UNIVERSIDAD de VALLADOLID

ESCUELA de INGENIERÍAS INDUSTRIALES

Ingeniero Técnico Industrial, Especialidad Electrónica Industrial

Proyecto Fin De Carrera

Aplicación del framework UNICOS-CERN en la implantación de las capas de supervisión y control de una planta de laboratorio.

Autores:

Mínguez García, Juan Manuel

Ortega Aguado, Vicente

Tutores:

Mazaeda Echevarría, Rogelio	Ingeniería de sistemas y automática
Martí Martínez, Rubén	Ingeniería de sistemas y automática

NOVIEMBRE – 2013

ÍNDICE

Capítulo 1:

INTRODUCCIÓN

Este proyecto está basado en la elaboración de una solución, tanto para el control, como para la supervisión de una planta de presión, que será generada automáticamente a través de las herramientas proporcionadas por el *framewwork* UNICOS. Para profundizar más en este tema, se debe hacer referencia a diversos temas relacionados con la automatización de procesos y la capacidad que tiene UNICOS en este marco.

La automatización en el campo de la industria se basa en el uso de sistemas y elementos computarizados y electromecánicos con el fin de controlar una maquina determinada y/o un proceso industrial.

Los primeros automatismos industriales aparecen en la revolución industrial, en estos automatismos se utilizan elementos mecánicos y electromagnéticos, de grandes dimensiones. A través de los años los mecanismos automáticos evolucionan hasta conseguir dar solución a problemas simples a través de circuitos cableados, suponen un gran avance en este campo, pero tienen varios inconvenientes, son de gran tamaño, caros de construir y únicamente son capaces de controlar el proceso para el que están diseñados, este último problema marca la evolución de los automatismos. La aparición en los años 60 de computadoras digitales abre una nueva puerta en la evolución de los automatismos, debido a la gran flexibilidad que presentan. La evolución de estas máquinas permite crear autómatas programables o PLCs (*programmable logic controller*) que sean capaces de adaptarse a cualquier proceso de la industria además de la opción de que los procesos sean supervisados a través de estas computadoras, sistemas denominados SCADA (*Supervisory Control And Data Acquisition*). En este momento nace el sistema de control que conocemos hoy en día como puede verse en la figura 1 y que esta instaurado en toda la industria, sistemas basados en autómatas programables o PLC que son supervisados por sistemas SCADA.

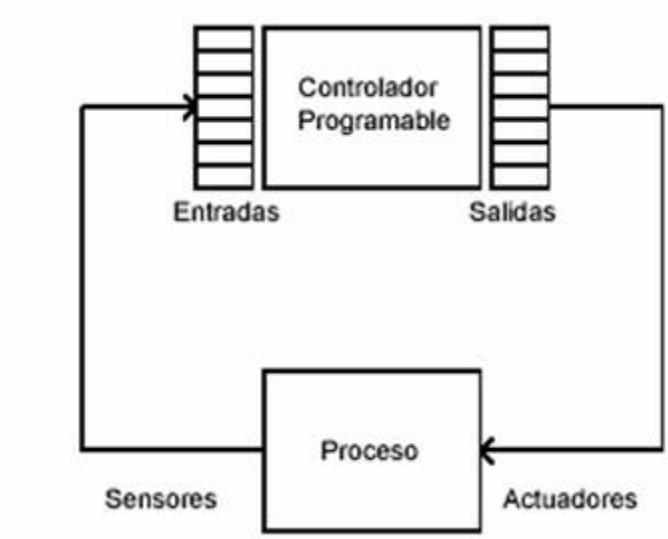


Figura 1: Proceso de control mediante PLC

La instauración de todos estos sistemas requiere de una estandarización en los procesos industriales. Esta estandarización está determinada por la norma ISA-95[1] que establece una serie de normas y pautas a seguir en todos los procesos industriales, Esta norma divide los procesos en 5 partes:

- Parte 1: consiste en terminología estándar y modelos de objetos, que se pueden utilizar para decidir qué información, deber ser intercambiada
- Parte 2: consiste en las cualidades de cada objeto, que se defina en la parte 1, los objetos y cualidades pueden utilizarse para el intercambio de información.
- Parte 3: se enfoca en las funciones y las actividades (capa de producción/planta).
- Parte 4: hace referencia a los modelos del objeto y las cualidades de la gerencia de las operaciones de fabricación.
- Parte 5: se refiere a las transacciones de la fabricación.

La aplicación de esta norma pretende estandarizar la automatización en los procesos industriales, este tipo de estandarización es complejo y ha estas alturas es más una guía para el desarrollo de buenas prácticas en este ámbito y de normas más concretas, para ello establece unos niveles jerárquicos que deben ser cumplidos por todos los procesos:

- Nivel 0: Proceso físico
- Nivel 1: Actividades que envían y controlan el proceso físico
- Nivel 2: Monitoreo y control del proceso físico
- Nivel 3: Detalles de producción
- Nivel 4: Inventario, personal, cuentas, calendario de producción

Figura 2: niveles de la pirámide de control o de la norma *ISA-95*

En este PFC se hace referencia a los primeros niveles, desde el nivel de evaluación de variables, donde los datos obtenidos pasaran a una capa de control y esta capa exigirá unos requisitos y por ultimo estos datos serán reflejados en el último nivel, el nivel de supervisión, que a través de la monitorización de los dos niveles anteriores será posible un seguimiento continuo del proceso en cuestión.

Este estándar no es suficiente para la normalización de este tipo de procesos, debido a que establece las normas a seguir en todos los procesos industriales pero no define la constitución de los PLC o sus lenguajes de programación.

Esto plantea un problema debido a que cada marca comercial construye su modelo de autómata con sus propios lenguajes de programación lo que provoca la imposibilidad de coexistencia de modelos diferentes en un mismo proceso industrial.

Para evitar este problema aparece la norma IEC 61131-3 [2] creada por la Comisión Electrotécnica Internacional que establece tanto la sintaxis como la semántica que deben cumplir los lenguajes de programación soportados por los PLC, consiguiendo así que independientemente de la marca comercial de estos los usuarios utilicen las mismas pautas para su programación.

En la norma IEC 61131-3 se definen cuatro tipos lenguajes de programación, donde se pueden distinguir dos lenguajes gráficos y dos textuales

- Versión textual:
  - Lista de instrucciones *(Instruction List o IL)*
  - Texto estructurado *(Structured Texto o ST)*

- Versiones gráficas:
  - Diagrama de Escalera *(Ladder Diagram o LD)*
  - Diagrama de bloques de funciones (*Function Block o FB*)

Estas normas establecen las características de la automatización de procesos, a continuación se hace una reseña de las capacidades de la que dispone UNICOS para adaptarse a este marco.

El proyecto UNICOS surgió en 2001 para controlar toda la criogenia del LHC, actualmente se utiliza como estándar en el CERN, debido a que facilita la producción de sistemas de control homogéneos, empleando la terminología y los modelos de la norma ISA-88, para los sistemas de control de proceso por lotes, ahorrando de esta manera mucho tiempo y trabajo.

UNICOS ha ido evolucionando hasta conseguir desarrollar tecnologías de control de una forma cada vez más simple, incorporando paradigmas de programación cada vez más homologables con los del software general.

Un ejemplo de esta evolución en busca del estándar en aplicaciones de automatización de procesos es claramente visible en la incorporación de la norma IEC

61499 [3] a UNICOS, que es soportada por SIEMENS entre otros grandes fabricantes, y que distribuye el programa de funcionamiento de un PLC en Bloques funcionales o FB. Estos bloques disponen de una memoria asignada mediante otro bloque (Bloque de datos de instancia DB) que acumula los parámetros del proceso incluso tras la ejecución completa del FB, permitiendo así la posterior reutilización de estos valores sin la necesidad de ejecutar de nuevo el bloque.

UNICOS aporta al campo de la automatización industrial una nueva forma de diseño de soluciones que cubran las capas de supervisión y control a través de la creación del *framework* UNICOS-CPC. Este *framework* basado como se ha dicho en normas aceptadas por grandes marcas comerciales establece toda una infraestructura bien definida, basada en las experiencias del grupo de automatización que trabaja en el CERN. Este grupo ha desarrollado toda una serie de manuales, que pueden ser encontrados en la página oficial del CERN [4].

UNICOS-CPC pretende estandarizar aún más el diseño de aplicaciones en el campo de la automatización industrial, lo que supone una ventaja, debido a que la aparición de errores durante el diseño se verá claramente reducida. Para llevar a cabo esta estandarización asigna una serie de objetos a cada uno de los elementos de un proceso y establece una jerarquía entre ellos, como podrá verse más adelante en esta memoria. La distribución del proceso de forma jerárquica proporciona una gran cantidad de ventajas, un ejemplo aparece cuando uno de los elementos se detiene de forma automática debido a un error, en ese momento la distribución jerárquica permite detener todos los elementos subordinados a él, este mecanismo se denomina, mecanismo de enclavamiento o *interlocks*.

Este *framework* proporciona además varios modos de operación, manual, automático y forzado, además de la posibilidad de simulación de sensores, manipulación de actuadores, diferentes tipos de accesos al sistema de control (experto, operador, monitorización). Brinda también una independencia entre las diferentes capas del sistema de control, lo que permite desarrollar las capas por separado y una metodología de trabajo con dos partes fundamentales:

- Documentación y especificaciones necesarias para realizar el análisis funcional del proceso a controlar.

- Documentación para el diseño y desarrollo del código de control

Para unificar todos estos conceptos y así estandarizar la solución tanto en la capa de control como en la de supervisión UNICOS-CPC genera automáticamente una solución estándar a partir de las especificaciones del proceso. Es evidente que esta solución en la mayoría de los casos no es suficiente, ya que es imposible prever absolutamente todas las necesidades de cada posible planta concreta. Es por eso que UNICOS es un sistema abierto que brinda la posibilidad de adaptar esta solución tanto

en el nivel del PLC como en el SCADA, incluyendo por ejemplo lógica asociada a esta solución o con la creación de un Grafcet de funcionamiento.

Este proyecto fin de carrera busca cumplir una serie de objetivos concretos mediante la utilización de esta herramienta. A continuación pueden verse enumerados:

- Diseñar e implementar una planta de laboratorio de aire comprimido

- Entender UNICOS y la solución automática que brinda aplicándola a la planta anterior.

- Adaptar la solución automática en dos sentidos

    - Creación de un programa secuencial que permita establecer diferentes regímenes de trabajo en el funcionamiento del compresor.

    - Modificación del objeto predefinido PID de UNICOS de manera que el mismo contenga código que autosintonice los parámetros del PID para un correcto funcionamiento.

- Implícitamente el cumplimiento de estos objetivos conlleva alcanzar conocimientos avanzados de nuevas herramientas de trabajo:

    - Programación y manejo avanzado de autómatas SIEMENS S7-300 y SCADA PVSS.

    - Conocimientos avanzados de regulación automática mediante bloques PID.

    - Manejo avanzado del software matemático MATLAB, para la realización de pruebas previas en la sintonización de bloques PID.

Por ultimo reseñar que este Proyecto Fin de Carrera no hubiera sido posible sin la ayuda de muchos colaboradores. Agradecer tanto a Rubén Martín como a Rogelio Mazaeda tutores de este proyecto, que hayan acompañado su desarrollo desde su nacimiento en todo momento. Agradecer también la ayuda presta al grupo de automatización del CERN que siempre estuvo en contacto y ofreció respaldo al trabajo desempeñado, en particular a Luis Gómez Palacín, que represento un contacto directo con el CERN para la solución de problemas, y sin el que el desarrollo de este proyecto habría sido impensable.

Capítulo 2:

INSTRUMENTACIÓN NECESARIA

2.1 PLC

Como ya se ha comentado con anterioridad el PLC es la parte fundamental de la capa de control de un proceso, realiza todas las operaciones lógicas necesarias para el correcto funcionamiento y toma además todos los valores suministrados por la planta. En este apartado se puede observar un desglose de los elementos que lo componen.

2.1.1 Estructura

El modelo utilizado en esta aplicación es un SIEMENS 315-2PN/DP. Este modelo consta de 6 bloques: bloque de alimentación, CPU, entradas digitales, salidas digitales, salidas analógicas y entradas analógicas, a continuación se muestra un desglose de las funciones de cada uno de los bloques que conforman el autómata.

El modelo de PLC SIEMENS 315-2PN/DP consta de seis bloques:

- Bloque 0 : Módulo de alimentación
- Bloque 1 : CPU
- Bloque 2: Entradas digitales
- Bloque 3: Salidas digitales
- Bloque 4: Salidas analógicas
- Bloque 5: Entradas analógicas

Figura 1: SIEMENS 315-2PN/DP

2.1.1.1 Bloque 0: Módulo de alimentación

El módulo de alimentación del PLC es el encargado de proporcionar corriente eléctrica al resto de módulos. Para el correcto funcionamiento de todo el PLC ha de ser configurado como puede verse en la figura 2.

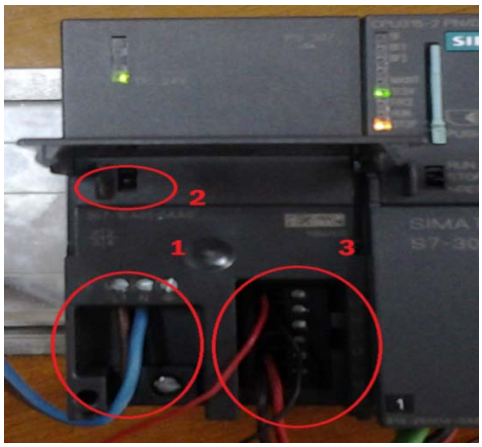


Figura 2: Modulo de Alimentación del PLC

Este módulo debe alimentarse con una señal eléctrica proveniente de la red de 220V/50Hz como se ve en la parte inferior izquierda de la imagen (1). Cuenta con un interruptor situado en la parte central (2) que enciende y apaga el PLC y de tres líneas de corriente continua de 24V (3) de las que se obtendrán la alimentación para los módulos restantes.

2.1.1.2 Bloque 1: CPU

Este bloque es el encardo del cálculo de todas las operaciones lógicas del proceso necesarias para el correcto funcionamiento del PLC. En la figura 3 se pude observar la configuración de este módulo:

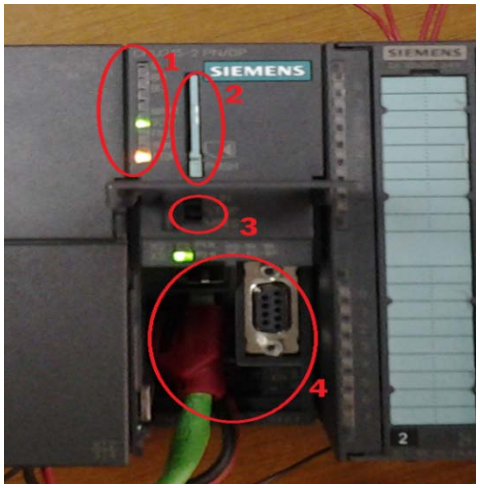


Figura 3: CPU del PLC

En este módulo pueden observarse diferentes zonas como se ve en la Figura 3:

- Los LED (1): Destinados a informar sobre el estado del PLC

- Tarjeta de memoria (2): Almacena todo el programa de funcionamiento

- Interruptor de funcionamiento (3): Selecciona el modo de funcionamiento del PLC. Consta de tres posiciones: *Run* (encendido), *Stop* (parada) y *Res* (reinicio)

- Puertos de conexión (4): Necesarios para la comunicación con el PLC, este modelo de CPU cuenta con 3 puertos.

    - Un puerto *Profibus* : destinados a la conexión de periféricos

    - Dos puertos de línea Ethernet: destinado a la comunicación con el PC o periféricos complejos.

Todos los datos técnicos y configuraciones adicionales de este módulo se pueden ver en el manual técnico: S7-300 CPU 31xC y CPU 31x: Datos Técnicos [5].

2.1.1.3 Bloque 2 y 3: entradas y salidas digitales

Los bloques 2 y 3 se corresponden con las entradas y salidas digitales. Han de ser alimentados con una señal de corriente continua de 24 V.

El bloque 2 de este PLC está destinado para las entradas digitales. Este modelo cuenta con 32 entradas distribuidas en dos líneas de 16, cada una de las cuales debe ser alimentada con una señal de 24 V, para ello se deben conectar los pines superiores de cada línea a la fuente de alimentación de 24V del módulo 0 y los inferiores a su respectiva masa. En la figura 4 se puede observar este cableado.

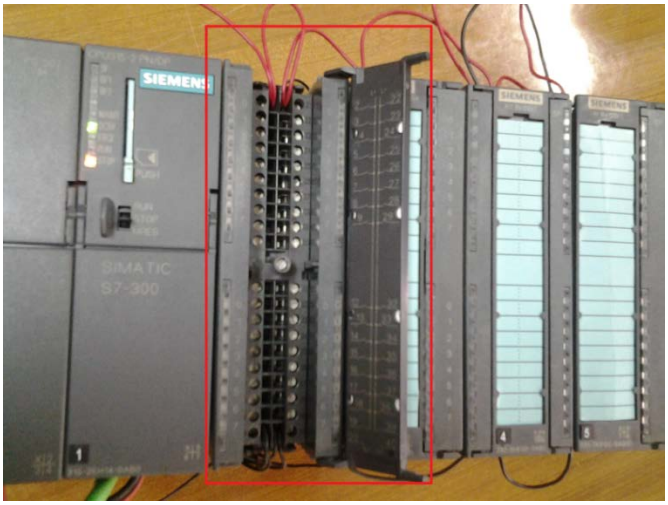


Figura 4: Módulo de salidas digitales alimentado a 24V

El bloque 3 contiene las salidas digitales del PLC. Este modelo cuenta con 16 salidas digitales, agrupadas en dos líneas de 8 y al igual que en las entradas se deben alimentar con una señal de corriente continua de 24 Voltios como se ve en la figura 5.

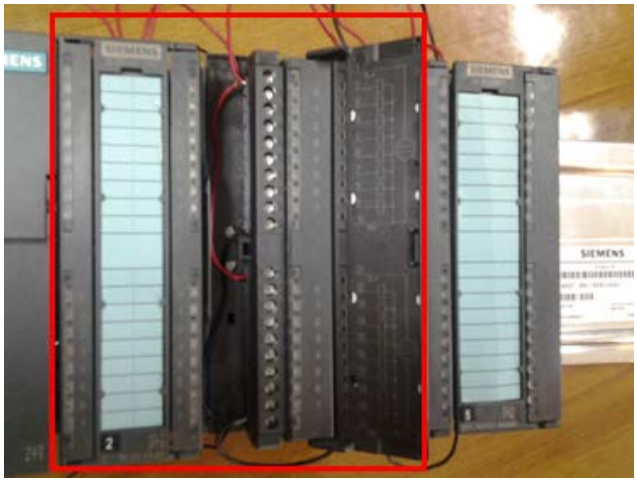


Figura 5: Módulo de salidas digitales alimentado a 24V

Estos dos módulos tienen *LED*, asociados a cada una de las entradas o salidas digitales, que se encenderán si su respectiva entrada o salida está activa, cuentan además con un *LED*, el primero de cada columna, que indicara mediante una luz roja la presencia de cualquier error tanto en su cableado como en su configuración en el software. La activación de las entradas como de las salidas se llevara a cabo mediante una señal eléctrica de 24V.

2.1.1.4 Bloque 4 y 5: Entradas y salidas analógicas

Los bloques 4 y 5 se corresponden con las entradas y salidas analógicas. Han de ser alimentados con una señal de corriente continua de 24 V.

El bloque 4 contiene las salidas analógicas, este modelo cuenta con 16 salidas repartidas en dos zonas de 8 canales. En el caso de las entradas analógicas, situadas en el bloque 5 este modelo cuenta con 8 canales.

Al igual que en los bloques de E/S digitales, estos bloques han de ser alimentados mediante una señal de 24 V. La configuración necesaria para el correcto funcionamiento se encuentra en las tapas de protección como se muestra en la figura 6.

En el caso de los bloques analógicos no basta con alimentarles, estos bloques tienen una configuración más compleja y extensa que los digitales. El PLC 315-2PN/DP permite que los bloques analógicos trabajen con intensidad a 2 hilos, 4 hilos o trabajen con voltaje. Para seleccionar estas opciones deben ser configurados según se muestra en el siguiente punto.

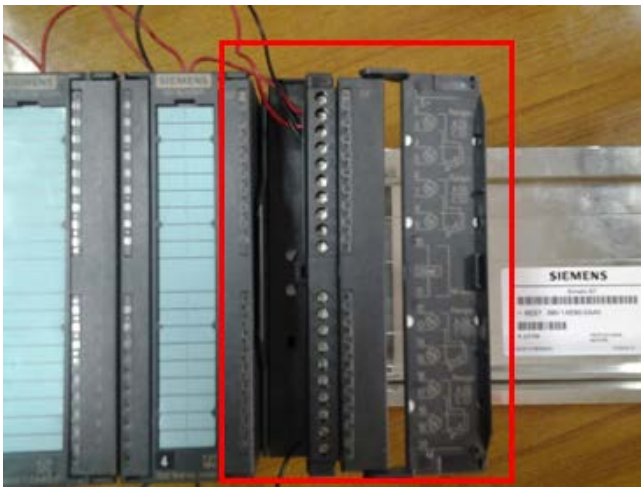


Figura 6: Modulo de entradas analógicas

Configuración de módulos analógicos

Los bloques analógicos han de ser correctamente configurados para que el funcionamiento sea óptimo. En el caso del módulo de entradas analógicas asociado a este PLC además de una configuración en el *software* se debe llevar a cabo una configuración en el *hardware*, donde se especifique el tipo de señal con la va a trabajar. Para ellos estos módulos disponen de unas pestañas colocadas en su parte inferior como puede verse en la figura 7, que permiten según la posición que adopten, trabajar con distintas magnitudes. Estas pestañas tienen cuatro posiciones que abarcan las siguientes configuraciones:

- Intensidad a 2 hilos
- Intensidad a 4 hilos
- Voltaje (mv)
- Voltaje (V)

Figura 7: Tapas de configuración bloques analógicos

Los datos técnicos de este módulo pueden verse detallados en el manual técnico: *S7 300 SM331; AI 8x12 Bit Getting Started* [6].

Para la utilización estos bloques analógicos dependiendo de la magnitud con la que se trabaje (voltaje o intensidad), la conexión de los elementos externos varía. En la figura 8 se observan los 4 puntos de conexión de un canal y como deben ser cableados para que la lectura o escritura de los valores sea correcta.

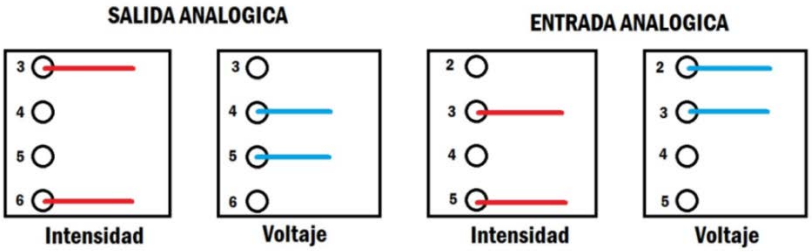


Figura 8: Tipos de cableado según las magnitudes

La configuración de los bloques tanto de entradas como salidas analógicas se debe completar en el programa *SIMATIC STEP7*, en la zona del programa dedicada al *Hardware*, se debe entrar en las tarjetas analógicas, y ajustar la magnitud de trabajo y el rango deseado para cada entrada o salida, como se muestra en la Figura 9.

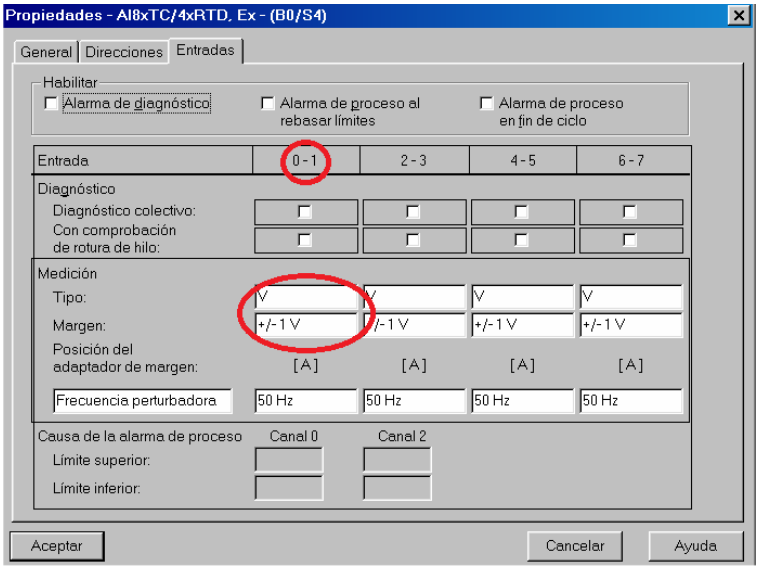


Figura 9: Configuración bloques analógicos SIMATIC S7

2.1.2 SIMATIC STEP7

El programa SIEMENS SIMATIC S7 permite trabajar con el PLC desde el ordenador de forma cómoda y eficaz, pero antes de ello será necesario llevar a cabo una buena configuración y preparación de las características del proyecto.

2.1.2.1 Ajustes previos

Antes de empezar a trabajar con SIEMENS SIMATIC STEP7 es necesario establecer la comunicación entre el PLC y el ordenador, para que sea posible la trasferencia de datos. Para ello se debe configurar correctamente la tarjeta de red instalada en el ordenador.

CONFIGURACIÓN DE TARJETA DE RED

Para la detección del PLC, la tarjeta de red del ordenador debe de ser configurada como se detalla a continuación paso a paso:

1 En primer lugar se debe acceder al menú de conexión de área local y modificar sus propiedades como se ve en la figura 10.

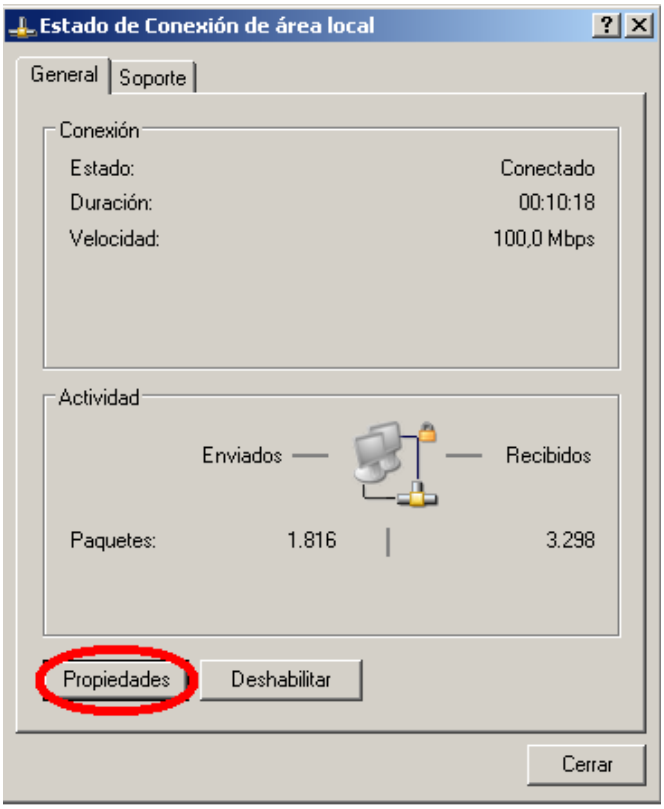


Figura 10: Conexión de área local

2 A continuación se debe pulsar sobre: Propiedades → Opciones avanzadas → Agregar. Para incluir la nueva conexión con el PLC, como puede verse en la figura 11.

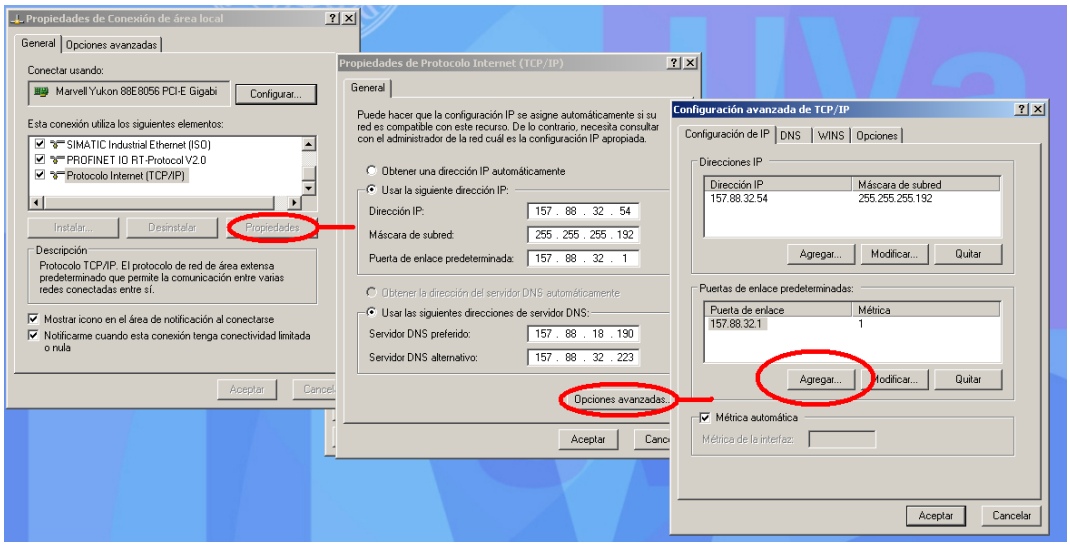
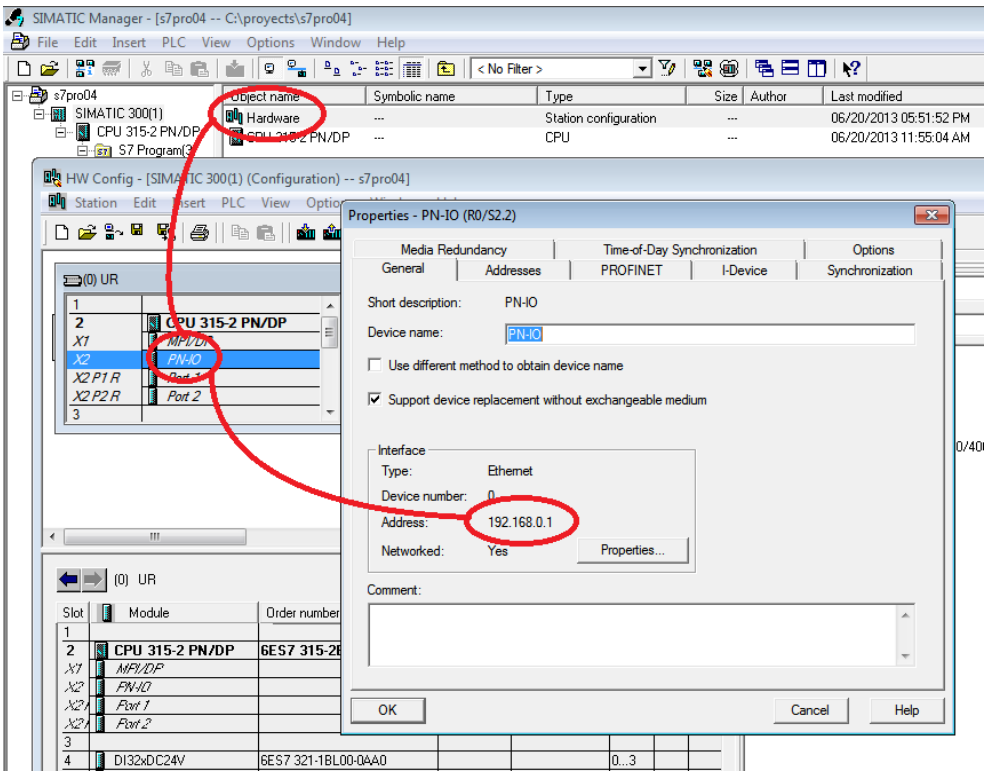


Figura 11: Opciones avanzadas de las conexiones de área local

3 En la ventana despegable que aparece, se debe introducir la dirección IP asignada al PLC, si esta se conoce, si no es así debe obtenerse desde el programa SIMATIC STEP 7 siguiendo la ruta Nuevo proyecto → Hardware → PN-IO como se muestra en la figura 12:



.Figura 12: Obtención de IP del PLC

4 Por ultimo para completar esta configuración, se debe añadir el modelo de la tarjeta de red instalada en el ordenador al programa *SIMATIC STEP7*, siguiendo la ruta Herramientas → Ajustar Interface PG/PC → Seleccionar y agregar como se ve en la figura 13:

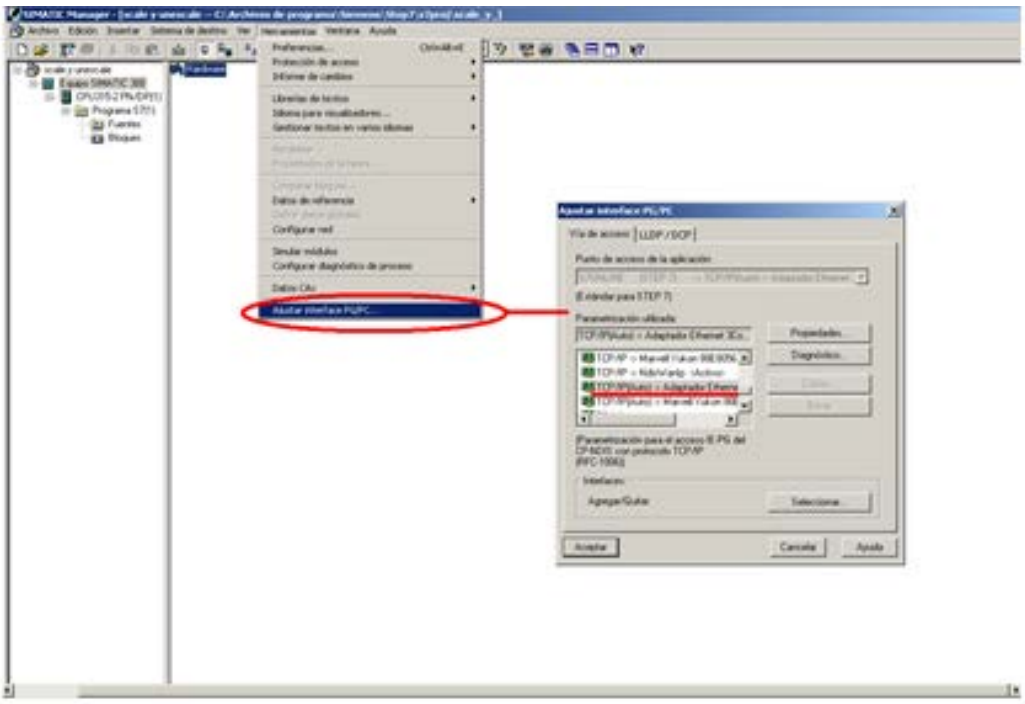


Figura 13: Configuración de tarjeta de red en *SIMATIC STEP7*

Este procedimiento deberá llevarse a cabo una sola vez, ya que es una configuración general que *SIMATIC STEP7* usara para todos los demás proyectos.

2.1.2.2 Creación de proyectos

La creación de un nuevo proyecto en *SIMATIC STEP7* es posible mediante dos formas diferentes. Una de las opciones consiste en crear un archivo vacío e ir incluyendo uno por uno todos los elementos que forman un proyecto en STEP 7. Esta opción no es recomendada en un inicio debido a que se necesita conocer perfectamente tanto la estructura de la CPU como la de los proyectos de SIMATIC.

La opción recomendada es la de utilizar un asistente incluido en el programa, denominado “Asistente de nuevo proyecto” donde únicamente se selecciona el modelo de la CPU, y el asistente creará un proyecto vacío que se debe configurar para que su funcionamiento sea correcto. Esta configuración puede verse detallada en los puntos sucesivos.

2.1.2.2.1 Configuración

Una vez creado el proyecto se deben incluir todos los módulos que contiene el PLC, además de establecer la conexión con el PLC

Para establecer la conexión entre el PLC y el programa se debe llevar a cabo mediante la opción *"NetPro"* de *SIMATIC STEP7*.

Esta opción despliega una pantalla como se ve en la figura 14 donde se debe incluir la nueva conexión. Para este modelo de PLC la conexión ha de realizarse mediante un cable Ethernet, este cable ha de incluirse mediante el menú de la derecha, en la opción subredes *Industrial Ethernet*. La conexión queda establecida uniendo esta línea con el PLC.

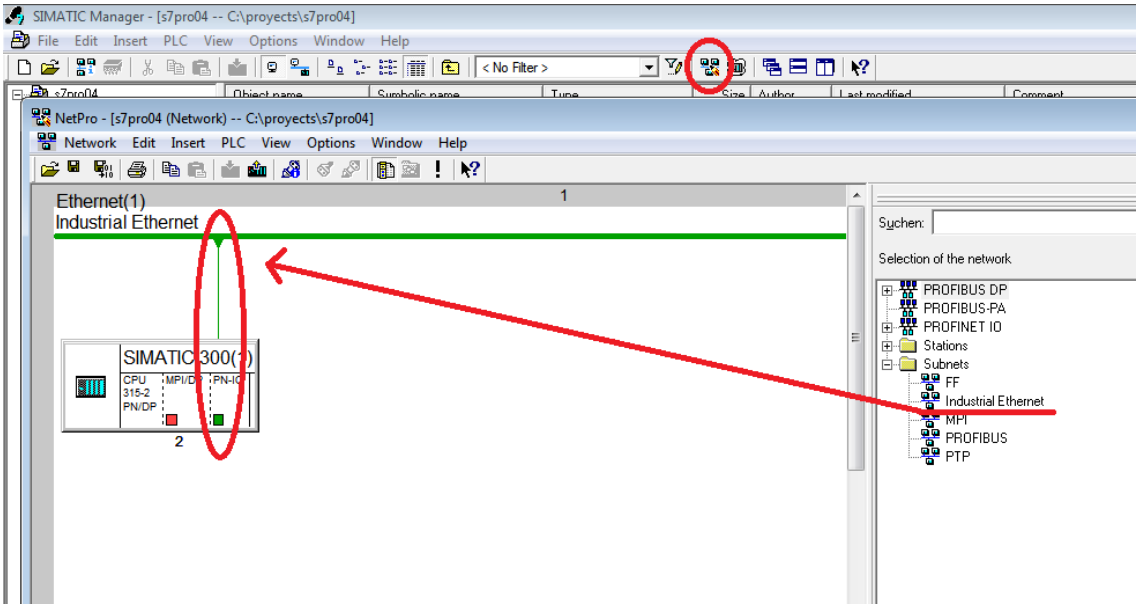


Figura 14: NetPro, línea Ethernet.

Para que el proyecto quede correctamente configurado se deben añadir los módulos de E/S tanto digitales como analógicos del PLC. Esto se lleva a cabo desde la opción de *Hardware* del programa. Esta opción requiere de los números identificativos de cada modelo de tarjeta, que aparecen en su puerta frontal. El número de cada tarjeta debe introducirse en la parte superior derecha, y el propio programa la buscará en su base datos, como puede verse en la figura 15.

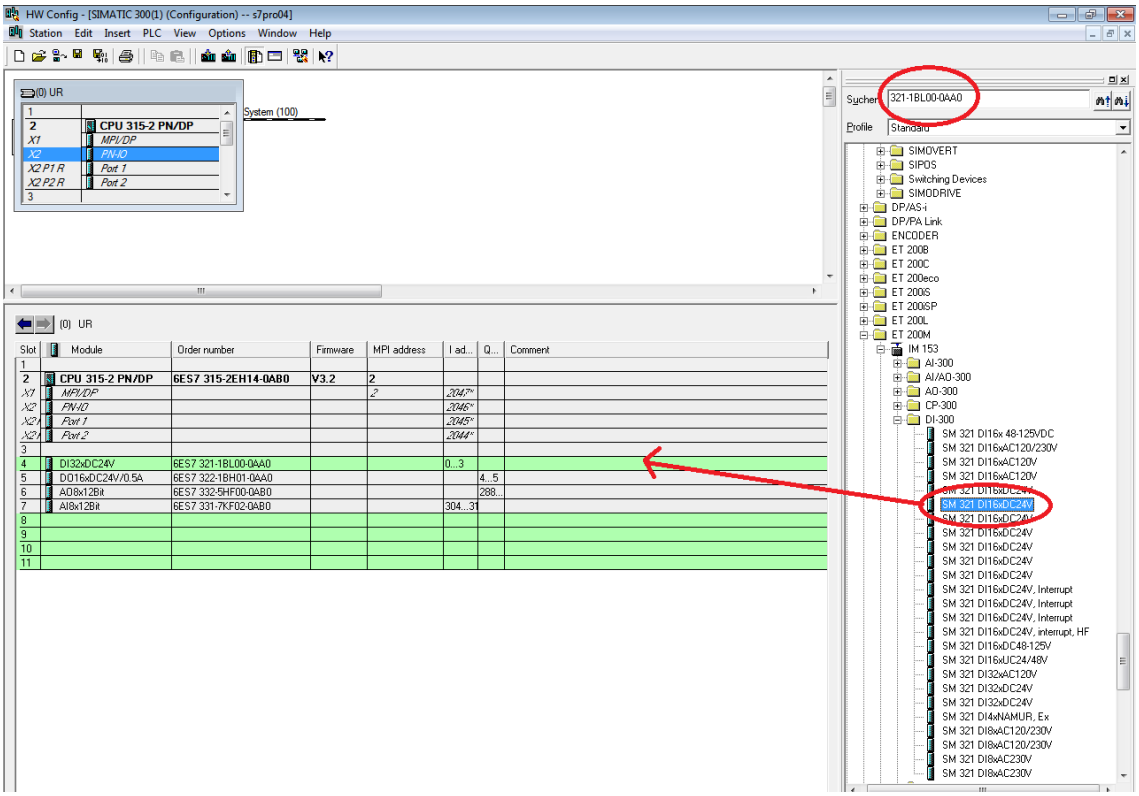


Figura 15: Identificación de las tarjetas de E/S

Por último se deben asignar a una de las ranuras arrastrando el modelo escogido a la derecha de la pantalla. En el este modelo de PLC escogido las referencias de las tarjetas de E/S, se corresponden con los siguientes códigos:

- DI 321-1BL00-0AA0
- DO 322-1BH01-0AA0
- AO 3325HF00-0AB0
- AI 331-7KF02-0AB0

Tras estas acciones el proyecto está completamente creado y ya es posible trabajar con normalidad.

2.1.2.2.2 Lenguajes de programación

Para la creación de un programa lógico que determine el funcionamiento del PLC, SIMATIC STEP7 permite trabajar en diferentes lenguajes de programación basados en el estándar IEC 61131-3:

a. STL
b. LAD
c. FBD (No utilizado en esta aplicación)

Lenguaje STL

El leguaje STL, se caracteriza por ser un lenguaje textual basado en el estándar IEC61131-3.Este lenguaje a partir de una serie de sentencias sencillas establece las ordenes lógicas que determinan el funcionamiento del programa. En el manual técnico de SIEMENS: *Statement List (STL) for S7-300 and S7-400 Programming* [7] se pueden encontrar todas las estructuras necesarias para la creación de un programa mediante este lenguaje de programación. En la figura 16, se observa una pequeña función importada de las librerías de SIMATIC STEP7 llamada *Scaling* a modo de ejemplo de este lenguaje.

```
CALL  "ReadPAValve"
 IN     :=PIW304
 HI_LIM :=2.765800e+004
 LO_LIM :=0.000000e+000
 BIPOLAR:=I0.0
 RET_VAL:=PQW304
 OUT    :=MD200
```

Figura 16: Scaling STL

Lenguaje LAD

El lenguaje LAD al igual que el lenguaje STL está basado en el estándar IEC61131-3, pero en este caso es un lenguaje gráfico, similar al leguaje de contactos. Es un lenguaje muy intuitivo, que permite trabajar de forma sencilla. En este caso aquellas funciones que se importan de *SIMATIC STEP7* son devueltas en forma de estructura, a diferencia de STL que lo devuelve en forma de líneas de código, se puede ver en la figura 17 la misma función *Scaling* pero esta vez en lenguaje LAD. En el manual técnico de SIEMENS: *Ladder Logic (LAD) for S7-300 and S7-400 Programming* [8], se pueden observar todas las estructuras lógicas capaces de ser implementadas en este lenguaje.

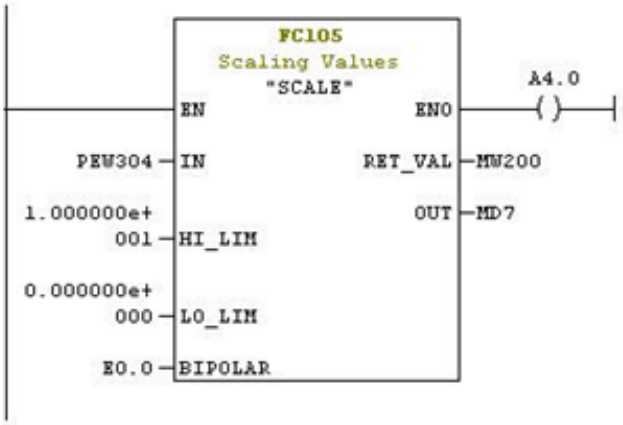


Figura 17: *Scaling AWL*

2.1.2.2.3 Librerías de operaciones lógicas STEP7

SIMATIC STEP 7 cuenta con una serie de funciones lógicas ya creadas e incluidas en una librería para que sean implementadas directamente en el programa sin necesidad de ser programadas por el usuario, estas funciones son imprescindibles para la creación de un programa que funcione de forma correcta.

Esta librería cuenta con muchas funciones, abarcando operaciones de comparación hasta bloques PID. Se pueden destacar las funciones vistas en las figuras 16 y 17, *scale* y *unscale*, son de gran importancia ya que van a permitir trabajar con las entradas y salidas analógicas escalando y desescalando los valores para su correcta comprensión

La función *SCALE* que puede verse en la figura 17, sirve para escalar un determinado valor procedente de una entrada analógica, el cual llega en forma de bits con rango desde 0 hasta 27648. Esta función dispone de varias entradas y salidas.

Como entradas a estos bloques aparecen los valores:

- La toma de datos, *IN* (entrada analógica)
- Los rangos de la escala (Max y min),
- Bipolar que nos servirá para doblar la escala en el rango negativo (Max, min,-Max);

Mientras que como salidas se pueden encontrase:

- *RET_VAL*, que es una memoria de tipo palabra que nos informa de los errores
- *OUT*, que es la salida, debe ir conectada una memoria interna que recogerá los datos ya escalados
- *ENO* al que se conecta una salida digital para verificar el correcto funcionamiento del bloque.

La función *UNSCALE* realiza la acción opuesta al *SCALE*, con esta función se consigue des-escalar unos datos en formato tipo bit, convirtiéndoles en valores entendibles por el usuario.

Esta función, que puede verse en la figura 18, cuenta en sus entradas y salidas con las mismas variables que la función *SCALE*.

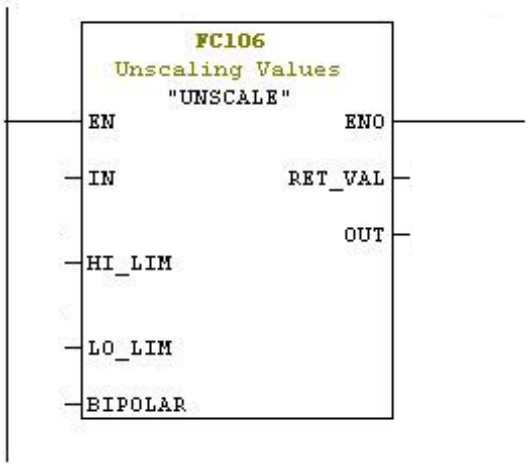


Figura 18: Función *UNSCALE*

La combinación de estas dos funciones es imprescindible para el trabajo con datos procedentes de un proceso externo. Debido a que si no son escalados correctamente no serán inteligibles para el usuario.

2.1.2.2.4 Bloques STEP7

El programa *SIMATIC STEP7* desglosa el proyecto en varias partes mediante diferentes tipos de bloques, cada uno de los cuales contendrá una parte diferente del proyecto. Esto permite optimizar el trabajo y obtener así un mayor rendimiento. Esta división nos aporta una serie de ventajas citadas a continuación:

- Los programas de gran tamaño se pueden programar de forma clara
- Se pueden estandarizar determinadas partes del programa
- Se simplifica la organización del programa
- Las modificaciones del programa pueden ejecutarse más fácilmente
- Se simplifica el test del programa, ya que puede ejecutarse por partes
- Se simplifica la puesta en servicio.

2.1.2.2.4.1 Tipos

Las divisiones en la que *SIMATIC STEP7* estructura el programa de funcionamiento se denominan bloques y se dividen en 6 tipos. Cada uno de estos bloques contiene una parte determinada de la lógica que forma el programa, por lo que se denominan también bloques lógicos. El número permitido de bloques de cada tipo y su longitud admisible dependen de la CPU.

En el gráfico que se ve en la figura 19 se observan todos los tipos de bloques con una descripción de la lógica que contienen.

Bloque	Descripción breve de la función
Bloques de organización (OB)	Los OBs definen la estructura del programa de usuario.
Bloques de función del sistema (SFBs) y funciones de sistema (SFCs)	Los SFBs y SFCs están integrados en la CPU S7, permitiéndole acceder a importantes funciones del sistema.
Bloques de función (FB)	Los FBs son bloques con "memoria" que puede programar el mismo usuario.
Funciones (FC)	Las FCs contienen rutinas de programa para funciones frecuentes.
Bloques de datos de instancia (DBs de instancia)	Al llamarse a un FB/SFB, los DBs de instancia se asocian al bloque. Los DBs de instancia se generan automáticamente al efectuarse la compilación.
Bloques de datos (DB)	Los DBs son áreas de datos para almacenar los datos de usuario. Adicionalmente a los datos asociados a un determinado bloque de función, se pueden definir también datos globales a los que pueden acceder todos los bloques.

Figura 19: Tipos de bloques presentes en *SIMATIC STEP7*

SIMATIC STEP 7 permite también crear una fuente que genere estos bloques. Esta fuente es una hoja de texto compilable donde el usuario debe especificar el tipo de bloque que desea generar y escribir la lógica asociada en lenguaje SCL, lenguaje muy similar a C que permite simplificar mucho la tarea de diseño de un programa.

2.1.2.2.4.2 GRAPH

SIMATIC STEP 7 permite también la creación de un grafcet que controle todo el funcionamiento del programa. Este archivo puede crearse mediante dos procedimientos: 1) A través de la creación de una fuente, que posteriormente será compilada o 2) A través de la creación de un bloque. En los dos casos el proyecto debe haber sido creado antes. En la figura 20 puede verse un gráfico con el procedimiento general para esta creación.

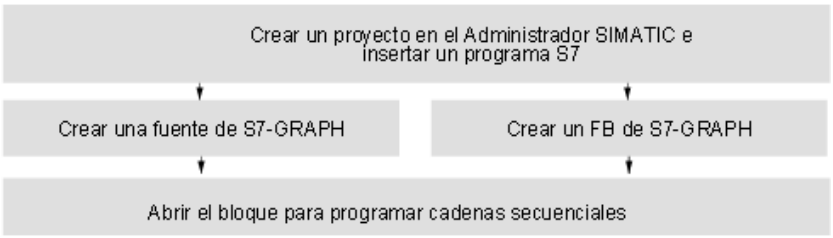
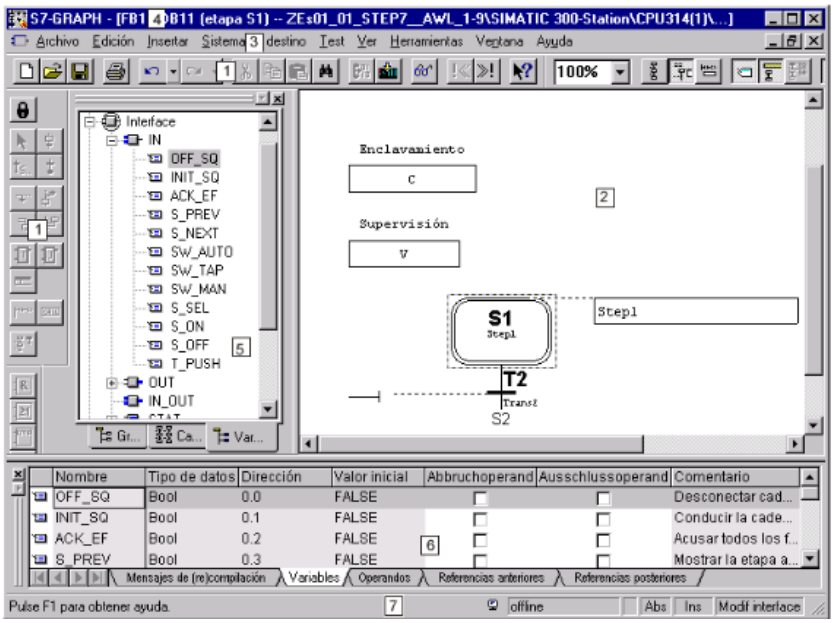


Figura 20: Creación de un Grafcet en SIMATIC STEP7

La opción escogida en el desarrollo de este proyecto ha sido la creación del Grafcet a través de un bloque. Para llevar a cabo esta elección se debe crear un bloque de función (*FB*) en el menú Insertar → Selección Bloque de función y seleccionar el lenguaje *GRAPH*. Al abrir este bloque aparece el siguiente terreno de trabajo como se puede ver en la figura 21.



(1)Barras de herramientas
(2)Área de trabajo
(3) Barra de menús
(4) Barra de título
(5) Ventana "Vista general"
(6) Ventana de detalle
(7) Barra de estado

Figura 21: Panel de trabajo del bloque Grafcet.

En este interfaz se pueden diferenciar 3 áreas de trabajo desde donde se llevará cabo gran parte de la creación:

• En el área de trabajo (2) se muestra el control secuencial para su edición. Aquí se puede definir la estructura de la cadena secuencial o se pueden programar las diversas acciones y condiciones.

• La ventana "Vista general" (5) ofrece una vista general de toda la estructura del control secuencial, los parámetros del bloque y las variables, así como del entorno del bloque de función en el programa S7.

• La ventana de detalle (6) contiene información específica necesaria para las diversas fases de programación, como p. ej., mensajes de compilación o indicaciones acerca de los puntos de aplicación de los operandos.

En el manual de SIEMENS: *S7 Graph V5.2 for S7-300/400 Programming Sequential Control Systems* [9], pueden verse todas las funciones asociadas a este de lenguaje.

2.2 Compresor

Esta planta cuenta con un compresor *JUN-AIR*, el modelo es un "*JUN-AIR Quiet Air 6-25*". Este modelo puede verse en la figura 22.

Figura 22: Compresor *JUN-AIR Quiet Air 6-25*

Se ha escogido este modelo debido a que produce un bajo nivel de ruido y una gran capacidad de compresión, alcanzando los 8 bares y produciendo únicamente 45 dB de ruido en funcionamiento normal, lo cual le convierte en la opción óptima para este tipo de trabajo en laboratorio.

Características compresor *JUN-AIR Quiet Air 6-25*

Voltaje	230 V
Frecuencia normal	50 Hz
Rango de Frecuencias admitidas	[15,85] Hz
Motor	0.36 kW
Presión Máxima	8 bar
Tamaño del tanque	25 l
Peso	29 kg
Nivel de Ruido	45 dB

Este compresor cuenta con dos partes claramente diferenciadas, el motor eléctrico y el tanque de almacenamiento del fluido

- Motor eléctrico

Este modelo de compresor cuenta con un motor eléctrico de 360W que debe de ser alimentado con señal de 230v y 50Hz para su funcionamiento normal.

El compresor no cuenta con ningún elemento para variar la velocidad de giro del motor, esta ha de ser controlada variando la frecuencia de la señal de alimentación.

- Tanque

El compresor cuenta con un tanque de almacenamiento de 25l, el motor girará comprimiendo el aire y almacenándolo hasta alcanzar una presión máxima de 8 bares, por motivos de seguridad, el compresor cuenta con un apagado de emergencia que se activará cuando se alcancen los 8bares.

Esta presión máxima solo podrá ser alcanza si la salida de aire está cerrada, si no es así el motor a máxima potencia consigue proporcionar una presión cercana a 1bar.

En los anexos se incluye la hoja de características proporcionada por la marca comercial JUN AIR.

2.3 Variador de frecuencia

El variador de frecuencia utilizado es un "*VS mini J7*" de la marca *OMROM*. Este modelo puede verse en la figura 23.

Figura 23: Variador de frecuencia *OMROM VS mini J7*

Este modelo de variador de frecuencia cuenta con una gran cantidad de modos de funcionamientos. En esta aplicación únicamente ha de ser utilizado para variar la frecuencia de una señal de salida. Esta frecuencia viene determinada por una señal de control comprendida entre 0 y 10 voltios, en corriente continua.

El variador ha de ser configurado mediante la correcta conexión de los pines que se encuentran bajo sus tapas. La configuración correcta para el control de la frecuencia mediante una señal de control puede observarse en la Figura Para que funcione a través La señal analógica de control se conecta según la figura 24:

Figura 24: Terminales de conexión del Variador de frecuencia con el PLC

Además de la correcta configuración de la señal de control, se deben conectar también la alimentación general, desde la red eléctrica, y la salida del variador. Esta conexión debe llevarse a cabo como se ve en la figura 25:

Figura 25: Conexiones Variador de frecuencia

El resto de cableados junto con las características técnicas se encuentran en el manual técnico OMROM: VS mini J7 Variador compacto de empleo general. Manual del usuario [10]

2.4 Electroválvula

La electroválvula utilizada es “M&M D263DVH”. Esta modelo puede verse en la figura 26.

Figura 26: Electroválvula *M&M D263DVH*

Es una válvula todo o nada que ha de ser alimentada con 24V para que se abra y permita el paso de aire, si no permanecerá cerrada.

Características electroválvula:

Alimentación	24V
Diámetro Nominal	3 mm
Peso	0.24Kg
Tipo de fluidos aceptados	Agua, aceite y aire
Temperatura de fluidos	[-10,130]ºC

En esta aplicación se utilizará como medida de seguridad, dependerá de una señal de control proveniente del PLC que dará orden de apertura para el vaciado completo del circuito antes de su apagado.

En los anexos se incluye la hoja de características proporcionada por la marca comercial M&M.

2.5 Transmisor de presión

El transmisor de presión es un "*Desing Instruments* TPR-14/N". Este modelo puede verse en la figura 27.

Figura 27: Transmisor de presión *Desing Instruments TPR-14/N*

Este transmisor de presión proporciona una corriente eléctrica de 4 a 20 mA para informar del valor de la presión en la red de tuberías instalado.

La intensidad eléctrica que proporciona este elemento está comprendida en un rango de 4 a 20 mA que equivale al rango de presiones de 0 a 10 bares y que ha de ser llevada al PLC donde se escalará para su correcta comprensión.

Ha de ser alimentado a una fuente de 24V como se ve en la figura 28, para su correcto funcionamiento:

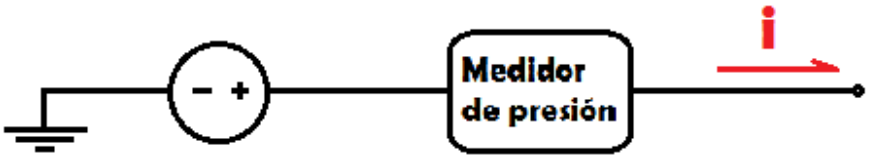


Figura 28: Alimentación del transmisor de presión

En al anexo se incluye la hoja de características proporcionada por la marca comercial Desing Instruments.

Capítulo 3:

FRAMEWORK UNICOS CPC6

3.1 Introducción

UNICOS *(UNified Industrial Control System)* es un *framework* creado por el CERN destinado al desarrollo de aplicaciones de control industrial. Este sistema abarca las dos capas superiores de un sistema de control clásico: Supervisión y Control.

El proyecto UNICOS surge en 2001 para controlar toda la criogenia del LHC, actualmente se utiliza como estándar en el CERN, debido a que facilita la producción de sistemas de control homogéneos, empleando la terminología y los modelos de la norma ISA-88, para los sistemas de control de proceso por lotes, ahorrando de esta manera mucho tiempo y trabajo.

El objetivo de UNICOS es estandarizar el desarrollo de aplicaciones de control en el *CERN* debido a que con esta metodología se consigue:

- Hacer hincapié en las buenas prácticas de las aplicaciones de control de procesos continuos
- Reducir el costo de la automatización de procesos continuos (por ejemplo, refrigeración, climatización...)
- Esfuerzos de ingeniería del ciclo de vida (por ejemplo, Optimizar el uso de herramientas de generación automática de código)

Este sistema tiene una gran cantidad de ventajas:

Para los operadores UNICOS proporciona:

- Modos de operación (manual, automático, forzado...).
- Simulación de sensores.
- Manipulación de actuadores.
- Diferentes tipos de accesos al sistema de control (experto, operador, monitorización).

Para los desarrolladores proporciona:

- Independencia entre las diferentes capas del sistema de control, lo que permite desarrollar las capas por separado.
- Una metodología de trabajo con dos partes fundamentales:
    - Documentación y especificaciones necesarias para realizar el análisis funcional del proceso a controlar.
    - Documentación para el diseño y desarrollo del código de control.

Ofrece herramientas de generación automática de código de control basado en objetos:

- Objetos PLC y SCADA.
- Configuración automática de la comunicación entre el SCADA y el PLC.
- Es una herramienta de generación muy útil para procesos repetitivos.

- El tiempo de desarrollo y de mantenimiento del sistema de control se reduce considerablemente gracias al uso de esta metodología.

En la figura 1 se puede observar un gráfico donde queda definida toda la arquitectura de UNICOS.

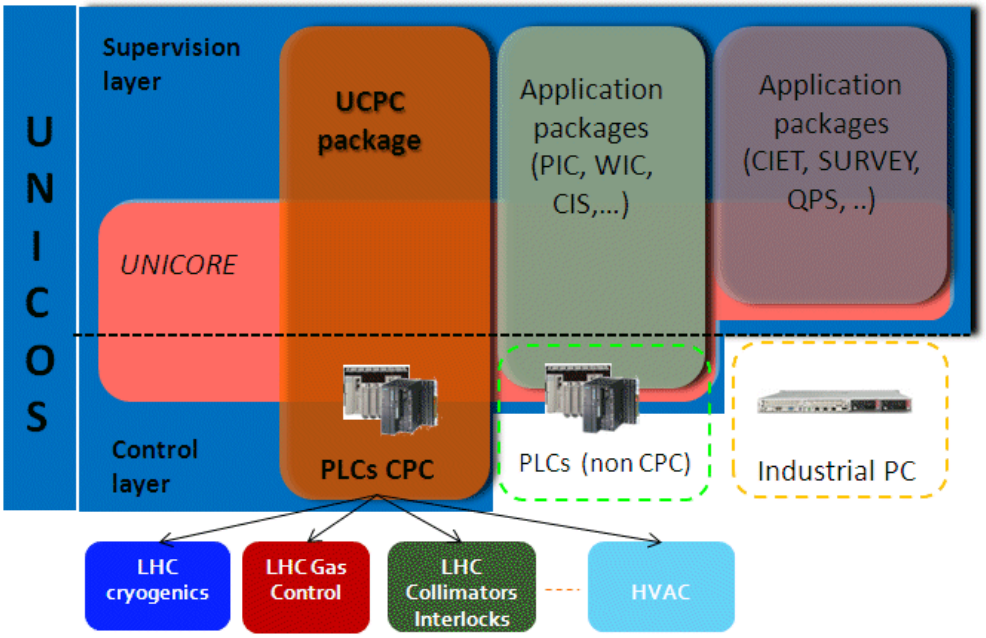


Figura 1: Arquitectura UNICOS

El presente proyecto, como ha podido verse anteriormente, se basa en la compresión del paquete UNICOS-CPC *(Continuous Process Control)* o UCPC, este paquete dispone de un manual muy extenso: UCP-MANUAL [11]**.** Este es un paquete destinado al diseño de sistemas de control de procesos continuos y está diseñado para sistemas de control con PLCs Siemens o Schneider y sistemas de supervisión PVSS.

3.2 UNICOS-CPC6 (UCPC6)

UNICOS CPC6 es uno de los componentes de UNICOS, como puede verse en la figura 1. Proporciona una librería de objetos, una metodología y una serie de herramientas para el diseño y la implementación de aplicaciones de control industrial.

Este paquete contiene una herramienta llamada *UNICOS Application Builder* o UAB destinada a la creación de aplicaciones CPC. Esta herramienta es un generador que a través de una hoja de especificaciones, introducida por el usuario, que defina el funcionamiento de un proceso, consigue generar las instancias y la lógica que lo definen.

UNICOS CPC6 pretende continuar con la filosofía de la unificación y simplificación del diseño y desarrollo de aplicaciones. Establece una jerarquía de objetos que identifican cada uno de los elementos del proceso a controlar. Estos objetos

están relacionados entre sí y además contienen todos los datos relativos al funcionamiento del proceso que controlan

La metodología de UNICOS proporciona una serie de documentos y pasos a seguir para el desarrollo de aplicaciones CPC. Mediante esta metodología es posible trasformar toda la información que se tenga sobre el proceso en una aplicación CPC que cumple la jerarquía de objetos mencionada.

Como ya se ha mencionado estos objetos deberán ser relacionados con cada uno de los elementos de nuestro proceso, se debe tener presente la jerarquía establecida por UNICOS en todo momento a la hora del diseño de una aplicación para que el funcionamiento del proceso sea correcto.

Esto se hace evidente por ejemplo a la hora de controlar elementos de una planta mediante controladores PID. Si no se tiene presente esta jerarquía, se asociaría directamente la salida del PID con el elemento a controlar, según UNICOS esto no es correcto y produciría errores. Se debe crear un elemento intermedio que asocie estas dos señales. En la figura 2 aparece un ejemplo de esta jerarquía.

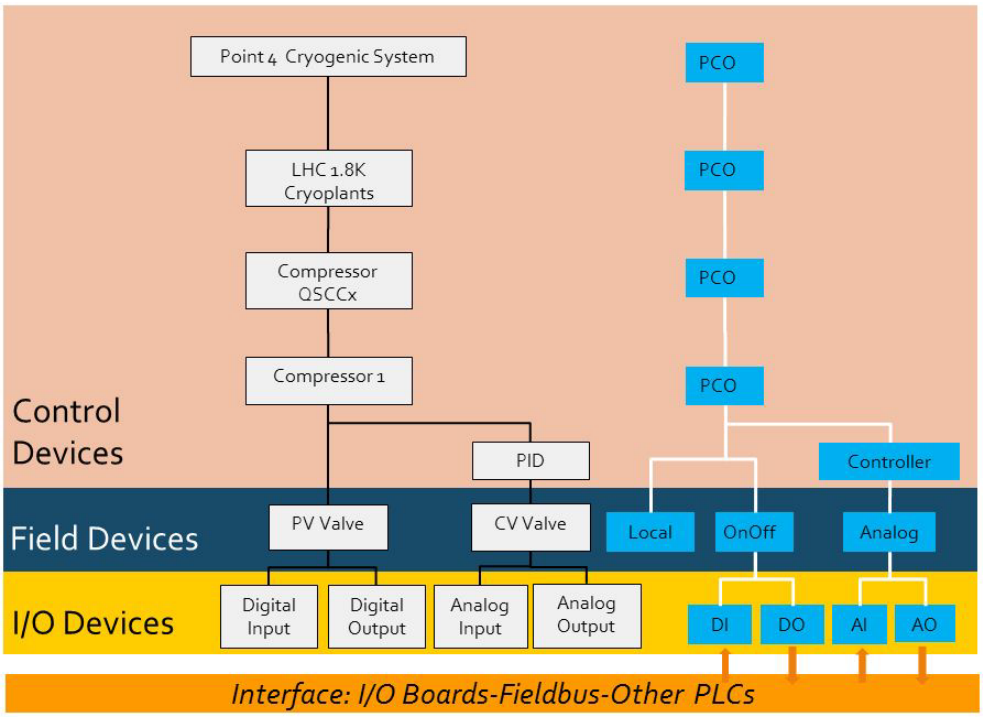


Figura 2: Ejemplo de la jerarquía de objetos CPC

La metodología UNICOS que nos permite la creación de las aplicaciones CPC contiene los siguientes pasos, que además pueden verse en la figura 3:

- Análisis funcional: el ingeniero de proceso transfiere su conocimiento sobre el proceso en el documento llamado UNICOS *Functional Analysis.*
- Diseño de lógica UNICOS: el ingeniero de proceso y el ingeniero de control trabajan juntos para definir el comportamiento del proceso en lenguaje UNICOS.
- Configuración de las especificaciones: todos los objetos UNICOS son definidos y parametrizados utilizando un archivo Excel/XML.
- Generación automática: el código de instancias y de lógica para el PLC y el archivo de configuración para PVSS se generan automáticamente utilizando las herramientas de generación UAB.
- Configuración de la lógica: completar la generación automática de lógica si fuese necesario, siguiendo la lógica de diseño UNICOS.
- Compilación del código de control.
- Test de Comprobación
- Puesta en marcha
- Operación

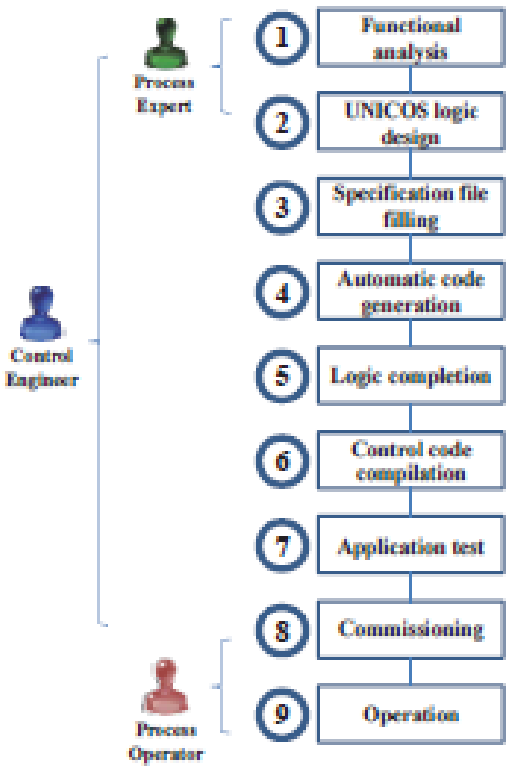


Figura 3: Metodología de UNICOS-CPC

3.3 Arquitectura de los sistemas de control

El diseño *hardware* se basa en un modelo de tres capas utilizando E/S distribuidas, conectadas a la capa de control de proceso y la de supervisión, como puede verse en la figura 4. El diseño software construye un modelo del proceso completo con la jerarquía de componentes o dispositivos (canales de E/S, actuadores, conjunto de sensores y actuadores, constituyendo una entidad del proceso).

Los componentes del proceso se implementan en dos partes, una en la capa de control del proceso, y una en la capa de supervisión, las cuales están comunicadas. La capa del PLC se encarga de todas las instalaciones para que se maneje el dispositivo bien por el operador (modo manual) o bien por la lógica de alto nivel (modo automático), además de su respuesta a entradas externas tales como *interlocks*. La parte correspondiente en la capa de SCADA es proporcionada por el operador para supervisar y manipular los dispositivos.

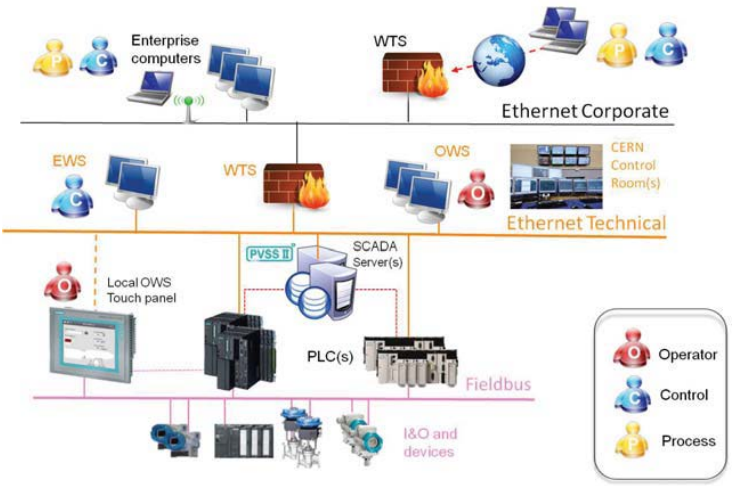


Figura 4: Arquitectura UNICOS

3.4 Estructura de los objetos UNICOS

UNICOS CPC está basado en una jerarquía de objetos elementales que se identifican a continuación:

- Objetos de Entrada/Salida: información de E/S.
- Objetos de interfaz: memoria de interfaz y/o información de periferia. Normalmente más ligeros que los objetos de E/S.
- Objetos de campo: equipo real (ej.: válvulas, motores…) y actúan en los objeto E/S.
- Objetos de control: llevan a cabo las acciones de la lógica de control (incluyendo *interlocks*).Actúan sobre los objetos de campo.

Toda la estructura que identifica los objetos UNICOS puede verse en la figura 5:

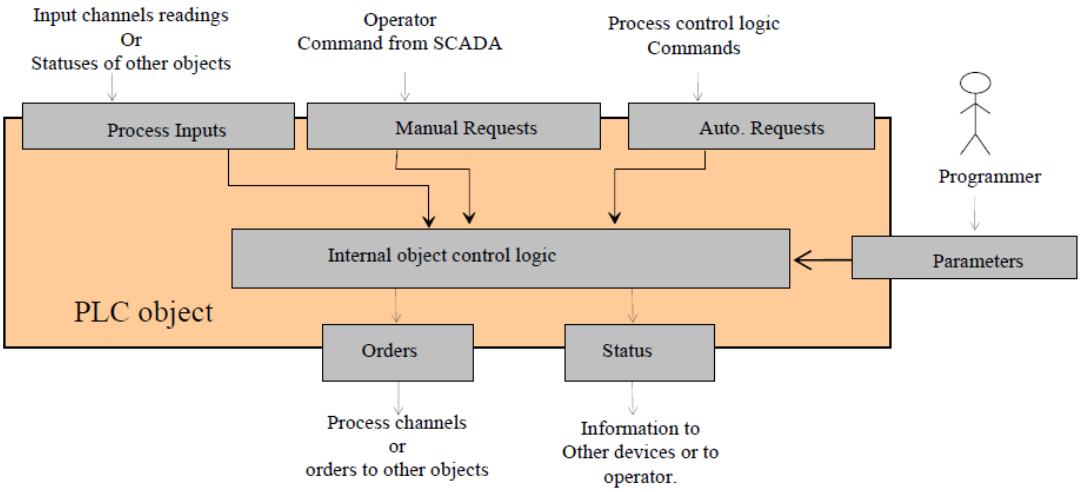


Figura 5: Estructura general de los objetos UNICOS

Todos los objetos que han sido desarrollados para PLC Siemens están representados por bloques funcionales (FB) genéricos que serán inicializados en un bloque de datos (DB). Los objetos se dividen en tres grupos:

- Dispositivos de entrada/salida
  - Entradas y salidas digitales (DI, DO)
  - Entradas y salidas analógicas (AI,AO)

- Objetos de campo (Ver figura 6)
  - Objetos locales
  - Objetos ON/OFF
  - Objetos analógicos
  - Objetos analógicos-digitales

- Objetos de control.
  - Objetos controlador
    - El más utilizado es el PID, en la figura 7 se puede ver, su implementación eléctrica.
  - Objetos alarmas
  - Objetos de control de proceso (PCO)

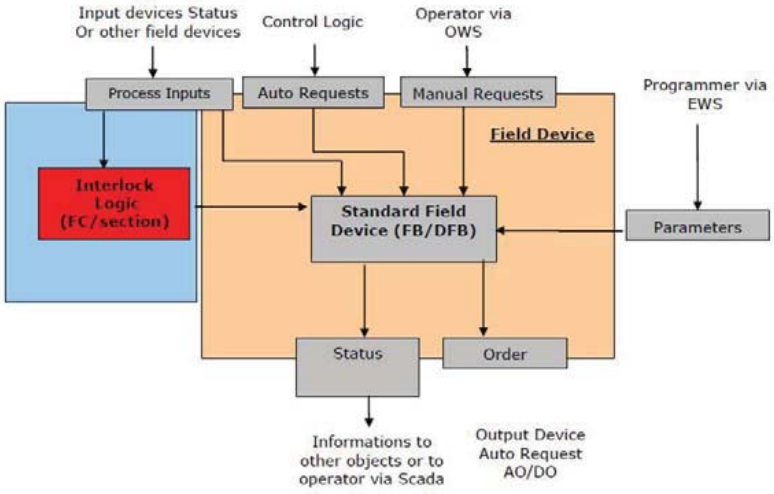


Figura 6: Estructura de los objetos de campo

En esta lista se pueden identificar todos los objetos que forman UNICOS-CPC6. Para diseñar una aplicación basada en un proceso determinado se deben identificar cada uno de los elementos y decidir que bloque lo identifica, no solo los elementos físicos, sino también las partes del proceso que no están asociadas a ningún elemento físico, como podría ser una alarma.

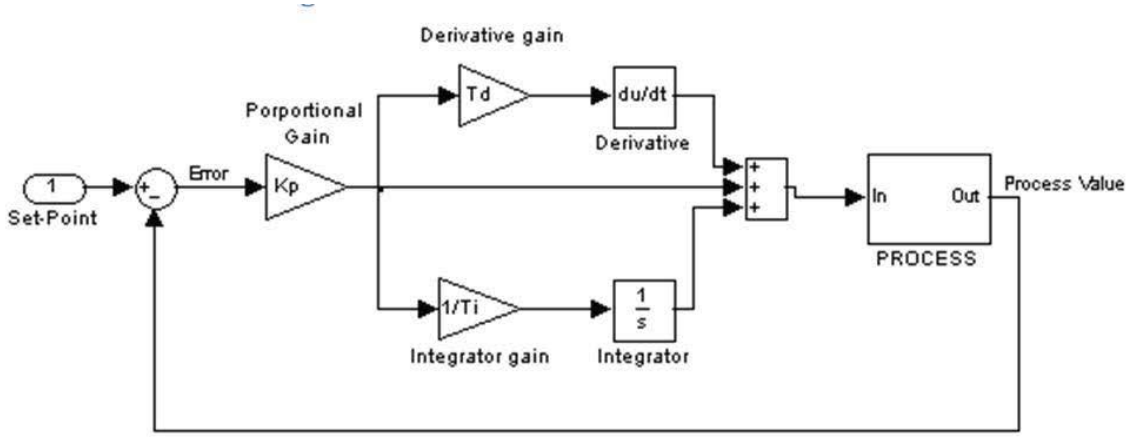


Figura 7: Estructura mixta de los controladores PID

Los objetos PCOs son los más relevantes en el proceso, debido a que según la jerarquía UNICOS son los que encabezarán y administrarán todo el proceso. La estructura general de estos bloques puede verse en la figura 8.

La lógica de los PCOs se divide en varias sub-funciones: lógica de *interlocks*, lógica de configuración, lógica estándar del objeto y lógica específica del proceso. A continuación se detallan sus características:

- Lógica de *interlocks*, tres *interlocks* son posibles: *interlock* de arranque (evita el arranque del objeto); *interlock* de parada total (para el objeto), el re-arranque requiere la eliminación del *interlock* y su reconocimiento por parte del operador; *interlock* de parada temporal (para el objeto), el re-arranque no requiere la intervención del operador, a no ser que se encuentre en modo manual o forzado.

- Lógica de configuración: esta lógica determina si el objeto estará activado o desactivado dependiendo de los estados de los objetos dependientes.

- Lógica estándar del objeto: existen ocho modos posibles de funcionamiento que pueden ser activados por la lógica o por el operador. Si se para el proceso, se puede ir de un modo a otro sin restricciones. Por el contrario, si el proceso está activo, el paso de un modo a otro debe ser autorizado.

- Lógica específica del objeto: el punto de funcionamiento del proceso depende de la lógica de *interlocks*, la de configuración y la estándar. Un comando es enviado a la lógica específica del proceso.

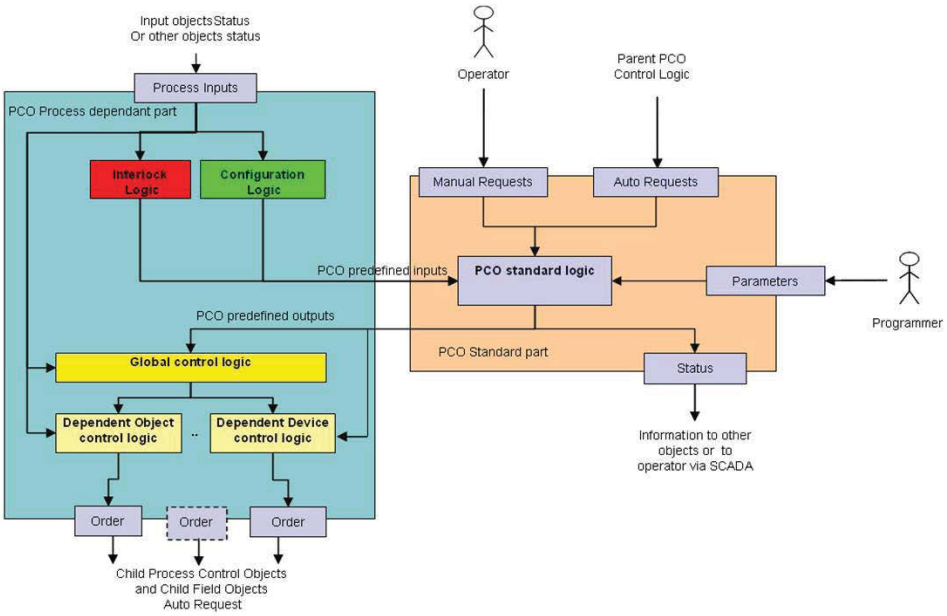


Figura 8: Estructura general de los objetos PCO.

3.5 Generador UAB: UNICOS Application Builder

UAB es una herramienta de generación, que engloba todos los generadores UNICOS en uno, que tenga una parte común que gestione y permita compartir una gran cantidad de recursos. Los requisitos para esta herramienta de generación son:

- Flexibilidad y escalabilidad (mejora de los *templates*, creación de nuevos objetos, adición de nuevas plataformas).
- Mejorar el tiempo de generación.
- Un interfaz intuitivo y fácil de usar.
- Gestión de versiones.

3.5.1 Arquitectura UAB

Para cumplir con estos requisitos, UAB cuenta con una arquitectura dividida en dos partes, el núcleo (*Core*) y los *plugins*. En la figura 9 se puede un gráfico con esta arquitectura desglosada.

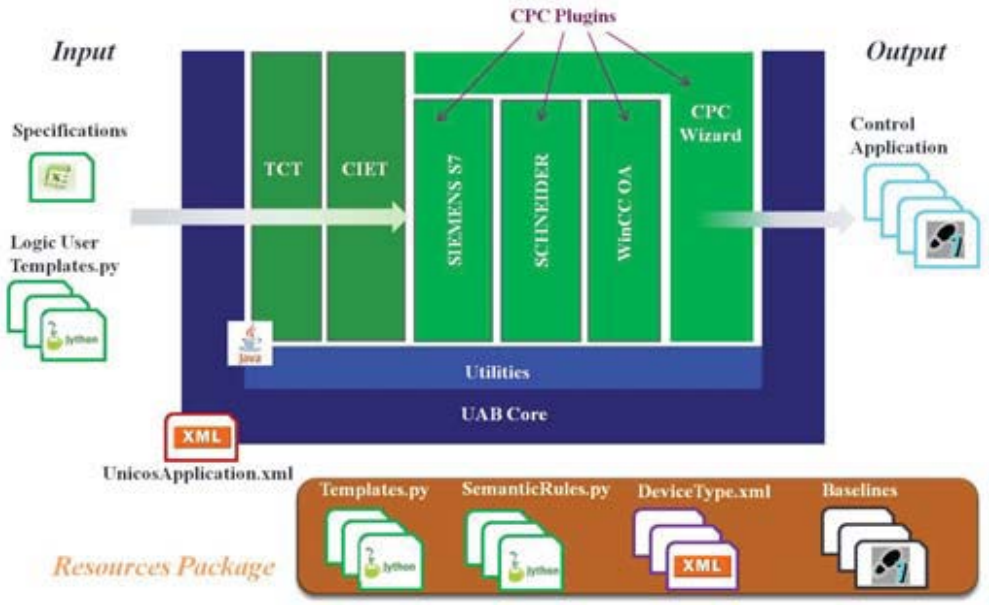


Figura 9: UNICOS Application Builder

Núcleo (*CORE*):

Es la parte principal del generador, gestiona toda la aplicación. Todos los datos de entrada pasan por este núcleo, los obtiene, los procesa y proporciona una serie de funciones para que los diferentes *plugins* puedan acceder a esos datos de entrada. Está escrito en lenguaje JAVA

PLUGINS

El *plugin* tiene como función crear una estructura para la generación de código de una plataforma concreta. Al igual que el núcleo está escrito en JAVA. Actualmente en el UAB están desarrollando varios *plugins* para UNICOS-CPC, a continuación se pueden ver los necesarios para la creación de la aplicación de este proyecto:

- Generador de instancias S7 (*S7 Instance Code Generator Plugin)*: encargado de generar código de las instancias para el PLC Siemens.

- Generador de lógica S7 (*S7 Logic Generator Plugin)*: encargado de generar el código de lógica para el PLC Siemens, es decir, todas las interconexiones entre los PCOs y los objetos de campo del proyecto a controlar

- Generador de instancias PVSS *(PVSS Instances Generator)*: encargado de crear el fichero de importación para el SCADA.

- Asistente UAB (UAB *wizard*): diseñado para facilitar el uso del *UAB* a los desarrolladores de sistemas de control, proporcionándoles una interfaz gráfica sencilla donde solamente se muestran parámetros no expertos para configurar.

Estos *plugins* son necesarios para la creación del proyecto, son analizados en profundidad en el punto “Creación de proyectos > UAB” del presente proyecto.

El generador UAB emplea estos *plugins* para la creación de todos los archivos necesarios para el correcto funcionamiento tanto de la capa de supervisión como de la capa de control. Únicamente necesita que se le aporte un archivo XML-EXCEL que deberá estar correctamente cumplimentado por el Ingeniero, definiendo todos los elementos a controlar en el proceso.

3.5.2 Especificaciones del proyecto

El archivo XML-Excel de especificaciones, contiene todos los datos que identifican cada uno de los bloques que componen el proceso. Si por ejemplo el proceso contiene una salida analógica, en este archivo de especificaciones se debe incluir la dirección del PLC donde debe estar situada, el rango de valores con los que trabajara, si tiene alguna lógica asociada etc.

Para optimizar el trabajo antes de comenzar a rellenar la hoja de especificaciones se deben tener presente todos los elementos del proceso y distribuirles de tal forma que se cumpla con la jerarquía UNICOS, en la figura 10 se puede ver una distribución simplificada del proceso descrito en este proyecto, para la correcta comprensión de como rellenar este archivo.

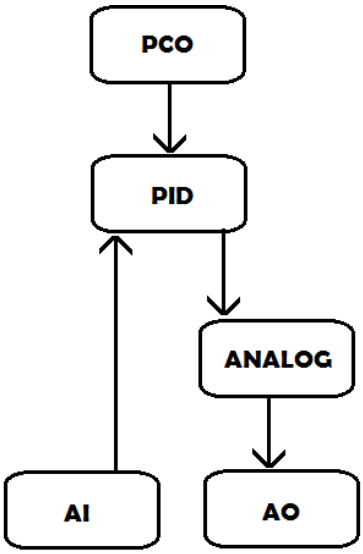


Figura 10: Grafico Planta según Jerarquía UNICOS

Este gráfico describe un proceso en el que mediante un PID se controla una variable de salida dependiendo del valor que tenga la entrada. Como se puede ver en la figura 10 se necesita describir cinco bloques en la hoja de especificaciones para que después el generador UAB cree los archivos correspondientes correctamente. Esta hoja de especificaciones es un archivo XML-EXCEL que cuenta con una estructura unificada. Este libro Excel está conformado por varias hojas:

- Hoja de documentación (*ProjectDocumentation*):

Esta hoja debe de estar rellena con los nombres de los ingenieros y la fecha de modificación, como se ve en la figura 11, si no es así el generador producirá errores.

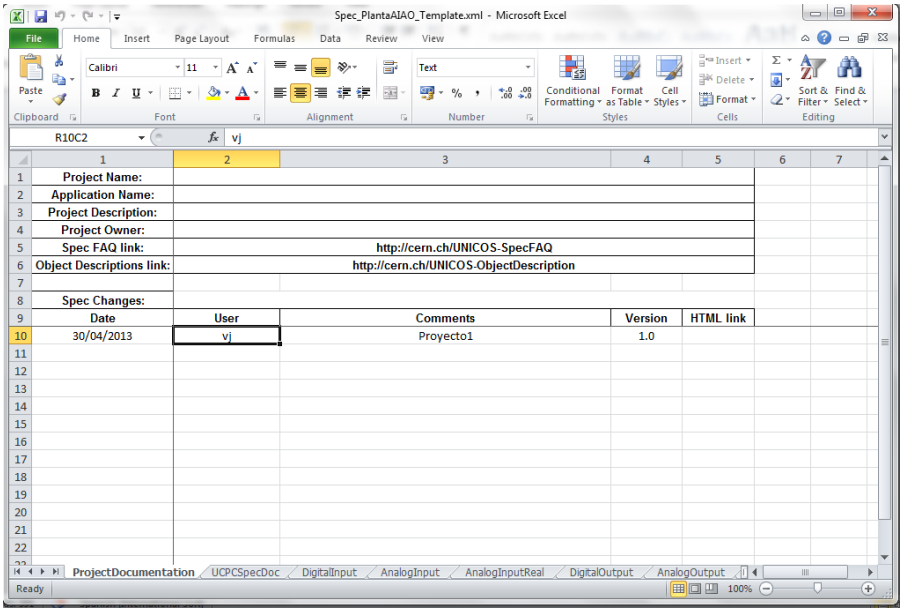


Figura 11: Hoja de Documentación del archivo EXCEL

- Hoja de ayuda (*UCPSpecDoc*)

Esta hoja contiene todas las ayudas para la correcta cumplimentación de los campos de la hoja. Como se puede ver en la figura 12 cada campo cuenta con una breve descripción del valor que debe contener.

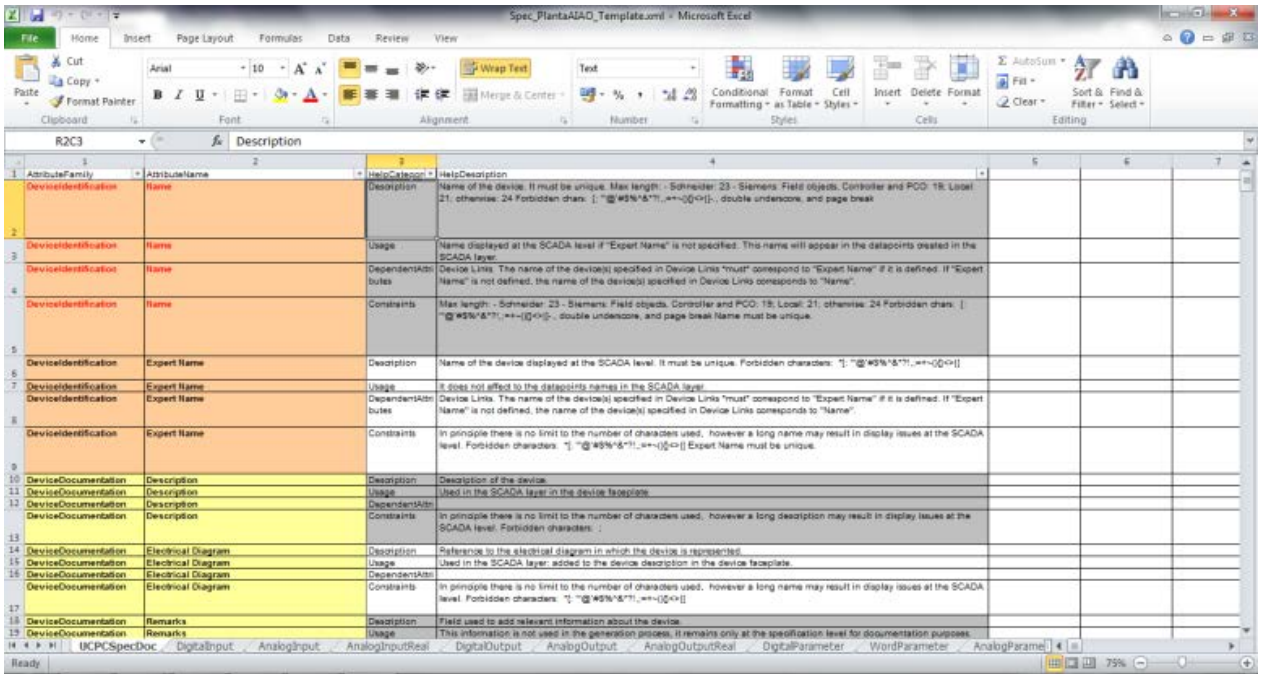


Figura 12: anexo de ayuda

- Hojas de especificaciones de cada bloque

El archivo genérico Excel cuenta con una hoja para cada tipo de objeto de UNICOS-CPC, se deben rellenar UNICAMENTE las hojas que coincidan con los bloques de nuestro proceso.

Se pueden identificar dos tipos de campos en estas hojas, los campos obligatorios, que aparecen en color rojo, y los campos complementarios, en color negro. Para conseguir una aplicación que funcione basta con rellenar los campos obligatorios, pero es aconsejable añadir los campos complementarios de los que se conozcan los valores.

Cada objeto tiene asignada una hoja dentro del archivo de especificaciones donde se pueden diferenciar cuatro partes. Una para incluir la descripción general del objeto, otra dos con la información relativa al PLC y al SCADA, y por ultimo una más para incluir archivos de lógica asociados al elemento.

Esta lógica debe estar escrita en un archivo fuente del lenguaje Phyton que se incluirá en la carpeta del proyecto de UAB en *"Resouces/S7LogicGenerator/Rules/UserSpecific"*, y que será llamando desde el Excel. Esta lógica puede añadirse posteriormente con el proyecto creado en *SIMATIC* como se verá en el presente proyecto.

Si aparece cualquier tipo de duda para rellenar algún campo del archivo se deben pinchar sobre el botón *"help"*, que aparece sobre todos los campos, que despliega la hoja de ayuda donde se indica que valores se deben introducir. En las figuras 13, 14 y 15 se ve una hoja Excel correspondiente a una entrada analógica correctamente cumplimentada.

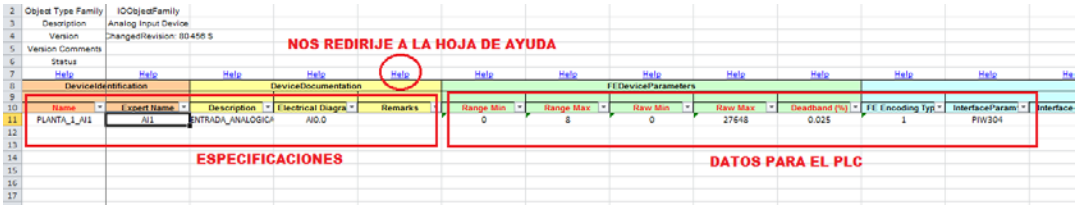


Figura 13: Hoja de especificaciones Excel 1

Figura 14: Hoja de especificaciones Excel 2

Figura 15: Hoja de especificaciones Excel 2

3.5.3 Creación de proyectos paso a paso

A continuación se describe como crear un proyecto paso a paso tanto en el generador UAB como en SIMATIC STEP7 y PVSS. Estos manuales deben ser seguidos paso a paso para la correcta creación del proyecto, la omisión de alguno de los pasos o la incorrecta ejecución provocará errores posteriores muy difíciles de resolver.

Para la creación de proyectos se ha creado además con un video-tutorial con el nombre Creación de aplicaciones UNICOS CPC6. Disponible en la plataforma youtube.com

3.5.3.1 UAB

A continuación se puede ver un manual paso a paso para el correcto uso del generador UAB. Además se dispone de un manual creado por el CERN [12] que amplia esta información.

1. Abrir el asistente UAB y en UAB *Bootstrap* y presionar *cpc-wizard* como se ve figura 16

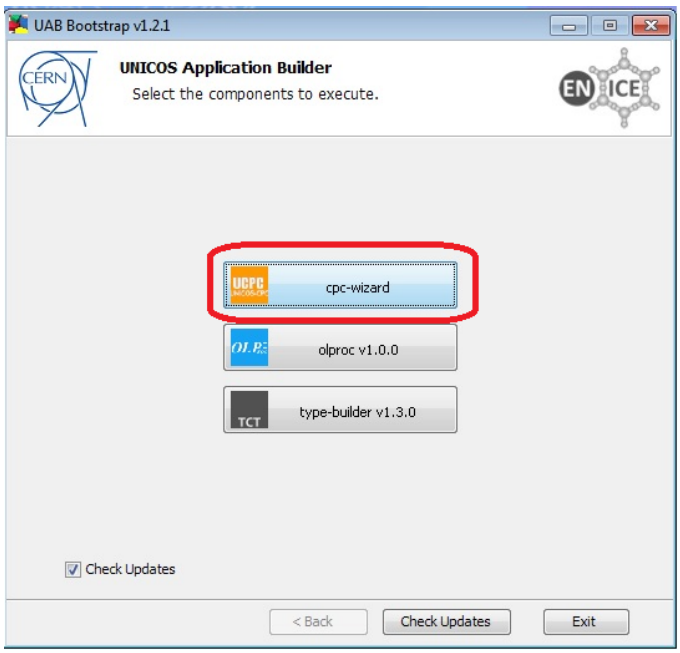


Figura 16: UAB Bootstrap

2. En la siguiente ventana se debe seleccionar la pestaña para la creación de un nuevo proyecto, y seleccionar la marca del PLC. Por último se debe escoger la carpeta contenedora del proyecto. Para evitar errores futuros es aconsejable ubicar el proyecto directamente el directorio raíz.

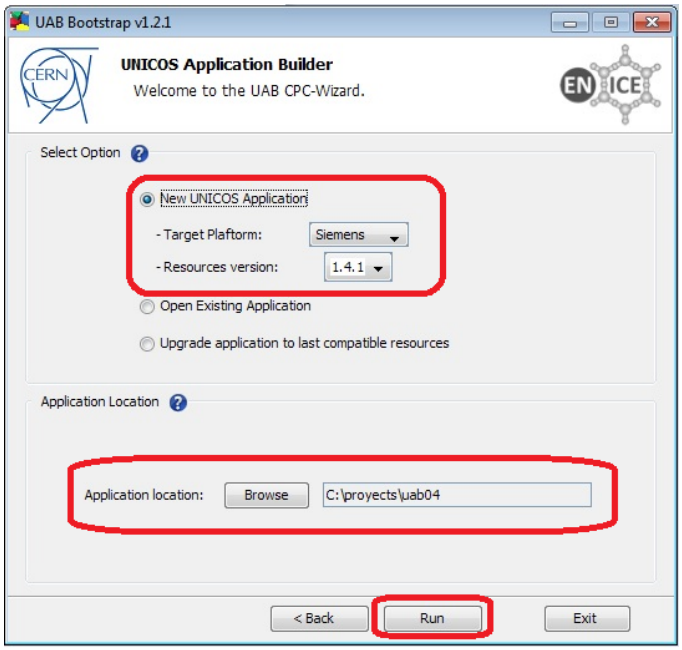


Figura 17: Nueva aplicación UNICOS para SIEMENS

3. En la siguiente ventana se deben escribir las especificaciones del proyecto como el nombre de la aplicación y la versión. Además se debe incluir el archivo de especificaciones previamente cumplimentado como se ve en la figura 18.

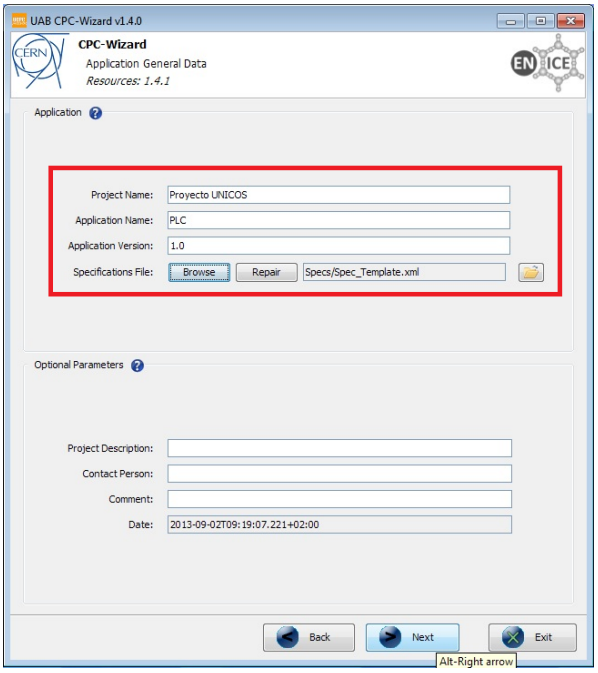


Figura 18: Datos de la aplicación para UAB

4. Se debe cumplimentar a continuación toda la información relativa a la configuración y la conexión entre el PLC y el SCADA. Los valores de estas casillas deben cumplimentarse como se ven en la figura 19. La dirección del PLC y el modelo debe ser acorde a la aplicación desarrollada.

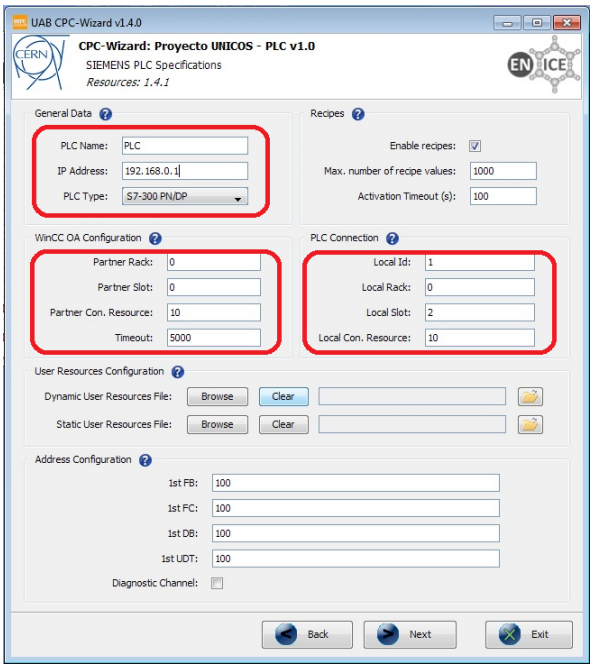


Figura 19: Configuración de PVSS y PLC en UAB

5. A continuación se deben ejecutar los tres *plugins* que se ven en la figura 20 para generar toda la lógica y las instancias que definen el proyecto.

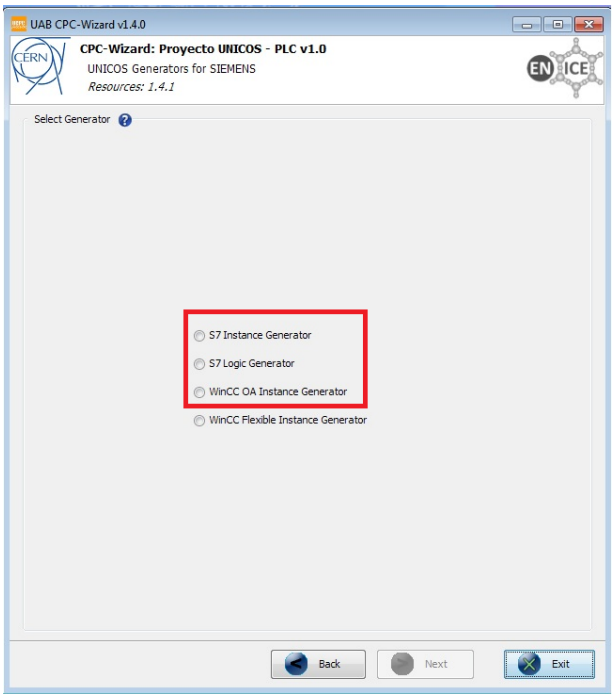


Figura 20: Plugins de generación

6. Generador de Instancias *(S7 Instance Generator)*. Para la correcta generación de las instancias se deben seleccionar todos los objetos y a continuación generar las instancias como se ve en la figura 21. Este proceso puede tardar varios minutos.

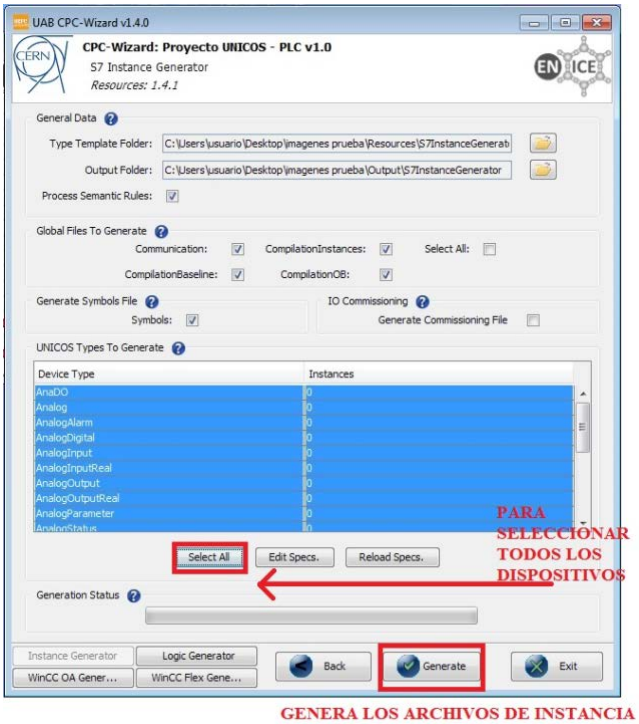


Figura 21: Generador de instancias

7. Generador de Lógica (*S7 logic Generator*). La operación para la generación de la lógica es la misma que para la de las instancias, se deben seleccionar todos los elementos y pulsa sobre generar, como se ve en la figura 22.

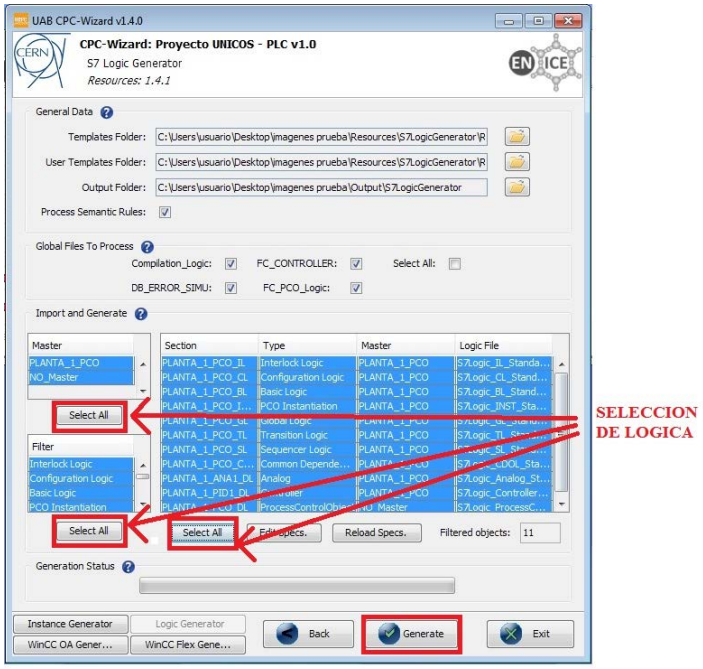


Figura 22: Generador de lógica

8. Generador PVSS (*PVSS Instance Generator*) La generación del archivo para la configuración del SCADA se lleva a cabo como como en el resto de *plugins*. Se seleccionan todos los objetos y se genera como se ve en la figura 23.

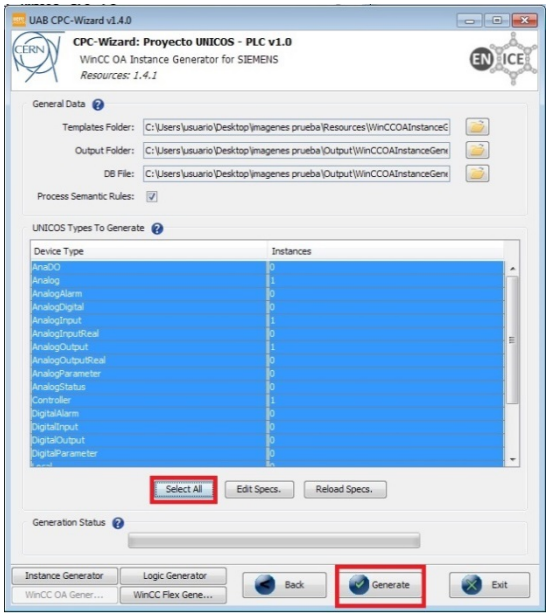


Figura 23: Generador ficheros PVSS

9. Toda esta generación de archivos se analiza desde la consola del generador como puede verse en la figura 24, que informa sobre los errores que se produzcan. Si una generación de archivos ha sido exitosa aparecerán los siguientes mensajes:

    a. **"The exit status of the S7CodeGenerator plug-in is SUCCED"**
    b. **"The exit status of the S7LogicGenerator plug-in is SUCCED"**
    c. **"The exit status of the PVSSInstanceCodeGenerator plug-in is SUCCED"**

    Esta consola ofrece varias herramientas como el copiado de sus mensajes o el borrado del área de trabajo, como puede verse en la figura 2

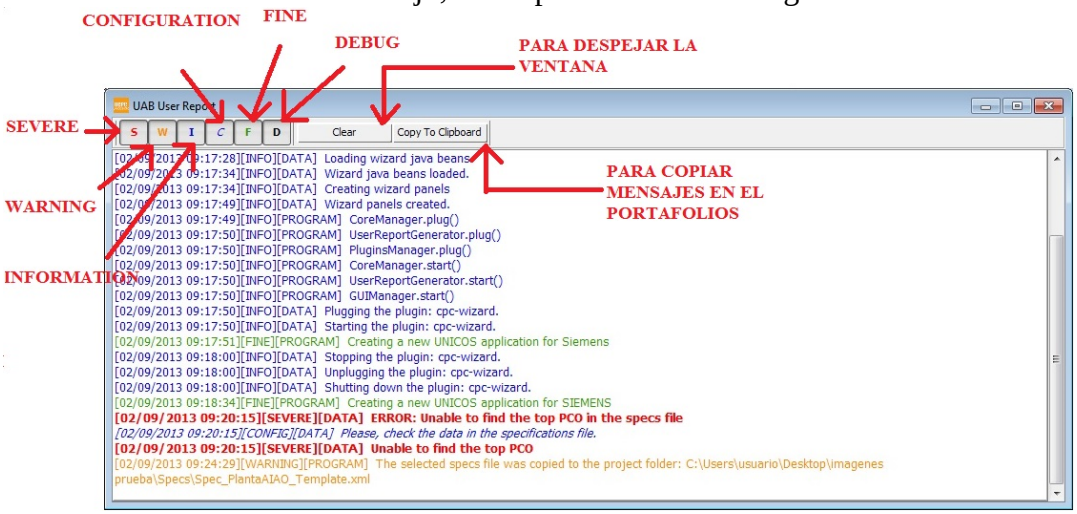


Figura 24: Consola PVSS

Si la generación no consigue completarse con éxito aparecerá un error en color rojo, como se ve en la figura 24 de la consola, que informa de los posibles errores. Los errores más comunes son provocados por la incorrecta cumplimentación de la hoja de características:

a. Olvidar rellenar un campo obligatorio de la hoja Excel.
b. El tamaño, formato o caracteres a la hora de nombrar los campos de la hoja Excel.
c. Escribir nombres de objetos en las especificaciones.

3.5.3.2 SIEMENS SIMATIC STEP 7

Para continuar con la creación de una aplicación en el marco de UNICOS CPC6 se deben importar los archivos generados por UAB creando un proyecto que cumpla con la filosofía UNICOS y que además contenga la lógica generada. Los archivos que se obtienen tras la generación son archivos de tipo fuentes S7 y bloques S7 que definen la estructura y la lógica del programa.

A continuación se detalla cómo se debe llevar a cabo la importación de esta lógica mediante un manual paso a paso. También se dispone de un manual creado por el CERN [13] donde también se detalla este proceso.

Antes de proceder con la importación de la lógica generada por UAB el proyecto de SIMATIC debe estar correctamente creado, como se detalla en los puntos anteriores y a continuación realizar tres ajustes para evitar errores futuros.

- Ajuste de la fecha y hora del PLC
- Configuración de la conexión entre el PLC y el SCADA
- Selección del idioma adecuado para el PLC.

Ajuste de la fecha y hora del PLC

Este ajuste es imprescindible para que el PLC y el SCADA funcionen de forma sincronizada. Para llevar a cabo este cambio de hora se debe seguir la siguiente ruta en el programa SIMATIC S7: PLC→*Diagnostic/Setting* →*SER TIME of DAY* como se puede ver en la figura 25. La hora y fecha del PLC debe coincidir con la Hora UTC (GMT):

- hora local -1h en inviernos
- hora local -2h en verano

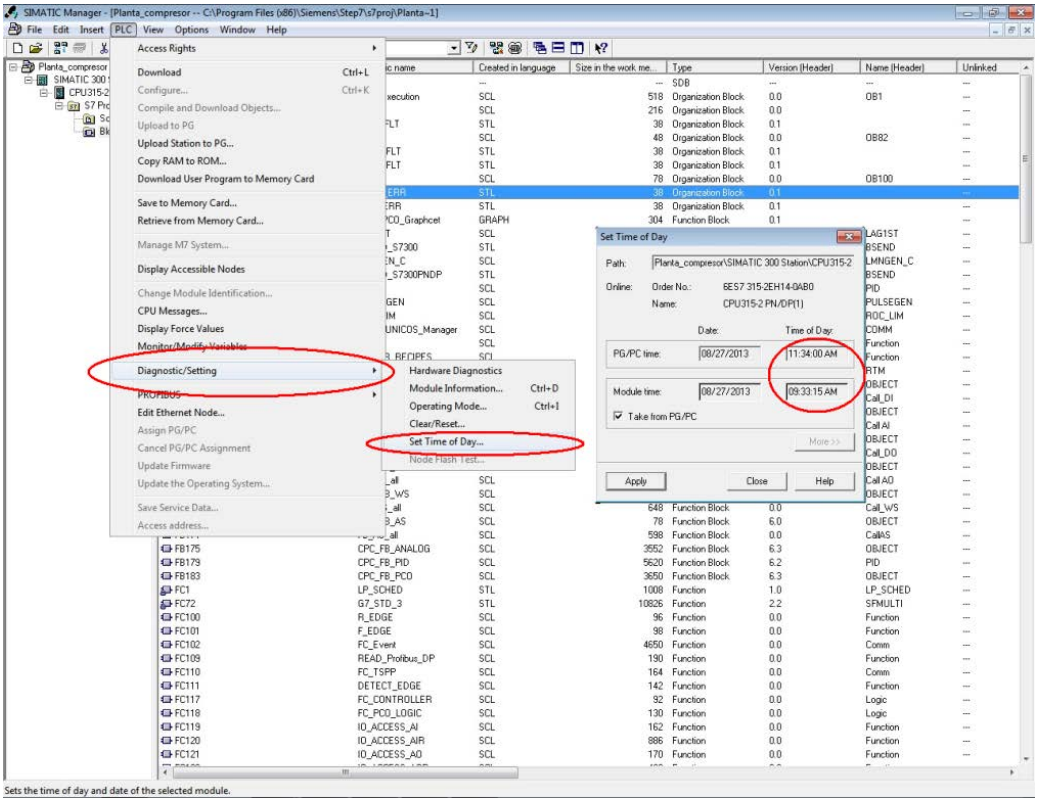


Figura 25: Ajuste horario PLC

Configuración de la conexión entre el PLC y el SCADA

Para el correcto intercambio de datos entre el PLC y el SCADA (PVSS) se debe crear una nueva conexión. Para ello debe abrirse el menú de conexiones del SIMATIC S7, que se encuentra en la barra superior de la ventana principal (NetPRO), seleccionar la CPU del PLC y con el botón derecho añadir una nueva conexión. Se deben rellenar los campos como se ve en la figura 26.

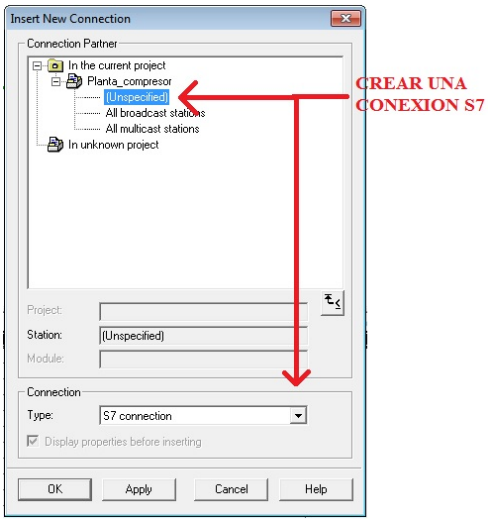


Figura 26: creación de la conexión entre PLC y SCADA

Una vez creada esta nueva conexión se desplegara una ventana para incluir las características de esta. Estos campos deben rellenarse como se en la figura 27:

Figura 27: Parámetros de la conexión de PVSS y SIMATIC I

La IP a introducir en el campo inferior derecho tiene que ser la de la tarjeta de red instalada en el ordenador.

Para terminar de configurar la conexión se debe pinchar sobre *Address Details*, (en la esquina inferior) e introducir los valores que se detallan en la figura 28

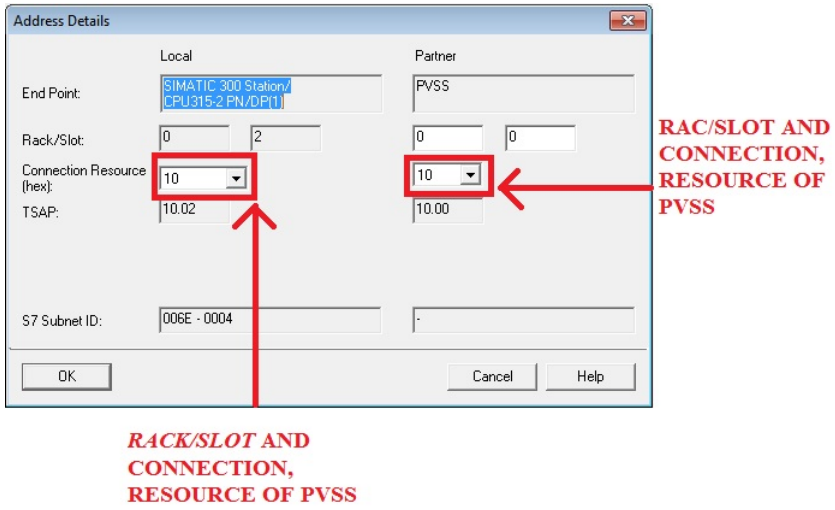


Figura 28: Parámetros de la conexión de PVSS y SIMATIC II

Una vez configurado con estos pasos, solo queda realizar un último ajuste en la conexión para evitar posibles errores con la sincronización del PLC con el PVSS. Este ajuste se lleva a cabo mediante la ruta: *Hardware → PN/IO → General → Properties → usar router*, como se puede ver en la figura 29.

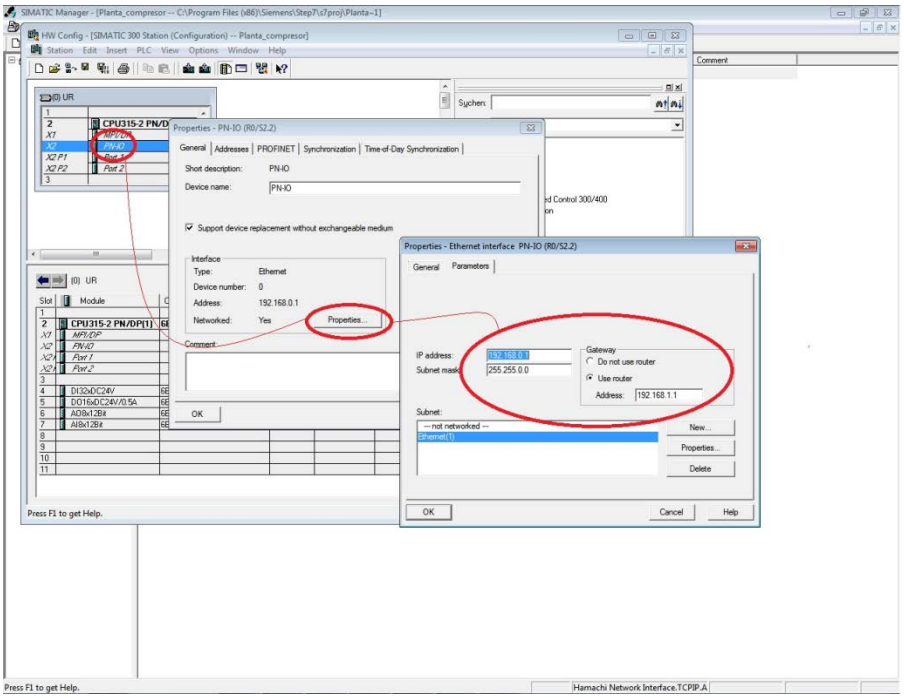


Figura 29: Configuración de las IP del PLC

Selección del idioma adecuado para el PLC

SIMATIC S7 ofrece trabajar en dos idiomas, inglés y alemán. Para trabajar con UNICOS-CPC6 se debe configurar en inglés. Esto es debido a que el generador UAB crea los archivos de lógica en inglés, y la incorrecta configuración del proyecto va a producir conflictos en la importación.

Para llevar a cabo este cambio de idioma se deben seguir los pasos que se detallan en la figura 30.

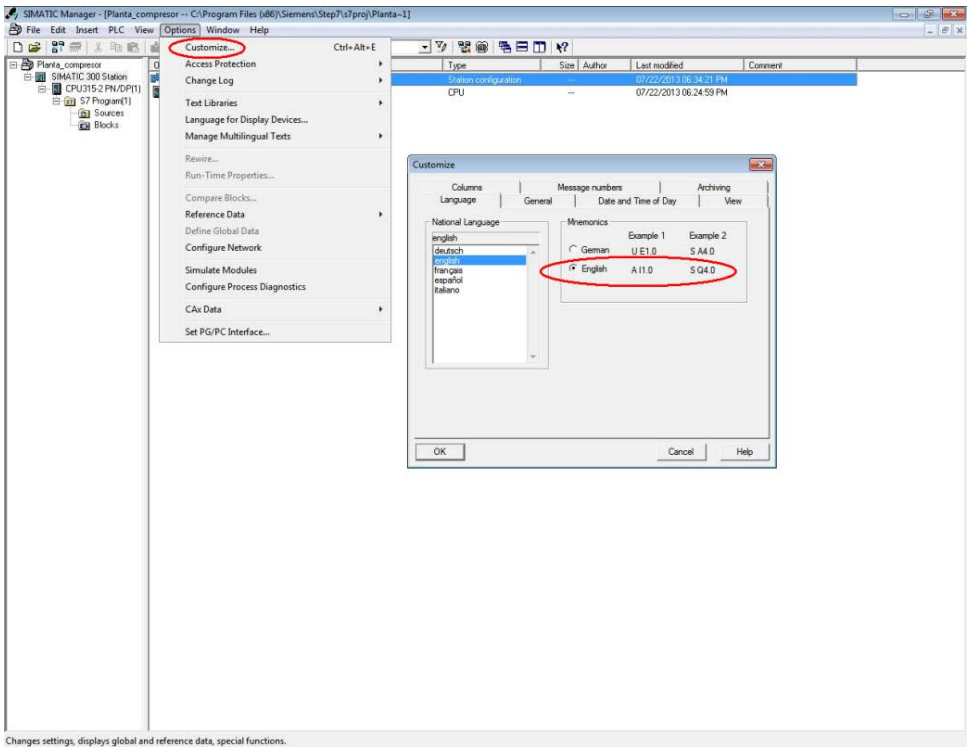


Figura 30: Selección del idioma

Una vez realizados estos ajustes el proyecto SIMATIC S7 ya está configurado para importar los archivos generados. Para ello se deben seguir los siguientes pasos:

1. Ir a *BASELINE* en la carpeta del proyecto UAB y descomprimir el archivo *ucp-plc-siemens-6.3*

2. Importar estos archivos *SIMATIC S7* desde →*File*→*Open*

3. Seleccionar librerías y buscar la librería dentro del proyecto como se ve en la figura 31.

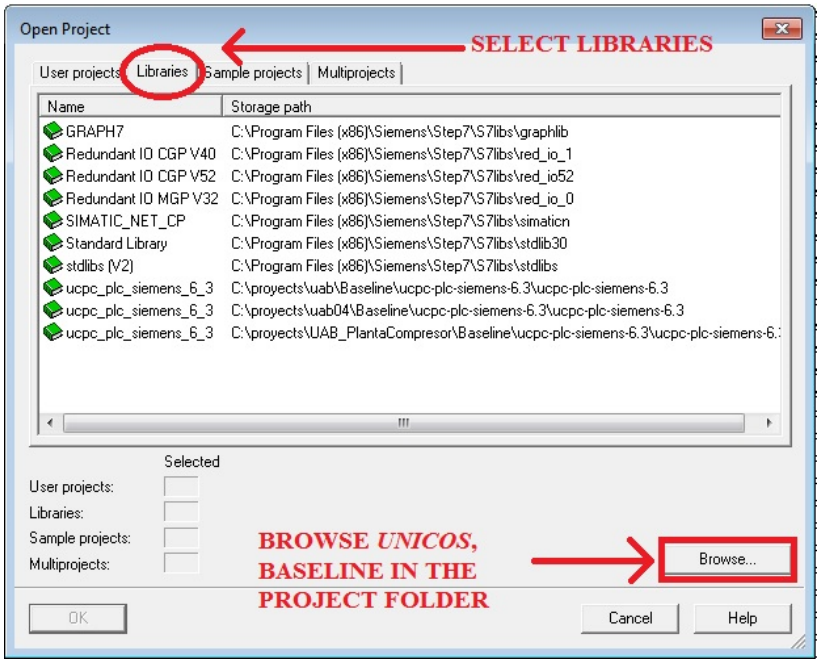


Figura 31: Librerías SIMAATIC S7

4. Copiar todas las fuentes y los bloques de la librería de UNICOS-CPC6 al proyecto nuevo. Estas fuentes pueden verse en la figura 32.

    a. Las fuentes son:
        i. *CPC_BASE_Unicos*: Contiene declaraciones de tipos de datos y funciones básicas.
        ii. *CPC_FB_AA*: Analog Alarm FB.
        iii. *CPC_FB_AI*: Analog Input FB.
        iv. *CPC_FB_AIR*: Analog Input Real FB.
        v. *CPC_FB_ANADIG*: Analog Digital FB.
        vi. *CPC_FB_ANALOG*: Analog FB.
        vii. *CPC_FB_AO*: Analog Output FB.
        viii. *CPC_FB_AOR*: Analog Output Real FB.
        ix. *CPC_FB_APAR*: Analog Parameter FB.
        x. *CPC_FB_AS*: Analog Status FB.
        xi. *CPC_FB_DA*: Digital Alarm FB.
        xii. *CPC_FB_DI*: Digital Input FB.
        xiii. *CPC_FB_DO*: Digital Output FB.
        xiv. *CPC_FB_DPAR*: Digital Parameter FB.
        xv. *CPC_FB_LOCAL*: Local FB.
        xvi. *CPC_FB_ONOFF*: On Off FB.
        xvii. *CPC_FB_PCO*: Process Control Object FB.
        xviii. *CPC_FB_PID*: Controller FB.
        xix. *CPC_FB_WPAR*: Word Parameter FB.
        xx. *CPC_FB_WS*: Word Status FB.

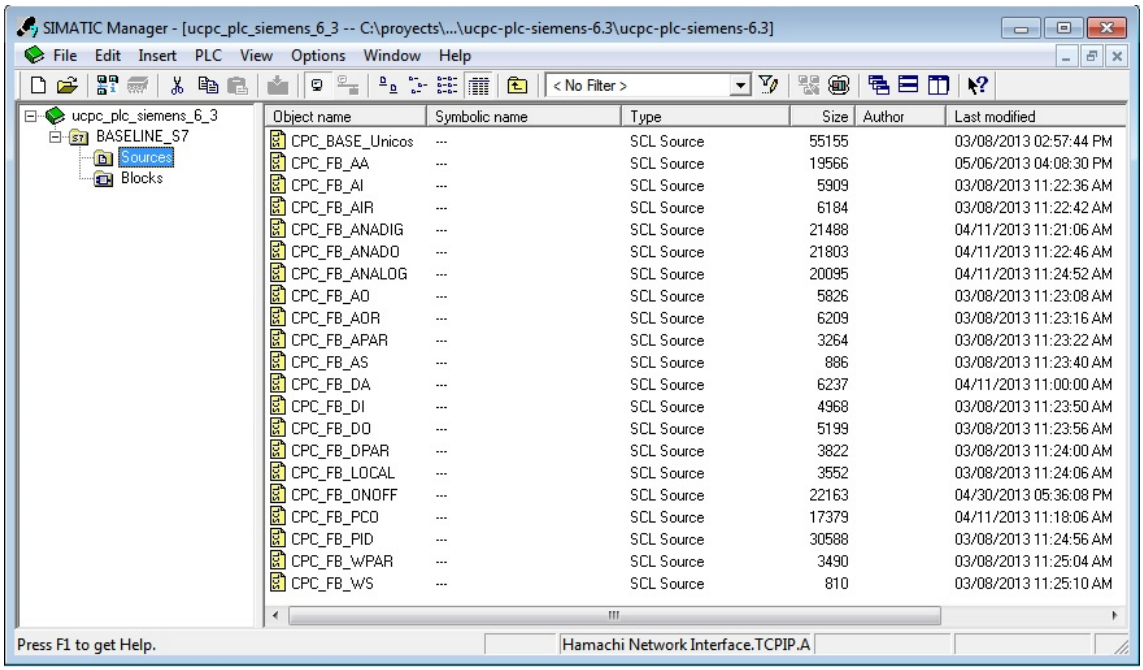


Figura 32: Fuentes de la librería importada desde BASELINE

b. Los bloques de la librería pueden verse en la figura 33:

i. *OB80*, *OB85*, *OB86*, *OB121*, *OB122*: tratamiento de errores de OBs.
ii. *FB9*: LAG1ST, intercambio de datos no coordinado.
iii. *FB12*: BSEND, bloque de intercambio de datos.
iv. *FB13*: LMNGEN_C, viene de la librería Modular PID library.
v. *FB19*: PID, viene de la librería Modular PID library.
vi. *FB20*: PULSEGEN, viene de la librería Modular PID library.
vii. *FB22*: ROC_LIM, viene de la librería Modular PID library.
viii. *FC1*: LP_SCHED, programador de tiempo que controla los sampling time de los
ix. controller.
x. *FC72*: función para la ejecución del grafcet.
xi. *VAT_TSPP*: tabla de visualización que permite una rápida resolución de problemas
xii. de las comunicaciones TSPP.
xiii. *SFB3*, *SFB4*, *SFB5*: IEC timers.

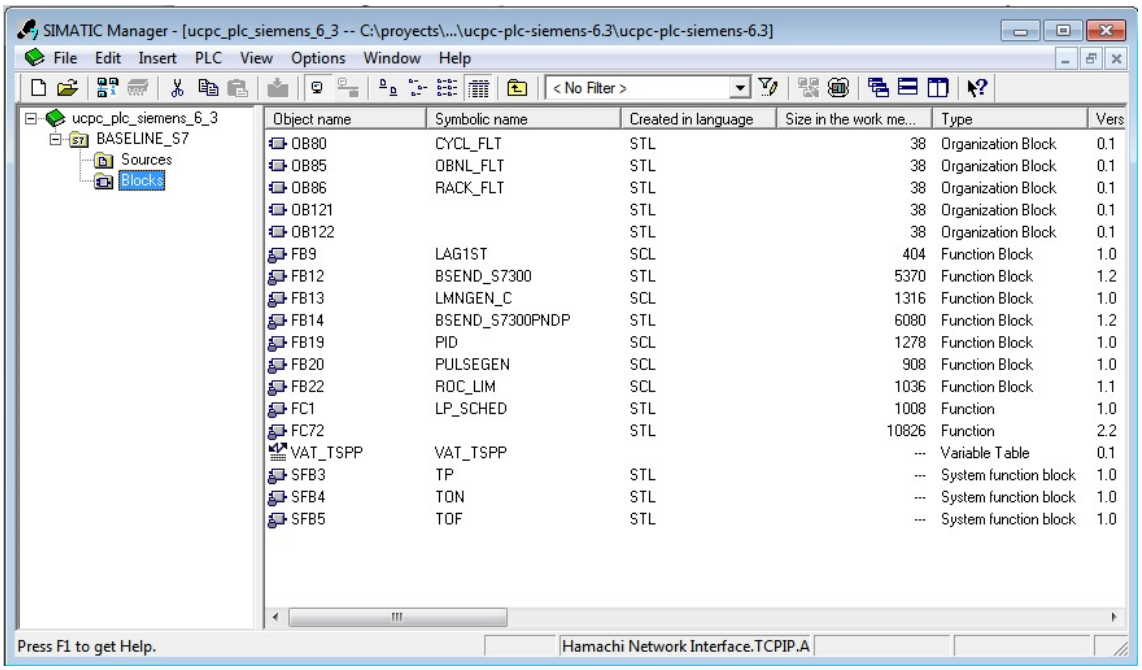


Figura 33: Bloques importados de la librería *BASELINE*

5. A continuación deben importarse las fuentes, de lógica y de archivos generados por UAB. En ambos casos la importación se lleva a cabo desde la opción insertar fuente externa, como se ve en la figura 34.

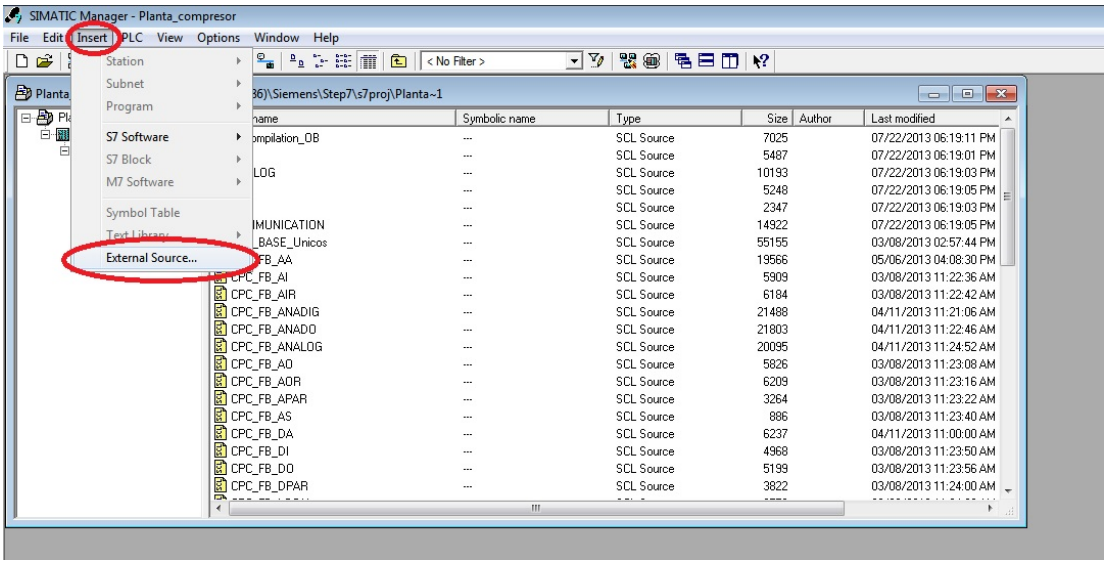


Figura 34: Importación de fuentes de instancia y de lógica I

6. La opción de insertar fuente externa despliega una ventana donde se debe incluir la ruta hasta los archivos generador. Estos archivos se encuentran en la carpeta donde se haya creado el proyecto UAB, *Project/Output/S7InstanceGenarator* en el caso de las instancias y *Output/S7LogicGenarator* para la lógica. En ambos casos se seleccionan todos los archivos y se abren. En la figura 35 puede verse el contenido de una de estas carpetas.

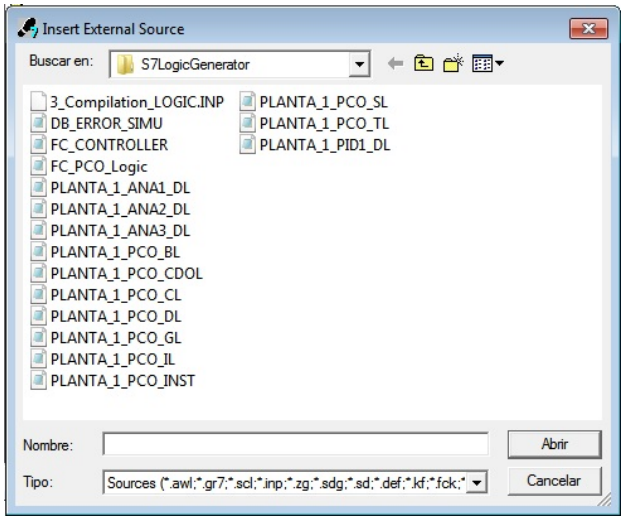


Figura 35: Importación de fuentes de instancia y de lógica II

7. Si se han seguido los pasos descritos el proyecto debe tener el aspecto de la figura 36.

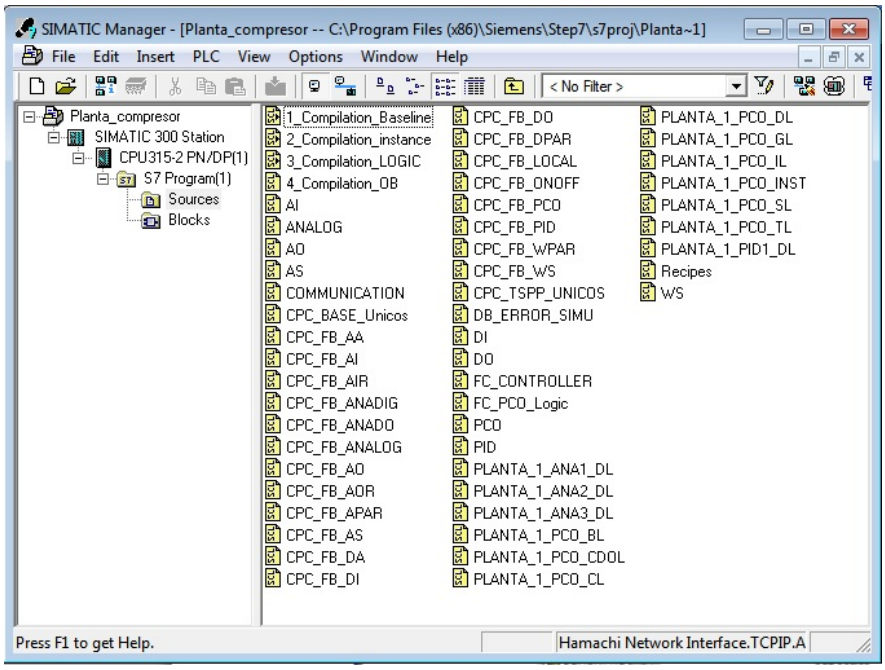


Figura 36: Fuentes del proyecto UNICOS CPC6

8. Por último debe importarse la tabla de símbolos, para ello se abre la tabla ya existente, como se ve en la figura 37, y se selecciona la opción de importar, como se ve en la figura 38.

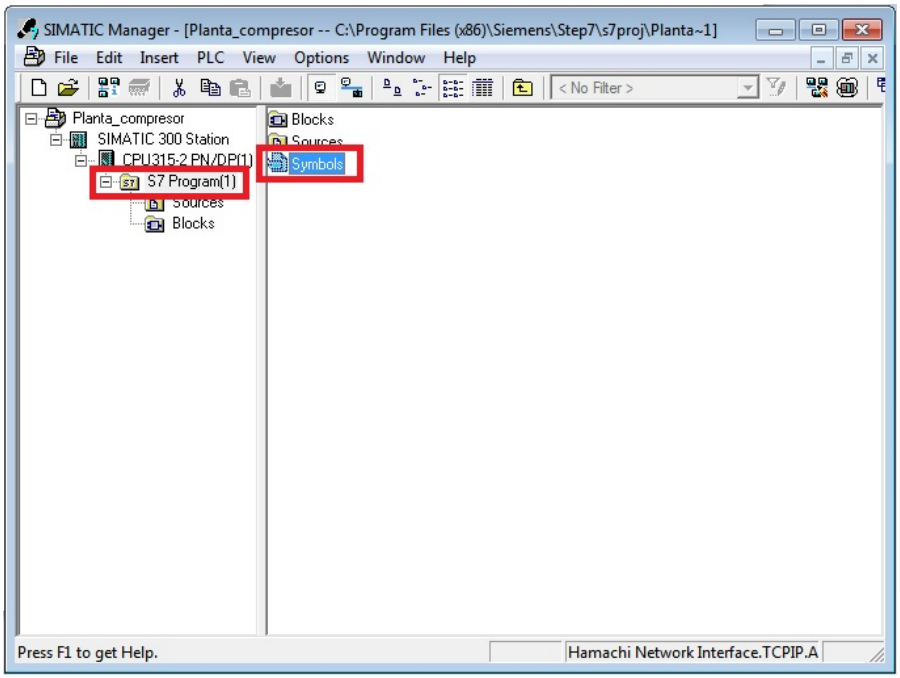


Figura 37: Importación de símbolos

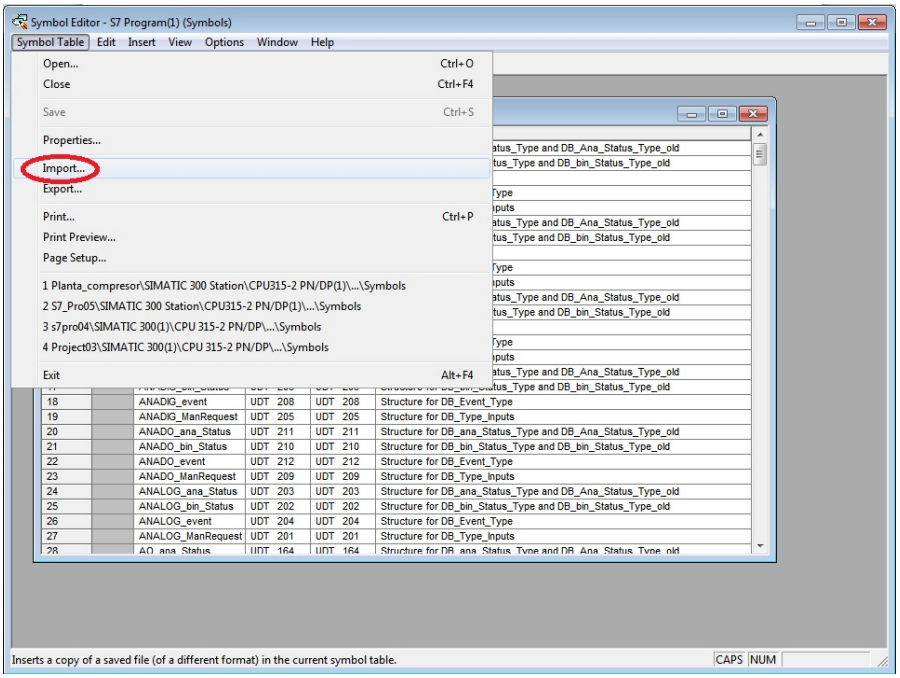


Figura 38: Importación de símbolos

Se debe abrir el archivo de símbolos del proyecto generado por UAB como se ve en la figura 39 ubicado en: *\Project\Output\S7InstanceGenerator*.

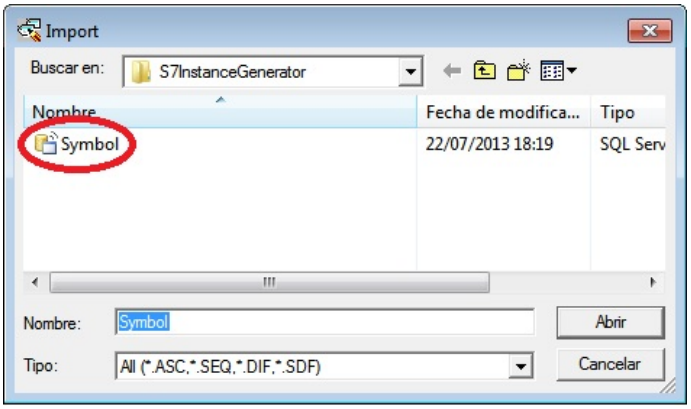


Figura 39: Importación de los símbolos del sistema generado por UAB

Es importante salvar la tabla de símbolos antes de cerrar la ventana que se ve en para que esta importación tenga validez.

9. Una vez importados todos los archivos se deben compilar las siguientes cuatro fuentes:

    a. 1_Compilation_Baseline
    b. 2_Compilation_instance
    c. 3_Compilation_LOGIC
    d. 4_Copilation_OB

La compilacion de estas fuentes debe hacerse por orden, según aparece en la lista superior, si no es asi podrian aparecer errores. Para llevar a cabo la compilacion se deben abrir cada una de ellas y pinchar en →*File*→*Compile* como puede verse en la figura 40.

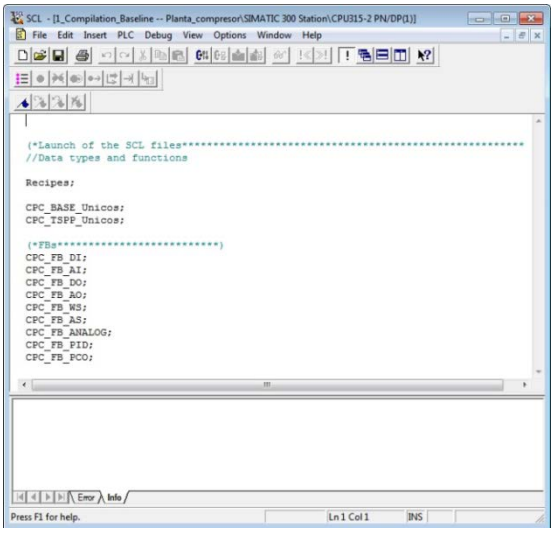


Figura 40: Compilación de fuentes

Es muy importante que no salgan errores durante las compilaciones ya que si se produce algún error no se puede continuar. Una vez compilada la última fuente el proyecto está completamente creado

3.5.3.3 PVSS

Para continuar con la creación del proyecto al igual que en SIMATIC se debe crear un proyecto en PVSS, continuando con la filosofía UNICOS. Para ello se importan los archivos creados por el generador UAB.

Durante de la creación del proyecto se deben tener en cuenta varias indicaciones previas para evitar la aparición de problemas posteriores:

- En primer lugar el PLC y el ordenador debe estar sincronizados, puestos en hora como se explicó en la creación del proyecto de SIMATIC: UTC -1 hora en invierno y UTC -2 en verano.

- La herramienta "*PVSS Project Administration*", deberá ser ejecutarla como administrador.

- No se utilizan los caracteres ' ' ni '-' en las carpetas que forman el *path* del proyecto, en su lugar se deben utilizar '_' o letras mayúsculas:

  '*C:\dev_disk\PVSS_projects\Demonstrator_SIEMENS*'

- No se deben crear proyectos en la carpeta de instalación de PVSS, se creara una carpeta directamente en el directorio raíz denominada *projects* que contenga todo el proyecto como se puede ver en la figura 41:

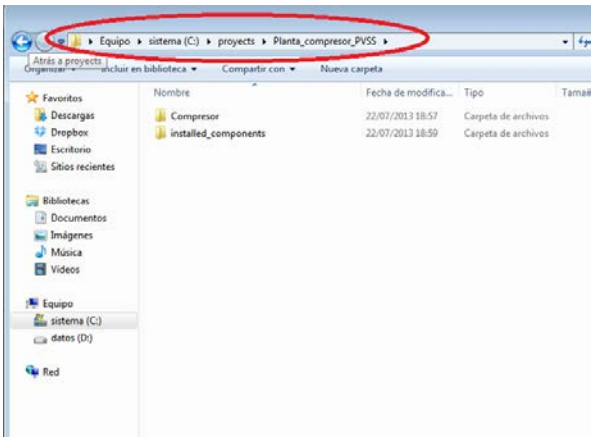


Figura 41: Carpeta creada para los proyectos PVSS

Una vez se tomen todas estas precauciones se puede comenzar con la creación del proyecto como se ve a continuación, mencionar que el CERN dispone también de un manual, donde se detallan estos pasos [14]:

1. Primero se ejecuta la herramienta *PVSS Project Administration* como administrador y se pincha en nuevo proyecto como se puede ver en la figura 42.

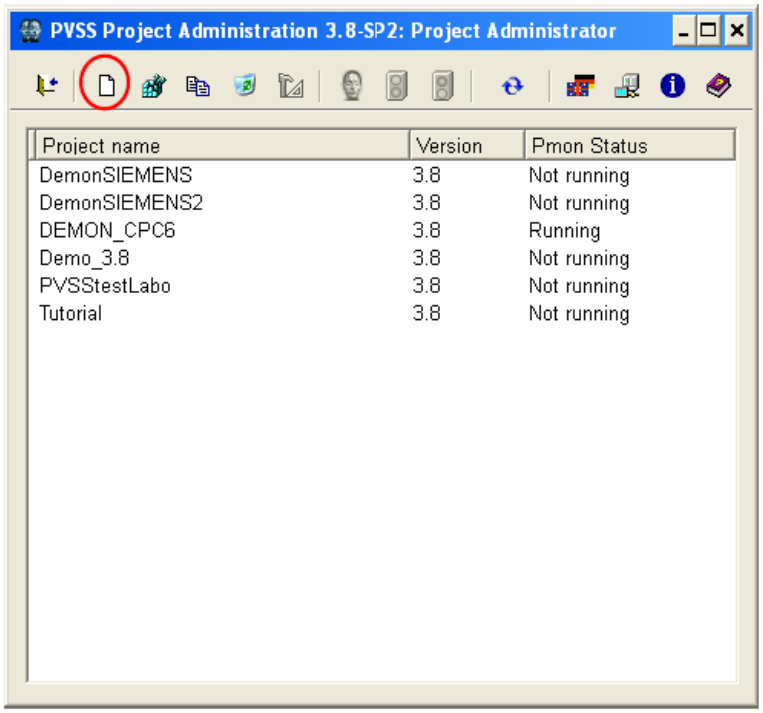


Figura 42: Creación de nuevo proyecto

2. A continuación se debe marcar la pestaña de *Distributed Project* como se ve en la figura 43 y pulsar en siguiente.

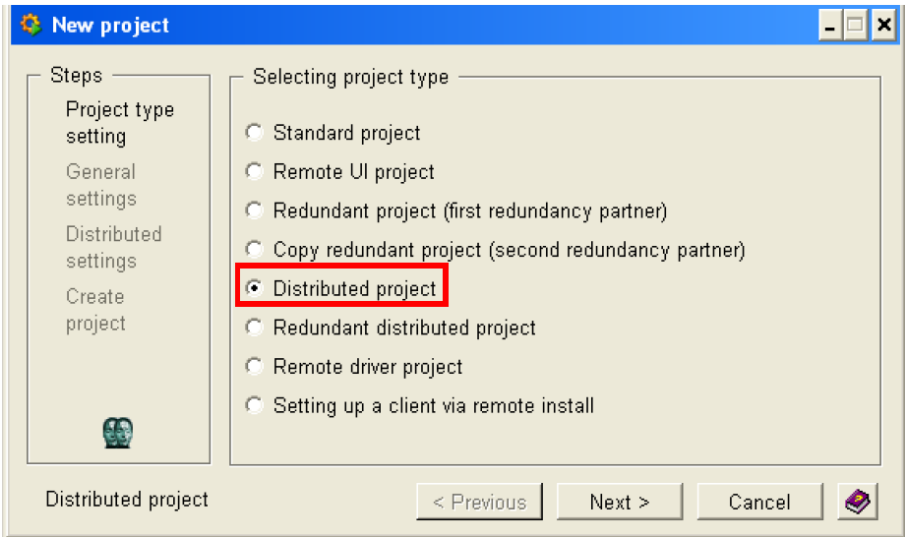


Figura 43: Proyecto Distribuido

3. En la siguiente ventana (figura 44) se le asigna un nombre al proyecto, y se ubica en la carpeta creada previamente, además se deben seleccionar “English - US” como idioma.

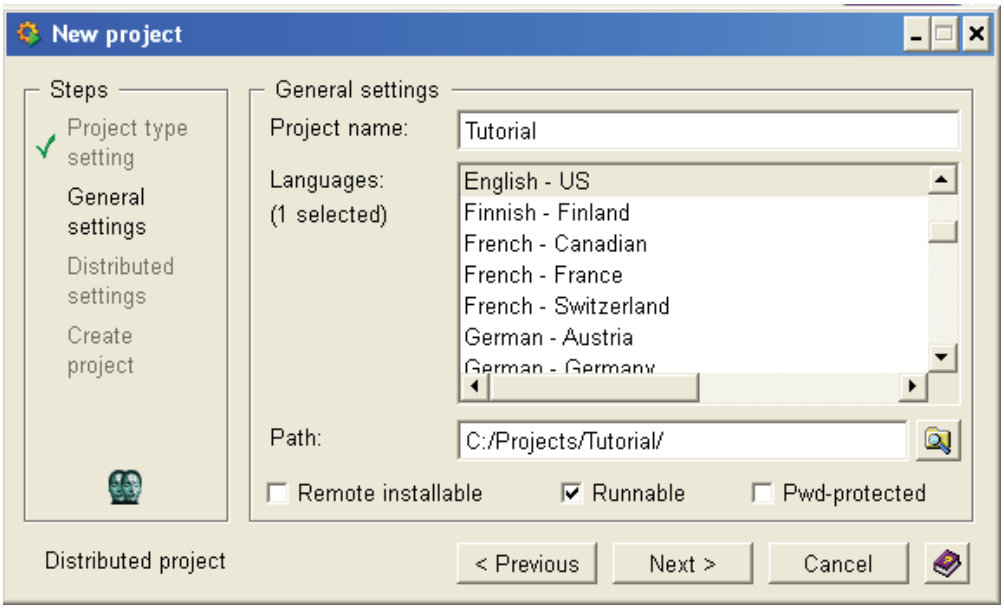


Figura 44: Selección de idioma y nombre

4. Se debe elegir ahora un número que identifique nuestro sistema, como se ve en la figura 45. Es importante que este número sea utilizado por otro proyecto.

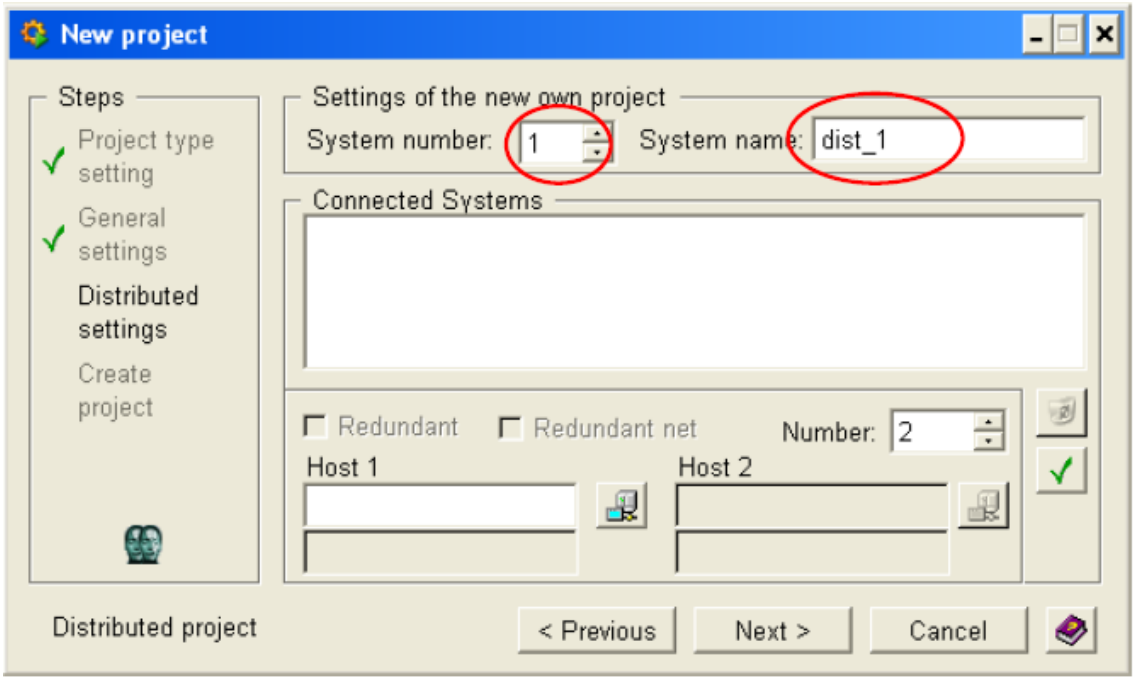


Figura 45: Número de sistema

5. Por último, se debe comprobar que la configuración es correcta como se ve en la figura 46.

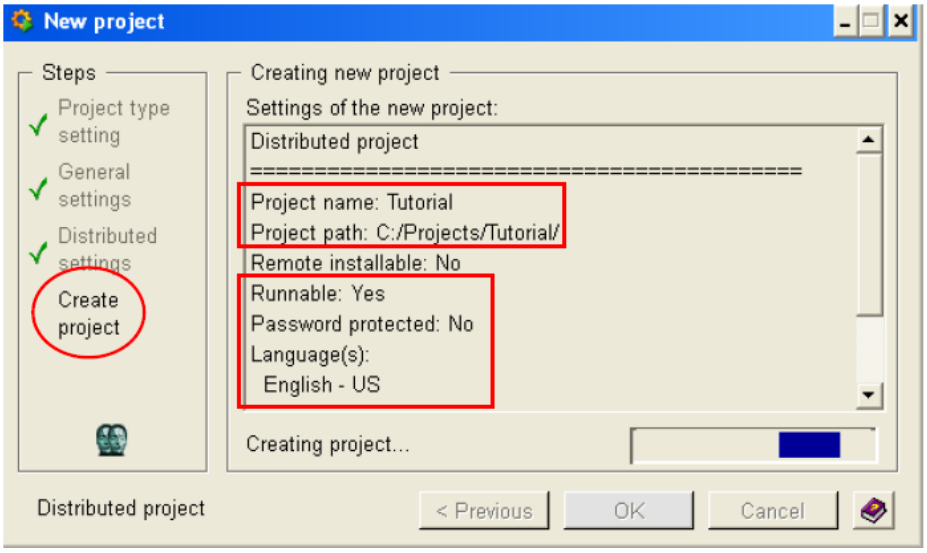


Figura 46: Creación de proyecto PVSS

6. Una vez creado el proyecto se debe abrir la consola y eliminar los archivos *Archive Manager* desde el *-num 1* hasta el *–num 5*, como se ve en la figura 47, para ello se puede utilizar el botón de borrar (parte derecha inferior) o el menú que se despliega con el botón derecho.

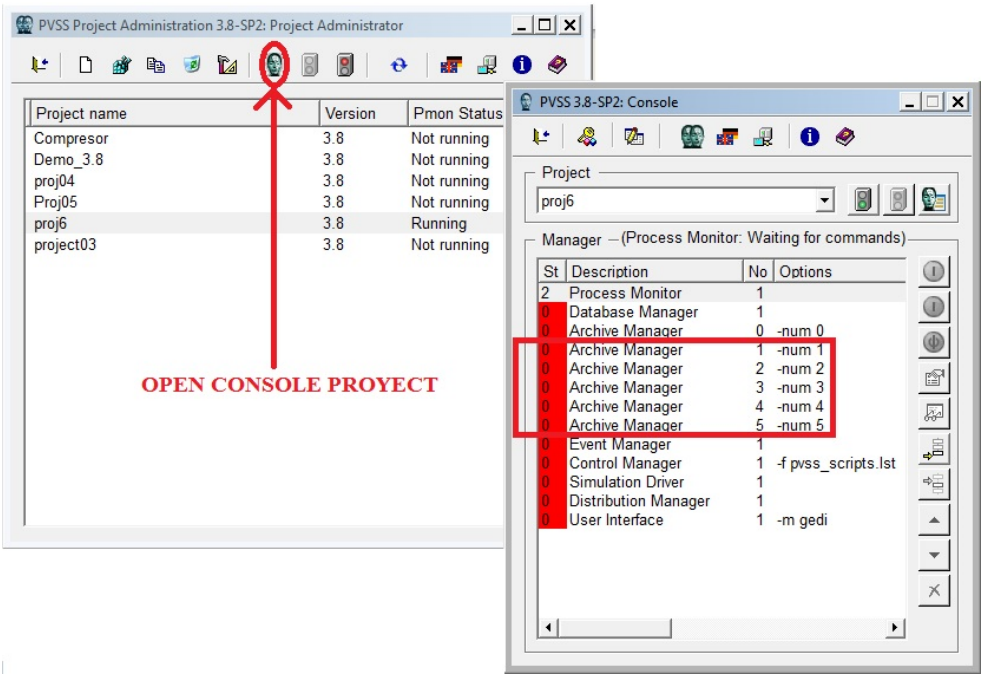


Figura 47: Eliminación de los archivos manager indicados

7. A continuación se debe parar el archivo *Distribution Manager* de la consola. Se selecciona y se pincha sobre el botón de la parte derecha que se indica la parte izquierda de la figura 48. Si no se para, se debe abrir el elemento y cambiar la pestaña de *always* a *manual* y repetir el proceso.

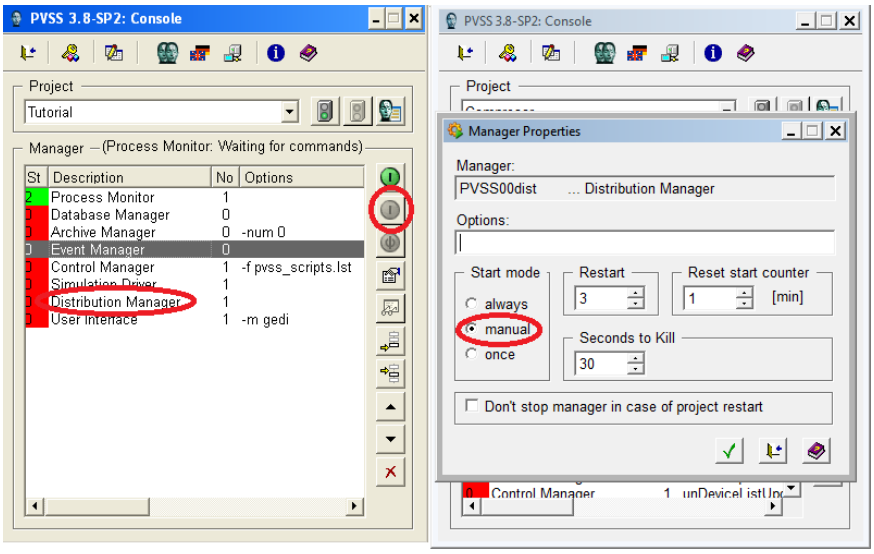


Figura 48: Parada del Distribution Manager

8. El proyecto ya está creado y listo para la importación de los archivos generados por UAB. En primer lugar se debe ir al proyecto UAB y descomprimir en carpetas temporales los siguientes archivos contenidos en la *BASELINE* generada por UAB (ver figura 49):

   - *Fw-installation tool-5.0.4*
   - *Unicos-framework-5.3.2*
   - *Ucp-win-cc-oa-6.1.2*

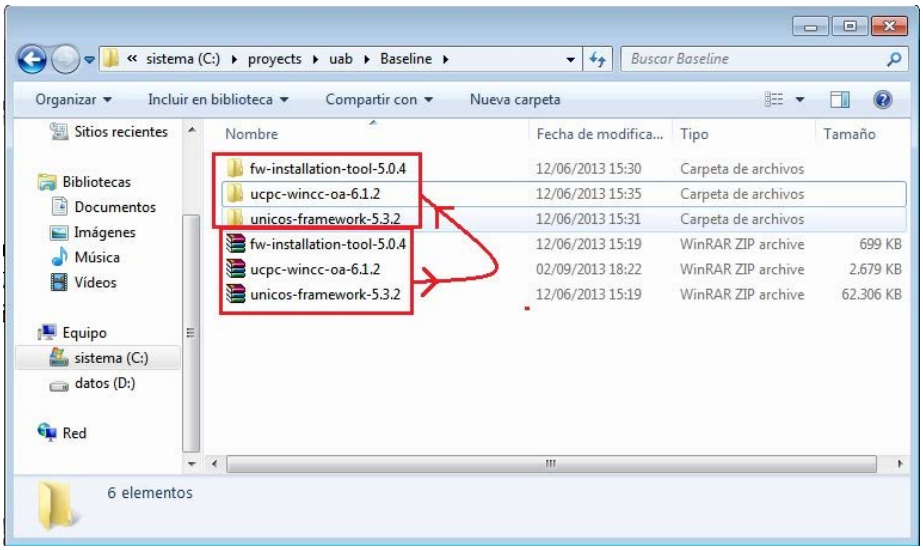


Figura 49: Extracción de ficheros de UAB/Baseline

9. En primer lugar se va a utilizar el archivo *fw-intallation-tool-5.0.4*. se deben copiar todos los elementos contenidos en esta carpeta y los copiados en la carpeta del proyecto PVSS. Es muy importarte que los archivos copiados tienen que sobrescribir los anteriores como se puede ver en la figura 50.

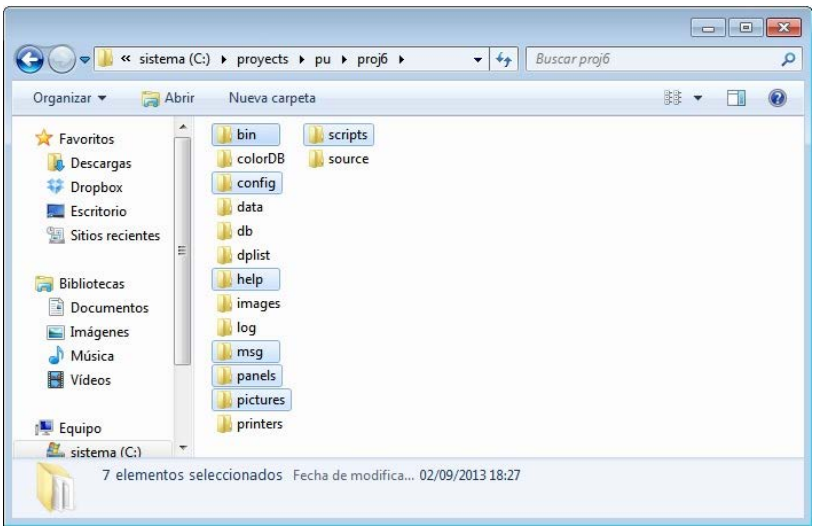


Figura 50: Archivos fw-intallation-tool-5.0.4

10. Se debe crear ahora un elemento en nuestra consola para instalar los elementos de UNICOS en el proyecto, con la herramienta *Append a new manager*. Se crea un nuevo elemento como se puede ver en la figura 51 con el nombre *-p fwInstallation/fwInstallation.pnl*.

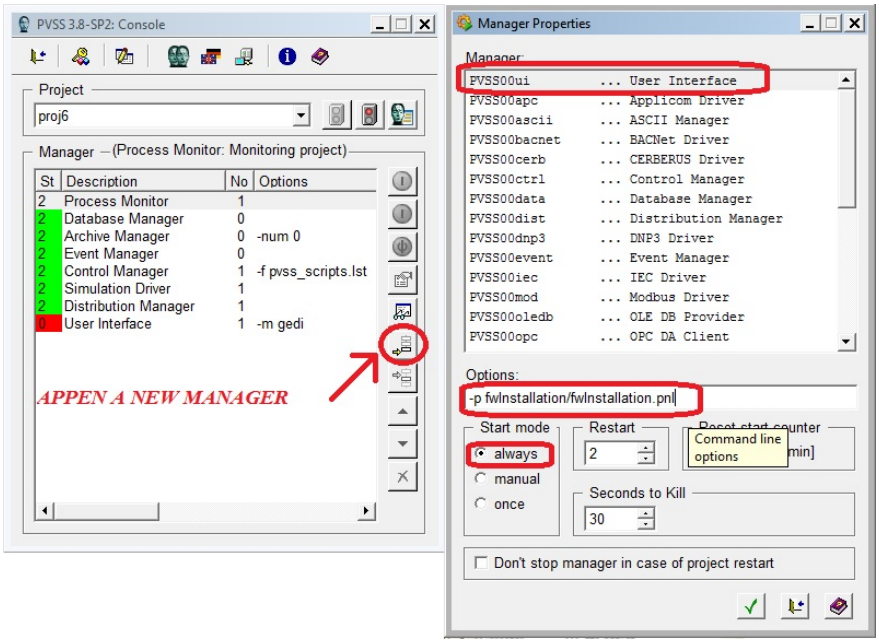


Figura 51: Creación del elemento *FwInstallation*

11. Ahora se arranca el nuevo elemento creado como se ve en la figura 52.

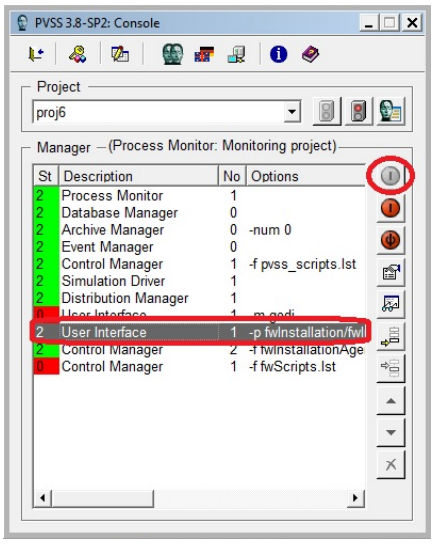


Figura 52: Arranque de FwInstallation

12. Una vez arrancado se debe crear un directorio donde se instalaran todos los elementos. Para ello se marca la pestaña, crear directorio, y se añade detrás de la última barra "*installed_components*" creando así una nueva carpeta donde se instalaran los componentes, como puede verse en la figura 53.

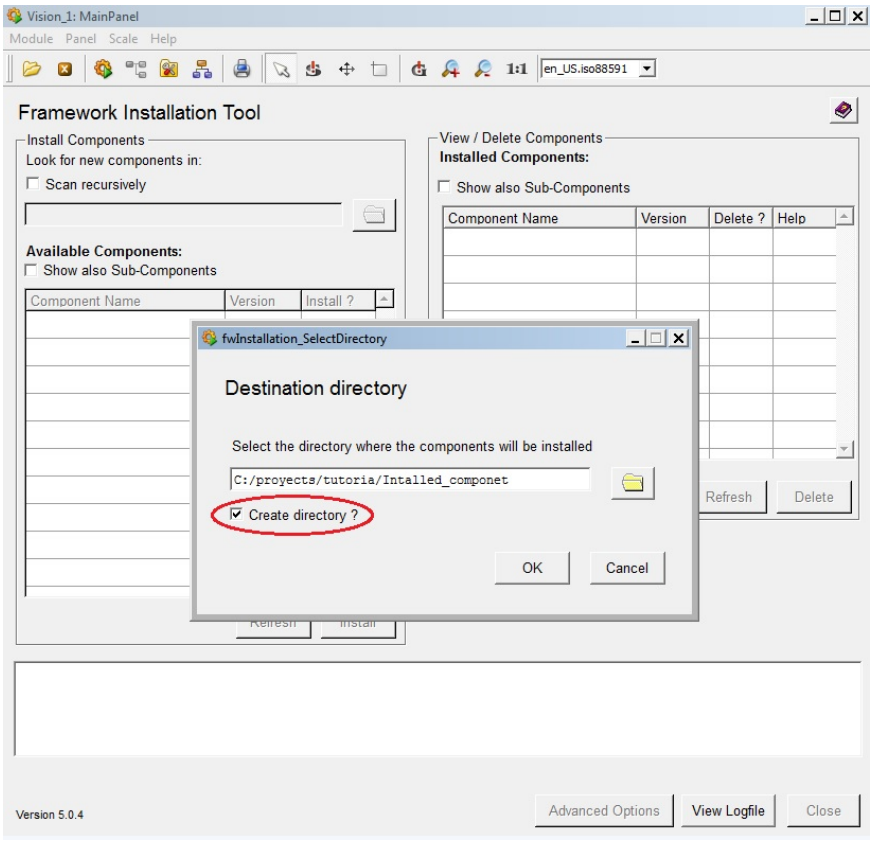


Figura 53: Creación de nuevo directorio

13. En la siguiente ventana da la opción de conectarlo a una base de datos, se debe pinchar en NO como se ve en la figura 54.

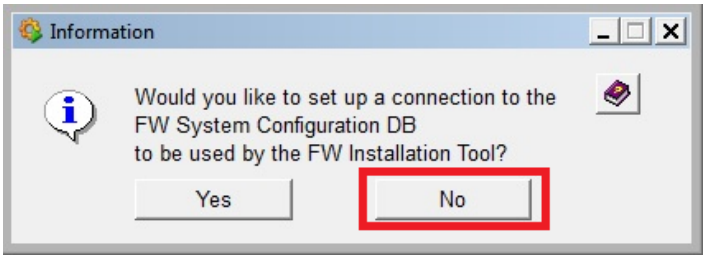


Figura 54: Conexión con base de datos

14. A continuación se deben buscar los componentes previamente extraídos en la carpeta del proyecto UAB en la carpeta *Unicos-framework-5.3.2*(ver figura 49). Para ello únicamente hay que incluir la ruta hasta este directorio, en la barra superior izquierda, como se ve en la figura 55. Si los elementos no aparecen marcar la opción *Scan recursively*.

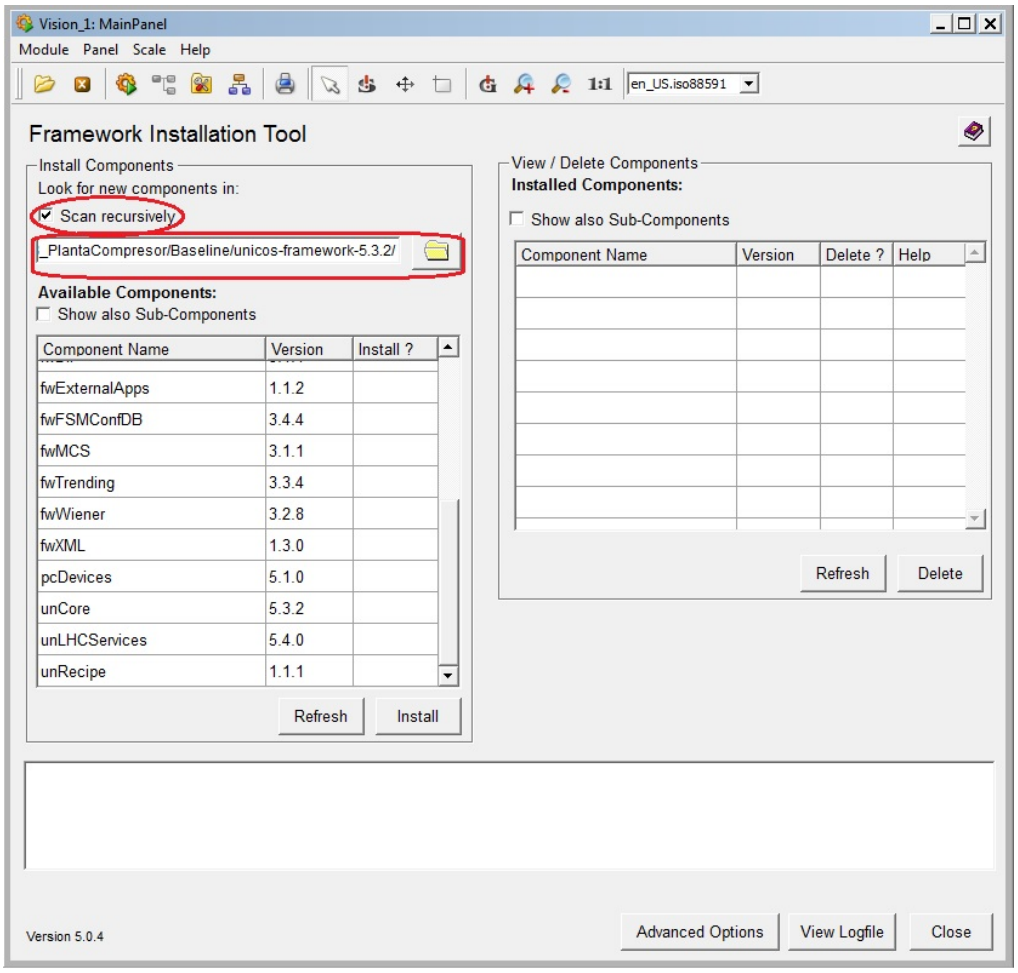


Figura 55: Búsqueda de componentes

15. Los componentes siguientes son los que se deben instalar. Para ello se seleccionan y se pulsa sobre instalar (ver figura 56):
    - *fwAccessControl*
    - *fwCore*
    - *fwTrending*
    - *unCore*
    - *fwConfigurationDB*
    - *unRecipe*

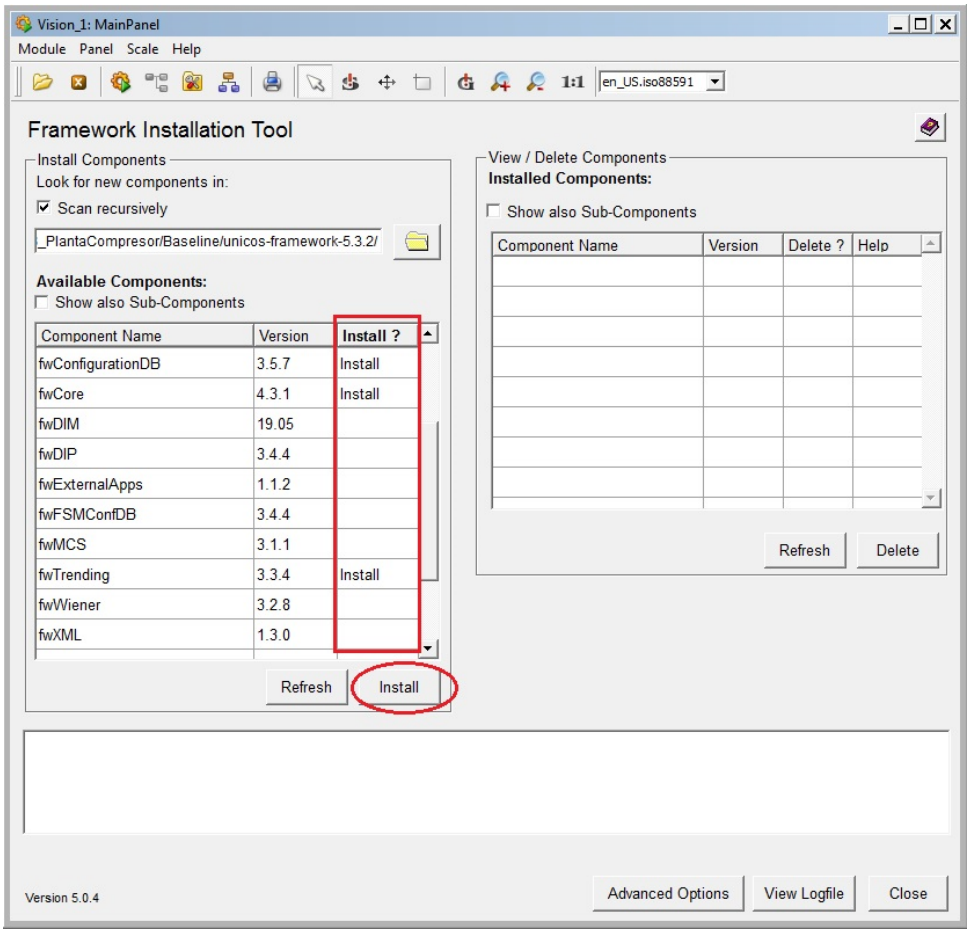


Figura 56: Instalación de componentes

16. Cuando el programa termine de instalar los componentes se debe reinicia el proyecto como se ve en la figura 57, puede tardar varios minutos, esperar hasta que se vuelva a iniciar por completo.

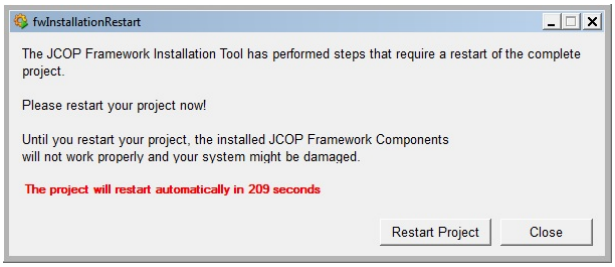


Figura 57: Reinicio de proyecto PVSS

17. Por último, una vez que el proyecto se haya reiniciado, se debe arrancar el elemento *fwScripts.Ist* y los nuevos componentes aparecerán en la consola Como se ve en la figura 58:

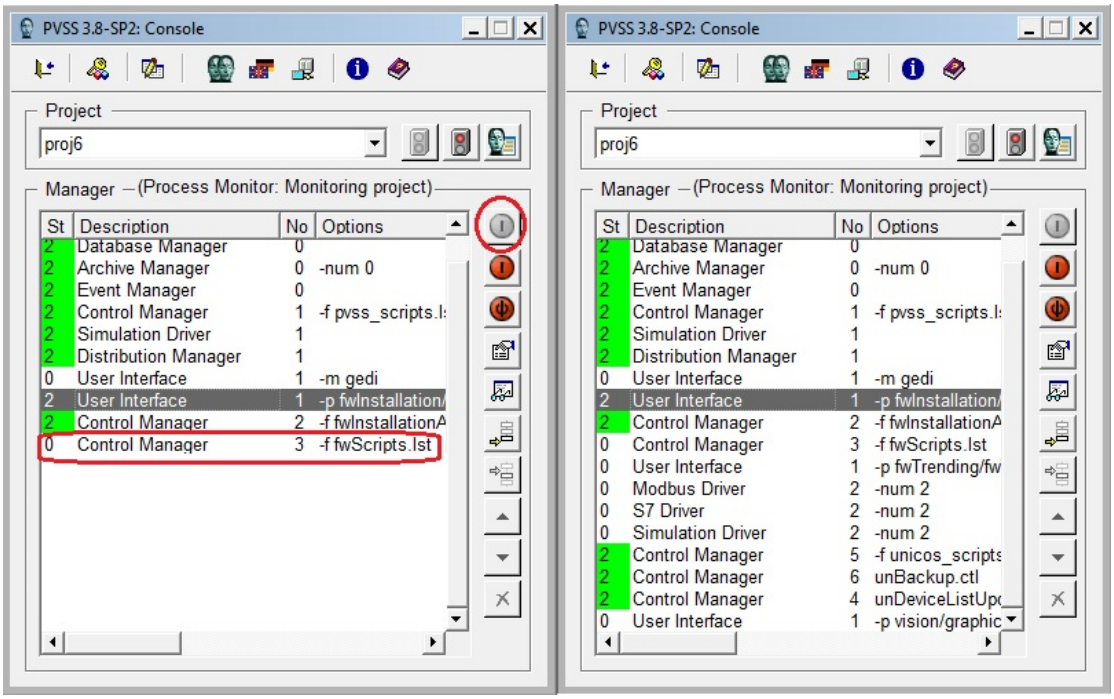


Figura 58: Arrancar fwScripts.Ist

18. Repetir ahora todo este proceso, desde el punto 14 al 17, pero esta vez con la carpeta de la *Baseline* de UAB *Ucp-win-cc-oa-6.1.2*.

    - Instalar ahora el elemento *unCPC6*
    - Reiniciar el proyecto
    - Arrancar *fwScripts.Ist*

19. Una vez están instalados todos los componentes se debe arrancar el elemento *Distribution Manager* que se ha parado en el punto 7 y a continuación arrancar el *-p visión/graphicalFrame/unicosHMI.pnl*.

20. Una vez este arrancado el elemento, se cliquea sobre la llave de la parte superior derecha e introducir la palabra *"admin"* en el Usuario y dejar el *pasword* vacío como se ve en la figura 59

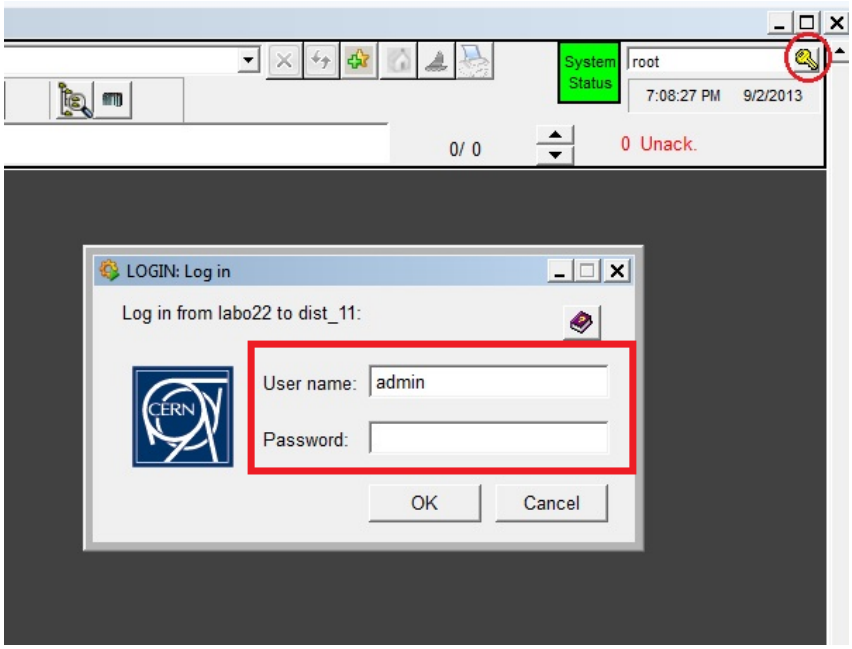


Figura 59: Acceso al proyecto PVSS como administrador

21. Ahora se debe hacer pinchar sobre el logo de UNICOS en la parte superior izquierda y seleccionar *Management>Utilities>Value Archive* como se ve en la figura 60:

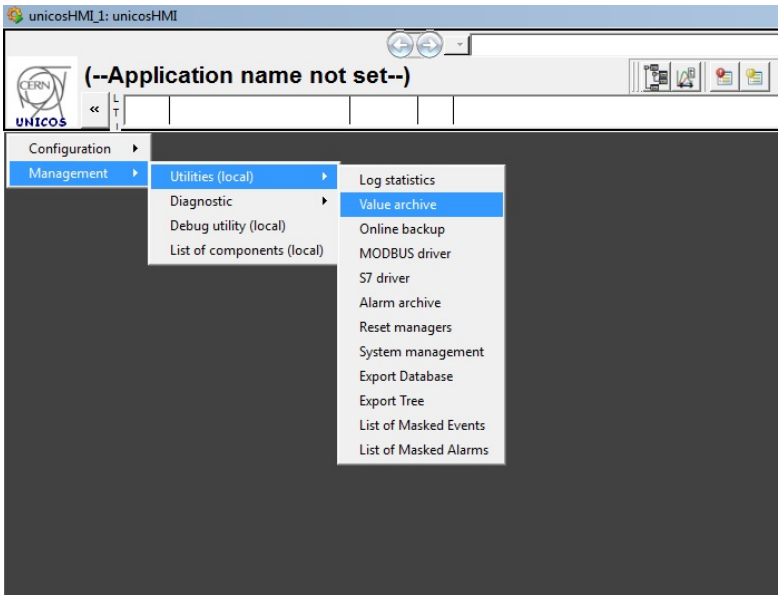


Figura 60: "Value archive"

22. En este punto se deben crear tres archivos, como se ve en la figura 61, con los nombres:
    - *boolean*
    - *analog*
    - *event*

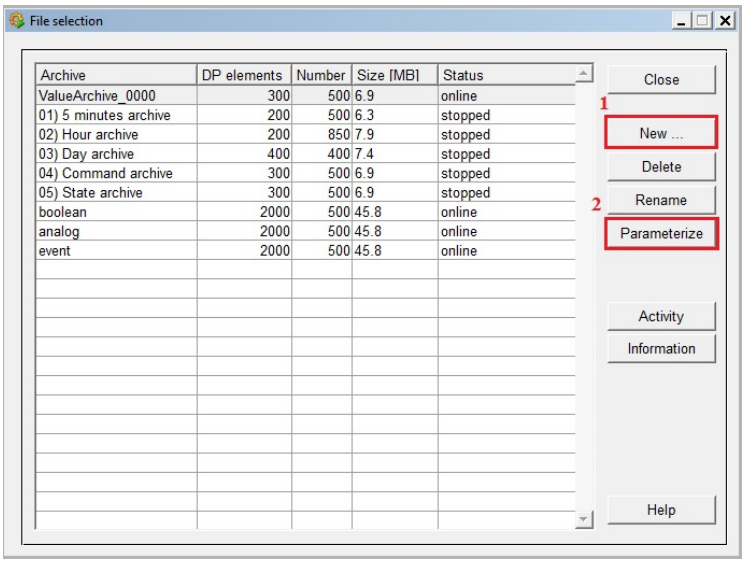


Figura 61: Creación de archivos Bool,Analog y Event

23. Pinchar en nuevo para crearles, y a continuación en parametrizar e introducir los siguientes valores en todos ellos como se ven en la figura 62:

    - *Max. number of DP elements: 200*
    - *Max. number of value entries: 500*
    - *Compression*
    - *Deletion: 2*
    - *Free entries after compression: 40*

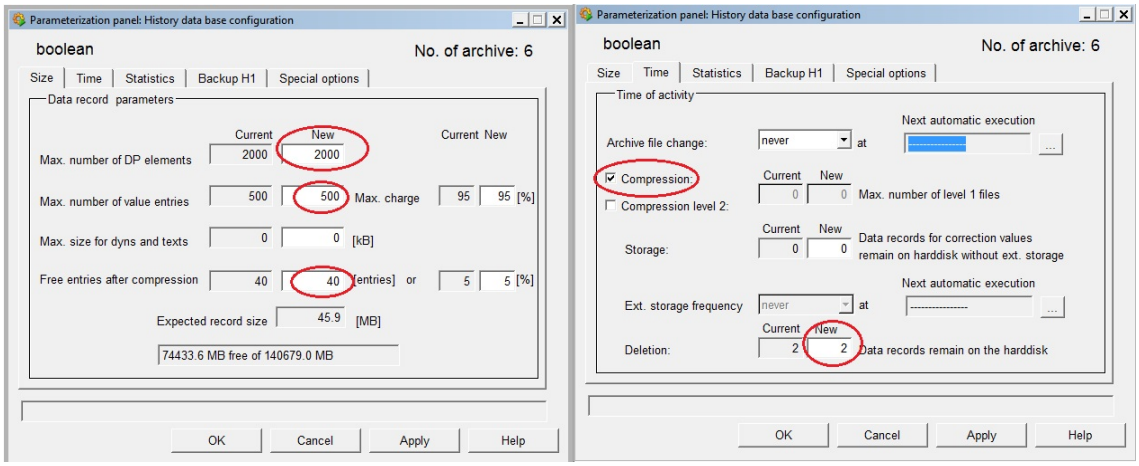


Figura 62: Parametrización de los archivos

24. Ahora se deben cerrar todas las ventanas hasta tener únicamente la consola y crear tres elementos como en la figura 63, igual que en el punto 11, asociados a los tres archivos que se han creado en el punto anterior. Deberán tener el nombre: -num 6, -num 7 y -num 8

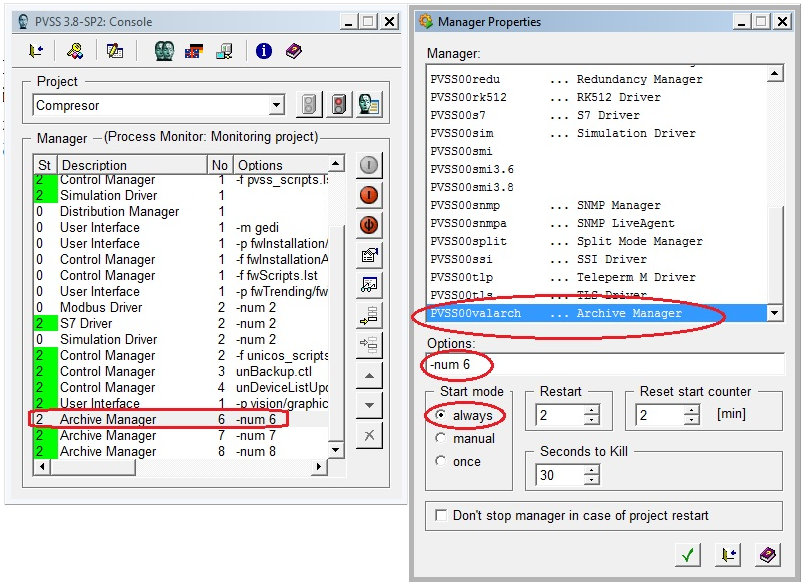


Figura 63: Creación de los "Archive manager"

25. Se debe arrancar de nuevo el *-pvisión/graphicalFrame/unicosHMI.pnl*, Para disponer de derechos de administrador, se deben introducir el usuario *admin*, como se vio anteriormente. A continuación seleccionar *Configuration>Import DataBase (local)*" como se ve en la figura 64.

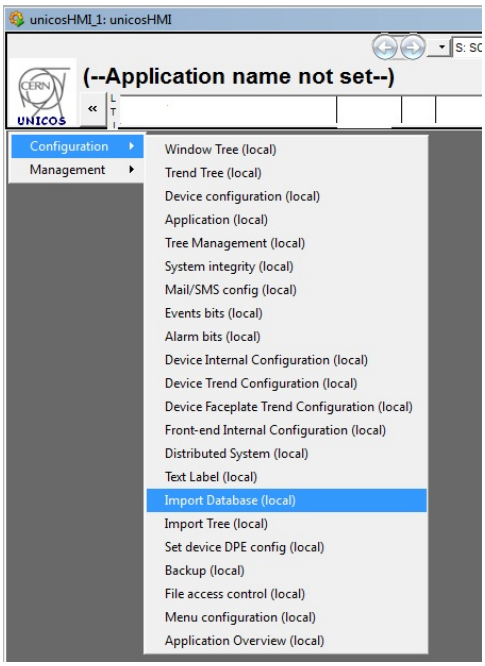


Figura 64: "Import DataBase"

26. En la pantalla que nos aparece hay que importar la información específica de nuestro proyecto generada por UAB, para ello se deben completar los campos como se ve en la figura 65.

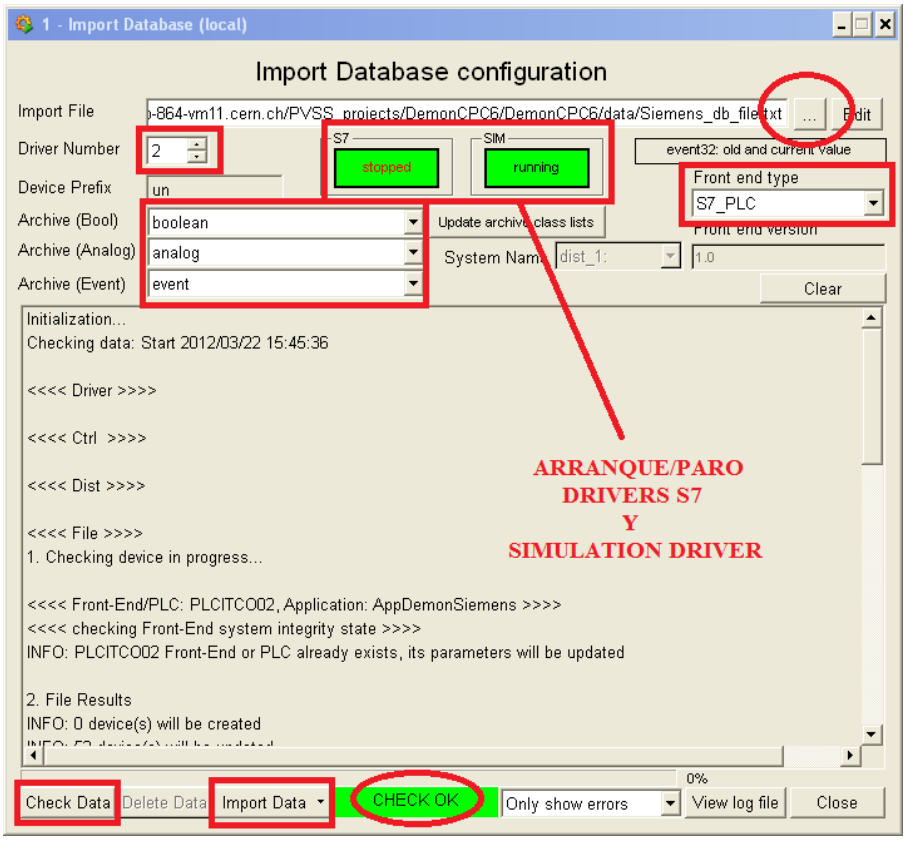


Figura 65: Importación de la lógica especifica

La mayor parte de los problemas en la creación del proyecto aparecen a la hora de completar esta pantalla, se deben seguir los siguientes pasos para evitar estos errores:

- Valores de los campos vacíos:
  - *Driver number : 2*
  - *Archive (Bool): boolean*
  - *Archive (Analog): analog*
  - *Archive (Event): event*
  - *Front end type: S7_PLC*
- Se deben indicar ahora la ruta hasta nuestro archivo en la parte superior de la pantalla, este archivo se encuentra en la carpeta del proyecto UAB en *Output/WinCCOAInstaceGenertor*
- Ahora con el driver *S7* parado y el *Simulation driver* arrancado se pincha sobre *Check Data*

- Cuando termine el proceso de chequeo, ya se pueden importar los datos pinchando sobre *Import Data*.

- Si se han realizado todos estos pasos correctamente, aparecerá un testigo en verde donde nos indica que la importación ha sido correcta

- Por último se debe parar el *Simulation driver* y arrancar el driver *S7*.

27. El proyecto ya está creado, por último hay que asegurarse que tanto la configuración como la conexión entre el PLC y el SCADA son correctas, para ello, abrir el *Front end Diagnostic* y cambiarla pestaña de *disable* a *enable*, como se ve en la figura 66.

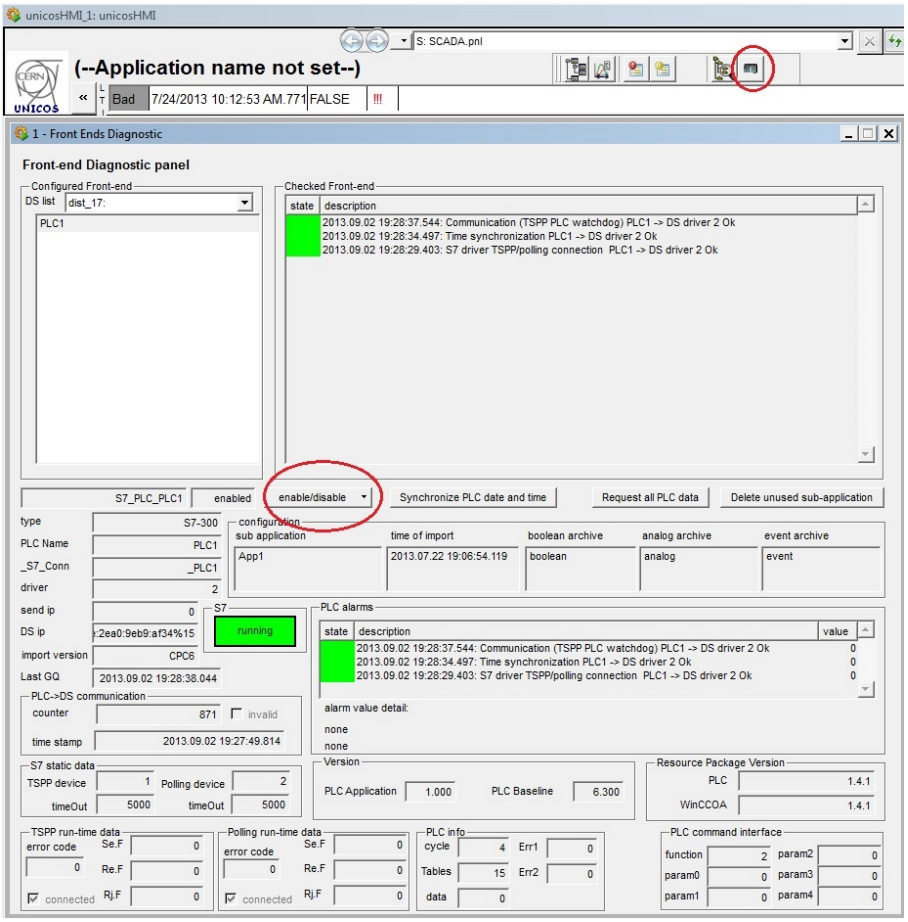


Figura 66: Conexión entre el PVSS y el PLC

Si se han seguido estos pasos correctamente, y el PLC está en modo *Run* los tres testigos de la parte inferior derecha deben de ponerse en verde. Si no es así los errores más comunes son los siguientes

- El PLC no está sincronizado, la hora que marca el PLC no es correcta

- Se debe poner la hora correcta como se indica el punto anterior "*Creación de un proyecto en SIMATIC*"

- La configuración de las IP no es correcta

- Este error es debido a que a la hora de la configuración del proyecto en UAB no se han introducido las IP correctamente, para resolverlo se debe entrar en la ventana de UNICOS, y abrir el *S7 driver* como se ve en la figura 67.

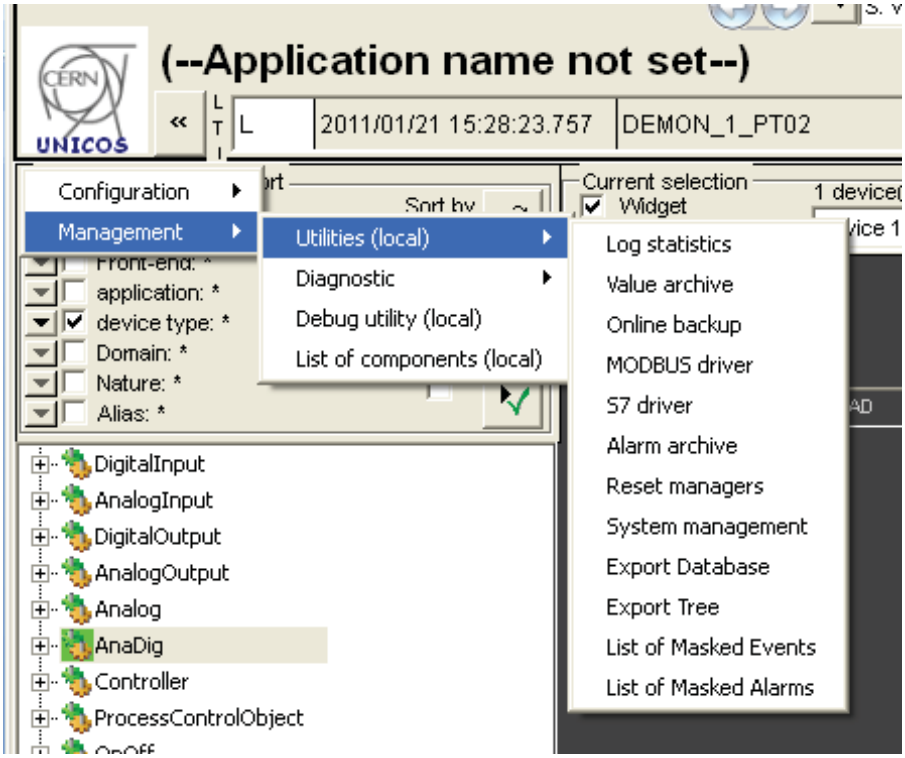


Figura 67: S7 Driver

A continuación se deben modificar las casillas necesarias con los valores correctos, como se ve en la figura 68. Una vez se apliquen estos cambios la conexión funcionara, correctamente.

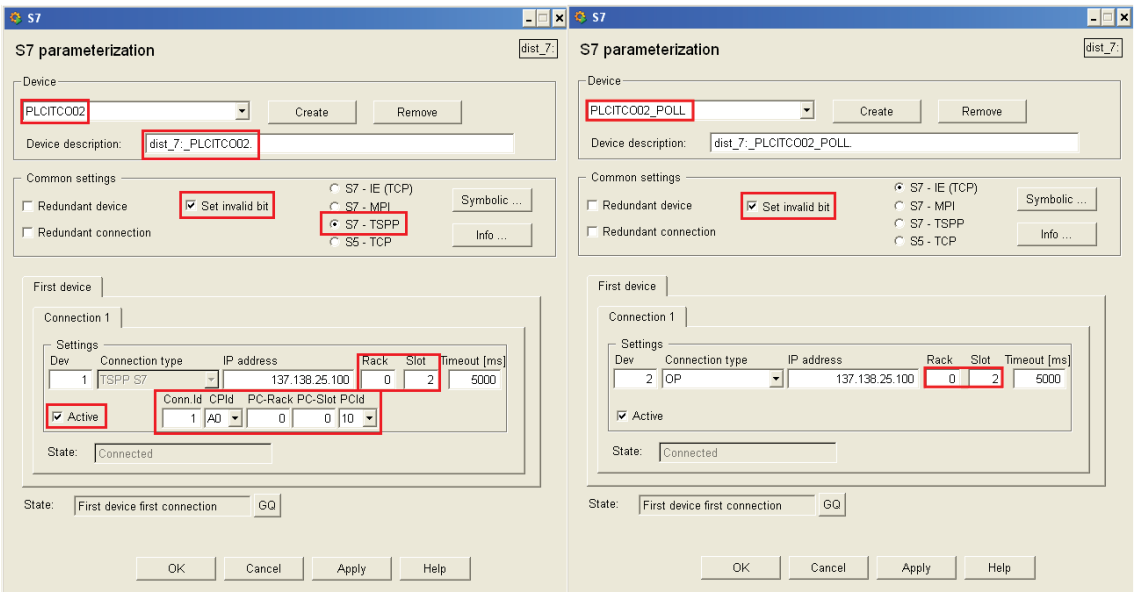


Figura 68: Configuración S7 Driver

Capítulo 4:

DESCRIPCIÓN DE LA PLANTA

4 Descripción de la planta

Este proyecto tiene como objetivo fundamental la compresión del *framework* UNICOS-CPC6, además de la demostración de que todos los bloques generados mediante este método son customizables.

Esto ha sido tenido muy en cuenta a la hora del diseño de la planta. El *framework* UNICOS-CPC6, era un elemento nuevo y se debía diseñar una planta sencilla que facilitase su comprensión.

Esta planta tenía que ser controlada por medio de un *PLC SIEMENS S300* Y supervisada mediante *PVSS*, elementos complejos, por lo que debía estar constituida por pocos elementos, fáciles de implementar y de los que se tuviese muy claro el funcionamiento.

Tras tomar en consideración todas estas pautas, se ha decidido crear una pequeña planta de presión. Formada por los siguientes elementos:

- PLC SIEMENS S315 PN/DP
- Compresor *JUN-AIR Quiet-Air 6-25*
- Variador de frecuencia Omrom *VS mini J7*
- Electroválvula *M&M D263DVH*
- Transmisor de presión *Desing Instruments TPR-14/N*
- Válvula manual

Distribuidos como puede verse en la figura 1, de tal forma que se suministre una presión constante a un consumidor situado al final de la red de tuberías. Esta presión es introducida por el usuario y debe de ser constante, con la menor oscilación posible. Para evitar las variaciones en la presión de salida el compresor debe cambiar el flujo de aire que suministra a la red dependiendo de la presión que tenga el sistema en ese momento. Para conseguir cambiar el flujo de aire el compresor debe variar la potencia con la que trabaja, esto se consigue mediante el variador de frecuencia. Este elemento suministra una señal eléctrica al compresor, para alimentar su motor eléctrico, de 220v pero de frecuencia variable, de esta forma se consigue variar el flujo de aire que se suministra a la red de tuberías.

De esta forma se puede variar la presión en el sistema, pero realizando esta tarea de forma manual es imposible conseguir una salida sin oscilaciones. Para solucionar esto se ha decidido incluir un controlador PI que regule la presión del sistema. Este controlador a través de una señal que le informa de la presión existente en la red y que proveniente del transmisor de presión varía la salida del variador de frecuencia, modificando así la alimentación del compresor, la velocidad de giro del motor de este y por tanto el flujo de aire y la presión en el sistema.

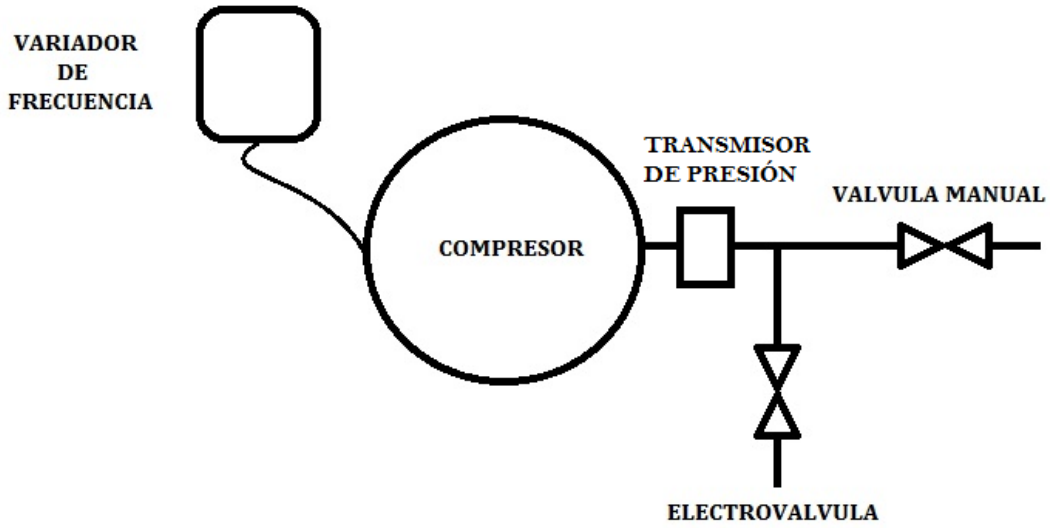


Figura 1: Diagrama de la planta de presión.

La implementación del bloque PI es llevada a cabo mediante UNICOS-CPC6 a través del PLC, de esta forma se puede comprender el funcionamiento de este *frammework* con una planta de funcionamiento sencillo.

Una vez creado el sistema a través de UNICOS CPC6 se ha decidido cambiar el funcionamiento, incluyendo varios modos de trabajo para continuar avanzado en la comprensión del *framework.*

El funcionamiento normal de la planta sigue siendo el descrito anteriormente, con la inclusión además de una lógica que autosintonice el bloque PI. Pero se han añadido modos adicionales, estableciendo una secuencia de funcionamiento como si se tratase de una planta industrial. En la figura 2 puede verse un pequeño diagrama de todos los modos y como funciona la planta final. Este diagrama será convertido a Grafect también a través del *framework* UNICOS CPC6.

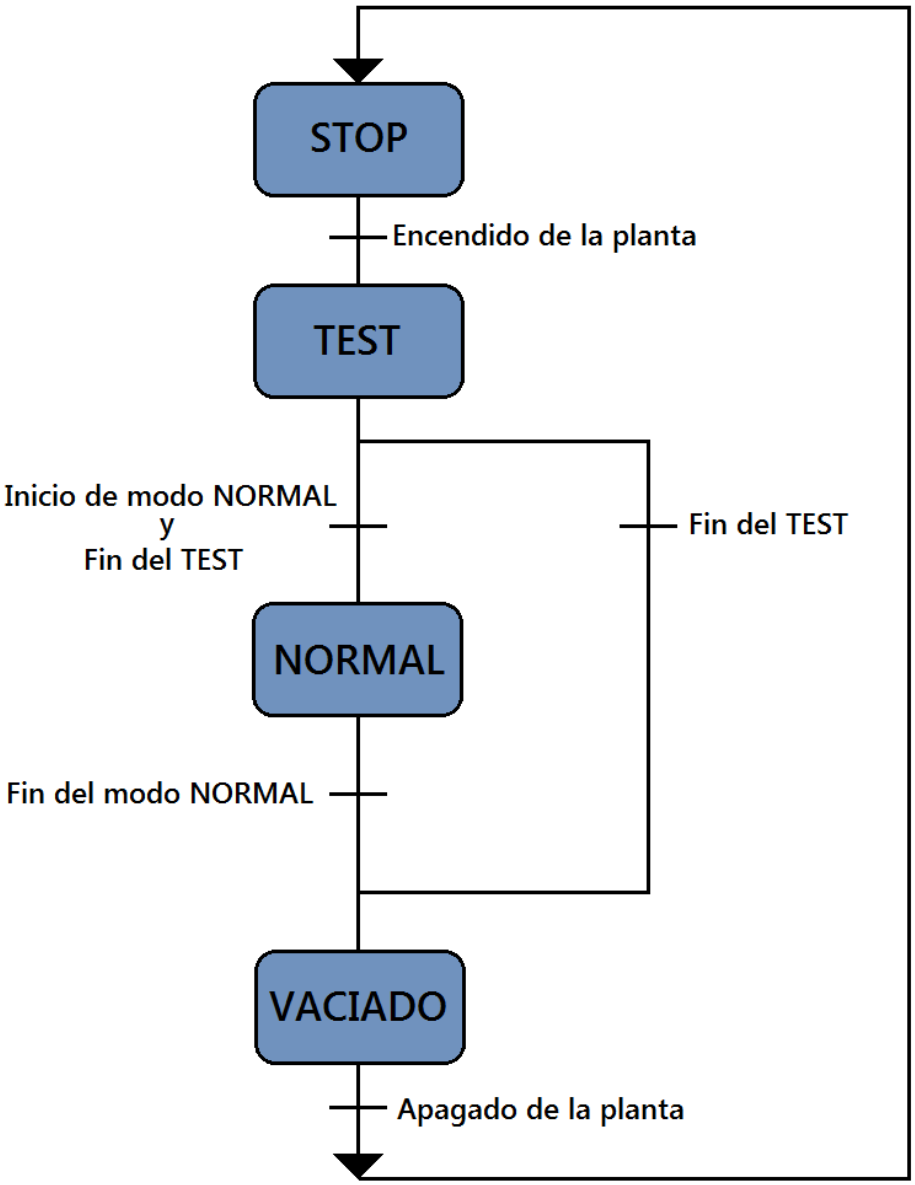


Figura 2: Diagrama de funcionamiento de la planta

Se puede ver en la figura 2 que se han creado 3 modos adicionales:

- Modo ***STOP***: En este modo la planta permanece en reposo hasta que se le da la orden de encendido.
- Modo ***TEST***. Este modo se activa únicamente cuando se encienda la planta, y realizar una rampa de frecuencias para el chequeo del funcionamiento.
- Modo ***NORMAL***: La planta funciona según el proceso descrito antes.
- Modo ***VACIADO***: La planta se vaciará por completo abriendo la electroválvula antes de su puesta en reposo para evitar posibles accidentes.

A estos modos hay que incluir un modo que está implícito durante el funcionamiento de todos ellos, el modo LOCAL. Este modo inhabilita todas las funciones del SCADA mientras un operario esté manipulando la planta desde el terreno.

Todos estos modos serán explicados con más detalle en los puntos sucesivos del proyecto.

En la figura 3 se puede ver el montaje final de la planta.

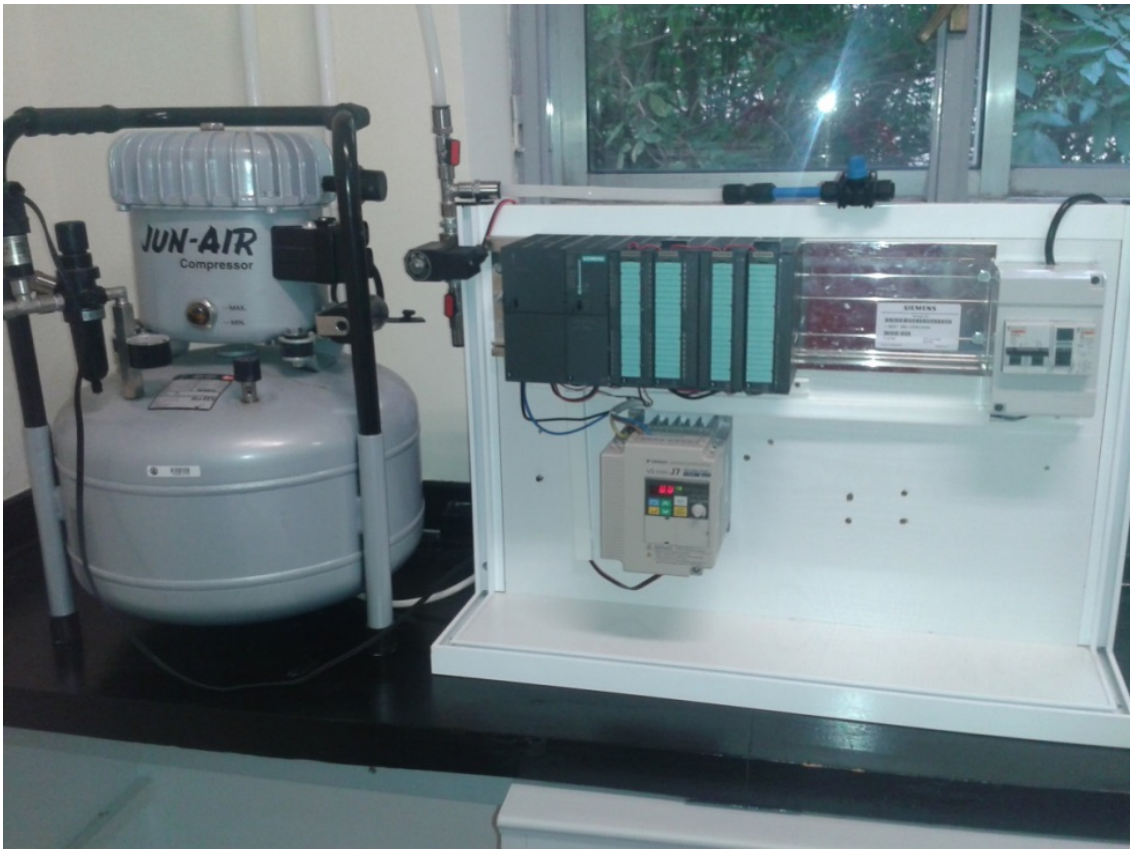


Figura 3: Planta de Presión

Capítulo 5:

DISEÑO

Para iniciar el diseño del control de una planta mediante el *framework* UNICOS-CPC6 se debe realizar un análisis exhaustivo de la planta que se desea controlar.

En el caso de este proyecto se ha decidido construir una planta de presión conectada a un PLC SIEMENS S315 PN/DP como se describe en el apartado anterior de este documento. Se opta por una configuración sencilla que facilite el cumplimiento del objetivo principal de este proyecto, la comprensión del *framework* UNICOS.

El diseño escogido puede verse en la figura 1. Consigue aportar una presión determinada a una red de tuberías por medio del compresor de aire. Este compresor funciona gracias a un motor eléctrico. Para conseguir variar la presión en el sistema se ha decido conectar este motor a un variador de frecuencia que controle su velocidad de giro, controlando así la presión que se suministra al sistema.

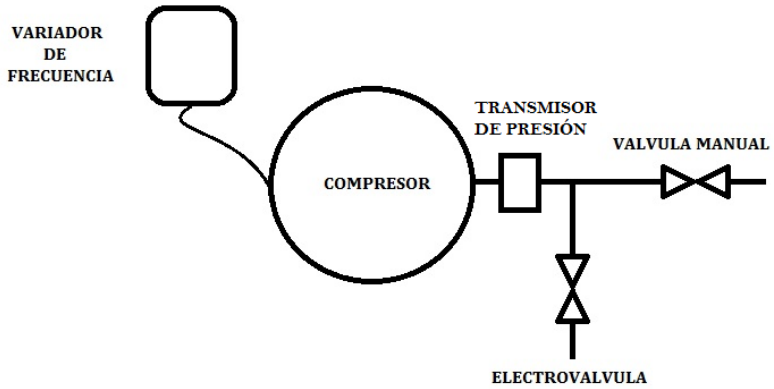


Figura 1: Diagrama de la planta de presión.

Para la compresión del funcionamiento de esta planta se debe entender que cuanto mayor sea la frecuencia de la señal proporcionada por el variador mayor será la velocidad de giro del motor y por lo tanto mayor será la presión en el sistema.

Para controlar la presión en el sistema de tuberías se ha decidido conectar un transmisor de presión en su inicio para que este informe en todo momento de la presión en el sistema. Las salidas se controlaran a través de dos válvulas, una manual y una electroválvula.

Como se ha podido ver en el punto de este documento donde se describe la planta, la presión en la salida debe de ser constante, evitando en la medida de lo posible que tenga oscilaciones. Para conseguir esto se ha optado por incluir un controlador PID que regule en todo momento la presión.

Para diseñar una solución mediante UNICOS-CPC6 en primer lugar se debe crear un diagrama que describa el proceso cumpliendo con la jerarquía establecida por UNICOS.

Este tipo de gráfico que describe el proceso completo pude verse en la figura 2. Para su creación se ha identificado cada uno de los elementos de la planta con un tipo de bloque UNICOS. Además se deben incluir los bloques que sean necesarios para identificar los procesos internos que definan el funcionamiento de la planta. En este caso el bloque que identifica el proceso es un PID que debe estar correctamente conectado con los bloques asociados a los elementos de la planta.

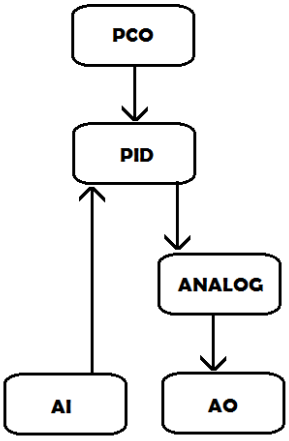


Figura 2: Grafico Planta según jerarquía UNICOS

Se puede observar en la Figura 2 que hay un objeto denominado PCO que no se corresponde con ningún elemento del proceso, este elemento será la cabeza del proceso, controla en todo momento el funcionamiento de la planta y es imprescindible que esté incluido para cumplir con la filosofía de UNICOS

Una vez se haya descrito el funcionamiento de la planta mediante este tipo gráfico, se debe proceder a la generación de los solución mediante el generador UAB, como se ha podido ver en el punto número tres de este documento.

UAB genera automáticamente una solución tanto para la capa de control como para la de supervisión a través de una serie de archivos que pueden verse desglosados a continuación.

Como herramienta auxiliar, este proyecto se hace referencia a un proyecto fin de carrera de Silvia María Izquierdo Rosas [15] que puede ayudar en la comprensión de los puntos sucesivos.

5.1 Solución generada automáticamente EN UNICOS

5.1.1 PLC

La solución generada automáticamente por UAB, proporciona una serie de archivos para el programa de SIEMENS SIMATIC STEP7 que son detallados a continuación y que constituyen la solución para la capa de control. Estos archivos que

se encuentran la carpeta *Output* del proyecto UAB, establecen la configuración del PLC siguiendo la filosofía UNICOS e incluyen la lógica que determina el funcionamiento completo del proceso.

Como se ha visto anteriormente el programa STEP7 divide los archivos que constituyen un proyecto en fuentes y bloques. UAB crea a partir de la hoja de especificaciones varios archivos de este tipo que deben ser importados a un nuevo proyecto siguiendo los pasos especificados en los puntos sucesivos de esta memoria.

Estos archivos se pueden dividir en dos tipos, los archivos comunes a todos los procesos, que establecen la configuración del proyecto cumpliendo con las especificaciones UNICOS, y los archivos que contienen la configuración y lógica del proceso a controlar.

- Archivos comunes a todos los proyectos

Los archivos comunes para todos los procesos, tienen la función de modificar el proyecto de SIMATIC, de tal forma que se cree un proyecto básico y estándar para el desarrollo de aplicaciones UNICOS CPC. En la figura 3 se puede ver una lista que contiene todos los archivos de este tipo generados por UAB.

FUENTES
CPC_BASE_UNICOS
CPC_FB_AA
CPC_FB_AI
CPC_FB_AIR
CPC_FB_ANADIG
CPC_FB_ANADO
CPC_FB_ANALOG
CPC_FB_AO
CPC_FB_AOR
CPC_FB_APAR
CPC_FB_AS
CPC_FB_DA
CPC_FB_DI
CPC_FB_DO
CPC_FB_DPAR
CPC_FB_LOCAL
CPC_FB_ONOFF
CPC_FB_PCO
CPC_FB_PID
CPC_FB_WPAR
CPC_FB_WS
1_Compilation_Baseline
2_Compilation_instance
3_Compilation_logic
4_Compilation_OB

BLOQUES
OB1
OB80
OB85
OB86
OB121
OB122
FB9
FB12
FB13
FB14
FB19
FB20
FB22
FC1
FC72
VAT_TSPP
SFB3
SFB4
SFB5

Figura 3: Bloques y fuentes generadas para la configuración UNICOS

Estos archivos en ningún caso deberán ser modificados, ya que contienen por ejemplo las variables que identifican a cada objeto o la interconexión entre estos, y si se modificasen el proyecto ya no cumpliría con el estándar que establece UNICOS para las aplicaciones CPC.

Con estos archivos comunes a todos los proyectos únicamente se establecen las bases del proyecto UNICOS-CPC6, para que la creación sea completa y poder disponer de todos los bloques necesarios para el correcto funcionamiento de la aplicación se deben compilar cuatro de las fuentes citadas anteriormente:

- 1_Compilation_Baseline
- 2_Compilation_Instance
- 3_Compilation_LOGIC
- 4_Compilation_OB

Se debe tener en cuenta que cada una de las fuentes del proyecto contiene una parte de la configuración o funcionamiento del proceso. La compilación de estas cuatro fuentes une toda esta lógica creando una serie de bloques que contienen toda la información requerida para el correcto funcionamiento del PLC.

En la figura 4 se pueden ver todos los bloques de un proceso completo, que han sido creados tras esta compilación

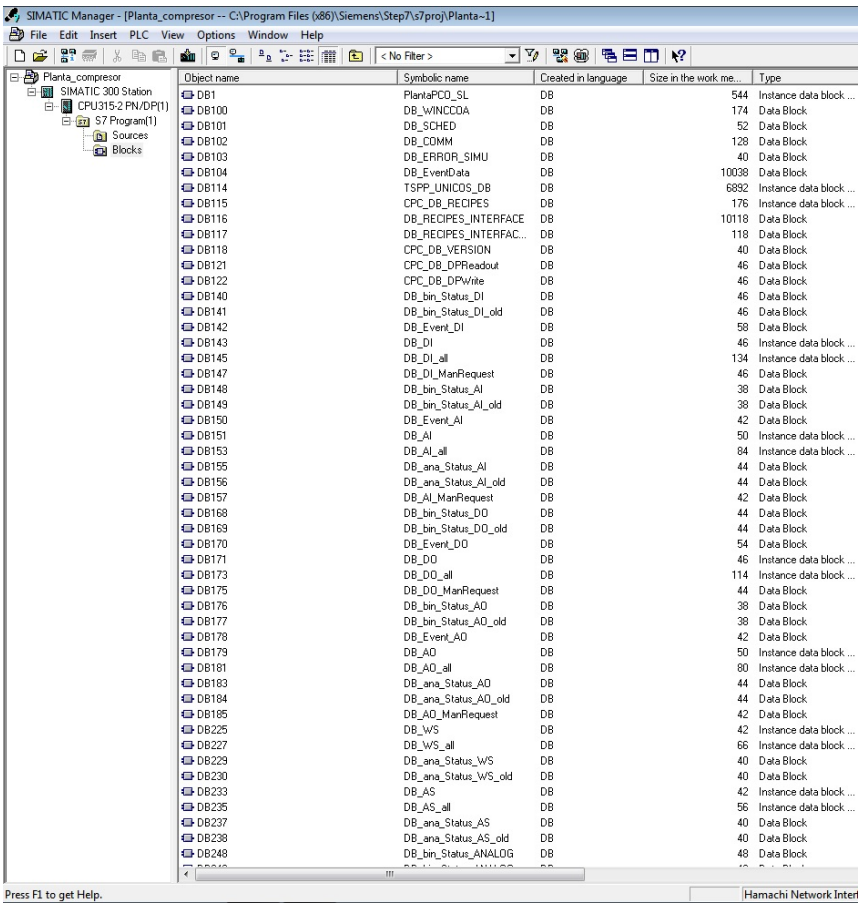
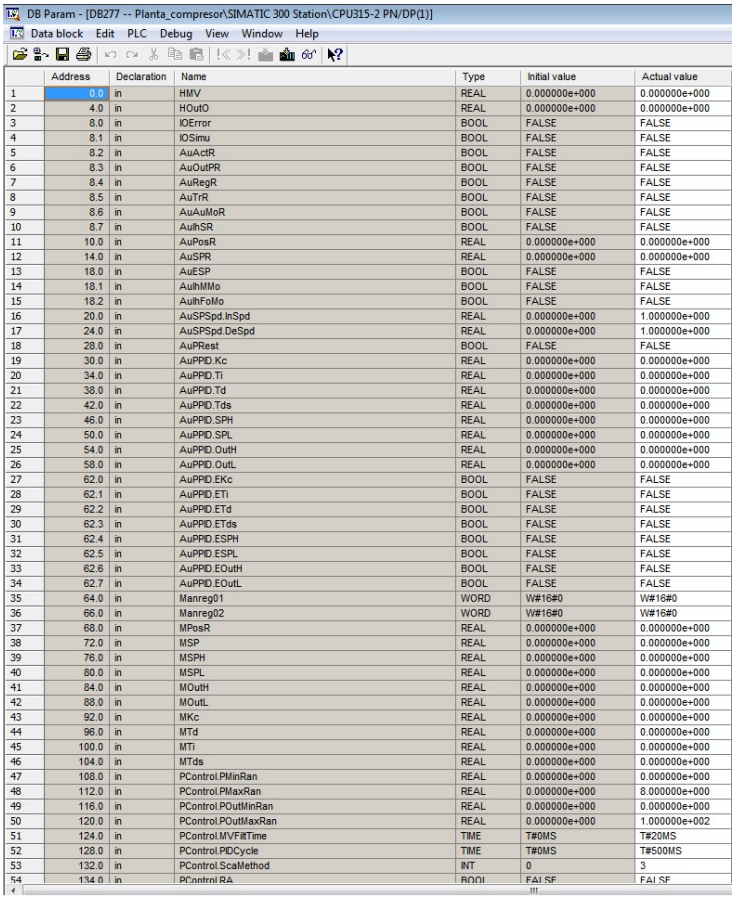


Figura 4: Conjunto de bloques del proyecto SIMATIC

En la figura 5 se puede ver el contenido de uno de los bloques generados tras esta compilación, pertenece a un objeto PID, y como se puede observar define todas las variables necesarias para el control del bloque.



DB Param - [DB277 -- Planta_compresor\SIMATIC 300 Station\CPU315-2 PN/DP(1)]

Data block Edit PLC Debug View Window Help

	Address	Declaration	Name	Type	Initial value	Actual value
1	0.0	in	HMV	REAL	0.000000e+000	0.000000e+000
2	4.0	in	HOutO	REAL	0.000000e+000	0.000000e+000
3	8.0	in	IOError	BOOL	FALSE	FALSE
4	8.1	in	IOSimu	BOOL	FALSE	FALSE
5	8.2	in	AuActR	BOOL	FALSE	FALSE
6	8.3	in	AuOutPR	BOOL	FALSE	FALSE
7	8.4	in	AuRegR	BOOL	FALSE	FALSE
8	8.5	in	AuTrR	BOOL	FALSE	FALSE
9	8.6	in	AuAuMoR	BOOL	FALSE	FALSE
10	8.7	in	AuIhSR	BOOL	FALSE	FALSE
11	10.0	in	AuPosR	REAL	0.000000e+000	0.000000e+000
12	14.0	in	AuSPR	REAL	0.000000e+000	0.000000e+000
13	18.0	in	AuESP	BOOL	FALSE	FALSE
14	18.1	in	AuIhMMo	BOOL	FALSE	FALSE
15	18.2	in	AuIhFoMo	BOOL	FALSE	FALSE
16	20.0	in	AuSPSpd.InSpd	REAL	0.000000e+000	1.000000e+000
17	24.0	in	AuSPSpd.DeSpd	REAL	0.000000e+000	1.000000e+000
18	28.0	in	AuPRest	BOOL	FALSE	FALSE
19	30.0	in	AuPPID.Kc	REAL	0.000000e+000	0.000000e+000
20	34.0	in	AuPPID.Ti	REAL	0.000000e+000	0.000000e+000
21	38.0	in	AuPPID.Td	REAL	0.000000e+000	0.000000e+000
22	42.0	in	AuPPID.Tds	REAL	0.000000e+000	0.000000e+000
23	46.0	in	AuPPID.SPH	REAL	0.000000e+000	0.000000e+000
24	50.0	in	AuPPID.SPL	REAL	0.000000e+000	0.000000e+000
25	54.0	in	AuPPID.OutH	REAL	0.000000e+000	0.000000e+000
26	58.0	in	AuPPID.OutL	REAL	0.000000e+000	0.000000e+000
27	62.0	in	AuPPID.EKc	BOOL	FALSE	FALSE
28	62.1	in	AuPPID.ETi	BOOL	FALSE	FALSE
29	62.2	in	AuPPID.ETd	BOOL	FALSE	FALSE
30	62.3	in	AuPPID.ETds	BOOL	FALSE	FALSE
31	62.4	in	AuPPID.ESPH	BOOL	FALSE	FALSE
32	62.5	in	AuPPID.ESPL	BOOL	FALSE	FALSE
33	62.6	in	AuPPID.EOutH	BOOL	FALSE	FALSE
34	62.7	in	AuPPID.EOutL	BOOL	FALSE	FALSE
35	64.0	in	Manreg01	WORD	W#16#0	W#16#0
36	66.0	in	Manreg02	WORD	W#16#0	W#16#0
37	68.0	in	MPosR	REAL	0.000000e+000	0.000000e+000
38	72.0	in	MSP	REAL	0.000000e+000	0.000000e+000
39	76.0	in	MSPH	REAL	0.000000e+000	0.000000e+000
40	80.0	in	MSPL	REAL	0.000000e+000	0.000000e+000
41	84.0	in	MOutH	REAL	0.000000e+000	0.000000e+000
42	88.0	in	MOutL	REAL	0.000000e+000	0.000000e+000
43	92.0	in	MKc	REAL	0.000000e+000	0.000000e+000
44	96.0	in	MTd	REAL	0.000000e+000	0.000000e+000
45	100.0	in	MTi	REAL	0.000000e+000	0.000000e+000
46	104.0	in	MTds	REAL	0.000000e+000	0.000000e+000
47	108.0	in	PControl.PMinRan	REAL	0.000000e+000	0.000000e+000
48	112.0	in	PControl.PMaxRan	REAL	0.000000e+000	8.000000e+000
49	116.0	in	PControl.POutMinRan	REAL	0.000000e+000	0.000000e+000
50	120.0	in	PControl.POutMaxRan	REAL	0.000000e+000	1.000000e+002
51	124.0	in	PControl.MVFiltTime	TIME	T#0MS	T#20MS
52	128.0	in	PControl.PIDCycle	TIME	T#0MS	T#500MS
53	132.0	in	PControl.ScaMethod	INT	0	3
54	134.0	in	PControl.RA	BOOL	FALSE	FALSE

Figura 5: Bloque de datos asociado al PID

Como se ha visto en esta memoria la compilación de estas fuentes debe ser llevada a cabo tras incluir los archivos que determinan el funcionamiento del proceso y en un orden preciso.

- Archivos específicos para cada aplicación.

UAB además de generar los archivos genéricos a todos los proyectos, citados anteriormente, va a generar otros específicos para cada aplicación. Estos archivos contienen todas las especificaciones incluidas en la hoja Excel.

En este caso UAB únicamente genera archivos de tipo fuente que contienen toda la información de cada uno de los bloques que caracterizan el proceso y como se verá en el punto siguiente de este documento son estas fuentes las que serán modificadas para asociar lógica al proceso.

En la figura 6 se pueden ver las fuentes de un proyecto completo, tanto las genéricas como las específicas. Las fuentes que se ven resaltadas son las destinadas a la inclusión de lógica.

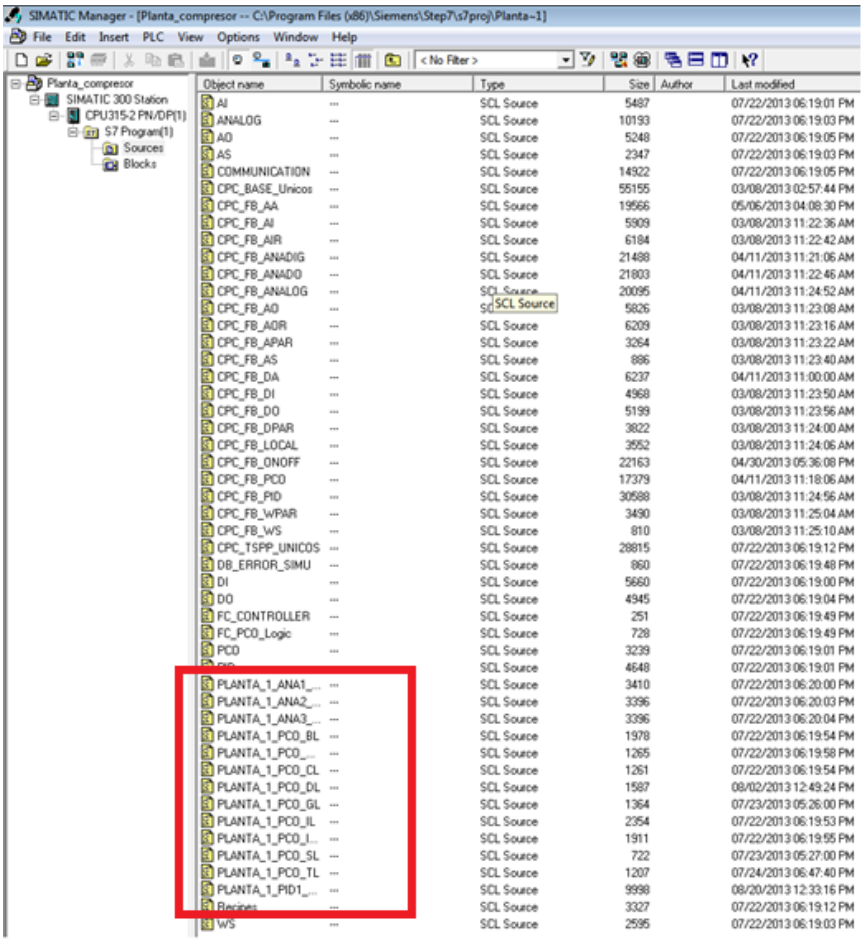


Figura 6: Conjunto de fuentes del proyecto SIMATIC

Son estas fuentes las verdaderamente importantes en la descripción del proceso ya que contienen la lógica asociada a cada uno de los objetos del proceso. UAB da un nombre genérico a estas fuentes para que sean fácilmente identificables. En la figura 7 se puede ver un gráfico que explica los criterios seguidos por el generador para nombrar estas fuentes.

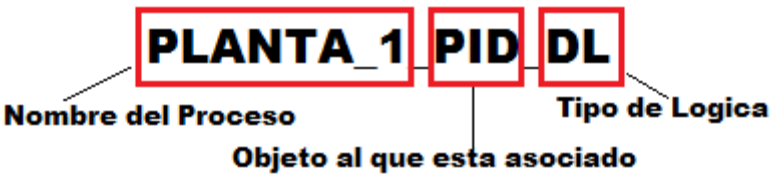


Figura 7: Criterio UNICOS para los nombres de las fuentes

Como se ve en la figura 6 UAB ha creado nueve archivos asociados al objeto PCO, esto es debido a la complejidad del elemento. Cada una de estas fuentes va a contener una parte de la lógica total del proceso. En la figura 8 se muestra una tabla donde se indican el tipo de lógica que debe ir incluido en cada uno se ellos:

Tipo de archivo	Lógica asociada
PLANTA_1_PCO_BL	Funcionamiento de avisos y alarmas
PLANTA_1_PCO_CDOL	Modo de funcionamiento del objeto
PLANTA_1_PCO_CL	Configuración lógica del PCO.
PLANTA_1_PCO_DL	Lógica dependiente del objeto
PLANTA_1_PCO_GL	Lógica global de funcionamiento
PLANTA_1_PCO_IL	Lógica asociada a los *interlock*
PLANTA_1_PCO_INST	Instancias asociadas al PCO
PLANTA_1_PCO_SL	Llamada a Grafcet de funcionamiento
PLANTA_1_PCO_TL	Transiciones del Grafcet

Figura 8: Fuentes asociadas al objeto PCO

Como se puede ver también en esta figura 8, la lógica que contiene cada una de las fuentes viene determinada por las siglas finales del nombre, por ejemplo:

PLANTA_1_PCO_GL → ***G****loblal* ***L****ogic*

En el caso del resto de objetos únicamente creara una fuente, como puede verse en la figura 6 para el caso del PID.

5.1.2 PVSS

El generador UAB al igual que para la capa de control, también genera archivos para la de supervisión. Esta generación es mucho más simple que el caso anterior, ya que UAB únicamente genera un archivo que contiene toda la información relativa al proceso.

Este archivo debe ser importado a un proyecto PVSS como se ha visto en los puntos anteriores de esta memoria, indicándole únicamente la ruta hasta su ubicación. Este archivo al igual que en el caso de los de control se encuentra en la carpeta del proyecto de UAB. A pesar de que el programa utilizado para la supervisión es el PVSS, UAB genera este archivo en la carpeta *WinCCOAInstaceGenertor*. En la figura 9 se puede ver una imagen con la ruta hasta el archivo generado.

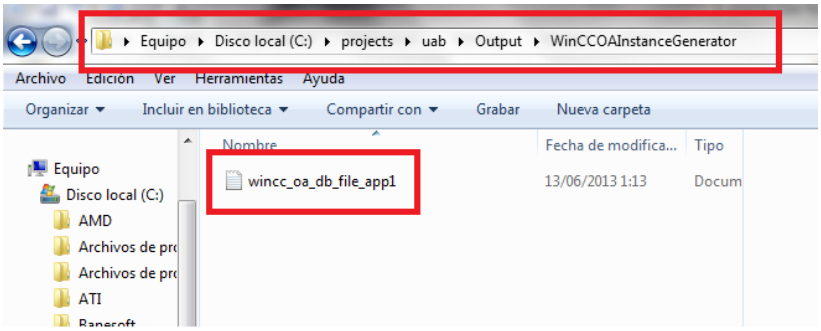


Figura 9: Archivo generado para PVSS

5.2 Adaptación de la solución en el marco de UNICOS para el nivel de control

El generador UAB como se ha comentado anteriormente crea una serie de archivos que contienen tanto la solución para el control, como para la supervisión de un determinado proceso.

Tras la importación y compilación de los archivos generados al programa SIMATIC, se crea una aplicación que es capaz de controlar todo el proceso. Esta aplicación es básica, tiene una estructura sencilla, determinada por la hoja de características Excel, y cuenta únicamente con estructuras predefinidas en UNICOS. Por lo tanto no se ha creado ningún tipo lógica asociada a los bloques, solo un espacio de trabajo.

5.2.1 Customización de la solución generada

La solución generada por UAB se puede modificar, ya que UNICOS es un sistema abierto, pudiendo incluir lógica que defina el funcionamiento de los objetos. La opción que presenta UNICOS para esta modificación, consiste en asociar una lógica creada por el usuario a un objeto predefinido en UNICOS de tal forma que se amplíe o modifique su funcionamiento. En la figura 10 se puede observar un gráfico que detalla esta incursión de lógica.

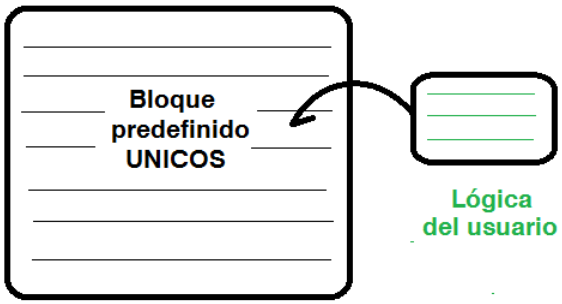


Figura 10: Incursión de lógica en UNICOS

UNICOS cuenta con tres métodos diferentes para la incursión de lógica al proceso:

- Modificar la programación del generador UAB.
- Crear un *Template* escrito en lenguaje *Phyton*
- Añadir la lógica directamente al PLC (método empleado en este proyecto)

Modificación del generador UAB

El primer método citado, consiste en la modificación de los *plugins* del generador. Para llevar a cabo este método se debe acceder a la programación interna del generador e incluir allí la lógica deseada, de esta forma todos los procesos generados la tendrán asociada de base. Este método es el más complejo de los tres, ya que requiere de un gran dominio del lenguaje de programación JAVA como de un conocimiento amplio de la arquitectura de UNICOS.

Creación de *template*

La opción de la inclusión de lógica mediante un *template* consiste en la creación de un archivo de texto que incluya la lógica deseada escrita en lenguaje *Phyton*. Este archivo tiene una estructura predefinida para cada objeto y únicamente se debe añadir la parte de la lógica asociada a él. Para utilizar este método se deben incluir los *template* creados en la carpeta del proyecto UAB, siguiendo la ruta *Resouces/S7LogicGenerator/Rules/UserSpecific*, y establecer una llamada desde la hoja de especificaciones, como se ve en la figura 11. Esta forma de incluir lógica es ideal para procesos con objetos que actúan de la misma forma, ya que se puede asignar el mismo *template* a varios elementos. Lo que supone un ahorro considerable de tiempo en la fase de diseño.

Es un método mucho menos complejo que el anterior, aunque requiere conocimientos avanzados del lenguaje de programación *Phyton.*

Figura 11: Llama a un *template* desde la hoja de especificaciones

Incursión de lógica directamente sobre el PLC

Por último la tercera opción permite introducir lógica directamente al PLC. Este método consiste en crear la lógica dentro de la fuente correspondiente al bloque que se desea modificar. Es la opción más sencilla aunque la menos potente, ya que la lógica incluida únicamente sirve para un objeto en una aplicación determinada. Es la opción escogida en el desarrollo de esta aplicación para demostrar así la custumizacion de los objetos UNICOS.

Para el uso de este método el proyecto SIMATIC debe estar completamente creado. Para comenzar se debe abrir la fuente destinada al objeto que desee modificar e

incluir ahí la lógica. En la figura 12 puede verse el interior de una fuente asignada a un bloque PID. Esta fuente tiene asignado un espacio para incluir la lógica que comienza en la línea de comentarios *User Code Begin*. El código que defina el funcionamiento debe estar escrito en lenguaje STL.

El lenguaje STL es un lenguaje basado en la norma IEC 61499[3]. Por lo tanto es un código orientado a objetos, que crea unas instancias de *Function Blocks* para definir cada objeto UNICOS. Este código tiene una estructura similar a C por lo que la programación resulta sencilla e intuitiva.

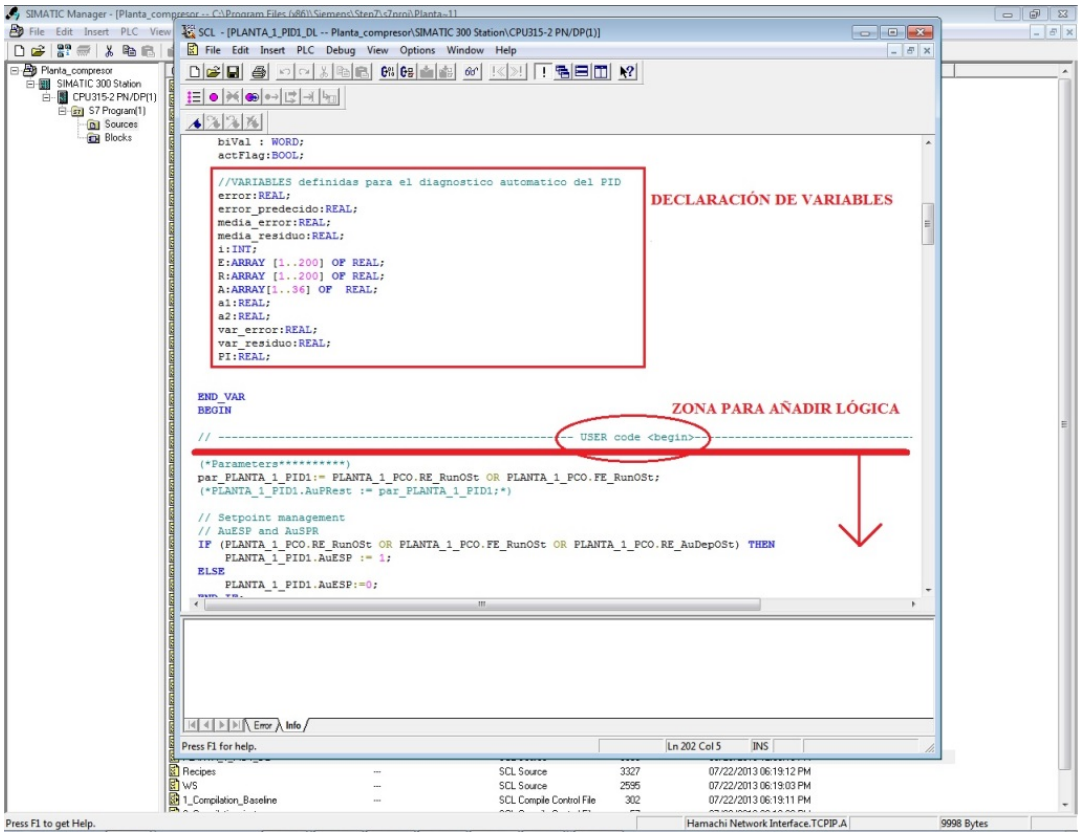


Figura 12: Zona de incursión de lógica

Una vez incluida la lógica en la fuente correspondiente se debe compilar como se ve en la figura 13 y cargar de nuevo todos los bloques al PLC. Solo entonces la lógica asociada tendrá efecto sobre el proceso.

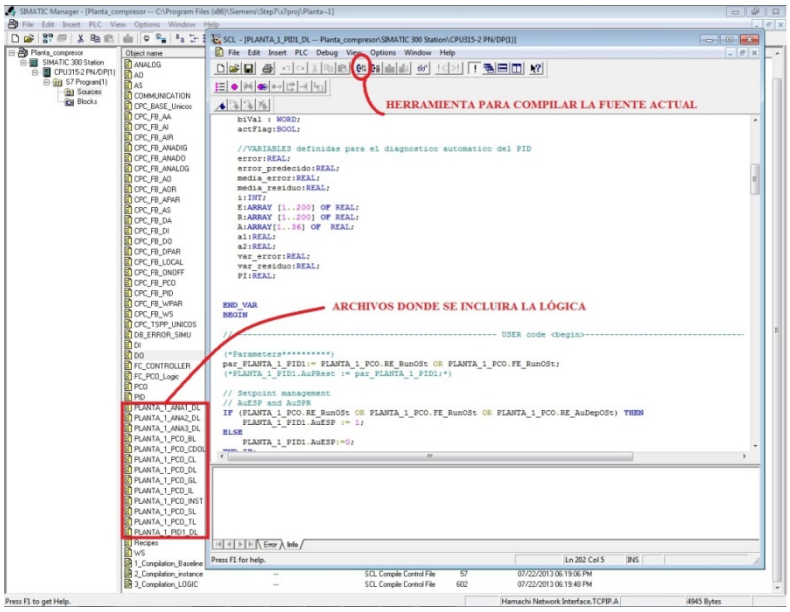


Figura 13: Adición de lógica a una fuente SIMATIC

Para añadir lógica mediante estos métodos se debe tener en cuenta que cada uno de los objetos tiene asociadas unas variables con las que se controla su funcionamiento, estas variables tienen asociada un nombre determinado según la acción que realicen y se pueden encontrar en el manual *UCPC6-Manual* [11]. En la figura 14 se puede ver un ejemplo de las variables que contiene un bloque PID.

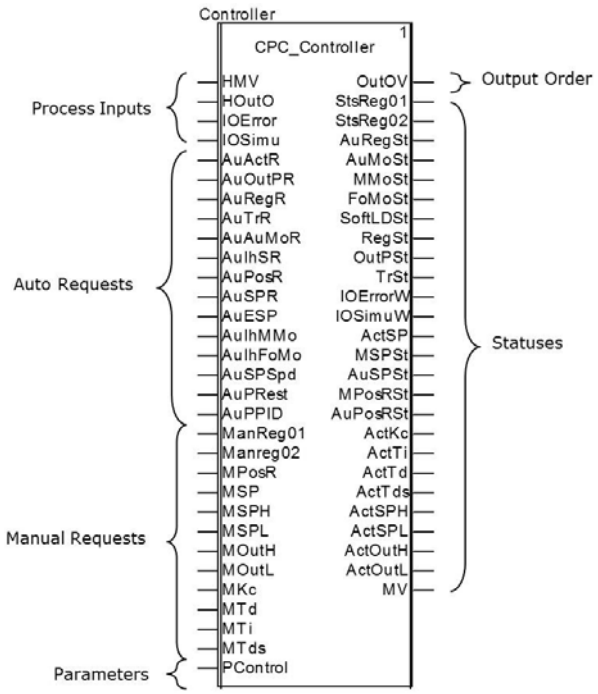


Figura 14: Bloque Variables PID

Para la correcta modificación de estas variables, en el código deben ser denominadas correctamente, si no es así el programa producirá errores. En primer lugar se debe incluir el nombre del objeto a modificar seguido de nombre de la variable, como puede verse en la figura 15.

Figura 15: Llamada de variables en UNICOS

La lógica en el diseño de esta aplicación se ha introducido directamente al PLC, se ha escogido este método por la naturaleza del propio proyecto. Como se ha comentado anteriormente el objetivo principal del proyecto es la comprensión del funcionamiento del *framework* UNICOS CPC6 y la demostración de que todos los objetos generados en UNICOS pueden customizarse.

En busca de una comprensión del funcionamiento de UNICOS se decide escoger esta opción debido a su simplicidad. Si las opciones escogidas hubieran sido cualquiera de las otras la comprensión del entorno UNICOS habría sido una tarea imposible. Estos métodos no utilizados se contemplaran en versiones futuras, una vez quede demostrada la funcionalidad de esta solución.

5.2.1.1 Customización del bloque PID

Esta aplicación pretende añadir un programa que sintonice de forma automática el objeto PID básico creado por UNICOS, demostrando así que es posible la customización. El objeto escogido para la modificación ha sido el PID debido a que este tipo de bloque brinda una gran cantidad de opciones de modificación como pueden ser diagnósticos o incluso auto-sintonías.

5.2.1.1.1 Creación de programa para la autosintonía del PID

En primer lugar se debe identificar que es y que función tiene un bloque PID. Un regulador PID se define como un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso. En definitiva se puede decir que el objetivo de un regulador PID es estabilizar una señal de salida lo máximo posible. En la figura 16 puede verse un esquema de un regulador PID genérico.

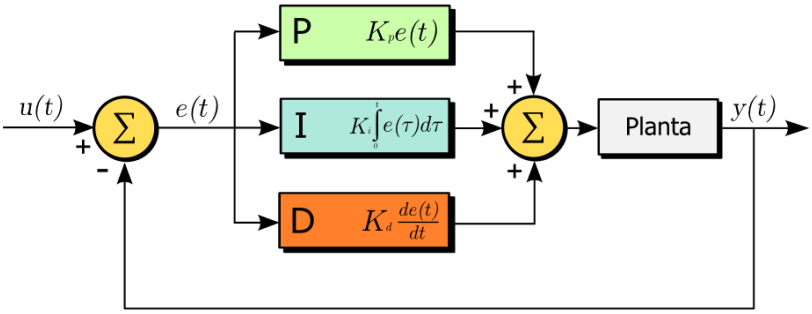


Figura 16: Regulador PID

Como se aprecia en la figura 16 el control de la señal se lleva a cabo mediante tres acciones, **P**roporcional, **I**ntegral y **D**erivativa. Estas tres acciones a partir del análisis del error en el sistema, modifican su salida hasta que el error disminuya a cero.

- La acción proporcional consiste en el producto entre la señal de error y una constante proporcional para lograr que el error en estado estacionario se aproxime a cero. El valor Proporcional determina la reacción del error actual y responde a la inicial K.

- El término integral, Ti, responde a la integral del error, esto asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero y la acción del término integral puede ser ajustada mediante el parámetro de sintonía.

- El término derivativo, Td, determina la reacción del tiempo en el que el error se produce.

Mediante el correcto ajuste de estos tres parámetros la respuesta del sistema controlado ante cambios en la referencia o frente a perturbaciones en el proceso, presentará unas características dinámicas aceptables, garantizando que el error del lazo de control sea aceptable en todo momento. Esta tarea se denomina sintonización del PID. La correcta sintonización de estos reguladores es compleja debido a que cada proceso tiene unos valores determinados, dependiendo de la situación de sus componentes o de la tarea que realicen.

En este proyecto se pretende diseñar un código que autosintonice el sistema, independientemente de la situación por ejemplo de las válvulas de la red de tuberías o del punto de funcionamiento en el que se desee trabajar.

Se dispone de varios métodos para sintonizar un PID, en este caso se ha escogido un método de sintonía denominado el método del relé. Este método calcula los valores de las variables del PID en lazo cerrado, lo que supone una gran ventaja ya que no es necesario modificar el montaje de la planta cada vez que se necesite sintonizar el sistema.

Método del Relé para la autosintonía de bloques PID.

Por autosintonía se entiende que se realiza a demanda de un usuario externo y no continuamente como sucedería en métodos de control adaptativo. Nada impide que el agente externo que lo demanda, sea un algoritmo de diagnóstico.

La importancia de la autosintonía de PIDs no puede ser sobreestimada. El PID es el controlador más utilizado en la industria de procesos. Una empresa típica puede tener cientos de lazos de control con este controlador. Existen reportes que indican que una mayoría de estos están mal sintonizados, y esto implica un coste económico importante. La sintonía manual de estos reguladores suele ser muy costosa en términos de recursos humanos.

Se debe tener en cuenta además que no basta con una sintonización inicial de este tipo de controladores, los elementos del proceso sufren desgastes a lo largo del tiempo, que pueden variar el funcionamiento completo de la planta, sin olvidar que muchas veces se llevaran a cabo rediseños que hagan que los parámetros establecidos ya no sean válidos. Por eso es importante disponer de un método que de forma rápida sintonice el sistema siempre que se crea necesario.

El método del relé tiene la ventaja de que puede aplicar en lazo cerrado, introduciendo perturbaciones mínimas en el proceso controlado. Este tipo de método, requiere de la existencia de un controlador que esté presintonizado y no necesitan de información extra para ser aplicado. El tamaño de la perturbación que se introducen a propósito en el lazo de control, se puede fácilmente parametrizar, en base al nivel de ruido existente, por una parte, y a los cambios admisibles de acuerdo a las características del proceso controlado, por la otra.

Una sintonía adecuada de un lazo de control, requeriría, en principio un conocimiento de las características dinámicas del proceso a controlar. En particular, es importante tener información fidedigna del comportamiento de la respuesta de frecuencia del sistema alrededor del punto crítico. Se conoce que cuando se aumenta la ganancia del lazo de control, eventualmente el sistema en lazo cerrado comenzará a oscilar. Estas oscilaciones, de frecuencia ω_{180}, se corresponderán con un ganancia del controlador igual $K_u=1/k_{180}$. Los parámetros ω_{180} y k_{180} se denominan respectivamente frecuencia y ganancia últimas o críticas y sus valores específicos dependerán de las características dinámicas del sistema. Es un hecho conocido, que a la frecuencia última, existe un desfase de 180º entre la entrada y la salida del proceso, de ahí la denominación de las misma. El conocimiento de estos valores permite diseñar una sintonía aceptable del PID y está en la base del conocido método de Ziegler y Nichols (ZN) en lazo cerrado.

La dificultad de implementar el método de ZN en un ambiente industrial, es que para obtener la información relevante, se debe llevar al sistema de control en lazo cerrado al borde la inestabilidad. Una solución a este problema fue brindada por Astrom

y colaboradores mediante el método del relé. La idea es sustituir el PID original, con un relé según se muestra en el esquema de la figura 17, en el momento en que se decida realizar la autosintonía se conecta el mismo en lazo cerrado. La inclusión del relé como controlador provocará la excitación de oscilaciones a la frecuencia última ω_{180}, puesto que provoca que la entrada del proceso, que será ahora una onda cuadrada y el armónico principal de la salida del mismo presenten un desfase de 180º tal y como se requiere. La ventaja fundamental del uso del relé estriba en que el diseñador puede elegir la amplitud a la salida de éste y de esa forma impide que el sistema se vaya de control. Esta amplitud del relé será un parámetro a escoger en cada caso, solucionando el compromiso entre el ruido de medida existente y las perturbaciones que el proceso pueda tolerar durante la sintonía. Se le suele añadir una histéresis al relé para mitigar los efectos del ruido de medida. Una explicación teórica rigurosa del método se puede encontrar en *Advanced PID Control . Karl J. Astrom y Tore Hagglund Editado por ISA: The instrumation, Systems, and Automation Society* [16].

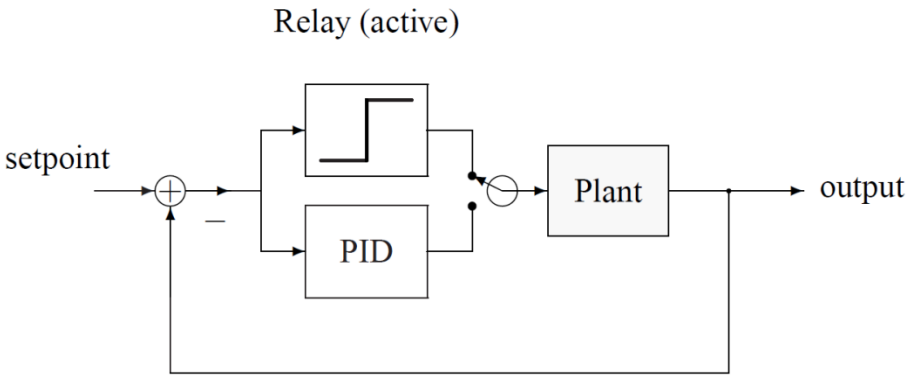


Figura 17: Esquema de funcionamiento del método del relé.

En la figura 18 puede apreciarse un esquema del funcionamiento clásico del método del relé. Se puede ver como se produce un armónico en la salida del sistema (rojo), cuando la entrada responde a una onda cuadrada generada a la frecuencia última.

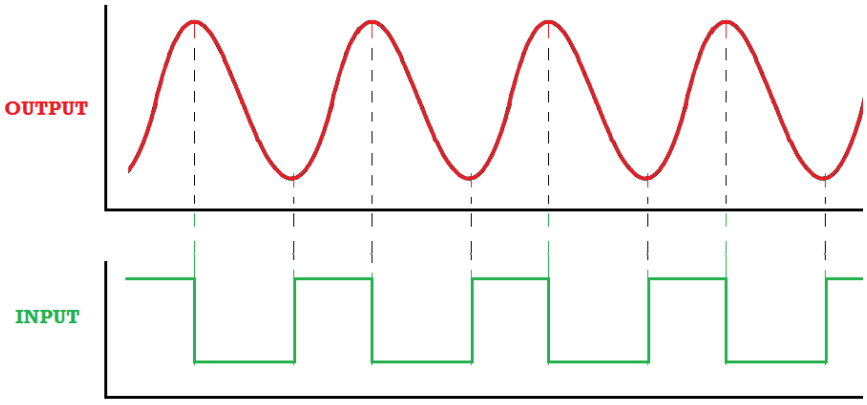


Figura 18: Ondas de entrada y salida en el método del relé

La generación de esta onda cuadrada debe cumplir con una serie de requisitos para que el cálculo de la frecuencia y ganancia última sea correcto. De esta forma la determinación posterior de los parámetros que establecen la sintonía del sistema será óptima.

En primer lugar se debe resaltar que la amplitud de esta onda responde a valores concretos que varían dependiendo del sistema que se vaya a sintonizar. Estos valores se determinan a partir del valor que tenga la entrada del sistema en el momento justo en que se desconecta el bloque PID, y se denomina Uo.

Para determinar el máximo de esta onda cuadrada se utiliza la variable mencionada Uo y se le suma una constante determinada por el usuario denominada d, de esta forma se controla en todo momento el sistema evitando que se dispare a valores que lo desestabilicen. En el caso del mínimo se lleva a cabo la misma operación, pero en este caso restado esta variable d. En la figura 19 puede observarse un gráfico que detalla este proceso.

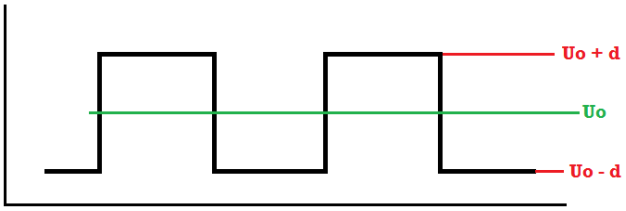


Figura 19: Amplitud del tren de pulsos del método del relé

Para el correcto establecimiento de la frecuencia última del sistema se debe tener en cuenta el criterio de signos establecido por el método del relé en cuanto al error cometido. Ya que este valor establecerá los cambios en la señal de entrada del sistema. Este método define el error como la diferencia entre el punto de trabajo – la salida actual. De esta forma si la salida se encuentra por encima del punto de trabajo el error será negativo y si se encuentra por debajo el error cometido será positivo. Puede verse en la figura 20 una representación gráfica de este concepto.

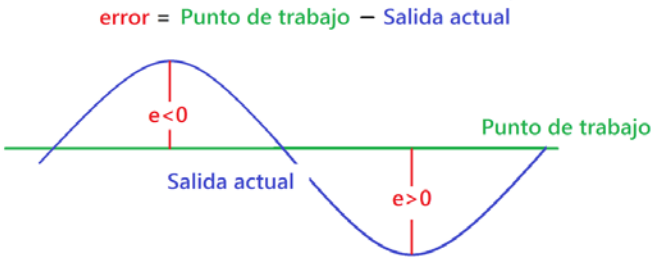


Figura 20: Criterio de símbolos en el método del relé

De esta forma tras el análisis del error cometido se realizaran los cambios en la señal de entrada del sistema, estableciendo así la frecuencia última. El método del relé para llevar a cabo esta acción analiza el signo del error en todo momento y establece un cambio en la onda cuadrada de entra del sistema cuando se produce un cambio en el signo del error. En la figura 21 puede verse un diagrama que detalla el proceso de la selección del valor en cada punto de la onda.

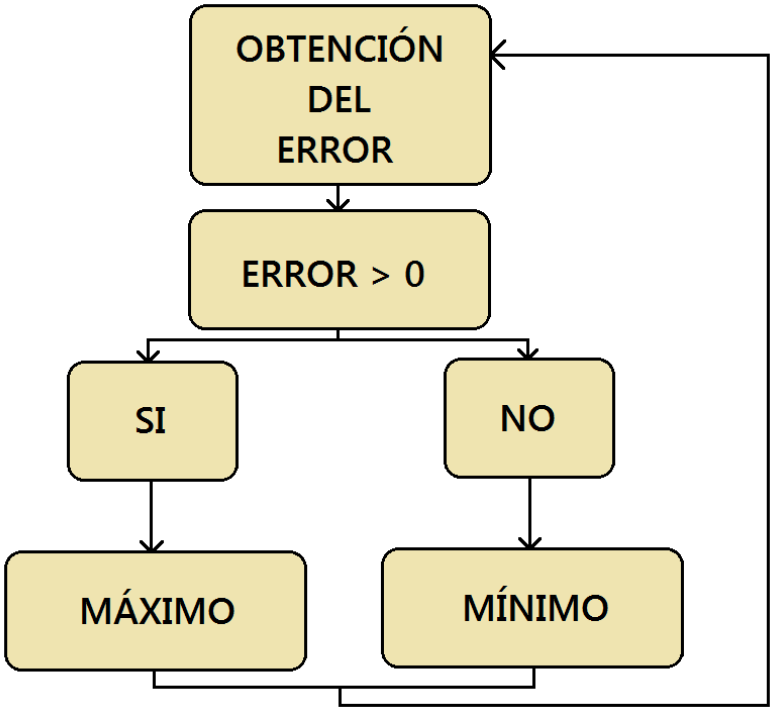


Figura 21: Diagrama análisis del error en el método del relé

La evaluación del error en un solo punto no es suficiente en el desarrollo de esta aplicación, debido a que el armónico obtenido en la salida con este método es imposible de distinguir debido al ruido presente en la planta. El método del relé, contempla otra posibilidad mucho más exacta donde se determina el cambio a través de una histéresis.

Continúa con la misma filosofía mencionada en la figura 21, pero en este caso establece un intervalo de valores donde se puede encontrar el error. En la figura 22 puede verse una representación de esta histéresis.

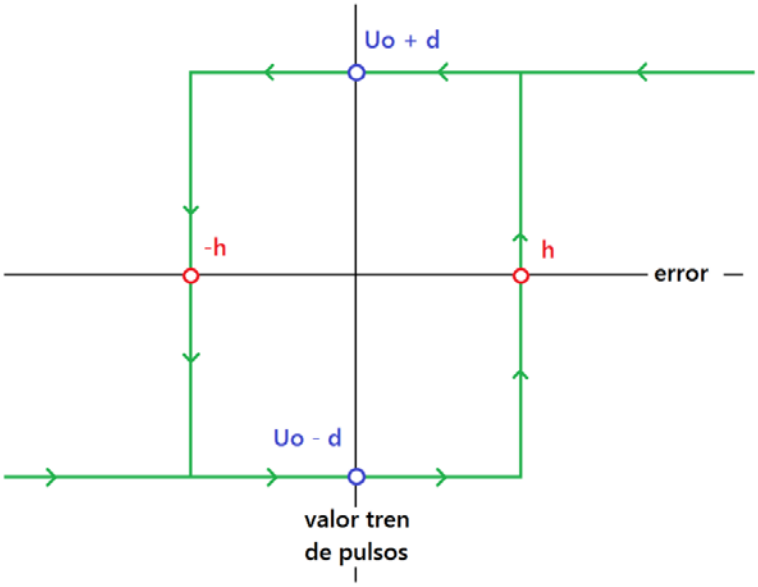


Figura 22: Histéresis del error en el método del relé

Se aprecia que si el error es mayor a un valor h establecido por el usuario, la entrada tomará el valor máximo, si es menor a -h tomará el mínimo, y en el caso en que se encuentre en el intervalo (-h, h) dependerá del valor de la iteración anterior. De esta forma podrá apreciarse en la señal de salida una onda de amplitud mayor, y por lo tanto mucho más fácil de identificar. En la figura 23 puede verse un gráfico que muestra el proceso para el establecimiento del valor de la onda de entrada.

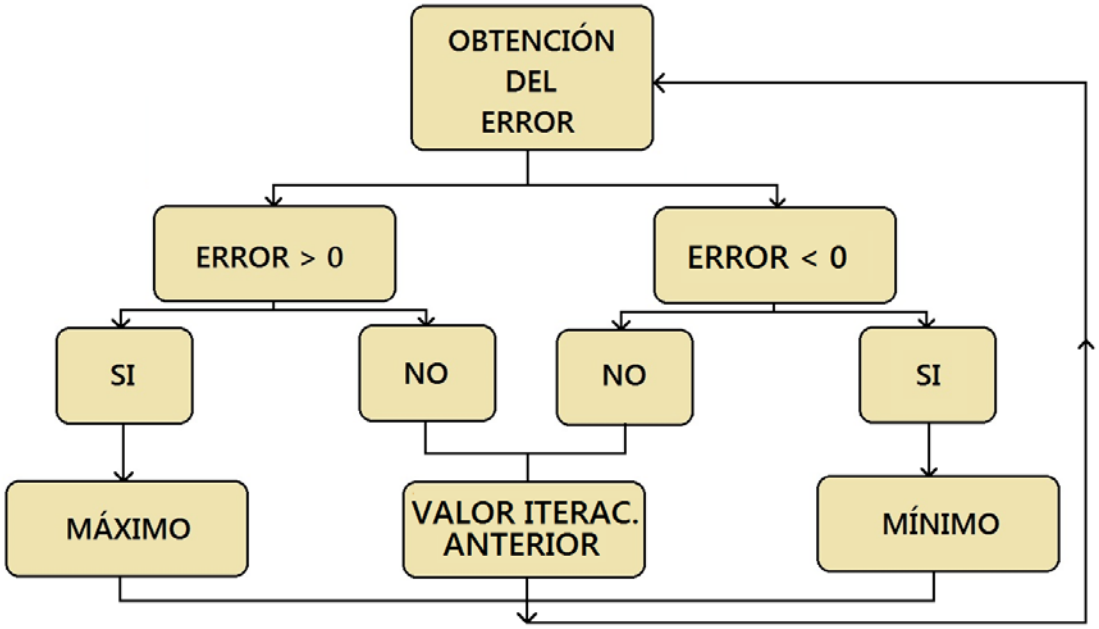


Figura 23: Diagrama análisis del error en el método del relé con Histéresis.

Una vez que la construcción de la onda cuadrada de entrada del sistema cumple con estos requisitos, se genera un armónico en la salida, de tal forma que la determinación de la frecuencia y ganancia última puedan establecerse de forma correcta.

El valor de la frecuencia última está definido como el tiempo necesario para que se produzca una oscilación en sistema, mientras que la ganancia última responde a una ecuación que necesita de las amplitudes de las ondas de entrada (d) y salida (a):

$$K_u = {}^{4d}/_{\pi a}$$

En la figura 24 puede apreciarse una representación gráfica de los valores de necesarios para la determinación de la ganancia y frecuencia última.

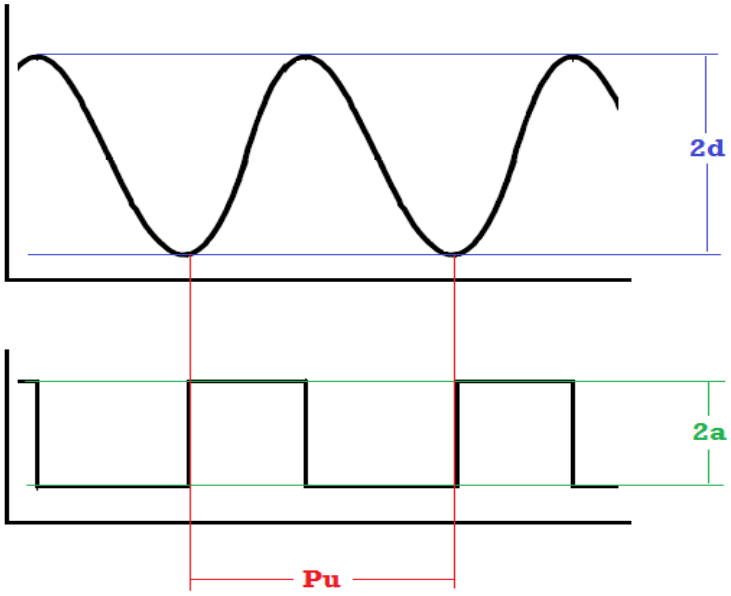


Figura 24: Variables del método del relé

Una vez que se dispone de los valores de la ganancia y frecuencia última se puede proceder al cálculo de los valores que sintonizan el sistema de forma correcta. Para ello el método del relé establece una serie de fórmulas. Estas fórmulas pueden verse en la figura 25.

Specification	K_c	τ_i	τ_d
Original	$0.6K_u$	$P_u/2$	$P_u/8$
Little overshoot	$0.33K_u$	$P_u/2$	$P_u/3$
No overshoot	$0.2K_u$	$P_u/2$	$P_u/3$

Figura 25: Formulas método del relé

Como puede verse en la figura 25 el método contempla diferentes variantes para la sintonía del PID. Todas ellas proporcionan una sintonía correcta, y únicamente se diferencian en la respuesta del sistema en el inicio de su funcionamiento. Para una mayor exactitud la fórmula escogida debe ser acorde con el sistema que se pretenda sintonizar.

Como ya se ha comentado se pretende incluir este método en el funcionamiento del proceso descrito anteriormente añadiendo una lógica al bloque PID predefinido por UNICOS, esto se llevará a cabo directamente sobre el PLC. Para ello el código creado debe incluirse en la fuente de SIMATIC STEP7 dedicada al PID, en la zona definida como *User code begin*. Como se ha detallado en el punto anterior del presente proyecto.

A continuación puede verse el código que autosintoniza el PID detallado y que representa todo el proceso del método del relé.

Código de autosintonía asociado al bloque PID.

Como puede entenderse de la explicación teórica del método del relé anterior, para crear un código que autosintonice el bloque PID únicamente se debe llevar a cabo dos acciones:

- Generación de una onda cuadrada que modifique la entrada y cumpla con las características mencionadas, haciendo oscilar la salida produciendo un armónico determinado.

- Medición en las ondas de entrada y salida de las variables necesarias para el cálculo de las variables del PID.

A continuación puede verse una explicación detallada de todas las partes del código creado que realiza estas dos acciones, detallando las decisiones tomadas.

En primer lugar, como ya se ha mencionado, el método del relé para empezar el proceso desconecta el PID cuando recibe la instrucción de autosintonía. En el diseño de este código se ha decido que esta petición de sintonización se establezca de forma manual, por medio de una entrada digital del PLC, debido a la inexistencia de un método de diagnóstico.

El código creado puede dividirse en dos partes muy diferenciadas, una que recoge las acciones que debe llevar a cabo el autómata mientras no se reciba la orden de sintonía y las que deberá realizar para la auto sintonización del bloque PID.

En la primera parte, mientras no se necesite calcular de nuevo los parámetros del PID, el código hace que todas las variables que definen el método se reinicien y acumula el valor de la entrada del proceso en la variable Uo, para su posterior uso. En la figura 26 puede verse el código característico de esta parte del proceso. Como se ve

también en esta figura debe tenerse en cuenta el criterio para nombrar las variables características de UNICOS citado en el punto anterior, ya que si no se cumple será imposible obtener resultados del código.

```
Variable UNICOS que indica el estado de una ENTRA DIGITAL
IF DB_DI_ALL.DI_SET.PLANTA_1_DI4.PosSt=0 THEN //inicializacion variables para la autosintonia

    Uo:= PLANTA_1_PID1.OutOV;  //Frecuencia de salida
    x1:=0;                      // Variable para indicar fase intermedia histeresis
    x2:=0;                      // Variable para indicar fase intermedia histeresis
    c:=0;                       // Numero de ciclos en el tren de pulsos
    maximo:=0;                  //maximo de la salida en el tren de pulsos
    minimo:=1000;               //minimo de la salida en el tren de pulsos
    HABILITACION:=FALSE;        //habilitacion del temporizador que determina el periodo

END_IF;
```

Figura 26: Código de autosintonía I

La segunda parte del código es más compleja, ya que es la destinada tanto a la construcción de la onda cuadrada que responde a la histéresis como al cálculo de las variables del PID. Para simplificar el diseño de esta parte del código se identificaros 6 acciones principales que debe cumplir para la correcta implementación del método:

- Desconexión del PID, para comenzar la acción de sintonía.
- Calculo del error en la salida
- Generación de la onda que responde a la histéresis
- Detección de un ciclo de la onda de entrada, para el cálculo posterior del periodo.
- Cálculo de las variables de método del relé.
- Transferencia de las variables al bloque PID

Para la desconexión del PID se deben modificar las variables UNICOS que controlan su funcionamiento. En este proceso la regulación se lleva a cabo únicamente en modo manual, por lo tanto para la desconexión del bloque PID bastará con desactivar este modo. Además se debe activar el modo automático para que las acciones posteriores tengan efecto. Esto es debido a que por recomendación directa del CERN se nos propuso modificar únicamente variables automáticas, para que no apareciesen problemas posteriores en el funcionamiento.

En cuanto al cálculo del error en la salida únicamente se deben analizar las variables relacionadas con el punto de trabajo del sistema o *SetPoint* y con la salida del proceso. En la figura 27 puede verse la parte del código que realiza estas dos acciones.

```
//Peticion de autosintonia
IF DB_DI_ALL.DI_SET.PLANTA_1_DI4.PosSt=1 THEN   Petición Autosintonía
    //cambio a modo a automatico
    PLANTA_1_PID1.MMoSt:=0;                     Desactivación Modo Manual
    PLANTA_1_PID1.AuMoSt:=1;                    Activación Modo Automatico

    //Acumulacion del error
    error := PLANTA_1_PID1.MSPSt- PLANTA_1_PID1.HMV ;   Error=setpoint-salida actual
```

Figura 27: Código de autosintonía II

Una vez se dispone del error y se está situado en el modo correcto se puede proceder con la generación del tren de pulsos y la detección de los ciclos de esta onda.

En caso de la creación de una onda que responda a una histéresis se cuenta con 4 puntos de trabajo, la parte derecha de la histéresis donde el error es mayor a h, y la salida toma el valor máximo (Uo + d), la parte izquierda de la histéresis, donde el error es menor a h y la salida debe toma el valor mínimo (Uo - d), y por último la parte central de la histéresis. En esta zona se cuenta con dos puntos de trabajo donde el error pertenece a el intervalo (-h, h) y la salida dependerá de los valores que haya tomado anteriormente.

El diseño de las dos primeras zonas es simple, ya que únicamente se debe establecer una condición evaluado el valor del error y asignando la salida correspondiente. Sin embargo en el caso de la zona central de la histéresis se deben identificar los valores anteriores que ha tomado la salida para asignar de forma correcta el nuevo valor. Para ello se ha optado por colocar dos variables a modo de testigo que recojan la información de los valores del error durante todo el proceso y permitan tomar la decisión correcta a la hora de asignar el nuevo valor.

En este punto aparece un problema importante, ya que si en el momento inicial de la autosintonía el error se encuentra en el intervalo (-h, h), el código no posee información suficiente para la toma de decisiones en la asignación de la salida. Para solucionarlo se ha optado por forzar la salida del proceso hasta que se alcance un error mayor a h. En este momento el código ya puede comenzar a generar la onda que responde a la histeresis desde la parte derecha. En la figura 28 puede verse el código comentado perteneciente a esta parte.

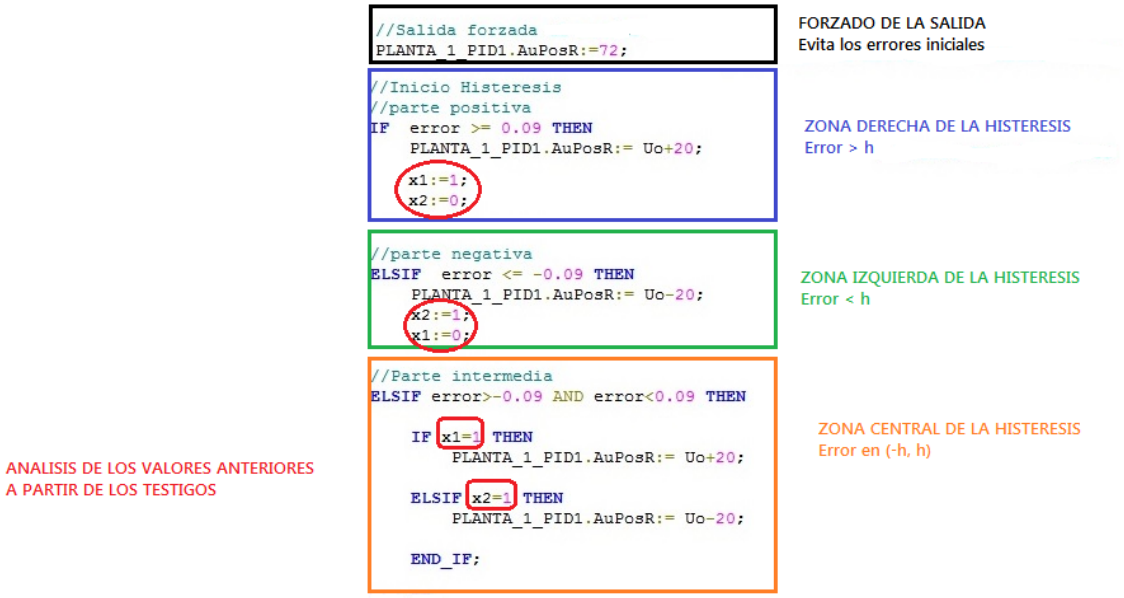


Figura 28: Código de autosintonía III

Este código genera una onda cuadrada que hace oscilar a la salida de una forma cíclica. Aparece en este punto la necesidad de detectar los ciclos o por lo menos uno de ellos para el cálculo posterior de las variables necesarias para los parámetros del PID. En la figura 29 puede verse una salida real del sistema obtenida desde el SCADA donde se aprecia la oscilación provocada.

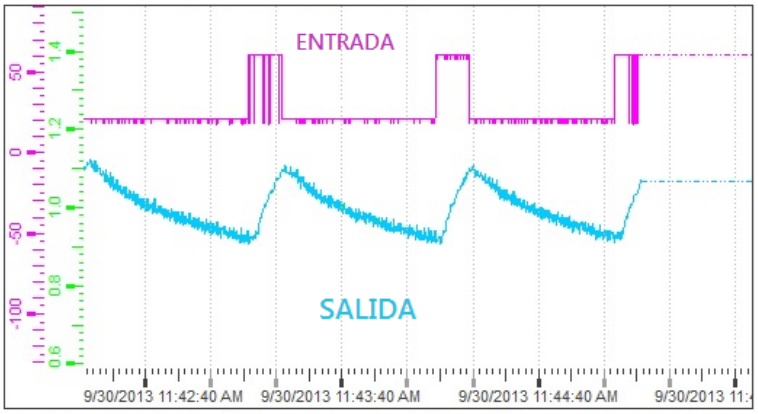


Figura 29: Oscilación del sistema

La decisión para la detección de cada uno de los ciclos ha sido analizar la salida del sistema antes y después de ejecutar el código de la histéresis. Detectando así cualquier cambio que se produzca. Más concretamente se va a detectar cada uno de los flancos de subida la onda cuadrada de entrada. Para ello se utiliza una variable denominada j que se activa siempre que el tren de pulsos se encuentre en la salida negativa, cuando cambie a salida positiva la variable se desactiva y se incrementa el número de ciclos, detectando así únicamente el flanco de subida de la onda. En la figura 30 puede verse el código comentado perteneciente a esta parte.

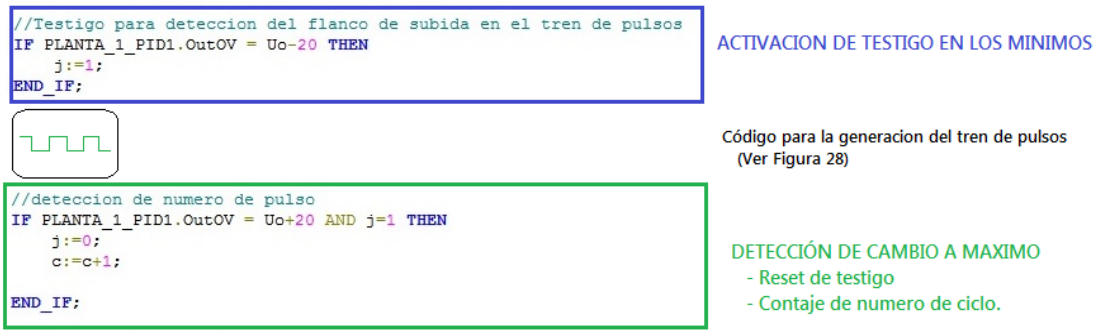


Figura 30: Código de autosintonía IV

En este momento ya se está en disposición de calcular las variables características del método del relé para la sintonización del PID. Estos valores son similares en todos los ciclos, y como se dispone en todo momento de la información del ciclo en el que se encuentra, se ha decido medir estos valores únicamente en un solo ciclo. Cuando este ciclo termine, los valores quedaran acumulados y se indicara que la autosintonía ha finalizado.

En el caso de la amplitud del tren de pulsos, denominada d, es un parámetro introducido por el usuario en el código, ya que al fijar el máximo y el mínimo de la onda se debe indicar este valor, por lo tanto se convierte en una constante en el método.

El cálculo de la amplitud de la onda de salida, denominada a, tampoco es estrictamente necesario. Esto es debido a que la respuesta de este sistema es muy rápida, en el momento en el que la entrada cambia, la salida también lo hace y por lo tanto los valores máximos y mínimos van a ser muy similares al valor establecido por la histéresis. De esta forma el valor de h, fijado por el usuario en el código, coincide con el valor a, que marca la amplitud de esta onda de salida. A pesar de ello se ha optado por calcular estos valores en aras de conseguir una mayor precisión del sistema. Se ha optado por utilizar un algoritmo clásico de detección de máximos y mínimos, donde se comparan los valores actuales con los anteriores y se acumula el mayor, en el caso del máximo o el menor en el caso del mínimo.

Por ultimo para calcular el valor de Pu, que es el tiempo que tarda en ejecutarse un ciclo, se ha optado por la utilización de un temporizador preconfigurado por SIEMENS que aporta el valor en binario de este tiempo. El uso de estos temporizadores no es aconsejable si no se conoce todo su funcionamiento.

En este caso se ha utilizado un temporizador denominado S_PULSE. Este temporizador mantiene su salida activa siempre que la habilitación también lo este y el tiempo preconfigurado no se haya sobrepasado. También dispone de una opción de *reset* para pararlo siempre que se desee, aunque habilitación y tiempo sigan activos. En la figura 31 puede verse una gráfica que explica su funcionamiento.

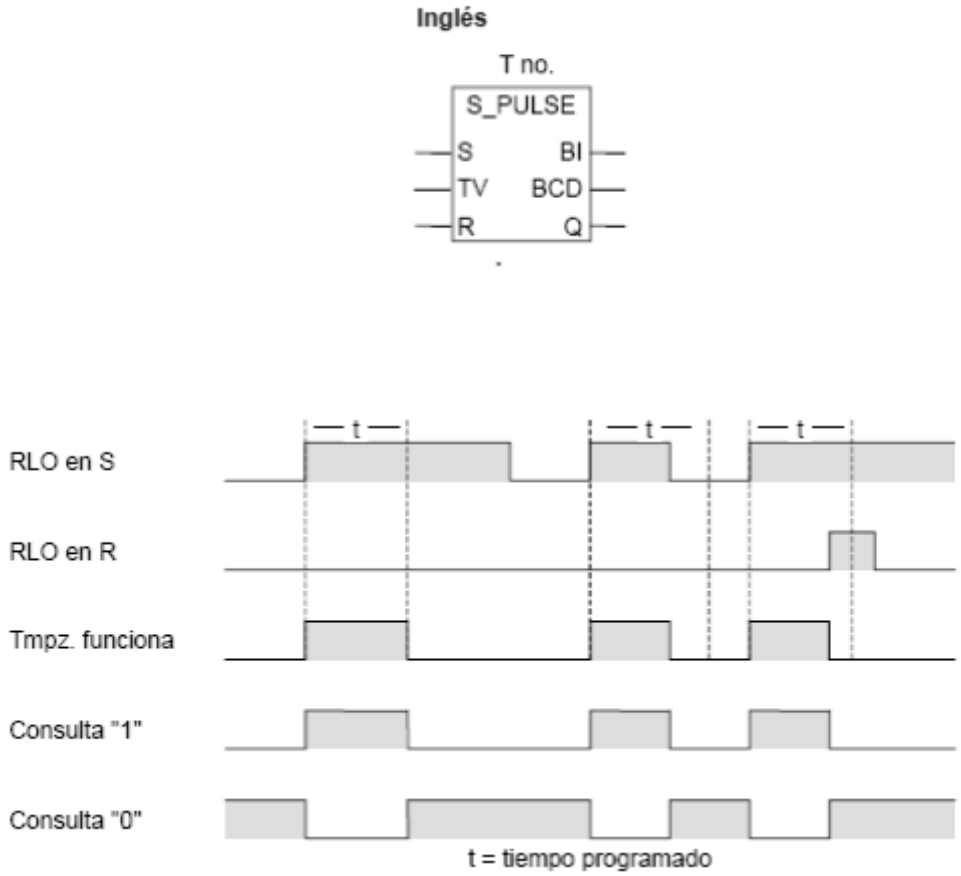


Figura 31: Temporizador S_PULSE.

Esta función se aprovecha de tal modo, que se configura para que solo este habilitado durante el ciclo donde se van a medir los valores, se le inserta además un tiempo lo suficientemente alto para que no se cumpla nunca. De esta forma a través de su variable binaria BI, el temporizador informa del tiempo que ha estado activo.

Se debe tener en cuenta que esta variable informa del tiempo que le queda al temporizador en binario para que se cumpla su tiempo establecido por defecto, es decir, al inicio por ejemplo estará en 100 segundos, e ira decrementando según se ejecute. Por lo tanto es necesario transfórmala, en primer lugar a un valor entero, con la función de SIEMENS *WORD_TO_INT* y posteriormente hacer una resta para que informe del tiempo que el temporizador ha estado activo.

En la figura 32 puede verse el código comentado que recoge todos estos cálculos de las variables necesarias para la posterior sintonización del método.

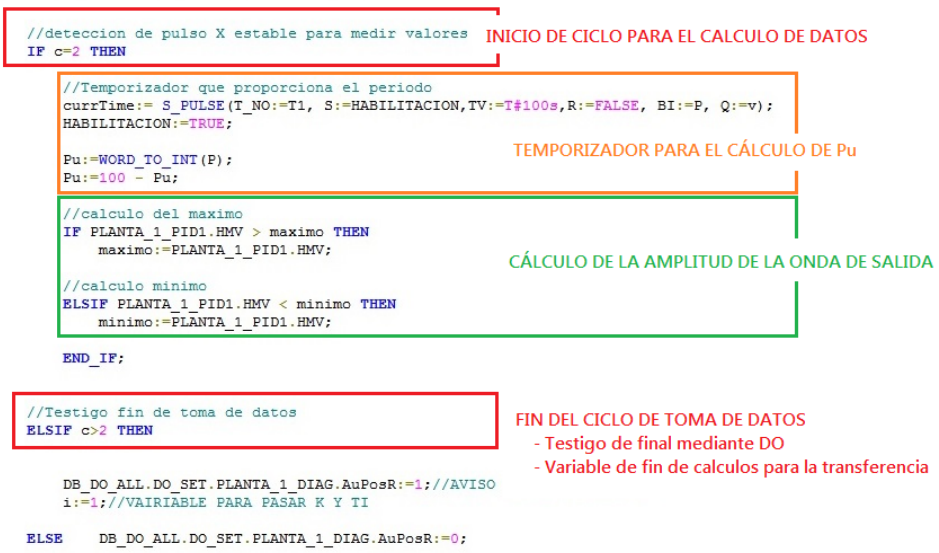


Figura 32: Código de autosintonía V

Por último se deben calcular las variables que sintonizan de forma correcta PID y transferirlas al bloque para que cuando se desee se ponga de nuevo en funcionamiento la regulación del bloque y este funcione de forma correcta.

Para ello se deben utilizar dos tipos de variables UNICOS. En primer lugar se debe habilitar la transferencia de valores a través de una variable Booleana, posteriormente transferirlos y a continuación volver a deshabilitar la variable anterior. Si no se sigue este proceso la trasferencia no se hará efectiva, a continuación puede verse un ejemplo simple con el uso de estas variables:

PLANTA_1_PID1.AuPPID.EKc := True	Variable que habilita la trasferencia
PLANTA_1_PID1.AuPPID.Kc := 300	Asignación de valor a la K del PID
PLANTA_1_PID1.AuPPID.Kc := False	Variable que deshabilita la trasferencia

A continuación, en la figura 33, puede verse el código creado para la transferencia, donde se incluyen además las formulas proporcionadas por el método del relé para el cálculo de K y Ti de PID.

```
//Calculo y transferencia de las variables al PID
IF i=false THEN
    PLANTA_1_PID1.AuPPID.EKc:=false;
    PLANTA_1_PID1.AuPPID.ETi:=false;

ELSIF i=true THEN
    PLANTA_1_PID1.AuPPID.EKc:=TRUE;
    PLANTA_1_PID1.AuPPID.ETi:=TRUE;
    PLANTA_1_PID1.AuPPID.Kc:=(0.6*4*20)/(3.1416*((maximo-minimo)/2));
    PLANTA_1_PID1.AuPPID.Ti:=Pu/2;
    i:=false;

END_IF;
```

Figura 33: Código de autosintonía V

En este punto la creación del código estaría finaliza, en la figura 34 puede verse el código completo, que incluye todo lo definido en los apartados anteriores. Puede verse además que este código sólo se ejecutara cuando el modo de funcionamiento del proceso sea el modo NORMAL.

```
//MODO NORMAL + AUTOSINTONIA

IF  DB_DI_ALL.DI_SET.PLANTA_1_ENCENDIDO.PosSt=1 THEN //Inicio modo normal


    IF(DB_DI_ALL.DI_SET.PLANTA_1_DI4.PosSt=0) THEN //inicializacion variables para la autosintonia

        Uo:= PLANTA_1_PID1.OutOV;  //Frecuencia de salida
        x1:=0;                     // Variable para indicar fase intermedia histeresis
        x2:=0;                     // Variable para indicar fase intermedia histeresis
        c:=0;                      // Numero de ciclos en el tren de pulsos
        maximo:=0;                 //maximo de la salida en el tren de pulsos
        minimo:=1000;              //minimo de la salida en el tren de pulsos
        HABILITACION:=FALSE;       //habilitacion del temporizador que determina el periodo

    END_IF;


    //Peticion de autosintonia
    IF DB_DI_ALL.DI_SET.PLANTA_1_DI4.PosSt=1 THEN
        //cambio a modo a automatico
        PLANTA_1_PID1.MMoSt:=0;
        PLANTA_1_PID1.AuMoSt:=1;

        //Acumulacion del error
        error := PLANTA_1_PID1.MSPSt- PLANTA_1_PID1.HMV ;

        //Testigo para deteccion del flanco de subida en el tren de pulsos
        IF PLANTA_1_PID1.OutOV = Uo-20 THEN
            j:=1;
        END_IF;


        //Salida forzada al maximo para provocar que la histeresis comience en zona positiva
        PLANTA_1_PID1.AuPosR:=72;

        //Inicio Histeresis
        //parte positiva
        IF  error >= 0.09 THEN                          //inicio tren de pulsos
            PLANTA_1_PID1.AuPosR:= Uo+20;
            x1:=1;
            x2:=0;

        //parte negativa
        ELSIF  error <= -0.09 THEN
            PLANTA_1_PID1.AuPosR:= Uo-20;
            x2:=1;
            x1:=0;

        //Parte intermedia
        ELSIF error>-0.09 AND error<0.09 THEN

            IF x1=1 THEN
                PLANTA_1_PID1.AuPosR:= Uo+20;

            ELSIF x2=1 THEN
                PLANTA_1_PID1.AuPosR:= Uo-20;

            END_IF;

        END_IF;
```

```
    END_IF;


    //deteccion de numero de pulso
    IF PLANTA_1_PID1.OutOV = Uo+20 AND j=1 THEN
        j:=0;
        c:=c+1;

    END_IF;

   //deteccion de pulso X estable para medir valores
   IF c=2 THEN

       //Temporizador que proporciona el periodo
       currTime:= S_PULSE(T_NO:=T1, S:=HABILITACION,TV:=T#100s,R:=FALSE, BI:=P, Q:=v);
       HABILITACION:=TRUE;

       Pu:=WORD_TO_INT(P);
       Pu:=100 - Pu;

       //calculo del maximo
       IF PLANTA_1_PID1.HMV > maximo THEN
           maximo:=PLANTA_1_PID1.HMV;

       //calculo minimo
       ELSIF PLANTA_1_PID1.HMV < minimo THEN
           minimo:=PLANTA_1_PID1.HMV;

       END_IF;


   //Testigo fin de toma de datos
   ELSIF c>2 THEN


       DB_DO_ALL.DO_SET.PLANTA_1_DIAG.AuPosR:=1;//AVISO FIN AUTOSINTONIA
       i:=1;//VAIRIABLE PARA PASAR K Y TI

   ELSE    DB_DO_ALL.DO_SET.PLANTA_1_DIAG.AuPosR:=0;


   END_IF;

   //Calculo y transferencia de las variables al PID
   IF i=false THEN
       PLANTA_1_PID1.AuPPID.EKc:=false;
       PLANTA_1_PID1.AuPPID.ETi:=false;


   ELSIF i=true THEN
       PLANTA_1_PID1.AuPPID.EKc:=TRUE;
       PLANTA_1_PID1.AuPPID.ETi:=TRUE;
       PLANTA_1_PID1.AuPPID.Kc:=(0.6*4*20)/(3.1416*((maximo-minimo)/2));
       PLANTA_1_PID1.AuPPID.Ti:=Pu/2;
       i:=false;

   END_IF;
END_IF;
```

Figura 34: Código de autosintonía completo

El código creado al igual que la mayoría de métodos de autosintonía requiere de una sintonización previa, que proporcione valores cercanos al punto de trabajo, pero en ningún caso que llegue a sintonizarle de forma perfecta, es una tarea sencilla que se puede llevar a cabo incluso de forma manual desde el SCADA. Esta sintonía previa es necesaria debido a que el método del relé requiere la entrada del sistema en el momento de desconexión del PID. Si la sintonía es demasiado mala el código creado no será capaz de construir una onda que consiga hacer oscilar al sistema.

En la figura 35 puede verse un ejemplo de un PID con una sintonía previa demasiado brusca. Se observa que la señal de control oscila entre los valores máximos y mínimo (0Hz y 70Hz), esto provoca que la variable inicial Uo del método del relé tome uno de estos dos valores, haciendo imposible la construcción de un tren de pulsos válido. La solución más aparente para este problema sería calcular el valor medio de estos valores y aplicarle al método, esto no es del todo correcto. En la mayoría de las situaciones el valor medio de la frecuencia de entrada calculado de esta forma toma valores que no consiguen hacer oscilar al sistema, es decir se le aplicará una frecuencia de entrada al sistema que mantenga el error constante impidiendo que se mueva a lo largo de la histéresis.

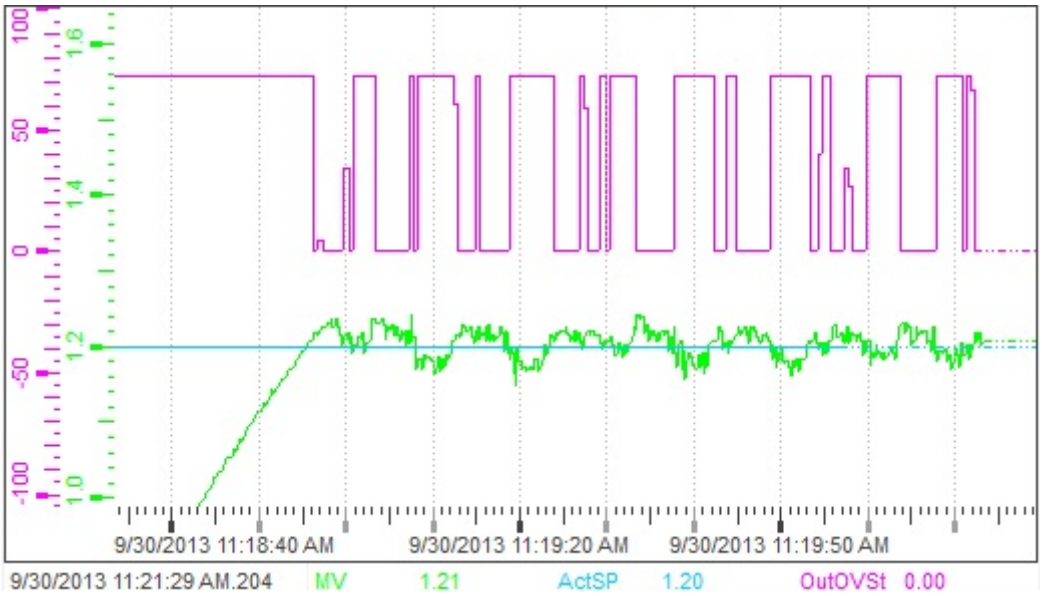


Figura 35: PID sin sintonía previa

Siempre que se haya tenido en cuenta esta consideración, de sintonización previa, este código consigue sintonizar el PID de forma rápida y sencilla. Además el método del relé cuenta con tres tipos de fórmulas que modifican el sobrepico inicial, se ha decido escoger las formulas originales debido de nuevo a la velocidad de la respuesta del sistema. El resto de opciones provocaban que la velocidad de establecimiento de la señal de salida aumentase sin apreciarse mejoras significativas.

A continuación pueden verse discutidos algunos de los resultados obtenidos tras la ejecución de este código.

Este método no consigue únicamente sintonizar un PID que no tuviese una respuesta estable en su salida, sino que también es capaz de mejorar la sintonía de bloque que ya está ajustado de forma correcta. En la figura 36 puede verse la gráfica de un PID correctamente sintonizado, pero con un sobrepico en el inicio y un tiempo de establecimiento demasiado grandes. La ejecución del método sobre este PID cambiara sus parámetros mejorando sus características.

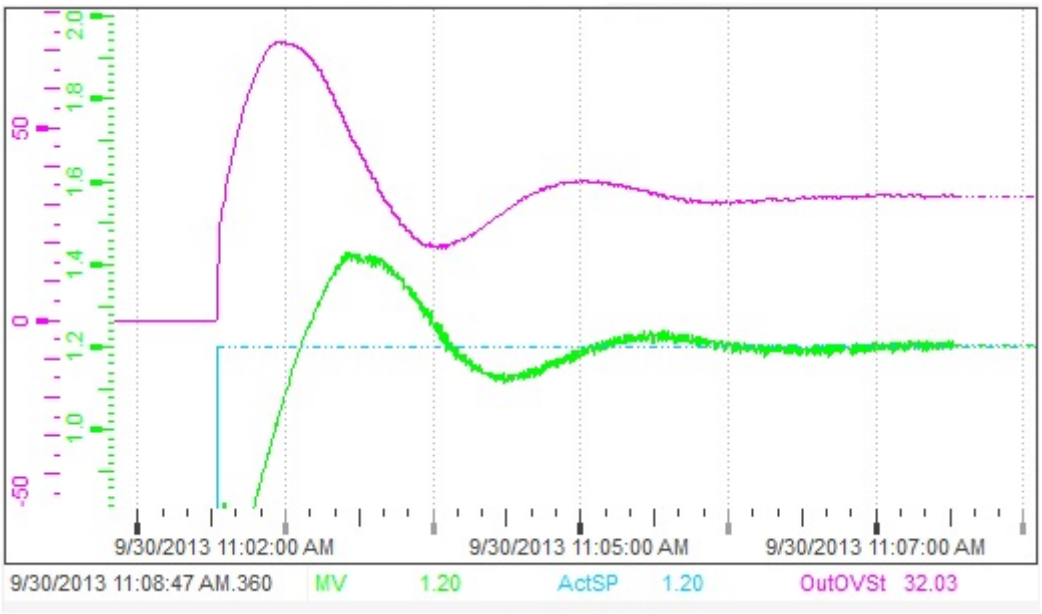


Figura 36: PID con sintonización no aceptable

En la figura 37 puede verse un ejemplo clásico de un PID mal sintonizado, como puede verse la salida del sistema se estabiliza por encima del punto de trabajo, tiene un error estacionario durante todo el funcionamiento. Este caso es el más común en la industria y a continuación se va a mostrar como el código creado estabiliza esta señal.

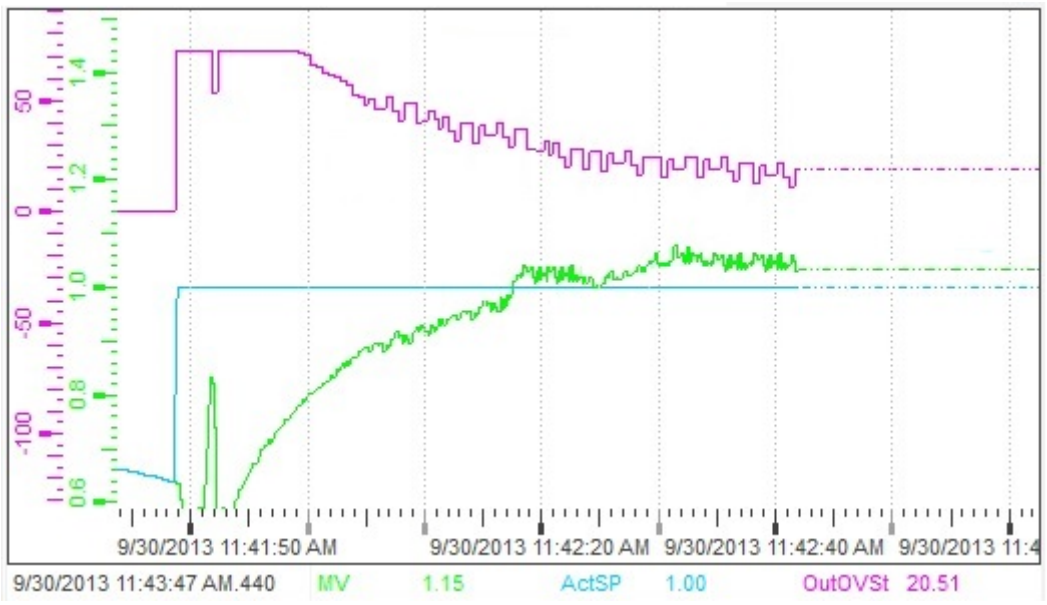


Figura 37: PID mal sintonizado.

Este regulador cuenta con unos valores de K=300 y Ti=10, que como puede apreciarse no proporcionan un rendimiento óptimo para la situación actual. La ejecución del código proporcionara unos nuevos valores para K y Ti (Termino proporcional, y termino integral) que proporcionen una respuesta estable.

Cuando el usuario ve esta respuesta entiende que el PID no tiene una sintonía aceptable y activa la autosintonía. En la gráfica de la figura 38 puede apreciarse un ejemplo de este momento y como la entrada del sistema comienza a generar la onda cuadrada descrita anteriormente. Puede apreciarse también, que no es necesario dejar al PID que estabilice la señal, basta con encender la autosintonía cuando la salida sea próxima al punto de trabajo.

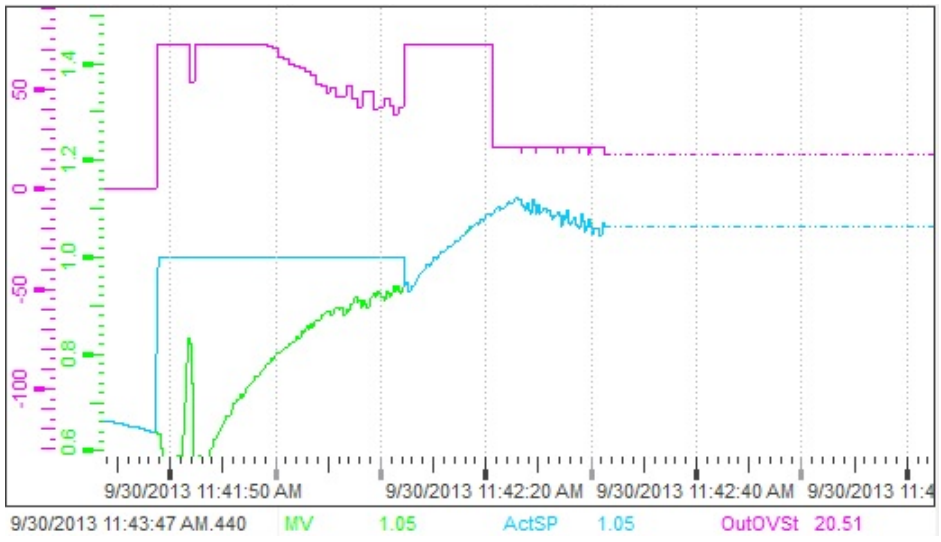


Figura 38: Inicio de autosintonía

La autosintonía debe dejarse encendida hasta que se estabilice la señal de salida, y pueda apreciarse un patrón repetitivo, como puede verse en la figura 39. En este momento el código realizara los cálculos para hallar los nuevos valores del PID e informara por medio de un LED al usuario que ha finalizado. Como podrá verse detallado en el punto del presente proyecto dedicado al y manejo del SCADA.

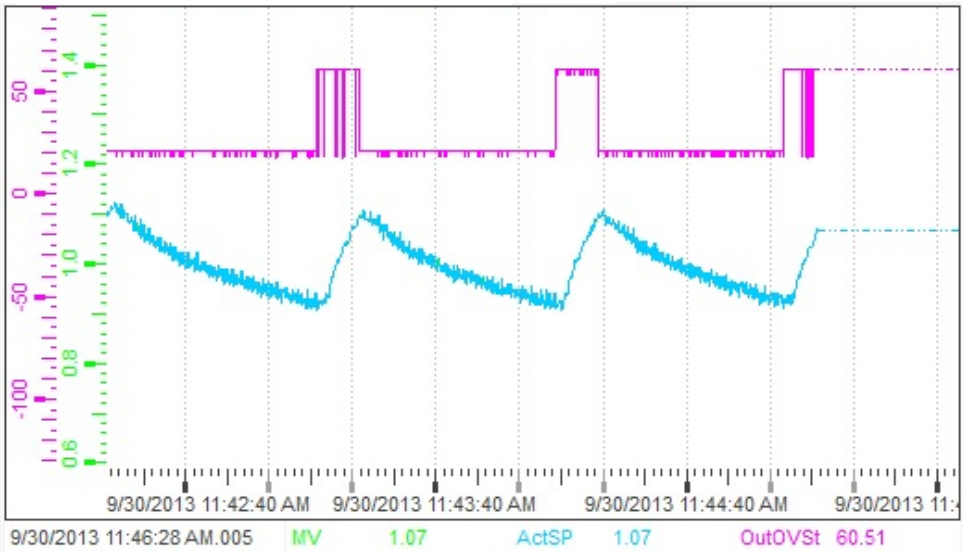


Figura 39: Gráficas del proceso de autosintonía

En este momento el método del relé ha terminado de realizar los cálculos y transfiere los nuevos valores al bloque PID. Como puede verse reflejado en la figura 40, que recoge la pantalla destinada al PID del SCADA.

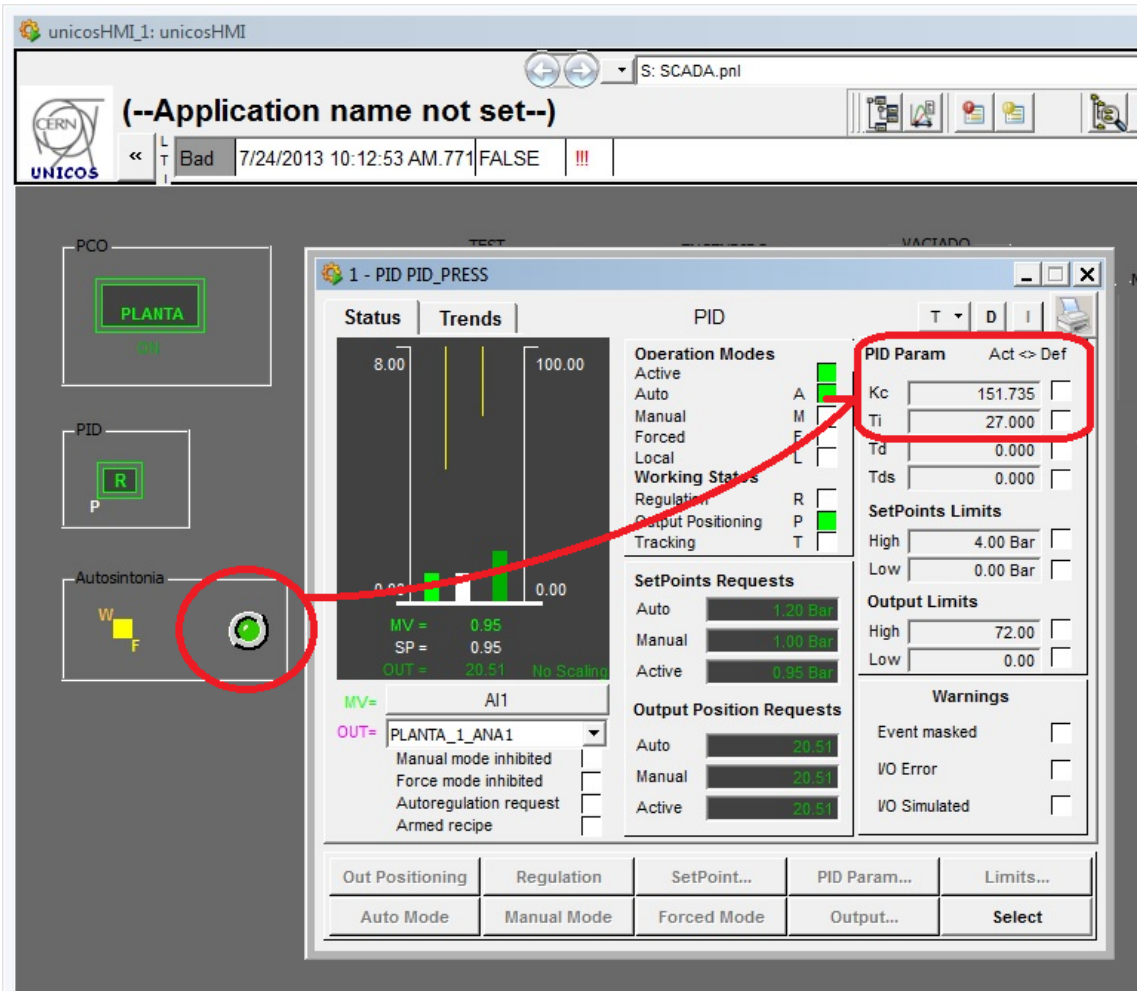


Figura 40: Fin de autosintonía

El regulador PID ya tiene introducidos los valores para el correcto funcionamiento del sistema, el usuario únicamente debe parar la autosintonía y volver a hacer regular al bloque. Como puede verse en la figura 40 los nuevos valores PID han sido modificados respecto a los iniciales:

- $K_{antigua} = 300$ $K_{nueva} = 151.735$
- $Ti_{antigua} = 10$ $Ti_{nueva} = 27$

A continuación en la figura 41 puede verse la gráfica del PID con esta nueva sintonía, donde puede apreciarse la mejora de las características de la salida, tras aplicar el método.

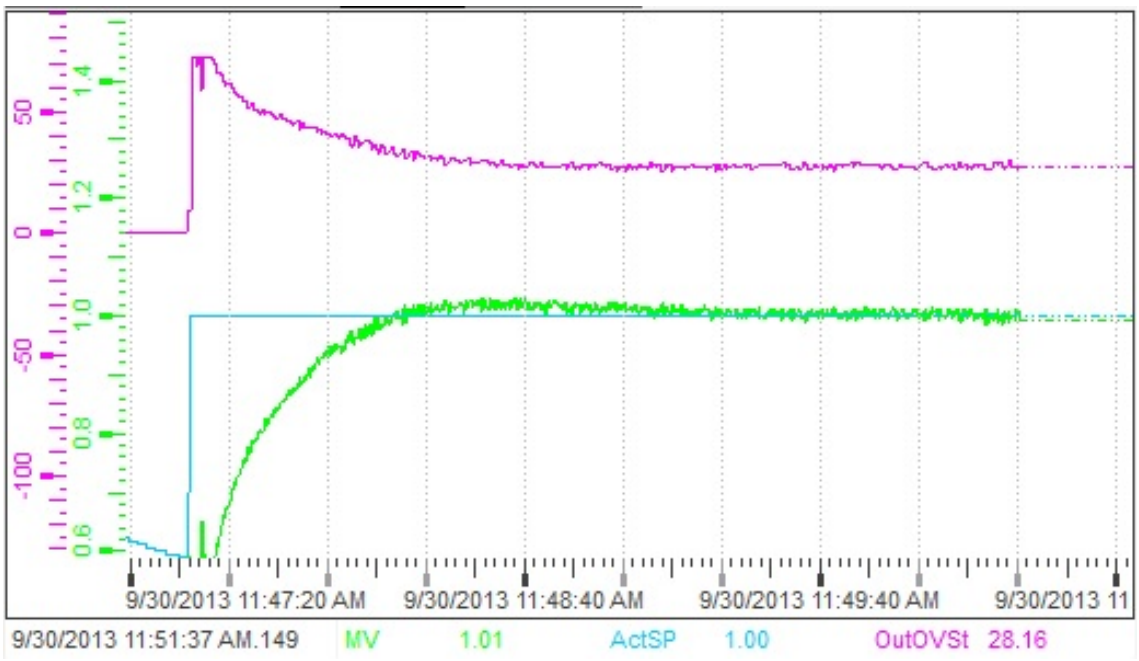


Figura 41: PID tras la aplicación del método del relé.

5.2.2 Creación bloque grafcet de funcionamiento (modos)

Para continuar con la comprensión del *framework* UNICOS, se ha decidido crear un Grafcet que controle el funcionamiento del proceso. En los puntos anteriores se detalla las opciones que presenta SIMATIC STEP 7 para la creación de un Grafcet. A continuación se detalla como UNICOS aprovecha estas funciones creando una secuencia que además cumple con su filosofía.

Como se ha visto en el punto donde se describe la planta del presente proyecto, además del funcionamiento normal de la planta se han incluido varios estados adicionales para crear un funcionamiento más complejo, que se aproxime al funcionamiento de una planta industrial.

Como puede verse en la figura 42, se han añadido tres modos adicionales, y una secuencia de funcionamiento que debe ser traducida de tal forma que cumpla con la filosofía de UNICOS y que el programa STEP 7 sea capaz de comprenderla.

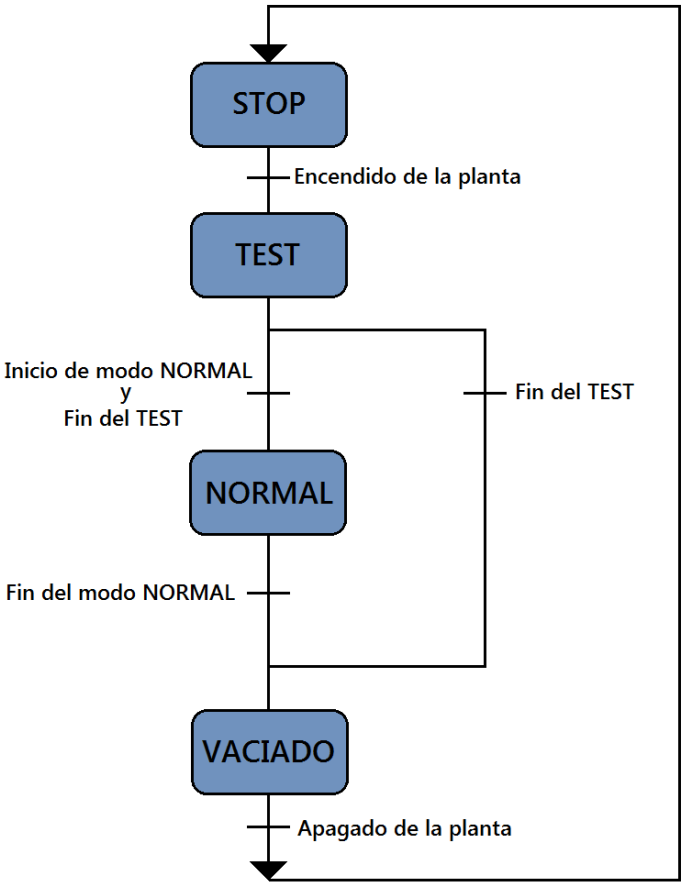


Figura 42: Diagrama de funcionamiento de la planta

Se debe hacer referencia en este punto a la presencia de un cuarto modo implícito en todo el funcionamiento de la planta, el MODO LOCAL.

MODO LOCAL

Este modo permite inhabilitar todas las funciones del SCADA a través de la activación de un interruptor situado en la planta, de esta forma se impide al técnico que maneja el SCADA modificar los valores de funcionamiento del proceso mientras un operario realiza por ejemplo acciones de mantenimiento en la planta. La naturaleza de este modo permite asociarle con los *interlocks* de UNICOS, que pueden verse detallados en el punto tres de esta memoria. Para ello únicamente se debe asociar el *interlock* deseado con la variable que identifica el interruptor situado en la planta. Esta asociación debe incluirse en la fuente del proyecto PLANTA_1_PCO_IL.

Antes de describir como crear el diagrama de funcionamiento cumpliendo con UNICOS, se debe tener en cuenta que un diagrama grafcet identifica cada uno de los estados de funcionamiento y las variables necesarias para el paso de un estado a otro, denominadas transiciones. En este caso viendo el diagrama de la figura 43, se pueden identificar cuatro estados y cinco transiciones, que pueden verse detalladas en la figura 17:

Estados
STOP
TEST
NORMAL
VACIADO

Transiciones
STOP→TEST
TEST→NORMAL
TEST→VACIADO
NORMAL→VACIADO
VACIADO→STOP

Figura 43: Estados y transiciones del Grafcet de funcionamiento

Para una correcta compresión del método deben describirse antes de empezar con la creación del Grafcet deben detallarse las funciones de cada uno de los estados y de las transiciones

Estados

STOP

En este estado la planta permanece en reposo, todos los elementos deben estar parados y debe ser imposible ponerlos en funcionamiento tanto desde el SCADA como desde el campo.

TEST

En este modo de funcionamiento permite chequear el correcto funcionamiento del compresor. Se le someterá a una rampa de diferentes frecuencias observando que la velocidad de giro del motor varíe. Como puede verse en la figura 44, cuando se da la orden para activar este modo, se envía una frecuencia de 40 Hz al motor, y se mantiene durante cinco segundos, a continuación la frecuencia vuelve a cero.

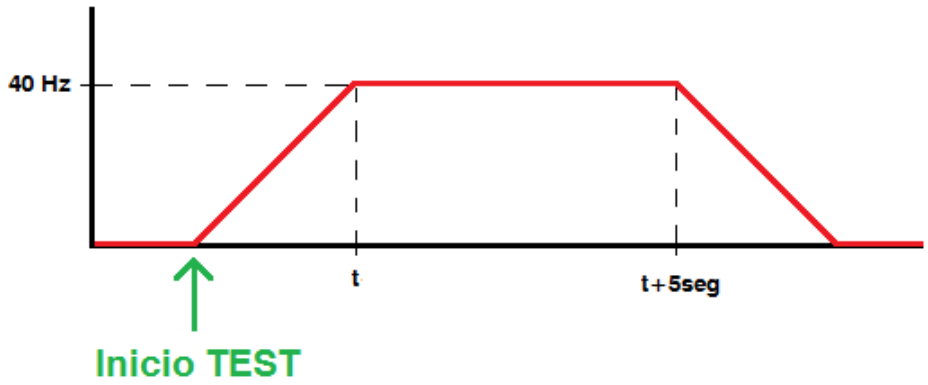


Figura 44: Rampa modo TEST

En la figura 45 puede verse el código asociado para la construcción de esta rampa. Pueden identificarse variables ajenas a esta construcción como la variable *M0.0*, pero IMPRESCINDIBLES para marcar el inicio y final del modo para la asociación posterior con las transiciones, como se verá más adelante.

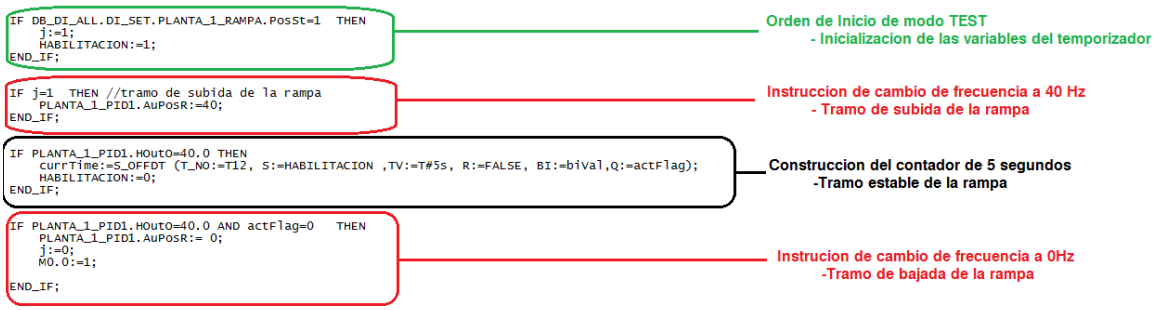


Figura 45: Código del modo TEST

NORMAL

Este modo representa el funcionamiento normal de la planta, es el bloque principal de funcionamiento, ya que aquí se dispone de la opción de la regulación de la salida mediante el bloque PID modificado, además de funciones de forzado de variables o apagado de la planta.

VACIADO

Este modo tiene la función de vaciar por completo de aire todo el sistema de tuberías. Cuando la planta recibe la orden de paro del modo normal la electroválvula de abre hasta que el usuario compruebe que el vaciado del tanque ha terminado.

Transiciones

STOP→TEST

Esta transición marca el arranque de la planta, se activara cuando desde el SCADA se dé la orden de iniciar el TEST.

TEST→NORMAL

En este caso la activación del modo normal y el fin de la construcción de la rampa activan la transición

TEST→VACIADO

Esta transición se establece para contemplar la posibilidad de que el usuario, únicamente desee verificar el correcto funcionamiento del compresor. Se activará cuando la construcción de la rampa haya finalizado y la orden de modo normal no esté activa.

NORMAL→VACIADO

Cuando el usuario desea apagar la planta tras haber regulado su salida, da la orden de vaciado, es en este momento cuando se activa esta transición.

VACIADO→STOP

La transición se activa cuando el usuario da la orden de apagado una vez se ha asegurado de que la presión en el sistema es lo suficientemente baja para no provocar accidentes.

Una vez identificados todos los estados y las transiciones se puede llevar a cabo la creación del Grafcet de funcionamiento. Para cumplir con la filosofía UNICOS los nombres de las transiciones como de los estados debe cumplir una serie de características. En el gráfico de la figura 46 se pueden ver todos los elementos con su nombre en UNICOS.

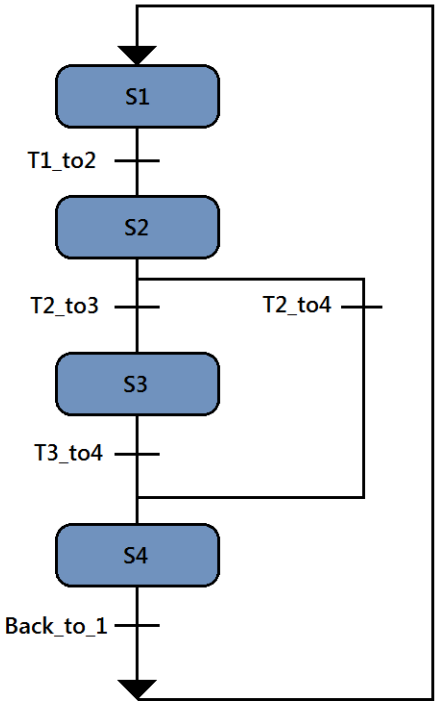


Figura 46: Diagrama de funcionamiento de la planta en UNICOS

Se puede observar en la figura 16 que el nombre de los estados únicamente señala el paso entre los estados que controla, su lógica debe incluirse en una fuente del proyecto como se verá más adelante. En el caso de los estados, únicamente se deben numerar dependiendo de la posición que ocupen en el proceso, las acciones que se deben realizar en cada uno de los estados estarán incluidas en las fuentes del proyecto. UNICOS no establece una sola fuente para esta lógica, sino que la incluye en la fuente correspondiente al elemento que se desee controlar, de esta forma las acciones de cada estado están distribuidas por todo el proyecto.

Como se mencionó en los puntos anteriores la opción escogida para la creación de Grafcet ha sido la de la creación de un bloque funcional (FB), esto es debido a que es la opción que se escoge en la arquitectura UNICOS. Par ello se debe crear el bloque funcional, seleccionando el lenguaje tipo GRAPH, además de un bloque de datos (DB) asociado a este FB. La creación de estos bloques debe ser llevada a cabo a través de la opción insertar del menú de SIMATIC STEP7.

En la figura 47 se puede ver la creación del bloque funcional. Se observa que el nombre asociado a este bloque debe de cumplir con la nomenclatura UNICOS. Debido a que este bloque controla todo el proceso está relacionado con el objeto PCO, de esta forma el nombre escogido ha sido PlantaPCO_Graphet. En el caso del bloque de datos (DB), se debe crear de la misma forma un bloque en este caso DB con el nombre PlantaPCO_Graphet_SL, asociado al bloque funcional anterior (PlantaPCO_Graphet).

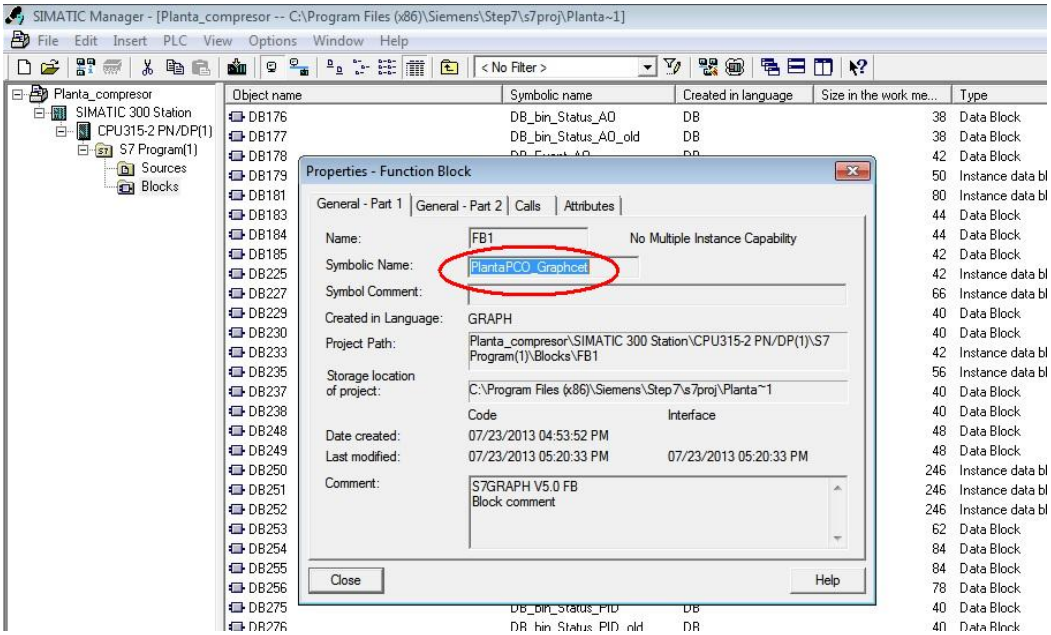


Figura 47: Creación de bloque para el Grafcet

Una vez creados los dos bloque ya se puede crear el grafcet, para ello debe abrirse el bloque recién creado, PlantaPCO_Graphet, y completarle con la secuencia elegida, se deben crear como variables de entrada todas las transiciones necesarias. Estas nuevas variables se declaran en la parte inferior de la pantalla como se vio en el punto de “GRAPH” del presente proyecto. En la figura 48 se puede ver el grafcet de funcionamiento creado para este proceso.

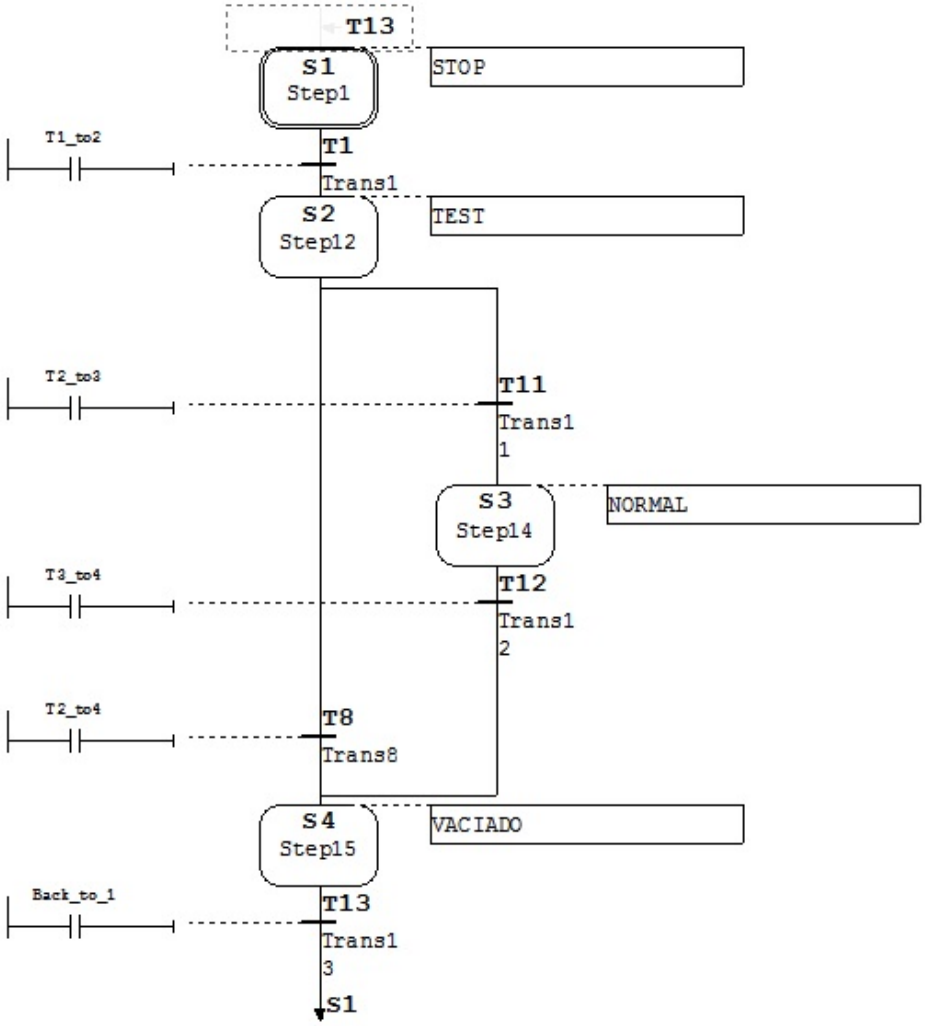


Figura 48: Grafcet de funcionamiento

El Grafcet ya está creado, pero el proceso no ha terminado, para que tenga validez se debe asociar con el bloque PCO, ya que como se ha mencionado anteriormente este objeto controla todo el funcionamiento, y por la tanto si se desea establecer una secuencia de funcionamiento deberá ser llevada a cabo por el PCO. Es importante saber que si no se asocia este Grafcet con el PCO el proyecto seguirá funcionando, incluso puede cumplir con las especificaciones establecidas en las transiciones, pero no cumple con la arquitectura UNICOS.

Para asociar el Grafcet con el PCO ello se deben modificar las fuentes del proyecto. En primer lugar las transiciones deben quedar definidas en la fuente PLANTA_1_PCO_GL, se acumularán en un vector como puede verse en la figura 49.

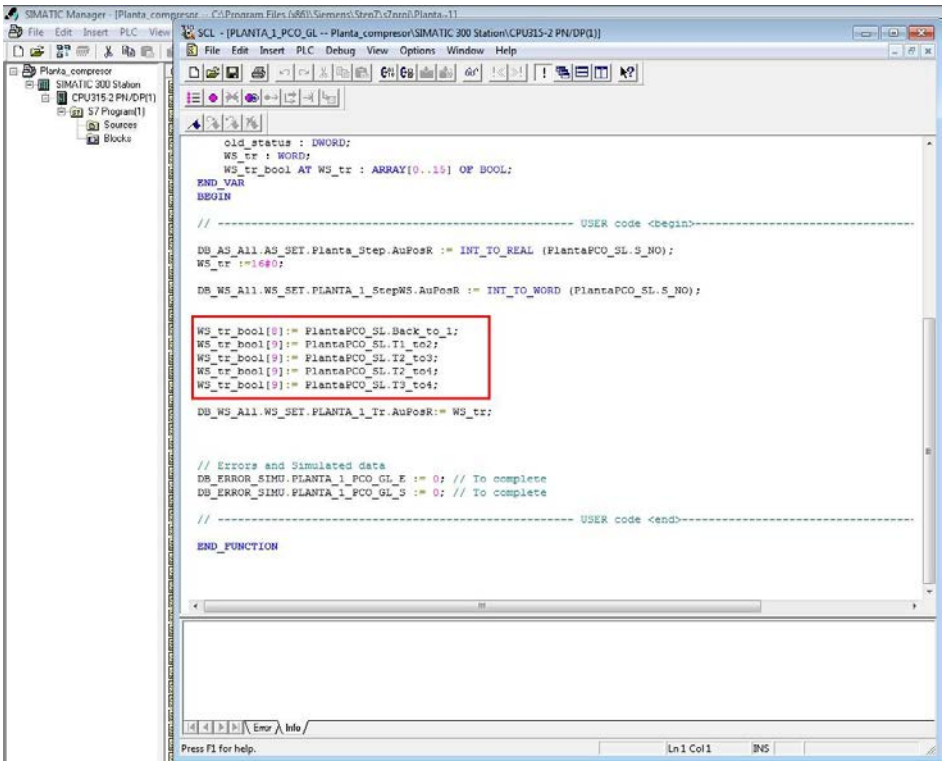


Figura 49: Creación de transiciones del grafcet

En este punto las transiciones están creadas, pero son variables vacías. La lógica que define el funcionamiento de estas transiciones se encuentra en la fuente PLANTA_1_PCO_TL. Debe abrirse e incluir aquí el funcionamiento de cada transición, en la parte del código de usuario, como se ve en la figura 50.

```
// ---------- USER code <begin>----------
//STEP 1 TO 2
PlantaPCO_SL.T1_to2 := DB_DI_ALL.DI_SET.PLANTA_1_RAMPA.PosSt ;

//STEP 2 TO 3
PlantaPCO_SL.T2_to3 := DB_DI_ALL.DI_SET.PLANTA_1_ENCENDIDO.PosSt AND M0.0;

//STEP 3 TO 4
PlantaPCO_SL.T3_to4 := (NOT DB_DI_ALL.DI_SET.PLANTA_1_ENCENDIDO.PosSt) ;

//STEP 2 TO 4
PlantaPCO_SL.T2_to4 := (NOT DB_DI_ALL.DI_SET.PLANTA_1_ENCENDIDO.PosSt) AND M0.0;

//BACK TO 1
PlantaPCO_SL.Back_to_1 := I0.3;

DB_ERROR_SIMU.PLANTA_1_PCO_TL_E := 0 ; // To complete
DB_ERROR_SIMU.PLANTA_1_PCO_TL_S := 0 ; // To complete


// ---------- USER code <end>----------
```

Figura 50: Lógica de las transiciones

Por ultimo para que la asociación con el PCO tenga validez se debe incluir una llamada al Grafcet en la fuente PLANTA_1_PCO_SL, como se indica en la figura 51.

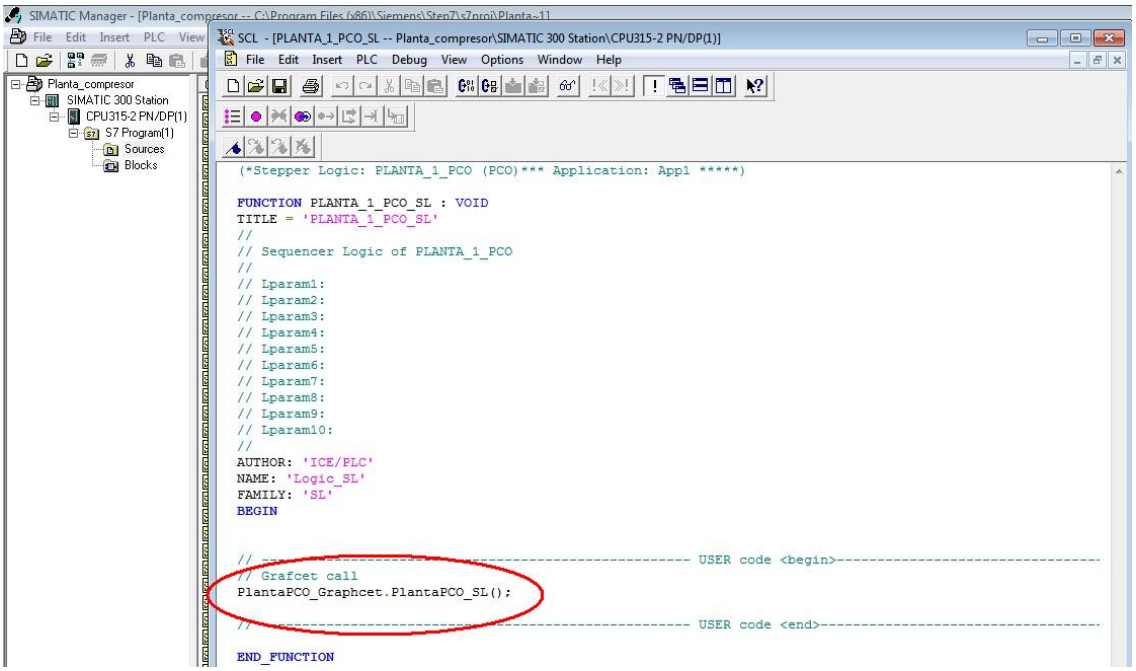


Figura 51: Llamada a el Grafcet de funcionamiento

En este punto el Grafcet está completamente creado, y define el funcionamiento del proceso. Los cambios en la planta o en SCADA activaran sus transiciones cambiando entre los distintos tipos de funcionamiento. El PVSS no muestra la imagen de este Grafcet, esta opción se encuentra en SIMATIC STEP7, pudiendo ver el paso entre estados en tiempo real. Esta opción no es demasiado recomendable, ya que requiere de una gran cantidad de recursos, lo que ralentizaría el funcionamiento del SCADA.

5.3 Adaptación de la solución en el marco de UNICOS en el nivel de supervisión

UNICOS al igual que en la capa de control permite adaptar la solución generada por UAB a la aplicación diseñada. Esta adaptación debe ser llevada a cabo cumpliendo siempre las directrices marcadas por UNICOS para seguir cumpliendo con su jerarquía y filosofía.

UNICOS ha generado una solución para el SCADA PVSS, pero cada elemento tiene a su disposición una ventana propia, no es posible en este momento el visionado de una sola vez de todo el proceso. Para conseguir controlar el proceso desde una sola ventana es necesario crear un panel de control que incluya todos los elementos.

5.3.1 Creación de panel principal

El panel principal es la zona donde se llevará a cabo todo el trabajo, debe contar con todos los elementos que identifican el proceso, además de con las herramientas necesarias para su control. Para su creación se utiliza una herramienta de PVSS denominada Gedi. Esta herramienta es un editor que permite la creación de paneles a través de unos *widgets*. Proporcionados por UNICOS.

La creación del panel de control a través de esta herramienta debe llevarse a cabo después de haber creado el proyecto completo en PVSS con el archivo de UAB ya importado, como se vio anteriormente.

El editor Gedi debe arrancarse desde la consola del proyecto como se puede observar en la figura 52.

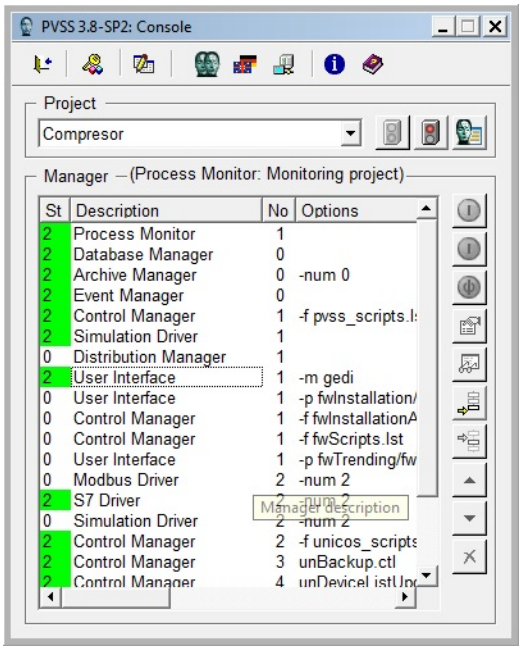


Figura 52: Editor de paneles Gedi

Una vez arrancado este editor se despliega una ventana que contiene el área de trabajo necesaria para la creación del panel. El área de trabajo está dividida en cuatro partes, como puede verse en la figura 53.

- Barra de Herramientas: Se encuentra en la parte superior de la ventana, contiene elementos como la creación de nuevos archivos o el visionado de los paneles creados.

- Editor de Propiedades: Situado en la parte izquierda de la ventana. Permite la modificación de los elementos añadidos al panel, como por ejemplo el cambio de color, tamaño, etc.

- Menú de objetos. Se encuentra en la parte central de la ventana y contiene todos los *Widgets* UNICOS necesarios para describir el funcionamiento del proceso en el panel.

- Zona de trabajo: Situada en la parte derecha de la ventana. Es el área destinada a la creación del panel, aquí se deben incluir los *Widgets* UNICOS que describan el proceso.

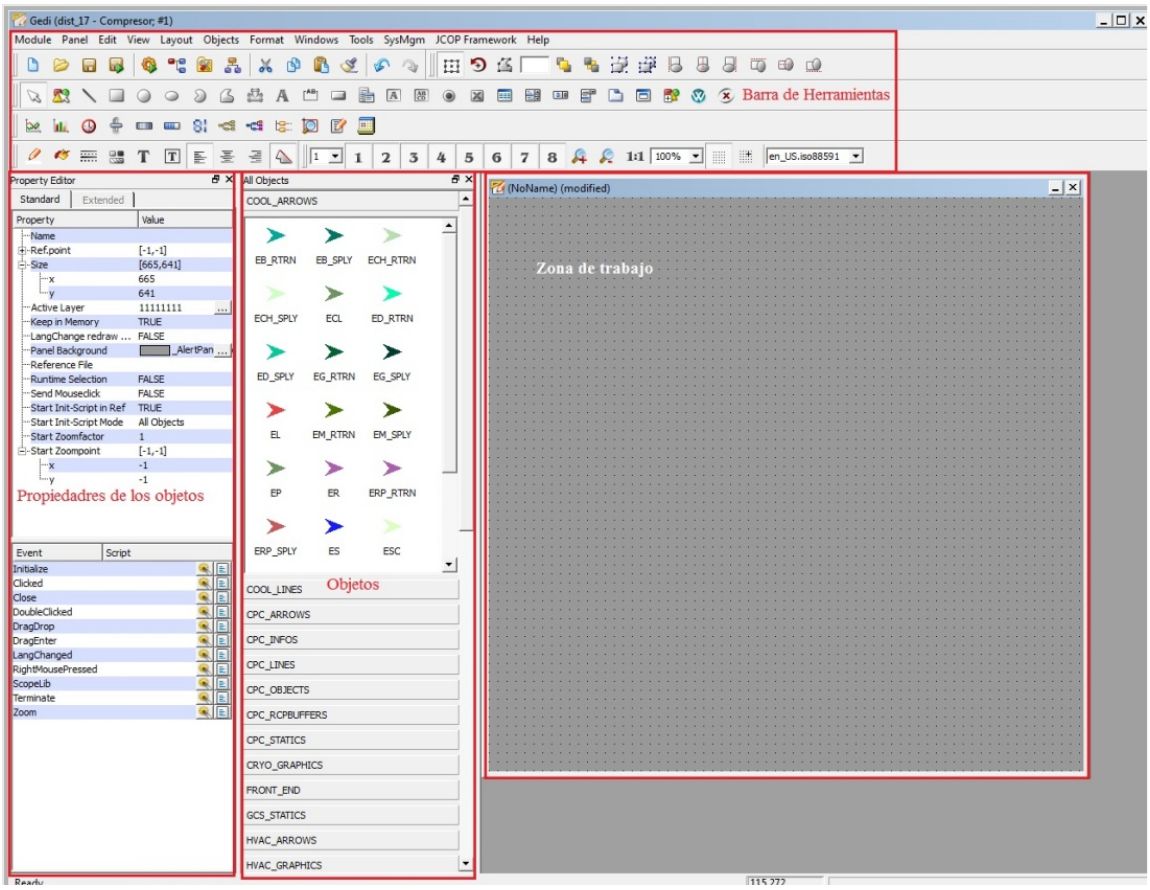


Figura 53: Espacio de trabajo Gedi

En esta última zona se incluirán los elementos necesarios para la descripción del proceso. UNICOS establece una norma propia de colores y aspecto para todos los *Widgets* que se incluyan en los paneles de control, en la figura 54 puede verse un el aspecto de estos además de una leyenda que indica su funcionamiento. En el punto siguiente puede verse como asociar correctamente estos *Widgets* con las variables del proceso.

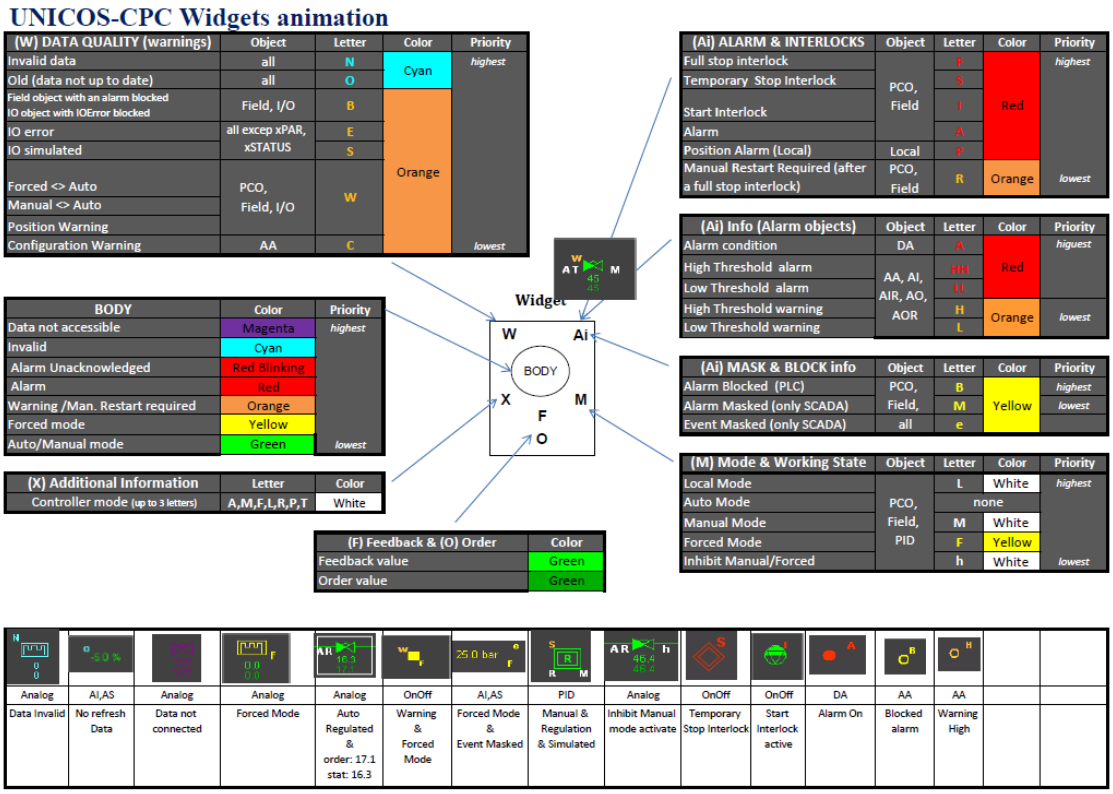


Figura 54: Widgets UNICOS

5.3.1.1 Asociación de variables

Es de gran importancia este apartado, ya que no todas las variables de un bloque trabajan de igual manera y por tanto hay que señalar la indicada para cumplir los requisitos impuestos.

Para incluir un *Widget* al panel se debe buscar en el menú de objetos el elemento que le caracterice y arrastrarle sobre la zona de trabajo. En este momento se despliega un menú para la asociación con las variables del proceso. A continuación se describe paso a paso esta asociación de variables. Esta secuencia puede verse en la figura 55.

1. Arrastrar el elemento deseado

2. Pulsar sobre el icono con forma de árbol de la parte inferior de la ventana.

3. Se ha desplegado una nueva ventana donde se debe buscar el elemento del proceso al que se desea asociar el *Widget*. Se dispone de un método manual de búsqueda y uno automático. En este menú aparecerán todos los bloques especificados en la hoja EXCEL que sean asociables al *Widget*.

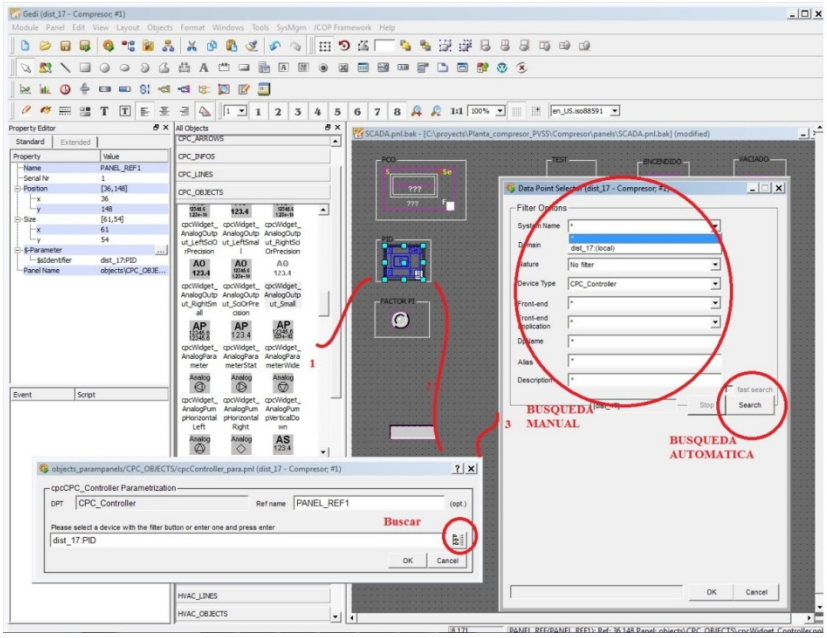


Figura 55: Direccionamiento de los objetos

Se debe reseñar que en ocasiones los elementos a representar en el panel no son objetos completos, como PIDs, PCOs, sino que únicamente se desea incluir un led o una pantalla de información sobre una única variable del bloque. En este caso el proceso para la asociación de la variable es diferente. El método a seguir coincide con el descrito anteriormente excepto que a la hora de la búsqueda no aparecen los objetos UNICOS para asociarles, en su lugar aparecerá una ventana donde se debe escoger el objeto y la variable deseada, como se ve en la figura 56.

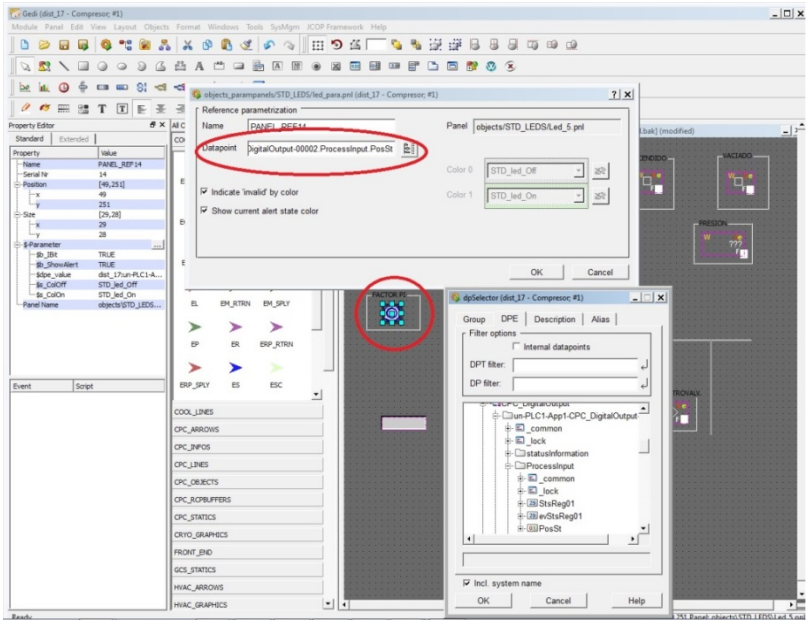


Figura 56: Asociación de variables a un led

Es importante que una vez completado el panel principal, debe ser guardado en la misma carpeta donde se encuentre el resto del proyecto PVSS

5.3.1.2 Derechos de administrador

El panel creado se puede abrir directamente desde el *Gedi*, pero esta opción únicamente permite el visionado de las variables, para tener derechos de administrador y poder modificar y controlar en todo momento las variables del sistema es imprescindible llevar a cabo la importación del archivo que contiene el panel a la ventana de UNICOS-HMI (-p visión/graphical) contenida en la consola de PVSS. Que puede verse en la figura 57.

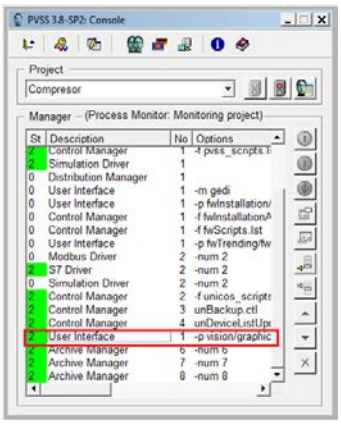


Figura 57: Ejecución de la ventana UNICOS HMI.

Se debe tener en cuenta que siempre que se ejecute esta ventana UNICOS-HMI debe introducirse el usuario *admin* como se vio en puntos anteriores, si no se hace no se podrán modificar ningún tipo de valores.

Para importar se debe seguir la ruta UNICOS→Configuration→ Application, esto despliega una ventana donde se incluye la ruta hasta el panel creado como puede verse en la figura 58.

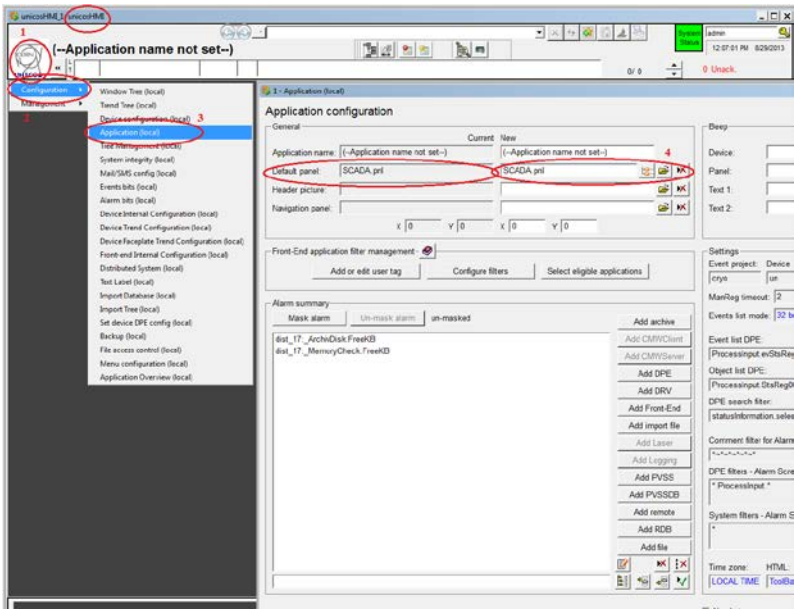


Figura 58: Importación del panel principal.

Esta acción ha importado el panel principal a la ventana UNICOS-HMI y para poder acceder a él, únicamente se debe pulsar sobre la casa de la parte superior de la ventana como se ve en la figura 59. Si el usuario *admin* esta introducido no debe de haber problemas para la modificación de parámetros.

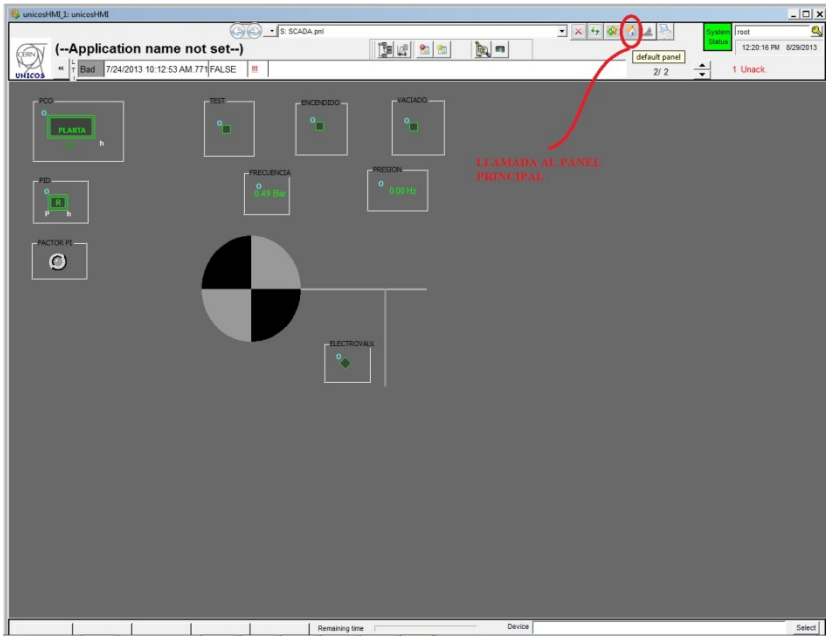


Figura 59: Ventana UNICOS-HMI

5.3.2 Adquisición de datos

Para el diseño de algunos procesos es necesario obtener valores del SCADA para que sean analizados fuera de línea, PVSS permite la extracción de datos a través de un archivo EXCEL. Para ello en primer lugar se debe seleccionar la gráfica del elemento que se desee extraer los datos como se ve en la figura 60.

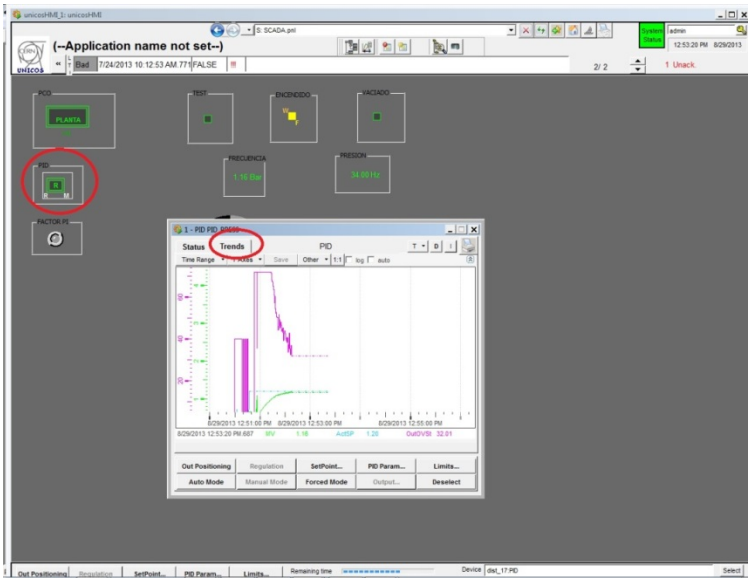


Figura 60: Grafica para extracción de datos

Con el elemento activo y situado en esta grafica debe seleccionarse la ruta que puede verse en la figura 61. Una vez se presione sobre *ok* aparecerá un archivo Excel con los datos de la gráfica en la ruta seleccionada.

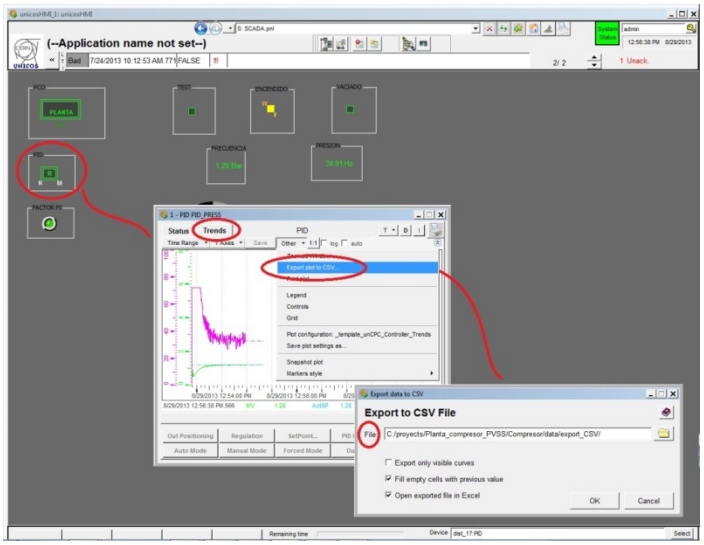


Figura 61: Exportación de datos a un fichero Excel

PVSS permite realizar esta operación en UNICOS-HMI con cualquier elemento que disponga en su menú de una representación gráfica.

5.4 Control de planta

Como ya se ha mencionado anteriormente la supervisión de la planta será llevada a cabo mediante el SCADA PVSS. El *framework* UNICOS-CPC aporta una serie de elementos para que la supervisión mediante este programa se lleve a cabo de forma sencilla e intuitiva. A continuación puede verse como se ha de manejar el SCADA creado para un correcto funcionamiento de la planta.

En primer lugar se debe hacer una reseña del aspecto y manejo que tienen cada uno de los elementos que proporciona UNICOS. Cada uno de estos elementos tiene un *Widget* asociado dependiendo de la tarea que realice. Todos ellos disponen además de un menú propio para la modificación del modo de funcionamiento, forzado de variables, etc. que se despliega pinchando dos veces sobre su figura.

Dependiendo del modo de funcionamiento del objeto, o de la situación de su salida, la figura que lo representa tomara una serie de colores. Este criterio esta unificado por UNICOS, como se vio anteriormente en este mismo punto (Ver figura 54)

Sin embargo los menús de cada objeto disponen de funciones y características diferentes dependiendo de la función que realicen. A continuación pueden verse detalladas las funciones de los menús de un bloque PID y de una entrada digital, para la explicación de este concepto

En la figura 62 puede verse el menú de funcionamiento de un bloque PID. En primer lugar para la modificación de cualquier valor, se debe seleccionar el objeto en la pestaña inferior derecha, *Select*, esto permite la selección del modo de funcionamiento, el forzado de las variables o incluso cambiar los parámetros que lo definen.

Es importante mencionar la diferencia de funcionamiento entre el modo manual y automático de los bloques UNICOS. El modo manual permite modificar directamente todos los valores del bloque desde el SCADA, pero en el caso del modo automático únicamente permite modificar parámetros en el código asociado al bloque, por este motivo no se deben incluir variables manuales en la lógica. Ademas no se debe olvidar que todos los bloques por defecto, se encuentran en modo automático. De esta forma si nos hay ninguna lógica asociada a él el objeto permanecerá en reposo, es decir con sus variables a cero.

Puede apreciarse en esta figura que el menú no solo permite modificar el funcionamiento del bloque con las pestañas inferiores, sí no que también informa de todos los valores que tiene en sus variables internas. Además objetos complejos como este, permiten visionar mediante una gráfica su comportamiento a lo largo del tiempo. Para ello únicamente se debe pulsar en la pestaña superior, *Trends*.

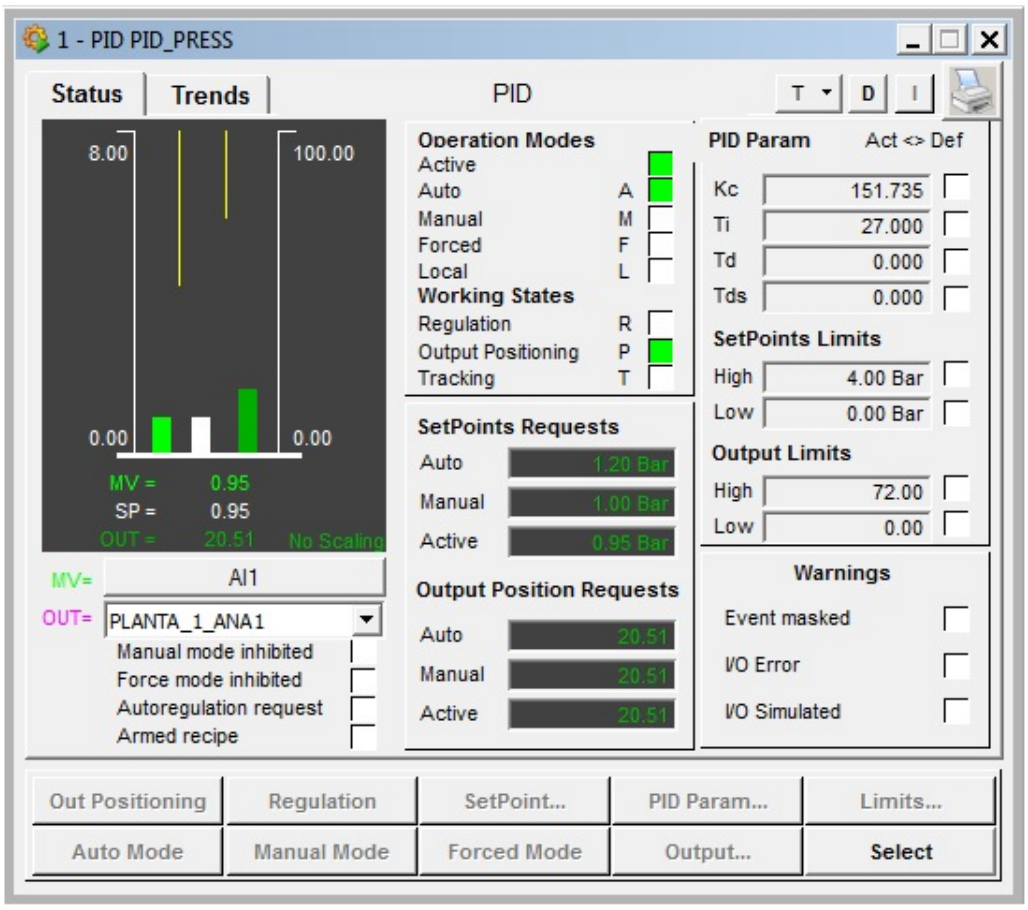


Figura 62: Menú de un bloque PID.

Todas estas características hacen que estos objetos tengan un funcionamiento sencillo. Por ejemplo en el caso de este bloque, si se desea poner a regular la salida, únicamente se debe seleccionar el objeto, escoger el modo manual y pulsar sobre la pestaña de regulación. En este momento el bloque comenzará a controlar la salida del sistema.

En la figura 63 puede verse el menú de una entrada digital, donde se aprecia a simple vista la diferencia con él mencionado anteriormente, En este caso el menú es mucho más simple, debido a que una entrada digital cuenta con muchas menos funciones que un bloque complejo como el PID.

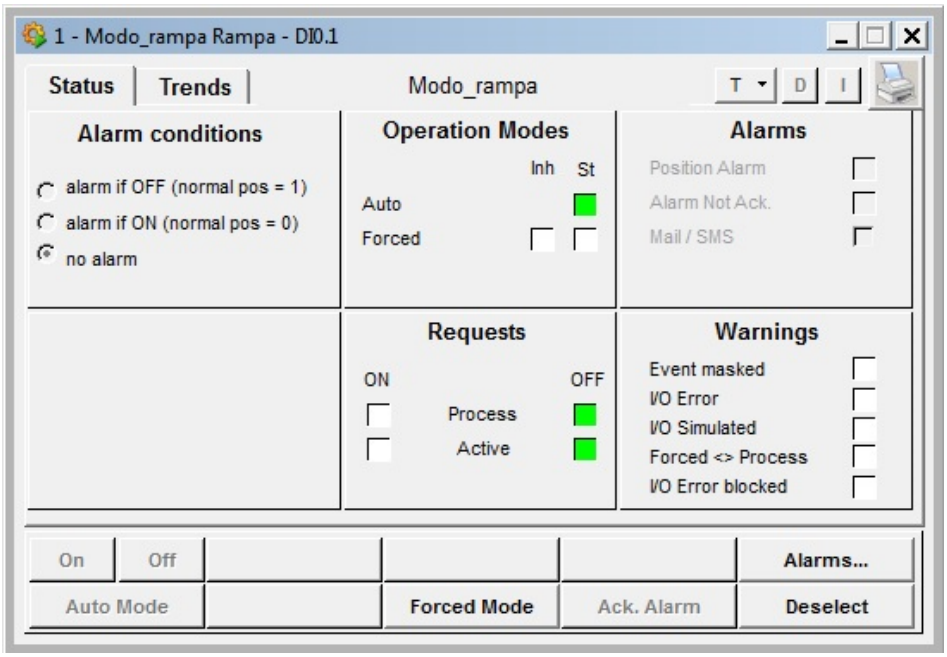


Figura 63: Menú de un bloque DI.

Como ya se ha mencionado antes se debe tener en cuenta que para la modificación de variables el bloque debe estar seleccionado y que por defecto estará en modo automático y en este caso apagado. Este último punto debe estar muy presente a la hora del forzado de variables, si por ejemplo se fuerza una entrada encendiéndola, en el momento en que se vuelva a seleccionar el modo automático la entrada volverá a su estado de reposo, apagada.

Esto ocurre en todos los bloques, por eso es importante ser cauto a la hora del forzado de variables, ya que se podrían modificar los estados seguros del bloque. Como se comentó en el punto del código asociado al PID este es uno de los motivos para no modificar variables manuales o forzadas en los programas.

Una vez mencionado el aspecto y funcionamiento de los objetos UNICOS en PVSS, se puede proceder a la explicación del funcionamiento del SCADA diseñado para supervisar el proceso diseñado.

Como se ha mencionado durante el documento, en busca de que el funcionamiento de la planta se asemejase al de una planta industrial se ha decido incluir un Grafcet de funcionamiento, demostrando también de esta forma la customización de la solución generada por UNICOS.

En la figura 64 puede verse el panel desde donde se supervisa todo el proceso. En esta figura puede ver además resaltadas en colores los *Widgets* que controlan el proceso, y que será detallada más adelante.

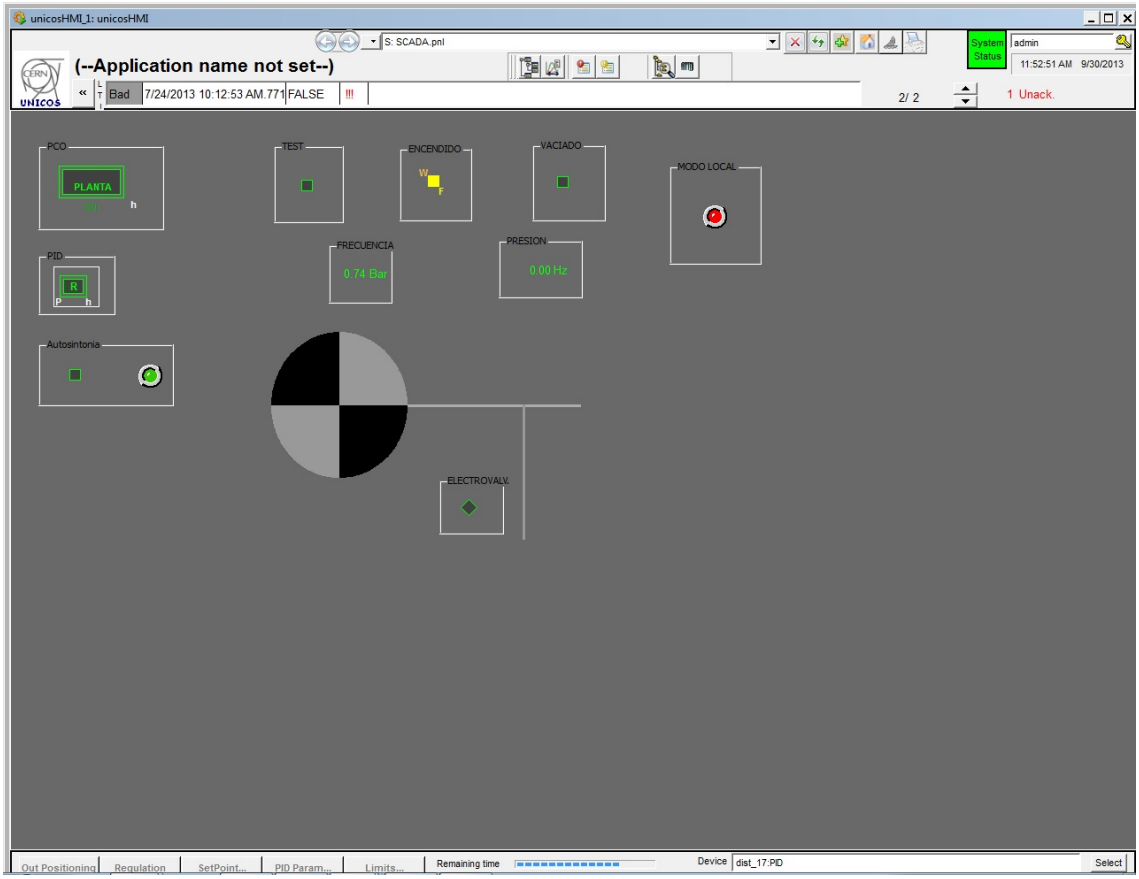


Figura 64: Panel de control PVSS.

Como se vio en los puntos anteriores del presente documento se cuenta con 4 modos de funcionamiento, STOP, TEST, NORMAL, VACIADO y un quinto modo implícito durante todo el proceso, MODO LOCAL.

Se debe empezar hablando de este último modo, el Modo LOCAL, que detiene todas las acciones del SCADA y se activa directamente desde la planta, se ha colocado un LED a modo de aviso en el panel principal para avisar al usuario de que la planta está siendo modificada y que no va a poder realizar ninguna acción hasta que esta luz roja se apague. En el caso de que este modo se hubiera asociado a un *Interlock* el aviso aparecería en la barra superior. Esto puede verse reflejado en la figura 65.

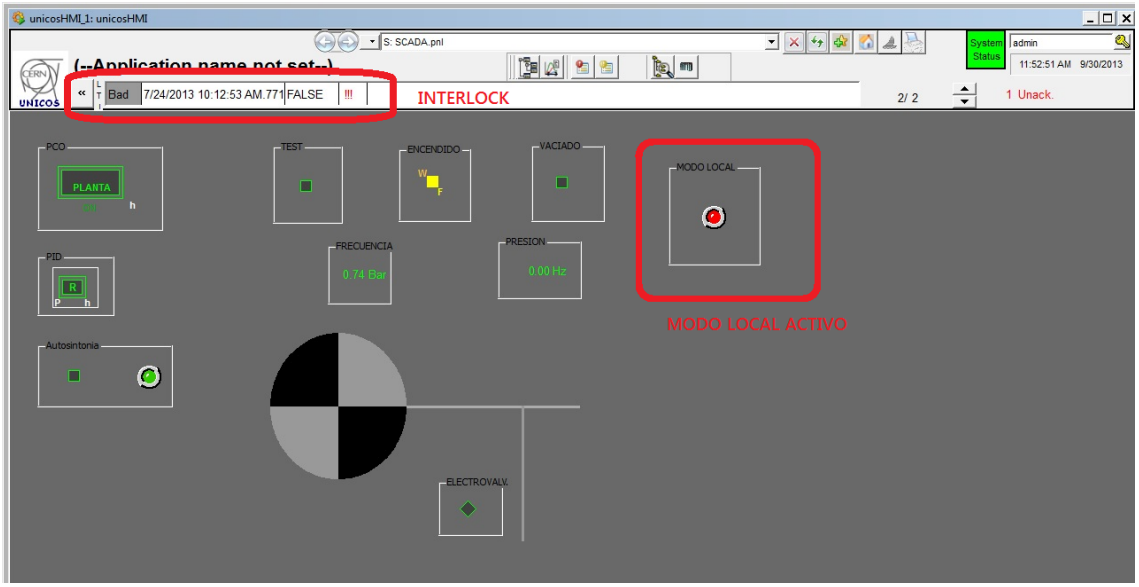


Figura 64: Activación del modo local.

Para la comprensión del funcionamiento del panel creado se debe mencionar el Grafcet de funcionamiento, el cambio de estados en este diagrama, como ya se vio anteriormente está controlado a través de entradas digitales, que deben ser forzadas por el usuario para que el cambio de estados se haga posible. En la figura 65 puede verse la zona del panel destinada a estas entradas digitales.

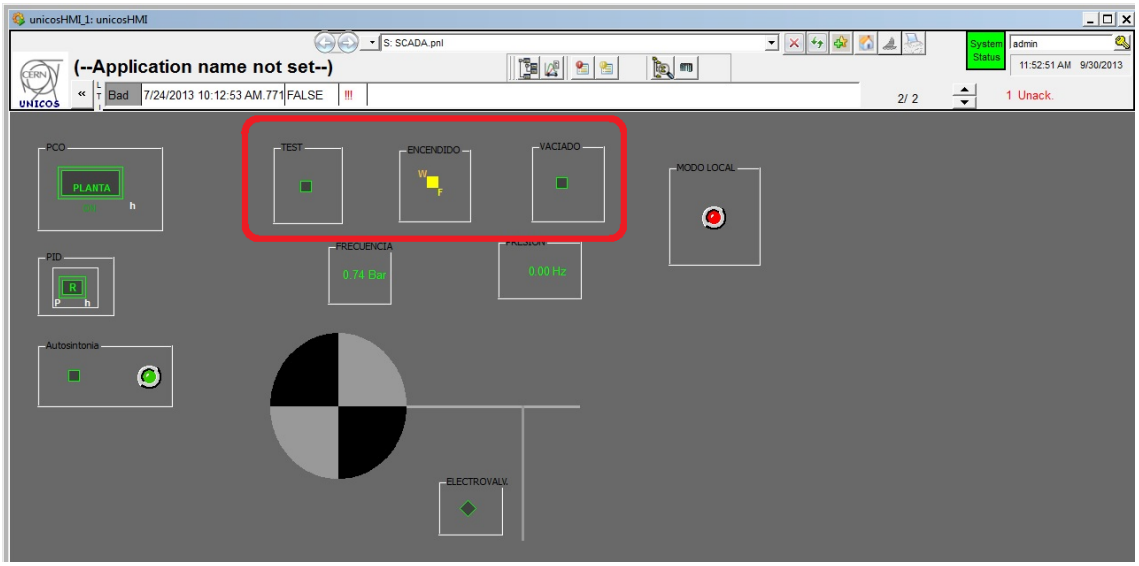


Figura 65: Entradas digitales del proceso.

Pueden verse tres entradas, la primera controla la entrada en el modo test, la seguda en el modo normal y la tercera marca el final del proceso.

Para hacer funcionar la planta se debe tener en siempre en mente el Grafcet de funcionamiento que puede verse en la figura 66. De esta forma se conseguirá un rendimiento óptimo del proceso.

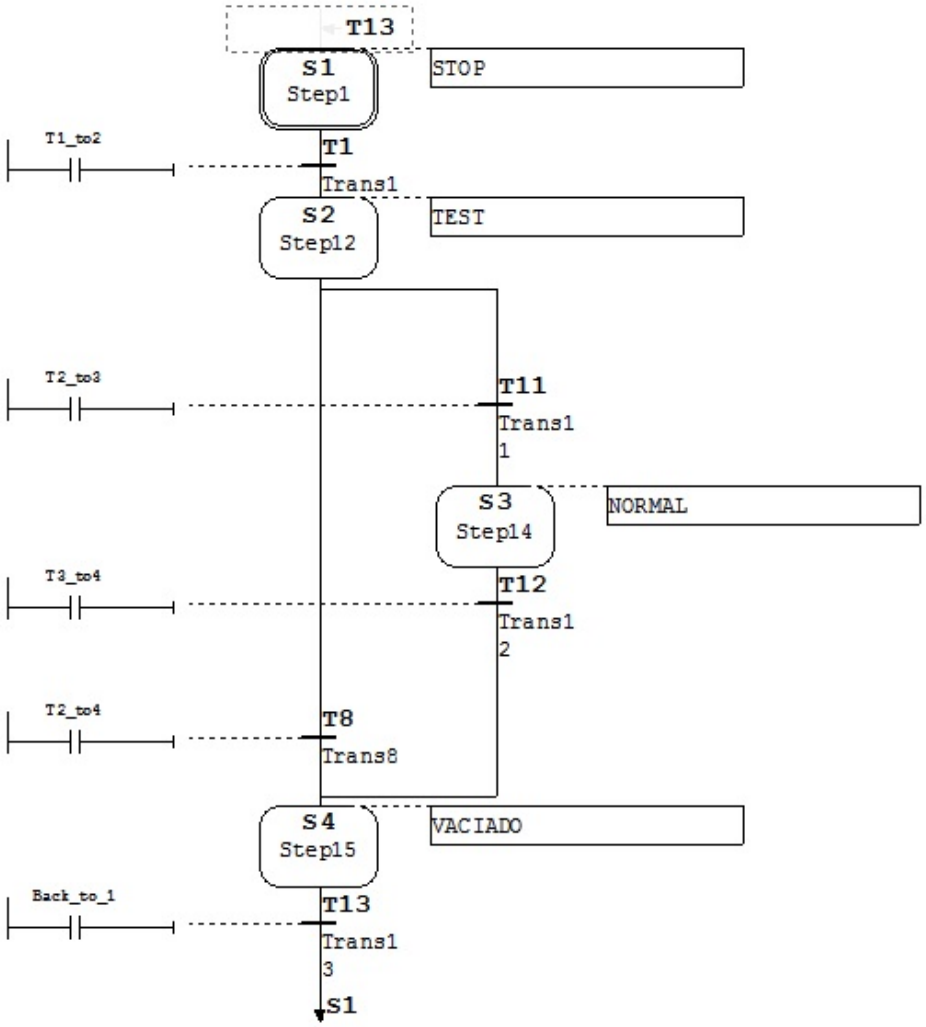


Figura 66: Grafcet de funcionamiento

Como puede observarse en este Grafcet siempre que se inicie el proceso deberá llevarse a cabo un test de funcionamiento, que proporciona una rampa de frecuencias. Este proceso se lleva a cabo forzando durante un instante la entrada digital destinada a este modo. Para ello se debe desplegar el menú de esta entrada forzarla a encendido y devolverla a modo automático. El la figura 67 puede verse este proceso numerado, además de la rampa de frecuencias obtenida.

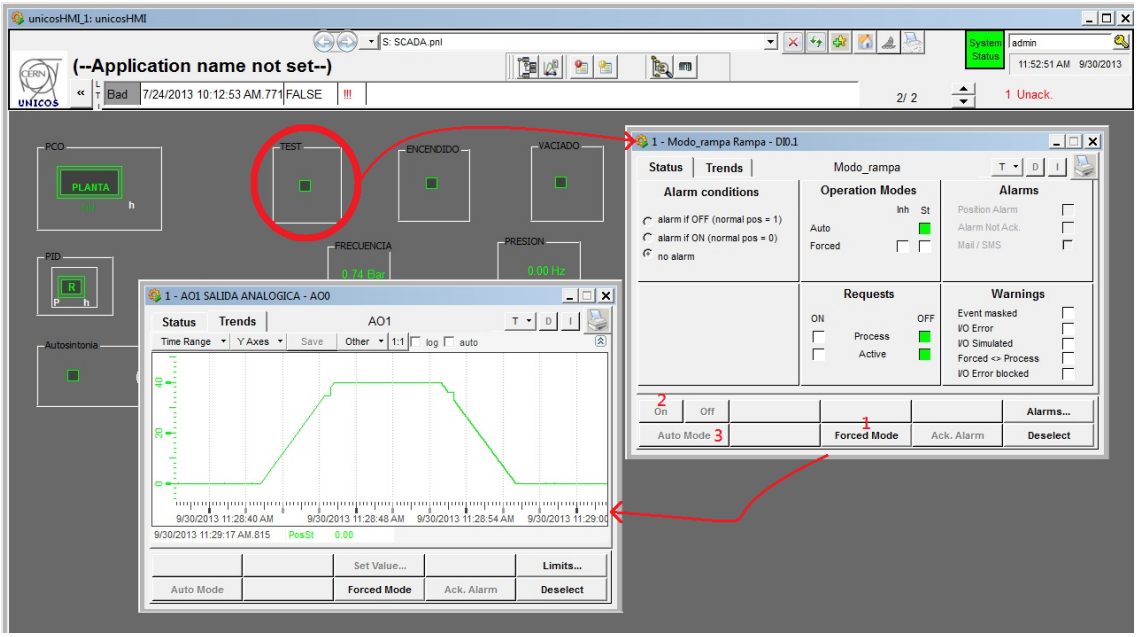


Figura 67: Modo TEST.

En este punto de funcionamiento pueden darse dos opciones, que se desee pasar a modo normal para regular la salida del sistema, forzar variables, etc., como se verá más adelante o que no se desee realizar ninguna opción, por lo que el proceso se pondría en modo vaciado.

Este último modo, el modo vaciado, abre la electroválvula para que el sistema se vacíe rápidamente, cuando el usuario desee poner de nuevo el sistema en reposo debe forzar la entrada digital destinada a esta acción, y la electroválvula se cerrará, colocando al proceso en el modo STOP. Al igual que el modo test la entrada debe ser forzada durante un instante.

En la figura 68 se puede ver como la electroválvula está abierta (Verde) y la salida que se debería forzar (Rojo) siguiendo el proceso descrito en el modo TEST para devolver el sistema al reposo.

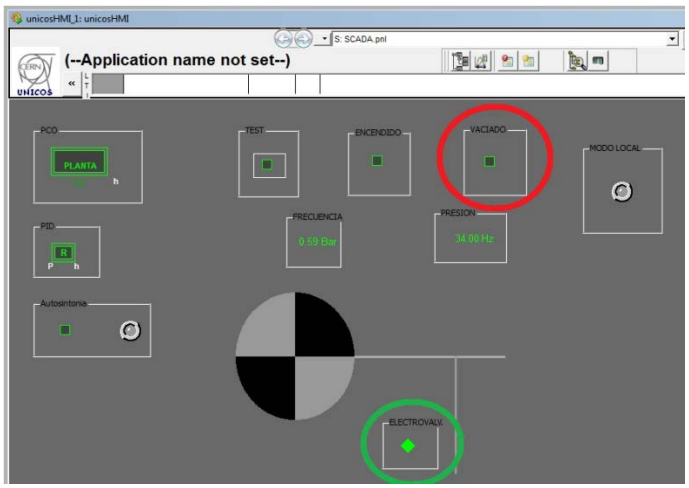


Figura 68: Modo VACIADO.

En el caso en el que se desease poner el sistema en modo normal, para controlar por completo su funcionamiento, la entrada digital destinada a esta tarea debe estar forzada antes de que la construcción de la rampa en el modo TEST termine, y quedarse así hasta que se desee apagar el sistema. En este punto se desactivará y el sistema pasará a modo vaciado. Mientras la entrada esta forzada debe aparecer en color amarillo como puede verse la figura 69.

Figura 69: Entrada Digital forzada.

Este modo, el modo normal, es el que más características posee, ya que aquí es donde se lleva a cabo todo el control de la planta, forzando variables, regulando la salida, etc. En este modo TODAS las características de cada uno de los objetos pueden ser manipuladas. A continuación se detalla el funcionamiento de la opción que va a ser la más utilizada, la regulación de la salida.

En el momento en el que se entra en el modo normal el proceso no realiza ningún tipo de acción, de esta forma se permite al usuario que escoja que función utilizar. Para comenzar a regular el sistema se debe abrir el menú del PID y en primer lugar ponerle en modo manual, y seleccionar la opción de regulación, como puede verse enumerado en la figura 70. En esta figura se pueden ver resaltadas dos opciones, *SetPoint* y *PID Param*, Son las destinadas a la modificación del punto del trabajo del sistema y de los valores del PID respectivamente. Es aconsejable modificar estos valores una vez puesto el modo manual, y antes de comenzar a regular para evitar errores posteriores

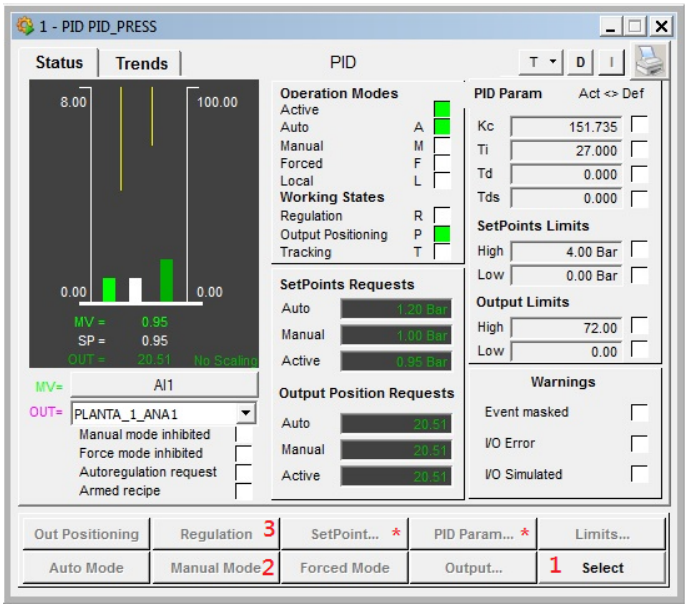


Figura 70: Regulación en modo normal.

En este momento el PID comenzará a regular la salida del sistema. Si esta salida no es estable o el ajuste no es el deseado el usuario podrá ejecutar la autosintonía únicamente forzando la entrada digital destinada a esta tarea y dejándola así hasta que el LED de su derecha se encienda, marcando así el fin del proceso. En la figura 71 puede verse la salida forzada mientras se construye el tren de pulsos para la sintonización.

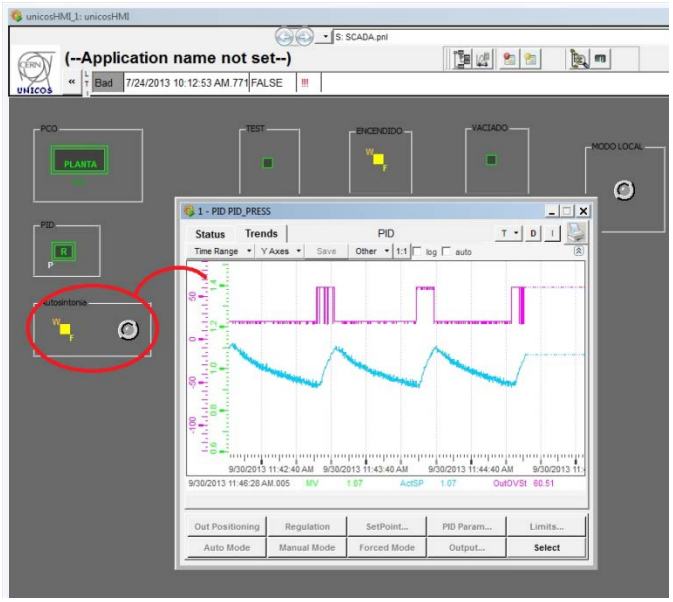


Figura 71: Inicio de Autosintonía.

Cuando el LED se encienda, la autosintonía ha finalizado y el PID ya tiene los valores correctos, entonces el usuario debe apagar la entrada digital de la autosintonía, y poner de nuevo el regulador PID a funcionar, como se ve en la figura 72.

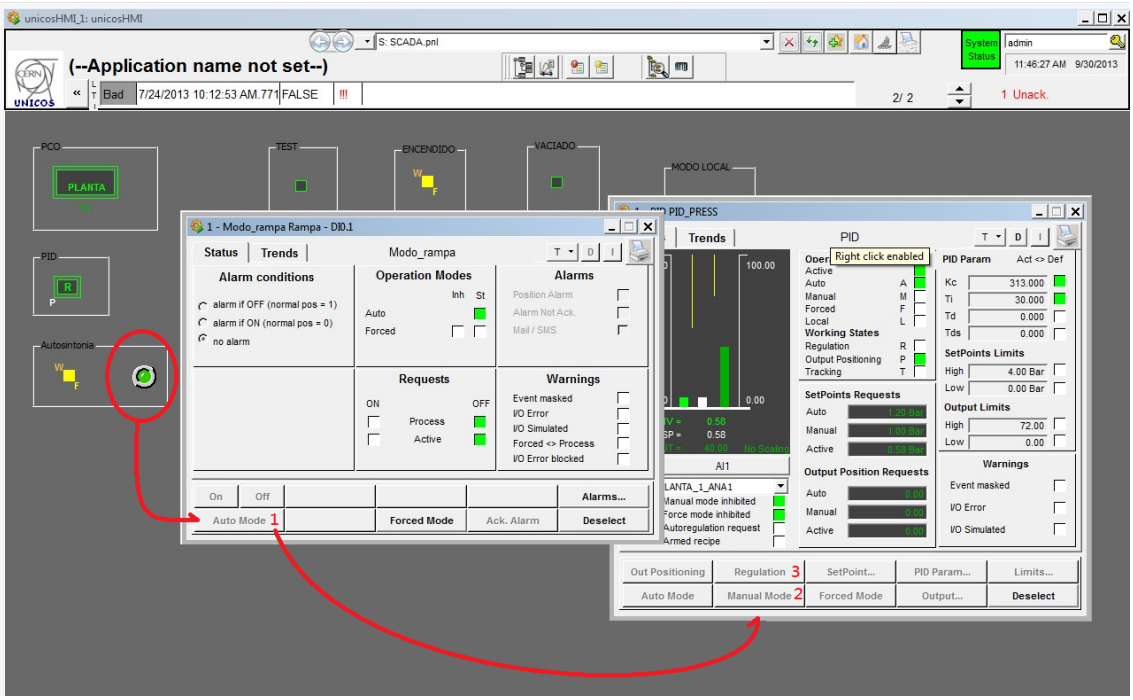


Figura 72: Inicio regulación con Autosintonía.

Capítulo 6:

TRABAJO FUTURO

6 Trabajo futuro

El proyecto descrito en esta memoria ha servido para asentar las bases en conocimiento y comprensión del software *framework* UNICOS-CERN, que tanto desde el punto de vista de la infraestructura de clases básicas que brinda de forma automática, como de las posibilidades de adaptación a la aplicación concreta de que se trate, tenía precisamente como uno de sus objetivos clave, el de servir de base para facilitar la incorporación posterior de diversos algoritmos de automatización y control.

Este proyecto ha tomado una de las muchas direcciones posibles en el campo de *framework* UNICOS-CERN, esta dirección ha sido explicada con detalle anteriormente y en este capítulo se procede a la explicación de una continuación futura del mismo proyecto o de otros caminos alternativos que puede tomar este software.

Para la continuación futura de este proyecto hay diferentes formas de ampliación, de mejora e incorporación mediante una lógica y unos dispositivos más eficaces y completos. Específicamente, en lo que se refiere a la aplicación concreta aquí expuesta sobre el ajuste de PID, existen muchas oportunidades de mejora u otros sistemas implantables de ajuste. Unos buenos pasos a seguir para continuar en la mejora son:

- Se debe realizar una comprobación más extensa de la robustez del algoritmo implementado.
- La implantación de la lógica en el autómata programable (PLC) tiene sus inconvenientes ya que como se conoce la memoria es un recurso muy escaso en un autómata programable y un prerrequisito para la explotación industrial del algoritmo sería la solución de este problema. En este caso, el método utilizado para el ajuste del PID es el Método del relé, y no condiciona la escasa memoria del PLC para su implantación. Como se ha dicho y en este caso no se requiere de una gran cantidad de memoria pero en cualquier caso sigue existiendo el problema de fondo y en este sentido, podrían existir varias alternativas:

    - La implantación de la lógica relativa al PID y a los demás bloques relacionados con la hoja de especificaciones al nivel de la capa supervisión y no de la de control. Esta parece la solución obvia, teniendo en cuenta además el carácter específico de esta aplicación concreta. De ser implementada a nivel de SCADA, la abundancia en recursos de las plataformas informáticas donde se despliegan estos sistemas de supervisión, eliminaría la relevancia del problema de la memoria.

- Se puede implantar varios sistemas de auto-sintonia PID y también de auto-diagnóstico PID para que el proyecto pueda tener una lógica más completa y poder mejorar la regulación de la planta, aunque si la lógica va incluida en la capa de control aparecería el problema de memoria ya que los auto-diagnósticos requieren de un mayor espacio por que necesitan almacenar datos y compararlos, se podría conservar el algoritmo dentro del PLC si los auto-diagnósticos ahorran memoria compartiendo los vectores. Una solución como está requeriría de cierto mecanismo de arbitraje entre los diferentes lazos de control.

Existen otros caminos alternativos al propuesto anteriormente, los cuales exigirán otros requisitos y nuevas características, pero manteniendo en común el diseño de la estructura UNICOS. El proceso de creación de un proyecto es el mismo y difiere de este en que se puede tomar otros caminos que nos proporcionen una manera diferente de obtener los sistemas de control y que a continuación se detalla:

- En este proyecto no se usan todos los objetos relacionados con el software *framework* UNICOS-CERN, por lo que una alternativa será la de incorporar nuevos elementos a la hoja de especificaciones Excel para ampliar el conocimiento, el rendimiento y la potencia de este software. Esto requiere una ampliación o una nueva instrumentación donde los dispositivos y elementos de control sean más variados, potentes y posean una función diferente de los actuales..

- La incursión de lógica se puede hacer de maneras diferentes, esto puede proporcionar una solución de errores en determinados casos y así poder profundizar más en el software *framework* UNICOS-CERN, las formas de incluir lógica son:

  - Capa de control, la lógica va incluida en el PLC, esta lógica se incluye en las fuentes generadas por el UAB desde el software de propio PLC (Siemens SIMATIC STEP 7, este método es el utilizado en este proyecto.

  - Capa de supervisión, este proceso soluciona los problemas de espacio de memoria limitado que imponen los PLC utilizando los recursos del ordenador que será donde se implante la capa de supervisión, la incursión de esta lógica se lleva a cabo en el SCADA desde el programa PVSS.

  - Otra forma de incluir lógica es mediante unos archivos de extensión *python* , los cuales son incluidos en la hoja de especificaciones Excel, esta técnica requiere de un conocimiento más extenso por que se tendría que programar y luego incluirlo, aunque proporciona más ventajas

respecto de las otras dos, como la eliminación del problema en la memoria, la utilización de la misma lógica para elementos comunes en la planta y es más rápida y flexible a la hora de incluirla.

- *Codesys*, *framework* UNICOS-CERN proporciona otros programas para generar un entorno de desarrollo para la programación de controladores conforme con el estándar industrial. Codesys puede ser útil a la hora de creación de Sistemas de Desarrollo de Controladores, pero esta línea se aleja más de lo desarrollado por que difiere en el software.

- Otra línea de actuación sería la de disponer el algoritmo PID de ajuste o auto-diagnóstico perfectamente validado y con las optimizaciones antes mencionadas, se puede pensar en hacer más estrecha su integración con el *framework* UNICOS-CERN utilizando las posibles vías mencionadas en la memoria: ya sea definiendo un *script* en *Python* que automatice la inserción del código necesario o incluso la creación de un nuevo *plugin* en JAVA para extender la funcionalidad del generador de aplicaciones (UAB).

- En este PFC se ha estudiado la posibilidad de la implementación de un método de autodiagnóstico de bloques PID. Estos métodos a partir del análisis del error, mediante una serie de operaciones matemáticas, establecen la calidad de la sintonía de un PID. La incursión de uno de estos métodos en este PFC mejoraría considerablemente su funcionamiento, haciendo desaparecer la necesidad del disparo manual de código de autosintonía, de tal forma que cuando el diagnostico establezca que la sintonía no es correcta, el código de diagnóstico comenzaría a ejecutarse.

  En concreto se ha estudiado incluir un método desarrollado en el departamento ISA de la Universidad de Valladolid, denominado *Performance monitoring of industrial controllers based on the predictability of controller behavior* [16]. Este método se basa en determinar la predictibilidad del error de la planta a partir de la aplicación de los métodos de los mínimos cuadrados. La implantación de este método a nivel del PLC ha sido imposible debido que las operaciones matemáticas necesarias para el cálculo de mínimos cuadrados requieren una gran cantidad de memoria, algo de lo que carecen los PLC. En este punto se establecen dos líneas de trabajo para posibles versiones futuras:

  - Implantación del método a nivel del SCADA: Esta posible solución hace desaparecer inmediatamente los problemas asociados al uso de memoria en el proceso de cálculo de los mínimos cuadrados, ya que cualquier plataforma informática convencional es capaz de desarrollar esto sin ningún tipo de problema. Los inconvenientes aparecen a la hora de la adaptación de esta solución en el marco de UNICOS.
  -

- Adaptación del método al algoritmo de mínimos cuadrados recursivos (RLS) como puede verse en el documento: Aplicación de Mínimos Cuadrados Recursivos al método de diagnóstico de lazos de control, de Rogelio Mazaeda, incluido en los anexos del presente documento. Este método permite obtener los mismos valores que con los algoritmos clásicos de mínimos cuadrados, con la ventaja de que el uso de memoria es mucho menor. Solucionando de esta forma los problemas citados y permitiendo la implementación en el PLC.

Capítulo 7:

CONCLUSIONES

7 Conclusiones

Los objetivos de este proyecto han sido debidamente cumplidos. El esfuerzo desarrollado en su elaboración buscaba cumplir con el requerimiento académico de aplicar una parte importante de los conocimientos obtenidos durante la carrera en la realización de un proyecto con un alto componente práctico y al mismo tiempo, desarrollar las habilidades en la adquisición independiente de conocimientos, competencia que es cada vez más necesaria para enfrentar un mundo laboral muy cambiante, fundamentalmente en las especialidades técnicas.

En este sentido, se han reafirmado y profundizado nuestros conocimientos al realizar satisfactoriamente actividades tales como:

- Diseño e implantación de una planta de laboratorio. Esta tarea incluye el estudio de la planta a controlar (en este caso el compresor de aire), así como de los elementos de control e instrumentación a partir manuales, catálogos y otras fuentes de información técnica.

- Estudio en profundidad de un autómata programable como el SIEMENS 315-2PN/DP, uno de los más difundidos en la industria. Aquí se incluye obviamente, el aprendizaje de sus lenguajes de tipo SIMATIC S7. Este grupo de lenguajes, aunque presentan especificidades propias, son en gran medida compatibles con el estándar IEC 61131-3, de manera que el esfuerzo invertido puede ser aprovechado a la hora de enfrentar otros autómatas programables compatibles con dicho estándar.

- Se ha aprendido a trabajar con la herramienta de tipo SCADA PVSS. Este sistema puede ser configurado para realizar loa supervisión de grandes sistemas de control industrial, que estén distribuidos geográficamente conectados a través de la red informática. El proyecto de la planta de laboratorio no explota todas las posibilidades de una herramienta de este grado de complejidad, pero constituye una experiencia práctica útil, fácilmente generalizable a herramientas similares.

- Se han consolidado los conocimientos relativos a los controladores de tipo PID, que como se conoce constituye el tipo de controlador más extendido en la industria de procesos. Se han aplicado herramientas de sintonía tradicionales, se ha simulado su funcionamiento utilizado herramientas basadas en MATLAB y finalmente se ha implantado en la planta real alojados en un PLC. Pero no solo se han aplicado algoritmos establecidos en la práctica industrial, también se han explorado soluciones novedosas extraídas de la amplia literatura científica de la especialidad del control automático, en este caso específico para la sintonía automática de lazos de control.

- Se ha estudiado el *framework* UNICOS-CERN. Esta infraestructura está destinada al diseño de sistemas de control y su posterior generación y despliegue, que se realiza de forma automática, en los equipos programables industriales situados en las capas de control y supervisión de la pirámide de control. El estudio detallado de la arquitectura que propone UNICOS nos ha permitido, en buena medida, aprender buenas prácticas que se han formado a partir de años de experiencia del grupo de automatización del CERN y que están, de alguna forma, "codificadas" en dicho *framework.*

- El trabajo con UNICOS-CERN requiere que la solución propuesta sea adaptada al caso específico de que se trate. En este caso, esa adaptación ha requerido la compresión de la jerarquía de objetos generada automáticamente por UNICOS con el objetivo de insertar, en los puntos adecuados previstos al efecto, el código necesario. Este código ha sido escrito en lenguaje del PLC, para adaptar los objetos desplegados en la capa de control. Los objetos generados automáticamente para la capa de supervisión fueron utilizados en la configuración del SCADA.

Pero los resultados logrados no se limitan al aprovechamiento académico de los autores. El estudio UNICOS-CERN que con este proyecto se ha iniciado, servirá de base a futuros desarrollos que se acometerán en el Departamento de Ingeniería de Sistemas y Automática y que utilizarán la mencionada infraestructura en el marco de cooperación establecido con el grupo de automatización del CERN.

Por otra parte, la implementación práctica, a nivel de autómata programable, de un algoritmo de autosintonía de los lazos de control, puede ser útil en sí misma.

Capítulo 8:

LISTA DE FIGURAS

4 Lista de figuras

Capítulo 9:

REFERENCIAS

9 Referencias

[1] Norma ANSI/ISA95

- http://www.isa.org

[2] Estándar IEC 61131-3

- http://www.infoplc.net/files/documentacion/estandar_programacion/infoPLC_net_Intro_estandar_IEC_61131-3.pdf

[3] Estándar IEC 61499

- http://www.holobloc.com/papers/iec61499/overview.htm

[4] Pagina web del grupo EN-ICE.

- https://j2eeps.cern.ch/wikis/display/EN/UNICOS

[5] S7-300 CPU 31xC y CPU 31x: Datos técnicos

- http://www.swe.siemens.com/spain/web/es/industry/automatizacion/simatic/controladores/Documents/S7300ManualProducto.pdf

[6] S7-300 SM331;AI 8x12 Bit Getting Started

- http://cache.automation.siemens.com/dnl/Dk/Dk1MDkxNQAA_17473828_HB/s7300_sm331_ai_8x12_bit_getting_started_en-US_en-US.pdf

[7] Statement list (SCL) for S7-300 and S7-400 Programming

- https://www.automation.siemens.com/doconweb/pdf/SINUMERIK_SINAMICS_03_2013_E/S7_AWL.pdf?p=1

[8] Ladder Logic (LAD) for S7-300 and S7-400 Programming

- https://www.automation.siemens.com/doconweb/pdf/SINUMERIK_SINAMICS_03_2013_E/S7_KOP.pdf?p=1

[9] S7 Graph V5.2 for S7-300/400 Programming sequential control systems

- http://w3.usa.siemens.com/us/internet-dms/ia/AutomationComm/Automation/Docs/Graph7_e.pdf

[10] VS mini J7 Variador compacto de empleo general. Manuel de Usuario

- http://www.valtek.es/ftp/Omron/Variadores/J7/I63E-ES-01+J7AZ+UsersManual.pdf

[11] Manual técnico UCP

- http://j2eeps.cern.ch/wikis/display/EN/UNICOS

[12] Manual técnico generador UAB

- https://j2eeps.cern.ch/wikis/display/UCPC15/UNICOS-CPC+Documentation

[13] Manual técnico S7 en UNICOS

- https://edms.cern.ch/file/1228441/1.5.0/Procedure_S7-UCPC_Application.pdf

[14] Manual técnico PVSS en UNICOS

- https://edms.cern.ch/file/1228441/1.5.0/Procedure_PVSS-UCPC_Application.pdf

[15] Desarrollo de programas PLC multiplataforma mediante el *framework* UNICOS de aplicación en banco de prácticas del LHC (*Large Hadron Collider*). Silvia María Izquierdo Rosas

[15] *Advanced PID Control . Karl J. Astrom y Tore Hagglund Editado por ISA: The instrumation, Systems, and Automation Society*

[16] *Performance monitoring of industrial controllers based on the predictability of controller behavior.* Rachid A. Ghraizi, Ernesto Martinez, Cesar de Prada, Francisco Cifuentes, Jose Luis Martinez.

Capítulo 10:

ANEXOS

Uso de los mínimos cuadrados recursivos en algoritmo de diagnóstico automático de las prestaciones de un lazo de control

1-Introducción

El PID constituye el algoritmo de control más difundido. En cualquier industria de procesos, los lazos de control regulado mediante PID se cuentan por centenares. Típicamente, una proporción muy alta de estos controladores están mal sintonizados, incluso existen referencias que sugieren que muchos controladores son desplegados en la fábrica sin modificar los parámetros que tiene por defecto. La robustez inherente de este algoritmo de control, combinado con las exigencias específicas del proceso a controlar, hace que, en muchos casos, una sintonía imperfecta no tenga consecuencias importantes. En otras ocasiones, sin embargo, un lazo de control mal controlado, que sea demasiado lento a la hora de rechazar perturbaciones o de seguir la referencia, o que presente oscilaciones o errores de estado estacionario, puede conllevar consecuencias económicas, que agregadas a nivel de toda la fábrica, pueden llegar a ser significativas.

Por otra parte, debido a la gran cantidad de PIDs existentes, la tarea de diagnosticar los lazos que funcionan incorrectamente primero, para posteriormente re-sintonizarlos, resulta muy exigente para el personal técnico de la fábrica. Es por eso que existe una motivación muy bien justificada para avanzar en la automatización de ambas tareas.

El objetivo de este informe técnico es modificar el algoritmo de diagnóstico descrito en Ghraizi *et al*, (2007) de forma que sea factible su implementación con los recursos computacionales y de memoria de los autómatas programables. En la sección 2 se discutirán los antecedentes relacionados con la implementación actual del índice de comportamiento, para luego discutir en 3 las modificaciones relacionadas con la aplicación del algoritmo de mínimos cuadrados recursivos, más apropiados para su implementación en línea. En la sección 4 se brinda el código en MATLAB de una posible implementación.

2-Antecedentes teóricos

El diagnostico automático de las prestaciones de un lazo de control ha sido tratado en múltiples ocasiones (Aström y Hägglund, 2005) pero quizá el procedimiento más conocido es el que se realiza a partir del cálculo del llamado índice de Harris.

Este índice pone en relación, la varianza actual de la señal de salida del lazo cerrado con la varianza mínima teórica que se obtendría si la variable de proceso estuviese controlada por el así llamado *controlador de varianza mínima*.

Pero se conoce que un PID no puede, en general, brindar una sintonía de varianza mínima. Por este motivo, el diagnosticar las prestaciones del lazo a partir del índice de Harris tiene la desventaja de que resulta difícil cuantificar, para cada lazo de control concreto, lo que constituye un valor que indica un comportamiento razonable, de aquellos otros que indican una mala sintonía.

En la reciente contribución de Ghraizi *et al*, (2007), realizada en el departamento ISA de nuestra Universidad, se plantea solucionar esta desventaja, al presentar un nuevo índice que a diferencia del de Harris, no brinde la comparación con un lazo teórico difícilmente realizable con un PID, sino que se base en el grado de predictibilidad del error.

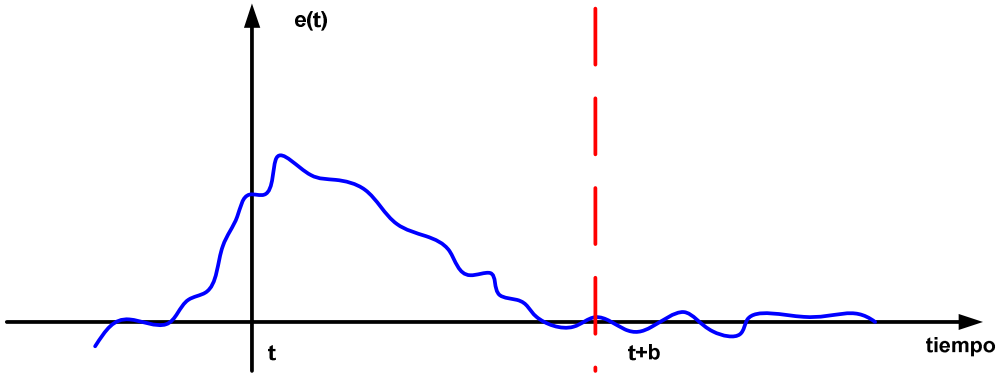


Figura 1. Diagnóstico a partir de la predictibilidad del error del lazo.

La idea de este índice de prestaciones se puede explicar con arreglo a la figura 1. La justificación surge a partir de comprender que para un lazo de control bien sintonizado, el error de control actual, o sea la diferencia entre la referencia deseada y la variable de proceso, debe ser independiente, desde el punto de vista estadístico, del error que existió ***b*** intervalos de muestreo en el pasado. Explicado de otra forma: un PID bien ajustado debe ser capaz de eliminar las perturbaciones o de seguir los cambios de referencia que se produzcan en el instante ***t***, de manera que los errores correspondientes en ***t+b*** no estén correlacionados con los errores que ocurrieron hasta el instante ***t***. En un controlador mal sintonizado, si habrá cierto grado de correlación más allá del horizonte ***b***, puesto que el controlador no ha sido capaz, durante ese tiempo, de anular los efectos de los cambios a los que se enfrentó en el pasado. Entre los instantes ***t*** y ***t+b***, en todos los casos, si habrá correlación estadística entre los errores, que es debida, evidentemente, a las acciones de control realizadas por el PID para corregir el error.

El parámetro ***b*** se escogerá, teniendo en cuenta fundamentalmente la dinámica del proceso subyacente a controlar pero también aspectos tales como la energía de control que se quiera invertir en cada caso concreto.

A continuación, una explicación más detallada del cálculo del índice. El error de control es evidentemente la diferencia entre el valor de referencia (*w(t)*) y la variable de proceso (*y(t)*):

$$e(t) = w(t) - y(t) \qquad (1)$$

Mediante *e(t)* haremos referencia al error real que efectivamente se mide, el cual se debe comparar con el error predicho, al que se le designará mediante $\hat{e}(t)$. El residuo *r(t)* sería la diferencia entre ambos, de acuerdo a:

$$r(t) = e(t) - \hat{e}(t) \qquad (2)$$

Se requiere ahora estimar errores futuros a partir de los actuales y para ello se utilizará un modelo de regresión lineal de orden *m* con *m+1* parámetros *[a_0, a_1 .. a_m]* del tipo:

$$\hat{e}(t+b) = a_0 + a_1 e(t) + a_2 e(t-1) + a_3 e(t-2) + \cdots + a_m e(t-m+1) \quad (3)$$

El ajuste del modelo de regresión se realiza mediante el conocido algoritmo de mínimos cuadrados de acuerdo a:

$$[a_0\,, a_1 \ldots \ldots a_m]^T = (X^T X)^{-1} X^T Y \quad (4)$$

Donde la salida a predecir es el vector *Y* conformado por *n* valores del error del lazo, tomados a partir del horizonte de predicción *b*:

$$Y = [e(m+b)\ \ e(m+b+1) \ldots\ \ e(n)]^T \quad (5)$$

Mientras *X* es la matriz conformada con los regresores, definida según se describe a continuación:

$$X = \begin{pmatrix} 1 & e(1) & e(2) & \cdots & e(m) \\ 1 & e(2) & e(3) & & e(m+1) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e(n-b-m+1) & \cdots & \cdots & e(n-b) \end{pmatrix} \quad (6)$$

El índice de prestación del lazo se determina entonces a partir de la expresión:

$$PI = \frac{\sigma_r^2}{\sigma_e^2} \quad (7)$$

Los términos σ_e^2 y σ_r^2 se corresponden con las varianzas del error y de los residuos respectivamente, determinadas de acuerdo a:

$$\sigma_r^2 = \frac{1}{n-1}\sum_{i=1}^{n}(r(i) - \bar{r})^2 \quad (8)$$

$$\sigma_e^2 = \frac{1}{n-1}\sum_{i=1}^{n}(e(i) - \bar{e})^2 \quad (9)$$

El índice PI nos informará de cuán buena es la sintonía del PID, sus valores estarán comprendidos entre 0 y 1. Cuanto más similares serán las varianzas mejor será la sintonía, puesto que esto significará que el modelo ajustado es incapaz de predecir los errores más allá de un horizonte de *b* intervalos de muestreo, y estos será únicamente atribuibles a ruido aleatorio. Por otro lado, un PI cercano a cero, implicará que la varianza de los residuos es también cercana a cero y serán por tanto perfectamente predecibles, cuando no deberían serlo, implicando entonces una mala sintonía. Desde el punto de vista práctico, solo restaría definir un umbral específico para distinguir las buenas de las malas prestaciones del lazo de control.

La implementación del método de diagnóstico descrito implica la elección de los parámetros ***b***, ***m*** y ***n***. Los criterios para la elección de *b* ya se han comentado anteriormente. El orden del modelo de predicción ***m*** debe ser elegido de acuerdo a la dinámica que se desee del lazo cerrado, evitando valores demasiado altos que implicarían el riesgo de sobre-entrenamiento. Por otra parte, el parámetro ***n*** es el tamaño de los datos que conforman la matriz X a partir del

conjunto de regresores y su elección debe resolver el compromiso entre la sensibilidad a los datos locales (para valores pequeños) y el no mezclar datos heterogéneos, lo que se produciría para valores demasiado grandes.

3-Modificación del índice PI para implementación en autómata programable

En la implementación práctica de un algoritmo como el descrito, existen varias alternativas que dependen, en primer lugar, de la posición específica dentro de la conocida pirámide CIM, donde la misma se pretenda desplegar. Lo más evidente, sería el elegir la capa de supervisión conformada fundamentalmente por sistemas de tipo SCADA. Estos sistemas comerciales brindan diferentes opciones a la hora de adaptar las soluciones estándar a los objetivos específicos para instalación concreta. La solución es creada a partir de un sistema de software configurable, que en muchos casos permite ejecutar código especialmente creado por el usuario, en algún lenguaje propietario o estándar. No es tampoco raro que brinden la posibilidad de enlazar con código externo, creado por ejemplo en lenguaje C. En este último caso, se tendría un gran nivel de control sobre la solución de supervisión creada, puesto que sería trivial el poder, por ejemplo, acceder a bibliotecas matemáticas externas. Por otra parte, el uso de las plataformas informáticas más o menos convencionales que alojan a los sistemas SCADA, elimina cualquier preocupación en cuanto a los recursos computacionales y de memoria necesarios.

A pesar de lo anterior, podrían darse algunos argumentos a favor de desplegar aplicaciones hechas a la medida en el nivel del control de la pirámide CIM, utilizando a ese efecto los dispositivos programables tales como autómatas o PLCs. Se conoce la tradicional reticencia de los tecnólogos y operarios de la industria a la hora, no ya de acoger con entusiasmo, sino de siquiera aceptar, el despliegue de aplicaciones novedosas. Estas prevenciones por parte de la industria, que están en buena medida en el origen de la brecha existente entre la teoría y la práctica del control automático, no dejan de tener su justificación. Desde el punto de vista de la industria, el compromiso entre los beneficios esperables de la implantación de determinada solución nueva y el riesgo económico que implica un probable mal funcionamiento o falta de robustez de la misma, no siempre se decanta a favor de la introducción de mejoras.

En este sentido, la utilización de lenguajes de PLC, que son por diseño, lenguajes de variabilidad reducida, con un conjunto limitado de estructuras computacionales, que están bien estudiadas y recogidas en estándares como IEC 61131-3, aporta un grado extra de fiabilidad que puede inclinar la balanza a favor de la innovación. Desde este punto de vista, las limitaciones de expresividad de los lenguajes y la estructura relativamente simple de los PLC serían, no es ya una desventaja, sino una motivación importante para su utilización.

En cualquier caso, las limitaciones de recursos computacionales del PLC constituyen una barrera a la hora de implementar algoritmos de control más o menos complejos. En este contexto sería de interés el encontrar, en la medida de lo posible, variantes de esos algoritmos que fueran aplicables con los recursos disponibles en estos dispositivos.

Las limitaciones de los PLCs al abordar algunas soluciones más exigentes, se hacen evidentes, por ejemplo, en el caso de una posible implementación del algoritmo de diagnóstico descrito en la sección anterior.

El método descrito requiere, por una parte, del almacenaje de varios vectores y matrices, algunos de dimensiones considerables. Por ejemplo, en la referencia citada se recomienda una longitud de 1000 para el vector de regresores, y entre 20 y 30 para como orden del modelo. Por otra parte, se necesita el cálculo de la inversa de la matriz $(X^T X)$ como parte de la determinación de la pseudo-inversa de la matriz de regresores *X* de acuerdo a $(X^T X)^{-1} X^T$ imprescindible para la implantación de los mínimos cuadrados.

En este caso, una posible solución, casi inmediata, es el apelar a la versión recursiva del algoritmo de mínimos cuadrados. Se trata de ir modificando el modelo lineal actual con los nuevos datos en la medida en que estos estén disponibles, sin necesidad de hacer el cálculo partiendo de cero. Sería necesario utilizar la versión recursiva de los mínimos cuadrados que incluye un factor de olvido exponencial, que hace que no se requiera ajustar toda la historia precedente sino que se atienda únicamente a los últimos datos. La función a minimizar en el ajuste sería entonces:

$$V(\theta,t) = \frac{1}{2}\sum_{i=1}^{t} \lambda^{t-i}\left(y(i) - \varphi^T(i)\theta\right)^2 \qquad (10)$$

Donde λ es el factor de olvido exponencial, que debe estar entre 0 y 1, $\varphi(t) \in \mathbb{R}^{mx1}$ es el vector de regresores a ajustar y $\theta \in \mathbb{R}^{mx1}$ el vector de parámetros del modelo lineal. El problema (10) puede ser resuelto mediante la expresión recursiva (11), que calcula el vector actual que caracteriza al modelos ($\hat{\theta}(t) \in \mathbb{R}^{mx1}$) es obtenido a partir de corregir la estimación anterior ($\hat{\theta}(t-1)$) por un término que es proporcional al tamaño del error de predicción del modelo anterior a la hora de explicar los nuevos datos, utilizando el regresor ($\varphi(t)$) y el valor ($y(t)$) actuales.

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1)) \qquad (11)$$

La constante de proporcional $K(t)$ se determina de acuerdo a:

$$K(t) = P(t)\varphi(t) = P(t-1)\varphi(t)\left[\lambda I + \varphi^T(t)P(t-1)\varphi(t)\right]^{-1} \qquad (12)$$

En (12) $P(t) \in \mathbb{R}^{mxm}$ es la matriz de covarianzas del modelo y se actualiza, también de forma recurrente, como:

$$P(t) = (I - k(t)\varphi^T(t))P(t-1)/\lambda \qquad (13)$$

Obsérvese, que para el caso de una sola salida $y(t) \in \mathbb{R}$, como es el que nos ocupa, en (12) el término a invertir es un escalar.

Queda por definir el problema de la inicialización. En caso de ausencia inicial de conocimiento sobre el problema, lo más prudente sería inicializar a cero el modelo estimado ($\hat{\theta}(t=0) = \mathbf{0}$) y, compatible con lo anterior, asignar un valor alto a la matriz de covarianzas, indicando la falta de confianza en ese modelo inicial y forzando una rápida convergencia hacia valores ajustados de los datos medidos ($P(t=0) = I\sigma$) con σ alto).

La aplicación del algoritmo recursivo recién descrito al cálculo del índice de comportamiento del lazo de control es casi inmediata. Se debe sustituir y(t) por el error del lazo que se mide en el instante actual e(t). El regresor $\varphi(t)$ estará constituido por los n valores anteriores del error que comienzan en e(t-b) hasta e(t-b-m) donde m es la dimensión del modelo lineal y b es el horizonte antes discutido, a partir del cual los errores no estarán correlacionados para un controlador bien ajustado.

Con el uso del algoritmo recursivo, por una parte, desaparece la necesidad de invertir una matriz y por la otra, también se reduce la necesidad de almacenamiento en memoria, un recurso como se sabe, escaso en un PLC. Para comprobar este segundo punto, basta analizar que la matriz de regresión definida en (6) y que pertenecía a $X(t) \in \mathbb{R}^{(n-b-m+1)x\,m+1}$ donde *n* era un número cuyo valor se consideraba en torno a 1000, y que surgía producto de adoptar un método de ajuste por lotes, ya no es necesaria. Se debe recordar que en la descripción original, se discutía que el tamaño de *n* debía ser elegido de forma que no fuera excesivamente sensible a datos recientes pero que tampoco se tornara inútil al mezclar datos tomados en condiciones completamente diferentes. Una función similar la viene a cumplir el parámetro λ, un valor de 1 hace que el modelo elegido dependa de todos los datos obtenidos desde la inicialización del algoritmo, lo cual equivaldría a un valor excesivamente grande (en principio infinito) de *n*. En la medida, en que λ adopta un valor menor que la unidad, el modelo se hace progresivamente más dependiente de los datos recientes. En Aström y Wittenmark (1989) se ofrece la regla aproximada (14), útil a la hora de elegir λ.

$$N = \frac{2}{1-\lambda} \qquad (14)$$

De esta manera, por ejemplo, $\lambda = 0.998$ implica considerar aproximadamente los últimos 1000 pasos a la hora de estimar el modelo.

Implementación en MATLAB del algoritmo recursivo

En la figura 2 se describe el esquema de prueba del algoritmo de diagnóstico recursivo. El algoritmo en sí, está implementado en la S-función llamada diag_lms. En las figuras 3,4 y 5 se recogen fragmentos significativos del código de MATLAB de la función utilizada.

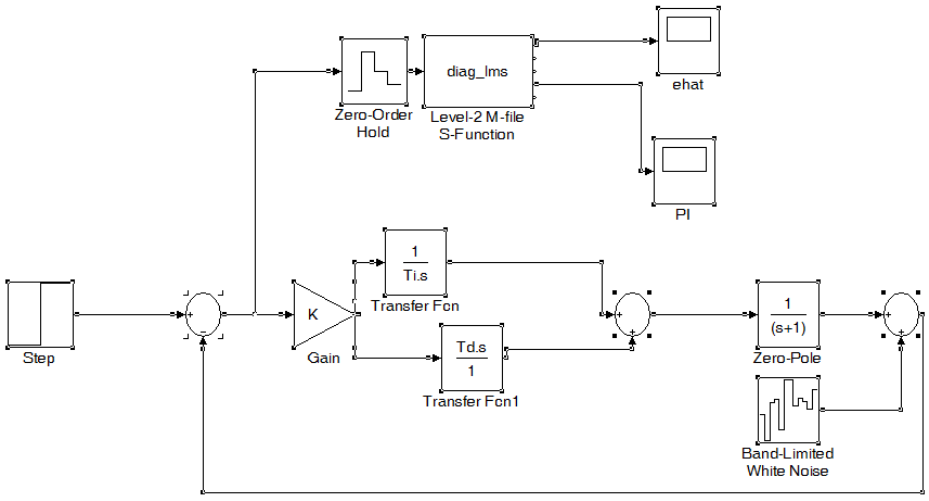


Figura 2. Esquema de prueba del algoritmo de diagnóstico en MATLAB.

```
function Outputs(block)

  lambda = block.RuntimePrm(1).Data;    %factor de olvido
  N  = block.RuntimePrm(2).Data;        %orden del modelo
  b  = block.RuntimePrm(3).Data;        %parámetro b
  total=N+b;
  residuesSum  = block.Dwork(3).Data;
  residuesSquare= block.Dwork(6).Data;
  errorsSquare= block.Dwork(7).Data;
  P=eye(N,N);

  D=block.Dwork(8).Data;
  for i=1:N
    P(i,1:N) =D((i-1)*N+1:(i-1)*N+N);       %Se conforma la matriz P a partir del vector
                                            %proveniente de la iteración
                                            %anterior
  end
```

Figura 3. Código de S-función para recuperar valores de los parámetros

```
et = block.InputPort(1).Data(1);          %Se lee error actual

X = block.Dwork(1).Data;                  %Se lee el regresor de la iteración anterior

H = block.Dwork(2).Data;                  %Se lee el modelo de la iteración anterior


X(2:total) = X(1:total-1);                %Se actualiza el regresor con el nuevo error
X(1)   = et;

Xatras(1:N)=X(1+b:total)                  %Se toman los elmentos del regresor b instantes
                                          %de tiempo en el pasado

ehat      = Xatras'*H  ;                  %Se calcula el error predicho
e      = (et-ehat);                       %se calcula residuo actual

K=P*Xatras/(lambda+Xatras'*P*Xatras)      %Se actualiza matriz de ganancias K


H      = H+K*e                            %Se actualiza recursivamente el modelo
P=(eye(N,N)-K*Xatras')*P/lambda           %Se actualiza matriz P


R(2:total)=R(1:total-1);                 %Se alamacenan los residuos

R(1)=e;
```

Figura 4. Código de S-función que implementa algoritmo recursivo.

```
residuesSum=0;
residuesSquare=0;
errorsSum=0;
errorsSquare=0;
 for i=1:total
  residuesSum=residuesSum+R(i);
  errorsSum=errorsSum+X(i);
 end
 residuesSum=residuesSum/total
 errorsSum=errorsSum/total
 for i=1:total
  residuesSquare=residuesSquare+(R(i)-residuesSum)^2;
  errorsSquare=errorsSquare+(X(i)-errorsSum)^2;
 end
PI=residuesSquare/errorsSquare
```

Figura 5. Código de S-función donde se calcula índice de prestaciones.

Bibliografía

1989. *Adaptive Control*. Aström y Wittenmark. Addison-Wesley, 1989.

TRANSMISORES DE PRESIÓN DE BAJO COSTE
para aplicación general

TPR-14/N .../LP .../HP

Desin Instruments

DESCRIPCION

TPR-14/N para Neumática y Vacío

Transmisor de presión de BAJO COSTE con salida 4/20 mA para aplicaciones en **Neumática y Vacío**, con rangos de 250 mbar a 10 bar

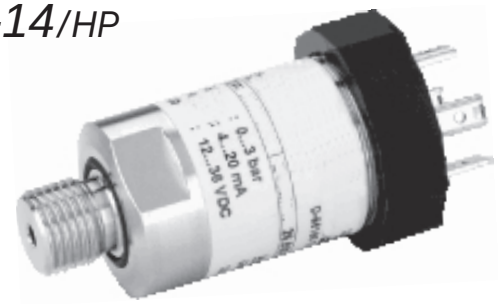
TPR-14/N

- **SENSOR DE SILICIO EN BASE CERÁMICA**
- **EXACTITUD 0,5 % FSO IEC 60770**
- **SEÑAL DE SALIDA 4 ... 20 mA a 2 HILOS (OPCIÓN: 0 ... 10 V a 3 HILOS)**
- **CONEXIÓN A PROCESO ¼" GAS**
- **CONECTOR DIN 43650**
- **CAJA ESTANCA ALUMINIO ANODIZADO IP 65**

TPR-14/LP .../HP para líquidos y gases no corrosivos

Transmisor de presión de BAJO COSTE con salida 4/20 mA para aplicaciones en **líquidos y gases no corrosivos**, con rangos de 1 a 400 bar

TPR-14/LP
TPR-14/HP



- **SENSOR CERÁMICO**
- **EXACTITUD 0,5 % FSO IEC 60770**
- **SEÑAL DE SALIDA 4 ... 20 mA A 2 HILOS (OPCIÓN: 0 ... 10 V a 3 HILOS)**
- **CONEXIÓN A PROCESO ¼" GAS**
- **CONECTOR DIN 43650**
- **CAJA INOXIDABLE ESTANCA IP 65**
- **MODELOS: TPR-14/LP (1 a 25 bar) Y TPR-14/HP (40 a 400 bar)**

CE

DESCRIPCION ESPECIFICA

Fabricamos una amplia gama de instrumentos de medida y control con entrada 4-20 mA para conectar al transmisor de presión relativa **TPR-14**:

- Reguladores digitales PID configurables, con alarmas: **serie BS-2000, LS-3000 y HS-7000**
- Indicadores digitales configurables, con o sin alarmas: **serie BS-2000, LS-3000 y HS-7000**
- Indicadores digitales autoalimentados por la misma línea de señal 4-20 mA: **PM-3650 y PM-6670**
- Fuentes de alimentación conmutadas y lineales de 24 Vdc y 1 A: **FAC-24/1000**
- Sistemas inteligentes de adquisición de medidas y control por PC: **DAS-8000 y HS-7000**

TÜV SÜD ISO 9001

CARACTERISTICAS TECNICAS

TPR-14/N

- Rangos (bar): 0.25 0.4 0,6 1 1,6 2.5 4 6 10
- Sobrepresión (bar): 1 1 3 3 7 7 12 12 25
- Exactitud: ..... ≤± 0,5 % FSO
- Conexión a proceso: ..... ¼"GAS
- Protección encapsulado: IP-65
- Salida: ..... 4-20 mA 2 hilos (opción 0-10 V 3 hilos)
- Conexión eléctrica: ..... conector macho y hembra DIN 43650
- Tensión alimentación: ..... entre 12 y 36 Vcc ..... remota, superpuesta a la línea de medida
- Carga máxima:
  2 hilos ..... [UB (V) -12 V] / 0,02 A
  3 hilos ..... >1 MΩ
- Deriva térmica: ..... ≤ ± 0,3 % FSO / 10 K
- Margen de temp. compensada: ..... 0 a 60 °C
- Temperatura de trabajo
  Sensor: ..... -25 a +90 °C
  Ambiente: ..... -25 a +85 °C
- Unidades: ..... bar (normalizada)
- Tiempo de respuesta: ..... ≤ 3 ms
- Resistencia a vibraciones: ..... 10 g / 20 a 2000 Hz
- Resistencia a choques: ..... 100 g / 11 ms
- Materiales:
  Caja ..... Aluminio anodizado
  Sensor (P_N hasta 1 bar) ..... Acero inox. 1.4305, RTV, cerámica Al_2O_3, silicona
  Sensor (P_N desde 1 bar) ..... Cerámica Al_2O_3
  Junta ..... FKM

TPR-14/LP.../HP

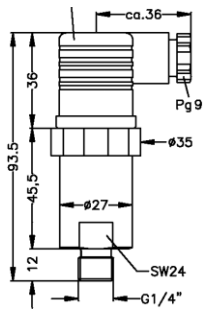
- Presión (bar):

	LP								HP					
Relativa:	1	1,6	2,5	4	6	10	16	25	40	60	100	160	250	400
Absoluta:	-	1,6	2,5	4	6	10	16	25	-	-	-	-	-	-
Sobrepresión:	3	7	7	12	12	25	50	50	120	120	250	500	500	600

- Captador: ..... piezorresistivo con compensación de temperatura integrada (otros bajo demanda)
- Exactitud: ..... ≤± 0,5 % FSO (BFSL: ≤± 0,5 % FSO) (*1 % FSO)
- Conexión a proceso: ..... ¼"GAS
- Protección encapsulado: IP-65
- Salida: ..... 4-20 mA 2 hilos (opción 0-10 V 3 hilos)
- Conexión eléctrica: ..... conector macho y hembra DIN 43650
- Tensión alimentación: ..... entre 12 y 36 Vcc ..... remota, superpuesta a la línea de medida
- Carga máxima:
  2 hilos ..... [UB (V) -12 V] / 0,02 A
  3 hilos ..... >1 MΩ
- Deriva térmica: ..... ≤ ± 0,3 % FSO / 10 K
- Margen de temp. compensada: ..... 0 a 70 °C
- Temperatura de trabajo
  Sensor: ..... -25 a +125 °C
  Ambiente: ..... -25 a +85 °C
- Unidades: ..... bar (normalizada)
- Tiempo de respuesta: ..... ≤ 3 ms
- Resistencia a vibraciones: ..... 10 g / 20 a 2000 Hz
- Resistencia a choques: ..... 100 g / 11 ms
- Materiales:
  Caja ..... Acero inox. 1.4305
  Membrana ..... Cerámica Al_2O_3
  Junta sensor ..... FKM / P_N ≥100 bar: NBR

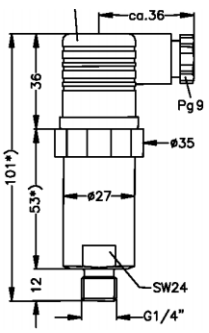
MEDIDAS

TPR-14/N



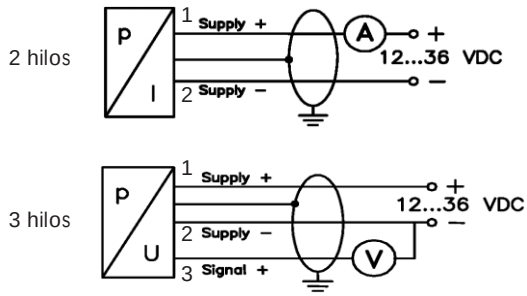
Peso: aprox. 0,200 kg

TPR-14/LP.../HP



Peso: aprox. 0,200 kg

CONEXIÓN ELECTRICA



APLICACIONES

TPR-14/N

· Aplicaciones generales en neumática y vacío, aire limpio, etc

TPR-14/LP.../HP

· Aplicaciones generales gases y líquidos, proceso, riego, etc

COMO PEDIRLO

MODELOS:

TPR-14/N Neumática y Vacío
TPR-14/LP Líquidos y gases 1 a 25 bar
TPR-14/HP Líquidos y gases 40 a 400 bar

Opciones:

- Para aplicaciones en oxígeno
- Presión Absoluta
- Salida 0-10 V a 3 hilos
- Prensaestopas PG7+ 2m
- Cada metro de cable

Desin Instruments, s.a.
Av. Frederic Rahola, 49 - 08032 BARCELONA (España)
Tel. (+34) 93 358 6011* - Fax (+34) 93 357 6850
e-mail:desin@desin.com - http://www.desin.com

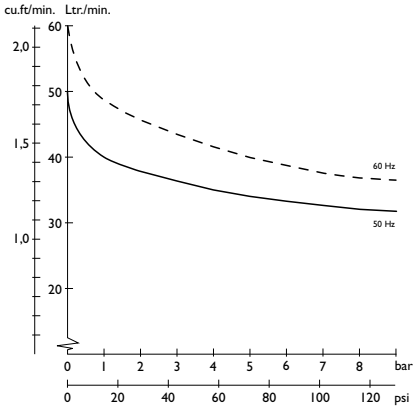
TPR-14 /N.../LP.../HP

127.43

0715 H127.43-1

6-25 (oil-lubricated)

Voltage	V	100	100	120	200	200	230	230				
Frequency	Hz	50	60	60	50	60	50	60				
Motor	HP kW	0.54 0.40	0.54 0.40	0.54 0.40	0.46 0.34	0.46 0.34	0.46 0.34	0.46 0.34				
Displacement	l/min CFM	50 1.77	60 2.12	60 2.12	50 1.77	60 2.12	50 1.77	60 2.12				
FAD @ 8 bar	l/min CFM	32 1.13	37 1.31	37 1.31	32 1.13	37 1.31	32 1.13	37 1.31				
Max. pressure[1)]	bar psi	8 120	8 120	8 120	8 120	8 120	8 120	8 120				
Max. current	A	6.2	6.2	6.2	2.9	2.9	2.9	2.9				
Tank size	litres gallon	25 6.6	25 6.6	25 6.6	25 6.6	25 6.6	25 6.6	25 6.6				
Weight	kg lbs	29 64	29 64	29 64	29 64	29 64	29 64	29 64				
Noise level	dB(A)/1m	45	45	45	45	45	45	45				
Thermal protection		Yes	Yes	Yes	Yes	Yes	Yes	Yes				
Duty cycle		50%	50%	50%	50%	50%	50%	50%				



Dimensions (l x w x h)
380 x 380 x 550 mm / 15 x 15 x 21 5/8 inch

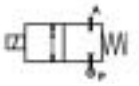
[1)] Higher pressure available upon request

Technical modifications reserved

JUN-AIR®

E-mail: info@jun-air.dk
Internet: www.jun-air.com

2/2 WAY DIRECT ACTING SOLENOID VALVE, G 1/8" - G 1/4"



normally closed

TYPE: D262/263

TECHNICAL SPECIFICATIONS

Media: water, oil, air	
Media temperature: -10°C ÷ +130°C	
Ambient temperature: -10°C ÷ +50°C	
Body material: brass (CW617N EN 12165)	
Orifice material: stainless steel (AISI 303 EN 10088-3)	
Operator material: stainless steel	
Seal material: FKM	
Coil power:	AC 18VA (holding)
	AC 36VA (inrush)
	DC 14W
Protection class: IP 65 (with connector)	

OPTIONS

Normally open (Ex. code RD263DVG) with coils class “H” only

Manual override (Ex. code D262DVHM)

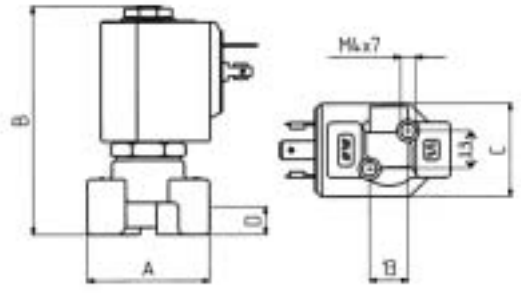
EPDM seal for air and hot water MAX 120°C (Ex. code D262DEH)

RUBY seal -10°C +180°C for high temperature with class "H" coils (Ex. code D262DRC 7201)

SELECTION TABLE

VALVE	G connection	Nominal Diameter	Flow rate kvs	OPD min	OPD max		COILS	
Code	[ISO 228]	[mm]	[l/min]	[bar]	AC [bar]	DC [bar]	Code	[Volts/Hz]
D262DVA	1/8”	1.0	0.5	0	30	30	7250	24V DC
D262DVC	1/8”	1.5	1.3	0	24	24	7200	24V 50/60Hz
D262DVG	1/8”	2.5	3.4	0	18	16	7400	110V 50Hz - 120V 60Hz
D262DVH	1/8”	3.0	4.5	0	15	8	7600	200V 50Hz – 220V 60Hz
D263DVC	1/4”	1.5	1.3	0	24	24	7700	230V 50Hz – 240V 60Hz
D263DVG	1/4”	2.5	3.4	0	18	16		
D263DVH	1/4”	3.0	4.5	0	15	8		
D263DVL*	1/4”	4.0	6.0	0	8	5		
D263DVN*	1/4”	5.0	7.5	0	5	2.5		
D263DVP*	1/4”	6.0	8.5	0	3	1		

* NO version not available



DIMENSIONS & WEIGHTS

G connection	A	B	C	D	weight
[ISO 228]	[mm]	[mm]	[mm]	[mm]	[Kg]
1/8”	40	77.5	32	11	0.26
1/4”	40	77.5	32	11	0.26

Flow direction overseat 1→2