Universidad Zaragoza

1542

Trabajo de fin de grado

Supervisión y automatización de una maqueta de procesos continuos.

Supervision and automation of a continuous process model.

Autor:

Monzón Castelló, Sergio

Director:

Ramón Piedrafita Moreno

ESCUELA DE INGENIERÍA Y ARQUITECTURA

FEBRERO 2017

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

D./Dª. SERGIO MONZÓN CASTELLÓ,

con nº de DNI 17769388W en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) Supervisión y automatización de una maqueta de procesos continuos
Supervision and automation of a continuous process model)

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 3 DE FEBRERO DE 2017

Fdo: SERGIO MONZÓN CASTELLÓ

Supervisión y automatización de una maqueta de procesos continuos.

Supervision and automation of a continuous process model.

RESUMEN

En este proyecto, se ha llevado a cabo la programación y puesta en marcha de una maqueta educativa de procesos continuos que trabaja con líquidos (agua destilada en este caso para evitar oxidaciones), con su correspondiente programación para llevar a cabo los objetivos posteriormente dispuestos. La intencionalidad de esta maqueta es el aprendizaje del control de variables analógicas por parte del alumnado de la escuela.

Este proyecto parte de la continuación de un Proyecto de Final Carrera de un compañero de la antigua titulación, que no alcanzaba a desarrollar los objetivos deseables para esta maqueta, y por lo tanto hacía necesaria una replanteación y su actualización, lo cual se lleva a cabo en este proyecto [1]. Dicha maqueta se encuentra en uno de los laboratorios del departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza.

Las variables analógicas que han sido controladas son: caudal, temperatura y nivel del líquido en el tanque central, y presión del aire del tanque izquierdo. El control de dichas variables puede realizarse tanto en un autómata programable (caso que se ha abordado en este proyecto) como en unos reguladores industriales que posee la maqueta y externos al autómata, funcionando independientes a este. Aunque existe un sistema SCADA desarrollado por el propio fabricante de la maqueta, se ha optado por hacer uno propio que permite mayor flexibilidad y control.

El control anteriormente mencionado se ha hecho por 2 métodos: control por regulador PID y por regulador IMC, pudiendo alternar entre ambos para caudal, nivel y temperatura dependiendo cual nos convenga más. Para el regulador PID se le ha añadido la funcionalidad adicional de calcular automáticamente los parámetros de regulación óptimos, a través del bloque de Autotuning que proporciona el entorno de desarrollo.

Para darle mayor funcionalidad a la maqueta, se han creado varios archivos *Simulink* de Matlab, que por medio de comunicación OPC con el autómata permiten tanto la visualización de variables del sistema como el control de procesos. Así mismo, se ha configurado un procedimiento de exposición que, de manera automática, muestra al usuario el funcionamiento de los elementos de la maqueta con una explicación de la operativa de los mismos.

ÍNDICE

Tabla de Ilustraciones

Capítulo 1. Introducción

En este capítulo vamos a encuadrar los aspectos principales de esta memoria, como pueda ser sus objetivos, alcance, contenido y siglas empleadas en la misma.

1.1 Objetivos

Los objetivos que se plantean alcanzar en este proyecto son los siguientes:

- Estudiar el proyecto anterior, el cuales sirve de base para la realización de este. Se deberá hacer un análisis detallado de los aspectos críticos, tanto aquellos que funcionan y se pueden mejorar como los que tienen algún tipo de fallo para corregir.
- Adecuación y mejora de la programación ya existente del autómata, rehaciendo el programa del PLC (en ST, *Estructured Text*) a lenguaje LD (*Leadder Diagram*) las partes necesarias, separando el único regulador PID existente en tantos individuales como variables vayamos a controlar, permitiendo el control multivariable independiente.
- Solucionar los problemas con el control de temperatura, ya que actualmente se desarrolla un control independiente para calentar y otro para enfriar, no permitiendo un control preciso de la temperatura del líquido.
- Incorporación de una pantalla de explotación en el programa Unity del autómata, para permitir el control de la maqueta desde el PC sin necesidad de un SCADA.
- Desarrollo de una nueva aplicación SCADA sobre VIJEO CITECT versión 7, de la empresa SCHNEIDER. Se deberá hacer una actualización y ampliación de la aplicación SCADA existente para la versión 6 de VIJEO CITECT, añadiendo las funcionalidades para llevar a cabo los objetivos dispuestos en este apartado, y poder visualizar el estado de esta maqueta desde un PC y poder realizar los diversos controles.
- Desarrollo de una clase automática, de forma que de manera autónoma el Sistema de Supervisión puede demostrar el funcionamiento de los diferentes controles implementados, mostrando al usuario de manera ordenada y detallada todas las características disponibles en la maqueta. Esto servirá para enseñar a los alumnos de la carrera los fundamentos de los diferentes controles que pueden ser implementados en un autómata programable para un entorno industrial.
- Incorporación de un *Autotuning* a los reguladores para evitar los posibles fallos de llevar a cabo un modelado manual del sistema. Este módulo permite el ajuste automático de los reguladores PID. Durante el inicio de la instalación, el ajuste automático del regulador estabiliza la regulación y ahorra tiempo.
- Incorporación de control IMC (corrector del modelo), que permite afrontar los retardos graves, si existen, comparándolos con la constante de tiempo principal del proceso, por lo que será la opción elegida si los PIDs no nos proporcional la respuesta adecuada o se ve una clara no linealidad.
- Estudio de las posibilidades de comunicación del autómata con MATLAB junto con la creación de una aplicación para permitir realizar gráficas con los datos tomados de las variables analógicas en tiempo real. También se verá la posibilidad de realidad algún tipo de control sobre la maqueta con MATLAB, siempre y cuando sea posible.

- Realizar un estudio de las posibles mejoras sobre la maqueta a partir de los conocimientos obtenidos durante la realización de este TFG.

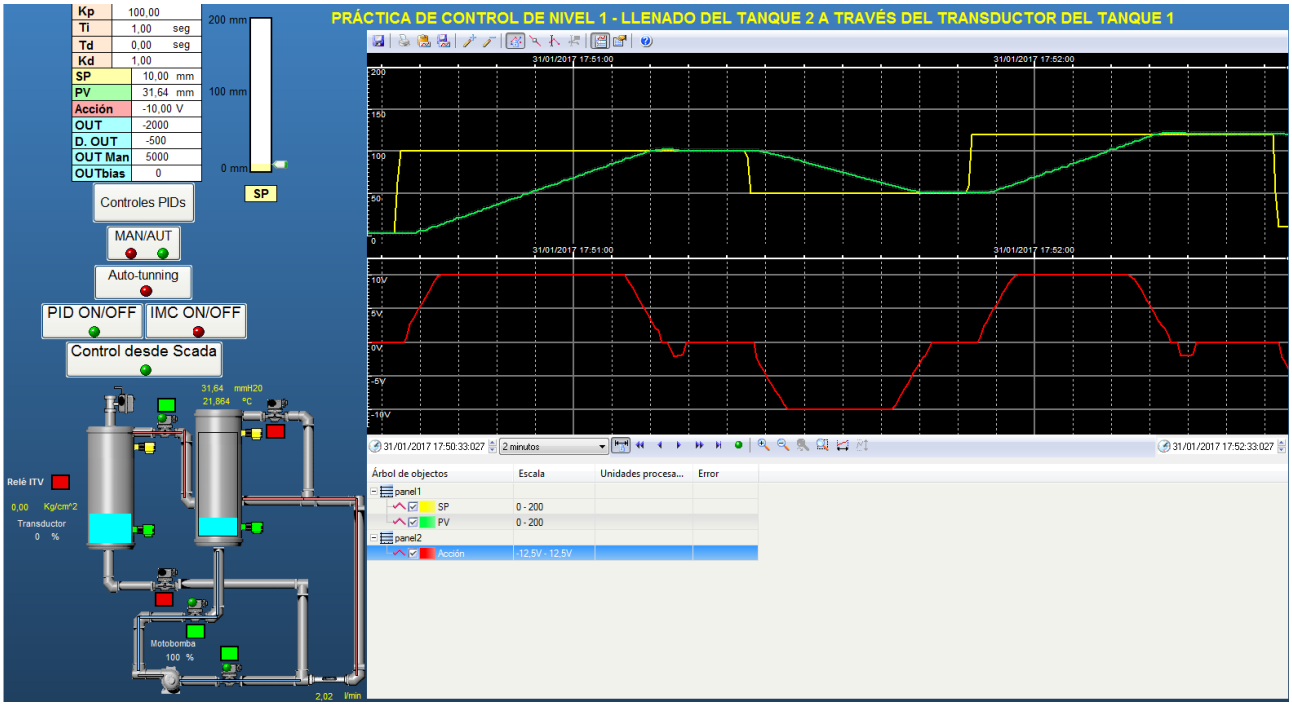


Figura 1. Pantalla SCADA de Control de Nivel

1.2 Alcance

Los elementos a desarrollar que conforman el proyecto tienen de propósito general la enseñanza en el control de procesos continuos mediante autómatas programables.

Si bien el ámbito de este proyecto, en un principio, es la formación de alumnos de la Escuela de Ingeniería y Arquitectura, puede ser aplicado a otros colectivos que requieran de un aprendizaje en estos sistemas.

El uso debe ser sencillo, ya que se supone que el alumno parte de unos conocimientos básicos de estos sistemas, además de garantizar que ni el estudiante ni la maqueta puedan sufrir daños mediante su ejecución, además de resultar cómoda de manejar para permitir una correcta asimilación de conocimientos.

Mediante este proyecto, el alumno adquirirá y reforzará sus conocimientos de sistemas continuos, aprendiendo uno de las muchas maneras de controlarlos.

1.3 Contenido de la memoria

La memoria está estructurada en 6 capítulos.

Capítulo 1 - Introducción. Se indican los objetivos principales del proyecto, su alcance, su estructura y las siglas empleadas en el mismo.

Capítulo 2 - Descripción de la estación de depósitos. Se indican las principales características del aparato, como son los sensores, los actuadores y el autómata programable.

Capítulo 3 – Control de la estación de depósitos. Descripción y estudio del control del sistema mediante PID e IMC, así como del *Autotuning* en asociación con estos para la obtención automática de sus parámetros óptimos.

Capítulo 4 – Supervisión de la estación. Estudio e implementación de la comunicación entre el autómata y el PC, para el posterior desarrollo de la nueva aplicación de control mediante el software SACADA, junto con la comunicación y representación de variables en tiempo real en MATLAB y el desarrollo de pantallas gráficas en la propia aplicación del PLC.

Capítulo 5 – Desarrollo de un sistema de demostración Automático. En este apartado se abordará la creación de un programa en lenguaje SFC de control en el autómata que hará una demostración que irá recorriendo automáticamente los diferentes aspectos que componen el control de la maqueta. Dicha demostración irá mostrando diferentes pantallas explicativas en el SCADA.

Capítulo 6 - Conclusiones finales. La memoria finaliza con un capítulo en el que se presentan las conclusiones finales del proyecto, realizando un análisis de los objetivos establecidos junto con su cumplimento o sus defectos, proponiéndose posibles mejoras a futuro o líneas de trabajo.

1.4 Lista de acrónimos

- PLC: *Programmable Logic Controller* (controlador lógico programable o también autómata programable).
- SCADA: *Supervisory Control And Data Acquisition* (Supervisión, Control y Adquisición de Datos).
- OPC: *OLE for Process Control*.
- OLE: *Object Linking and Embedding*.
- SFC: *Secuencial Function Chart*.
- SP: *Set point* o punto de consigna.
- PV: *Process value* o valor del proceso.

Capítulo 2. Descripción de la estación de depósitos

En este capítulo se va a proceder a indicar las principales características del aparato, como son los sensores, los actuadores y el autómata programable, explicando brevemente cada uno de ellos y adjuntando su hoja de características.

2.3 PLC TSX-Premium P57 2634M

El PLC es un dispositivo diseñado para controlar procesos secuenciales en entornos industriales, por lo que van asociados a la maquinaria que desarrolla procesos de producción y controlan su trabajo. PLC son las siglas en inglés de Controlador Lógico Programable (*Programmable Logic Controller*) siendo un sistema completo, ya que contiene todo lo necesario para operar, y es industrial, por tener todos los registros necesarios para operar en los ambientes hostiles que se encuentran en la industria [2].

Un PLC realiza, entre otras, las siguientes funciones:

- Recoger datos de los puertos de entrada analógicos y digitales.
- Ejecuta sentencias previamente programadas en función de esos datos.
- Almacena datos en su memoria y lee los datos almacenados, para modificarlos si es preciso e interactuar con las señales de entrada.
- El autómata viene controlado por ciclos de tiempo, en cada uno de los cuales se ejecutan y evalúan todas aquellas funciones que hayan sido programadas y deban comprobarse.
- Genera cálculos matemáticos sobre los datos de entrada para obtener unos datos de salida.
- Dichos datos de salida actuarán sobre dispositivos externos mediante las salidas analógicas y digitales.
- Tienen la habilidad de comunicarse con otros sistemas externos, tanto autómatas como dispositivos diversos.

Al contrario que otro tipo de controladores, los PLCs están diseñados para manejar cualquier tipo de maquinaria industrial, además de ser automáticos manteniendo estable la operación de las máquinas sin requerir participación humana.

Figura 2. Imagen del PLC TSX-Premium P57 2634M

2.1 Sensores

En esta sección se van a describir los diferentes sensores que incluye la maqueta y van a ser utilizados en el proyecto.

Presostato digital ISE40-01-62L

Un presostato está basado en un sensor electrónico que ofrece también la funcionalidad de un transmisor de presión [3].

Por defecto un presostato solo emite las señales digitales si la presión ha alcanzado el punto de interrupción o no, pero no indican la diferencia entre presión actual y presión del punto de interrupción. Muchos presostato disponen por lo tanto de un display que permite la lectura de la presión actual o de una señal analógica suplementaria.

Este sistema permite además la transmisión de la presión captada mediante la señal analógica a una unidad de control y regulación. En nuestro caso, el presostato será el encargado de darnos la lectura de la presión del aire en el depósito de la derecha, ya que proporciona dos salidas digitales y una analógica que pueden ser empleadas para realizar un control de la presión del depósito.

Figura 3. Imagen preostato digital ISE40-01-62L

Caudalímetro PF2W504-F03-2

Un caudalímetro es un instrumento de medida para la medición de caudal o gasto volumétrico de un fluido o para la medición del gasto másico. Estos aparatos suelen colocarse en línea con la tubería que transporta el fluido [4]. Los caudalímetros se conocen por muchos nombres, como flujómetro, indicador de flujo, medidor de líquido, etc., según la industria. No obstante, la función de medición de caudal es siempre la misma.

En nuestro caso, usaremos el caudalímetro para medir el caudal que circula hacia los depósitos.

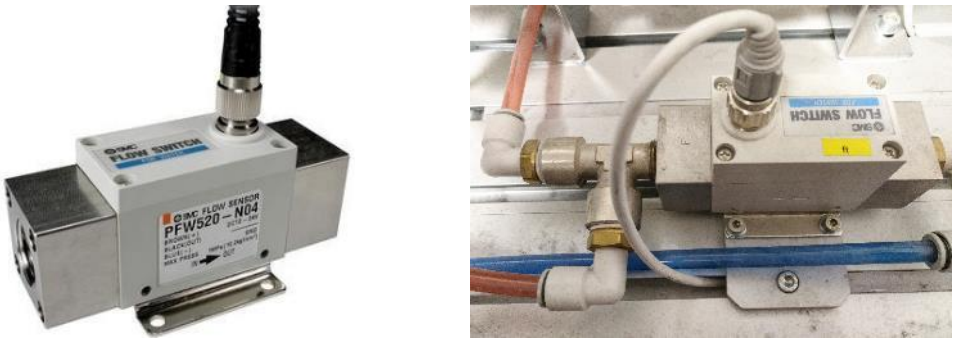


Figura 4. Imagen caudalímetro PF2W504-F03-2

Sensor Nivel. Presión diferencial PSE550-AC2

Este sistema se basa en el principio de medir la carga o presión hidrostática. La carga se define como el peso de líquido que existe por encima de un plano de referencia, y es independiente del volumen de líquido implicado o de la forma del recipiente. El sistema no mide el nivel de líquido sino la presión ejercida y, como la presión es proporcional a la altura de columna de líquido, el medidor «infiere» la posición actual del nivel [5].

En la maqueta, utilizamos dicho sensor para medir el nivel de líquido contenido en el tanque central.

Figura 5. Imagen sensor de nivel por presión diferencial PSE550-AC2

Sonda de temperatura PTC100 con cabeza amplificadora

La PT100 es un sensor de temperatura consistente en un alambre de platino que a 0 °C tiene 100 ohms y que al aumentar la temperatura aumenta su resistencia eléctrica. El incremento de la resistencia de la PT100 no es lineal, pero si creciente y característico del platino de tal forma que mediante tablas es posible encontrar la temperatura exacta a la que corresponde.

Normalmente las sondas PT100 industriales se fabrican encapsuladas en la misma forma que los termopares, es decir dentro de un tubo de acero inoxidable u otro material (vaina). En un extremo está el elemento sensible (Sensor RTD) y en el otro está el terminal eléctrico de los cables protegido dentro de una caja redonda de aluminio (cabezal) [6].

Dicha sonda se usará para medir la temperatura del tanque central de la maqueta.

Figura 6. Imagen Sonda de temperatura PTC100 con cabeza amplificadora

Sensores capacitivos

Los sensores capacitivos son un tipo de sensor eléctrico que reaccionan ante metales y no metales, los cuales al aproximarse a la superficie activa sobrepasan una determinada capacidad. Su funcionamiento se basa en un circuito oscilante RC y las líneas del campo eléctrico que se cierran a través del aire. La aproximación de un objeto con una constante dieléctrica superior a la del aire, ocasiona el desequilibrio del circuito y el inicio de las oscilaciones [5].

Sensibles a la mayoría de líquidos y materiales, permitiendo la detección de otros materiales a través de materiales o paredes no conductores (presencia de agua en el interior de una tubería plástica o depósito). Válidos para materiales no conductores como plástico, cristal, goma y conductivos como metales o agua.

Emplearemos los sensores capacitivos para detectar si el nivel de agua en los 3 tanques está comprendido entre un mínimo y un máximo, colocando dichos sensores en la parte inferior y superior de los depósitos.

Figura 7. Imagen sensores capacitivos

2.2 Actuadores

En esta sección se han a describir los diferentes actuadores que incluye la maqueta y tendremos que controlar en nuestra aplicación.

Electroválvulas monoestables 3/2 VKF334Y-5DZ-01F / VCW21-5D-4-02F

Una electroválvula es una válvula electromecánica, diseñada para controlar el paso de un fluido por un conducto o tubería. La válvula se mueve mediante una bobina solenoide. Generalmente no tiene más que dos posiciones: abierto y cerrado, o todo y nada. Las electroválvulas se usan en multitud de aplicaciones para controlar el flujo de todo tipo de fluidos.

En nuestro caso, dichas válvulas serán accionadas por nuestro autómata cuando el control lo requiera. Las utilizaremos para controlar tanto el paso de agua como de aire.

Figura 8. Imagen electroválvulas monoestables

Electroválvula proporcional Burkert 6022

Al contrario que las que las anteriores, las electroválvulas proporcionales sirven para hacer una regulación más precisa del fluido que circula por ellas, permitiendo controlar y regular dicho líquido en función de un valor de consigna. Su funcionamiento se basa en que la cantidad de fluido al que permiten el paso es directamente proporcional a la tensión de consigna aplicada.

Dicho valor de consigna será proporcionado por el autómata en función de la cuantía calculado en el control correspondiente.

Figura 9. Imagen electroválvula proporcional Burkert 6022

Regulador de presión proporcional ITV1030-01F2BN2-Q

Los reguladores reductores de presión son equipos de control de flujo diseñados para mantener una presión constante aguas abajo de ellos, independientemente de las variaciones de presión a la entrada o los cambios de requerimientos de flujo. Los mecanismos internos que componen un regulador, automáticamente controlan o limitan las variaciones de presión a un valor previamente establecido [8].

Usaremos el regulador para controlar la presión del aire en el tanque de la izquierda.

Figura 10. Imagen regulador de presión proporcional ITV1030-01F2BN2-Q

Motobomba 12V para procesos continuos

Una bomba hidráulica es una máquina generadora que transforma la energía mecánica (en nuestro caso un motor accionado eléctricamente) con la que es accionada en energía del fluido incompresible que mueve.

Al incrementar la energía del fluido, se aumenta su presión, su velocidad o su altura, todas ellas relacionadas según el principio de Bernoulli, utilizándose dicha propiedad para incrementar la presión de un líquido añadiendo energía al sistema hidráulico, para mover el fluido de una zona de menor presión o altitud a otra de mayor presión o altitud [9].

Figura 11. Imagen motobomba 12V

Células Peltier 12v – 73 Watt Qmax

La refrigeración termoeléctrica utiliza el efecto Peltier para crear un flujo térmico a través de la unión de dos materiales diferentes, como metales o semiconductores tipo P y N. Un refrigerador o calentador usando uno de estos dispositivos transfiere calor de un lado del aparato al otro oponiéndose al gradiente de temperatura, consumiendo para ello energía eléctrica, por lo que será necesario alimentarlo con una diferencia de tensión continua [10].

No es la manera más sencilla de conseguir calentamiento, pero como tiene la posibilidad de enfriar si invertimos la polaridad, se suelen usar para situaciones en las que el montaje requiera de ambas funciones.

Figura 12. Imagen de algunos modelos de Células Peltier

Capítulo 3. Control de la estación de depósitos

En este capítulo, se desarrollan los aspectos principales correspondientes con el autómata programable de la estación de depósitos, los cuales nos proporcionaran la funcionalidad primordial con la cual alcanzaremos los objetivos propuestos.

En el *Anexo B. Programa PLC* se detalla el programa implementado en el autómata con los principales módulos que lo componen (expuestos en este capítulo), junto con el resto de elementos para obtener la funcionalidad deseada.

3.1 Control PID

En este primer apartado se va a hablar acerca del control PID necesario a implementar en la maqueta para controlar las variables que podamos manejar.

Un controlador PID es un mecanismo de control por realimentación ampliamente usado en sistemas de control industrial. Este calcula la desviación o error entre un valor medido y un valor deseado [11].

La misión de estos controladores es:

- Conseguir que la variable controlada siga la señal de referencia lo más exactamente posible.
- Atenuar el efecto de las perturbaciones lo más rápidamente posible.
- Conseguir una determinada y aceptable respuesta temporal del sistema controlado

Existen 4 tipos de controladores clásicos:

- Tipo P: o control proporcional. En los reguladores proporcionales la acción suministrada por el regulador es proporcional al error. La respuesta del regulador se produce sin retraso en el tiempo; existe un error en régimen permanente.
  La acción proporcional permite obtener diferentes velocidades de respuesta del proceso. Si la ganancia es elevada se acelera la respuesta del sistema y se reduce el error estático, pero la estabilidad se ve perjudicada. Es necesario encontrar un buen compromiso entre la velocidad y la estabilidad.

$$U(s) = K_p \cdot E(s)$$

Ecuación 1: Ecuación de un regulador tipo P

U: Salida del controlador proporcional.
Kp: Ganancia proporcional.
E: Error de proceso instantáneo en el tiempo t. E=SP-PV.
SP: Punto establecido. PV: Proceso variable.

- Tipo PI: o control proporcional integral. Añadimos un regulador Proporcional Integral en la cadena directa tal que la acción es proporcional a la señal de error y a la integral de la señal de error en el tiempo.
  La acción integral permite cancelar el error estático (desviación entre la medida y el valor de consigna). Cuanto más elevada es la acción integral (Ki/τi mayor) la respuesta se acelera, pero la estabilidad se deteriora. También es necesario encontrar un buen compromiso entre la velocidad y la estabilidad.

$$U(s) = \left(K_p + K_i \cdot \frac{1}{\tau_i \cdot s}\right) \cdot E(s)$$

Ecuación 2: Ecuación de un regulador tipo PI

  Ki: Ganancia integral. τi: Tiempo de acción integral.

- Tipo PD: o control proporcional diferencial. Los reguladores proporcionales diferenciales son aquellos en que su variable de salida es proporcional a la señal de error y la derivada de la señal de error.
  La acción derivada tiene una característica de anticipación, ya que agrega un elemento que tiene en cuenta la velocidad de variación del error, lo que permite anticipar acelerando la respuesta del proceso cuando la desviación aumenta y decelerando cuando la desviación disminuye. Cuanto más elevada es la acción derivada (Kd*τd mayor), más se acelera la respuesta.

$$U(s) = \left(K_p + K_d \cdot \tau_d \cdot s\right) \cdot E(s)$$

Ecuación 3: Ecuación de un regulador tipo PD

  Kd: Ganancia derivativa. τd: Tiempo de acción derivativo.

- Tipo PID: o control proporcional integral diferencial. Son aquellos que su variable de salida es proporcional a la suma de la señal de error, de la derivada de la señal de error y de la integral de la señal de error. El valor Proporcional depende del error actual, el Integral de los errores pasados y el derivativo es una predicción de los errores futuros (como ya hemos dicho anteriormente).
  Es un mecanismo de control por realimentación ampliamente usado en sistemas de control industrial, ya que históricamente se ha considerado que el controlador PID es el más adecuado ya permite una gran flexibilidad dependiendo del tipo de proceso que tengamos, porque con un solo controlador podemos tener además las 3 tipologías anteriores. Aun así, el uso del PID para control no garantiza control óptimo del sistema o la estabilidad del mismo.

$$U(s) = \left(K_p + K_i \cdot \frac{1}{\tau_i \cdot s} + K_d \cdot \tau_d \cdot s\right) \cdot E(s)$$

Ecuación 4: Ecuación de un regulador tipo PID

  Este último será el controlador utilizado en el proyecto, para poder obtener el comportamiento adecuado según lo que queramos conseguir.

Viendo la disponibilidad de sensores y actuadores en la maqueta, se ha decidido hacer los siguientes controles:

- Control de caudal: utilizando el caudalímetro, se va a proceder al control del caudal que pasa por el mismo mediante 3 posibles combinaciones entre los 3 depósitos:
  - Recirculación del agua en el tanque central: se utilizará la motobomba colocada a la salida del tanque intermedio para impulsar el agua de nuevo a la parte superior del mismo, controlado el caudal mediante la acción impuesta a dicha motobomba.
  - Transvase del depósito izquierdo al central: emplearemos el regulador de presión proporcional del depósito de origen, aplicándole una acción calculada, para empujar el agua a través del caudalímetro para introducirlo a continuación en el depósito intermedio con la velocidad deseada.
  - Transvase del depósito derecho al central: recurriremos a la electroválvula proporcional, junto con la válvula para presurizar el tanque derecho, con el objetivo de inducir la circulación del agua a través del caudalímetro para introducirla en el tranque intermedio igual que en el caso anterior.
- Control de temperatura: se emplearán las placas peltier situadas bajo el tanque central para controlar la temperatura del mismo junto con un disipador y un ventilador adheridos a la parte inferior, y un elemento giratorio a modo de homogeneizador para favorecer este proceso:
  - Para calentar aplicaremos una acción calculada sobre las placas peltier que, junto con el elemento giratorio, nos permitirán aumentar la temperatura del líquido.
  - Para enfriar nos ayudaremos del ventilador con disipador situados en la parte inferior, favoreciendo su efecto con el agitador ya mencionado para disminuir la temperatura del agua.
- Control de nivel: para este control utilizaremos el sensor de nivel por presión diferencial situado en el tanque central para hacer el control de la cantidad de agua contenida en el mismo. Tendremos 2 posibilidades para regular el nivel del tanque:
  - Transferir agua entre el tanque izquierdo y el central. En esta modalidad nos basaremos en la medida del sensor de nivel para que el regulador del autómata decida qué hacer, si hace falta sacar agua del depósito intermedio o llenarlo. Una vez tengamos esto y calculada la acción a emplear, está ultima irá impuesta sobre la motobomba para el primer caso o sobre el regulador de presión proporcional para el segundo caso
  - Transferir agua entre el tanque derecho y el central. De esta manera, realizaremos la medida como en el caso anterior con la diferencia de utilizar esta vez el depósito derecho para meter agua en el central o para sacar agua del mismo. Calculada la acción, la volcaremos sobre la motobomba para extraer agua del central para enviarlo al derecho, o por el contrario la impondremos en la electroválvula proporcional para hacer la acción inversa.

- Control de presión: en este último control utilizaremos el regulador de presión proporcional situado en el tanque izquierdo para obtener la presión deseada del mismo. Como su propio nombre indica, el regulador lleva internamente los elementos necesarios para hacer el control de la presión que se le impone, por lo que no será necesario implementar dicho control en el autómata, sino que directamente impondremos la consigna.

  Aunque con la medida de presión del regulador obtengamos el valor real que hay en el tanque, la consigna que nosotros imponemos es un valor relativo a la presión con la que alimentamos el regulador. Esto es debido a que nosotros podemos tener en nuestro sistema neumático una presión inferior a la máxima que admite el mismo, y por lo tanto no podremos llegar a alcanzar una superior a la disponible. Por esta razón, en nuestro caso trataremos la presión deseada como un porcentaje sobre la máxima disponible, imponiendo esta consigna en el regulador.

Una vez tenemos elegido qué tipo de controles vamos a hacer, nos vamos a la documentación del entorno de desarrollo del PLC (*Unity Pro XL*). En dicha documentación localizamos el bloque *PIDFF: Controlador PID completo*. Dicho bloque se basa en un algoritmo PID para realizar las funciones que queremos implementar [12].

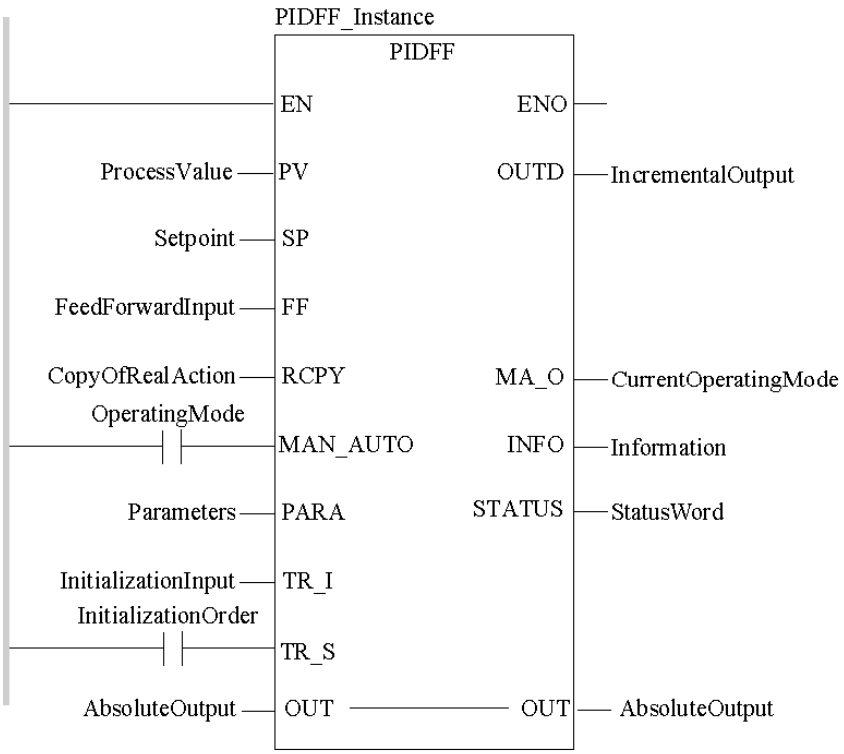


Figura 13. Bloque PIDFF

De este bloque utilizaremos las siguientes entradas y salidas:

Entradas

- *EN:* Dicha entrada servirá para habilitar o deshabilitar el bloque *PIDFF*.
- *PV:* Entrada donde escribiremos el valor actual de la variable a controlar.
- *SP:* Entrada donde establecemos el valor que queremos de la variable a controlar.
- *MAN_AUTO:* Entrada que determina si el bloque funciona en modo Automático o Manual. En el primero, las salidas *OUT* y *OUTD* corresponden al resultado de los cálculos efectuados por el bloque; en la segunda, la salida *OUT* no se ajusta por medio del bloque. El usuario puede ajustar este valor directamente.
- *PARA:* Variable que contiene los parámetros a configurar del bloque.
  - *PV_INF:* Límite inferior del rango del valor real.
  - *PV_SUP:* Límite superior del rango del valor real.
  - *OUT_INF:* Límite inferior del rango de valores de salida cuando el controlador está en modo Manual.
  - *OUT_SUP:* Límite superior del rango de valores de salida cuando el controlador está en modo Manual.
  - *REV_DIR:* Variable que indica si el controlador funciona de modo directo u opuesto. En nuestro caso, lo estableceremos en modo directo.
  - *MIX_PAR:* Variable que establece si la estructura del controlador es paralela o mixta. Para nuestro objetivo, la establecemos en mixta para obtener el tipo PID explicado antes.
  - *BUMP:* Variable que deshabilitaremos para que la conmutación entre el modo Manual al Automático se haga sin brusquedad.
  - *OUTBIAS:* Compensación manual de la desviación estática. Será una variable que podremos modificar externamente.
  - *OUT_MIN:* Límite inferior del rango de valores de salida cuando el controlador está en modo Automático.
  - *OUT_MAX:* Límite superior del rango de valores de salida cuando el controlador está en modo Automático.
  - *Kp, ti, td, kd:* Los parámetros más importantes que debemos proporcionar a estos PIDs serán la contante proporcional *kp*, la contante de tiempo integral *ti*, la constante de tiempo diferencial *td*, y la contante de ganancia diferencial *kd*. Dichos parámetros serán los que determinen la dinámica del sistema.
- *TR_I:* Entrada de inicialización. El valor introducido aquí se transfiere a la salida *OUT* cuando *TR_S* está activado. Como en la modalidad manual, *OUT* se encuentra entre los límites *out_inf* y *out_sup*.
- *TR_S:* Comando de inicialización. En nuestro caso, estas 2 últimas entradas irán controladas por el *Autotuning* que describiremos más adelante.

Salidas

- *OUT:* Valor absoluto de la acción calculada por el regulador.
- *OUTD:* Salida de valor incremental: diferencia entre la acción del ciclo actual y la del ciclo anterior

Una vez tenemos analizado el bloque que vamos a emplear, procederemos a su implementación en *Unity Pro XL* en el *Anexo B. Programa PLC*.

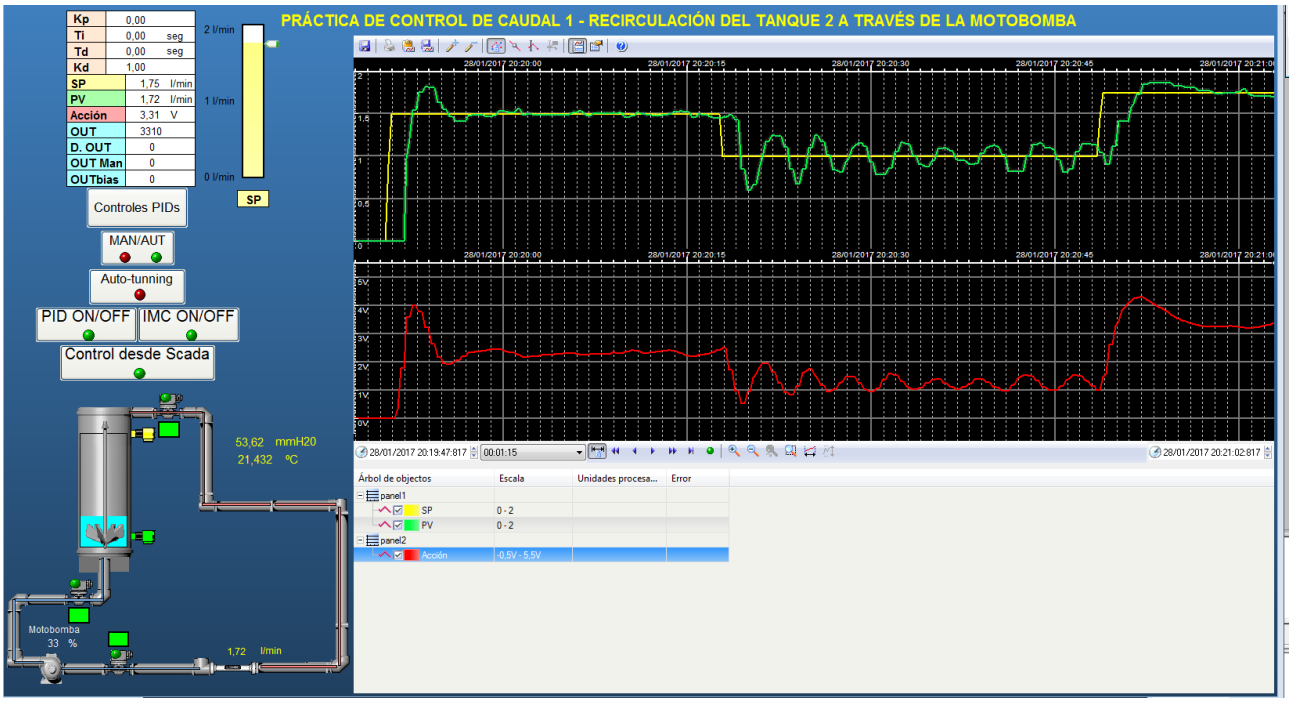


Figura 14. Ejemplo de control PID de caudal en SCADA

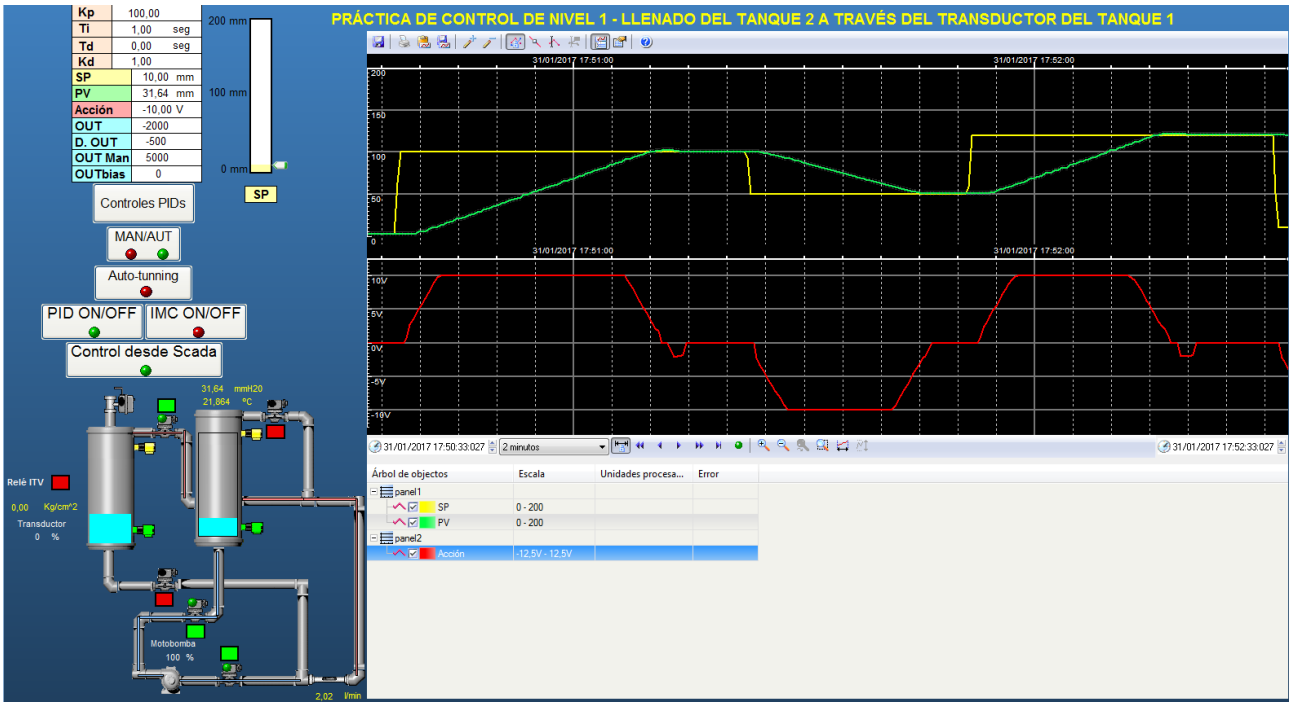


Figura 15. Ejemplo de control PID de nivel en SCADA

Se puede observar un video demostrativo del control PID de Caudal en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdslytnZp1_8sxvsg

Se puede observar un video demostrativo del control PID de Nivel en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdpOXVNY9HgOrD7wA

Se puede observar un video demostrativo del control PID de Temperatura en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdoz2QDjwq6rihUBw

3.2 Control ON/OFF

En este segundo punto se introduce uno de los controladores más sencillos que existen, el regulador "si/no", "encendido/apagado" o "todo/nada". Este tipo de controlador se basa en enviar una señal de activación al actuador cuando el valor que lee el sensor es menor que la consigna definida, para de esta manera alcanzar este valor; por el contrario, si se da la situación opuesta se desactiva la señal para hacer decrecer la variable controlada [13].

Este tipo de controles son enormemente utilizados en termostatos de aire acondicionado, ya que son procesos lentos los cuales muchas veces se pueden controlar de esta forma sencilla y económica. Estos activan el aire frío cuando la temperatura es mayor que la de referencia) y lo desactivan cuando la temperatura ya es menor (o igual) que la de referencia.

Como el objetivo de este proyecto no es controlar usando este tipo de regulador, simplemente haremos un pequeño ejemplo montando una alternativa a la opción primera de control de caudal con PID.

Buscamos en la documentación del autómata y encontramos que el bloque que permite este tipo de control es el *STEP2: Controlador de dos puntos*.

Una vez tenemos elegido qué tipo de controles vamos a hacer, nos vamos a la documentación del entorno de desarrollo del PLC (*Unity Pro XL*). En dicha documentación localizamos el bloque *PIDFF: Controlador PID completo*. Dicho bloque se basa en un algoritmo PID para realizar las funciones que queremos implementar.

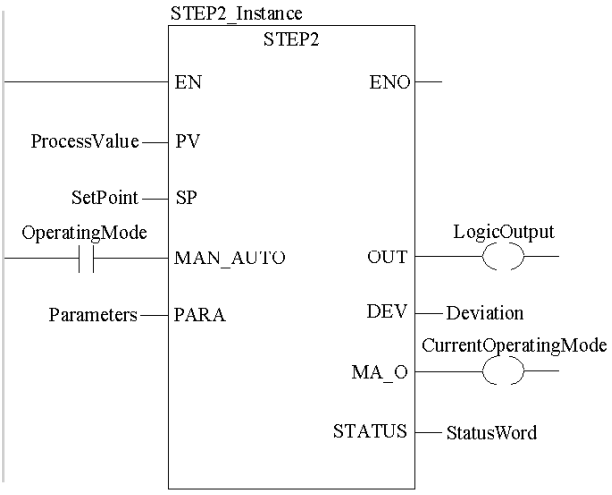


Figura 16. Bloque STEP2

De este bloque utilizaremos las siguientes entradas y salidas:

Entradas

- *EN:* Dicha entrada servirá para habilitar o deshabilitar el bloque *STEP2*.
- *PV:* Entrada donde escribiremos el valor actual de la variable a controlar.
- *SP:* Entrada donde establecemos el valor que queremos de la variable a controlar.
- *MAN_AUTO:* Entrada que determina si el bloque funciona en modo Automático o Mantener. En el primero, la salida *OUT* corresponde al resultado de los cálculos efectuados por el bloque, que la pondrá dicha salida a 0 o 1; en la segunda, la salida *OUT* mantiene el último valor calculado.
- *PARA:* Variable que contiene los parámetros a configurar del bloque.
  - *PV_INF:* Límite inferior del rango del valor real.
  - *PV_SUP:* Límite superior del rango del valor real.
  - *DEV_LL:* Valor umbral inferior de la desviación por debajo de la cual la salida *OUT* se activa.
  - *DEV_HL:* Valor umbral superior de la desviación a partir de la cual la salida *OUT* se desactiva.

Salidas

- *OUT:* Salida lógica. Esta variable habrá que transformarla en un valor numérico para poder imponerla en la motobomba.

Una vez hemos analizado el bloque, hacemos la configuración de prueba en *Unity Pro XL* en el *Anexo B. Programa PLC*.

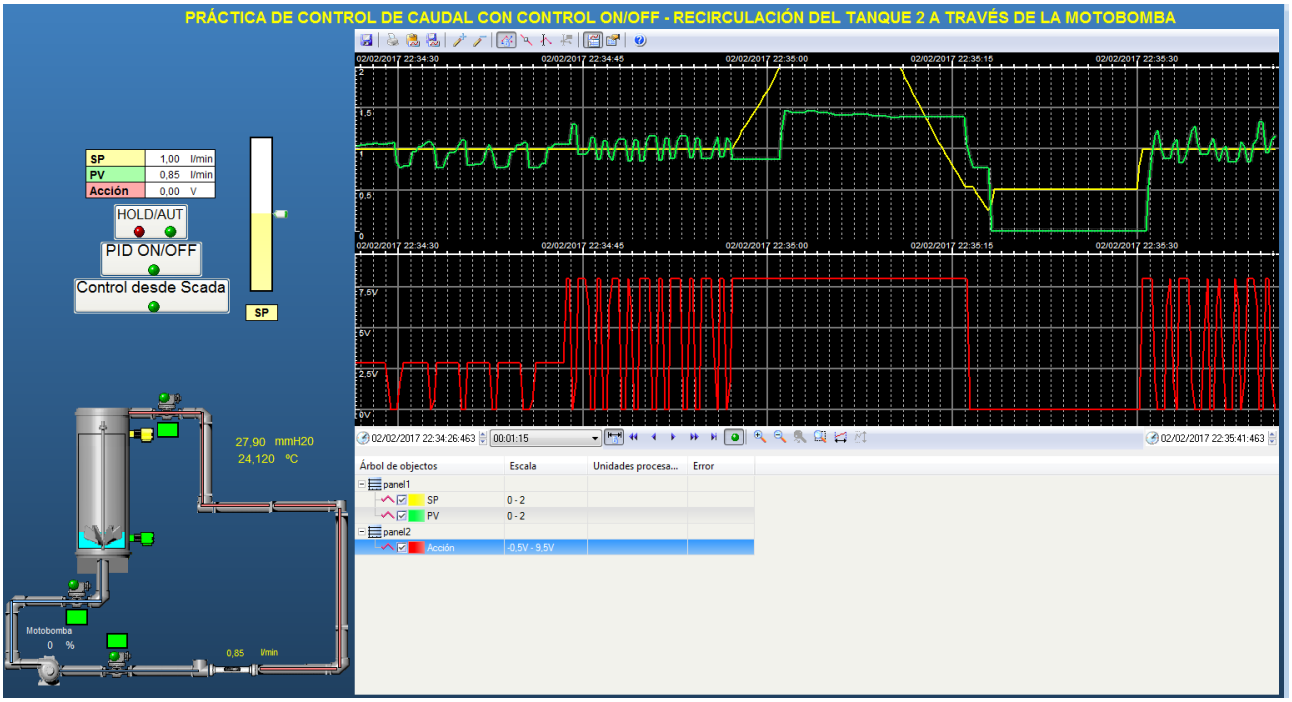


Figura 17. Ejemplo de control ON/OFF de caudal en SCADA

Se puede observar un video demostrativo del control ON/OFF de Caudal en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdnsz_UXZWOzRx3jg

3.3 Autotuning

En este apartado se introduce el concepto de Autotuning, que sirven de apoyo a los PID expuesto en el apartado anterior.

El Autotuning permite que, durante el inicio de la instalación, se haga un ajuste automático del regulador para estabilizarlo la regulación y, por tanto, ahorra tiempo a la hora de poner en marcha el sistema al proporcionar unos parámetros óptimos que pueden ser impuestos directamente. También permite utilizarlo más adelante si se diera el caso que quisiéramos obtener unos nuevos parámetros, por ejemplo, si se ha hecho alguna modificación en el sistema.

El Autotuning puede ser aplicado a los siguientes tipos de procesos:

- Procesos con sólo una entrada/salida
- Procesos con estabilidad natural o con componente integral
- Procesos asimétricos dentro de los límites permitidos por el algoritmo del PID

Los Autotuning de colocan en conjunto con los PID, permitiendo establecer sus parámetros óptimos cuando los primeros son ejecutados. A continuación, estudiamos la posibilidad de implementarlo en nuestros controles:

- Control de caudal: en este caso, conectaremos el Autotuning al PID, siendo este primero el que gobierne la acción sobre el segundo, para de esta manera hacer el proceso necesario para obtener los parámetros. Se ha decidido que dicha configuración se haga sobre la primera combinación de control (recirculación del agua en el tanque central), ya que es la única de las 3 opciones que permite una ejecución ininterrumpida en el tiempo.
- Control de temperatura: al igual que en el caso anterior, utilizaremos el Autotuning en conjunto con el PID para hacer la configuración necesaria para obtener los parámetros. El Autotuning irá imponiendo valores de acción sobre las placas Peltier que conseguirán modelar el sistema y conseguir dichos parámetros.
- Control de nivel: para calcular los valores de este regulador con el control de nivel se ha decidido hacer para la segunda modalidad, ya que el regulador de presión proporcional del tanque izquierdo no es apropiado para este objetivo. La acción necesaria del Autotuning esta vez se impondrá sobre la motobomba o sobre la electroválvula proporcional, según sea necesario sacar agua o introducirla en el tanque centrar respectivamente.
- Control de presión: como ya se ha mencionado anteriormente, al ser la regulación de la presión del tanque izquierdo independiente del usuario, no podremos variar los parámetros que por construcción tiene el regulador, y por tanto no será posible realizar el Autotuning sobre este control.

Una vez tenemos elegido sobre qué reguladores vamos a poner los Autotuning, nos vamos a la documentación de *Unity Pro XL* al igual que el caso anterior. En la documentación localizamos el bloque *AUTOTUNE: Ajuste automático del sintonizador*, que es el que necesitamos.

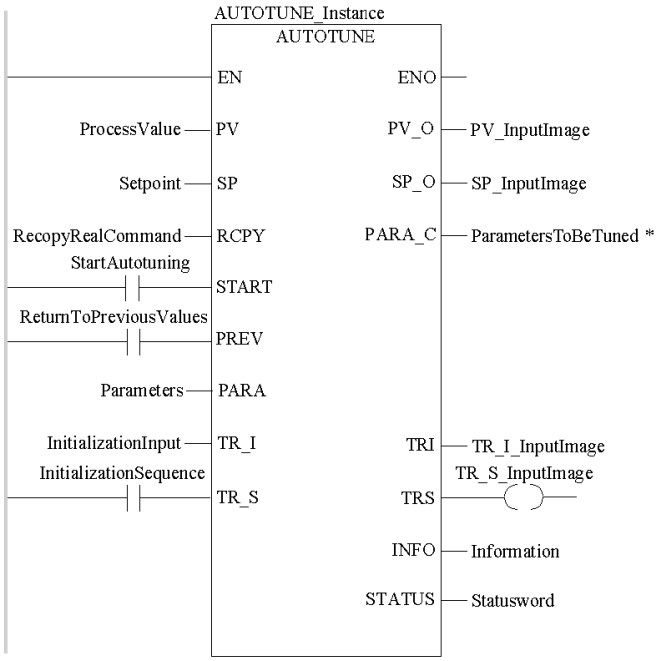


Figura 18. Bloque AUTOTUNE

El algoritmo se basa en reglas heurísticas, como, por ejemplo, el método de Ziegler-Nichols. Primero se efectúa un análisis que corresponde aproximadamente a dos veces y media el tiempo de reacción del bucle abierto. De esta forma se puede identificar el proceso como proceso de primer orden con retardo. A partir de este modelo se crea un juego de parámetros de regulación basado en reglas heurísticas y valores experimentales.

De este bloque utilizaremos las siguientes entradas y salidas:

Entradas

- *EN:* Dicha entrada servirá para habilitar o deshabilitar el bloque *AUTOTUNE*.
- *PV:* Entrada donde escribiremos el valor actual de la variable a controlar.
- *SP:* Entrada donde establecemos el valor que queremos de la variable a controlar.
- *START:* Variable para activar el *AUTOTUNE*. El bloque se activa en la transición de esta variable de 0 a 1. La variable debe permanecer a 1 durante la ejecución del bloque. En caso contrario se cancela el sintonizado.
- *PARA:* Variable que contiene los parámetros a configurar del bloque.
  - *STEP_AMPL:* Valor del impulso de posicionado de salida en porcentaje (entre 0 y 100). Este valor viene tabulado en función de la aplicación del PID (20 para control de caudal y nivel; 50 para control de temperatura)
  - *TMAX*: Duración del impulso de posicionado con ajuste automático del regulador. Después de este tiempo, hay un pequeño retraso en el cual se calculan los parámetros y se sustituyen en el PID.

- *TR_I:* Entrada de inicialización.
- *TR_S:* Comando de inicialización. En este caso, como no vamos a utilizar estas 2 últimas entradas, las desactivaremos anulando esta última.

Salidas

- *PV_O:* Copia de la entrada *PV* que hace de puente para al PID al que va unido.
- *SP_O:* Copia de la entrada *SP* que hace de puente para al PID al que va unido.
- *TRI:* Esta salida se conecta a la entrada *TR_I* del PID al que va unido. Durante la ejecución del bloque, el *AUTOTUNE* envía por esta variable la acción a imponer sobre el actuador.
- *TRS:* Variable que indica al PID que el *AUTOTUNE* está activado, y por lo tanto que debe aplicar la acción que recibe por la entrada *TR_I*.

Después de haber caracterizado el bloque, procederemos a su implementación en *Unity Pro XL* en el *Anexo B. Programa PLC*.

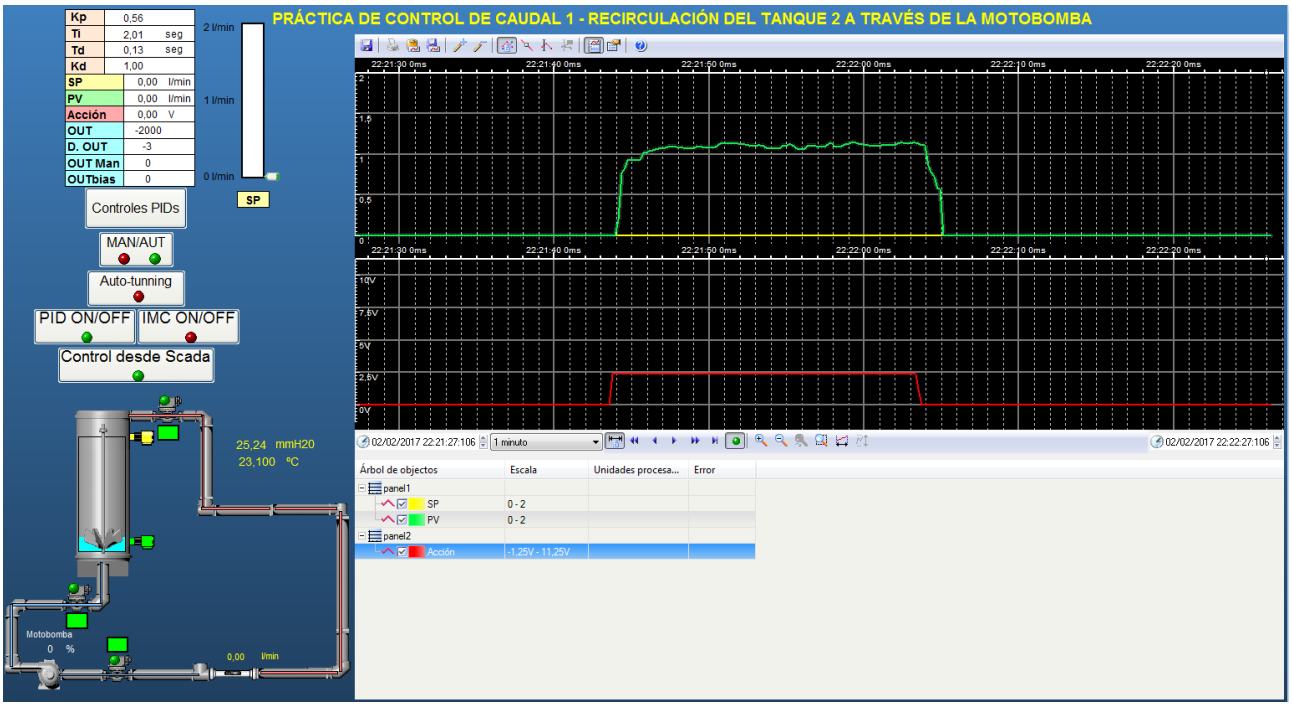


Figura 19. Ejemplo de Autotuning de caudal en SCADA

Se puede observar un video demostrativo del Autotuning de Caudal en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdqsuhOjcQVOV1QRg

3.4 Control IMC

En este apartado se introduce el concepto de control por IMC (o corrector del modelo), que sirve de sustituto al clásico control PID.

El corrector de modelos permite afrontar los retardos graves, si existen, comparándolos con la constante de tiempo principal del proceso; esta situación no se puede resolver satisfactoriamente mediante un control de procesos estándar PID. El corrector de modelos también es importante para regular procesos no lineales.

Una vez más, nos vamos a la documentación de *Unity Pro XL* y localizamos el bloque *IMC, Corrector del modelo*, que nos hará la función que deseamos.

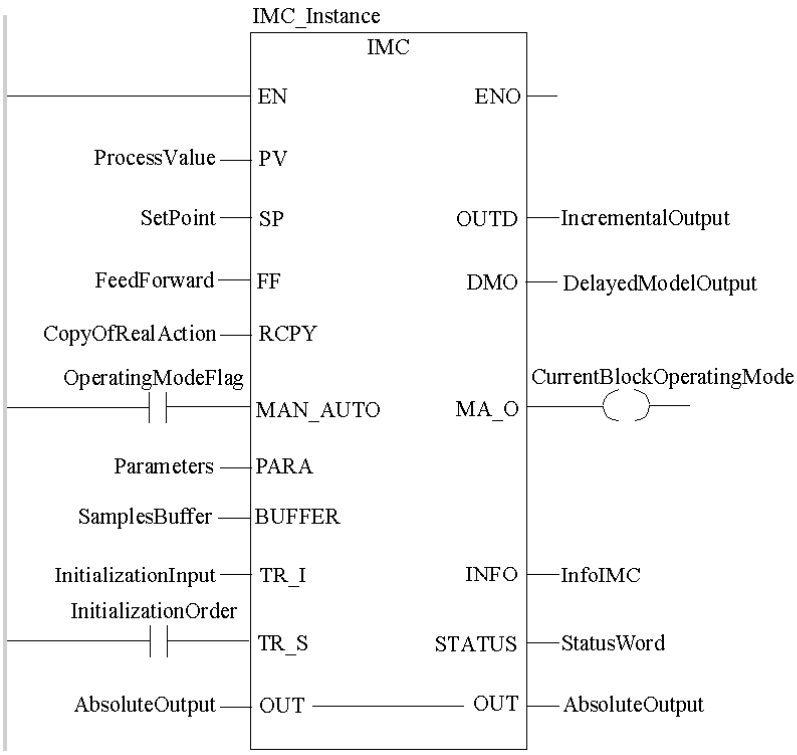


Figura 20. Bloque IMC

El modelo es de primer orden + retardo. Sin embargo, este corrector puede tratar cualquier proceso estable y no periódico en cualquier orden.

De este bloque utilizaremos las siguientes entradas y salidas:

Entradas

- *EN:* Dicha entrada servirá para habilitar o deshabilitar el bloque IMC.
- *PV:* Entrada donde escribiremos el valor actual de la variable a controlar.
- *SP:* Entrada donde establecemos el valor que queremos de la variable a controlar.
- *MAN_AUTO:* Entrada que determina si el bloque funciona en modo Automático o Manual. En el primero, las salidas *OUT* y *OUTD* corresponden al resultado de los cálculos efectuados por el bloque; en la segunda, la salida *OUT* no se ajusta por medio del bloque. El usuario puede ajustar este valor directamente.

- *PARA:* Variable que contiene los parámetros a configurar del bloque.
  - *PV_INF:* Límite inferior del rango del valor real.
  - *PV_SUP:* Límite superior del rango del valor real.
  - *OUT_INF:* Límite inferior del rango de valores de salida cuando el controlador está en modo Manual.
  - *OUT_SUP:* Límite superior del rango de valores de salida cuando el controlador está en modo Manual.
  - *REV_DIR:* Variable que indica si el controlador funciona de modo directo u opuesto. En nuestro caso, lo estableceremos en modo directo.
  - *OUT_MIN:* Límite inferior del rango de valores de salida cuando el controlador está en modo Automático.
  - *OUT_MAX:* Límite superior del rango de valores de salida cuando el controlador está en modo Automático.
  - *KS:* Ganancia estática del proceso en bucle abierto. La ajustaremos dividiendo la lectura del valor de la variable a controlar en bucle abierto (*PV*) y la acción necesaria para obtener dicho valor (*OUT*).
  - *OL_TIME:* Contante de tiempo del proceso el bucle abierto. La mediremos dividiendo para 3 el tiempo que tarda en completar el transitorio el sistema en el caso anterior.
  - *T_DELAY:* Tiempo de retardo puro. Es el tiempo que tarda el sistema en reaccionar ante un cambio en *OUT*.
  - *T_ECH:* Periodo de muestreo al que el bloque tomará los valores. Dejamos el valor por defecto de 0.3 segundos.
- *TR_I:* Entrada de inicialización.
- *TR_S:* Comando de inicialización. En este caso, como no vamos a utilizar estas 2 últimas entradas, las desactivaremos anulando esta última.

Salidas

- *OUT:* Valor absoluto de la acción calculada por el regulador.
- *OUTD:* Salida de valor incremental: diferencia entre la acción del ciclo actual y la del ciclo anterior

Como ha podido verse, al contrario que para el PID los valores a ajustar son *KS*, *T_DELAY* y *OL_TIME*, no siendo necesario proporcionar al IMC unos valores de regulación, sino que el propio corrector de modelo realizará la regulación con la mejor dinámica que se pueda conseguir.

Ahora que ya tenemos el bloque estudiado, lo implementamos en *Unity Pro XL* en el *Anexo B. Programa PLC*.

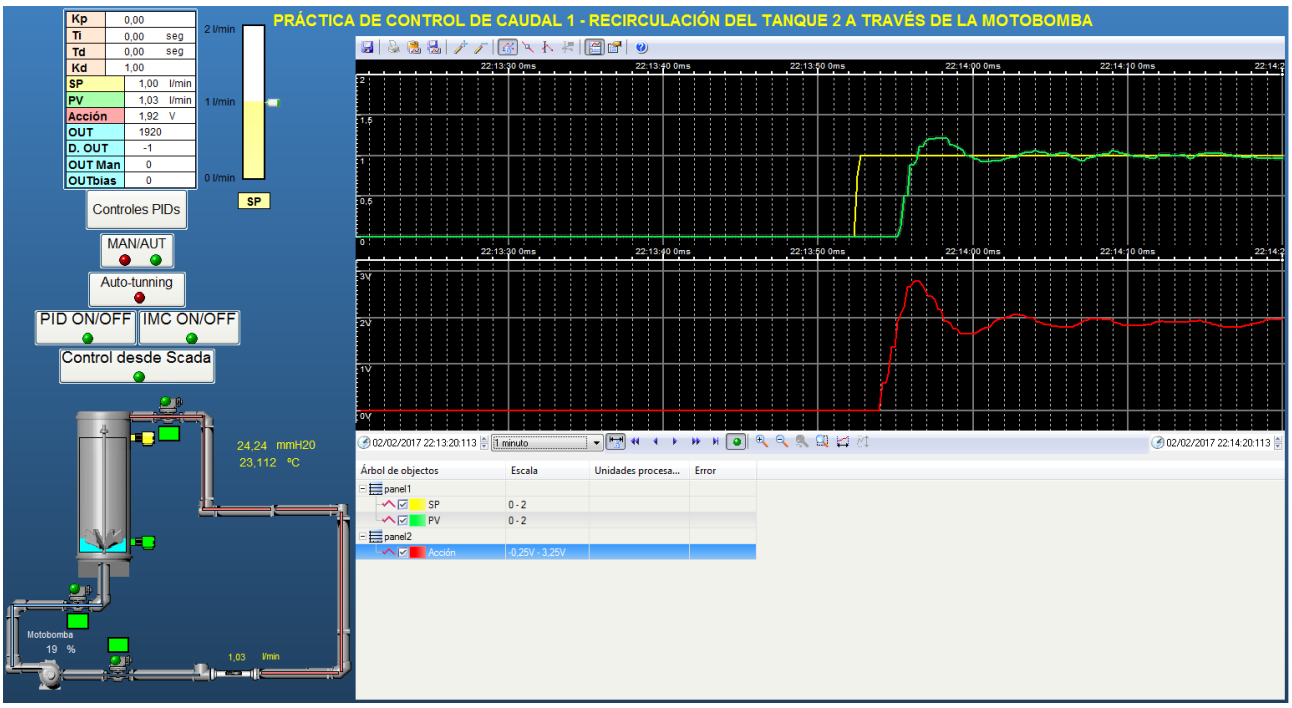


Figura 21. Ejemplo de control IMC de caudal en SCADA

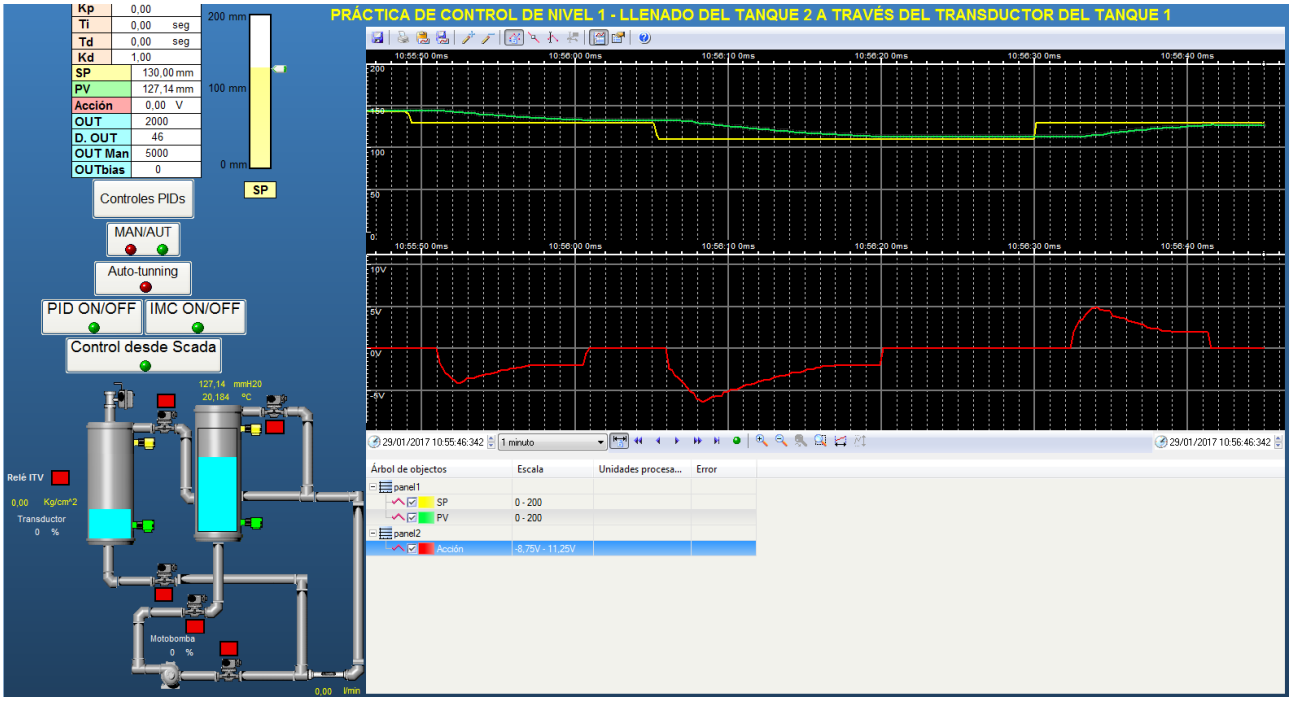


Figura 22. Ejemplo de control IMC de nivel en SCADA

Se puede observar un video demostrativo del control IMC de Caudal en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdrNvZpMNycpcXN7A

Se puede observar un video demostrativo del control IMC de Nivel en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIdmhY2WGeq6skfR5A

Capítulo 4. Supervisión de la estación

En este capítulo, se desarrollan los apartados correspondientes con la supervisión de la estación, empezando por la comunicación de la misma con el sistema SCADA, el desarrollo de pantallas gráficas de operador en el propio entorno de desarrollo del autómata, y la comunicación con Matlab para realizar varias funciones.

4.1 Comunicación con SCADA

En primer lugar, deberemos crear la comunicación necesaria para poder interconectar el autómata a un sistema SCADA, para realizar varias funciones:

- Control manual de todos los sensores y actuadores de la maqueta.
- Realizar unas pantallas para cada control elegido en el *Capítulo 3. Control de la estación de depósitos*, para poder controlarlos, variar sus parámetros y hacer gráficas de los datos.
- Activar y visualizar un sistema de demostración automático.

El sistema SCADA con todas sus pantallas y controles creado para este proyecto se detalla en el *Anexo C. Sistema SCADA*.

La comunicación entre el autómata y el PC donde estará ejecutándose el sistema SCADA la haremos mediante el protocolo '*MODBUS TCP/IP*', que es un protocolo de comunicaciones basado en la arquitectura cliente/servidor (TCP/IP) diseñado por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de comunicaciones estándar en la industria, es el que goza de mayor disponibilidad para la conexión de dispositivos como el que estamos empleando. Las razones por las cuales el uso de *Modbus* es superior a otros protocolos de comunicaciones son [14]:

- Es público.
- Su implementación es fácil y requiere poco desarrollo
- Maneja bloques de datos sin suponer restricciones
- Se puede conectar a la red mediante puerto serie o Ethernet (Modbus/TCP).

Por lo tanto, como tanto el autómata como el PC están conectados mediante Ethernet a la red de la universidad, optaremos por este protocolo para hacer la comunicación entre ambos. Todo el proceso de configuración de la comunicación, así como un ejemplo de funcionamiento de una aplicación SCADA sencilla viene explicado en el *Anexo F. Seminario sobre diseño y programación de una aplicación de supervisión (SCADA)*.

4.2 Desarrollo pantallas gráficas

No solo se puede controlar el PLC desde una aplicación SCADA, sino que esta labor puede ser realizada también desde el propio entorno de desarrollo del PLC (*Unity Pro XL*).

Esto se consigue mediante las llamadas "*pantallas de operador*", que permite implementar una funcionalidad similar que el sistema SCADA con un entorno gráfico más sencillo, pero al uso suficiente para la tarea que necesitamos de controlar los aspectos básicos del PLC. Al igual que con SCADA, se crearán estas pantallas con el objetivo de realizar los controles de los PID nombrados en el *Capítulo 3. Control de la estación de depósitos*.

Para usar dichas pantallas solamente será necesario tener una conexión activa entre el PC y el PLC, es decir, que ambos estén conectados por el protocolo mencionado en el punto anterior. El desarrollo y explicación de dichas pantallas se hará en el *Anexo B. Programa PLC*.

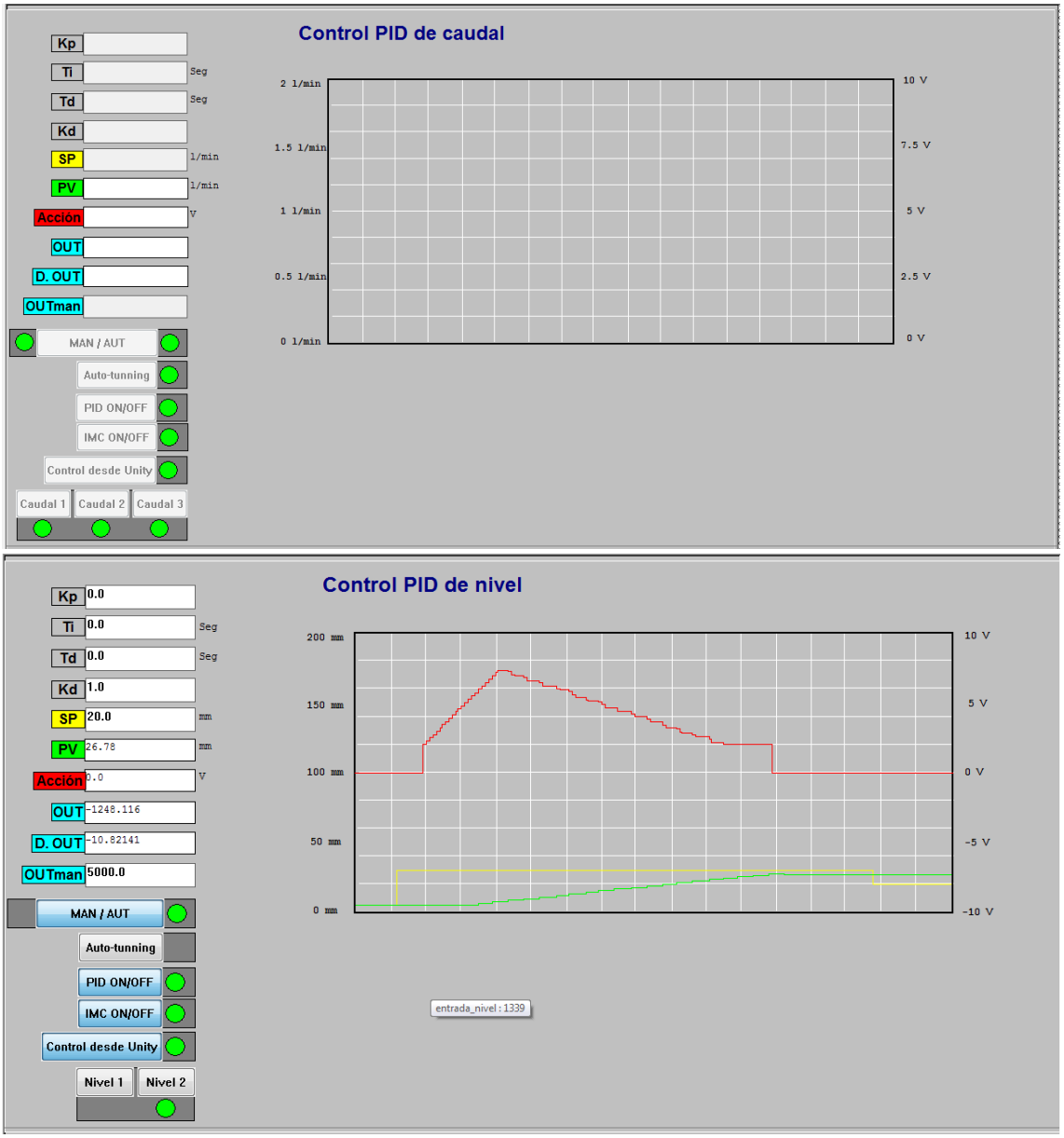


Figura 23. Ejemplo de pantallas gráficas de operador en Unity

4.3 Comunicación con Matlab

Por último, tal y como se especificó en los objetivos del proyecto, debemos crear un canal de comunicación entre Matlab y el PLC para realizar las siguientes funciones:

- Representación gráfica de los controles de temperatura, presión en el tanque 1, caudal y nivel que se expusieron en el Capítulo 3, para poder tener un paralelismo en la representación con respecto al SCADA. De dichos controles representaremos la consigna, la realimentación captada por los sensores y la acción que imponen los PIDs.
- Habilitar una pantalla en la *interface* gráfica de Matlab que permite cambiar entre los varios tipos de control de caudal y de nivel.
- Poder habilitar desde Matlab un modo de control en el cual se tome el control del autómata y sea el propio Matlab el que lea la consigna y la realimentación, y calcule la acción a aplicar. Este control se hará solamente sobre 1 de los controles, el cual elegido es el primer tipo de control de caudal (recirculación del agua en el tanque central). En esta modalidad, el PLC funciona solamente leyendo la acción calculada en Matlab y volcándola sobre el actuador correspondiente, a la vez que envía la señal de los sensores de vuelta a Matlab.

El diseño e implementación de las funciones expuestas está desarrollado en el *Anexo D. Programa Matlab*. A continuación, se explica el sistema de comunicación elegido entre Matlab y el PLC necesario para llevar a cabo lo dicho anteriormente.

Matlab no permite de manera predeterminada la comunicación con PLCs industriales, por lo que será necesario utilizar una aplicación intermedia para este fin. Entre la documentación de Matlab encontramos varios métodos para hacer esta comunicación [15]:

- *Simulink PLC Coder*: Esta *toolbox* de Matlab genera código en formato de texto estructurado IEC 61131 y funciones de Matlab embebidas. El texto estructurado es generado en PLCOpen XML y otros formatos soportados por la mayoría de entornos de desarrollo. Por esta razón, se pueden compilar aplicaciones para PLCs.
- *OPC Toolbox*: Esta *toolbox* puede ser usada directamente en código Matlab o Simulink para comunicarse con un servidor OPC previamente configurado y conectado en el PC. Esta *toolbox* es un conjunto de funciones y bloques que permiten la manipulación de los datos almacenados en el servidor en tiempo real (siempre y cuando sincronicemos la simulación con el reloj de sistema), tanto la lectura como la escritura. Estos bloques pueden conectarse directamente como entradas o salidas (respectivamente) del modelo.
- *Real-Time Workshop Embedded Coder*: Algunos PLCs puede ser programados en código C generado por Simulink y Matlab que facilita la programación de una manera clásica. Este código generado es extraordinariamente compacto y rápido. Este soporto para que sea fácilmente integrable se consigue mediante aplicaciones, funciones y datos heredados. Este conjunto de librerías está soportado por la mayoría de estándares industriales como IEC 61508 o ISO 26262.
- *Simulink Real-time/Speedgoat*: Usado para comunicar a través de protocolos *fieldbus*. Matlab permite este tipo de protocolos a parte del Ethernet/IP y Modbus a través de

una aplicación de Simulink en tiempo real (*fieldbus* requiere que sea en tiempo real). La mayoría de PLCs soportan alguna modalidad de protocolo *fieldbus*, dependiendo del hardware.

- *Instrument Control toolbox*: Esta *toolbox* está pensada para establecer una comunicación serie con el PLC, ya que tiene los drivers de RS-232 para implementar en Simulink.

Una vez vistos los métodos anteriores, y percibiendo la posibilidad y rapidez de su implementación se ha optado por elegir hacer un servidor OPC en el PC que se comunique con el PLC, y que sea Simulink el que maneje los datos a través de la *toolbox* de Matlab. Como servidor OPC, nos descargaremos el *Modbus Ehternet OPC Server*, que es fácil de usar y de configurar y además es gratuito [16].

Una vez que tenemos instalado el programa, se ha creado un nuevo servidor OPC que lo hemos denominado *PLC_depositos*. La configuración elegida se muestra en la siguiente figura.

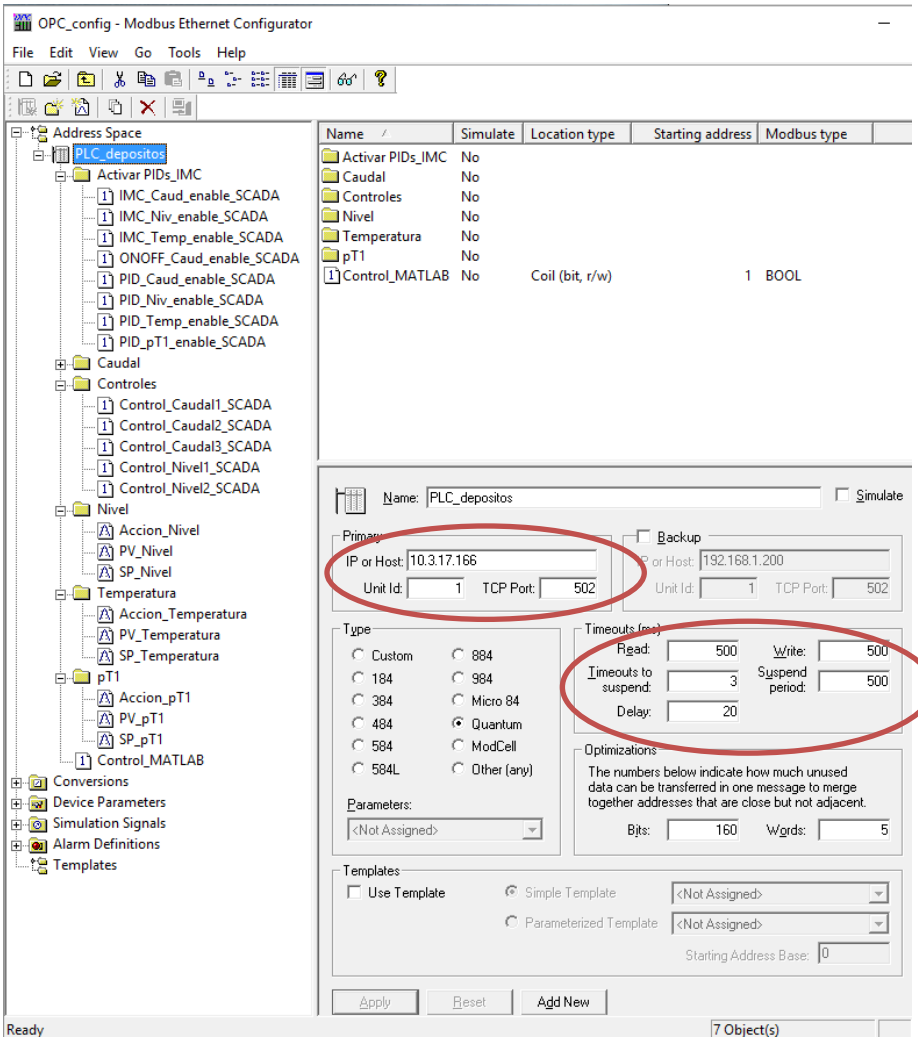


Figura 24. Configuración del OPC

Los elementos más importantes son la dirección IP y los periodos de adquisición de datos (marcados en rojo). Una vez tenemos esto, añadiremos tantos *ítems* como variables queramos leer y/o escribir en el autómata. En nuestro caso, hemos puesto una variable para comunicar al PLC con Simulink para indicarle al primero que ceda el control para usar el PID hecho en Simulink

para el control de caudal. Por otra parte, se han hecho 6 agrupaciones para facilitar su lectura: *Caudal*, *Nivel*, *Temperatura*, y *pT1* (presión en el tanque izquierdo) contendrán las variables que muestran los valores de acción, valor leído del sensor y consigna (respectivamente) de estos 4 PID en el autómata; el grupo *Controles* contiene las variables que desde Simulink permitirán cambiar entre los tipos de control de caudal y de nivel; por último, el grupo de *Activar PIDs_IMC* tiene una serie de variables para poder habilitar o deshabilitar los PIDs y IMC del programa del PLC.

En la siguiente figura se muestra como configurar dichas variables, tomando de ejemplo la primera mencionada de activación del control desde Matlab.

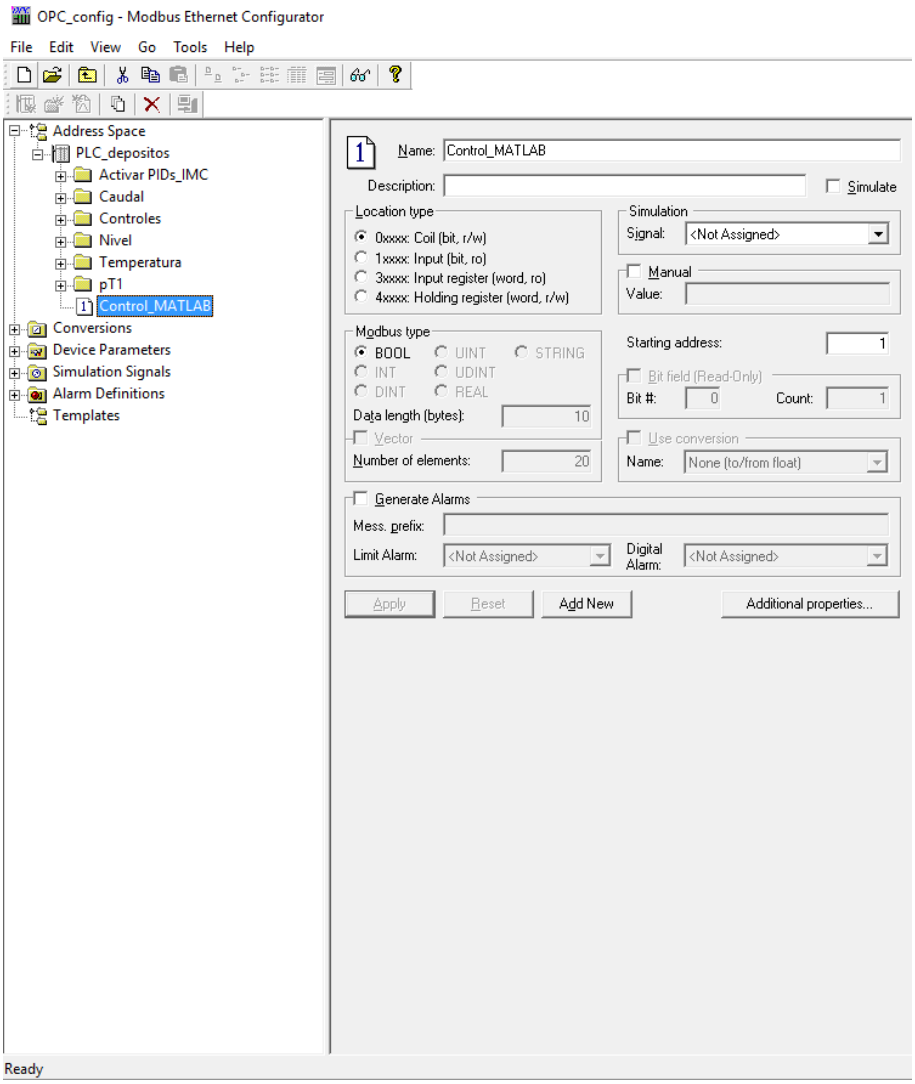


Figura 25. Configuración de variables

Como podemos ver, fundamentalmente deberemos elegir qué tipo de variable es (lógica, entera, real, …), así como su dirección de memoria y si es una variable de sólo lectura (*Input/Input register*) o por el contrario también es de escritura (*Coil/Holding register*).

Una vez tenemos esto, pasamos a crear el programa *Simulink*, desarrollado en el *Anexo D. Programa Matlab*. Se muestra a continuación los archivos Simulink para representación gráfica de variables y de control PID de Caudal desde Matlab, así como la ejecución de esta última.

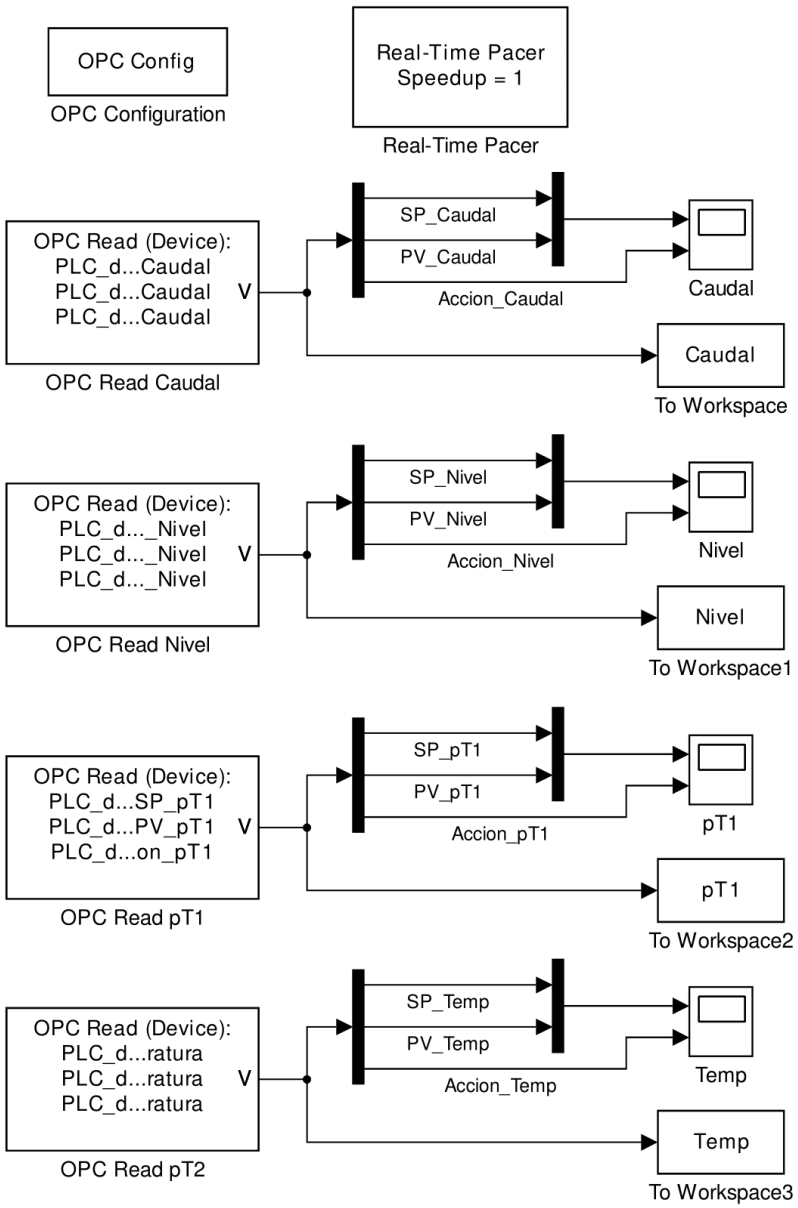


Figura 26. Visualización gráfica de las variables de los reguladores del PLC en Matlab

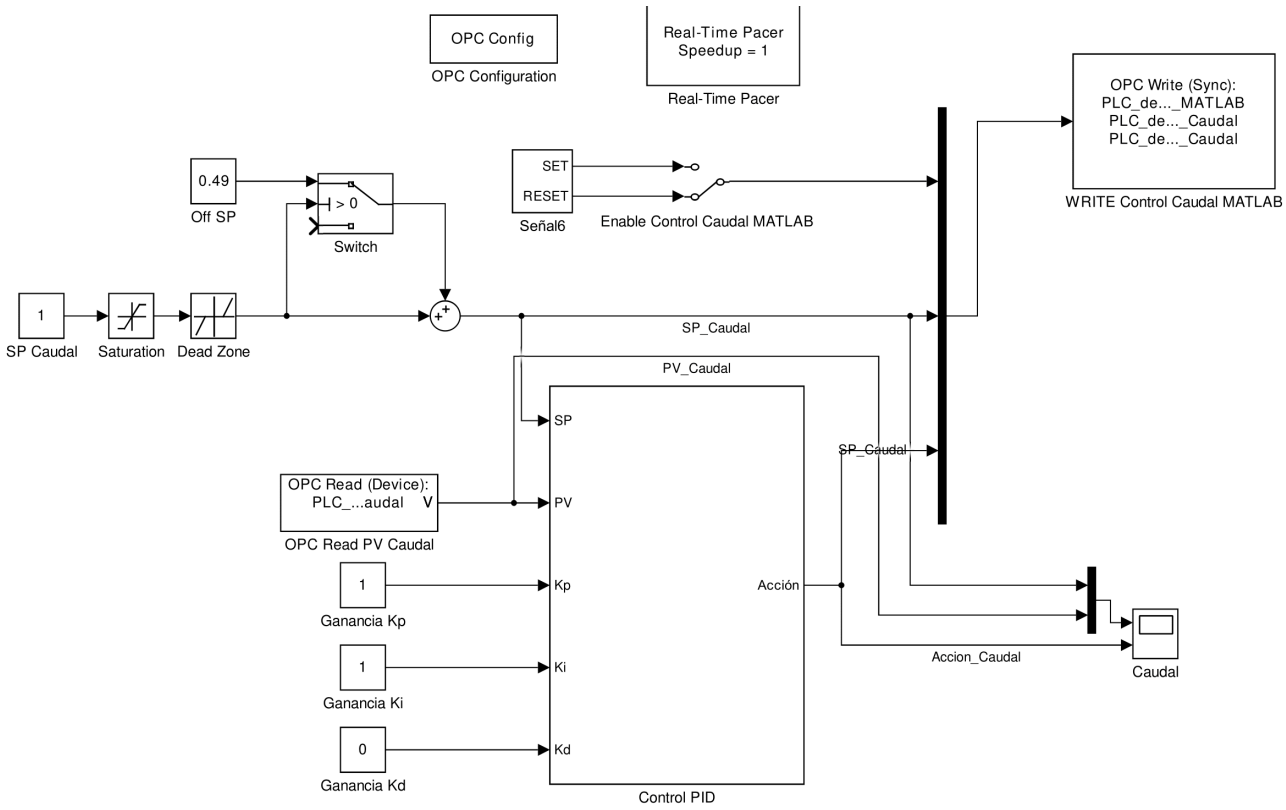


Figura 27. Control PID de Caudal tipo 1 desde Matlab

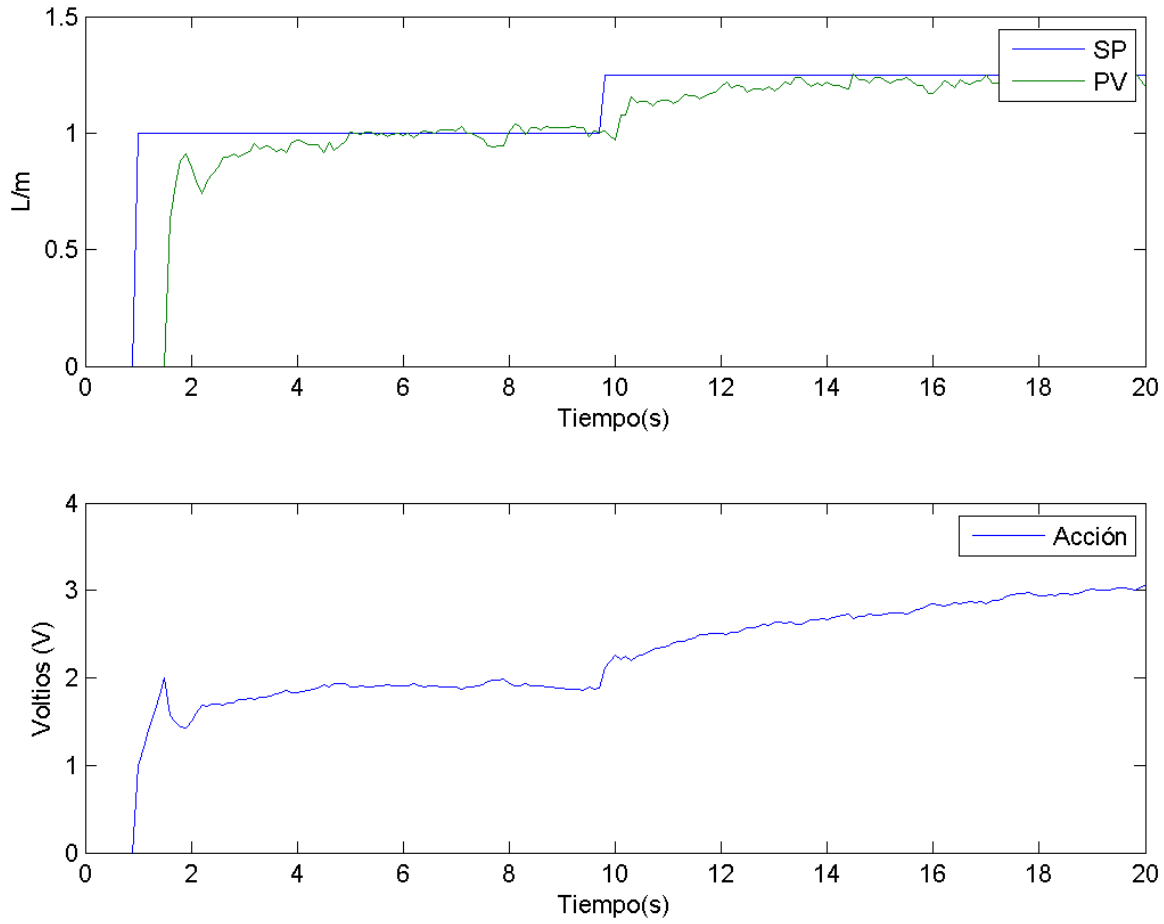


Figura 28. Gráfica obtenida del control de Caudal a través del PID en *Simulink*

Capítulo 5. Desarrollo de un sistema de demostración Automático

En este capítulo se estudia la viabilidad de desarrollar un sistema de demostración automático del programa del PLC, que permitirá recorrer de manera autónoma diversos aspectos de la maqueta, como son los PID, Autotuning e IMC. También se irá mostrando por una pantalla del SCADA la explicación de dichos controles.

Se puede observar un video demostrativo de la clase Automática en este link: https://1drv.ms/v/s!Aqx4_6xP88_RiIoy6yAqHNU23uxJKA

5.1 La clase Automática

En primer lugar, en este punto vamos a establecer los elementos que contendrá este sistema de demostración, así como su organización y ejecución.

Una vez que tenemos estudiados las posibilidades de control sobre la maqueta, se decide hacer la siguiente estructura para la demostración:

- Primero deberemos preparar la maqueta para la demostración, para lo cual transferiremos toda el agua de los 3 depósitos al de la izquierda.
- Proceso de transvase entre depósitos: transferimos agua entre los depósitos de manera temporizada para comprobar que los actuadores funcionan correctamente. Primero del tanque central al derecho, luego del derecho al izquierdo, y por ultimo de vuelta al central. Este ciclo lo haremos 3 veces completas.
- Control de nivel del agua en el depósito intermedio con ayuda del depósito izquierdo, para lo cual utilizaremos el PID puesto para este fin con una consigna de 100 milímetros de altura. Lo temporizaremos a 30 segundos para asegurarnos que se alcanza la altura deseada.
- Autotuning de nivel para obtener los parámetros de control para el siguiente paso. Usaremos el tanque derecho como apoyo, esperando a la señal de finalización de este para continuar.
- Control de nivel del agua en el depósito intermedio con ayuda del depósito derecho, utilizando el PID y los parámetros calculados en el paso anterior. Iremos alternando entre consignas de 20 y 80 milímetros, ciclo que repetiremos 3 veces cambiando entre ellos cada 30 segundos.
- Control de caudal con regulador ON/OFF para enseñar su funcionamiento. Lo ejecutaremos sobre la primera modalidad de recirculación, recirculando el agua desde el tanque intermedio a través de la motobomba.
- Autotuning de caudal para obtener los parámetros de control para el siguiente paso. Lo haremos también sobre la primera modalidad
- Control de caudal recirculante por el tanque central utilizando su PID y los parámetros calculados en el Autotuning. Iremos variando entre consignas de 1 y 1.5 litros/minuto, ciclo que repetiremos durante 2 minutos alternando cada 20 segundos.

- Control de temperatura del depósito intermedio con PID, en el que diferenciaremos 2 casos: si la temperatura ambiente está a menos de 25º, la regularemos para alcanzar este valor, estableciendo que la temperatura ambiente es *FRIA*; si por el contrario el ambiente está por encima de este umbral, pondremos como condición alcanzar una temperatura 5 grados superior a la del ambiente, estableciendo que la temperatura ambiente es *CALIENTE*. Una vez que hayamos determinado esta condición, la mantendremos para el resto de etapas de control de temperatura. El paso al siguiente paso se hará una vez superados los 10 minutos.
- Autotuning de temperatura del depósito intermedio para obtener los parámetros de control para los siguientes pasos de temperatura.
- Control de temperatura del depósito intermedio con PID en el que volvemos a diferenciar 2 casos: si la temperatura ambiente es *FRIA*, regularemos para obtener 30º; si por el contrario es *CALIENTE*, regularemos para obtener 10 grados por encima de la temperatura ambiente obtenida en el primer control.
- Ahora volvemos a hacer los mismos 2 controles de temperatura, pero esta vez utilizando el IMC para la regulación en vez del PID.
- Por último, volvemos a hacer un control de nivel del agua en el depósito intermedio con ayuda del depósito izquierdo con el PID, pero esta vez con una consigna de 20 milímetros. Lo temporizaremos también a 30 segundos para asegurarnos que se alcanza la altura deseada.

Una vez tenemos determinado la estructura del programa de demostración, procederemos a su implementación en *Unity Pro XL* y a la creación de las pantallas en SCADA.

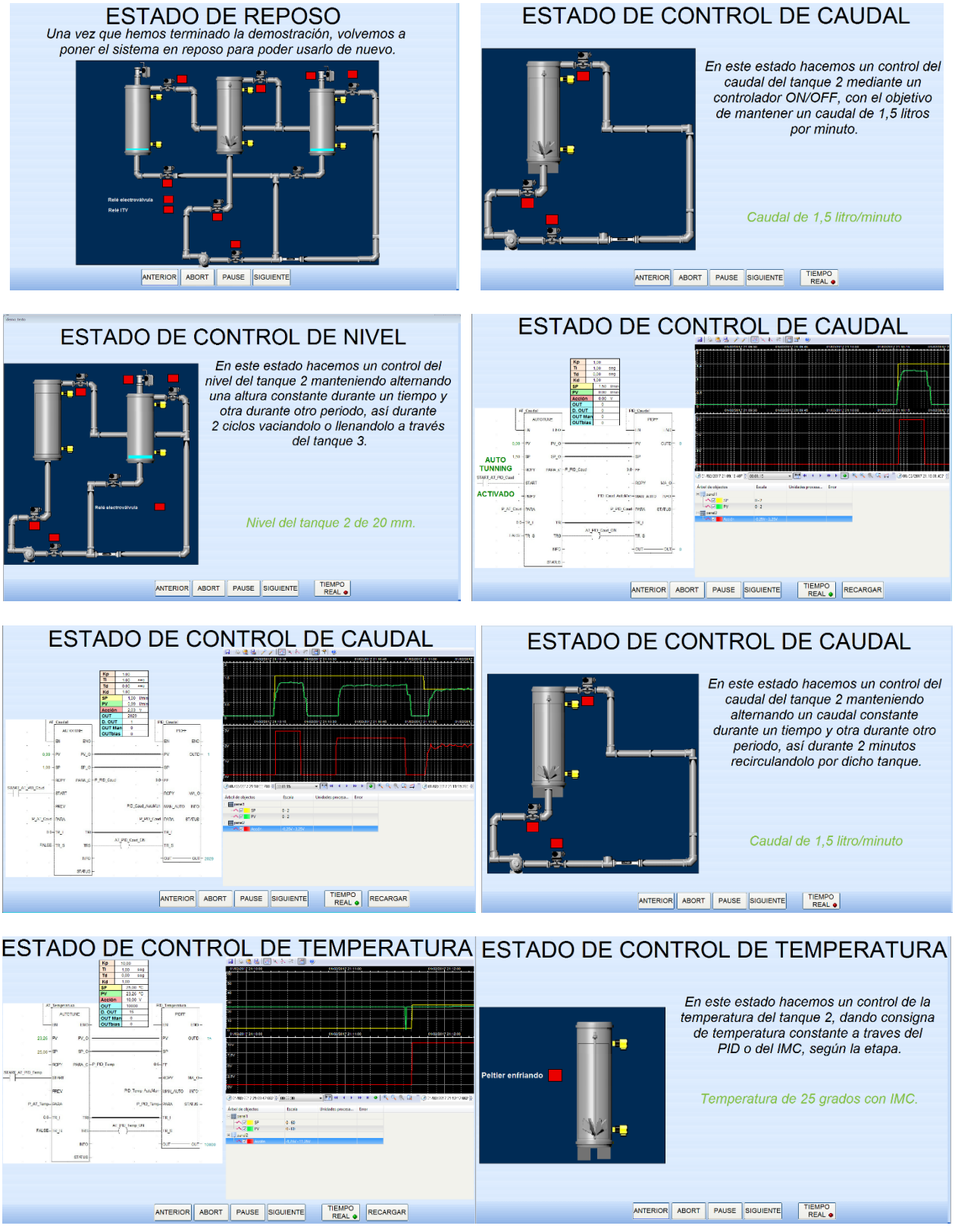


Figura 29. Ejemplos de pantallas de la demo

5.2 SFC de control

Para implementar todo lo anteriormente mencionado se ha decidido hacer en un nuevo módulo SFC, debido a su facilidad y rapidez de integración y a su claro entendimiento visual como diagrama de flujo.

Los estados de dicho SFC serán todos y cada uno de los puntos explicados en el apartado *5.1*, siendo las transiciones las condiciones de salida de estas, ya sean por tiempo o por otras condiciones como:

- Se ha decidido implementar la posibilidad de que la demostración se salte el estado que se está ejecutando en ese momento, pasando inmediatamente al siguiente. En el caso de la temperatura, si hacemos esto en cualquiera de sus estados, pasaremos al último de nivel.
- También se ha habilitado la opción de poder volver al estado anterior, saltándose el estado que se está ejecutando en ese momento, poniendo como límite que no podemos volver más atrás del estado de preparación.
- Otra opción es poder abortar la demo en caso de desearlo.
- Así mismo, se puede pausar la demostración, es decir, que no pasará al estado siguiente, aunque se cumpla la condición de finalización.
- Por último, la posibilidad de alternar entre 2 pantallas diferentes, una que muestra una explicación de lo que se está haciendo en cada momento junto con un esquema de los sensores y actuadores empleados; otra que muestra en tiempo real los valores de las variables que intervienen en el control.

Una vez que hemos establecido qué elementos queremos tener en la demostración, pasamos a su implementación tanto en el PLC (*Anexo B. Programa PLC*) como la creación de las pantallas en SCADA. (*Anexo C. Sistema SCADA*).

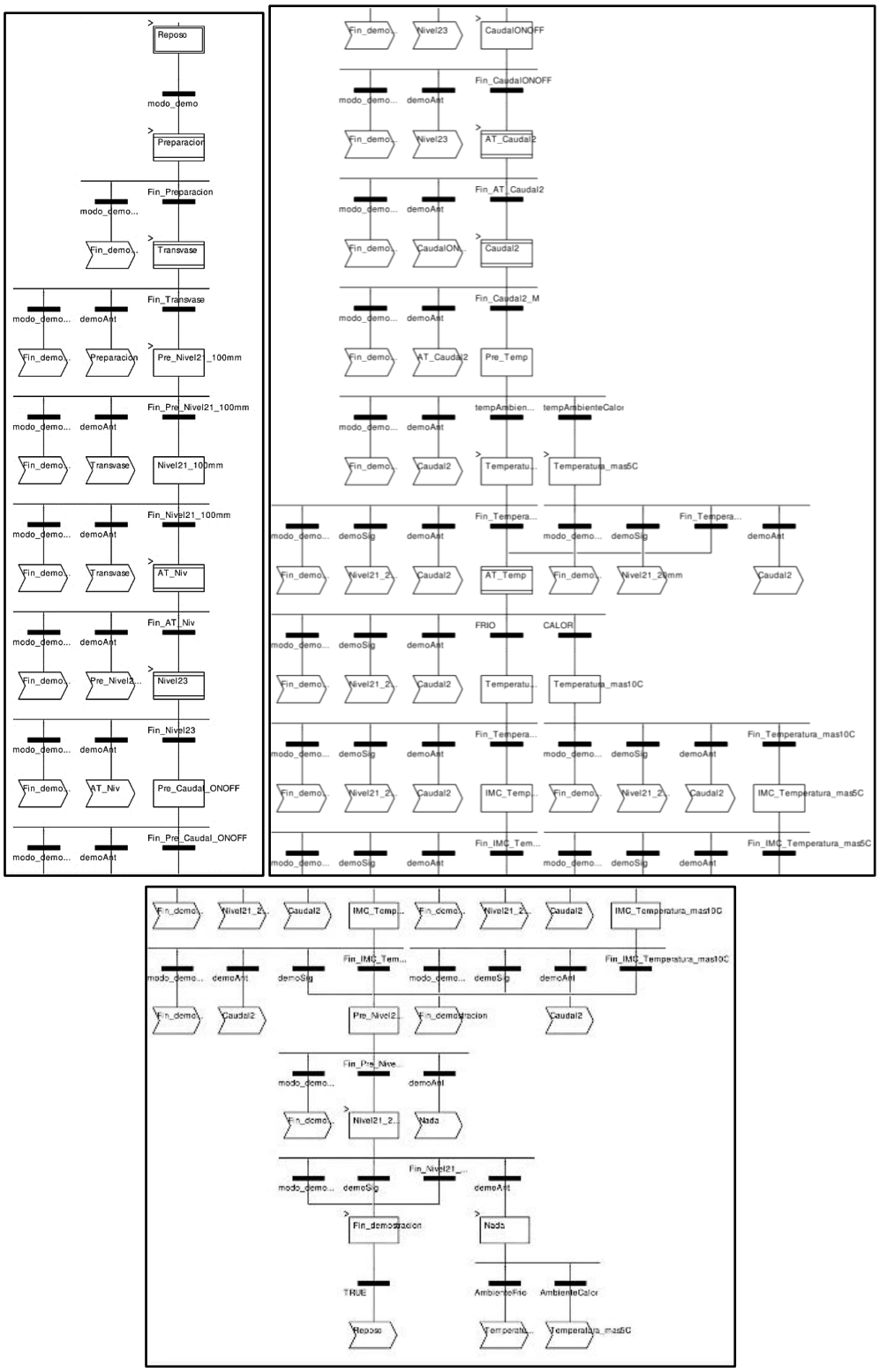


Figura 30. SFC de control de la demo

Capítulo 6. Conclusiones finales

En este capítulo se comentan las conclusiones finales del proyecto, así como las posibles líneas futuras de trabajo sobre la maqueta.

6.1 Conclusiones

La conclusión principal que se debe aportar con este proyecto es la amplia capacidad de la maqueta para realizar diversos controles con los elementos de lo que dispone, más concreto como ha sido en nuestro caso caudal, nivel, temperatura y presión.

También recalcar la mejora con respecto al proyecto antiguo, siendo el cambio más importante la posibilidad de tener un solo control de temperatura que permita calentar y enfriar; también se han separado los PID, y en vez de tener uno solo en el que se cambiaban las variables a controlar, se ha colocado un PID individual para los controles nombrados, permitiendo de esta manera la posibilidad de tener varios controles simultáneos de manera más sencilla e independientes entre ellos. También la inclusión de los Autotuning para facilitar al usuario un correcto funcionamiento de estos calculando automáticamente los parámetros de regulación óptimos.

Por otro lado, otro aspecto importante a mencionar es el buen control que se ha conseguido con los reguladores IMC que, a su vez, aunque son más complicados de configurar que los PID, permiten un uso más sencillo por parte del usuario, que no tiene que preocuparse de poner correctamente los parámetros de regulación, ya que los IMC siempre trataran de conseguir la mejor dinámica para el sistema controlado.

Asimismo, recalcar la gran utilidad del programa de demostración automático, que permite recorrer los elementos más importantes de la maqueta para ver su funcionamiento. La demo, aunque un poco larga en el tiempo, emplea la mayor parte de los reguladores que han sido programados, por lo que ya no solo sirve para enseñar al usuario el funcionamiento de la maqueta, sino comprobar que todos los elementos funcionan correctamente.

Por la parte del SCADA se han mejorado y ampliado las pantallas disponibles, al tiempo que se ha creado una para cada tipo de controles, lo que permite ver los valores del regulador y variar los parámetros de este, así como elegir si queremos usar PID, IMC o activar el Autotuning del primero para calcular sus parámetros.

También tener la posibilidad de activar el modo demostración, mostrando por una pantalla los diversos pasos que recorre la demo mostrando los elementos usados y proporcionando una explicación de la actividad realizada en el momento; de la misma manera permite ver visualmente los valores del regulador, así como sus parámetros.

Por ultimo mencionar la utilidad que supone poder ejecutar controles desde Matlab, ya sea en ventanas emergentes como con archivos *Simulink* interactuando con el autómata para cambiar el tipo de control en el primer caso o sustituir los reguladores del PLC en el segundo caso, pasando a controlarlo desde Matlab como el caso práctico que se ha hecho sobre caudal, así como obtener gráficas de las variables del autómata.

Como conclusión final resaltar los conocimientos adquiridos durante la realización del proyecto, ya sea aprendiendo de manera más profunda las herramientas empleadas como su interrelación para conseguir los objetivos deseados, así como los métodos de control que permiten implementar los autómatas programables para el entorno industrial, con gran utilidad de cara al futuro profesional propio de la Ingeniería.

6.2 Líneas futuras

En el punto de líneas futura, podemos mencionar en primer lugar la posibilidad de ampliación de los archivos *Simulink*, ya que partiendo de la base que hemos implementado, se puede hacer una réplica funcional de la programación del PLC, para de esta manera poder tener todos con reguladores y controles en Matlab y ser independiente casi totalmente del autómata, siendo solo necesario la comunicación con este para la lectura de sensores y el uso de actuadores.

Por otra parte, como modificación de la propia maqueta se propone actualizar los actuadores de control de temperatura del depósito intermedio. Las razones de esto son que, al tener el foco de calor en la parte inferior del mismo, el depósito tarde un tiempo elevado en calentarse cuando está lleno de agua, siendo además la medida de temperatura poco precisa al variar entre la parte inferior más caliente y la superior. Por esto se propone la instalación de un sistema que permita el calentamiento en una mayor parte de la superficie del depósito para facilitar el proceso. Otro punto sería la mejora del sistema de refrigeración, ya que el actual no permite conseguir temperaturas por debajo del ambiente, y como en el caso del calentamiento, es un proceso enormemente lento manejando un gran volumen de agua.

Con respecto al control de presión en el tanque izquierdo, se plantea el desarrollo de un sistema que permita establecer un valor de presión en valor real, frente al método relativo actualmente instalado, siempre teniendo en cuenta que no se puede superar la presión de aire del suministro externo. Para el depósito derecho, se podría cambiar la electroválvula neumática instalada (que solo permite apertura o cierre total) por otra que admita un control más preciso.

Por último, se plantea la posibilidad de implementar el control del PLC mediante otros sistemas alternativos a SCADA o Matlab, como pueden ser las pantallas de explotación externas que permiten la ejecución de menús y pantallas de manera similar a lo ya realizado.

Bibliografía

[1] Proyecto de Fin de Carrera "CONTROL DE NIVEL, CAUDAL, PRESIÓN Y TEMPERATURA CON UN SCADA EN UN SISTEMA MEZCLADOR DE LÍQUIDOS".
Ingeniería industrial curso 2008-2009. Aut. José Miguel Gil Narvión

[2] Definición PLC
http://recursostic.educacion.es/observatorio/web/gl/component/content/article/502-monografico-lenguajes-de-programacion?start=2

[3] Definición presostato electrónico
http://www.bloginstrumentacion.com/blog/2011/05/02/que-es-un-presotato-electronico/

[4] Definición caudalímetro
https://es.wikipedia.org/wiki/Caudal%C3%ADmetro

[5] Definición medida de nivel por presión diferencial
http://instrumentacionycontrol.net/cursos-libres/instrumentacion/item/81-nivel-medidas-por-presi%C3%B3n-diferencial.html

[6] Definición sensor PT100
http://srcsl.com/que-es-un-sensor-pt100/

[7] Definición sensor capacitivo
http://www.sensorstecnics.net/es/productos/category/96/sensores-y-transmisores/sensores-capacitivos
https://es.wikipedia.org/wiki/Sensor_capacitivo

[8] Definición regulador de presión proporcional
http://www.asconumatics.eu/images/site/upload/_es/pdf1/Tecnologiaproporcional.pdf
http://www.oilproduction.net/files/Reguladoresdepresion.pdf

[9] Definición motobomba
https://es.wikipedia.org/wiki/Bomba_hidr%C3%A1ulica

[10] Definición refrigeración termoeléctica
https://es.wikipedia.org/wiki/Refrigeraci%C3%B3n_termoel%C3%A9ctrica

[11] Definición controlador PID
https://es.wikipedia.org/wiki/Controlador_PID
Apuntes de la asignatura "Sistemas de control automático", Aut.: **Ramón Piedrafita**.

[12] Librería de bloques para regulación en *Unity Pro XL* de Schneider Electric
https://1drv.ms/b/s!AiG6opqiuee6holOUWI2SSOju-S7xw

[13] Definición controlador ON/OFF
https://es.wikipedia.org/wiki/Control_S%C3%AD/No

[14] Definición protocolo Modbus
https://es.wikipedia.org/wiki/Modbus

[15] Métodos de conexión entre un PLC y MATLAB
http://es.mathworks.com/matlabcentral/answers/92196-how-can-i-interface-to-plcs-like-the-modicon-or-allen-bradley-plc-with-matlab-or-simulink?requestedDomain=www.mathworks.com

[16] Ejemplo de aplicación de control de un PLC desde MATLAB
http://www.rifqion.com/menulis/plc-plant-simulation-with-matlab-simulink/

[17] Curso de formación sobre Vijeo Citect de Schneider Electric
https://1drv.ms/b/s!AiG6opqiuee6holcRcugvQulkDNrWQ
https://1drv.ms/b/s!AiG6opqiuee6holak3BgrMPCAvra5w

[18] Manual básico sobre Matlab
http://www.lawebdelprogramador.com/cursos/Matlab/3573-Manual-basico-de-Matlab.html

ANEXOS

Anexos

En esta sección desarrollaremos los Anexos del proyecto que hemos ido nombrando en la memoria.

Anexo A. Resultados Experimentales

En este Anexo se muestras diferentes imágenes de la ejecución del programa del PLC tanto desde las pantallas de operador de *Unity Pro*, SCADA y Matlab.

C1. Control de Caudal

En primer lugar, probamos los diferentes reguladores de caudal y sus modalidades.

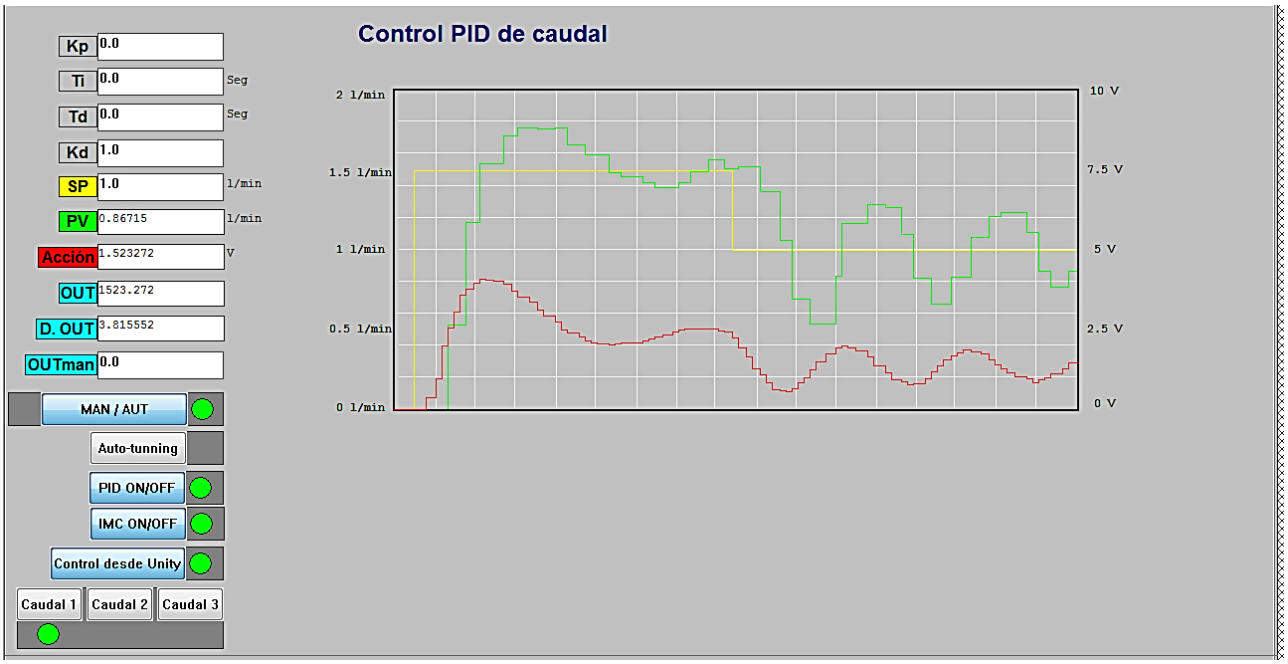


Figura 31. Pantalla de operador en Unity de Control tipo 1 de Caudal con IMC

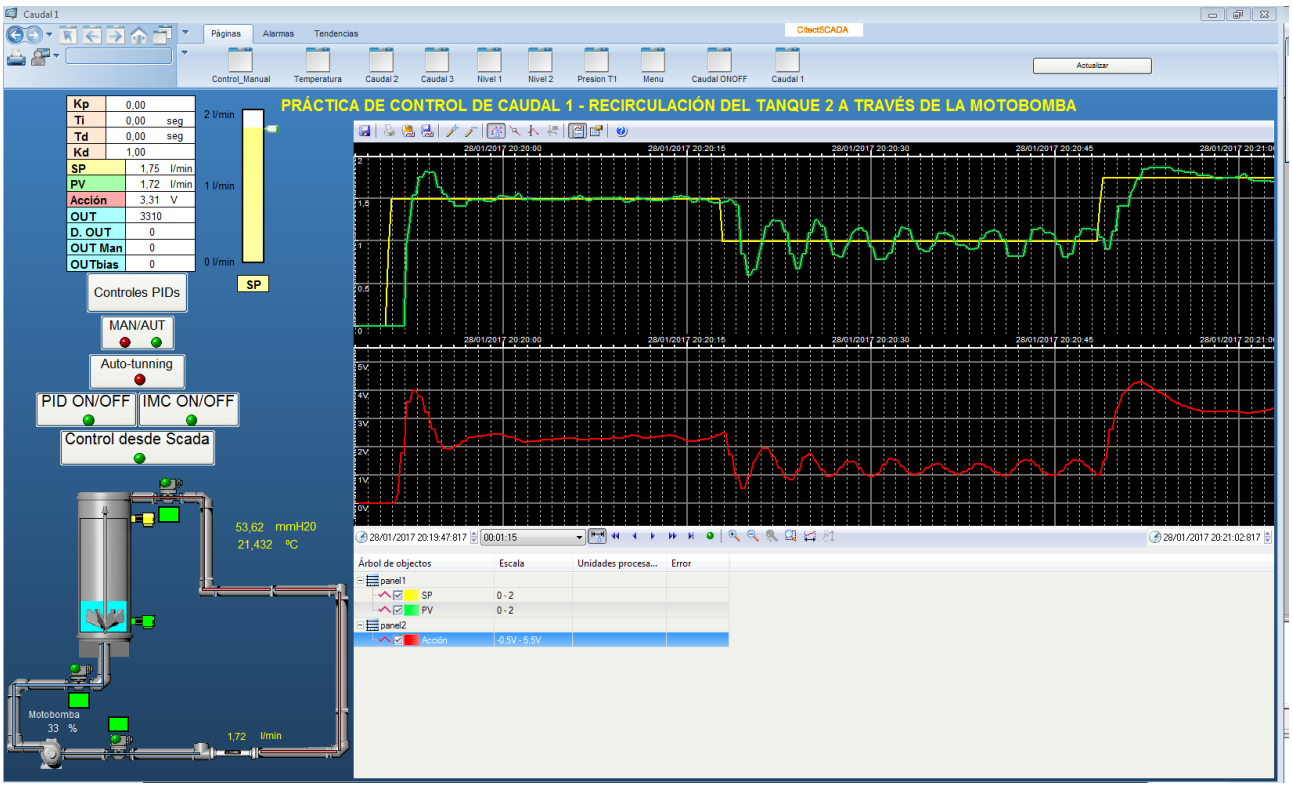


Figura 32. Pantalla SCADA de Control tipo 1 de Caudal con IMC

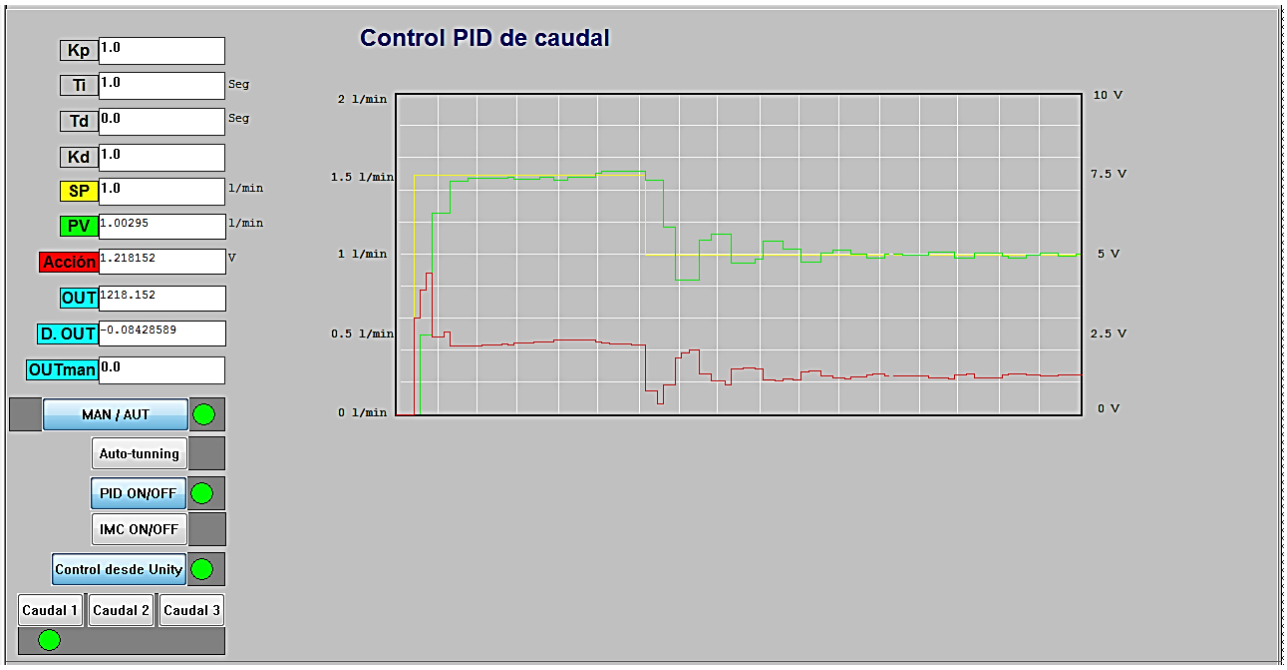


Figura 33. Pantalla de operador en Unity de Control tipo 1 de Caudal con PID

Figura 34. Pantalla SCADA de Control tipo 1 de Caudal con PID

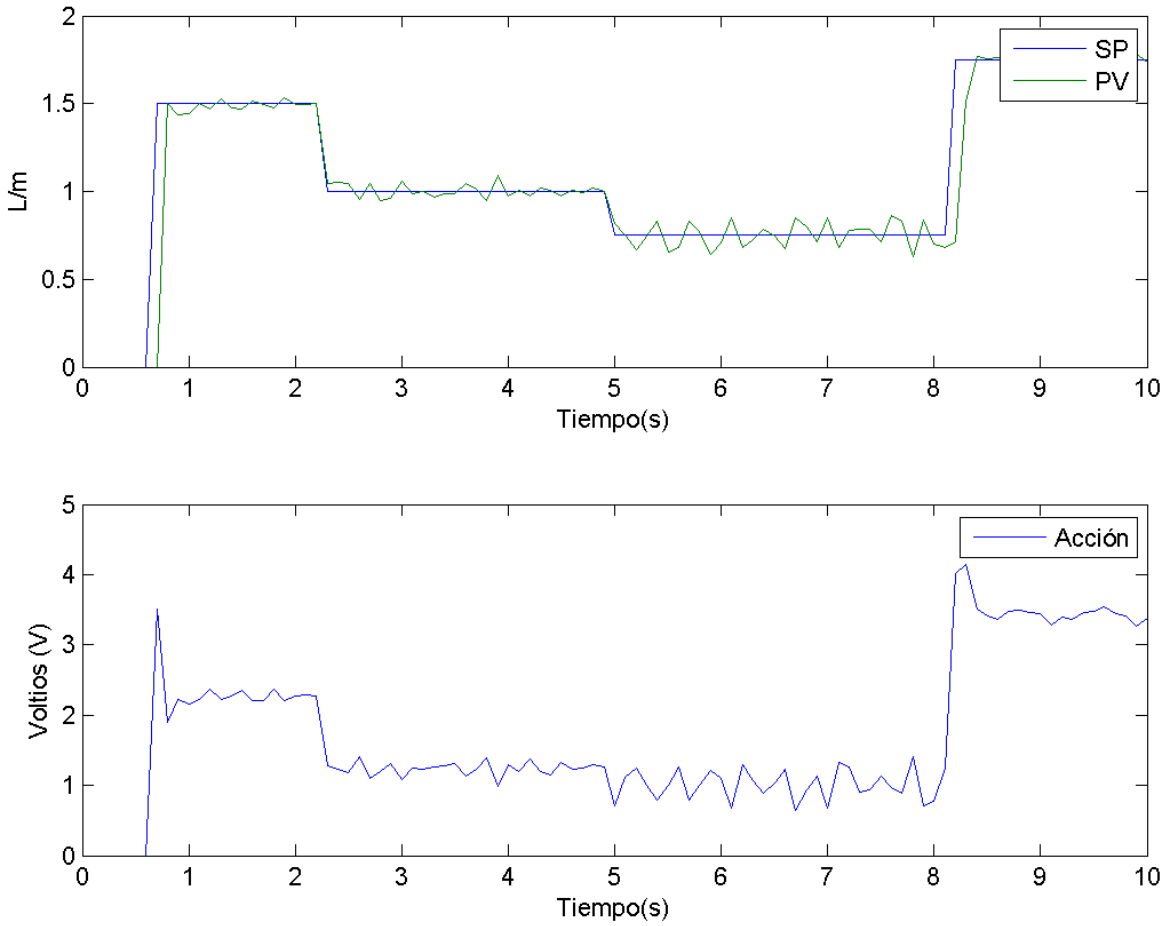


Figura 35. Gráfica obtenida de Matlab con las variables de Control tipo 1 de Caudal

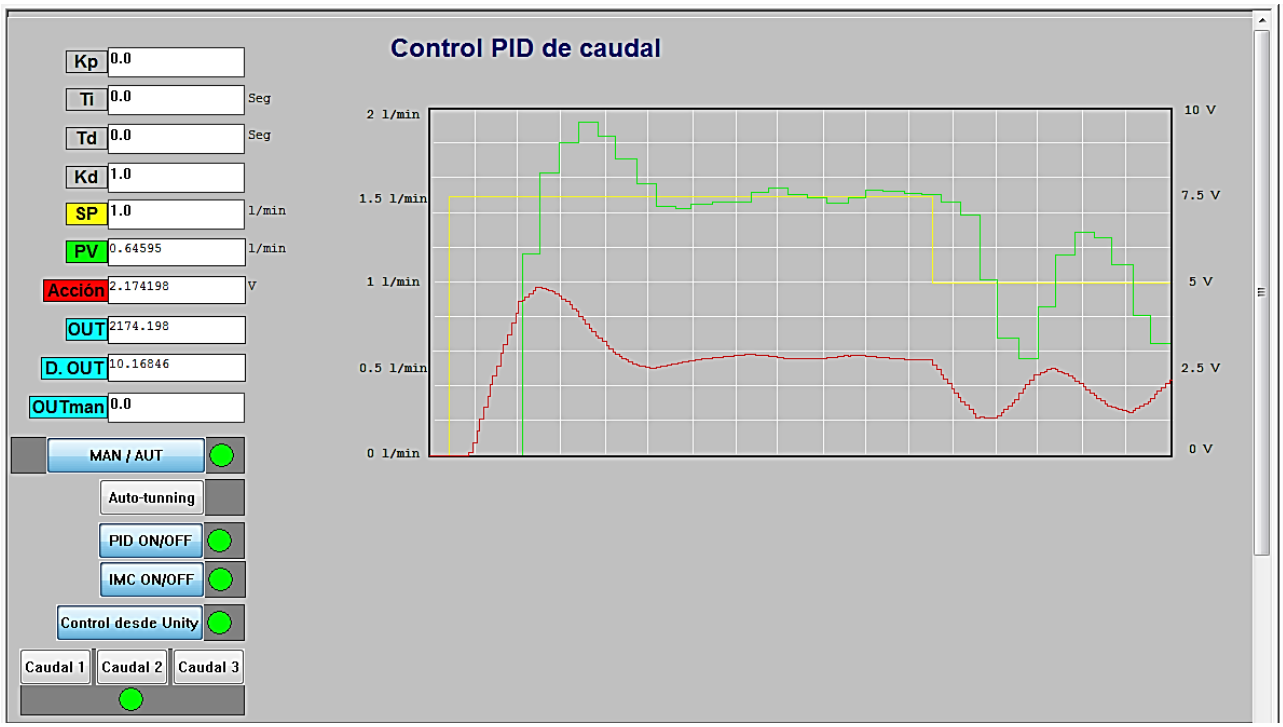


Figura 36. Pantalla de operador en Unity de Control tipo 2 de Caudal con IMC

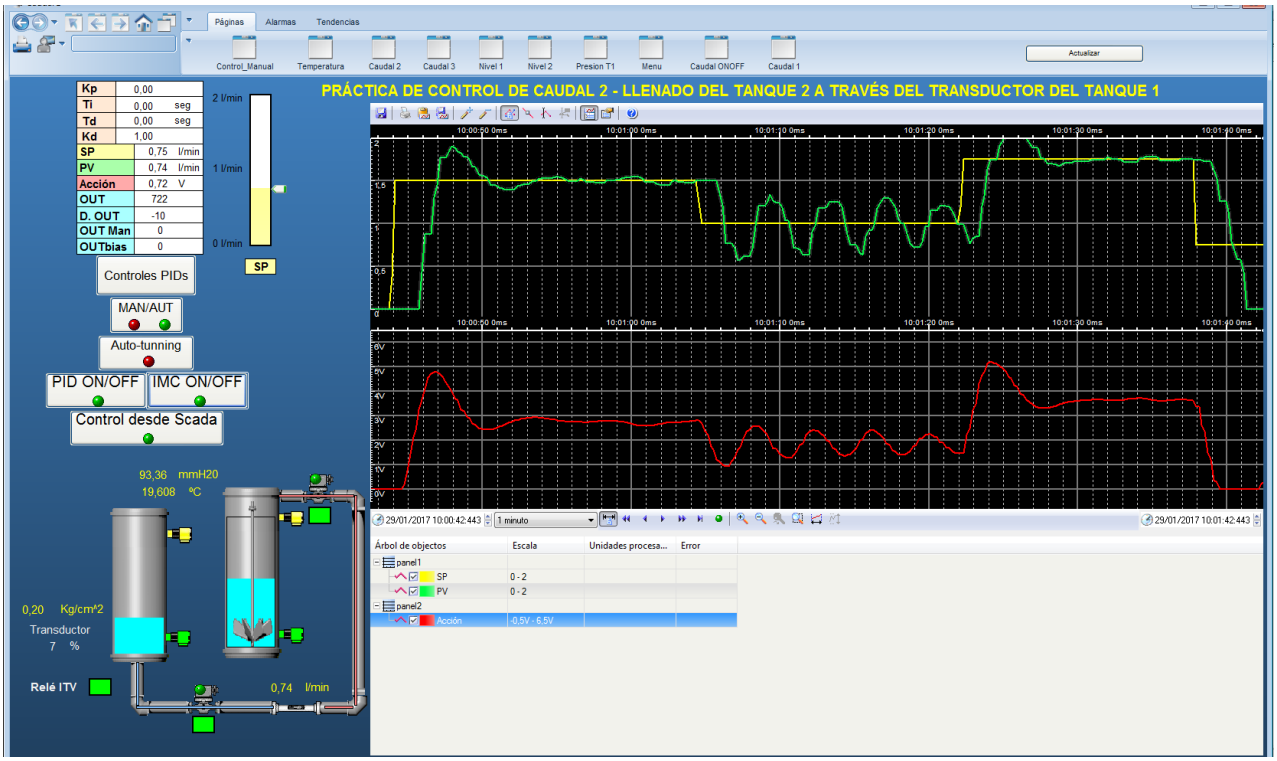


Figura 37. Pantalla SCADA de Control tipo 2 de Caudal con IMC

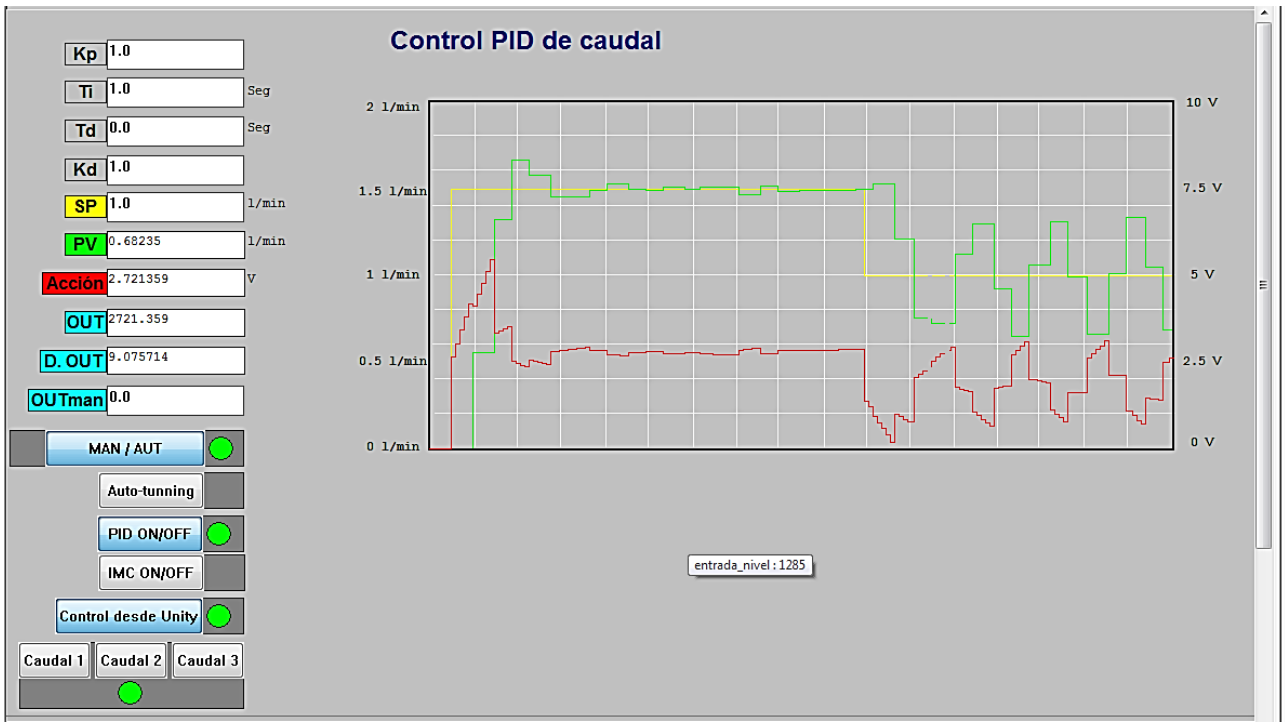


Figura 38. Pantalla de operador en Unity de Control tipo 2 de Caudal con PID

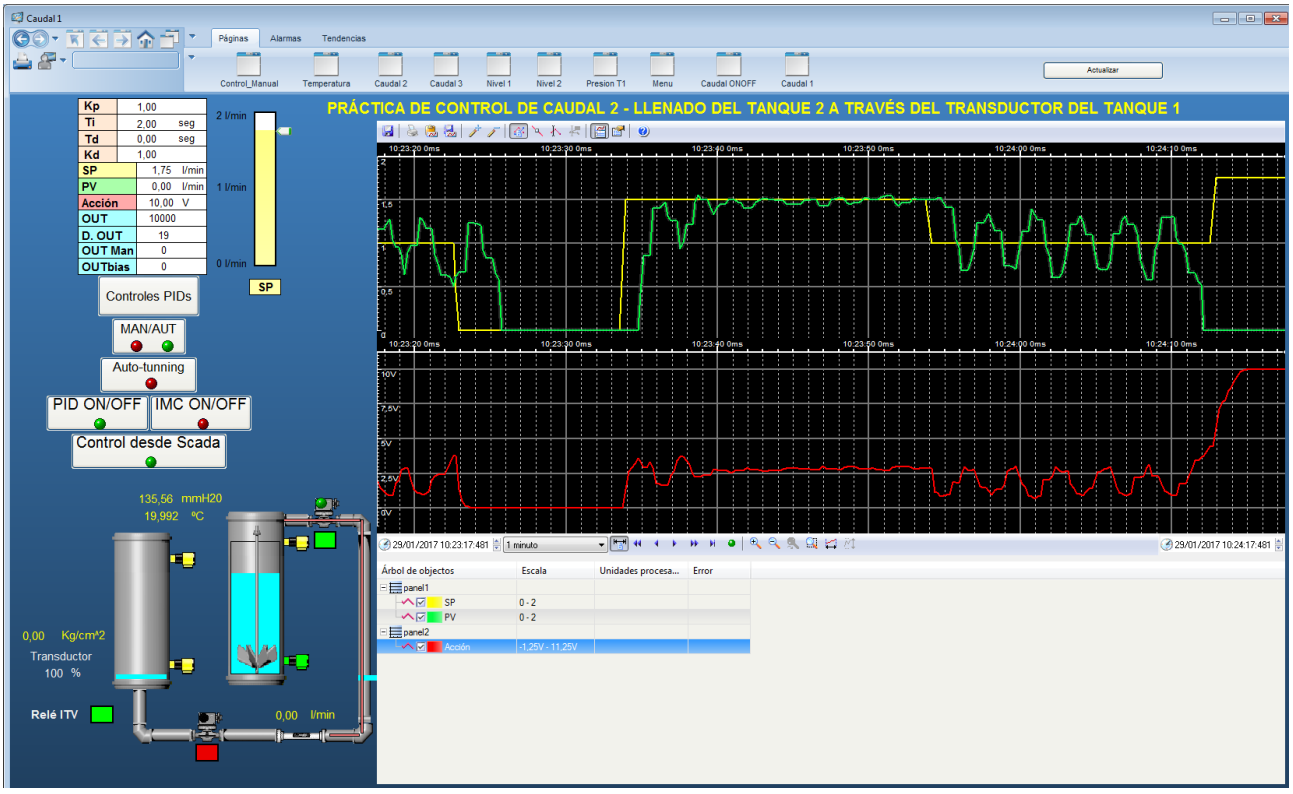


Figura 39. Pantalla SCADA de Control tipo 2 de Caudal con PID

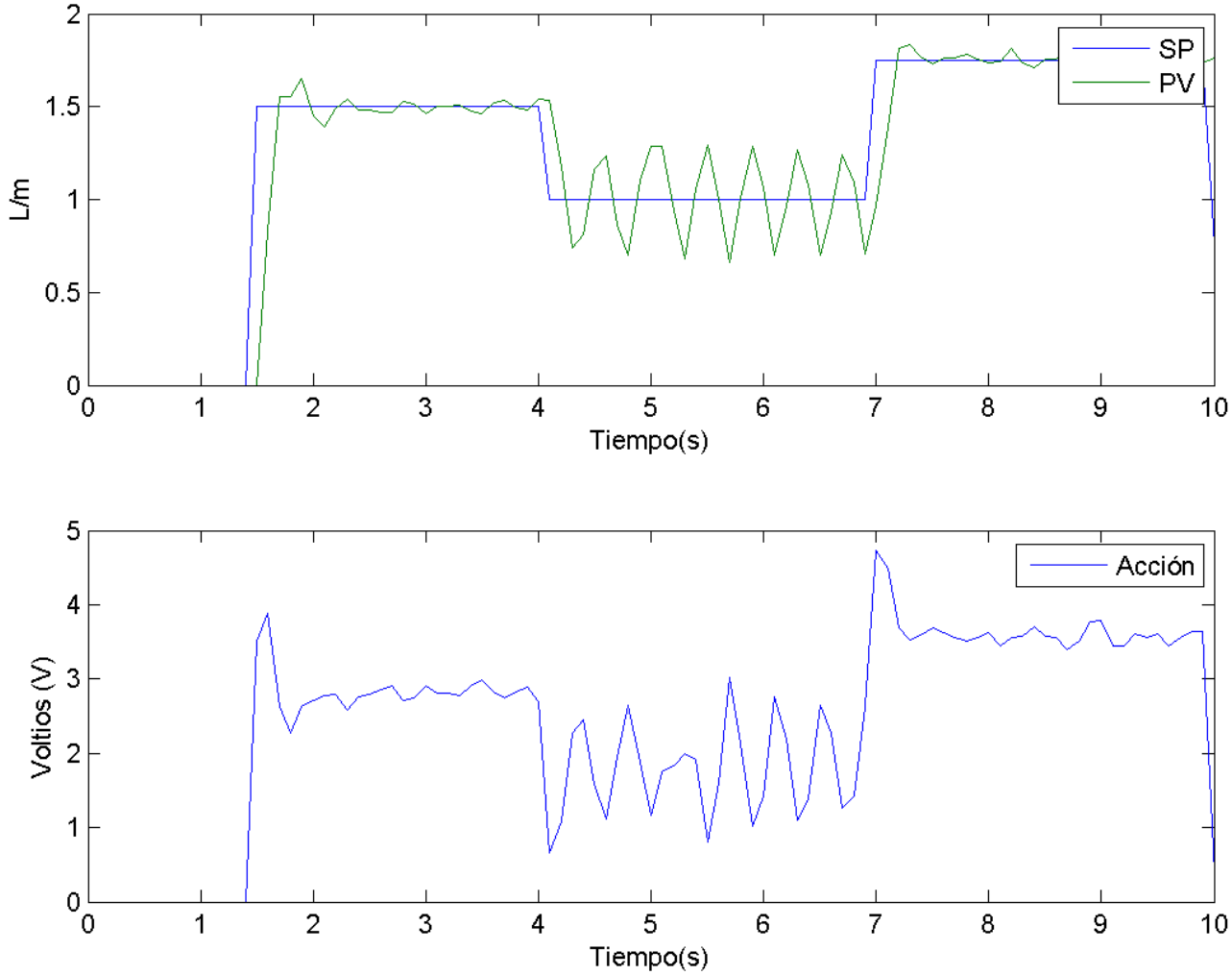


Figura 40. Gráfica obtenida de Matlab con las variables de Control tipo 2 de Caudal

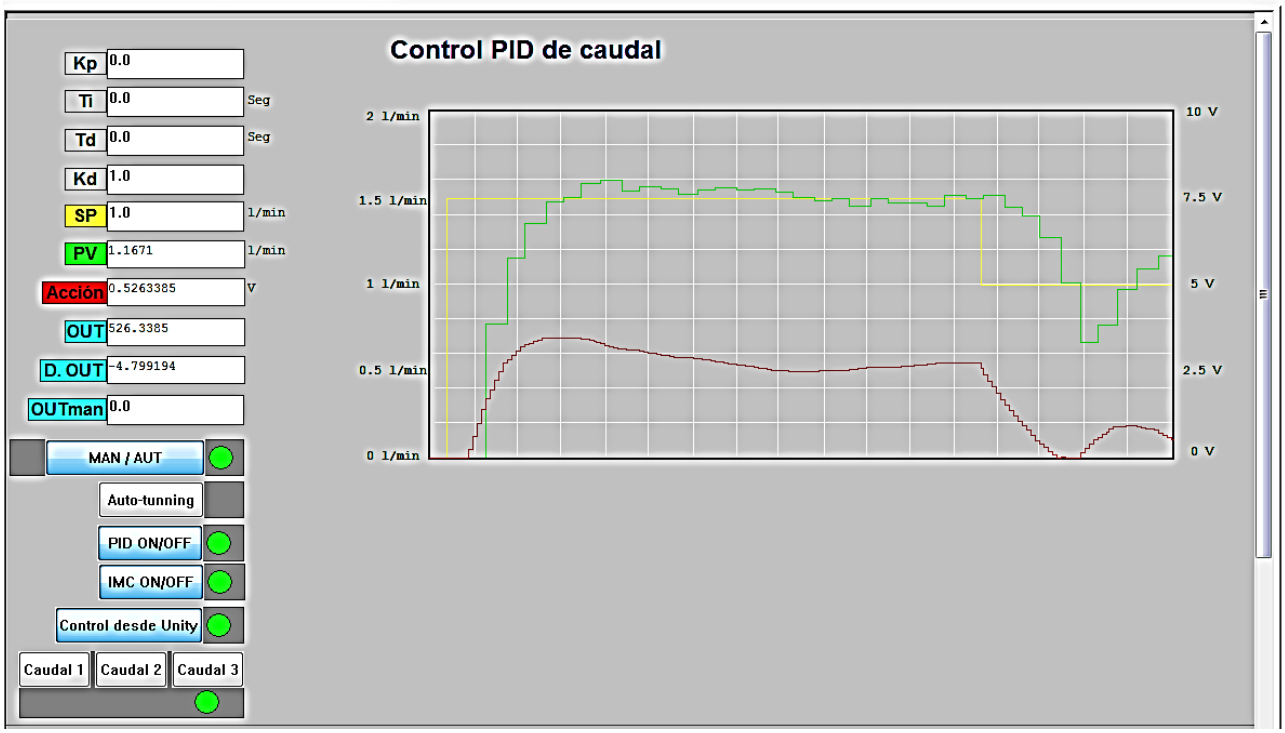


Figura 41. Pantalla de operador en Unity de Control tipo 3 de Caudal con IMC

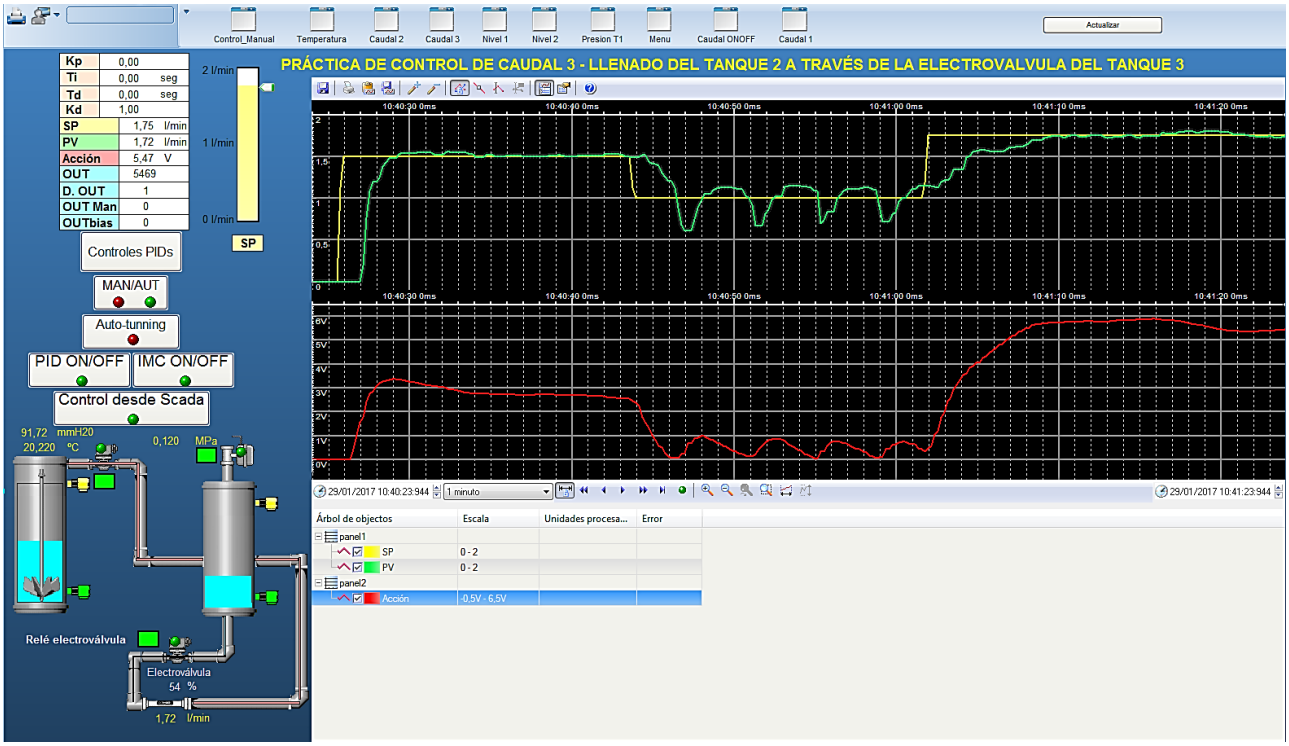


Figura 42. Pantalla SCADA de Control tipo 2 de Caudal con IMC

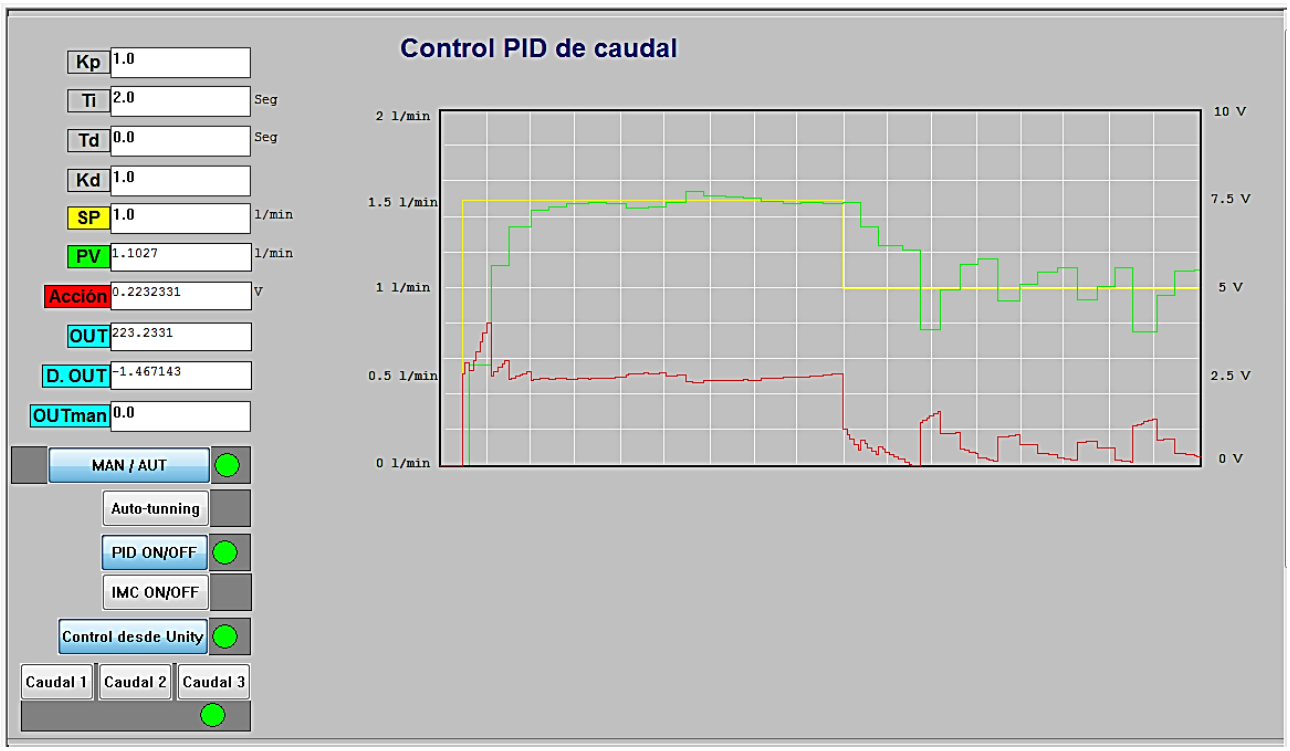


Figura 43. Pantalla de operador en Unity de Control tipo 3 de Caudal con PID

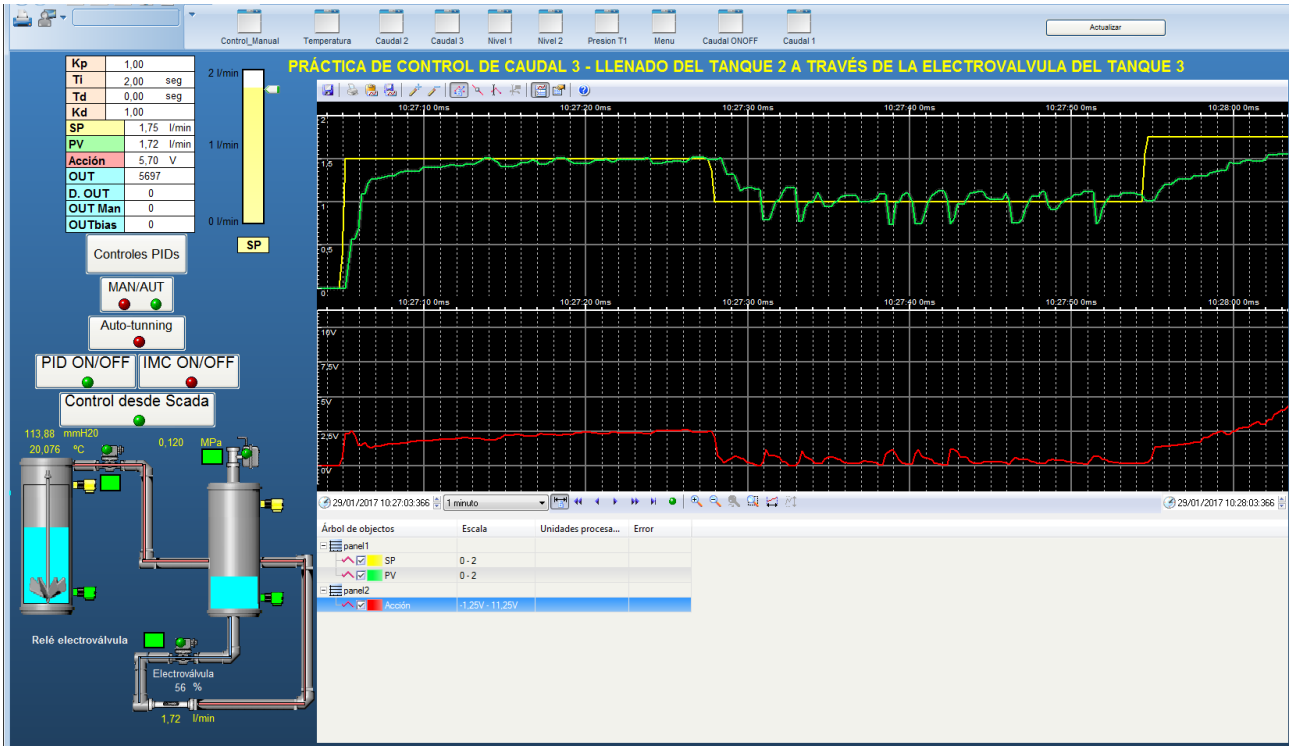


Figura 44. Pantalla SCADA de Control tipo 3 de Caudal con PID

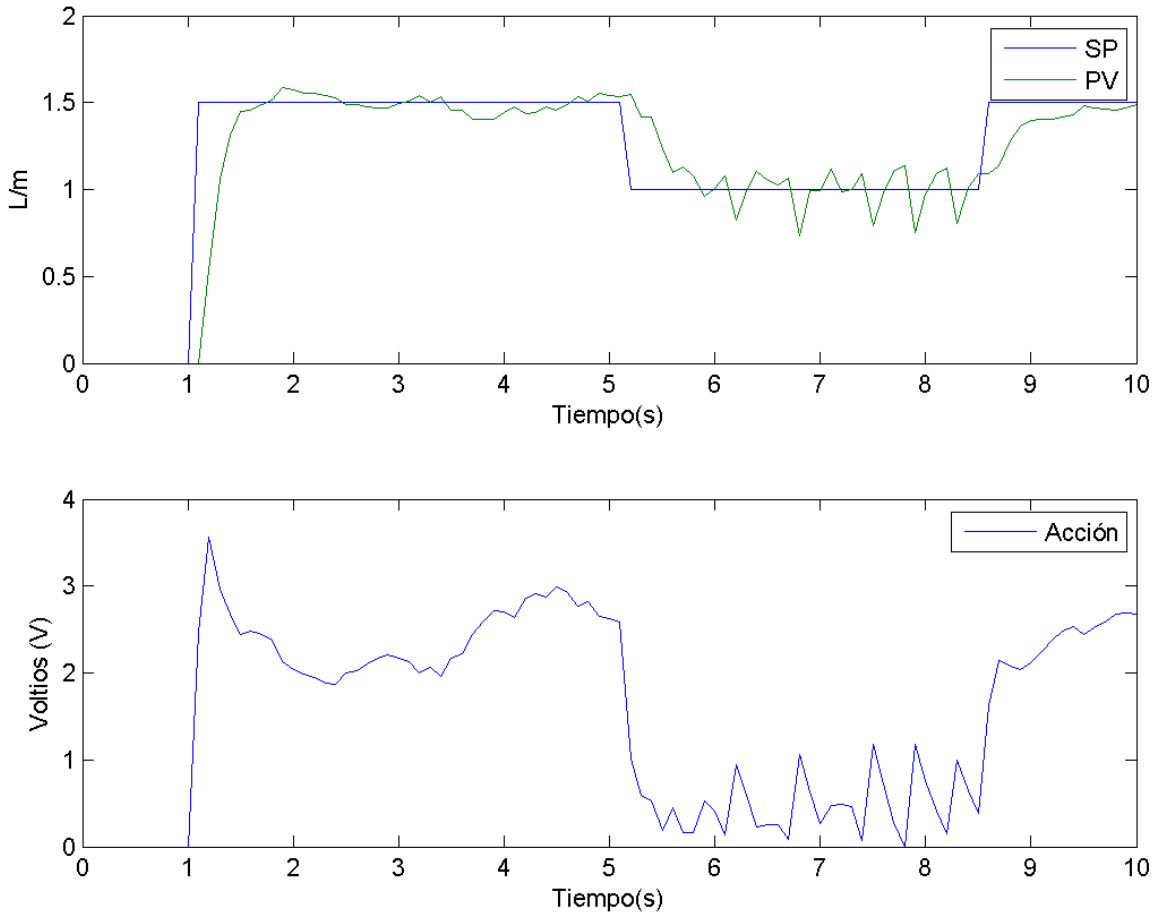


Figura 45. Gráfica obtenida de Matlab con las variables de Control tipo 3 de Caudal

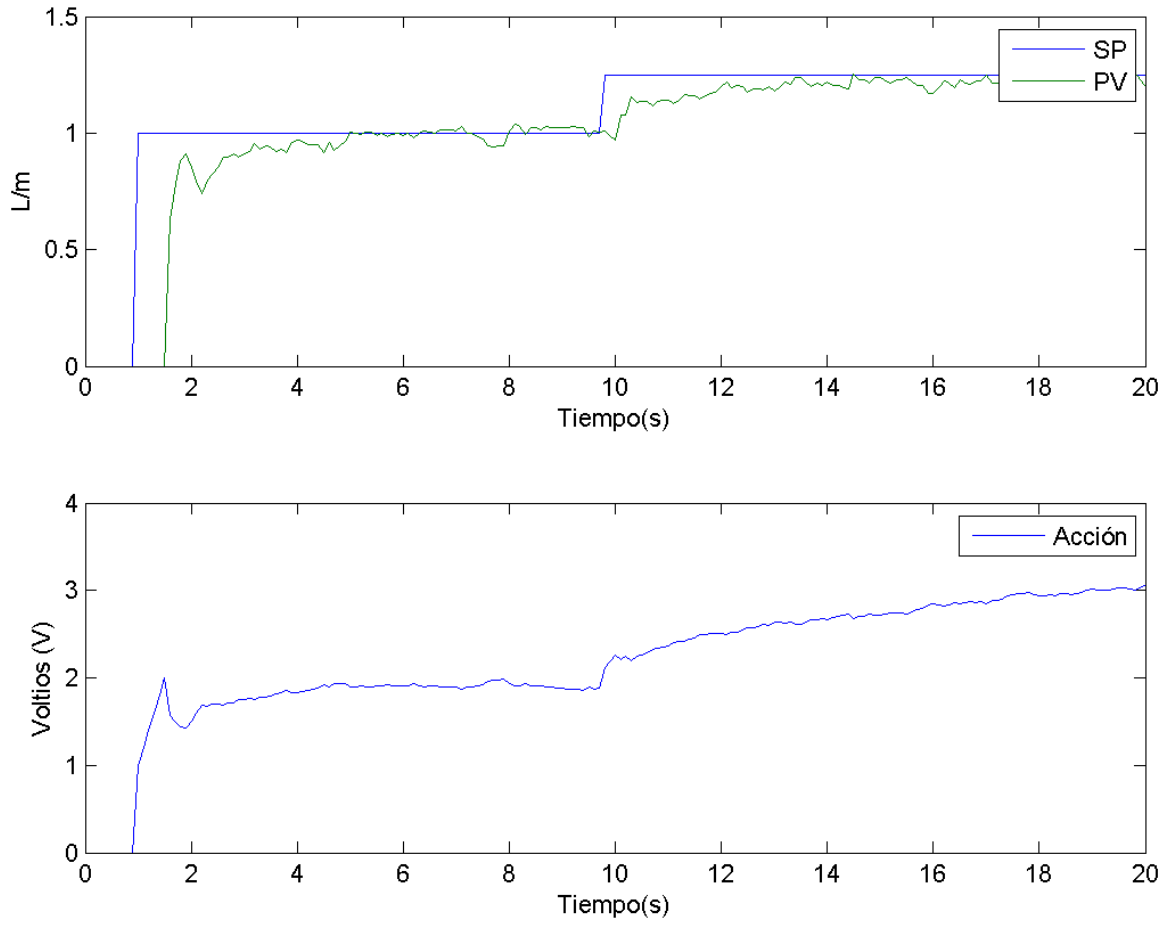


Figura 46. Gráfica obtenida del control de Caudal a través del PID en *Simulink*

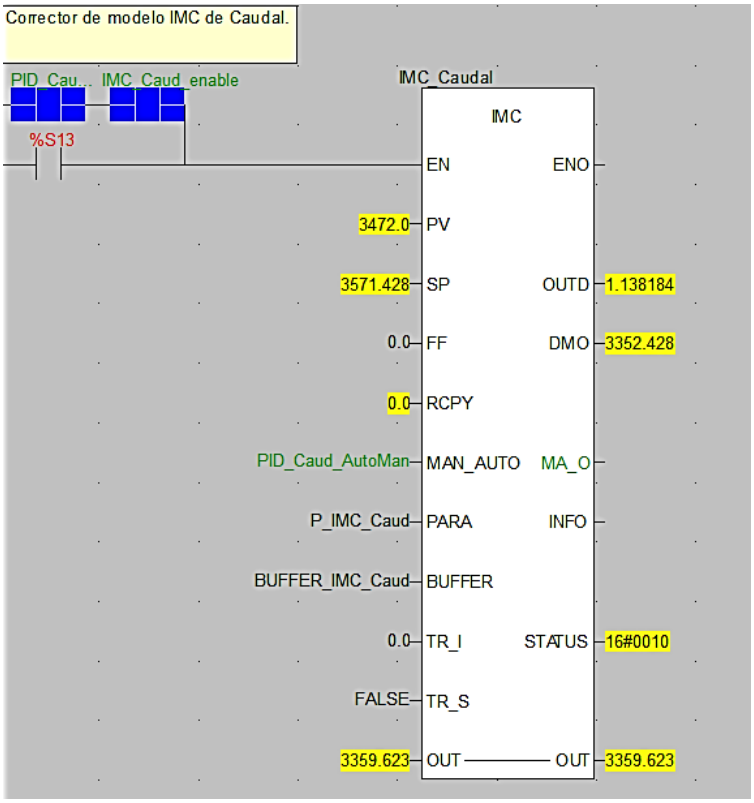


Figura 47. Bloque IMC de Control de Caudal

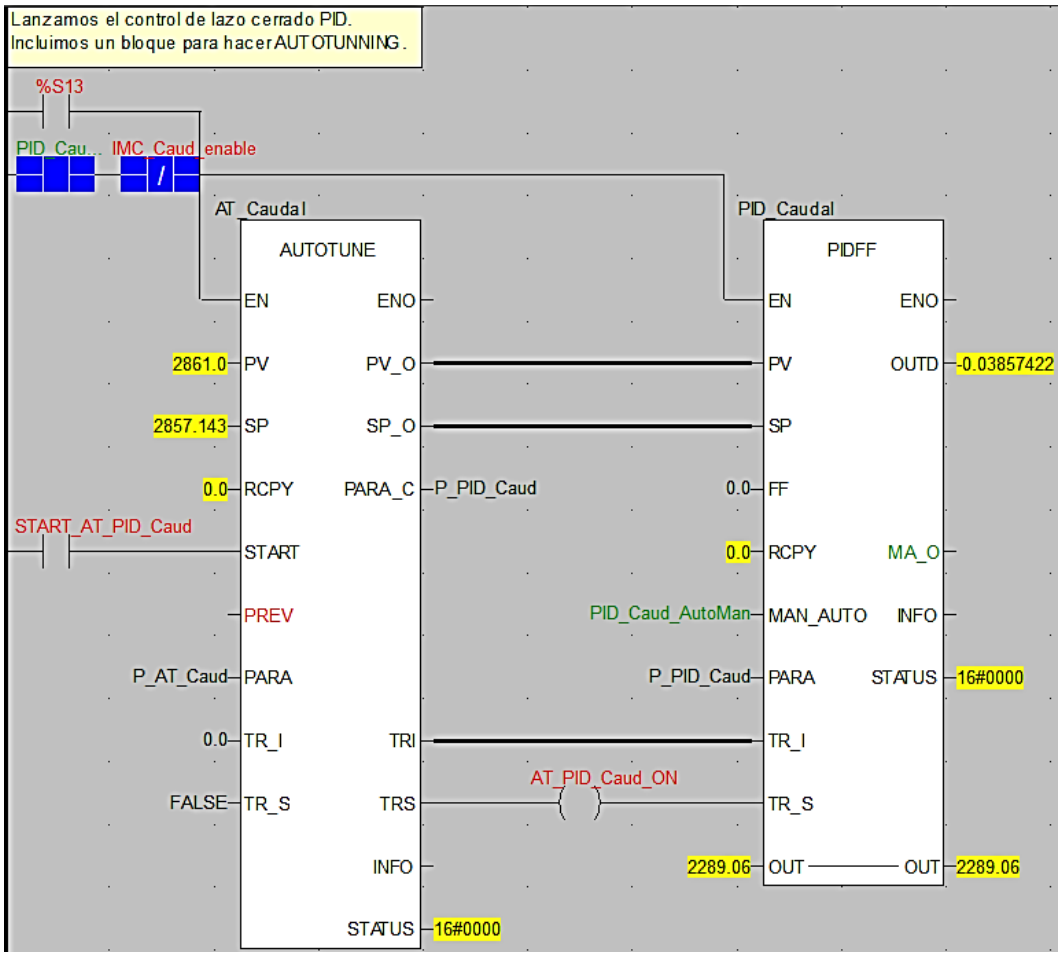


Figura 48. Bloque PID con Autotuning de Control de Caudal

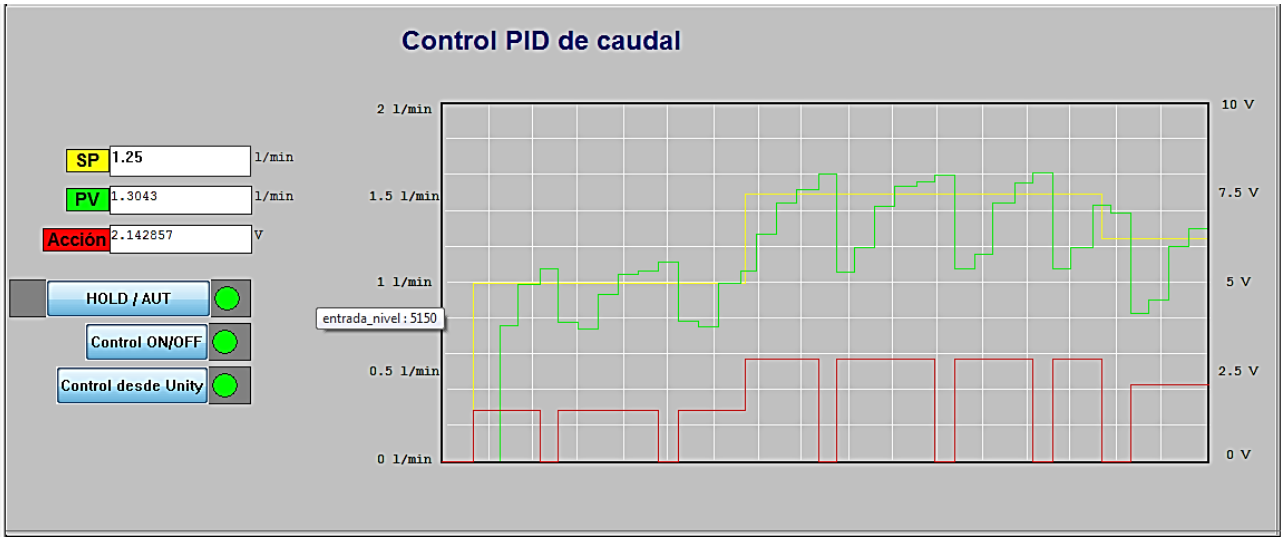


Figura 49. Pantalla de operador en Unity de Control ON/OFF de Caudal

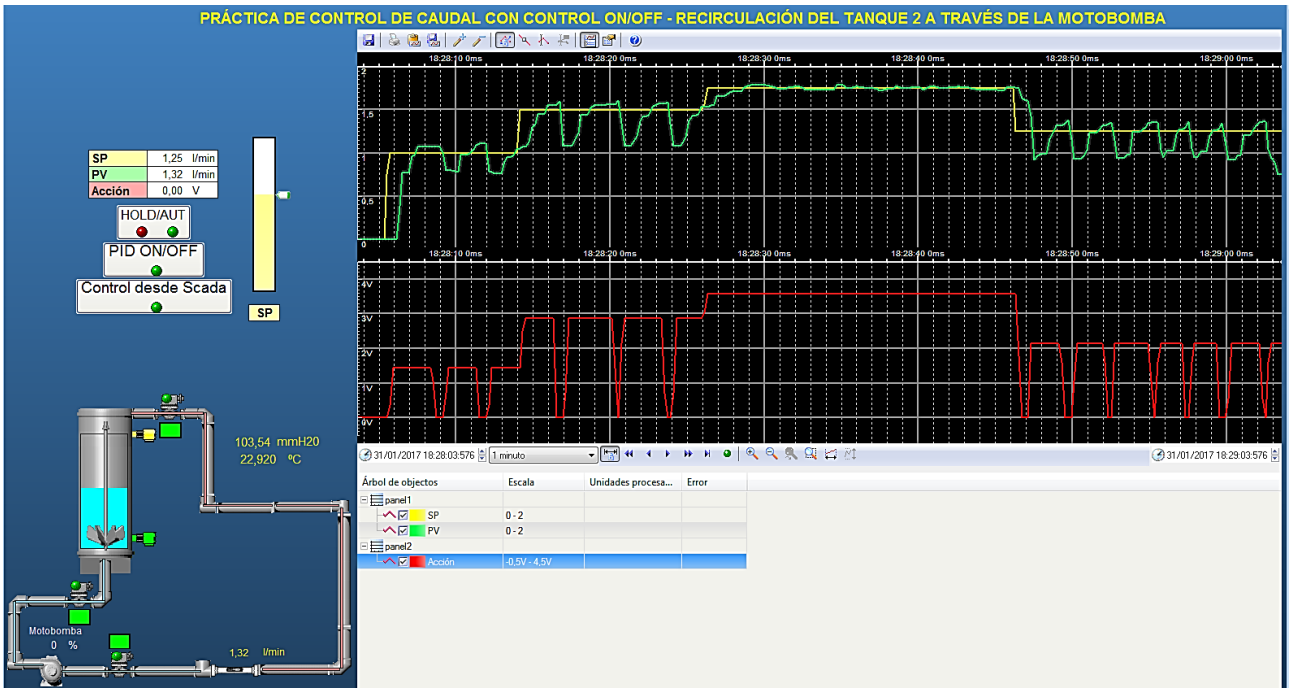


Figura 50. Pantalla SCADA de Control ON/OFF de Caudal

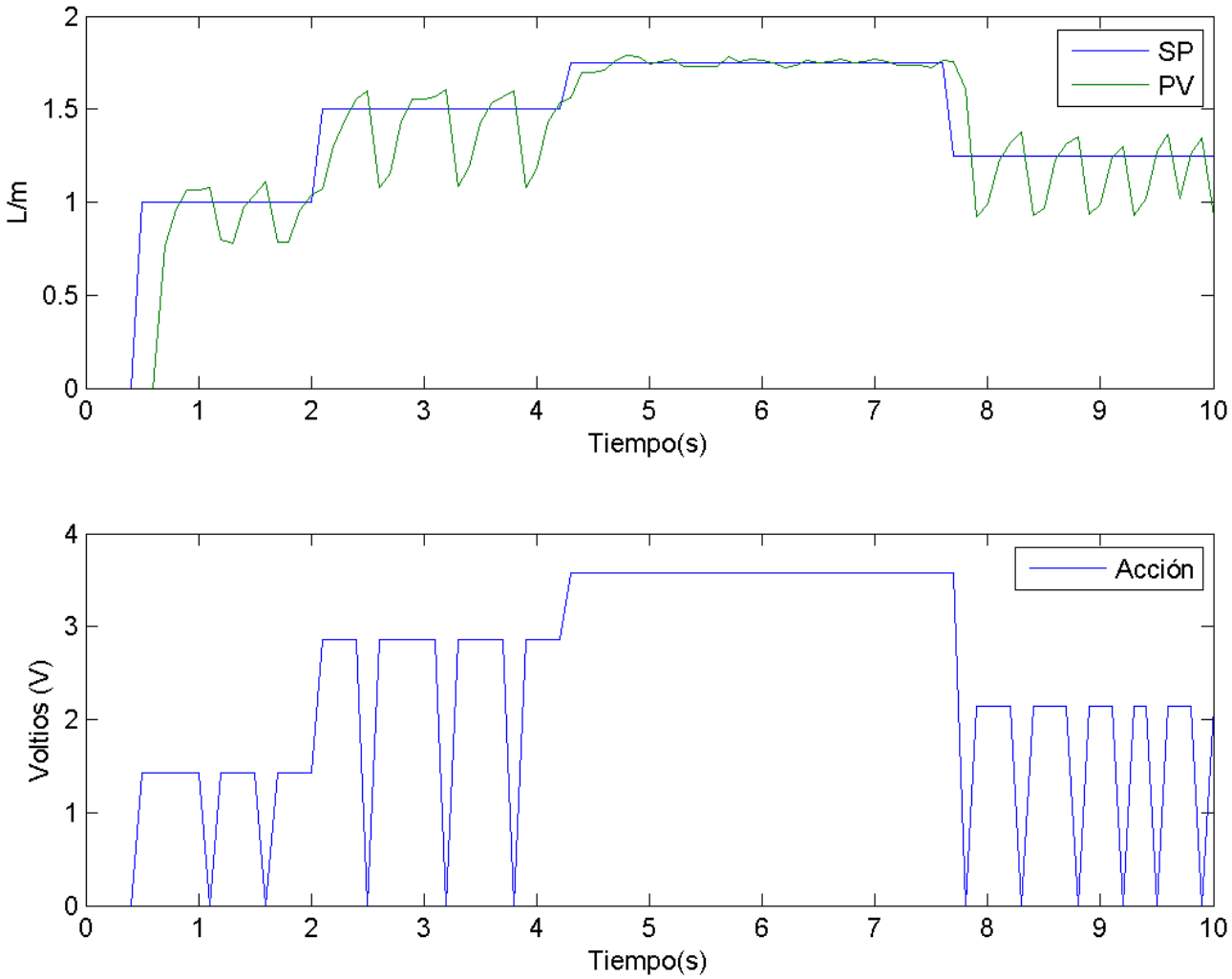


Figura 51. Gráfica obtenida de Matlab con las variables Control ON/OFF de Caudal

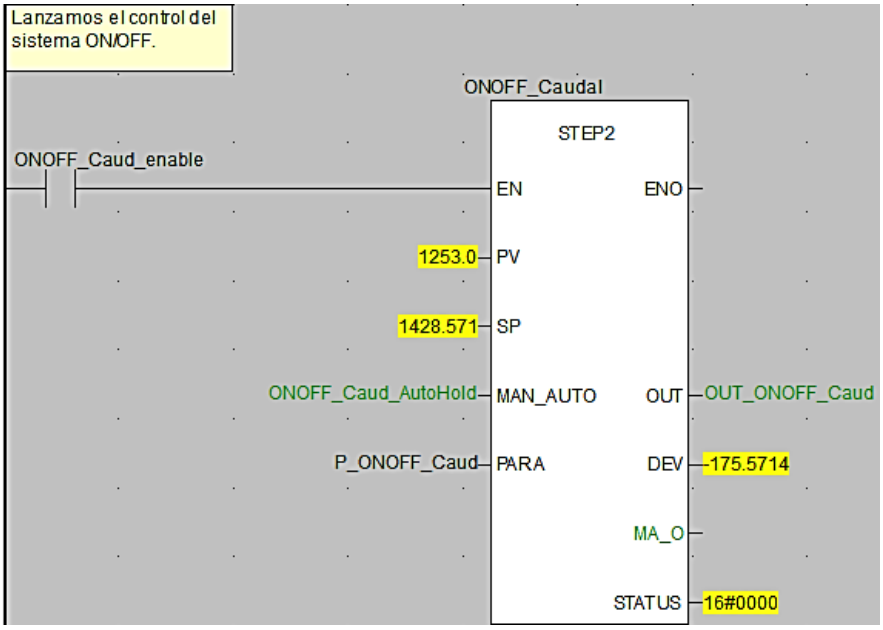


Figura 52. Bloque de Control ON/OFF de Caudal

C2. Control de Nivel

En segundo lugar, probamos los diferentes reguladores de nivel configurados y sus modalidades.

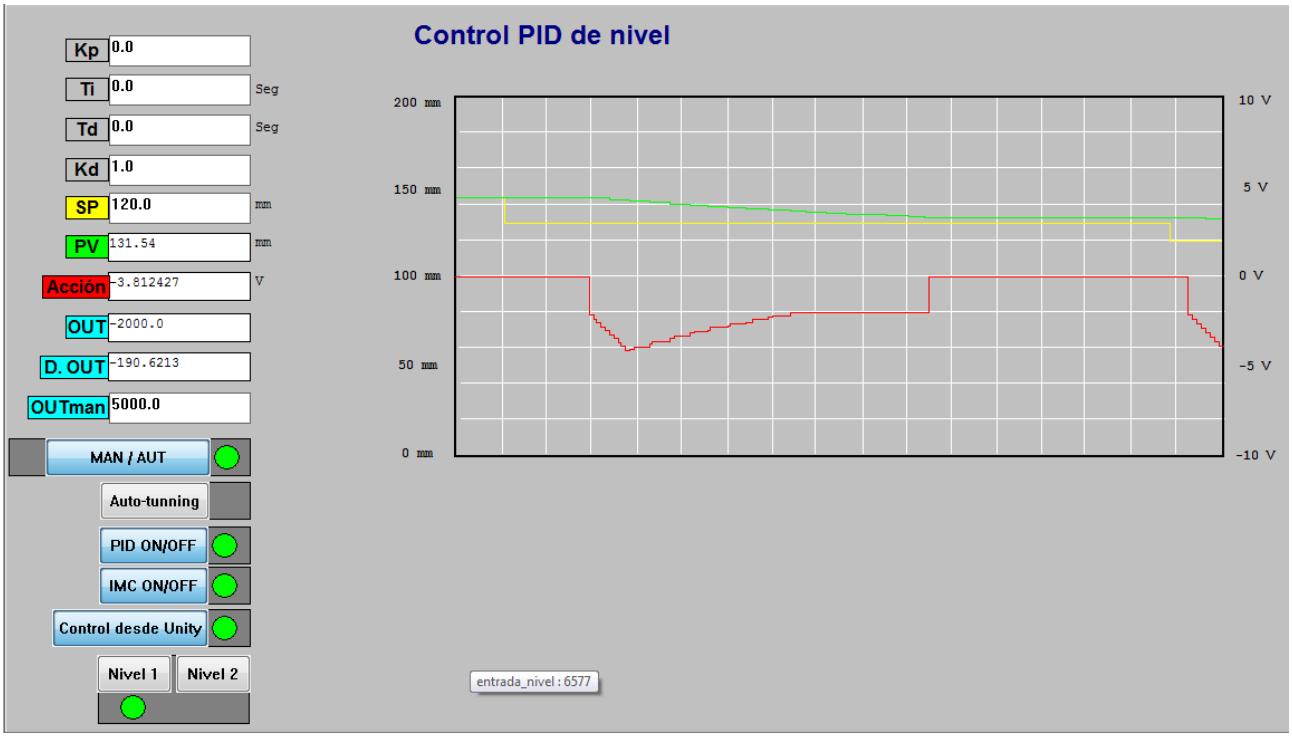


Figura 53. Pantalla de operador en Unity de Control tipo 1 de Nivel con IMC

Figura 54. Pantalla SCADA de Control tipo 1 de Nivel con IMC

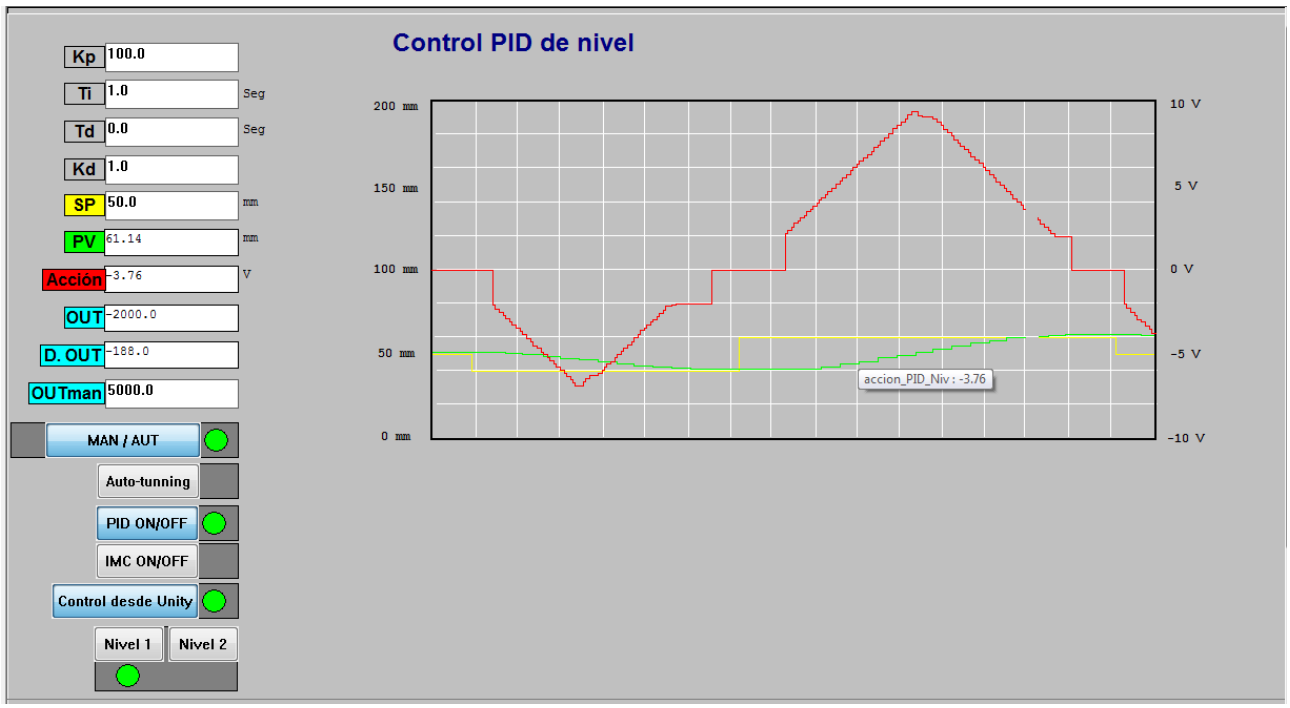


Figura 55. Pantalla de operador en Unity de Nivel tipo 1 de Caudal con PID

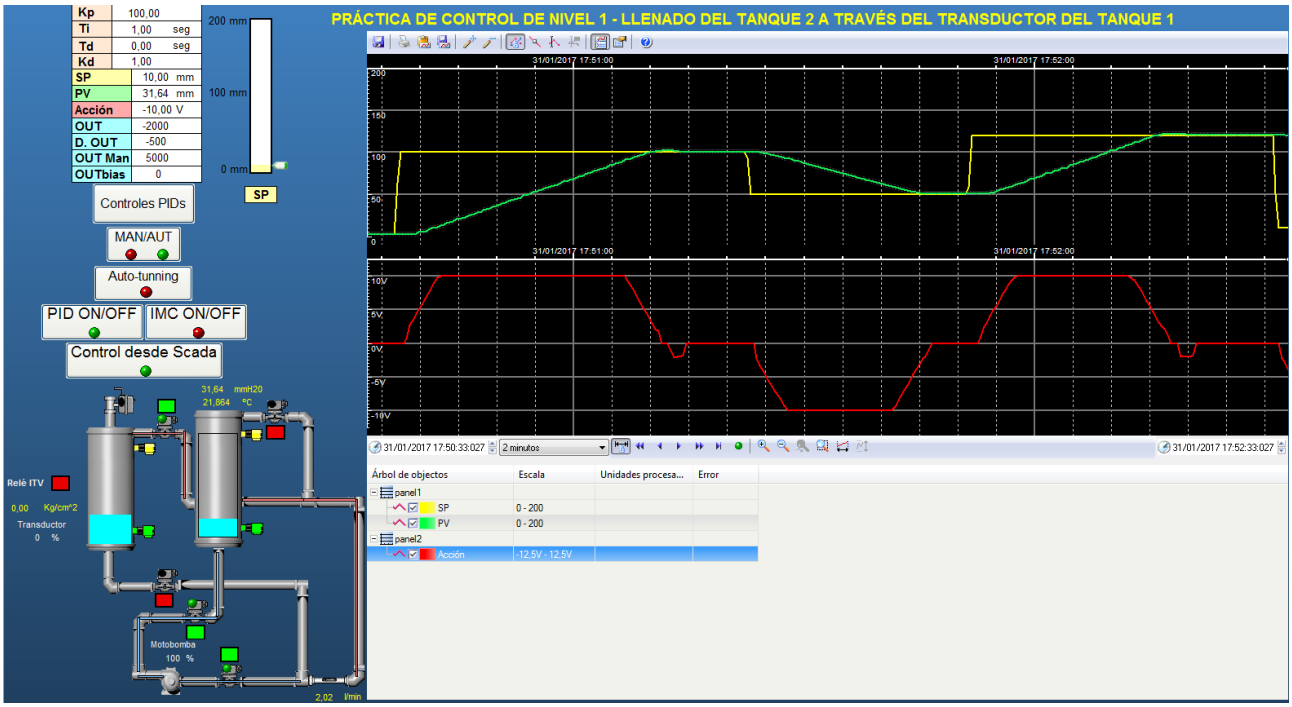


Figura 56. Pantalla SCADA de Control tipo 1 de Nivel con PID

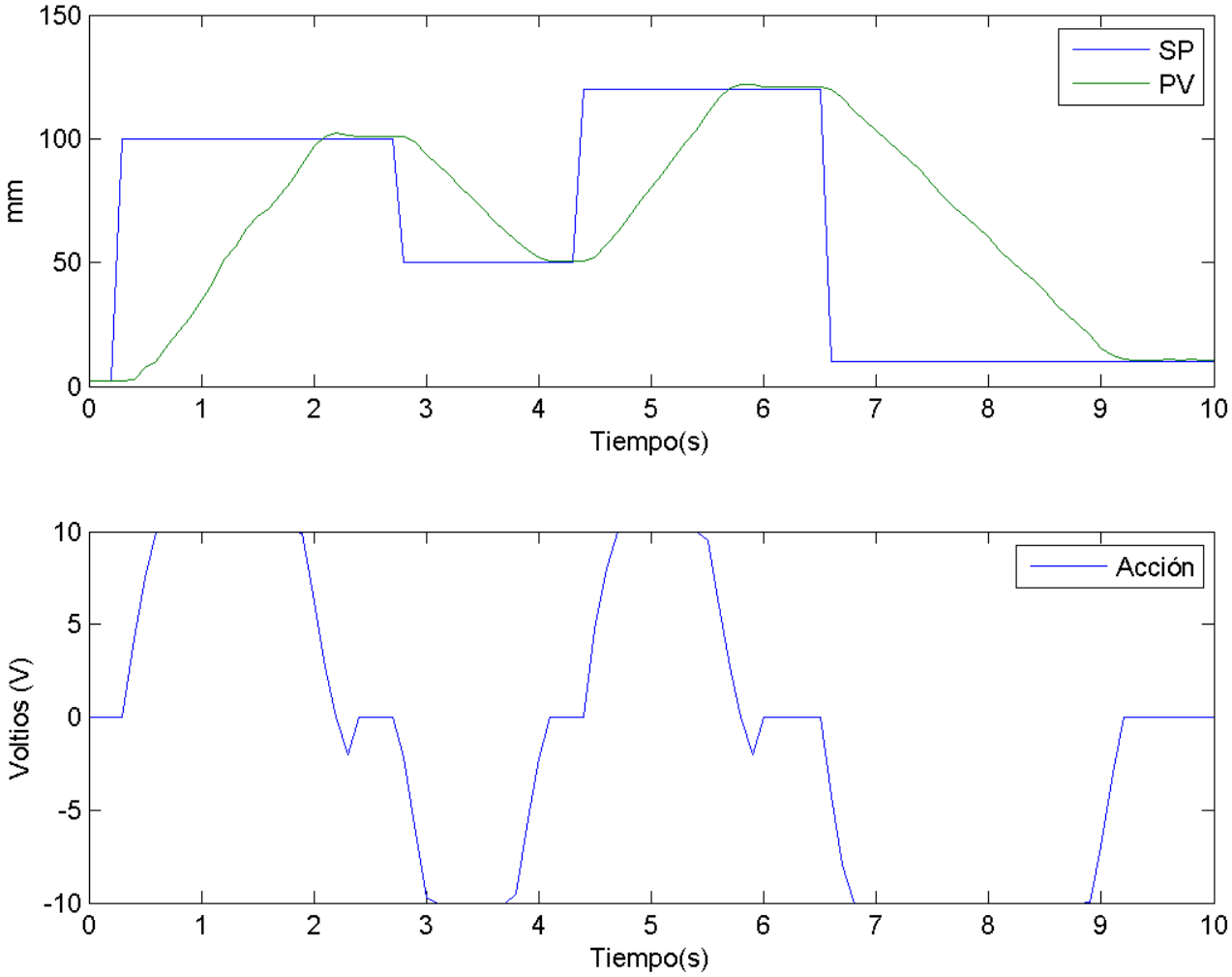


Figura 57. Gráfica obtenida de Matlab con las variables de Control tipo 1 de Nivel

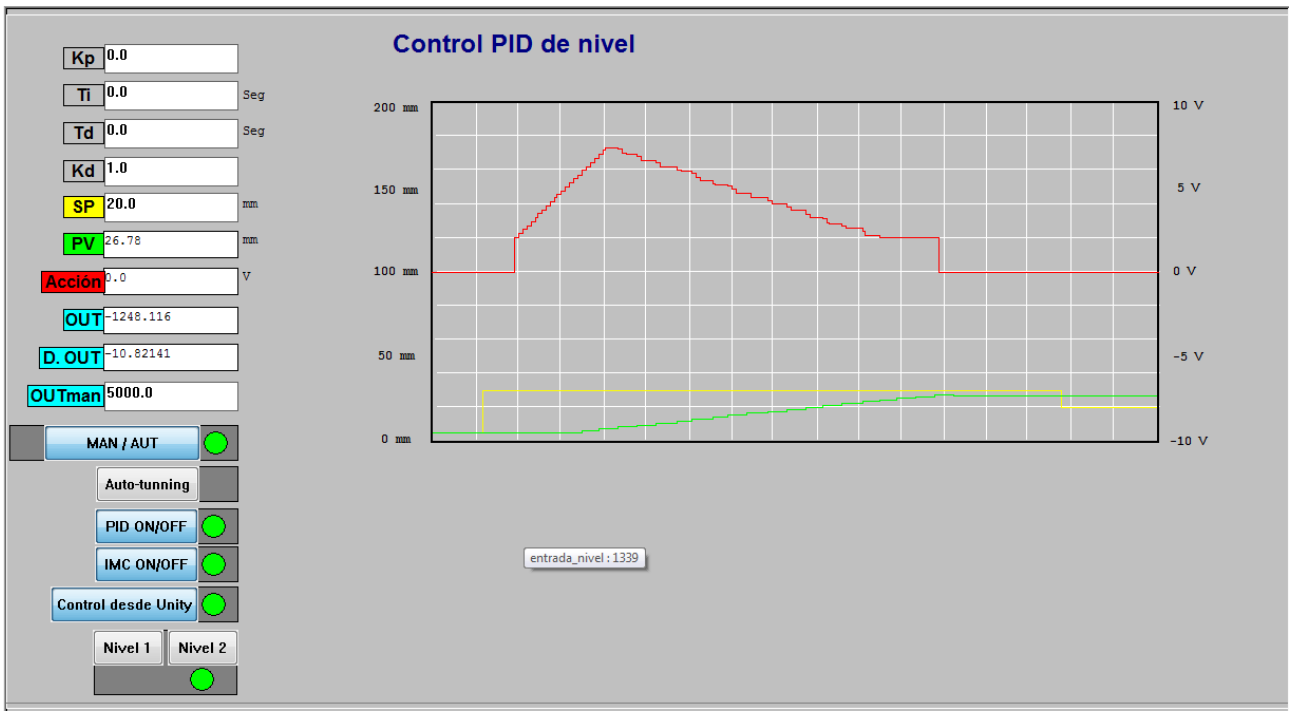


Figura 58. Pantalla de operador en Unity de Control tipo 2 de Nivel con IMC

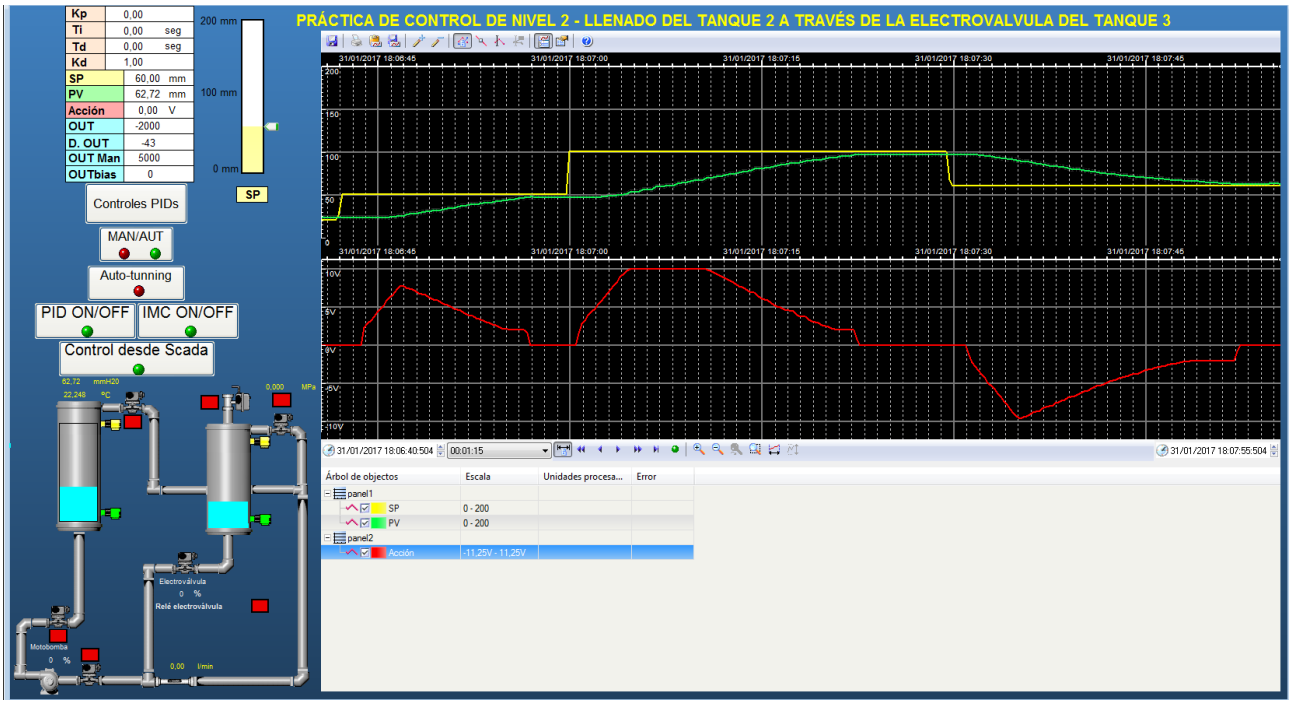


Figura 59. Pantalla SCADA de Control tipo 2 de Nivel con IMC

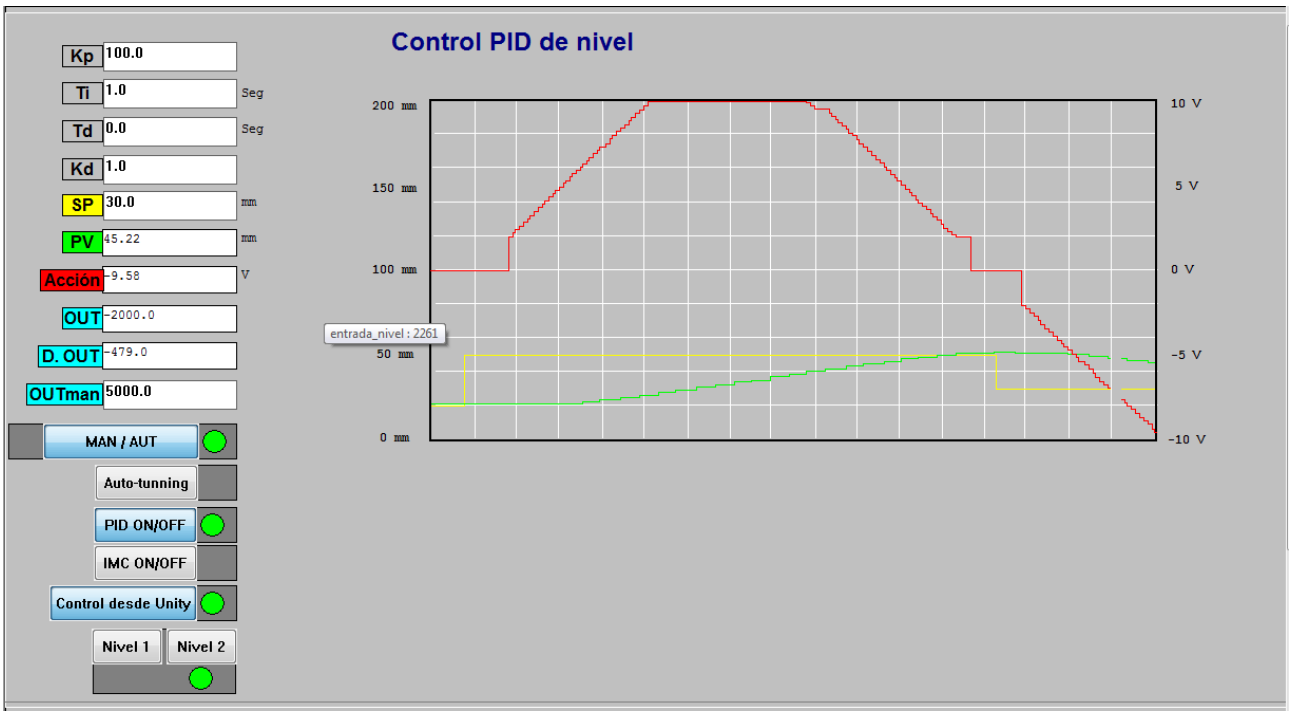


Figura 60. Pantalla de operador en Unity de Control tipo 2 de Nivel con PID

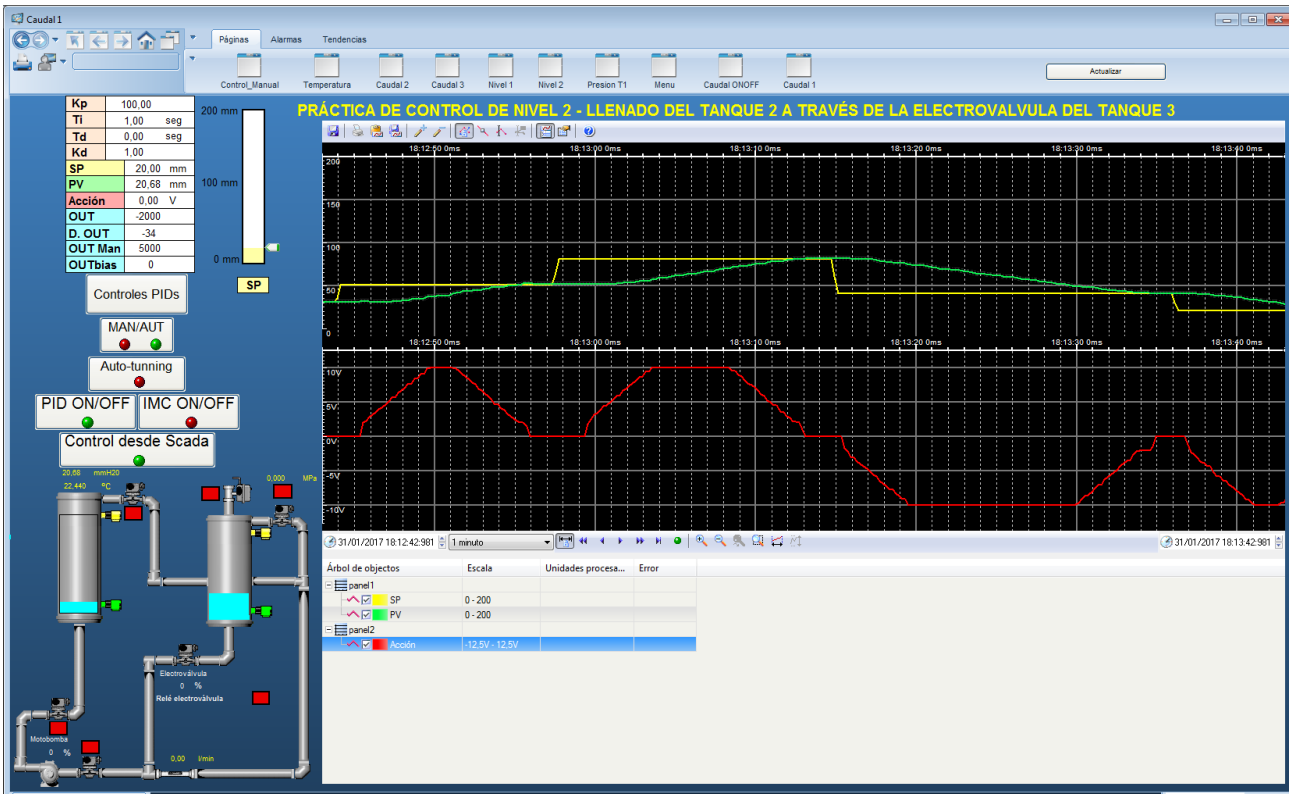


Figura 61. Pantalla SCADA de Control tipo 2 de Nivel con PID

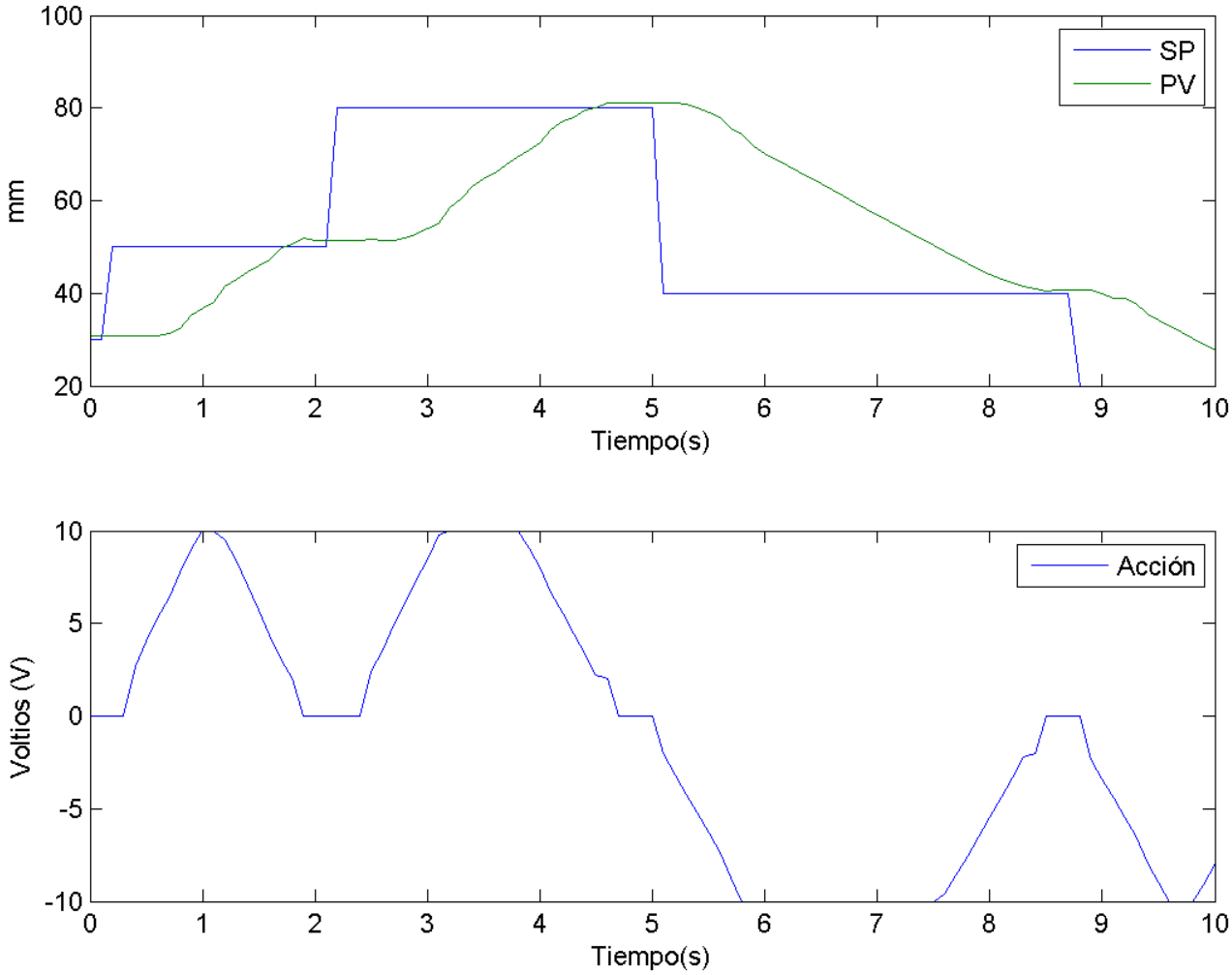


Figura 62. Gráfica obtenida de Matlab con las variables de Control tipo 2 de Caudal

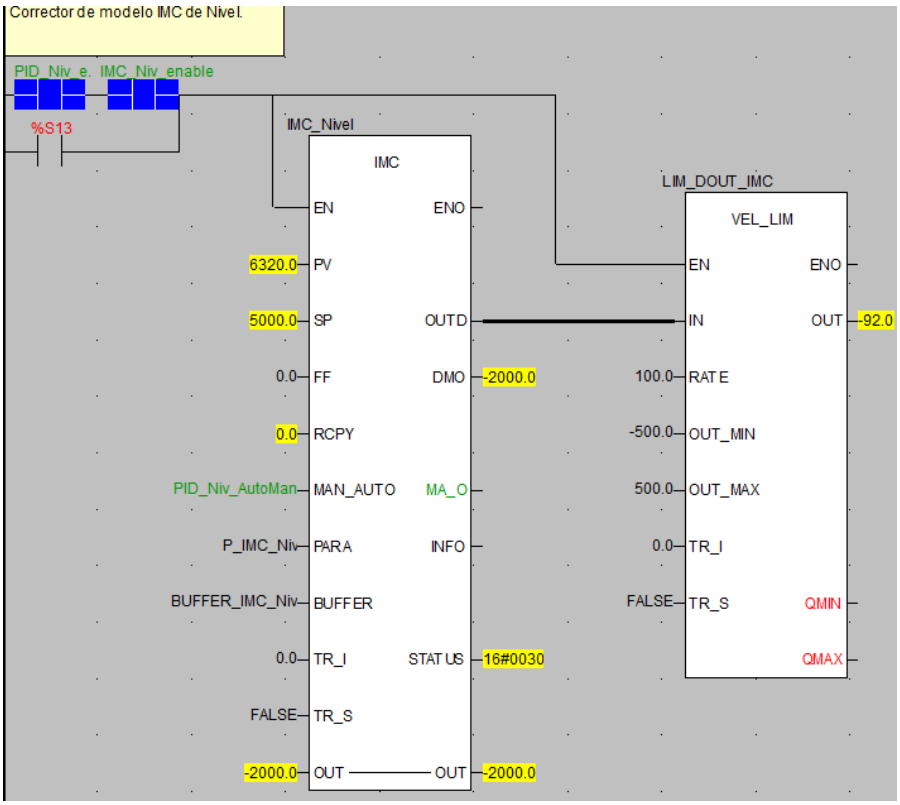


Figura 63. Bloque IMC de Control de Nivel

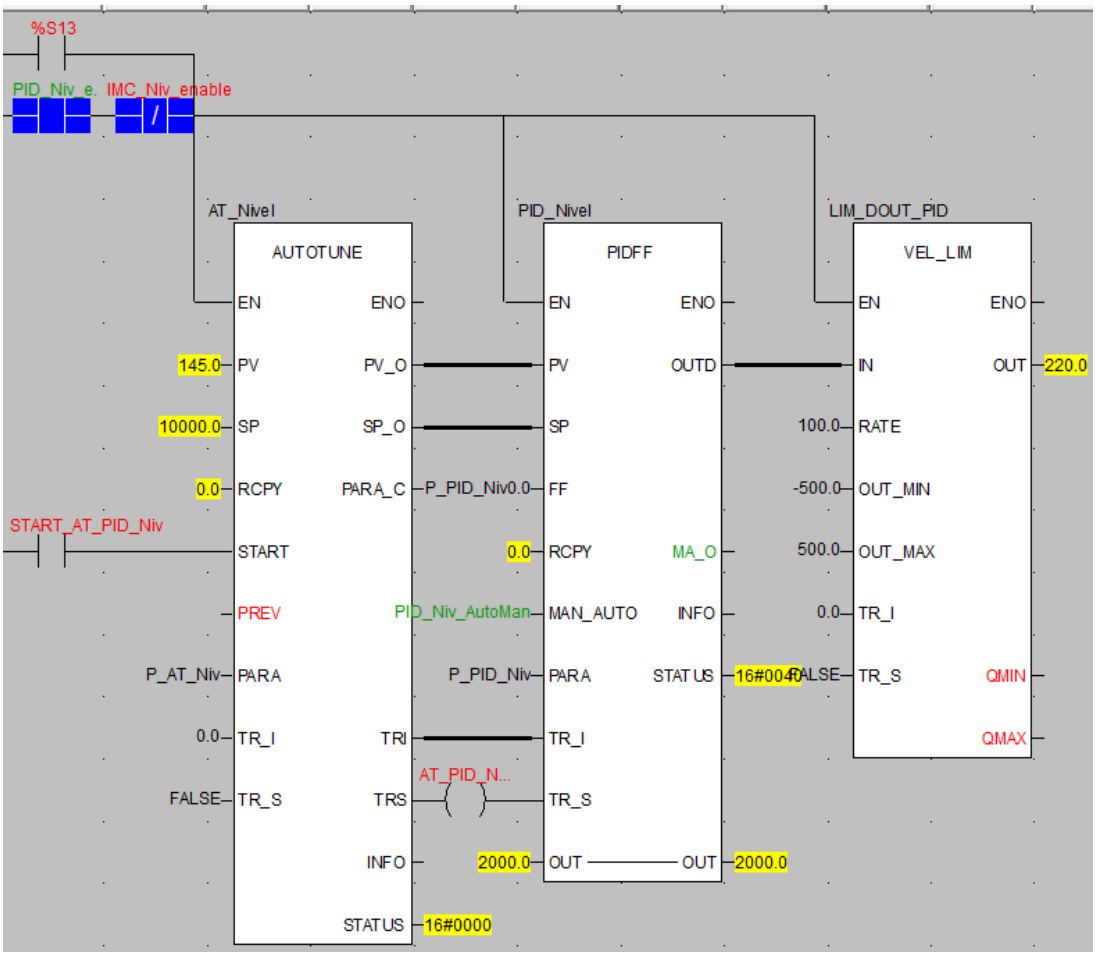


Figura 64. Bloque PID con Autotuning de Control de Nivel

C3. Control de Temperatura

En tercer lugar, probamos los diferentes reguladores de temperatura.

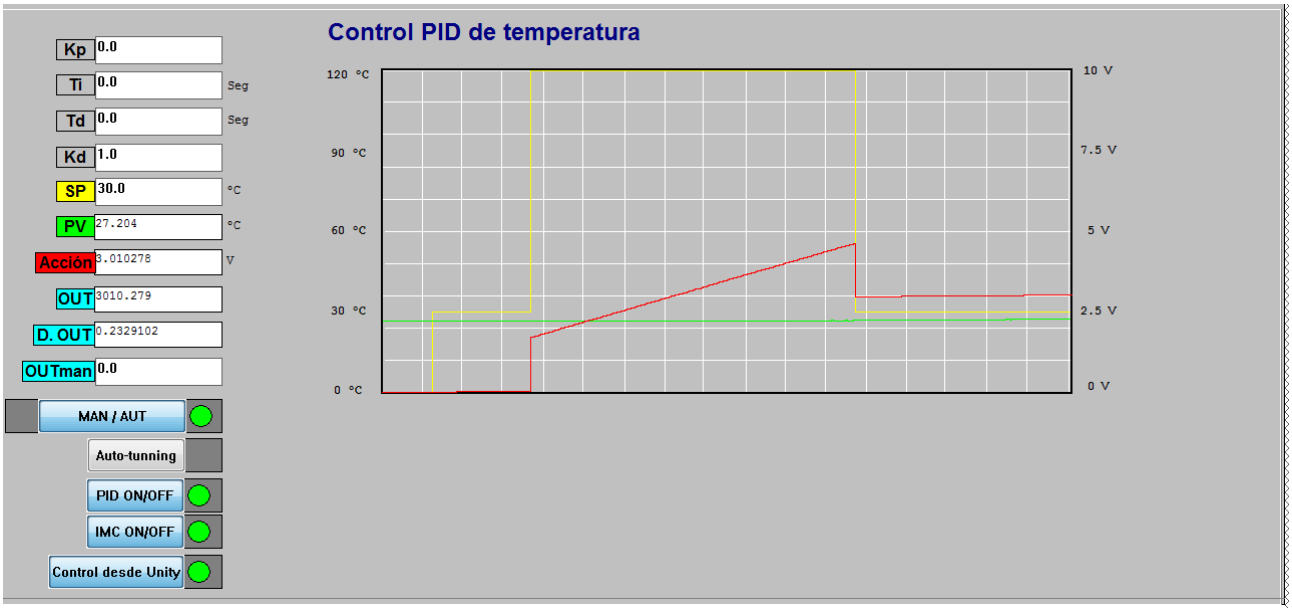


Figura 65. Pantalla de operador en Unity de Control de Temperatura con IMC

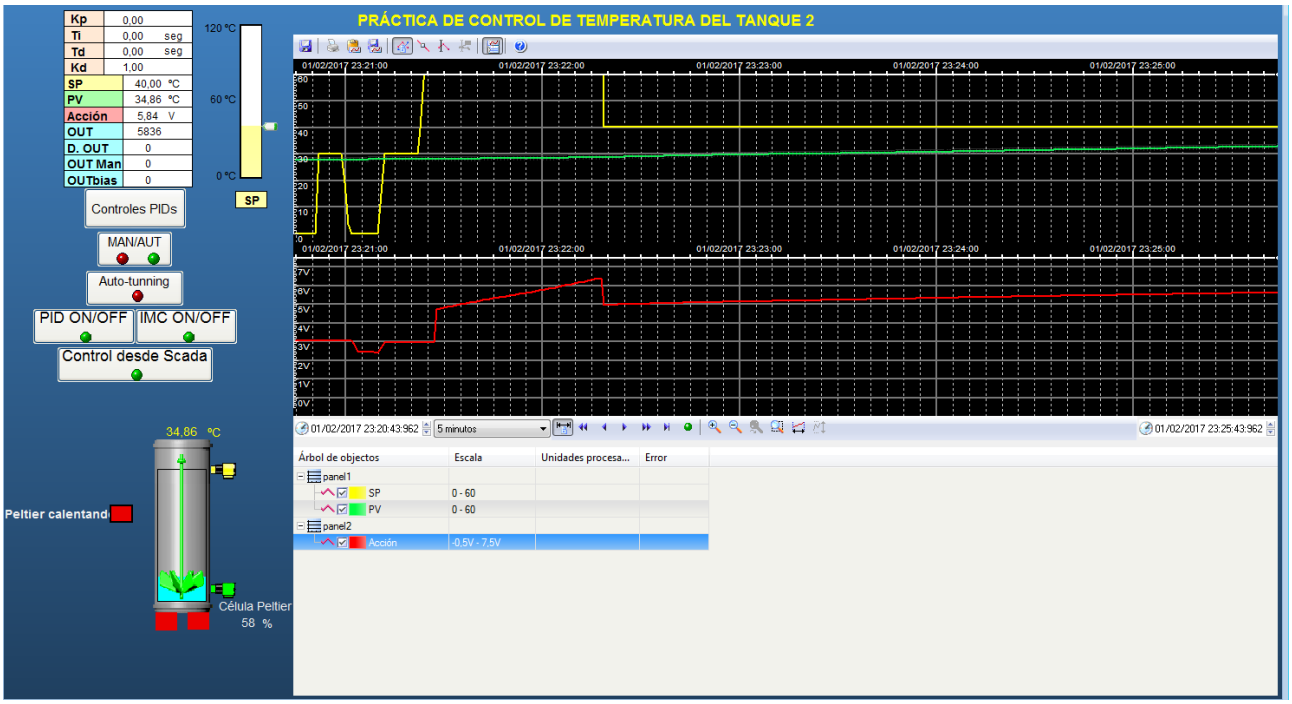


Figura 66. Pantalla SCADA de Control de Temperatura con IMC

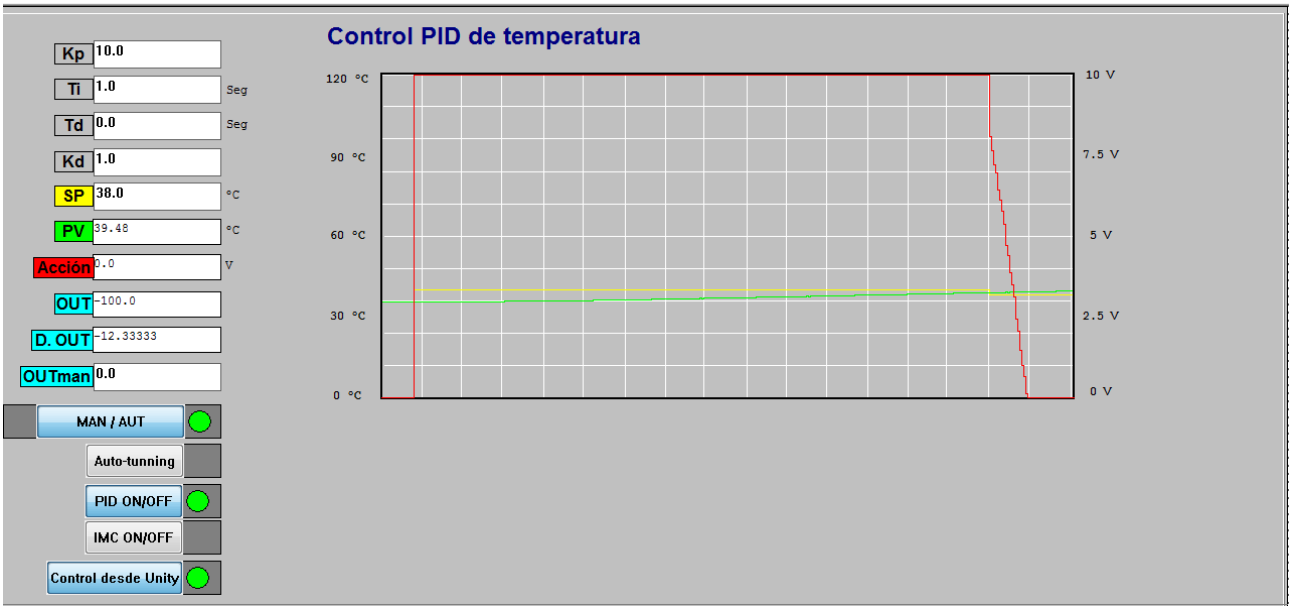


Figura 67. Pantalla de operador en Unity de Control de Temperatura con PID

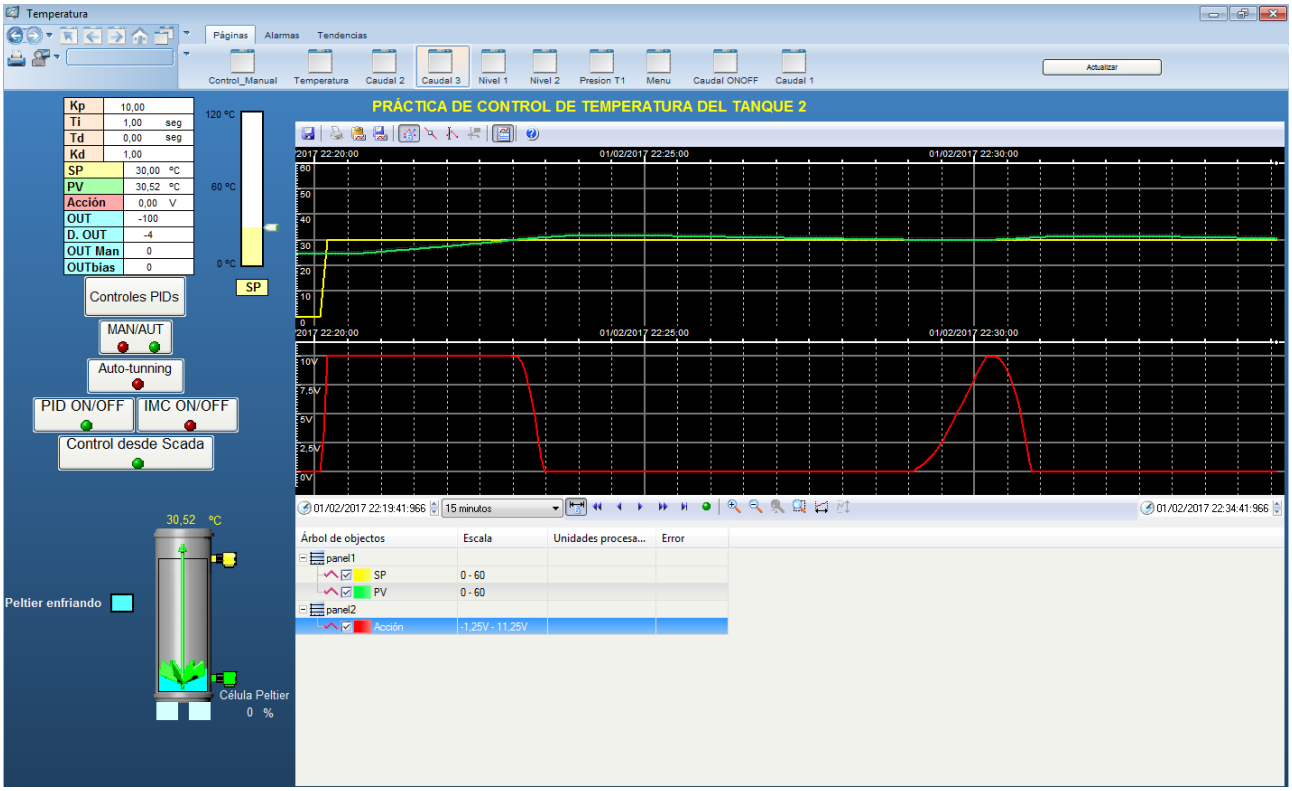


Figura 68. Pantalla SCADA de Control de Temperatura con PID

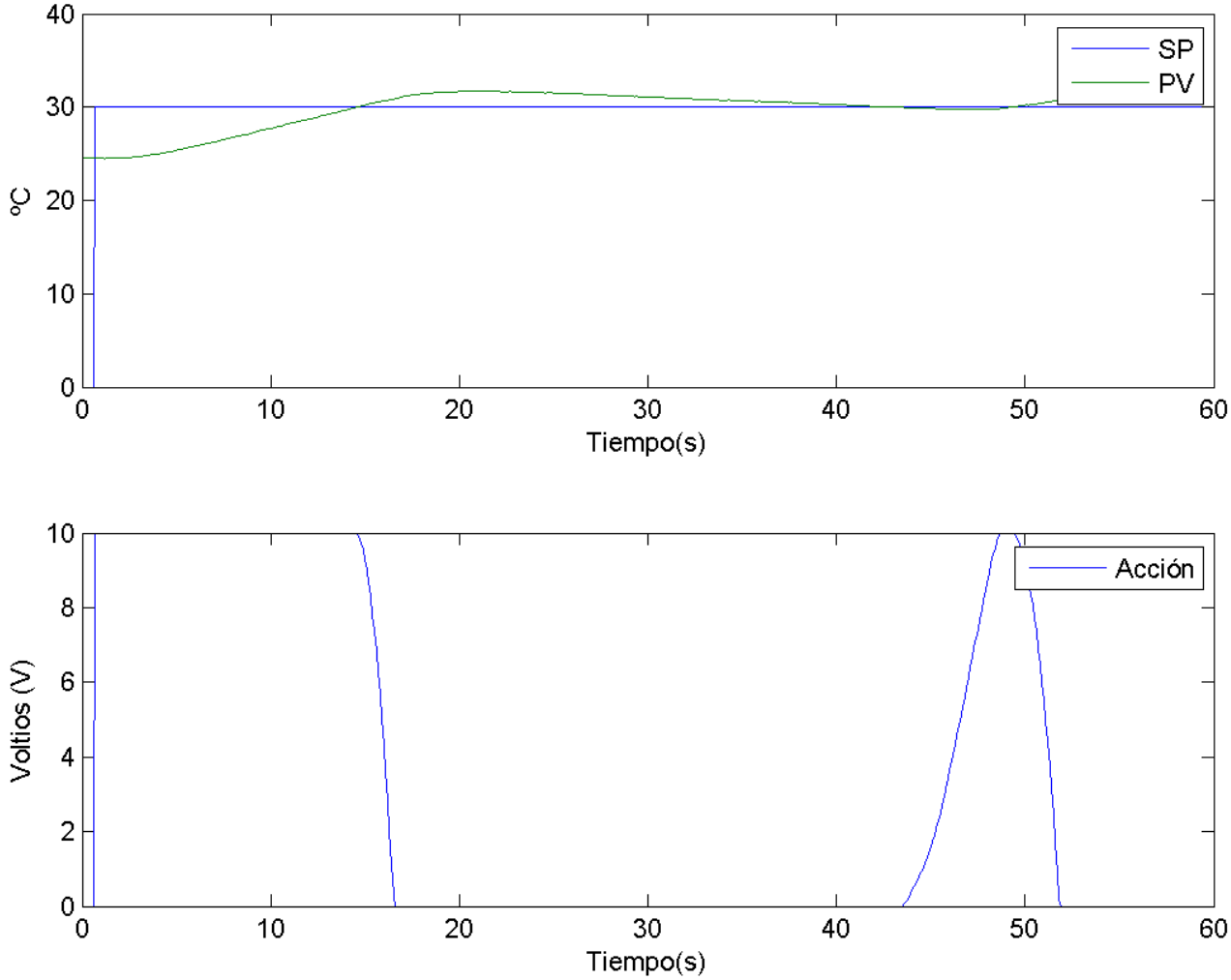


Figura 69. Gráfica obtenida de Matlab con las variables de Control de Temperatura

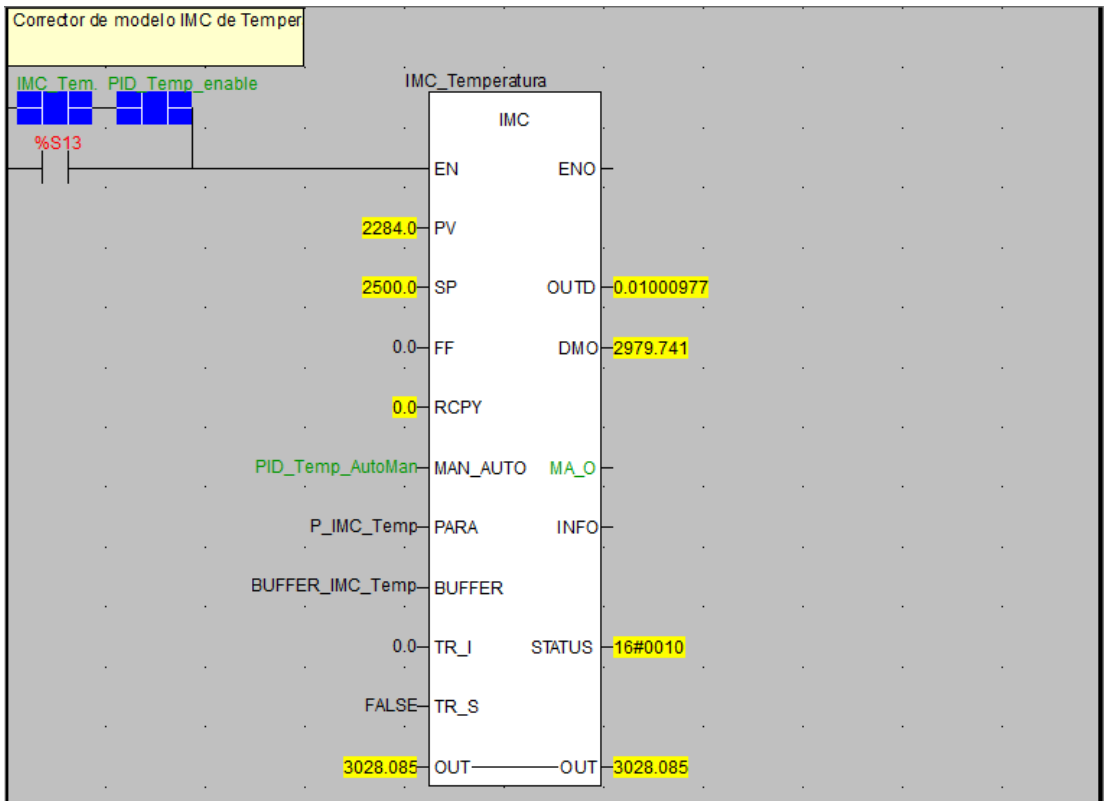


Figura 70. Bloque IMC de Control de Temperatura

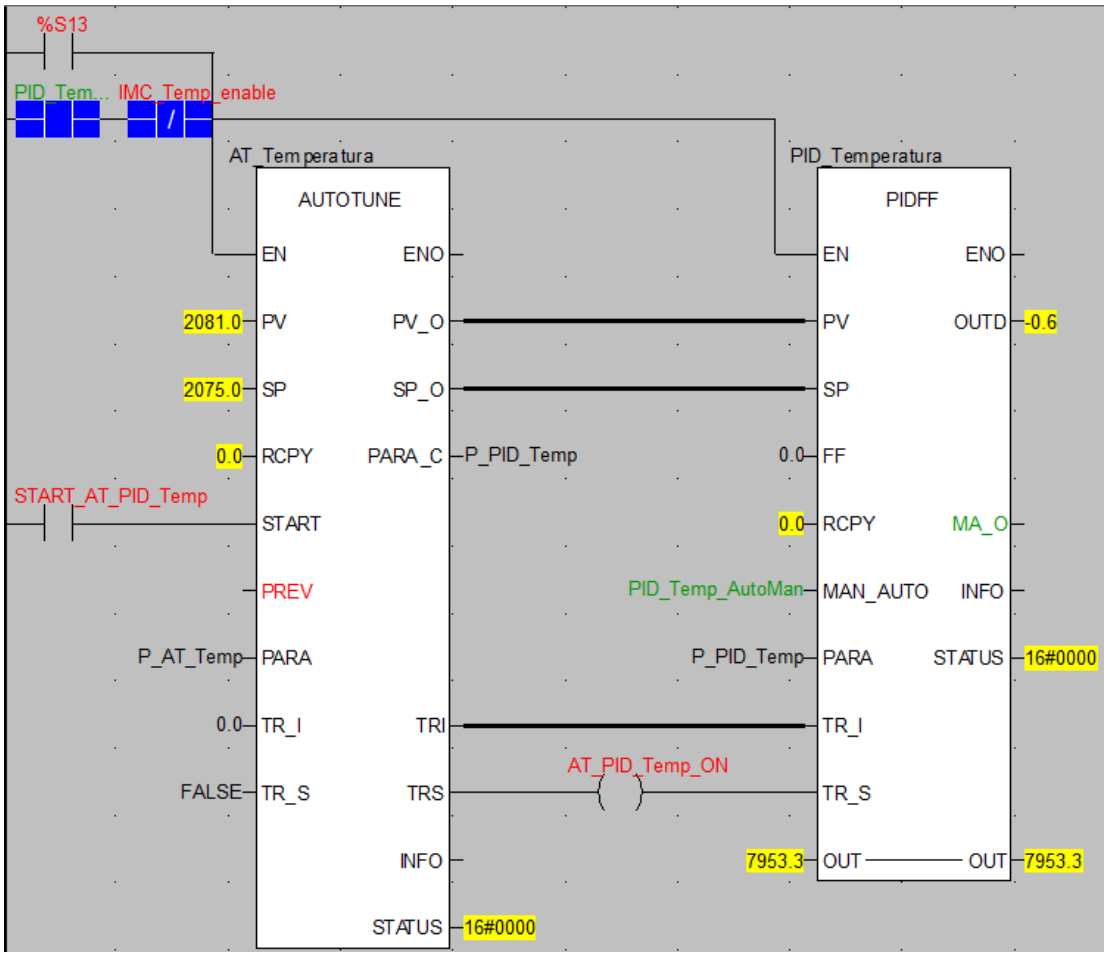


Figura 71. Bloque PID con Autotuning de Control de Temperatura

C4. Control de Presión del tanque izquierdo

En último lugar, probamos el control de presión con consigna proporcional.

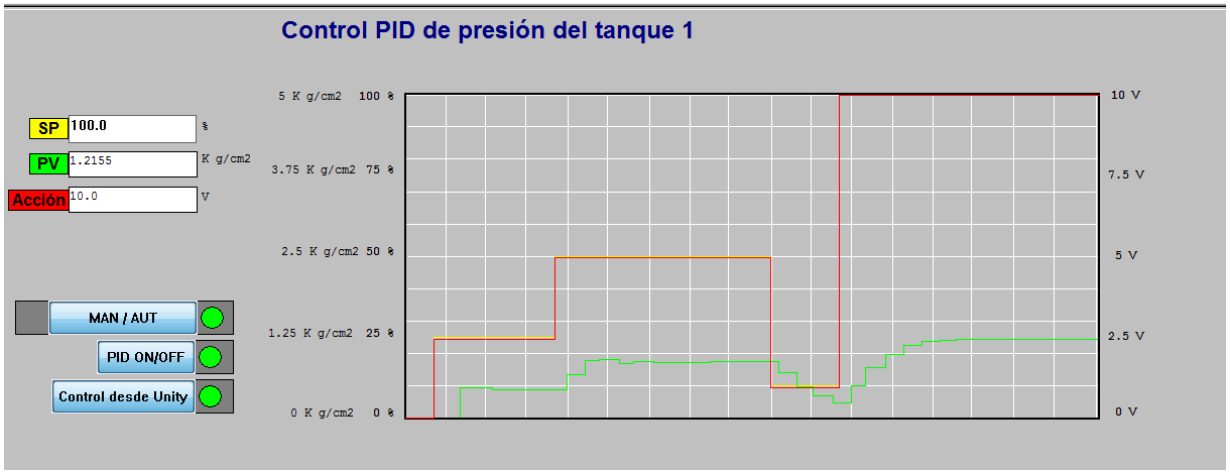


Figura 72. Pantalla de operador en Unity de Control de Presión del tanque izdo

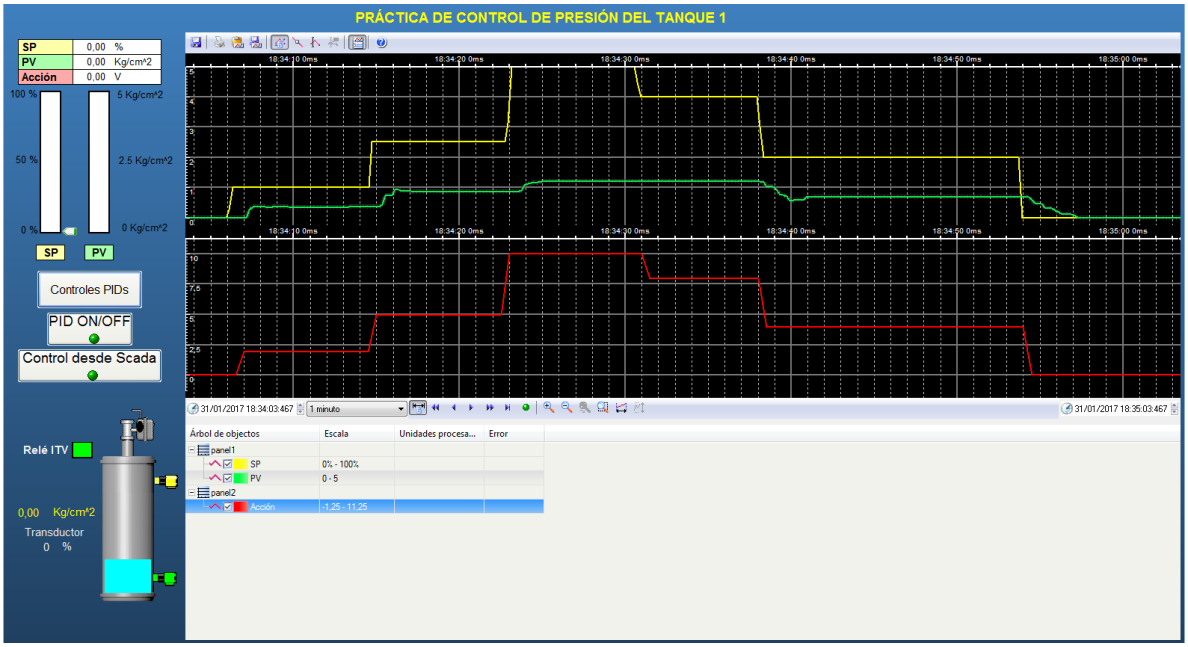


Figura 73. Pantalla SCADA de Control de Presión del tanque izdo

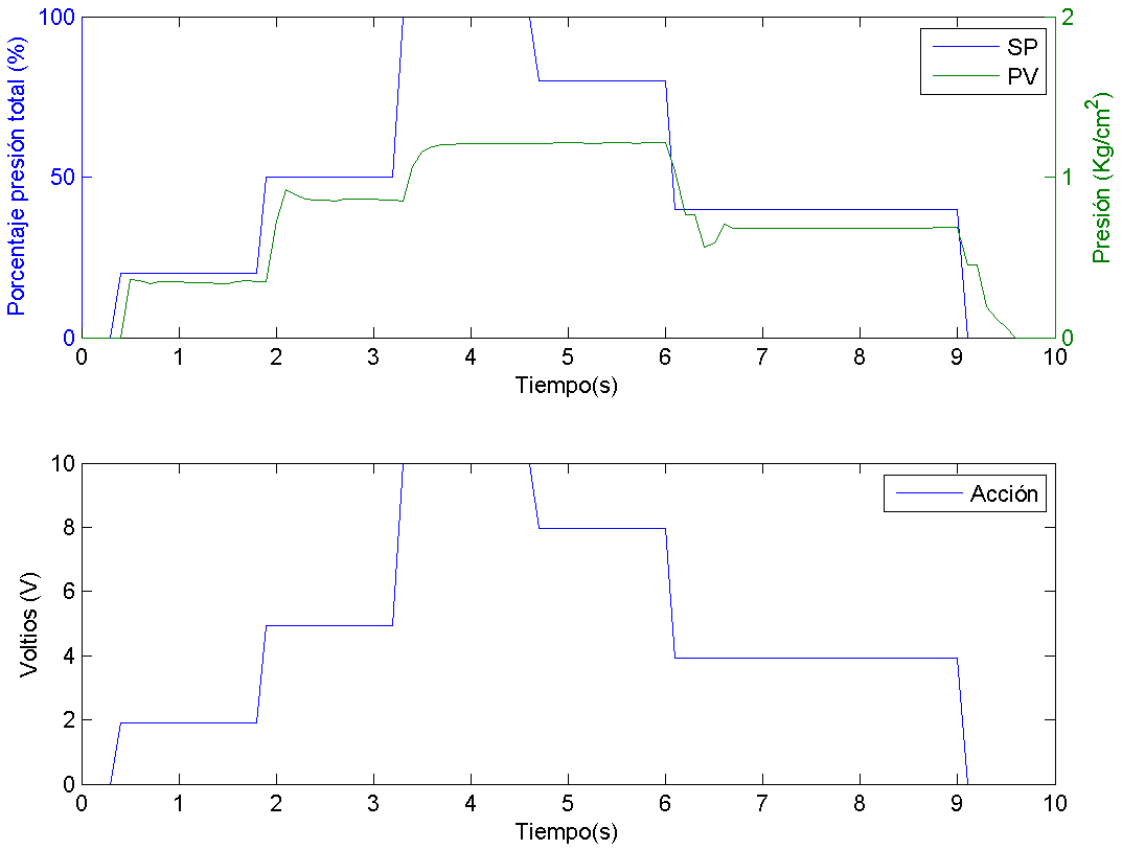


Figura 74. Gráfica obtenida de Matlab con las variables de Control de Presión del tanque izdo

Anexo B. Programa PLC

En este Anexo vamos a mostrar los bloques principales del programa del autómata. Lo dividiremos en las siguientes partes: Inicialización, lectura de entradas, comunicación con SCADA, control de prácticas, control ON/OFF de caudal, PID de caudal, PID de temperatura, PID de presión en el tanque 1, SFC de demostración, bloques varios, y escritura de salidas. También tendremos las variables empleadas y las pantallas de operador propias del autómata.

Inicialización

Es este primer apartado se hace la inicialización del programa del PLC, estableciendo unos factores de conversión para el SCADA y los valores iniciales de los PIDs.

```
(*seccion: INICIALIZAR.
Esta seccion activa varios bits de estado dentro del
autómata solamente durante el primer ciclo de ejecución. *)

If %S13 THEN (* %S13 solo está activo en el primer ciclo. *)
        SET(running);    (*running - estado del automata *)

        const_Caud := 2.0 / 32000.0;
        const_Temp := 120.0 / 32000.0;
        const_Niv := 200.0 / 32000.0;
        const_pT1 := 100.0 / 32000.0;

        kp_PID_Caud := 1.0;
        kp_PID_Niv := 100.0;
        kp_PID_Temp := 10.0;

        Ti_PID_Caud := 1.0;
        Ti_PID_Niv := 1.0;
        Ti_PID_Temp := 1.0;

        Td_PID_Caud := 0.0;
        Td_PID_Niv := 0.0;
        Td_PID_Temp := 0.0;

        kd_PID_Caud := 1.0;
        kd_PID_Niv := 1.0;
        kd_PID_Temp := 1.0;

        outbias_PID_Caud := 0.0;
        outbias_PID_Niv := 0.0;
        outbias_PID_Temp := 0.0;
END_IF;
```

Figura 75. Código de inicialización

Lectura de entradas

En este apartado se hace la lectura de las entradas de los sensores, así como su gestión y adaptación para poder usarlas.

```
(*seccion: ENTRADAS.
Aqui se realizan operaciones con algunas entradas, bien sea alguna temporización
o alguna corrección en la lectura de las variables, así como el comportamiento de
algunas salidas cuando las entradas ordenan que se pare el control del
autómata. *)

(* Se añade un retraso a las señales de nivel mínimo de los
tanques. De esta manera, cuando pasen tres segundos (PT) desde que el
sensor se ha puesto a 1, la salida de la función TON dará 1. *)

temporizador_vaciado_tanque_1 (IN :=sensor_min_tank1, PT:=T#3s, Q=>retraso1);
temporizador_vaciado_tanque_2 (IN :=sensor_min_tank2, PT:=T#3s, Q=>retraso2);
temporizador_vaciado_tanque_3 (IN :=sensor_min_tank3, PT:=T#3s, Q=>retraso3);

(* El bit interno min_tank se activará tras tres segundos de que se active
  "min" gracias a la señal retraso temporizada en el codigo anterior, y
    se desactivará cuando la señal min pase a 0. *)

if (NOT(sensor_min_tank1)) THEN reset (min_tank1);
elsif (retraso1) then set (min_tank1);
END_IF;

if (NOT(sensor_min_tank2)) THEN reset (min_tank2);
elsif (retraso2) then set (min_tank2);
END_IF;

if (NOT(sensor_min_tank3)) THEN reset (min_tank3);
elsif (retraso3) then set (min_tank3);
END_IF;

(* El control se activa con el botón de marcha y se desactiva con el botón de
   parar (start y stop). *)

IF marcha then set(running);
ELSIF paro then reset(running);
END_IF;

(* Leemos los conversores A/D de los sensores, y hacemos una copia con acomodación
para enviar el valor leido al SCADA para su visualización. *)

(* Sensor de presión del tanque 1 *)
PV_pT1 := INT_TO_REAL(entrada_transductor);
IF (PV_pT1 < 0.0) THEN
        PV_pT1_SCADA := 0.0;
ELSE
        PV_pT1_SCADA := (PV_pT1 * 5.0 / 10000.0) + 0.05;
END_IF;

(* Sensor de presión del tanque 3 *)
PV_pT3 := INT_TO_REAL(entrada_presostato);
IF (PV_pT3 < 0.0) THEN
        PV_pT3_SCADA := 0.0;
ELSE
        PV_pT3_SCADA := PV_PT3 * 1.04 / 10000.0;
END_IF;

(* Sensor de caudal *)
PV_Caud := INT_TO_REAL(entrada_caudal);
IF (PV_Caud < 0.0) THEN
        PV_Caud_SCADA := 0.0;
ELSE
        PV_Caud_SCADA := (PV_Caud * 3.5 / 10000.0) + 0.5;
END_IF;

(* Sensor de nivel del tanque 2 *)
PV_Niv := INT_TO_REAL(entrada_nivel);
PV_Niv_SCADA := PV_Niv * 200.0 / 10000.0;

(* Sensor de tamperatura del tanque 2 *)
PV_Temp := INT_TO_REAL(entrada_temperatura);
PV_Temp_SCADA := PV_Temp * 120.0 / 10000.0;


(* Si el control está desactivado,
 se resetea la entrada al PID*)

IF (not(running)) then
        RESET(PID_pT1_enable);
        RESET(PID_Temp_enable);
        RESET(PID_Niv_enable);
        RESET(PID_Caud_enable);

        RESET(Control_Caudal1);
        RESET(Control_Caudal2);
        RESET(Control_Caudal3);

        RESET(Control_Nivel1);
        RESET(Control_Nivel2);
END_IF;
```

Figura 76. Código de lectura de entradas

Comunicación con SCADA

En este apartado se hacen las operaciones necesarias para enviar los datos necesarios al SCADA y para recibir del mismo la manipulación del usuario.

```
(*seccion: SCADA.
Se realiza una lectura de algunas de las entradas digitales
y se almacenan para que puedan ser visualizadas por el SCADA. *)

estado_min1:=sensor_min_tank1;
estado_min2:=sensor_min_tank2;
estado_min3:=sensor_min_tank3;
estado_max1:=sensor_max_tank1;
estado_max2:=sensor_max_tank2;
estado_max3:=sensor_max_tank3;

estado_plcreg:=auto_man;
estado_valvula_salida_tank1:=valvula_salida_tank1;
estado_valvula_entrada_tank1:=valvula_entrada_tank1;
estado_valvula_salida_tank2:=valvula_salida_tank2;
estado_valvula_entrada_tank2:=valvula_entrada_tank2;
estado_valvula_entrada_tank3:=valvula_entrada_tank3;
estado_valvula_salida_tankaux:=valvula_salida_tankaux;
estado_valvula_salida_motobomba:=valvula_salida_motobomba;


(* Tambien comprobamos los valores de los parámetros para los PID que el usuario establece en el SCADA,
para comprobar que son corrector y adaptarlos en caso contrario *)

(* Comprobamos las consignas establecidas por el usuario. *)
(* Rango de Presión entre 1% y 100%. *)
IF (SP_pT1_SCADA <= 0.0) THEN
        SP_pT1_SCADA := 0.0;

ELSIF (SP_pT1_SCADA < 0.01 * 32000.0) THEN
        SP_pT1_SCADA := 0.01 * 32000.0;

ELSIF (SP_pT1_SCADA > 32000.0) THEN             (* 100% *)
        SP_pT1_SCADA := 32000.0;                (* 100% *)
END_IF;

(* Rango de Temperatura entre 0ºC y 120ºC. *)
IF (SP_Temp_SCADA <= 0.0) THEN
        SP_Temp_SCADA := 0.0;

ELSIF (SP_Temp_SCADA > 32000.0) THEN            (* 120ºC *)
        SP_Temp_SCADA := 32000.0;               (* 120ºC *)
END_IF;

(* Rango de Caudal entre 0.5 l/min y 2 l/min. *)
IF (SP_Caud_SCADA <= 0.0) THEN
        SP_Caud_SCADA := 0.0;

ELSIF (SP_Caud_SCADA < 32000.0 / 4.0) THEN      (* 0.5 l/min *)
        SP_Caud_SCADA := 32000.0 / 4.0;         (* 0.5 l/min *)

ELSIF (SP_Caud_SCADA > 32000.0) THEN
        SP_Caud_SCADA := 32000.0;
END_IF;

(* Rango de Nivel entre 0mm y 200mm. *)
IF (SP_Niv_SCADA <= 0.0) THEN
        SP_Niv_SCADA := 0.0;

ELSIF (SP_Niv_SCADA > 32000.0) THEN             (* 200mm *)
        SP_Niv_SCADA := 32000.0;                (* 200mm *)
END_IF;

(* Comprobamos los parámetros de los PID *)
(* Kp *)
IF (kp_PID_Caud < 0.0)THEN
        kp_PID_Caud := 0.0;
END_IF;

IF (kp_PID_Niv < 0.0)THEN
        kp_PID_Niv := 0.0;
END_IF;

IF (kp_PID_Temp < 0.0)THEN
        kp_PID_Temp := 0.0;
END_IF;

(* Ti *)
IF (Ti_PID_Caud < 0.0)THEN
        Ti_PID_Caud := 0.0;
END_IF;

IF (Ti_PID_Niv < 0.0)THEN
        Ti_PID_Niv := 0.0;
END_IF;

IF (Ti_PID_Temp < 0.0)THEN
        Ti_PID_Temp := 0.0;
END_IF;

(* Td *)
IF (Td_PID_Caud < 0.0)THEN
        Td_PID_Caud := 0.0;
END_IF;
```

Figura 77. Código de comunicación SCADA. Parte I

```
 95
 96  IF (Td_PID_Niv < 0.0)THEN
 97          Td_PID_Niv := 0.0;
 98  END_IF;
 99
100  IF (Td_PID_Temp < 0.0)THEN
101          Td_PID_Temp := 0.0;
102  END_IF;
103
104  (* Kd. Ponemos el mínimo en 0.1 para evitar comportamientos indesesados en los PID *)
105  IF (kd_PID_Caud < 1.0)THEN
106          kd_PID_Caud := 1.0;
107  END_IF;
108
109  IF (kd_PID_Niv < 1.0)THEN
110          kd_PID_Niv := 1.0;
111  END_IF;
112
113  IF (kd_PID_Temp < 1.0)THEN
114          kd_PID_Temp := 1.0;
115  END_IF;
116
117  (* Copiamos los estados del SFC de demostracíón a unas variables para poder controlar las pantallas del SCADA
117>>*)
118  IF Reposo.X THEN
119          EstadoSFCdemo := 0;
120  ELSIF Preparacion.X THEN
121          EstadoSFCdemo := 1;
122  ELSIF Transvase_23.X THEN
123          EstadoSFCdemo := 2;
124  ELSIF Transvase_31.X THEN
125          EstadoSFCdemo := 3;
126  ELSIF (Transvase_12.X OR Transvase_Fin.X OR Transvase_OUT.X) THEN
127          EstadoSFCdemo := 4;
128  ELSIF Pre_Nivel21_100mm.X THEN
129          EstadoSFCdemo := 5;
130  ELSIF Nivel21_100mm.X THEN
131          EstadoSFCdemo := 6;
132  ELSIF AT_Niv_IN.X THEN
133          EstadoSFCdemo := 7;
134  ELSIF AT_Niv.X THEN
135          EstadoSFCdemo := 8;
136  ELSIF Nivel23_IN.X THEN
137          EstadoSFCdemo := 9;
138  ELSIF Nivel23_20mm.X THEN
139          EstadoSFCdemo := 10;
140  ELSIF (Nivel23_80mm.X OR Fin_Nivel2.X OR Nivel23_OUT.X) THEN
141          EstadoSFCdemo := 11;
142  ELSIF Pre_Caudal_ONOFF.X THEN
143          EstadoSFCdemo := 12;
144  ELSIF CaudalONOFF.X THEN
145          EstadoSFCdemo := 13;
146  ELSIF AT_Caudal2_IN.X THEN
147          EstadoSFCdemo := 14;
148  ELSIF AT_Caudal2.X THEN
149          EstadoSFCdemo := 15;
150  ELSIF Caudal2_IN.X THEN
151          EstadoSFCdemo := 16;
152  ELSIF Caudal2_1.X THEN
153          EstadoSFCdemo := 17;
154  ELSIF (Caudal2_15.X OR Fin_Caudal2.X OR Caudal2_OUT.X) THEN
155          EstadoSFCdemo := 18;
156  ELSIF Pre_Temp.X THEN
157          EstadoSFCdemo := 19;
158  ELSIF Temperatura_25C.X THEN
159          EstadoSFCdemo := 20;
160  ELSIF Temperatura_mas5C.X THEN
161          EstadoSFCdemo := 21;
162  ELSIF AT_Temp_IN.X THEN
163          EstadoSFCdemo := 22;
164  ELSIF AT_Temp.X THEN
165          EstadoSFCdemo := 23;
166  ELSIF Temperatura_30C.X THEN
167          EstadoSFCdemo := 24;
168  ELSIF Temperatura_mas10C.X THEN
169          EstadoSFCdemo := 25;
170  ELSIF IMC_Temperatura_25C.X THEN
171          EstadoSFCdemo := 26;
172  ELSIF IMC_Temperatura_mas5C.X THEN
173          EstadoSFCdemo := 27;
174  ELSIF IMC_Temperatura_30C.X THEN
175          EstadoSFCdemo := 28;
176  ELSIF IMC_Temperatura_mas10C.X THEN
177          EstadoSFCdemo := 29;
178  ELSIF Nivel21_20mm.X THEN
179          EstadoSFCdemo := 30;
180  END_IF;
```

Figura 78. Código de comunicación SCADA. Parte II

Control de prácticas

En este punto hacemos la gestión de las diferentes prácticas que están programadas con los PID, así como la activación y desactivación de los PIDs e IMCs cuando el usuario lo solicita.

```
(* Sección de control de los elementos de las diferentes practicas. *)


(* Cuando activemos una modalidad, desactivaremos otras que puedan causar conflicto por estar usando los mismo
>>s elementos. *)

        (* Prácticas de Nivel. Incompatibles con las prácticas de Caudal y con la de presión (solo la de Nivel
>> 1). *)
IF Control_Nivel1_SCADA THEN
        SET(Control_Nivel1);
        RESET(Control_Nivel2);
        RESET(Control_Nivel1_SCADA);

        RESET(Control_Caudal1);
        RESET(Control_Caudal2);
        RESET(Control_Caudal3);

        RESET(PID_pT1_enable_SCADA);
        RESET(PID_Caud_enable_SCADA);
        RESET(ONOFF_Caud_enable_SCADA);

ELSIF Control_Nivel2_SCADA THEN
        RESET(Control_Nivel1);
        SET(Control_Nivel2);
        RESET(Control_Nivel2_SCADA);

        RESET(Control_Caudal1);
        RESET(Control_Caudal2);
        RESET(Control_Caudal3);

        RESET(PID_Caud_enable_SCADA);
        RESET(ONOFF_Caud_enable_SCADA);
END_IF;

        (* Prácticas de Caudal. Incompatibles con las prácticas de Nivel y con la de presión (solo la de Cauda
>>l 2). *)
IF Control_Caudal1_SCADA THEN
        SET(Control_Caudal1);
        RESET(Control_Caudal2);
        RESET(Control_Caudal3);
        RESET(Control_Caudal1_SCADA);

        RESET(Control_Nivel1);
        RESET(Control_Nivel2);

        RESET(PID_Niv_enable_SCADA);

ELSIF Control_Caudal2_SCADA THEN
        RESET(Control_Caudal1);
        SET(Control_Caudal2);
        RESET(Control_Caudal3);
        RESET(Control_Caudal2_SCADA);

        RESET(Control_Nivel1);
        RESET(Control_Nivel2);

        RESET(PID_pT1_enable_SCADA);
        RESET(PID_Niv_enable_SCADA);
        RESET(ONOFF_Caud_enable_SCADA);

ELSIF Control_Caudal3_SCADA THEN
        RESET(Control_Caudal1);
        RESET(Control_Caudal2);
        SET(Control_Caudal3);
        RESET(Control_Caudal3_SCADA);

        RESET(Control_Nivel1);
        RESET(Control_Nivel2);

        RESET(PID_Niv_enable_SCADA);
        RESET(ONOFF_Caud_enable_SCADA);
END_IF;

        (* Práctica de Presión del Tanque 1. Incompatibles con las prácticas de Nivel 1 y con la de Caudal 2.
>>*)
IF PID_pT1_enable_SCADA THEN
        RESET(Control_Nivel1);
        RESET(Control_Caudal2);
END_IF;

        (* Práctica de Control de Caudal Mediante Control ON/OFF. Incompatibles con las prácticas de Nivel y C
>>audal 2 y 3 (ya que lo usaremos con la de Caudal 1). *)
IF Re(ONOFF_Caud_enable_SCADA) THEN
        SET(Control_Caudal1);
        RESET(Control_Caudal2);
        RESET(Control_Caudal3);

        RESET(Control_Nivel1);
        RESET(Control_Nivel2);

        RESET(PID_Caud_enable_SCADA);
        RESET(PID_Niv_enable_SCADA);
END_IF;

```

Figura 79. Código de control de prácticas. Parte I

```
90          (* Si volemos a activar el control normal de Caudal, reseteamos en Control ON/OFF. *)
91  IF PID_Caud_enable_SCADA THEN
92          RESET(ONOFF_Caud_enable_SCADA);
93  END_IF;
94
95          (* Si activamos la modalidad de control desde Matlab, anilamos todo el resto de controles. *)
96  IF Control_Matlab THEN
97          SET(Control_Caudal1);
98          RESET(Control_Caudal2);
99          RESET(Control_Caudal3);
100
101         RESET(Control_Nivel1);
102         RESET(Control_Nivel2);
103
104         RESET(PID_Caud_enable_SCADA);
105         RESET(IMC_Caud_enable_SCADA);
106         RESET(ONOFF_Caud_enable_SCADA);
107
108         RESET(PID_Niv_enable_SCADA);
109         RESET(IMC_Niv_enable_SCADA);
110
111         RESET(PID_pT1_enable_SCADA);
112
113         RESET(PID_Temp_enable_SCADA);
114         RESET(IMC_Temp_enable_SCADA);
115 END_IF;
116
117
118         (* Comprobamos las ordenes de activación de los PID. Los PID de Caudal y Nivel sólo
119         los activaremos si hay una práctica de estos modos seleccionada. *)
120
121 IF NOT (modo_demo) THEN
122         PID_Caud_enable := (PID_Caud_enable_SCADA and
123                   (Control_Caudal1 or Control_Caudal2 or Control_Caudal3));
124         IMC_Caud_enable := PID_Caud_enable and IMC_Caud_enable_SCADA;
125         ONOFF_Caud_enable := ONOFF_Caud_enable_SCADA and Control_Caudal1;
126
127         PID_Niv_enable := (PID_Niv_enable_SCADA and
128                  (Control_Nivel1 or Control_Nivel2));
129         IMC_Niv_enable := PID_Niv_enable and IMC_Niv_enable_SCADA;
130
131         PID_pT1_enable := PID_pT1_enable_SCADA;
132
133         PID_Temp_enable := PID_Temp_enable_SCADA;
134         IMC_Temp_enable := PID_Temp_enable and IMC_Temp_enable_SCADA;
135
136 END_IF;
137
138         (* Volcamos las variables que indican si los PID y/o IMC están habilitados, para visualizar su estado
138>>desde el SCADA. *)
139
140 PID_Caud_enable_SCADA := PID_Caud_enable;
141 PID_Niv_enable_SCADA := PID_Niv_enable;
142 PID_pT1_enable_SCADA := PID_pT1_enable;
143 PID_Temp_enable_SCADA := PID_Temp_enable;
144
145 IMC_Caud_enable_SCADA := IMC_Caud_enable;
146 IMC_Niv_enable_SCADA := IMC_Niv_enable;
147 IMC_Temp_enable_SCADA := IMC_Temp_enable;
148
149 ONOFF_Caud_enable_SCADA := ONOFF_Caud_enable;
150
151
152 (* Activamos las válvulas y relés necesarios en cada práctica *)
153
154 IF NOT(modo_manual) THEN
155         accion_agitador := PID_Temp_enable;
156         accion_ventilador :=  PID_Temp_enable and (accion_Peltier = 0);
157         accion_rele_Peltier := PID_Temp_enable and (accion_Peltier = 0);
158
159         accion_rele_itv := (PID_Caud_enable and Control_Caudal2) or PID_pT1_enable or (PID_Niv_enable and Cont
159>>rol_Nivel1 and (accion_itv > 0));
160         accion_rele_electrovalvula := (PID_Caud_enable and Control_Caudal3) or (PID_Niv_enable and Control_Niv
160>>el2 and (accion_electrovalvula > 0));
161
162         accion_valvula_salida_tank1 :=  (PID_Caud_enable and Control_Caudal2) or (PID_Niv_enable and Control_N
162>>ivel1 and (accion_itv > 0));
163         accion_valvula_entrada_tank1 := PID_Niv_enable and Control_Nivel1 and (accion_motobomba > 0);
164
165         accion_valvula_salida_tank2 := ((PID_Caud_enable or Control_Matlab or ONOFF_Caud_enable) and Control_C
165>>audal1) or (PID_Niv_enable and (Control_Nivel1 or Control_Nivel2) and (accion_motobomba > 0));
166         accion_valvula_entrada_tank2 := ((PID_Caud_enable or Control_Matlab or ONOFF_Caud_enable) and (Control
166>>_Caudal1 or Control_Caudal2 or Control_Caudal3)) or (PID_Niv_enable and (Control_Nivel1 or Control_Nivel2) and
166>> ((accion_itv > 0) or (accion_electrovalvula > 0)));
167
168         accion_valvula_presion_tank3 := (PID_Caud_enable and Control_Caudal3) or (PID_Niv_enable and Control_N
168>>ivel2 and (accion_electrovalvula > 0));
169         accion_valvula_entrada_tank3 := PID_Niv_enable and Control_Nivel2 and (accion_motobomba > 0);
170
171         accion_valvula_salida_motobomba := ((PID_Caud_enable or Control_Matlab or ONOFF_Caud_enable) and Contr
171>>ol_Caudal1) or (PID_Niv_enable and (Control_Nivel1 or Control_Nivel2) and (accion_motobomba > 0));
172 END_IF;
```

Figura 80. Código de control de prácticas. Parte II

Control ON/OFF de caudal

Aquí vemos los elementos en lenguaje LD que se han colocado para dar la funcionalidad al control ON/OFF de caudal.

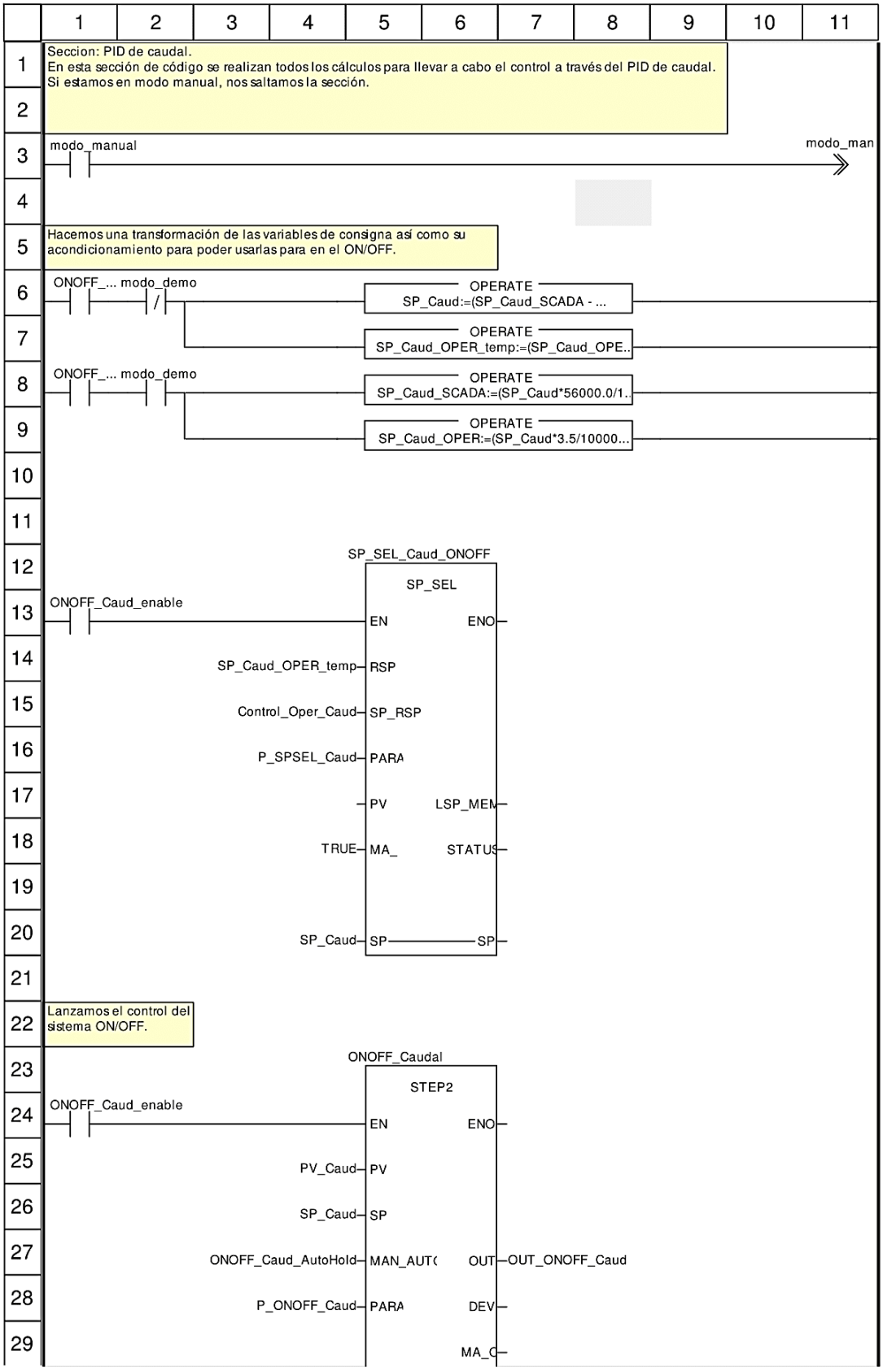


Figura 81. Código del control ON/OFF del caudal. Parte I

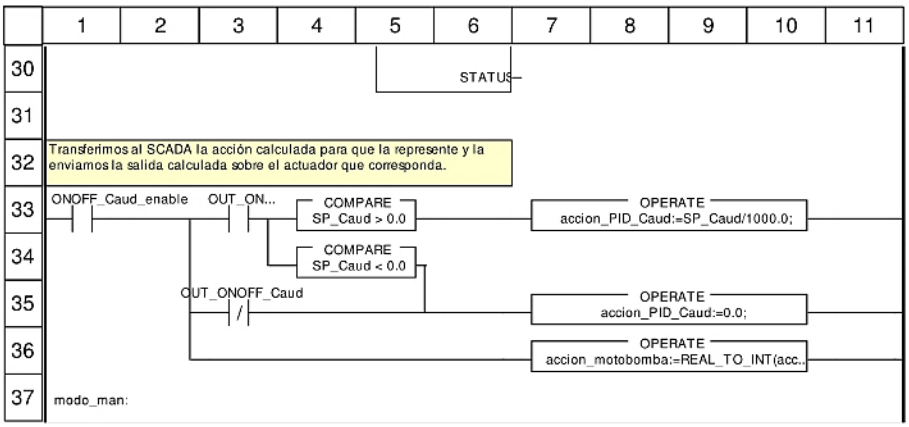


Figura 82. Código del control ON/OFF del caudal. Parte II

PID de caudal

Aquí vemos los elementos en lenguaje LD que se han colocado para dar la funcionalidad al control de caudal tanto por PID como por IMC, así como la gestión de la acción para enviarla al actuador correcto.

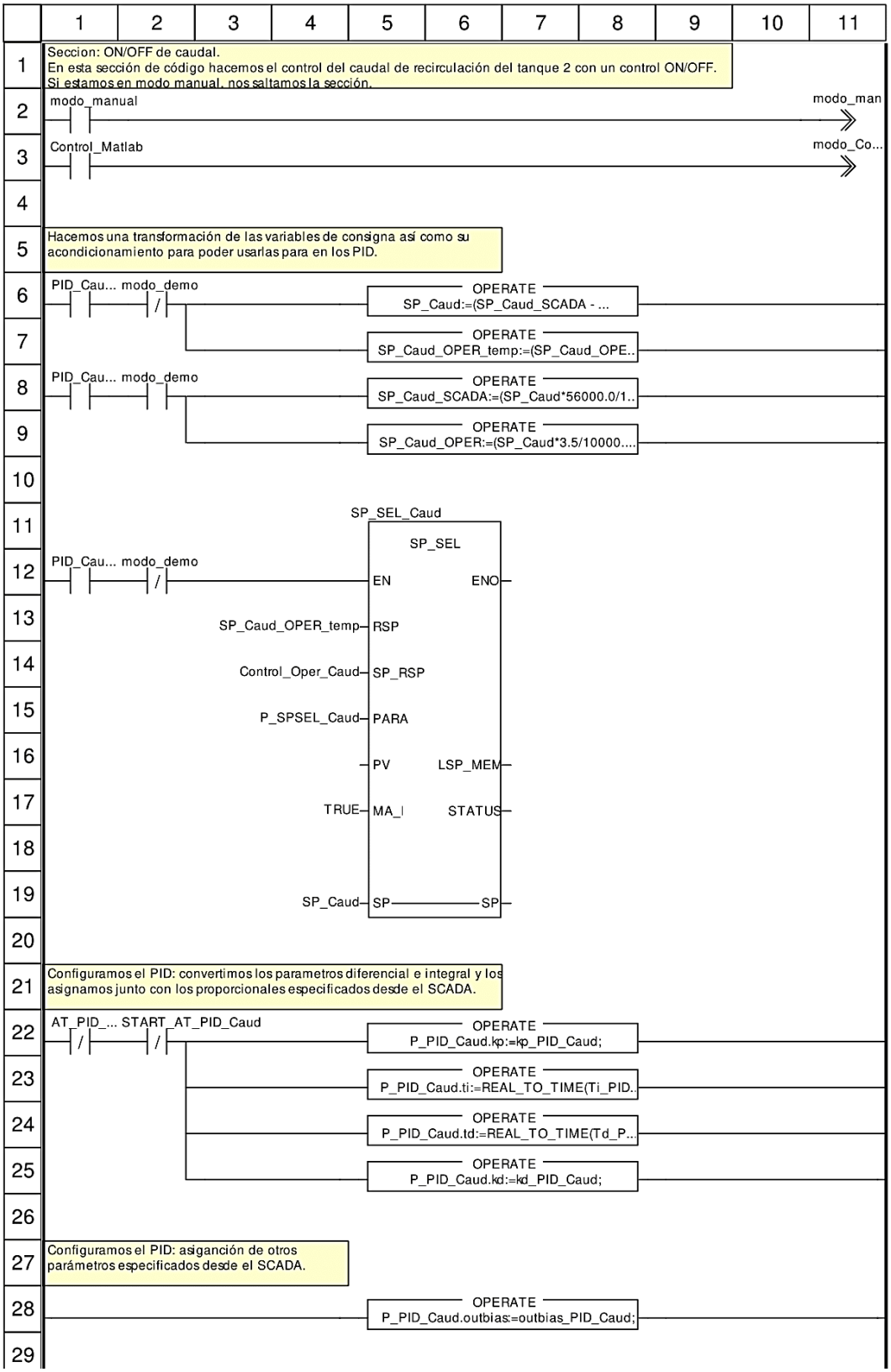


Figura 83. PID Caudal. Parte II

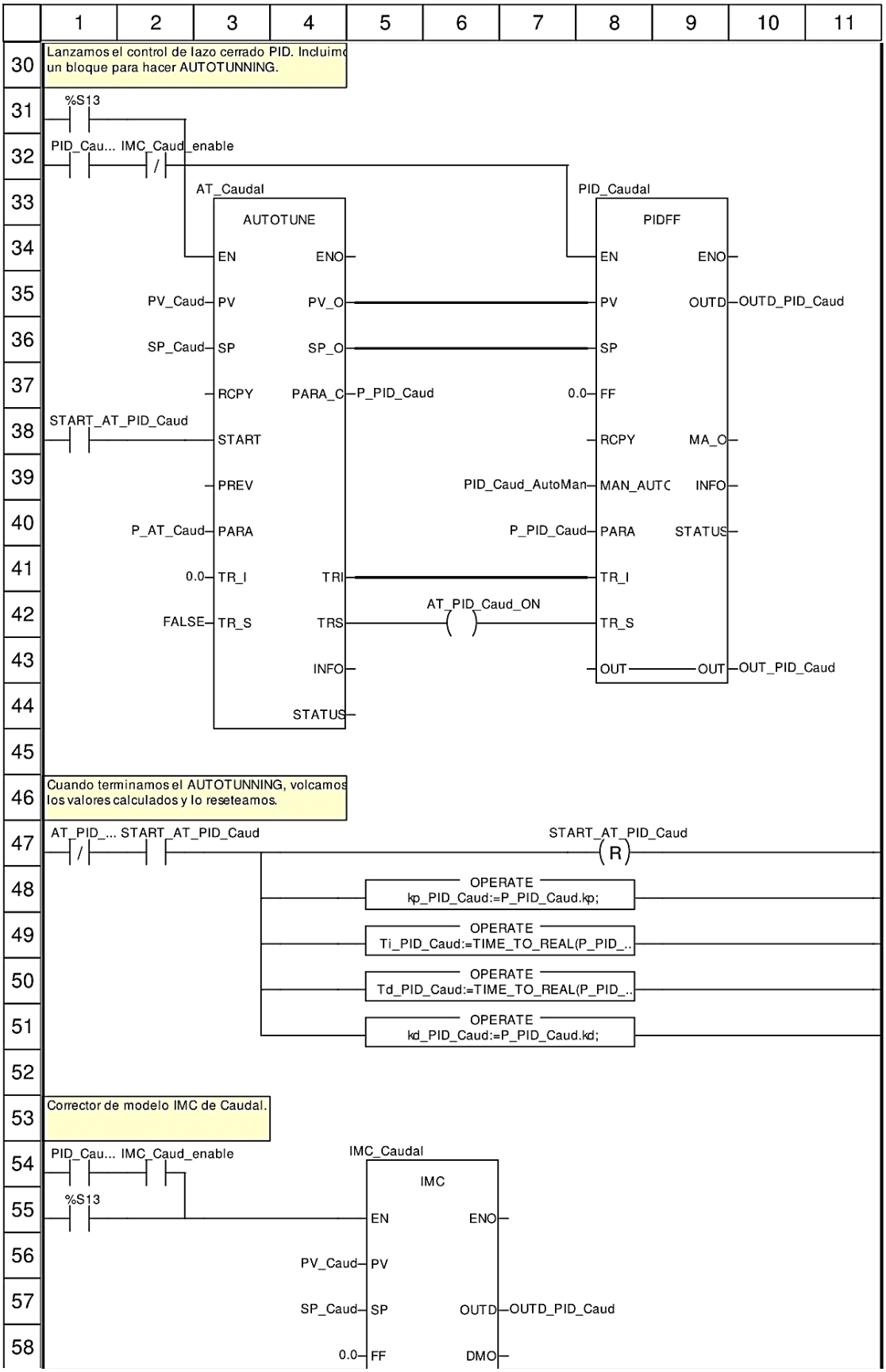


Figura 84. PID Caudal. Parte II

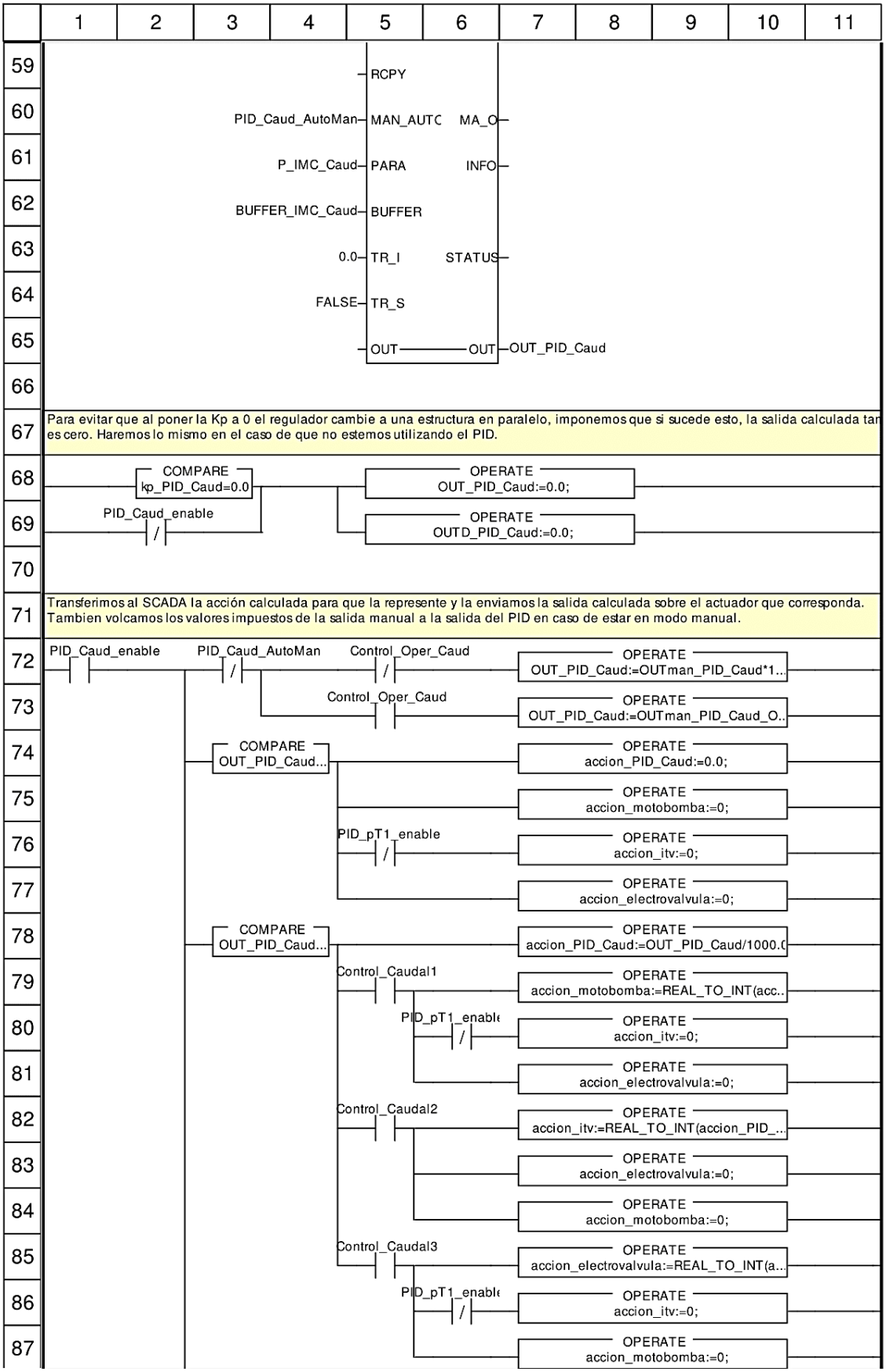


Figura 85. PID Caudal. Parte III

PID de nivel

Aquí vemos los elementos en lenguaje LD que se han colocado para dar la funcionalidad al control de nivel tanto por PID como por IMC, así como la gestión de la acción para enviarla al actuador correcto dependiendo de la modalidad escogida.

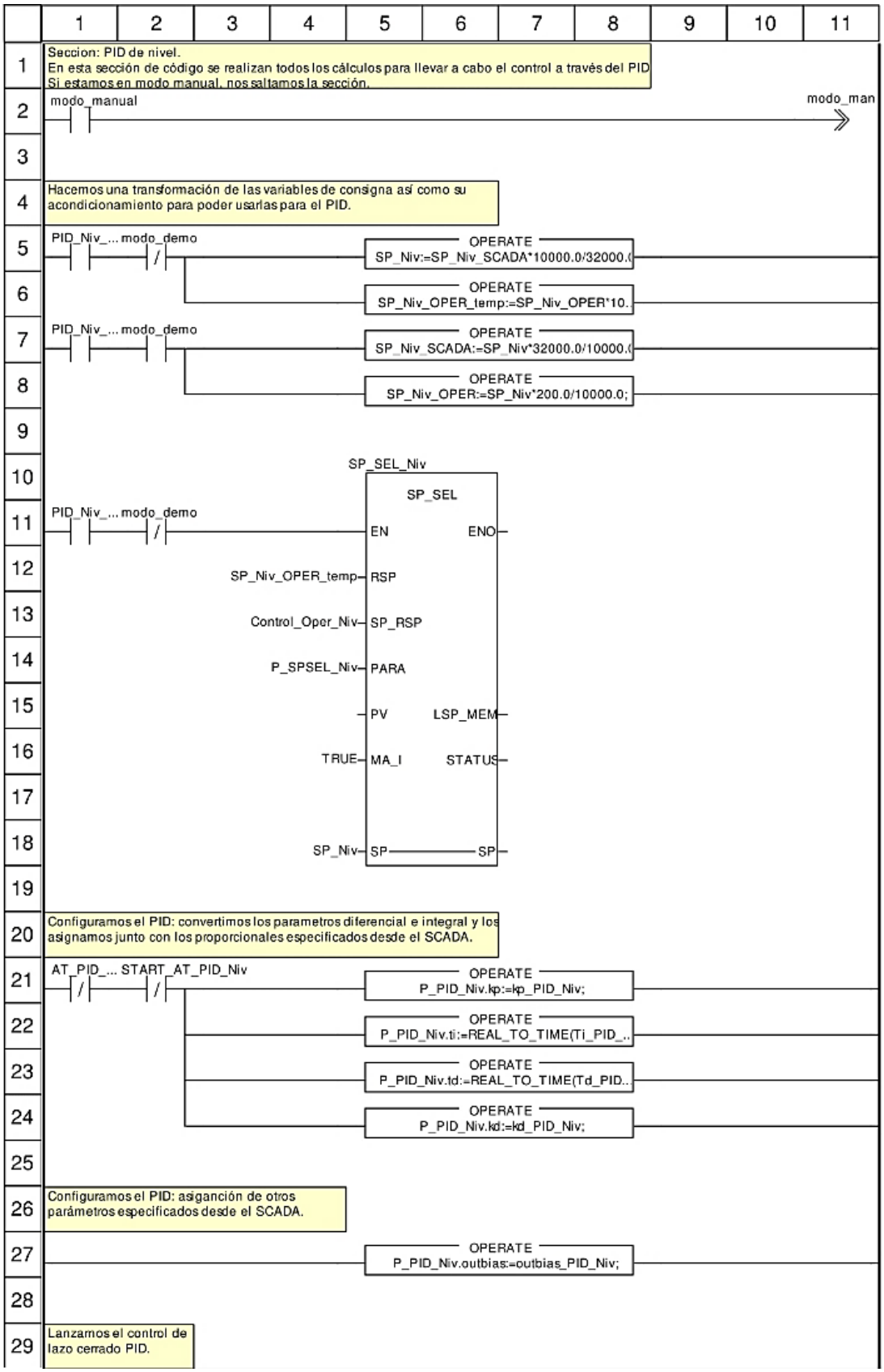


Figura 86. PID Nivel. Parte I

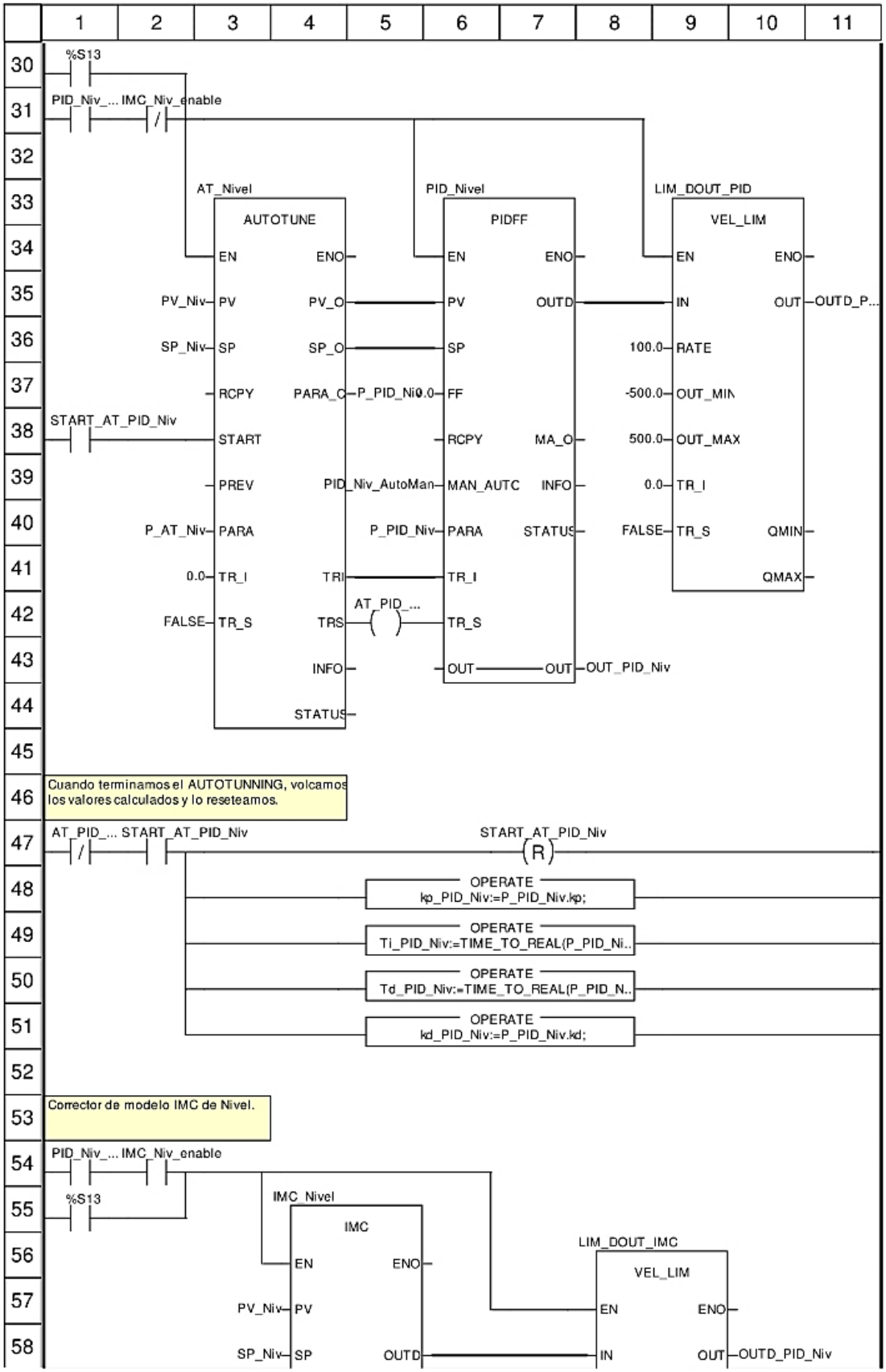


Figura 87. PID Nivel. Parte II

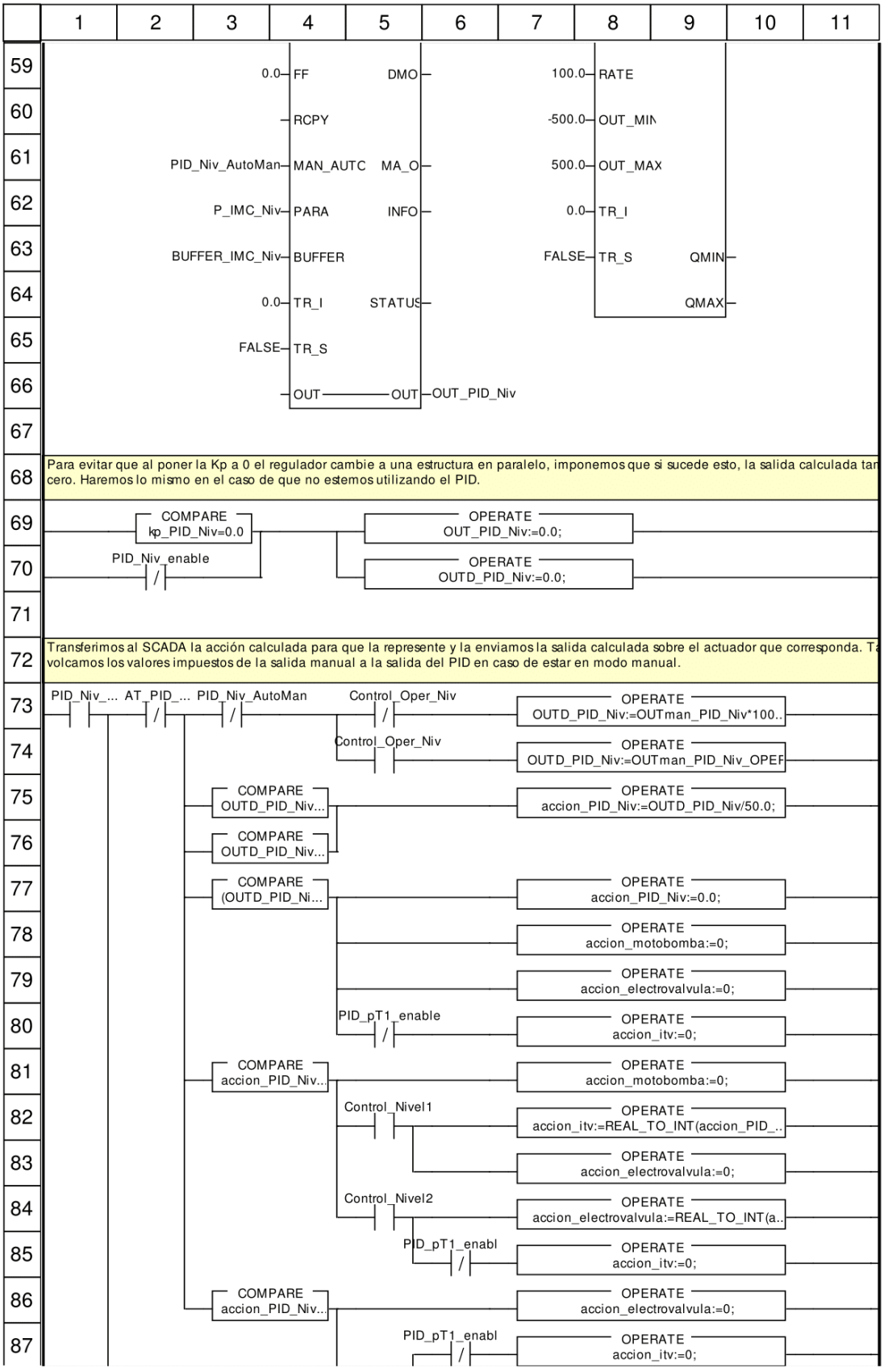


Figura 88. PID Nivel. Parte III

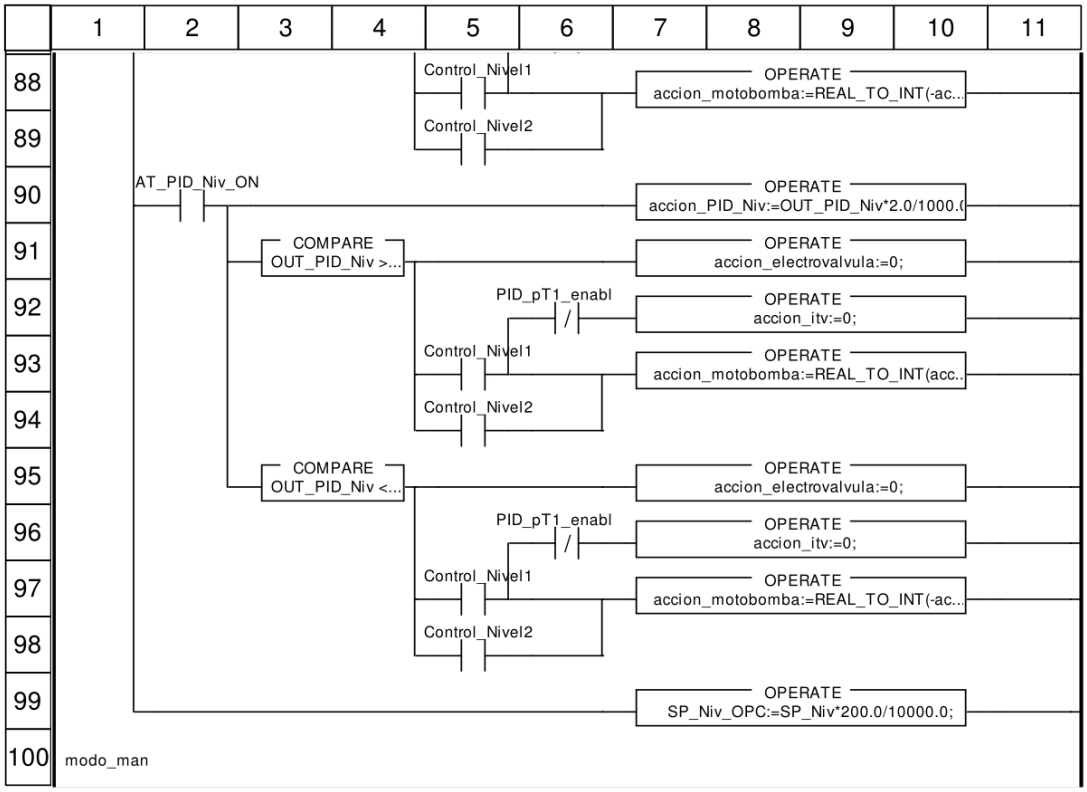


Figura 89. PID Nivel. Parte IV

PID de temperatura

Aquí vemos los elementos en lenguaje LD que se han colocado para dar la funcionalidad al control de temperatura tanto por PID como por IMC.

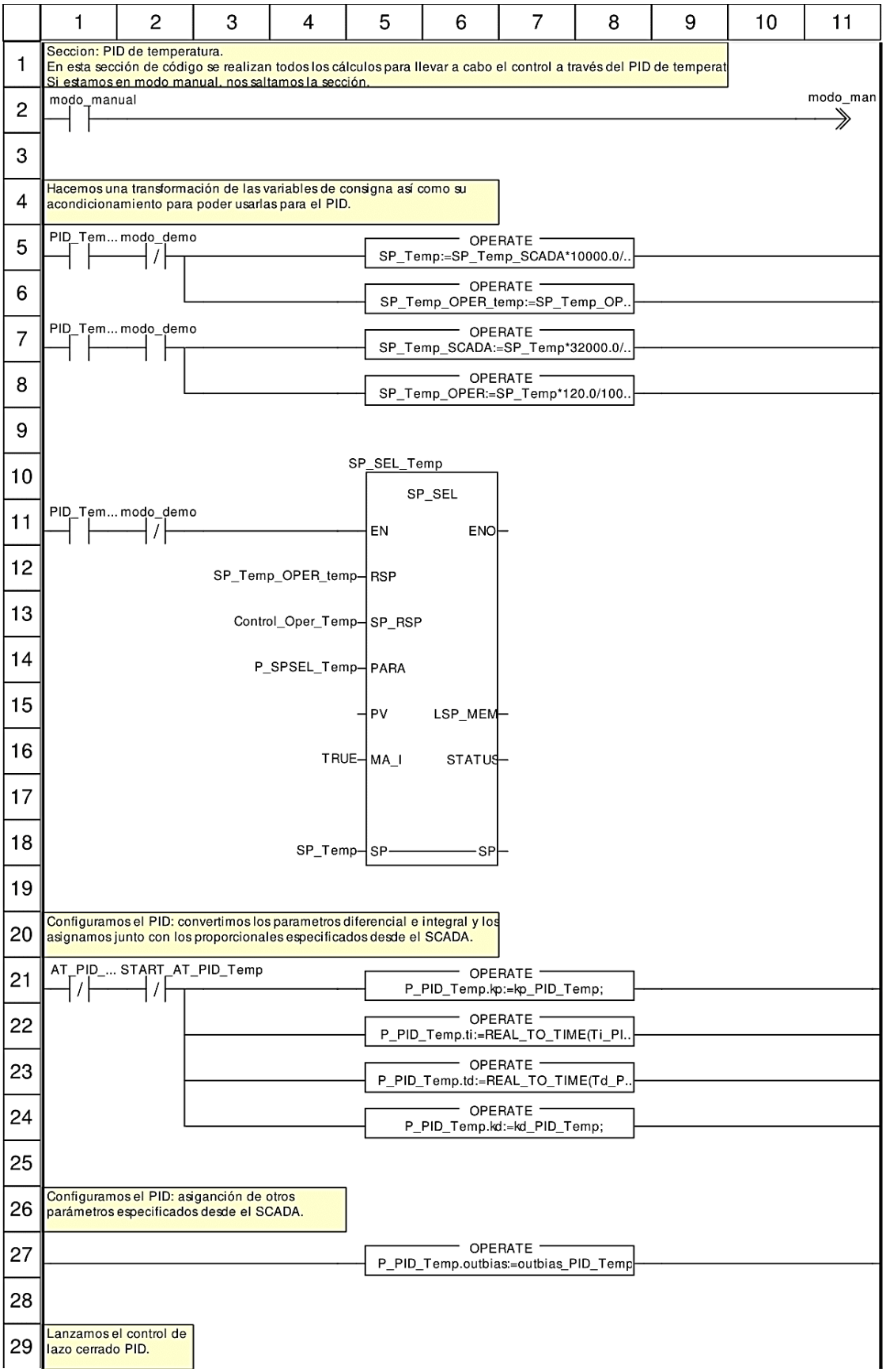


Figura 90. PID de temperatura Parte I

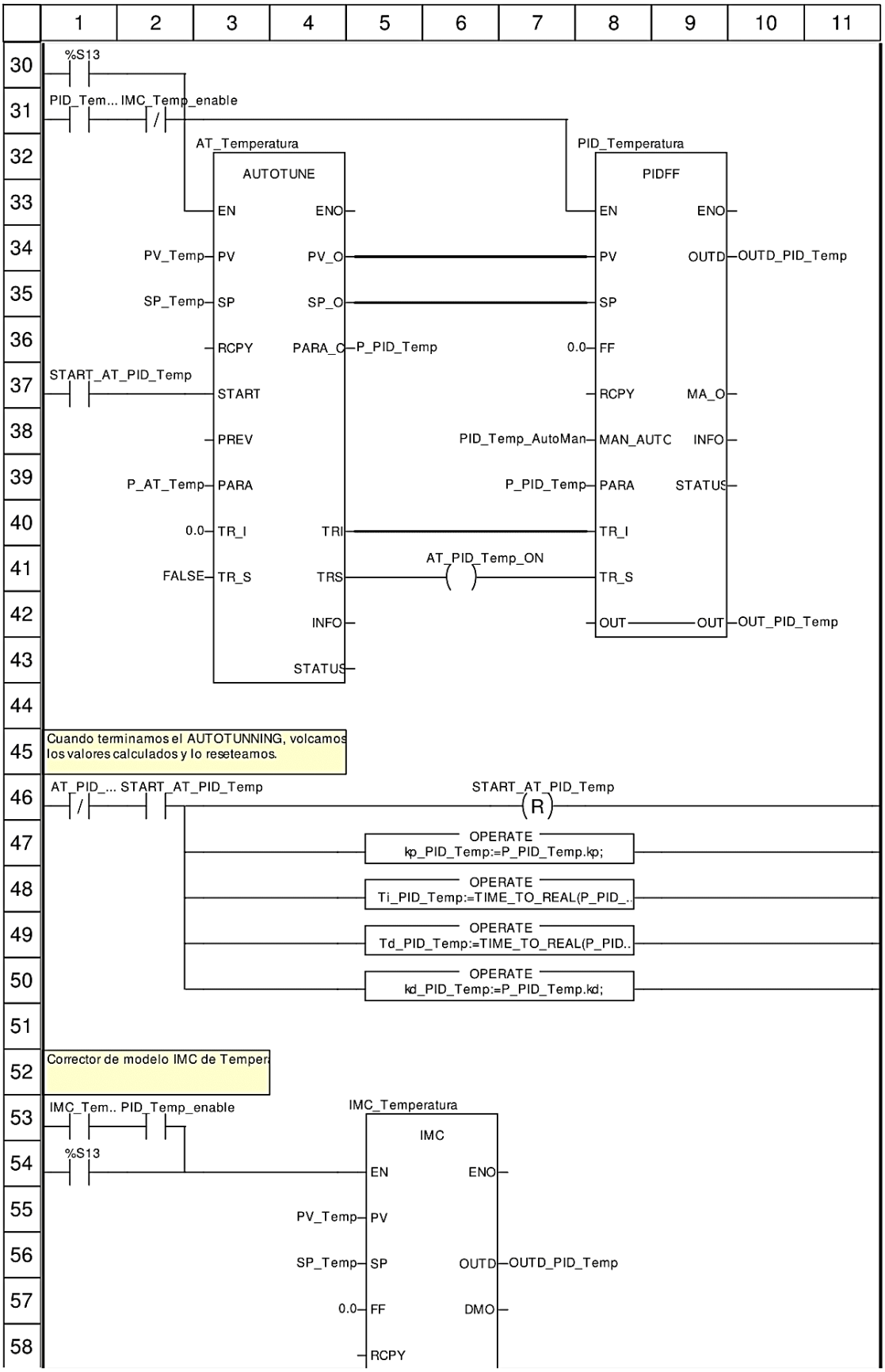


Figura 91. PID de temperatura Parte II

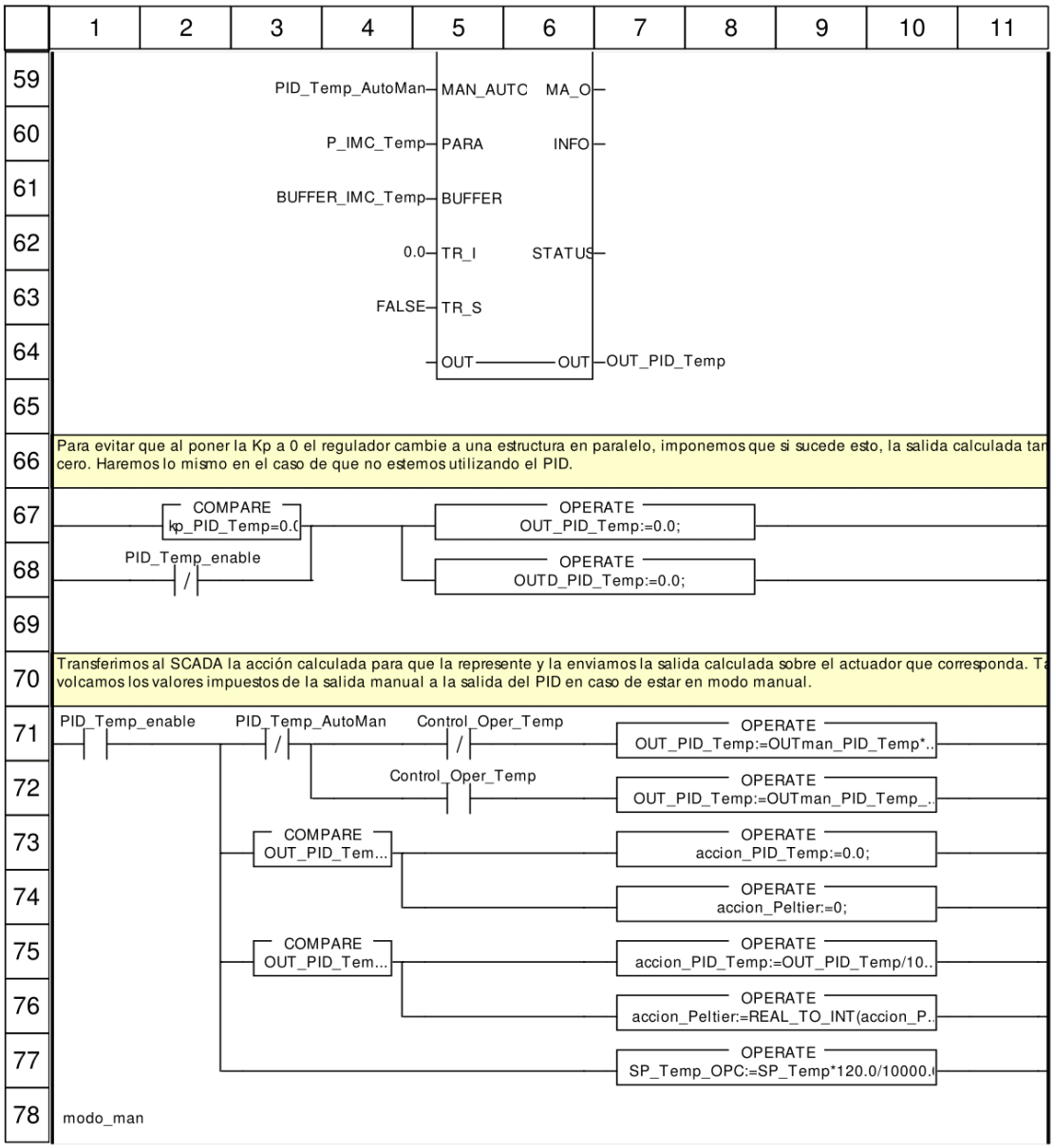


Figura 92. PID de temperatura Parte III

PID de presión en el tanque izquierdo

Aquí vemos los elementos en lenguaje LD que se han colocado para dar la funcionalidad al control de presión del tanque izquierdo. En ese caso no tenemos PID ni IMC, por lo que la consigna se envía directamente al regulador de presión proporcional.

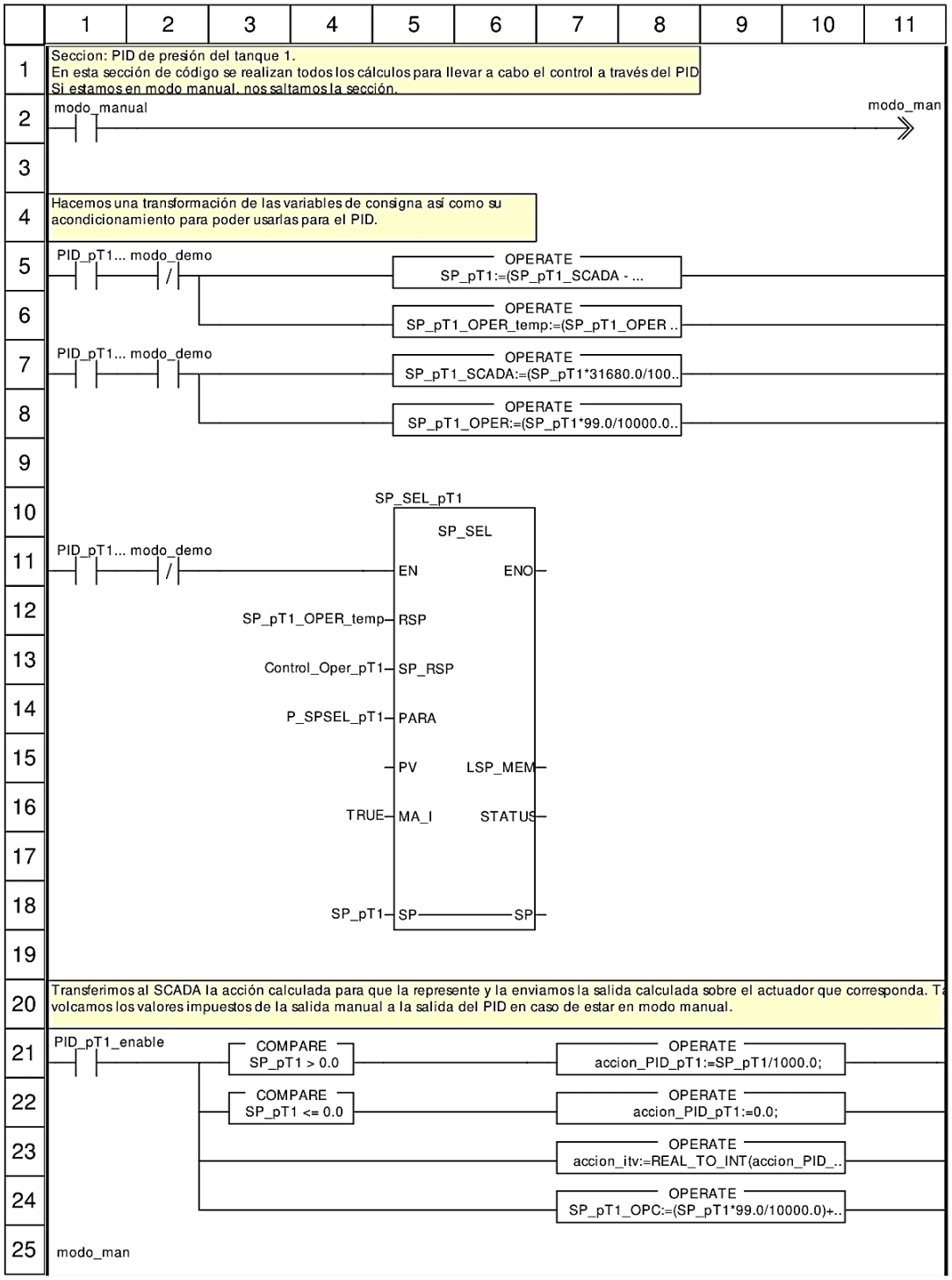


Figura 93. PID de presión en el tanque izdo

SFC de demostración

A continuación, se muestra el SFC de control del modo de demostración automático, junto con el contenido de las macro-etapas creadas para simplificar el diagrama principal, agrupando los estados relacionados con el mismo objetivo.

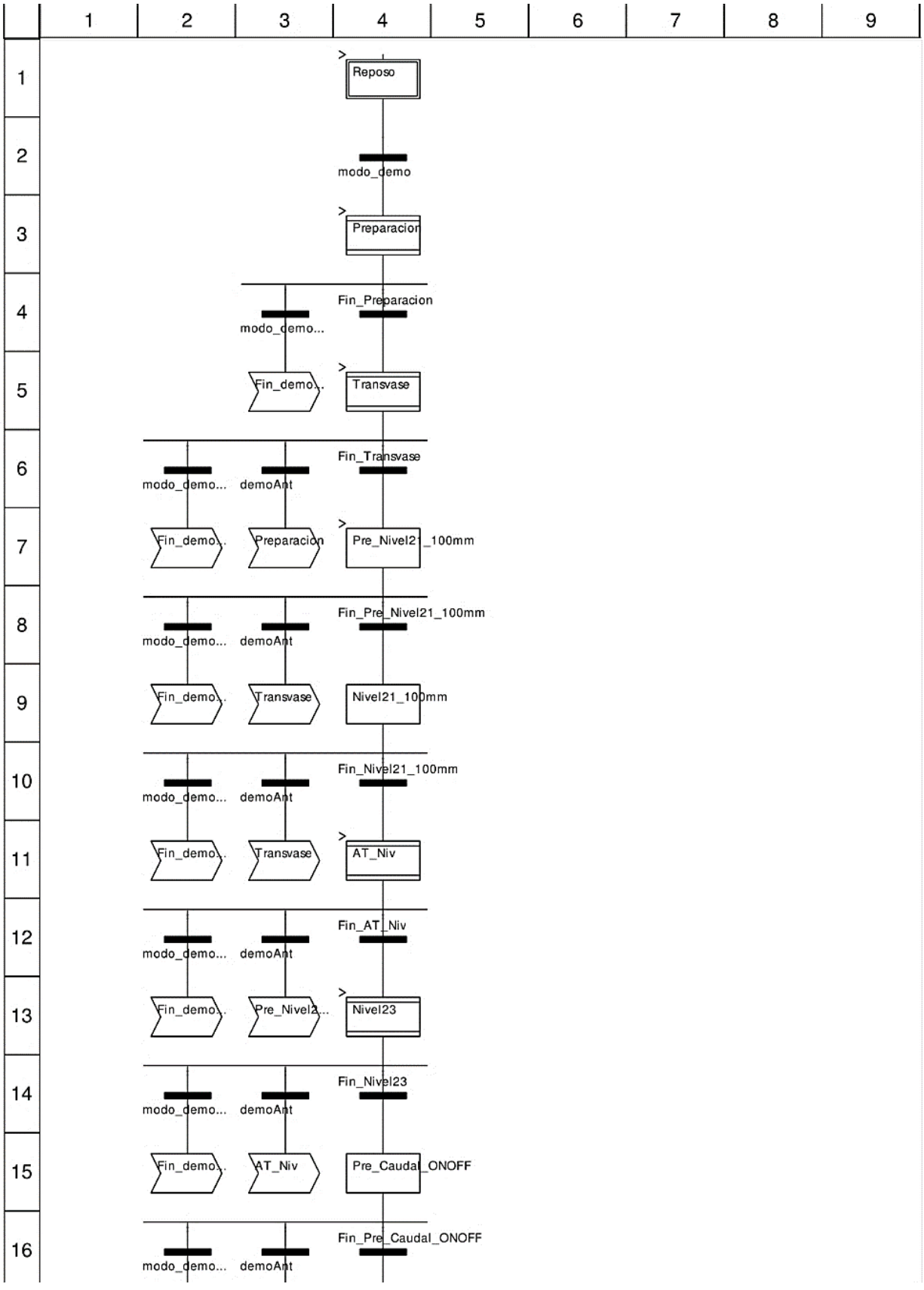


Figura 94. SFC de Demostración. Parte I

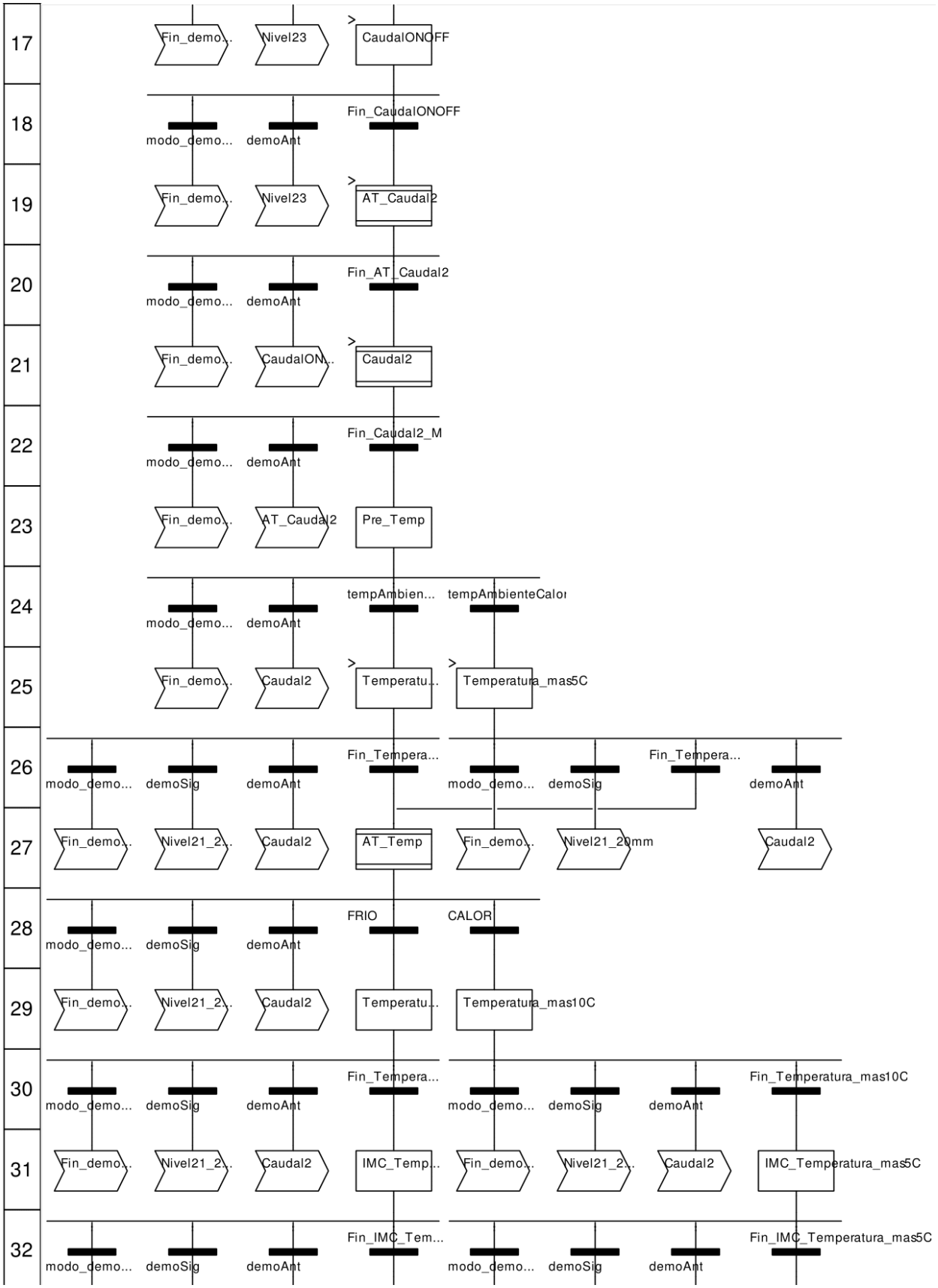


Figura 95. SFC de Demostración. Parte II

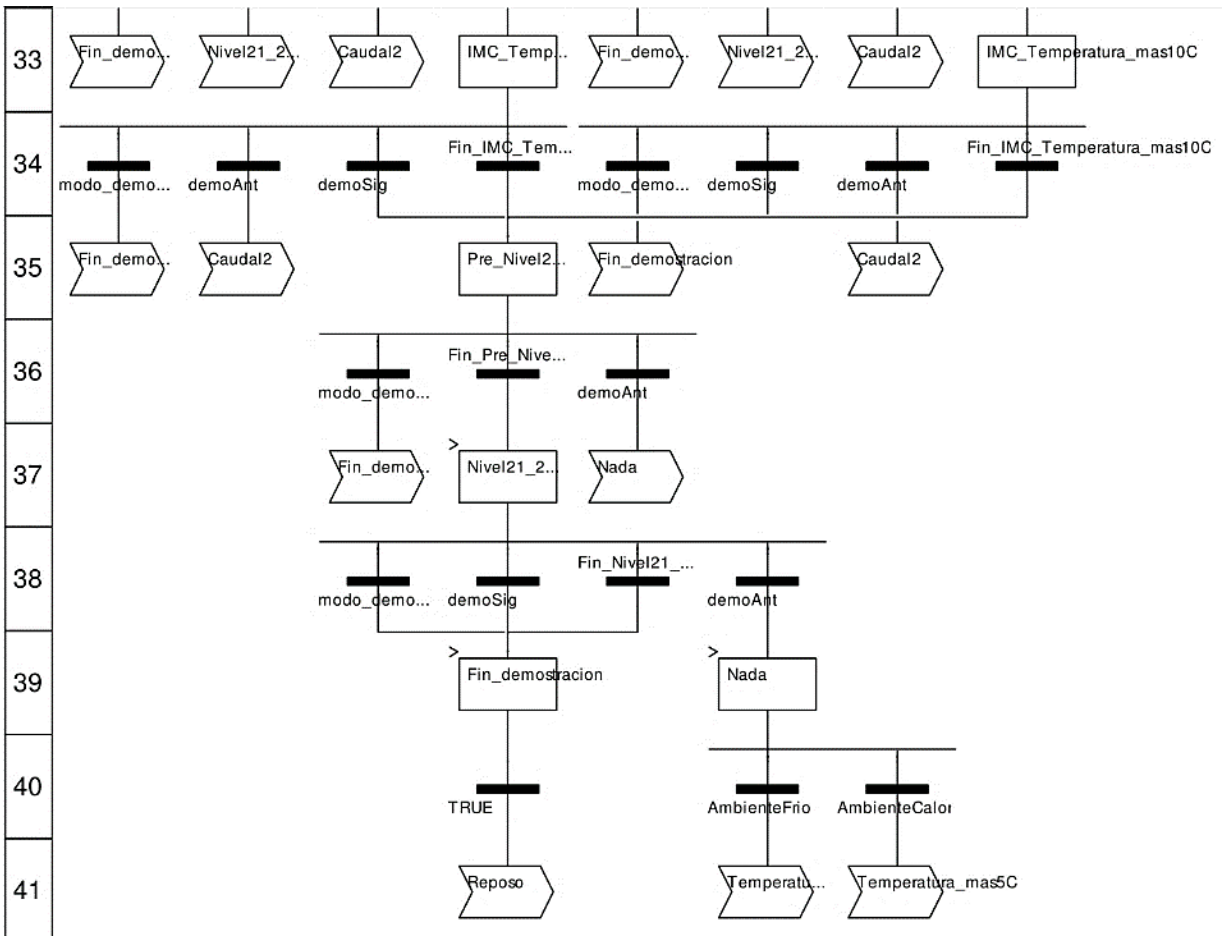


Figura 96. SFC de Demostración. Parte III

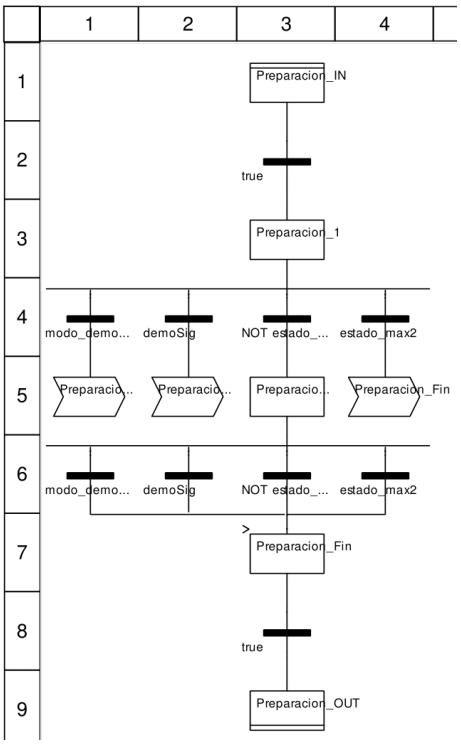


Figura 97. Macro-etapa de Preparación

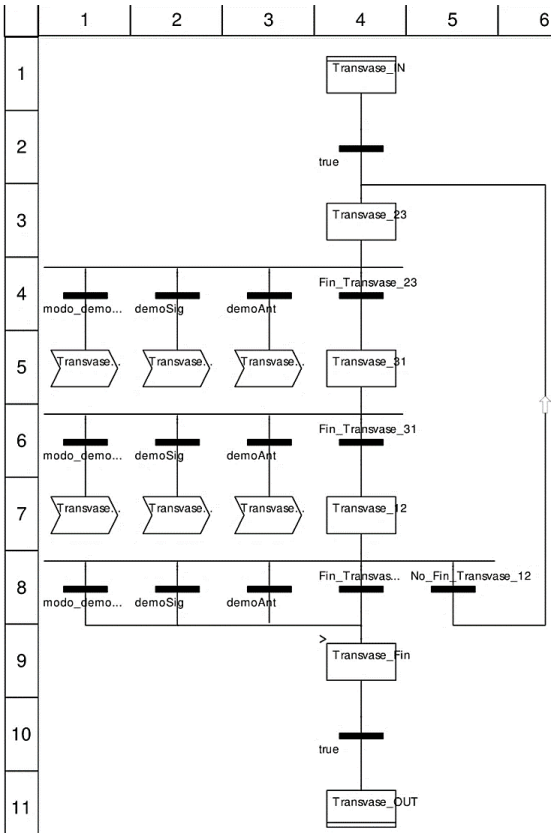


Figura 98. Macro-etapa de Transvase

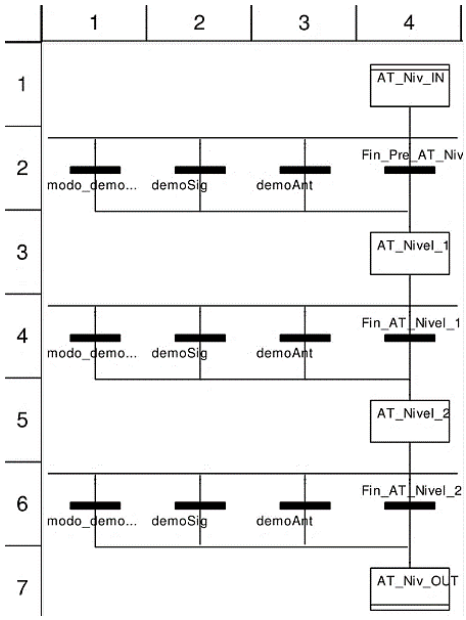


Figura 99. Macro-etapa de Autotuning de Nivel

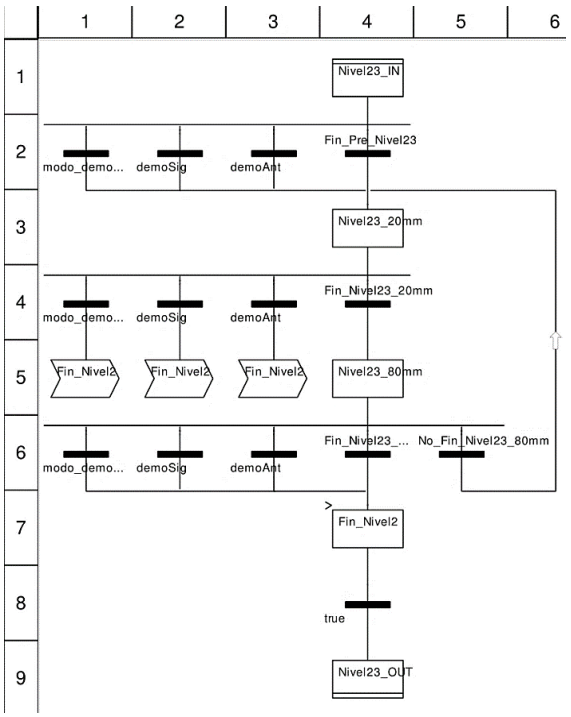


Figura 100. Macro-etapa de Nivel 23

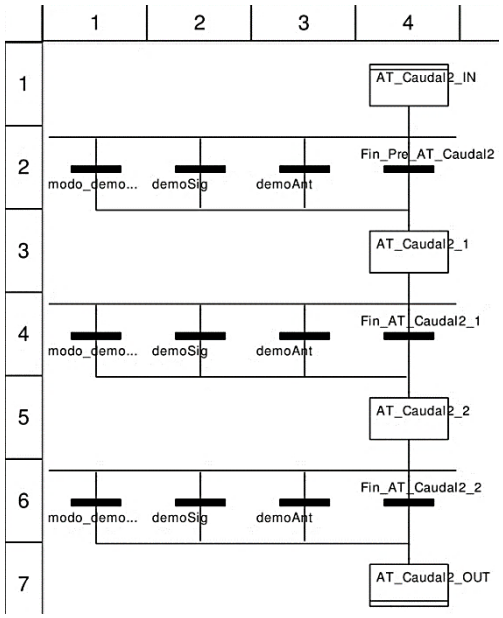


Figura 101. Macro-etapa de Autotuning de Caudal

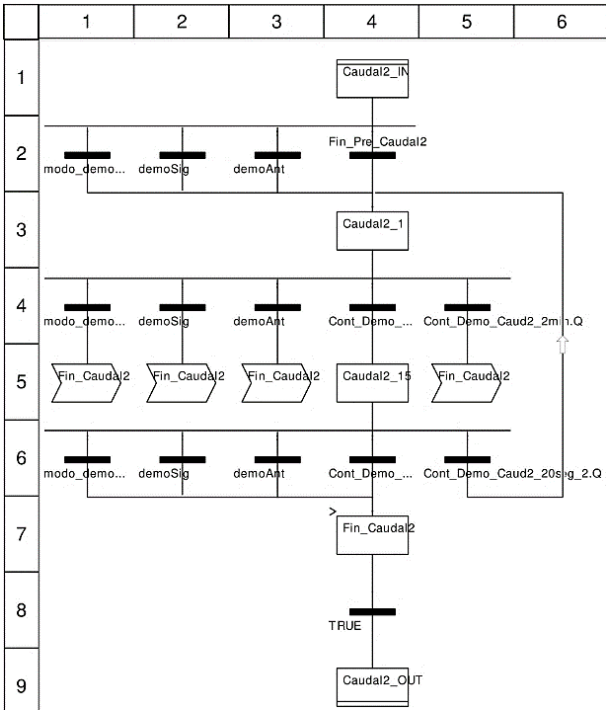


Figura 102. Macro-etapa de Caudal 2

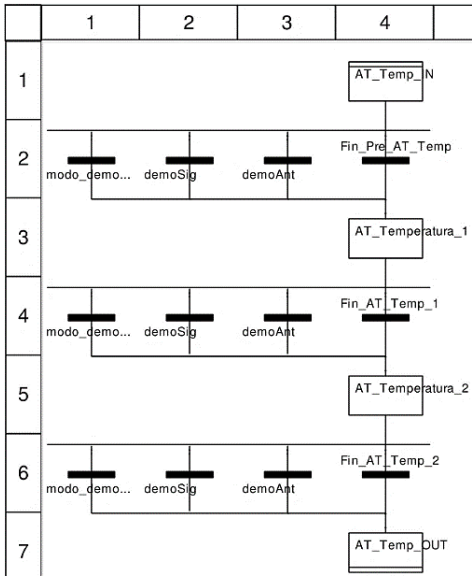


Figura 103. Macro-etapa de Autotuning de Temperatura

Contadores y temporizadores

En este apartado se muestras varios bloques que se ha usado tanto para temporizar algunas etapas de la demo como contadores para sabes los ciclos hechos en otras macro-etapas.

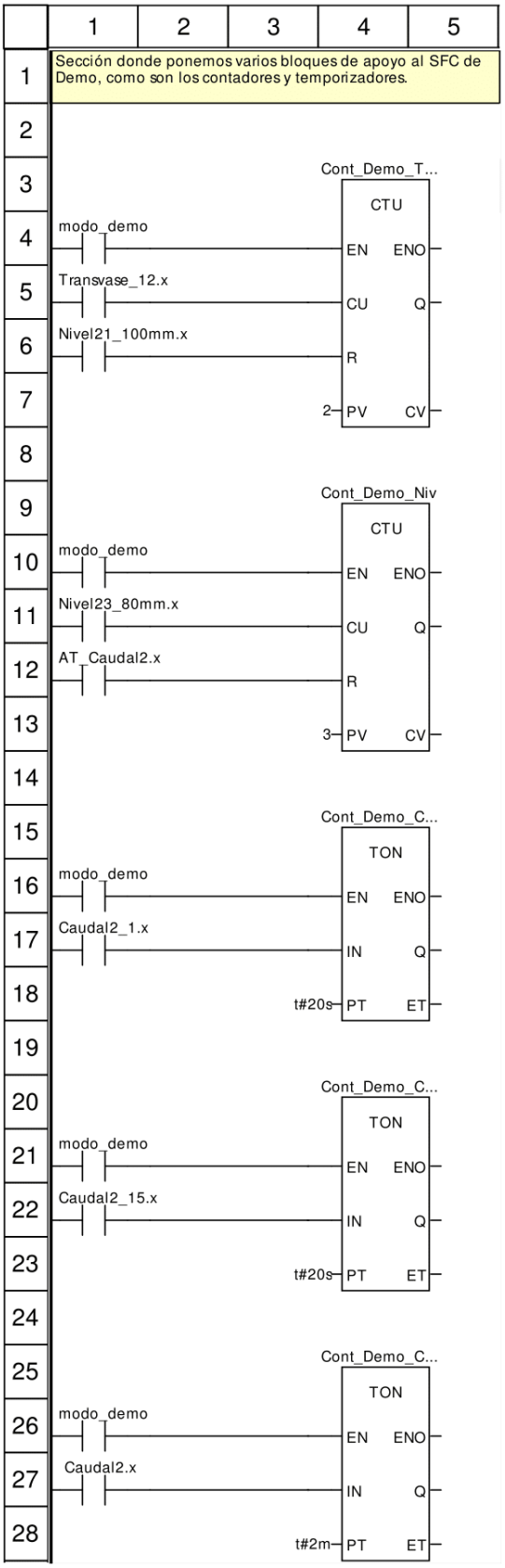


Figura 104. Contadores y temporizadores

Escritura de salidas

En esta sección se muestra el código del autómata que permite escribir sobre los actuadores, ya sea los analógicos (como el regulador de presión proporcional, la electroválvula proporcional, las células Peltier o la motobomba) como para los digitales (los relés que controlan las válvulas monoestables y los que habilitan el agitador, el ventilador, el regulador de presión proporcional, la electroválvula y las Peltier).

```
(*seccion: SALIDAS.
Aqui se realizan operaciones con algunas entradas como por ejemplo protecciones
para no hacer trabajar los actuadores sin líquido, entre otras cosas *)


(* Se lleva hasta la salida analógica la acción que se calcula desde el PID
   o que se ordena manualmente desde el SCADA *)

salida_itv:=accion_itv;
salida_electrovalvula:=accion_electrovalvula;
salida_celulapeltier:=accion_Peltier;
motobomba:=accion_motobomba;

(* Las valvulas de drenaje solo se activan si hay un mínimo de nivel y las de
llenado solo pueden activarse si no se ha alcanzado el máximo *)

valvula_salida_tank1:=accion_valvula_salida_tank1 AND min_tank1;
valvula_entrada_tank1:=accion_valvula_entrada_tank1 AND not(sensor_max_tank1);

valvula_salida_tank2:=accion_valvula_salida_tank2 AND min_tank2;
valvula_entrada_tank2:=accion_valvula_entrada_tank2 AND not(sensor_max_tank2);

valvula_entrada_tank3:=accion_valvula_entrada_tank3 AND not(sensor_max_tank2);

(* La válvula de presión del depósito 3 solo se activa si hay un mínimo de
nivel en el tanque*)

valvula_presion_tank3:=accion_valvula_presion_tank3 AND min_tank3;

(* La válvula del depósito auxiliar se activa directamente sin protecciones
cuando se ordena desde el SCADA. Lo mismo ocurre con la válvula de salida de la
motobomba *)

valvula_salida_tankaux:=accion_valvula_salida_tankaux;
valvula_salida_motobomba:=accion_valvula_salida_motobomba;

(* El agitador del depósito central, el ventilador de la Peltier y el relé que activa
la celula Peltier se activan mientras en PID de temperatura esté funcionando. *)

Agitador := accion_agitador;
ventilador := accion_ventilador;
peltierV := accion_rele_Peltier;

(* El relé que habilita el control desde el PLC o desde los E5CN se controla
desde el SCADA o desde el selector en la maqueta *)

plcreg_s:=accion_rele_control OR auto_man;

(* El relé que habilita la accion sobre la electrovalvula se activa desde el
SCADA o desde el control automático *)

plcreg_0:=accion_rele_electrovalvula;

(* El relé que habilita la accion sobre el transductor se activa desde el SCADA
o desde el control automático *)

plcreg_1:=accion_rele_itv;

(* Cuando no se está ejecutando el control, la baliza parpadea. %S6 es una señal
cuadrada de un segundo para que parpadee *)
```

Figura 105. Código de escritura de las salidas Parte I

```
61
62  baliza := not(running) AND %S6;
63
64  (* La motobomba debe desactivarse por seguridad cuando se alcanzan máximos en
65  los depósitos o cuando en el deposito central no haya líquido, o cuando no
66  estén activadas las válvulas de paso de la bomba para evitar un funcionamiento
67  en seco. Si se está vaciando el depósito 2 y hay una transición indicando que
68  no queda líquido, también debe detenerse*)
69
70  IF (NOT(min_tank2) OR sensor_max_tank1 OR sensor_max_tank2 OR sensor_max_tank3 OR not(valvula_salida_motobomba
70>> AND (valvula_salida_tank2 OR valvula_salida_tankaux))) then
71          motobomba:=0;
72  END_IF;
73
74  (* De manera similar debe desactivarse la salida del transductor *)
75
76  IF (NOT(min_tank1) OR sensor_max_tank2 or sensor_max_tank3) THEN
77          salida_itv:=0;
78  END_IF;
79
80  (* Y también debe desactivarse la accion sobre la electrovalvula *)
81
82  IF (NOT(min_tank3) OR sensor_max_tank1 OR sensor_max_tank2) THEN
83          salida_electrovalvula:=0;
84  END_IF;
85
86  (* Si el control está desactivado, se desactivan las salidas analogicas
87   y se resetan las lecturas de caudal y presion de la itv a la par que
88   se resetea la salida proporcionada por el PID*)
89
90  IF (not(running)) then
91          motobomba:=0;                   (*Salida motobomba *)
92          salida_celulapeltier:=0;        (*Salida celula Peltier *)
93          salida_electrovalvula:=0;       (*Salida electrovalvula *)
94          salida_itv:=0;                  (*Salida del transductor electroneumatico *)
95
96          OUT_PID_pT1:=0.0;
97          OUT_PID_Temp:=0.0;
98          OUT_PID_Niv:=0.0;
99          OUT_PID_Caud:=0.0;
100 END_IF;
```

Figura 106. Código de escritura de las salidas Parte II

Variables

En este punto se muestras las variables que ha sido necesario poner en el programa del autómata.

Nombre	N°	Tipo
AT_Caudal		AUTOTUNE
AT_Caudal_0		AUTOTUNE
AT_Nivel		AUTOTUNE
AT_Temperatura		AUTOTUNE
Cont_Demo_Caud2_2 min		TON
Cont_Demo_Caud2_2 0seg_1		TON
Cont_Demo_Caud2_2 0seg_2		TON
Cont_Demo_Niv		CTU
Cont_Demo_Trans		CTU
IMC_Caudal		IMC
IMC_Caudal_0		IMC
IMC_Nivel		IMC
IMC_Temperatura		IMC
LIM_DOUT_IMC		VEL_LIM
LIM_DOUT_PID		VEL_LIM
ONOFF_Caudal		STEP2
PID_Caudal		PIDFF
PID_Caudal_0		PIDFF
PID_Nivel		PIDFF
PID_Presion_T1		PIDFF
PID_Temperatura		PIDFF
RAMP_Caudal		RAMP
SP_SEL_Caud		SP_SEL
SP_SEL_Caud_ONO FF		SP_SEL
SP_SEL_Niv		SP_SEL
SP_SEL_pT1		SP_SEL
SP_SEL_Temp		SP_SEL
temporizador_vaciado _tanque_1		TON
temporizador_vaciado _tanque_2		TON
temporizador_vaciado _tanque_3		TON

Figura 107. Instancias elementales

Nombre	Tipo
AT_Caudal2	SFCSTEP_STATE
AT_Caudal2_1	SFCSTEP_STATE
AT_Caudal2_2	SFCSTEP_STATE
AT_Caudal2_IN	SFCSTEP_STATE
AT_Caudal2_OUT	SFCSTEP_STATE
AT_Niv	SFCSTEP_STATE
AT_Niv_IN	SFCSTEP_STATE
AT_Niv_OUT	SFCSTEP_STATE
AT_Nivel_1	SFCSTEP_STATE
AT_Nivel_2	SFCSTEP_STATE
AT_Temp	SFCSTEP_STATE
AT_Temp_IN	SFCSTEP_STATE
AT_Temp_OUT	SFCSTEP_STATE
AT_Temperatura_1	SFCSTEP_STATE
AT_Temperatura_2	SFCSTEP_STATE
Caudal2	SFCSTEP_STATE
Caudal2_1	SFCSTEP_STATE
Caudal2_15	SFCSTEP_STATE
Caudal2_IN	SFCSTEP_STATE
Caudal2_OUT	SFCSTEP_STATE
CaudalONOFF	SFCSTEP_STATE
Demostracion	SFCCHART_STATE
Fin_Caudal2	SFCSTEP_STATE
Fin_demostracion	SFCSTEP_STATE
Fin_Nivel2	SFCSTEP_STATE
IMC_Temperatura_25C	SFCSTEP_STATE
IMC_Temperatura_30C	SFCSTEP_STATE
IMC_Temperatura_mas5C	SFCSTEP_STATE
IMC_Temperatura_mas10C	SFCSTEP_STATE
Nada	SFCSTEP_STATE
Nivel21_20mm	SFCSTEP_STATE
Nivel21_100mm	SFCSTEP_STATE
Nivel23	SFCSTEP_STATE
Nivel23_20mm	SFCSTEP_STATE
Nivel23_80mm	SFCSTEP_STATE
Nivel23_IN	SFCSTEP_STATE
Nivel23_OUT	SFCSTEP_STATE
P_AT_Caud	Para_AUTOTUNE
P_AT_Niv	Para_AUTOTUNE
P_AT_Temp	Para_AUTOTUNE
P_IMC_Caud	Para_IMC
P_IMC_Niv	Para_IMC
P_IMC_Temp	Para_IMC
P_ONOFF_Caud	Para_STEP2
P_PID_Caud	Para_PIDFF
P_PID_Niv	Para_PIDFF
P_PID_Temp	Para_PIDFF
P_SPSEL_Caud	Para_SP_SEL
P_SPSEL_Niv	Para_SP_SEL
P_SPSEL_pT1	Para_SP_SEL
P_SPSEL_Temp	Para_SP_SEL
Pre_Caudal_ONOFF	SFCSTEP_STATE
Pre_Nivel21_20mm	SFCSTEP_STATE

Nombre	Tipo
Pre_Nivel21_100mm	SFCSTEP_STATE
Pre_Temp	SFCSTEP_STATE
Preparacion	SFCSTEP_STATE
Preparacion_1	SFCSTEP_STATE
Preparacion_2	SFCSTEP_STATE
Preparacion_Fin	SFCSTEP_STATE
Preparacion_IN	SFCSTEP_STATE
Preparacion_OUT	SFCSTEP_STATE
Reposo	SFCSTEP_STATE
Temperatura_25C	SFCSTEP_STATE
Temperatura_30C	SFCSTEP_STATE
Temperatura_mas5C	SFCSTEP_STATE
Temperatura_mas10C	SFCSTEP_STATE
Transvase	SFCSTEP_STATE
Transvase_12	SFCSTEP_STATE
Transvase_23	SFCSTEP_STATE
Transvase_31	SFCSTEP_STATE
Transvase_Fin	SFCSTEP_STATE
Transvase_IN	SFCSTEP_STATE
Transvase_OUT	SFCSTEP_STATE

Figura 108. Variables derivadas

Nombre	Tipo	Dirección	Valor	Comentario
AmbienteCalor	EBOOL			
AmbienteFrio	EBOOL			
AT_IMC_Caud_ON	EBOOL			
BUFFER_IMC_Caud	ARRAY[1..10] OF REAL			
BUFFER_IMC_Niv	ARRAY[1..10] OF REAL			
BUFFER_IMC_Temp	ARRAY[1..50] OF REAL			
CALOR	BOOL			
Fin_AT_Caudal2	BOOL			
Fin_AT_Caudal2_1	BOOL			
Fin_AT_Caudal2_2	BOOL			
Fin_AT_Niv	BOOL			
Fin_AT_Nivel_1	BOOL			
Fin_AT_Nivel_2	BOOL			
Fin_AT_Temp_1	BOOL			
Fin_AT_Temp_2	BOOL			
Fin_Caudal2_15	BOOL			
Fin_Caudal2_M	BOOL			
Fin_CaudalONOFF	BOOL			
Fin_IMC_Temperatura_25C	BOOL			
Fin_IMC_Temperatura_30C	BOOL			
Fin_IMC_Temperatura_mas5C	BOOL			
Fin_IMC_Temperatura_mas10C	BOOL			
Fin_Nivel21_20mm	BOOL			
Fin_Nivel21_100mm	BOOL			
Fin_Nivel23	BOOL			
Fin_Nivel23_20mm	BOOL			
Fin_Nivel23_80mm	BOOL			
Fin_Pre_AT_Caudal2	BOOL			
Fin_Pre_AT_Niv	BOOL			
Fin_Pre_AT_Temp	BOOL			
Fin_Pre_Caudal2	BOOL			
Fin_Pre_Caudal_ONOFF	BOOL			
Fin_Pre_Nivel21_20mm	BOOL			
Fin_Pre_Nivel21_100mm	BOOL			
Fin_Pre_Nivel23	BOOL			
Fin_Preparacion	BOOL			
Fin_Temperatura_25C	BOOL			
Fin_Temperatura_30C	BOOL			
Fin_Temperatura_mas5C	BOOL			
Fin_Temperatura_mas10C	BOOL			
Fin_Transvase	BOOL			
Fin_Transvase_12	BOOL			
Fin_Transvase_23	BOOL			
Fin_Transvase_31	BOOL			

Figura 109. Variables elementales. Parte I

Nombre	Tipo	Dirección	Valor	Comentario
FRIO	BOOL			
kd_PID_Caud	REAL	%MW280	1.0	
kd_PID_Niv	REAL	%MW284	1.0	
kd_PID_Temp	REAL	%MW296	1.0	
kp_PID_Caud	REAL	%MW220		
kp_PID_Niv	REAL	%MW224		
kp_PID_Temp	REAL	%MW236		
No_Fin_Nivel23_80mm	BOOL			
No_Fin_Transvase_12	BOOL			
OUT_ONOFF_Caud	EBOOL			
SP_Caud_OPER_temp	REAL			
SP_Niv_OPER_temp	REAL			
SP_pT1_OPER_temp	REAL			
SP_Temp_OPER_temp	REAL			
START_AT_PID_Caud_memo	EBOOL			
tempAmbienteCalor	BOOL			
tempAmbienteFrio	BOOL			
tiempoPre	TIME		t#0s	
modo_demo_abort	EBOOL	%M53	FALSE	Aborta la demostracion.
accion_PID_pT1	REAL	%MW208		Accion calculada por el controlador PID de la presión del tanque 1
accion_PID_Temp	REAL	%MW216		Accion calculada por el controlador PID de la temperatura en el tanque 2
accion_PID_Caud	REAL	%MW200		Accion calculada por el controlador PID del caudal que sale del tanque 2
accion_PID_Niv	REAL	%MW204		Accion calculada por el controlador PID del nivel del tanque 2
accion_itv	INT	%MW9		Accion para el transductor
accion_Peltier	INT	%MW11		Accion para la celula Peltier
accion_electrovalvula	INT	%MW10		Accion para la electrovalvula proporcional
accion_motobomba	INT	%MW12		Accion para la motobomba
modo_demo	EBOOL	%M52	FALSE	Activa el modo demostracion.
alarma	EBOOL	%M50		Alarma en forma de onda cuadrada.
auto_man	EBOOL	%I0.2.2		automatico/manual
AT_PID_Caud_ON	EBOOL	%M125	FALSE	AUTOTUNE de caudal ON
AT_PID_Niv_ON	EBOOL	%M126	FALSE	AUTOTUNE de nivel ON
AT_PID_Temp_ON	EBOOL	%M129	FALSE	AUTOTUNE de temperatura ON
START_AT_PID_Caud	EBOOL	%M120	FALSE	Comienzo del AUTOTUNE de caudal

Figura 110. Variables elementales. Parte II

Nombre	Tipo	Dirección	Valor	Comentario
START_AT_PID_Niv	EBOOL	%M121	FALSE	Comienzo del AUTOTUNE de nivel
START_AT_PID_Temp	EBOOL	%M124	FALSE	Comienzo del AUTOTUNE de temperatura
SP_Caud	REAL			Consigna de caudal para el PID
salida_itv	INT	%QW0.6.0		consigna de la ITV
SP_Niv	REAL			Consigna de nivel para el PID
SP_pT1	REAL			Consigna de presión del tanque 1 para el PID
SP_Temp	REAL			Consigna de temperatura para el PID
const_Caud	REAL	%MW50		Constante de adaptación
const_Niv	REAL	%MW54		Constante de adaptación
const_pT1	REAL	%MW58		Constante de adaptación
const_Temp	REAL	%MW66		Constante de adaptación
Control_Caudal1	EBOOL	%M113		Control de caudal 1
Control_Caudal1_SCADA	EBOOL	%M143	FALSE	Control de caudal 1 desde SCADA
Control_Caudal2	EBOOL	%M114		Control de caudal 2
Control_Caudal2_SCADA	EBOOL	%M144	FALSE	Control de caudal 2 desde SCADA
Control_Caudal3	EBOOL	%M115		Control de caudal 3
Control_Caudal3_SCADA	EBOOL	%M145	FALSE	Control de caudal 3 desde SCADA
Control_Nivel1	EBOOL	%M110		Control de nivel 1
Control_Nivel1_SCADA	EBOOL	%M140	FALSE	Control de nivel 1 desde SCADA
Control_Nivel2	EBOOL	%M111		Control de nivel 2
Control_Nivel2_SCADA	EBOOL	%M141	FALSE	Control de nivel 2 desde SCADA
Control_Oper_Niv	EBOOL	%M151	FALSE	Control desde autómata de nivel
Control_Oper_pT1	EBOOL	%M152	FALSE	Control desde autómata de presión
Control_Oper_Temp	EBOOL	%M153	FALSE	Control desde autómata de temperatura
Control_Oper_Caud	EBOOL	%M150	FALSE	Control desde autómatal de cauda
plcreg_s	EBOOL	%Q0.3.9		control desde PLC o E5CN (R6&R7&R8&R9)
ONOFF_Caud_enable	EBOOL	%M98		Control ONOFF caudal enable
ONOFF_Caud_enable_SCADA	EBOOL	%M97	FALSE	Control ONOFF caudal enable desde SCADA
medida_caudal_1	EBOOL	%I0.2.14		display de caudalimetro (salida 1)
medida_caudal_2	EBOOL	%I0.2.15		display de caudalimetro (salida 2)
medida_nivel_1	EBOOL	%I0.2.12		display de nivel (salida 1)
medida_nivel_2	EBOOL	%I0.2.13		display de nivel (salida 2)
Td_PID_Caud	REAL	%MW260		En segundos.
Td_PID_Niv	REAL	%MW264		En segundos.
Td_PID_Temp	REAL	%MW276		En segundos.
Ti_PID_Caud	REAL	%MW240		En segundos.

Figura 111. Variables elementales. Parte III

Nombre	Tipo	Dirección	Valor	Comentario
Ti_PID_Niv	REAL	%MW244		En segundos.
Ti_PID_Temp	REAL	%MW256		En segundos.
entrada_caudal	INT	%IW0.4.1		entrada analogica de caudal
entrada_nivel	INT	%IW0.4.2		entrada analogica de nivel
entrada_presostato	INT	%IW0.5.0		entrada analogica de presostato digital
entrada_temperatura	INT	%IW0.4.0		entrada analogica de temperatura
entrada_transductor	INT	%IW0.4.3		entrada analogica de transductor electroneumatico ITV
sensor_min_tank1	EBOOL	%I0.2.5		entrada sensor limite de nivel minimo en tanque 1
sensor_min_tank2	EBOOL	%I0.2.7		entrada sensor limite de nivel minimo en tanque 2
sensor_min_tank3	EBOOL	%I0.2.9		entrada sensor limite de nivel minimo en tanque 3
estado_valvula_salida_tank1	EBOOL	%M33		Estado de la valvula A1 para el SCADA
estado_valvula_salida_tank2	EBOOL	%M35		Estado de la valvula A2 para el SCADA
estado_valvula_salida_tankaux	EBOOL	%M38		Estado de la valvula A4 para el SCADA
estado_valvula_salida_motobomba	EBOOL	%M39		Estado de la valvula A5 para el SCADA
estado_valvula_entrada_tank1	EBOOL	%M34		Estado de la valvula B1 para el SCADA
estado_valvula_entrada_tank2	EBOOL	%M36		Estado de la valvula B2 para el SCADA
estado_valvula_entrada_tank3	EBOOL	%M37		Estado de la valvula B3 para el SCADA
estado_plcreg	EBOOL	%M32		Estado del rele plcreg para el SCADA
estado_max1	EBOOL	%M29		Estado del sensor max1 para el SCADA
estado_max2	EBOOL	%M30		Estado del sensor max2 para el SCADA
estado_max3	EBOOL	%M31		Estado del sensor max3 para el SCADA
estado_min1	EBOOL	%M26		Estado del sensor min1 para el SCADA
estado_min2	EBOOL	%M27		Estado del sensor min2 para el SCADA
estado_min3	EBOOL	%M28		Estado del sensor min3 para el SCADA
IMC_Caud_enable	EBOOL	%M160	FALSE	IMC caudal enable
IMC_Caud_enable_SCADA	EBOOL	%M155	FALSE	IMC caudal enable desde SCADA
IMC_Niv_enable	EBOOL	%M161	FALSE	IMC nivel enable
IMC_Niv_enable_SCADA	EBOOL	%M156	FALSE	IMC nivel enable desde SCADA
IMC_Temp_enable	EBOOL	%M162	FALSE	IMC temperatura enable
IMC_Temp_enable_SCADA	EBOOL	%M157	FALSE	IMC temperatura enable desde SCADA
OUTD_PID_Caud	REAL	%MW300		Incremento de salida del PID de caudal
OUTD_PID_Niv	REAL	%MW304		Incremento de salida del PID de nivel

Figura 112. Variables elementales. Parte IV

Nombre	Tipo	Dirección	Valor	Comentario
OUTD_PID_Temp	REAL	%MW316		Incremento de salida del PID de temperatura
peltierV	EBOOL	%Q0.3.8		inversion de polaridad celulas peltier (R2 & R3)
plcreg_2	EBOOL	%Q0.3.14		lazos caudal y nivel (R11 & R12)
PV_Temp_SCADA	REAL	%MW176		Lectura del sensor de temperatura
PV_Caud	REAL			Lectura tomada del sensor de caudal
PV_Caud_SCADA	REAL	%MW160		Lectura tomada del sensor de caudal
PV_Niv	REAL			Lectura tomada del sensor de nivel en tanque 2
PV_Niv_SCADA	REAL	%MW164		Lectura tomada del sensor de nivel en tanque 2
PV_pT1	REAL			Lectura tomada del sensor de presion en tanque 1
PV_pT1_SCADA	REAL	%MW168		Lectura tomada del sensor de presion en tanque 1
PV_PT3	REAL			Lectura tomada por el presostato en el tanque 3
PV_pT3_SCADA	REAL	%MW172		Lectura tomada por el presostato en el tanque 3
PV_Temp	REAL			Lectura tomada por el sensor de temperatura
sensor_max_tank1	EBOOL	%I0.2.4		limite de nivel maximo en tanque 1
sensor_max_tank2	EBOOL	%I0.2.6		limite de nivel maximo en tanque 2
sensor_max_tank3	EBOOL	%I0.2.8		limite de nivel maximo en tanque 3
baliza	EBOOL	%Q0.3.7		luz de alarma de la baliza
Agitador	EBOOL	%Q0.3.0		mezclador (agitador)
ONOFF_Caud_AutoHold	EBOOL	%M99	TRUE	Modo de control ONOFF caudal: TRUE => Automático; FALSE => Hold.
PID_Caud_AutoMan	EBOOL	%M105	TRUE	Modo de control PID caudal: TRUE => Automático; FALSE => Manual.
PID_Niv_AutoMan	EBOOL	%M106	TRUE	Modo de control PID nivel: TRUE => Automático; FALSE => Manual.
PID_Temp_AutoMan	EBOOL	%M109	TRUE	Modo de control PID temperatura: TRUE => Automático; FALSE => Manual.
PID_pT1_AutoMan	EBOOL	%M107	TRUE	Modo de control transductor: TRUE => Automático; FALSE => Manual.

Figura 113. Variables elementales. Parte V

Nombre	Tipo	Dirección	Valor	Comentario
min_tank1	EBOOL			nivel minimo alcanzado para el tanque1
min_tank2	EBOOL			nivel minimo alcanzado para el tanque2
min_tank3	EBOOL			nivel minimo alcanzado para el tanque3
accion_valvula_salida_tank1	EBOOL	%M11		Orden a la valvula A1 desde el SCADA
accion_valvula_salida_tank2	EBOOL	%M13		Orden a la valvula A2 desde el SCADA
accion_valvula_salida_tankaux	EBOOL	%M17		Orden a la valvula A4 desde el SCADA
accion_valvula_salida_motobomba	EBOOL	%M18		Orden a la valvula A5 desde el SCADA
accion_valvula_entrada_tank1	EBOOL	%M12		Orden a la valvula B1 desde el SCADA
accion_valvula_entrada_tank2	EBOOL	%M14		Orden a la valvula B2 desde el SCADA
accion_valvula_entrada_tank3	EBOOL	%M15		Orden a la valvula B3 desde el SCADA
accion_valvula_presion_tank3	EBOOL	%M16		Orden a la valvula P3 desde el SCADA
accion_agitador	EBOOL	%M19		Orden al agitador del deposito central desde el SCADA
accion_rele_control	EBOOL	%M22		Orden al relé que habilita el control desde el SCADA
accion_rele_itv	EBOOL	%M24		Orden al relé que habilita el transductor desde el SCADA
accion_rele_electrovalvula	EBOOL	%M23		Orden al relé que habilita la electrovalvula desde el SCADA
accion_rele_Peltier	EBOOL	%M21		Orden al relé que habilita la Peltier desde el SCADA
accion_ventilador	EBOOL	%M20		Orden al ventilador de la Peliter desde el SCADA
outbias_PID_Caud	REAL	%MW340	0.0	Outbias para compensación del PID de caudal
outbias_PID_Niv	REAL	%MW344	0.0	Outbias para compensación del PID de nivel
outbias_PID_Temp	REAL	%MW356	0.0	Outbias para compensación del PID de temperatura
demoAnt	EBOOL	%M55		Pasa al paso anterior de la demo.
demoSig	EBOOL	%M54		Pasa al siguiente paso de la demo.
PID_Caud_enable	EBOOL	%M100	FALSE	PID caudal enable
PID_Caud_enable_SCADA	EBOOL	%M130	FALSE	PID caudal enable desde SCADA
PID_Niv_enable	EBOOL	%M101	FALSE	PID nivel enable
PID_Niv_enable_SCADA	EBOOL	%M131	FALSE	PID nivel enable desde SCADA
PID_pT1_enable	EBOOL	%M102	FALSE	PID presión tanque 1 enable

Figura 114. Variables elementales. Parte VI

Nombre	Tipo	Dirección	Valor	Comentario
PID_pT1_enable_SCADA	EBOOL	%M132	FALSE	PID presión tanque 1 enable desde SCADA
PID_Temp_enable	EBOOL	%M104	FALSE	PID temperatura enable
PID_Temp_enable_SCADA	EBOOL	%M134	FALSE	PID temperatura enable desde SCADA
plcreg_0	EBOOL	%Q0.3.10		PLC/E5CN Index 0. AV-M (R4 & R5)
plcreg_1	EBOOL	%Q0.3.11		PLC/E5CN Index 1.ITV (R10)
presion_tank3out1	EBOOL	%I0.2.10		presostato digital tanque3 (salida 1)
presion_tank3out2	EBOOL	%I0.2.11		presostato digital tanque3 (salida 2)
pulsador_reset	EBOOL	%I0.2.3		pulsador de reset
marcha	EBOOL	%I0.2.0		pulsador de start
paro	EBOOL	%I0.2.1		pulsador de stop
salida_electrovalvula	INT	%QW0.6.1		salida analogica para la electrovalula proporcional del tanque3
motobomba	INT	%QW0.6.3		salida analogica para la motobomba
salida_celulapeltier	INT	%QW0.6.2		salida analogica para las celulas peltier
OUT_PID_pT1	REAL	%MW188		Salida del PID de presión del tanque 1
OUT_PID_Caud	REAL	%MW180		Salida del PID de caudal
OUT_PID_Niv	REAL	%MW184		Salida del PID de nivel
OUT_PID_Temp	REAL	%MW196		Salida del PID de temperatura
OUTman_PID_Caud	REAL	%MW320	0.0	Salida manual del PID de caudal
OUTman_PID_Caud_OPER	REAL		0.0	Salida manual del PID de caudal desde Pantalla de Operador
OUTman_PID_Niv	REAL	%MW324	16000.0	Salida manual del PID de nivel
OUTman_PID_Niv_OPER	REAL		5000.0	Salida manual del PID de nivel desde Pantalla de Operador
OUTman_PID_Temp	REAL	%MW336	0.0	Salida manual del PID de temperatura
OUTman_PID_Temp_OPER	REAL		0.0	Salida manual del PID de temperatura desde Pantalla de Operador
retraso1	EBOOL			Se activa cuando min1 lleva 3 segundos en false
retraso2	EBOOL			Se activa cuando min2 lleva 3 segundos en false
retraso3	EBOOL			Se activa cuando min3 lleva 3 segundos en false
modo_mant	EBOOL	%M49	FALSE	selecciona si se entra en modo mantenimiento
modo_manual	EBOOL	%M51	FALSE	Selecciona si se entra en modo manual.
running	EBOOL			sistema en ejecucion
SP_Caud_SCADA	REAL	%MW120		Valor deseado de caudal desde el SCADA
SP_Caud_OPER	REAL			Valor deseado de caudal desde Pantalla de Operador

Figura 115. Variables elementales. Parte VII

Nombre	Tipo	Dirección	Valor	Comentario
SP_Temp_SCADA	REAL	%MW136		Valor deseado de caudal temperatura desde el SCADA
SP_Temp_OPER	REAL			Valor deseado de caudal temperatura desde Pantalla de Operador
SP_Niv_SCADA	REAL	%MW124		Valor deseado de nivel desde el SCADA
SP_Niv_OPER	REAL			Valor deseado de nivel desde Pantalla de Operador
SP_pT1_SCADA	REAL	%MW128		Valor deseado de presión del tanque 1 desde el SCADA
SP_pT1_OPER	REAL			Valor deseado de presión del tanque 1 desde Pantalla de Operador
valvula_entrada_tank 1	EBOOL	%Q0.3.2		valvula de entrada al tanque 1 (llenado)
valvula_entrada_tank 2	EBOOL	%Q0.3.4		valvula de entrada al tanque 2 (llenado)
valvula_entrada_tank 3	EBOOL	%Q0.3.6		valvula de entrada al tanque 3 (llenado)
valvula_presion_tank 3	EBOOL	%Q0.3.5		Valvula de presurización del tanque 3
valvula_salida_motob omba	EBOOL	%Q0.3.15		valvula de salida de la bomba
valvula_salida_tank1	EBOOL	%Q0.3.1		valvula de salida del tanque 1 (vaciado)
valvula_salida_tank2	EBOOL	%Q0.3.3		valvula de salida del tanque 2 (vaciado)
valvula_salida_tankau x	EBOOL	%Q0.3.13		valvula de salida del tanque auxiliar
PauseDemo	EBOOL	%M56	FALSE	Variable para pausar el SFC de demostración
SP_Caud_OPC	REAL	%MW360		Variable para visualizar el SP de Caudal en el OPC
SP_Niv_OPC	REAL	%MW364		Variable para visualizar el SP de Nivel en el OPC
SP_pT1_OPC	REAL	%MW368		Variable para visualizar el SP de pT1 en el OPC
SP_Temp_OPC	REAL	%MW372		Variable para visualizar el SP de Temperatura en el OPC
EstadoSFCdemo	INT	%MW0		Variable que almacena el estado del SFC de demostrracion
Control_Matlab	EBOOL	%M0	FALSE	Variable que habilita el control desde MATLAB
ventilador	EBOOL	%Q0.3.12		ventilador de las celulas peltier

Figura 116. Variables elementales. Parte VIII

Pantallas de operador

A continuación, se muestran las pantallas de operador creadas en *Unity Pro XL* con el objetivo de controlar los reguladores del Autómata y servir de alternativa a SCADA directamente en su entorno de desarrollo.

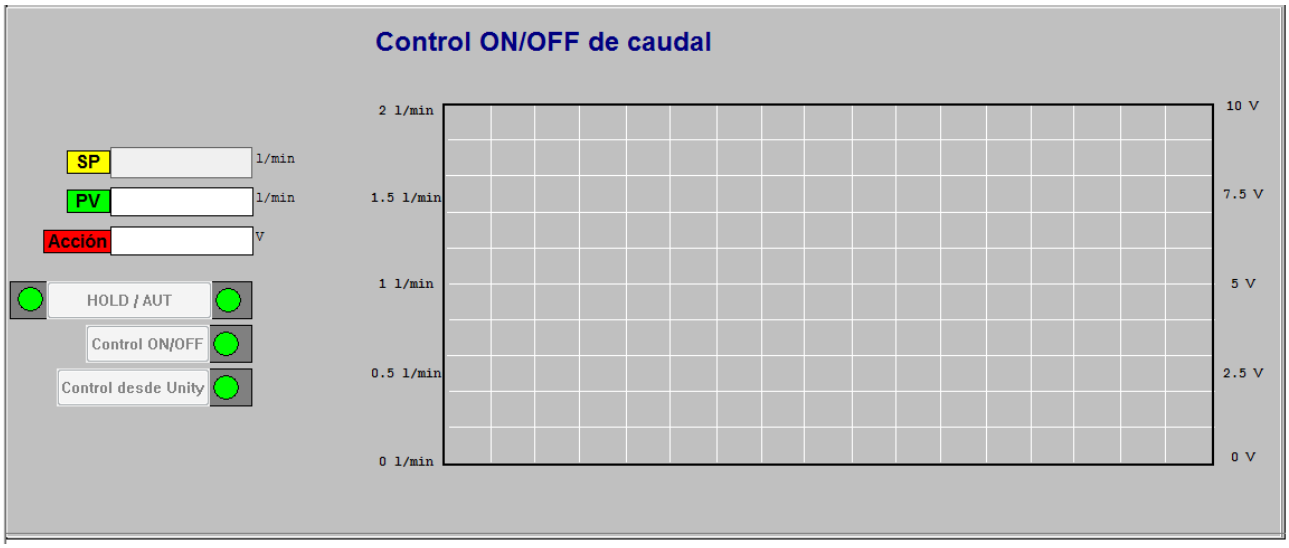


Figura 117. Pantalla del control ON-OFF de caudal

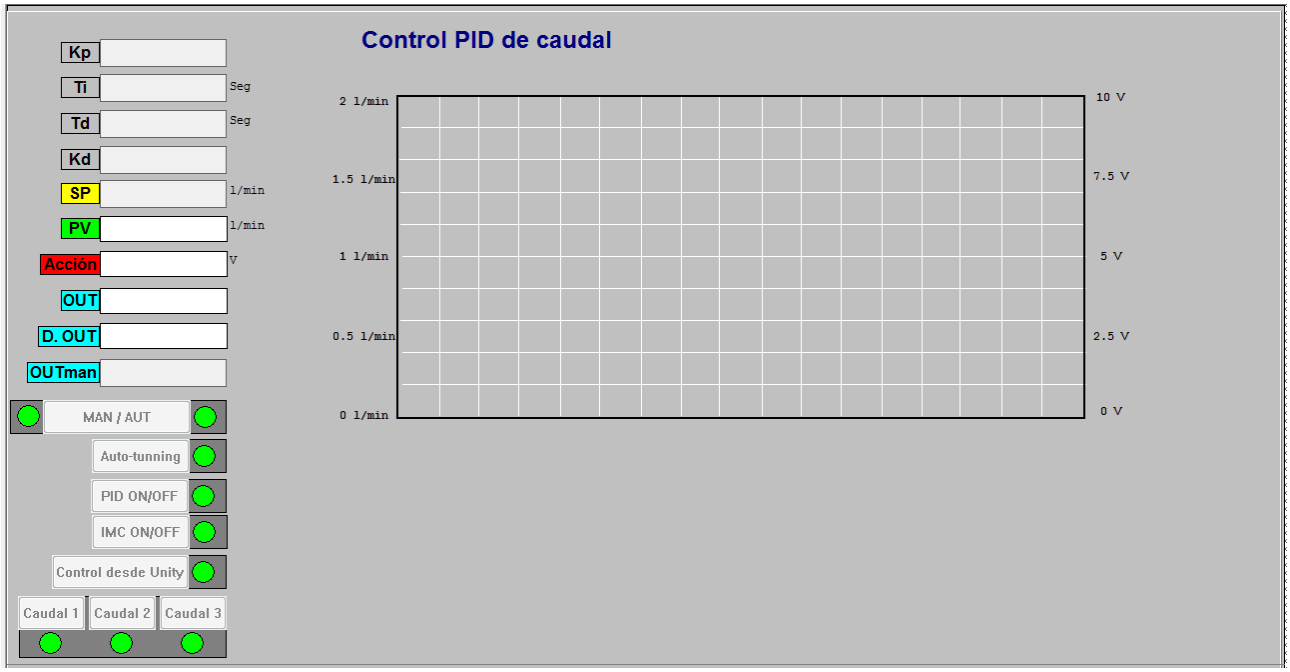


Figura 118. Pantalla de control PID e IMC del caudal

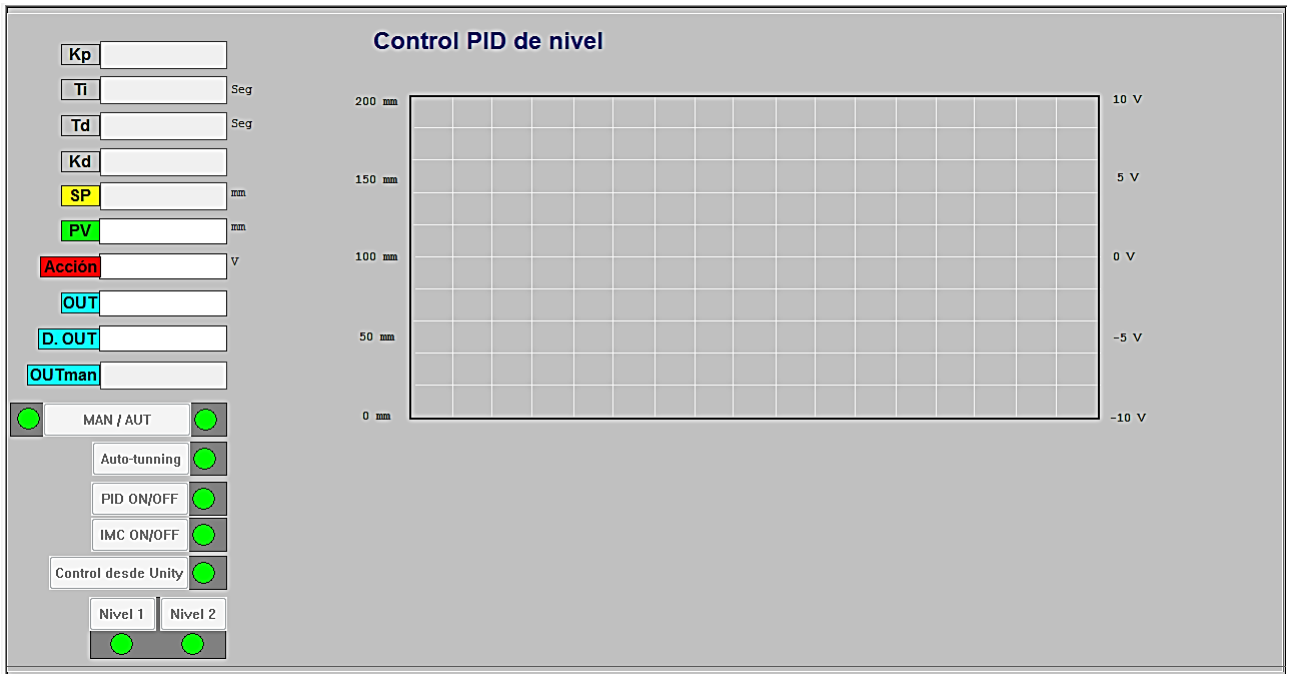


Figura 119. Pantalla de control PID e IMC de nivel del depósito intermedio

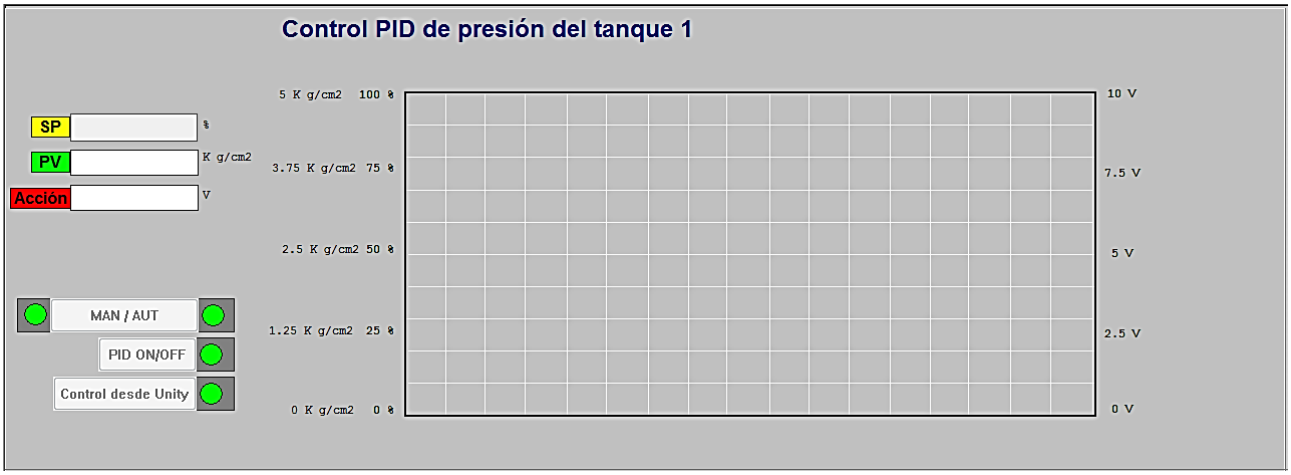


Figura 120. Pantalla de control PID e IMC de presión del depósito izdo

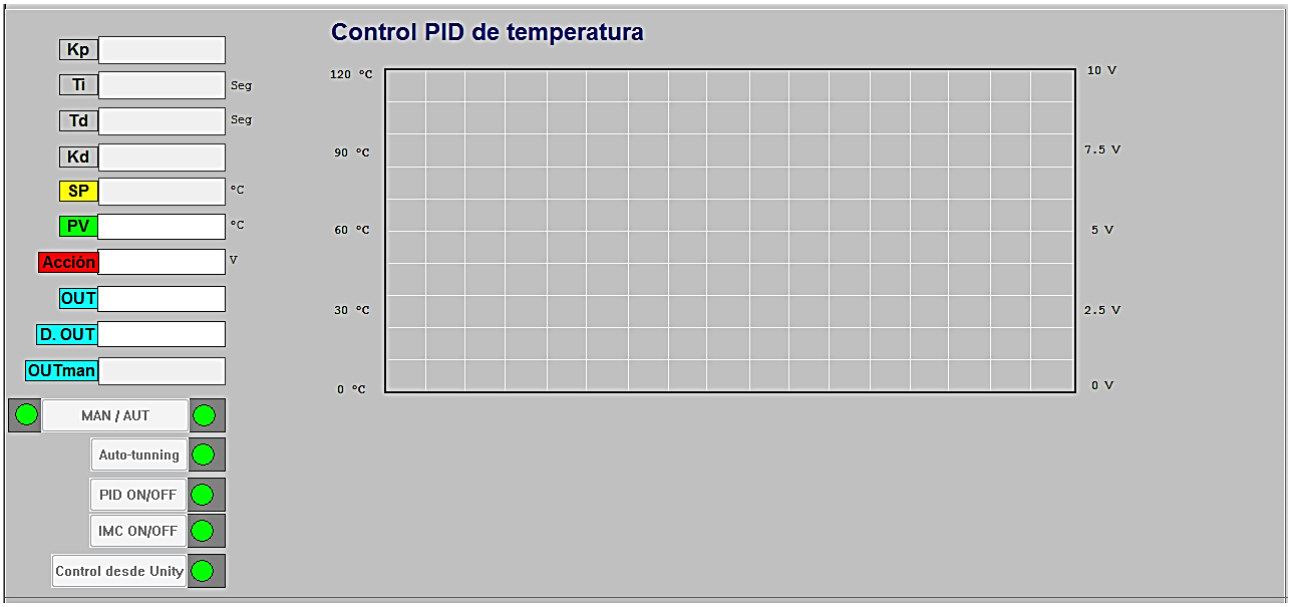


Figura 121. Pantalla de control PID e IMC de temperatura del depósito intermedio

Anexo C. Sistema SCADA

En este Anexo vamos a mostrar las pantallas que componen el programa SCADA desarrollado para este proyecto: pantallas de Menú, control Manual, diferentes tipos de controles de Caudal y Nivel, control de Temperatura y control de Presión del tanque izquierdo. También se muestran las pantallas creadas para el sistema de demostración automático, que explican brevemente la fase en la que están y muestran los sensores, actuadores y variables de los reguladores [17].

Pantallas principales

Pantallas que componen la estructura principal del SCADA.

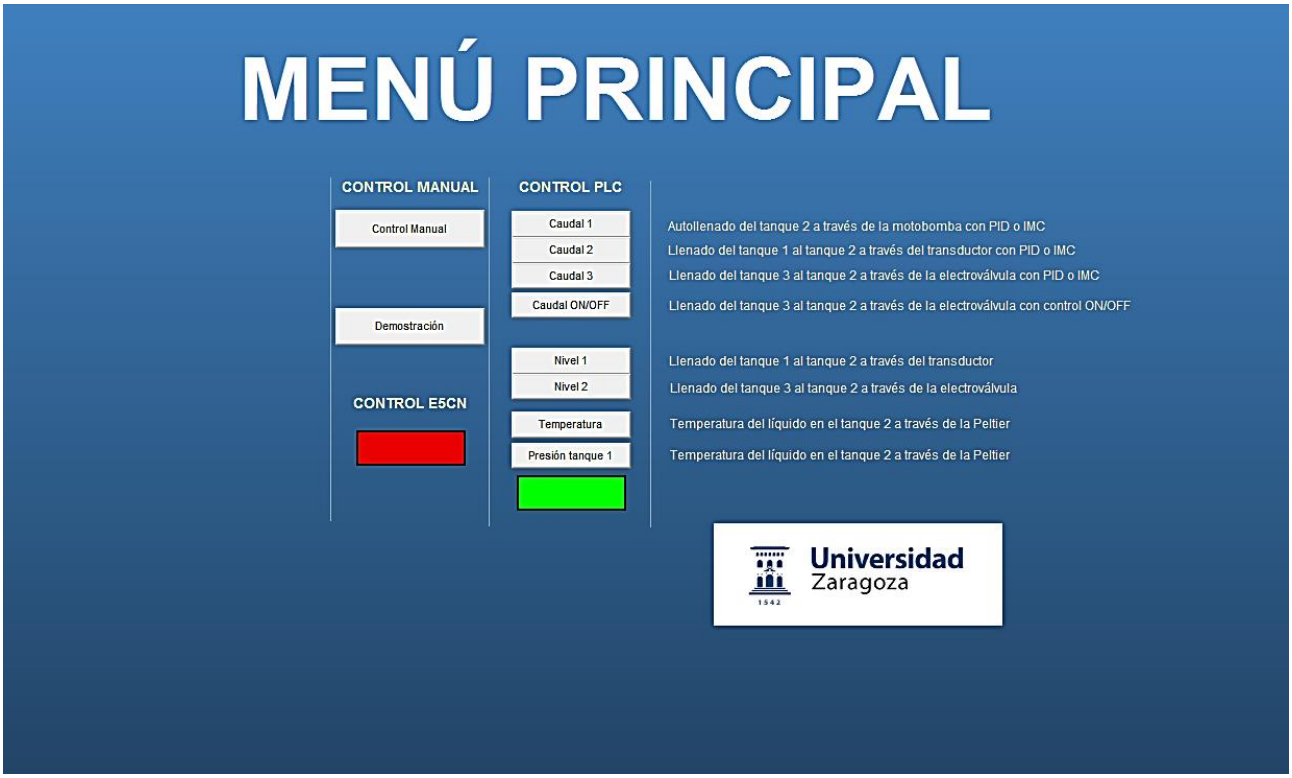


Figura 122. Pantalla de Menú principal

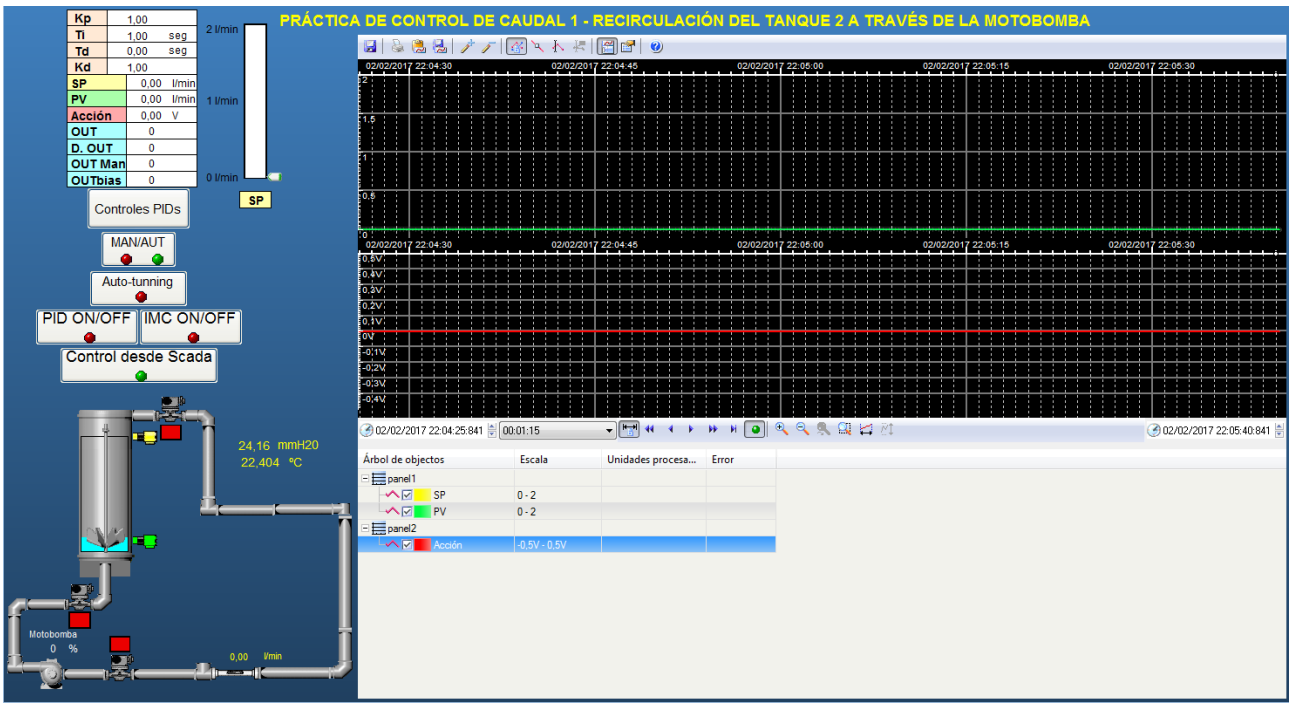


Figura 123. Pantalla de control de Caudal de tipo 1 con PID e IMC

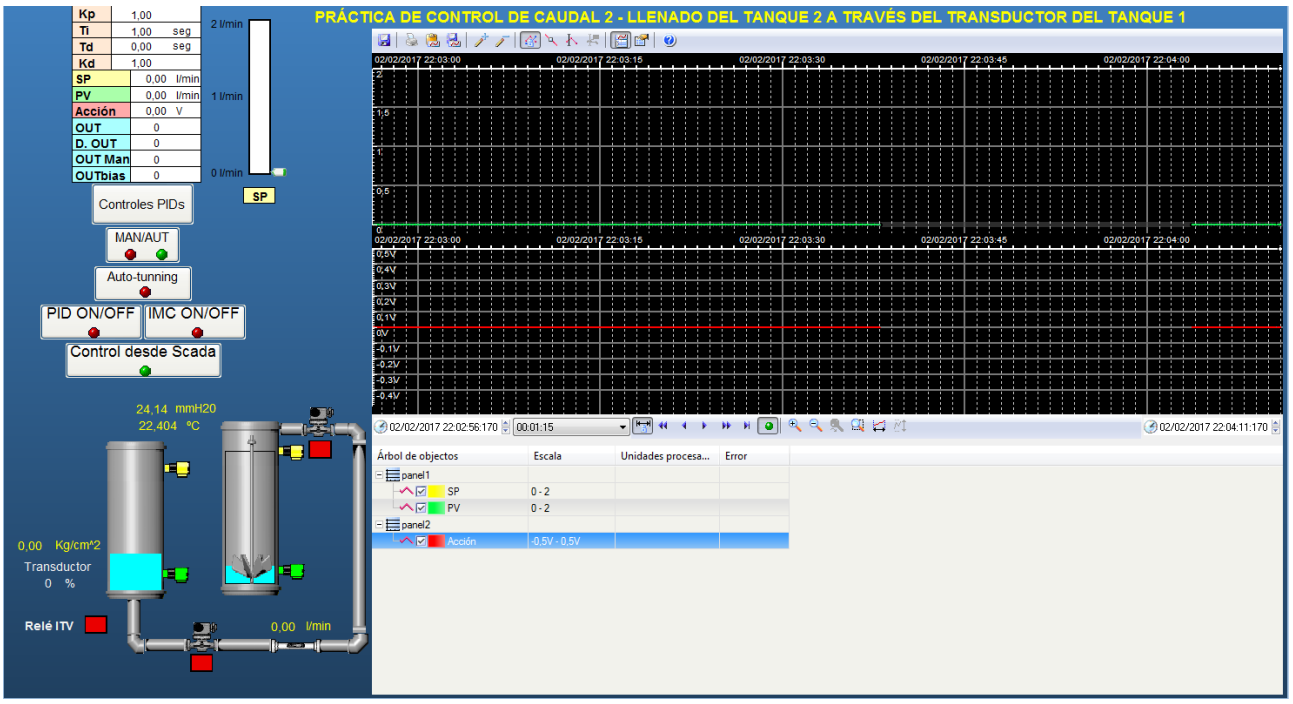


Figura 124. Pantalla de control de Caudal de tipo 2 con PID e IMC

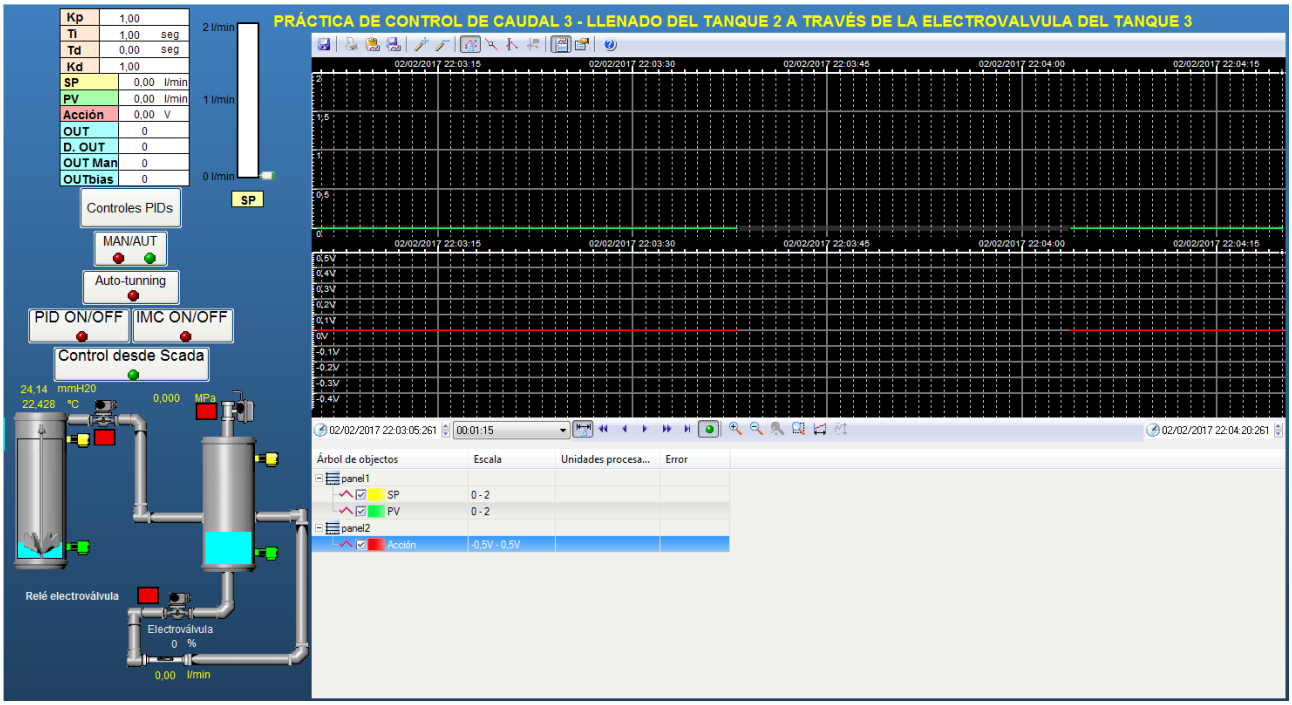


Figura 125. Pantalla de control de Caudal de tipo 3 con PID e IMC

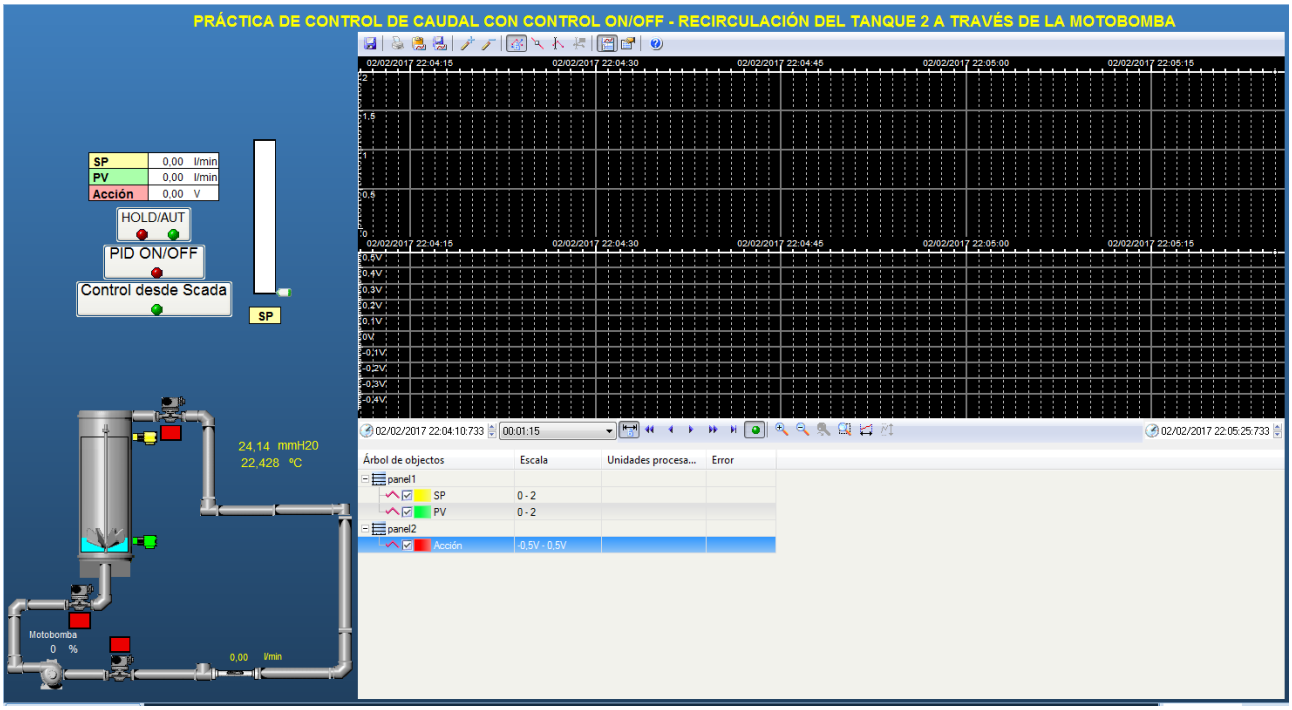


Figura 126. Pantalla de control de Caudal con regulador ON/OFF

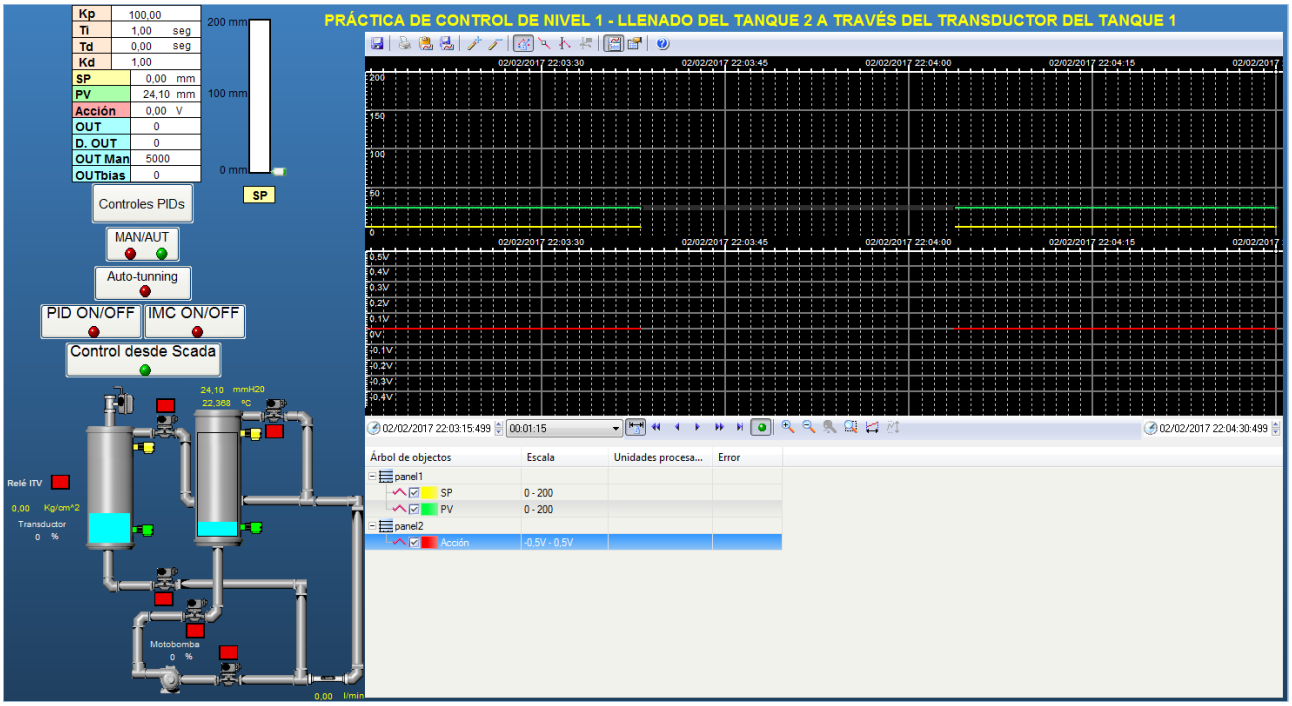


Figura 127. Pantalla de control de Nivel de tipo 1 con PID e IMC

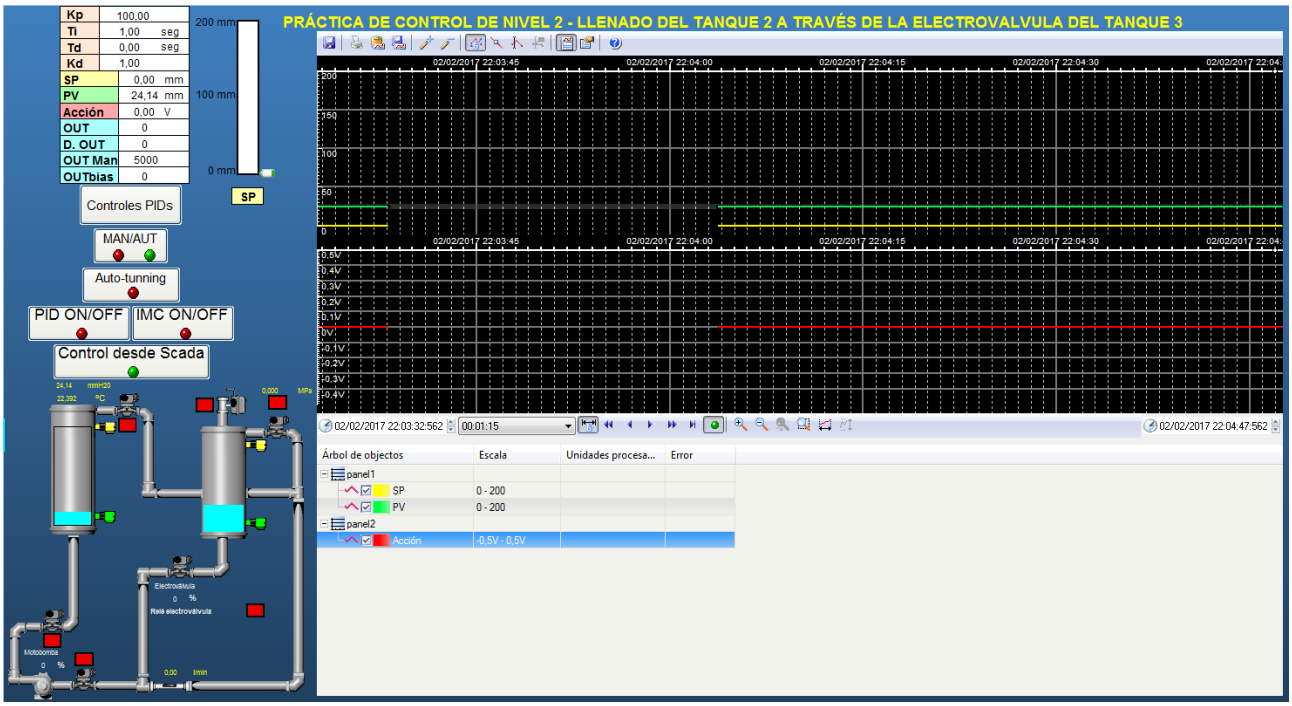


Figura 128. Pantalla de control de Nivel de tipo 2 con PID e IMC

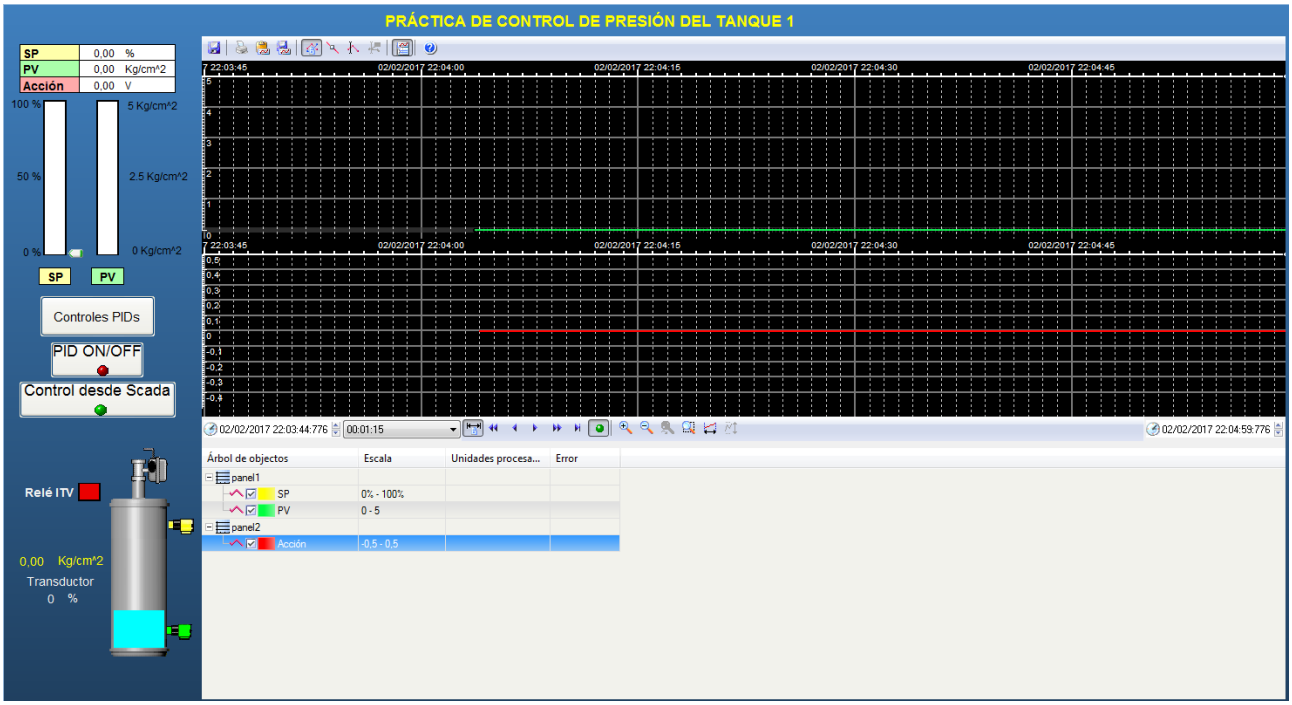


Figura 129. Pantalla de control de Presión del tanque izdo

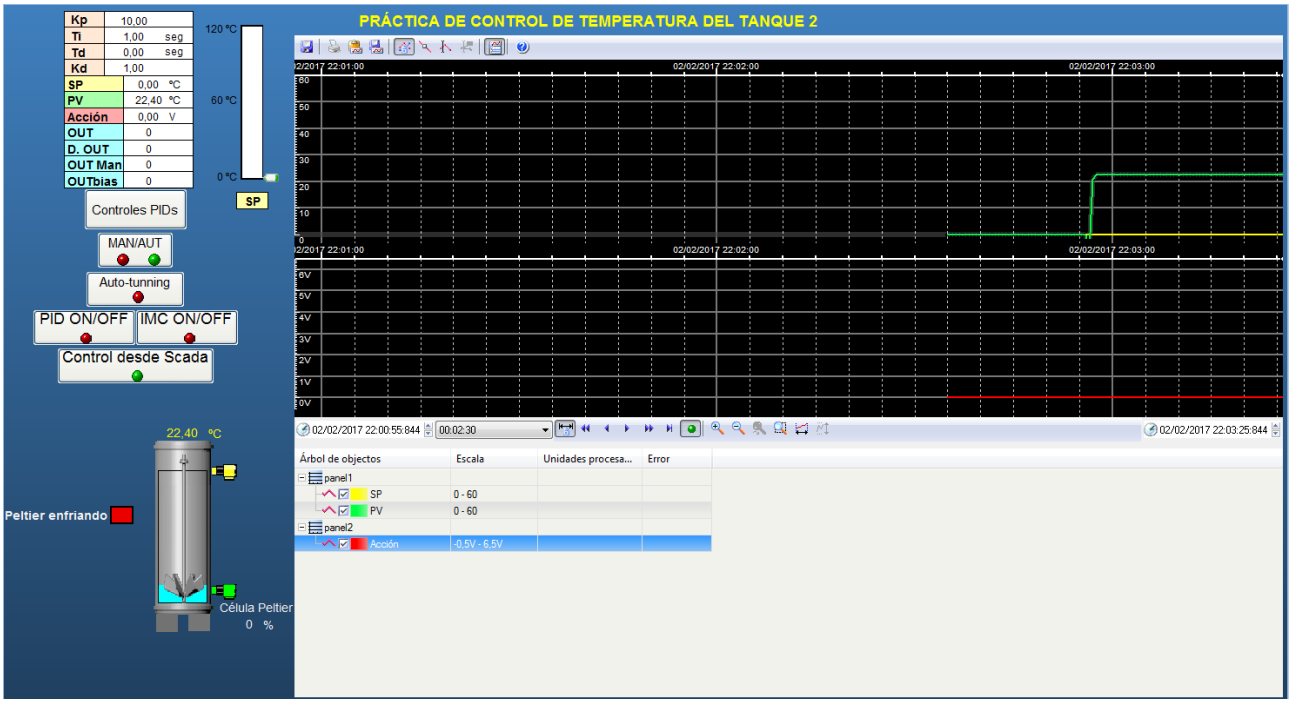


Figura 130. Pantalla de control de Temperatura con PID e IMC

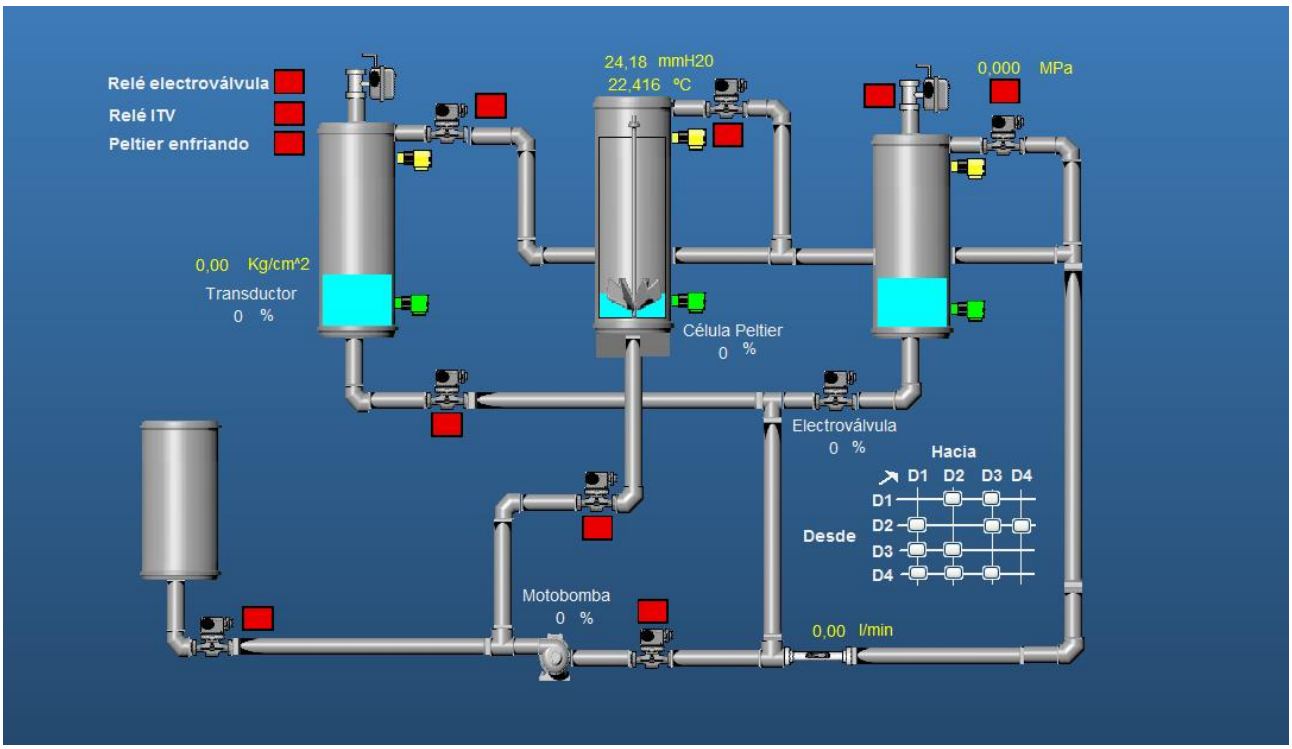


Figura 131. Pantalla de Control Manual de los sensores y actuadores

Pantallas de la demostración automática

Pantallas creadas con la explicación de las etapas de la demostración automática. También se dispone de una visualización en tiempo real de las variables de los reguladores.

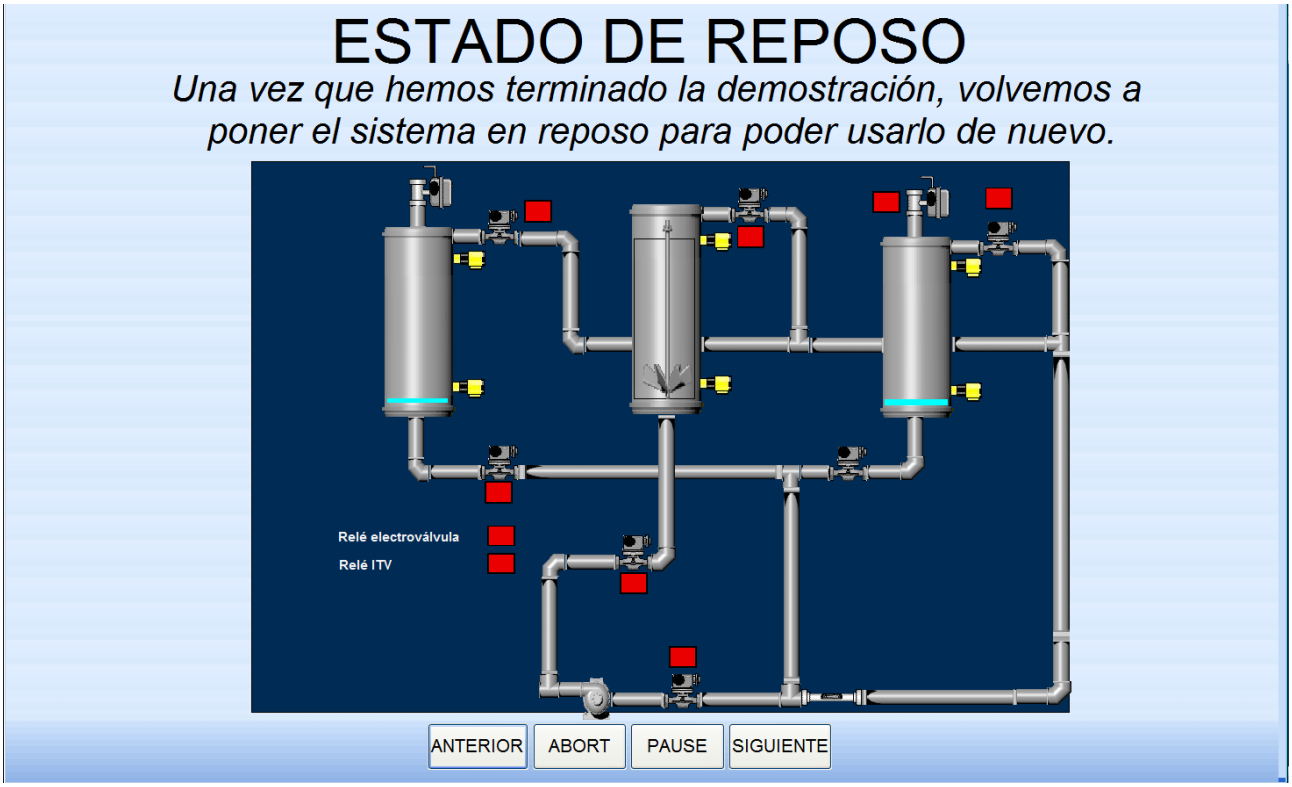


Figura 132. Estado de Reposo

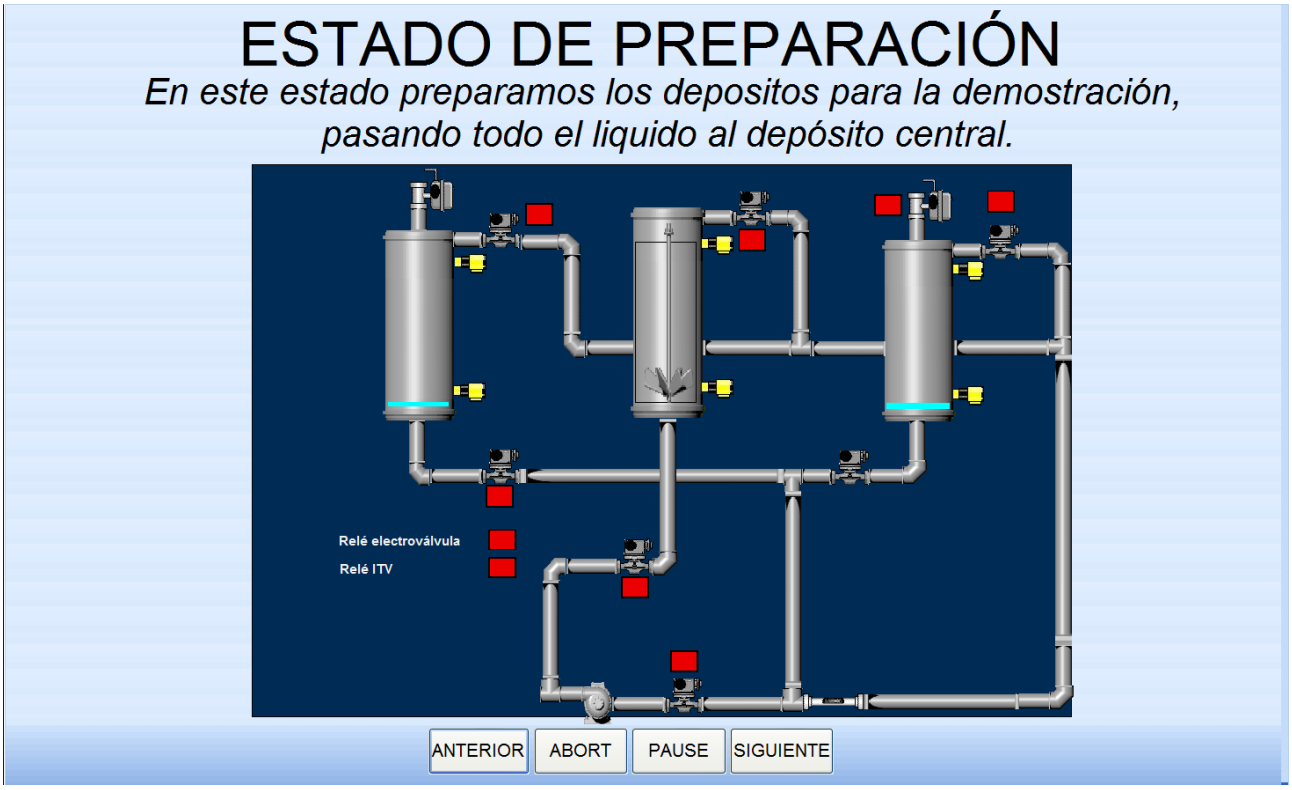


Figura 133. Estado de Preparación de los depósitos

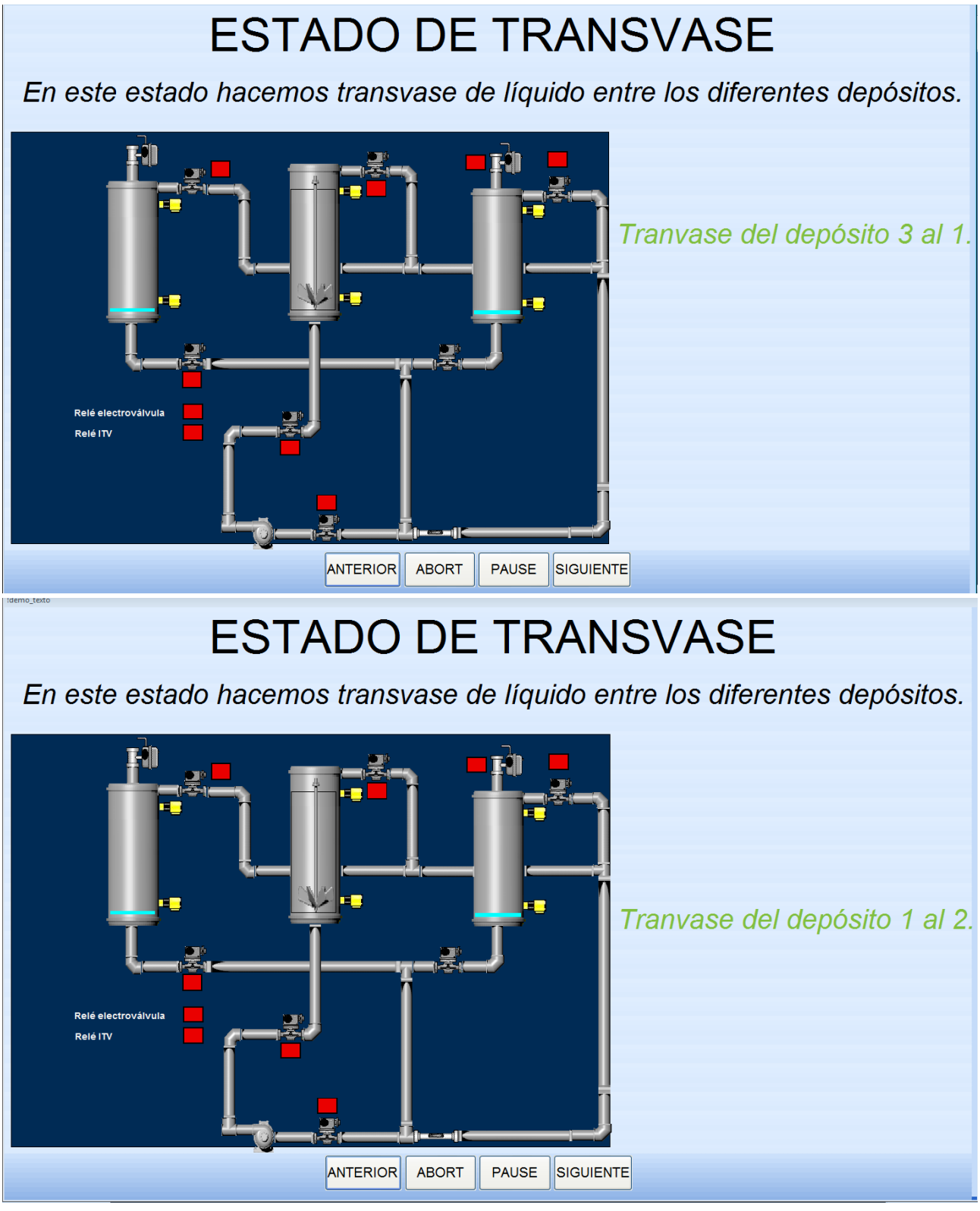


Figura 134. Estados de Transvase entre depósitos

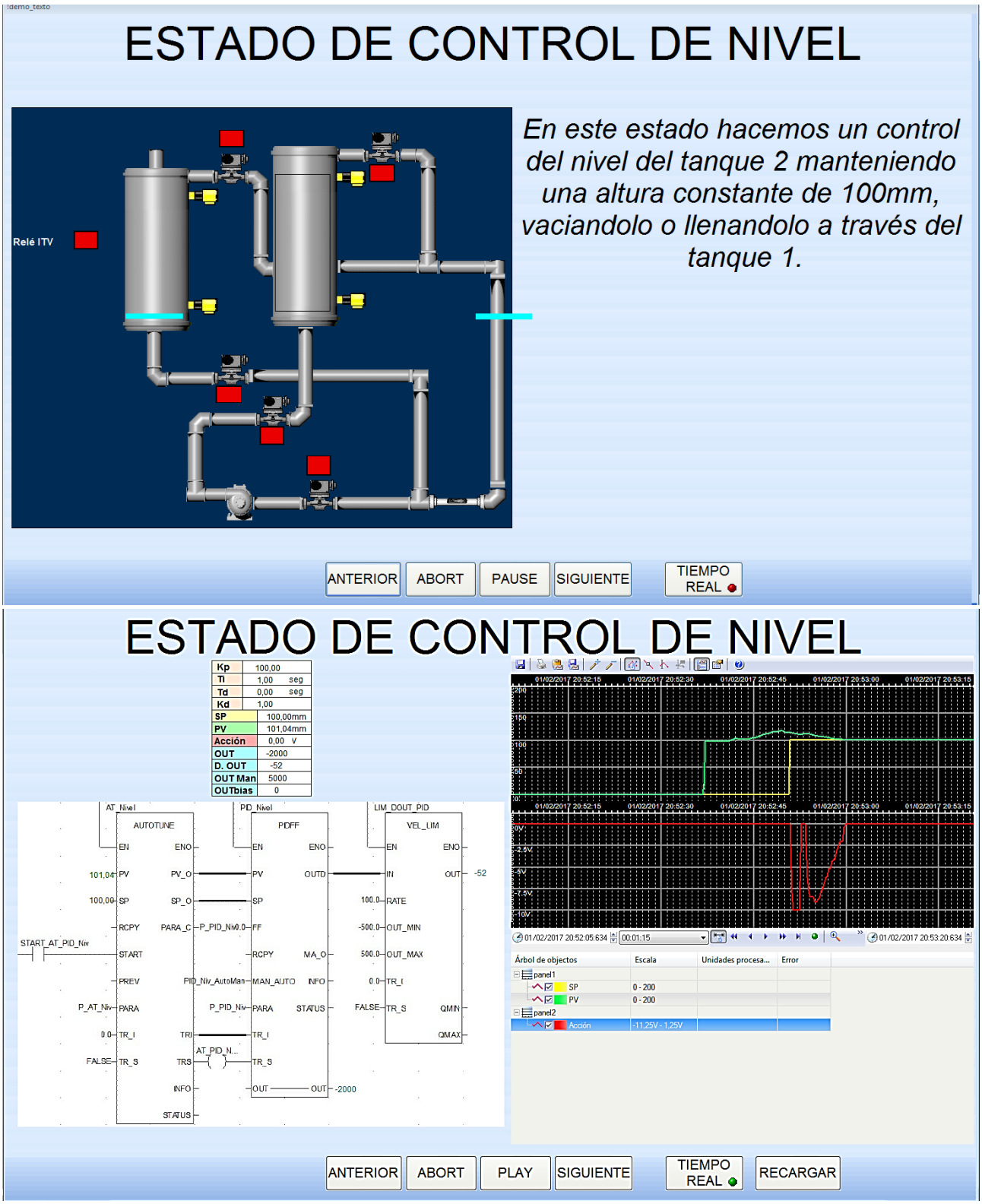


Figura 135. Estados de Control de Nivel tipo 1 con PID

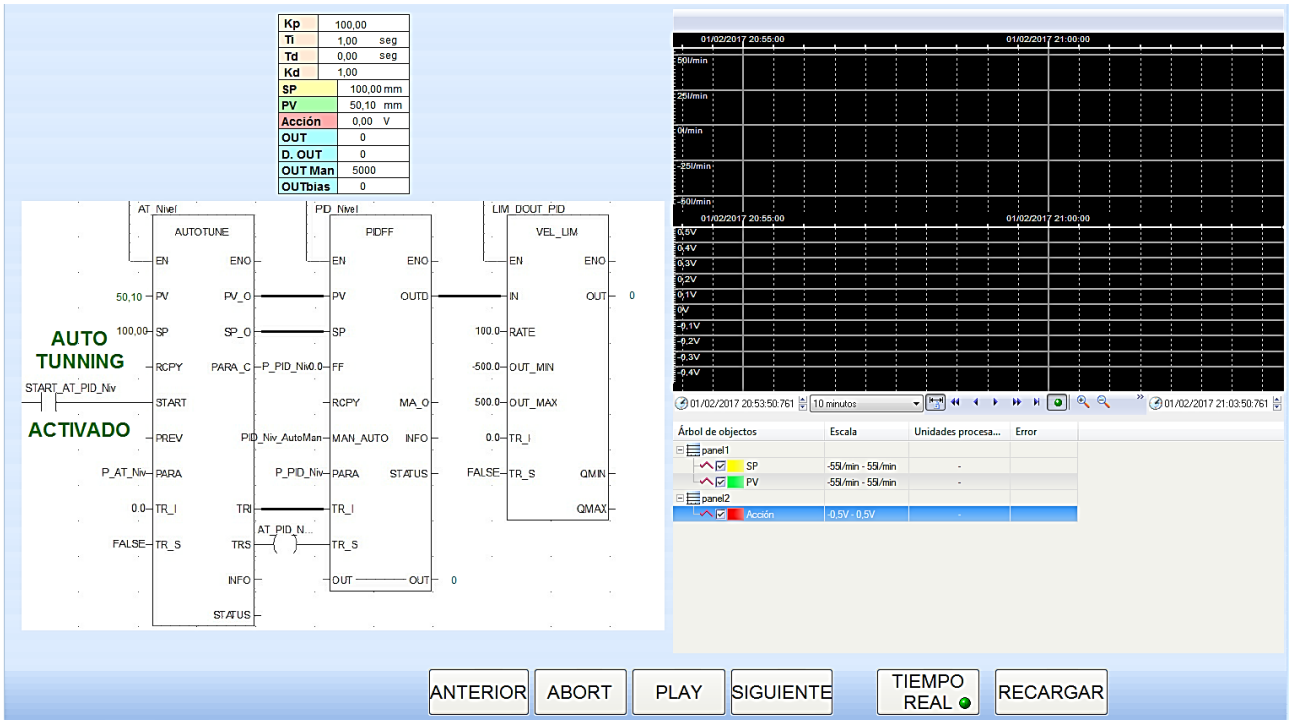
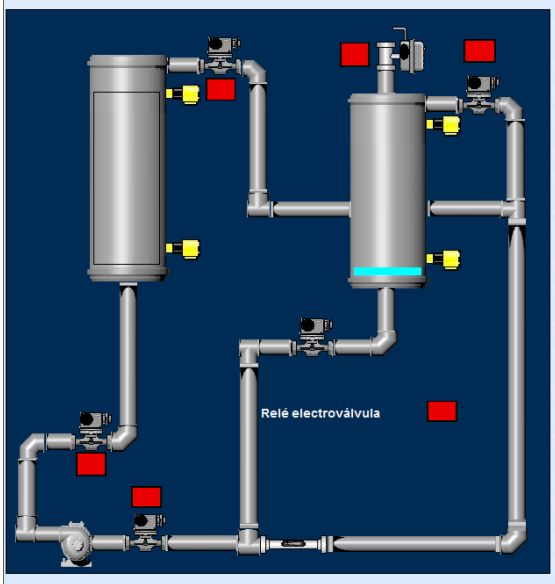


Figura 136. Estados de Autotuning para PID de Nivel tipo 2

ESTADO DE CONTROL DE NIVEL

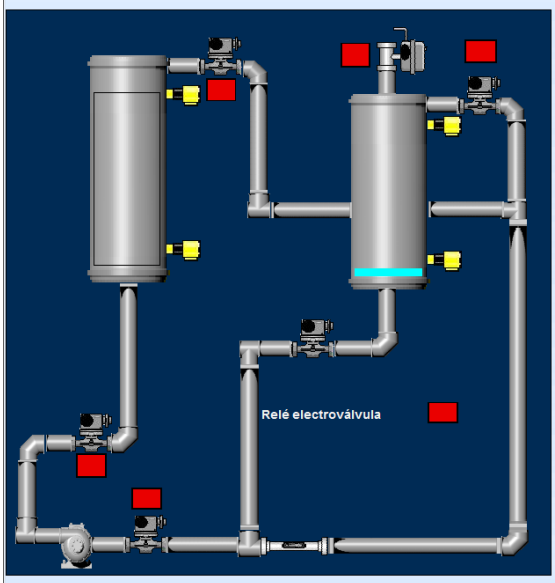


En este estado hacemos un control del nivel del tanque 2 manteniendo alternando una altura constante durante un tiempo y otra durante otro periodo, así durante 2 ciclos vaciandolo o llenandolo a través del tanque 3.

Nivel del tanque 2 de 20 mm.

ANTERIOR | ABORT | PAUSE | SIGUIENTE | TIEMPO REAL

ESTADO DE CONTROL DE NIVEL



En este estado hacemos un control del nivel del tanque 2 manteniendo alternando una altura constante durante un tiempo y otra durante otro periodo, así durante 2 ciclos vaciandolo o llenandolo a través del tanque 3.

Nivel del tanque 2 de 80 mm.

ANTERIOR | ABORT | PAUSE | SIGUIENTE | TIEMPO REAL

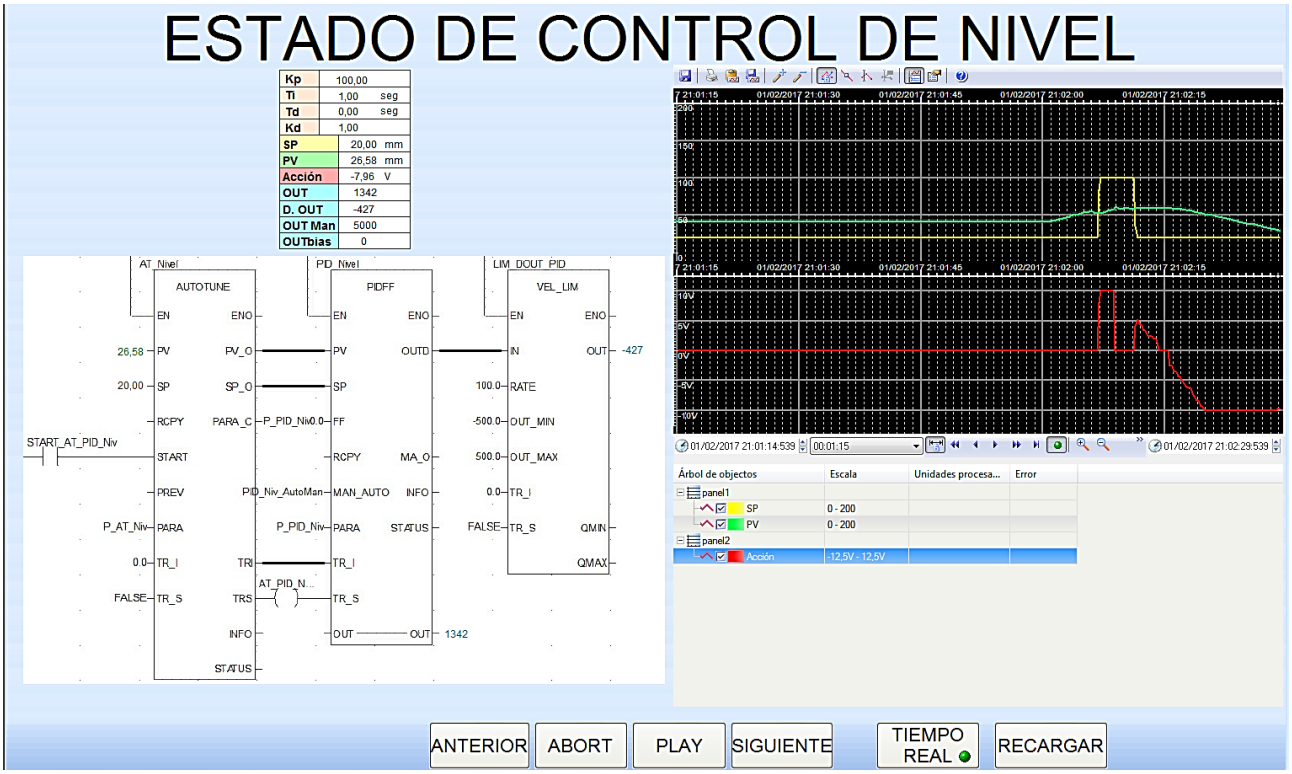


Figura 137. Estados de Control de Nivel tipo 2 con PID

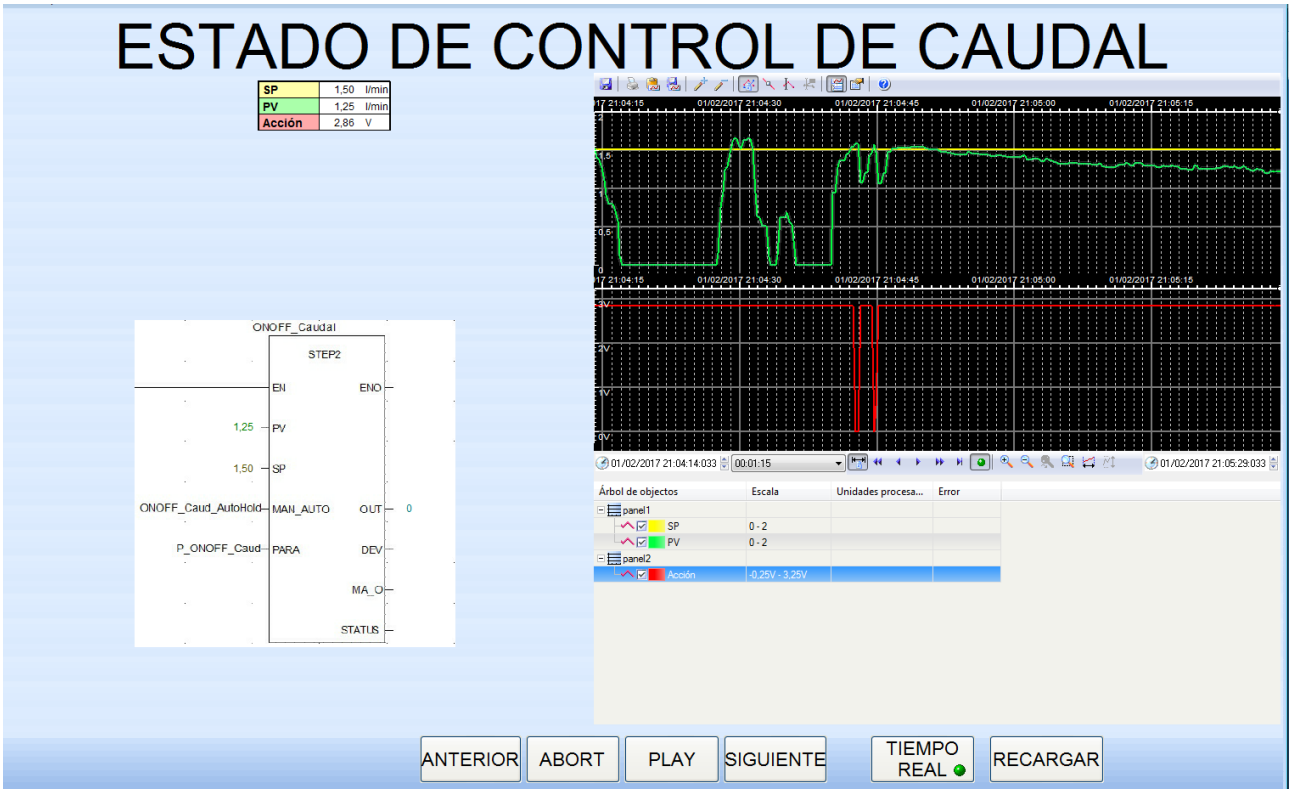


Figura 138. Estados de Control de Caudal con regulador ON/OFF

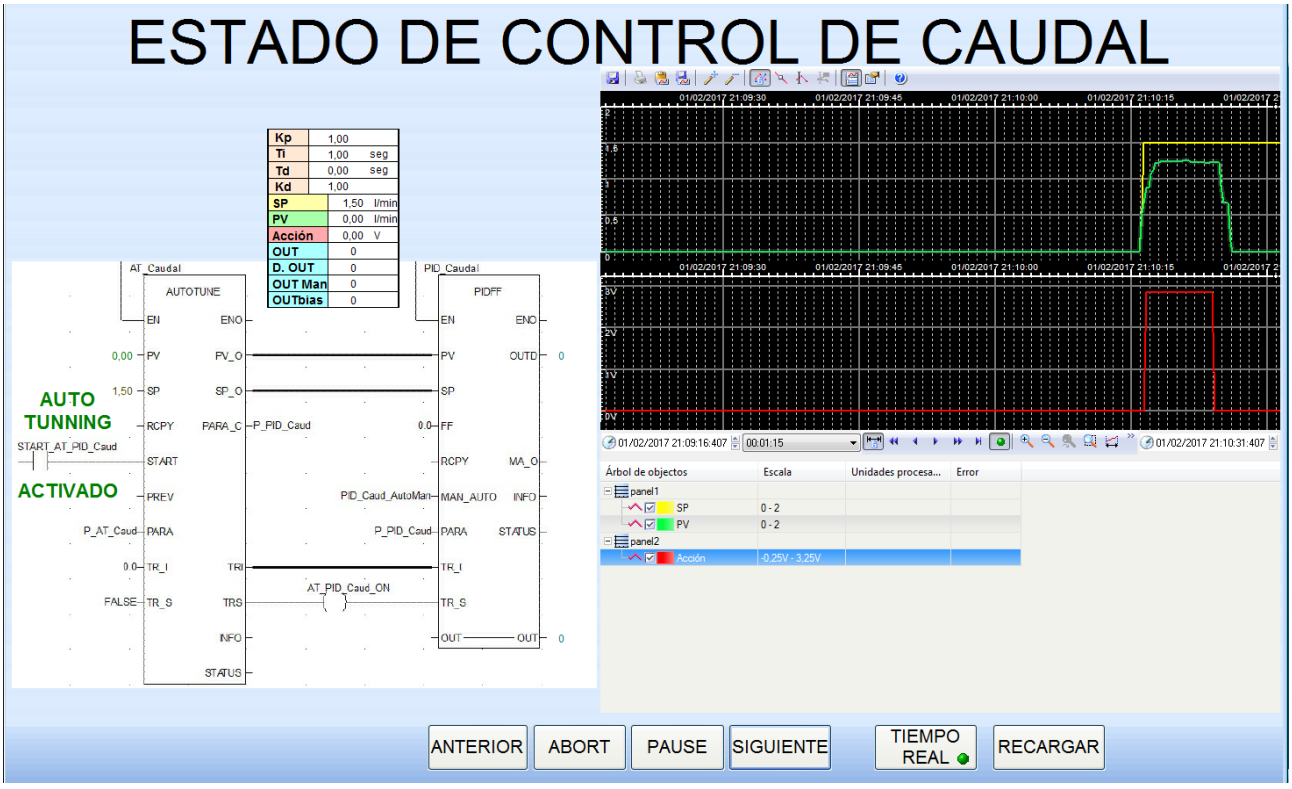
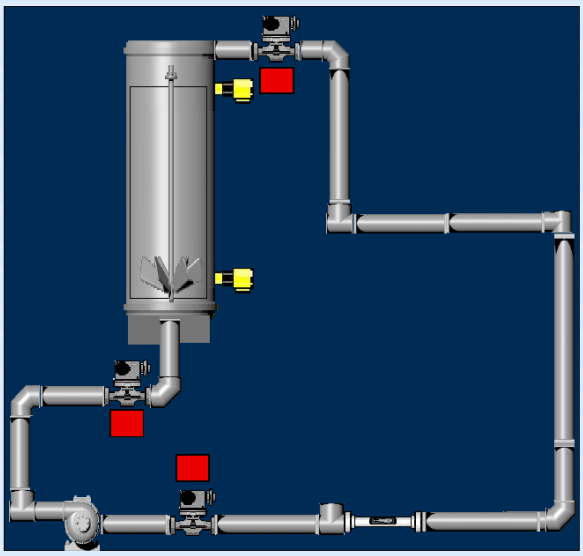


Figura 139. Estados de Autotuning para PID de Caudal tipo 1

ESTADO DE CONTROL DE CAUDAL

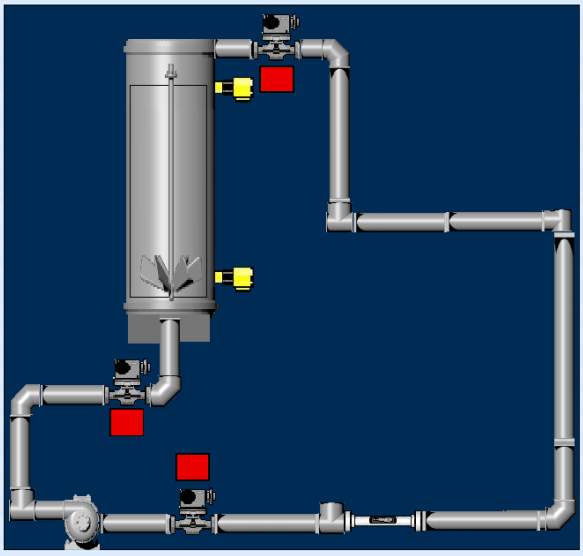


En este estado hacemos un control del caudal del tanque 2 manteniendo alternando un caudal constante durante un tiempo y otra durante otro periodo, así durante 2 minutos recirculandolo por dicho tanque.

Caudal de 1 litro/minuto.

ANTERIOR | ABORT | PAUSE | SIGUIENTE | TIEMPO REAL

ESTADO DE CONTROL DE CAUDAL



En este estado hacemos un control del caudal del tanque 2 manteniendo alternando un caudal constante durante un tiempo y otra durante otro periodo, así durante 2 minutos recirculandolo por dicho tanque.

Caudal de 1,5 litro/minuto

ANTERIOR | ABORT | PAUSE | SIGUIENTE | TIEMPO REAL

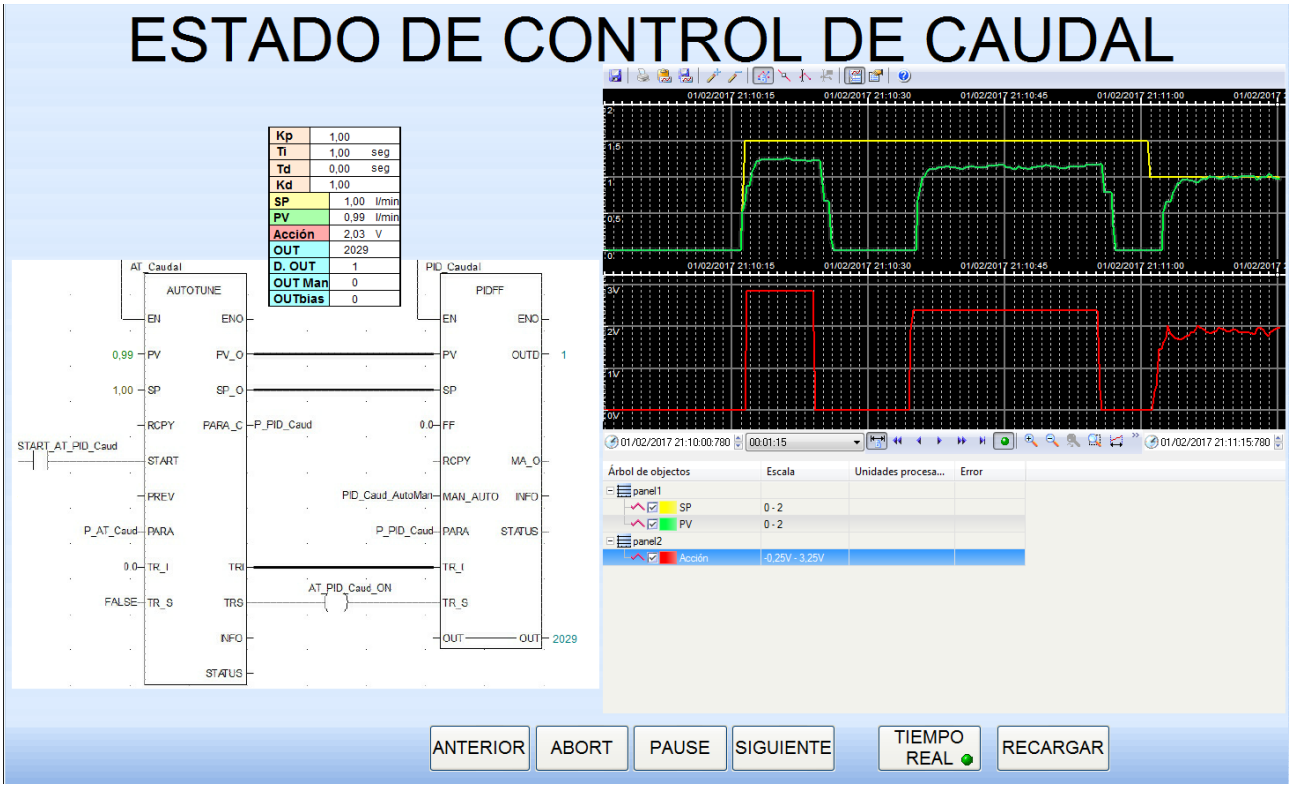
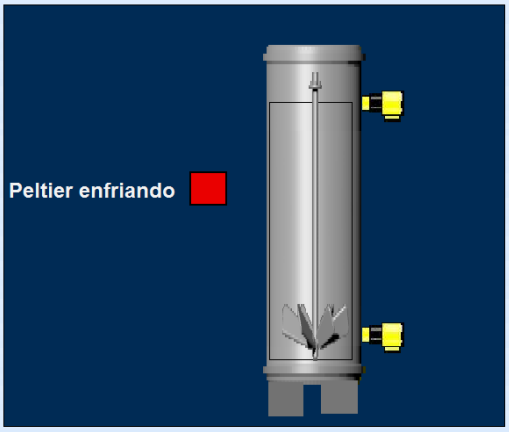


Figura 140. Estados de Control de Caudal tipo 1 con PID

ESTADO DE CONTROL DE TEMPERATURA

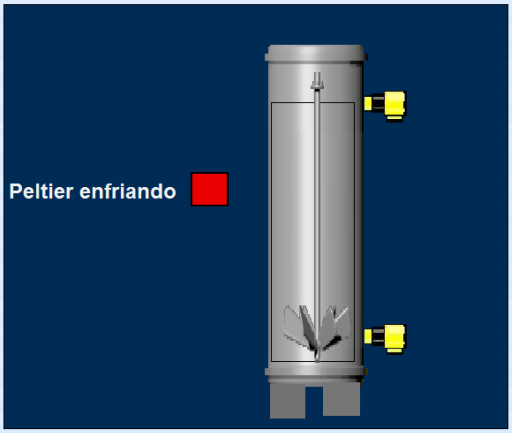


En este estado hacemos un control de la temperatura del tanque 2, dando consigna de temperatura constante a traves del PID o del IMC, según la etapa.

Temperatura de 25 grados con PID.

ANTERIOR | ABORT | PAUSE | SIGUIENTE | TIEMPO REAL

ESTADO DE CONTROL DE TEMPERATURA



En este estado hacemos un control de la temperatura del tanque 2, dando consigna de temperatura constante a traves del PID o del IMC, según la etapa.

Temperatura de 5 grados superior al ambiente con PID.

ANTERIOR | ABORT | PAUSE | SIGUIENTE | TIEMPO REAL

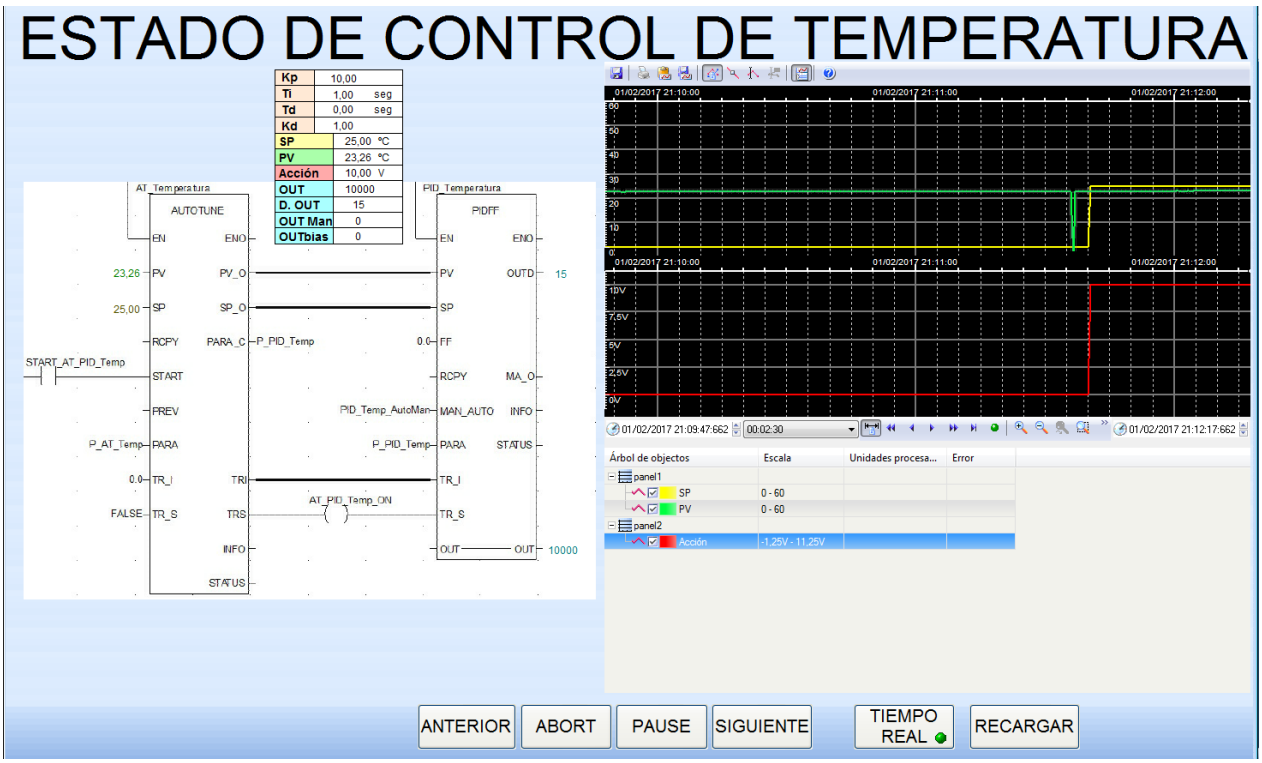


Figura 141. Estados de Control de Temperatura con PID

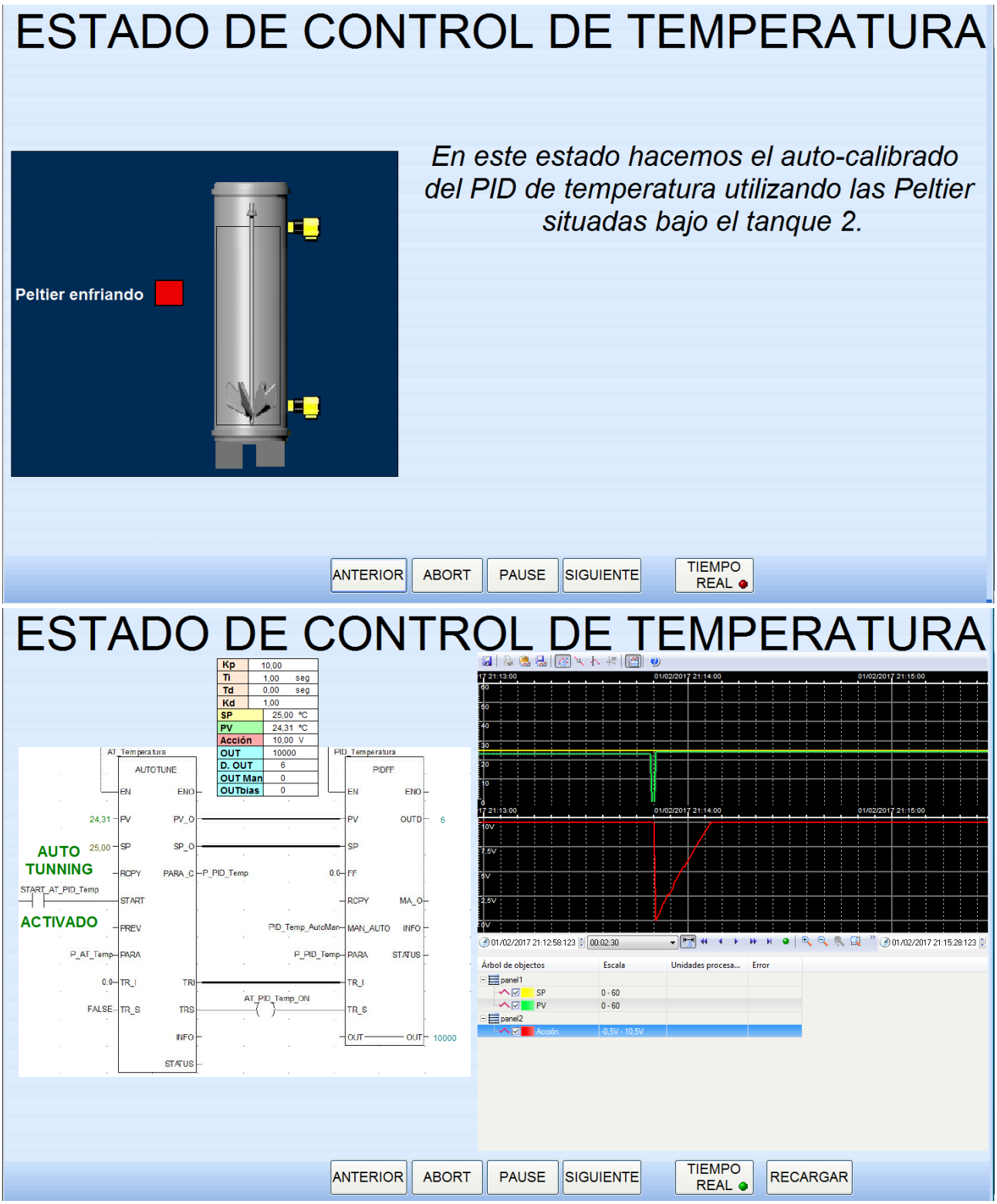


Figura 142. Estados de Autotuning para PID de Temperatura

Figura 143. Estados de Control de Temperatura con PID

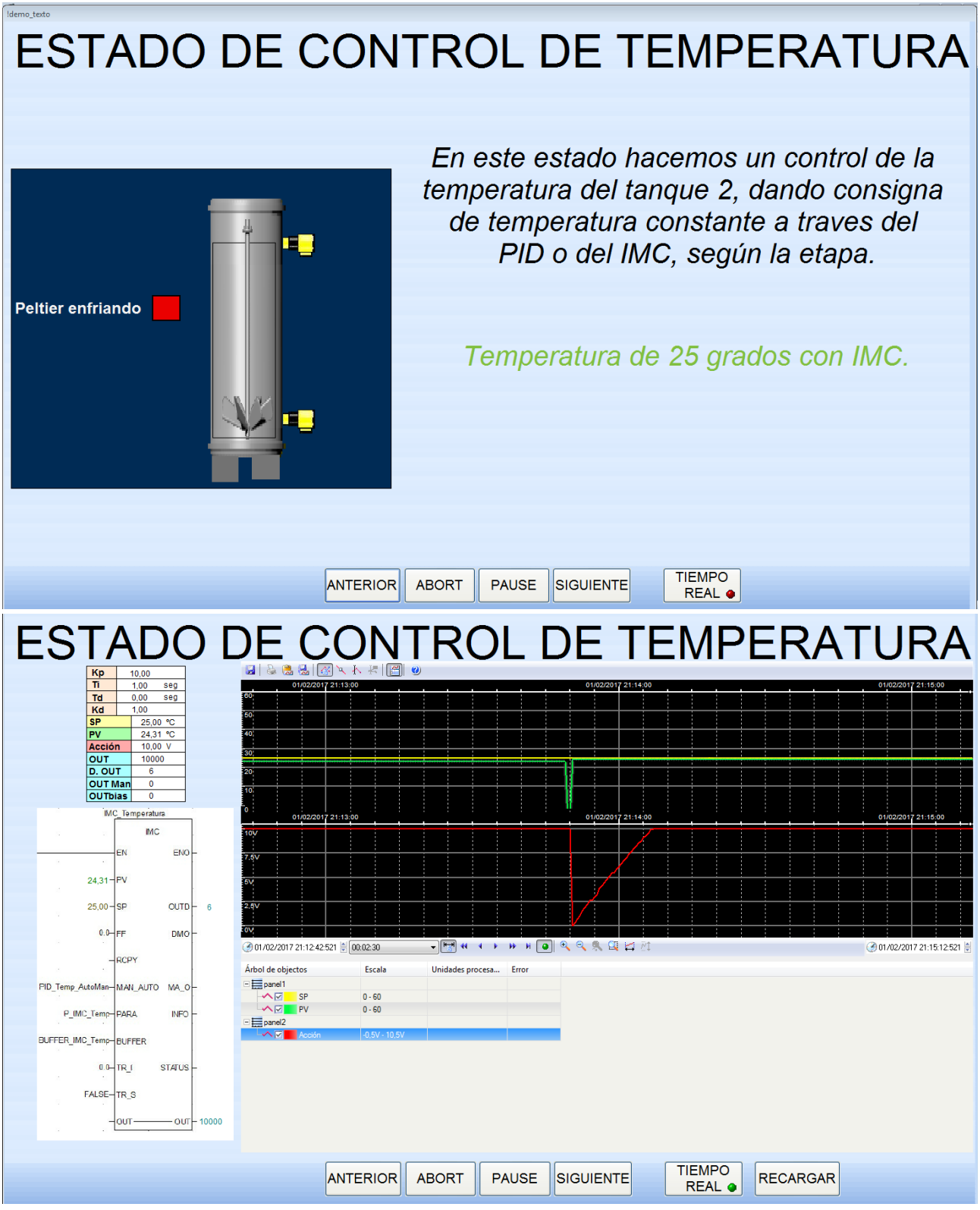


Figura 144. Estados de Control de Temperatura con IMC

Figura 145. Estados de Control de Temperatura con IMC

Figura 146. Estados de Control de Temperatura con IMC

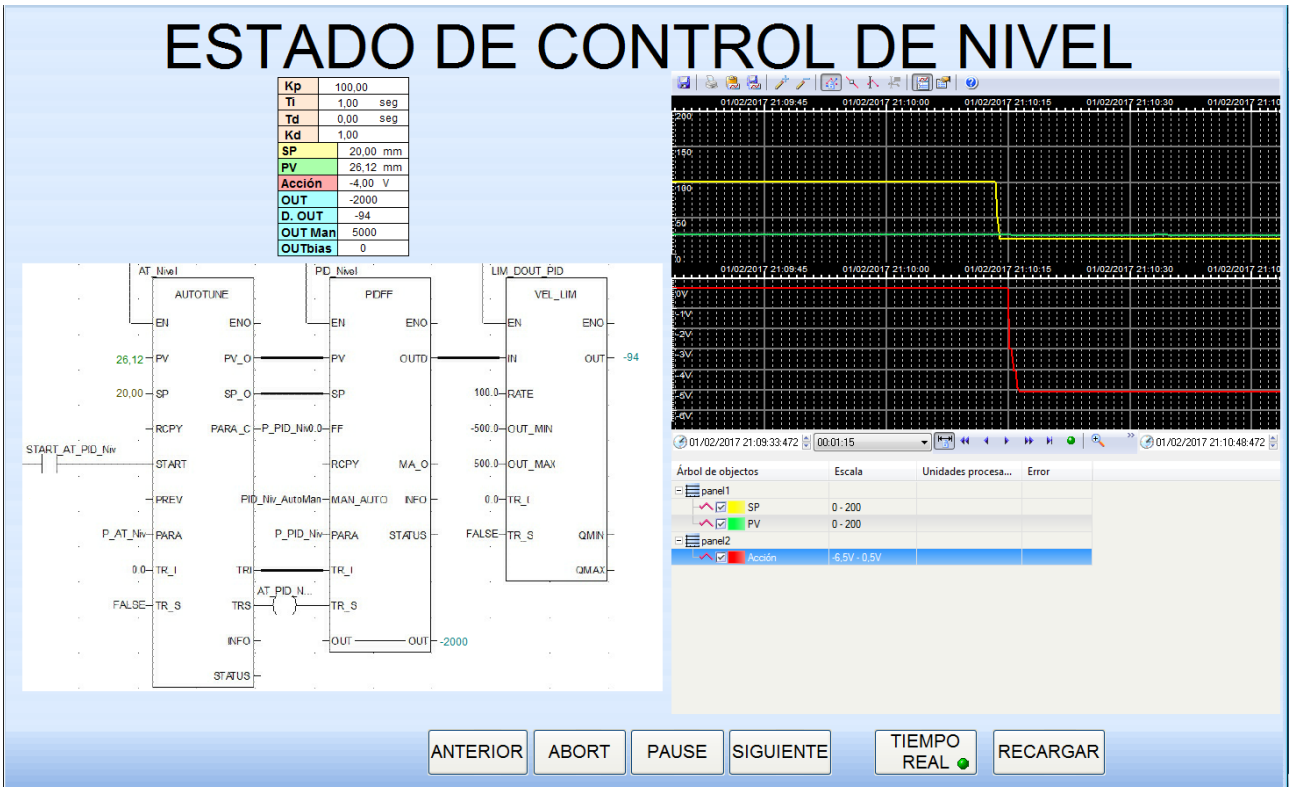


Figura 147. Estados de Control de Nivel tipo 2 con PID

Anexo D. Programa Matlab

En este Anexo se muestran los componentes del programa hecho en Matlab. Se divide *Scripts*, Modelos *Simulink* y figuras [18].

Scripts

En este apartado se exponen los códigos de lenguaje Matlab de nuestro programa. Empezaremos con la administración de la ventana de selección de tipo de control tanto de caudal como de nivel, para luego mostrar el código que permite la sincronización de Matlab con el reloj de sistema.

```
function varargout = Controles(varargin)
% CONTROLES MATLAB code for Controles.fig
%      CONTROLES, by itself, creates a new CONTROLES or raises the existing
%      singleton*.
%
%      H = CONTROLES returns the handle to a new CONTROLES or the handle to
%      the existing singleton*.
%
%      CONTROLES('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CONTROLES.M with the given input arguments.
%
%      CONTROLES('Property','Value',...) creates a new CONTROLES or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Controles_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Controles_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Controles

% Last Modified by GUIDE v2.5 10-Jan-2017 22:44:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Controles_OpeningFcn, ...
                   'gui_OutputFcn',  @Controles_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
% End initialization code - DO NOT EDIT


% --- Executes just before Controles is made visible.
function Controles_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Controles (see VARARGIN)

% Choose default command line output for Controles
```

Figura 148. Código de la figura de controles (parte I)

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Controles wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Controles_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in botonCaudal1.
function botonCaudal1_Callback(hObject, eventdata, handles)
% hObject    handle to botonCaudal1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in botonCaudal2.
function botonCaudal2_Callback(hObject, eventdata, handles)
% hObject    handle to botonCaudal2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in botonCaudal3.
function botonCaudal3_Callback(hObject, eventdata, handles)
% hObject    handle to botonCaudal3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in botonNivel1.
function botonNivel1_Callback(hObject, eventdata, handles)
% hObject    handle to botonNivel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in botonNivel2.
function botonNivel2_Callback(hObject, eventdata, handles)
% hObject    handle to botonNivel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Figura 149. Código de la figura de controles (parte II)

```
% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over botonCaudal1.
function botonCaudal1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to botonCaudal1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

inpgui_sf([],[],[],'pushBtnDwnCaud1')


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over botonCaudal2.
function botonCaudal2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to botonCaudal2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

inpgui_sf([],[],[],'pushBtnDwnCaud2')
% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over botonCaudal3.
function botonCaudal3_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to botonCaudal3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

inpgui_sf([],[],[],'pushBtnDwnCaud3')


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over botonNivel1.
function botonNivel1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to botonNivel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

inpgui_sf([],[],[],'pushBtnDwnNiv1')
% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over botonNivel2.
function botonNivel2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to botonNivel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

inpgui_sf([],[],[],'pushBtnDwnNiv2')


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

Figura 150. Código de la figura de controles (parte III)

```
%---part 1 ---------------------%
%set initial values
ud.botonCaudal1 = 0 ;
ud.botonCaudal2 = 0 ;
ud.botonCaudal3 = 0 ;
ud.Nivel1 = 0;
ud.Nivel2 = 0;

set(hObject, 'UserData', ud) ; %store GUI initial values

%Store figure handle in block userdata - handle can then be used for
%manipulating the figure params
set_param(gcbh,'Userdata',hObject);

%-----end of part 1 --------------%
```

Figura 151. Código de la figura de controles (parte IV)

```
%  Nitin Skandan, 11-Aug-2011
% This function collects data from the GUI and passes to the model
% The editable portions are commented with %-NS- way
function [sys,x0,str,ts] = inpgui_sf(t,x,u,flag,Ts)
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(Ts);
  case 2,
    sys=mdlUpdate(t,x,u,Ts);
  case 3,
    sys = mdlOutputs(t,x,u); % Calculate outputs
  case { 1, 4, 9 },
    sys = [0 0 0 0 0];
  case 'pushBtnDwnCaud1', %Call coming from input gui when button pressed
    ud = get(gcbf,'UserData');
    %set(gcbo,'BackGroundColor','Yellow') ;    %change color
    set(gcbf,'WindowButtonUpFcn','inpgui_sf([],[],[],''pushBtnUp'');'); % specify the
button release function
     ud.botonCaudal1 = 1 ;   %update gui config
     set(gcbf,'Userdata',ud); %update gui confg data struct
   case 'pushBtnDwnCaud2', %Call coming from input gui when button pressed
     ud = get(gcbf,'UserData');
     %set(gcbo,'BackGroundColor','Yellow') ;    %change color
     set(gcbf,'WindowButtonUpFcn','inpgui_sf([],[],[],''pushBtnUp'');'); % specify the
button release function
     ud.botonCaudal2 = 1 ;   %update gui config
     set(gcbf,'Userdata',ud); %update gui confg data struct
   case 'pushBtnDwnCaud3', %Call coming from input gui when button pressed
     ud = get(gcbf,'UserData');
     %set(gcbo,'BackGroundColor','Yellow') ;    %change color
     set(gcbf,'WindowButtonUpFcn','inpgui_sf([],[],[],''pushBtnUp'');'); % specify the
button release function
```

Figura 152. Código que comunica la pantalla de controles con *Simulink* (parte I)

```
    ud.botonCaudal3 = 1 ;   %update gui config
    set(gcbf,'Userdata',ud); %update gui confg data struct
  case 'pushBtnDwnNiv1', %Call coming from input gui when button pressed
    ud = get(gcbf,'UserData');
    %set(gcbo,'BackGroundColor','Yellow') ;    %change color
    set(gcbf,'WindowButtonUpFcn','inpgui_sf([],[],[],''pushBtnUp'');'); % specify the
button release function
    ud.Nivel1 = 1 ;   %update gui config
    set(gcbf,'Userdata',ud); %update gui confg data struct
  case 'pushBtnDwnNiv2', %Call coming from input gui when button pressed
    ud = get(gcbf,'UserData');
    %set(gcbo,'BackGroundColor','Yellow') ;    %change color
    set(gcbf,'WindowButtonUpFcn','inpgui_sf([],[],[],''pushBtnUp'');'); % specify the
button release function
    ud.Nivel2 = 1 ;   %update gui config
    set(gcbf,'Userdata',ud); %update gui confg data struct
  case 'pushBtnUp',
    ud = get(gcbf,'UserData');
    chnd = findobj(gcbf,'Tag','pbuttn') ;
    set(chnd,'BackGroundColor',[0.831 0.816 0.784]) ;
    set(gcbf,'WindowButtonUpFcn','');
    ud.botonCaudal1 = 0 ;
    ud.botonCaudal2 = 0 ;
    ud.botonCaudal3 = 0 ;
    ud.Nivel1 = 0;
    ud.Nivel2 = 0;
    set(gcbf,'Userdata',ud);
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes(Ts)

% call simsizes for a sizes structure, fill it in and convert to a sizes array.
sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 5; %-NS Specify the number of outputs
sizes.NumInputs      = 0; %-NS Specify the number of inputs
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [0];
str = [];
ts  = [Ts 0];
% create the figure, if necessary
Controles;
% end mdlInitializeSizes
```

Figura 153. Código que comunica la pantalla de controles con *Simulink* (parte II)

```
function sys=mdlUpdate(t,x,u,Ts)
fig = get_param(gcbh,'UserData');
sys=x;


function sys = mdlOutputs(t,x,u)
 fig = get_param(gcbh,'UserData') ;
 if ishandle(fig),
%     chnd = findobj(fig,'Tag','spd_edt') ;
    ud = get(fig,'UserData') ;
%     set(chnd,'String', num2str(ud.VehSpd)) ;
    sys = [ud.botonCaudal1 ud.botonCaudal2 ud.botonCaudal3 ud.Nivel1 ud.Nivel2];
 else
     sys = [0 0 0 0 0] ;
 end

% end mdlOutputs
```

Figura 154. Código que comunica la pantalla de controles con *Simulink* (parte III)

```
function msfun_realtime_pacer(block)
% Help for Writing Level-2 M-File S-Functions:
%   web([docroot '/toolbox/simulink/sfg/f7-67622.html']
%   http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/sfg/f7-67622.html

% Copyright 2009, The MathWorks, Inc.

% instance variables
mySimTimePerRealTime = 1;
myRealTimeBaseline = 0;
mySimulationTimeBaseline = 0;
myResetBaseline = true;
myTotalBurnedTime = 0;
myNumUpdates = 0;

setup(block);
%% ------------------------------------------------------
    function setup(block)
        % Register the number of ports.
        block.NumInputPorts  = 0;
        block.NumOutputPorts = 0;

        % Set up the states
        block.NumContStates = 0;
        block.NumDworks = 0;

        % Register the parameters.
        block.NumDialogPrms     = 1; % scale factor
        block.DialogPrmsTunable = {'Nontunable'};

        % Block is fixed in minor time step, i.e., it is only executed on major
        % time steps. With a fixed-step solver, the block runs at the fastest
        % discrete rate.
        block.SampleTimes = [0 1];

        block.SetAccelRunOnTLC(false); % run block in interpreted mode even w/↙
Acceleration
```

Figura 155. Código de sincronización con reloj de sistema. Parte I

```
        % methods called during update diagram/compilation.
        block.RegBlockMethod('CheckParameters', @CheckPrms);

        % methods called at run-time
        block.RegBlockMethod('Start', @Start);
        block.RegBlockMethod('Update', @Update);
        block.RegBlockMethod('SimStatusChange', @SimStatusChange);
        block.RegBlockMethod('Terminate', @Terminate);
    end

%%
    function CheckPrms(block)
        try
            validateattributes(block.DialogPrm(1).Data, {'double'},{'real', 'scalar',
'>', 0});
        catch %#ok<CTCH>
             throw(MSLException(block.BlockHandle, ...
                 'Simulink:Parameters:BlkParamUndefined', ...
                 'Enter a number greater than 0'));
         end
    end

%%
    function Start(block)
        mySimTimePerRealTime = block.DialogPrm(1).Data;
        myTotalBurnedTime = 0;
        myNumUpdates = 0;
        myResetBaseline = true;
        if strcmp(pause('query'),'off')
             fprintf('%s: Enabling MATLAB PAUSE command\n', getfullname(block.
BlockHandle));
             pause('on');
        end
    end
%%
    function Update(block)
        if  myResetBaseline
            myRealTimeBaseline = tic;
            mySimulationTimeBaseline = block.CurrentTime;
            myResetBaseline = false;
        else
            if isinf(mySimTimePerRealTime)
                return;
            end
            elapsedRealTime = toc(myRealTimeBaseline);
            differenceInSeconds = ((block.CurrentTime - mySimulationTimeBaseline) /
mySimTimePerRealTime) - elapsedRealTime;
            if differenceInSeconds >= 0
                pause(differenceInSeconds);
                myTotalBurnedTime = myTotalBurnedTime + differenceInSeconds;
                myNumUpdates = myNumUpdates + 1;
            end
        end
    end
```

Figura 156. Código de sincronización con reloj de sistema. Parte II

```
%%
    function SimStatusChange(block, status)
        if status == 0,
            % simulation paused
            fprintf('%s: Pausing real time execution of the model (simulation time = %
g sec)\n', ...
                getfullname(block.BlockHandle), block.CurrentTime);
        elseif status == 1
            % Simulation resumed
            fprintf('%s: Continuing real time execution of the model\n', ...
                getfullname(block.BlockHandle));
            myResetBaseline = true;
        end
    end
%%
    function Terminate(block)
        if myNumUpdates > 0
            fprintf('%s: Average idle real time per major time step = %g sec\n', ...
                getfullname(block.BlockHandle),  myTotalBurnedTime / myNumUpdates);
        end
    end

end
```

Figura 157. Código de sincronización con reloj de sistema. Parte III

Simulink

En este apartado se exponen los modelos de *Simulink* que controlan los diferentes aspectos del programa.

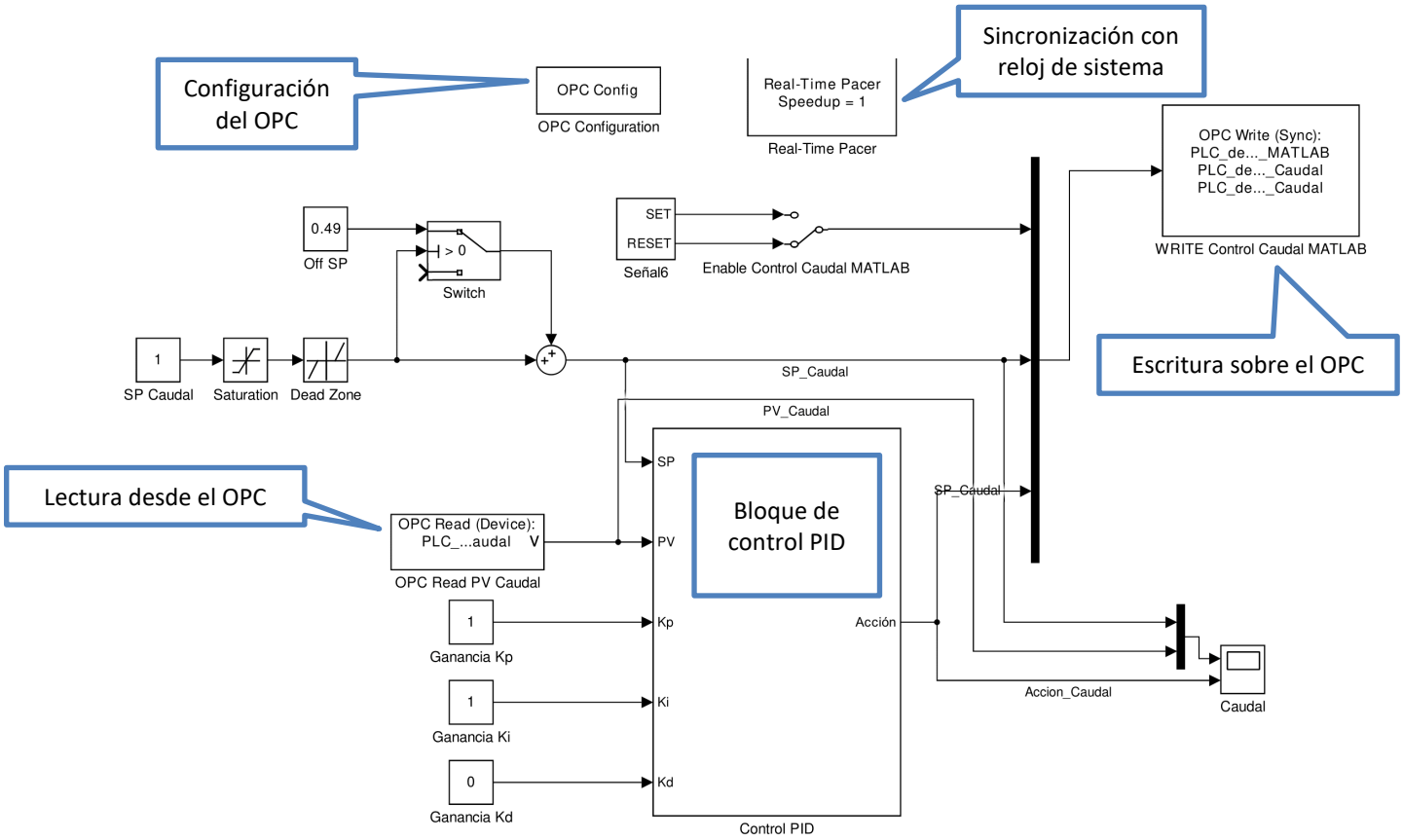


Figura 158. Control PID de Caudal tipo 1. Parte I

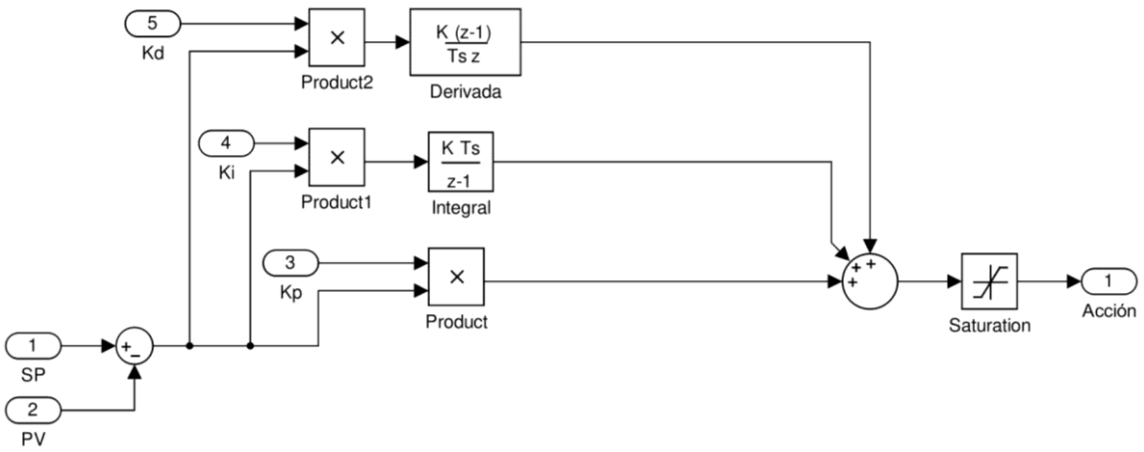


Figura 159. Distribución interna de bloque PID

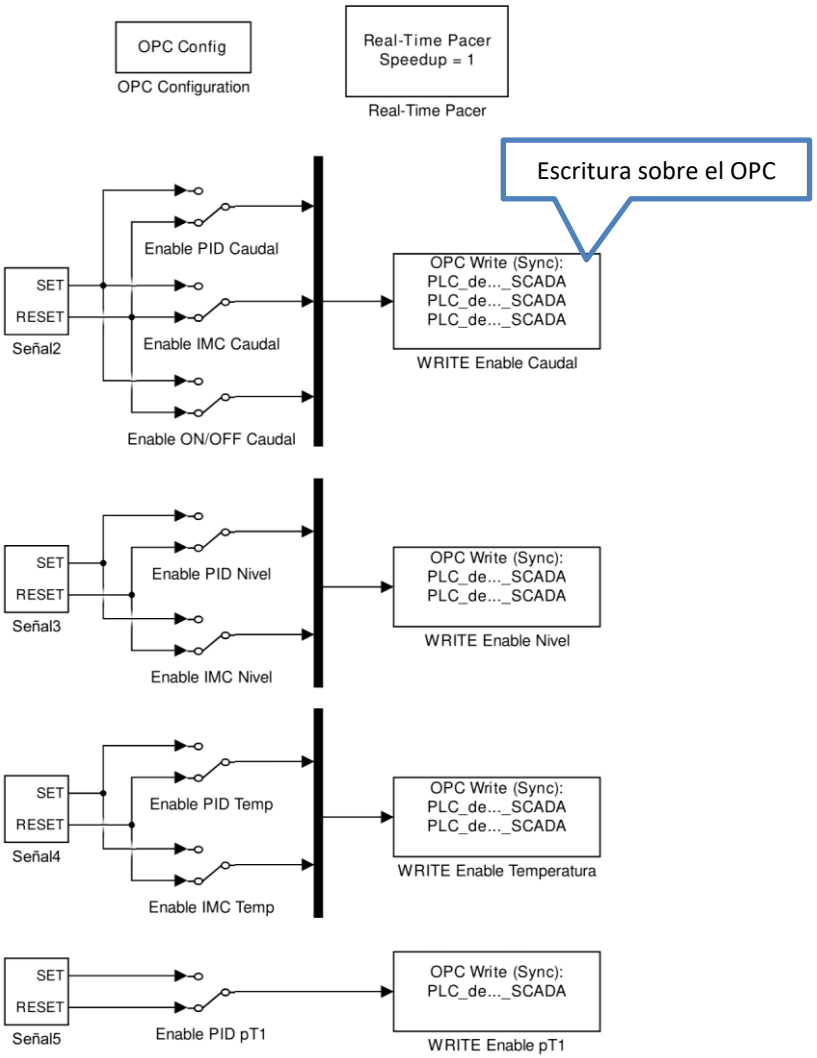


Figura 160. Controles PID e IMC

Controles alternativos para activar/desactivar tanto los PID como los IMC del PLC.

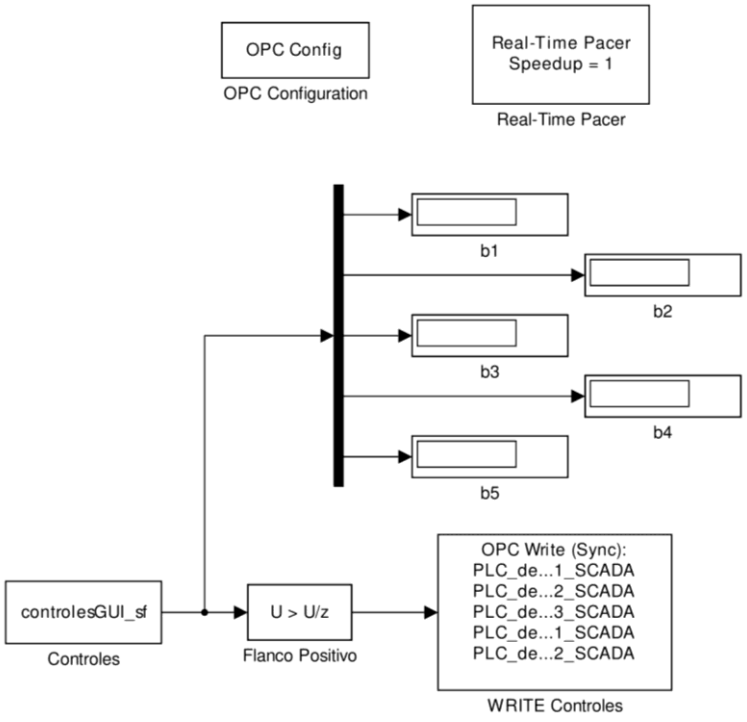


Figura 161. Gestión de los botones de la pantalla de controles

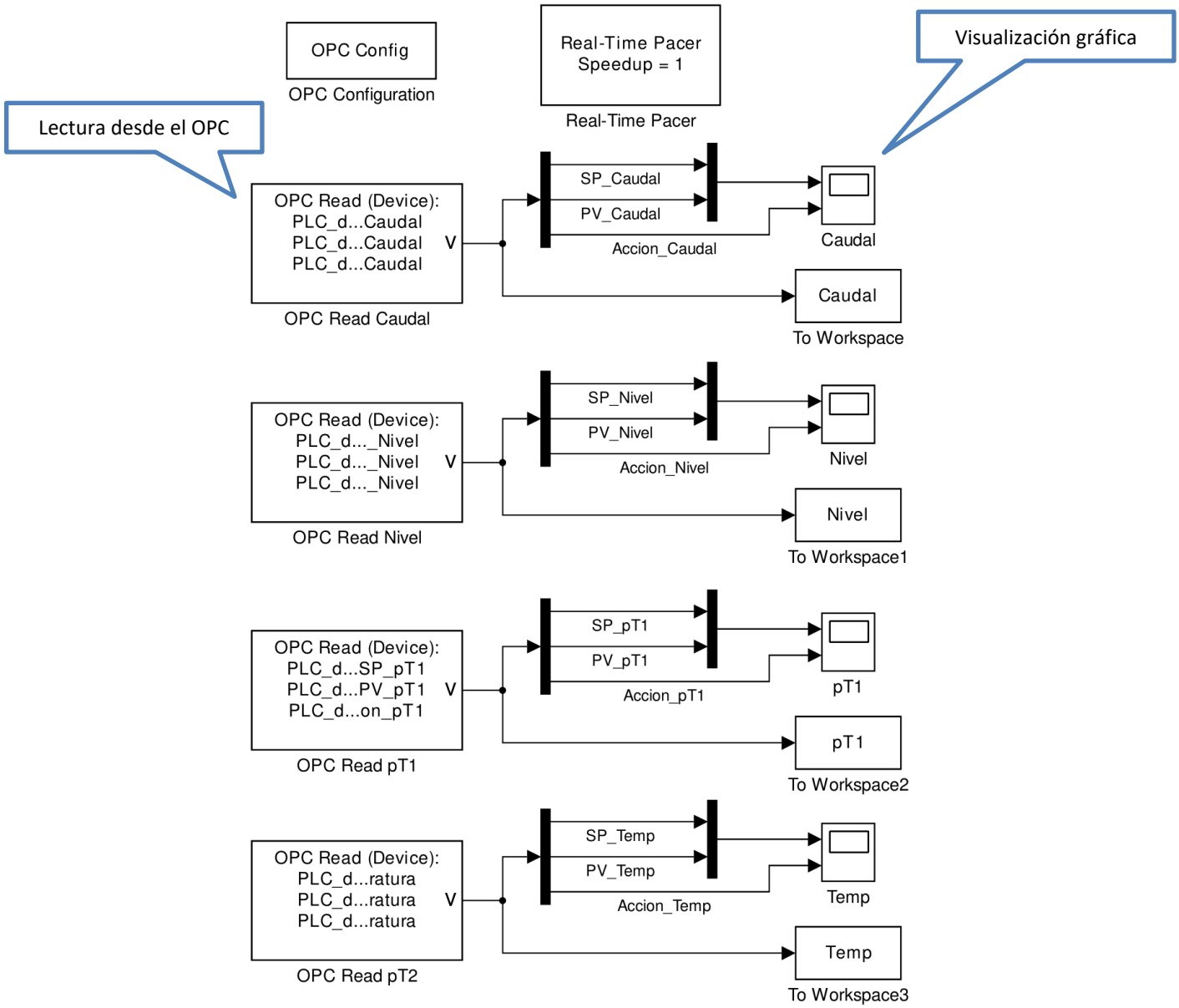


Figura 162. Visualización gráfica de las variables de los reguladores del PLC

Figuras

Por último, tenemos la pantalla de los pulsadores que permiten alternar entre las diferentes combinaciones de control de caudal y nivel.

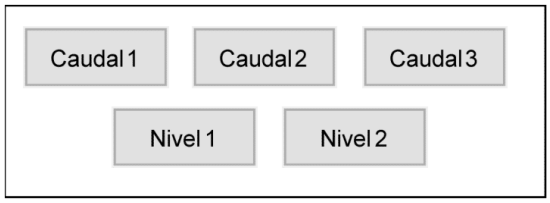


Figura 163. Pantalla de controles

Anexo E. Hojas de características de elementos de la estación

En este Anexo se muestran las hojas de características de los diversos elementos que componen la estación (sensores, actuadores y PLC), que fueron ya explicados en el *Capítulo 2. Descripción de la estación de depósitos* de esta memoria.

PLC TSX-Premium P57 2634M

Characteristics and performance

Types of processor				TSX P57 0244M	TSX P57 104M	TSX P57 1634M	TSX P57 154M	TSX P57 204M	TSX P57 2634M *(1)*	TSX P57 254M
Maximum configuration	No. of racks	4/6/8 slots		1	4			16		
		12 slots		1	2			8		
	Max. no. of slots for modules			12	32			128		
Functions	Max. no in-rack *(3)*	Discrete I/O		192/256 *(2)*	512			1024		
		Analog I/O		12	24			80		
		Process control channels		–				10 (up to 30 parameterizable simple loops)		
				Programmable loops via EFB control blocks (with Unity Pro Large and Extra Large)						
		Application-specific channels, number		4	8			24		
		Application-specific channels, type		Counter, axis control, weighing and serial links (Modbus, Uni-Telway and asynchronous)						
	Integrated connections	Ethernet		–		1	–		1	–
		Fipio manager		–			1 (63 agents)	–		1 (127 agents)
		Serial link		1 link with 2 connectors (TER and AUX) 19.2 Kbit/s						
	Max. no. of connections	Network (Ethernet, Fipway, Ethway, Modbus Plus)		1		1 integrated Ethernet port	1		1 integrated Ethernet port	1
		AS-Interface bus		1	2			4		
		CANopen or Modbus Plus bus		1 integrated CANopen	1		1 Modbus Plus only	1		
		INTERBUS or Profibus DP bus		–				1		
Memories	Maximum capacity	Without PCMCIA card	**Kb**	96 prog. + data				160 prog. + data		192 prog.+ data
		With PCMCIA card	**Kb**	128 prog. 96 data	224 prog. 96 data			768 prog. 160 data		768 prog. 192 data
		Data storage	**Kb**	256				16,384 (limited to 8192 with current PCMCIA cards)		
	Maximum size of object zones	Located internal bits (%Mi)	**bits**	4096				8132		
		Located internal data	**Kb**	64 for internal words %Mei 64 for constant words %Kei						
		Unlocated internal data	**Kb**	Elementary EDT and derived DDT data: 32 Kb				Elementary EDT and derived DDT data: 64 Kb		
				DFB and EFB function blocks: Size per instance: 64 Kb, unlimited number of instances *(7)*						
Application structure	Tasks	Master		1	1			1		
		Fast		1	1			1		
		Auxiliary		–	–			–		
		Event-triggered		32 (1 of which has priority)				64 (1 of which has priority)		
Execution time for one instruction	Without PCMCIA card	Boolean	µs	0.19	0.19			0.19		
		On word or fixed-point arithmetic	µs	0.25	0.25			0.25		
		On floating points	µs	1.75...2.60 *(7)*	1.75...2.60 *(7)*			1.75...2.60 *(7)*		
	With PCMCIA card	Boolean	µs	0.25	0.25			0.21		
		On word or fixed-point arithmetic	µs	0.50	0.50			0.42		
		On floating points	µs	1.75...2.60 *(7)*	1.75...2.60 *(7)*			1.75...2.60 *(7)*		
No. of Kinstructions executed every ms	Without PCMCIA card	100% Boolean	**Kinst/ ms**	4.76	4.76			4.76		
		65% Boolean and 35% fixed arithmetic	**Kinst/ ms**	3.71	3.71			3.71		
	With PCMCIA card	100% Boolean	**Kinst/ ms**	3.10	3.10			3.70		
		65% Boolean and 35% fixed arithmetic	**Kinst/ ms**	2.10	2.10			2.53		
System overhead	Tasks	Master	**ms**	1.00	1.00			1.00		
		Fast	**ms**	0.30	0.30			0.30		

Figura 164. Tabla de características del PLC TSX-Premium P57 2634M

Sensores

Presostato digital ISE40-01-62L

		ZSE40F (Presión combinada)	ZSE40 (Presión de vacío)	ISE40 (Presión positiva)
Rango de presión nominal		-100 a 100 kPa	0 a -101,3 kPa	0 a 1 MPA
Rango de presión de trabajo/Rango de presión de disparo		-100 a 100 kPa	10 a -101,3 kPa	-0,1 a 1 MPa
Presión de prueba		500kPa		1,5 MPa
Resolución de la presión de regulación Nota 1)	kPa	0.1		—
	MPa	—		0.001
	kgf/cm2	0.001		0.01
	bar			0.01
	psi	0.02	0.01	0.1
	mmHg	1		—
		0.1		—
Fluido aplicable		Aire, gas no corrsivo/no inflamable		
Tensión de alimentación				
Consumo de corriente		55mA o menos		
Salida digital		2 salidas NPN o PNP Corriente de carga m·x.: 80mA Tensión m·x. aplicada: 30 VDC(con salida NPN) Tensión residual:1V o menos (con corriente de carga 80mA)		
Repetitividad		±0,2% Fondo de escala o menos		
Histéresis	Modo de histéresis	Variable		
	Modo ventana comparativa	Fijo (3 dígitos) Nota 4)		
Tiempo de respuesta (con función antivibración)		2,5ms o menos (con función antivibración: selección 25ms, 192ms y 768ms		
Protección de cortacircuitos de salida		Sí		
Unidad de indicación		LED 3 1/2 dígitos (Ciclo de muestra: 5 veces/seg.)		
Precisión de display		±2% Fondo de escala ±1 dígito o menos (a temperatura ambiente de 25±3?C		
Indicador		LED verde (Se enciende cuando se activa OUT1), LED rojo (Se enciende cuando se activa OUT2)		
Salida analógica Nota 2)		Tensión de salida: 1 a 5V 5% fondo de escala o menos (en rango de presión nominal) Linealidad: 1% fondo de escala o menos Impedancia de salida: 1kW aprox.	Tensión de salida: 1 a 5V 2,5% fondo de escala o menos (en rango de presión nominal) Linealidad: 1% fondo de escala o menos Impedancia de salida: 1kW aprox.	
Entrada autodiagnóstico Nota 3)		Sin tensión (Reed o de estado sólido), entrada 5ms o m·s		
Resistencia medioambiental	Protección	IP65 Nota 6)		
	Rango temperatura ambiente	En funcionamiento: 0 a 50 , Almacenado: -10 a 60 (sin condensación ni congelación)		
	Rango de humedad ambiental	En funcionamiento/Almacenado: 35 a 85% (sin condensación)		
	Resistencia dieléctrica	1000VAC durante 1min. entre los cables y el cuerpo		
	Resistencia al aislamiento	50M o m·s (a 500 VDC) entre los cables y el cuerpo		
	Resistencia a vibraciones	10 a 500Hz a una amplitud de 1,5mm o una aceleración de 98m/s2 en las direcciones X, Y, Z, 2 horas cada una (Desactivado)		
	Resistencia a impactos	980m/s2 en las direcciones X, Y, Z, tres veces cada una (Desactivado)		
Características de temperatura		En un rango de temperatura de 0 a 50 , 2% fondo de escala o menos de presión medida a 25		
Conexión		01: R1/8, M5x0.8, T1: NPT1/8, M5x0.8, W1: Rc1/8, WF1: G1/8 C4: con 4 conexiones instant·neas, C6: con 6 conexiones instant·neas, M5: roscas hembra M5		
Cable		Cable de 5 hilos óleoresistente para cargas pesadas (0,15mm2)		
Peso		Modelo 01/T1 aprox. 60g, modelo W1 aprox. 80g, modelo C4/C6/M5 aprox. 92g (cada uno con cables de 0,6m incluidos)		

Figura 165. Especificaciones preostato digital ISE40-01-62L

Caudalímetro PF2W504-F03-2

Model		PF2W504	PF2W520	PF2W540	PF2W511
Measured fluid		Water			
Detection type		Karman vortex			
Rated flow range		0.5 to 4 L/min	2 to 16 L/min	5 to 40 L/min	10 to 100 L/min
Operating pressure range		0 to 1 MPa			
Withstand pressure		1.5 MPa			
Operating fluid temperature		0 to 50°C			0 to 50°C
Accuracy Note 1)		±5% F.S.			±3% F.S.
Repeatability Note 1)		±3% F.S.			±1% F.S. (connected with PF2W33□) ±3% F.S. (connected with PF2W2□□)
Temperature characteristics		±2% F.S. (15 to 35°C, 25°C reference), ±3% F.S. (0 to 50°C, 25°C reference)			
Output specifications Note 2)	Output for display unit	Pulse output, N channel, open drain, output for monitor unit PF2W3□□. (Specifications: Maximum load current of 10 mA; Maximum applied voltage of 30 V)			
	Analog output	Voltage output 1 to 5 V Accuracy: ±5%F.S., Min. load impedance: 100 kΩ (Output impedance: 1 kΩ)			
		Current output 4 to 20 mA Accuracy: ±5%F.S., Max. load impedance: 300 Ω or less (at 12 VDC), 600 Ω or less (at 24 VDC)			
Power supply voltage		12 to 24 VDC ±10%			
Current consumption (No load)		20 mA or less			
Environment	Enclosure	IP65			
	Operating temperature range	Operating: 0 to 50°C, Stored: −25 to 85°C (with no freezing and condensation)			
	Withstand voltage	1000 VAC for 1 minute between terminals and housing			
	Insulation resistance	50 MΩ or more (500 VDC measured via megohmmeter) between terminals and housing			
	Noise resistance	1000 Vp-p, Pulse width 1 μs, Rise time 1 ns			
Weight Note 3)		410 g	470 g	650 g	1,100 g
Port size (Rc, NPT, G)		3/8	3/8, 1/2	1/2, 3/4	3/4, 1

Figura 166. Especificaciones caudalímetro PF2W504-F03-2

Sensor Nivel. Presión diferencial PSE550-AC2

Model		PSE550	PSE550-28
Rated differential pressure range		0 to 2 kPa	
Operating pressure range		−50 to 50 kPa (Note)	
Extension analog output range		−0.2 to 0 kPa	—
Proof pressure		65 kPa	
Applicable fluid		Air/Non-corrosive gas/Non-flammable gas	
Power supply voltage		12 to 24 VDC±10%, Ripple (p-p) 10% or less (with power supply polarity protection)	
Current consumption		15 mA or less	—
Output specification		Analog output: 1 to 5 VDC (within rated differential pressure range) 0.6 to 1 VDC (with extension analog output range) Output impedance: Approx. 1 kΩ	Analog output: 4 to 20 mADC (within rated differential pressure range) Allowable load impedance: 500 Ω or less (at 24 VDC) 100 Ω or less (at 12 VDC)
Accuracy (Operating temperature at 25°C)		±1% F.S. or less (with rated pressure range), ±3% F.S. or less (with extension analog output range)	
Linearity		±0.5% F.S. or less	
Repeatability		±0.3% F.S. or less	
Indication light		Orange light is turned on. (When energized)	
Environmental resistance	Enclosure	IP40	
	Operating temperature range	Operating: 0 to 50°C, Stored: −20 to 70°C (No freezing or condensation)	
	Operating humidity range	Operating/Stored: 35 to 85% RH (No condensation)	
	Withstand voltage	1000 VAC, 50/60 Hz for 1 minute between live parts and case	
	Insulation resistance	50 MΩ or more between live parts and case (at 500 VDC Mega)	
	Vibration resistance	10 to 150 Hz at whichever is smaller of 1.5 mm amplitude or 100 m/s² acceleration, in X, Y, Z directions, for 2 hours each (De-energized)	
	Impact resistance	300 m/s² in X, Y, Z directions, 3 times each (De-energized)	
Temperature characteristics		±3% F.S. or less (Based on 25°C)	
Port size		ø4.8 (ø4.4 in the end) resin piping (Applicable to I.D. ø4 air tubing)	
Wetted parts material		Resin pipe: Nylon, Piston area of sensor: Silicon	
Sensor cable		Oil proof heavy-duty vinyl cable (ellipse), 3 cores, 2.7 x 3.2, 3 m Conductor area: 0.15 mm², Insulator O.D.: 0.9 mm	Oil proof heavy-duty vinyl cable (ellipse), 2 cores, 2.7 x 3.2, 3 m Conductor area: 0.15 mm², Insulator O.D.: 0.9 mm
Mass	With sensor cable	75 g	
	Without sensor cable	35 g	
Standards		Compliant with CE marking, UL (CSA)	

Figura 167. Especificaciones sensor de nivel por presión diferencial PSE550-AC2

Sonda de temperatura PTC100 con cabeza amplificadora

Figura 168. Especificaciones Sonda de temperatura PTC100 con cabeza amplificadora

Sensores capacitivos

Datos eléctricos		
Alimentación		DC PNP
Tensión de alimentación	[V]	10...36 DC
Consumo	[mA]	< 20
Clase de protección		II
Protección contra inversiones de polaridad		sí
Salidas		
Función de salida		normalmente abierto / normalmente cerrado programable
Caída de tensión	[V]	< 2,5
Corriente de salida	[mA]	200
Protección contra cortocircuitos		pulsada
Resistente a sobrecargas		sí
Frecuencia de conmutación	[Hz]	10
Rango de detección		
Alcance	[mm]	12

Figura 169. Especificaciones Sensores capacitivos

Actuadores

Electroválvulas monoestables 3/2 VKF334Y-5DZ-01F / VCW21-5D-4-02F

Valve model		Operating pressure range (MPa)	Port size	Flow characteristics						Mass (g)	
				1 → 2 (P → A)			2 → 3 (A → R)			Grommet	DIN terminal
				C [dm³/(s·bar)]	b	Cv	C [dm³/(s·bar)]	b	Cv		
Body ported	VKF33²₃	0 to 0.7	M5 x 0.8 Rc 1/8	0.67	0.10	0.15	0.41	0.39	0.11	80 (1)	90 (1)
	VKF33²₃Y			0.56	0.13	0.13	0.32	0.25	0.09		
	VKF33²₃E			0.56	0.13	0.13	0.32	0.25	0.09		
	VKF33²₃V	−101.2 KPa to 0.1		0.67	0.10	0.15	0.41	0.39	0.11		
	VKF33²₃W			0.56	0.13	0.13	0.32	0.25	0.09		
Base mounted (With sub-plate)	VKF334	0 to 0.7		0.68	0.13	0.15	0.59	0.31	0.14	120	130
	VKF334Y			0.56	0.13	0.13	0.32	0.25	0.09		
	VKF334E			0.56	0.13	0.13	0.32	0.25	0.09		
	VKF334V	−101.2 kPa to 0.1		0.68	0.13	0.15	0.59	0.31	0.14		
	VKF334W			0.56	0.13	0.13	0.32	0.25	0.09		

Figura 170. Especificaciones electroválvulas monoestables

Electroválvula proporcional Burkert 6022

Port Connection [inch]	Orifice [inch]	Body Material	C_v	Pressure Range [PSI]	Power Consumption [W]	Maximum Coil Current [mA]	Controller Signal	Weight [lbs.]	Item No.		
									Controller	Valve	Valve with Controller
1/4 NPT	5/64	Brass	0.12	0 - 115	8	300	4 - 20 mA	1.1	060 644 J	456 967 U	702 618 M
1/4 NPT	5/64	Brass	0.12	0 - 115	8	300	0 - 10 V	1.1	060 459 R	456 967 U	702 619 N
1/4 NPT	5/32	Brass	0.38	0 - 28	8	300	4 - 20 mA	1.1	060 644 J	456 968 D	702 620 K
1/4 NPT	5/32	Brass	0.38	0 - 28	8	300	0 - 10 V	1.1	060 459 R	456 968 D	702 621 G
3/8 NPT	5/32	Brass	0.47	0 - 55	15	530	4 - 20 mA	2.0	060 644 J	456 965 S	702 623 A
3/8 NPT	5/32	Brass	0.47	0 - 55	15	530	0 - 10 V	2.0	060 459 R	456 965 S	702 624 B
3/8 NPT	1/4	Brass	0.82	0 - 28	15	530	4 - 20 mA	2.0	060 644 J	456 966 T	702 625 C
3/8 NPT	1/4	Brass	0.82	0 - 28	15	530	0 - 10 V	2.0	060 459 R	456 966 T	702 626 D
1/4 NPT	5/64	Stainless Steel	0.12	0 - 115	8	300	4 - 20 mA	1.1	060 644 J	458 402 H	704 241 M
1/4 NPT	5/64	Stainless Steel	0.12	0 - 116	8	300	0 - 10 V	1.1	060 459 R	458 402 H	704 242 N
1/4 NPT	5/32	Stainless Steel	0.38	0 - 28	8	300	4 - 20 mA	1.1	060 644 J	458 403 A	704 243 P
1/4 NPT	5/32	Stainless Steel	0.38	0 - 28	8	300	0 - 10 V	1.1	060 459 R	458 403 A	704 244 Q

Figura 171. Especificaciones electroválvula proporcional Burkert 6022

Regulador de presión proporcional ITV1030-01F2BN2-Q

Modelo		ITV101□ (Nota 10) / ITV201□ / ITV301□	ITV103□ (Nota 10) / ITV203□ / ITV303□	ITV105□ (Nota 10) / ITV205□ / ITV305□
Mínima presión de alimentación		Presión de regulación +0.1 MPa		
Máxima presión de alimentación		0.2 MPa	1.0 MPa	
Rango de presión de regulación (Nota 1)		0.005 a 0.1 MPa	0.005 a 0.5 MPa	0.005 a 0.9 MPa
Alimentación	Tensión	24 VDC ± 10%, 12 a 15 VDC		
	Consumo de corriente	Tensión de alimentación 24 VDC: 0.12 A o menos Tensión de alimentación 12 a 15 VDC: 0.18 A o menos		
Señal de entrada	Tipo corriente (Nota 7)	4 a 20 mA, 0 a 20 mA (tipo COM+)		
	Tipo tensión	0 a 5 VDC, 0 a 10 VDC		
	Entrada preajustada	4 puntos		
Entrada de carga	Tipo corriente	250 Ω máx. (Nota 6)		
	Tipo tensión	Aprox. 6.5 kΩ		
	Entrada preajustada	Aprox. 2.7 kΩ		
Señal de salida (salida de monitor) (Nota 3)	Salida analógica	1 a 5 VDC (impedancia de carga: 1 kΩ o más) 4 a 20 mA (tipo COM+) (Impedancia de carga: 250 Ω o menos) Precisión de salida en el rango de ±6% (extensión completa)		
	Salida digital	Salida de colector abierto NPN: Máx. 30 V, 30 mA Salida de colector abierto PNP: Máx. 30 mA		
Linealidad		En el rango de ±1% (extensión completa)		
Histéresis		En el rango de 0.5% (extensión completa)		
Capacidad de repetición		En el rango de ±0.5% (extensión completa)		
Sensibilidad		En el rango de 0.2% (extensión completa)		
Características de temperatura		En el rango de ±0.12% (extensión completa)/°C		
Indicación de presión de salida (Nota 4)	Precisión	±3% (extensión completa)		
	Unidad mínima	MPa: 0.01, kgf/cm²: 0.01, bar: 0.01, PSI: 0.1 (Nota 5), kPa: 1		
Temperatura ambiente y de fluido		0 a 50°C (sin condensación)		
Protección		IP65		
Peso (Nota 9)	ITV10□□	Aprox. 250 g (sin opciones)		
	ITV20□□	Aprox. 350 g (sin opciones)		
	ITV30□□	Aprox. 645 g (sin opciones)		

Figura 172. Especificaciones regulador de presión proporcional ITV1030-01F2BN2-Q

Motobomba 12V para procesos continuos

Rendimiento bomba (50Hz) por hora desde aprox.	150,00 l
Rendimiento bomba (50Hz) por hora hasta aprox.	300,00 l
Altura bombeo con 50 Hz aprox. (Al máx.)	0,50 m
Consumo eléctrico (50 Hz) aprox.	5,00 Vatios

Figura 173. Especificaciones Motobomba 12V

Células Peltier 12v – 73 Watt Qmax

Catalog Number	Imax (Amps)	Th = 25°C			N	Dimensions (mm)			
		Qmax[1] (Watts)	Vmax (Volts)	ΔTmax (°C)		A	B	C	D[2]
CP1.4-11-045L	8.5	6.0	1.33	65	11	10	15	10	3.3
CP1.4-17-045L	8.5	9.2	2.06	65	17	15	15	15	3.3
CP1.4-31-045L	8.5	16.8	3.75	65	31	20	20	20	3.3
CP1.4-35-045L	8.5	19.0	4.24	65	35	15	30	15	3.3
CP1.4-51-045L	8.5	28.9	6.18	65	51	9.5	62	9.5	3.3
CP1.4-71-045L	8.5	38.5	8.60	65	71	30	30	30	3.3
CP1.4-127-045L	8.5	72.0	15.40	65	127	40	40	40	3.3
CP2-17-10L	9.0	10.3	2.06	68	17	22	22	22	5.6
CP2-31-10L	9.0	18.8	3.75	68	31	30	30	30	5.6
CP2-49-10L	9.0	29.7	5.93	68	49	36	36	36	5.6
CP2-71-10L	9.0	43.1	8.60	68	71	44	44	44	5.6
CP2-127-10L	9.0	77.1	15.40	68	127	62	62	62	5.6
CP2-17-06L	14.0	16.0	2.06	67	17	22	22	22	4.6
CP2-31-06L	14.0	29.3	3.75	67	31	30	30	30	4.6
CP2-49-06L	14.0	46.2	5.93	67	49	36	36	36	4.6
CP2-71-06L	14.0	67.0	8.60	67	71	44	44	44	4.6
CP2-127-06L	14.0	120.0	15.40	67	127	62	62	62	4.6
CP2.8-31-06L	24.0	50.2	3.75	67	31	40	40	40	5.0
CP5-31-10L	39.0	81.5	3.75	68	31	55	55	55	5.8
CP5-31-06L	60.0	125.0	3.75	67	31	55	55	55	4.9

Figura 174. Tabla de características de algunos modelos de Células Peltier

Anexo F. Seminario sobre diseño y programación de una aplicación de supervisión (SCADA)

Objetivos

Programar una aplicación sencilla de adquisición de datos, control y supervisión de un proceso industrial sencillo. Dadas las capacidades y la complejidad del software utilizado, el seminario se plantea como una introducción al trabajo con software SCADA.

Trabajo previo

Puesto que no es necesario realizar la programación del autómata que realiza el control del caudal, el único trabajo previo requerido es haberse leído el guion.

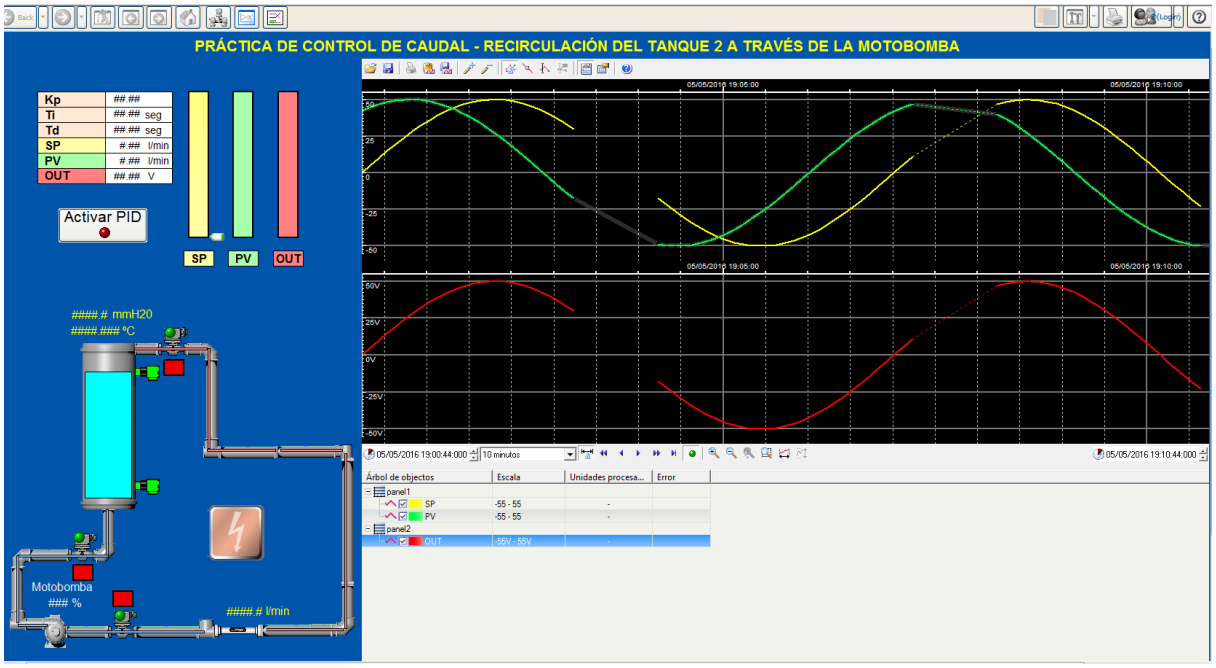


Fig. 0. Pantalla de control de la aplicación a realizar

Descripción

La práctica consiste en desarrollar una aplicación SCADA para el control de un sistema. El control del sistema se realiza desde un autómata (quien lee el estado del sistema y finalmente es el que actúa sobre el mismo). Se podrán enviar consignas al sistema a través del SCADA siempre que no se esté en condición de alarma, en cuyo caso será el autómata quien tomará el control. El sistema es un depósito con agua sobre el que se va a realizar un control de caudal, recirculando el agua a través de la motobomba situada a su salida.

El programa del autómata no hay que desarrollarlo, ya está cargado en el autómata y ejecutándose durante la práctica.

El sinóptico de la aplicación SCADA debería quedar algo similar al representado en la figura 0.

Los elementos que aparecen en dicha figura pueden agruparse en dos categorías:

a) Visualizadores, que muestran información del estado del sistema: habilitación del control, baliza luminosa de emergencia, nivel de agua en el depósito, temperatura del depósito, estado de las válvulas de entrada y salida del tanque, así como la de la motobomba, consigna de caudal, medida del caudal, acción sobre la motobomba, y medida del nivel.
b) Controles, que permiten actuar sobre él en determinadas condiciones: botón control ON/OFF (permite comandar el control de caudal), campos numéricos para introducir la nueva consigna de caudal (que se dará en l/min) y parámetros del controlador y puntero deslizante para controlar la consigna de caudal.

Realización de la aplicación

Crear una nueva aplicación

Para lanzar la aplicación de desarrollo nos vamos al menú de inicio, *Todos los programas*, *Automática*, *Vijeo Citect 7.30*, donde encontrarás el *Explorador de Vijeo Citect.*

Esto lanza tres ventanas, siendo la principal *Explorador de Citect*. Menú *Archivo* → *Nuevo proyecto*. Pon tu nombre de pila (para identificar claramente tu proyecto entre los varios existentes), la descripción es opcional, asegúrate que está desmarcada la opción *¡¡¡¡Crear proyecto basado en proyecto inicial!!!!!*, y *Aceptar*.

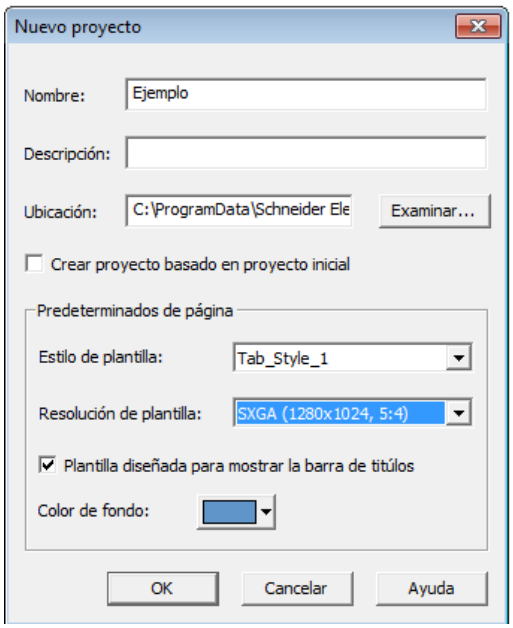


Fig. 1. Nuevo proyecto.

Creación del Cluster

Para poder seguir con la configuración, es necesario crear un cluster que vaya registrando y almacenando los elementos que vamos a necesitar (Tags, alarmas, …). En el árbol de tu proyecto doble clic en *Comunicaciones*, y doble clic en *Clusters*. Eliminamos el existente si existe (*Eliminar)*, y creamos uno nuevo con el nombre *cluster*, *Añadir.*

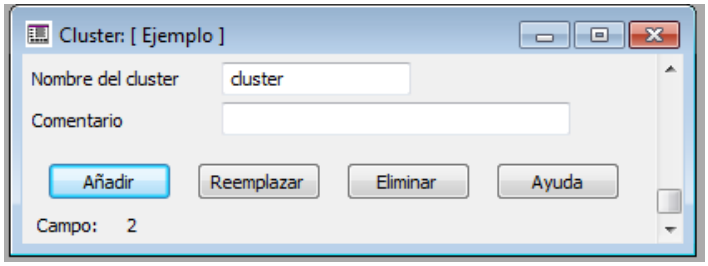
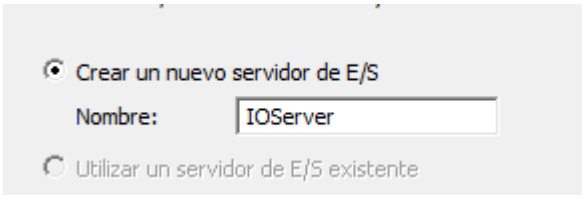


Fig. 2. Cluster.

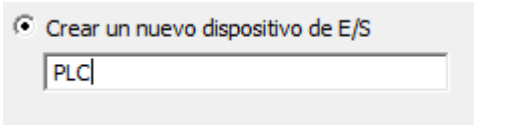
Configuración de comunicaciones y dispositivos

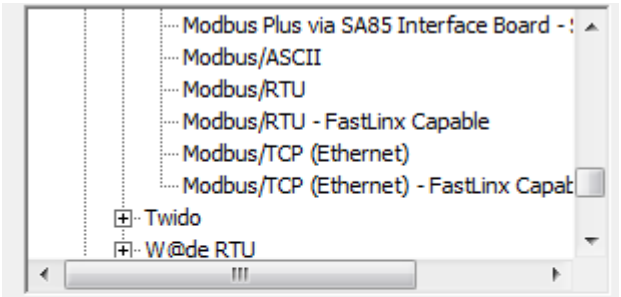
Ahora vamos a configurar un servidor de comunicaciones y un dispositivo de E/S.

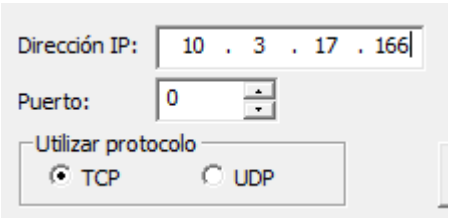
En el árbol de tu proyecto doble clic en *Comunicaciones*, y doble clic en *Configuración rápida de dispositivo de E/S*. Primero le damos a *Siguiente*, marcamos la opción *Crear un nuevo servidor de E/S*, introducimos el nombre *IOServer*, *Siguiente.*



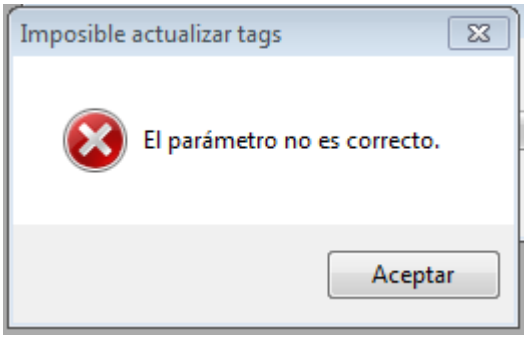
A continuación, crea un nuevo dispositivo de E/S con el nombre *PLC, Siguiente*, *Dispositivo de E/S externo*, *Siguiente*. Ahora hay que seleccionar el driver a utilizar con dicho dispositivo: *Schneider-Electric → Quantum→ Modbus/TCP (Ethernet), Siguiente*. El número IP del autómata es "10.3.17.166", y el puerto de comunicaciones por defecto debería funcionar. Seleccionar protocolo *TCP* y *Siguiente*, *Siguiente*, *Finalizar*.







Si os sale un error *Imposible actualizar tags. El parámetro no es correcto* le dais a *Aceptar*.



Dirección de Red

Debemos también configurar la dirección de Red de nuestro ordenador. Para ello, nuevamente en el árbol de tu proyecto doble clic en *Comunicaciones*, y doble clic en *Direcciones de Red*. Introducimos de nombre *net* y en *Dirección* ponemos "127.0.0.1", *Añadir.*

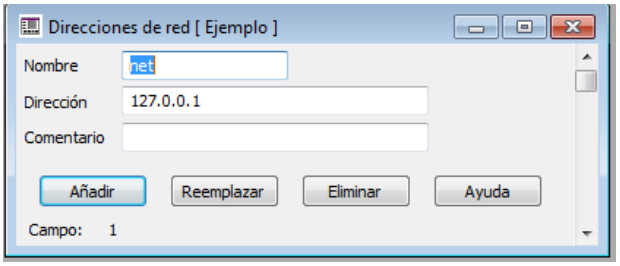


Fig. 3. Dirección de Red.

Debemos cambiar un par de cosas en nuestro proyecto. En el árbol de tu proyecto doble clic en *Comunicaciones*, y doble clic en *Servidor de E/S*. En el campo *Nombre del cluster* abrid el desplegable, y seleccionar el nombre que hemos definido antes, clic en *Reemplazar*. Hacemos lo mismo en el campo *Direcciones de red*, seleccionando la red *net* que acabamos de definir.

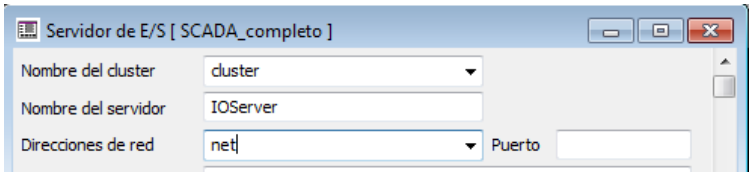


Fig. 4. Servidor E/S.

Definición de usuarios

Para poder seguir trabajando, deberemos definir un usuario para nuestra aplicación, ya que, aunque no es necesario el programa nos lo exige. En el árbol de tu proyecto doble clic en *Sistema*, y doble clic en *Usuarios*. Escribimos en el campo *Nombre de usuario* VUESTRO NOMBRE en minúsculas, y le damos a *Añadir*.

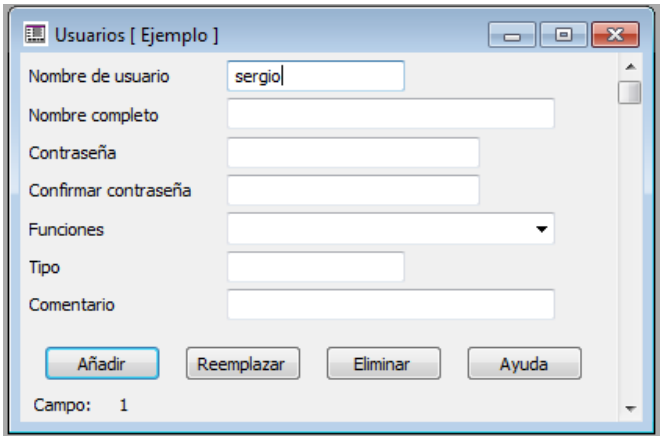


Fig. 5. Usuario.

Configuración del computador

Una vez creado un proyecto, hay que indicar al software que queremos que ejecute nuestro proyecto (de entre los muchos que pueda haber desarrollados en nuestro computador). Primero deberemos compilar el proyecto: ve a la ventana *Editor de proyectos de Citect* pulsando en el botón, y compila el proyecto con el botón o con la opción *Compilar* del menú *Archivo* (o ALT+F10). Si nos sale *Nada que compilar. ¿Todavía desea compilar?*, le damos a *Sí*. Si no hay errores, debería decir *La compilación ha sido exitosa.*

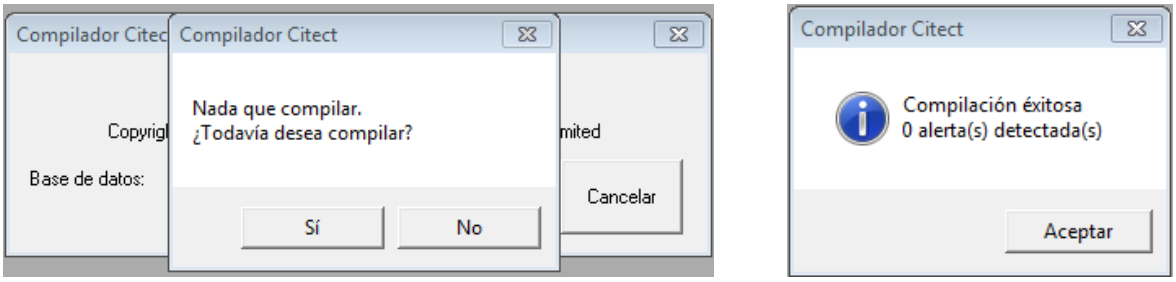


Fig. 6. Compilación.

De vuelta en el *Explorador de Citect*, ejecutamos el asistente de instalación: *Herramientas* → *Asistente de instalación del ordenador*, donde se elegirán las siguientes opciones: *Configuración exprés, Siguiente, Nombre del proyecto: 'El nombre de vuestro proyecto'*, *Siguiente, Servidor y cliente de control (Multiproceso)*, *Siguiente, Desconectada, Siguiente, Contraseña: seminario*, *Siguiente, Finalizar.*

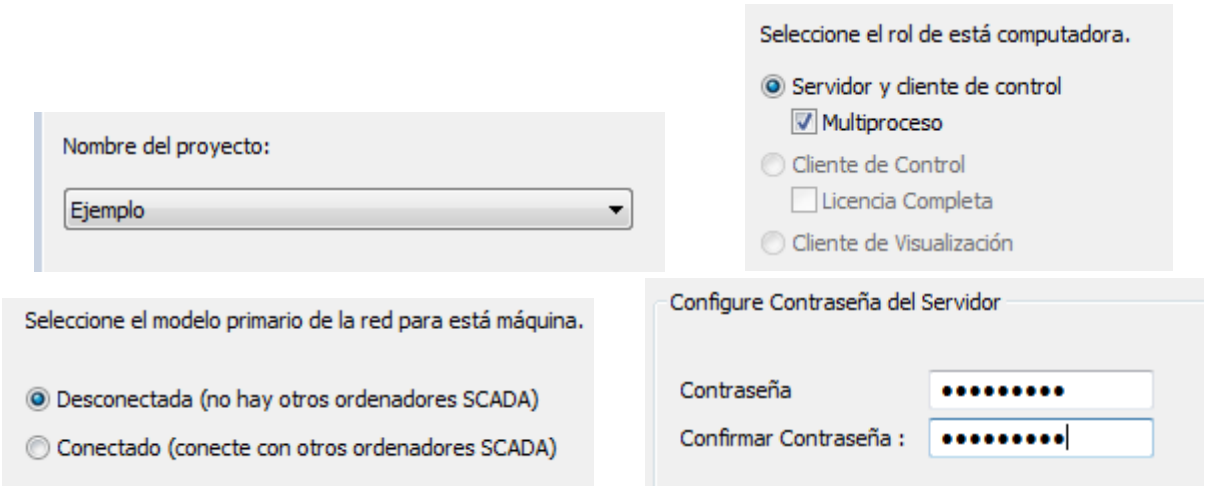


Fig. 7. Asistente de instalación.

Definición de Tags

Para probar que realmente tenemos comunicación con el autómata, vamos a leer la temperatura actual del agua del depósito. Hay que definir una *tag*, o variable de la base de datos que es gestionada por el SCADA en tiempo real, en la cual se recoge toda la información del proceso supervisado.

Para ahorrar tiempo, se os habrá proporcionado un fichero *varible.dbf*, que contiene ya definidos los *tags* que vamos a usar. Pero antes, para refrescaros la memoria, vamos a crear un *tag* para la temperatura (aunque ya esté creado). En el árbol de tu proyecto clic en *Tags*, doble clic en *Tags de variable*. Aquí podremos ver los *tags* definidos y crear nuevos. Define la primera *tag* según la figura 8. *Añadir.*

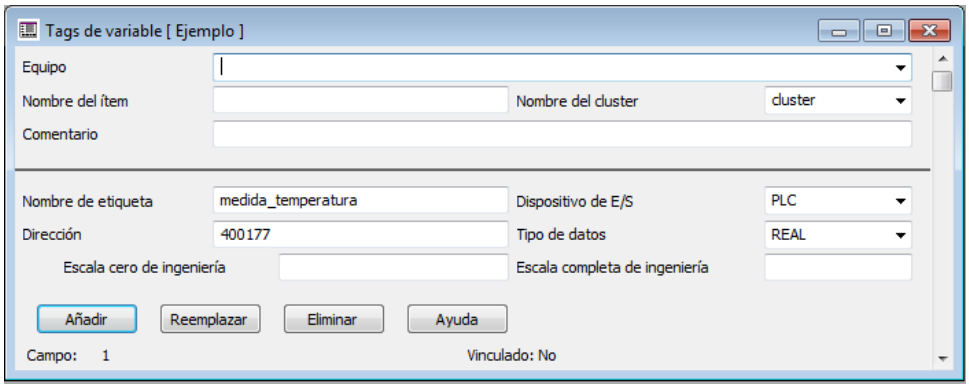


Fig. 8. Creación de un Tag.

El direccionamiento en Unity es diferente al de Vijeo Citect. Unity utiliza el estándar IEC 61131, pero en cambio Vijeo Citect utiliza el direccionamiento Modbus. Además, desde Vijeo Citect sólo se pueden leer y escribir palabras de memoria (%MWi) y bits de memoria (%Mi). No se pueden leer el resto de variables %I; %Q, %K, %S… Tampoco se pueden leer o escribir variables que no tengan dirección asignada. La equivalencia entre direccionamientos viene indicada en la figura 9.

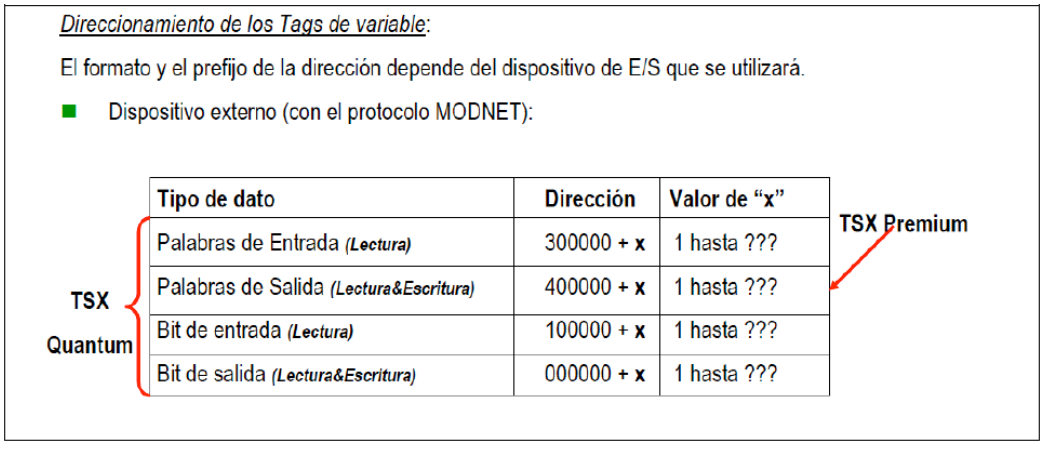


Fig. 9. Direccionamiento de variables.

Para importar los *tags* ya creados, vamos a nuestro proyecto y le damos con el botón derecho desde el árbol de proyectos, y luego clic en *Abrir carpeta de proyectos.* Antes de importar la base de datos, cerramos el *Explorador de Citect* para evitar problemas. Pegamos y reemplazamos el fichero *varible.dbf* proporcionado sobre la carpeta que hemos abierto.

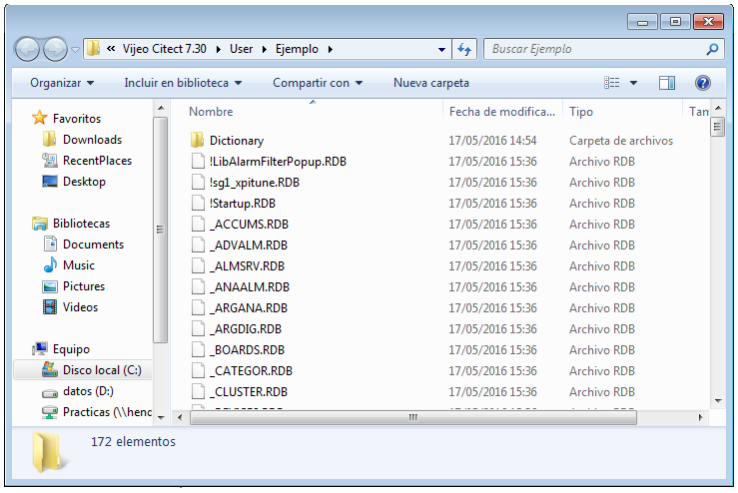


Fig. 10. Carpeta del proyecto.

Cerramos la carpeta y volvemos a abrir el programa. Para que la importación sea correcta, nos vamos al *Editor de proyectos de Citect* → *Archivo* → *Empacar*. Para comprobar que los *tags* se han cargado correctamente, accedemos a los *tags* como hemos hecho antes, y veremos que tenemos los necesarios para realizar nuestra aplicación.

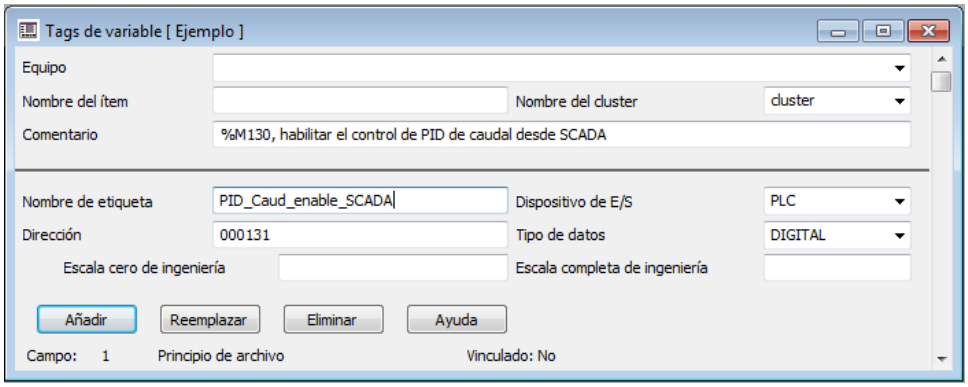


Fig. 11. Variables.

Prueba de la primera aplicación

Para poder probar la lectura de la temperatura debemos realizar una página que incluya dicha *tag*, para lo cual procedemos a crear nuestra primera página: en el árbol de tu proyecto, doble clic en *Gráficos*, doble clic en *Páginas*, doble clic en *Crear una nueva página*, y seleccionar como se ve en la figura 12. Se damos a *OK* y aparecerá nuestra primera página en blanco.

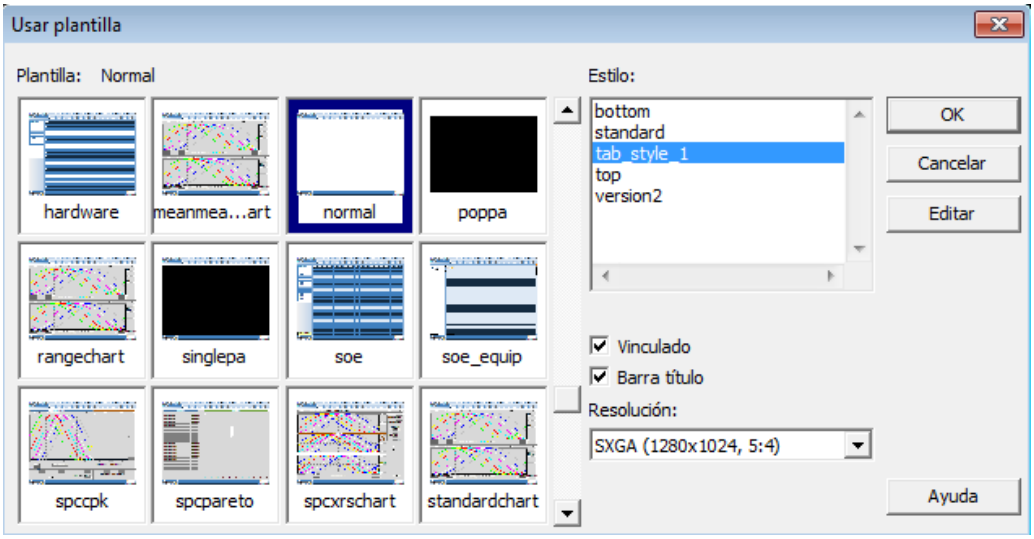


Fig. 12. Creación de la primera página.

Selecciona la herramienta '##' (*Número)* de la paleta y posiciona el objeto haciendo clic en la página, con lo que saldrá el cuadro de la figura 13.

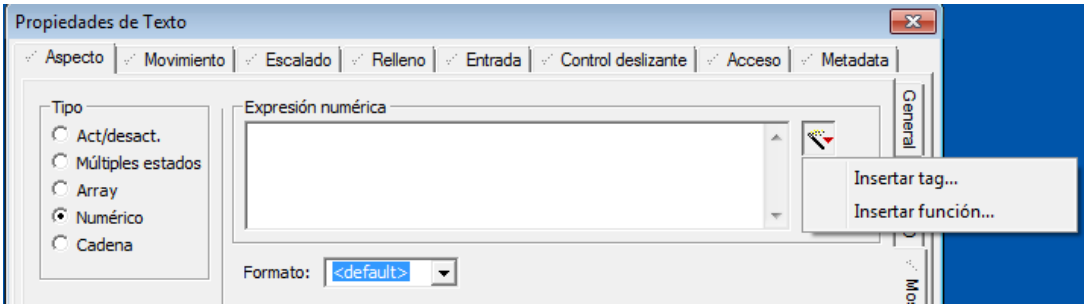


Fig. 13. Visualización de un valor numérico.

El asistente te permite seleccionar una tag para ser visualizada: *PV_Temp_SCADA.*

Fig. 14. Selección del tag.

En la pestaña vertical *General* (a la derecha de la ventana de la figura 13) puedes seleccionar el tamaño de letra y el color (18 puntos y color amarillo suelen ir bien sobre ese fondo azul). Una vez le damos a *Aceptar*, tendría que quedar algo parecido a la figura 15.

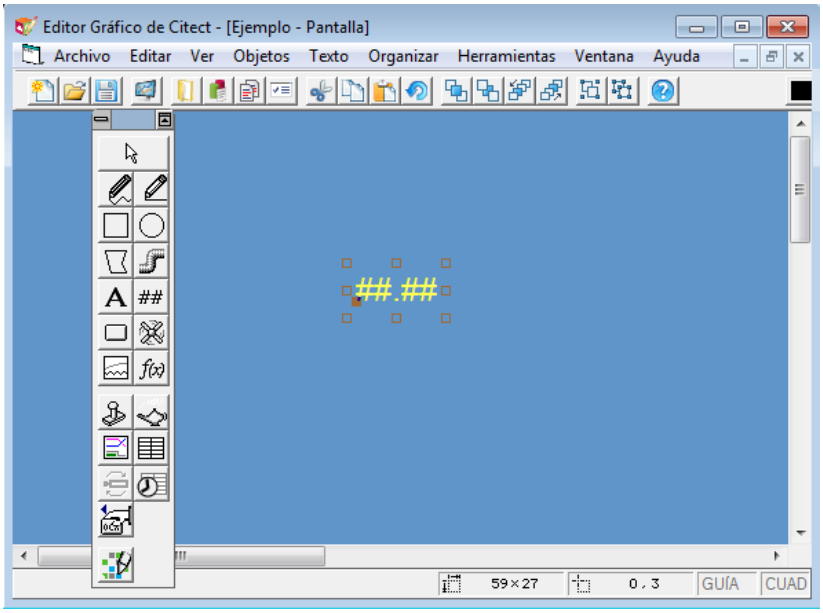


Fig. 15. Objeto número.

Falta guardar la página: Archivo → Guardar, como nombre de la página *seminario_SCADA*, asegurándoos de que en la lista *proyecto* tenéis seleccionado el nombre de vuestro proyecto (en mi caso el proyecto se llama *ejemplo)*.

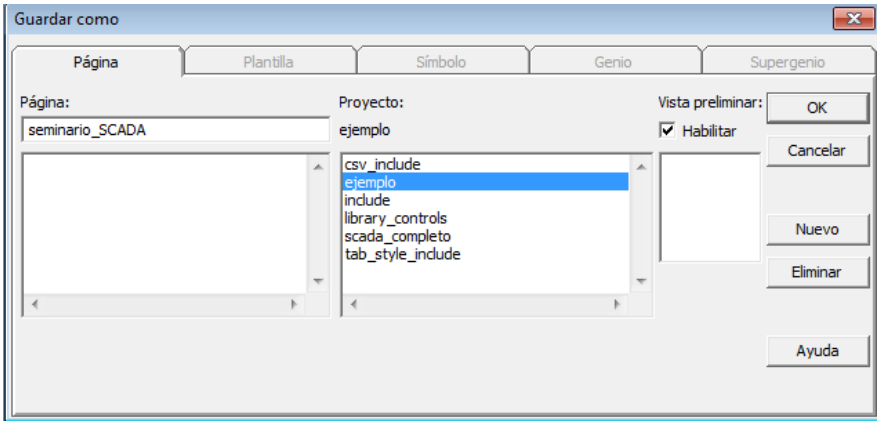


Fig. 16. Guardar una página.

Nos vamos de nuevo a la ventana *Editor de proyectos de Citect* y compilamos nuestra aplicación como hemos hecho antes. Nuevamente, si nos sale *Nada que compilar. ¿Todavía desea compilar?*, le damos a *Sí*. Si hay algún error, lo corregimos antes de seguir.

Ha llegado el momento de probar nuestra primera aplicación SCADA: usa el botón *Ejecutar proyecto*  , que encontrarás en cualquiera de las tres ventanas de la aplicación (o con la opción *Ejecutar* del menú *Archivo*).

Te saldrá un mensaje diciendo que no encuentra la llave de protección (dispositivo USB que permite ejecutar el proyecto sin límite de tiempo). Acepta ejecutar en modo demostración, con lo que tendrás el control sólo durante 15-20 minutos (suficiente para jugar un rato con el equipo).

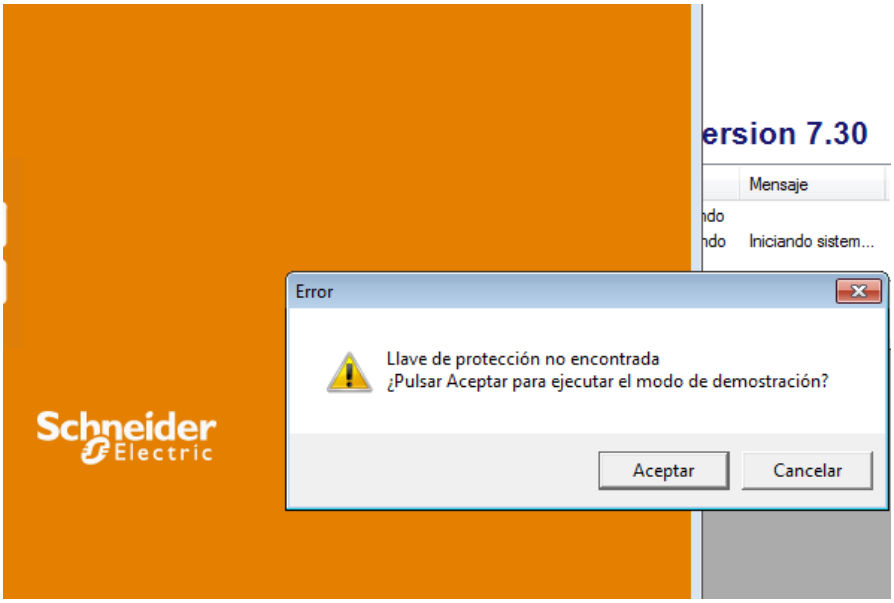


Fig. 17. Ejecutando el programa.

Tras unos instantes aparecerá una ventana como la de la figura 18: en la sección a *Pages* seleccionamos la página que hemos creado antes. Deberías visualizar la temperatura del depósito en tiempo real (si no es así, consulta con tu profesor de prácticas).

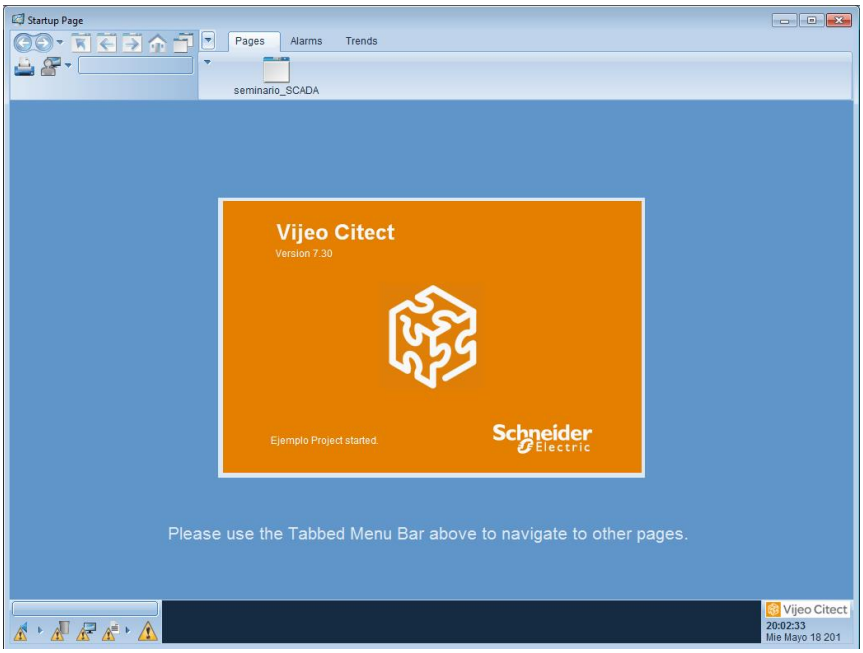


Fig. 18. Ventana de ejecución.

Creación de la pantalla de control

Creación de botones. Para crear el botón *Activar PID*, inserta un botón (búscalo en la paleta de herramientas), asigna el texto *Activar PID* (no pierdas tiempo intentando modificar la fuente del texto), haz clic en la pestaña *Entrada* y escribe en la zona de inserción de *Arriba* el comando: *Toggle(PID_Caud_enable_SCADA)* (para la selección de la *tag* puedes ayudarte del asistente, como antes).

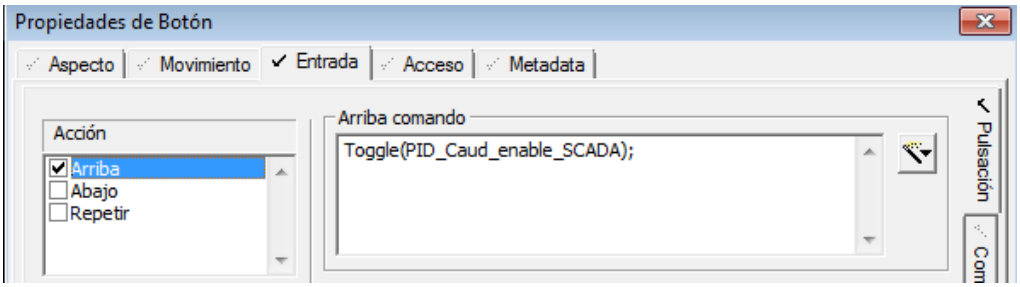


Fig. 19. Creación del botón.

Visualización de variables digitales. Tanto los indicadores circulares de la pantalla como los rectangulares son controlados mediante variables digitales. Los primeros se hacen clicando en la herramienta *Grupo de símbolos* y después en un lugar en la pantalla (figura 20). El *tag* a asociar para el botón *Activar PID* es *PID_Caud_enable_SCADA* al igual que en el botón, pero esta vez con el objetivo de ver cuándo está activado. En la opción '*Símbolo act:*' seleccionaremos el color del indicador para cuando el *tag* valga 1 en el autómata, y de igual manera en '*Símbolo desact:*' para cuando el *tag* valga 0. Nos aseguraremos que en la opción *Tipo* esté seleccionado el modo *Act/desact*.

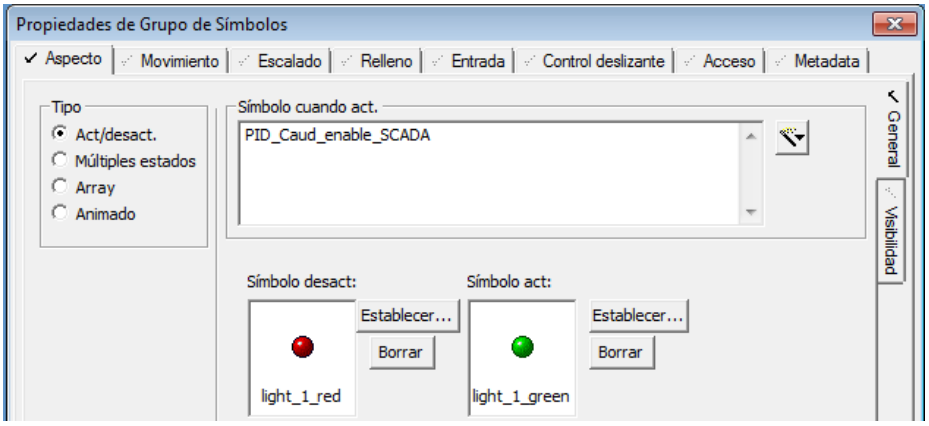


Fig. 20. Tag digitales.

Otra forma de visualizar condiciones es el uso de objetos es que sólo aparezcan en pantalla al cumplirse alguna condición. Vamos a probarlo con el símbolo del "Rayo" (figura 21), que nos servirá para indicar que la alarma está activa si dicho símbolo se visualiza. Añade dicho símbolo desde la herramienta *Pegar símbolo*  , bilbioteca *dangersigns*, simbolo *DangerSign3*.

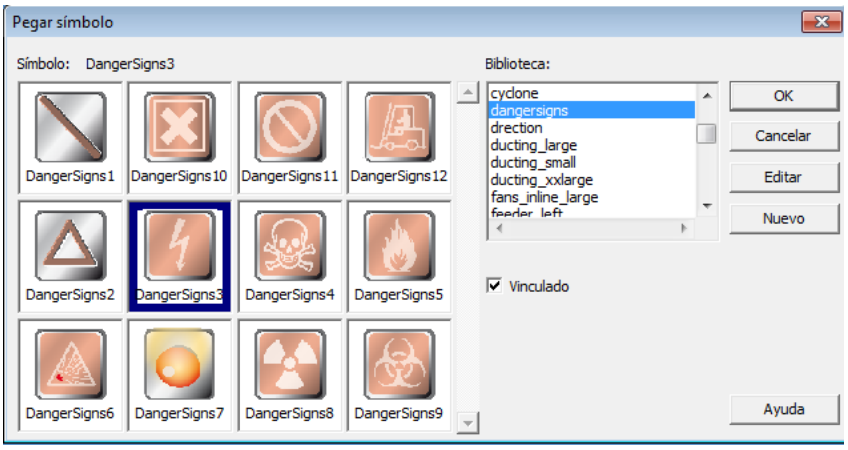


Fig. 21. Tag digitales.

Una vez que lo tengamos situado, doble clic sobre él y vamos a la pestaña *Aspecto → Visibilidad* (panel derecho). Añadimos la condición de que esté oculto cuando no haya alarma, tal y como se ve en la figura 22.

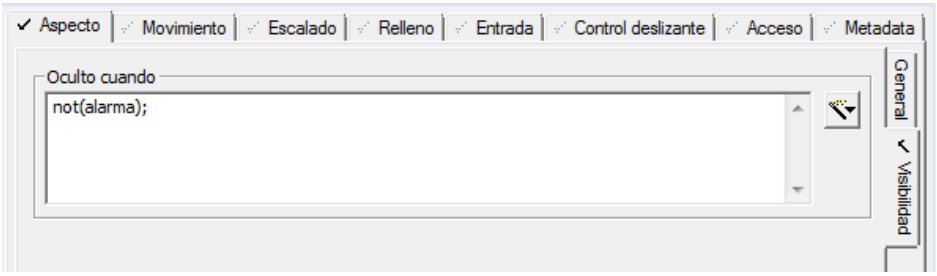


Fig. 22. Tag digitales.

Dibujo del depósito. Usa la herramienta *Pegar símbolo* de nuevo para añadir una imagen de un depósito usando la biblioteca *tanks_cylindrical* (por ejemplo, un tank_tall).

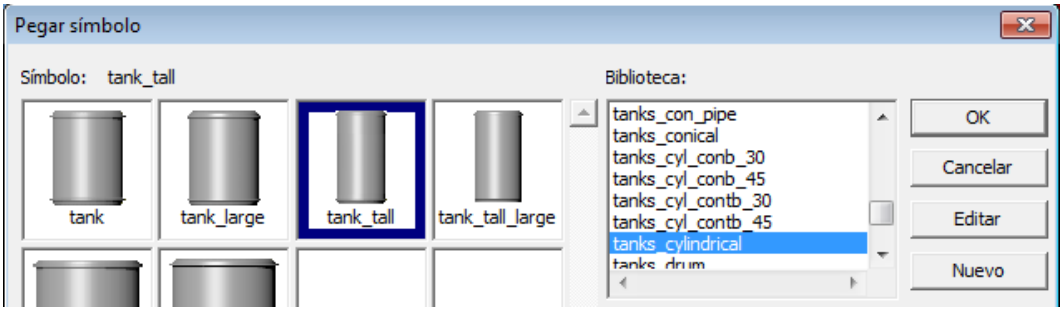


Fig. 23. Depósito.

Sobre él, pon un rectángulo que permitirá visualizar el nivel de líquido, con las siguientes propiedades: en la pestaña *Aspecto*, marcar la casilla *Relleno*, y elegir un color azulado (con el que se simulará el nivel de agua), y en la pestaña *Relleno* (pestaña vertical *Nivel*) asociarle la *tag PV_Niv_SCADA*, especificar rango mínimo de 0.0 y máximo de 200.0 (marcando la casilla *Especificar rango*), dirección de relleno hacia arriba, y como color de fondo transparente.

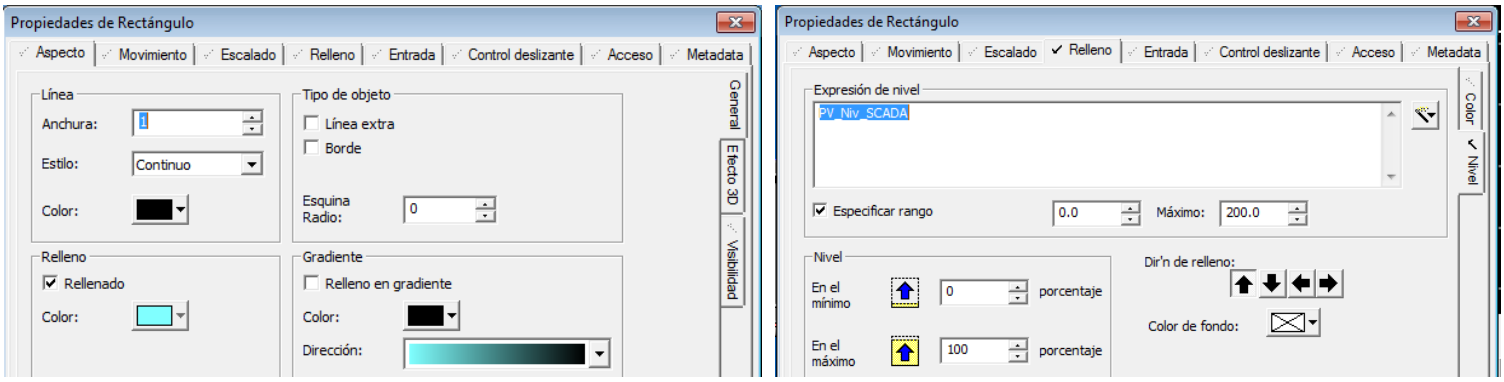


Fig. 24. Visualización del nivel.

Introducción y visualización de variables analógicas. El cuadro que aparece a la izquierda en la Fig.0 permite visualizar las siguientes variables: constante proporcional (Kp), contante de tiempo integral (Ti), contante de tiempo proporcional (Td), consigna de caudal (SP), valor actual del caudal (PV), y acción en Voltios sobre la motobomba (OUT) respectivamente. También permite cambiar los parámetros del PID (Kp, Ti, Td) y la consigna (SP). Para crearlos, seguimos el mismo procedimiento de antes para la visualización de la temperatura, con la siguiente relación de *tags*: *kp_PID_Caud* → Kp; *Ti_PID_Caud* → Ti; *Td_PID_Caud* → Td; *SP_Caud_SCADA* → SP; *PV_Caud_SCADA* → PV; *acción_PID_Caud* →OUT.

Ahora vamos a añadir la función para poder introducir nuevos parámetros mediante un teclado numérico: doble clic sobre el objeto *Número* de la Kp, pestaña *Entrada*, subpestaña vertical *Comandos de teclado*. Escribimos *LBUTTON_DBL* en el campo *Secuencia de teclas*, e introducimos el código de la figura 25 en el panel *LBUTTON_DBL comando*.

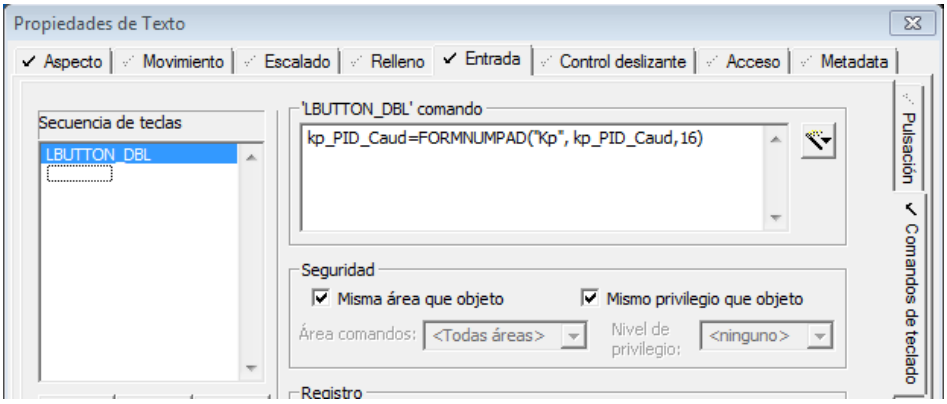


Fig. 25. Configuración del teclado.

Ahora repetimos el proceso con el resto de *tags* que vayamos a modificar durante la ejecución. Con todo ello, se consigue utilizar un único objeto para introducir y visualizar la consigna de caudal y los parámetros del PID. Antes de pasar al punto siguiente, termina poner en la pantalla todos los objetos *Número* de la aplicación según la figura 0.

La configuración del *Número* correspondiente al SP se hace un poco diferente debido a la configuración del autómata, y para poder poner un control deslizante más adelante. Hacemos la configuración según las figuras 26 y 27, tanto en la visualización como en el teclado.

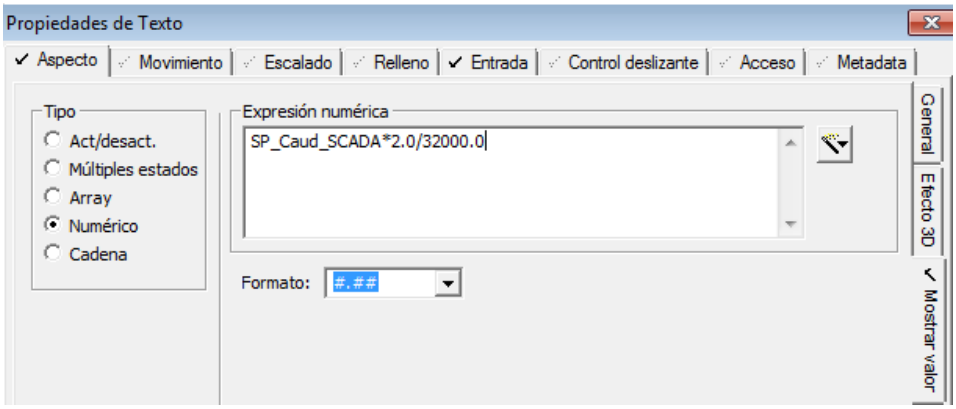


Fig. 26. Configuración del SP. Aspecto.

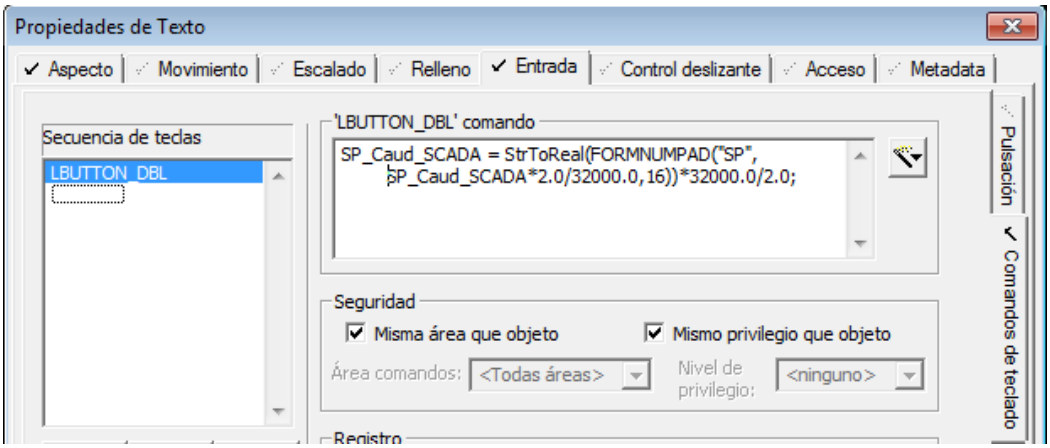


Fig. 27. Configuración del SP. Entrada.

PRECAUCIÓN: El código de la figura 27 debe estar todo en la misma línea. Se ha puesto así en la imagen para una mejor interpretación.

Para las barras verticales SP, PV, y OUT seguimos el mismo procedimiento que hemos hecho para el rectángulo de nivel del depósito, con los mismos *tags* utilizados en los objetos *Número* de estas 3 variables en los siguientes rangos: 0.0-32000.0 para SP, 0.0-2.0 para PV, y 0.0-10.0 para OUT.

Control mediante Punteros Deslizantes (Sliders). Ahora vamos a aprender otra manera de introducir valores mediante un objeto deslizante (puntero del borde derecho de la barra SP). Para hacerlo, usa la herramienta *Pegar símbolo* de nuevo para añadir una imagen de un puntero, por ejemplo, el de la figura 28:

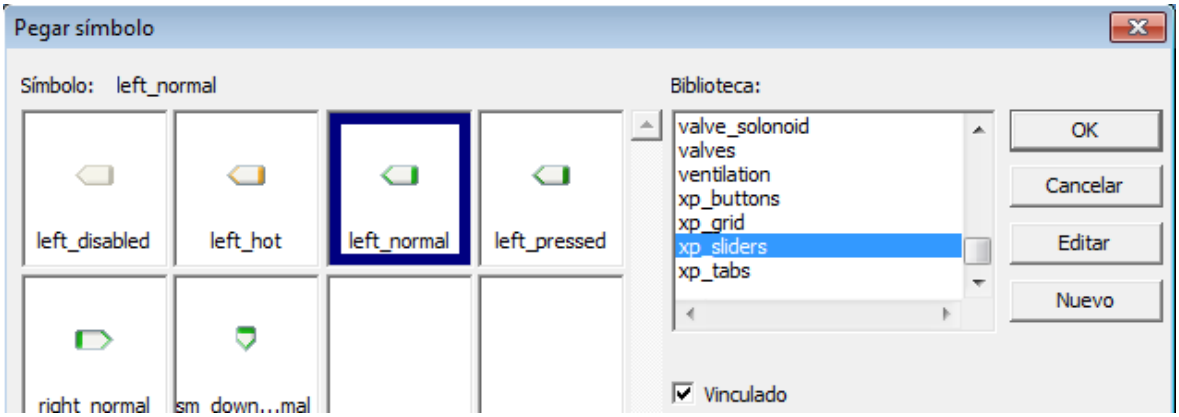


Fig. 28. Selección del Slider.

Lo situamos en la parte inferior de la barra. Ahora tenemos que saber el alto en pixeles de dicha barra para la configuración del Slider. Hacemos clic sobre la barra y nos fijamos en un pequeño recuadro en la parte inferior derecha de la pantalla (figura 29).

Fig. 29. Ancho x Alto.

El segundo número nos dice el alto en píxeles de la barra, en nuestro caso 203 píxeles. Ahora damos doble clic en el Slider, pestaña *Control deslizante*, subpestaña derecha *Vertical*, donde asignaremos el *tag* que se actualizará al mover el Slider, así como la distancia de desplazamiento, para lo cual nos fijamos en la figura 30. Puede suceder que, aunque pongamos correctamente el desplazamiento del Slider según la altura de la barra, se desplace demasiado, si esto es así ajustamos dicho desplazamiento hasta que recorra justamente dicha altura.

Fig. 30. Configuración del Slider.

Aunque el rango de caudal está comprendido entre 0 y 2 l/min, como hemos visto en la consigna dicho rango lo establecemos entre 0 y 32000. Esto es debido a que los Slider trabajan en este último rango, y para poder utilizarlos en el autómata está hecha una conversión para ajustar el rango del Slider al rango real.

Una vez tengamos hecha nuestra página, guardamos, compilamos y ejecutamos el programa para comprobar que todo funciona correctamente. Si no es así, arréglalo antes de continuar.

Gráficos de tendencia en tiempo de ejecución.

Existen dos posibilidades para visualizar una gráfica de un valor analógico en función del tiempo (y tanto en tiempo real como en modo histórico: el analizador de procesos (*Process Analyst*) y las gráficas de tendencia ya incorporadas en el entorno. Debido a su gran utilidad, vamos a usar el *Process Analyst* para monitorizar gráficamente la evolución de la consigna de caudal (SP), el valor actual de caudal (PV) y la acción sobre la motobomba (OUT).

Es necesario en primer lugar definir *tags de tendencia*: en el árbol de tu proyecto, clic en *Tags*, doble clic en *Tags de tendencia*, y cámbiate al *Editor de proyectos de Citect*, donde configurarás las 3 *tags de tendencia* que usaremos según las figuras 31, 32 y 33.

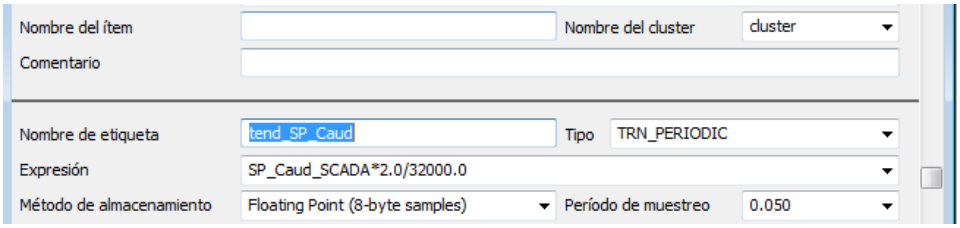


Fig. 31. Tendencia de consigna.

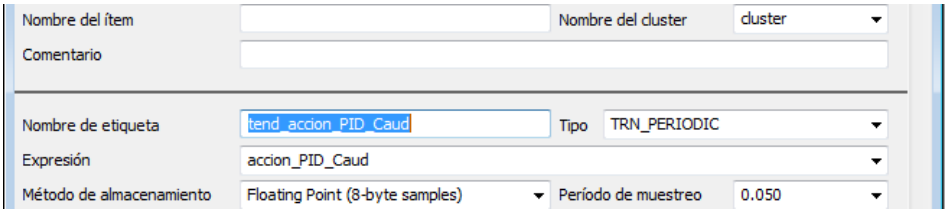


Fig. 32. Tendencia de acción.

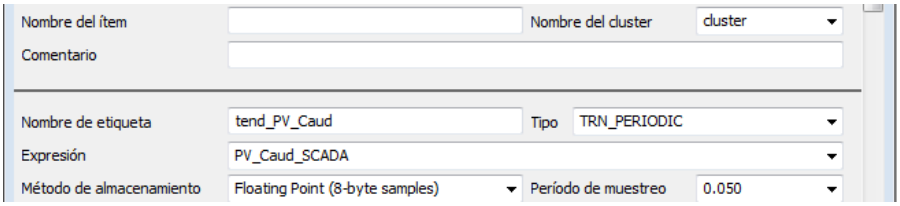


Fig. 33. Tendencia de caudal actual.

Como ves, en *Expresión* puede ponerse una expresión sencilla que involucre la *tag* que se va a historiar. Al igual que en la visualización de la consigna, hacemos la conversión para que la tendencia esté situada en el rango real 0-2 l/min.

Ahora deberemos crear un servidor de tendencias que vaya guardando los puntos para representarlos. Para ello, vamos al árbol de tu proyecto, doble clic en *Comunicaciones*, y doble clic en *Servidores de tendencia*. Eliminamos el existente si existe (*Eliminar)*, y creamos uno nuevo según la figura 34.

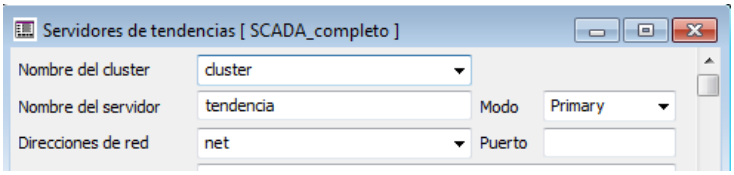


Fig. 34. Servidor de tendencia.

Alarmas.

Para el ejercicio de alarmas, se ha establecido una alarma desde el autómata a través del *tag* '*alarma'*, que está activa mientras el PID de caudal NO esté funcionando. Para crear una alarma digital, vamos al árbol de tu proyecto, doble clic en *Alarmas*, doble clic en *Alarmas digitales*, y configura una alarma según la figura 35.

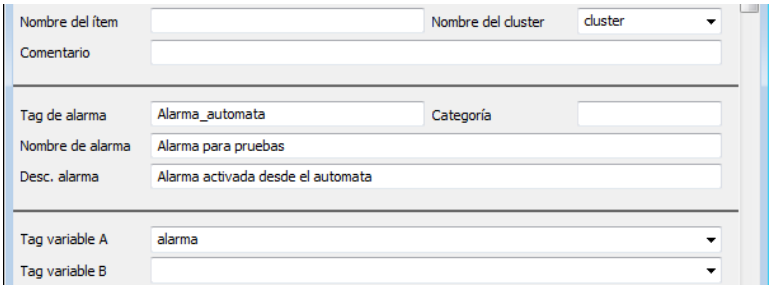


Fig. 35. Alarma digital.

Además, si la acción sobre la motobomba sobrepasa 7V queremos avisar con una alarma para evitar que la motobomba sufra en exceso, y si supera 9V, con un nivel más elevado de alarma. Para ello es preciso configurar una *Alarma analógica* según la figura 36.

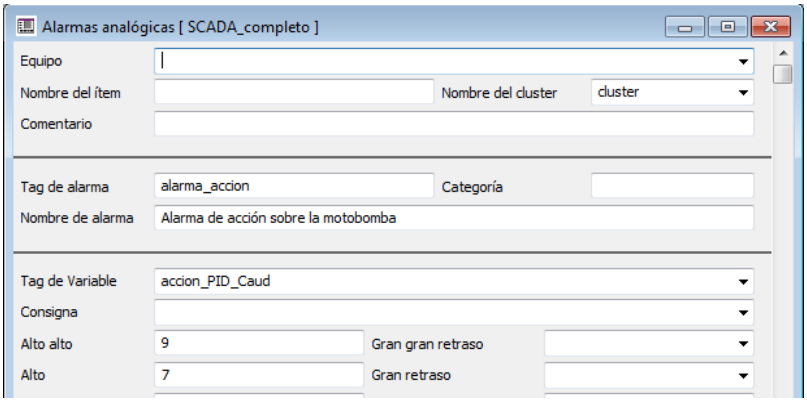


Fig. 36. Alarma analógica.

Ahora deberemos crear un servidor de alarmas que va a guardar un registro de todas las alarmas que vayan activándose, para poder mostrárselas al usuario. Vamos al árbol de tu proyecto, doble clic en *Comunicaciones*, y doble clic en *Servidores de alarma*. Eliminamos el existente si existe (*Eliminar)*, y creamos uno nuevo según la figura 37.

Fig. 37. Servidor de alarmas.

Trabajo con tendencias.

En tu aplicación, añade un *Process Analyst* desde la herramienta *Analista de procesos*  , y ponlo en tu pantalla. Puedes cambiar su tamaño pinchando y arrastrando en las esquinas. Habrá aparecido también una pantalla de configuración como la de la figura 38, y si no es así doble clic sobre el *Analista de procesos*. Pinchando en el menú desplegable de la izquierda *Vista del analizador…*, configuramos la pantalla como muestra la figura 38 (fondo blanco, velocidad de refresco 500ms, …).

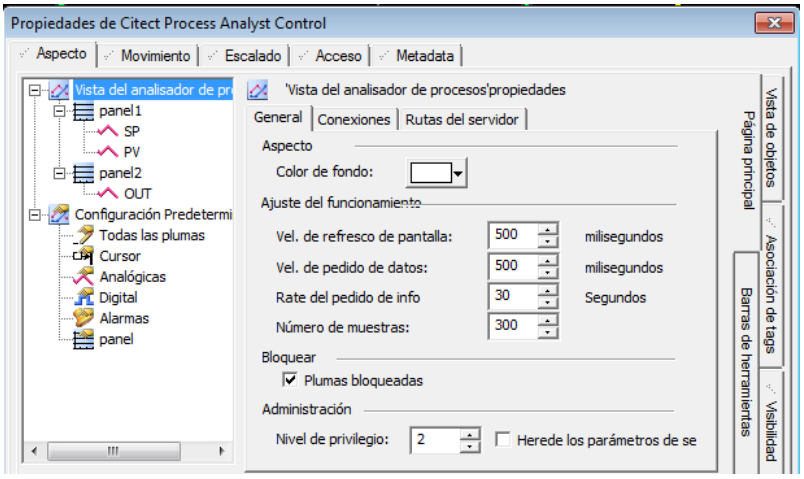


Fig. 38. Configuración del analista de procesos.

Ahora pasamos a añadir los *tags* de tendencia que hemos creado antes. Distribuiremos las tendencias en 2 paneles tal y como se ve en la figura 0: la mitad superior para la consigna de caudal (amarillo) y su medida (verde) ya que tienen el mismo rango de valores, y la mitad inferior para la acción sobre la motobomba.

Para crear el panel superior, clic derecho sobre *Vista del analizador…* y clic *Agregar Panel*. En *color de fondo* ponemos negro y *altura* seleccionamos *variable 100 peso/píxeles.* Añadimos la tendencia del SP: clic derecho en el panel que hemos creado → *Agregar pluma* → *Analógica*. Hacemos la configuración como en las figuras 39, 40, 41, 42, y 43. En la pestaña conexión seleccionaremos el *tag* de tendencia que queremos visualizar, en su promedio (figura 43).

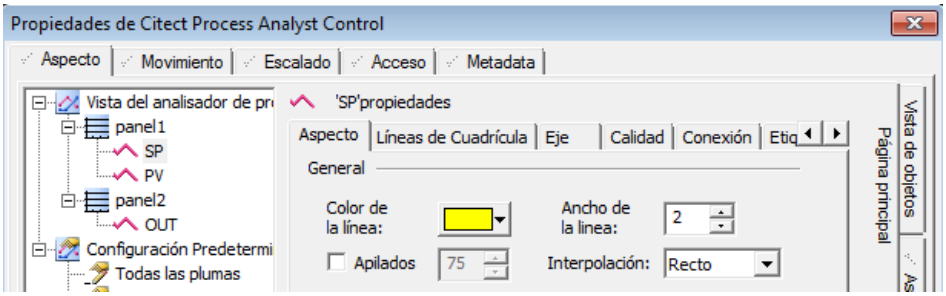


Fig. 39. Pestaña *Aspecto*.

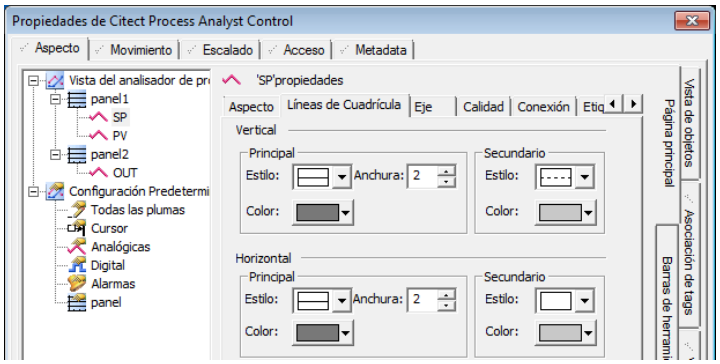


Fig. 40. Pestaña *Líneas de Cuadrícula*.

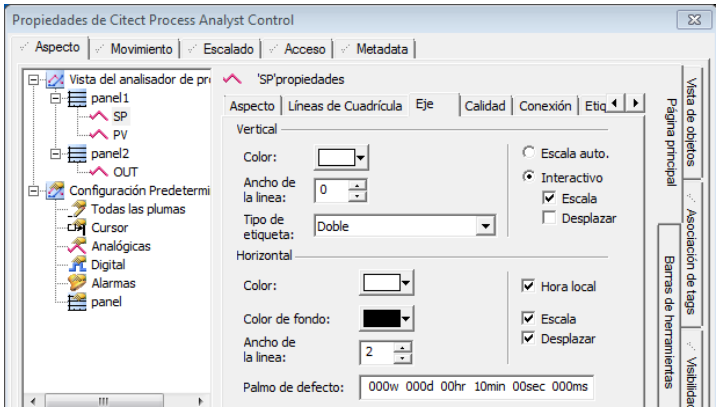


Fig. 41. Pestaña *Eje*.

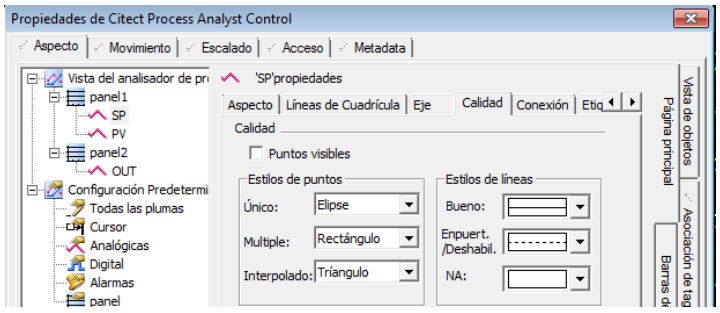


Fig. 42. Pestaña *Calidad*.

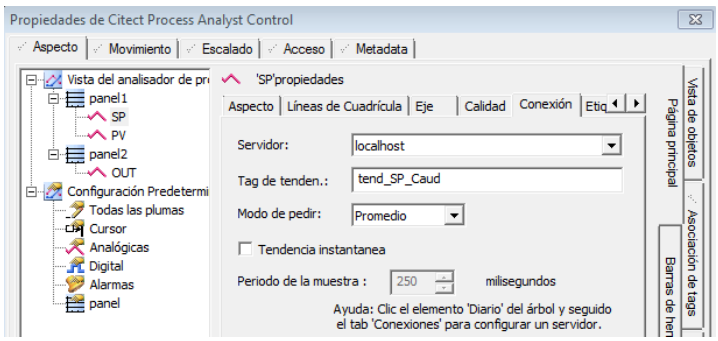


Fig. 43. Pestaña *Conexión*.

Aplicar y *Aceptar*. Veremos que se ha creado el panel ocupando todo el analista (se reajustará automáticamente al añadir el otro panel) y una línea amarilla que representa el SP. Repetimos el proceso con las otras 2 tendencias, con la precaución de poner la acción en otro panel.

Prueba de la aplicación final.

Compila la aplicación y ejecútala. Observa que puedes detener el control de caudal (lo que hará que cambie la baliza a estado intermitente), imponer una nueva consigna de caudal en l/min, y la modificación de los parámetros del PID. Prueba con varios valores para ver cómo se comporta el sistema.

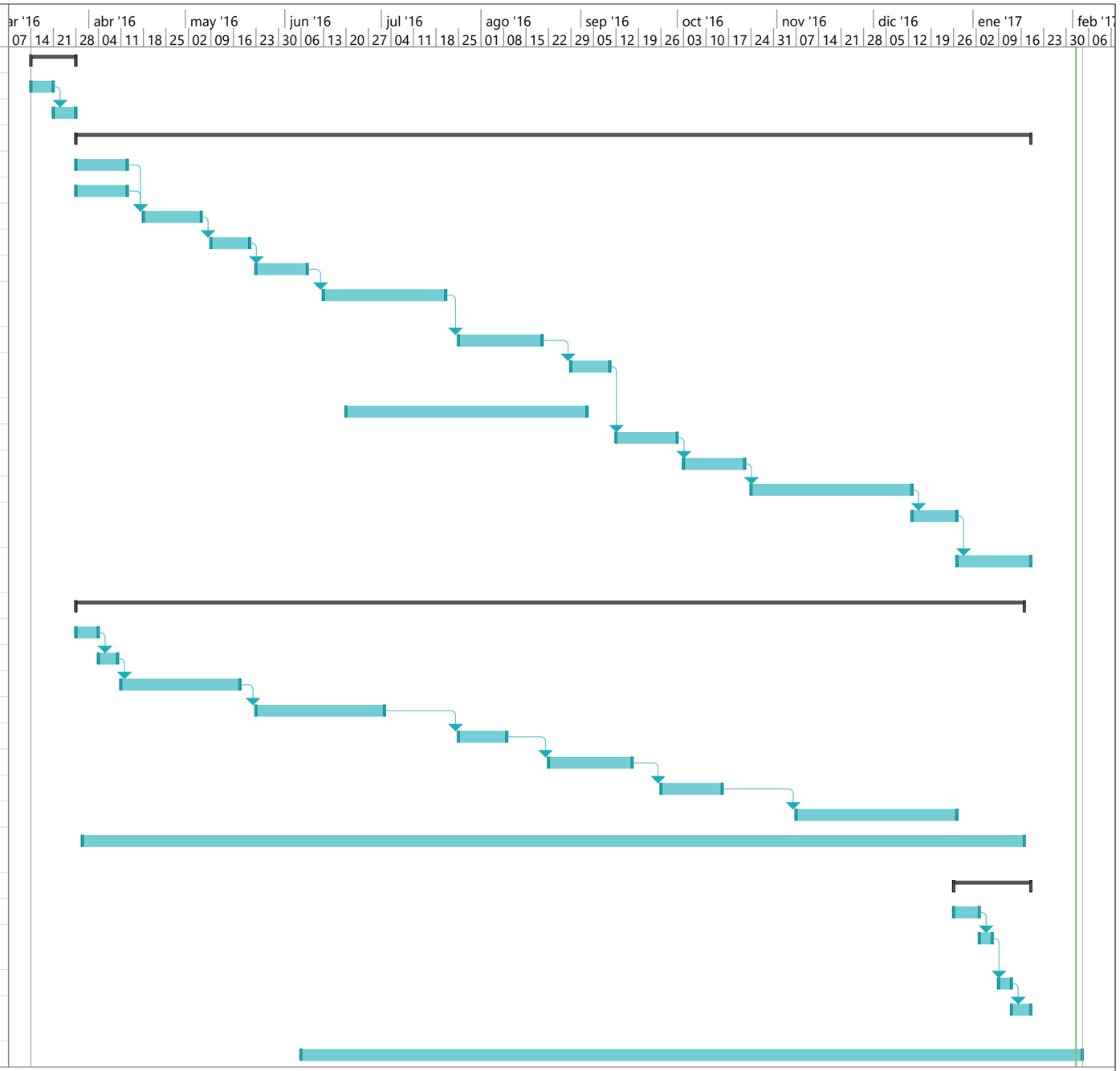
Desactiva el control PID, accede a las alarmas con el menú *Alarms*. Vuelve a activar el PID. ¿Qué sucede? Pon una consigna elevada suficiente para que la acción esté entre 7 y 9V. ¿Se disparan las alarmas analógicas?

Observa cómo se van representando los *tags* en el *Analista de Procesos*. Cambia los ajustes para obtener una visualización óptima.

Realiza las modificaciones que consideres oportunas para que la pantalla quede lo mejor posible.

Anexo G. Diagrama temporal de Gantt

Id	Nombre de tarea	Comienzo	Fin	Duración
1	**Estudio de los proyectos anteriores**	**lun 14/03/16**	**dom 27/03/16**	**10 días**
2	Estudio del programa del autómata	lun 14/03/16	dom 20/03/16	6 días
3	Estudio del programa SCADA	lun 21/03/16	dom 27/03/16	6 días
4	**Adecuación adecuación del programa antiguo del autómata**	**lun 28/03/16**	**mié 18/01/17**	**213 días**
5	Actualización a la versión nueva de Unity Pro XL	lun 28/03/16	mar 12/04/16	12 días
6	Conversión del programa del autómata de ST a LD	lun 28/03/16	mar 12/04/16	12 días
7	Separar en PID en 4 diferentes	lun 18/04/16	jue 05/05/16	14 días
8	Limpiar elementos obsoletos	lun 09/05/16	vie 20/05/16	10 días
9	Configurar PIDs de los diferentes controles	lun 23/05/16	mar 07/06/16	12 días
10	Gestionar la recepción de los parámetros de los PID desde SCADA	lun 13/06/16	mié 20/07/16	28 días
11	Crear el control para gestionar las diferentes prácticas	lun 25/07/16	vie 19/08/16	20 días
12	Gestionar las acciones de los PIDs para mandarlas a los accionadores correspondientes	lun 29/08/16	vie 09/09/16	10 días
13	Hacer pantallas de operador	lun 20/06/16	vie 02/09/16	55 días
14	Poner y configurar los IMC	lun 12/09/16	vie 30/09/16	15 días
15	Poner y configurar los Autotunning	lun 03/10/16	vie 21/10/16	15 días
16	Crear SFC de demostración automático	lun 24/10/16	lun 12/12/16	36 días
17	Gestion de una variable para mostrar en SCADA el estado de la demo	mar 13/12/16	lun 26/12/16	10 días
18	Configurar variables para mandar a MATLAB las gráficas	mar 27/12/16	mié 18/01/17	17 días
19	**Modificación del programa SCADA**	**lun 28/03/16**	**lun 16/01/17**	**211 días**
20	Modificación de la pantalla de menú	lun 28/03/16	dom 03/04/16	6 días
21	Modificación de la pantalla de control manual	lun 04/04/16	sáb 09/04/16	6 días
22	Crear una pantalla para cada modalidad de control	lun 11/04/16	mar 17/05/16	27 días
23	Hacer genio de control de los PIDs para las pantallas	lun 23/05/16	vie 01/07/16	30 días
24	Hacer supergenio para introducir parámetros	lun 25/07/16	lun 08/08/16	11 días
25	Hacer supergenio de control de prácticas	lun 22/08/16	vie 16/09/16	20 días?
26	Configurar las gráficas para cada pantalla	lun 26/09/16	vie 14/10/16	15 días
27	Hacer supergenios para visualizar la demo	lun 07/11/16	lun 26/12/16	36 días
28	Hacer correspondencia entre variables del PLC y de SCADA	mié 30/03/16	lun 16/01/17	209 días
29	**Programa MATLAB**	**lun 26/12/16**	**mié 18/01/17**	**18 días**
30	Configurar comunicación con OPC	lun 26/12/16	lun 02/01/17	6 días
31	Hacer pantalla simulink para visualizar graficamente las variables de control	mar 03/01/17	vie 06/01/17	4 días
32	Hacer patalla gráfica de control de prácticas	lun 09/01/17	jue 12/01/17	4 días
33	Hacer pantalla simulink para control de caudal con PID propio	vie 13/01/17	mié 18/01/17	4 días
34	Redacción de la memoria del proyecto	lun 06/06/16	vie 03/02/17	175 días



Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual