Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

DAVID MARQUÉS DEL RÍO

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

ANÁLISIS E IMPLEMENTACIÓN DE POSIBLES ESTRATEGIAS DE EFICIENCIA ENERGÉTICA EN LOS EDIFICIOS DE PHOENIX CONTACT

DOCUMENTO Nº3: MANUAL DEL PROGRAMADOR

ÍNDICE

ÍNDICE DE TABLAS E IMÁGENES

1. GESTIÓN DE LA ILUMINACIÓN

En este apartado se describen las particularidades de la programación del sistema de gestión de la iluminación en el edificio de oficinas que Phoenix Contact SAU posee en Llanera.

Comprende 2 partes básicas:

- El grupo POU denominado *Iluminacion*, que se basa en el programa diseñado por la propia empresa, y que se ha modificado convenientemente para su trabajo conjunto con el paquete de SCADA Atvise.

  En este grupo se incluyen los siguientes programas/bloques de función, que se detallarán en próximos apartados:

Grupo POU	Tipo	Nombre	Lenguaje IEC
Iluminación	Programa	P2	FBD
Iluminación	Programa	P1	FBD
Iluminación	Programa	PB	FBD
Iluminación	Programa	PSS	FBD
Iluminación	Programa	P2_mecanismos_luminarias	ST
Iluminación	Programa	P2_mecanismos_luminarias	ST
Iluminación	Programa	PB_mecanismos_luminarias	ST
Iluminación	Programa	PSS_mecanismos_luminarias	ST
Iluminación	Programa	Puesta_en_hora	ST
Iluminación	Programa	dias_festivos	ST
Iluminación	Programa	Modo_automatico	FBD
Iluminación	Bloque Funcional	control_iluminacion	FBD
Iluminación	Bloque Funcional	T_activacion	ST
Iluminación	Bloque Funcional	rearme_manual	FBD
Iluminación	Bloque Funcional	dia_festivo	ST
Iluminación	Bloque Funcional	dia_semana	ST
Iluminación	Bloque Funcional	tiempo_restante	ST

Tabla 1.1 - Grupo Iluminación

- El grupo POU denominado *SCADA_Atvise*, que comprende los distintos bloques funcionales y programas que se han creado para permitir la comunicación entre el programa de control en sí, y el servidor de Atvise. También se realizan aquí las conversiones entre formatos necesarias para mostrar los valores convenientemente en las distintas pantallas de operador.

En este grupo se incluyen los siguientes programas/bloques de función, que se detallarán en próximos apartados:

Grupo POU	Tipo	Nombre	Lenguaje IEC
SCADA_Atvise	Programa	P2_Habitaciones	FBD
SCADA_Atvise	Programa	P1_Habitaciones	FBD
SCADA_Atvise	Programa	PB_Habitaciones	FBD
SCADA_Atvise	Programa	PSS_Habitaciones	FBD
SCADA_Atvise	Programa	P2_Grupos	FBD
SCADA_Atvise	Programa	P1_Grupos	FBD
SCADA_Atvise	Programa	PB_Grupos	FBD
SCADA_Atvise	Programa	PSS_Grupos	FBD
SCADA_Atvise	Programa	P2_Tiempos	FBD
SCADA_Atvise	Programa	P1_Tiempos	FBD
SCADA_Atvise	Programa	PB_Tiempos	FBD
SCADA_Atvise	Programa	PSS_Tiempos	FBD
SCADA_Atvise	Programa	Horarios	FBD
SCADA_Atvise	Bloque Funcional	Escribe_Trearme	ST
SCADA_Atvise	Bloque Funcional	Muestra_Trearme	ST
SCADA_Atvise	Bloque Funcional	Muestra_Trestante	ST

Tabla 1.2 - Grupo SCADA_Atvise

1.1 SOFTWARE UTILIZADO

Las herramientas informáticas utilizadas en esta parte son las siguientes:

- **PCWorx 6.30.1202:** Para la programación de PLCs y dispositivos de la firma Phoenix Contact.
- **PCWorx SRT 1.1**: Es el simulador de PLCs de Phoenix Contact que corre en un PC Windows, se ha usado para realizar pruebas menores cuando no está disponible el equipo físico.
- **AX-OPC Server 3.00:** Se trata del servidor OPC-DA de Phoenix Contact, que recoge las variables publicadas por el programa de control mediante el protocolo estandarizado OPC, para poder ser utilizadas por otros clientes (Atvise, en nuestro caso).
- **Atvise 2.5.8:** Es un paquete SCADA basado en web, concretamente en tecnología HTML5, que permite recrear las pantallas de operador en multitud de dispositivos siempre y cuando posean navegadores web compatibles con este estándar.

Posee su propio servidor OPC-UA integrado, que se encarga de replicar las variables que le llegan por OPC-DA desde el PLC a modo de variables de pantalla (*MirrorInput y MirrorOutput*), para que puedan ser utilizadas en el *builder* de Atvise para el diseño del SCADA.

1.2 LIBRERÍAS

No se han usado librerías adicionales en esta parte del código.

1.3 RELACIÓN DE TAREAS

Para configurar la ejecución de las distintas funcionalidades implementadas en el programa de gestión de la iluminación, se han utilizado tareas de tipo CYCLIC.

Este programa de gestión posee también dos tareas adicionales que se ejecutan en los arranques en frío y templado del PLC, para que él mismo calcule los días no laborables dentro del año actual, en los que no debe activar los contactores del edificio.

En cuanto a la tarea correspondiente al *Atvise*, se han agrupado aquí las instancias a los programas anteriormente mencionados, que sirven para la comunicación con el SCADA.

Tarea	Tipo	Intervalo
ATVISE	CYCLIC	100ms
GESTION	CYCLIC	100ms
Frio	SYSTEM	-
Templad	SYSTEM	-

Tabla 1.3 - Estructura de tareas

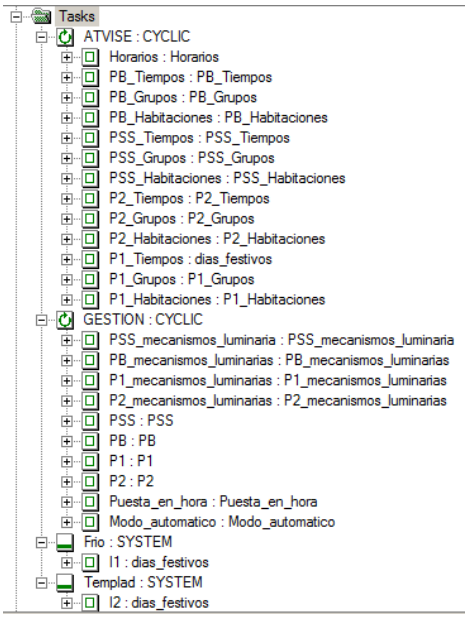

Imagen 1.1 - Relación de tareas

1.4 RELACIÓN DE VARIABLES

Las variables usadas en la parte del Proyecto correspondiente a la gestión de la iluminación del edificio de oficinas, responden a la división a modo de cuadrícula, de las plantas de dicho edificio en casillas de la forma *letra+número*, donde, las filas son representadas mediante letras, mientras que las columnas son definidas por el número que les sigue a continuación. Por ejemplo:

A1

De esta forma, en el ejemplo anterior, nos referimos a la primera fila (A), y primera columna (1), de la cuadrícula.

1.4.1 Variables que definen las plantas del edificio

Para cada una de las plantas del edificio, se definen los siguientes nombres de variables de acuerdo a la estructura presentada:

- C_P*x*_*x*: son los contactores identificados por planta en la que se encuentran y su número ordinal.
- L_P*x*_*filacolumna*: los grupos de luminarias identificados por planta, fila y columna en la que se encuentran.
- M_P*x*_*filacolumna*: los mecanismos identificados por planta, fila y columna en la que se encuentran.
- Luminarias_C_P*x*_x: array de luminarias pertenecientes a un contactor, definidos por la planta en la que se encuentran, y el ordinal del contactor.
- Mecanismos_C_P*x*_x: array de mecanismos pertenecientes a un contactor, definidos por la planta en la que se encuentran, y el ordinal del contactor.
- Manual_C_P*x*_*x*: pulsadores de rearme de cada contactor, identificados por la planta en la que se encuentra dicho contactor y su ordinal dentro de la misma.
- Manual_DEF_C_P*x*_*x*: pulsadores de rearme durante tiempo por defecto de cada contactor, identificados por la planta en la que se encuentra dicho contactor y su ordinal dentro de la misma.
- TR_C_P*x*_*x*: tiempo restante de cuenta atrás para el contactor de la planta referida, y su número ordinal.
- tiempo_rearme_P*x*_*x:* tiempo de rearme introducido para el contactor de la planta referida, y su número ordinal.

Todas ellas son variables globales, agrupadas según la planta a la que pertenecen dentro de la lista de variables.

1.4.2 Variables adicionales para su uso en Atvise

Se han creado, además, variables globales para la muestra y modificación, en algunos casos, de variables del programa en Atvise. Son las siguientes:

- L_SC_P*x*_*Habitacion*: se refieren a la agrupación de luminarias por nombre de habitación y piso en el que se encuentran.
- L_SC_P*x*_Grupos_*Habitacion_filacolumna*: responden a las agrupaciones de luminarias similares en la casilla dada, perteneciente a dicha planta y habitación.
- SC_tiempo_rearme_P*x*_C*x*: es el tiempo de rearme del contactor mencionado dentro de su planta, en minutos (INT).
- TR_SC_P*x*_C*x*: cadena de texto que contiene el tiempo restante para la apertura del contactor dado dentro de la planta a la que pertenece.

1.4.3 Variables generales

Son las referentes a horarios de trabajo, otras variables de tipo temporal (mes, año, minuto, etc.) que sirven, además de internamente al programa, para formar cadenas de texto para su visualización por el operador. También tenemos aquí las variables correspondientes a los modos de funcionamiento automático, así como la sincronización de relojes en caso de utilizarse.

La lista completa de variables para cada planta, adicional a esta explicación de su estructura, se acompaña al código fuente del Proyecto.

1.5 DATOS PROPIOS

Ha sido necesaria la definición de 4 tipos adicionales de datos propios, dentro del programa original de gestión de la iluminación:

- Array de 12 posiciones de tipo INT:
  `m_1_12_INT : ARRAY [1..12] OF INT;`
- Array de 15 posiciones de tipo STRING:
  `m_1_15_STRING : ARRAY [1..15] OF STRING;`
- Array de 14 posiciones de tipo BOOL
  `m_1_14_BOOL : ARRAY [1..14] OF BOOL;`
- Array de 15 posiciones de tipo WORD:
  `m_1_15_WORD : ARRAY [1..15] OF WORD;`

1.6 GRUPO GESTIÓN - DETALLES DE PROGRAMACIÓN

Como ya se ha mencionado, esta sección del Proyecto parte del programa original desarrollado por Phoenix Contact para su uso propio en el edificio de oficinas de Llanera. Los programas y bloques funcionales usados para implementar este software de gestión no se han modificado, a menos que se indique lo contrario en su correspondiente explicación, por lo que este apartado se limitará a explicar la funcionalidad y particularidades de cada uno de ellos, ya que el desarrollo de este programa completo se escapa al alcance de este Proyecto.

1.6.1 Programas de control de cada planta

Realizados en lenguaje FBD, relacionan a través de los bloques funcionales *control_iluminacion,* los arrays de mecanismos que están contenidos en los circuitos de los distintos contactores, con sus corresponientes arrays de luminarias, declaración esta que se realiza dentro de los programas correspondientes a cada planta (*Px_mecanismos_luminarias*), y que se explicará en el próximo subapartado. En este punto es donde entran también en juego las variables de rearme manual de los contactores, teniendo como salida las variables de tiempo restante de funcionamiento.

Existe uno para cada planta, denominados de igual forma que el resto de programas, de acuerdo a los nombres de cada piso. Estos programas son:

- P2: Programa de control de la Planta 2
- P1: Programa de control de la Planta 1
- PB: Programa de control de la Planta Baja
- PSS: Programa de control de la Planta (semi) Sótano

1.6.2 Programas de relación mecanismos-luminarias

Programados en texto estructurado (ST), se encargan de establecer una relación entre las variables que identifican individualmente, por un lado a cada mecanismo, y por el otro a cada grupo de luminarias de esa planta en concreto. El objetivo de estos programas es reagrupar estos datos en sendos arrays del tipo *m1_12_bool*, para ser usados después en los bloques *control_iluminacion*, en el programa de control correspondiente a cada planta. Son también cuatro:

- P2_mecanismos_luminarias: relación de mecanismos y luminarias, y arrays de la Planta 2.
- P1_mecanismos_luminarias: relación de mecanismos y luminarias, y arrays de la Planta 1.
- PB_mecanismos_luminarias: relación de mecanismos y luminarias, y arrays de la Planta Baja.
- PSS_mecanismos_luminarias: relación de mecanismos y luminarias, y arrays de la Planta (semi) Sótano.

1.6.3 Programa Puesta_en_hora

Implementado en ST, realiza la sincronización de la hora del programa mediante los registros internos del firmware del PLC, todas las noches, para evitar que el controlador pierda la hora en caso de reinicio. Además, se prevé que pueda sincronizarse también con fuentes externas, como la hora del SCADA, aunque estas funciones no serán usadas en esta implementación.

1.6.4 Programa dias_festivos

Se trata de una simple declaración de los días festivos, tanto nacionales como autonómicos/locales, en los que el modo automático de gestión de la iluminación no debe ser activado por tratarse de días no laborables.

Para ello, se almacenan dichos días a modo de cadena de texto en las distintas posiciones de un array de datos propios descritos anteriormente, del tipo *m_1_15_STRING,* identificado como *dias_festivos_anyo_actual[]*.

Este programa se instancia en las tareas *Templado* y *Frio*, que son invocadas por evento en cada arranque del PLC.

1.6.5 Programa Modo_automatico

En este programa, implementado en lenguaje de bloques funcionales (FBD), se activa y desactiva la variable booleana *Modo_automatico*, con la cual se gestiona la apertura o cierre de los contactores principales del edificio (de tipo N/C, para permitir el funcionamiento de las luces en caso de caída del sistema o si se desea un funcionamiento manual).

Esta variable, que se mantiene automáticamente a *TRUE* durante el horario laboral, es necesario desactivarla transcurrido el horario de trabajo, y en días libres o festivos. Para ello, se utiliza un bloque *T_activacion*, de creación propia, al que le llega información desde otros dos bloques llamados *dia_festivo* y *dia_semana*, que generan la información del día en que nos encontramos.

En el bloque *T_activacion* también entran en juego las horas de encendido y apagado de los contactores, que serán diferentes para los días L-J, Viernes o fin de semana.

Estos bloques funcionales se detallarán a continuación dentro de este mismo apartado.

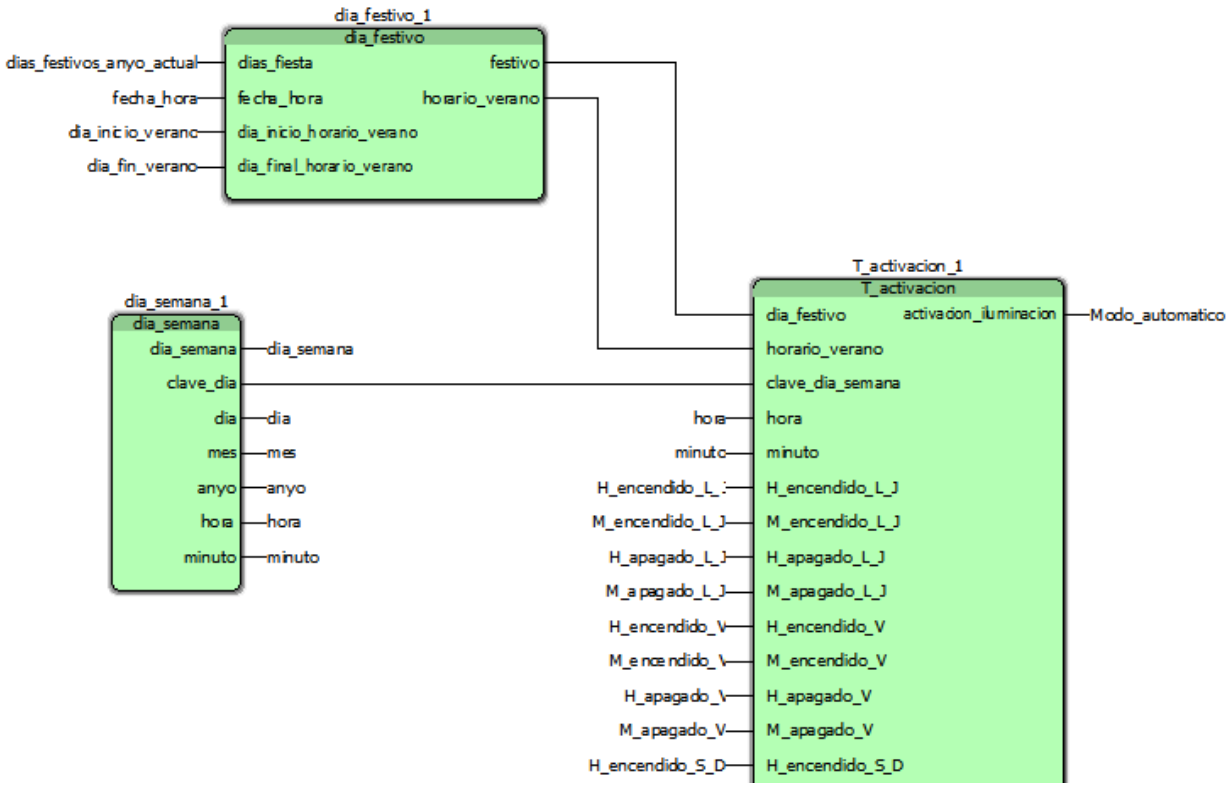

Imagen 1.2 - Vista del programa Modo_automatico

1.6.6 Bloque funcional control_iluminacion

Con este bloque, realizado en lenguaje FBD, se hace posible la relación de los distintos arrays de mecanismos, con aquellos que contienen las luminarias correspondientes a cada contactor que están presentes en las distintas plantas del edificio.

Este bloque sirve también para comandar la apertura o cierre del contactor asociado, mediante actuación sobre el pulsador de rearme o uno de los mecanismos que acciona grupos de luminarias de este contactor. El tiempo durante el que esto ocurre, le llega al bloque en forma de variable temporal por la entrada al respecto, generando internamente el tiempo restante para la apertura, que se utilizará como salida del bloque.

La descripción de dicho bloque es la siguiente:

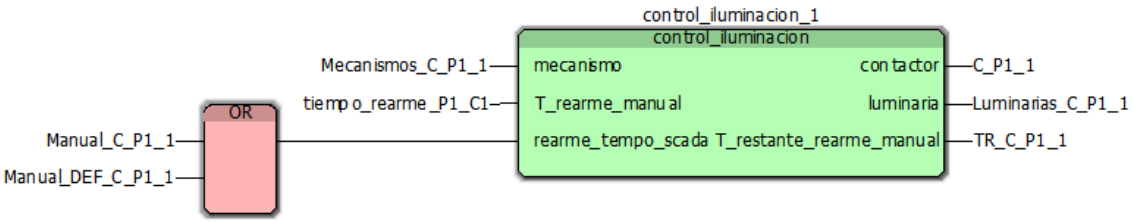

Imagen 1.3 - Ejemplo de utilización del bloque control_iluminacion

VAR_INPUT (entradas)	Tipo	Descripción
mecanismo	m_1_14_BOOL	Array de mecanismos del contactor.
T_rearme_manual	TIME	Tiempo que dura el rearmado manual del contactor
rearme_tempo_scada	BOOL	Activa el rearme manual del contactor

Tabla 1.4 - Entradas bloque control_iluminacion

VAR_OUTPUT (salidas)	Tipo	Descripción
contactor	BOOL	Salida de activación del contactor N/C (TRUE contactor cerrado, FALSE contactor abierto)
m_1_14_BOOL	m_1_14_BOOL	Array de luminarias que gestiona el contactor
T_restante_rearme_manual	STRING	Cadena de texto con el tiempo restante para la apertura tras un rearme.

Tabla 1.5 - Salidas bloque control_iluminacion

En el caso del ejemplo que nos ocupa, utilizamos dicho bloque para gestionar el contactor P1_1 (es decir, el contactor número 1 de la planta primera P1). Este bloque está conectado a los arrays de luminarias y mecanismos que le corresponden.

El rearme, por su parte, viene dado a través de una puerta OR por sendas variables *Manual_C* y *Manual_DEF_C*, que se corresponden con pulsadores en el SCADA. La primera de ellas se encarga del rearme del mismo durante un tiempo -*tiempo_rearme* - dado por el usuario, mientras que la segunda (DEF - por defecto), efectúa dicho rearme durante un tiempo predeterminado dado (60 minutos en nuestro caso). Esto se detallará más en profundidad cuando se expliquen las particularidades de programación del grupo POU *SCADA_Atvise*.

1.6.7 Bloque funcional T_activacion

Este bloque, que ha sido implementado en ST, es el encargado de activar o desactivar el funcionamiento automático del sistema, es decir, de conmutar la variable *Modo_automatico* dependiendo de si estamos en horario laboral, o si le llega información de que nos encontramos en día festivo o hay cambios en el horario habitual por ser verano.

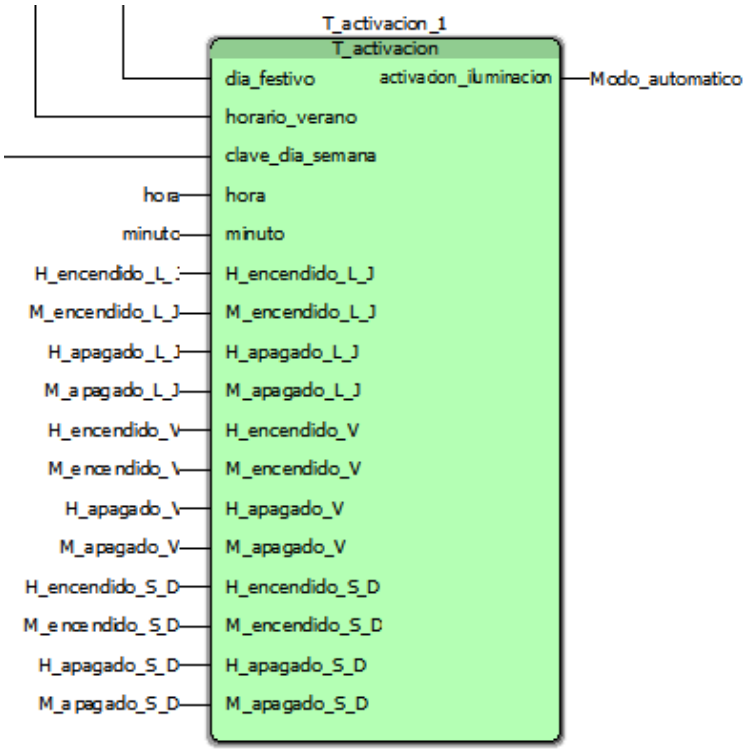

Imagen 1.3 - Ejemplo de utilización del bloque T_activacion

VAR_INPUT (entradas)	Tipo	Descripción
dia_festivo	BOOL	TRUE - Dia actual es festivo
horario_verano	BOOL	TRUE - En horario de verano
clave_dia	INT	Día de la semana actual (1-7)
hora	INT	Hora actual
minuto	INT	Minuto actual
H_encendido_L_J	INT	Hora encendido Lunes-Jueves
M_encendido_L_J	INT	Minuto encendido Lunes-Jueves
H_apagado_L_J	INT	Hora apagado Lunes-Jueves
M_apagado_L_J	INT	Minuto apagado Lunes-Jueves
H_encendido_V	INT	Hora encendido Viernes
M_encendido_V	INT	Minuto encendido Viernes
H_apagado_V	INT	Hora apagado Viernes
M_apagado_V	INT	Minuto apagado Viernes
H_encendido_S_D	INT	H. encendido Sábado/Domingo
M_encendido_S_D	INT	M. encendido Sábado/Domingo
H_apagado_S_D	INT	H. apagado Sábado/Domingo
M_apagado_S_D	INT	M. apagado Sábado/Domingo

Tabla 1.6 - Entradas bloque T_activacion

VAR_OUTPUT (salidas)	Tipo	Descripción
activación_iluminacion	BOOL	TRUE - Modo Automático activo

Tabla 1.7 - Salidas bloque T_activacion

1.6.8 Bloque funcional rearme_manual

Mediante este bloque de función, programado también en FBD, se lleva a cabo el rearme de los contactores al llegarle la señal al respecto desde uno de los mecanismos, y se utilizará como parte interna de los bloques de *control_iluminacion.* A continuación se muestra una vista de uno de estos bloques dentro del programa de control:

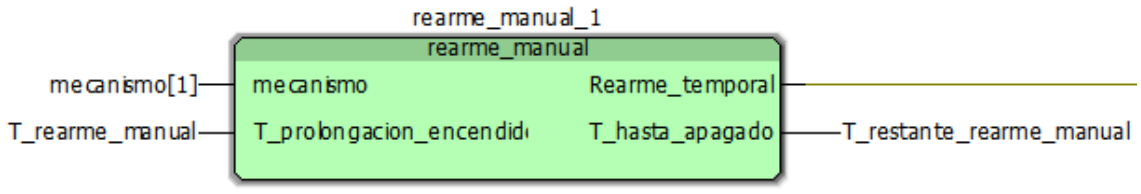

Imagen 1.4 - Ejemplo de utilización del bloque rearme_manual

VAR_INPUT (entradas)	Tipo	Descripción
mecanismo	BOOL	Detección del flanco positivo del mecanismo
T_prolongacion_encendido	TIME	Tiempo que dura el rearmado manual del contactor

Tabla 1.8 - Entradas bloque rearme_manual

VAR_OUTPUT (salidas)	Tipo	Descripción
Rearme_temporal	BOOL	TRUE - Salida a contactor rearmado
T_hasta_apagado	STRING	Tiempo restante hasta el apagado

Tabla 1.9 - Salidas bloque rearme_manual

Del ejemplo anterior se deduce que, de detectarse un flanco positivo en el mecanismo, se rearma el contactor conectado a la salida (*Rearme_temporal*), durante un tiempo *T_rearme_manual* dado, cuya cuenta se recoge en la salida *T_restante_rearme_manual*.

1.6.9 Bloque funcional dia_festivo

Implementado en texto estructurado (ST), comprueba a través de una cadena (STRING) que contiene la hora y el día actuales, si nos encontramos en festivo y/o horario de verano. De esta forma, origina dos salidas booleanas *festivo* y *horario_verano*, que tomarán el valor TRUE si se dan alguna de estas condiciones.

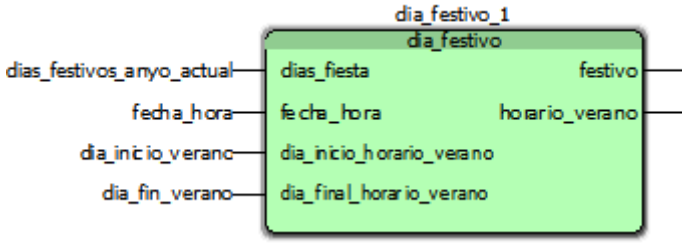

Imagen 1.5 - Ejemplo de utilización del bloque dia_festivo

VAR_INPUT (entradas)	Tipo	Descripción
dias_fiesta	m_1_15_STRING	Array que contiene los días festivos
fecha_hora	STRING	String con fecha y hora actuales
dia_inicio_horario_verano	STRING	String con día de inicio h. verano
dia_final_horario_verano	STRING	String con día de fin h. verano

Tabla 1.10 - Entradas bloque dia_festivo

VAR_OUTPUT (salidas)	Tipo	Descripción
festivo	BOOL	TRUE - Dia actual es festivo
horario_verano	BOOL	TRUE - Día actual es horario de verano

Tabla 1.11 - Salidas bloque dia_festivo

1.6.10 Bloque funcional dia_semana

Este bloque especifica, a partir de la fecha actual, el día de la semana en que nos encontramos. Esto lo realiza a partir de la extracción de los datos de día, mes, año (*anyo*), etc. de la cadena que contiene la fecha y hora actuales. Estas variables se reaprovechan, además, como salidas de dicho bloque para, por ejemplo, su uso en el SCADA.

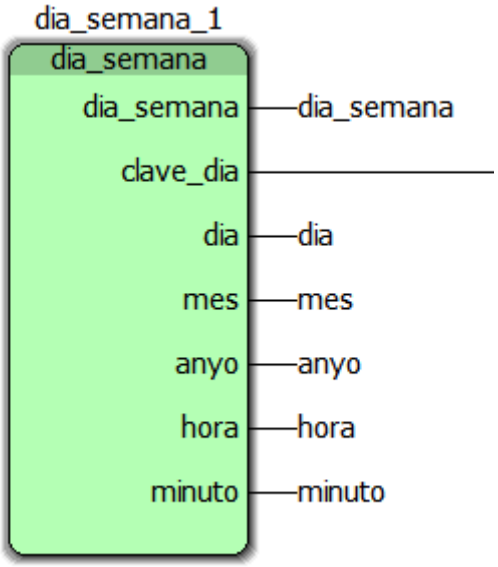

Imagen 1.6 - Ejemplo de utilización del bloque dia_semana

VAR_OUTPUT (salidas)	Tipo	Descripción
dia_semana	STRING	Día de la semana
clave_dia	INT	Clave del día de la semana en el rango 1-7 (INT)
dia	INT	Día actual
mes	INT	Mes actual
anyo	INT	Año actual
hora	INT	Hora actual
minuto	INT	Minuto actual

Tabla 1.12 - Salidas bloque dia_semana

1.6.11 Bloque funcional tiempo_restante

Este último bloque que forma parte de este grupo POU, está también programado en ST. Su función es la de generar una cadena de texto (formato STRING) a partir de la conversión de formato de tiempo, que le llega en milisegundos (ms) en una variable tipo TIME.

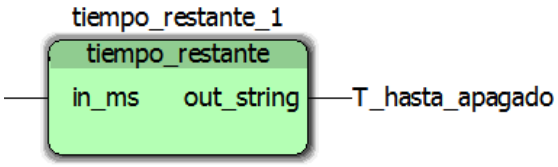

Imagen 1.7 - Ejemplo de utilización del bloque tiempo_restante

VAR_INPUT (entradas)	Tipo	Descripción
in_ms	TIME	Tiempo en milisegundos (ms)

Tabla 1.13 - Entradas bloque tiempo_restante

VAR_OUTPUT (salidas)	Tipo	Descripción
out_string	STRING	Cadena de tiempo en formato hh:mm:ss

Tabla 1.14 - Salidas bloque tiempo_restante

1.7 GRUPO SCADA_ATVISE - DETALLES DE PROGRAMACIÓN

Este grupo POU engloba las funcionalidades implementadas sobre el programa de gestión de iluminación de Phoenix Contact SAU, para la comunicación de dicho programa con el SCADA Atvise.

El envío de valores al SCADA de las distintas variables se realiza mediante el servidor OPC integrado de Phoenix Contact (AX-OPC Server), por lo que buena parte de los programas pertenecientes a este grupo se destinarán a adaptar los formatos en los que estos datos se muestran, así como en agrupar y convertir la información necesaria para permitir la interacción entre el usuario y el sistema de iluminación del edificio.

1.7.1 Programas Px_Habitaciones

En los programas cuyo nombre sigue el formato *Px_Habitaciones*, se agrupan las variables correspondientes a las luminarias de cada habitación del edificio, para dar lugar a una variable de salida identificativa de dicha habitación.

Estas variables, que identifican la habitación en la que alguna de las luces está encendida (de ahí que estos programas sean una mera agrupación de variables de entrada que pasan por una puerta OR, para dar lugar a una variable BOOL de salida), se utilizan luego en la vista general de cada una de las plantas dentro del SCADA, para saber al primer vistazo, qué zona del edificio está iluminada total o parcialmente.

Estos programas son cuatro, siguiendo la estructura general de las distintas POUs de gestión:

- P2_Habitaciones: identificación de habitaciones de la Planta 2
- P1_Habitaciones: identificación de habitaciones de la Planta 1
- PB_Habitaciones: identificación de habitaciones de la Planta Baja
- PSS_Habitaciones: identificación de habitaciones de la Planta (semi)Sótano

1.7.2 Programas Px_Grupos

Estos programas, que también siguen la estructura de división en plantas utilizada anteriormente, están realizados en lenguaje de bloques funcionales (FBD). Su función es similar a los explicados en el apartado anterior, *Px_Habitaciones*, pero en este caso, el agrupamiento de variables se realiza por grupos en vez de por habitaciones.

Las luminarias del edificio, tal y como se ha explicado en la definición de las variables, se definen según la división imaginaria a modo de cuadrícula sobre los planos de las plantas del edificio, estando las filas representadas por una letra, y las columnas por un número (ej. P1_C_10).

El objetivo de este programa es entonces, el agrupamiento de las variables de algunos de los grupos de luminarias que ocupan la misma casilla dentro de la cuadrícula del plano, para facilitar su visualización en el SCADA. Esto ocurre, en su mayoría, en las casillas correspondientes a los pasillos. Dichos programas son los siguientes:

- P2_Grupos: identificación de grupos luminarias de la Planta 2
- P1_Grupos: identificación de grupos luminarias de la Planta 2
- PB_Grupos: identificación de grupos luminarias de la Planta Baja
- PSS_Grupos: identificación de grupos luminarias de la Planta (semi)Sótano

1.7.3 Programas Px_Tiempos

Los programas de este tipo, forman el núcleo de la comunicación entre el SCADA y el programa para el rearme de contactores del edificio. Existiendo uno para cada planta, su función es triple:

- Mostrar el tiempo restante durante el cual el contactor permanecerá cerrado fuera del horario laboral.
- Escribir los valores de los tiempos de rearme introducidos en el SCADA, en las variables propias del programa de gestión que manejan la apertura y cierre de los contactores del edificio.
- Generar una cadena de texto con los tiempos de rearme (TIME) almacenados en las variables del programa de gestión, para su muestra en el SCADA.

Estas tres funciones se llevan a cabo gracias a los bloques de función programados al respecto (*Muestra_Trestante, Escribe_Trearme y Muestra_Trearme*, respectivamente, que serán tratados en el siguiente apartado del presente documento).

Los programas de tiempos serán, siguiendo la estructura del resto del Proyecto:

- P2_Tiempos: gestiona los tiempos de rearme/t.restante en el SCADA para la Planta 2
- P1_Tiempos: gestiona los tiempos de rearme/t.restante en el SCADA para la Planta 1
- PB_Tiempos: gestiona los tiempos de rearme/t.restante en el SCADA para la Planta Baja
- PSS_Tiempos: gestiona los tiempos de rearme/t.restante en el SCADA para la Planta (semi)Sótano.

1.7.4 Programa Horarios

El objetivo de este programa es generar cadenas de texto (STRING), en formato HH:MM, a partir de los datos de horarios laborables que el programa de gestión utiliza en formato INT, para definir las horas del día en las que se activa el modo automático de gestión de los contactores del edificio de oficinas.

Las cadenas de texto que aquí se generan se utilizan en la pantalla de operador llamada del mismo modo -Horarios-, como simple información al empleado de los horarios de trabajo que están vigentes (además de la hora actual, horarios de verano, etc.).

1.7.5 Bloque funcional Temporizador

Este bloque de función se ha programado en ST, con el objetivo de crear un temporizador con función de *reset* incorporada, ante la imposibilidad de utilizar los temporizadores IEC estándar (TP, TON, TOF), sin que ello implique grandes cambios en el código ya escrito.

Para ello, se recurre a la variable *PLC_SYS_TICK_CNT*, que lleva un registro incremental de los milisegundos (ms) transcurridos desde el arranque el PLC, hasta el momento actual. Es una función interna de la serie de controladores de Phoenix Contact.

La forma de reseteo del temporizador se realiza pues, en base al valor de este registro, comparándolo con el tiempo fijado por el usuario, y originando una salida booleana que depende de si este tiempo se ha superado (FALSE), o no (TRUE). Si en xInicio aparece un flanco positivo, se fija un nuevo tiempo límite y se resetea la cuenta.

Imagen 1.8 - Ejemplo de utilización del bloque Temporizador

VAR_INPUT (entradas)	Tipo	Descripción
xInicio	BOOL	Flanco para la activación/reseteo del bloque
tTiempo	TIME	Tiempo hasta desactivación

Tabla 1.15 - Entradas bloque Temporizador

VAR_OUTPUT (salidas)	Tipo	Descripción
xSalida	BOOL	Contando (TRUE)/Fin de cuenta (FALSE)

Tabla 1.16 - Salidas bloque Temporizador

Este bloque de temporización se usa dentro del bloque *Muestra_Trestante* para sacar por pantalla los tiempos restantes hasta el apagado del contactor, dependiendo de si el rearme ha sido manual o por activación de alguno de los mecanismos, pudiendo resetear dicha cuenta, y evitando así mostrar valores incorrectos en caso de rearmes sucesivos.

1.7.6 Bloque funcional Muestra_Trestante

El bloque de función *Muestra_Trestante*, implementado en ST, se encarga de mostrar por pantalla los tiempos restantes hasta la apertura de cada contactor, en caso de rearme de los mismos. Pueden darse los siguientes casos:

- Si nos encontramos en horario laborable, se muestra por pantalla la cadena de texto 'H. Laboral'
- Si estamos fuera del horario laborable, y no solicito ningún rearme, se muestra una cadena de texto con el formato 'HH:MM:SS' estática con el tiempo de rearme que se ha introducido. El valor de esta cadena, cambiará lógicamente al proporcionar un nuevo valor hasta el rearme.
- En caso de no estar en horario laborable, y haber solicitado el rearme mediante alguno de los pulsadores de rearme en pantalla, la salida será una cadena de texto que contiene la cuenta atrás del tiempo restante hasta la apertura del contactor. Cuando dicha cuenta atrás termine, y el contactor se abra (salida del contactor a TRUE), se volverá al caso anterior.
- Si nos encontramos fuera del horario laboral, y se solicita un rearme por medio de la activación de alguno de los mecanismos que comandan dicho contactor, la salida de la cadena de texto mostrará 'Mecanismo' para informar al usuario de este hecho.

Ya explicado el comportamiento de dicho bloque, se resumen las entradas y salidas del mismo:

VAR_INPUT (entradas)	Tipo	Descripción
xSel_Modo	BOOL	Modo automático (TRUE), fuera del horario laboral (FALSE)
sTiempo_in	STRING	Cadena de entrada con el tiempo restante hasta apertura
xInput_Rearme	BOOL	Pulsador de rearme manual
tTiempo_Predet	TIME	Tiempo de rearme
xInput_Contactor	BOOL	Estado del contactor correspondiente: cerrado (FALSE), abierto (TRUE)

Tabla 1.17 - Entradas bloque Muestra_Trestante

VAR_OUTPUT (salidas)	Tipo	Descripción
sTiempo_out	STRING	Cadena de salida para la muestra de valores por pantalla

Tabla 1.18 - Salidas bloque Muestra_Trestante

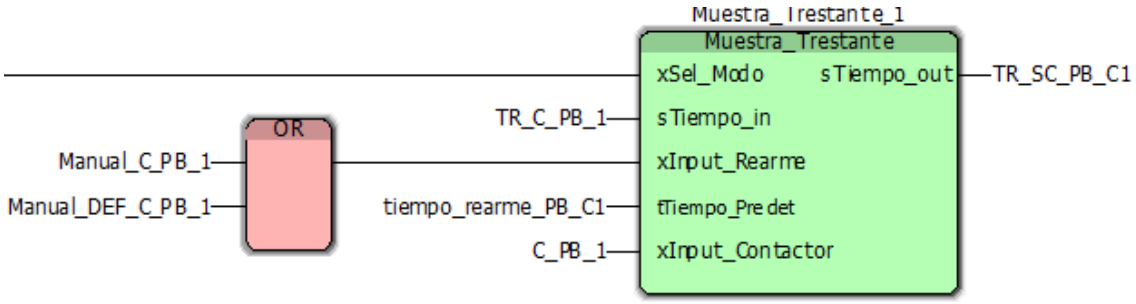

Imagen 1.9 - Ejemplo de utilización del bloque Muestra_Trestante

1.7.7 Bloque funcional Escribe_Trearme

Este bloque se ha creado con la intención de escribir en las variables que gestionan los tiempos de rearme en el programa de gestión de la iluminación, los valores introducidos por el usuario en el SCADA, limitándolos a unos valores máximo y mínimo definidos, y reseteando además dichos tiempos de rearme a valores por defecto de una hora (T#3600s -en segundos tipo TIME-) en caso de solicitar un rearme por defecto -DEF- o detectarse la entrada en horario laborable (para homogeneizar dichos valores entre distintos circuitos).

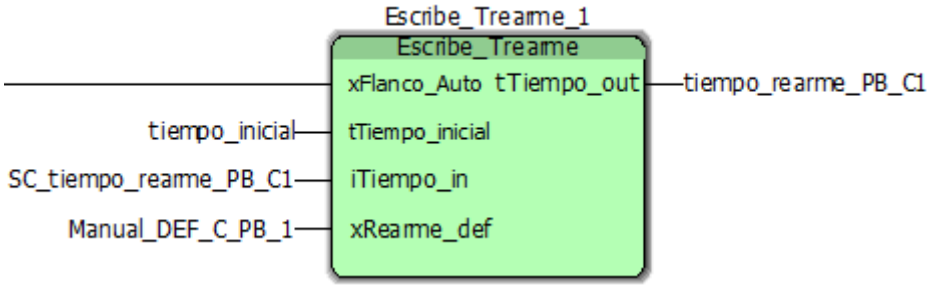

Imagen 1.10 - Ejemplo de utilización del bloque Escribe_Trearme

VAR_INPUT (entradas)	Tipo	Descripción
xFlanco_Auto	BOOL	Detección flanco entrada modo automático (inicio horario laboral)
tTiempo_inicial	TIME	Tiempo por defecto duración del rearme
iTiempo_in	INT	Tiempo de rearme, introducido en minutos en el SCADA, en forma de número entero (INT)
xRearme_def	BOOL	Detección flanco entrada para el rearme por defecto (durante tiempo inicial)

Tabla 1.19 - Entradas bloque Escribe_Trearme

VAR_OUTPUT (salidas)	Tipo	Descripción
tTiempo_out	TIME	Valor (TIME) de salida del bloque, para gestión de iluminación.

Tabla 1.20 - Salidas bloque Escribe_Trearme

En este bloque se efectúa, además, la conversión de valores de tipo INT en el SCADA, a TIME en el programa de control, lo que hace posible este intercambio de información entre el SCADA y el software.

1.7.8 Bloque funcional Muestra_Trearme

Es un simple bloque que convierte valores tipo TIME (presentes en el programa), a tipo INT (en el SCADA). La implementación de este bloque es necesaria para evitar problemas de representación de valores debido al *mirroring* que Atvise realiza de las variables *SC_tiempo_rearme_Px_Cx*, que es del tipo '*Mirror input&output*' (una sola variable es modificada, y registrado su nuevo valor por OPC, y leída a la vez por distintos elementos del SCADA).

En ocasiones, sin la utilización de este bloque, no se refrescan correctamente los valores nuevos introducidos, ya que las variables de pantalla que utiliza Atvise como copia de las originales, poseen memoria interna y guardan el valor anterior con este tipo de *mirroring*.

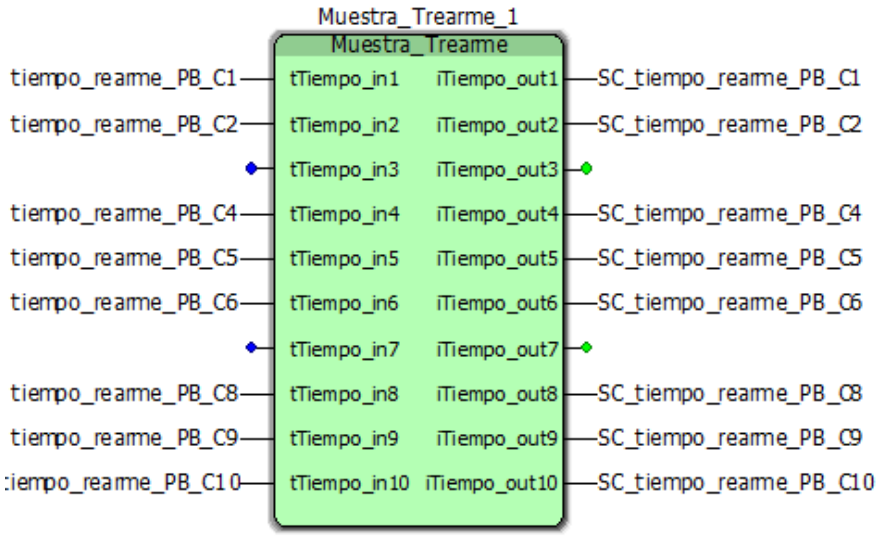

Imagen 1.11 - Ejemplo de utilización del bloque Muestra_Trearme

Sus entradas (VAR_INPUT) son las variables de rearme en formato TIME, mientras que sus salidas son la re-transformación de los valores a tipo INT (en minutos).

2 MONITORIZACIÓN DE PARÁMETROS ELÉCTRICOS

La programación de las funciones de medición de las variables del suministro eléctrico pasará por la configuración de los bloques de función que permiten, por un lado, seleccionar el tipo de red eléctrica y los parámetros de conversión en caso de utilizar transformadores de intensidad, como seleccionar las variables que nos interesa.

Los programas correspondientes a este apartado se añaden al programa de gestión ya comentado, formando parte del mismo proyecto de PC-Worx:

Grupo POU	Tipo	Nombre	Lenguaje IEC
Medicion	Programa	Medicion_Acometida	FBD
Medicion	Programa	Medicion_P2_Fuerza	FBD
Medicion	Programa	Medicion_P2_Alumb	FBD
Medicion	Programa	Medicion_P1_Fuerza	FBD
Medicion	Programa	Medicion_P1_Alumb	FBD
Medicion	Programa	Medicion_PB_Fuerza	FBD
Medicion	Programa	Medicion_PB_Alumb	FBD
Medicion	Programa	Medicion_PSS_Fuerza	FBD
Medicion	Programa	Medicion_PSS_Alumb	FBD

Tabla 2.1 - Programas medición

2.1 SOFTWARE UTILIZADO

Las herramientas informáticas utilizadas en esta parte son las siguientes:

- **PCWorx 6.30.1202:** Para la programación de PLCs y dispositivos de la firma Phoenix Contact.
- **AX-OPC Server 3.00:** Se trata del servidor OPC-DA de Phoenix Contact, que recoge las variables publicadas por el programa de control mediante el protocolo estandarizado OPC-DA, para poder ser utilizadas por otros clientes (Atvise, en nuestro caso).
- **Atvise 2.5.8:** Es un paquete SCADA basado en web, concretamente en tecnología HTML5, que permite recrear las pantallas de operador en multitud de dispositivos siempre y cuando posean navegadores web compatibles con este estándar.
- **Acron 7.3 SP5:** Software de bases de datos para uso en plantas industriales que permite la edición avanzada de informes y gráficas a partir de los históricos. Se conecta por OPC-UA a Atvise para la lectura de variables.

2.2 LIBRERÍAS

- **AsynCom_V1_02_610200:** Librería que habilita la comunicación asíncrona para ciertos dispositivos. Sirve como dependencia de la librería de medición, en este caso.
- **PowerMeasurement_V1_06:** Librería que contiene los bloques que permiten la configuración del módulo IB IL PM 3P/N/EF-PAC.

2.3 RELACIÓN DE TAREAS

Los programas correspondientes a la medición de consumos eléctricos están contenidos en la siguiente tarea:

Tarea	Tipo	Intervalo
MEDICION	CYCLIC	100 ms

Tabla 2.2 - Relación de tareas medición

2.4 RELACIÓN DE VARIABLES

Las variables generales utilizadas en el programa de medición siguen la siguiente estructura lógica:

- Alumb_*planta_medición*: definen la variable a medir, dentro del circuito de *Alumbrado* de la planta correspondiente.
- Fuerza_*planta_medición*: definen la variable a medir, dentro del circuito de *Fuerza* de la planta correspondiente.
- Acometida_*medición*: definen la variable a medir, en la acometida general del edificio.

El resto de variables del programa sirven para activar los bloques de función y establecer los parámetros a su valor correspondiente. La lista completa de las mismas se adjunta en el documento Nº4: “Código Fuente”, dentro del presente Proyecto.

2.5 MONITORIZACIÓN Y REGISTRO DE PARÁMETROS ELÉCTRICOS

En este apartado se explica la configuración y la lectura de variables por medio de los dispositivos IB IL PM 3P/N/EF-PAC, gracias a la librería *PowerMeasurement*:

2.5.1 Programas de medición

Implementados en FBD, existen tanto para la parte de Alumbrado, como Fuerza y la Acometida general. Su función es configurar los bloques explicados a continuación, necesarios para dar como salida las variables de tensiones, corrientes, potencias, etc. que se enviarán por OPC al SCADA.

2.5.2 Bloque funcional PM_3P_N_EF_V1_06

Sirve para establecer la comunicación con el módulo de medición, diciéndole al programa de control cómo debe comunicarse con el mismo:

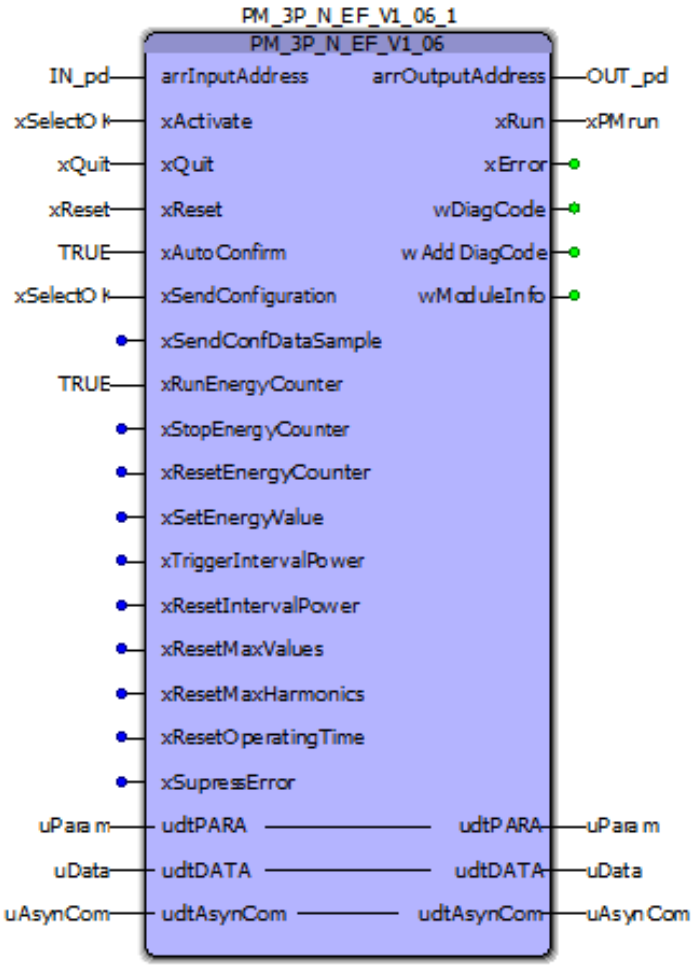

Imagen 2.1 - Bloque PM_3P_N_EF_V1_06

VAR_INPUT (entradas)	Tipo	Descripción
arrInputAddress	PM_ARR_W_0_11	IN Datos de proceso
xActivate	BOOL	Activación del bloque
xQuit	BOOL	ACK de los mensajes de error hasta el momento
xReset	BOOL	Reseteo del bloque
xAutoConfirm	BOOL	ACK automático
xSendConfiguration	BOOL	Flanco ascendente: envío de parámetros del bloque PM_Configuration al módulo
xSendConfDataSample	BOOL	Flanco ascendente: envío de parámetros del bloque PM_ConfDataSample al módulo
xRunEnergyCounter	BOOL	Activa medición
xStopEnergyCounter	BOOL	Detiene medición
xResetEnergyCounter	BOOL	Resetea medición
xSetEnergyValue	BOOL	Flanco ascendente: resetea el medidor con la información del bloque PM_SetEnergyValue
xTriggerInternalPower	BOOL	Flanco ascendente: medición de potencias por intervalos según bloque PM_IntervalPower
xResetIntervalPower	BOOL	Resetea intervalos de potencias
xResetMaxValues	BOOL	Resetea valores máximos
xResetMaxHarmonics	BOOL	Resetea armónicos
xResetOperatingTime	BOOL	Resetea tiempos de operación
xSupressError	BOOL	Resetea mensajes de errores

Tabla 2.3 - Entradas bloque PM_3P_N_EF_V1_06

VAR_OUTPUT (salidas)	Tipo	Descripción
arrOutputAddress	PM_ARR_W_0_11	OUT Datos de Proceso
xRun	BOOL	Bloque activo, configuración OK
xError	BOOL	(TRUE) Se ha producido un error
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código extendido de diagnóstico
wModuleInfo	WORD	ID y firmware del módulo

Tabla 2.4 - Salidas bloque PM_3P_N_EF_V1_06

VAR_INPUT _OUTPUT	Tipo	Descripción
udtPARA	PM_UDT_PARA	Estructura que contiene los parámetros de configuración del módulo
udtDATA	PM_UDT_DATA	Estructura para el intercambio de datos

Tabla 2.5 - Input/output bloque PM_3P_N_EF_V1_06

2.5.3 Bloque funcional PM_Configuration

Su función es la de escribir los parámetros de configuración del módulo en la estructura udtPARA:

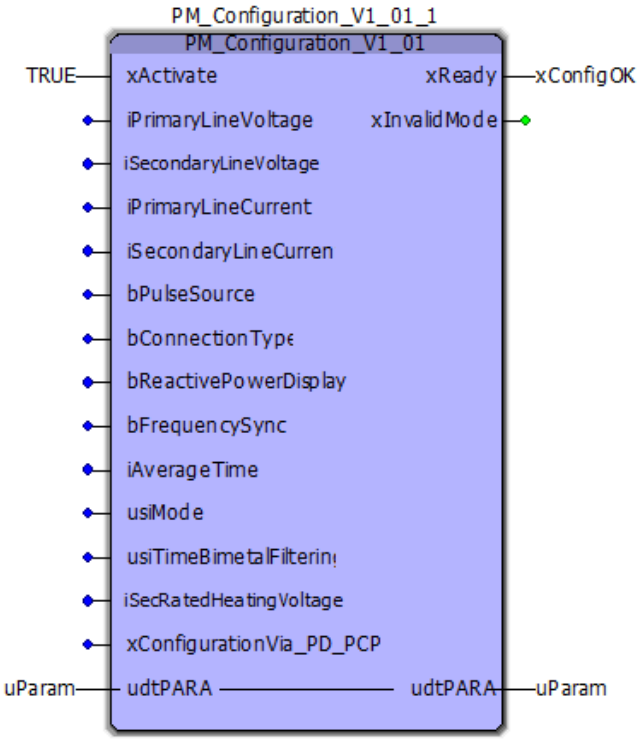

Imagen 2.2 - Bloque PM_Configuration

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	(TRUE) Bloque activo
iPrimaryLineVoltage	INT	Tensión de medición primario (si uso transformador de tensión)
iSecondaryLineVoltage	INT	Tensión de medición secundario (si uso transformador de tensión)
iPrimaryLineCurrent	INT	Corriente del primario (si uso transformador de corriente)
iSecondaryLineCurrent	INT	Corriente del secundario (si uso transformador de corriente)
bPulseSource	BYTE	Conmuta entre diversos modos de medición de potencia - Dejar por defecto a 0 -
bConnectionType	BYTE	Selección del tipo de línea (valor HEX, ver documentación de la librería)
bReactivePowerDisplay	BYTE	Visualización de la potencia reactiva (valor HEX, ver documentación de la librería)
bFrequencySync	BYTE	Frecuencia (valor HEX, ver documentación de la librería)
iAverageTime	INT	Opciones de medición de onda senoidal

usiMode	USINT	Modo de operación (Default 0: Basic measured values)
usiTimeBimetalFiltering	USINT	Opciones de tiempo para el filtrado
iSecRatedHeatingVoltage	INT	Opciones del secundario
xConfigurationVia_PD_PCP	BOOL	Permite seleccionar el envío de la configuración al módulo vía PD (datos de proceso) o PCP

Tabla 2.6 - Entradas bloque PM_Configuration

VAR_OUTPUT (salidas)	Tipo	Descripción
xReady	BOOL	(TRUE) Parámetros enviados correctamente
xInvalidMode	BOOL	Modo de operación no válido

Tabla 2.7 - Salidas bloque PM_Configuration

VAR_INPUT _OUTPUT	Tipo	Descripción
udtPARA	PM_UDT_PARA	Estructura que contiene los parámetros de configuración del módulo

Tabla 2.8 - Input/output bloque PM_Configuration

2.5.4 Bloque funcional PM_Select_PD

Tras la parametrización del módulo INLINE, es necesario indicarle qué valores nos interesa leer con el mismo. Para ello se utiliza el bloque *PM_Select_PD*, que realiza la llamada a los distintos bloques de lectura vía PD (Process Data):

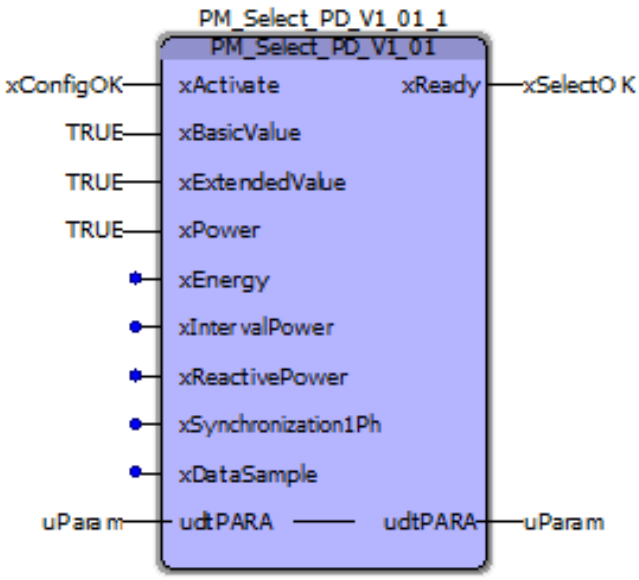

Imagen 2.3 - Bloque PM_Select_PD

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del bloque
xQuit	BOOL	ACK de los mensajes de error hasta el momento
xReset	BOOL	Reseteo del bloque
xBasicValue	BOOL	Activa lectura del bloque PM_BasicValue
xEntendedValue	BOOL	Activa lectura del bloque PM_ExtendedValue
xPower	BOOL	Activa lectura del bloque PM_Power
xEnergy	BOOL	Activa lectura del bloque PM_Energy
xIntervalPower	BOOL	Activa lectura del bloque PM_IntervalPower
xReactivePower	BOOL	Activa lectura del bloque PM_ReactivePower
xSynchronization1Ph	BOOL	Activa lectura del bloque PM_Sync1Ph
xDataSample	BOOL	Activa lectura del bloque PM_ConfDataSample

Tabla 2.9 - Entradas bloque PM_Select_PD

VAR_OUTPUT (salidas)	Tipo	Descripción
xReady	BOOL	(TRUE) Parámetros enviados correctamente

Tabla 2.10 - Salidas bloque PM_Select_PD

VAR_INPUT _OUTPUT	Tipo	Descripción
udtPARA	PM_UDT_PARA	Estructura que contiene los parámetros de configuración del módulo

Tabla 2.11 - Input/output bloque PM_Select_PD

2.5.5 Bloques de lectura: PM_BasicValue, PM_ExtendedValue, PM_Power

En esta sección se resume de forma general el funcionamiento de los bloques de lectura, cuya activación depende del anterior bloque *PM_Select_PD*. Su función es la de extraer las variables del suministro eléctrico que están almacenadas en la estructura *udtData*, para poder guardar su valor en nuevas variables de salida para ser utilizadas convenientemente en el SCADA y el software de históricos

En el caso que nos ocupa, los tres bloques usados son activados (Var. Input) por la señal de activación *xPM_Run*, que actúa como salida del bloque anterior *PM_3P_N_EF_V1_06*.

Las salidas de estos nuevos bloques serán, por tanto, las variables deseadas de tensiones, corrientes y potencias; que solo necesitarán ser adaptadas a las unidades deseadas.

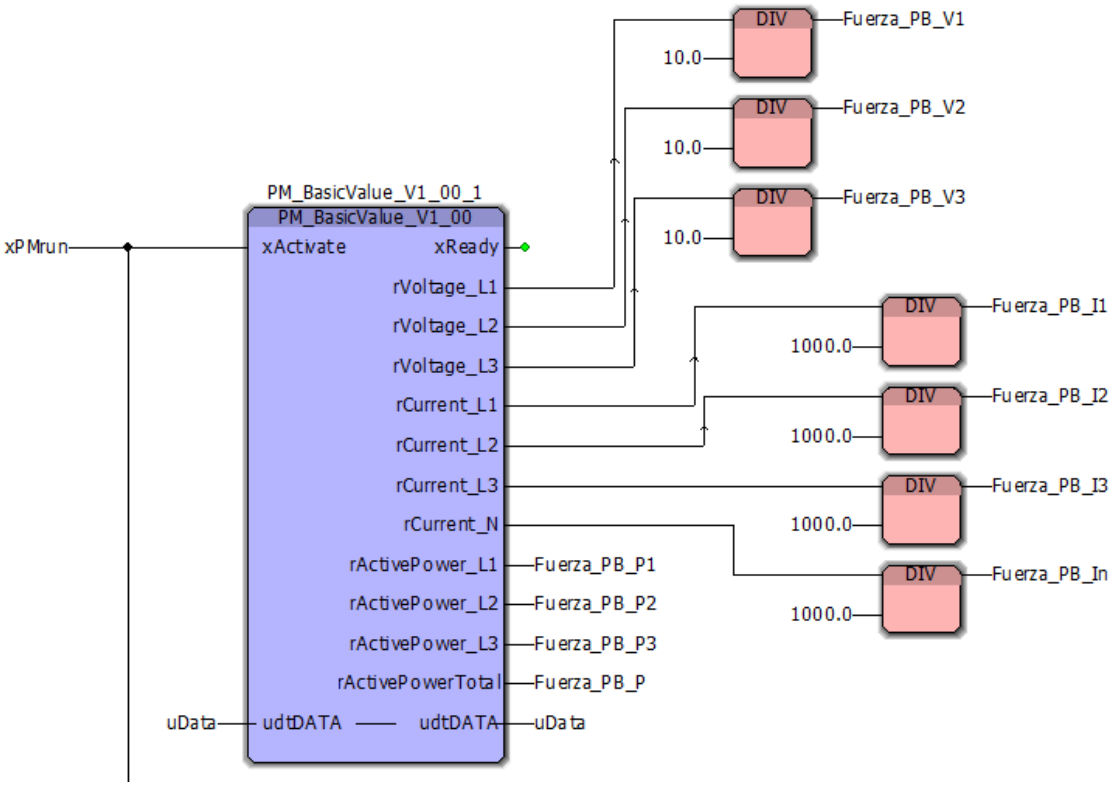

Imagen 2.4 - Ejemplo con bloques de medición librería PowerMeasurement

3 FUNCIONALIDADES ADICIONALES

Se conoce por funcionalidades adicionales a las capacidades de los controladores de Phoenix Contact utilizadas para la comunicación con los dispositivos EMPro, además de las características de envío de correo electrónico y datalogger.

3.1 CONFIGURACIÓN DE MEDIDORES DE ENERGÍA EMPRO

En este apartado se detallan las particularidades de configuración de las comunicaciones entre los dispositivos de la gama EMpro de Phoenix Contact, para la medición de parámetros del suministro eléctrico, y los equipos de control.

El objetivo de esta implementación es doble: por un lado mostrar la forma adecuada de establecer las comunicaciones entre un EMpro MA600 y un PLC de la familia ILC-191 ME/AN mediante los protocolos Modbus TCP/RS-485, y por el otro, servir como referencia para poder realizar programas de mayor envergadura que partan de esta base.

Los programas que forman parte de este ejemplo son:

Grupo POU	Tipo	Nombre	Lenguaje IEC
-	Programa	SQL_Server	FBD
-	Programa	Selector	ST
-	Programa	ModBus_TCP	FBD
-	Programa	ModBus_485	FBD

Tabla 3.1 - Programas configuración EMpro

3.1.1 Software utilizado

Las herramientas informáticas utilizadas en esta parte son las siguientes:

- **PCWorx 6.30.1202:** Para la programación de PLCs y dispositivos de la firma Phoenix Contact.
- **PCWorx SRT 1.1**: Es el simulador de PLCs de Phoenix Contact que corre en un PC Windows, se ha usado para realizar pruebas menores cuando no está disponible el equipo físico.
- **Microsoft SQL Server 2005:** Se trata del software de gestión de bases de datos SQL de Microsoft, siendo la versión que da mayor soporte a las librerías de programación de los PLCs de la gama ILC de Phoenix Contact.

3.1.2 Librerías

- **ILCME_ModBus_V1 _01:** Librería que habilita la comunicación Modbus en los PLCs de la serie ILC 191 ME. Contiene tanto los bloques de función para la parametrización de las comunicaciones, como el acceso a las distintas funciones y registros.
- **Modbus_V2_01:** Se trata de la librería que habilita de forma general las comunicaciones Modbus para todos los controladores de la marca que soporten dichos protocolos.
- **DBFL_SQL_V1_18:** Esta librería permite, gracias a la colección de bloques funcionales que incorpora, la escritura de valores de las variables de proceso deseadas en bases de datos Microsoft SQL Server / MySQL.

La explicación de alguno de los bloques utilizados que contienen dichas librerías se resumen de forma general en los apartados correspondientes. Para mayor información, se adjuntan los documentos de ayuda que acompañan a cada librería.

3.1.3 Relación de tareas

Tarea	Tipo	Intervalo
STD_TSK	DEFAULT	-

Tabla 3.2 - Relación de tareas EMpro

3.1.4 Relación de variables

Las variables generales utilizadas en el programa de ejemplo EMpro siguen la siguiente estructura lógica:

- aRead*variable*_ModBus: array de lectura de la variable deseada desde los registros de Modbus, utilizando una comunicación RS-485.
- aRead*variable*_TCP: array de lectura de la variable deseada desde los registros de Modbus, utilizando una comunicación Ethernet.
- r*variable*_ModBus: variable de salida resultante de los registros Modbus, utilizando una comunicación RS-485.
- r*variable*_TCP: variable de salida resultante de los registros Modbus, utilizando una comunicación Ethernet.
- r*variable*_Actual: variable de escritura en la tabla SQL, dependiendo del valor de *xSelector*, se tomará el valor obtenido por RS-485/Modbus TCP.

El resto de variables del programa sirven para activar los bloques de función y establecer los parámetros a su valor correspondiente. La lista completa de las mismas se adjunta en el documento Nº4: "Código Fuente", dentro del presente Proyecto.

3.1.5 Programa ModBus_485

Implementado en lenguaje de bloques de función (FBD), se encarga de la toma de variables a partir de los arrays de datos que el PLC toma utilizando un protocolo Modbus con conexión RS-485 desde el medidor EEM-MA600, a las bornas de comunicación del controlador ILC-191 ME/AN.

Dicha comunicación se configura en el bloque *MB191_485_Para_V1_00*:

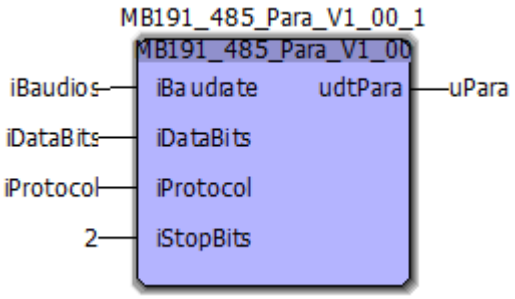

Imagen 3.1 - Bloque MB191_485_Para_V1_00

VAR_INPUT (entradas)	Tipo	Descripción
iBaudrate	INT	Velocidad de la transmisión (en baudios)
iDataBits	BYTE	Número de bits y paridad
iProtocol	BYTE	Utilización de RS-422/RS-485
iStopBits	BYTE	Número de bits de parada

Tabla 3.3 - Entradas bloque MB191_485_Para_V1_00

VAR_OUTPUT (salidas)	Tipo	Descripción
udtPara	MB191_udtILC1xx_RS485	Estructura que contiene los datos de parametrización

Tabla 3.4 - Salidas bloque MB191_485_Para_V1_00

El array que contiene los datos de parametrización ha de disponerse como entrada del bloque de activación, *MB191_485_T1_V1_00*:

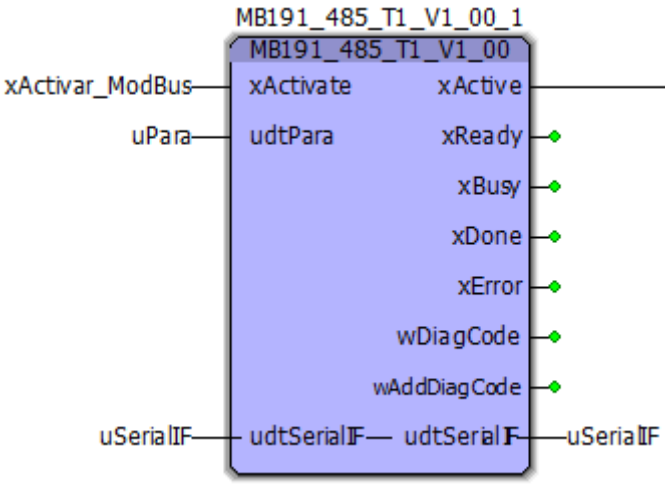

Imagen 3.2 - Bloque MB191_485_T1_V1_00

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación de la comunicación
udtPara	MB191_udtILC1xx_RS485	Estructura que contiene los datos de parametrización

Tabla 3.5 - Entradas bloque MB191_485_T1_V1_00

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo
xReady	BOOL	Bloque listo, comunicación activa
xBusy	BOOL	Bloque ocupado, comunicación en proceso
xDone	BOOL	Comunicación correcta
xError	BOOL	Si activo, se ha producido un error de comunicación
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código extendido de diagnóstico

Tabla 3.6 - Salidas bloque MB191_485_T1_V1_00

VAR_INPUT _OUTPUT	Tipo	Descripción
udtSerialIF	MB191_udtSerial_IF_T1	Estructura de intercambio para la comunicación serie

Tabla 3.7 - Input/output bloque MB191_485_T1_V1_00

Una vez parametrizada y activada la comunicación, se dispone el bloque *MB191_RTU_Gateway_V1_00* a modo de *Gateway* (enlace):

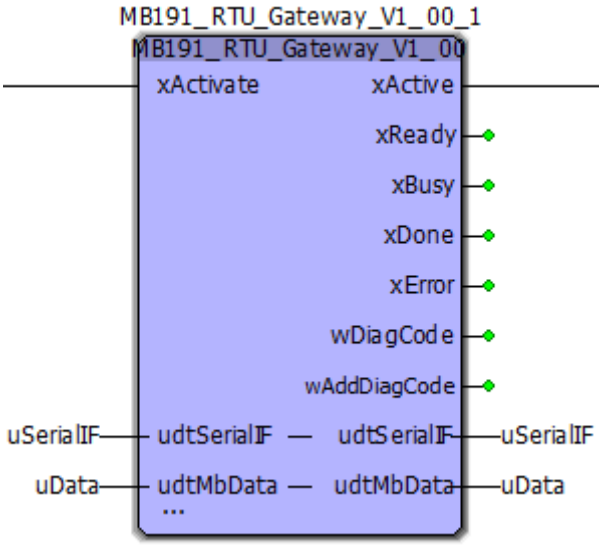

Imagen 3.3 - Bloque MB191_RTU_Gateway_V1_00

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del gateway

Tabla 3.8 - Entradas bloque MB191_RTU_Gateway_V1_00

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo
xReady	BOOL	Bloque listo, comunicación activa
xBusy	BOOL	Bloque ocupado, comunicación en proceso
xDone	BOOL	Comunicación correcta
xError	BOOL	Si activo, se ha producido un error de comunicación
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código extendido de diagnóstico

Tabla 3.9 - Salidas bloque MB191_RTU_Gateway_V1_00

VAR_INPUT _OUTPUT	Tipo	Descripción
udtSerialIF	MB191_udtSerial_IF_T1	Estructura de intercambio para la comunicación serie
udtMbData	MB191_udtMB_RTU_Data	Estructura de intercambio entre el gateway y los bloques de toma de datos

Tabla 3.10 - Input/output bloque MB191_RTU_Gateway_V1_00

Ya establecida la comunicación, y confirmado su correcto funcionamiento, necesitaremos colocar en el programa tantos bloques como datos queramos obtener mediante este protocolo.

En nuestro caso, utilizamos la función FC3 (Read Analog Output Holding Registers) del protocolo Modbus, incorporando su lectura al programa por medio del bloque *MB191_RTU_FC3FC4*:

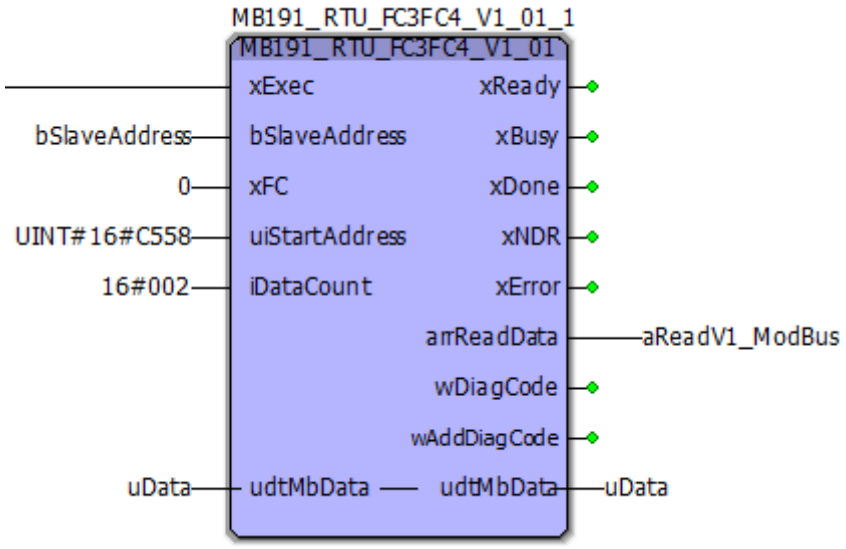

Imagen 3.4 - Bloque MB191_RTU_FC3FC4

VAR_INPUT (entradas)	Tipo	Descripción
xExec	BOOL	Flanco para el inicio de la comunicación
bSlaveAddress	BYTE	Dirección Modbus del esclavo (1…247)
xFC	BOOL	Selección entre FC3 (FALSE)/FC4 (TRUE)
uiStartAddress	UINT	Dirección de inicio lectura datos
iDataCount	INT	Número de datos a leer (1…125)

Tabla 3.11 - Entradas bloque MB191_RTU_FC3FC4

La forma de trabajo de este bloque, a la vista de las entradas del mismo, consistirá en la lectura de los datos contenidos en dicha dirección, dentro de los registros de Modbus, con la longitud *iDataCount* dada. Estos registros vienen indicados en las hojas de características de los dispositivos EMpro. De esta forma, y colocando tantos bloques como sea necesario, podemos leer los valores de tensión, corriente y potencia deseados.

VAR_OUTPUT (salidas)	Tipo	Descripción
xReady	BOOL	Bloque listo, comunicación activa
xBusy	BOOL	Bloque ocupado, comunicación en proceso
xDone	BOOL	Comunicación correcta
xNDR	BOOL	Lectura satisfactoria, valores actualizados correctamente
xError	BOOL	Si activo, se ha producido un error de comunicación
arrReadData	MB191_arrWord_0_125	Array de lectura Modbus
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código extendido de diagnóstico

Tabla 3.12 - Salidas bloque MB191_RTU_FC3FC4

VAR_INPUT_OUTPUT	Tipo	Descripción
udtMbData	MB191_udtMB_RTU_Data	Estructura de intercambio entre el gateway y los bloques de toma de datos

Tabla 3.13 - Input/output bloque MB191_RTU_FC3FC4

Para el acceso a los datos obtenidos, debemos tomar las posiciones necesarias del array y colocarlas en una variable nueva, que transformaremos de tipo WORD a INT/REAL, para su dimensionamiento a las unidades requeridas.

3.1.6 Programa ModBus_TCP

Implementado en lenguaje de bloques de función (FBD), se encarga de la lectura de variables a partir de los arrays de datos que el PLC toma utilizando un protocolo Modbus/TCP con conexión Ethernet desde el medidor EEM-MA600, mediante un cable de red estándar a la entrada del autómata.

En este caso, la comunicación se configura de manera más sencilla que en el caso de optar por una conexión serie RS-485. De la misma forma que en el caso anterior, es necesario introducir los parámetros de conexión del cliente (en este caso, el dispositivo EMpro) antes de proceder con la toma de datos.

Para la configuración de la conexión utilizamos el bloque *MB_TCP_Client_V2_00*, que se encarga de acceder a la dirección desde la que se van a obtener los datos, e informarnos del estado de la misma:

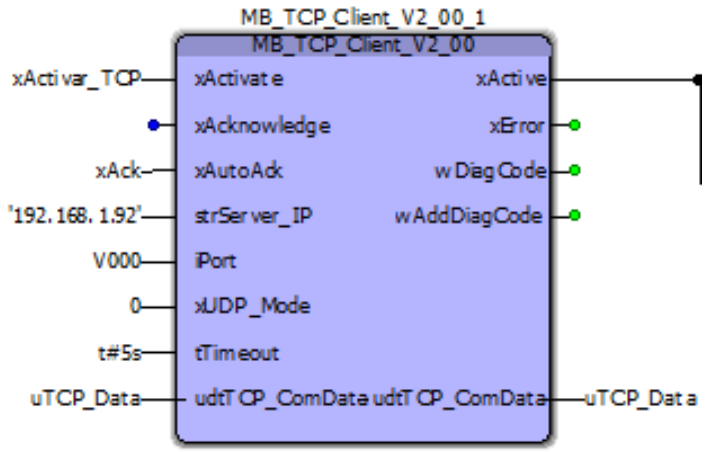

Imagen 3.5 - Bloque MB_TCP_Client_V2_00

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del bloque
xAcknowledge	BOOL	(TRUE) Reset automático en caso de errores
xAutoAck	BOOL	(TRUE) ACK automático de la conexión
strServer_IP	STRING	Cadena con la dirección IP del servidor en formato 'xxx.xxx.xxx.xxx'
iPort	INT	Puerto TCP
xUDP_Mode	BOOL	(FALSE) Usa TCP / (TRUE) Usa UDP
tTimeout	TIME	Timeout del servidor

Tabla 3.14 - Entradas bloque MB_TCP_Client_V2_00

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo
xError	BOOL	Si activo, se ha producido un error de comunicación
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código extendido de diagnóstico

Tabla 3.15 - Salidas bloque MB_TCP_Client_V2_00

VAR_INPUT_OUTPUT	Tipo	Descripción
udt_TCP_ComData	MB2_COM_UDT_COMMUNICATION_V2	Estructura de comunicación TCP

Tabla 3.16 - Input/output bloque MB_TCP_Client_V2_00

Dicha estructura de comunicación se comunica con los bloques de acceso a las funciones de Modbus. Idénticamente al caso de la comunicación serie RS-485, necesitamos acceder a la función FC3 para la toma de parámetros correspondientes al suministro eléctrico, recogidos por el EMpro. Para ello, existe el bloque *MB_TCP_FC3_V2_01*, siendo equivalentes los bloques que corresponden al resto de funciones:

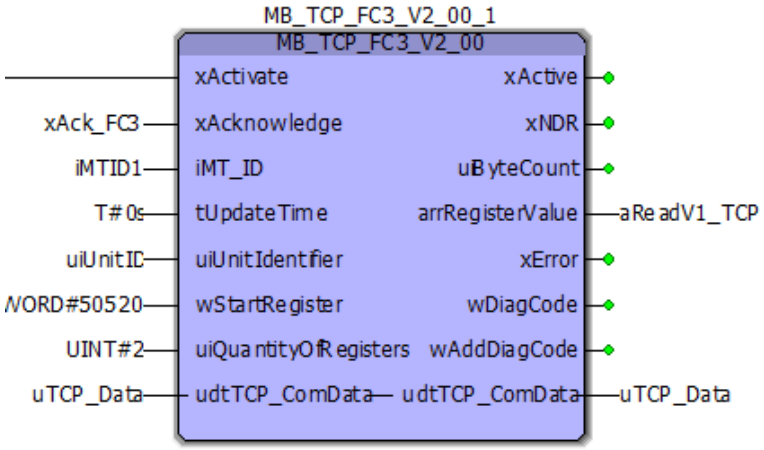

Imagen 3.6 - Bloque MB_TCP_FC3_V2_01

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del bloque
xAcknowledge	BOOL	(TRUE) Reset automático en caso de errores
iMT_ID	INT	(1-10) Parámetro estático que representa el orden de conexión de los bloques.
tUpdateTime	TIME	Tiempo de *delay* entre lecturas
uiUnitIdentifier	UINT	Identificador, dirección Modbus
wStartRegister	WORD	Dirección de inicio del registro de lectura
uiQuantityOfRegisters	UNIT	Número de registros a leer (1-125)

Tabla 3.17 - Entradas bloque MB_TCP_FC3_V2_01

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo
xNDR	BOOL	Los datos se han recibido correctamente
uiByteCount	UINT	Número de bytes recibidos
arrRegisterValue	MB2_TCP_ARR_W_1_125	Array que contiene los valores leídos por el registro.
xError	BOOL	Si activo, se ha producido un error de comunicación
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código extendido de diagnóstico

Tabla 3.18 - Salidas bloque MB_TCP_FC3_V2_01

VAR_INPUT_OUTPUT	Tipo	Descripción
udt_TCP_ComData	MB2_COM_UDT_COMMUNICATION_V2	Estructura de comunicación TCP

Tabla 3.19 - Input/output bloque MB_TCP_FC3_V2_01

El tratamiento de los datos se efectúa de la misma forma que en el caso de usar una conexión RS-485, debiendo extraer los valores deseados de su correspondiente posición en el array de salida para su tratamiento/conversión de unidades.

3.1.7 Programa Selector

Implementado en texto estructurado (ST), se trata de un simple programa que se encarga de conmutar entre la escritura de valores en la base de datos de ejemplo, de entre las dos fuentes de captura de datos: Modbus TCP y Modbus basado en RS-485.

Posee la función básica de copiar en las variables *rvalor_Actual*, el contenido de los registros que le llegan bien por TCP o RS-485, para escribir en la tabla correspondiente los deseados según el valor de la variable booleana *xSelector*.

3.1.8 Programa SQL_Server

Su objetivo es el de ofrecer un ejemplo básico de escritura de los valores importados por el programa desde el dispositivo EMpro, en una tabla SQL Server (ya existente). Para ello, se utiliza la siguiente sentencia SQL en el bloque de escritura, la cual se completa con los de entrada de valores para cada correspondiente tipo de variable:

'INSERT INTO Prueba_ModBus (Time,V1,I1,F,S,P,Q,PF) VALUES('

El modo de trabajo de los bloques de esta librería consiste, en primer lugar, en acceder a la tabla en la cual queremos escribir los registros dentro de la base de datos deseada, para lo que utilizamos el bloque de conexión *DBFL_TSQL_ACCESS_V1_15*, cuyo funcionamiento se detalla:

VAR_INPUT (entradas)	Tipo	Descripción
IP_Activate	BOOL	(TRUE) Se establece conexión TCP
DB_Activate	BOOL	Flanco para envío de secuencia SQL a BBDD
DB_User	STRING	Nombre de usuario para acceso a base de datos
DB_Password	STRING	Contraseña del usuario
IP_Address	STRING	Cadena que contiene la IP del servidor BBDD
IP_Port	STRING	Puerto de la base de datos
DB_Name	STRNG	Nombre de la base de datos
Time_Out	TIME	Timeout errores SQL
No_KeepAlive	BOOL	(TRUE) Desactiva TCP/IP-KeepAlive-Frames

Tabla 3.20 - Entradas bloque DBFL_TSQL_ACCESS_V1_15

VAR_OUTPUT (salidas)	Tipo	Descripción
TCP_Ready	BOOL	(TRUE) Conexión TCP establecida
SQL_Ready	BOOL	Comando SQL enviado
SQL_Done	BOOL	Comando SQL ejecutado correctamente
SQL_Server_Info	DBFL_UDT_MSSQL_Server_Info	Proporciona información acerca del estado de la comunicación con el servidor SQL
Error	BOOL	(TRUE) Se ha producido un error en la comunicación
TCP_Status	DINT	Código de error TCP (ver documentación)
SQL_Status	DINT	Código de error SQL (ver documentación)
SQL_ErrMsg	DBFL_UDT_STRING25	Texto de error correspondiente al SQL_STATUS
Rcv_Size	DINT	Tamaño de los datos recibidos
Row_Cnt	DINT	Número de filas afectadas
Col_Cnt	DINT	Número de columnas afectadas

Tabla 3.21 - Salidas bloque DBFL_TSQL_ACCESS_V1_15

VAR_INPUT_OUTPUT	Tipo	Descripción
SQL_IN	DBFL_ARR_BYTE_0_1439	Array que contiene comando SQL a enviar
RCV_BUFFER	DBFL_ARR_BYTE_0_10219	Contiene los datos recibidos desde el servidor (se evalúan con el bloque SQL_Decode)

Tabla 3.22 - Input/output bloque DBFL_TSQL_ACCESS_V1_15

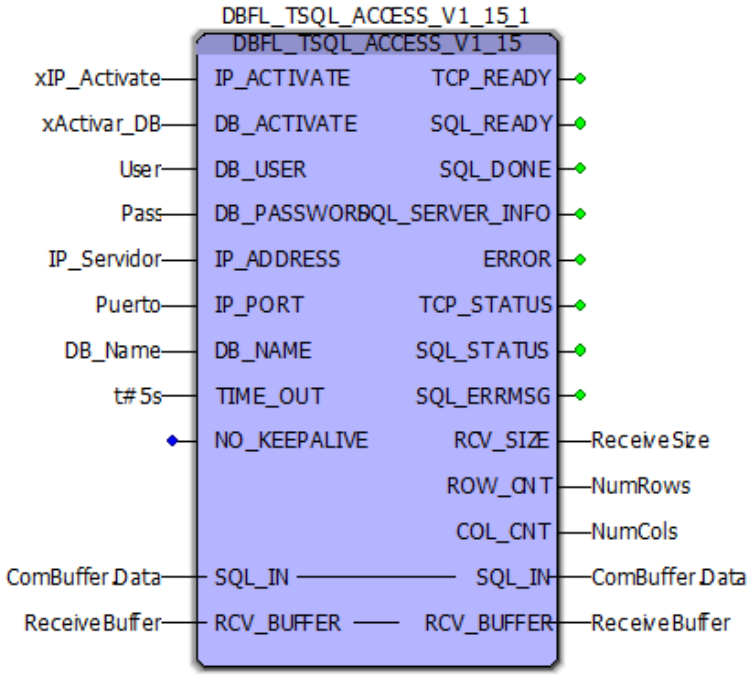

Imagen 3.7 - Bloque DBFL_TSQL_ACCESS_V1_15

Posteriormente al establecimiento de la comunicación, pasamos a formar el comando SQL deseado, encadenando cuantos bloques sean necesarios según el tipo de dato que corresponda en cada caso.

El primer bloque que ha de formar parte de esta cadena será el que envíe la sentencia SQL al servidor, para luego ser completada con los valores deseados por los demás bloques que le siguen. Este primer bloque es *DBFL_StartComT2_V1_01*:

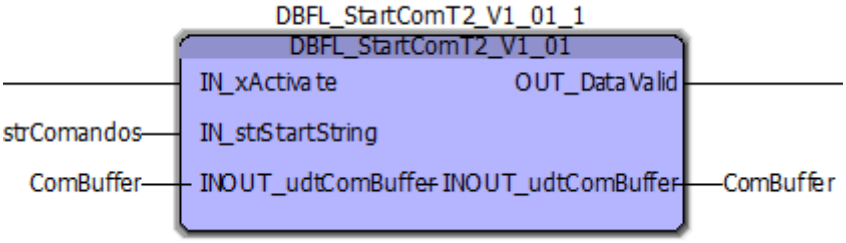

Imagen 3.8 - Bloque DBFL_StartComT2_V1_01

VAR_INPUT (entradas)	Tipo	Descripción
IN_xActivate	BOOL	Flanco que activa el envío de la sentencia introducida en IN_strStartString
IN_strStartString	STRING	Comando SQL de envío al servidor

Tabla 3.23 - Entradas bloque DBFL_StartComT2_V1_01

VAR_OUTPUT (salidas)	Tipo	Descripción
OUT_DataValid	BOOL	Flanco que confirma escritura de la cadena en el buffer

Tabla 3.24 - Salidas bloque DBFL_StartComT2_V1_01

VAR_INPUT_OUTPUT	Tipo	Descripción
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	Buffer que contiene el comando SQL

Tabla 3.25 - Input/output bloque DBFL_StartComT2_V1_01

En el caso que nos ocupa, en la entrada *IN_strStartString*, deberemos introducir el encabezado del comando SQL que vamos a enviar: **'INSERT INTO Prueba_ModBus (**, para ser luego completado por el resto de bloques.

Continuando con este ejemplo, para la introducción del valor temporal *Time*, el bloque que se ha de utilizar será *DBFL_DateTimeStr T2_V1_02*:

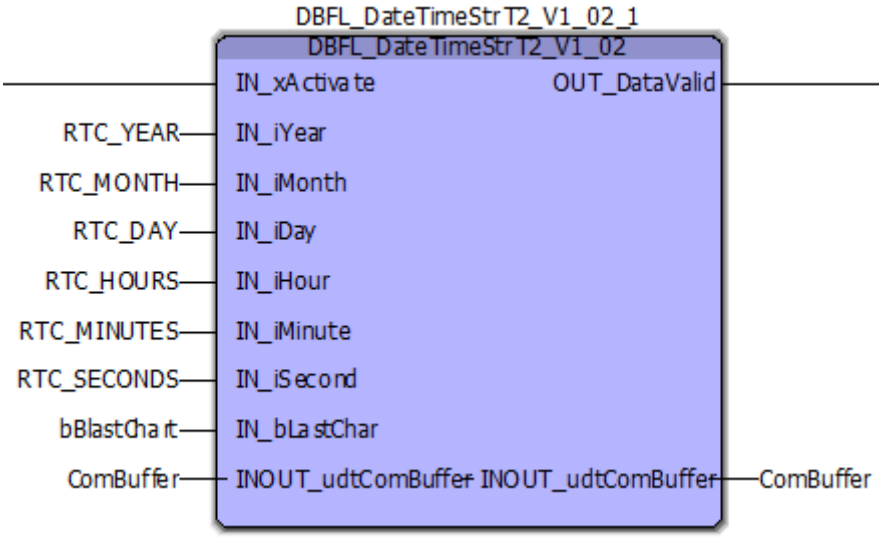

Imagen 3.9 - Bloque DBFL_DateTimeStr T2_V1_02

VAR_INPUT (entradas)	Tipo	Descripción
IN_xActivate	BOOL	Flanco que activa la escritura SQL del bloque
IN_iYear	INT	Año
IN_iMonth	INT	Mes
IN_iDay	INT	Día
IN_iHour	INT	Hora
IN_iMinute	INT	Minuto
IN_iSecond	INT	Segundo
IN_bLastChar	BYTE	Separador entre valores del comando SQL: ‘,’ : BYTE#44 (coma) ‘)’ : BYTE#41 (paréntesis de fin sentencia)

Tabla 3.26 - Entradas bloque DBFL_DateTimeStr T2_V1_02

VAR_OUTPUT (salidas)	Tipo	Descripción
OUT_DataValid	BOOL	Flanco que confirma escritura de la cadena en el buffer

Tabla 3.27 - Salidas bloque DBFL_DateTimeStr T2_V1_02

VAR_INPUT_OUTPUT	Tipo	Descripción
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	Buffer que contiene el comando SQL

Tabla 3.28 - Input/output bloque DBFL_DateTimeStr T2_V1_02

A la vista del bloque de ejemplo, estamos forzando a que para cada nueva entrada que se escribe en la tabla SQL, estamos añadiendo la hora y fecha en la que se encuentra el PLC, ya que dicho bloque accede a los registros internos RTC (Real Time Clock) del mismo.

Para finalizar esta plantilla de escritura de valores, utilizamos tantos bloques como sea necesario hasta completar el comando SQL. En este caso, que escribimos valores de tensiones, corrientes, potencias, etc., de tipo REAL, se encadenan bloques *DBFL_RealToComT2_V1_01*:

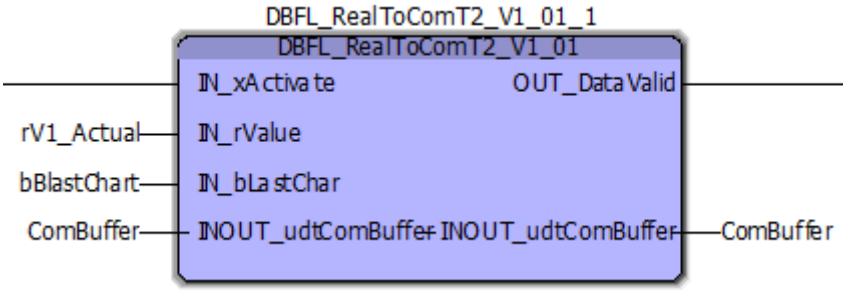

Imagen 3.10 - Bloque DBFL_RealToComT2_V1_01

VAR_INPUT (entradas)	Tipo	Descripción
IN_xActivate	BOOL	Flanco que activa la escritura SQL del bloque
IN_rValue	REAL	Valor a enviar de tipo REAL
IN_bLastChar	BYTE	Separador entre valores del comando SQL: ‘,’ : BYTE#44 (coma) ‘)’ : BYTE#41 (paréntesis de fin sentencia)

Tabla 3.29 - Entradas bloque DBFL_RealToComT2_V1_01

VAR_OUTPUT (salidas)	Tipo	Descripción
OUT_DataValid	BOOL	Flanco que confirma escritura de la cadena en el buffer

Tabla 3.30 - Salidas bloque DBFL_RealToComT2_V1_01

VAR_INPUT_OUTPUT	Tipo	Descripción
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	Buffer que contiene el comando SQL

Tabla 3.31 - Input/output bloque DBFL_RealToComT2_V1_01

De esta forma, podemos completar el comando SQL a enviar, introduciendo el caracter ‘,’ como elemento de separación entre bloques, reservando el paréntesis de cierre para el último de ellos.

Sirva este pequeño ejemplo para explicar la estructura básica que permite a los PLCs de Phoenix Contact la escritura de valores en tablas SQL Server ya existentes. Para la escritura de valores de otro tipo (INT, WORD, etc.), la estructura de dichos bloques específicos es equivalente a los aquí explicados.

3.2 ENVÍO DE CORREO ELECTRÓNICO / DATALOGGER

En este apartado se explican conjuntamente las funcionalidades de envío de correo electrónico y Datalogger soportadas por los controladores de Phoenix Contact, al guardar una estrecha relación en el ejemplo desarrollado, y por tanto, estar contenidas en el mismo proyecto PC-Worx.

El objetivo aquí es mostrar, por medio de un pequeño programa, la capacidad del PLC de recoger datos de proceso de distintos tipos de entre los más frecuentes (REAL, INT, BOOL, BYTE, etc. siendo el planteamiento similar para sus tipos derivados), y almacenarlos localmente, en la propia memoria del PLC, en forma de archivo *.csv* (*comma-separated values*).

Por último, se muestra también la configuración de bloques necesaria para subir dichos archivos generados a un servidor FTP, así como su envío a una cuenta de correo electrónico dada (de ahí que se utilice este ejemplo conjunto para enviar por e-mail uno de estos archivos).

Los programas que corresponden a este apartado son:

Grupo POU	Tipo	Nombre	Lenguaje IEC
-	Programa	Declara_byte	ST
-	Programa	Detec_cambios	ST
-	Programa	correo	FBD
-	Programa	DataLog	FBD

Tabla 3.32 - Programas Correo & DataLogger

3.2.1 Software utilizado

Las herramientas informáticas utilizadas en esta parte son las siguientes:

- **PCWorx 6.30.1202:** Para la programación de PLCs y dispositivos de la firma Phoenix Contact.
- **Filezilla 3.13.1**: Permite crear un servidor FTP de prueba, para comprobar la correcta subida de archivos para esta funcionalidad.

3.2.2 Librerías

- **DataLogger_V0_18Eval_08:** Agrupa los bloques necesarios para la recogida y procesamiento de datos de la función DataLogger, en formato binario, ASCII o directamente con salida en un archivo *.csv*. También ofrece la posibilidad de configurar parámetros de la memoria del PLC para poder guardar estos archivos, o subirlos a un servidor FTP.
- **IT_Library_1_32:** Librería que contiene diversas funciones de comunicación e IT para los dispositivos de Phoenix Contact, como la configuración DNS y DHCP, protocolos SMTP para el envío de correo electrónico, etc.

La explicación de alguno de los bloques utilizados que contienen dichas librerías se resumen de forma general en los apartados correspondientes. Para mayor información, se adjuntan los documentos de ayuda que acompañan a cada librería.

3.2.3 Relación de tareas

Tarea	Tipo	Intervalo
STD_TSK	DEFAULT	-

Tabla 3.33 - Relación de tareas Correo & DataLogger

3.2.4 Relación de variables

Las variables generales utilizadas en el programa de ejemplo Correo & DataLogger siguen la siguiente estructura lógica:

- iInt, rReal, xBool*x*, bByte: Representan los tipos básicos de datos con los que trabajaremos en el ejemplo.

El resto de variables del programa sirven para activar los bloques de función y establecer los parámetros a su valor correspondiente. La lista completa de las mismas se adjunta en el documento Nº4: “Código Fuente”, dentro del presente Proyecto.

3.2.5 Programa Declara_byte

Contiene una simple asignación de valores para formar una variable tipo *byte* a partir de una serie de variables booleanas, para poder así desarrollar el correspondiente ejemplo.

3.2.6 Programa Detec_cambios

Su función es la de detectar cambios, por parte del usuario, en los valores de alguna de las variables que vamos a escribir, generando además un flanco positivo que activa la escritura.

3.2.7 Programa correo

Sirve como ejemplo para el envío de archivos adjuntos por e-mail utilizando el protocolo SMTP, gracias a los parámetros de configuración del bloque *SMTP_Client_V1_16*:

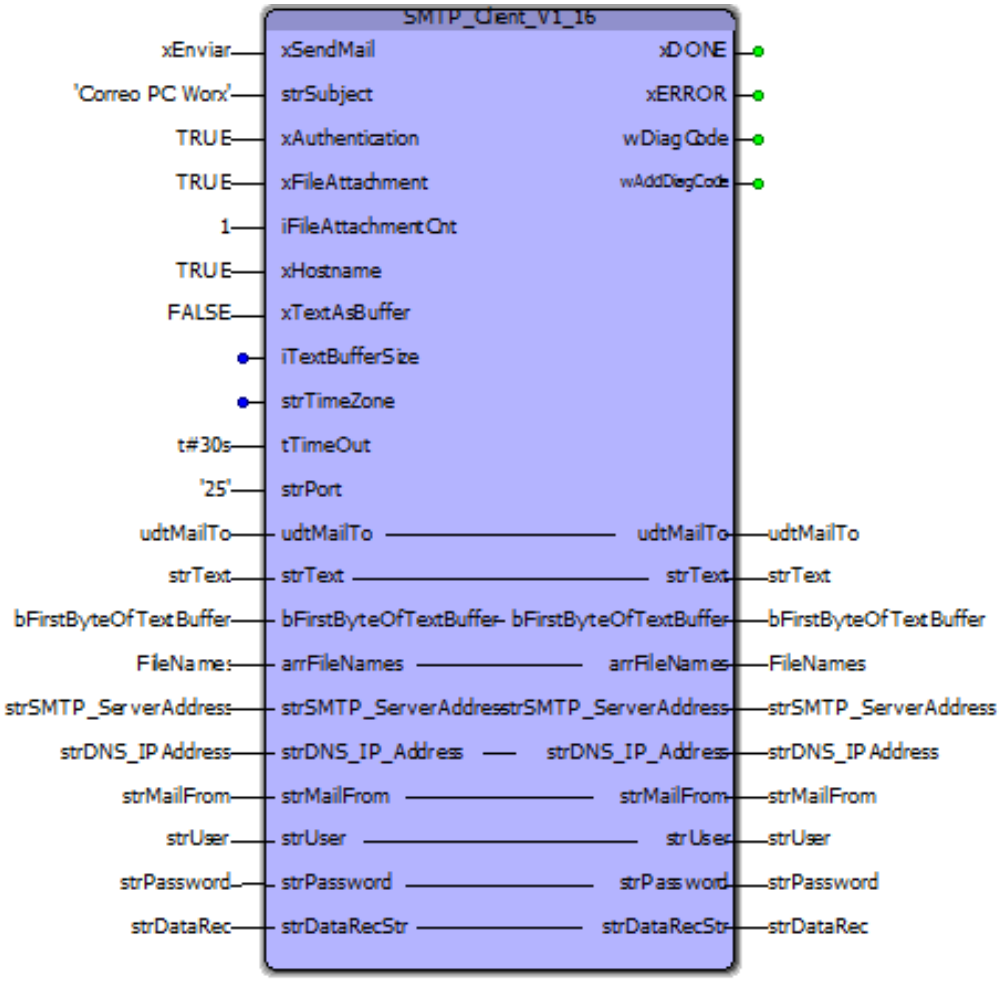

Imagen 3.11 - Bloque SMTP_Client_V1_16

VAR_INPUT (entradas)	Tipo	Descripción
xSendMail	BOOL	A la llegada de un flanco de subida, envía un correo
strSubject	STRING	Cadena que contiene el asunto del correo
xAuthentication	BOOL	A TRUE, requiere autenticación del usuario
xFileAttachment	BOOL	Envío de archivos adjuntos (TRUE)
iFileAttachmentCnt	INT	Número de adjuntos
xHostname	BOOL	Permite conmutar entre la utilización de la dirección web del servidor (TRUE) o su IP (FALSE)
xTextAsBuffer	BOOL	Envío de texto como buffer de caracteres (TRUE)
iTextBufferSize	INT	Tamaño del buffer
strTimeZone	STRING	Zona horaria
tTimeOut	TIME	Tiempo de espera a la conexión (Timeout)
strPort	STRING	Cadena que contiene el puerto de conexión

Tabla 3.34 - Entradas bloque SMTP_Client_V1_16

VAR_OUTPUT (salidas)	Tipo	Descripción
xDONE	BOOL	Marca que se activa una vez enviado el correo
xERROR	BOOL	Error en el envío si variable a TRUE
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código de diagnóstico (extendido)

Tabla 3.35 - Salidas bloque SMTP_Client_V1_16

VAR_INPUT_OUTPUT	Tipo	Descripción
udtMailTo	SMTP_UDT_EmailAddress	Direcciones de destino
strText	STRING	Texto del cuerpo del correo
bFirstByteOfTextBuffer	BYTE	Primer byte del buffer que se envía
arrFileNames	SMTP_ARR_STR_1_10	Estructura con los nombres y ubicación de los adjuntos a enviar
strSMTP_ServerAddress	STRING	Dirección del servidor de correo
strDNS_IPAddress	STRING	DNS
strMailFrom	STRING	Dirección de correo del usuario
strUser	STRING	Nombre de usuario
strPassword	STRING	Contraseña

Tabla 3.36- Input/output bloque SMTP_Client_V1_16

En el caso anterior, se envía como adjunto el último archivo creado por el DataLogger, a desde y hacia la dirección 'David_Prueba_Phoenix@gmx.es', cada vez que generemos manualmente un flanco de subida en la variable *xEnviar*, que está conectada a la entrada *xSendMail*.

Cada vez que esto ocurre, este programa envía un e-mail a todas las direcciones almacenadas en la estructura de datos *udtMailTo*, con el asunto *strSubject*, y llevando como adjuntos los archivos designados en el array *arrFilenames*.

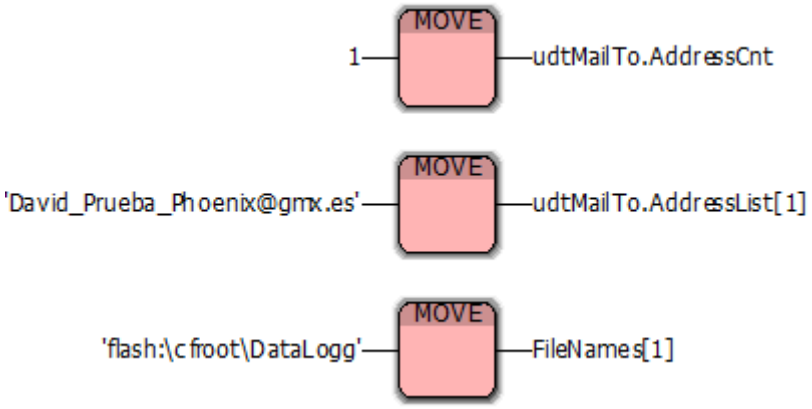

Imagen 3.12 - Ejemplo de asignación de direcciones de destino y archivos adjuntos

3.2.8 Programa DataLog

En este programa, se configuran los bloques que definen la funcionalidad DataLogger del PLC y que permiten generar los archivos *.csv* que se almacenarán en su espacio de memoria interna. Adicionalmente, se plantea la subida de dichos archivos a un servidor FTP como funcionalidad adicional dentro de las posibilidades IT de esta familia de PLCs de Phoenix Contact.

El primer bloque que debe llamar nuestra atención aquí, es el que permite configurar de forma general dicha funcionalidad de DataLogger, y que genera como salida valores almacenables en archivos de texto estándar. Se trata del bloque *DataLogger_V0_18*:

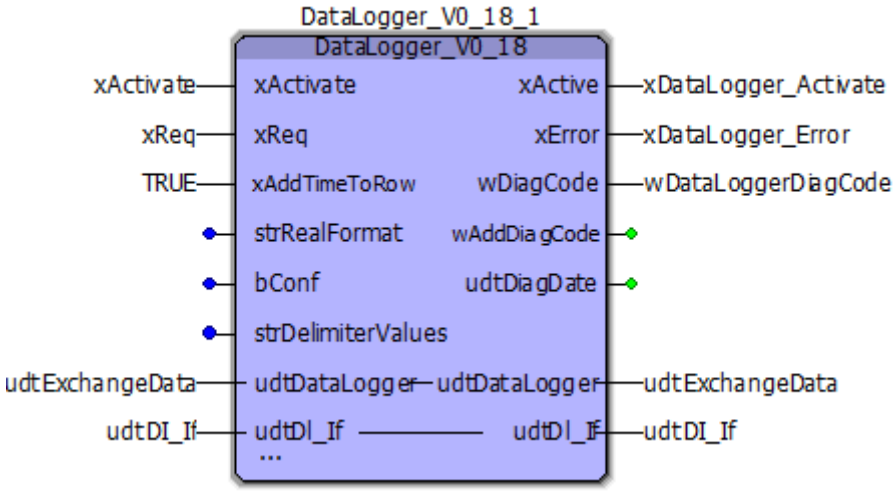

Imagen 3.13 - Bloque DataLogger_V0_18

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del bloque (TRUE)
xReq	BOOL	Flanco que realiza la carga de nuevos valores
xAddTimeToRow	BOOL	Añade la fecha a la fila de datos
strRealFormat	STRING	Resultado de la conversión de la cadena a STRING
bConf	BYTE	Diferentes opciones de formato: elimina encabezado (0), usa SQL (1), añade hito en ms. al tiempo (2), guarda datos por cada ciclo del PLC (3)
strDelimiterValues	STRING	Cadena escrita para la separación entre valores

Tabla 3.37 - Entradas bloque DataLogger_V0_18

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo (salida a TRUE)
xERROR	BOOL	Error en el envío si variable a TRUE
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código de diagnóstico (extendido)
udtDiagDate	udtDL_LoggerDiagData	Fecha y hora para diagnóstico del DataLogger

Tabla 3.38 - Salidas bloque DataLogger_V0_18

VAR_INPUT_OUTPUT	Tipo	Descripción
udtDataLogger	udtDL_V3	Estructura con los datos en formato binario
udt_DI_If	udtDataLoggerInterface_V1	Estructura con los datos en formato de salida, listos para su escritura

Tabla 3.39 - Input/output bloque DataLogger_V0_18

Una vez que hemos definido la función de DataLogger, hemos de indicar qué variables y en qué formato deseamos que sean registrados sus valores (por el bloque anterior, de ahí la estrecha relación entre ellos). Para ello, tenemos una serie de distintos bloques que dependen del tipo de dato que define la variable, aunque su configuración es idéntica:

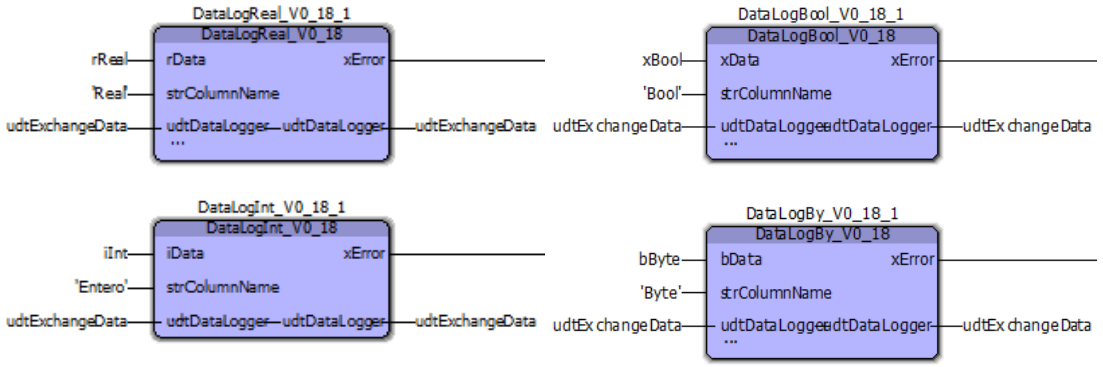

Imagen 3.14 - Bloques de registro DataLogger (por tipo de datos)

VAR_INPUT (entradas)	Tipo	Descripción
x Data	REAL, INT, BOOL, BYTE, WORD	Entrada de la variable a leer, por tipo de datos dependiendo del bloque usado
strColumnName	STRING	Cadena de texto para encabezado de la columna

Tabla 3.40 - Entradas bloques de registro DataLogger (por tipo de datos)

VAR_OUTPUT (salidas)	Tipo	Descripción
xERROR	BOOL	Error en el envío si variable a TRUE

Tabla 3.41 - Salidas bloques de registro DataLogger (por tipo de datos)

VAR_INPUT_OUTPUT	Tipo	Descripción
udtDataLogger	udtDL_V3	Estructura con los datos en formato binario

Tabla 3.42 - Input/output bloques de registro DataLogger (por tipo de datos)

Con los bloques de registro configurados, pasamos a decirle al programa dónde queremos guardar los archivos generados, en qué formato, y con qué características. Para ello, se muestran a modo de ejemplo tanto la forma de generación de archivos *.csv*, como la subida de los últimos registros, ya recogidos en este formato, a un servidor FTP:

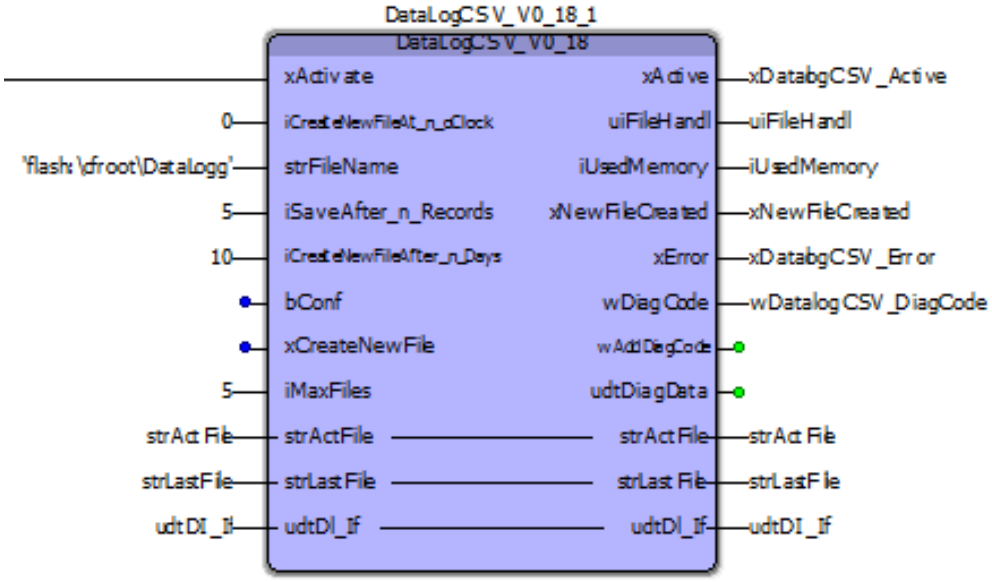

Imagen 3.15 - Bloque DataLogCSV_V0_18

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del bloque
iCreateNewFileAt_n_oClock	INT	Hora en la que se crea un nuevo fichero
strFileName	STRING	Nombre de archivo y ruta de guardado
iSaveAfter_n_Records	INT	Guarda el fichero cada *n* registros
iCreateNewFileAfter_n_Days	INT	Cada *n* días se crea un nuevo archivo
bConf	BYTE	Opciones de configuración: mantiene el archivo abierto (0), añade fecha al nombre del archivo (1), añade línea de encabezado (2)
xCreateNewFile	BOOL	Detección de flanco para generar un nuevo fichero
iMaxFiles	INT	Cantidad máxima de archivos simultáneos en la memoria del PLC

Tabla 3.43 - Entradas bloque DataLogCSV_V0_18

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo (salida a TRUE)
uiFileHandl	UINT	Informa si archivo abierto/cerrado
iUsedMemory	INT	Memoria usada en %
xNewFileCreated	BOOL	Se activa si se ha generado un nuevo archivo en el ciclo PLC
xERROR	BOOL	Error en el envío si variable a TRUE
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código de diagnóstico (extendido)
udtDiagDate	udtDL_LoggerDiagData	Fecha y hora para diagnóstico del DataLogger

Tabla 3.44 - Salidas bloque DataLogCSV_V0_18

VAR_INPUT_OUTPUT	Tipo	Descripción
strActFile	STRING	Archivo actual
strLastFile	STRING	Último archivo que se ha guardado
udt_DI_If	udtDataLoggerInterface_V1	Estructura con los datos en formato de salida, listos para su escritura

Tabla 3.45 - Input/output bloque DataLogCSV_V0_18

Con este bloque, y en el ejemplo descrito, logramos que a cada flanco que llega a la entrada *xActivate*, se guarden los datos almacenados por el DataLogger en el archivo que se encuentra en la ruta dada por *strFileName*, cada 5 registros.

La creación de nuevos archivos *.csv* se producirá a cada flanco en la variable *xCreateNewFile* o bien cada 10 días de forma automática (valor dado por la entrada *iCreateNewFileAfter_n_Days*). El nombre de los ficheros será el indicado, más la fecha y la hora en la que se han generado.

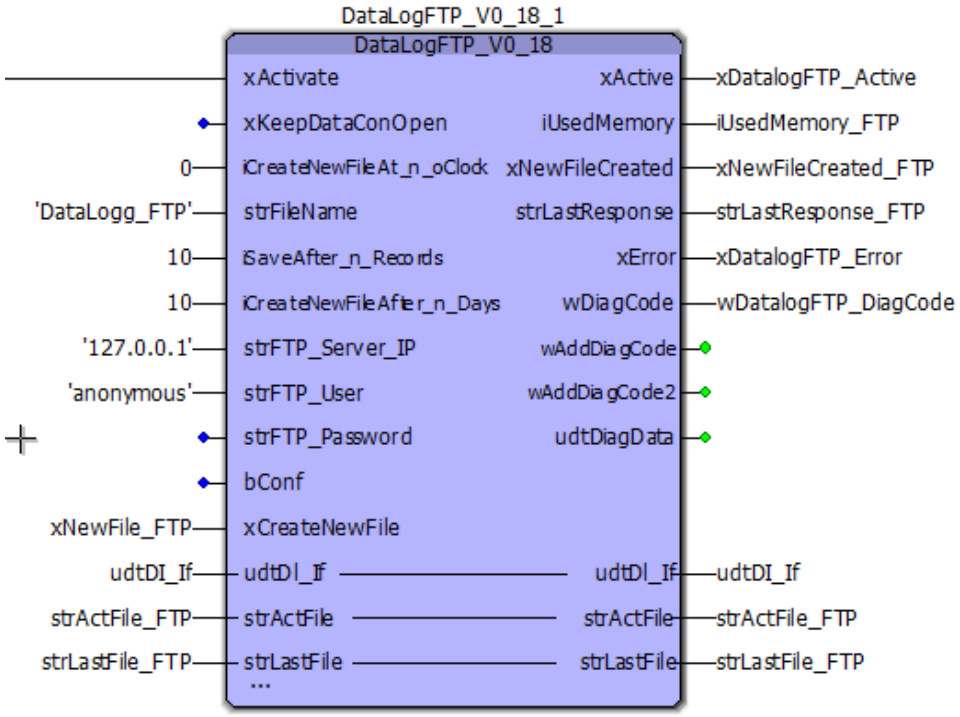

Imagen 3.16 - Bloque DataLogFTP_V0_18_1

VAR_INPUT (entradas)	Tipo	Descripción
xActivate	BOOL	Activación del bloque (TRUE)
xKeepDataConOpen	BOOL	Mantiene la conexión abierta (TRUE)
iCreateNewFileAt_n_oClock	INT	Hora en la que se crea un nuevo fichero
strFileName	STRING	Nombre de archivo y ruta de guardado
iSaveAfter_n_Records	INT	Guarda el fichero cada *n* registros
iCreateNewFileAfter_n_Days	INT	Cada *n* días se crea un nuevo archivo
strFTP_Server_IP	STRING	Cadena que contiene la dirección IP del servidor FTP
strFTP_User	STRING	Usuario con el que me *logeo* en el servidor
strFTP_Password	STRING	Contraseña para la autentificación
bConf	BYTE	Opciones de configuración: mantiene el archivo abierto (0), añade fecha al nombre del archivo (1), añade línea de encabezado (2)
xCreateNewFile	BOOL	Detección de flanco para generar un nuevo fichero

Tabla 3.46 - Entradas bloque DataLogFTP_V0_18

VAR_OUTPUT (salidas)	Tipo	Descripción
xActive	BOOL	Bloque activo (salida a TRUE)
uiFileHandl	UINT	Informa si archivo abierto/cerrado
xNewFileCreated	BOOL	Se activa si se ha generado un nuevo archivo en el ciclo PLC
strLastResponse	STRING	Fecha y hora de la última confirmación del servidor
xERROR	BOOL	Error en el envío si variable a TRUE
wDiagCode	WORD	Código de diagnóstico
wAddDiagCode	WORD	Código de diagnóstico (extendido)
wAddDiagCode2	WORD	Código de diagnóstico (extendido)
udtDiagDate	udtDL_LoggerDiagData	Fecha y hora para diagnóstico del DataLogger

Tabla 3.47 - Salidas bloque DataLogFTP_V0_18

VAR_INPUT_OUTPUT	Tipo	Descripción
udt_DI_If	udtDataLoggerInterface_V1	Estructura con los datos en formato de salida, listos para su escritura
strActFile	STRING	Archivo actual
strLastFile	STRING	Último archivo que se ha guardado

Tabla 3.48 - Input/output bloque DataLogFTP_V0_18

A la vista del programa creado como ejemplo, vemos que la dirección del servidor FTP que se ha utilizado para estas pruebas, es la del *localhost* (127.0.0.1). Esto es así, porque se ha optado por ubicar en el propio PC de las pruebas, un servidor FTP local al respecto que podemos crear de manera sencilla a través del programa Filezilla, con parámetros automáticos para el puerto utilizado y los usuarios:

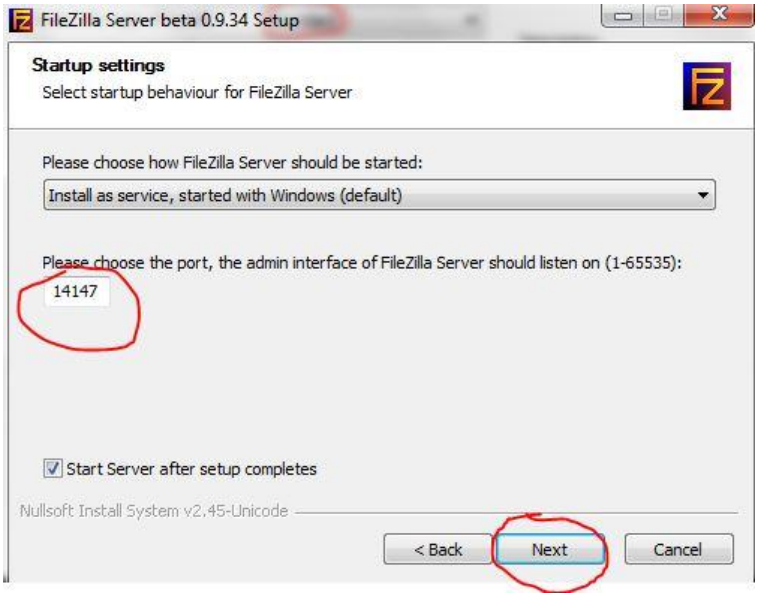

Imagen 3.17 - Creación de servidor FTP local

De esta manera, podemos hacer una prueba rápida del funcionamiento del servidor y la correcta subida de archivos desde el PLC de manera sencilla, sin tener que buscar por la red un servidor FTP gratuito cuya política de permisos y seguridad nos deje probar estas funcionalidades.

Para dichas pruebas se desaconseja utilizar el simulador PC-Worx SRT, ya que pueden existir conflictos si alojamos nuestro servidor en *localhost*, ya que aquí es donde dicho simulador genera, de forma virtual, los archivos que componen su memoria interna imitando a un PLC real. Por lo tanto, es conveniente detener el servicio del SRT antes de testear estas funcionalidades.