Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías Industriales

Control de un motor de inducción basado en un variador de velocidad

Autor: Fernando José Sánchez Gómez

Tutor: Daniel Limón Marruedo

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Control de un motor de inducción a través de un variador de velocidad

Autor:
Fernando José Sánchez Gómez

Tutor:
Daniel Limón Marruedo
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Control de un motor de inducción a través de un variador de velocidad

Autor: Fernando José Sánchez Gómez

Tutor: Daniel Limón Marruedo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia, quienes me han apoyado en todo momento y siempre están ahí.

A todos esos maestros de los que tanto he aprendido, que han sabido transmitirme las ganas y el aprendizaje necesarios para progresar, y de los que tanto me falta por aprender.

Agradecimientos

Me gustaría agradecer este proyecto, a:

Mi tutor, D. Daniel Limón Marruedo, por haber tenido la suerte de contar con su ayuda y con su gran conocimiento, por su dedicación y por su infinita paciencia, pues sin él este proyecto no sería posible.

D. Javier Romero Suárez, por haberme permitido continuar su excelente trabajo, y por ofrecerme su ayuda en todo momento.

D. Luis Fernando Castaño Castaño, por ofrecerme su experiencia y conocimientos en este ámbito, que me han ayudado enormemente con las comunicaciones y con la configuración del sistema.

D. Ignacio Alvarado Aldea, por su gran ayuda ofrecida en el proceso de impresión de las piezas.

D. Francisco Salas Gómez, por su asesoramiento y orientación en el aprendizaje de la programación en Tia Portal.

Mi familia y amigos, por el grandísimo apoyo que me ofrecen día tras día y por el cariño recibido.

D. Juan Manuel García Mahave y el resto de compañeros de Emasesa, quienes me han apoyado enormemente y tengo en gran estima.

D. Carlos Luis Nogales García y el resto de personal del departamento, por el excelente clima de trabajo que fomentan en el que todos ofrecen siempre su ayuda.

Resumen

Este proyecto puede clasificarse dentro de los proyectos de desarrollo, y se ha centrado en la integración de un equipo variador de velocidad con autómatas programables (PLCs).

Se ha llevado a cabo para finalizar la renovación de un equipo didáctico de Schneider Electric para el aprendizaje de control de motores de inducción mediante variador de frecuencia, mediante la incorporación de un control en velocidad en bucle cerrado y el uso de autómatas programables.

El objetivo del proyecto es, por tanto, configurar y ensayar algunas estrategias de control que permitan avanzar en el conocimiento práctico de estos equipos.

El variador dispone de entradas que permiten al operador obtener distintos valores de las características de salida del motor (velocidad, par, aceleración…), dentro de su rango de funcionamiento. Dichas entradas podrán ser manipuladas localmente (a partir de los elementos del armario) o de forma remota.

Se han contemplado varias alternativas para la gestión del control del equipo:

La primera, basada en el comando del variador a través de señales analógicas en tensión. Para este objetivo se empleará un PLC *S7-1200* del fabricante Siemens, junto con una pantalla de explotación ó HMI, desde la cual podremos visualizar el estado del proceso, alternar entre modos de funcionamiento y cambiar la consigna de velocidad. Ambos equipos han sido programados haciendo uso de un PC y del software Tia Portal v15 ©, empleando comunicación Ethernet TCP/IP en ambos casos, y un Switch de comunicaciones para la interconexión de los equipos.

En la segunda alternativa se propone una interesante idea para la gestión del variador en una configuración redundante. Normalmente, este tipo de redundancia es propuesta por los fabricantes con equipos específicos, y con un software determinado. Sin embargo, en este caso se han empleado PLCs de propósito general, sin ningún hardware ni software de fabricante pensados para este fin.

El enfoque ha sido desarrollar estrategias de control para equipos redundantes a través de una configuración Modbus a dos niveles: un primer nivel con comunicación Modbus RTU entre el variador y los PLCs, con configuración maestro-esclavo, y un segundo nivel con comunicación Modbus TCP con configuraciones cliente-servidor entre los PLCs.

El esquema desarrollado y ensayado tiene la particularidad de ser tolerante a fallos (avería de un equipo, desconexión de red). Se presentan soluciones a los problemas característicos de redundancia de equipos, como son la sincronización de máquinas y el paso de testigo para el control maestro (conmutación del equipo de control) sin que se produzcan saltos bruscos en las señales de control al motor.

El control es asumido por uno de los PLCs, con su propia función y parametrización PID. En caso de que este PLC interrumpa su comunicación con el variador, automáticamente otro PLC cogerá el relevo aplicando sus funciones PID, sin que se vea afectado el control que se estaba llevando a cabo a través del variador.

Abstract

This project can be classified as a development project, and is centered in a speed drive integration through Progammable Logic Computers (PLCs).

This project comes in to end-up a Schneider Electric's teaching equipment update, which aims the learning on how to control an induction motor through a frequency drive, adding a closed-loop speed control and programmable logic computers.

Therefore, project's objective is to configure and essay some control strategies to progress in this equipment's practice knowledge.

The motor drive has inputs that allow the operator getting different motor output characteristic values (speed, torque, acceleration…), within its operating range. These inputs can be manipulated locally (from the cabinet elements), or remotely.

In case of remote manipulation, we've contemplated few alternatives:

The first consists in commanding the speed drive through analog voltage signals. To do this, we will use an *S7-1200* PLC, from Siemens, with a Human-Machine Interface (HMI) display, which allows us to alternate between operating modes, changing the speed reference and see the status process. This equipment has been programmed using *Tia Portal v15* ©, using Ethernet TCP/IP communication in both cases, and a communication switch to connect the whole equipment.

The second alternative consists in an interesting idea to manage the speed drive with a redundant configuration. Normally, this functionality is purposed by manufacturers with specific software and equipment. However, in this case it has been implemented with general purpose PLCs, without any manufacturer's hardware of software thought to this application.

The idea has been to develop control strategies for redundant computers, through a two-level Modbus communication: first level consists in a Modbus RTU communication between PLCs and the speed drive, with a Master-Slave configuration, and second level consists in a Modbus TCP communication with a Client-Server configuration.

The control developed and tested has the peculiarity of being tolerant to failures (equipment faults, network disconnection). It brings solutions to the common problems of redundant equipment, as machine synchronisation and switching token to master control station (switching control station), away from abrupt steps in control signals to the motor.

The control is assumed by one PLC, with a proprietary PID function and parametrization. In case this PLC cuts off communication with the speed drive, another PLC will take control, applying its own PID function and parametrization, so speed drive's control won't be affected.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Actualmente, se conoce que el motor trifásico de inducción es uno de los elementos más empleados dentro de la industria. Estos equipos conectados a una fuente de tensión ofrecen unas prestaciones constantes de velocidad, fuerza, par… etc.

El variador de frecuencia surge precisamente de la necesidad de mejorar estas prestaciones para adaptarlas a la gran diversidad de tipos de procesos dentro de la industria. El mencionado equipo dispone de un conjunto de señales de control que permiten modificar estas variables de velocidad, par, etc.

Sin embargo, el variador de frecuencia común no es más que un actuador sobre el motor, que es capaz de modificar la potencia entregada al mismo pero que no tiene en cuenta las posibles perturbaciones que puedan existir sobre el medio, por lo que no se asegura de que se alcancen los valores de velocidad especificados.

En el presente proyecto se persigue el objetivo de alcanzar un control en bucle cerrado sobre el sistema variador-motor, de forma que se asegure en todo momento que la consigna de velocidad requerida sea alcanzada, a pesar de las posibles perturbaciones que puedan afectar al sistema.

Para ello, se ha implementado el control en bucle cerrado a través de PLCs, gracias a la medida de velocidad obtenida por un sensor digital inductivo, y el cálculo elaborado para obtener la magnitud de velocidad.

Se ha utilizado un PLC Siemens S7-1200, en el cual se ha probado el control en primer lugar sobre un modelo de sistema no lineal, y posteriormente sobre el variador.

Además, se ha implementado el mismo control utilizando un PLC Schneider M340. En este caso, además, se ha creado una red Modbus TCP con varios PLCs M340 redundantes, de forma que se asegura el control del variador en caso de que alguno de los PLCs caiga.

2 EQUIPO DIDÁCTICO

El obejtivo principal del TFG es el desarrollo de sistemas de control de velocidad para un equipo didáctico motor-variador existente en el departamento. En este capítulo se detallan las características del mismo y la elección e integración del sensor de velocidad necesario en el equipo.

2.1 Variador de frecuencia

2.1.1 ¿Qué es un variador de frecuencia?

Es muy común en el mundo industrial la presencia de motores de corriente alterna. Estos componentes conectados directamente a la red de suministro eléctrico se encuentran bajo una tensión y frecuencia constantes ofrecidas por dicha red. Esto, como ya se ha visto en la introducción, posee numerosas limitaciones que se solucionan en la industria mediante el uso de un variador de frecuencia.

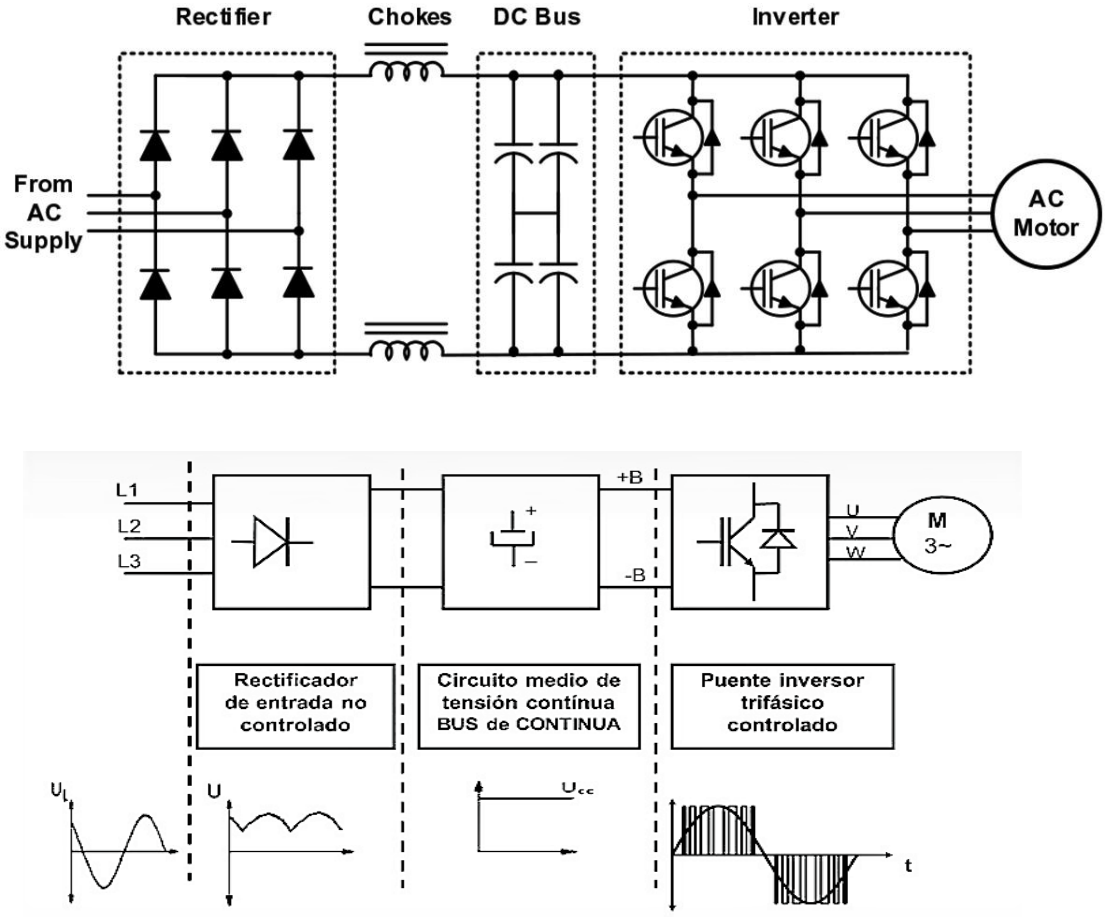
El variador de frecuencia es un sistema que controla la velocidad de giro de los motores de corriente alterna, controlando la frecuencia de alimentación suministrada al motor. [1] Este sistema se conectaría entre la alimentación energética y el motor, regulando la energía que llega al motor para luego ajustar la frecuencia y la tensión en función de los requisitos del procedimiento.

Por tanto, de esta importantísima aplicación se puede deducir que el uso de variadores de frecuencia trae consigo una gran cantidad de ventajas tanto a nivel de control, como energéticas, operativas e incluso medioambientales, ya que incrementa la eficiencia energética y alarga la vida útil de los equipos, previniendo el deterioro de los mismos.

2.1.2 Hardware y funcionamiento de un variador

El funcionamiento básico de un variador de frecuencia se divide en tres etapas, fácilmente entendibles si se dominan los conceptos básicos de electrónica de potencia.

- Primera etapa ó convertidor AC/DC, que toma la corriente alterna (200-240Vac) de la red y la rectifica mediante un puente de diodos.
- Segunda etapa ó bus de continua, en la que se aplica un filtrado LC de la señal, para eliminar el rizado.
- Tercera etapa ó convertidor DC/AC, en la que se genera una onda senoidal mediante un inversor, que permite ser modulada en amplitud y en frecuencia a través de la generación de pulsos PWM.



1 Etapas que determinan el funcionamiento del variador

2.1.3 Características

El variador de frecuencia con el que contamos en el equipo didáctico es el modelo "Altivar ATV320U07M2C", de Schneider, cuyas características principales son:

- Tensión de alimentación: 200 .. 240 V (- 15...10%)
- Frecuencia de alimentación: 50 .. 60 Hz
- Corriente de alimentación: 10.1 A a 200V / 8.5 A a 240V para altas cargas.
- Compatibilidad con motores síncronos / asíncronos. Potencia máxima de motor: 0.75 kW
- Frecuencia de salida: 0.1 .. 599 Hz
- Frecuencia de cambio nominal: 4 kHz
- Regulador PID interno ajustable
- Resolución de frecuencia: unidades en el display: 0.1 Hz, entrada analógica: 0.012/50 Hz
- Protocolos admitidos por el puerto de comunicaciones: CANopen / Modbus RTU
- Características extraídas directamente del datasheet [9]:

Entrada analógica	Tensión (AI1), estado 1 0...10 V CC, resolución 10 bits, impedancia 30000 Ohm Tensión diferencial bipolar (AI2), estado 1 +/- 10 V CC, resolución 10 bits, impedancia 30000 Ohm Corriente (AI3), estado 1 0...20 mA (o 4-20 mA, x-20 mA, 20-x mA u otros patrones según configuración), resolución 10 bits, impedancia 250 Ohm
Entrada digital	Programable (común positivo/común negativo) (DI1...DI4), estado 1 24...30 V CC, estado 1 PLC niv 1 Programable como entrada de pulsos 20 kpps (DI5), estado 1 24...30 V CC, estado 1 PLC niv 1 Sonda PTC configur, por conm, (DI6), estado 1 24...30 V CC Par de torsión seguro (STO), estado 1 24...30 V CC, impedancia 1500 Ohm
Salida analógica	Corriente configurable por software (AQ1), estado 1 0...20 mA, resolución 10 bits, impedancia 800 Ohm Tensión configurable por software (AQ1), estado 1 0...10 V, resolución 10 bits, impedancia 470 Ohm
Salida digital	Lógica relé configurable NA/NC (R1A, R1B, R1C): electrical durability 100000 ciclos Lógica relé configurable NA (R2A, R2B): electrical durability 100000 ciclos Lógica (LO)

Tabla 1. E/S del variador

2 El variador de frecuencia ATV 320U07M2C

2.2 Armario

El variador se integra en un armario que permite operar el equipo de forma cómoda y segura, facilitando el conexionado con el motor, y tanto la visualización como la modificación de ciertas variables (cuando se opera en modo manual).

Este armario fue actualizado (incorporando el nuevo variador) en el trabajo de Fin de Grado de Javier [4].

2.2.1 Introducción

Atendiendo a la ilustración 3, se muestran los distintos componentes con los que cuenta el armario:

(1) Marcha: Abre la conexión de la red eléctrica con el variador.

(2) Paro: Cierra la conexión de la red eléctrica con el variador.

(3) Disyuntor: aislamiento del motor

(4) Bornero del freno: Habilita el funcionamiento del freno acoplado al motor

(5) Bornero del Motor: Existe una borna para cada fase más la de tierra.

(6) Variador

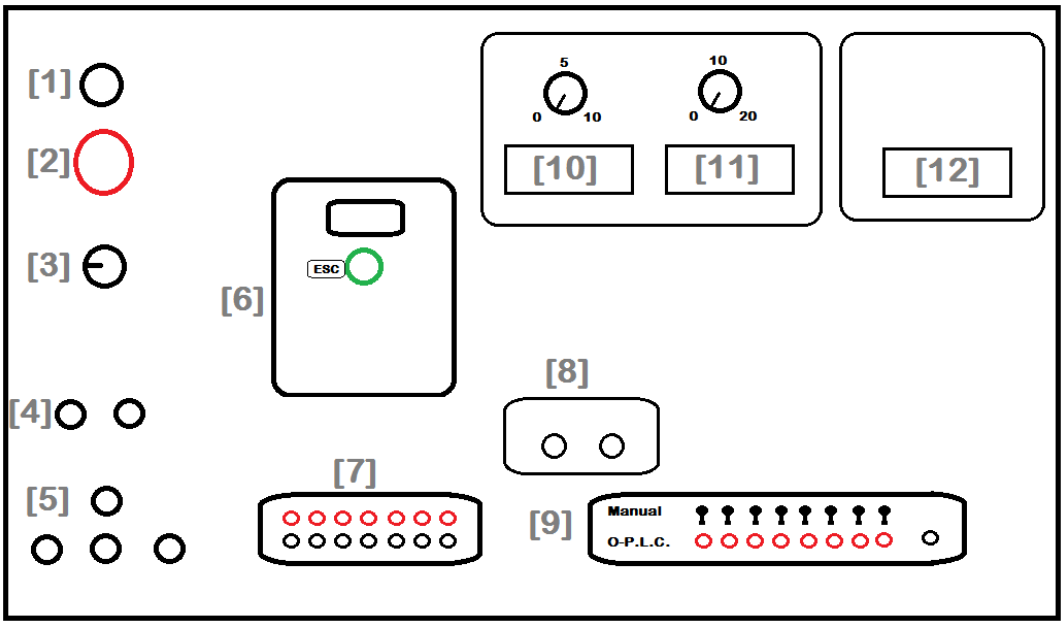
(7) Bornero para chequeo de señales: Permite testear ciertas corrientes y voltajes.

(8) Relés: Comunican mediante leds estados y/o errores que van sucediendo.

(9) Bornero y contactores para entradas lógicas: Pueden ser gestionadas de forma manual o conectándolas a un dispositivo remoto.

(10) Display y potenciómetro de la referencia en tensión 0-10 Vdc.

(11) Display y potenciómetro de la referencia en corriente 4-20 mA.



3 Esquema configuración del armario

4 Vista del armario de control

2.3 Motor de inducción asíncrono de jaula de ardilla

Atendiendo a la referencia [2]:

El principal tipo de motor usado en la industria es el motor eléctrico de inducción trifásico tipo jaula de ardilla. Fundamentalmente por los siguientes aspectos:

- Altos niveles de eficiencia comparado con otros tipos de motor
- Bajos costos de mantenimiento
- Bajo costo y facilidad de adquisición
- Altos grados de protección y posibilidad de uso en áreas clasificadas.

2.3.1 Funcionamiento

[2] El funcionamiento de un motor de inducción trifásico se basa en la interacción de los campos magnéticos del rotor y del estator. El estator posee un devanado trifásico, por el cual circulan unas intensidades que crean un campo magnético giratorio en el entrehierro que gira a la velocidad de sincronismo. Este campo induce corrientes trifásicas en el rotor, que a su vez generan un campo magnético giratorio a la velocidad síncrona respecto al estator. Como consecuencia de esto y de acuerdo con el principio de alineamiento de campos magnéticos, el rotor se ve sometido a un par, que tiende a alinear ambos campos. Se produce así por tanto el giro del rotor en la misma dirección que el campo magnético del estator.
Cuando la velocidad del rotor se aproxima a la del campo estatórico, la magnitud de las corrientes inducidas en el rotor disminuye. De esta forma el par aplicado al rotor disminuye, hasta que a la velocidad de sincronismo el par es nulo. Así pues, en condiciones normales de funcionamiento, el rotor girará a una

velocidad ligeramente inferior a la de sincronismo, de ahí que a estos motores se les llame motores asíncronos. **La diferencia entre la velocidad del campo magnético y la del rotor se denomina deslizamiento**. Cuanto mayor es el deslizamiento, mayor es el par que puede ser aplicado a la carga.
La velocidad de sincronismo es:

$$\Omega = \frac{\omega}{p} = \frac{2 \cdot \pi \cdot f}{p}$$

siendo:

Ω = velocidad de sincronismo expresada en radianes geométricos por segundo (rad/s)

w = pulsación del sistema de corrientes trifásicas (rad/s)

f = frecuencia del sistema de corrientes trifásicas en Hertzios (Hz)

p = nº de pares de polos del estator (= al del rotor)

En vueltas por minuto:

$$n = \frac{60f}{p}$$

siendo n la velocidad de sincronismo en revoluciones por minuto (rpm).

El deslizamiento se define como:

$$s = \frac{\Omega - \Omega_r}{\Omega} = \frac{n - n_r}{n}$$

siendo:

s = deslizamiento (adimensional)

Ωr = velocidad a la que gira el rotor en radianes por segundo (rad/s)

nr = velocidad a la que gira el rotor en revoluciones por minuto (rpm)

En funcionamiento como motor, el deslizamiento tiene un valor comprendido entre 0<s<1, siendo típico un valor entre 0.02 y 0.05."

2.4 Freno de polvos magnéticos

Existen dos tipos de cargas a emular en nuestro sistema: dinámicas y estáticas. Para ello se usa este dispositivo el cual estará dispuesto axialmente con el eje de salida del motor de inducción de jaula de ardilla.

La evidente finalidad de un sistema de frenado es la de reducir la velocidad de un cuerpo, llevándolo en ocasiones a un estado de reposo total.

2.4.1 Principio de funcionamiento

Para hablar sobre el funcionamiento de este dispositivo de frenado se acude a la siguiente cita [11]:

Para el presente caso, el freno utilizado para emular la presencia de cargas dinámicas es un freno por partículas magnéticas. El funcionamiento de este tipo de frenos se basa en la disposición de un polvo ferromagnético

totalmente “libre” en un compartimento estanco de reducidas dimensiones, compartimento que separa dos partes: un disco que gira libremente y una parte fija donde se encuentra alojada la bobina que crea el campo magnético. Al energizar la bobina mediante el empleo de corriente continua, se produce una magnetización de la bobina creando unas líneas de flujo magnético.

El par se transmite a través de las partículas de hierro resistentes al desgaste, las cuales forma cadenas de partículas en función de la intensidad del campo electromagnético, formando un aglutinamiento que transmite el par al dificultar el movimiento relativo entre el disco y la carcasa. La intensidad del campo establece de forma lineal la rigidez de las cadenas y en consecuencia la cantidad de par transferible. Resulta evidente que a mayor intensidad circulando por la bobina, mayor será la fuerza ejercida independientemente de las revoluciones a las que gire el eje.

Por tanto, la inexistencia de fricción o contacto directo entre el disco y la carcasa en los frenos de partículas magnéticas se torna beneficioso al no producir desgastes ni reglaje alguno, lo que repercute en una mayor vida, buena estabilidad con alto grado de repetitividad y un funcionamiento completamente silencioso.

El uso que se le da en este proyecto, exige al freno de partículas magnéticas trabajar en modo de deslizamiento permanente, siempre que la cantidad de calor que tenga que ser disipado no sea excesiva. En dicho modo de deslizamiento, las partículas magnéticas trabajan sometidas a un bajo régimen de desgaste. Sin embargo, puede ocurrir que el par caiga por debajo de los requerimientos necesarios, obligando a reemplazar dichas partículas por unas nuevas. Por tanto, es evidente que la vida útil del freno dependerá de las condiciones de operación.

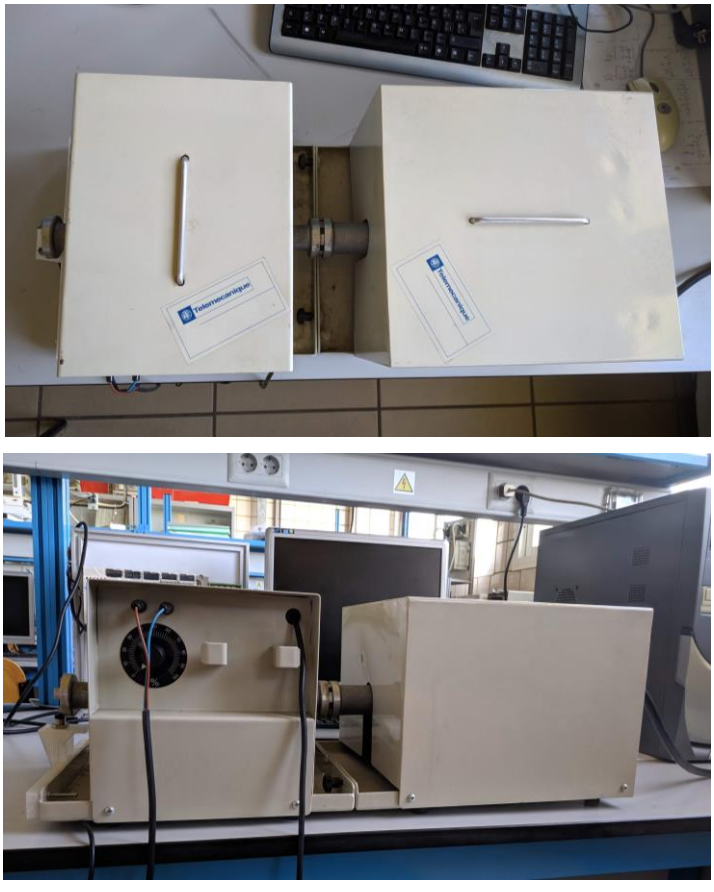
Las ventajas que presenta utilizar un freno como el expuesto anteriormente son:

· Permiten establecer un control del par de forma precisa, mediante la corriente de excitación.

· Dinámica rápida.

· Mayores tiempos de vida ante la inexistencia de fricciones en lo referente al acoplamiento mecánico.

· Bajo ruido.”



5. Vista de planta y perfil del motor (derecha) + freno (izquierda)

2.5 Autómata Siemens S7-1200

Como ya se ha comentado, se cuenta en primer lugar con un autómata Siemens S7-1200, modelo "6ES7 214-1BG40-0XB0", dotado además de un terminal HMI Basic Panel. Las comunicaciones se realizan a través de un switch Siemens, modelo "Scalance XB005".

2.5.1 Principales características del autómata

El autómata posee una cpu denominada "1214C AC/DC/Rly", cuyo nombre ya nos está indicando que se alimenta a la tensión de la red AC (120/220Vac), que tiene entradas digitales de continua (24Vdc) y que las salidas digitales son a relé, es decir, servirán para ser conectadas a relés, que habiliten el paso de la corriente a través de equipos de mayor potencia, sin embargo estas salidas estarán más limitadas en cuanto a los tiempos de conmutación.

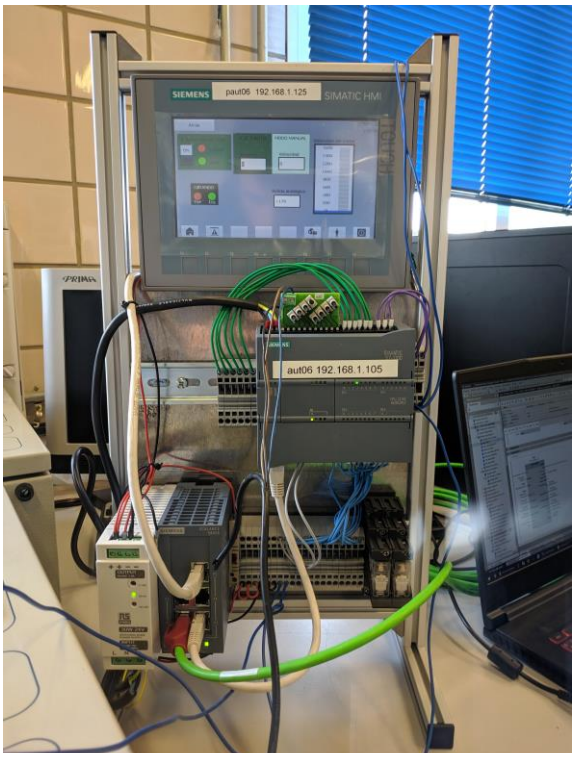
A continuación, se listan las características que aportan la información más relevante para nuestra aplicación:

- Paquete de programación: STEP 7 v14 ó superior
- Tensión de alimentación: 120 V AC / 230 V AC
- Alimentación de sensores 24V: 20,4 a 28,8 V
- Sensor compatible: sensor a 2 hilos
- Entradas digitales:
    - Nº DI: 14, de ellas 6 HSC (High Speed Counting)
    - Fuente/sumidero: sí
    - Tensión de entrada: 24V
        - Para señal "0": 5Vdc, con 1mA
        - Para señal "1": 15Vdc, con 2,5mA
- Salidas digitales:
    - Nº DO: 10 salidas a relé
- Entradas analógicas:
    - Nº AI: 2, en tensión (0-10V)
- Salidas analógicas:
    - Nº AO: 2, en tensión (±10V) ó intensidad (0..20mA)
- Protocolos de comunicación disponibles: PROFINET IO, PROFIBUS

2.5.2 Métodos de comunicación

Como se puede apreciar en las especificaciones, este autómata cuenta con diversas formas de comunicación posibles que se han planteado a la hora de comunicarlo con nuestro variador.

Aprovechando que la lectura de algunas DI (entradas digitales) es de tipo HSC (High Speed Counting), hemos escogido éstas para realizar la lectura del sensor inductivo.



6 Equipo Siemens para el control analógico

2.6 Autómata Schneider M340

En la segunda fase de este proyecto, se ha realizado el control a través de un autómata del fabricante *Schneider Electric*, modelo *Modicon M340*. En este caso no contamos con terminal gráfico HMI, por lo que el PC desde el que se programa hará las veces de HMI.

2.6.1 Principales características del autómata

El PLC cuenta con una CPU denominada "BMX P34 2020", y en el caso de este proyecto, contaremos con posibilidad de comunicación Modbus y comunicación Ethernet, a través de los puertos RJ45 de la tarjeta. Adicionalmente permite la programación a través de un puerto USB.

También contaremos con una tarjeta de entradas/salidas digitales *DDM 3202K*, de una tarjeta de entradas analógicas *AMI0410* y de una tarjeta de entradas/salidas digitales *DDM 16022*.

7 Vista del PLC Modicon M340 y de las tarjetas de E/S

2.7 Sensor inductivo

2.7.1 Elección del sensor de velocidad

A la hora de escoger un sensor para detectar la velocidad del sistema, se han valorado distintos tipos de sensores analógicos, así como tacómetros ó motores DC, que ofrecerían una señal continua de velocidad.

Sin embargo, el sensor escogido para transmitir la señal de velocidad ha sido un sensor digital de tipo inductivo. La principal razón de dicha elección reside en la disponibilidad (ya disponíamos de este equipo en el laboratorio), y en el precio, ya que es un tipo de sensor bastante económico. El problema principal en la elección de este sensor era que se necesita que la frecuencia de conmutación sea suficiente para determinar todo el rango de velocidades en el que nos moveremos.

De esta manera, se ha calculado la frecuencia de conmutación máxima que tendría que soportar el sensor:

$$fc_{max} = \frac{n * v_{max}}{60}$$

Siendo n el número de conmutaciones por vuelta y v_{max} el régimen de vueltas máximo al que se someterá.

Considerando que el régimen de vueltas máximo al que se va a llevar el motor es de 1500rpm y el nº de conmutaciones por vuelta es 6 (3 dientes en el eje, 6 flancos a detectar por el sensor), concluimos que se necesita un sensor cuya frecuencia de conmutación esté por encima de los **150Hz**.

Observando el datasheet del sensor inductivo:

- *Size: 50 mm*
- *Type of output signal: Discrete*
- *Wiring technique: 3-wire*

- *Nominal sensing distance: 4 mm*
- *Discrete output type: PNP*
- *Rated supply voltage: 12...24 V DC with reverse polarity protection*
- *Supply voltage limits: 10...36 V DC*
- *Switching capacity in mA: <= 200 mA with overload and short-circuit protection*
- ***Switching frequency: <= 2500 Hz***
- **…**

Las especificaciones cumplen con creces las características mínimas requeridas.

2.7.2 Principio básico de funcionamiento del sensor

Como se explica en la referencia [12]:

Los sensores de proximidad inductivos contienen un devanado interno. Cuando una corriente circula por el mismo, un campo magnético es generado. Cuando un metal es acercado al campo magnético generado por el sensor de proximidad, éste es detectado.

La bobina, o devanado, del sensor inductivo induce corrientes de Foucault en el material por detectar. Estas, a su vez, generan un campo magnético que se opone al de la bobina del sensor, causando una reducción en la inductancia de la misma. Esta reducción en la inductancia de la bobina interna del sensor trae aparejado una disminución en la impedancia de ésta.

$$X_L = 2\pi * f * L$$

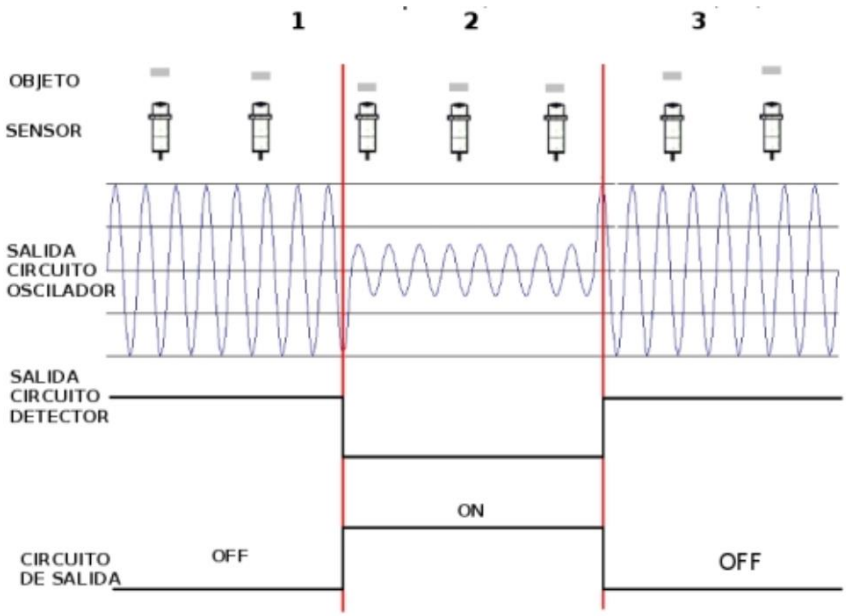
XL: Reactancia inductiva medida en ohms

f: Frecuencia del sistema medida en Hertz (Hz)

L: Inductancia medida en Henrios (H)

El oscilador podrá generar nuevamente el campo magnético con su amplitud normal. Es en este momento en que el circuito detector nuevamente detecta este cambio de impedancia y envía una señal al amplificador de salida para que sea éste quien, nuevamente, restituya el estado de la salida del sensor.

Si el sensor tiene una configuración “Normal Abierta”, este activará la salida cuando el metal a detectar ingrese en la zona de detección. Lo opuesto ocurre cuando el sensor tiene una configuración "Normal Cerrada".Estos cambios de estado son evaluados por unidades externas tales como: PLCs, relés, PCs, etc.



8. Funcionamiento sensor inductivo de proximidad

En esta aplicación usamos una configuración “Normal Abierta”, es decir, el sensor genera una señal cuadrada que se mueve en los valores de 0 y +24Vdc, y que es enviada al motor cuando está próximo a uno de los dientes del eje. En concreto, cada uno de los flancos de subida indica el paso por 1 tercio de vuelta.

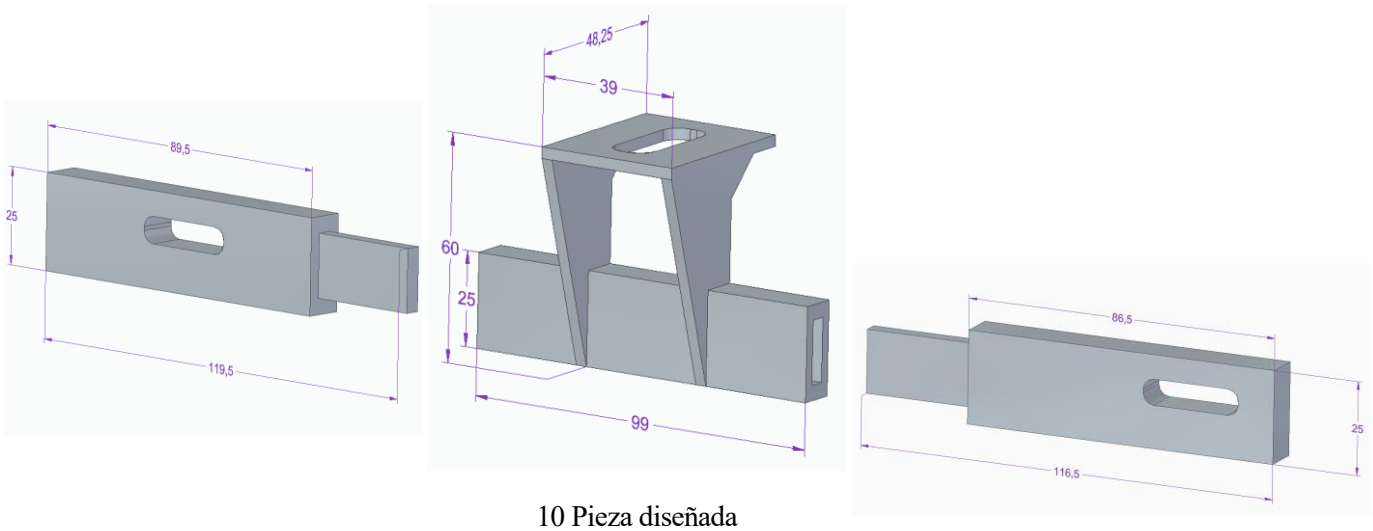
2.7.3 Montaje del sensor

Para el montaje del sensor sobre el eje, se requiere que la parte frontal del sensor inductivo se sitúe a una distancia menor de 4mm con respecto a los dientes del eje, sin llegar a tocarlos.

9 Vista del eje previa al montaje del sensor

Para lograrlo, y tras valorar distintas opciones, se ha diseñado una pieza a través de Solid Edge ©, de manera que pueda ser atornillada en la base que ya posee el propio motor, y que pueda sostener el sensor. La altura a la que se coloca puede ser regulada gracias a las tuercas que ya incorpora el propio sensor.

Debido a las restricciones de tamaño impuestas por la impresora, la pieza se diseña en tres partes encajadas (cotas en mm):



10 Pieza diseñada

Finalmente, tras la impresión y el montaje con los tornillos, el sensor queda fijado en su correcta posición, evitando así la pérdida de pulsos y lectura errónea de la velocidad.

11 Montaje final del sensor

3 SISTEMA DE CONTROL EN VELOCIDAD A TRAVÉS DE SALIDAS ANALÓGICAS

En este apartado se abordará la solución al problema de control a través de una tarjeta de salidas analógicas integrada en el PLC *S7-1200* del fabricante *Siemens*.

Para realizar este montaje será necesario el uso del software Tia Portal v15©, y del software adicional PLCSim©, y de los equipos mencionados en el apartado 2.5. de este documento.

3.1 Objetivo

El objetivo en esta fase es realizar el control realimentado de la velocidad del motor a través de un autómata, que establecerá el régimen de giro del motor y será capaz de controlar el sistema más allá de posibles perturbaciones, como podrían ser el envejecimiento, el freno, o cualquier otro elemento que haga que el motor no gire al régimen de vueltas requerido.

Para ello, se va a realimentar el sistema mediante un sensor inductivo que enviará pulsos al autómata, permitiendo obtener una medida elaborada de velocidad.

En la programación del PLC se empleará un bloque PID que recibirá realimentación con la señal de velocidad, y que realizará el control, habiendo sido sintonizado de manera adecuada para obtener una buena respuesta del sistema. La salida de este bloque PID será la actuación sobre el variador, y se realizará a través de la tarjeta de salidas analógicas, enviando una señal analógica de tensión en la entrada del variador de ±10V.

3.2 Montaje

Para poner en marcha el sistema, en primer lugar y por seguridad, se han realizado algunas simulaciones del sistema observando el comportamiento del mismo. Estos primeros pasos son únicamente comprobaciones que se han realizado por motivos de seguridad y para garantizar la correcta respuesta del equipo, por lo que no son necesarios para la puesta en marcha del equipo y podrían omitirse.

3.2.1 Comprobaciones previas

Para realizar estas comprobaciones será necesario el software adicional PLCSim©.

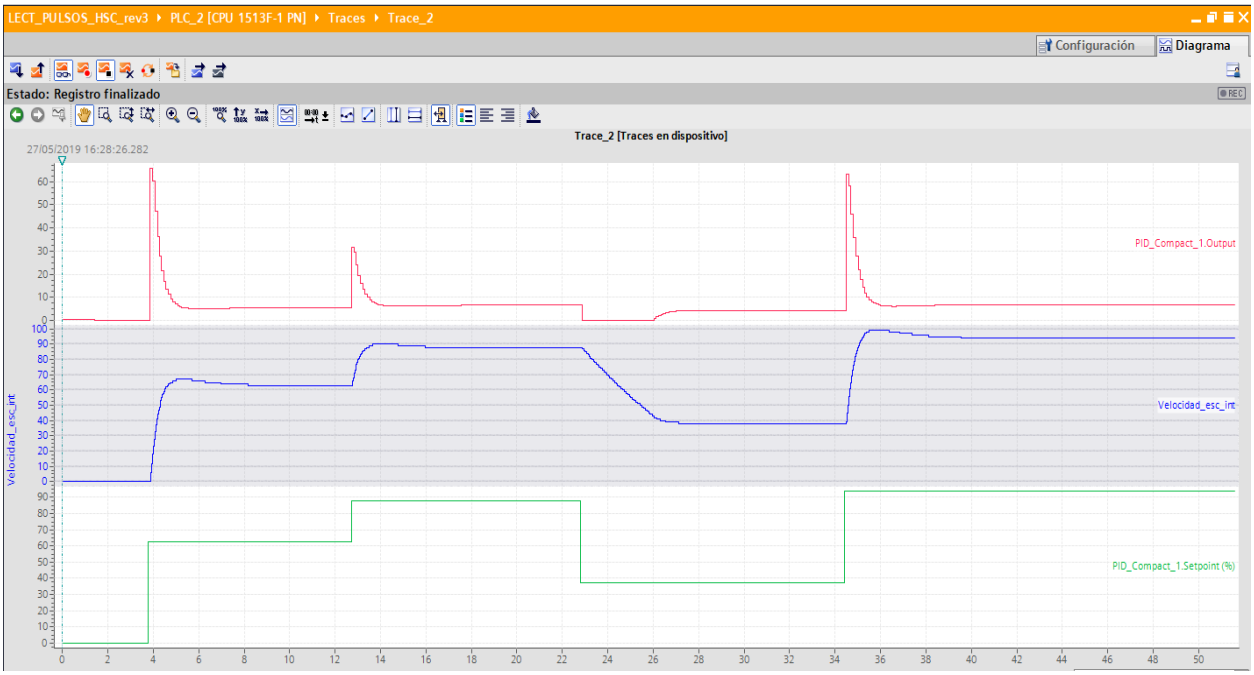
Para la primera comprobación, se ha programado un modelo de primer orden no lineal, que simulará el comportamiento de nuestro sistema y nos dará una idea de su comportamiento.

$$y_{k+1} = y_k + T_m * \frac{1}{2} * (u_k - \frac{1}{\sqrt{2}} * \sqrt{y_k})$$

Parametrizando el PID de la siguiente manera:

$$Kp = 1\ ;\ \ Ti = 2s\ ;\ \ Td = 0s$$

Empleando este modelo de primer orden y el PID con esa parametrización, se ha empleado el software PLCSim para realizar una simulación del modo automático del PID, de la que se ha obtenido lo siguiente:



12 Respuesta del modelo no lineal de primer orden reespecto a diferentes valores de Setpoint

La segunda comprobración consistió en conectar un multímetro a la salida analógica del PLC para comprobar la correcta actuación sobre la entrada analógica del variador.

13 Conexión del multímetro a la salida analógica del autómata

La comprobación consiste en probar los distintos rangos de salida (0%, 25%, 50%, 75%, 100%) y asegurarse de que corresponden a los niveles de tensión adecuados respecto al escalado.

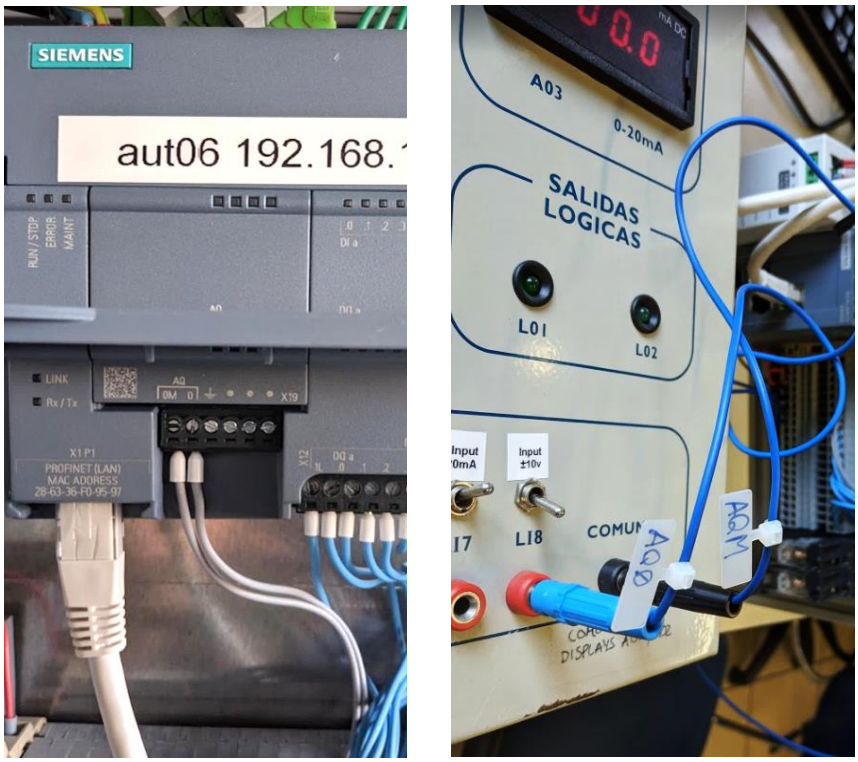
Una vez realizadas las comprobaciones pertinentes, podremos proceder a la puesta en marcha del equipo real.

3.2.2 Proceso de montaje

3.2.2.1 Montaje hardware

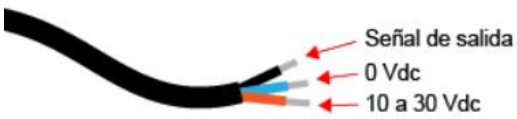
Para el montaje del hardware, configuraremos la salida analógica (actuación del PID) y la entrada digital (lectura de pulsos del sensor inductivo).

Localizaremos la salida analógica del autómata y la conectaremos a la entrada LI8 del variador (±10V). Los voltajes probados en el anterior ensayo serán los que se apliquen al variador a través de la entrada LI8, tal y como está configurado.



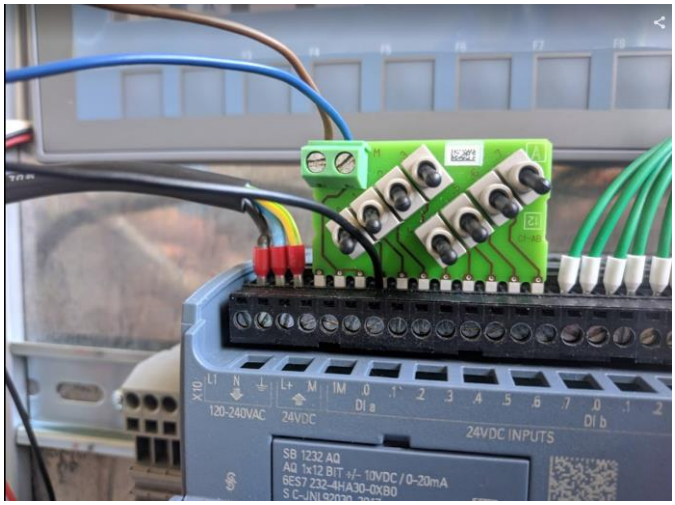
14 Cableado señal analógica

En cuanto al sensor, tal y como observamos en el datasheet [3], funciona a 3 hilos, dos para la alimentación y uno que enviaría la señal digital al sistema.



15 Hilos del sensor inductivo

Por tanto, empleamos la alimentación a +24Vdc del propio autómata y conectamos la señal de salida a la entrada digital DI 0.0



16 Conexionado sensor inductivo a *S7-1200*

3.2.2.2 Montaje software

A fin de transmitir la programación realizada a los equipos físicos, es necesario realizar la comunicación entre nuestro PC y el PLC junto con el terminal gráfico. Para ello, conectaremos ambos equipos (HMI y autómata) vía Ethernet al switch de comunicaciones, a través de los puertos RJ-45 integrados. Después, de la misma forma, conectaremos nuestro ordenador a dicho switch.

Una vez dentro del programa, hay que establecer conexión con nuestro hardware configurando las direcciones IP. Una vez tenemos conexión con el autómata, es necesario configurar las entradas/salidas físicas dentro del programa.

Entraremos en la opción *Configuración de dispositivos* **(1)** de la barra lateral seleccionamos nuestro PLC **(2).**

Para la salida analógica, entramos en *AQ1 Signal Board* **(3)** → *Salidas analógicas* → *Canal0*. Ahí seleccionamos el tipo de salida analógica en tensión**.** Comprobamos también que la salida está direccionada correctamente entrando en el apartado *Direcciones E/S* → *Dirección inicial* y comprobando que está en la dirección 80.

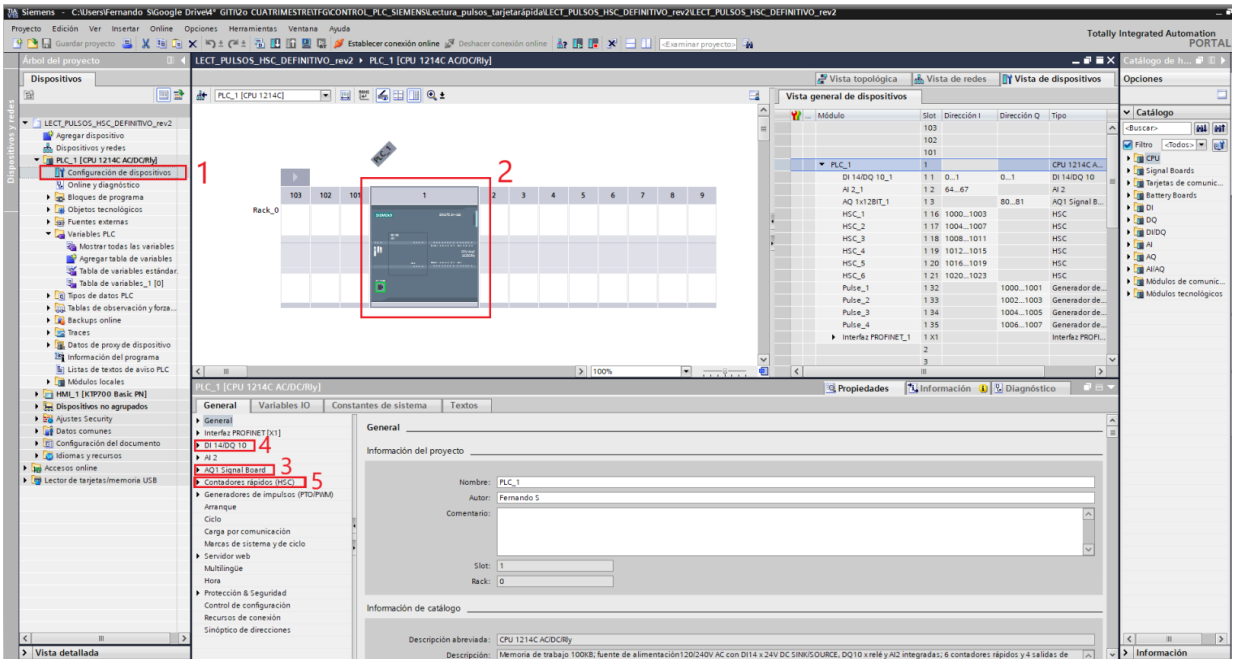
Por otro lado, es imprescindible para el correcto funcionamiento de la lectura de pulsos que la entrada digital en la que hemos conectado el sensor sea adaptada y configurada como contador rápido de pulsos (HSC).

Para ello, buscamos la opción *DI 14/DQ 10* **(4)** → *General* → *Entradas digitales* → *Canal0* → *Filtros de entrada*, y lo configuramos en *0.1 microsec*.

Después buscamos la opción *Contadores rápidos (HSC)* **(5)** → *HSC1*. Ahí vamos a configurar lo siguiente:

- *General* → *Activar este contador rápido*
- *Función* → ...
    - ... *Modo de contaje* → *Duración del período*
    - ... *Fase servicio* → *Monofásica*
    - ... *Sentido de contaje dado por* → *Programa de usuario*
    - ... *Sentido de contaje inicial* → *Incrementar contador*
    - ... *Período de medición de frecuencia* → *1.0*

- *Entradas de hardware → Entrada del generador de impulsos del reloj → %I0.0*
- *Direcciones E/S → Dirección inicial → 1000.0*

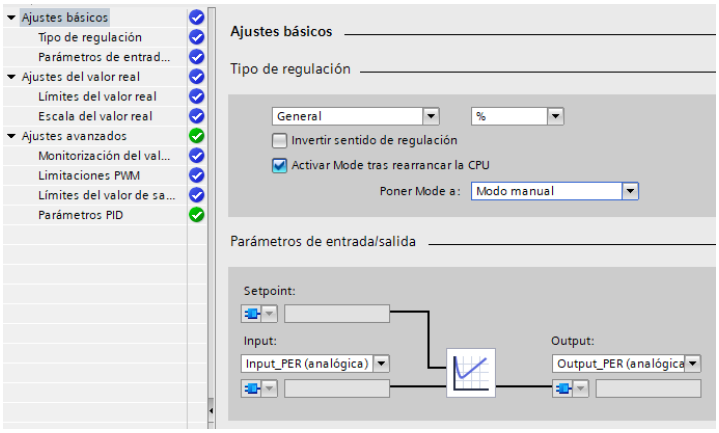


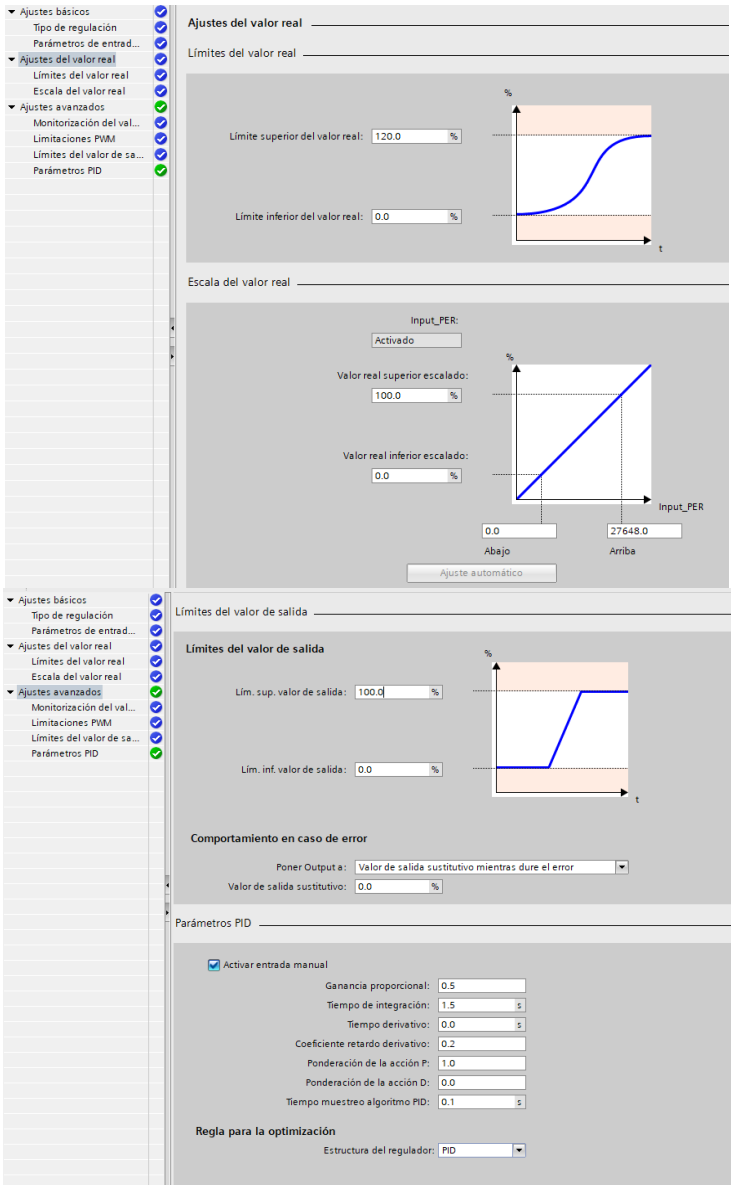
17 Vista general del programa en TIA Portal v15 para la configuración E/S

Una vez han quedado correctamente configuradas las entradas y salidas, transferimos el programa al autómata y conmutamos al estado RUN. Así quedará el sistema en funcionamiento en modo manual.

Antes de conmutar al modo automático, hay que parametrizar y poner en marcha el PID. Para ello, en la barra lateral seleccionamos *Objetos tecnológicos → PID_Compact_1 [DB4] → Configuración*. En este apartado podemos modificar la parametrización del controlador.

En las pruebas se ha utilizado la siguiente configuración:

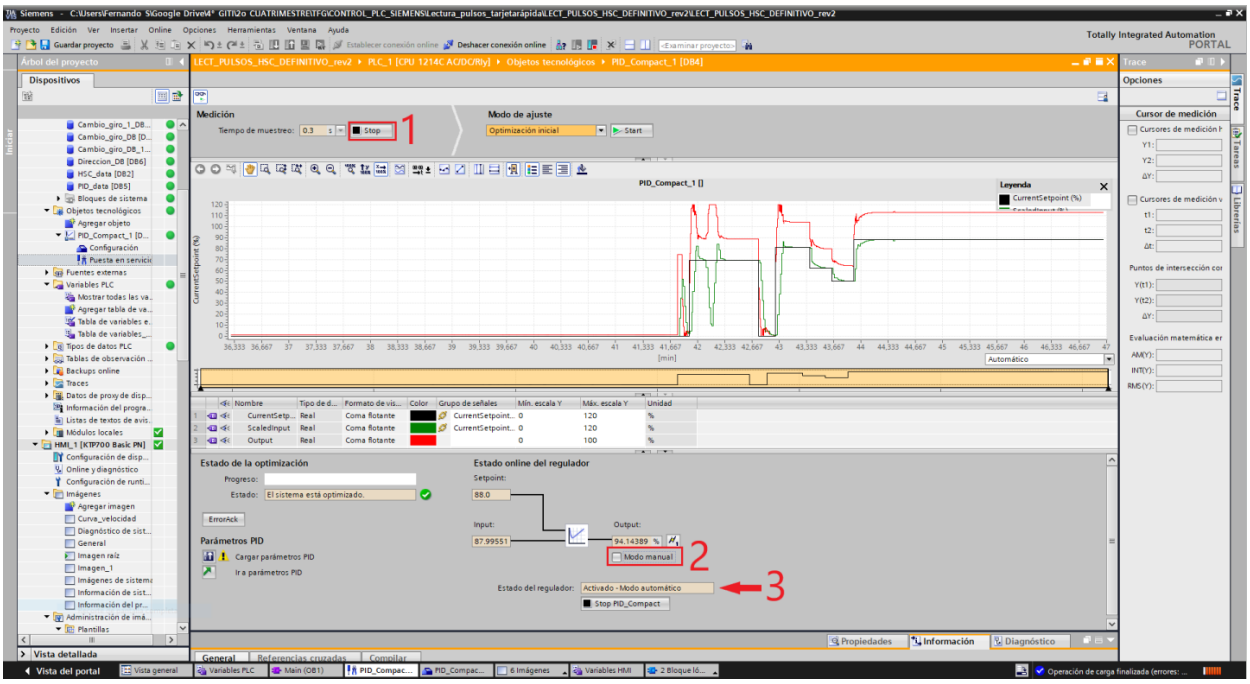




18 Configuración PID probada

Posteriormente, vamos a la sección *Objetos tecnológicos → PID_Compact_1 [DB4] → Puesta en servicio* y en el campo *Medición* comprobamos que el tiempo de muestreo está seleccionado en *0,3* y pulsamos en *Start* **(1)**.

Después, comprobar en la parte inferior que la casilla *Modo manual* **(2)** está desactivada, y que el PID está regulando en modo automático **(3).**



19 Puesta en servicio del bloque PID_Compact

Por último, cabe destacar la funcionalidad del software TIA Portal que permite realizar un "Auto ajuste" de la parametrización del PID, en el apartado *Modo de Ajuste → Optimización inicial / Optimización fina*.

La optimización inicial se realizaría en primer lugar, cuando el sistema parte del reposo, y una vez se establece el régimen permanente, se pasaría a la optimización fina.

Dicha funcionalidad no ha sido integrada en este proyecto.

3.3 Programación

En este apartado se explicará de la manera más resumida posible el funcionamiento básico de la programación del autómata para la ejecución correcta del control.

Está compuesta por los siguientes bloques de programa:

20 Bloques de programa del usuario

Y en la ejecución del programa se usarán las variables globales indicadas en la figura 20.

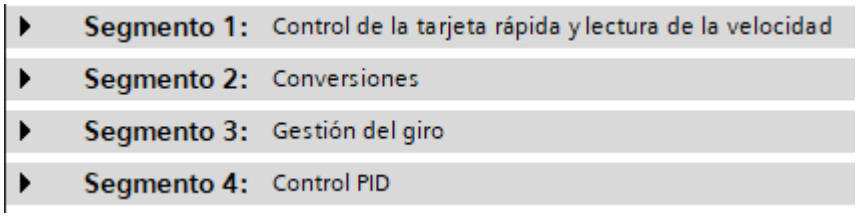
Variables PLC

	Nombre	Tabla de variables	Tipo de datos	Dirección	Rema...	Acces...	Escrib...	Visibl...	Comentario
1	Velocidad_porc	Tabla de variables e...	Real	%MD0		☑	☑	☑	Valor de la velocidad leída (en porcentaje)
2	Velocidad_analog	Tabla de variables e...	Real	%MD4		☑	☑	☑	Valor de la velocidad leída (valor analógico)
3	Analog_salida	Tabla de variables e...	Int	%QW80		☑	☑	☑	Señal analógica (+-10V) de salida
4	Setpoint	Tabla de variables e...	Real	%MD8		☑	☑	☑	Setpoint seleccionado en la regulación PID (en rpm)
5	Valor_manual	Tabla de variables e...	Int	%MW12		☑	☑	☑	Valor seleccionado en la regulación manual (en rpm)
6	Activa_control	Tabla de variables e...	Bool	%M14.0		☑	☑	☑	Variable que activa/desactiva el control
7	Analog_hmi	Tabla de variables e...	Real	%MD16		☑	☑	☑	Analógica de salida expresada en V, para la visualización en HMI
8	DIR	Tabla de variables e...	Bool	%M14.1		☑	☑	☑	Sentido de giro (1, izquierda; 0, derecha)

21 Variables globales

Bloque principal (*main*)

Este bloque estará constituido por 4 segmentos de código. En él se harán las llamadas a las distintas funciones.

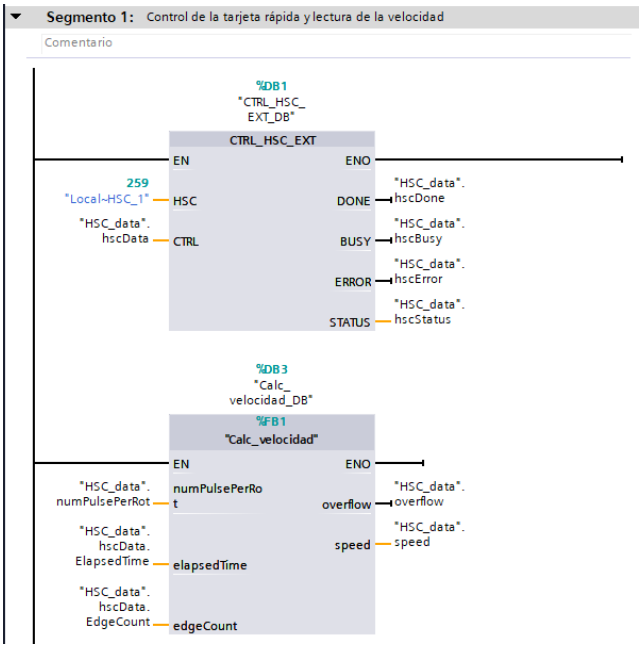


22 Segmentos del bloque principal

En el **segmento 1** encontramos dos llamadas a funciones.

La primera función *CTRL_HSC_EXT* se ha extraído de la librería de funciones HSC de Tia Portal, y se encarga del control de la tarjeta rápida. Realiza la lectura del canal y determina el estado de la lectura de dicha entrada.

En la llamada a la función *Calc_velocidad* se encuentra el método de cálculo que, a partir de los pulsos recibidos, determina la velocidad en rpm. Dicha función recibe como parámetros el número de pulsos por vuelta, el tiempo transcurrido entre pulsos y la cuenta de pulsos en dicho tiempo. Devuelve el valor de la velocidad en rpm y un booleano que indica si se ha excedido el tiempo máximo entre pulsos.



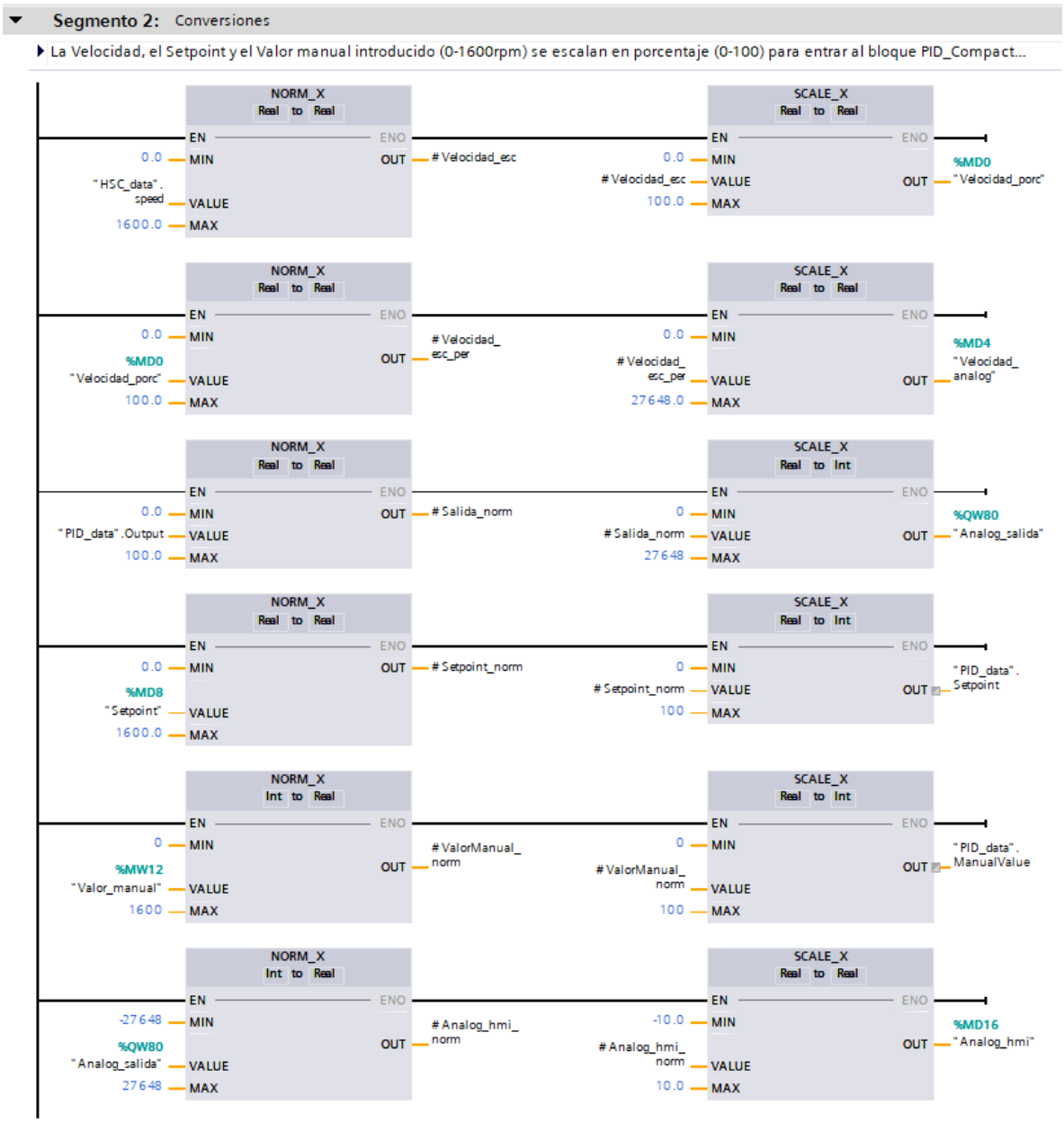
23 Segmento 1 del bloque *main*

En el **segmento 2** se realiza el escalado de todos los parámetros del programa.

La Velocidad (*HSC_data.speed*), el Setpoint *(Setpoint)* y el Valor manual (*Valor_manual)* introducidos en el HMI (0-1600rpm) se escalan en porcentaje (0-100) para entrar al bloque PID_Compact.

La *Velocidad_porc* y la salida del PID *(PID_data.Output)* (0-100) se escalan en valor analógico *(Analog_salida)* (0-27648). En el caso de la salida, este será el valor que actuará sobre el variador en V.

Por último, la analógica de salida *(Analog_salida)* se escala en Voltios para ser mostrados en el terminal HMI.



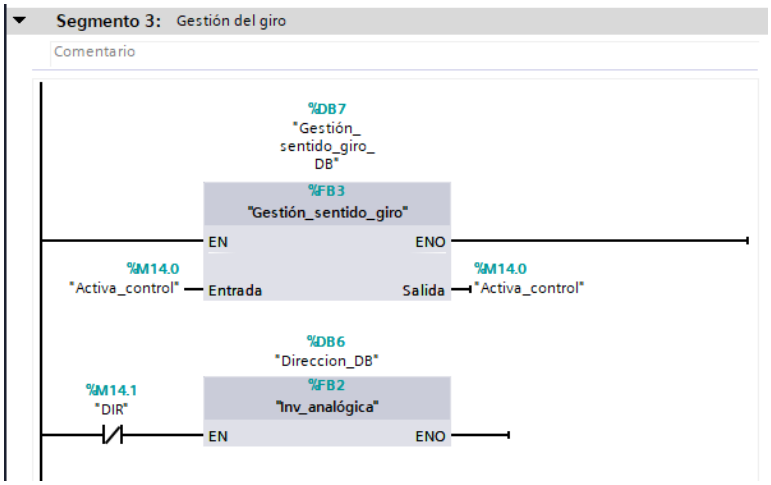
24 Segmento 2 del bloque *main*

En el **segmento 3** se gestiona el sentido de giro. Se realizan llamadas a dos funciones.

La primera de ellas, *Gestion_sentido_giro*, se encarga de gestionar el proceso cuando estamos controlando el sistema en una dirección y realizamos un cambio de sentido. Cuando esto sucede es necesario que el proceso esté controlado, si no el PID respondería a este cambio dando una salida del 100% en el sentido contrario, lo cual produciría un cambio brusco en la referencia que podría dañar el sistema. Por ello, esta función detiene el

control PID durante este proceso.

La segunda función tiene un único cometido que es cambiar el signo de la salida analógica si cambiamos la dirección a través de *DIR*. (Analog_salida := - Analog_salida)



25 Segmento 3 del bloque *main*

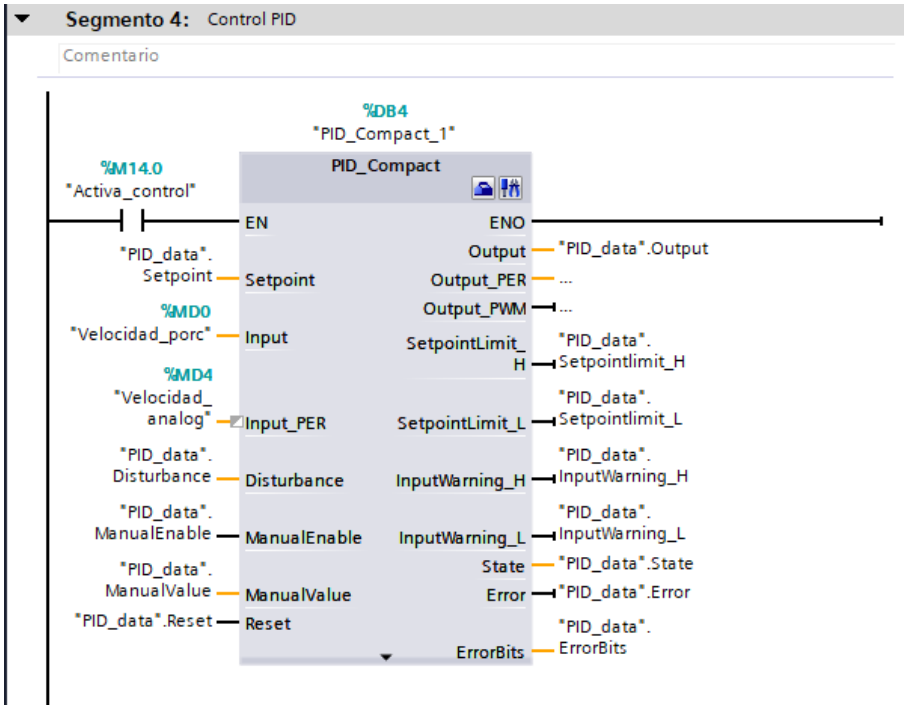
Por último, el **segmento 4** gestiona la llamada al bloque *PID_Compact*, es decir, el propio control. Como variables de entrada, encontramos:

- Setpoint. Consigna manual a alcanzar en el modo de regulación automático.
- Input. Entrada de la realimentación, en este caso la señal de velocidad, en porcentaje (0-100).
- Input_PER. Entrada de la realimentación, en variable analógica (0-27648).
- Disturbance. Permite el forzado de una perturbación a través de esta entrada.
- ManualEnable. Activa el modo manual.
- ManualValue. Valor de consigna en modo manual.
- Reset. Resetea el controlador.

Como variables de salida, tenemos:

- Output. Actuación del controlador, en pocentaje (0-100).
- Output_PER. Actuación del controlador, en variable analógica (0-27648)
- Setpointlimit_H. Límite superior del setpoint
- Setpointlimit_L. Límite inferior del setpoint
- Inputwarning_H. Umbral superior de alarma para la entrada
- Inputwarning_L. Umbral inferior de alarma para la entrada
- State. Estado del PID
- Error. Booleano que indica si hay error
- ErrorBits. Mensaje de error

Adicionalmente, se ha condicionado la señal de habilitación con la variable *Activa_control*, gestionada por el segmento 3.



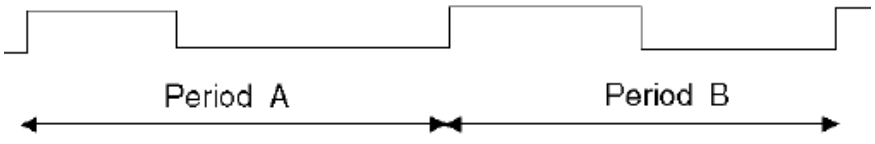
26 Segmento 4 del bloque *main*

3.3.1 Función de cálculo de la velocidad *(Calc_velocidad)*

Como se ha explicado en el punto anterior, el cometido de esta función es incluir un método de cálculo que determine la velocidad en rpm a partir de la entrada de pulsos.

La capacidad de hacer una correcta lectura de la velocidad de la forma más rápida y precisa posible es fundamental a la hora de implementar el control, pues cuanto más actualizada y fiable sea la realimentación del controlador, mayor será la exactitud y menor el retraso con el que se actúe. Esta, por lo tanto, es una etapa crítica en el proceso de control.

El método de cálculo se basa en medir el período transcurrido entre cada par de flancos. Conociendo esta magnitud y el número de pulsos por cada rotación completa, podemos obtener la velocidad. Realizar esta operación es posible gracias a la existencia de la función del sistema *CTRL_HSC_EXT*, la cual nos devuelve los datos necesarios para ello.



27 Periodo entre flancos del sensor

Se muestran en la siguiente figura las variables empleadas y la programación de la función.

Calc_velocidad

	Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible desde HMI/..	Escribible desde HMI..	Visible en HMI ..	Valor de ajuste	Comentario
1	Input				☐	☐	☐	☐	
2	numPulsePerRot	Int	3	No remanente	☑	☐	☑	☐	Nº pulsos por vuelta
3	elapsedTime	UDInt	0	No remanente	☐	☐	☐	☐	Tiempo transcurrido entre pulsos
4	edgeCount	UDInt	0	No remanente	☐	☐	☐	☐	Cuenta de flancos
5	Output				☐	☐	☐	☐	
6	overflow	Bool	false	No remanente	☑	☐	☑	☐	Alarma de sobretiempo
7	speed	Real	0.0	No remanente	☑	☐	☑	☐	Velocidad
8	InOut				☐	☐	☐	☐	
9	<Agregar>				☐	☐	☐	☐	
10	Static				☐	☐	☐	☐	
11	statPeriod	Real	0.0	No remanente	☐	☐	☐	☐	Periodo entre pulsos
12	statOverflow	Bool	false	No remanente	☐	☐	☐	☐	Sobretiempo
13	Temp				☐	☐	☐	☐	
14	<Agregar>				☐	☐	☐	☐	
15	Constant				☐	☐	☐	☐	
16	BILLION	Real	1.0E+09		☐	☐	☐	☐	
17	OVERFLOW_VALUE	UDInt	4294967295		☐	☐	☐	☐	Umbral de sobretiempo

28 Variables de la función *Calc_velocidad*

```
25
26  // Si hay sobretiempo, el periodo es 0
27 IF (#statOverflow = TRUE)
28  THEN
29      #statPeriod := 0;
30  END_IF;
31
32  // Convierte el periodo (tiempo para 1 vuelta), a rpm (vueltas para 1 min)
33 IF (#statPeriod > 0)
34  THEN
35      #speed := (1 / #statPeriod) * 60.0;
36  ELSE
37      #speed := 0;
38  END_IF;
39
40  // Indica sobretiempo como variable de salida
41  #overflow := #statOverflow;
```

29 Función *Calc_velocidad*

En el primer IF se calcula el periodo de giro del motor (tiempo en dar una vuelta):

$$Período = \frac{tiempo\ entre\ pulsos}{cuenta\ de\ flancos * 10^9} * pulsos\ por\ vuelta$$

Para obtener finalmente la velocidad en rpm, basta con hacer la siguiente conversión:

$$Velocidad\ (rpm) = \frac{1}{Período} * 60$$

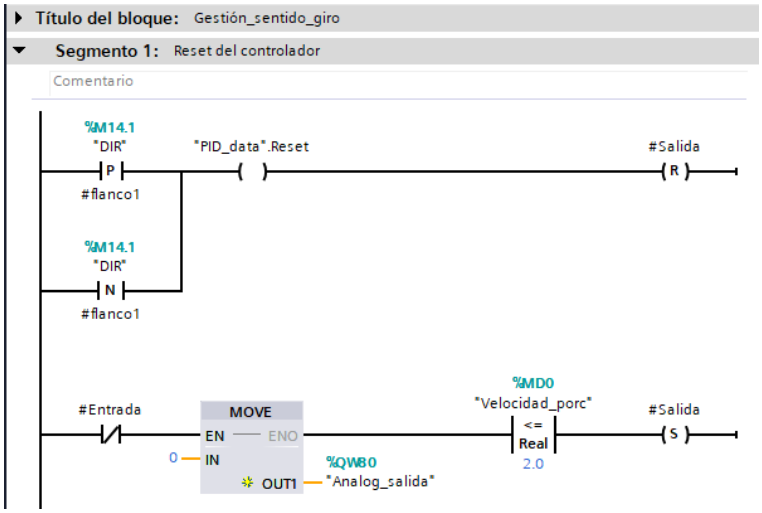
Por otro lado, se incluyen otros dos condicionantes para determinar si hay sobretiempo. En caso de que el tiempo transcurrido (*elapsedTime)* sea igual al umbral definido *(OVERFLOW_VALUE)*, se fija el periodo a 0 y se determina que la velocidad es 0. En el caso de nuestro programa, el umbral está definido en *4294967295*, es decir, ***4.3s***. En el caso de nuestro motor, resulta imposible que transcurra ese tiempo entre dos pulsos a no ser que el se encuentre en reposo.

3.3.2 Función de gestión del giro *(Gestión_sentido_giro)*

Como se resumió en el apartado 3.3.1, el cometido de esta función es gestionar el control del motor cuando se decide invertir el sentido de giro y el motor está en un régimen de vueltas determinado.

Cuando esto sucede, el control por defecto cambiaría bruscamente la referencia, transmitiendo una

sobreactuación al sistema y posiblemente dañándolo. Por ello, ante ese cambio esta función debe resetear el controlador, detenerlo, parar el motor y posteriormente activarlo cuando el sistema ya se haya quedado prácticamente en reposo.



30 Función *Gestión_sentido_giro*

Como se observa, la función reacciona cuando hay un flanco de subida ó bajada en la dirección, reseteando el controlador y enviando un reset a la salida (la salida es la señal de habilitación del control). Cuando el control está deshabilitado, se envía un 0 en la analógica de salida y se espera a que la velocidad sea prácticamente nula. Entonces, se restablece el control.

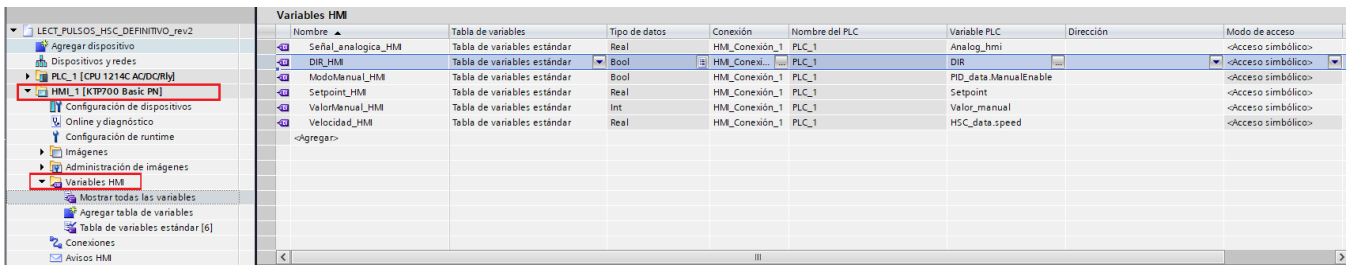
3.4 Pantalla de explotación (HMI)

En este apartado se abordará todo lo relacionado con la ventana de explotación desde la cual el usuario controla el sistema.

3.4.1 Programación

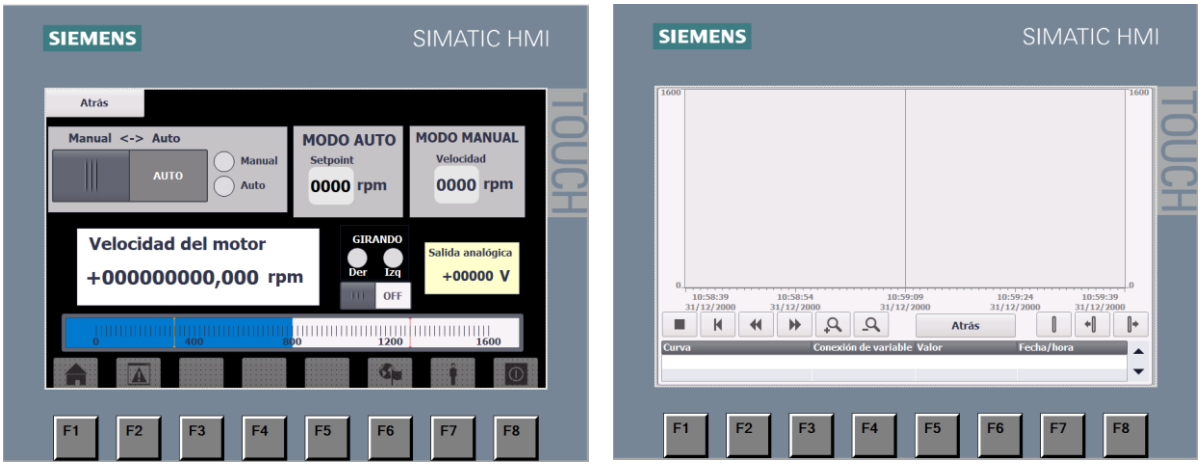
La programación de la ventana gráfica se ha realizado, al igual que la del autómata, a través del software *Tia Portal v15©*.

Para diseñar la ventana, en primer lugar se definen las variables de proceso que se van a mostrar/modificar a través de ella. Con ello, se define una lista de variables que se conecta con la lista de variables globales del autómata.



31 Listado de variables HMI

El siguiente paso sería la creación de la imagen, es decir, la interfaz con la que va a interactuar el usuario. En este proyecto se han programado dos imágenes: un panel de operación, que permite alternar entre modos de funcionamiento, enviar consignas, cambiar la dirección, visualizar la velocidad del sistema y la señal analógica enviada al variador. Por otro lado, la otra imagen consta de una gráfica en la que se representan el Setpoint y la Velocidad, a fin de monitorizar el sistema cuando se está controlando.



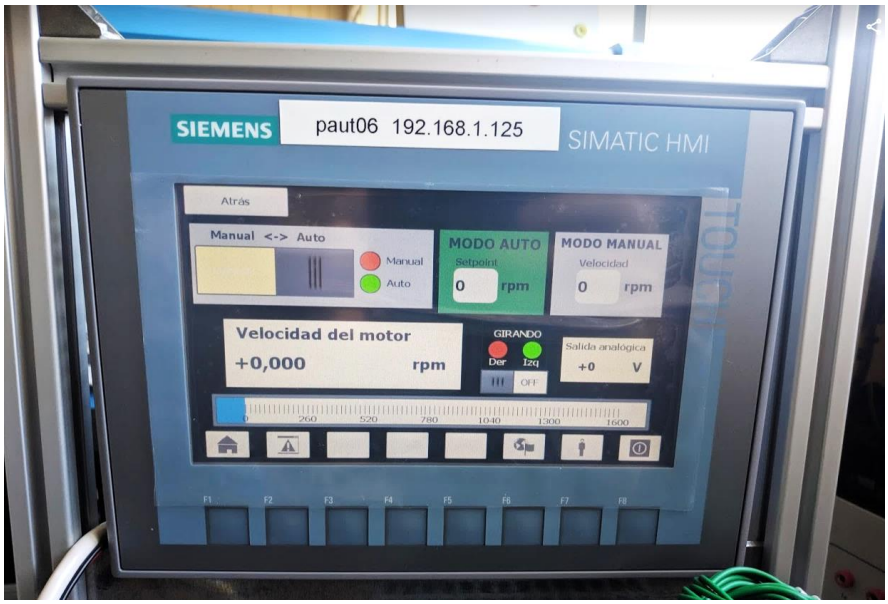
32 Imágenes programadas para el HMI

3.5 Funcionamiento del sistema

Realizando las conexiones, programación, y configuraciones indicadas, se procede a probar el control en funcionamiento.

Para realizar la prueba, en primer lugar, se va a llevar al sistema a un régimen de vueltas de 0 hasta 1300rpm, y observaremos la respuesta del mismo a través del terminal gráfico.

Cuando conectamos el sistema, éste se encuentra en modo manual. Pulsamos el selector *Manual <-> Auto* para alternar al modo automático. En ese momento el cuadro de entrada del setpoint se colorea de verde, lo cual quiere decir que podemos introducir una referencia.

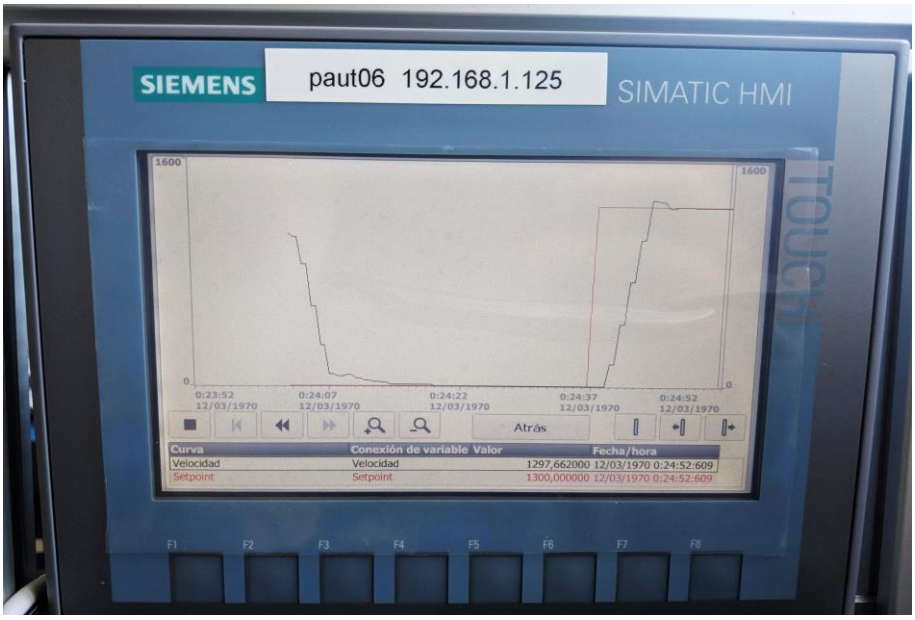


33 Vista del panel de operador previo a la introducción de la referencia

Una vez introducida la referencia, observamos cómo el sistema empieza a entregar voltaje al variador *(ventana salida analógica),* y la velocidad empieza a aumentar con ello.

34 Vista del panel de operador posterior a la introducción de la referencia

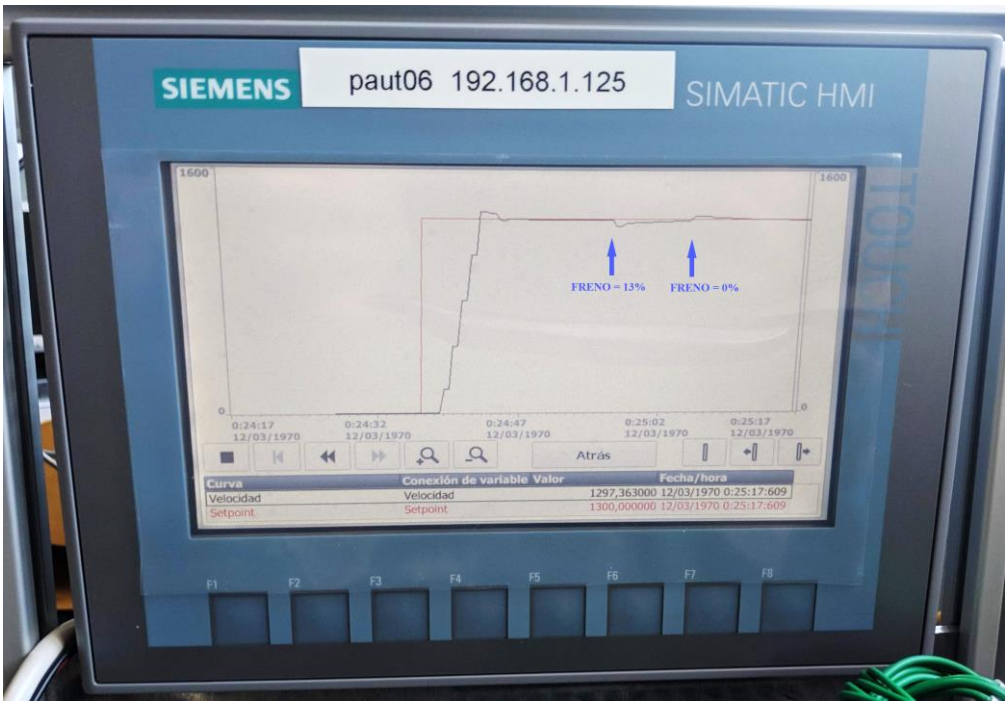
Podemos monitorizar la evolución de la velocidad en el tiempo, gracias al gráfico programado en la otra imagen.



35 Vista de la evolución velocidad-setpoint

En este punto en el que el sistema se ha estabilizado en torno al setpoint (1300rpm), vamos a añadir una perturbación real al sistema para observar cómo reacciona el controlador.

Para ello, vamos a activar el freno, que ejercerá un par opuesto al movimiento del motor y que tratará de reducir la velocidad. El control debería entregar más potencia al variador para alcanzar el setpoint. Posteriormente, vamos a soltar el freno, liberando ese par y haciendo que la velocidad aumente. Entonces el control debería reducir la potencia hasta volver a estabilizarse en torno al régimen permanente.



36 Vista del gráfico velocidad-setpoint posterior al uso del freno

4 SISTEMA DE CONTROL EN VELOCIDAD BASADO EN COMUNICACIÓN MODBUS RTU

En este apartado, se abordará la solución al problema de control a través de comunicación Modbus RTU, usando el autómata *Modicon M340* del fabricante *Schneider Electric*.

Para realizar este montaje será necesario el uso del software Unity Pro XL©, y del equipo mencionado en el apartado 2.6 de este documento.

4.1 Objetivo

El objetivo de este apartado es exactamente idéntico al que se pretendía en el apartado anterior, con la única salvedad de que, en este caso, en lugar de introducir la consigna de referencia al variador a través de señales analógicas de tensión, la consigna será enviada haciendo uso de un bus de comunicaciones Modbus serie (RTU), el cual ofrece suna serie de ventajas sobre el uso de la analógica.

La comunicación se realiza a través de un bus de comunicación serie de tipo RS485 en el cual, a diferencia de la comunicación Ethernet TCP/IP, sólo se usarán los pines 4, 5 y 7. El cable requerido para realizar dicha comunicación es un cable tipo RJ45 directo (no cruzado).

4.2 Montaje

Para poner en marcha el sistema se recomienda realizar una simulación de la planta a través de un modelo de primer orden no lineal, a fin de hacer una prueba del control y garantizar la eficacia del mismo.

4.2.1 Consideraciones previas

Antes de comenzar con el proceso de montaje del sistema, conviene hacer algunas consideraciones previas.

En primer lugar es necesario configurar la parametrización del variador.

Cuando se conecte el variador, ya sea por primera vez o bien esté configurado de otra forma, es recomendable la aplicación de una macroconfiguración (es decir, un modo de funcionamiento con configuraciones prestablecidas), en concreto el modo *Uso general*. Para ello, se debe seguir la ruta *CONF > CFG > Gen* en el display del variador.

A continuación, configurar las E/S físicas del variador siguiendo la siguiente tabla [4]:

NOMBRE	DESCRIPCIÓN	RUTA DE CONFIGURACIÓN
AI1	Canal de referencia 1 - Canal de control 1	CONF > FULL > CTL > FR1 > AI1 (*)
AI2	Canal de referencia sumatoria 2 al canal 1	CONF > FULL > FUN > OAI > SA2 > AI2 (*)
AI3	Canal de referencia sumatoria 3 al canal 1	CONF > FULL > FUN > OAI > SA2 > AI3 CONF > FULL > I_O >AI3- > CRL3 > 4.0 CONF > FULL > I_O >AI3- > CRH3 > 20.0 (*)

LI1	Marcha adelante	CONF > FULL > I_O > FRD > LI1(*)
LI2	Marcha atrás	CONF > FULL > I_O > RRS > LI2 (*)
LI3	Jog Frecuencia JOG	CONF > FULL > FUN > JOG-- > JOG > LI3 (*) CONF > FULL > SET-- > JGF > 10.0 (*)
LI4	Borrado de fallos	CONF > FULL > FLT- > RST > RPA > LI4
AQ1	Frecuencia de salida del motor en Hz Salida en intensidad 0-20 mA	CONF > FULL > I_O >AO1- > AO1 > OFR CONF > FULL > I_O >AO1- > AO1t > OA CONF > FULL > I_O >AO1- > AOL1 > 0.0 CONF > FULL > I_O >AO1- > AOH1 > 20.0
R1	ON cuando se detecta algún fallo	CONF > FULL > I_O > R1- > R1 > FLT
R2	ON cuando se habilita el funcionamiento del variador	CONF > FULL > I_O > R2- > R2 > RUN

Tabla 2. Tabla ajuste E/S y sus rutas de configuración

(*) No es necesaria la configuración manual al venir incluido en la macroconfiguración *Uso General*.

Además de los parámetros de E/S indicados en la tabla, es necesaria la configuración del parámetro que fija el modo de entrega de potencia del variador al motor.

Por defecto, el variador entrega la potencia al variador a intensidad constante, aumentando/disminuyendo la tensión según un perfil en rampa ascendente/descendente. Este perfil puede ser modificado en la configuración del variador.

Además, también permite modificar la pendiente de la rampa, permitiendo hacer la entrega de potencia con mayor o menor velocidad.

Esta parametrización se puede modificar en el display del variador accediendo a *CONF > FULL > FUN- > RPT-*. Para seleccionar el perfil, acceder al menú *rPt-*, y para seleccionar el tiempo de la rampa, al menú *Inr*.

Para más información acerca de la rampa de potencia, acuda al Manual de Programación [6], página 169.

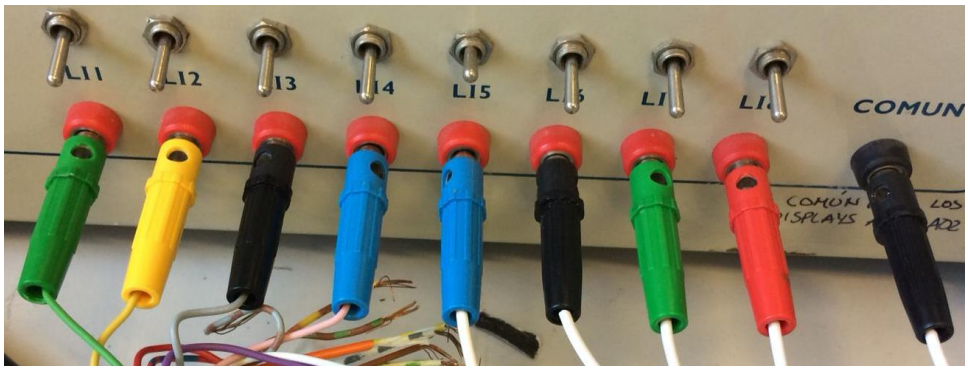
El software diseñado para esta aplicación posee únicamente posibilidad de comunicación Modbus, debido a la disponibilidad de tarjetas de E/S existente en el laboratorio en el que se va a emplear.

Sin embargo, es posible utilizar el modo de conexión analógica siempre que se configure el hardware, ya que el *Modo Analógico* ha sido añadido a la máquina de estados en la programación.

En la parte de hardware, se realizaría empleando tarjetas de salidas analógicas *(AMO 0210)* y una tarjeta de salidas digitales *(DDM3202K)*, y la conexión se realizaría utilizando una manguera con las bananas conectadas tal y como se muestra en la tabla.

BORNA	COLOR BANANA	CORRESPONDENCIA PLC
LI1	VERDE	Salida de la tarjeta digital %Q0.1.18
LI2	AMARILLO	Salida de la tarjeta digital %Q0.1.19
LI3	NEGRO	Salida de la tarjeta digital %Q0.1.20
LI4	AZUL	Salida de la tarjeta digital %Q0.1.21
LI5	AZUL	Salida en tensión 0-10 Vdc (%QW0.6.0)
LI6	NEGRO	Entrada en intensidad 4-20 mA (%IW0.4.0)
LI7	VERDE	Salida en intensidad 4-20 mA (%QW0.5.0)
LI8	ROJO	Salida en tensión ±10 Vdc (%QW0.5.1)

Tabla 3. Correspondencia colores de las bananas de entrada a los interruptores LIi



37 Entradas analógicas al variador

En la parte de software, como ya se ha comentado, existe la posibilidad de alternar entre los dos modos de funcionamiento.

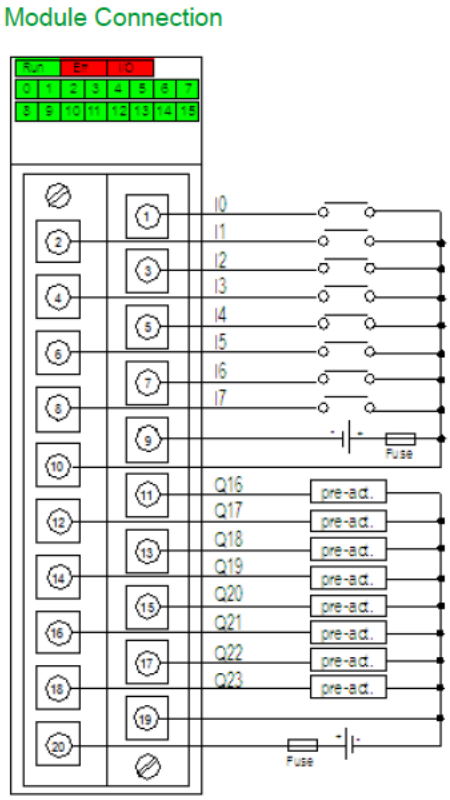
En este estado, las referencias se enviarían a través de las salidas analógicas y digitales del PLC.

Para conocer los detalles sobre cómo se realizaría la programación de dicho software, acudir a la referencia [4].

4.2.2 Proceso de montaje

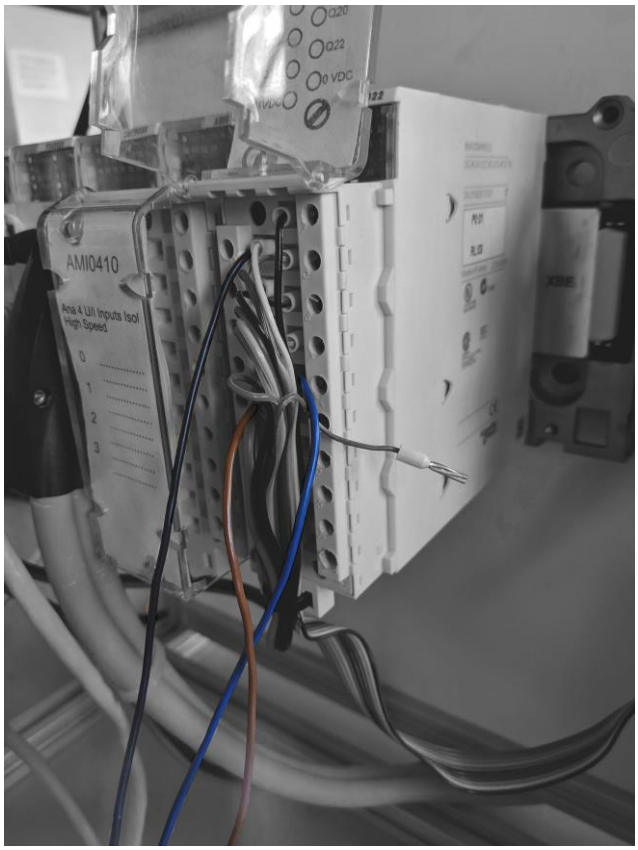
4.2.2.1 Montaje hardware

El montaje del sensor inductivo se realizará de forma análoga a como se hizo en el montaje del autómata S7-1200. Atendiendo al conexionado de la tarjeta *DDM16022* especificado en el datasheet [7]:



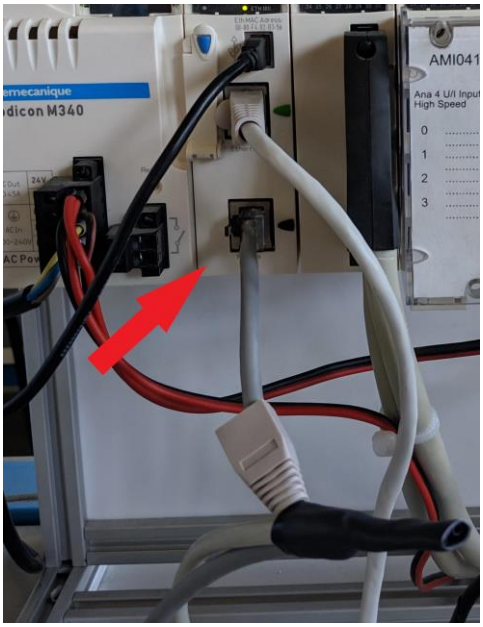
38 Conexionado *DDM 16022*

En este caso la señal controlada será recibida por dicha tarjeta, la cual ya tiene sus terminales (9) y (10) conectados a +24Vcc y a tierra, respectivamente, por lo que se debe conectar a esos dos terminales la alimentación del sensor, y al terminal (1) la señal de control.



39 Cableado sensor inductivo (a color)

Para el cableado de la comunicación Modbus se ha tomado una derivación de la red Modbus RTU existente en el laboratorio y conectarlo con el terminal RJ45 al puerto Modbus del PLC. Para conocer los detalles de la red Modbus existente en el laboratorio, acudir a la referencia [5].



40 Conexión Modbus RTU

4.2.2.2 Montaje sotware

En primer lugar, habría que realizar la configuración en el propio variador para adaptarlo a la comunicación Modbus serie con el PC. Para ello, se deben configurar los parámetros especificados en la siguiente tabla [4]:

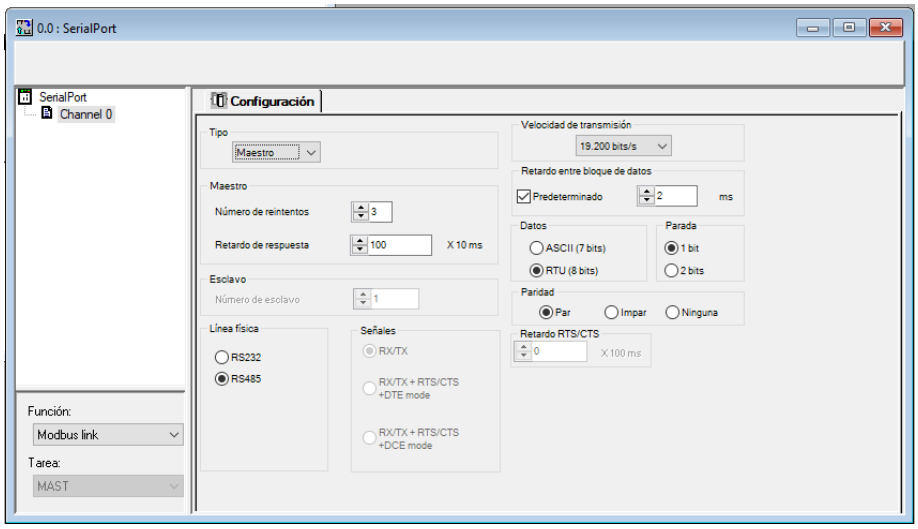
NOMBRE	DESCRIPCIÓN	RUTA DE CONFIGURACIÓN	VALOR
Add	Dirección de la trama Modbus	CONF > FULL > CON-- > ND1-- > ADD	1[1]
tbr	Velocidad de transmisión Modbus	CONF > FULL > CON-- > ND1-- > TBR	19.2 kbps
tFo	Formato de la trama	CONF > FULL > CON-- > ND1-- > TFO	8-E-1(**)
tto	Timeout Modbus	CONF > FULL > CON-- > ND1-- > TTO	10 s

Tabla 4. Parámetros de configuración Modbus del variador

(**) Estos tres dígitos hacen referencia a los ajustes de *Datos-Paridad-Parada* del Serial Port, respectivamente.

En el caso de la configuración en el PLC, habría que acudir al programa *Unity Pro XL©* y desplazarse a: *Explorador de Proyectos > Configuración > 0: Bus PLC > 0: BMX XBP 0800 > 0: BMX P34 2020 > Serial Port*.

En ese apartado, aplicar la configuración indicada en la figura.



41 Configuración puerto Modbus del PLC

[1] **IMPORTANTE**: Si usted cambia la dirección Modbus del variador (Add) mientras éste tiene tensión puede que existan fallos de comunicación. En caso de que se desee cambiar dicho parámetro, una vez cambiado quitar tensión al variador y volverlo a encender. Además, en el PC, habría que modificar el parámetro *"Direccion_Esclavo"* con la nueva dirección.

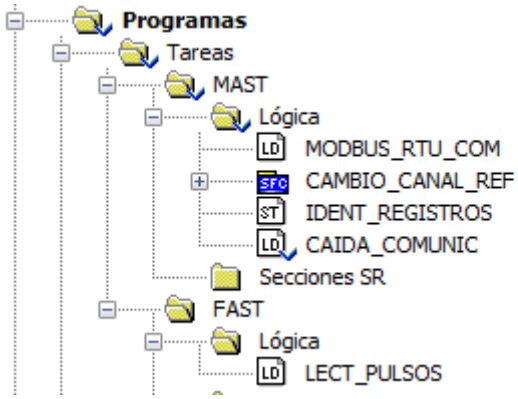
Encontrará información al respecto más extendida en la página 279 del manual de programación [6].

En el caso del sensor inductivo, la tarjeta a la que se conecta no se trata de una tarjeta rápida. Lo único a tener en cuenta es el direccionamiento de la entrada de pulsos del sensor en la %I0.3.1, de acuerdo con la configuración del hardware expuesta en el punto anterior.

4.3 Programación

En este apartado se detalla de la forma resumida el funcionamiento básico de la programación del autómata para la ejecución correcta del control.

Está compuesta por los siguientes bloques de programa:



42 Bloques de programa

4.3.1 Bloque de comunicación Modbus *(MODBUS_RTU_COM)*

El primero de los bloques a describir será el que hace posible la comunicación Modbus entre el PC (Maestro) y el variador (Esclavo).

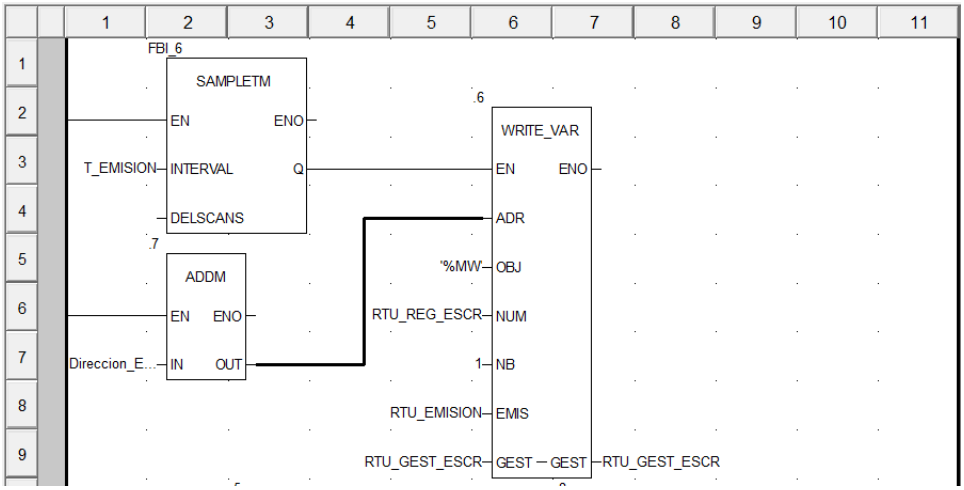
En este bloque se realizan las operaciones de lectura y escritura de parámetros.

Para la operación de escritura se emplea el bloque *WRITE_VAR*, que cuenta con los siguientes parámetros:

- *EN.* Habilitación del bloque
- *ADR.* Dirección en la que escribimos
- *OBJ.* Tipo de objeto que escribimos
- *NUM.* Dirección de registro en la que escribimos
- *NB.* Número de registros consecutivos a escribir
- *EMIS.* Búfer de transmisión
- *GEST.* Variable de entrada/salida. Vector que hace la gestión de la escritura.

Como la escritura no puede realizarse de forma continua (eso congestionaría el bus), vamos a emplear un bloque que realice la escritura en un tiempo de muestreo contenido en la variable *T_EMISION*.

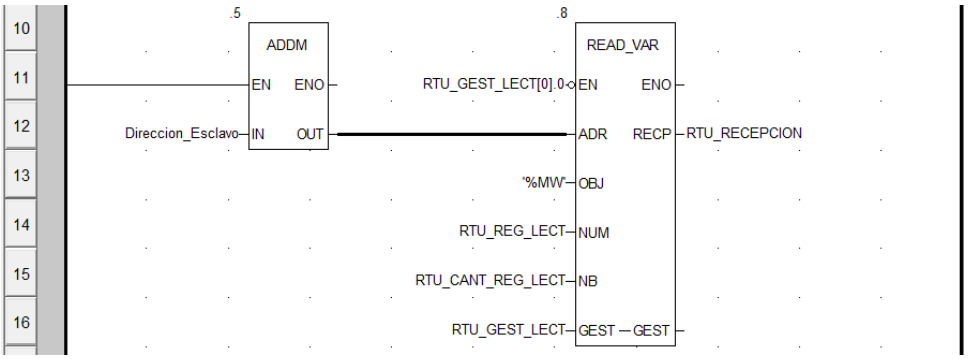
La dirección en la que se escribe vendrá especificada en la variable *DIRECCION_ESCLAVO*, que por defecto es *'0.0.0.1'*, de acuerdo con la configuración software realizada.



43 Operación de escritura Modbus RTU

Para la operación de lectura se emplea el bloque *READ_VAR*, que tiene los siguientes parámetros, cuya parametrización es similar a la del bloque *WRITE_VAR*, con alguna salvedad.

En este caso, la habilitación del bloque va a venir definida por el bit de habilitación de la variable de gestión del bloque. También existirá una variable de salida, el vector *RTU_RECEPCION,* cuyo tamaño corresponderá con la cantidad de registros que estemos leyendo.

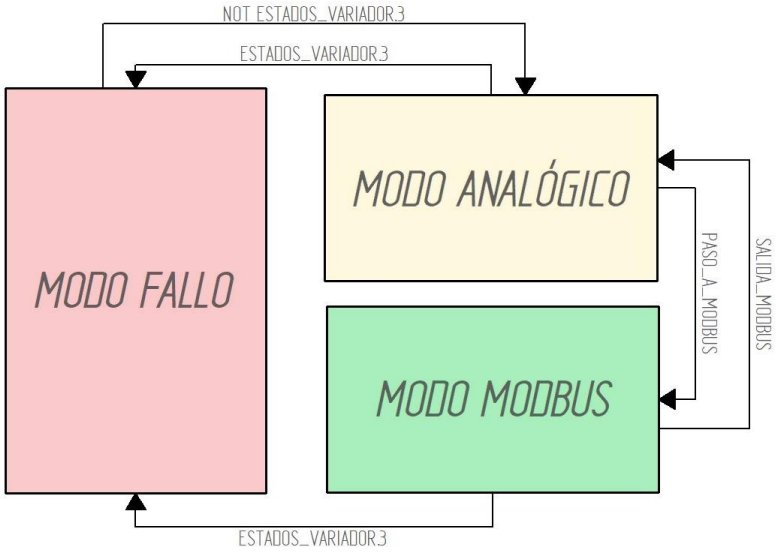


44 Operación de lectura Modbus RTU

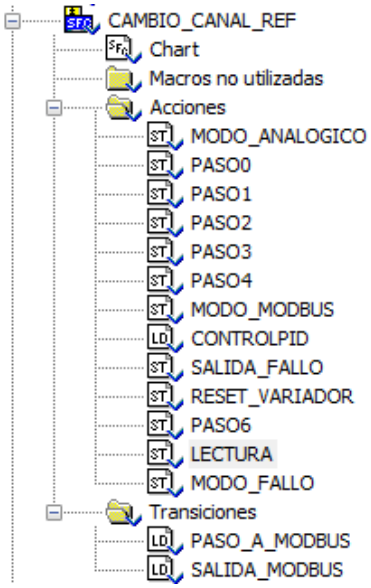
4.3.2 Bloque maestro *(CAMBIO_CANAL_REF)*

Este es el bloque principal, y tiene como objetivo principal definir en qué estado se encuentra el sistema. Permite alternar entre los dos modos de funcionamiento *(ANALÓGICO o MODBUS)*, y puede entrar en un modo adicional *(MODO FALLO)* en caso de que el variador entre en fallo, ya sea por un fallo en el propio variador, o por una caída en las comunicaciones.

Es por tanto, el que administra la máquina de estados que atiende a la guía GEMMA de la figura.



45 Guía GEMMA



46 Acciones/transiciones del bloque

Antes de describir el proceso de cambio entre estados, se van a definir las acciones más relevantes que se repiten en algunos lugares del bloque, referentes a la comunicación Modbus RTU.

4.3.2.1 Operación de lectura

Existen 3 modos de funcionamiento básicos: Modo Analógico, Modbus ó Fallo. En los tres modos se realiza la misma operación de lectura, a través de la acción *LECTURA*.

```
RTU_CANT_REG_LECT:=12;
RTU_REG_LECT:=3201;


ESTADOS_VARIADOR:=RTU_RECEPCION[0];
FREC_OUT:=RTU_RECEPCION[1];
REF_FRECUENCIA_ANTES_DE_LA_RAMPA:=RTU_RECEPCION[2];
CORRIENTE_ENTREGADA_INT:=RTU_RECEPCION[3];
PAR_ENTREGADO_INT:=RTU_RECEPCION[4];
ESTADOS_VARIADOR_AMPL:=RTU_RECEPCION[5];
```

47 Transición *LECTURA*

En todo momento se leen 12 registros (realmente sólo se almacenarán en variables los 6 primeros), a partir del 3201. La información acerca de los registros se puede ver con más detalle en el documento [8].

DIRECCIÓN	CÓDIGO	NOMBRE VARIABLE	NOMBRE	UNIDADES
3201	**[ETA]**	RTU_RECEPCION [0]	Registro de estados	-
3202	**[RFR]**	RTU_RECEPCION [1]	Frecuencia de salida	0.1Hz
3203	**[FRH]**	RTU_RECEPCION [2]	Ref. frec. antes de rampa	0.1Hz
3204	**[LCR]**	RTU_RECEPCION [3]	Corriente del motor	0.1A
3205	**[OTR]**	RTU_RECEPCION [4]	Par motor	0.1%
3206	**[ETI]**	RTU_RECEPCION [5]	Registro de estados ampliado	-

Tabla 5 Registros leídos a través de Modbus RTU

En concreto, los registros de estado **[ETA]** y **[ETI]** serán leídos bit a bit, ya que transmiten la información del estado de 16 booleanos. Los registros más significativos que van a ser empleados en nuestra programación son los expuestos en la tabla. El resto de registros pueden verse en la referencia [8].

Bit 0	Listo para encendido. (A la espera de entrega de potencia)
Bit 1	Encendido (Estado *ready*)
Bit 2	Operación activada (*running)*
Bit 3	Detección de falta
Bit 4	Presencia de voltaje en la alimentación.
Bit 5	Parada rápida / parada de emergencia
Bit 6	Alimentación de potencia bloqueada
Bit 7	Alarma
Bit 8	*Reservado*
Bit 9	Tipo de referencia:
	= 0 Comando ó referencia enviada a través de Ventana gráfica ó terminal gráfico.
	= 1 Comando ó referencia enviada a través de red.
Bit 10	Referencia alcanzada
Bit 11	Alcanzado límite interno
Bit 12	*Reservado*
Bit 13	*Reservado*
Bit 14	Tecla STOP pulsada
Bit 15	Dirección de rotación

Tabla 6. Registro de estados ETA *(ESTADOS_VARIADOR)*

Bit 0	Acceso a memoria EEPROM en ejecución.
Bit 1	Checkeo de consistencia de parámetros.
Bit 2	= 0: El variador no se encuentra en estado de fallo ó fallo detectado.
	= 1: Fallos despejados pero el variador continúa en estado de fallo.
Bit 3	Reserved (= 0).
Bit 4	Variador en modo velocidad.
Bit 5	Frenada regenerativa,
Bit 6	Variador en régimen permanente (0) / transitorio (1)
Bit 7	Alcanzado límite umbral de temperatura del motor.
Bit 8	Sobrefrenado.
Bit 9	Aceleración en curso.
Bit 10	Deceleración en curso.
Bit 11	Límite de corriente.
Bit 12	Parada rápida en curso.
Bit 13	Bit 13 = 0 y Bit 14 = 0: Variador controlado por terminal ó teclado local.
Bit 14	Bit 13 = 1 y Bit 14 = 0: Variador controlado por teclado remoto.
	Bit 13 = 0 y Bit 14 = 1: Variador controlado a través de Modbus.
	Bit 13 = 1 y Bit 14 = 1: Variador controlado a través de CANOpen ó tarjeta de red.
Bit 15	=0: Operación hacia delante aplicada antes de la rampa.
	=1: Operación hacia detrás aplicada antes de la rampa.

Tabla 7. Registro de estados ampliado ETI *(ESTADOS_VARIADOR_AMPL)*

REGISTRO	BIT	NOMBRE VARIABLE	DESCRIPCIÓN
[ETA]	**1**	RTU_RECEPCION [0].1	Estado [RdY] en el variador
[ETA]	**3**	RTU_RECEPCION [0].3	Variador en fallo
[ETA]	**15**	RTU_RECEPCION[0].15	Sentido de giro
[ETI]	**6**	RTU_RECEPCION[5].6	Variador en régimen permanente/transitorio
[ETI]	**9**	RTU_RECEPCION[5].9	Variador acelerando
[ETI]	**10**	RTU_RECEPCION[5].10	Variador decelerando

Tabla 8. Registros de estado leídos a través de Modbus RTU

4.3.2.2 Operación de escritura

Lo que caracteriza a cada uno de los tres modos de funcionamiento es, básicamente, la operación de escritura que realiza sobre el variador.

En el **modo Analógico** se escribirá en el registro 8413 [FR1], que activa el Canal de Referencia 1.

En el **modo Modbus** se escribe en dos registros en el 8501 [CMD] y 8502 [LFR]. El primero de ellos permite realizar operaciones sobre el variador modificando los bits de dicho registro, y el segundo de ellos es en el que escribiremos la referencia en frecuencia que será enviada al variador.

En el **modo Fallo** se escribe sobre dos registros distintos. En la acción *SALIDA_FALLO* se escribe sobre el registro 8501 [CMD], en concreto sobre el bit 3, que permite sacar al variador del estado de fallo de comunicaciones. Por otro lado, en la acción *RESET_VARIADOR* se escribe sobre el registro 7128 [RP], en concreto sobre el bit 1, que realizaría un reset del variador y lo sacaría de cualquier otro tipo de fallo.

El contenido a escribir en los registros se almacena en las variables *RTU_EMISION[1]* y *RTU_EMISION[2]*.

Para más información acerca de los registros mencionados, acudir al Manual de Programación y a la lista de parámetros de comunicación del variador, referencias [6] y [8].

4.3.2.3 Funcionamiento general

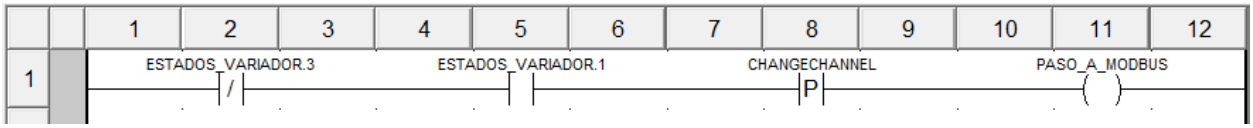
Para la definición de los distintos modos de funcionamiento, se ha definido un entero, *MODO_OPERACION,* que definirá el modo en el que estamos según su valor.

VALOR	MODO DE OPERACIÓN
0	Sin conexión
1	Alternando entre modo Modbus-Analógico
2	Modo Modbus
3	Modo Fallo
4	Saliendo del Modo Fallo
5	Haciendo *RESET* al equipo
6	Modo Analógico

Tabla 9. Valores posibles de la variable *MODO_OPERACION*

Por defecto, el sistema arrancará en Modo Analógico. En este modo, como se ha explicado en el apartado 4.2.1, se debería poder enviar una referencia analógica al variador a través de una tarjeta de salidas analógicas.

Para conmutar al Modo Modbus, es necesario activar la variable CHANGECHANNEL. Sin embargo, esta conmutación no sucederá si el variador encuentra un fallo (*ESTADOS_VARIADOR.3*) ó no se encuentra en estado [RdY] (Variador listo) [6].

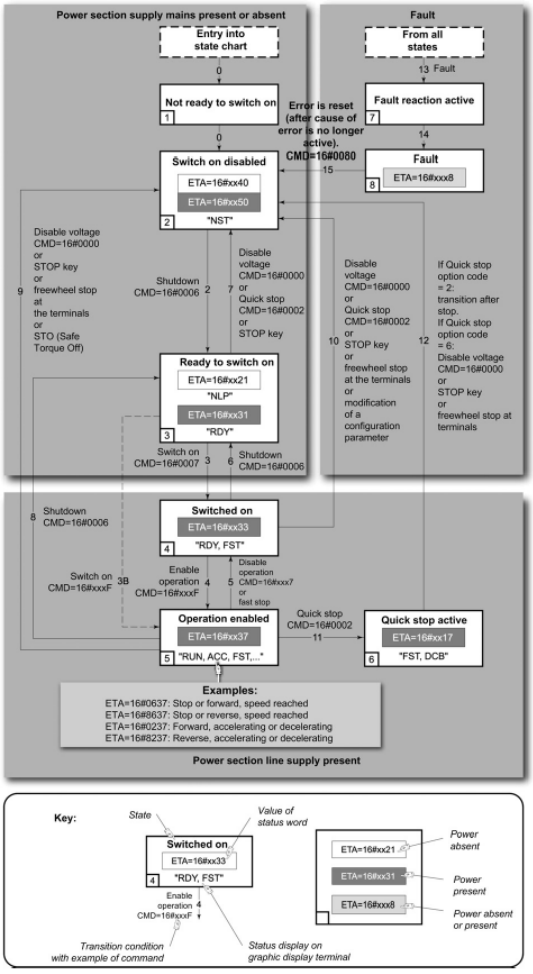


48 Transición *PASO_A_MODBUS*

Si se cumple la transición *PASO_A_MODBUS*, comienza la transición al Modo Modbus. Para realizar este cambio entre modos, el fabricante indica una secuencia de activación que hay que seguir para que el cambio se

produzca con éxito.

En la página 42 del manual Modbus del variador [9] encontramos la secuencia a seguir para hacer el cambio entre modos.



49 Secuencia de activación de los distintos modos de funcionamiento a través de Modbus

Una vez realizada la secuencia y pasado al modo Modbus, el variador debe mostrar los caracteres *0.0* en el display. Esto significa que está en Modo Modbus y a la espera de una nueva referencia.

Como se ha anticipado en el punto 4.3.2.2, en este modo enviaremos referencias de velocidad al registro 8502 [LFR], a través de la variable *VELOC_MODBUS*. Esto se realiza en la acción *MODO_MODBUS*.

```
MODO_OPERACION:=2;
RTU_REG_ESCR:=8501;
RTU_EMISION[2]:=VELOC_MODBUS;
T_EMISION:=t#1s;
```

50 Acción *MODO_MODBUS*

La variable *VELOC_MODBUS* vendrá definida de un modo u otro dependiendo de si vamos a controlar el sistema en manual o vamos a regular a través de un PID. Esta parte del programa guarda una cierta similitud con el control a través del autómata Siemens del punto anterior, al tratarse del mismo control, aunque el método de

envío de la referencia cambia.

Para realizar dicho control se ejecuta paralelamente otra acción en este estado, *CONTROL_PID*, en la que se realizan varias operaciones.

En primer lugar, debe aclararse que en este apartado se ha seguido la misma filosofía respecto al escalado de variables (aunque el bloque *PIDFF* en este caso ofrezca flexibilidad en ese aspecto). Por tanto, las variables referentes a velocidad que visualizamos en la ventana de explotación (en rpm, 0-1500) serán escaladas a valores en porcentaje (0-100), para volver a realizar el escalado de la señal de salida una vez se ha realizado el control. En este caso, la actuación del PID se escalará a un valor comprendido entre 0 y 600, al enviarse la referencia en décimas de Hz, tal y como especifica el registro 8502 [LFR] [8].

Por tanto, se realiza en primer lugar un escalado de las variables que se muestran por la pantalla de explotación, bien para introducirlas como consigna (*SETPOINT ó CONSIGNA_MANUAL*) o bien como visualización *(VELOCIDAD).*

A continuación se incluye un bloque PID (*PIDFF)* que realiza el control, con la parametrización dada por la variable *PARAM_PID,* por defecto parametrizada con *Kp=0.3, Ti=2s, Td=0s*, como se indica en la figura.

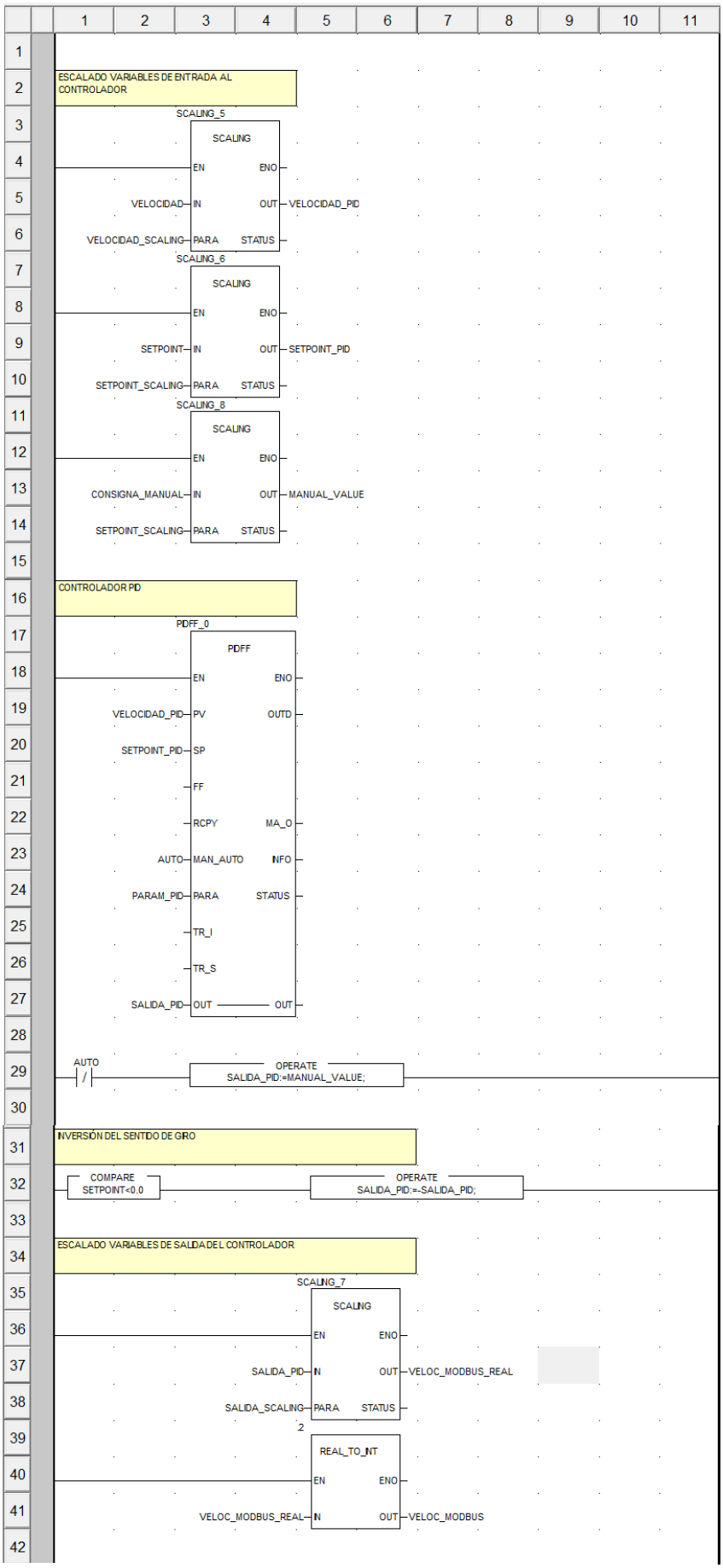
Finalmente, se ajusta el signo de la actuación del PID *(SALIDA_PID)* en función del sentido de giro, y se vuelve a escalar y convertir la actuación del PID *(SALIDA_PID)* para ser enviada al variador en décimas de Hz y de tipo entero.

Variables | Tipos de DDT | Bloques de funciones | Tipos de DFB

Filtro — Nombre = PARAM_PID

Nombre	Tipo	Valor	Comentario
PARAM_PID	Para_PIDFF		
id	UINT		
pv_inf	REAL	0.0	Límite inferior de la variable de proceso
pv_sup	REAL	100.0	Límite superior de la variable de proceso
out_inf	REAL	0.0	Límite inferior de la actuación
out_sup	REAL	100.0	Límite superior de la actuación
rev_dir	BOOL	FALSE	
mix_par	BOOL	FALSE	
aw_type	BOOL	FALSE	
en_rcpy	BOOL	FALSE	
kp	REAL	0.3	Kp
ti	TIME	t#2s	Ti
td	TIME	t#0s	Td
kd	REAL		
pv_dev	BOOL		
bump	BOOL		
dband	REAL		
gain_kp	REAL		
ovs_att	REAL		
outbias	REAL		
out_min	REAL	0.0	
out_max	REAL	100.0	
outrate	REAL		
ff_inf	REAL		
ff_sup	REAL		
otff_inf	REAL		
otff_sup	REAL		

51 Estructura *PARAM_PID*



52 Acción *CONTROLPID*

Si en alguno de los dos modos de funcionamiento (*Analógico/Modbus*) el variador entra en fallo (*ESTADOS_VARIADOR.3=TRUE),* se conmuta al Modo Fallo, y el programa quedará a la espera de que el usuario realice alguna acción para salir de ese modo (Acción *MODO_FALLO*).

O bien realiza un reset del equipo mediante *BOTON_RESET (*Acción *RESET_VARIADOR)* o bien un borrado de fallo de comunicación mediante *BOTON_SALIDA_FALLO (*Acción *SALIDA_FALLO)*. Si se pulsa el botón de salida de fallo y el variador continúa sin salir del fallo, se puede efectuar un reset del equipo para salir del fallo.

Tanto la acción *RESET_VARIADOR* como la acción *SALIDA_FALLO* envían un 1 al registro correspondiente especificado en la sección 4.3.2.2, para realizar la acción correspondiente.

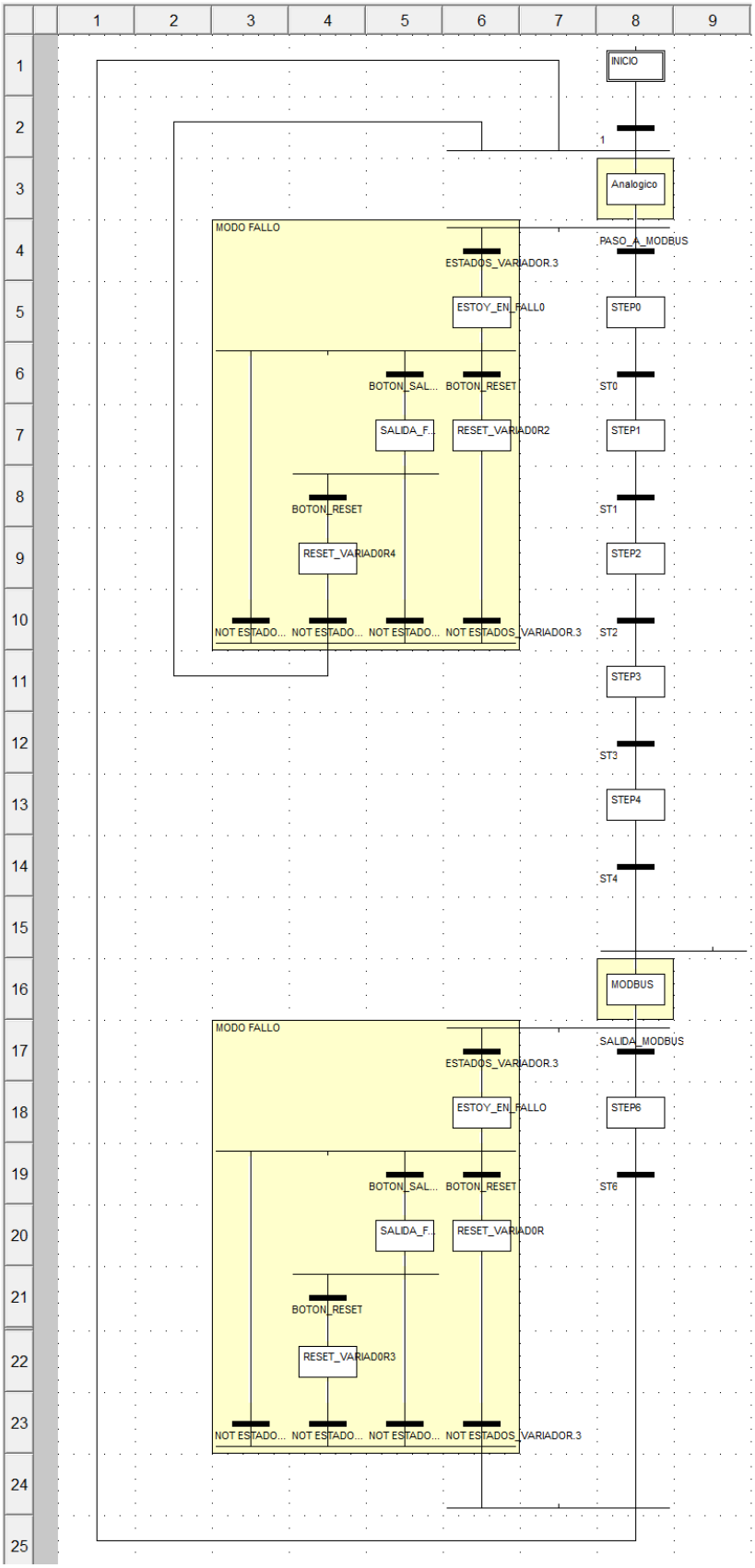
```
MODO_OPERACION:=3;
```

```
RTU_REG_ESCR:=7128;
RTU_EMISION[1]:=1;
T_EMISION:=t#1s;

MODO_OPERACION:=5;
```

```
RTU_REG_ESCR:=8501;
RTU_EMISION[1].7:=1;
T_EMISION:=t#1s;

MODO_OPERACION:=4;
```

53 Acciones: *MODO_FALLO(1), RESET_VARIADOR(2), MODO_FALLO(3)*

Finalmente, para volver del modo Modbus al modo Analógico, se realiza igualmente la secuencia indicada para ello en el Manual de Modbus [9].



54 Bloque *CAMBIO_CANAL_REF* completo

4.3.3 Bloque de identificación de registros *(IDENT_REGISTROS)*

Este bloque, programado en ST, tiene un único cometido y es activar las transiciones correspondientes para alternar entre los pasos de la secuencia de activación/desactivación del modo Modbus.

```
(*Sección para facilitar el trabajo
con los registros leídos*)
CASE RTU_RECEPCION[0] OF
164: ST0:=1; (*Enviamos un 164 al registro del Canal de referencia*)
     ST1:=0;
     ST2:=0;
     ST3:=0;
     ST4:=0;
     ST6:=0;
561: ST0:=0;
     ST1:=1;(*Enviamos un 6 al CMD*)
     ST2:=0;
     ST3:=0;
     ST4:=0;
     ST6:=0;
563: ST0:=0;
     ST1:=0;
     ST2:=1;(*Enviamos un 7 al CMD*)
     ST3:=0;
1:   ST0:=0;
     ST1:=0;
     ST2:=0;
     ST3:=1;(*Enviamos una referencia al LFR*)
     ST4:=0;
567: ST0:=0;
     ST1:=0;
     ST2:=0;
     ST3:=0;(*Enviamos un 15 aL CMD*)
     ST4:=1;
     ST6:=0;
1591: ST0:=0;
     ST1:=0;
     ST2:=0;
     ST3:=0;(*Enviamos un 15 aL CMD*)
     ST4:=1;
     ST6:=0;
592: ST0:=0;
     ST1:=0;
     ST2:=0;
     ST3:=0;(*Enviamos un 15 aL CMD*)
     ST4:=0;
     ST6:=1;
END_CASE;
```

55 Bloque *IDENT_REGISTROS*

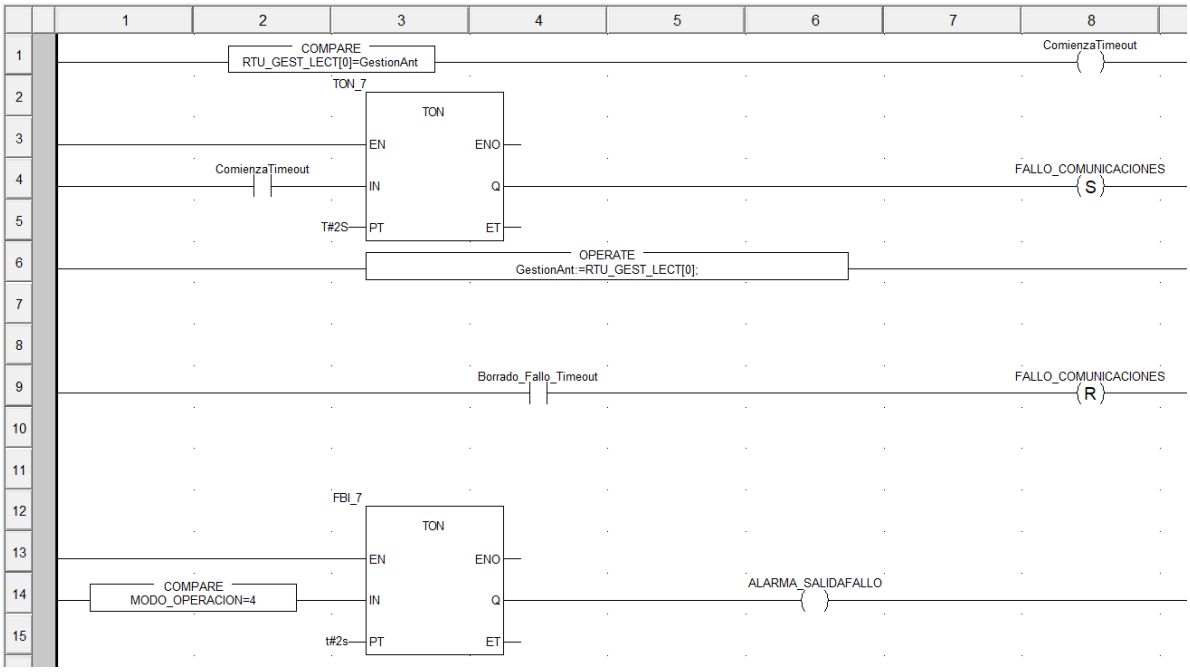
4.3.4 Bloque de administración de la comunicación *(CAIDA_COMUNIC)*

El objetivo de este bloque es detectar cuándo ha caído la comunicación, y por cuánto tiempo.

Se basa en que cuando la comunicación está activa y estable, el bit de estado de la matriz de gestión de lectura cambia de forma rápida, parándose cuando la comunicación se interrumpe. Por tanto, cuando este bit no cambie durante más de 2 segundos, se considerará que la comunicación ha caído y se lanzará la alarma *FALLO_COMUNICACIONES*.

Para borrar dicha alarma, cuando se reestablezca la comunicación, habría que pulsar el botón *Borrado_Fallo_Timeout*.

Además, si el sistema lleva más de 2 segundos intentando salir del modo fallo *(MODO_OPERACION=4)* sin éxito, activará otra alarma *(ALARMA_SALIDAFALLO)* en la que se indicará que no se está saliendo del fallo con éxito y se sugerirá realizar un RESET del variador.

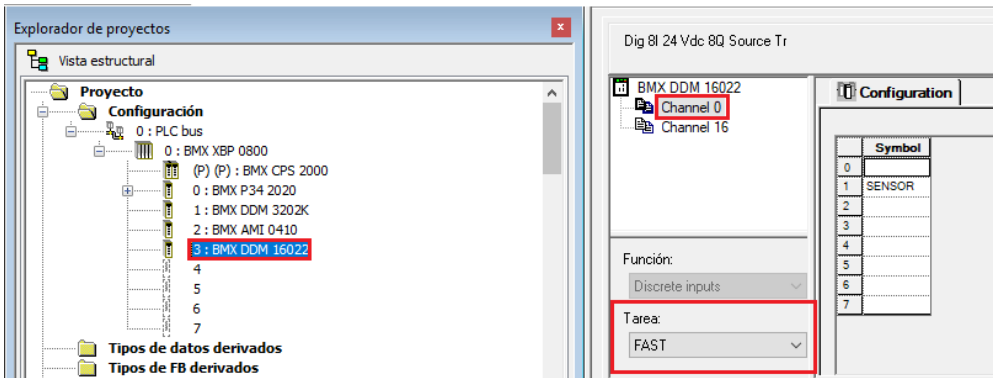


56 Bloque *CAIDA_COMUNIC*

4.3.5 Bloque de lectura de pulsos *(LECT_PULSOS)*

Antes de describir el presente bloque, cabe aclarar que éste ha sido programado como una tarea *FAST*, es decir, su tiempo de ciclo es menor al del resto de bloques. Esta es una funcionalidad que permite la programación del PLC M340 con el software Unity PRO. Permite que lo programado en esta parte del programa tenga un ciclo de programa menor al del resto de bloques, permitiendo incluir aquí las acciones más críticas. (Esta funcionalidad es permitida gracias a la CPU multi-núcleo que se incorpora.

Para ello, previamente se ha establecido dicha configuración en la tarjeta de E/S discretas DDM 16022 en la opción de la barra lateral *Configuración > PLC bus > BMX XBP 0800 > BMX DDM 16022,* en ese menú buscamos *Channel 0 > Tarea* y seleccionamos *FAST*.



57 Configuración tarea FAST

El objetivo de la función *LECT_PULSOS* es, al igual que en el apartado 3 de este documento, realizar de la forma más precisa y exacta posible, ya que cuanto más actualizada y fiable sea esta medida, mayor será la exactitud y menor será el retraso con el que se actúe. Esta sección es crítica para el correcto funcionamiento del control.

Si se dispusiera de una tarjeta rápida que realice la lectura de pulsos en alta frecuencia, como es el módulo de conteo *BMX EHC 0200*, podría replicarse con facilidad el método de conteo de pulsos del apartado anterior, basado en el cálculo del período entre pulsos del sensor, ofreciendo una medida actualizada y poco ruidosa de la velocidad.

En el caso de disponerse de la tarjeta mencionada, podría realizarse esta sección del programa siguiendo las indicaciones de la página 81 del Manual de Usuario [10] de dicho módulo.

58 Cálculo del período de la tarjeta rápida EHC0200

Al no disponerse en nuestro caso de dicha tarjeta, no es posible la programación de funciones ni variables que nos aporten los parámetros suficientes como para replicar el método de cálculo del apartado anterior.

Debido a esto, se va a emplear un método de cálculo alternativo que nos dará una señal menos precisa y actualizada. Consiste, inversamente al método anterior, en medir la cantidad de pulsos leídos en un período fijo. Por lo tanto, el tiempo de muestreo para el refresco de la medida es precisamente ese período. Cuanto menor sea ese tiempo, más actual será la medida pero a su vez será más imprecisa, pues se cuentan menos pulsos.

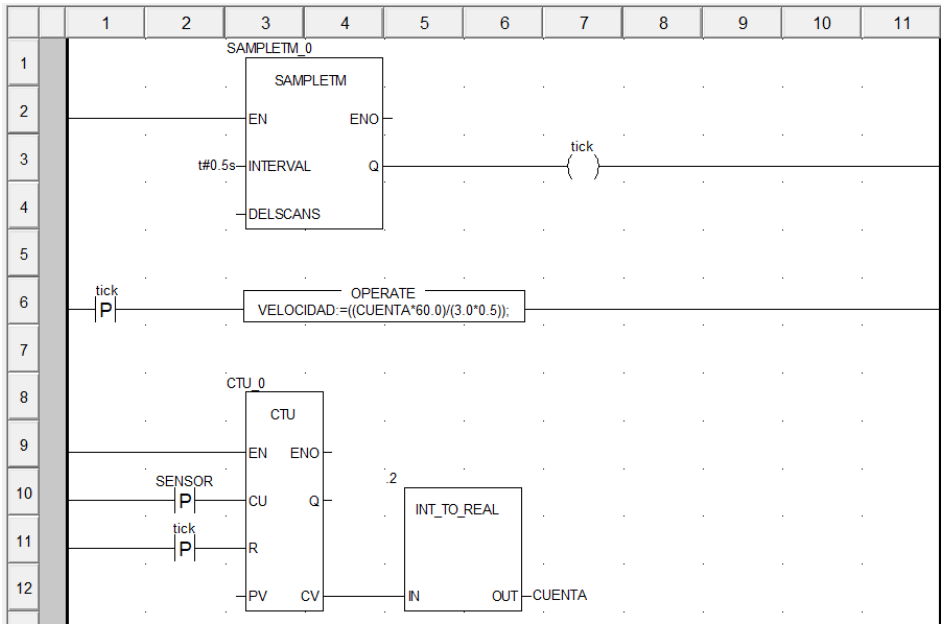
Y la fórmula que resume esta deducción es la que sigue.

$$Velocidad\ (rpm) = \frac{N * 60}{n * T}$$

Siendo n el número de pulsos por rotación completa, T el período en segundos y N el número de pulsos contados en el período.

El programa funciona de la siguiente manera. Un bloque contador *CTU* cuenta pulsos en la variable *SENSOR* (conectada a la entrada física *%I0.3.1*), y por cada 0,5s se genera un pulso virtual mediante el bloque *SAMPLETM*. En cada golpe de tick se hace el cálculo de la velocidad con la cuenta de pulsos primero y posteriormente se resetea el contador.

El período escogido en las demostraciones es de 0,5 segundos.



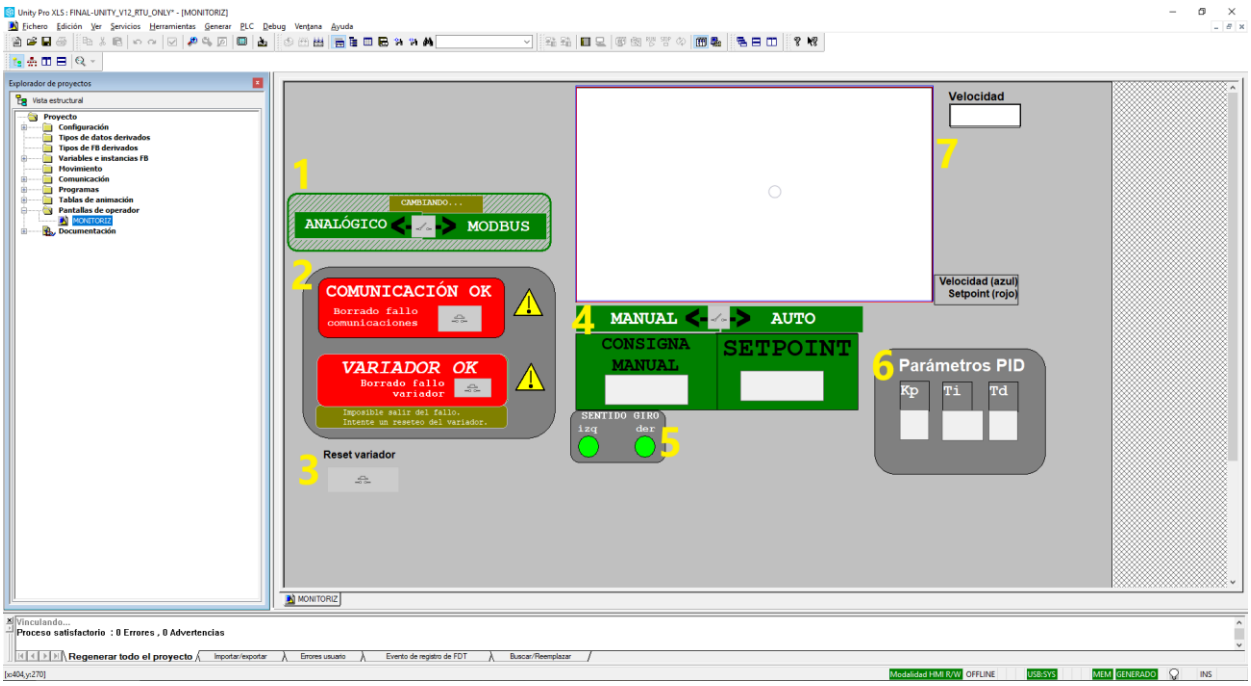
59 Bloque *LECT_PULSOS*

4.4 Pantalla de explotación

En este caso, como ya se ha indicado, no se dispone de un terminal físico HMI, por lo que se usa la estación PC a modo de HMI, siendo equivalente en términos de funcionalidad.

La ventana posee selectores y campos de entrada que permiten modificar los parámetros que gobiernan el programa.

- (1) Selector de modo Analógico <-> Modbus. Pulsador con retención que alterna el valor de la variable *CHANGECHANNEL*. El modo activo será el lado que se coloree de verde.
- (2) Panel de alarmas. Panel en el que se muestra si la comunicación ha caído o, en su defecto, si la comunicación es correcta, si el variador se encuentra en fallo o no. En caso de que no esté en fallo se colorearían de verde, y disponen de pulsadores sin retención para el borrado de fallos. Si se pulsa el botón de borrado de fallo en el variador y no se elimina el fallo en un tiempo de 2 segundos, aparecería la ventana inferior que sugiere hacer un reset.
- (3) Botón reset del variador
- (4) Selector de modo Manual <-> Automático. El pulsador con retención sólo aparece en caso de que nos encontremos en modo Modbus. En tal caso, se coloreará de verde el lado en el que nos encontremos. Los campos de entrada nos permiten introducir la consigna manual / el setpoint.
- (5) Indicador sentido de giro
- (6) Parámetros PID. Por defecto posee la parametrización Kp=0.3, Ti=2s, Td=0s; pero permite modificar dichos parámetros durante la ejecución del programa. Se recomienda alternar al modo manual antes de modificar los parámetros, para evitar picos de tensión que puedan dañar el sistema.
- (7) Monitor de velocidad. Gráfica en la que se muestra el Setpoint (rojo) frente a la velocidad actual en rpm (azul). También dispone de un display en el que se observa el valor numérico de la velocidad.



60 Pantalla de explotación diseñada

Las pruebas de funcionamiento, así como las curvas de respuesta del sistema, podrán apreciarse en el apartado 5 del presente documento.

5 IMPLEMENTACIÓN DE UN SISTEMA REDUNDANTE DE CONTROL A TRAVÉS DE UNA RED MODBUS TCP

Este apartado surge como ampliación del apartado anterior, y surge de la necesidad de incrementar la disponibilidad del sistema, para cubrir la demanda de actuaciones críticas dentro de la industria.

Para realizar este montaje será necesario el uso del software Unity Pro XL©, y del equipo mencionado en el apartado 2.6 de este documento.

5.1 Objetivo

El objetivo en esta fase es implementar el mismo algoritmo de control (salvando ciertas diferencias) que el expuesto en el apartado anterior en varias estaciones.

La particularidad en este montaje es que se realiza en una configuración redundante. En caso de que la comunicación Modbus RTU entre el autómata y el actuador (en este caso, el variador) sea interrumpida, otro autómata inmediatamente inferior en prioridad tomará el testigo y continuará regulando el sistema, sin que el actuador vea interrumpida su ejecución en ningún momento.

En el momento en el que el PLC más prioritario establezca de nuevo la comunicación, éste será capaz de recuperar el control del actuador sobre los PLCs de menor prioridad que en ese momento estuviesen operando.

Si por el contrario, se da el improbable caso en el que las comunicaciones de todos los PLC cayeran, el variador parará el motor por seguridad y hará disparar una señal de error. Cuando el usuario reestablezca la comunicación con cualquiera de las estaciones, podrá enviar una señal digital para eliminar dicho error, y restablecer el modo de funcionamiento en el variador.

5.2 Montaje

5.2.1 Funcionamiento básico

Para entender mejor el funcionamiento de la red Modbus TCP, se presentan a continuación algunas pautas básicas sobre el montaje del sistema.

El principio de funcionamiento para que el sistema de control trabaje en sincronía es la compartición de datos, parámetros y señales entre las estaciones a través de una red bajo el protocolo Modbus TCP. El montaje físico y el cableado de la propia red puede verse con detalle en la referencia [5].

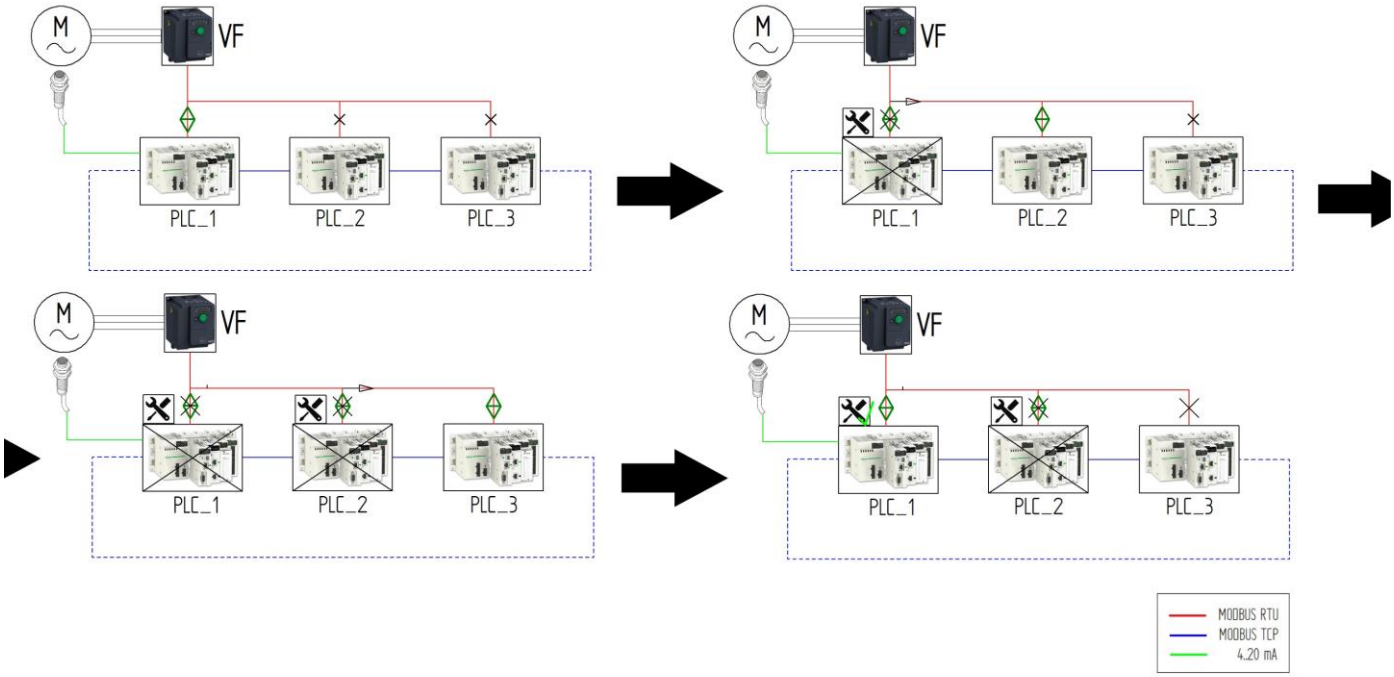
Todas las estaciones están conectadas entre sí a través de la red Modbus TCP, y conectadas al variador a través de Modbus RTU.

Para entender con detalle qué ocurre en el sistema cuando la comunicación Modbus RTU cae, se puede considerar el caso en el que el PLC principal (*PLC_1)* se encuentra regulando, es decir, el PID integrado en este PLC está enviando una actuación al sistema. Mientras tanto, este mismo PLC comparte con el resto datos de interés, así como la lectura del sensor de velocidad.

En el caso en el que la comunicación entre el PLC_1 y el variador se vea interrumpida, el PLC_2 detectará esta caída de conexión y tomará el testigo, continuando la regulación exactamente en el mismo punto donde se quedó el PLC_1.

De la misma manera, el PLC_3 haría el mismo proceso en caso de la caída de la comunicación del PLC_2.

Si el PLC_1 reestablece su comunicación, automáticamente continuará él con la regulación.



61 Esquema explicativo

***Aclaración:** cuando se indica la caída de un PLC, quiere decir que es su comunicación **con el variador** (Modbus RTU) la que ha caído. La red Modbus TCP permanece inalterada durante todo el proceso, por lo que los parámetros compartidos a través de esta red continúan transmitiéndose con normalidad.

En el presente caso, el montaje hardware del sistema es el mismo (exceptuando la propia red Modbus TCP) ya expuesto en el apartado anterior, por lo que ese apartado será omitido. Para la información sobre el montaje hardware, acudir al apartado 4.2.1.

5.2.2 Montaje software

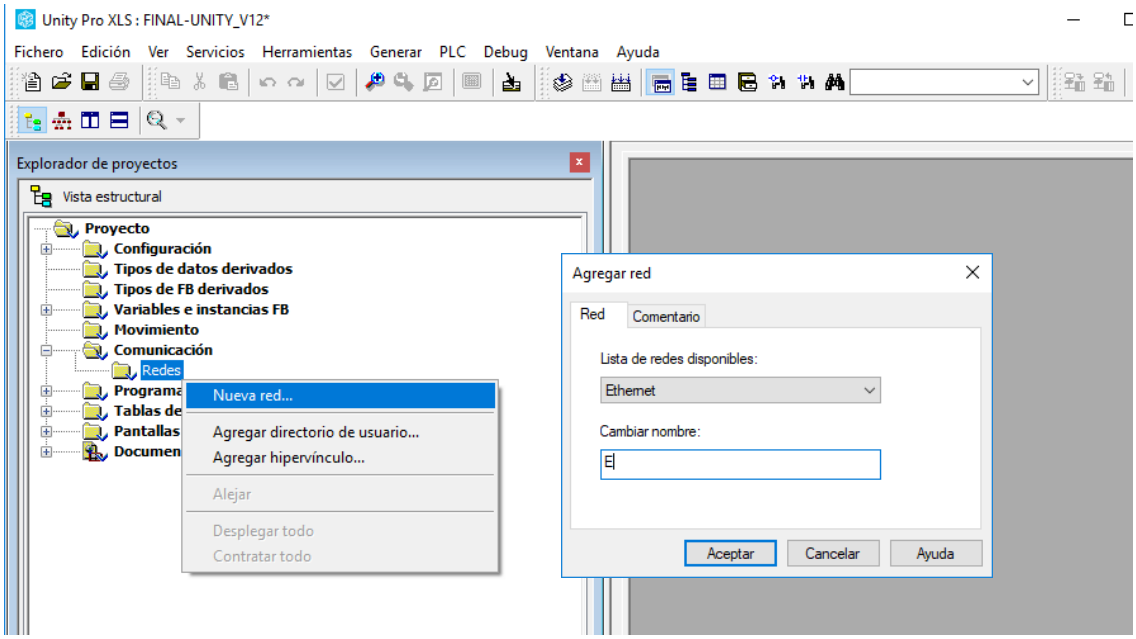
En el caso del montaje software, se harán algunas observaciones de importancia a tener en cuenta a la hora de poner en marcha el software.

Cada una de las estaciones posee su propia dirección IP única para la comunicación TCP. Es por ello que a la hora de iniciar el programa esta dirección debe quedar definida y configurada.

Para configurar una nueva conexión, acudir a la barra lateral, *Proyecto* > *Comunicación* > *Redes*, hacer click derecho y seleccionar *Nueva red…*

En el campo *Lista de redes disponibles* seleccionar *Ethernet* y dar el nombre de *"E"* [2].

[2] **IMPORTANTE.** El uso del nombre "E" para la red no es aleatorio, la conexión se encuentra así descrita en el código, por lo que no debe usarse otro nombre. Si en cualquier caso desea emplearse otra denominación, más adelante en el apartado de programación se detalla qué campos deben renombrarse.



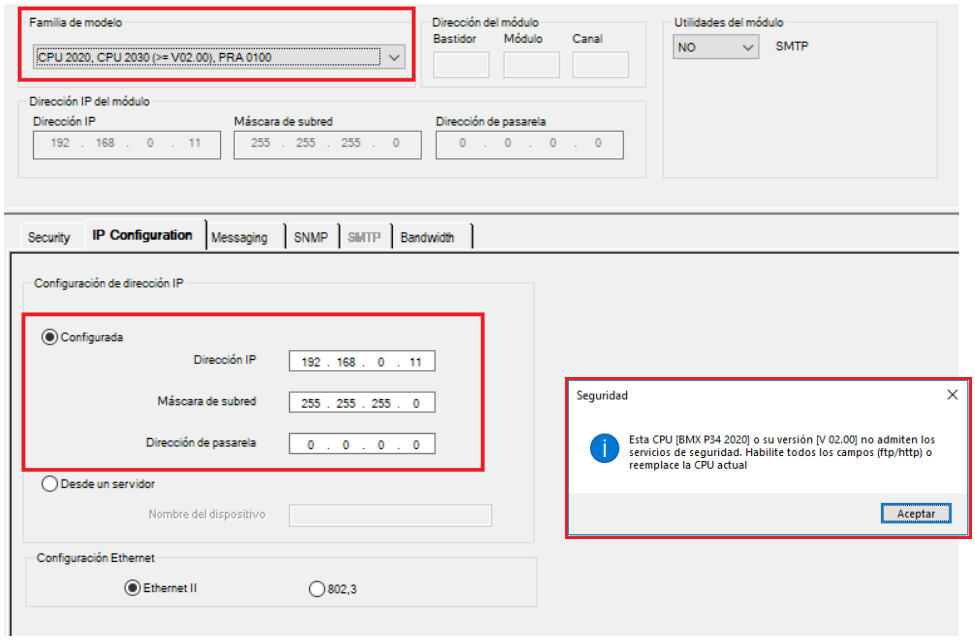
62 Creación de la nueva red

Al hacer click en aceptar, aparecerá la nueva red en esa localización. Hacer click en ella y configurarla asignándole la dirección IP y máscara correspondiente a la estación.

ESTACIÓN	DIRECCIÓN[3]
PLC_1	192.168.0.11
PLC_2	192.168.0.12
PLC_3	192.168.0.13

Tabla 10. Direcciones predefinidas

[3] De nuevo, se recomienda el uso de las direcciones descritas en la tabla para un correcto funcionamiento, ya que se encuentran así incluidas en el código. Si en cualquier caso desean cambiarse, consultar el apartado de programación en el que se detallan los campos a modificar.

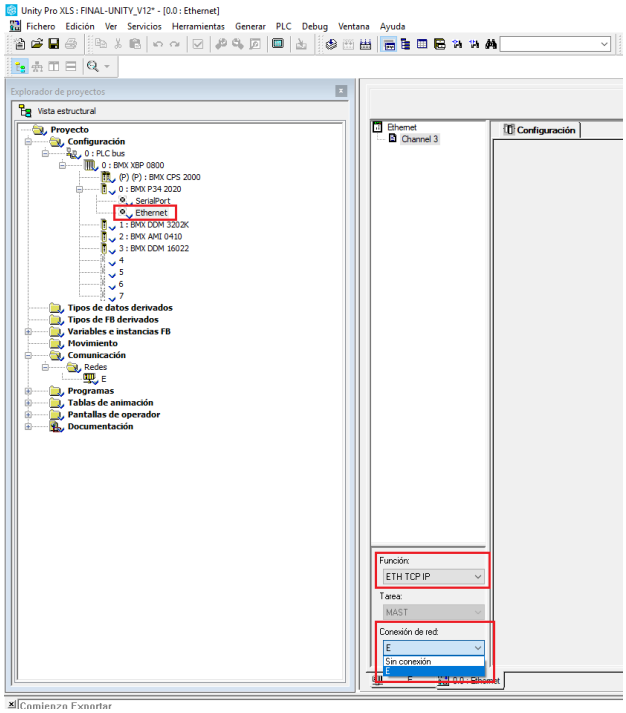


63 Configuración de la nueva red

Es posible que al cerrar la ventana y guardar la configuración aparezca el mensaje que aparece en la figura. En ese caso, acudir a la pestaña "Security" y habilitar las dos opciones. Después, cerrar la ventana y validar los cambios.

A continuación, configurar el puerto Ethernet para que funcione con la red creada. Para ello, acudir de nuevo a la barra lateral *Proyecto > 0 PLC bus > 0 BMX XBP 0800 > 0 BMX P34 2020 > Ethernet.*

En el campo *Función* seleccionar *ETH TCP IP* y en el campo *Conexión de red* seleccionar nuestra red creada *E*. Finalmente cerrar la ventana y aplicar los cambios.

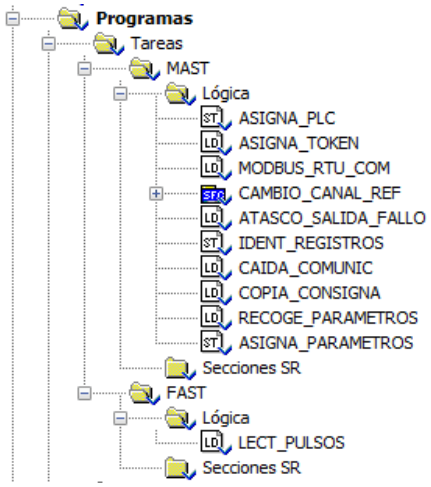


64 Configuración del puerto Ethernet

5.3 Programación

La programación de este sistema consiste en una ligera evolución del programa descrito en el apartado anterior, conservando los aspectos básicos, pero añadiendo todo el código referente a la comunicación y sincronización entre las estaciones.

Se compone de los siguientes bloques de programa



65 Bloques de programa

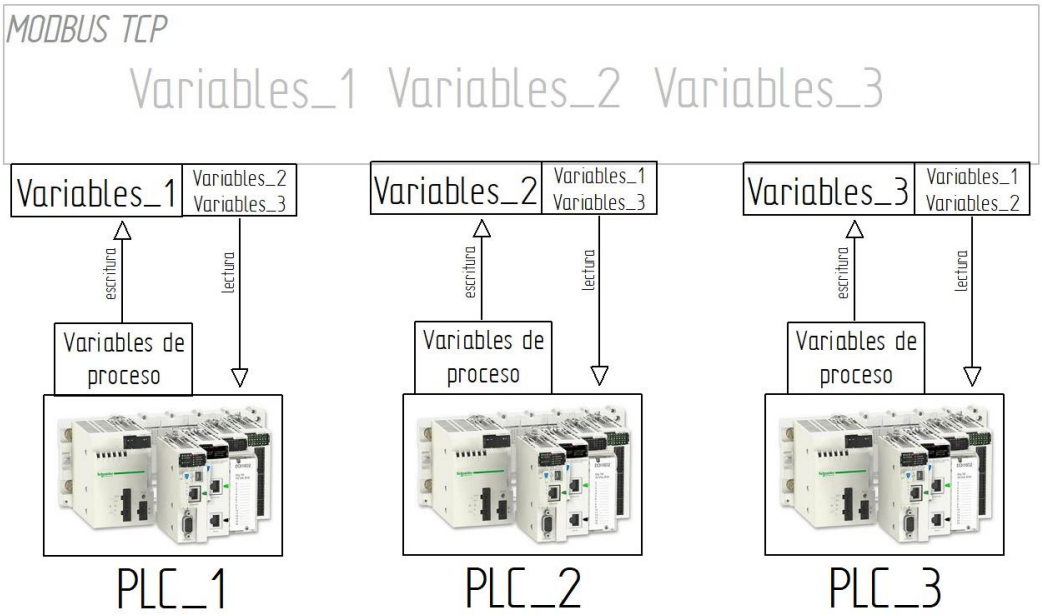
5.3.1 Concepto general

Antes de comenzar con la descripción de los bloques de programa, es necesario conocer la idea genérica por la que se ha optado a la hora de solucionar el problema de la comunicación y la sincronización entre las estaciones.

Empezando por la **comunicación**, las tres estaciones comparten las variables expuestas en la tabla de variables (*apartado 5.3.2*), de forma que cualquier estación dispone de la información de lo que está ocurriendo en todos los PLCs que componen el sistema.

Por tanto, estas variables compartidas van a ser de sólo lectura. De este modo, ejemplificando lo que se explica en la figura, el PLC *X* va a "volcar" sus variables de proceso (*VARIABLES DE PROCESO*) sobre las variables compartidas correspondientes a su estación (*VARIABLES_X)*, y va a leer el resto de variables compartidas *(VARIABLES_Y, VARIABLES_Z)*.

De esta forma, el programa nunca actúa directamente sobre las variables de proceso, aislando el proceso de control de la "nube" de parámetros compartidos.



66 Esquema explicativo de la comunicación Modbus TCP

En cuanto a la **sincronización**, se ha realizado de forma que todos los PLCs realizan el algoritmo de control, pero sólo el autómata que posee el "token" (permiso), hace su control efectivo.

Es decir, los tres poseen el bloque PID y lo ejecutan, pero la actuación de dicho bloque sólo puede ser transmitida al variador por el PLC que posee el token. El criterio de asignación de dicho token está explicado en el apartado 5.2.1 de este capítulo.

Para evitar la desincronización entre el control en las estaciones, se ha empleado una entrada al bloque de PID llamada *RCPY*, que, cuando se habilita, envía a la salida aquello que le sea introducido en dicha entrada. Por tanto, la sincronía se realizará de tal forma que los PLCs que no están controlando tendrán esta entrada habilitada, y enviarán a la salida aquello que envíe el PID que sí está regulando. De esta forma, cuando se produce el cambio, cualquiera de los demás bloques PID están enviando la misma señal de actuación y no tendrán que dar un salto brusco.

5.3.2 Variables compartidas

En este apartado se detallan las variables que se comparten entre las estaciones a través de la red Modbus TCP, es decir, aquellas que aparecen como "*VARIABLES_X"* en el esquema anterior.

Las variables de proceso de cada programa tienen la misma denominación, omitiendo el sufijo con la numeración.

El PLC principal (PLC_1) comparte adicionalmente el valor de la velocidad leída, y el de la parametrización del PID. Esto es debido a que, al tratarse ésta de la estación principal, es la que recibe la lectura del sensor, y la encargada de transmitirla al resto de estaciones. Además, se asume que la parametrización de este PLC va a ser adecuada, por lo que el resto de estaciones tendrán disponibles esta configuración por si en algún momento quieren volcarla y realizar exactamente el mismo control que en el PLC principal.

VARIABLE	DESCRIPCIÓN	TIPO	DIRECCIÓN
FALLO_COMUNICACIONES_1	*Valor alto si la comunicación RTU ha caído, bajo si está activa*	BOOL	**%M10**
DIR_1	*Valor alto o bajo según el sentido de giro*	BOOL	**%M11**
CHANGECHANNEL_1	*Valor alto para modo Modbus, bajo para modo Analógico*	BOOL	**%M12**
TOKEN_1	*Valor alto si se posee el token*	BOOL	**%M13**
SETPOINT_1	*Valor de Setpoint introducido*	INT	**%MW10**
VELOCIDAD_INT_1	*Velocidad leída por el sensor*	INT	**%MW11**
Kp_1	*Constante proporcional PID*	INT	**%MW12**
Ti_1	*Tiempo integral PID*	INT	**%MW13**
Td_1	*Tiempo derivative PID*	INT	**%MW14**
SALIDA_PID_1	*Actuación PID*	INT	**%MW15**
FALLO_COMUNICACIONES_2	*Valor alto si la comunicación RTU ha caído*	BOOL	**%M20**
DIR_2	*Valor alto o bajo según el sentido de giro*	BOOL	**%M21**
CHANGECHANNEL_2	*Valor alto para modo Modbus, bajo para modo Analógico*	BOOL	**%M22**
TOKEN_2	*Valor alto si se posee el token*	BOOL	**%M23**
SETPOINT_2	*Valor de Setpoint introducido*	INT	**%MW20**
SALIDA_PID_2	*Actuación PID*	INT	**%MW21**
FALLO_COMUNICACIONES_3	*Valor alto si la comunicación RTU ha caído*	BOOL	**%M30**
DIR_3	*Valor alto o bajo según el sentido de giro*	BOOL	**%M31**
CHANGECHANNEL_3	*Valor alto para modo Modbus, bajo para modo Analógico*	BOOL	**%M32**
TOKEN_3	*Valor alto si se posee el token*	BOOL	**%M33**
SETPOINT_3	*Valor de Setpoint introducido*	INT	**%MW30**
SALIDA_PID_3	*Actuación PID*	INT	**%MW31**

Tabla 11. Variables compartidas entre las estaciones

5.3.3 Bloques de programa

5.3.3.1 Bloque de asignación (*ASIGNA_PLC*)

El objetivo de este bloque es el volcado de las variables de proceso sobre las variables compartidas. Además, también se definen otras configuraciones como las direcciones del resto de estaciones o la cantidad de registros a enviar y recibir por cada una de las estaciones.

En función de qué estación sea la que realiza el volcado, se pueden apreciar algunas diferencias. Por ejemplo, la velocidad es escrita por el PLC_1 mientras las otras estaciones sólo realizan la lectura de la misma.

```
IF PLC_ASIGNADO=1 THEN

(* Cantidad de objetos de tipo entero a enviar y a recibir *)
        OBJETOSW_ENV:=6;
        OBJETOSW_RECEP_1:=2;
        OBJETOSW_RECEP_2:=2;

(* Direcciones / registros con los que se va a comunicar *)

        DIRECCION_PLC_1:= 'E{192.168.0.012}';
        DIRECCION_PLC_2:= 'E{192.168.0.013}';
        REGISTRO_1:= 20;
        REGISTRO_2:= 30;

(* Cambio de modo en alguna de las demás estaciones *)
        CHANGECHANNEL_ALT_1:=CHANGECHANNEL_2;
        CHANGECHANNEL_ALT_2:=CHANGECHANNEL_3;

(* Volcado de variables de proceso a las variables compartidas *)

        FALLO_COMUNICACIONES_1:= FALLO_COMUNICACIONES;
        DIR_1:= DIR;
        CHANGECHANNEL_1:= CHANGECHANNEL;
        TOKEN_1:= TOKEN;
        SETPOINT_1:= REAL_TO_INT(SETPOINT);
        VELOCIDAD_INT_1:= REAL_TO_INT(VELOCIDAD);
        SALIDA_PID_1:=REAL_TO_INT(SALIDA_PID);
        Kp_1:=REAL_TO_INT(PARAM_PID.Kp*100.0);  (*La Kp se multiplica por 100 antes de transformarse, para no perder resolución*)
        Ti_1:=TIME_TO_INT(PARAM_PID.Ti);
        Td_1:=TIME_TO_INT(PARAM_PID.Td);

END_IF;
IF PLC_ASIGNADO=2 THEN

(* Cantidad de objetos de tipo entero a enviar y a recibir *)

        OBJETOSW_ENV:=2;
        OBJETOSW_RECEP_1:=6;
        OBJETOSW_RECEP_2:=2;

(* Direcciones / registros con los que se va a comunicar *)

        DIRECCION_PLC_1:= 'E{192.168.0.011}';
        DIRECCION_PLC_2:= 'E{192.168.0.013}';
        REGISTRO_1:= 10;
        REGISTRO_2:= 30;


(* Cambio de modo en alguna de las demás estaciones *)

        CHANGECHANNEL_ALT_1:=CHANGECHANNEL_1;
        CHANGECHANNEL_ALT_2:=CHANGECHANNEL_3;

(* Volcado de variables de proceso a las variables compartidas *)

        FALLO_COMUNICACIONES_2:= FALLO_COMUNICACIONES;
        DIR_2:= DIR;
        CHANGECHANNEL_2:= CHANGECHANNEL;
        TOKEN_2:= TOKEN;
        SETPOINT_2:= REAL_TO_INT(SETPOINT);
        VELOCIDAD:=INT_TO_REAL(VELOCIDAD_INT_1);
        SALIDA_PID_2:=REAL_TO_INT(SALIDA_PID);

(*Si pulsamos el botón de copiar parámetros, vuelco la parametrización del PID_1 en el de esta estación*)

        IF BOTON_COPIA_PID=TRUE THEN
                PARAM_PID.Kp:=INT_TO_REAL(Kp_1)/100.0;
                PARAM_PID.Ti:=INT_TO_TIME(Ti_1);
                PARAM_PID.Td:=INT_TO_TIME(Td_1);
        END_IF;

END_IF;
```

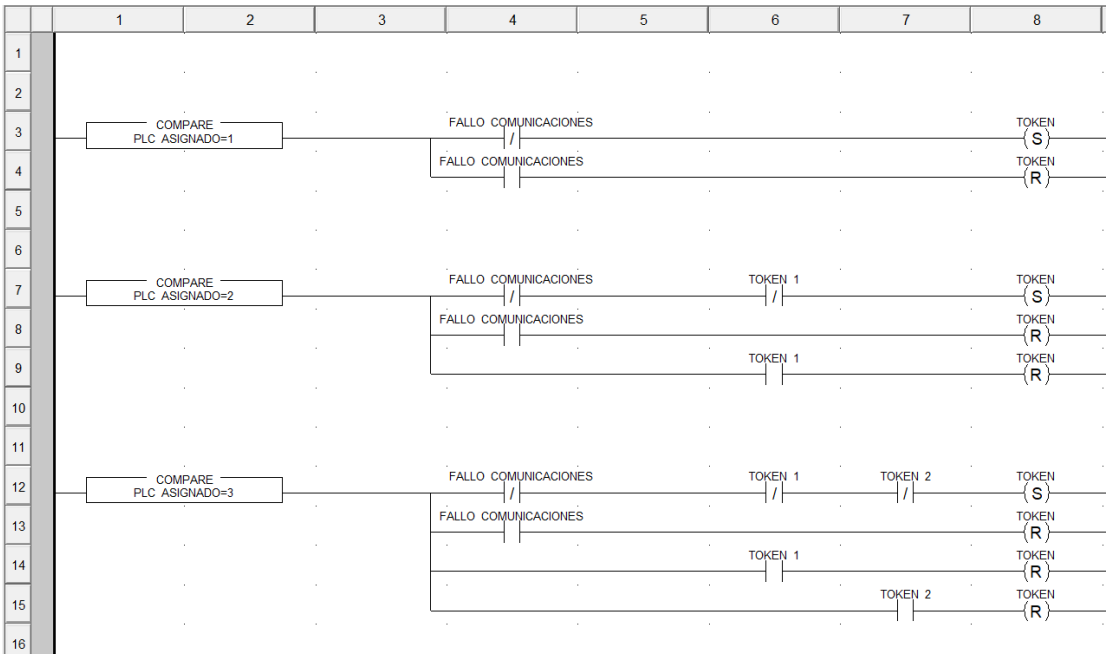
```
IF PLC_ASIGNADO=3 THEN

(* Cantidad de objetos de tipo entero a enviar y a recibir *)

        OBJETOSW_ENV:=2;
        OBJETOSW_RECEP_1:=6;
        OBJETOSW_RECEP_2:=2;

(* Direcciones / registros con los que se va a comunicar *)

        DIRECCION_PLC_1:= 'E{192.168.0.011}';
        DIRECCION_PLC_2:= 'E{192.168.0.012}';
        REGISTRO_1:= 10;
        REGISTRO_2:= 20;

(* Cambio de modo en alguna de las demás estaciones *)

        CHANGECHANNEL_ALT_1:=CHANGECHANNEL_1;
        CHANGECHANNEL_ALT_2:=CHANGECHANNEL_2;

(* Volcado de variables de proceso a las variables compartidas *)

        FALLO_COMUNICACIONES:= FALLO_COMUNICACIONES_3;
        DIR_3:= DIR;
        CHANGECHANNEL_3:= CHANGECHANNEL;
        TOKEN_3:= TOKEN;
        SETPOINT_3:= REAL_TO_INT(SETPOINT);
        VELOCIDAD:=INT_TO_REAL(VELOCIDAD_INT_1);
        SALIDA_PID_3:=REAL_TO_INT(SALIDA_PID);

(*Si pulsamos el botón de copiar parámetros, vuelco la parametrización del PID_1 en el de esta estación*)

        IF BOTON_COPIA_PID=TRUE THEN
                PARAM_PID.Kp:=INT_TO_REAL(Kp_1)/100.0;
                PARAM_PID.Ti:=INT_TO_TIME(Ti_1);
                PARAM_PID.Td:=INT_TO_TIME(Td_1);
        END_IF;

END_IF;
```

67 Programación del bloque *ASIGNA_PLC*

5.3.3.2 Bloque de asignación de token (*ASIGNA_TOKEN*)

El objetivo de este bloque es esencial para el correcto funcionamiento de la sincronización, pues en en este bloque se gestiona cuál de las estaciones tiene el token, y, por tanto, realiza la comunicación.

Como puede observarse en la programación, el PLC más supeditado a los demás es el más bajo en prioridad, mientras que el PLC principal sólo debe observar si tiene comunicación o no, sin tener en cuenta lo que hagan las demás estaciones.

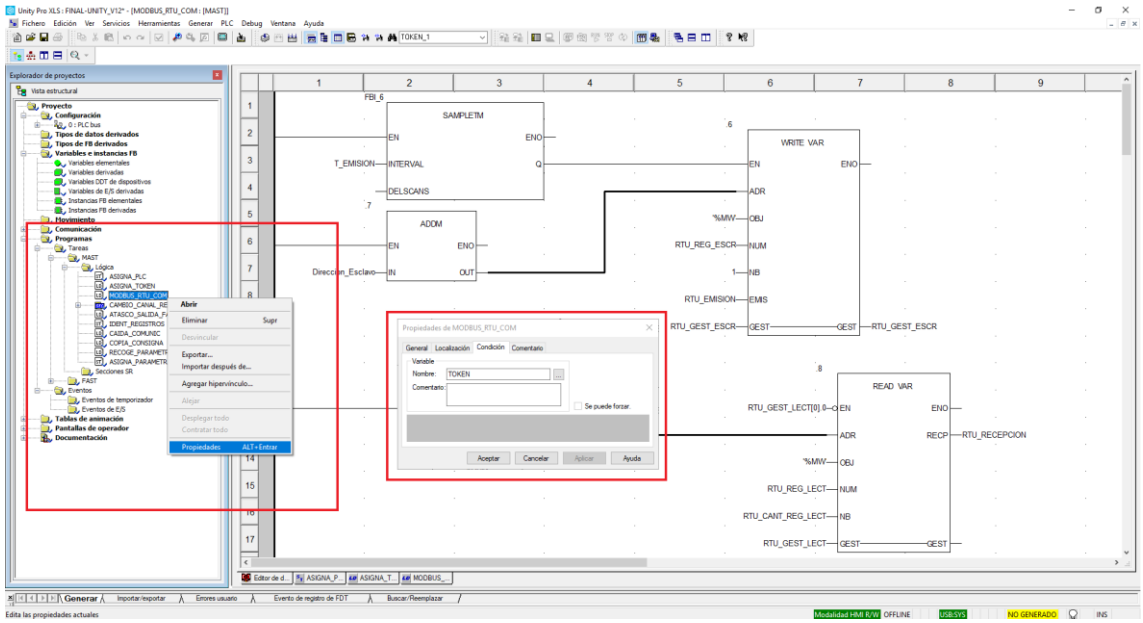


68 Programación del bloque *ASIGNA_TOKEN*

5.3.3.3 Bloque de comunicación Modbus RTU (*MODBUS_RTU_COM*)

Tanto el objetivo como la programación de este bloque son totalmente idénticos al ya expuesto en el apartado 4.3.1 del capítulo anterior.

Sin embargo, en este caso la ejecución de este bloque de programa está supeditada a si se tiene el token o no. Para hacer esta configuración, se ha hecho click derecho sobre el programa > *Propiedades* > *Condición* > *Variable* > *"Token"*.



69 Configuración de bloque de programa sujeto a condición

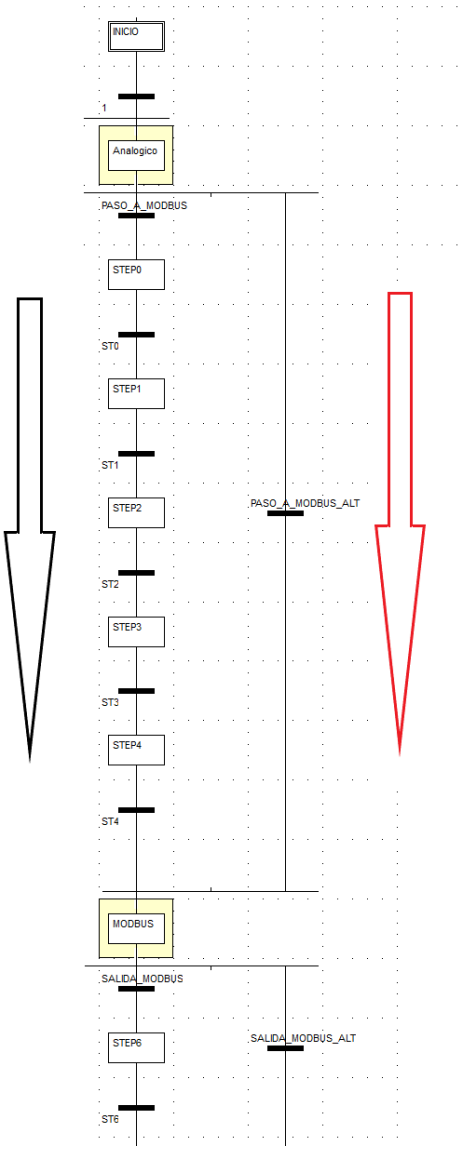
5.3.3.4 Bloque maestro (*CAMBIO_CANAL_REF*)

Aunque las similaridades entre este bloque y su homónimo del capítulo anterior son bastantes, conviene señalar algunas diferencias entre ambos. El funcionamiento básico (sin estaciones redundantes) ya puede observarse en el apartado 4.3.2 del capítulo anterior, por lo que el contenido ya mencionado será excluido de este apartado.

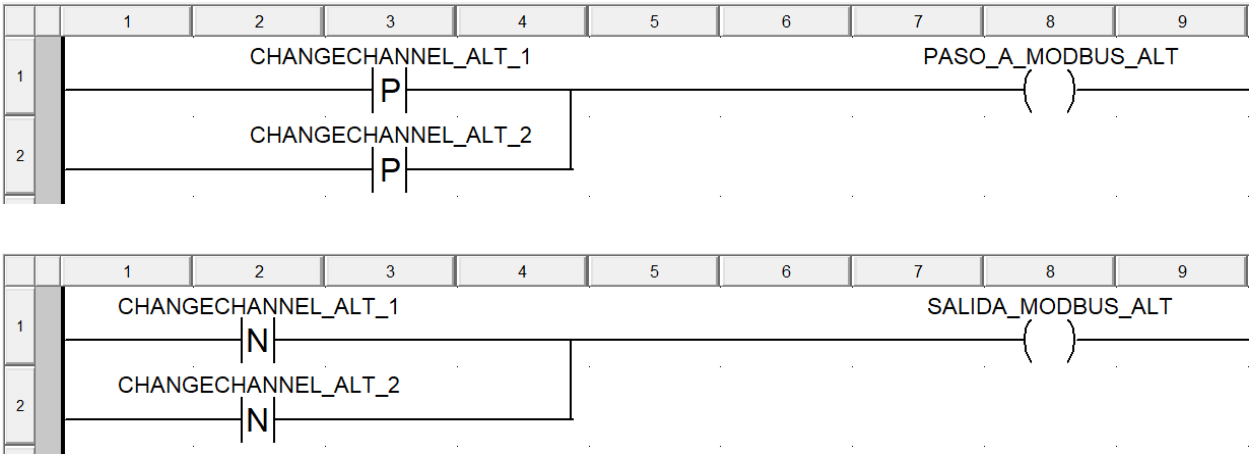
Las diferencias residen principalmente en todo lo referente a la sincronización entre estaciones.

En este bloque se producen los cambios entre modos de operación del sistema. Por ello, cuando una de las estaciones realiza un cambio entre modos de operación (cambio en la variable *CHANGECHANNEL*), el resto de estaciones deben realizar el cambio, aunque sólo la estación que tiene el token debe realizar la secuencia de activación del modo Modbus comentada en el apartado 4.3.2.3.

Con este fin, se han creado las variables *CHANGECHANNEL_ALT_X*, que realizan el cambio directo al modo Modbus en el resto de estaciones, mientras que la estación que sí modifica *CHANGECHANNEL* realiza la secuencia de activación. Ídem para la salida del modo Modbus.



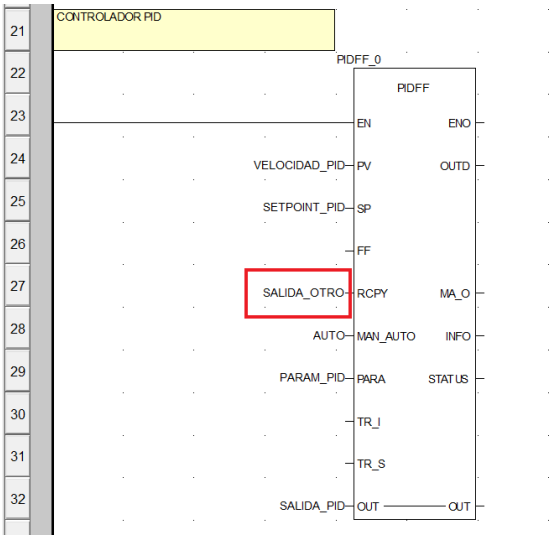
70 Alternativas para la activación/desactivación del modo Modbus



71 Transiciones *PASO_A_MODBUS_ALT* y *SALIDA_MODBUS_ALT*

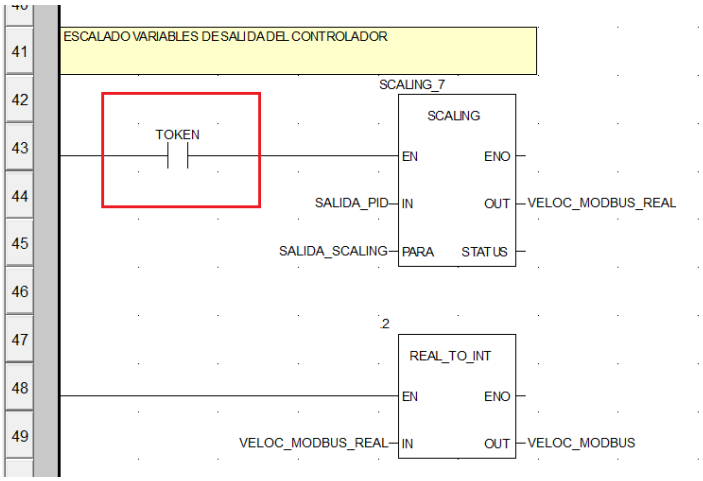
Respecto a la acción *CONTROLPID,* se hace uso de la entrada *RCPY* del bloque *PIDFF*. Esta entrada aplica directamente el valor que sea introducido a través de ella como actuación a la salida, siempre que el campo *PID_PARAM.en_rcpy* esté habilitado.

De esta forma, el PLC que está regulando tendrá el campo *PID_PARAM.en_rcpy* deshabilitado, pues realizaría el control normal tal y como se abordó en el anterior capítulo, mientras que el resto de estaciones tendrán dicho campo habilitado, y leerán por la entrada *RCPY* la actuación del PID que regula *(SALIDA_OTRO)*, estando así la actuación de los tres controladores en sincronía.



72 Entrada *RCPY*

Otra diferencia importante con el caso expuesto en el apartado anterior es que la actuación del PID sólo se transmite al variador en caso de que se tenga el *TOKEN*.



73 Condición de volcado de la actuación del PID

5.3.3.5 Bloque de identificación de registros (*IDENT_REGISTROS*)

Para conocer su programación, acudir al apartado 4.3.3.

5.3.3.6 Bloque de administración de la comunicación (*CAIDA_COMUNIC*)

El objetivo de este bloque, a pesar de ser parecido a su homónimo del capítulo anterior, guarda algunas diferencias importantes respecto a éste, referentes a la sincronización.

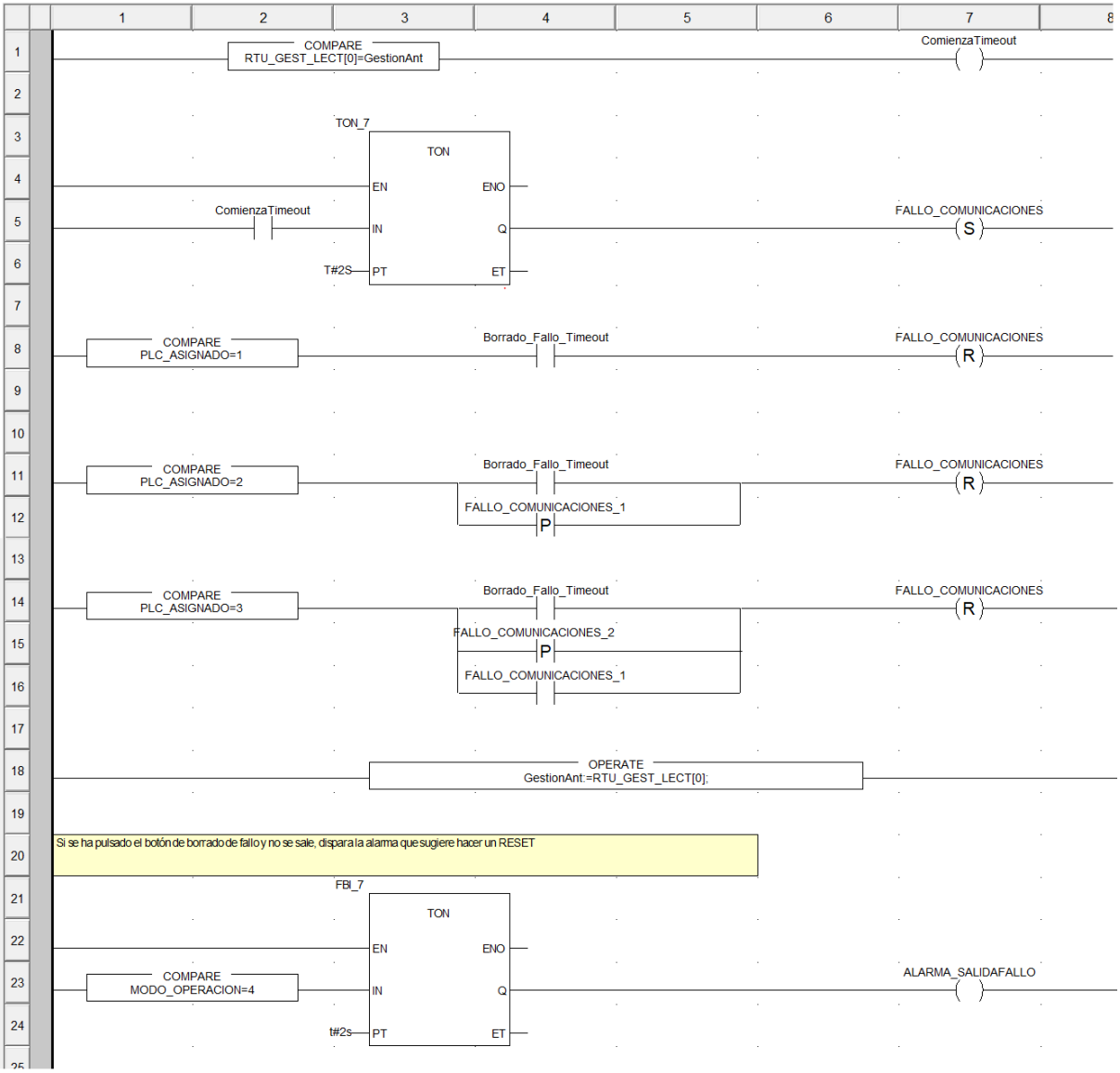
El proceso de detección de caída de la comunicación es exactamente el mismo. Se observará el bit de estado de la matriz de gestión de lectura, y si éste no cambia durante más de 2 segundos, se considera que la comunicación ha caído.

Ahora bien, conviene explicar cómo se administra la comunicación RTU en cada una de las estaciones, y cómo se produce el cambio de una a otra cuando las estaciones caen.

En primer lugar, al arrancar los programas, todas las estaciones tendrán la alarma *FALLO_COMUNICACIONES* activada por seguridad. En cualquiera de ellos se puede pulsar el botón de borrado de fallo, y automáticamente establecerá la conexión con el variador.

Si en algún momento la comunicación de un PLC cae, el PLC de prioridad inmediata inferior establecerá la comunicación automáticamente (*líneas 11 y 12 del código para el PLC_2, líneas 14, 15 y 16 de código para el PLC_3*).

Sin embargo, en este caso volvemos a tener en cuenta la prioridad, y es por eso que si en el PLC_1 se pulsa el botón de borrado de fallo y tiene comunicación RTU disponible, la estación que estuviese transmitiendo en ese momento suspenderá su comunicación.



74 Programación del bloque *CAIDA_COMUNIC*

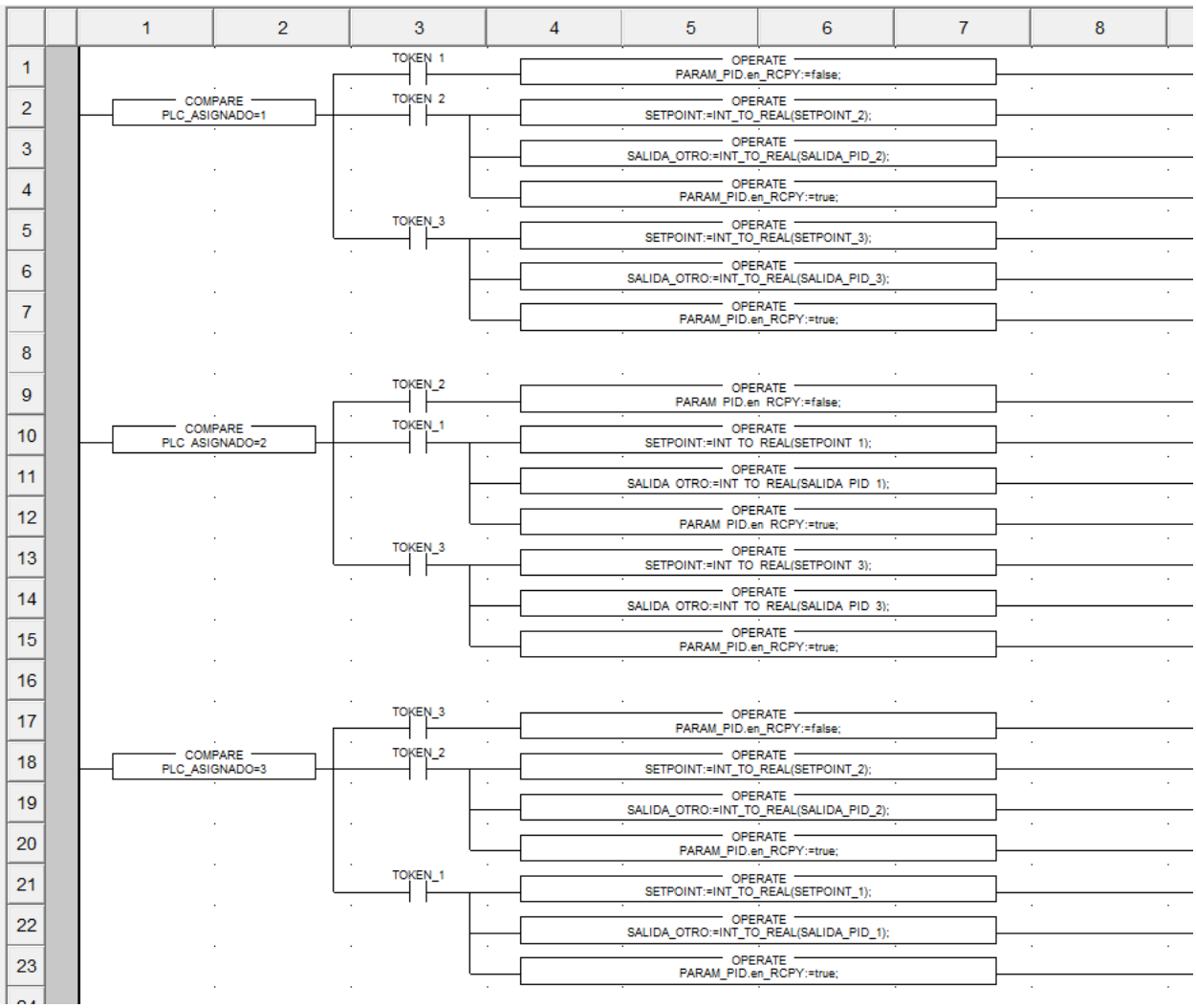
5.3.3.7 Bloque de sincronización de los controladores (*COPIA_CONSIGNA*)

El objetivo de este bloque es realizar la sincronía en la actuación de los reguladores comentada en el apartado 5.3.3.4, a través del uso de la entrada *RCPY* del bloque *PIDFF*.

Para ello, en función del PLC asignado y de quién tiene el token, se asignan unos parámetros u otros.

El caso del PLC_1 puede extrapolarse a las otras dos estaciones.

- Si el PLC_1 tiene el token, la única acción a realizar es desactivar la entrada *RCPY*.
- Si el token está en una de las dos estaciones, es necesario realizar tres acciones
    - Habilitar la entrada *RCPY (línea 5 de código).*
    - Copiar el valor de la actuación del PID que regula en la variable *SALIDA_OTRO*, para sincronizar las actuaciones *(línea 4 de código).*
    - Copiar el valor del Setpoint del PID que regula, para que en caso de que se produzca el cambio no haya un setpoint distinto fijado en esta estación *(línea 3 de código).*



75 Programación del bloque *COPIA_CONSIGNA*

5.3.3.8 Bloque de lectura de variables compartidas (*RECOGE_PARAMETROS)*

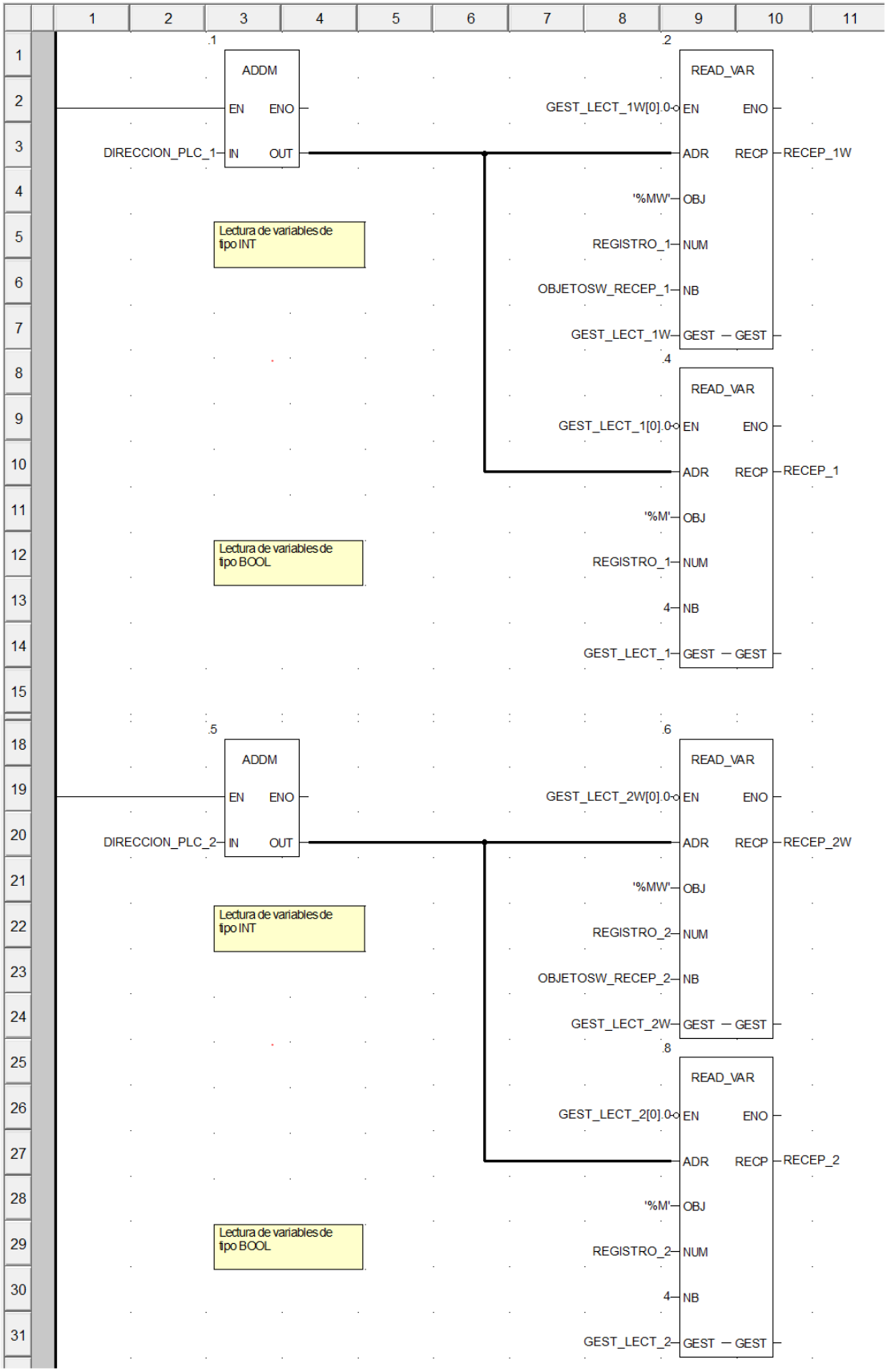
El objetivo de este bloque es la lectura de las variables compartidas a través de Modbus TCP.

Para ello, al igual que en la comunicación RTU, se emplean los bloques *READ_VAR*. Se leerán los parámetros que provienen de las direcciones *DIRECCION_PLC_1* y *DIRECCION_PLC_2*, que en cada estación cambiará según lo establecido en el bloque *ASIGNA_PLC*, al igual que la cantidad de objetos leídos y la dirección del registro a leer.

En el resto de aspectos, se sigue la misma dinámica que en la comunicación RTU, con las matrices de gestión y la habilitación del bloque *READ_VAR*, dada por el bit de estado de la matriz de gestión.

La nomenclatura para las variables de lectura es la siguiente

- Número 1 ó 2 según de qué estación se lee. Si estamos en PLC_1, el número 1 corresponderá con la estación 2, y el número 2 con la estación 3; tal y como se define en la acción *ASIGNA_PLC*.
- Terminación acabada en W para las matrices de recepción de enteros.



76 Programación del bloque *RECOGE_PARAMETROS*

5.3.3.9 Bloque de asignación de variables compartidas (*ASIGNA_PARAMETROS)*

Las variables compartidas leídas en el bloque anterior son almacenadas en los vectores de recepción *RECEP_1W, RECEP_1, RECEP_2W* y *RECEP_2*.

El objetivo del presente bloque es, simplemente, la copia de estos parámetros a las variables compartidas de nombres conocidos (VARIABLES_X).

```
IF PLC_ASIGNADO=1 THEN

        SETPOINT_2:=RECEP_1W[1];
        SALIDA_PID_2:=RECEP_1W[2];
        FALLO_COMUNICACIONES_2:=RECEP_1[1].0;
        DIR_2:=RECEP_1[1].1;
        CHANGECHANNEL_2:=RECEP_1[1].2;
        TOKEN_2:=RECEP_1[1].3;

        SETPOINT_3:=RECEP_2W[1];
        SALIDA_PID_3:=RECEP_1W[3];
        FALLO_COMUNICACIONES_3:=RECEP_2[1].0;
        DIR_3:=RECEP_2[1].1;
        CHANGECHANNEL_3:=RECEP_2[1].2;
        TOKEN_3:=RECEP_2[1].3;

END_IF;

IF PLC_ASIGNADO=2 THEN

        SETPOINT_1:=RECEP_1W[1];
        VELOCIDAD_INT_1:=RECEP_1W[2];
        Kp_1:=RECEP_1W[3];
        Ti_1:=RECEP_1W[4];
        Td_1:=RECEP_1W[5];
        SALIDA_PID_1:=RECEP_1W[6];
        FALLO_COMUNICACIONES_1:=RECEP_1[1].0;
        DIR_1:=RECEP_1[1].1;
        CHANGECHANNEL_1:=RECEP_1[1].2;
        TOKEN_1:=RECEP_1[1].3;

        SETPOINT_3:=RECEP_2W[1];
        SALIDA_PID_3:=RECEP_1W[2];
        FALLO_COMUNICACIONES_3:=RECEP_2[1].0;
        DIR_3:=RECEP_2[1].1;
        CHANGECHANNEL_3:=RECEP_2[1].2;
        TOKEN_3:=RECEP_2[1].3;

END_IF;

IF PLC_ASIGNADO=3 THEN

        SETPOINT_1:=RECEP_1W[1];
        VELOCIDAD_INT_1:=RECEP_1W[2];
        Kp_1:=RECEP_1W[3];
        Ti_1:=RECEP_1W[4];
        Td_1:=RECEP_1W[5];
        SALIDA_PID_1:=RECEP_1W[6];
        FALLO_COMUNICACIONES_1:=RECEP_1[1].0;
        DIR_1:=RECEP_1[1].1;
        CHANGECHANNEL_1:=RECEP_1[1].2;
        TOKEN_1:=RECEP_1[1].3;

        SETPOINT_2:=RECEP_2W[1];
        SALIDA_PID_2:=RECEP_1W[2];
        FALLO_COMUNICACIONES_2:=RECEP_2[1].0;
        DIR_2:=RECEP_2[1].1;
        CHANGECHANNEL_2:=RECEP_2[1].2;
        TOKEN_2:=RECEP_2[1].3;

END_IF;
```

77 Programación del bloque *ASIGNA_PARAMETROS*

5.3.3.10 Bloque de lectura de pulsos (*LECT_PULSOS*)

Para acceder a la información sobre la programación de este bloque, acudir al apartado 4.3.5.

5.4 Pantalla de explotación

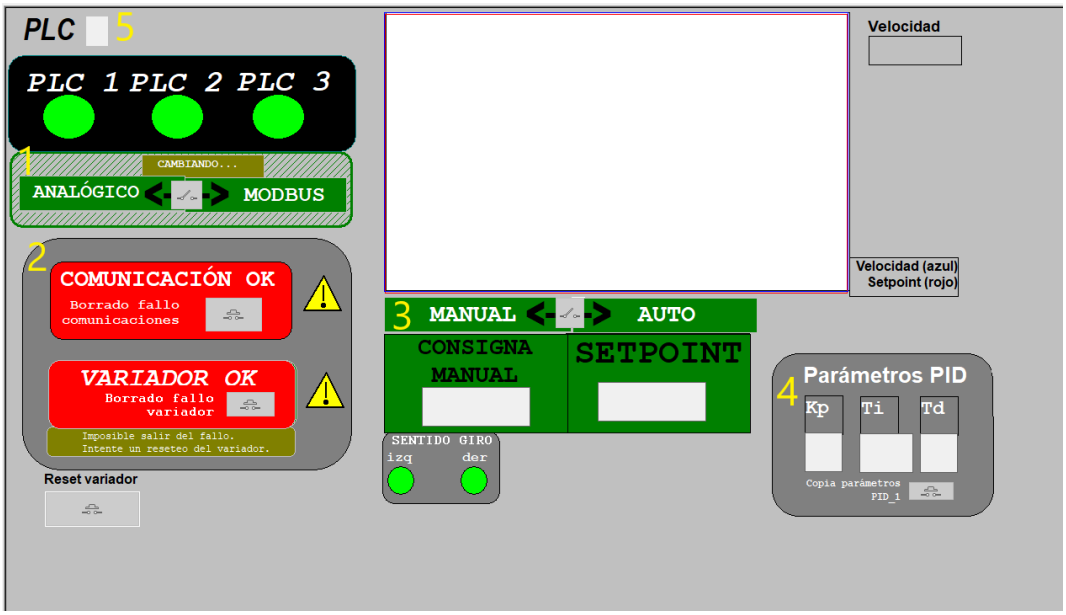
En este caso, y al igual que en el caso anterior, se usa la estación PC a modo de HMI, siendo equivalente en términos de funcionalidad.

Al guardar bastante similitud con el programa expuesto en el capítulo anterior, la pantalla de explotación en este capítulo será una versión ampliada de la anterior. Por lo tanto, la explicación del funcionamiento básico del panel de mandos será omitido de esta sección, pudiendo verse en el apartado 4.4 de este documento.

- (1) Selector de modo Analógico <-> Modbus. El pulsador sólo es visible en la estación cuyo PLC tenga el *TOKEN*. Cuando este PC alterna entre modos, se colorea de verde la opción correspondiente en todas las estaciones.
- (2) Panel de alarmas. Al hacer un arranque del sistema, se accionará el pulsador de borrado fallo comunicaciones para el PLC con el que queramos comenzar a controlar. En el resto de estaciones esta opción aparecerá en fallo.
- (3) Selector de modo Manual <-> Automático. El pulsador con retención sólo aparece en caso de que nos encontremos en modo Modbus, en cualquiera de las estaciones.

IMPORTANTE. Para evitar cambios bruscos en la referencia de velocidad, debe procurarse que el selector está en modo AUTO en todas las estaciones, de forma que si cae una comunicación todas puedan regular.

- (4) Parámetros PID. Por defecto posee la parametrización Kp=0.3, Ti=2s, Td=0s; pero permite modificar dichos parámetros durante la ejecución del programa. El resto de estaciones puede tener su propia parametrización, más agresiva o menos que la implementada por defecto en el PLC principal. En cualquier momento puede copiarse la parametrización de este PLC en cualquiera de las estaciones accionando el botón inferior.
- (5) Número de estación. En este cuadro de texto seleccionamos manualmente qué número de estación es el asignado a este PLC.



78 Pantalla de explotación diseñada

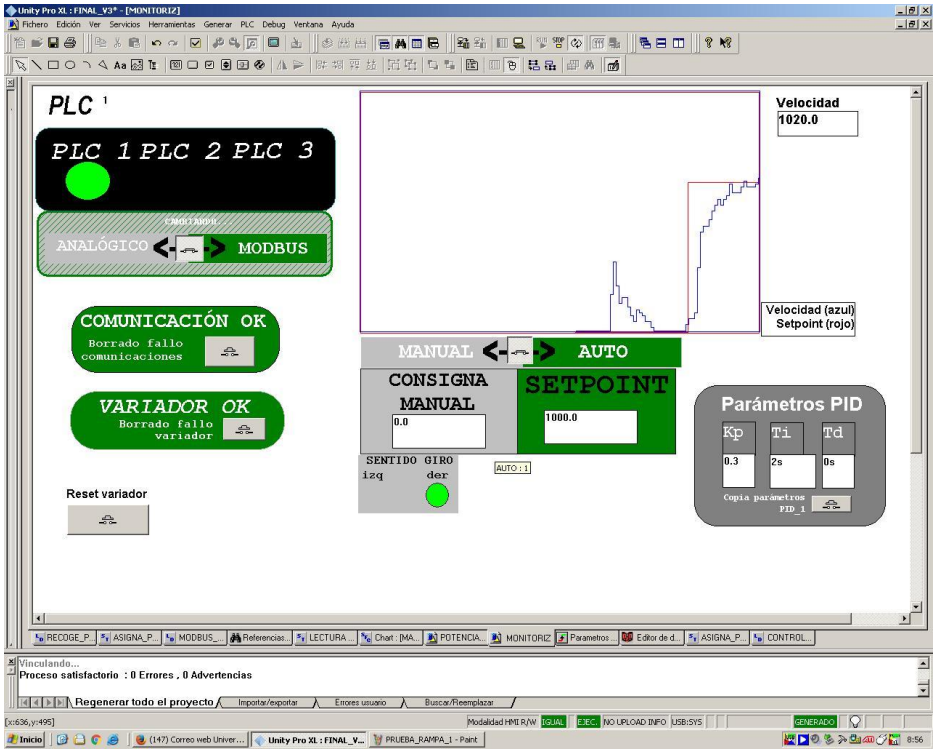
5.5 Funcionamiento del sistema

Finalmente, se presenta una prueba del sistema en la que se puede observar el funcionamiento de todo lo explicado en los capítulos 4 y 5.

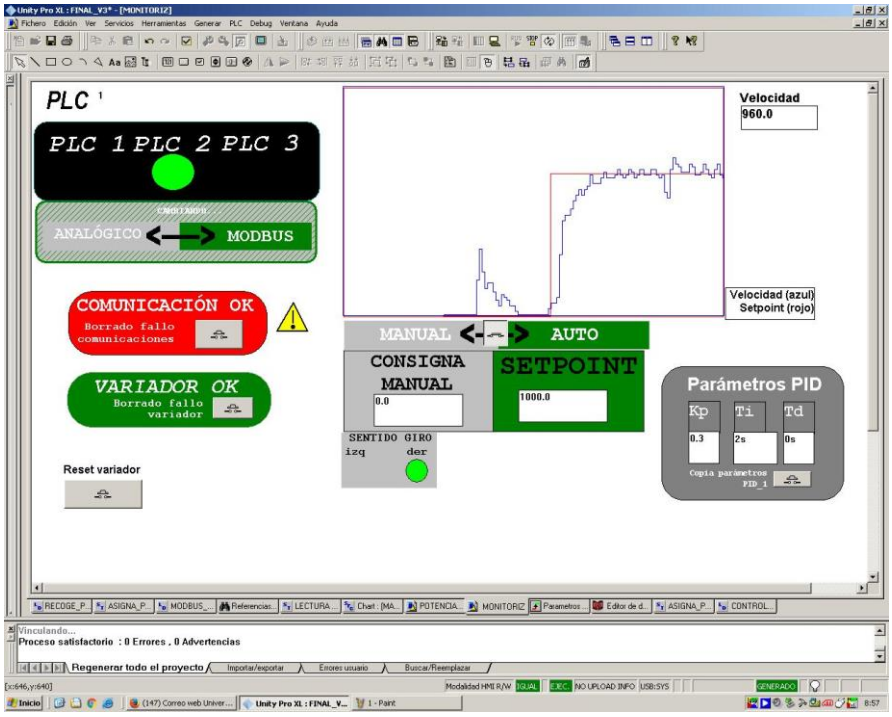
Antes de mostrar el proceso experimental, se deben definir ciertas condiciones para poder entender mejor la visualización

- El experimento se ha realizado únicamente haciendo uso de dos estaciones, PLC_1 y PLC_2.
- Con motivo de una mejor interpretación, todas las capturas son tomadas en el PC de la estación 1, observándose cómo desaparece la botonera de cambio de modo al controlar en la estación 2, y aparece el fallo de comunicaciones.
- El método utilizado para pasar el token de una estación a otra es el que se ha considerado más realista para hacer el cambio de 1 a 2 "tiramos" la comunicación desconectando el cable, mientras que para hacer el cambio de 2 a 1 reconectamos el cable y borramos el fallo.
- La parametrización PID empleada ha sido la misma en las dos estaciones, exceptuando el último caso, en el que se ha querido probar un control más agresivo.
- Se ha omitido en este documento la demostración experimental del modo manual, por su simplicidad.

En primer lugar, y para comprobar la utilidad de la entrada *RCPY*, se emplea una variante del programa que no usa esta entrada. Primero, se controla el sistema a través del PLC_1. Se pone en marcha el sistema, se pasa a modo Modbus y se fija una referencia.

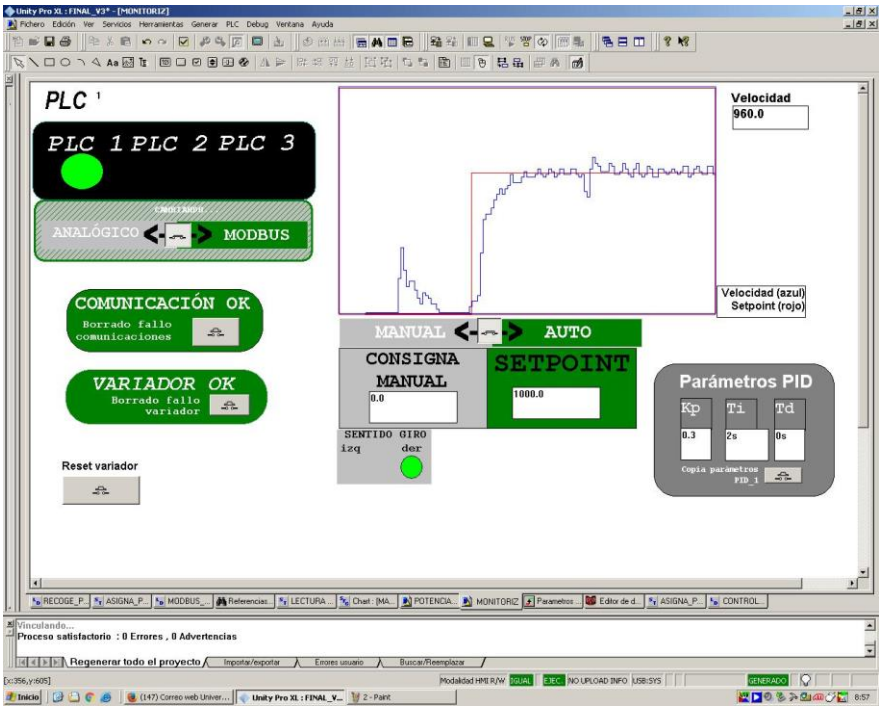


Una vez establecido el régimen permanente, se desconecta la comunicación Modbus RTU del PLC_1 y el sistema conmuta automáticamente al control a través del PLC_2.

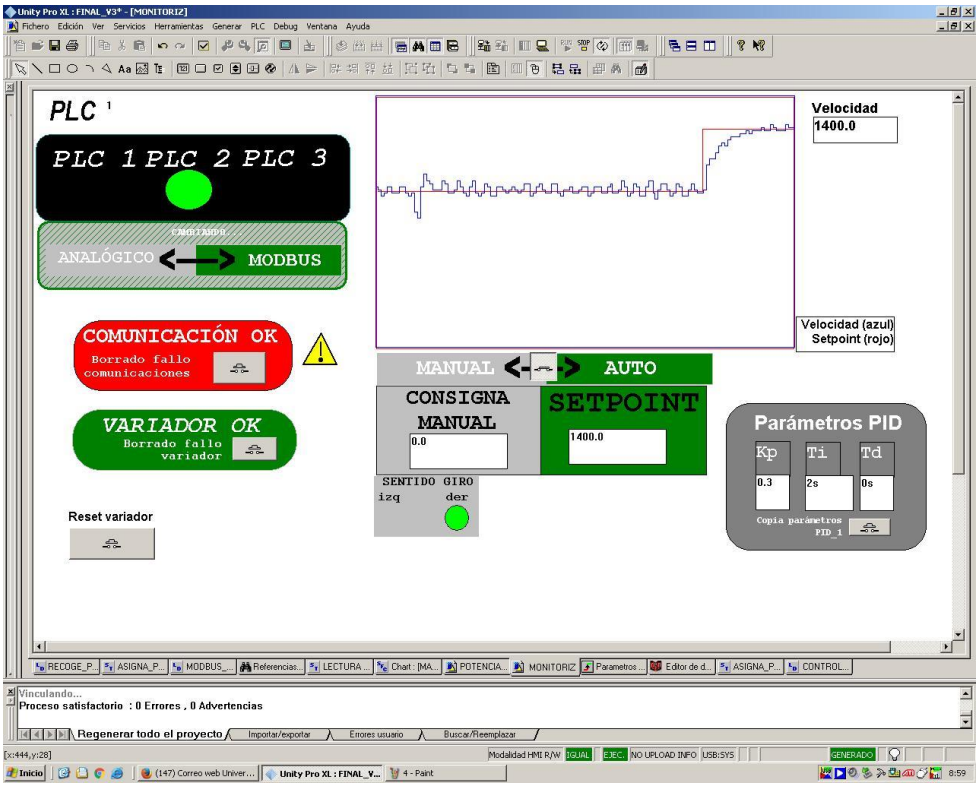


Como se puede apreciar, se ha producido una discontinuidad, debido a que el controlador del PLC_2 tenía una actuación menor a la del PLC_1 en el momento de hacer el cambio, y ha tenido que adaptarse rápidamente cuando ha empezado a controlar.

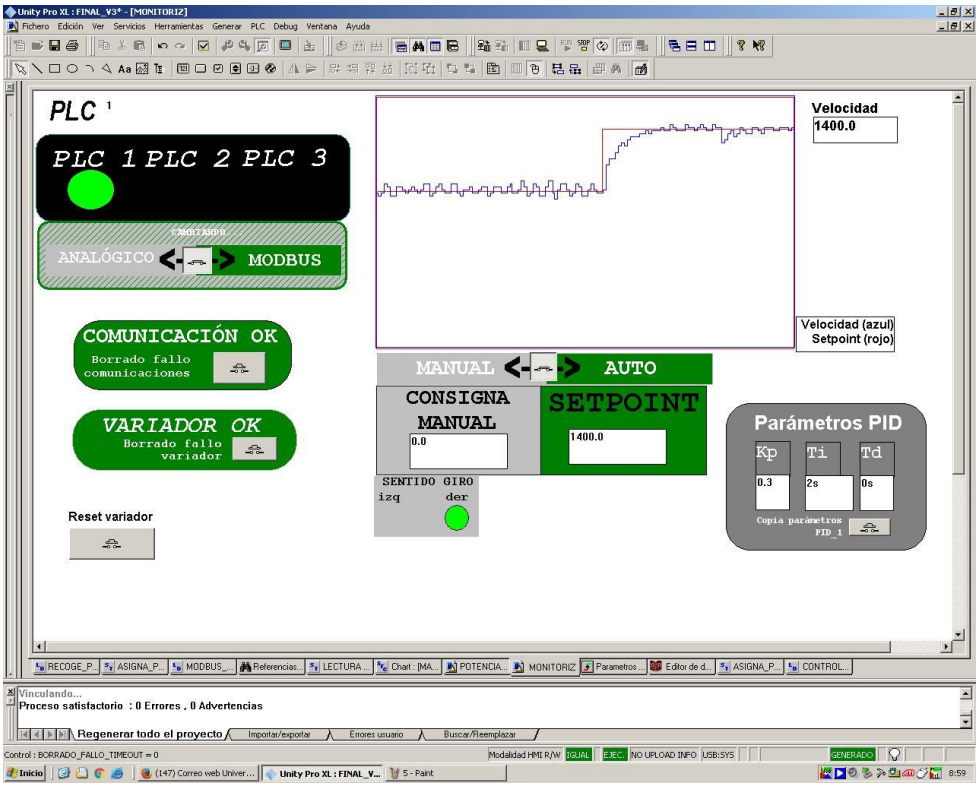
Puesto que el PLC_1 ha dejado el sistema con un Setpoint de 1000rpm de forma estable, cabe esperar que si volvemos a controlar a través de él, no se va a producir la discontinuidad.



Efectivamente, la discontinuidad no aparece. A continuación vamos a conmutar al PLC_2 y a cambiar el Setpoint desde 1000 hasta 1400rpm.

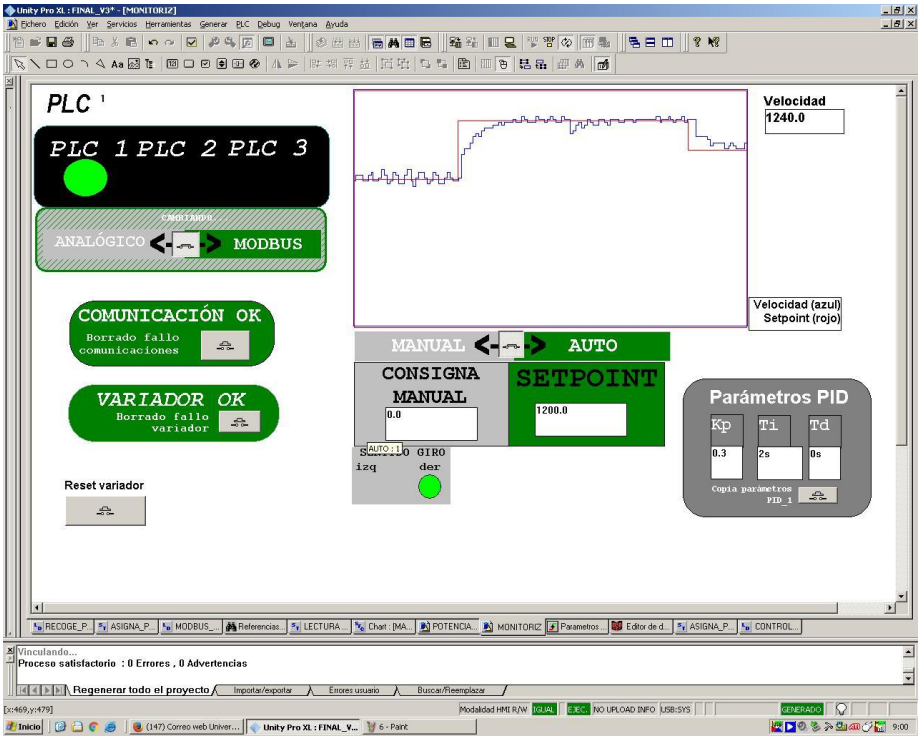


Después, volvemos a conmutar al PLC_1 y se asume que sí debería producirse la discontinuidad, al haber cambiado la referencia.

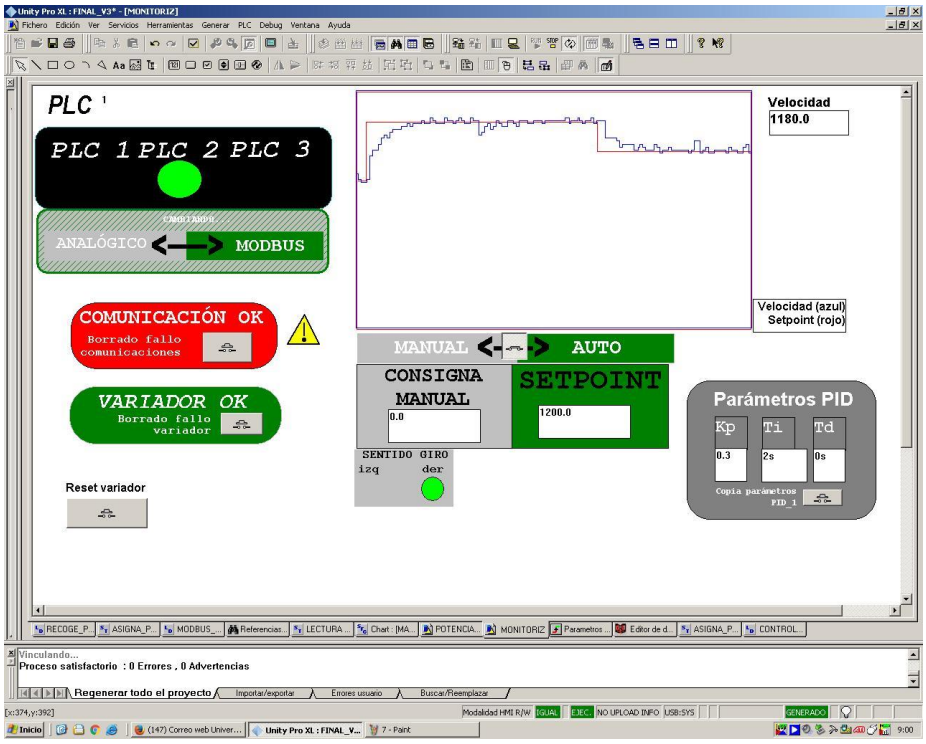


Efectivamente, se observa una discontinuidad en este caso de menor magnitud, ya que el salto en la referencia previo era menor.

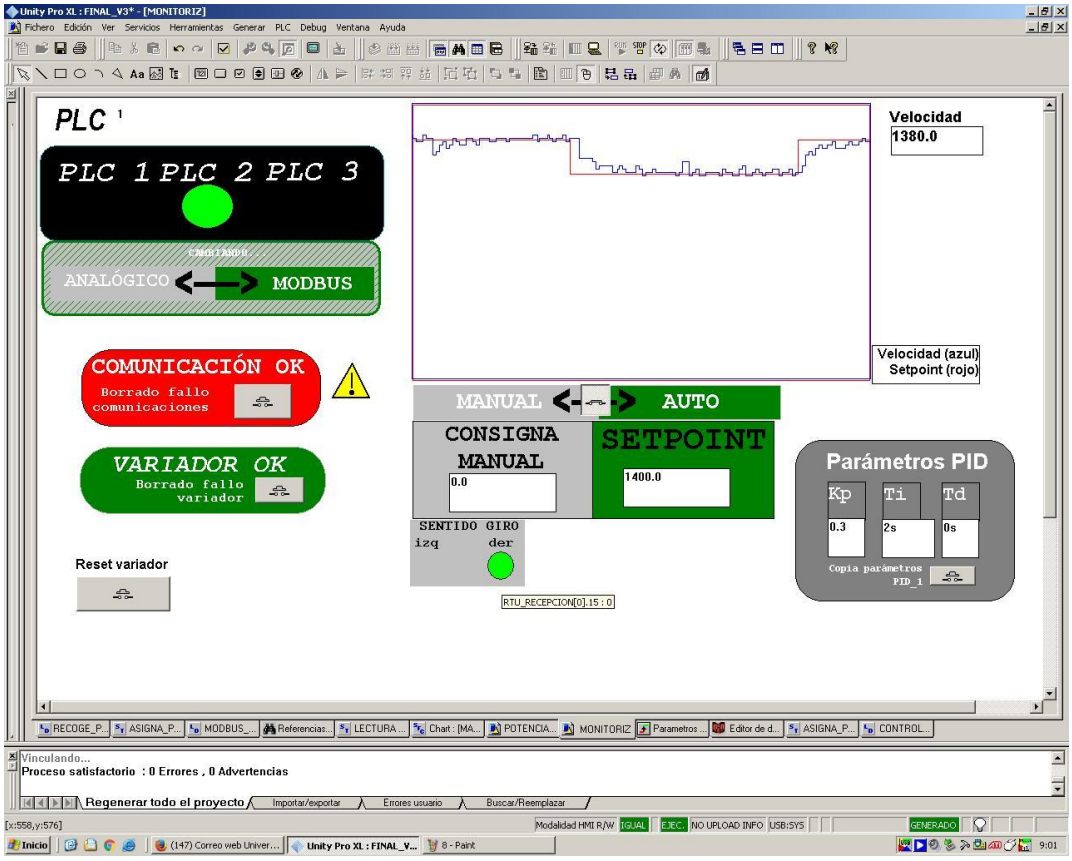
A continuación, se sigue controlando a través del PLC_1, y vamos a cambiar el Setpoint a 1200. Además, se activa la entrada *RCPY*, por lo que el PID del PLC_2 va a empezar a replicar la salida del PLC_1.

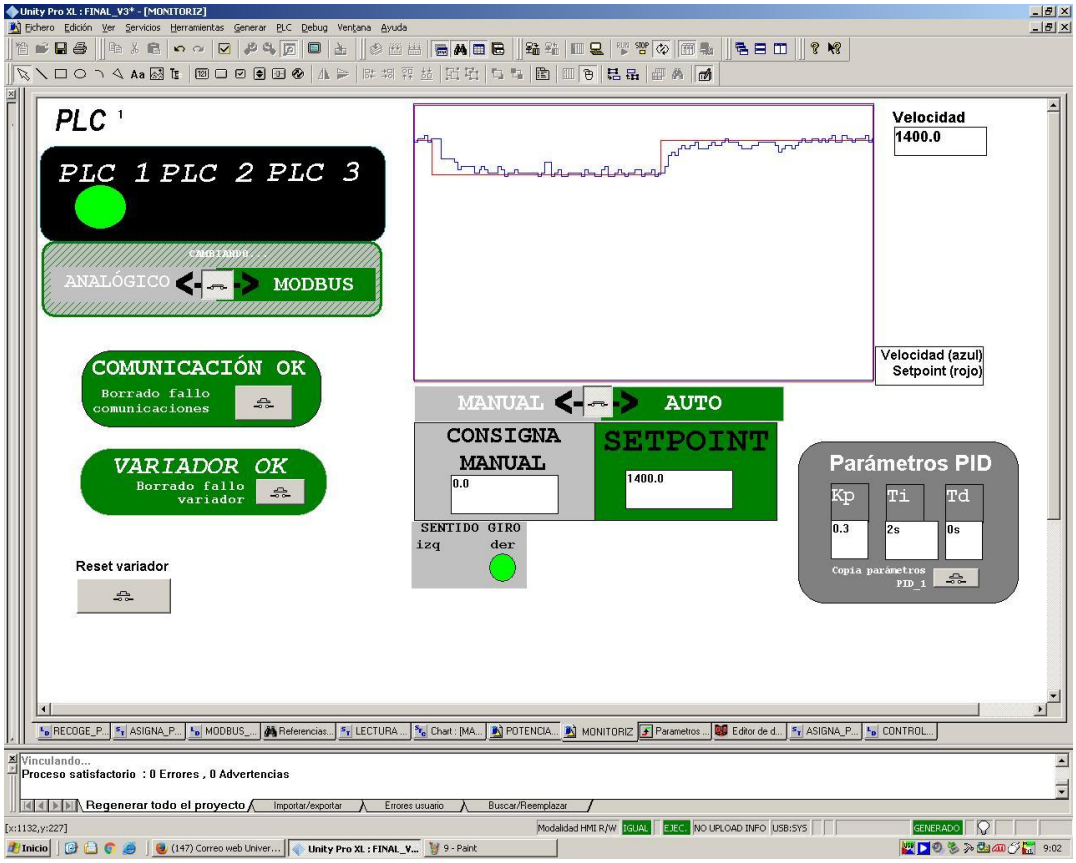


Ahora se conmuta al PLC_2. Se esperaría una ligera discontinuidad; sin embargo, observamos cómo se ha reducido considerablemente gracias al uso de la entrada *RCPY*.

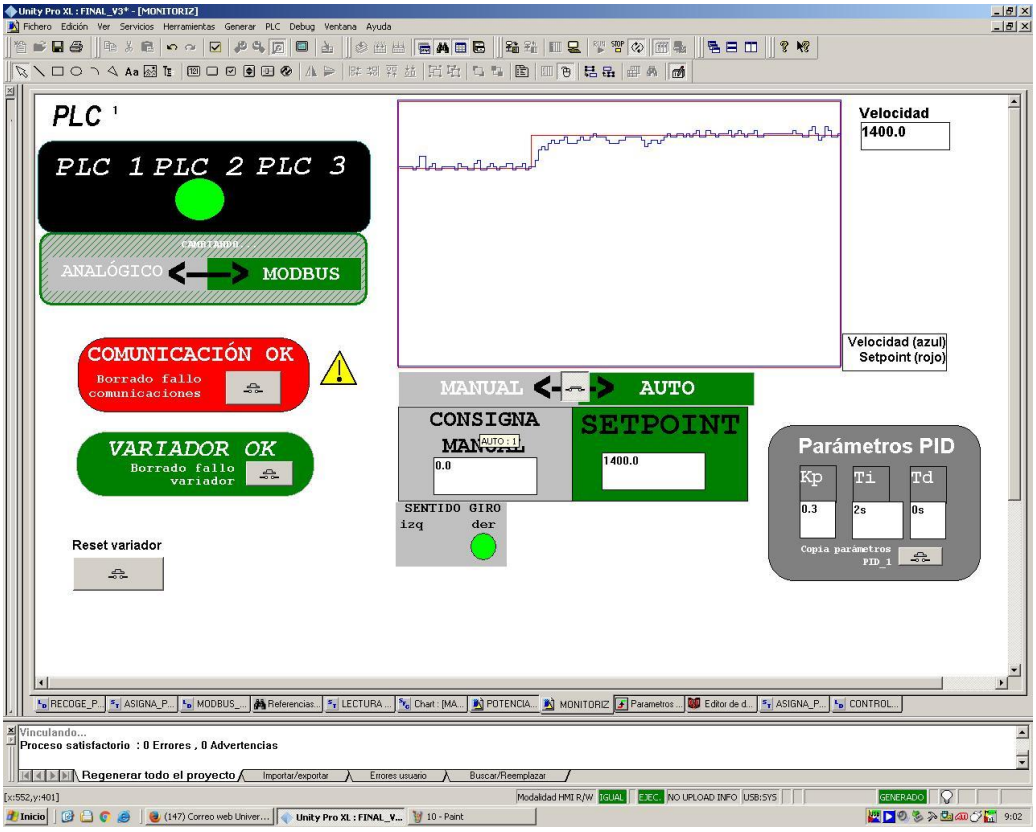


Se cambia la referencia a 1400rpm en el PLC_2 y se conmuta al PLC_1, observando de nuevo la ausencia de esa discontinuidad.

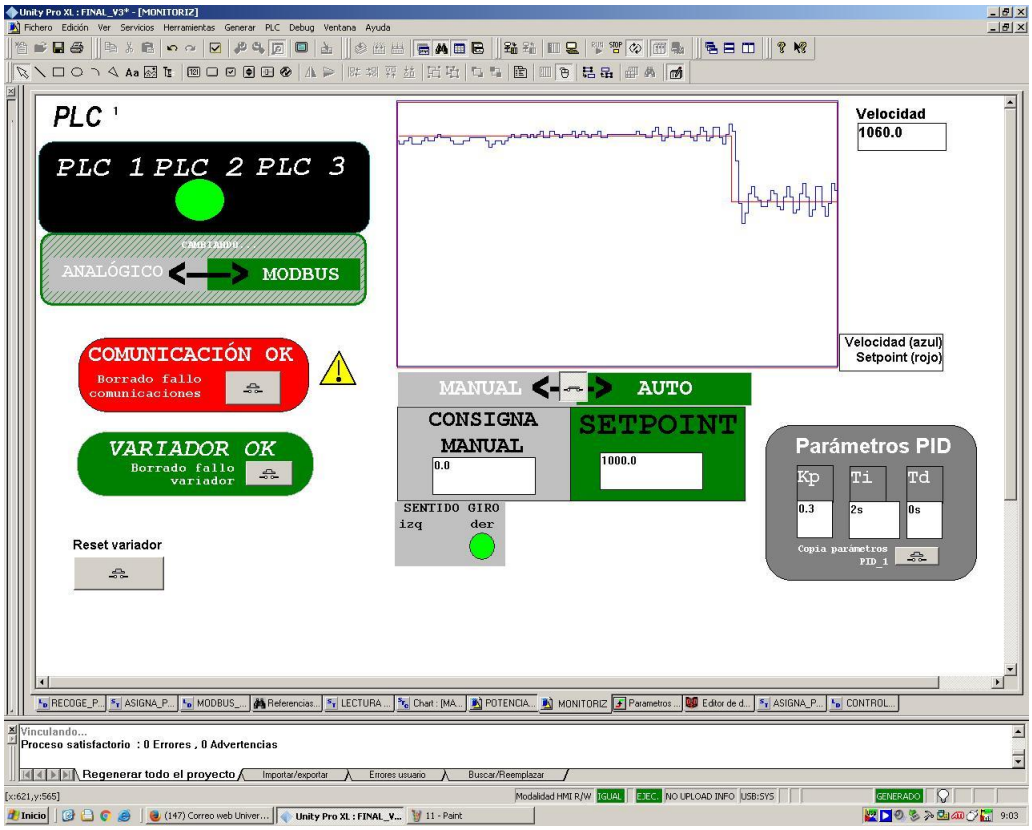


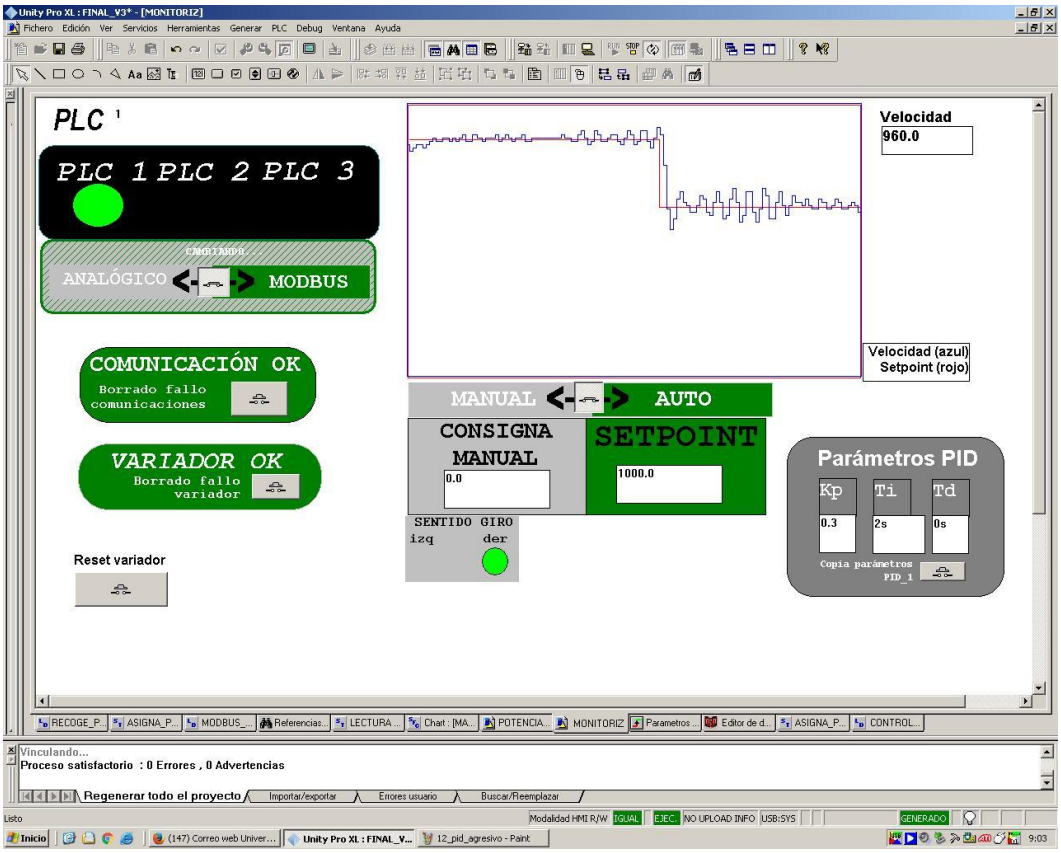


Finalmente, se realiza la misma prueba con un control más agresivo. Se conmuta al PLC_2 (pues no se recomienda modificar la parametrización del PLC_1), se cambia la Kp a 0.5, se baja el Ti a 1s y se cambia el Setpoint de 1400 a 1000rpm.



Como observamos, el cambio de referencia se produce más rápidamente, pero de forma más inestable. Finalmente se conmuta de nuevo al PLC_1 y recuperamos la estabilidad.





79 Simulación del control a través del sistema de PLCs redundantes

REFERENCIAS

[1] ABB, https//new.abb.com/drives/es/que-es-un-variador

[2] W.C.L.

«http//catalogo.weg.com.br/files/wegnet/WEG-seleccion-y-aplicacion-de-variadores-de-velocidad-articulo-tecnico-espanol.pdf »

[3] Datasheet sensor inductivo

« https//docs-emea.rs-online.com/webdocs/161e/0900766b8161eee2.pdf »

[4] TFG "Actualización de sistema de control del motor trifásico basado en el variador ATV320"

Javier Romero Suárez, 2018.

[5] TFG "Desarrollo e implementación de una red de datos basada en Modbus y Ethernet para autómatas industriales"

Marina Hernández Tinoco, 2016

« http://bibing.us.es/proyectos/abreproy/90946/fichero/marhertin.pdf »

[6] Variador *ATV320* - Manual de programación

« https//www.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/308000/FA308064/es_ES/ATV320_Programming_manual_SP_NVE41298_01b.pdf »

[7] Tarjeta E/S Discretas *BMX DDM 16022* – Datasheet

« https//www.schneider-electric.com/en/product/download-pdf/BMXDDM16022 »

[8] Variador *ATV320* - Communication Parameters

« https//www.schneider-electric.com/en/download/document/NVE41316/ »

[9] Variador *ATV320* - Modbus Manual

« https//www.schneider-electric.com/en/download/document/NVE41308/ »

[10] Módulo de conteo *BMX EHC 0200* - Manual de usuario

«https//www.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/33000/FA33770/en_US/Unity%20v50%20-%20Contaje%20BMXEHC0200.pdf »

[11] F.I.S.L. <<FRENOS INDUSTRIALES>>

« http//www.frenos.info/ »

[12] « https//es.wikipedia.org/wiki/Sensor_inductivo »