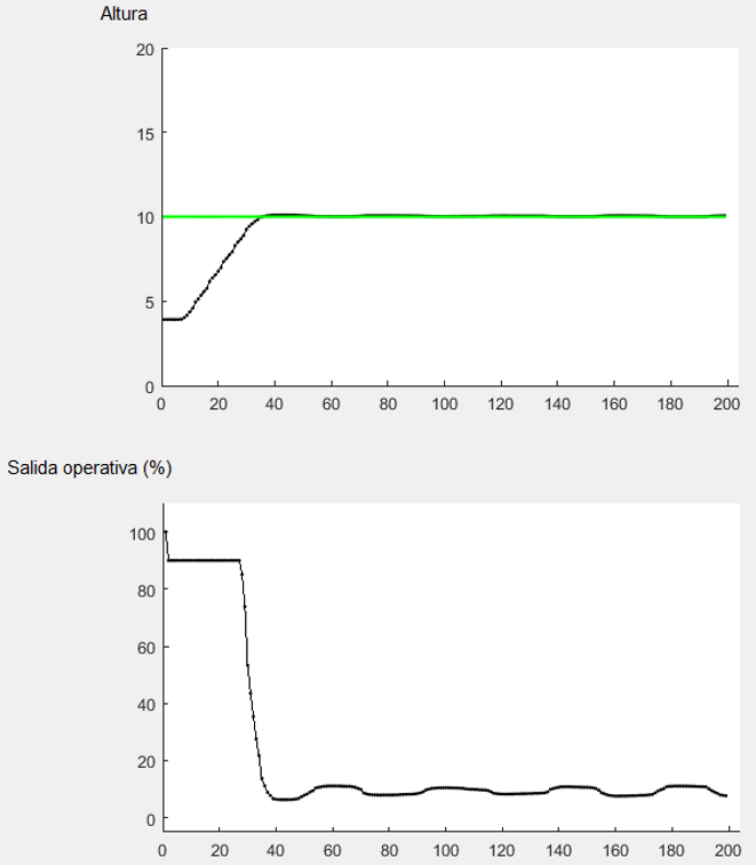

Figura 67. Prueba 5 – Simulación de un PI con BP = 9 y Ti = 500.

En la quinta prueba, *Figura 67,* se observa que, con un valor de Ti no muy elevado, el valor de PV tampoco llega a estabilizarse y está en continua oscilación, pero de esta forma son menos pronunciadas que en la prueba 4.

Prueba 6

BP	Ti	Td
9	3000	0

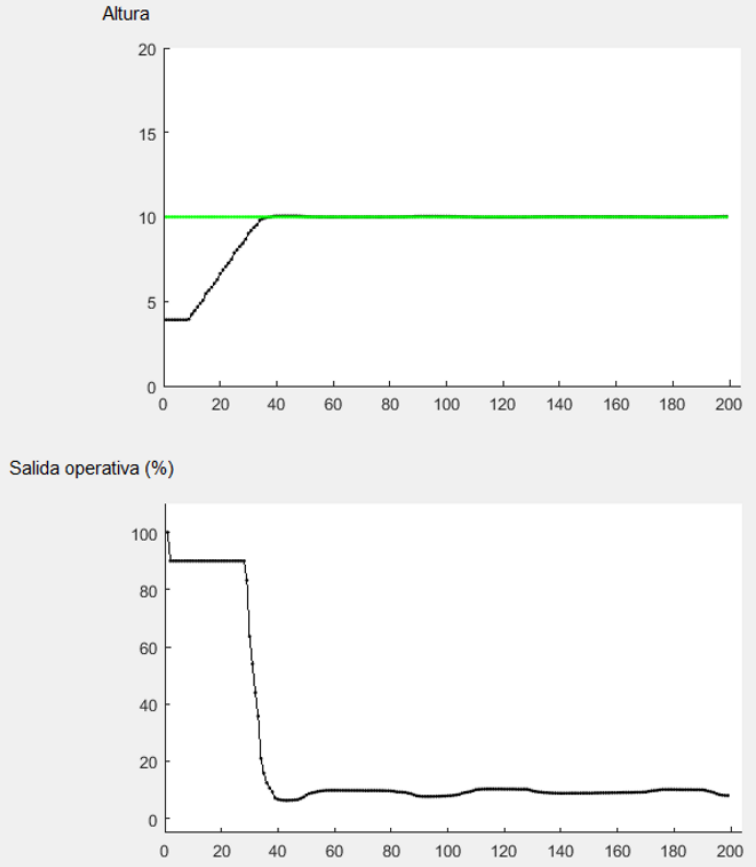

Figura 68. Prueba 6 – Simulación de un PI con BP = 9 y Ti = 3000.

Por último, en la sexta prueba, *Figura 68,* las oscilaciones del valor de PV son cada vez menos pronunciadas llegando finalmente a un valor más próximo y estable del SP.

Para analizar mejor las tres últimas pruebas, se representan en una misma gráfica, *Figura 69.*

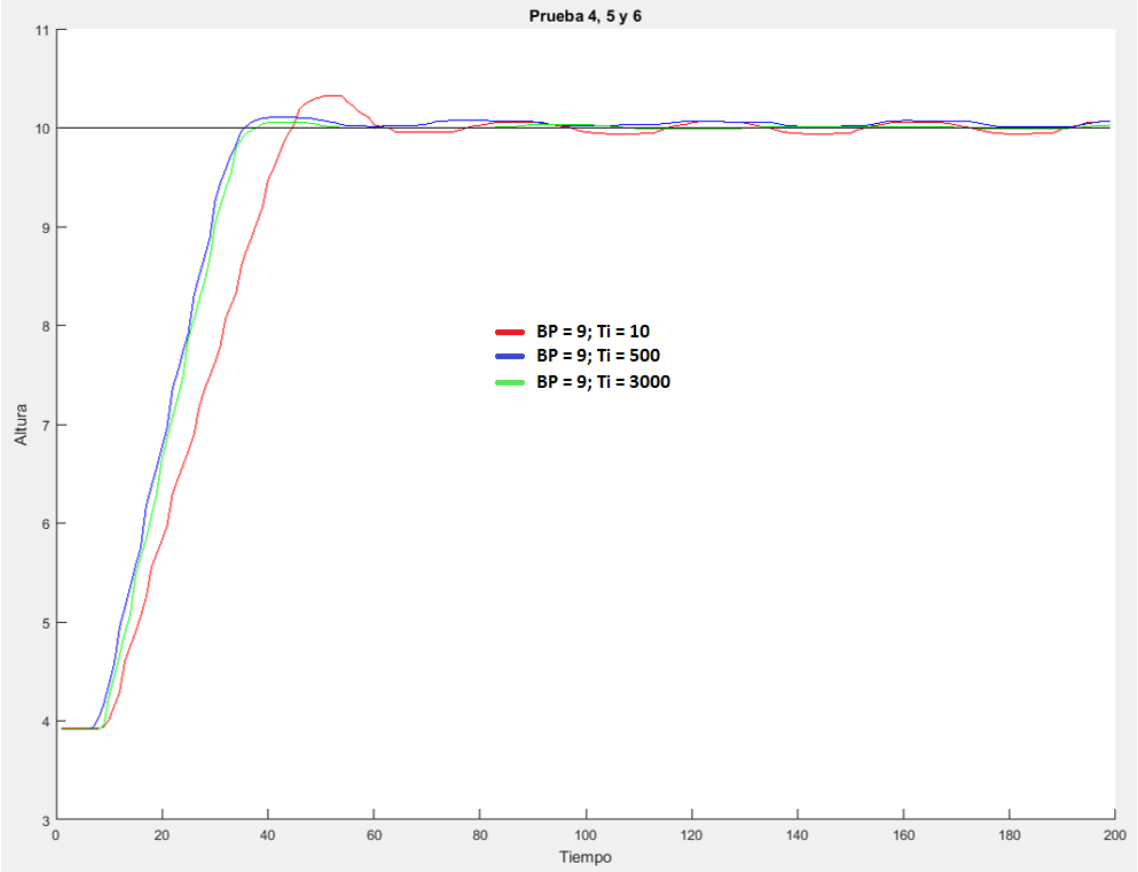

Figura 69. Comparación de pruebas 4, 5 y 6.

De las tres pruebas realizadas, la mejor respuesta es la de color verde, que corresponde con la tercera prueba:

BP	Ti	Td
9	3000	0

Es la respuesta que más se acerca al valor de la consigna deseada y con oscilaciones menos pronunciadas que el resto, es decir, es la respuesta con menor error de las tres.

6.2. Resultados de la implementación del control

Con el fin de conseguir un control lo más óptimo posible, se han estudiado los efectos de cada uno de los parámetros que intervienen en este en el apartado *6.1 Resultados del análisis de la influencia de los parámetros en el control.* En este apartado, se han realizado tres últimas pruebas, *Figura 70,* con el objetivo de alcanzar los valores de los parámetros BP y Ti más óptimos para el control del sistema.

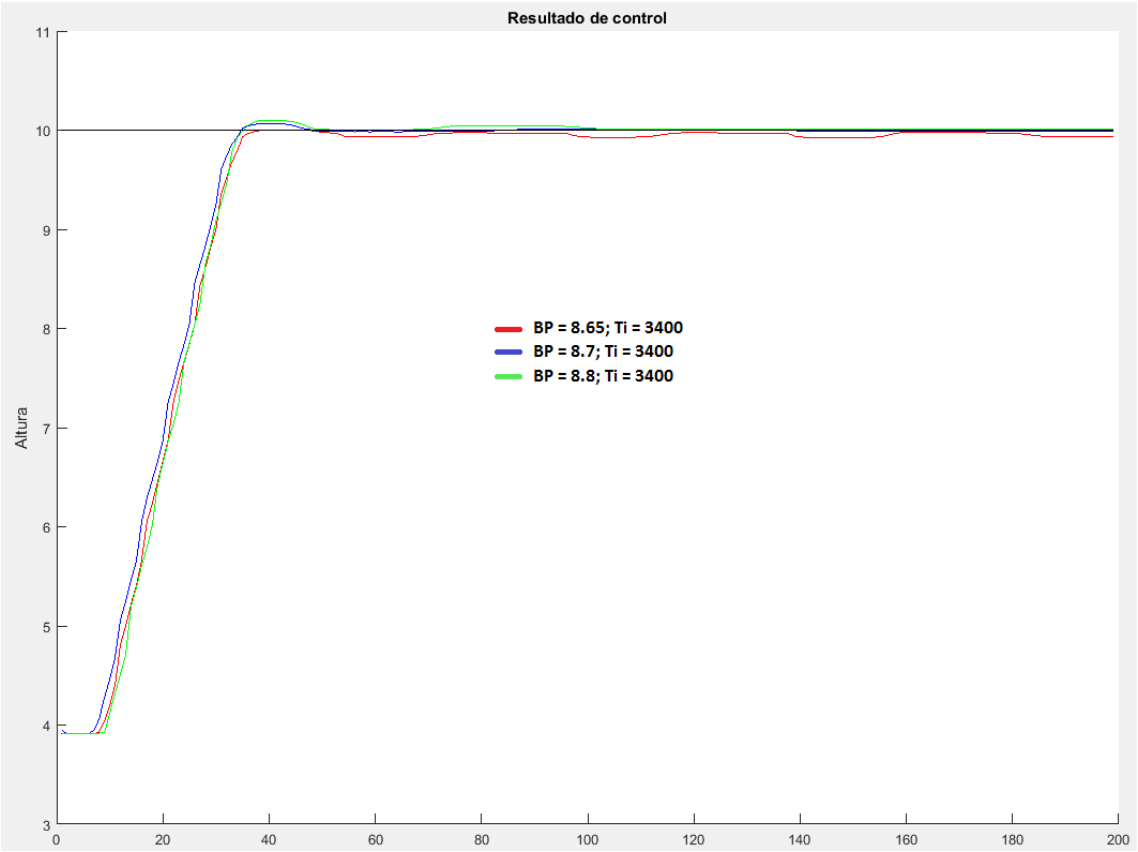

Figura 70. Resultado de control.

Finalmente, se ha llegado a la conclusión de que los parámetros que más se aproximan al control óptimo del sistema son los siguientes.

BP	Ti	Td
8.7	3400	0

7. PRESUPUESTO

En este apartado, se analiza el plan de trabajo para la realización de este proyecto con el fin de estimar el presupuesto necesario para llevarlo a cabo. Para ello, se ha dividido el presupuesto en tres partes diferentes: coste de los materiales, coste de la mano de obra y coste total del proyecto.

7.1. Coste de los materiales

Descripción	Cantidad (ud)	Coste unitario (€/ud)	Coste total (€)
Registrador/controlador nanodac	1	1.100	1.100
Resistencia	1	0,10	0,10
Cables de hilo unipolar	4	0,30	1,20
Cables banana banana	4	6	24
Cable de Ethernet	1	7	7
Monitor de ordenador	1	70	70
Ordenador	1	500	500
Teclado de ordenador	1	20	20
Ratón de ordenador	1	10	10
Coste total de materiales			**1.732,3**

7.2. Coste de mano de obra

Concepto	Cantidad (h)	Coste unitario (€/h)	Coste total (€)
Tiempo de análisis de la planta	2	15	30
Tiempo de instalación	6	15	90
Tiempo de configuración	60	15	900
Tiempo de elaboración del programa para el sistema de monitorización y control	200	15	3.000
Tiempo de realización de pruebas	40	15	600
Coste total de mano de obra			**4.620**

7.3. Coste total del proyecto

Concepto	Coste total (€)
Coste total de materiales	1.732,3
Coste total de mano de obra	4.620
Coste total del proyecto	**6.352,3**

8. CONCLUSIONES

El objetivo principal de este proyecto ha sido el estudio de la solución para la monitorización y control de una planta de nivel de tanques mediante un controlador industrial. Para ello, se han empleado los tres primeros niveles de la pirámide de la automatización: nivel de campo, nivel de control y nivel de supervisión.

En primer lugar, en el nivel de campo se ha estudiado la planta a controlar, analizando cada una de las entradas y salidas del sistema para determinar posteriormente las variables que influyen en el control de la misma.

Por otro lado, en el nivel de control se ha analizado el proceso de control a llevar a cabo y se ha instalado y configurado el dispositivo de control empleado, el resgistrador/controlador *nanodac* de *Eutotherm-Schneider Electric*. Para establecer la conexión entre el nivel de campo y el nivel de control se han utilizado cables banana banana conectados por un extremo a la planta y por el otro a diferentes cables de hilo unipolar que a su vez están conectados al *nanodac.*

Por último, se ha diseñado y desarrollado el sistema de monitorización y control en el nivel de supervisión. Para ello, se ha diseñado un sistema SCADA utilizando la herramienta MATLAB. El SCADA diseñado es capaz de mostrar el rendimiento de la planta en tiempo real, así como leer y modificar los valores del dispositivo de control que influyen en el sistema. Para establecer la comunicación entre el nivel de control y el nivel de supervisión, se ha utilizado el protocolo TCP ModBus. Para la conexión entre estos dos niveles se ha empleado un cable de Ethernet estándar entre el conector RJ45 del registrador/controlador nanodac y el ordenador. Para la transferencia de los datos entre el controlador y el servidor OPC, para poder leer y/o modificar los valores del *nanodac* desde el SCADA se ha utilizado el protocolo de transferencia de archivos, FTP.

Una vez finalizado el sistema de monitorización y control de la planta, se ha estudiado el efecto de los parámetros del PID en el sistema de control con el fin de encontrar los valores de estos que mejor se adapten al control de la planta. Para finalizar el trabajo, se ha estimado el presupuesto necesario para la realización de este proyecto.

Conclusions

The main objective of this project has been the study of the solution for the monitoring and control of a tank level plant using an industrial controller. For this, the first three levels of the automation pyramid have been used: field level, control level and supervision level.

First, at the field level, the plant to be controlled was studied, analyzing each of the inputs and outputs of the system to later determine the variables that influence in the control of this system.

On the other hand, at the control level, the control process has been analyzed and the control device used, the nanodac controller of Eutotherm-Schneider Electric, has been installed and configured. To establish the connection between the field level and the control level banana banana cables have been used. These cables are connected at one end to the plant and on the other to different unipolar wire cables that are connected at the same time to the nanodac.

Finally, the monitoring and control system at the supervisory level has been designed and developed. To do this, a SCADA system has been designed using the MATLAB tool. The SCADA designed shows the performance of the plant in real time, and it can read and modify the values of the control device that influence the system as well. To establish communication between the control level and the supervision level, the ModBus TCP protocol has been used. For the connection between these two levels, a standard Ethernet cable has been used between the RJ45 connector of the nanodac controller and the computer. For the transfer of data between the controller and the OPC server, in order to read and / or modify the nanodac values from the SCADA, the file transfer protocol, FTP, has been used.

Once the monitoring and control system of the plant has been completed, the effect of the PID parameters in the control system have been studied in order to find the values that adapt better to the control of the plant. To finish the work, the necessary budget for the realization of this project has been estimated

9. BIBLIOGRAFÍA

- SCADA: https://es.wikipedia.org/wiki/SCADA
- HMI: https://es.wikipedia.org/wiki/Interfaz_de_usuario
- MES:
- ERP: https://es.wikipedia.org/wiki/Sistema_de_planificación_de_recursos_empresariales
- PID: https://es.wikipedia.org/wiki/Controlador_PID
- OPC: https://es.wikipedia.org/wiki/OPC
- OLE: https://es.wikipedia.org/wiki/Object_Linking_and_Embedding
- FTP: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_archivos
- Arquitectura cliente-servidor: https://es.wikipedia.org/wiki/Cliente-servidor
- Modbus: https://es.wikipedia.org/wiki/Modbus
- Válvula de control: https://es.wikipedia.org/wiki/Válvula_de_control
- CPU: https://es.wikipedia.org/wiki/Unidad_central_de_procesamiento
- IDE: https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado
- Diferencias entre PLCs y PACs: http://www.machinedesign.com/mechanical/what-s-difference-between-plc-and-pac
- Pirámide de la automatización: https://www.smctraining.com/webpage/indexpage/311/
- Ingeniería de la Automatización Industrial. Autor: Ramón Piedrafita Moreno
- Figura PLC: https://upload.wikimedia.org/wikipedia/commons/f/f3/Siemens_Simatic_S7-416-3.jpg
- Figura PAC: http://www.machinedesign.com/mechanical/what-s-difference-between-plc-and-pac
- Comparación entre PLCs y PACs: http://www.logicelectronic.com/BECKHOFF/Que%20es%20un%20PAC.htm
- Figura *Problema sin OPC*: https://es.wikipedia.org/wiki/OPC#/media/File:PROBLEOPC.JPG
- Figura *Solución con OPC*: https://es.wikipedia.org/wiki/OPC#/media/File:SOLUCIOPC.JPG
- Guía del usuario de nanodac: file:///E:/nanodac%20Guía%20del%20usuario%20(HA030554SPA%202).pdf
- OPC UA: http://larraioz.com/articulos/opc-desde-el-clasico-al-nuevo-opc-ua

10. ANEXOS

10.1 Especificaciones técnicas del registrador/controlador nanodac

10.1.1 Categoría de instalación y grado de contaminación

Este producto ha sido diseñada de acuerdo con BS EN61010 para categoría de instalación II y grado de contaminación 2, que se definen como sigue:

Categoría de instalación II

La tensión nominal impulsiva para equipos con alimentación nominal de 230 V es de 2.500 V.

Grado de contaminación 2

Normalmente sólo se genera contaminación no conductiva. No obstante, en ocasiones se debe esperar una conductividad temporal causada por condensación.

10.1.2 Especificaciones del registrador

Tipos E/S		
	Entrada analógica	Cuatro
	E/S digital	Una
	Entrada digital	Dos
	Salida de relé (estándar)	Dos + dos con conexión común compartida, o bien
	Salida de relé (con opción de salida CC)	Una + dos con conexión común compartida
	Salida CC:	Una
Características		Formato de archivos CSV Protocolo de transferencia de archivos (FTP) Mensajes TCP ModBus esclavo Compresión uhh (archivo de historial) Puerto USB en la parte posterior del dispositivo Tablas de linealización de usuario (cuatro) Dos lazos de control (opcional) Compatibilidad con sonda de zirconio (opcional) 14 canales virtuales (cada uno de ellos configurable como canal matemático, totalizador o contador)
Especificaciones ambientales		
Temperatura ambiente		
	Operación:	0 a 55°C
	Almacenamiento:	-20 a +70°C
Humedad relativa	Operación:	5% a 85% sin condensación
	Almacenamiento:	5% a 85% sin condensación
Protección	Panel delantero	IP65, NEMA4X (internacional)
	Panel posterior:	IP10 (internacional)
Golpes/vibraciones		Según BS EN61131-2 (5 a 150 Hz a 1 G; 1 octava por minuto)
Altitud		<2.000 metros
Atmósfera		No apto para uso en atmósferas explosivas o corrosivas
Seguridad eléctrica		BS EN61010-1 (categoría de instalación II; grado de contaminación 2)
Compatibilidad electromagnética		
	Emisiones (unidades estándar):	BS EN61326 Límite B - Industrial ligera
	Emisiones (opción de baja tensión):	BS EN61326 Límite A - Industrial pesada
	Inmunidad	BS EN61326 Industrial
Otras certificaciones y normativas		
	General:	CE y cUL, EN61010
	Entrada PV	Según AMS2750D
	RoHS	UE; China
Embalaje		BS EN61131-2 sección 2.1.3.3.
Especificaciones físicas		
Montaje en panel		1/4 DIN
Peso		0,44 kg
Dimensiones de corte del panel		92 mm x 92 mm (ambos -0,0 +0,8) (Figura 2.1)
Profundidad por detrás del panel		90 mm (Figura 2.1) sin cables
Interfaz de operador		
Pantalla		TFT en color de 3" (320 píxeles de ancho x 240 píxeles de alto)
Controles		Cuatro botones de navegación debajo de la pantalla (Página, Desplazamiento, Bajar y Subir)

Alimentación eléctrica		
Tensión	Estándar:	100 a 230 V CA ± 15% ((48 a 62Hz))
	Opción de baja tensión:	24 V CA (+20% - 15%) (48 a 62 Hz), o bien 24 V CC (+10% -20%)
Disipación de potencia		9 W
Tipo de fusible		Ninguno
Protección contra interrupción		
	Estándar:	Retención >10 ms a 85 V RMS de alimentación
	Opción de baja tensión:	Retención >10 ms a 20,4 V RMS de alimentación

Batería	
Datos en memoria	Valores de hora, fecha
Autonomía (para reloj de tiempo real)	Mínimo 1 año con la unidad sin alimentación
Período de sustitución	Tres años (típico)
Tipo	Monofluoruro de policarbonato/litio (BR2330) (PA260195)

Comunicaciones Ethernet	
Tipo	10/100baseT Ethernet (IEEE802.3)
Protocolos	TCP/IP ModBus esclavo, FTP, DHCP
Tipo de cable	Categoría 5
Longitud máxima	100 metros
Terminación	RJ45; LED verde encendido = conectado; LED ámbar intermitente = actividad

Puerto USB	
Número de puertos	Uno en la parte posterior del dispositivo
Estándar	USB1.1
Velocidad de transferencia	1,5 Mbits/s (dispositivo de baja velocidad)
Corriente máxima	< 100 mA
Periféricos compatibles	Unidad de memoria (8 GB máx.)

Frecuencias de actualización/copia	
Frecuencia de muestreo (entrada/salida)	8 Hz
Actualización de representación	8 Hz máx.
Valor de muestreo de archivos	Último valor en el momento de la copia
Valor en pantalla	Último valor en el momento de la actualización

10.1.3 Especificaciones de entradas analógicas

Especificaciones generales		
Número de entradas analógicas		Cuatro
Tipos de entradas		V CC, mV CC, mA CC (requiere derivación externa), termopar, RTD (2 y 3 hilos), digital (cierre de contacto)
Combinación de tipos de entradas		Configurable a voluntad
Frecuencia de muestreo		8 Hz (125 ms)
Método de conversión		16 bits delta sigma
Rangos de entrada		Véase a continuación
Rechazo de interferencias (48 a 62 Hz)		
	Modo serie:	> 95 dB
	Modo común:	> 179 dB
Tensión en modo común		250 V CA máx.
Tensión en modo común		280 mV en rango mínimo; 5 Vp-p en rango máximo
Impedancia de entrada		Véanse las especificaciones de rango a continuación
Protección contra sobretensión	Continua:	±30 V RMS
	Transitoria (< 1 ms):	±200 Vp-p entre terminales
Detección de desconexión de sensor	Tipo:	Desconexión CA de sensor en cada entrada, con respuesta rápida y sin errores CC asociados
	Tiempo de reconocimiento:	<3 segundos
	Resistencia mínima de desconexión:	Rangos de 40 mV y 80 mV: 5 kΩ; otros rangos: 12,5 kΩ
Derivación (sólo entradas de mA)	Valores:	1Ω a 1kΩ, montaje externo
	Error adicional debido a derivación:	0,1% de la entrada
Aislamiento	Entre canales:	300 V RMS o CC (doble aislamiento)
	Entre canal y circuitos electrónicos:	300 V RMS o CC (doble aislamiento)
	Entre canal y tierra:	300 V RMS o CC (doble aislamiento)
Resistencia dieléctrica	Prueba:	BS EN61010, prueba tipo de 1 minuto
	Entre canales:	2.500 V CA
	Entre canal y tierra:	1.500 V CA

Rangos de entrada CC		
Rangos		40 mV, 80 mV, 2 V; 10 V (-4,0 a +10 V)
Rango de 40 mV	Rango:	-40 mV a +40 mV
	Resolución:	1,9 µV (sin filtrar)
	Ruido de la medida:	1,0 µVp-p con filtro de entrada de 1,6 s
	Error de linealización:	0,003% (línea recta con mejor ajuste)
	Error de calibración:	±4,6 µV ± 0,053% de la medida a 25°C
	Coeficiente de temperatura:	±0,2 µV/°C ± 13 ppm/°C de la medida a partir de 25°C
	Corriente de fugas de entrada:	±14 nA
	Resistencia de entrada:	100 MΩ
Rango de 80 mV	Rango:	-80 mV a +80 mV
	Resolución:	3,2 µV (sin filtrar)
	Ruido de la medida:	3,3 µVp-p con filtro de entrada de 1,6 s
	Error de linealización:	0,003% (línea recta con mejor ajuste)
	Error de calibración:	±7,5 µV ± 0,052% de la medida a 25°C
	Coeficiente de temperatura:	±0,2 µV/°C ± 13 ppm/°C de la medida a partir de 25°C
	Corriente de fugas de entrada:	±14 nA
	Resistencia de entrada:	100 MΩ
Rango de 2 V	Rango:	±2 V
	Resolución:	82 µV
	Ruido de la medida:	90 µVp-p con filtro de entrada de 1,6 s
	Error de linealización:	0,003% (línea recta con mejor ajuste)
	Error de calibración:	±420 µV ± 0,044% de la medida a 25°C
	Coeficiente de temperatura:	±125 µV/°C ± 13 ppm/°C de la medida a partir de 25°C
	Corriente de fugas de entrada:	±14 nA
	Resistencia de entrada:	100 MΩ
Rango de 10 V	Rango:	-3 V a +10 V
	Resolución:	500 µV
	Ruido de la medida:	550 µVp-p con filtro de entrada de 1,6 s
	Error de linealización:	0,007% (línea recta con mejor ajuste) para resistencia interna nula; sume 0,003% por cada 10 O de resistencia interna y de carga
	Error de calibración:	±1,5 mV ± 0,063% de la medida a 25°C
	Coeficiente de temperatura:	±66 µV/°C ± 45 ppm/°C de la medida a partir de 25°C
	Resistencia de entrada:	62,5 kΩ para tensiones de entrada > 5,6 V; 667 kΩ para tensiones de entrada < 5,6 V

Rangos de entrada de resistencia

Escala de temperatura	ITS90
Tipos, rangos y precisiones de RTD	Véase la tabla A3a
Corriente interna máxima	200 µA

Entrada de resistencia

Rango:	0 a 400 Ω (-200 a +850°C)
Resolución:	0,05°C
Ruido de la medida:	0,05°C p-p con filtro de entrada de 1,6 s
Error de linealización:	0,0033% (línea recta con mejor ajuste)
Error de calibración:	±0,3°C ± 0,023% de la medida en °C a 25°C
Coeficiente de temperatura:	±0,01 °C/°C ± 25 ppm/°C de la medida en °C a partir de 25°C
Resistencia de carga	0 a 22 Ω, resistencias de carga iguales
Corriente de bulbo:	200 µA nominales

Tipo de RTD	Rango global (°C)	Estándar	Error máx. de linealización
Cu10	-20 a + 400	General Electric Co.	0,02°C
Cu53	-70 a + 200	RC21-4-1966	< 0,01°C
JPT100	-220 a + 630	JIS C1604:1989	0,01°C
Ni100	-60 a + 250	DIN43760:1987	0,01°C
Ni120	-50 a + 170	DIN43760:1987	0,01°C
Pt100	-200 a + 850	IEC751	0,01°C
Pt100A	-200 a + 600	Eurotherm Recorders SA	0,09°C

Tabla A3a Tipos de RTD

Datos de termopares

Escala de temperatura		ITS90
CJC	Tipos:	Off, interno, externo, remoto
	Fuente de CJC remota:	Un canal de entrada o matemático
	Error de CJC interna:	< 1°C máx. con el dispositivo a 25°C
	Tasa de rechazo de CJC interna:	40:1 a partir de 25°C
Impulso hacia arriba/abajo en la escala		Alto, bajo o ninguno, configurable por separado para la detección de desconexión de sensor en cada canal
Tipos, rangos y precisiones		Véase la tabla A3b

Tipo de termopar	Rango global (°C)	Estándar	Error máx. de linealización
B	0 a + 1.820	IEC584.1	0 a 400°C = 1,7°C 400 a 1.820°C = 0,03°C
C	0 a + 2.300	Hoskins	0,12°C
D	0 a + 2.495	Hoskins	0,08°C
E	-270 a + 1.000	IEC584.1	0,03°C
G2	0 a + 2.315	Hoskins	0,07°C
J	-210 a + 1.200	IEC584.1	0,02°C
K	-270 a + 1.372	IEC584.1	0,04°C
L	-200 a + 900	DIN43710:1985 (hasta IPTS68)	0,02°C
N	-270 a + 1.300	IEC584.1	0,04°C
R	-50 a + 1.768	IEC584.1	0,04°C
S	-50 a + 1.768	IEC584.1	0,04°C
T	-270 a + 400	IEC584.1	0,02°C
U	-200 a + 600	DIN43710:1985	0,08°C
NiMo/NiCo	-50 a + 1.410	ASTM E1751-95	0,06°C
Platinel	0 a + 1.370	Engelhard	0,02°C
Mi/NiMo	0 a + 1.406	Ipsen	0,14°C
Pt20%Rh/Pt40%/Rh	0 a + 1.888	ASTM E1751-95	0,07°C

Tabla A3b Tipos, rangos y precisiones de termopares

10.1.4 Especificaciones de relés y E/S lógica

Especificaciones de relés y E/S lógicas OP1 y OP2

Salida lógica con fuente de corriente activa (corriente On)	
Salida de tensión en los terminales	+11 V mín.; +13 V máx.
Corriente de salida en cortocircuito	6 mA mín. (estado estacionario); 44 mA máx. (corriente de conmutación)
Salida lógica con fuente de corriente inactiva I/O1 (corriente Off)	
Salida de tensión en los terminales	0 V mín.; 300 mV máx.
Corriente de fugas de salida en cortocircuito	0 µA mín.; 100 µA máx.
Entrada lógica con fuente de cierre de contacto activa I/O1 (corriente On)	
Corriente de entrada Entrada a 12 V:	0 mA mín.; 44 mA máx.
Entrada a 0 V:	6 mA mín. (estado estacionario); 44 mA máx. (corriente de conmutación)
Tensión de entrada en circuito abierto	11 V mín.; 13 V máx.
Resistencia en circuito abierto (inactiva)	500 Ω mín.; ∞ máx.
Resistencia en circuito cerrado (activa)	0 Ω mín.; 150 Ω máx.
Contactos de relé	
Potencia de conmutación de contacto (resistiva)	Máx.: 2 A a 230 V RMS ± 15%; Mín.: 100 mA @ 12 V
Corriente máxima en los terminales	2 A

10.1.5 Entradas digitales

DigInA, DigInB, entrada lógica de cierre de contacto

Cierre de contacto	
Corriente de detección en cortocircuito (fuente)	5,5 mA mín.; 6,5 mA máx.
Resistencia en circuito abierto (inactiva)	600 Ω mín.; ∞ máx.
Resistencia en circuito cerrado (activa)	0 Ω mín.; 300 Ω máx.

10.1.6 Salidas CC (Opción)

Salidas analógicas OP3DC

Salidas de corriente	
Rangos de salida	Configurable entre 0 y 20 mA
Resistencia de carga	500 O máx.
Precisión de calibración	<±100 µA ±1% de la medida
Salidas de tensión	
Rango de salida	Configurable entre 0 y 10 V
Impedancia de salida	500 O mín.
Precisión de calibración	<±50 mV ±1% de la medida
Especificaciones generales	
Aislamiento	300 V CA con doble aislamiento del instrumento y otras E/S
Resolución	>11 bits
Deriva térmica	<100 ppm/°C

10.2 Código del sistema de monitorización y control en Matlab

```
function varargout = SCADA(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SCADA_OpeningFcn, ...
                   'gui_OutputFcn',  @SCADA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end



function SCADA_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
guidata(hObject, handles);



function varargout = SCADA_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;



function logo_ULL_CreateFcn(hObject, eventdata, handles)

xticks([])
yticks([])
imshow(['logoull.png'])



function conectar_da_Callback(hObject, eventdata, handles)

global h_pv;
global cont_xpv;
global h_u;
global h_sp;

global guardar_altura_pv;
global guardar_altura_sp;
global guardar_eje_tiempo;

guardar_altura_pv = [];
guardar_altura_sp = [];
guardar_eje_tiempo = [];
```

```
h_pv = animatedline(handles.grafica_altura_tanque,'Marker','.');
h_u = animatedline(handles.grafica_salida_operativa,'Marker','.');
h_sp = animatedline(handles.grafica_altura_tanque,'Color','g',
'Marker','.');

cont_xpv = 1;

handles.valor_leido_periodo = 1;


IP = '169.254.85.136';
IP2 = 'localhost';
serverID = 'Eurotherm.ModbusServer.1'

opcreset;

hostInfo = opcserverinfo(IP2)
allServers = hostInfo.ServerID
da = opcda(IP2, serverID)
set(da, 'EventLogMax', inf);

connect(da);

handles.da = da
guidata(hObject,handles)

grp = addgroup(handles.da)

handles.grp = grp
guidata(hObject,handles)


allitems = serveritems(handles.da, '*');

variable_1 = serveritems(handles.da, '*Loop.1.Main.PV*');
item_1 = additem(handles.grp, variable_1);
handles.item_1 = item_1
guidata(hObject,handles)

variable_2 = serveritems(handles.da, '*Loop.1.SP.SP1*');
item_2 = additem(handles.grp, variable_2);
handles.item_2 = item_2
guidata(hObject,handles)

variable_3 = serveritems(handles.da, '*Loop.1.PID.ProportionalBand*')
item_3 = additem(handles.grp, variable_3);
handles.item_3 = item_3
guidata(hObject,handles)

variable_4 = serveritems(handles.da, '*Loop.1.PID.IntegralTime*');
item_4 = additem(handles.grp, variable_4);
handles.item_4 = item_4
guidata(hObject,handles)
```

```
variable_5 = serveritems(handles.da, '*Loop.1.PID.DerivativeTime*');
item_5 = additem(handles.grp, variable_5);
handles.item_5 = item_5
guidata(hObject,handles)

variable_6 = serveritems(handles.da, '*Loop.1.OP.Ch1Out*');
item_6 = additem(handles.grp, variable_6);
handles.item_6 = item_6
guidata(hObject,handles)

variable_7 = serveritems(handles.da, '*Loop.1.Main.AutoMan*');
item_7 = additem(handles.grp, variable_7);
r_auto_manual = read(item_7)
handles.r_auto_manual = r_auto_manual;
value_auto_manual = getfield(r_auto_manual, 'Value')
handles.value_auto_manual = value_auto_manual;
handles.item_7 = item_7
guidata(hObject,handles)

if (value_auto_manual == 0)
    set(handles.modo_automatico, 'Value', 1)
else
    set(handles.modo_manual, 'Value', 1)
end

variable_8 = serveritems(handles.da, '*Loop.1.OP.ForcedOP*');
item_8 = additem(handles.grp, variable_8);
handles.item_8 = item_8
guidata(hObject,handles)

variable_9 = serveritems(handles.da, '*Loop.1.OP.OutputHighLimit*');
item_9 = additem(handles.grp, variable_9);
handles.item_9 = item_9
guidata(hObject,handles)

if value_auto_manual == 0

    r_limite_sup_salida = read(handles.item_9)
    handles.r_limite_sup_salida = r_limite_sup_salida

    value_limite_sup_salida = getfield(r_limite_sup_salida, 'Value')
    handles.value_limite_sup_salida = value_limite_sup_salida

    r_salida_operativa = read(handles.item_6)
    handles.r_salida_operativa = r_salida_operativa

    value_salida_operativa = getfield(r_salida_operativa, 'Value')
    handles.value_salida_operativa = value_salida_operativa

end

variable_10 = serveritems(handles.da, '*Loop.1.OP.ManualOutVal*');
item_10 = additem(handles.grp, variable_10);
handles.item_10 = item_10
guidata(hObject,handles)
```

```
r_valor_salida_manual = read(handles.item_10)
handles.r_valor_salida_manual = r_valor_salida_manual
value_salida_manual = getfield(r_valor_salida_manual, 'Value')
handles.value_salida_manual = value_salida_manual
set(handles.texto_conectado, 'Visible', 'On')

disp('Conectado')



function configurar_da_Callback(hObject, eventdata, handles)

global cont_xpv;
global h_pv;
global h_u;
global h_sp;

global guardar_altura_pv;
global guardar_altura_sp;
global guardar_eje_tiempo;


disp('entrando')


r_pv = read(handles.item_1)
value_pv = getfield(r_pv, 'Value')
handles.value_pv = value_pv;
set(handles.valor_pv_leido_tanque, 'String' , value_pv);

r_sp = read(handles.item_2)
value_sp = getfield(r_sp, 'Value')
handles.value_sp = value_sp;
set(handles.valor_sp_leido_tanque, 'String' , value_sp);

r_BP = read(handles.item_3)
value_BP = getfield(r_BP, 'Value')
handles.value_BP = value_BP;
set(handles.valor_Kp_leido, 'String' , value_BP);

r_Ti = read(handles.item_4)
value_Ti = getfield(r_Ti, 'Value')
handles.value_Ti = value_Ti;
set(handles.valor_Ki_leido, 'String' , value_Ti);

r_Kd = read(handles.item_5)
value_Kd = getfield(r_Kd, 'Value')
handles.value_Kd = value_Kd;
set(handles.valor_Kd_leido, 'String' , value_Kd);

r_auto_manual = read(handles.item_7)
value_auto_manual = getfield(r_auto_manual, 'Value')

if value_auto_manual == 0
    disp('esta en modo auto')
    hObject = handles.modo_automatico
```

```
    if (hObject ~= 1)
        get(hObject, 'Value')
        set(hObject, 'Value', 1)
    end

    get(handles.valor_salida_operativa_forzada, 'Visible')
    set(handles.valor_salida_operativa_forzada, 'Visible', 'Off')

else
   disp('esta en modo manual')
   hObject = handles.modo_manual

   if (hObject ~= 1)
        get(hObject, 'Value')
        set(hObject, 'Value', 1)
   end

    get(handles.valor_salida_operativa_forzada, 'Visible')
    set(handles.valor_salida_operativa_forzada, 'Visible', 'On')

end

if(hObject == handles.modo_automatico)
    r_salida_operativa = read(handles.item_6)
    handles.r_salida_operativa = r_salida_operativa
    value_salida_operativa = getfield(r_salida_operativa, 'Value')
    handles.value_salida_operativa = value_salida_operativa
    set(handles.salida_operativa, 'String' , value_salida_operativa);

else
   r_salida_forzada = read(handles.item_8)
   value_salida_forzada = getfield(r_salida_forzada, 'Value')
   set(handles.salida_operativa, 'String' , value_salida_forzada)

end


x_pv = cont_xpv;
cont_xpv = cont_xpv + handles.valor_leido_periodo;

y_pv = double(value_pv);

addpoints(h_pv,x_pv,y_pv);
axes(handles.grafica_altura_tanque);
xlim(handles.grafica_altura_tanque,[0, x_pv+5])
ylim(handles.grafica_altura_tanque,[0,20])
drawnow

fprintf ("%d %f\n",cont_xpv, y_pv);


y_sp = double(value_sp);

addpoints(h_sp,x_pv,y_sp);
axes(handles.grafica_altura_tanque);
xlim(handles.grafica_altura_tanque,[0, x_pv+5])
```

```
ylim(handles.grafica_altura_tanque,[0,20])
drawnow

fprintf ("%d %f\n",cont_xpv, y_sp );


    if(hObject == handles.modo_automatico)
        y_u_auto = double(value_salida_operativa);
        addpoints(h_u,x_pv,y_u_auto);
        axes(handles.grafica_salida_operativa);
        xlim(handles.grafica_salida_operativa,[0, x_pv+5])
        ylim(handles.grafica_salida_operativa,[-5, 110])
        drawnow

        fprintf ("%d %f\n", cont_xpv, y_u_auto);

    else
        y_u_manual = double(value_salida_forzada);
        addpoints(h_u,x_pv,y_u_manual);
        axes(handles.grafica_salida_operativa);
        xlim(handles.grafica_salida_operativa,[0, x_pv+5])
        ylim(handles.grafica_salida_operativa,[-5, 110])
        drawnow

        fprintf ("%d %f\n", cont_xpv, y_u_manual);
    end



   if (value_pv > (value_sp + 2))

       set(handles.indicador_alarma, 'Visible', 'on')
       pause(0.5)

       set(handles.texto_alarma, 'Visible', 'on')
       set(handles.boton_ok_alarma, 'Visible', 'on')
       set(handles.indicador_alarma, 'Visible', 'off')
       pause(0.5)
       set(handles.indicador_alarma, 'Visible', 'on')

   end

   guardar_altura_pv = [guardar_altura_pv y_pv];
   guardar_altura_sp = [guardar_altura_sp y_sp];
   guardar_eje_tiempo = [guardar_eje_tiempo x_pv];



function texto_titulo_CreateFcn(hObject, eventdata, handles)



function texto_conectado_CreateFcn(hObject, eventdata, handles)
```

```
function pv_tanque_CreateFcn(hObject, eventdata, handles)



function sp_tanque_CreateFcn(hObject, eventdata, handles)



function valor_sp_tanque_Callback(hObject, eventdata, handles)

valor_sp_tanque = str2double(get(hObject, 'String'));
handles.valor_sp_tanque = valor_sp_tanque;
set(handles.valor_sp_leido_tanque, 'String' , valor_sp_tanque)
write(handles.item_2, valor_sp_tanque)
guidata(hObject,handles)



function valor_sp_tanque_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function valor_pv_leido_tanque_CreateFcn(hObject, eventdata, handles)

valor_pv_leido_tanque = str2double(get(hObject, 'String'));
handles.valor_pv_leido_tanque = valor_pv_leido_tanque;
guidata(hObject,handles)



function valor_sp_leido_tanque_CreateFcn(hObject, eventdata, handles)

valor_sp_leido_tanque = str2double(get(hObject, 'String'));
handles.valor_sp_leido_tanque = valor_sp_leido_tanque;
guidata(hObject,handles)



function borrar_datos_Callback(hObject, eventdata, handles)

delete(handles.da)



function apagar_Callback(hObject, eventdata, handles)

disconnect(handles.da)
```

```
function valor_Kp_leido_CreateFcn(hObject, eventdata, handles)

valor_Kp_leido = str2double(get(hObject, 'String'));
handles.valor_Kp_leido = valor_Kp_leido;
guidata(hObject,handles)



function valor_Ki_leido_CreateFcn(hObject, eventdata, handles)

valor_Ki_leido = str2double(get(hObject, 'String'));
handles.valor_Ki_leido = valor_Ki_leido;
guidata(hObject,handles)



function valor_Kd_leido_CreateFcn(hObject, eventdata, handles)

valor_Kd_leido = str2double(get(hObject, 'String'));
handles.valor_Kd_leido = valor_Kd_leido;
guidata(hObject,handles)



function valor_Kp_introducido_Callback(hObject, eventdata, handles)

valor_Kp_introducido = str2double(get(hObject, 'String'));
handles.valor_Kp_introducido = valor_Kp_introducido;
set(handles.valor_Kp_leido, 'String' , valor_Kp_introducido)
write(handles.item_3(1), valor_Kp_introducido);
guidata(hObject,handles)



function valor_Kp_introducido_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function valor_Ki_introducido_Callback(hObject, eventdata, handles)

valor_Ki_introducido = str2double(get(hObject, 'String'));
handles.valor_Ki_introducido = valor_Ki_introducido;
set(handles.valor_Ki_leido, 'String' , valor_Ki_introducido)
write(handles.item_4(1), valor_Ki_introducido);
guidata(hObject,handles)
```

```
function valor_Ki_introducido_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function valor_Kd_introducido_Callback(hObject, eventdata, handles)

valor_Kd_introducido = str2double(get(hObject, 'String'));
handles.valor_Kd_introducido = valor_Kd_introducido;
set(handles.valor_Kd_leido, 'String' , valor_Kd_introducido)
write(handles.item_5(1), valor_Kd_introducido);
guidata(hObject,handles)



function valor_Kd_introducido_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function start_timer_Callback(hObject, eventdata, handles)

a = timer;
handles.a = a
set(handles.a, 'ExecutionMode', 'fixedRate')
set(handles.a, 'Period', handles.valor_leido_periodo)
get(handles.a, 'Period')
set(handles.a, 'TimerFcn',{@configurar_da_Callback, handles})

disp('start')
start(handles.a)
disp('despues start')
guidata(hObject,handles)



function stop_Callback(hObject, eventdata, handles)

global guardar_altura_pv;
global guardar_altura_sp;
global guardar_eje_tiempo;

assignin('base','altura_pv', guardar_altura_pv);
assignin('base','altura_sp', guardar_altura_sp);
assignin('base','eje_tiempo', guardar_eje_tiempo);

stop(handles.a)
```

```
function texto_salida_operativa_CreateFcn(hObject, eventdata, handles)



function salida_operativa_CreateFcn(hObject, eventdata, handles)

salida_operativa = str2double(get(hObject, 'String'));
handles.salida_opeativa = salida_operativa;
guidata(hObject,handles)



function valor_salida_operativa_forzada_Callback(hObject, eventdata,
handles)

valor_salida_operativa_forzada = str2double(get(hObject, 'String'));
handles.valor_salida_operativa_forzada =
valor_salida_operativa_forzada;
write(handles.item_8, valor_salida_operativa_forzada);
write(handles.item_10, valor_salida_operativa_forzada);
set(handles.salida_operativa, 'String' ,
valor_salida_operativa_forzada)
guidata(hObject,handles)



function valor_salida_operativa_forzada_CreateFcn(hObject, eventdata,
handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function modo_automatico_Callback(hObject, eventdata, handles)

modo_automatico = get(hObject,'Value')
get(handles.valor_salida_operativa_forzada, 'Visible')
set(handles.valor_salida_operativa_forzada, 'Visible', 'Off')

write(handles.item_7, 0)

r_limite_sup_salida = read(handles.item_9)
value_limite_sup_salida = getfield(r_limite_sup_salida, 'Value')

set(handles.salida_operativa, 'String', value_limite_sup_salida);

handles.modo_automatico = modo_automatico
guidata(hObject,handles)



function modo_manual_Callback(hObject, eventdata, handles)
```

```
modo_manual = get(hObject,'Value')
get(handles.valor_salida_operativa_forzada, 'Visible');
set(handles.valor_salida_operativa_forzada, 'Visible', 'On');

write(handles.item_7, 1)

r_salida_forzada = read(handles.item_8)
value_salida_forzada = getfield(r_salida_forzada, 'Value')

set(handles.salida_operativa, 'String' , value_salida_forzada)

handles.modo_manual = modo_manual
guidata(hObject,handles)



function uibuttongroup5_SelectionChangedFcn(hObject, eventdata,
handles)

if(hObject == handles.modo_automatico)
   write(handles.item_7, 0);
   handles.r_limite_sup_salida
   handles.value_limite_sup_salida
   set(handles.salida_operativa, 'Value' ,
handles.value_limite_sup_salida);

else
   write(handles.item_7, 1);
   set(handles.salida_operativa, 'Value' ,
handles.valor_salida_operativa_forzada)
   write(handles.item_8, handles.valor_salida_operativa_forzada);
end



function uibuttongroup5_CreateFcn(hObject, eventdata, handles)



function texto_salida_op_grafica_CreateFcn(hObject, eventdata,
handles)



function grafica_altura_tanque_CreateFcn(hObject, eventdata, handles)



function grafica_salida_operativa_CreateFcn(hObject, eventdata,
handles)



function altura_CreateFcn(hObject, eventdata, handles)
```

```
function indicador_alarma_CreateFcn(hObject, eventdata, handles)


function texto_alarma_CreateFcn(hObject, eventdata, handles)


function boton_ok_alarma_Callback(hObject, eventdata, handles)

set(handles.texto_alarma, 'Visible', 'off')
set(handles.indicador_alarma, 'Visible', 'off')
set(handles.boton_ok_alarma, 'Visible', 'off')


function boton_ok_alarma_CreateFcn(hObject, eventdata, handles)

boton_ok_alarma = get(hObject,'Value');
handles.boton_ok_alarma = boton_ok_alarma;
guidata(hObject,handles)


function texto_periodo_CreateFcn(hObject, eventdata, handles)


function valor_periodo_introducido_Callback(hObject, eventdata,
handles)

valor_periodo_introducido = str2double(get(hObject, 'String'));

stop(handles.a)

handles.valor_periodo_introducido = valor_periodo_introducido;

set(handles.a, 'Period', handles.valor_periodo_introducido)

start(handles.a)

handles.valor_leido_periodo = valor_periodo_introducido;
guidata(hObject,handles)


function valor_leido_periodo_CreateFcn(hObject, eventdata, handles)

valor_leido_periodo = str2double(get(hObject, 'String'));
handles.valor_leido_periodo = valor_leido_periodo;
guidata(hObject,handles)
```