UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS
INDUSTRIALES

Ingeniería Técnica Industrial especialidad en Electrónica Industrial

Departamento de Ingeniería de Sistemas y Automática

CONTROL ALGEBRÁICO DE NIVEL SOBRE UN SISTEMA DE TANQUES COMUNICANTES DE FLUIDOS

Julio 2012

Autor: Ismael García Sanz
Tutor: Francisco Javier García Ruiz

ÍNDICE

1. INTRODUCCIÓN 7
1.1 Antecedentes 7
1.2 Objetivo del proyecto 7
1.3 Planteamiento adoptado 8
2. MATERIALES UTILIZADOS 9
2.1 Introducción 9
2.2 Sistema de tanques 10
2.3 Electrobomba de corriente continua 11
2.4 Transductor de nivel 13
2.4.1 Introducción 13
2.4.2 Sonda de nivel DC11 TEN 14
2.4.3 Dispositivo electrónico FEC 12 16
2.5 Dispositivos de protección 18
2.6 Alimentación, etapa de potencia y convertidor corriente-tensión 19
2.6.1 Introducción 19
2.6.2 Módulo 20
2.6.3 Fuente de alimentación 24
2.6.4 Transformador reductor 25
2.6.5 Etapa de potencia 26
2.6.6 Convertidor corriente tensión 28
2.6.7 Interface del módulo 29
2.7 Autómata 30
2.7.1 Introducción 30
2.7.2 Bastidor (BMX XBP 0400) 32
2.7.3 Fuente de alimentación (BMX CPS 2000) 34
2.7.4 CPU (BMX P34 1000 37
2.7.5 Módulo E/S (BMX AMM 0600) 43
2.7.5.1 Introducción 43
2.7.5.2 Descripción y características técnicas 43
2.7.5.3 Esquema de conexiones 47
2.7.6 Módulo E/S (BMX DDM 16025) 49
2.7.6.1 Introducción 49
2.7.6.2 Descripción y características técnicas 49
2.7.6.3 Esquema de conexiones 52
3. SOFTWARE UTILIZADO 55
3.1 Introducción 55
3.2 Unity Pro 56
3.2.1 Introducción 56
3.2.2 Entorno gráfico del programa 57
3.2.3 Variables del programa 61
3.2.4 Programación de aplicaciones 64
3.2.5 Simulador del PLC 70
3.2.6 Depuración de programas 71
3.2.6.1 Animación dinámica 71
3.2.6.2 Tablas de animación 72

3.2.6.3 Pantallas del operador 73
4. IDENTIFICACIÓN DEL SISTEMA 75
4.1 Modelado del sistema de tanques 75
4.2 Diseño del controlador del sistema 81
4.2.1 Consideraciones previas 81
4.2.2 Cálculo de los parámetros del controlador 82
5. DESARROLLO DE LA APLICACIÓN 89
5.1 Proyecto en Unity Pro 89
5.1.1 Creación del proyecto 90
5.1.2 Configuración 91
5.1.3 Creación y asignación de variables 98
5.1.4 Programación en lenguaje LD y ST 101
5.1.5. Tabla de animación 111
5.1.6. Pantallas del operador 113
5.1.7. Generación del proyecto 118
5.1.8. Conexión con el autómata 120
5.1.9. Transferencia del proyecto al autómata 121
5.1.10. Ejecución 123
5.2. Secciones programadas 124
5.2.1. Principal 125
5.2.2. Funcionamiento 1 126
5.2.3. Funcionamiento 2 129
5.2.4. Emergencia 132
5.2.5. Tanque lleno 133
5.2.6. Llenado tanque 1 134
5.2.7. Llenado tanque 2 138
5.2.8. Error sonda 140
5.2.9. Reseteos 142
5.2.10. Parada del motor 146
5.2.11. Estado de los niveles 149
5.2.12. Estado de los motores 152
5.2.13. Estado de las válvulas 155
5.2.14. Representación de las válvulas 158
6. RESULTADOS Y CONCLUSIONES 161
6.1 Resultados obtenidos 161
6.2. Conclusiones 168
6.3. Líneas de desarrollo futuras 169
7. BIBLIOGRAFIA 171
8. ANEXOS 173
ANEXO I 173
ANEXO II 175

MEMORIA

1. INTRODUCCIÓN

1.1 Antecedentes

En la actualidad, la electrónica está presente en muchas aplicaciones de medición, monitorización, control y automatización de procesos industriales.

El control de nivel en un sistema de tanques comunicados entre sí mediante distintos tipos de válvulas es uno de los problemas más comunes en el control de sistemas de proceso industriales, siendo necesario controlar en todo momento el nivel de los tanques y el caudal que atraviesa las válvulas.

Este control de nivel y el flujo se hace cada día más preciso debido al gran desarrollo de la electrónica, la cual nos suministra sistemas de medición de nivel cada vez más sofisticados y económicos. No obstante, los principios físicos tradicionales tienen particularidades técnicas o prácticas que los mantienen siempre vigentes, una combinación efectiva de ambos dará lugar a la solución correcta.

1.2 Objetivo del proyecto

La aplicación consiste en controlar el nivel de un sistema de tanques de fluidos comunicados entre sí, cada uno de los cuales tiene un sistema de desagüe por gravedad.

La aplicación será capaz de controlar los niveles de fluido de los tanques en todo momento. Estos niveles se miden mediante dos sondas capacitivas y siempre deben estar en torno a los niveles de referencia requeridos con la menor deriva y en el menor tiempo posible.

Nuestro sistema también deberá detectar los posibles atascos en cualquiera de las tres válvulas, tanto en la válvula de cada tanque como en la que comunica ambos. Además, tendrá que detectar que válvula es la que se encuentra cerrada.

La introducción del nivel de referencia deseado por el usuario y la detección de la posible válvula cerrada se podrán visualizar en una de las pantallas de las que la aplicación de software utilizada dispone.

El usuario de la aplicación también podrá enviar órdenes al sistema y conocer información sobre él mediante una caja de mando diseñada para la aplicación.

Esta caja constará de 3 botones: marcha/paro, emergencia y reset; y de 3 leds: emergencia, bomba 1 activada y bomba 2 activada.

1.3 Planteamiento adoptado

Para la realización del proyecto hay que seguir una serie de fases:

1) Identificar la planta para calcular su función de transferencia correspondiente.

2) Diseñar un controlador adecuado a nuestros requerimientos y para las características de la planta identificada.

3) Una vez tenemos las funciones de transferencia de la planta y del controlador, se desarrolla el software propio del autómata mediante la implementación del controlador diseñado.

4) Desarrollar e implementar la parte de software encargada de la detección de atasco de las válvulas.

5) Diseñar la caja de mandos con los elementos nombrados anteriormente.

6) Por último, realizar una serie de pruebas que verifiquen el correcto funcionamiento del sistema.

2. MATERIALES UTILIZADOS

2.1 Introducción

Nos basaremos en los denominados diagramas de bloques para realizar una introducción sobre los materiales utilizados. En dichos diagramas se representan tanto las variables controlables como las no controlables mediante flechas, mientras que los bloques representan los diferentes procesos del sistema.

Tenemos un sistema de control en lazo cerrado ya que se usa retroalimentación desde el resultado final para ajustar la acción de control.

A continuación mostramos el diagrama de bloques de nuestra aplicación:

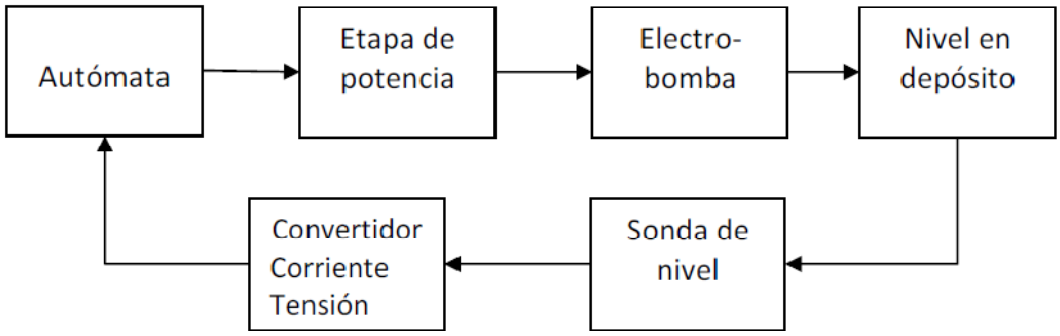


Figura 2.1.1. Diagrama de bloques de la aplicación

La acción de control será el accionamiento de la electrobomba por parte del autómata a través de la etapa de potencia que adecuará esta acción de control a los rangos de funcionamiento del motor. La realimentación será la señal correspondiente al nivel de líquido en el depósito que enviará la sonda capacitiva hacia el autómata mediante un convertidor que transforma la corriente proporcionada por la sonda a tensión a través del dispositivo FEC12.

A continuación describiremos todos los elementos físicos que intervienen en nuestra aplicación.

2.2 Sistema de tanques

El conjunto de depósitos utilizados proviene de aplicaciones y proyectos 'fin de carrera' anteriores.

Se trata de un sistema de tanques fabricados en metacrilato compuesto de tres depósitos, dos de los cuales están en posición vertical para realizar medidas de nivel y el otro se encuentra en posición horizontal, actuando como depósito auxiliar de los otros tanques. Cada uno de los tanques verticales está comunicado con el tanque horizontal y con el otro tanque vertical, los cuales son controlados manualmente por medio de llaves de paso. Estas tienen un diámetro de paso de 5mm.

En la siguiente imagen mostramos las dimensiones y características del montaje utilizado:

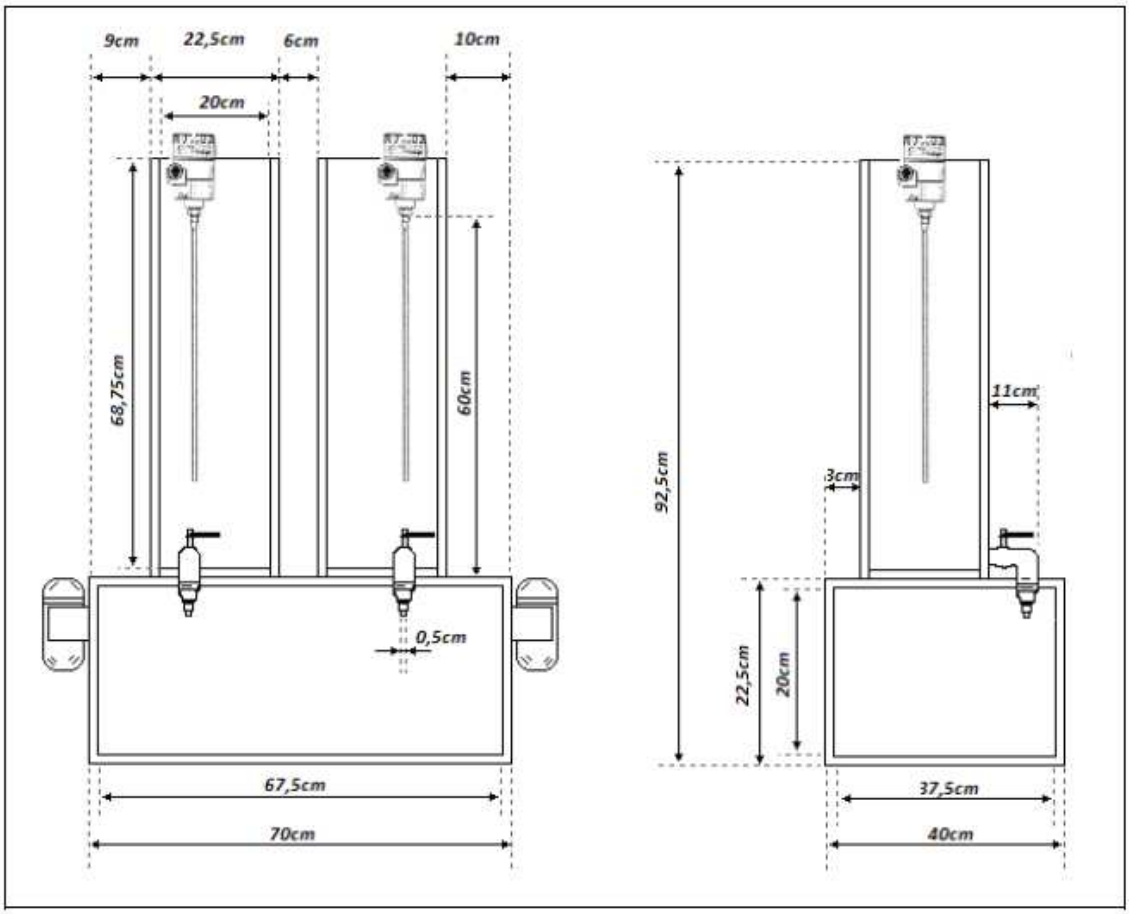


Figura 2.2.1 Esquema del sistema de tanques

2.3 Electrobomba de corriente continua

Estas electrobombas son las encargadas de suministrar caudal a los tanques. Se trata del modelo *Fiamma SuperFlow SF126 Pro.*

Es una bomba de tipo membrana o diafragma con desplazamiento positivo. El aumento de presión en el interior de esta se realiza mediante unas paredes elásticas que varía el volumen de la cámara aumentándolo o disminuyéndolo alternativamente. Dispone también de unas válvulas de retención que controlan que el movimiento del fluido vaya de la zona de menor presión a la de mayor presión.

Este tipo de bombas son autocebantes, es decir, no es necesario para su funcionamiento llenar la columna de aspiración de líquido. Esto resultará útil cuando se quiera extraer líquidos de depósitos en los que la cañería de aspiración pueda tener aire al inicio del bombeo.

Otras características importantes de este tipo de bombas son la resistencia y la robustez a la corrosión, lo que hace que sean de las más utilizadas en el ámbito industrial.

En la siguiente imagen mostramos la bomba descrita:

Figura 2.3.1 Aspecto exterior de la autobomba Fiamma SuperFlow SF126 Pro

El caudal suministrado por la bomba depende de la intensidad suministrada, es decir, del voltaje, el cual será objeto de nuestro control.

No hay una descripción muy extensa sobre este tipo de bombas, así que daré las características obtenidas:

Tensión de alimentación	24 V
Consumo de corriente	0.6 A
Caudal máximo teórico	6 l/min
Diferencia de presión máxima	1.3 bar

Tabla 2.3.1 Características de la autobomba Fiamma SuperFlow SF126 Pro

2.4 Transductor de nivel

2.4.1 Introducción

Los transductores, constituidos por un sensor y circuitos electrónicos, posibilitan la conversión de magnitudes físicas no eléctricas en magnitudes eléctricas, ya sean tensiones o corrientes DC.

Los transductores de nivel son muy utilizados en todo tipo de procesos industriales ya que el control de nivel es imprescindible cuando se quiere tener una producción continua, cuando un proceso requiere el control y medición de niveles de líquidos o también para evitar que un líquido se derrame.

Nuestro sistema requiere de un transductor de nivel continuo que convierta el nivel de líquido en una intensidad de corriente determinada. Nuestro transductor de nivel tiene el siguiente aspecto:

Figura 2.4.1.1 Transductor de nivel

Como podemos apreciar en la imagen nuestro transductor de nivel está compuesto por dos partes diferenciadas: la sonda de nivel encargada de medir la altura del líquido y un dispositivo electrónico que se encargará de convertir el nivel en la magnitud eléctrica correspondiente.

2.4.2 Sonda de nivel DC11 TEN

La sonda que vamos a utilizar en nuestra aplicación es de tipo capacitivo, la cual está diseñada para realizar medidas de nivel y detectar niveles límites. Esta sonda se basa en una barra cilíndrica hueca que permanecerá parcialmente sumergida en el líquido. Su capacidad eléctrica variará según el nivel de líquido. Cuanto mayor sea este nivel menor será la capacidad eléctrica de la sonda.

A continuación se muestra una imagen detallada de la sonda y su tabla de características.

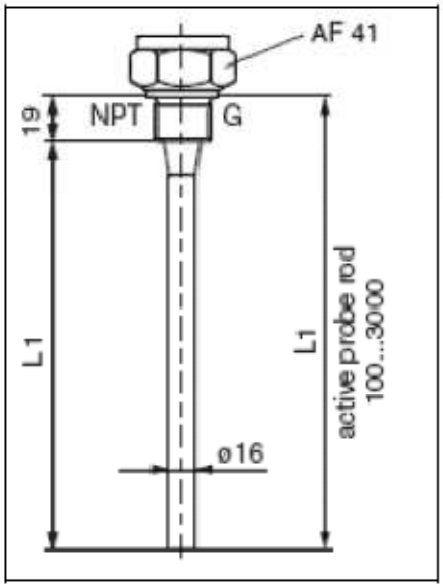


Figura 2.4.2.1 Sonda de nivel DC11 TEN

Error de longitud de la sonda	0mm-5mm
Dilatación de la sonda	<1% *por grado Kelvin*
Presión máxima de operación	25 *bar*
Temperatura máxima de operación	200ºC
L1	40cm

Tabla 2.4.2.1 Características técnicas de la sonda de nivel DC11 TEN

Las características máximas de operación se relacionan gráficamente de la siguiente manera:

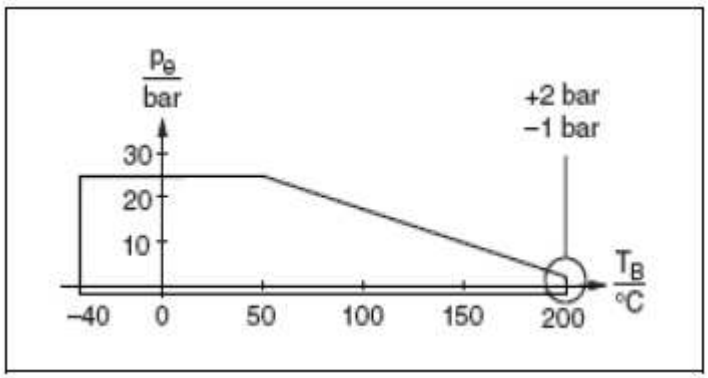


Figura 2.4.2.2 Relación presión-temperatura de utilización

2.4.3 Dispositivo electrónico FEC 12

Este dispositivo se encarga de transformar la capacidad eléctrica, correspondiente con el nivel de líquido, en una variación de corriente proporcional. Se encuentra situada en la parte superior de la sonda.

El dispositivo precisa de corriente continua. El cable de alimentación se usa además para transmitir la medida realizada por el sensor.

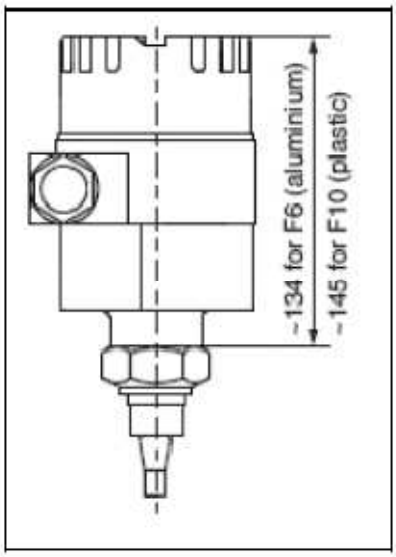


Figura 2.4.3.1 Dispositivo electrónico FEC 12

El conjunto que forman la sonda y el dispositivo se dispone en el tanque de la siguiente manera:

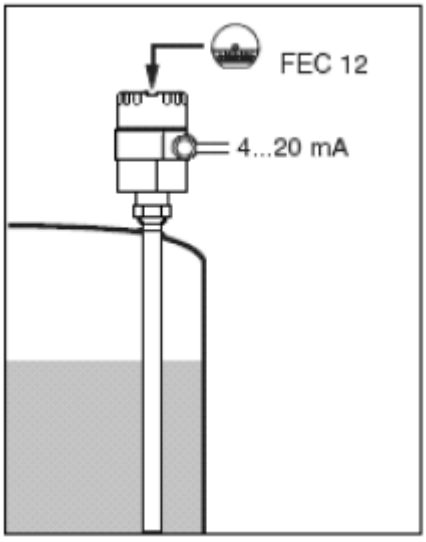


Figura 2.4.3.2 Montaje del conjunto sonda-dispositivo electrónico

La principal característica del dispositivo electrónico FEC 12 es la corriente proporcional al nivel del tanque. Podemos observar las características de este dispositivo en la tabla adjunta:

Resistencia de carga	0Ω – 720Ω
Fuente de alimentación necesaria	12V – 30V DC
Máximo rizado admisible	3% del voltaje de la fuente
Consumo	3.8mA – 22mA
Capacidad para tanque "vacío" ajustable	0 pF – 350 pF
Capacidad para tanque "lleno" ajustable	25pF – 2000pF
Corriente impuesta para tanque "vacío"	4mA
Corriente impuesta para tanque "lleno"	20mA
Resolución	14μA
Corriente impuesta como señal de alarma	22mA ± 0.1mA
Constante de tiempo 0 segundos	Ajustable hasta 40seg
Temperaturas de operación	0º C – 70º C

Tabla 2.4.3.1 Características técnicas del dispositivo electrónico FEC 12

2.5 Dispositivos de protección

A la hora de diseñar nuestro sistema hay que tener en cuenta la protección necesaria ante fallos eléctricos eventuales para que únicamente se interrumpa el circuito respectivo sin perjudicar el resto de la instalación. Para ello utilizaremos dispositivos de protección para proteger a nuestro sistema de sobrecargas, cortocircuitos y fallos de aislamiento.

En nuestro caso disponemos de tres tipos de protección: un interruptor diferencial, un interruptor magnetotérmico y un telerruptor.

El interruptor diferencial es un elemento destinado a la protección de las personas contra contactos indirectos. Corta la corriente que circula por la fase y el neutro desconectando el circuito si se produce un fallo de aislamiento en alguno de los componentes del montaje, es decir, si la fase entra en contacto con algún elemento conductor se origina una descarga a tierra con lo que la corriente que circula por el neutro es menor que la que circula por la fase, teniendo estas que ser iguales.

El interruptor magnetotérmico cuenta con un sistema mágnetico de respuesta rápida ante sobrecorrientes elevadas (cortocircuitos), y una protección térmica basada en un bimetal que desconecta ante sobrecorrientes más lentas (sobrecargas). Su misión es proteger cada circuito de la instalación, resguardando principalmente los conductores eléctricos ante sobrecorrientes que puedan producir elevaciones de temperaturas peligrosas.

Un telerruptor es un interruptor gobernado a distancia. Está constituido por un electroimán y uno o varios contactos. Al recibir un impulso eléctrico la bomba del electroimán hace cambiar la posición del contacto, permaneciendo en esta posición hasta recibir un nuevo impulso. Me permite maniobrar en baja tensión los circuitos alimentados a red.

El conjunto de estos tres dispositivos de protección está ubicado dentro de la caja del autómata,

2.6 Alimentación, etapa de potencia y convertidor corriente-tensión

2.6.1 Introducción

Se ha optado por usar como base la caja de mando de un antiguo proyecto fin de carrera ya que sólo es necesario realizar la parte de software para esta aplicación.

Dicha caja de mando engloba una serie de módulos diferenciados: fuente de alimentación, la transformación de la tensión de la red a la tensión de la fuente de alimentación con la que trabajemos, dos placas de circuito impreso similares encargadas de la etapa de potencia y de el convertidor corriente tensión de cada uno de los subgrupos bomba-sensor, las cuales están situadas en dos lugares distintos debido a la reutilización de esta caja de mando; y la placa de circuito impreso que engloba la adaptación de la alimentación y el montaje físico de los leds de información de estado.

Procederemos a un estudio más detallado de cada uno de estos módulos en los apartados posteriores.

2.6.2 Módulo

El módulo de mando se trata de una caja de aluminio que se encarga de gobernar el funcionamiento de la aplicación y visualizar el estado de ejecución de dicha aplicación.

En una de sus partes frontales se encuentran los correspondientes conectores encargados del envío de información al exterior y de la alimentación de los diferentes módulos. También se encuentran en esta parte de la caja el interruptor general de puesta en marcha y las carcasas que alojan los fusibles anteriormente mencionados para tener un fácil acceso a la hora de sustituirles.

En el reverso de la tapa se encuentran dos placas de circuito impreso. El primero de ellos es el encargado de adaptar la alimentación de cada uno de los leds de información y dar situación de los mismos. Esta placa de circuito impreso consta de tres leds: uno verde a la derecha que indica el funcionamiento de la bomba 1, uno rojo en el centro indicando emergencia y uno verde a la izquierda que indica el funcionamiento de la bomba 2. También incorpora toda la botonería de control de la aplicación: botón de encendido, botón de reset y pulsador de emergencia.

El segundo circuito impreso implementa conjuntamente la etapa amplificadora de potencia y el convertidor corriente-tensión del subgrupo bomba-sensor 1.

La caja de mando también dispone de un ventilador que canaliza un flujo de aire fresco del exterior al interior de la caja. Esto sirve para disipar las altas temperaturas interiores que se pueden alcanzar como consecuencia del funcionamiento del transistor de la etapa de potencia.

Podemos ver todo el conjunto de montaje en la siguiente imagen:

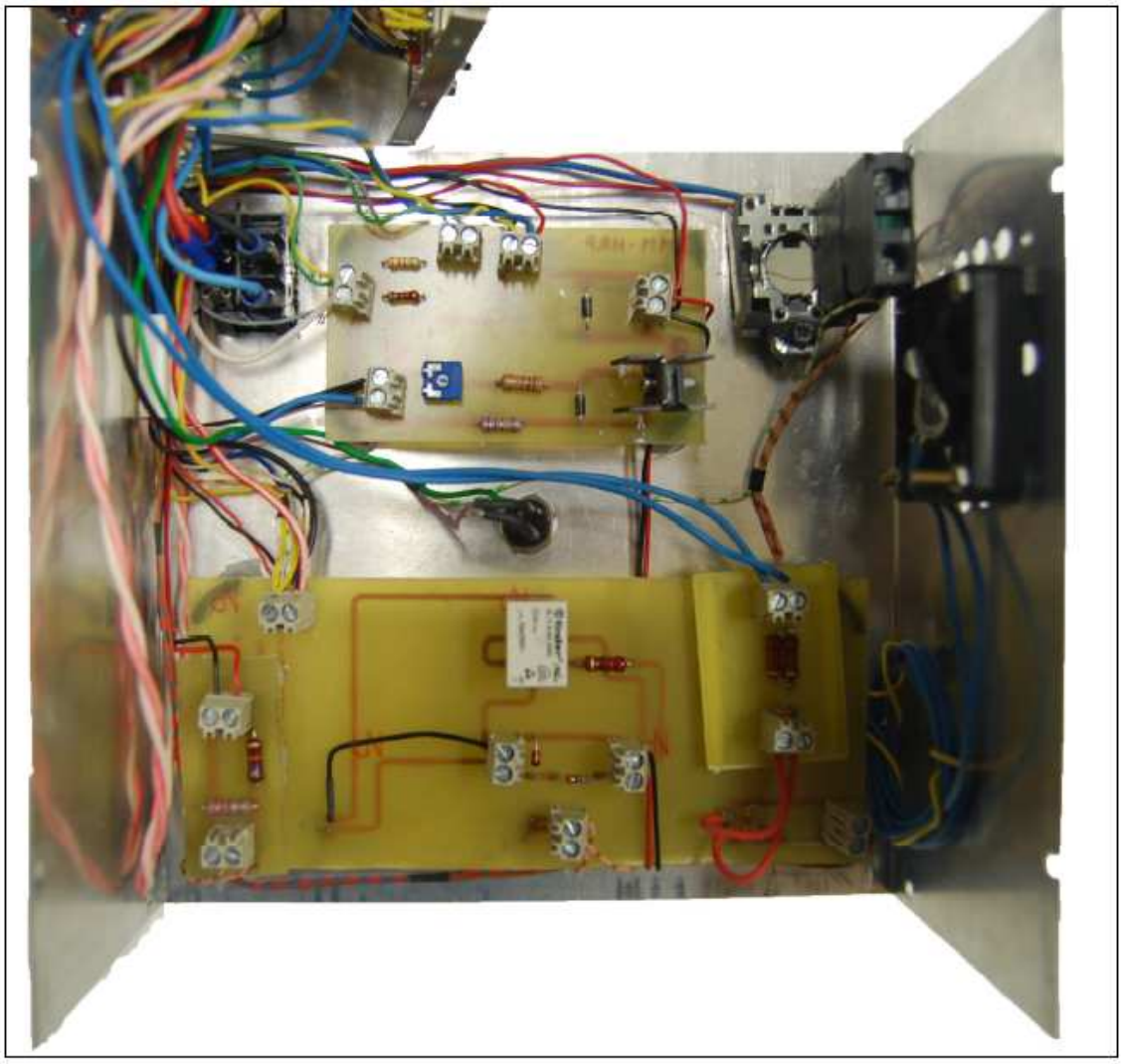


Figura 2.6.2.1 Reverso de la tapa con los circuitos impresos

Como podemos ver, el interior del módulo de mando está compuesto por los siguientes elementos: un circuito impreso similar al del reverso de la tapa el cual implementa conjuntamente la etapa amplificadora de potencia y el convertidor corriente-tensión del subgrupo bomba-sensor 2, el transformador de alimentación, la fuente de corriente y el interface exterior, encargado de las conexiones con el exterior y del emplazamiento de los elementos de protección del conjunto.

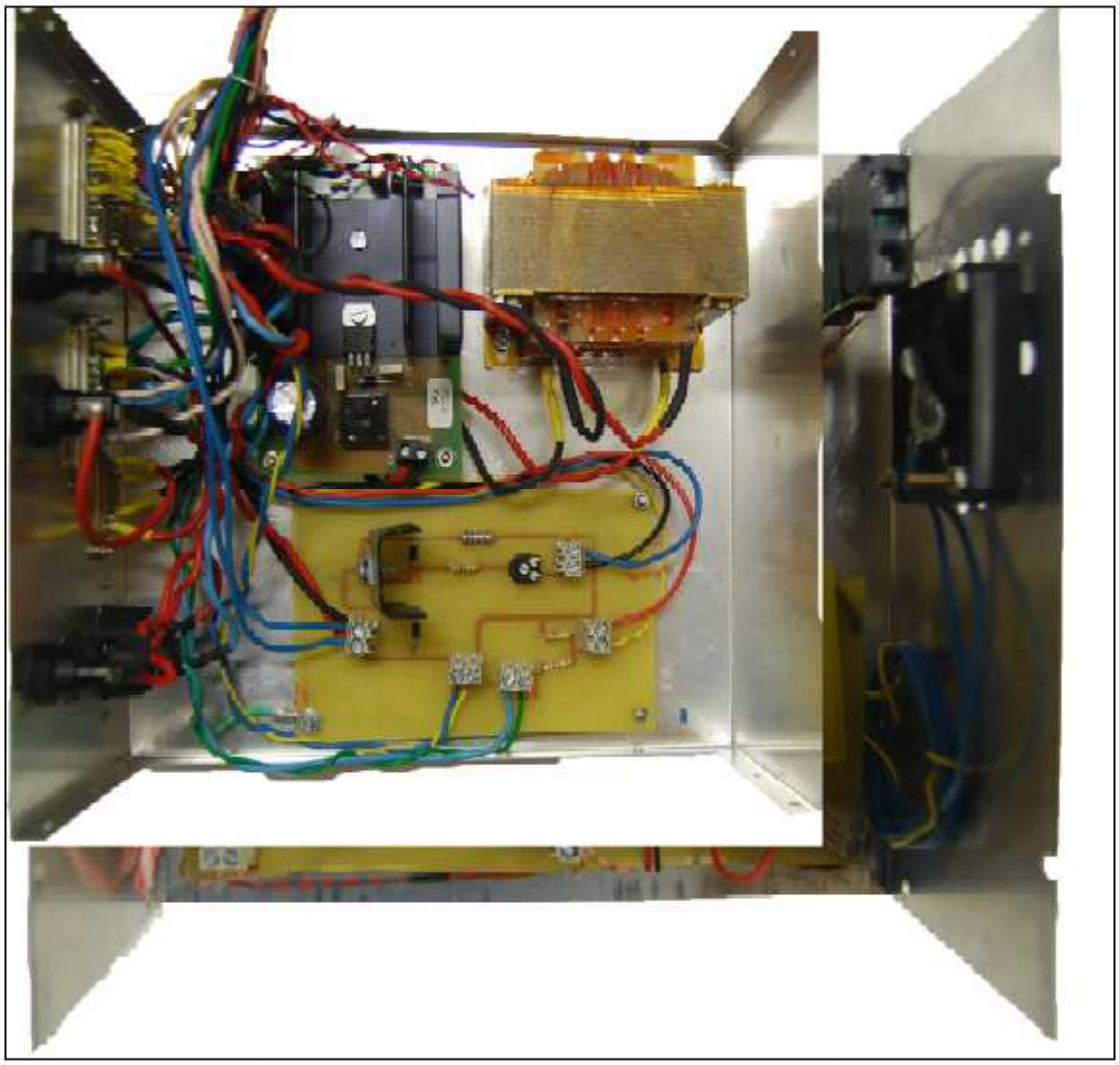


Figura 2.6.2.2 Interior de la caja de mando

En las dos siguientes imágenes mostramos un esquema con las dimensiones del módulo de control y una vista real de la botonería y los leds de dicho módulo.

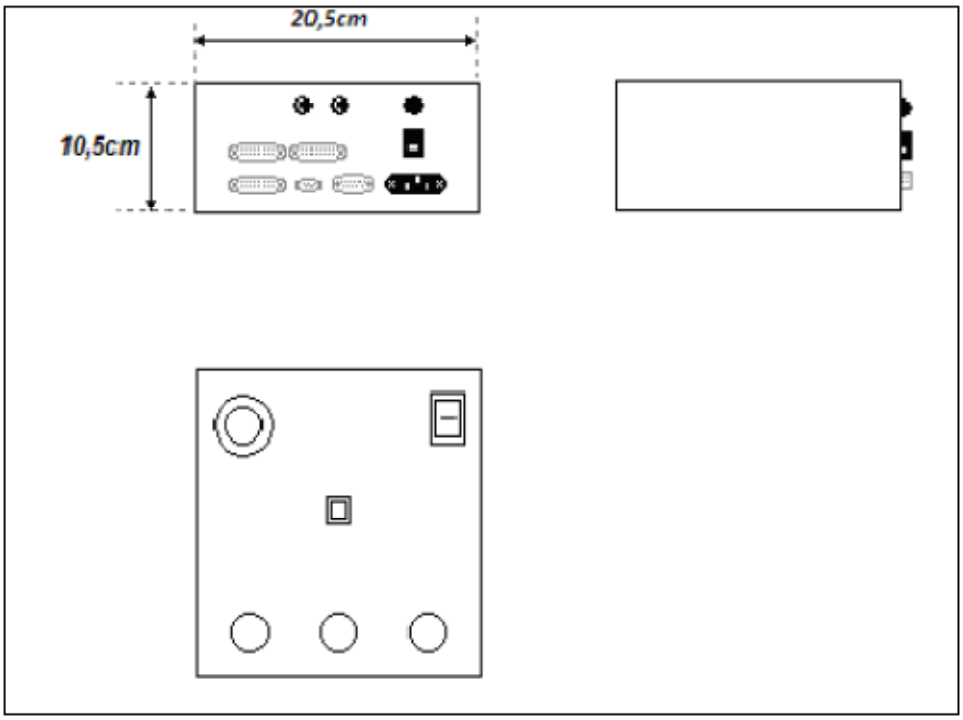


Figura 2.6.2.3 Esquema del módulo de mando

Figura 2.6.2.3 Botonería y leds del módulo

2.6.3 Fuente de alimentación

La fuente de alimentación utilizada es el modelo *FE 14* de la marca *Cebek.* Se trata de una fuente de alimentación perfectamente estabilizada y cortocicuitable de 24 V, tiene una intensidad máxima de salida de 2 A y su rizado máximo con carga no pasa de 10mV.

Además la fuente de alimentación incorpora bornes de conexión para facilitar su montaje y un led indicador de trabajo. También dispone de protección contra la inversión de polaridad.

Otra característica importante es que también dispone de protección contra cortocircuitos siendo su tiempo máximo de actuación de 30 segundos. Por este motivo, cuando esta actúe deberá desconectarse el aparato alimentado y dejar enfriar la fuente durante un tiempo superior a 60 segundos.

En la siguiente tabla podemos ver las características de esta fuente:

Fuente de alimentación	CEBEK FE 14
Tensión que suministra	24*V*
Rizado máximo con carga	7*mV*
Tolerancia tensión de salida	2%
Intensidad máxima de pico	2*A*
Intensidad máxima constante	1.5*A*
Tiempo respuesta a cortocircuito salida	(Con carga) 0.4*seg*
Tiempo recuperación de Vo tras cortocircuito	(Con carga) 12*seg*

Tabla 2.6.3.1 Características técnicas de la fuente de alimentación

2.6.4 Transformador reductor

El transformador reductor es un dispositivo que convierte la energía eléctrica alterna de un cierto nivel de tensión en energía alterna de un nivel de tensión menor por medio de interacción electromagnética. En nuestro caso reduce la tensión de red a otra tensión más adecuada para ser tratada.

Está constituido por dos o más bobinas de material conductor, aisladas entre sí eléctricamente y por lo general enrolladas alrededor de un mismo núcleo de material ferromagnético. La única conexión entre las bobinas la constituye el flujo magnético común que se establece en el núcleo.

La corriente que circula por el arrollamiento primario genera una circulación de corriente magnética por el núcleo del transformador. A su vez, la corriente que circula por el núcleo genera una tensión en el arrollamiento secundario que será tanto mayor cuanto mayor sea el número de espiras de este y cuanto mayor sea la corriente magnética que circula por el núcleo.

Este transformador tiene una relación de transformación de 220V a 24V. Sus características generales son las siguientes:

Transformador	Saber modelo 10042
Primario	230*V*
Secundario	24*V*
Corriente máxima en el secundario	2 **A**
Protección del transformador	Fusible en el primario de 0.5ª

Figura 2.6.4.1 Características técnicas del transformador Saber 10042

2.6.5 Etapa de potencia

Los módulos amplificadores de potencia tienen como finalidad transformar el rango de tensión que entrega la salida del autómata [0-10V] en el rango de control de la bomba [0-24V].

En esta parte se utiliza una etapa amplificadora en emisor común utilizando un transistor BD135 NPN (características en el Anexo II). Este transistor permite modificar la corriente que alimenta al motor a partir de la tensión de base del transistor que tomaremos como salida analógica.

La configuración en emisor común que utilizamos en nuestra aplicación es la siguiente:

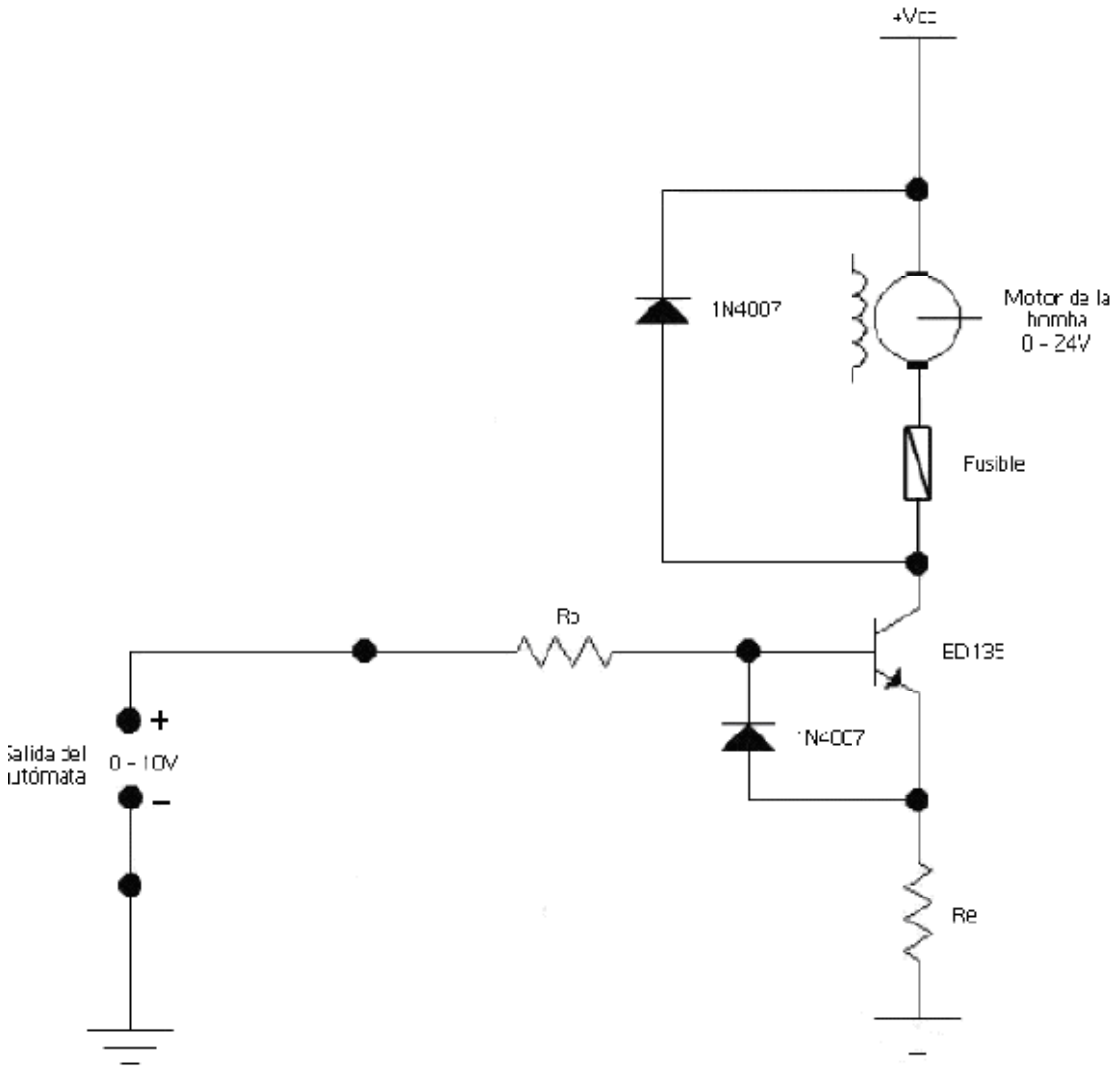


Figura 2.6.5.1 Configuración de la etapa de potencia en emisor común

Como podemos ver en el esquema, la tensión de entrada procede de la salida analógica que proporciona el autómata, la cual se encuentra en el rango [0-10V].

En serie con la salida del autómata tenemos la resistencia de base y el montaje del transistor en emisor común, responsable de la amplificación de la señal de entrada. Cabe destacar que la resistencia de emisor en este montaje del transistor aporta al sistema la estabilidad necesaria sin producir una caída de tensión excesivamente alta al tener un valor pequeño.

La resistencia de base consta de una resistencia conectada en serie con un potenciómetro, cuya función es posibilitar el ajuste del equipo en caso de que haya que sustituir el transistor por otro con distinta ganancia a causa de una avería. Pero si permitimos un ajuste fino de la resistencia de base y escogemos adecuadamente el transistor podemos volver a disponer del equipo con unas características de funcionamiento iguales a las anteriores.

Se añade un diodo polarizado en inversa para compensar las variaciones que puedan surgir por los cambios de temperatura y permitir un funcionamiento estable del transistor.

La carga conectada al colector es la electrobomba. En su estructura interna consta de una inductancia que puede dar lugar a transitorios durante la modificación de la corriente pudiendo llegar a producir una sobrecorriente desaconsejable para el transistor. Para evitar que esta sobrecorriente atraviese al transistor se conecta un diodo polarizado en inversa.

2.6.6 Convertidor corriente tensión

Disponemos también de un convertidor corriente-tensión que adapta la corriente del dispositivo FEC 12, el cual funciona como una fuente ideal con un rango de trabajo de [4-20 mA], al rango de tensión de la entrada analógica de la tarjeta, que tiene un rango de tensión de [0-10 V].

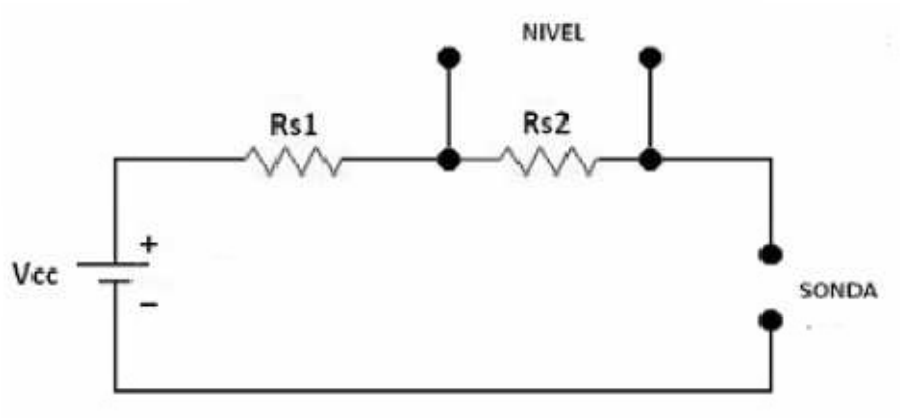


Figura 2.6.6.1 Convertidor corriente-tensión

La resistencia R_{s2} mide el nivel proporcional de tensión. R_{s1} se encarga de la protección de la sonda ante posibles valores peligrosos, como pudieran ser corotcircuitos. De esta forma el protocolo de comunicación interna que tienen las sondas para lectura y ajuste con terminales externos no sufriría ningún daño ante estos posibles problemas. Este circuito se encuentra implementado en la misma placa que la etapa amplificadora.

Los valores de los componentes utilizados en el convertidor son los siguientes:

Componente	*Descripción*	*Característica*
Vcc	Tensión de alimentación	*24V*
Rs1	Resistencia	*220Ω*
Rs2	Resistencia	*470Ω*
SONDA	Corriente del FEC 12	*4mA – 20mA*
NIVEL	Tensión respecto a la altura	*0V – 10V*

Tabla 2.6.6.1 Características componentes del convertidor

2.6.7 Interface del módulo

Los diversos conectores de alimentación, comunicación con el autómata y adquisición de señales de la sonda se encuentran en la parte trasera de la caja de mando.

En esta parte de la caja también se encuentra un conector de conexión a la red eléctrica y un interruptor basculante de corte de alimentación con su correspondiente fusible de protección.

Además, para la comunicación con el autómata la caja dispone de dos conectores RS232 de 25 pines cada uno, uno para la parte analógica y otro para la parte digital. Las señales correspondientes al nivel de líquido en el tanque provenientes de la sonda también se adquieren a través de este tipo de conector pero en este caso consta de 15 pines. Este mismo conector pero de tipo joystick se utiliza para el control del motor, estando protegida esta conexión por su fusible correspondiente.

Todos los elementos descritos en los párrafos anteriores se pueden distinguir en la siguiente imagen.

Figura 2.6.7.1 Aspecto de las conexiones de la caja de mando

La parte digital, conectada al módulo DDM 16025, es la encargada de gestionar las entradas de RESET y PARO de EMERGENCIA.

La parte analógica, conectada al módulo AMM 0600, es la encargada de gestionar las entradas NIVEL1_SONDA, NIVEL2_SONDA y ON/OFF, además de las salidas de MOTOR_1 Y MOTOR_2.

2.7 Autómata

2.7.1 Introducción

Puede definirse como un equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales.

En la actualidad los conocemos con el nombre de PLC's o controlador programable por lógica.

Lo que hace el PLC es revisar el estado de sus entradas y, dependiendo del estado de estas, cambia el estado de sus salidas, encendiéndolas o apagándolas.

Las partes fundamentales de un PLC son la CPU, la memoria y el sistema de entradas y salidas. Estas partes están descritas en los siguientes párrafos.

La CPU realiza el control interno y externo del autómata y la interpretación de las instrucciones del programa. A partir de las instrucciones almacenadas en la memoria y de los datos que recibe de las entradas, genera las señales de las salidas.

La memoria se encarga de almacenar el programa y los datos del proceso, además de diversas instrucciones, datos leídos de las entradas y programas generales para el correcto funcionamiento del sistema. La memoria se divide en dos bloques: la memoria RAM y la ROM. La primera se encarga de almacenar la información de los estados de las variables de entrada, salida e internas y también de almacenar el programa con el que trabajará el autómata. La segunda se encarga de almacenar programas para el correcto funcionamiento del sistema.

El sistema de entradas y salidas recoge la información del proceso controlado (entradas) y envía las acciones de control del mismo (salidas).

El modo de operación de un PLC consiste en efectuar un barrido continuo del programa. Estos ciclos de barrido, llamados *scan,* se pueden entender como una ejecución consecutiva de tres pasos principales como podemos ver en el siguiente gráfico.

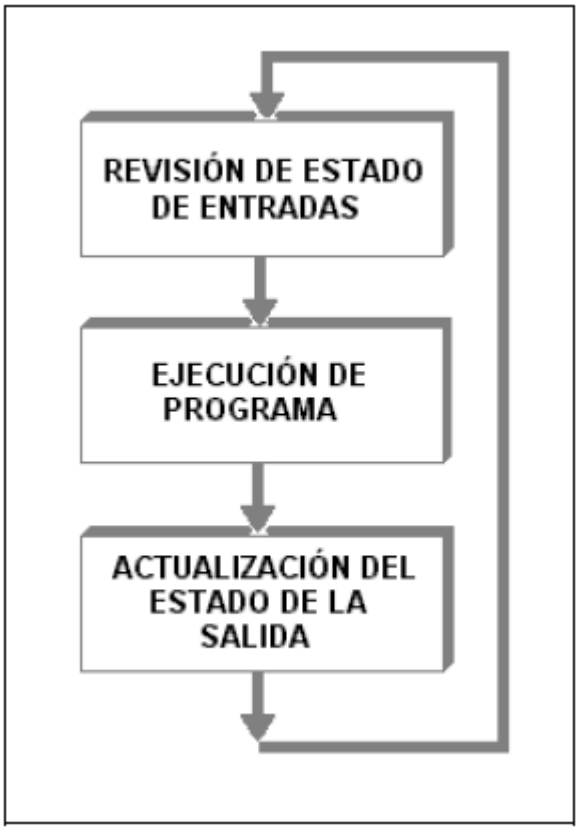


Figura 2.7.1.1 Pasos principales de la operación de un PLC

En la imagen podemos distinguir tres pasos en la operación de un PLC:

1. Revisar cada una de las entradas grabando su estado en la memoria para ser usadas en el siguiente paso.

2. Ejecución del programa instrucción por instrucción. En función del estado de las entradas el programa decidirá los valores que va a tomar cada una de las salidas, almacenándose estos en registros para ser utilizados en el último paso.

3. El PLC toma los valores almacenados tras la ejecución del programa y los va asociando uno a uno a cada una de las salidas en el orden que se haya establecido.

Los distintos elementos del autómata son descritos en apartados posteriores.

2.7.2 Bastidor (BMX XBP 0400)

El bastidor o rack es un elemento metálico que actúa de soporte para la plataforma del autómata usado en la aplicación, el *Modicon M340* de *Schneider Electrics.*

Se trata de un bastidor adaptable en carril DIN con 4, 6, 8 ó 12 módulos y una función de intercambio en funcionamiento *'Hot Swap'* para un mantenimiento sencillo.

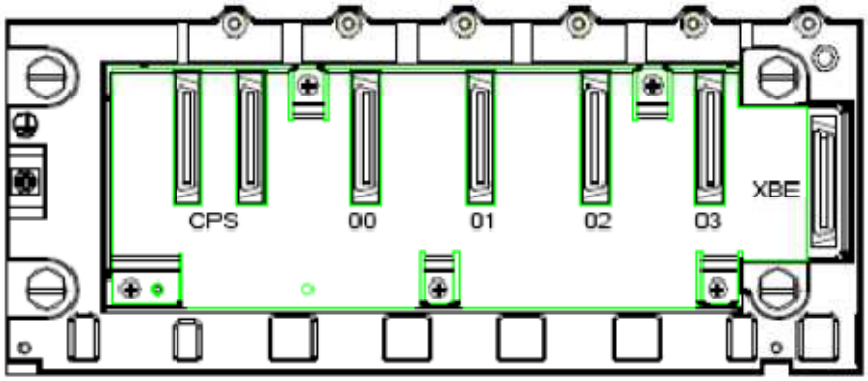


Figura 2.7.2.1 Esquema de conexiones del bastidor BMX XBP 0400

Como podemos ver en la imagen, utilizamos un modelo de 4 ranuras de conexión de módulos.

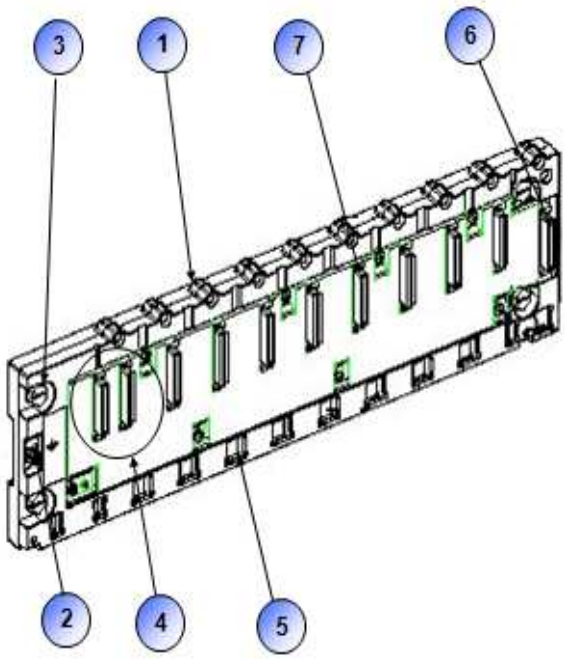
La ranura CPS queda reservada a la conexión del módulo de alimentación, la ranura 00 para la conexión del procesador y las restantes ranuras para los diferentes módulos de E/S específicos de cada aplicación. La ranura XBE se utiliza para el módulo de extensión.

La configuración de nuestra aplicación tiene el siguiente mapa de conexiones en el bastidor:

Elemento	*Descripción*	*Ranura*
BMX CPS2000	Fuente de alimentación	CPS
BMX P34100	Procesador	00
BMX DDM16025	E/S digitales	01
BMX AMM0600	E/S analógicas	02

Tabla 2.7.2.1 Mapa de conexiones de los módulos

Un rack estándar contiene una serie de elementos con una función tanto mecánica como eléctrica. En la siguiente imagen se muestran los elementos que componen un rack.



	Descripción
1	Soporte de metal el cual incorpora la carta electrónica de X-Bus , y protecciones contra interferencias EMI y ESD.
2	Terminales para puesta a tierra del Rack.
3	Agujeros para el montaje del Rack en un soporte (tornillos de M6)
4	2 Conectores 40 pines dedicados para la conexión del rack y de la fuente de alimentación
5	Parte mecánica dedicada a fijar el módulo
6	Conector 40 pines Dedicados para la extensión del Rack
7	Conector 40 pines dedicados para la conexión rack / modulo

Figura 2.7.2.2 Esquema general del bastidor

2.7.3 Fuente de alimentación (BMX CPS 2000)

Los módulos de alimentación están diseñados para alimentar cada rack BMX XBP xxx0 y sus módulos. El módulo de alimentación se selecciona de acuerdo a la distribución de red (continua o alterna) y la potencia requerida.

Tipo de módulo	Módulo		Consumo medio (*mA*)		
	Referencia	Descripción	Tensión; 3,3V	Tensión; rack 24V	Tensión; sensores 24V
Procesador	BMX P341000	CPU 340-10 Modbus	-	72	-
E/S analógicas	BMX AMM0600	4 entradas 10*VCC* 2 salidas 10*VCC*	150	130	-
E/S digitales	BMX DDM16025	8 entradas 24*VCC* binarias 8 salidas binarias	100	50	30

Corriente total (*mA*)	250	252	30
	x 3,3V	*x 24V*	*x 24V*
Consumo tensión (*mW*)	0,825	6,048	0,720
Potencia total necesaria (*W*)			7,593

Figura 2.7.3.1 Tabla consumos de potencia

A la vista de estos resultados, el modelo de fuente elegido es el BMX CPS 2000 ya que la red de alimentación es de corriente alterna y la potencia máxima de dicha red es de 20W en este modelo, por lo que se adapta perfectamente a nuestros requerimientos.

Las características de este módulo se exponen en la siguiente tabla.

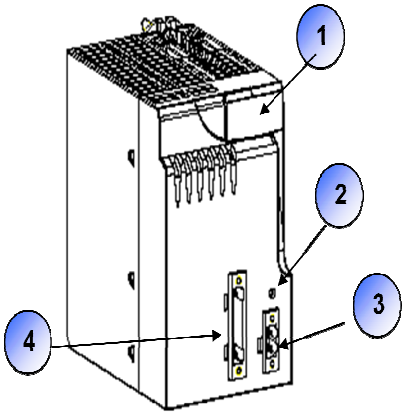
Características del bloque primario	Tensión	Nominal	100 – 120V/ 200 – 240V
		Límite (oscilación incluida)	85 – 264V
	Frecuencias	Nominal/límite	50 – 60/47 – 63Hz
	Potencia	Aparente	70VA
	Corriente	Nominal de entrada	0.61A a 115V 0.31A a 240V
	Conexión inicial a 25ºC (1)	Corriente de llamada	≤30A a 120V ≤60A a 240V
		Potencia en la activación	$\leq 0.5A^2s$ a 120V $\leq 2A^2s$ a 240V
		$I \cdot t$ de accionamiento	≤0.03As a 120V ≤0.06As a 240V
	Duración aceptada en los microcortes		≤10ms
	Protección integrada		Por fusible interno no accesible
Características del bloque secundario	Potencia útil	Max. global	20W
		Max. en tensiones de salida de rack	16.5W
	Tensión de salida a 3.3V (2)	Tensión nominal	3.3V
		Corriente nominal	2.5A
		Potencia típica	8.3W
	Tensión de salida a 24V (3)	Tensión nominal	24Vcc
		Corriente nominal	0.7A
		Potencia típica	16.5W
	Salida de captadores a 24V (4)	Tensión nominal	24Vcc
		Corriente nominal	0.45A
		Potencia típica	10.8W
	Protecciones integradas para tensiones (5)		Contra sobrecargas, cortocircuitos y sobretensiones

(1) Se tomarán en cuenta estos valores cuando estos equipos arranquen simultáneamente y para dimensionar los elementos de protección.
(2) Tensión de 3.3 V destinada a la alimentación lógica de los módulos de entrada/salida.
(3) Tensión de 24 V destinada a la alimentación de los módulos de entrada/salida y del procesador.
(4) Salida de captadores 24 V destinada a la alimentación de los captadores.
(5) Protegidas mediante un fusible al que no se puede acceder.

Potencia máxima disipada			8.5*W*
Características del aislamiento	Resistencia dieléctrica	Primaria/secundaria (24*V* / 3.3*V*)	1500*Vef*
		Primaria/secundaria (24*V* captadores)	2300*Vef*
		Primario/tierra	1500*Vef*
		Salida de captadores 24*V* / tierra	500*Vef*
	Resistencia de aislamiento	Primaria/secundaria y primaria/tierra	100*MΩ*

Tabla 2.7.3.2 Características del módulo de alimentación

El módulo de alimentación tiene el siguiente aspecto:



	Descripción
1 Led OK Led 24V	Bloque display conteniendo: LED OK (verde), encendido si el voltaje está presente y correcto, LED 24V (verde), encendido cuando el voltaje de alimentación sensores está correcto. Este LED solamente está presente en los módulos de alimentación alterna TSX CPS 2000/3500.
2	Botón de RESET. -El INIT_BAC está activado y se resetean todos los módulos en el Rack. -El relé de alarma se fuerza 'OPEN'. - El LED OK se apaga La acción sobre el botón de RESET es igual a un ARRANQUE EN FRIO.
3	Bloque terminal para el contacto de relé de alarma
4	Bloque terminal para la alimentación. Alimentación de sensores (TSX CPS 2000/3500)

Figura 2.7.3.1 Aspecto del módulo de alimentación

2.7.4 CPU (BMX P34 1000)

Las CPU MODICON M340 BMX P34 1000 es un procesador de la plataforma de automatización diseñado para controlar módulos de E / S discretas, módulos de E/ S analógicas y módulos de aplicaciones específicas. Este se conectará sobre el rack.

Este procesador puede trabajar en *Modbus* maestro/esclavo bajo protocolo RTU/ASCII o bajo protocolo modo de caracteres.

Las características generales de este procesador son las siguientes:

Configuración máxima	Número de Racks	4, 6, 8 o 12 emplazamientos	1
	Número máximo de emplazamientos para procesadores y módulos (sin contabilizar el módulo de alimentación)		12
Funciones	Número máximo (1)	E/S digitales	512
		E/S analógicas	128, 66 en configuración monorack (*4E/2S x 11*)
		Vías de regulación	Bucles programables (a través de biblioteca bloques de función)
		Vías de contaje	20
		Control de movimiento	-
	Conexiones integradas	Ethernet TCP/IP	-
		Bus CANopen maestro	-
		Conexión serie	1 puerto RJ45, Modbus maestro/esclavo RTU/ASCII o modo caracteres. 0,3...19,2*Kbits/s*
		Puerto USB	1 puerto 12*Mbits/s*
Memoria RAM de usuario interna	Módulo de comunicación	Ethernet TCP/IP	1 puerto RJ45, 10/100*Mbits/s*
	Capacidad total		2048*Kbits*
	Programa, constantes y símbolos		1792*Kbits*
	Datos		128*Kbits*

Tarjeta de memoria	Suministrada de serie (referencia BMX RMS 008MP)		Grabación del programa, constantes, símbolos y datos
Tamaño máximo de las zonas objetos	Bits internos localizados	Máximo	16250 *%Mi*
		Por defecto	256 *%Mi*
	Datos internos localizados	Máximo	32.464 palabras internas *%Mwi*, 32.760 palabras constantes *%Kwi*
		Por defecto	512 palabras int. *%Mwi* 128 palabras const. *%Kwi*
	Datos internos no localizados máximos		128*Kbits* (2)
Estructura de la aplicación	Tarea maestra		1 cíclica o periódica
	Tarea rápida		1 periódica
	Tareas auxiliares		-
	Tareas por suceso		32 (de los cuales 2 son prioritarios)
Tiempo de ejecución para una instrucción	Booleana		0,18 μs
	En palabras o aritmética de coma fija	Palabras de longitud simple	0,38 μs
		Palabras de longitud doble	0,26 μs
	En flotantes		1,72 μs
Número de *K*instrucciones ejecutadas por ms	100% booleano		5,4 $Kinst/ms$
	65% booleano y 35% aritmético fijo		4,2 $Kinst/ms$
Sistema Overhead	Tarea maestra		1,05 ms
	Tarea rápida		0,2 ms
Consumo	En tensión continua 24*V*		72 mA

(1) Sólo se refiere a los módulos 'in rack'. Las entradas/salidas distantes en bus CANopen no se tiene en cuenta en estos números máximos.

(2) El tamaño de los datos localizados (bits y datos internos) y el tamaño de los datos de configuración se restan de este valor.

Tabla 2.7.4.2 Tabla de características del procesador BMX P34 1000

El procesador BMX P34 1000 tiene el siguiente aspecto:

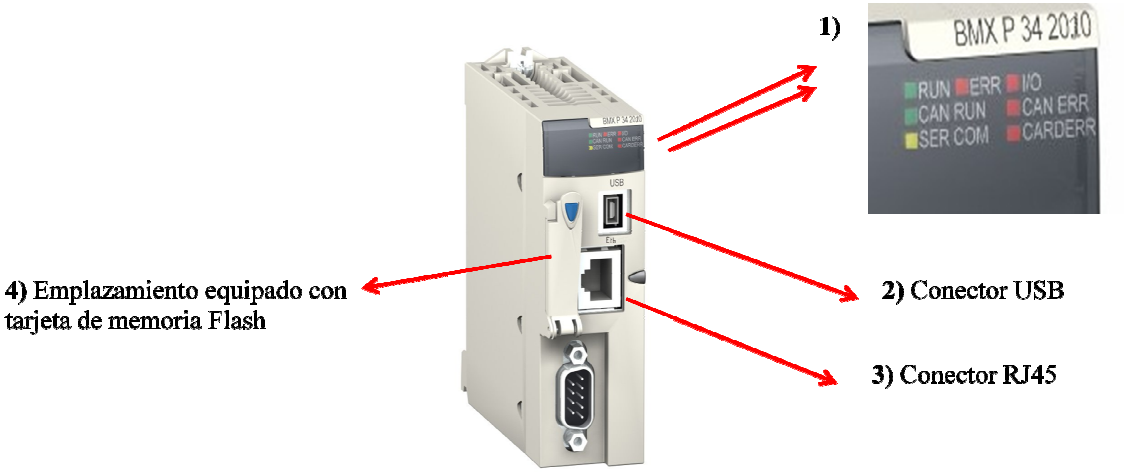


Figura 2.7.4.1 Aspecto general del procesador BMX P34 1000

En los siguientes puntos ampliamos la información sobre cada parte del procesador:

1) Bloque de visualización que dispone de 5 pilotos:

    a. RUN: Procesador en funcionamiento.
    b. ERR: Fallos del procesador o del sistema.
    c. E/S: Fallo procedente de los módulos de E/S.
    d. SER COM: Actividad en el enlace serie Modbus.
    e. CARD ERR: Ausencia o fallo de la tarjeta de memoria.

2) Conector tipo USB mini B para la conexión de un terminal de programación.

3) Conector tipo RJ45 para enlace serie *Modbus* o el enlace de modo caracteres.

4) Este emplazamiento con tarjeta de memoria sirve para realizar la copia de seguridad de la aplicación. Un piloto situado encima de este indica el reconocimiento o el acceso a la tarjeta de memoria Flash. Los procesadores CPU BMX P34 xxx de MODICON M340 usan una tarjeta de memoria BMX RMS 008MPx.

La memoria total de la aplicación está estructurada en dos bloques principales: la memoria RAM interna de la aplicación y la tarjeta de memoria.

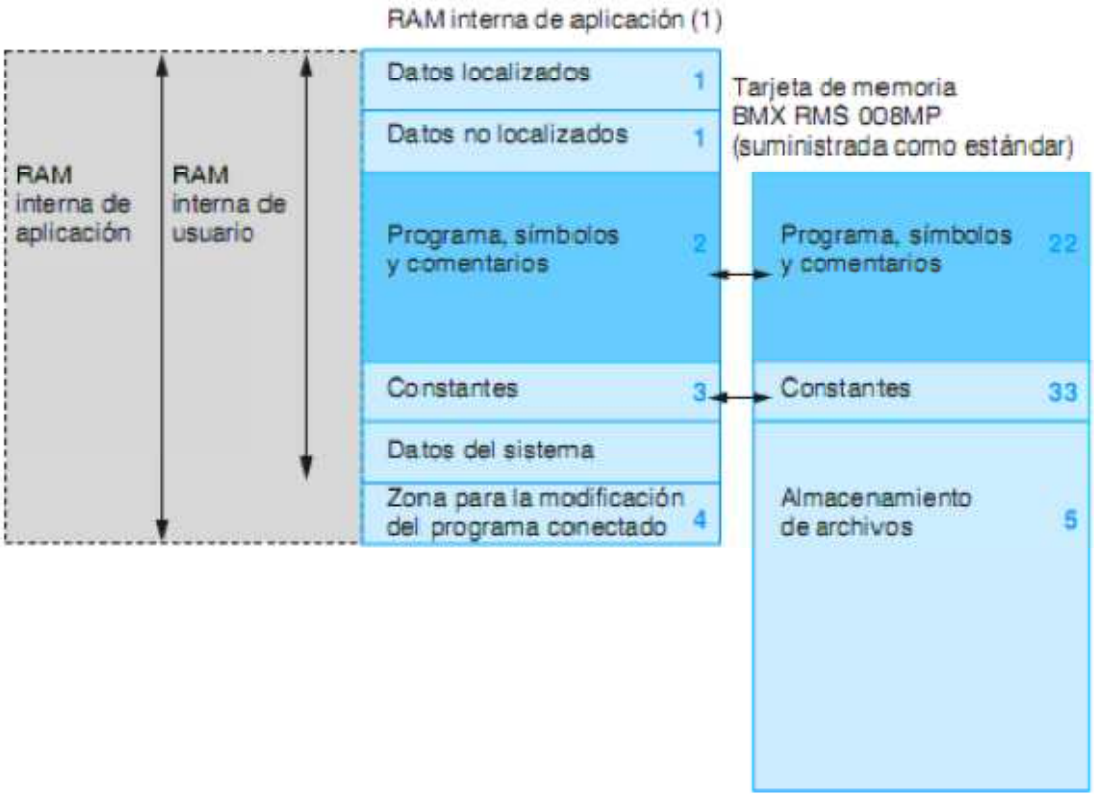


Figura 2.7.4.2 Estructura de la memoria

Como podemos apreciar en la imagen la memoria RAM se compone de las siguientes zonas:

1. Zona de datos de la aplicación de 2 tipos:

   - Datos localizados: Datos definidos por una dirección a la que se puede asociar un símbolo.

   - Datos no localizados: Datos definidos únicamente por un símbolo. La utilización de este tipo de datos elimina los problemas de gestión de la localización de la memoria debido a la atribución automática de direcciones además de permitir la estructuración y reutilización de los datos.

   Al desconectar el autómata, se realiza automáticamente una copia de esta zona de datos en una memoria interna no volátil de 256KBytes, la cual se encuentra integrada en el propio procesador.

2. Zona de programa, símbolos y comentarios: esta zona contiene el código binario ejecutable y el código fuente IEC del programa.

3. Zona de constantes: se almacenan en ella los datos de tipo constante.

4. Zona de modificación de programas en modo conectado.

   Esta zona permite al usuario de la aplicación transferir el código fuente con el programa ejecutable en el autómata. Esto permite, al conectar al autómata un terminal de programación, sin ninguna aplicación, restablecer en el terminal todos los elementos necesarios para la puesta a punto o evolución de la aplicación

Por otro lado, tenemos la tarjeta de memoria *Flash SD Card* (Secure Digital Card), la cual se suministra con todos los procesadores de la gama Modicon M340.

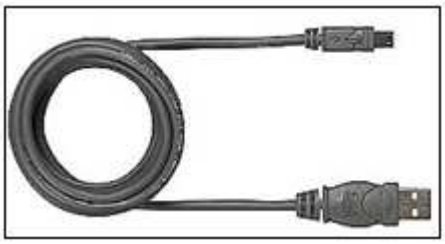
Esta tarjeta está destinada para la copia de la zona de programa, símbolos y comentarios y de la zona de constantes.

Las operaciones de duplicación, que se realizan en las zonas 22 y 33 respectivamente, y de restitución las gestiona automáticamente el sistema.

Esta tarjeta de memoria tiene una referencia de elemento de repuesto BMX RMS 008MP y garantiza la copia de seguridad de la aplicación soportada en memoria RAM interna volátil del procesador.

Respecto a las comunicaciones con el exterior, estas se pueden llevar a cabo por diferentes medios: para la conexión hombre-máquina se utiliza la conexión mediante USB y para las comunicaciones serie se utiliza la conexión/protocolo *Modbus*.

Para la conexión del interface hombre-máquina se utiliza un cable USB del modelo BMX XCA USB 018. En un extremo del cable tiene un conector USB de tipo A para conectar a la consola y en el extremo opuesto tiene un conector USB de tipo B que se conecta al procesador. En la siguiente imagen se muestra dicho cable.



2.7.4.3 Cable BMX XCA USB 018

Por otra parte, también tenemos la conexión mediante protocolo *Modbus,* el cual se ha llevado a *Ethernet TCP/IP*, para juntos formar *Modbus TCP/IP*, un protocolo totalmente abierto en *Ethernet*.

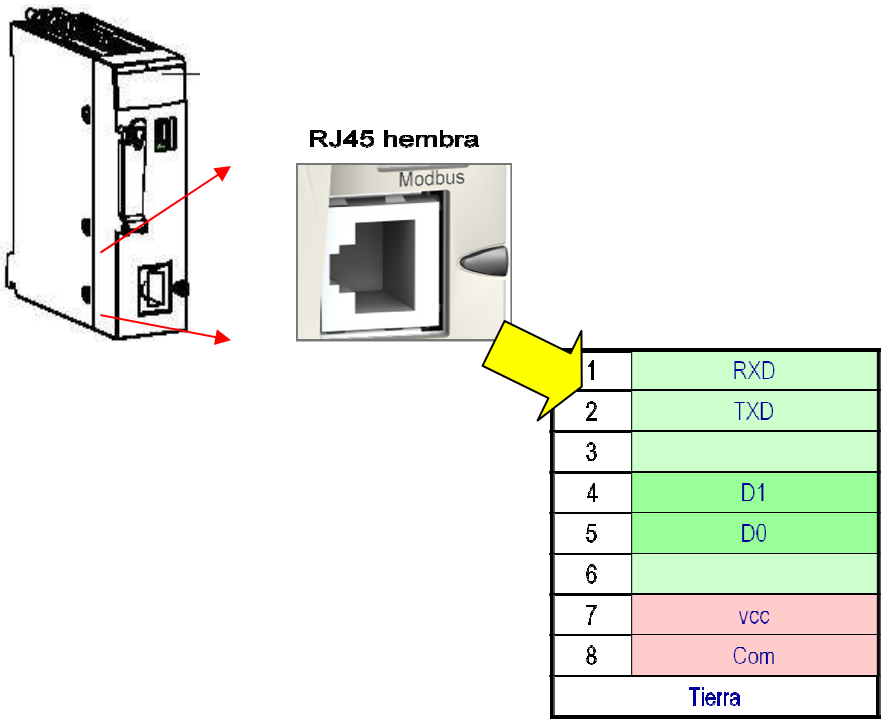
El desarrollo de una conexión *Modbus TCP/IP* no necesita ningún componente propietario y es de dominio público, no siendo necesario comprar licencias de ningún

tipo. Una de las ventajas de este protocolo es que se puede trasladar fácilmente a cualquier producto compatible con una pila de comunicación *TCP/IP* estándar.

El nivel de aplicación *Modbus* es muy sencillo y se conoce universalmente con sus nueve millones de conexiones instaladas. Un gran número de fabricantes utilizan ya este protocolo y muchos de ellos han desarrollado una conexión *Modbus TCP/IP* y numerosos productos están disponibles actualmente.

Este protocolo permite a cualquier producto de campo pequeño comunicarse en *Ethernet* sin necesidad de disponer de un potente microprocesador o de gran cantidad de memoria interna. Gracias a la gran velocidad de *Ethernet* (100 Mbits/s) se puede utilizar este tipo de red en aplicaciones de tiempo real.

Los datos que se transportan en una red Ethernet *Modbus TCP/IP* se denominan datos *CANopen* y reciben su estandarización por parte de *CiA DSP* 309-2. La especificación reserva el código de función *Modbus 43/13* a este uso.



2.7.4.4 Puerto serie RJ45 del BMX P34 1000

2.7.5 Módulo E/S (BMX AMM 0600)

2.7.5.1 Introducción

La gama M340 de Modicon incluye diferentes módulos de entradas/salidas, tanto binarias como analógicas. Los módulos analógicos son módulos de formato estándar y ocupan un solo emplazamiento en los racks, pudiéndose emplazar en cada uno de dichos emplazamientos excepto en los dos primeros (PS y 00). Estos se encuentran reservados para el módulo de alimentación del rack y el módulo del procesador respectivamente.

La alimentación de las funciones analógicas se suministra mediante el bus interno del rack (3.3 V y 24 V).

Los módulos de E/S analógicas pueden conectarse y desconectarse bajo tensión.

El número máximo de vías analógicas que pueden conectarse está limitado por el número de emplazamientos del rack.

2.7.5.2 Descripción y características técnicas

En nuestra aplicación utilizamos el módulo de entradas/salidas analógicas BMX AMM 0600, el cual consta de 4 entradas analógicas y 2 salidas analógicas no aisladas. Los rangos de tensión y corriente se muestran a continuación:

- Entradas analógicas:
    - Voltaje: +/- 10 V
    - Corriente: 0 a 20 mA y 4 a 20 mA
- Salidas analógicas:
    - Voltaje: +/-10 V, 0 a 10 V
      0 a 5 V / 0 a 20 mA
      1 a 5 V / 4 a 20 mA
      +/- 5 V / +/- 20 mA

Además de la misión de lectura y escritura de las diferentes variables, el módulo BMX AMM 0600 dispone de las siguientes funciones:

- Protección del módulo contra sobretensiones.
- Adaptación a los distintos accionadores: salida de tensión o de corriente
- Conversión de las señales digitales (10 bits o 12 bits según el rango) en señales analógicas.
- Transformación de los datos de la aplicación en datos compatibles con el convertidor analógico digital.

- Control del módulo e indicación de los fallos de la aplicación: test del convertidor, test de rebasamiento de rango, test 'WATCH P06'.

El proceso de ejecución y la descripción todas estas funciones se muestran a continuación:

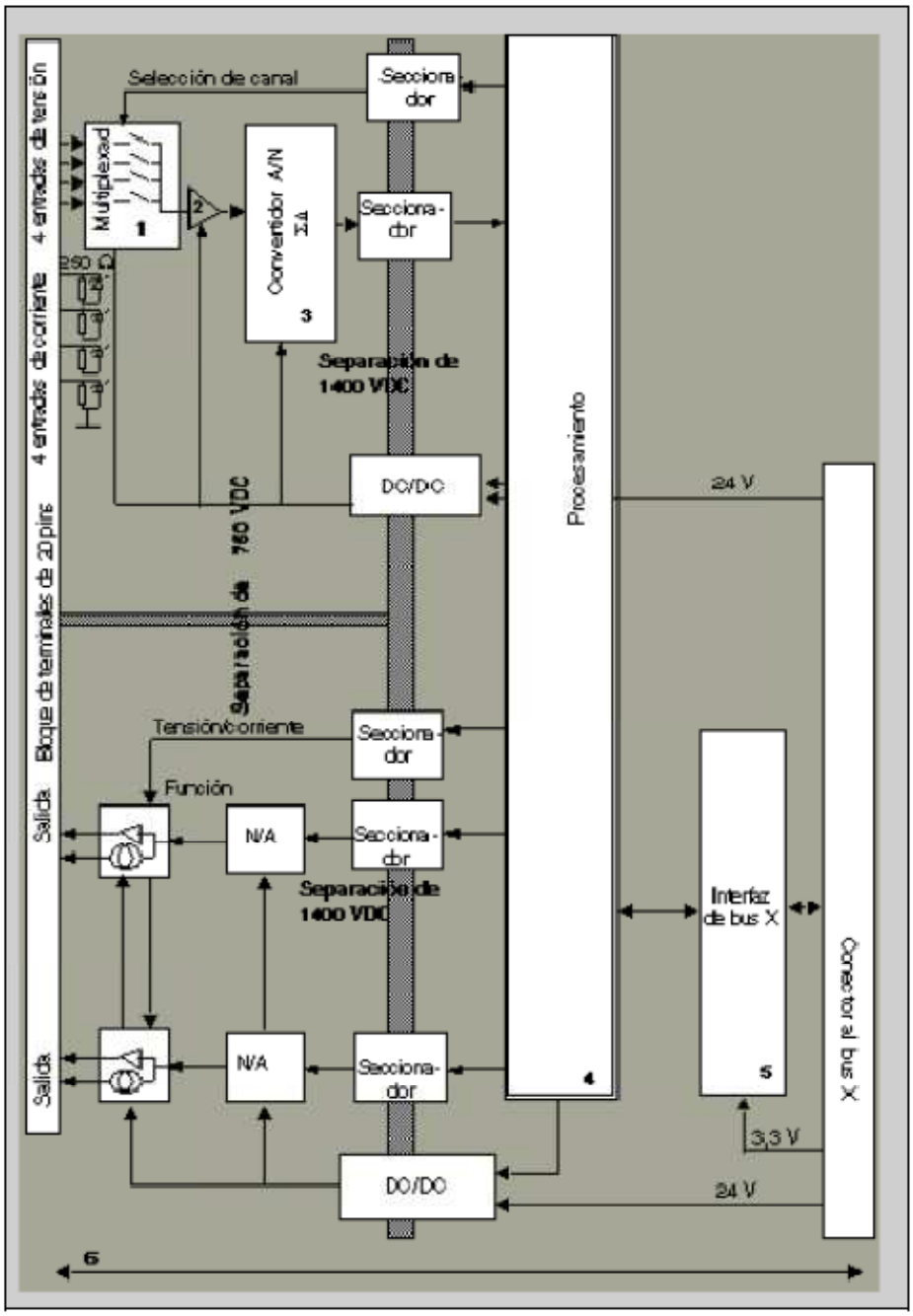


Figura 2.7.2.1 Gráfico de funciones

Dirección	Proceso	Características
1	Adaptación	- Conexión física al proceso - Protección contra sobretensiones
2	Adaptación de la señal	- Adaptación sobre tensión o corriente mediante configuración del software
3	Conversión	- Conversión en 13 bits con signo de polaridad - Encuadre automático de los datos del programa
4	Transformación de los valores de la aplicación en datos válidos para el convertidor analógico/digital	- Utilización de parámetros de calibración de equipo
5	Comunicación con la aplicación	- Gestión de los intercambios con la CPU - Direccionamiento geográfico - Recepción de los parámetros de configuración del módulo y de los canales, además de los valores numéricos de los canales
6	Supervisión del módulo e indicación de los posibles fallos de la aplicación	- Prueba de desborde de rango en los canales - Prueba de fallos externos (circuitos abiertos o cortocircuitos) - Prueba del watchdog - Capacidades de retorno programables

Tabla 2.7.5.2.1 Descripción de funciones

El módulo de entradas/salidas analógicas BMX AMM 0600 se encuentra en formato estándar (1 emplazamiento). Se presentan en forma de caja que garantiza una protección IP20 de toda la parte electrónica y se enclavan en cada emplazamiento mediante un tornillo imperdible.

El aspecto de este módulo y la descripción de cada una de sus partes se muestra a continuación:

Figura 2.7.5.1 Módulo BMX AMM 0600

Estos módulos incluyen en su parte frontal:

1. Cuerpo rígido que cumple las funciones de soporte y de protección de la tarjeta electrónica.
2. Marcado de la referencia del módulo (etiqueta que se encuentra igualmente en la parte derecha de este).
3. Bloque de visualización del estado del módulo y de las vías.
4. Conector que recibe el bornero de 20 contactos, para la conexión de los captadores o de los preaccionadores.

A continuación podemos ver las características generales del módulo BMX AMM 0600:

Tipo de vías		Entradas de alto nivel no aisladas	Salidas de alto nivel no aisladas
Número de vías		4	2
Rangos	Tensión	± 10*V*, 0...5*V*, 0...10*V*, 1...5*V*	± 10*V*
	Corriente	0...20*mA*,4...20*mA*	0...20*mA*, 4...20*mA*
Valor de conversión máximo	Tensión	± 11,25*V*	± 11,25*V*
	Corriente	0...30*mA*	0...24*mA*
Resolución		14 *bits* (± 10*V*), 12 *bits* (0...5*V*), 13 *bits* (0...10*V*), 12 *bits* (1...5*V*).	12 *bits* (0...20*mA*), 11 *bits* (4...20*mA*).
Filtrado		Filtrado numérico de 1er orden mediante firmware	-
Precisión de la resistencia interna de conversión		250 Ω, 0,2% -25 *ppm/* ˚*C*	-
Tiempo del ciclo de adquisición	Rápido	1*ms* + 1ms x n.º de vías utilizadas (adquisición periódica del nº de vías)	-
	Por defecto	5*ms* para 4 vías	-
Tiempo de recuperación		-	<2*ms*
Sobrecarga autorizada en las vías de entradas	Tensión	± 30*V*	± 11,25*V*
	Corriente	± 30*mA*	0...24*mA*
Errores de medida (1)	A 25 ˚*C*	0,25%*PE*(V), 0,6%*PE*(I)	0,25%*PE*
	Máximo a 0...60 ˚*C*	0,35%*PE*(V), 0,5%*PE*(I)	0,6%*PE*

Deriva de la temperatura		30 *ppm/ ˚C*(V) 50 *ppm/ ˚C*(I)	1000 *ppm/ ˚C*
Recalibración		Interno	Ninguna, calibrado en fábrica
Modo de secuencia (2)		-	Por defecto o configurable
Aislamientos	Entre grupo de vías de entradas y grupo de vías de salidas	1400*V*	
	Entre vías y bus	2000*V*	
	Entre vías y tierra	2000*V*	
Consumos	3,3*V*	Típico	0,54*W*
		Máximo	0,8*W*
	24*V*	Típico	1,8*W*
		Máximo	2,8*W*

(1) %PE: error en % de escala completa.
(2) Por defecto: salida a 0 (V o mA). Configurable: mantenimiento en el ultimo valor o ajuste al valor predefinido para cada vía.

Tabla 2.7.5.2 Características del módulo BMX AMM 0600

2.7.5.3 Esquema de conexiones

El conexionado del módulo con el exterior se realiza a través de un bornero desenchufable de 20 contactos que se suministra por separado cuya referencia es BMX FTB 2000.

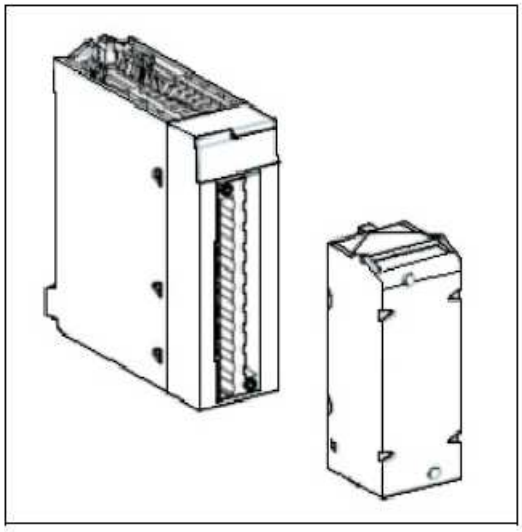


Figura 2.7.5.2 Esquema del conexionado de módulos

A continuación mostramos la conexión del bloque de terminales, los sensores y el cableado de los actuadores:

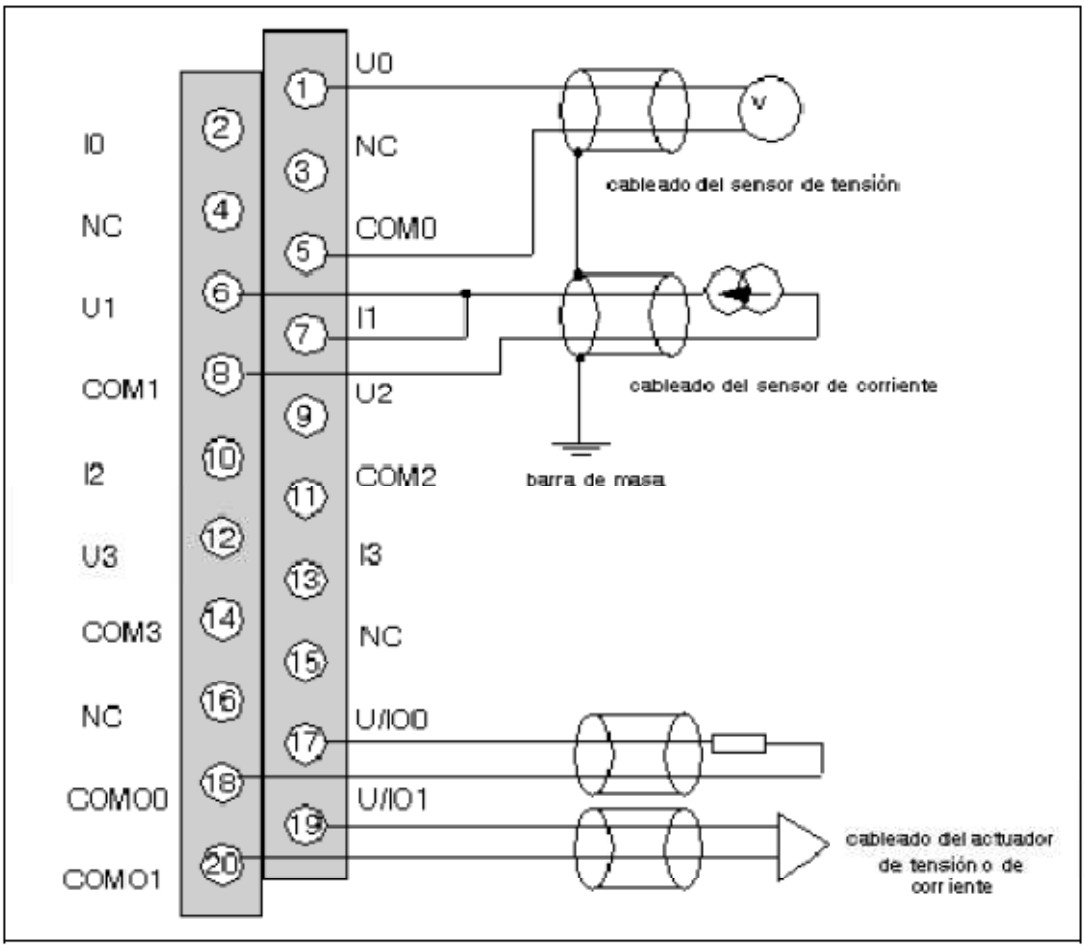


Figura 2.7.5.3 Cableado de conexión al bornero

Las entradas reciben la denominación Ux para el polo positivo y COMx para el polo negativo, correspondientes al canal x. Y las salidas se denominan U/IOx para el terminal positivo y COMOx para el terminal negativo, correspondientes también al canal x.

2.7.6 Módulo E/S (BMX DDM 16025)

2.7.6.1 Introducción

Además del módulo de entradas analógicas que acabamos de describir utilizamos el módulo de salidas binarias BMX DDM 16025.

Los módulos de entradas/salidas binarias del rango Modicon M340 son módulos de formato estándar (ocupan una sola posición) y están equipados con un bloque de terminales de 20 pins o con 1 ó 2 conectores de 40 pins. En este caso el bloque de terminales es de 20 pins.

2.7.6.2 Descripción y características técnicas

El módulo BMX DDM 16025 es un módulo binario de 24 VCC conectado a través de un bloque de terminales de 20 pins. Se trata de un módulo de lógica positiva en el cual los 8 canales de entrada reciben corriente de los sensores (común positivo). Las 8 salidas del relé funcionan con corriente alterna o con corriente continua.

El módulo se encuentra en formato estándar (1 emplazamiento). Viene de fábrica con una protección IP20 de toda la parte electrónica la cual asegura una protección contra la penetración de cuerpos sólidos con diámetro superior a 12mm.

En la siguiente imagen se muestra el aspecto del módulo descrito:

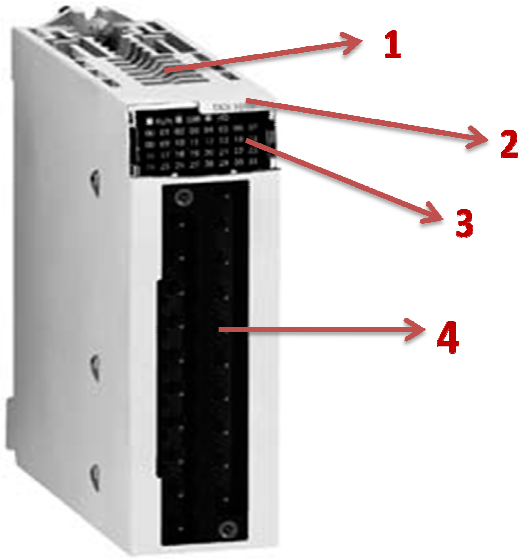


Figura 2.7.6.2.1 Aspecto general del módulo BMX DDM 16025

1. Cuerpo rígido que desempeña las funciones de soporte y protección de la tarjeta electrónica.
2. Referencia del módulo
3. Bloque de visualización del estado de las vías.
4. Conector para el bornero desenchufable de 20 contactos para la conexión de los captadores o de los preaccionadores.

En la siguiente tabla mostramos las características generales del módulo:

Tipo de vías			Ocho entradas de lógica positiva de 24 V CC	Ocho salidas de relé de 24 V CC/24–240 V CA
Valores Nominales	Tension Corriente		24 V CC 3,5 mA	
	Conmutación de tensión continua			Carga resistiva de 24 V CC
	Conmutación de corriente continua			Carga resistiva de 2 A
	Conmutación de tensión alterna			220 V CA, Cos Φ= 1
	Conmutación de corriente alterna			2 A, Cos Φ = 1
Valores límite de entrada	En 1	Tensión	≥11 V	
		Corriente	≥2 mA para U ≥11 V	
	En 0	Tensión	5 V	
		Corriente	< 1,5 mA	
	Alimentación del sensor (ondulación incluida)		De 19 a 30 V	
Impedancia de Entrada	En U nominal		6,8 KΩ	
Corriente de carga mínima	Tensión/corriente			5 V CC/1 mA.
Corriente de carga máxima	Tensión			264 V CA/125 V CC
Tiempo de respuesta	Típico		4 ms	
	Máximo		7 ms	
	Activación			≤ 8 ms
	Desactivación			≤ 10 ms

Conformidad con IEC 1131-2		Tipo 3	
Polaridad inversa		Con protección	
Fiabilidad	MTBF para funcionamiento continuo, en horas a temperatura ambiente(30º)	835 303	835 303
Rigidez dieléctrica	Primaria/secundaria	15,000 V reales a 50/60 Hz durante 1min	
	Entre grupos de entrada/salida	500 V CC	
	Tensión máxima		2.830 V CA rms/ciclos
Resistencia de aislamiento		> 10 MΩ	10 MΩ
Tipo de entrada		Corriente de común positivo	
Paralelización de las entradas		No	
Tensión del sensor: umbral de monitorización	Aceptar	> 18 V	
	Error	< 14 V	
Tensión del sensor: tiempo de respuesta de control a 24V	En la aparición	8 ms < T < 30 ms	
	En la desaparición	1 ms < T < 3 ms	
Consumo de alimentación de 3,3V	Típico	35 mA	79 mA
	Máximo	50 mA	111 mA
Consumo del preactuador de 24V	Típico	79 mA	36 mA
	Máximo	111 mA	58 mA
Potencia Disipada		3,1 W como máx.	3,1 W como máx.
Descenso de Temperatura		Ninguno	Ninguno
Fusibles	Internos	Ninguno	Ninguno
	Externos	1 fusible de acción rápida de 0,5 A por grupo de entrada	1 fusible de acción rápida de 12 A por grupo de salida

Tabla 2.7.6.2.1 Tabla características módulo BMX DDM 16025

2.7.6.3 Esquema de conexiones

El conexionado del módulo con el exterior se realiza mediante un bornero desenchufable de 20 contactos que se suministra por separado. Se fija mediante tornillos y su referencia es BMX FTB 2000.

El módulo BMX DDM 16025 tiene incorporado un bloque de terminales de 20 pines extraíble para la conexión de 8 canales de entrada y 8 canales de salida de relé aislados.

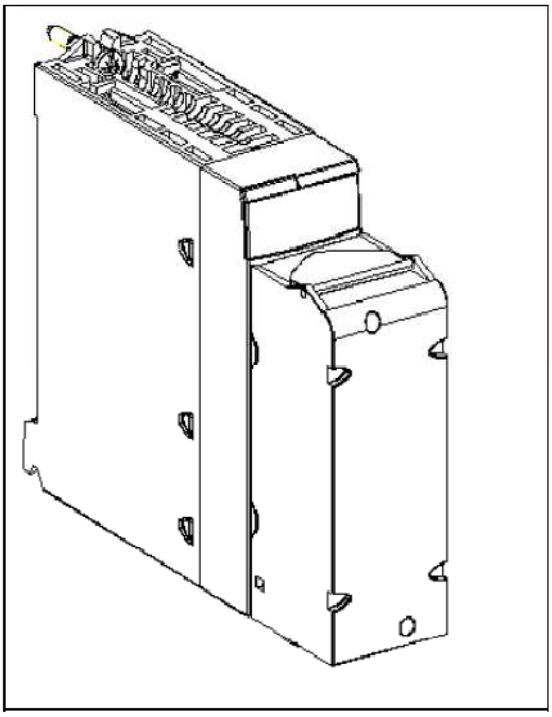


Figura 2.7.6.3.1 Esquema del bloque de terminales extraíble

En las siguientes imágenes mostramos el circuito de entrada de corriente directa (lógica positiva) y el circuito de salida de relé:

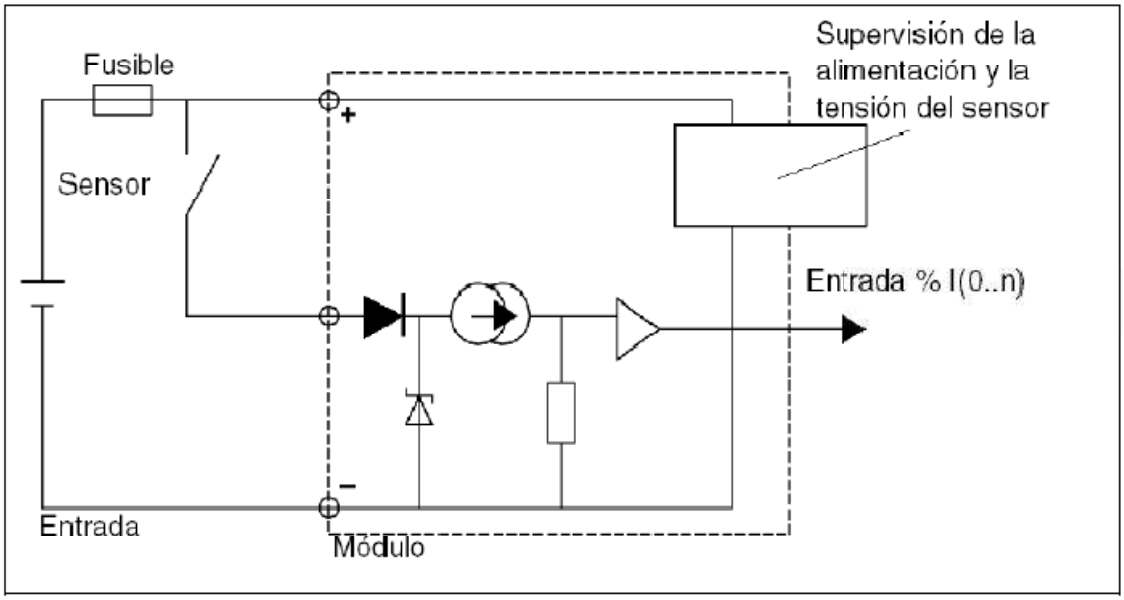


Figura 2.7.6.3.2 Diagrama del circuito de entrada

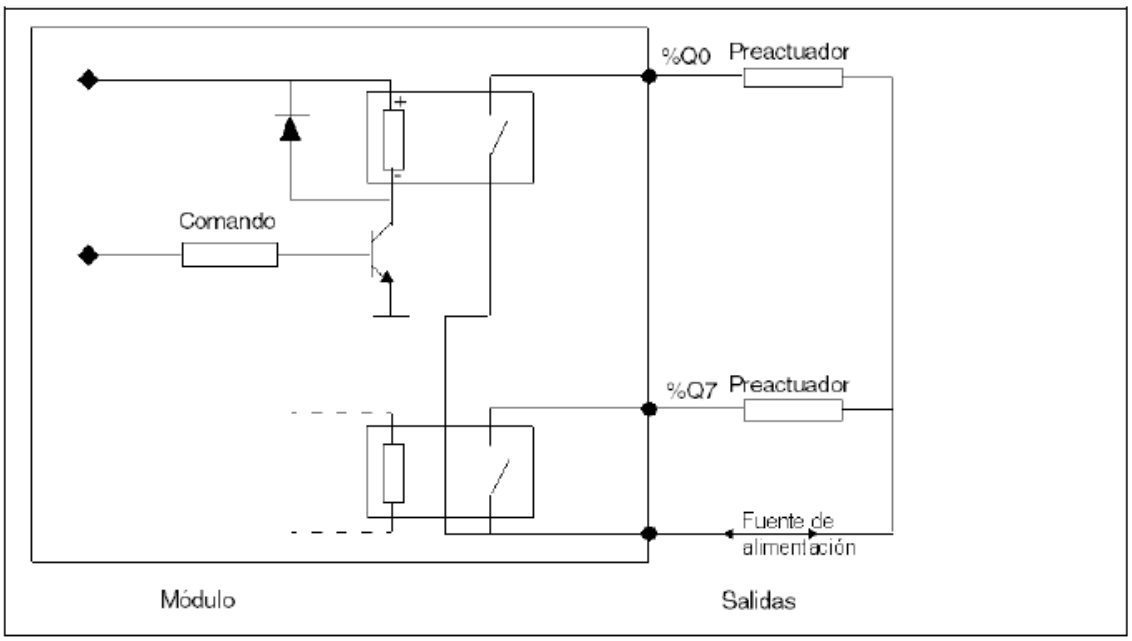


Figura 2.7.6.3.3 Diagrama del circuito de salida

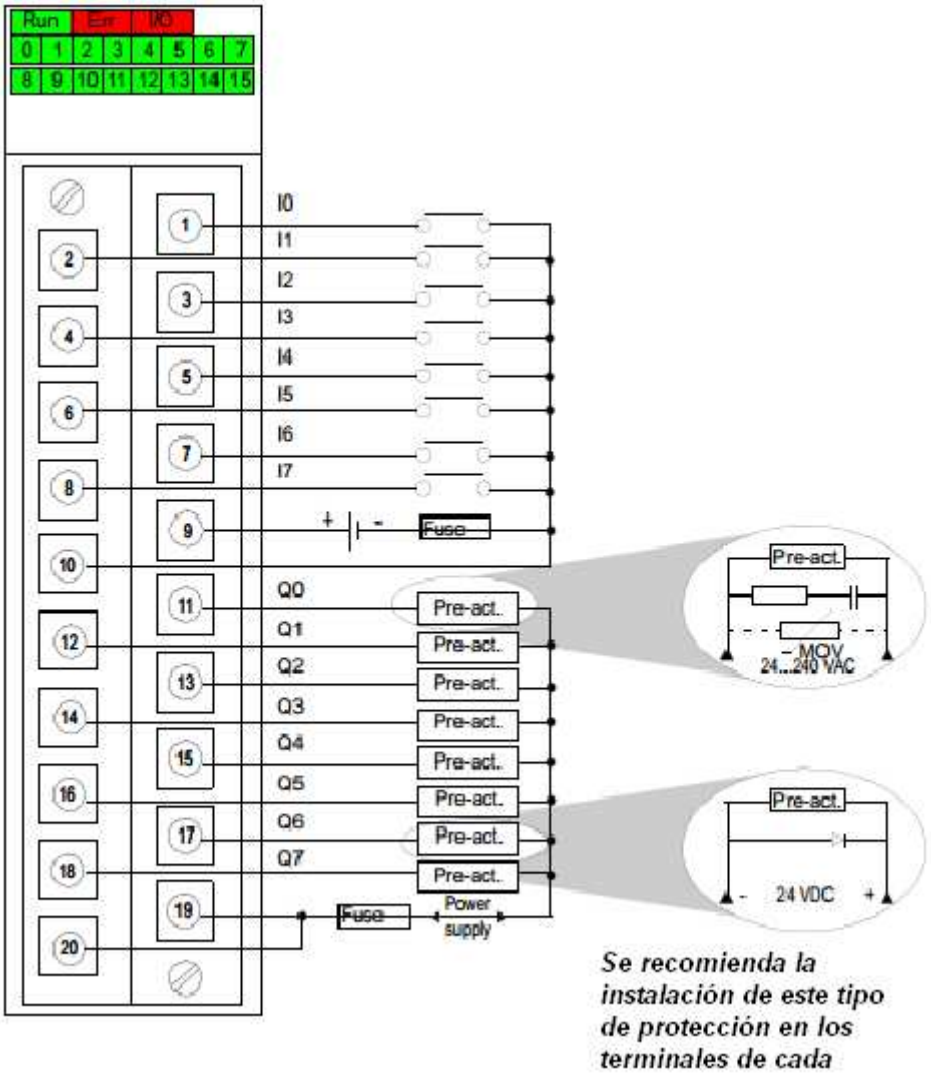


Figura 2.7.6.4 Cableado de conexión al bornero

3. SOFTWARE UTILIZADO

3.1 Introducción

El software común de programación, puesta a punto y explotación de los autómatas Modicon M340 se denomina *Unity Pro.*

El software *IEC 61131-3*, *Unity Pro* surge de la experiencia en los software *PL7* y *Concept*. Además abre las puertas de un conjunto completo de nuevas funcionalidades para obtener una mayor productividad:

- Elevado diseño funcional
- Óptima estandarización para la reutilización de los desarrollos
- Numerosas herramientas de diagnóstico de programas y mejora en la explotación de los sistemas
- Nuevos servicios de diagnóstico integrados

Unity Pro también hace posible la migración de las aplicaciones existentes. De este modo, se aumenta al máximo la inversión en software, se reducen los costes de formación y se beneficia de un potencial de evolución y de compatibilidad incomparable.

El catálogo de software de Unity propone software especializado para obtener más productividad:

- Apertura para desarrollos en lenguaje C o VBA (*Visual Basic Applications*).
- Diseño y generación de aplicaciones de forma automática.

3.2 Unity Pro

3.2.1 Introducción

Este software aprovecha las ventajas de los interfaces gráficos y contextuales de *Windows,* propone un conjunto completo de funcionalidades y herramientas que permiten calcar la estructura de la aplicación en la estructura del proceso o de la máquina.

El programa se divide en los siguientes módulos funcionales:

- Secciones de programa.
- Tablas de animación.
- Pantallas del operador.
- Hipervínculos.

El proceso de programación sigue una serie de pasos ordenados para el desarrollo correcto de la aplicación como podemos ver en el siguiente diagrama de bloques:

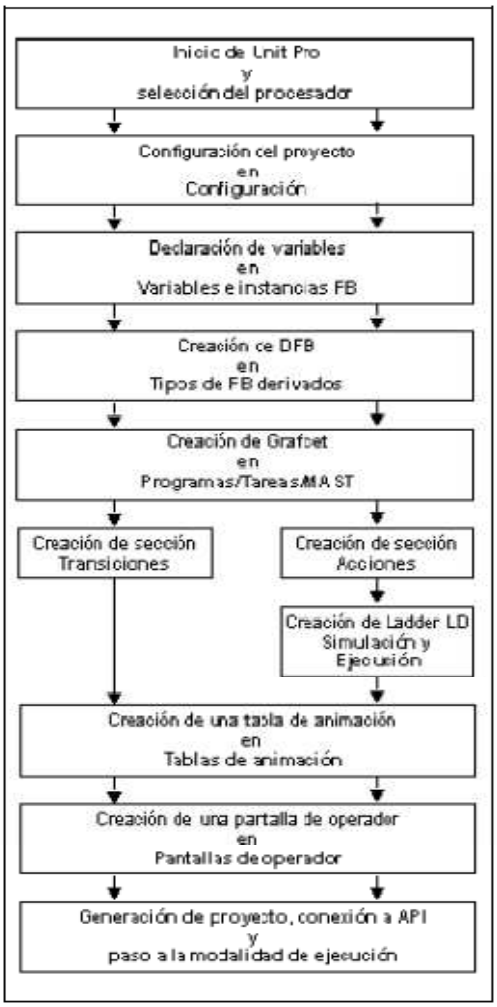


Figura 3.2.1.1 Diagrama con los pasos del proceso en Unity Pro

3.2.2 Entorno gráfico del programa

El software de *Unity Pro* ofrece acceso a todas las herramientas con el fin de aprovechar los resultados de los software de diseño de aplicaciones *Concept* y *PL7 Junior/Pro.*

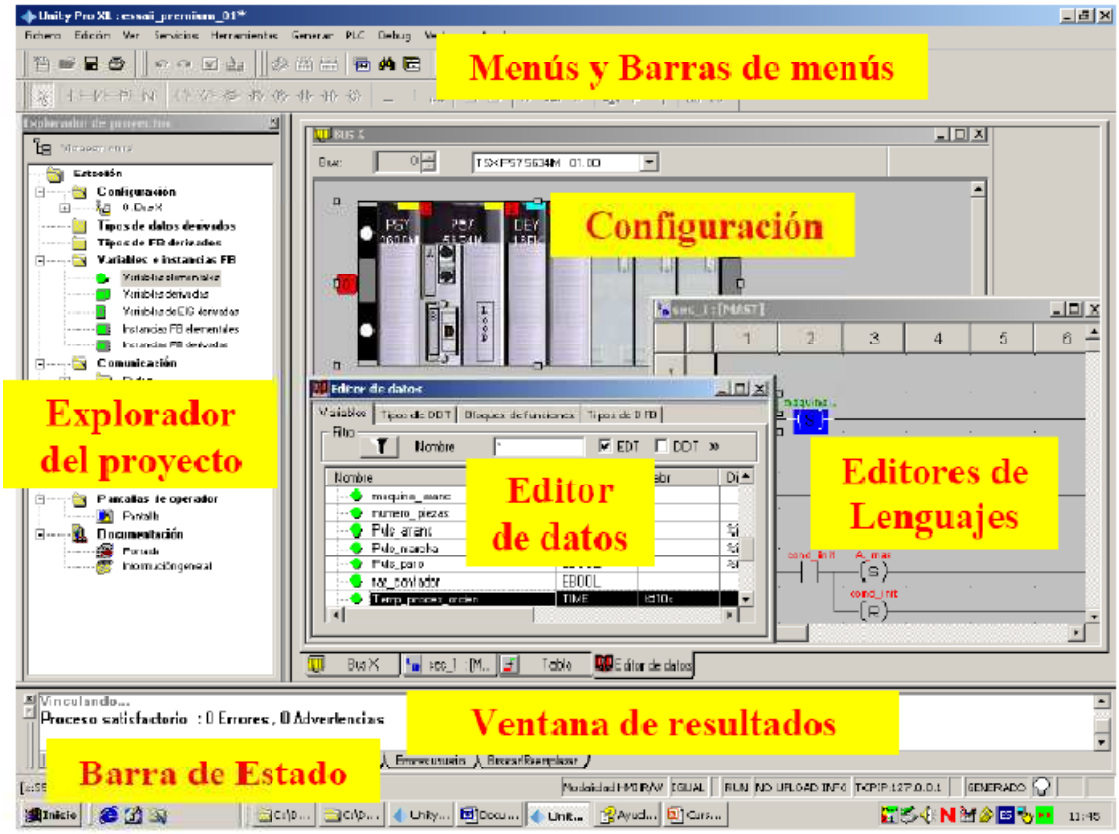


Figura 3.2.2.1 Interface de usuario

Mediante las barras de menú podemos acceder a todas las funciones de la herramienta Unity Pro y los iconos a las funciones más utilizadas. También podemos personalizar la barra de iconos haciendo clic derecho en la zona de los iconos y seleccionando los grupos de iconos que nos interesa visualizar.

El explorador de proyectos representa el árbol del proyecto y permite acceder a las diferentes partes de este. Hay dos visualizaciones disponibles:

- **Estructural**: Vista de un proyecto según el punto de vista del proyecto: configuración, editor de datos, sección de programación…
- **Funcional**: Vista del proyecto según el punto de vista de la máquina creando módulos funcionales que pueden representar las diferentes partes de la máquina.

A continuación mostramos los 2 tipos de visualizaciones:

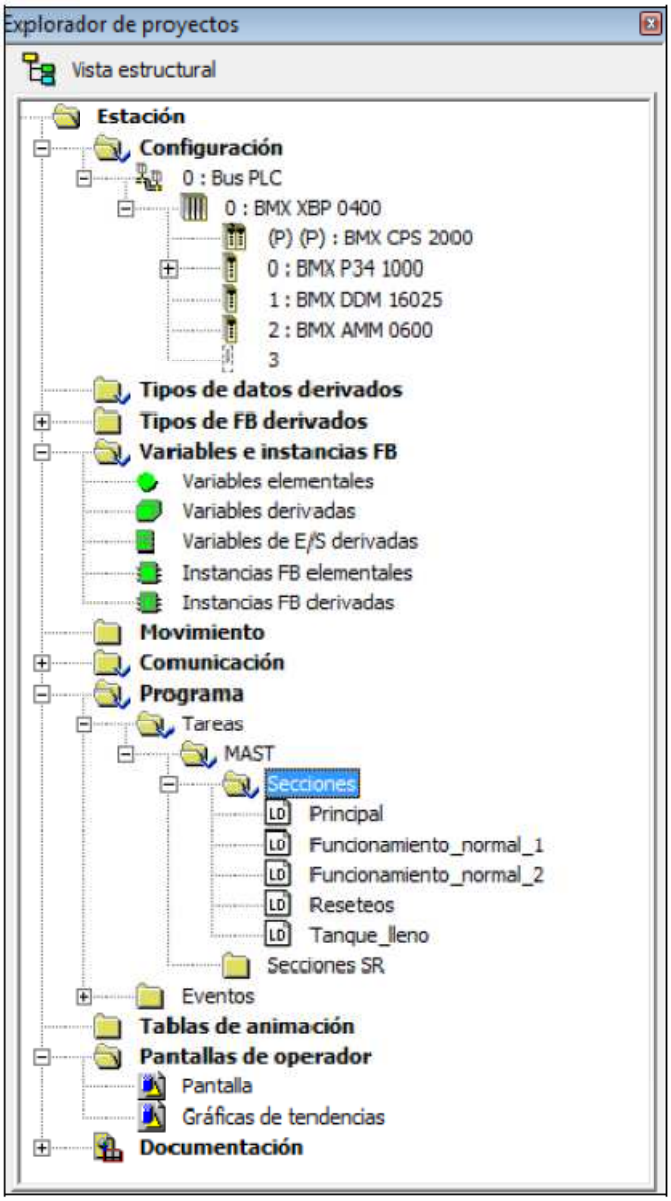


Figura 3.2.3.1 Vista estructural

Este tipo de vista permite tener una visión global del programa y acceder rápidamente al conjunto de componentes de la aplicación: editor de configuración editor de bloques de usuario (DFB) y de datos derivados (DDT),… .

Permite desde cualquier nivel del árbol tanto crear un hipervínculo para acceder a un comentario o a una descripción como crear un directorio en el que se guarden los hipervínculos para acceder a un conjunto de carpetas del usuario.

En la vista estructural, el explorador ofrece, entre otras cosas, las siguientes prestaciones:

- Crear y eliminar elementos.
- Símbolo de la sección que indica el lenguaje de programación de esta.
- Visualización de las propiedades de los elementos.
- Creación de directorios del usuario.
- Inicio de los editores.
- Inicio de la función de importación/exportación.

En cuanto a la vista funcional, esta permite estructurar la aplicación en módulos funcionales que se componen de secciones, tablas de animación y pantallas del operador.

Figura 3.2.3.2 Vista funcional

Entre las prestaciones de este tipo de vista caben destacar las siguientes:

- Creación de módulos funcionales.
- Insertar secciones, tablas de animación,… mediante la función 'arrastrar' y 'soltar' de la vista estructural.
- Creación de secciones.
- Visualización de las propiedades de los elementos.
- Inicio de los editores.
- Símbolo de la sección que muestra el lenguaje de programación de esta.

El usuario puede definir una estructura de árboles multinivel de la aplicación de automatización.

Además, en cada nivel, se pueden adjuntar secciones de programa escritas en los diferentes tipos de lenguaje que son: lista de instrucciones (IL), lenguaje de contactos (LD), bloques funcionales (FBD), texto estructurado (ST) y diagrama funcional en secuencia (SFC); tablas de animación y pantallas del operador.

Llegados a este punto cabe destacar la función del editor de configuración, que se estructura en las siguientes partes:

- El configurador de hardware que permite, de forma intuitiva y gráfica, modificar y completar la configuración de la aplicación con los siguientes elementos:

  - Racks y fuente de alimentación.
  - Tarjetas PCMCIA, de memoria o de comunicación en el procesador.
  - Módulos de entradas/salidas digitales, analógicas o específicas.

- El configurador y encargado del parametraje de los módulos de entrada/salida y específicos que permiten definir las características y los parámetros de funcionamiento de la función específica elegida, por ejemplo:

  - Valores de filtrado en digitales.
  - Rango de tensiones o de corrientes en modo analógico.
  - Valores de los umbrales en función contaje.
  - Trayectoria de los ejes en posicionamiento.
  - Calibrado de báscula en pesaje.

- El configurador y encargado de la parametrización de las redes de comunicación que permite definir la lista de las redes que están conectadas a las estación del autómata.

- El configurador de equipos *CANopen* integrado al completo en el editor de configuración.

3.2.3 Variables del programa

Los diferentes datos del proceso de automatización se manejan mediante variables definidas dentro del programa. *Unity Pro* permite el uso de los diferentes tipos de variables que nombraremos en los siguientes apartados.

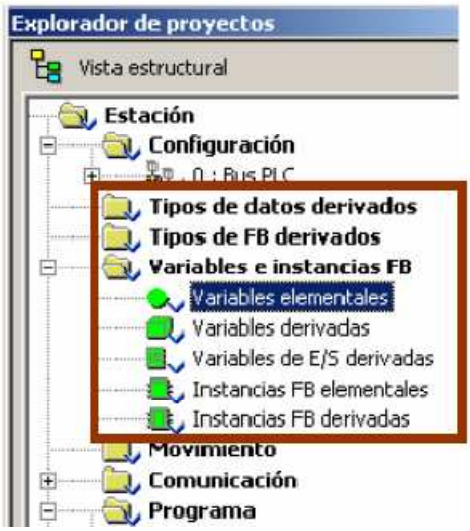


Figura 3.2.3.1 Variables del programa en el explorador de proyectos.

El editor de datos es la herramienta de creación, edición, búsqueda y todo lo que esté relacionado en si con los datos del programa.

Podemos distinguir los siguientes tipos de variables del programa:

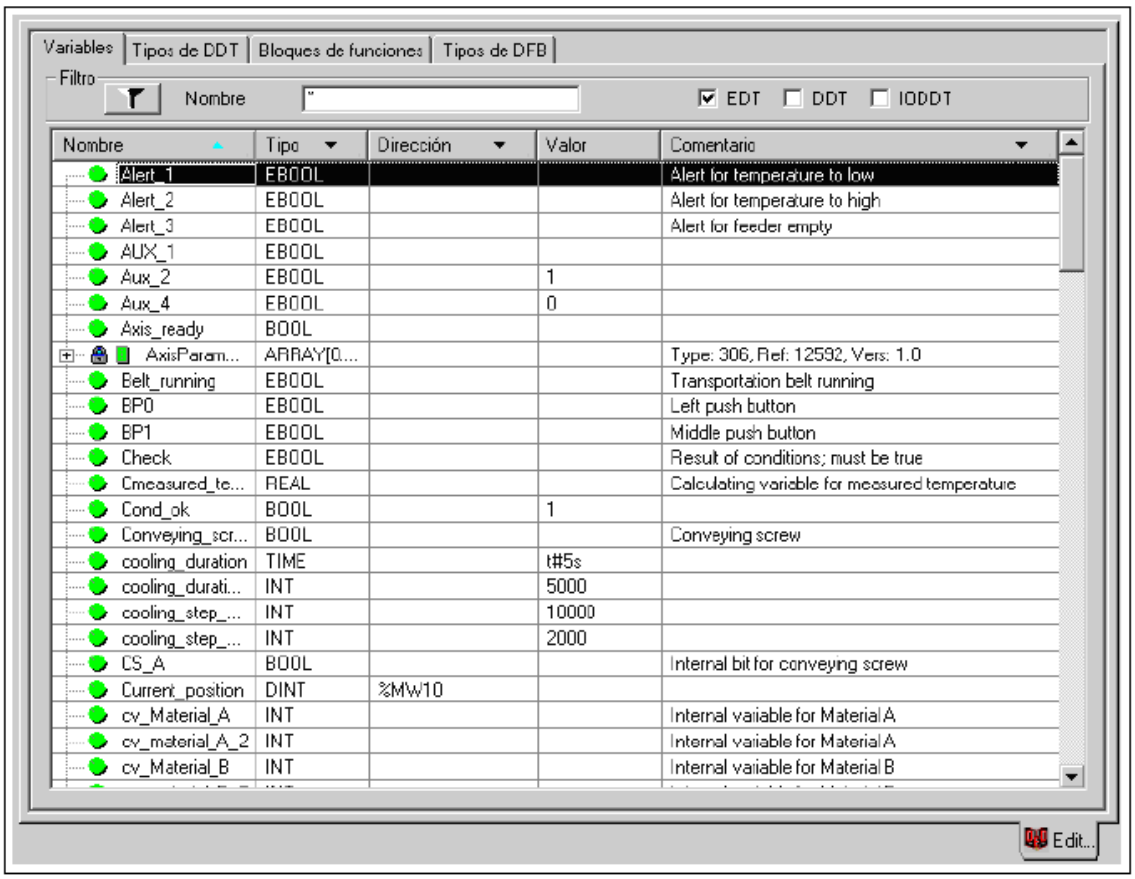
- Variables elementales, a las cuales se puede acceder mediante la pestaña '*Variables*'. Permite la creación y gestión de instancias de datos de bits, palabras, palabras dobles, entradas/salidas, tablas y estructuras.
- Variables derivadas, variables de E/S derivados, tipos de datos derivados. Se puede acceder a estas variables mediante la pestaña '*Tipos de DDT',* la cual permite la creación de tipos de datos compuestos.
- Instancias de bloques de función elementales y derivados (EFB/DFB), los cuales disponen de 2 pestañas:

    - 'Bloques de funciones' para el alta de las instancias de datos de los bloques de funciones EFB/DFB.
    - 'Tipos de DFB' para la creación de los tipos de datos de los bloques de funciones de usuario DFB.

Cada variable incluye una serie de atributos, de los cuales el nombre y el tipo son obligatorios mientras que la dirección física en la memoria, el comentario y el valor inicial son opcionales.

Se puede acceder a este editor de datos en cualquier momento durante la programación para crear o modificar datos.

En la siguiente imagen aparece la ventana con las pestañas correspondientes a los diferentes tipos de variables:



Nombre	Tipo	Dirección	Valor	Comentario
Alert_1	EBOOL			Alert for temperature to low
Alert_2	EBOOL			Alert for temperature to high
Alert_3	EBOOL			Alert for feeder empty
AUX_1	EBOOL			
Aux_2	EBOOL		1	
Aux_4	EBOOL		0	
Axis_ready	BOOL			
AxisParam...	ARRAY[0....			Type: 306, Ref: 12592, Vers: 1.0
Belt_running	EBOOL			Transportation belt running
BP0	EBOOL			Left push button
BP1	EBOOL			Middle push button
Check	EBOOL			Result of conditions; must be true
Cmeasured_te...	REAL			Calculating variable for measured temperature
Cond_ok	BOOL		1	
Conveying_scr...	BOOL			Conveying screw
cooling_duration	TIME		t#5s	
cooling_durati...	INT		5000	
cooling_step_...	INT		10000	
cooling_step_...	INT		2000	
CS_A	BOOL			Internal bit for conveying screw
Current_postion	DINT	%MW10		
cv_Material_A	INT			Internal vaiable for Material A
cv_material_A_2	INT			Internal vaiable for Material A
cv_Material_B	INT			Internal vaiable for Material B

Figura 3.2.3.2 Extracto del editor de datos

Unity Pro incluye un gestor de bibliotecas de funciones y de bloques de funciones que agrupa todos los elementos proporcionados por el programa. Esta información se almacena en bibliotecas que incluyen a su vez familias. Según el tipo de autómata seleccionado y el modelo del procesador, el usuario dispone de un subconjunto de estas librerías para escribir sus aplicaciones (Véase figura 3.2.3.3).

Dispone de las siguientes librerías: *Base Lib, Communication, CONT_CTL, Custom Lib, Diagnostics, I/O Management, MotionfunctionBlock, Motion, Obsolete Lib, Safety y System*.

La más importante de las librerías enumeradas es la de *Base Lib* ya que incluye un conjunto de bloques y funciones cuya compatibilidad es independiente de la plataforma. Esta librería incluye las siguientes familias:

- Temporizadores y contadores.

- Regulación de procesos.
- Gestión de tablas.
- Comparación.
- Gestión del tiempo, la fecha y la hora.
- Tratamiento lógico.
- Tratamiento matemático.
- Tratamiento estático.
- Tratamiento en cadenas de caracteres.
- Conversión de tipos de datos.

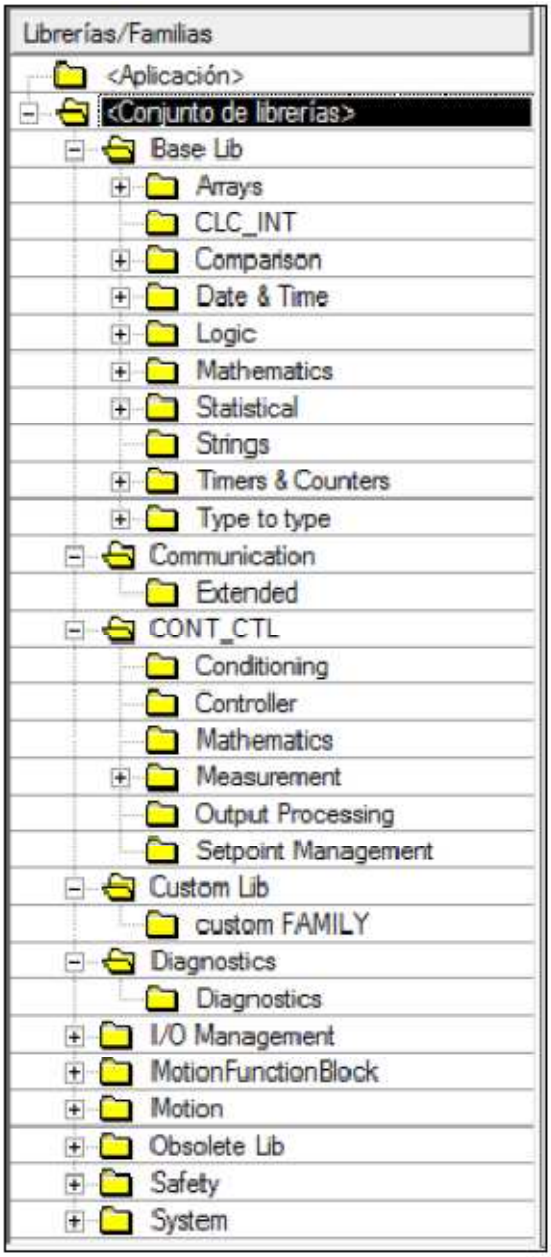


Figura 3.2.3.3 Bibliotecas de los bloques de funciones

3.2.4 Programación de aplicaciones

El software de *Unity Pro* se basa en un modelo de programación estructurada, modular y portátil.

Las plataformas *Modicon M340* instalada con el software *Unity Pro* ofrecen 2 tipos de estructura de aplicación:

- Monotarea: es la estructura simplificada ofrecida de forma predeterminada en la que sólo se ejecuta la tarea maestra.
- Multitarea: Esta estructura se compone de una tarea maestra, de una tarea rápida, de tareas periódicas y de tareas por suceso prioritarias. Es más adecuada para aplicaciones en tiempo real de alto avanzado.

Las tareas enumeradas en el anterior párrafo se componen de secciones y subprogramas. Estas secciones y subprogramas se pueden programar en los siguientes lenguajes: literal estructurado (ST), lista de instrucciones (IL), lenguaje de contactos (LD) y bloques funcionales (FBD). El lenguaje de diagrama funcional en secuencia (SFC) o *Grafcet* está reservado a las secciones de la tarea maestra.

Las denominadas secciones son entidades autónomas de programación en las que se crea la lógica del proyecto. Se ejecutan en el mismo orden en el que se representan en el explorador de proyectos. La división por secciones permite crear un programa estructurado y generar o añadir fácilmente módulos al programa.

En cuanto a los subprogramas, estos se crean como unidades independientes en secciones de subrutina. Su llamada se realiza desde las secciones o desde otra subrutina. Se puede acceder a ellos desde cualquier sección de la tarea a la que pertenecen o desde otros subprogramas de la misma tarea.

A continuación podemos ver las diferentes secciones y subrutinas de la aplicación desarrollada.

Figura 3.2.4.1 Secciones del programa

Los cinco lenguajes de programación se pueden clasificar en dos grupos:

- Lenguajes gráficos:
  - Diagrama de contactos (LD)
  - Diagrama de bloques funcionales (DFB)

- Lenguajes textuales:
  - Lista de instrucciones (IL).
  - Texto estructurado (ST).

El lenguaje de diagrama funcional en secuencia (SFC) es a menudo categorizado como un lenguaje IEC 1131-3 pero, en realidad, es una estructura organizacional que coordina los cuatro lenguajes nombrados anteriormente.

Para estos cinco lenguajes, la utilización del juego de instrucciones básicas conforme a la norma *IEC 61131-3* permite crear aplicaciones portátiles de una plataforma a otra. Además, el software Unity Pro aporta ampliaciones a este juego de instrucciones básicas. Estas extensiones específicas de los autómatas *Modicon M340 Atrium / Premium* y *Quantum* permiten desarrollar aplicaciones más complejas y aprovechar las características específicas de cada una de las plataformas.

En primer lugar vamos a describir brevemente el lenguaje de contactos (LD). Este se caracteriza porque cada sección o subprograma se compone de una serie de redes de contactos que el autómata ejecuta de forma secuencial. Las secciones LD tienen una rejilla de fondo que divide la sección en filas y columnas.

Este lenguaje está basado en celdas, es decir, en cada celda se puede colocar un único objeto. Dichos objetos pueden ser contactos, enlaces, bobinas, bloques de operaciones, bloques de funciones (EF/DFB/EFB), saltos, llamada de subprograma,….

La mayor parte de nuestro programa se escribe en este lenguaje.

En cuanto al lenguaje de bloques funcionales (FBD) podemos decir que es un lenguaje gráfico construido a base de bloques de funciones asociados a variables o parámetros y organizados entre sí mediante enlaces. En la siguiente imagen se muestra un ejemplo:

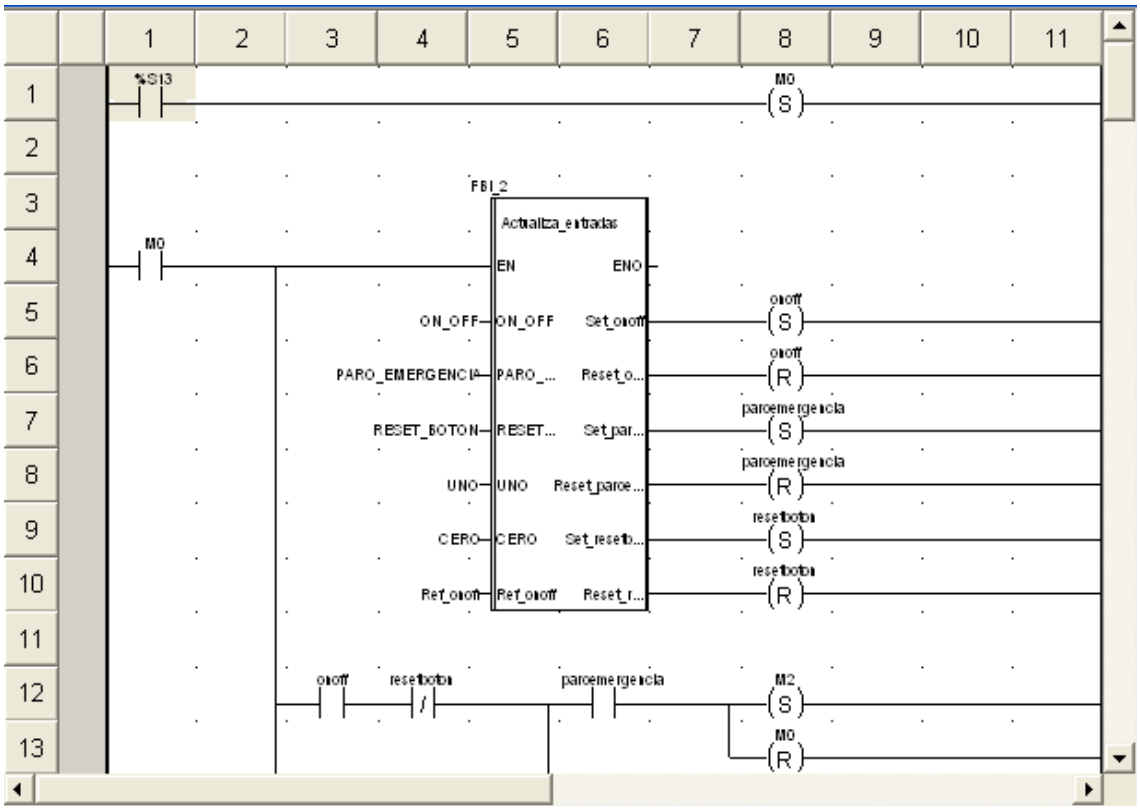


Figura 3.2.4.2 Ejemplo de programación en lenguaje de bloques funcionales

El lenguaje de diagrama funcional en secuencia (*SFC*) o *Grafcet* permite describir de forma sencilla y gráfica la parte secuencial de un automatismo a partir de etapas y transiciones. El lenguaje *SFC* presenta con respecto al lenguaje *Grafcet* una diferencia en cuanto a la ejecución de los gráficos:

- El lenguaje *SFC* sólo permite un único testigo en un mismo gráfico.
- El lenguaje *Grafcet* permite tener varios testigos en un mismo gráfico.

El software Unity Pro presenta un editor único para estos dos lenguajes con la posibilidad de definir el comportamiento con las características de la aplicación (*menú Tools / Project Settings / Language extensions*). Este lenguaje no le utilizaremos en el programa pero, aún así, mostraremos un ejemplo:

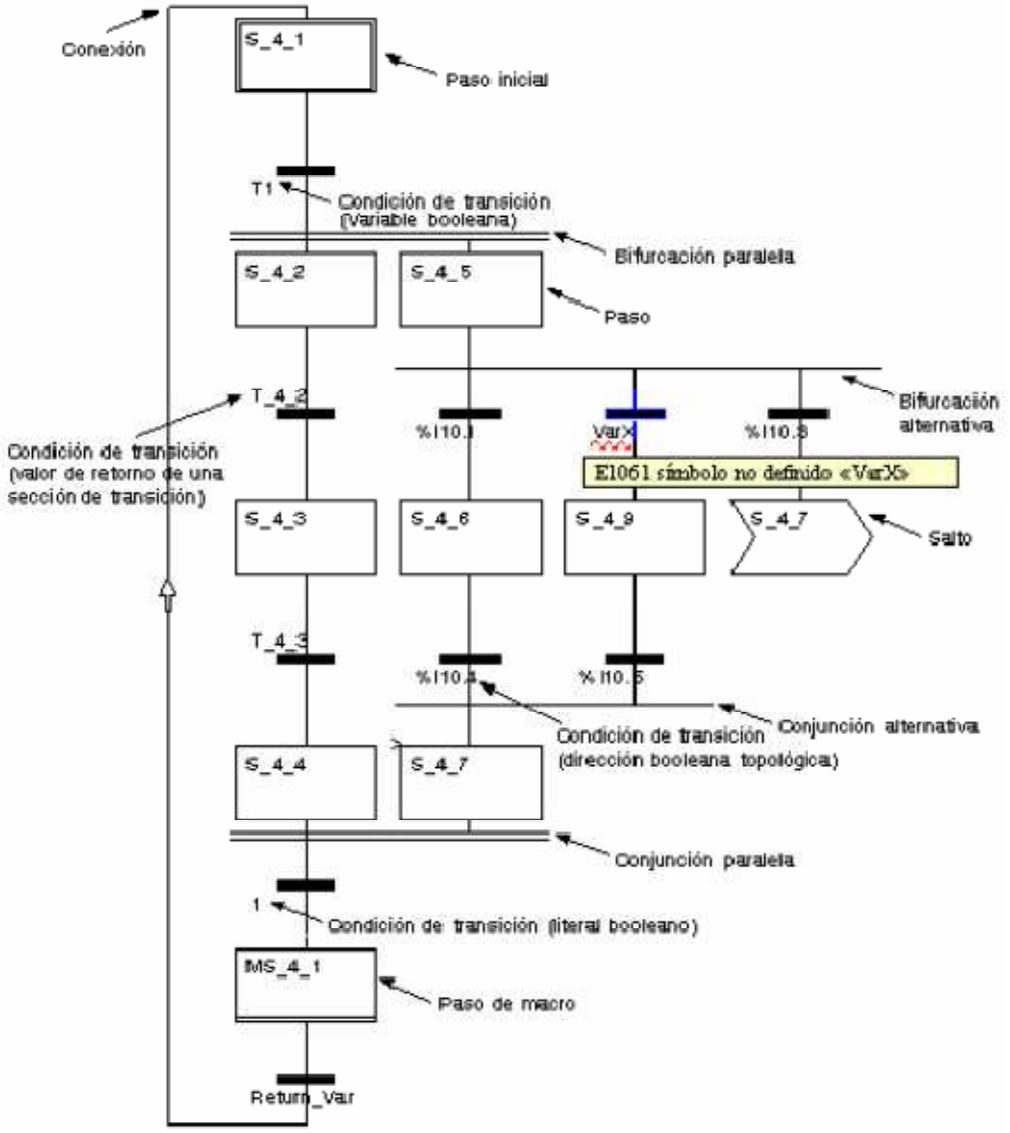


Figura 3.2.4.3 Ejemplo de programación en SFC

El lenguaje de texto estructurado (ST) es un lenguaje de alto nivel que permite la programación estructurada, lo que significa que muchas tareas complejas pueden ser divididas en unidades más pequeñas. En este lenguaje se pueden usar subrutinas para llevar a cabo diferentes partes de las funciones de control y paso de parámetros y valores entre las diferentes secciones del programa.

Incluye estructuras de cálculo repetitivo y condicional, tales como: FOR ... TO; REPEAT..... UNTIL X; WHILE X... ; IF ... THEN ...ELSE. Además soporta operaciones Booleanas (AND, OR, etc.) y una variedad de datos específicos, tales como fecha, hora.

Los operandos manipulados en las expresiones son de tipo variables de bits, variables de palabras o variables relativas a los bloques de funciones. La lectura es más

fácil gracias a los colores utilizados para diferenciar los objetos, las palabras clave y los comentarios de programa.

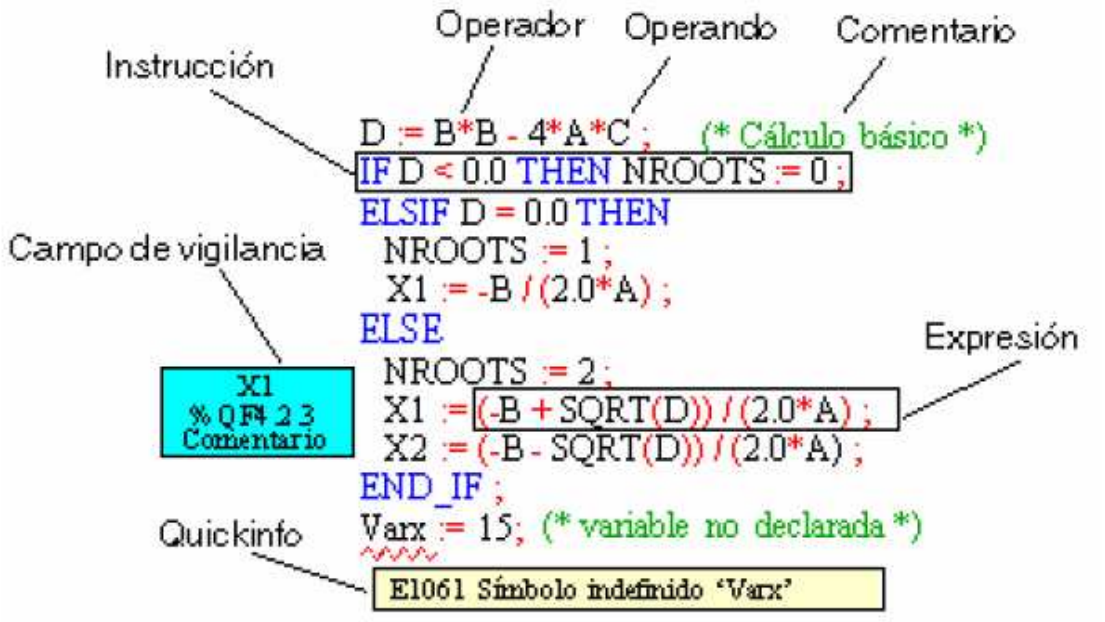


Figura 3.2.4.4 Ejemplo de programación en lenguaje literal estructurado (ST)

El lenguaje lista de instrucciones (*IL*) representa en forma de texto el equivalente a un esquema de relés. Permite escribir ecuaciones booleanas y aritméticas y utilizar todas las funciones disponibles del lenguaje Unity Pro (recuperación de funciones y bloques funcionales, asignación de variables, realización de saltos de programa, conexión con subprogramas en el interior de una sección de programa…).

Un programa en lenguaje lista de instrucciones se compone de una serie de instrucciones clasificadas según las siguientes familias diferentes:

- Instrucciones sobre bit.
- Instrucciones en bloque de función, por ejemplo, recuperación de un temporizador.
- Instrucciones numéricas sobre entero de formato sencillo, doble o flotante, por ejemplo, hacer una suma.
- Instrucciones sobre tablas de palabras, cadenas de caracteres, por ejemplo, asignar.
- Instrucciones sobre programa, por ejemplo, llamar a un subprograma.

Los operandos manipulados en las expresiones son de tipo variables de bits, variables de palabras o variables relativas a los bloques de funciones.

Este lenguaje tampoco será utilizado en nuestra aplicación.

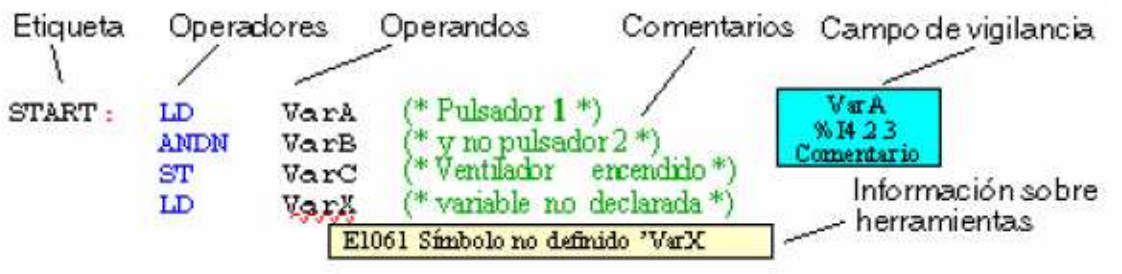


Figura 3.2.4.5 Ejemplo de programación en lista de instrucciones (IL)

3.2.5 Simulador del PLC

El simulador integrado en el software *Unity Pro* permite, a partir del terminal del ordenador, probar el programa de la aplicación sin utilizar ninguna conexión con el autómata, reproduciendo fielmente el comportamiento del programa en el PC.

Las funciones que ofrecen las herramientas de puesta a punto están disponibles para las tareas maestras, rápidas y auxiliares. La única diferencia con un PLC real consiste en la ausencia de E/S y redes de comunicaciones.

Puesto que el simulador no gestiona las E/S del autómata, las tablas de animación permiten el forzado a '0' ó a '1' del estado de las entradas.

El simulador se puede conectar a otras aplicaciones a través de un servidor OPC con el software OFS (OPC Factory Server).

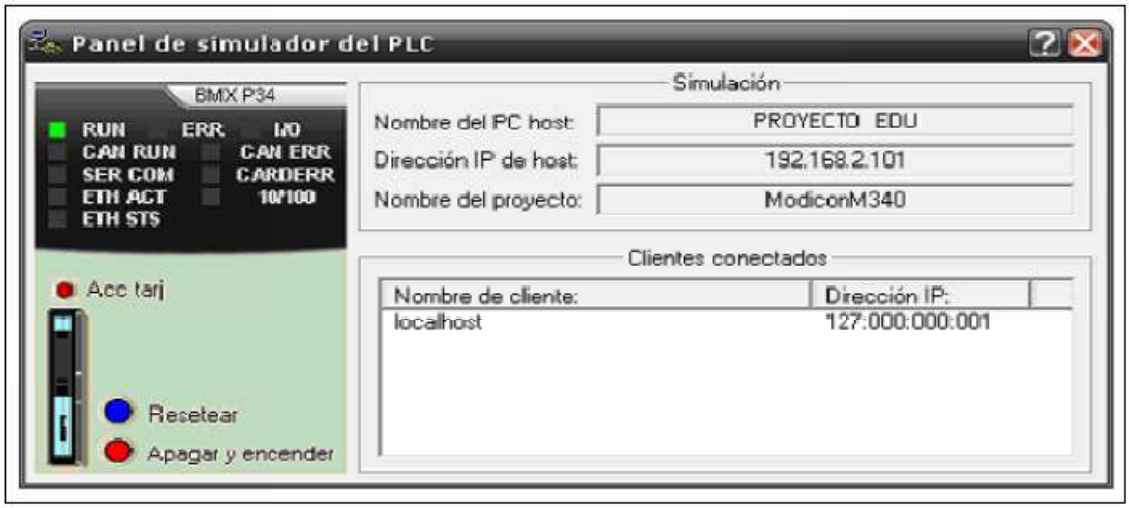


Figura 3.2.5.1 Panel del simulador

Una de sus desventajas, es que este simulador no se puede utilizar en versiones modernas de *Windows* como pueden ser *Windows 7* y *Windows Vista*.

3.2.6 Depuración de programas

El software Unity Pro ofrece un conjunto de herramientas completo para la puesta a punto de las aplicaciones Modicon M340. Una paleta de herramientas permite acceder directamente a las funciones principales, las cuales vemos a continuación.

3.2.6.1 Animación dinámica

La animación dinámica se gestiona sección a sección. Un botón de la barra de herramientas permite activar o desactivar la animación de cada sección.

Cuando el autómata está en ejecución permite visualizar simultáneamente tanto la animación de una parte del programa, independientemente del lenguaje utilizado, como la ventana de variables que contiene los objetos de la aplicación creada automáticamente a partir de la sección visualizada.

Es posible visualizar y animar simultáneamente varias ventanas mediante la función '*Visualizar valor'*, la cual permite ver al mismo tiempo una variable y su contenido cuando el ratón selecciona este objeto.

Existen dos tipos de animación:

- Estándar: las variables de la sección activa se actualizan al final de la tarea maestra (MAST).
- Sincronizada: el punto de visualización permite sincronizar la visualización de las variables animadas con un elemento de programa con el fin de conocer su valor en ese punto preciso del programa.

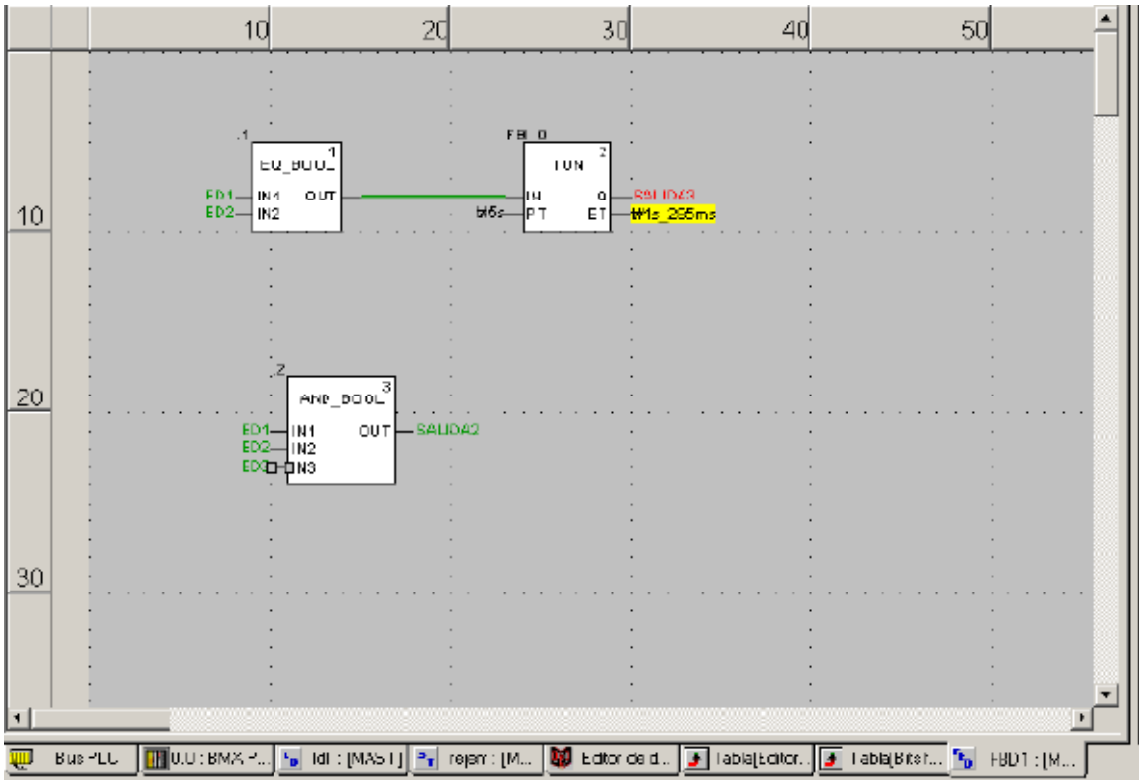


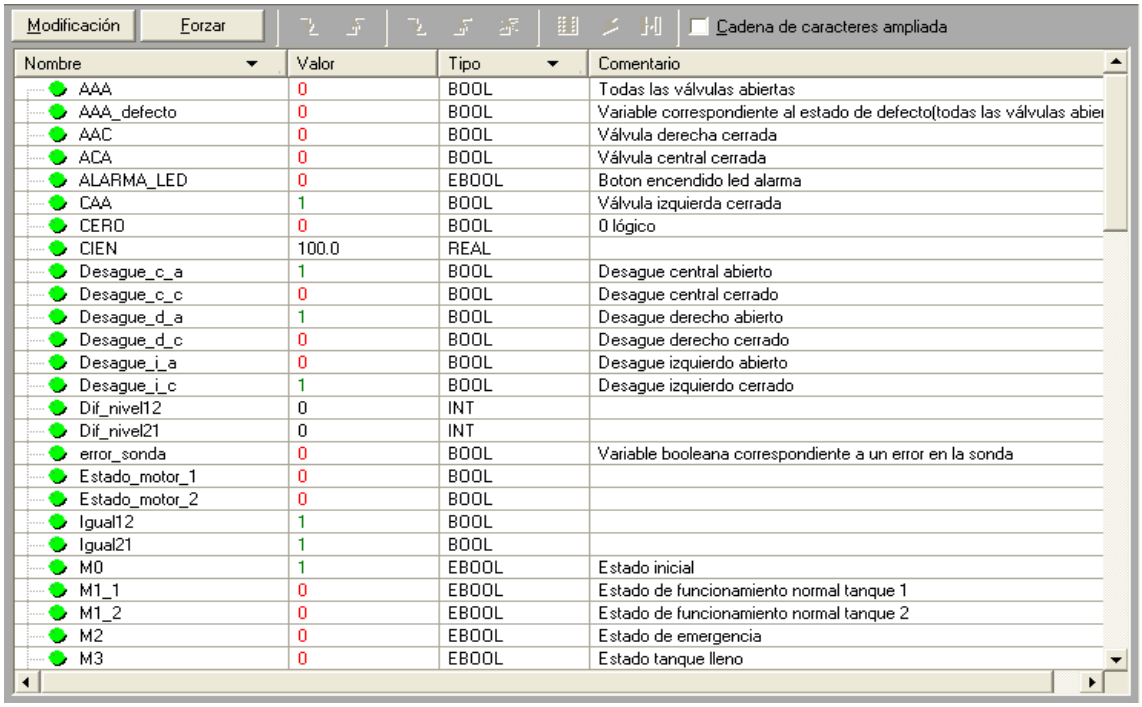
Figura 3.2.6.1.1 Animación de una sección

3.2.6.2 Tablas de animación

Estas tablas contienen las variables de la aplicación que se va a supervisar o a modificar mediante la introducción de datos o bien se pueden iniciar automáticamente a partir de la parte del programa seleccionada.

Además de la animación de datos, se pueden realizar otra serie de acciones:

- Modificar y forzar a 0 o a 1 las variables de bits.
- Cambiar el formato de visualización.
- Copiar y desplazar variables.
- Realizar búsquedas por referencias cruzadas.
- Visualizar la lista de los bits forzados.



Modificación | Forzar | Cadena de caracteres ampliada

Nombre	Valor	Tipo	Comentario
AAA	0	BOOL	Todas las válvulas abiertas
AAA_defecto	0	BOOL	Variable correspondiente al estado de defecto(todas las válvulas abier
AAC	0	BOOL	Válvula derecha cerrada
ACA	0	BOOL	Válvula central cerrada
ALARMA_LED	0	EBOOL	Boton encendido led alarma
CAA	1	BOOL	Válvula izquierda cerrada
CERO	0	BOOL	0 lógico
CIEN	100.0	REAL	
Desague_c_a	1	BOOL	Desague central abierto
Desague_c_c	0	BOOL	Desague central cerrado
Desague_d_a	1	BOOL	Desague derecho abierto
Desague_d_c	0	BOOL	Desague derecho cerrado
Desague_i_a	0	BOOL	Desague izquierdo abierto
Desague_i_c	1	BOOL	Desague izquierdo cerrado
Dif_nivel12	0	INT	
Dif_nivel21	0	INT	
error_sonda	0	BOOL	Variable booleana correspondiente a un error en la sonda
Estado_motor_1	0	BOOL	
Estado_motor_2	0	BOOL	
Igual12	1	BOOL	
Igual21	1	BOOL	
M0	1	EBOOL	Estado inicial
M1_1	0	EBOOL	Estado de funcionamiento normal tanque 1
M1_2	0	EBOOL	Estado de funcionamiento normal tanque 2
M2	0	EBOOL	Estado de emergencia
M3	0	EBOOL	Estado tanque lleno

Figura 3.2.6.2.1 Tabla de animación de la aplicación

3.2.6.3 Pantallas del operador

La herramienta pantalla del operador, también denominadas pantallas de explotación están integrada en el software Unity Pro. Estas pantallas están destinadas a facilitar la utilización de los procesos automatizados durante su puesta a punto, su arranque y su mantenimiento.

Las pantallas del operador se componen de un conjunto de información y permiten realizar acciones sencillas y rápidas en la modificación y la supervisión dinámica de las variables del autómata.

El editor de las pantallas de explotación proporciona todos los elementos de tipo *IHM* (*Interface Hombre / Máquina*) necesarios para diseñar y visualizar de forma animada los procesos. Permite diseñar estas pantallas utilizando las siguientes herramientas:

- Pantalla: creación de las pantallas de explotación que se pueden agrupar por familia.
- Mensaje: creación de los mensajes que se van a visualizar.
- Objetos: creación de una biblioteca de objetos gráficos a partir de:
    - Elementos geométricos (línea, rectángulo, elipse, incorporación de imágenes, partes frontales de regulador…).
    - Elementos de control (botones, campos de introducción de datos, navegación por pantallas…).
    - Elementos de animación (colores, parpadeo, gráfico de barras…).

Cuando el dispositivo equipado con el software Unity Pro está conectado al autómata, el usuario puede visualizar las pantallas de forma dinámica en función del estado del proceso. Las pantallas pueden visualizarse sucesivamente según la prioridad que se atribuya a cada una de ellas, desde el teclado o por petición del autómata.

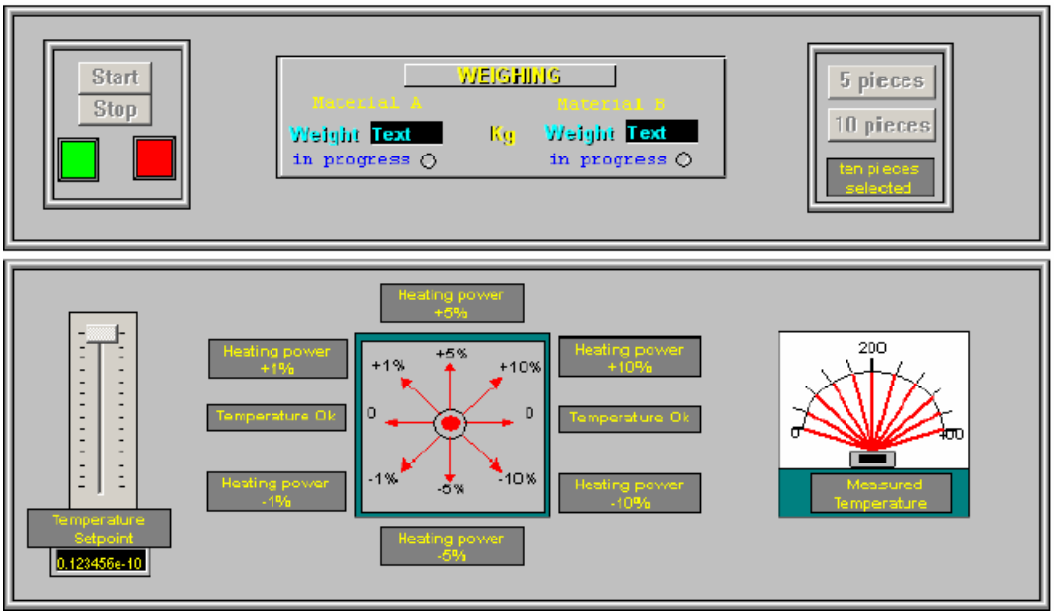


Figura 3.2.6.3.1 Ejemplo pantalla del operador

4. IDENTIFICACIÓN DEL SISTEMA

4.1 Modelado del sistema de tanques

El modelo matemático de un sistema dinámico se define como un conjunto de ecuaciones que representan la dinámica del sistema con precisión. Un sistema puede representarse de muchas formas diferentes, por lo que puede tener muchos modelos matemáticos, dependiendo del sistema en sí y de las circunstancias específicas un modelo puede ser más conveniente que otros.

Al obtener un modelo matemático se debe establecer un compromiso entre la simplicidad del mismo y la precisión de los resultados del análisis. El fin del modelado del sistema es la obtención de la función de transferencia del mismo.

Estamos ante dos tanques idénticos de descarga por gravedad mediante sendos orificios de desagüe de la misma sección, a través de los cuales el fluido descargado se almacena en un depósito inferior, del cual a su vez toman las bombas el fluido necesario que suministrar a los depósitos para su rellenado. Además los dos tanques están comunicados entre sí mediante una tubería sitiada en la parte inferior de ambos tanques como puede apreciarse en la siguiente figura.

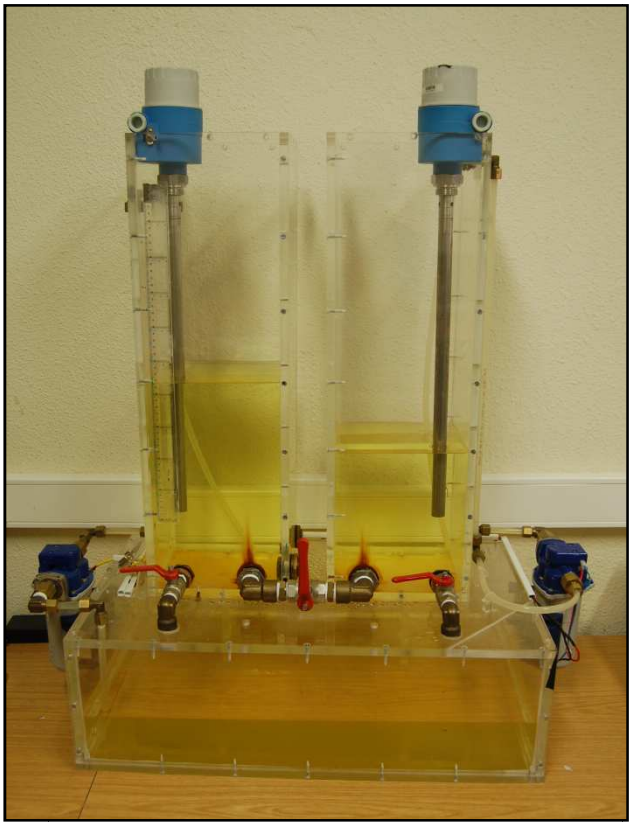


Figura 4.1.1 Sistema de tanques

Los caudales que circulan a través de este sistema de tanques son los siguientes:

Q_1, Q_2: Caudales de entrada de cada tanque suministrados por su bomba correspondiente

Q_{10}, Q_{20}: Caudales de descarga por gravedad a través de los orificios de desagüe de los tanques 1 y 2 respectivamente.

Q_{12}: Flujo que circula entre los dos tanques a través de la tubería que les comunica. Este flujo dependerá en cada instante de la altura del líquido en cada tanque.

En este tipo de sistemas se plantea la ecuación de continuidad que dice que la variación de la masa del líquido en el depósito por unidad de tiempo es igual a la masa de líquido que entra menos la que sale, despreciando fenómenos de evaporación, condensación,… .

$$\frac{dV}{dt} = \sum (Qi - Qo)$$

Al ser nuestros depósitos paralelepípedos, el volumen será igual a la sección transversal del tanque (A = 0.04 m^2) por su altura (h):

$$A\frac{dh}{dt} = \sum (Qi - Qo)$$

Por tanto, la ecuación de cada uno de los dos tanques aplicando la igualdad anterior será:

Tanque 1: $A\frac{dh_1}{dt} = Q_1 - Q_{12} - Q_{10}$

Tanque 2: $A\frac{dh_2}{dt} = Q_2 + Q_{12} - Q_{20}$

Según las leyes de la hidrodinámica, podemos calcular los caudales de descarga de cada tanque a través de la siguiente expresión:

$$Q_{10} = a \cdot C_d \cdot \sqrt{2 \cdot h_1 \cdot g}$$

$$Q_{20} = a \cdot C_d \cdot \sqrt{2 \cdot h_2 \cdot g}$$

a: Área transversal del orificio de desagüe ($1.9365 \cdot 10^{-5}$ m^2)
Cd: Coeficiente de descarga de la válvula.
g: Gravedad (9.8 m/s^2)

Basándonos en la ley de Torricelli, obtenemos la expresión del caudal que circula entre los dos tanques:

$$Q_{12} = a_{z12} \cdot S_n \cdot signo(h_1 - h_2)\sqrt{2 \cdot g \cdot (h_1 - h_2)}$$

az12: Coeficiente de correlación asociado a la tubería que comunica ambos tanques.
Sn: Sección de la tubería que comunica los dos tanques ($1.9365 \cdot 10^{-5}$ m^2).

Por último, para los caudales de entrada de cada tanque tenemos las siguientes expresiones:

$$Q_1 = V_{m1} \cdot C_m$$

$$Q_2 = V_{m2} \cdot C_m$$

Vm1 y Vm2: Tensiones aplicadas a las bombas de los tanques 1 y 2 respectivamente.
C_m: Coeficiente de relación entre el caudal de entrada al tanque y el voltaje aplicado a la bomba.

Para saber el caudal de las electrobombas para diferentes tensiones de entrada, representamos la siguiente bomba:

Tensión aplicada a la bomba	Caudal
10	0.00006451
9,5	0.00006349
9	0.00006289
8,5	0.00006079
8	0.00005952
7,5	0.00005680
7	0.00005600
6,5	0.00005555
6	0.00005080
5,5	0.00004484
5	0.00003980
4,5	0.00003686
4	0.00002985

3,5	0.00002185
3	0.00001600
2,5	0
2	0
1,5	0
1	0
0,5	0

Tabla 4.1.1 Relación caudal/tensión de entrada

El valor de C_m se calcula para un punto de trabajo determinado, en este caso tomaremos la mitad de la altura del depósito (h = 0.3 m) y para un voltaje medio teniendo en cuenta la zona muerta de trabajo de la bomba según la siguiente ecuación:

$$V_m = \frac{V_1}{2} + V_2 = \frac{10-3}{2} + 2.5 = 6V$$

V_1: Zona de funcionamiento normal de la bomba.
V_2: Zona muerta que tiene la bomba.

Con estos datos obtengo el valor de C_m:

$$C_m = \frac{Q_i}{V_m} = \frac{0.00005080}{6} = 0.00000846666$$

Donde **Q_i** es el caudal suministrado por cualquiera de las dos bombas, al ser éstas idénticas.

El coeficiente de descarga de la válvula (**C_d**) se rige por la siguiente ecuación:

$$C_d = \frac{Q_o}{a\sqrt{2 \cdot h \cdot g}}$$

Donde **Q_0** es el caudal que fluye por el orificio de desagüe de uno de los dos tanques ya que, como hemos dicho antes, ambos tanques son idénticos.

Habrá que elaborar una tabla para calcular **C_d**:

Altura h	Caudal de salida Q_o	Coeficiente C_d
0.45 - 0.40	$1.90476 \cdot 10^{-5} m^3/s$	0.484
0.40 – 0.35	$1.90476 \cdot 10^{-5} m^3/s$	0.510
0.35 – 0.30	$1.78571 \cdot 10^{-5} m^3/s$	0.507
0.30 – 0.25	$1.66667 \cdot 10^{-5} m^3/s$	0.506
0.25 – 0.20	$1.57480 \cdot 10^{-5} m^3/s$	0.517
0.20 – 0.15	$1.48148 \cdot 10^{-5} m^3/s$	0.533
0.15 – 0.10	$1.33333 \cdot 10^{-5} m^3/s$	0.536
0.10 – 0.05	$1.25000 \cdot 10^{-5} m^3/s$	0.580
0.05 – 0.00	$1.05263 \cdot 10^{-5} m^3/s$	0.598

Tabla 4.1.2 Relación entre la altura, caudal y coeficiente de descarga

Calculamos el valor medio de C_d y nos queda:

$$\mathbf{C_d = 0.530}$$

El coeficiente de correlación de la tubería que comunica ambos tanques se calcula a partir de la siguiente tabla:

Altura h_1	Altura h_2	Caudal Q_{12}	Coeficiente a_{z12}
0.60	0.10	$3.3333 \cdot 10^{-5} m^3/s$	0.5216
0.55	0.15	$2.8751 \cdot 10^{-5} m^3/s$	0.5237
0.50	0.20	$2.5000 \cdot 10^{-5} m^3/s$	0.4917
0.45	0.25	$1.8181 \cdot 10^{-5} m^3/s$	0.4716

Tabla 4.1.3 Relación entre alturas, caudal y coeficiente de descarga

Calculando el valor medio de $\mathbf{a_{z12}}$ obtenemos:

$$\mathbf{a_{z12} = 0.5022}$$

Con todos estos datos ya tenemos suficiente información para calcular la función de transferencia del sistema descrito. Para ello realizaremos un modelo en la herramienta de simulación SIMULINK de MATLAB el cual podemos ver a continuación.

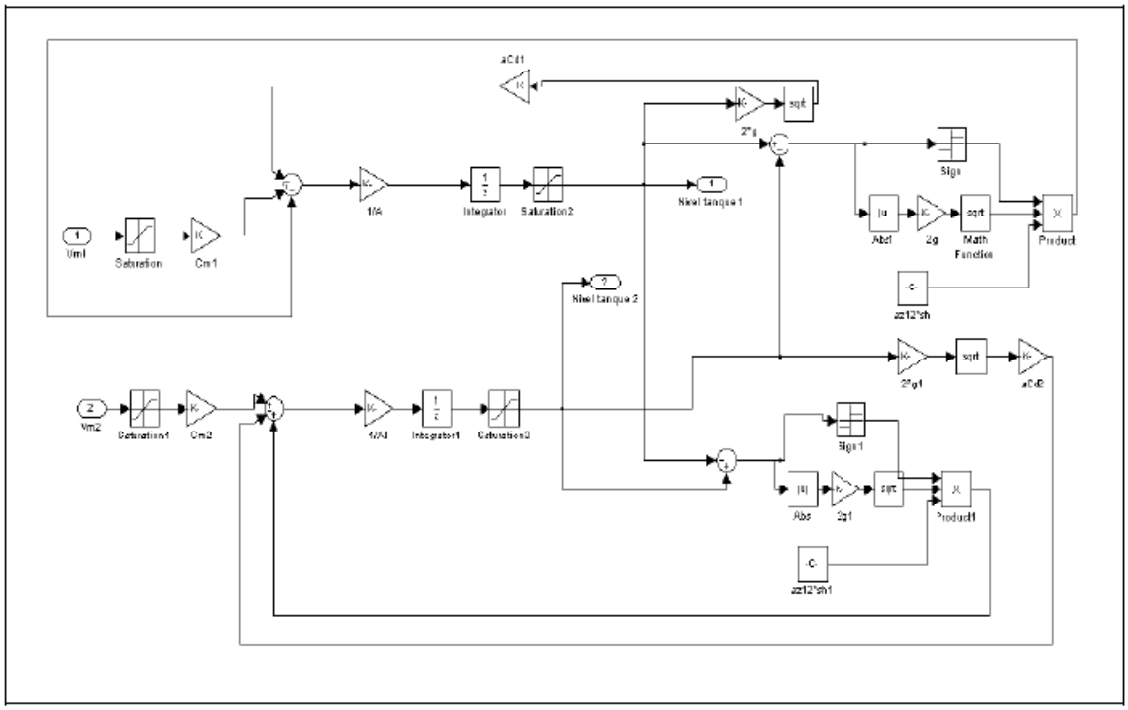


Figura 4.1.2 Modelo de la función de transferencia en SIMULINK

A partir de este modelo y utilizando las funciones 'LINMOD' y 'TRIM', presentes en la librería de *Matlab*, obtenemos la función de transferencia de cada uno de los dos tanques:

$$G_1(s) = \frac{h_1}{Vm_1} = \frac{0.0002117 \cdot s + 2 \cdot 10^{-7}}{s^2 + 0.002089 \cdot s + 1.091 \cdot 10^{-6}}$$

$$G_2(s) = \frac{h_1}{Vm_2} = \frac{0.0002117 \cdot s + 2 \cdot 10^{-7}}{s^2 + 0.002089 \cdot s + 1.091 \cdot 10^{-6}}$$

Como podemos ver las funciones de transferencia son iguales para los dos tanques. Simplificando obtenemos la función equivalente a la función de transferencia de la planta calculada anteriormente:

$$G_p(s) = \frac{0.1833}{858 \cdot s + 1}$$

4.2 Diseño del controlador del sistema

4.2.1 Consideraciones previas

El control de un proceso está basado en la elaboración de un modelo del mismo que permita el cálculo del correspondiente controlador.

El objetivo de nuestro proyecto es diseñar un controlador a partir de la función de transferencia de la planta y la función que engloba al controlador en serie con la planta y con realimentación unitaria. A continuación explicamos de manera más clara la obtención de esta función:

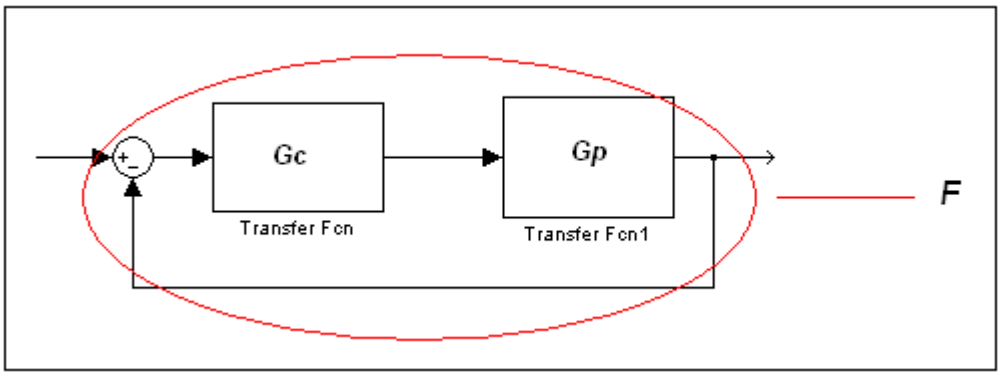


Figura 4.2.1.1 Función de transferencia del sistema

F: Función que se calcula a partir de los datos proporcionados.
Gp: Función de transferencia de la planta.
Gc: Función de transferencia del controlador.

$$F(s) = \frac{Gc \cdot Gp}{1 + Gc \cdot Gp}$$

Despejando de esta ecuación podemos escribir la expresión de 'Gc' en función de 'F' y 'Gp'.

$$Gc(s) = \frac{1}{Gp} \cdot \frac{F}{1 - F}$$

La función de transferencia 'F' debe cumplir las siguientes especificaciones:

- Tiempo de establecimiento (t_s) = 5 s.
- Factor de amortiguamiento (δ) = 0.9.

4.2.2 Cálculo de los parámetros del controlador

A partir de los datos proporcionados (t_s, δ) nos disponemos a calcular la función de transferencia 'F'.

Para ello escogemos un sistema de segundo orden cuya función de transferencia mostramos a continuación:

$$F(s) = \frac{K \cdot \omega n^2}{s^2 + 2 \cdot \delta \cdot \omega n + \omega n^2}$$

Para calcular la frecuencia propia no amortiguada (ωn) utilizamos la siguiente fórmula:

$$ts = \frac{3}{\delta \cdot \omega n}$$

Ponemos un '3' en el numerador ya que utilizamos el criterio del 5% (si utilizásemos el del 2% habría que poner un '4').

El valor de la frecuencia calculada es:

ωn = 0.667 rad/s

Sustituyendo valores en la función de transferencia de F(s) obtenemos:

$$F(s) = \frac{0.445}{s^2 + 1.2 \cdot s + 0.445}$$

Sustituyendo valores y despejando obtenemos la función de transferencia del controlador:

$$\boxed{Gc(s) = \frac{2083 \cdot s + 2.428}{s^2 + 1.2 \cdot s}}$$

Para nuestra aplicación serán necesarios dos controladores como este, uno para cada tanque.

Al no existir en el software *Unity Pro* ningún bloque para implementar esta función de transferencia como tal, habrá que escribirla como una ecuación en diferencias, la cual será implementada en lenguaje literal estructurado en nuestra aplicación.

Una ecuación en diferencias es una ecuación que muestra la relación entra valores consecutivos de una secuencia y la diferencia entre ellos. Usualmente se escribe en una ecuación recurrente para que la salida del sistema se pueda calcular de las entradas de la señal y sus valores anteriores.

La forma general de este tipo de ecuación es la siguiente:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k] \qquad (1)$$

También se puede expresar como una salida recurrente la cual se ve así:

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k] \qquad (2)$$

En esta ecuación y [n-k] representan las salidas y x [n-k] las entradas. El valor de **N** representa el orden de la ecuación de diferencia que corresponde a la memoria del sistema representado. Ya que la ecuación depende de los valores pasados de la salida, para calcular una solución numérica, algunos valores pasados, conocidos como condiciones iniciales, se deben saber.

En primer lugar hay que realizar la transformada 'z' de la función del controlador Gc (s):

$$Gc(z) = \frac{196.3 \cdot z - 196.277}{z^2 - 1.8869 \cdot z + 0.8869}$$

Una vez tenemos la función Gc (z) hay que dividir sus polinomios entre el orden más alto de 'z' (en este caso por z^2), quedándonos la siguiente expresión:

$$Gc(z^{-1}) = \frac{193.3 \cdot z^{-1} - 196.277 \cdot z^{-2}}{1 - 1.8869 \cdot z^{-1}\ \ 0.8869 \cdot z^{-2}}$$

De esta función de transferencia, los coeficientes de los dos polinomios serán nuestros valores de $\mathbf{a_k}$ y $\mathbf{b_k}$ que se encuentran el la forma general de la función de diferencias, ecuación 1. Usando estos coeficientes y la forma anterior de la función de transferencia, podemos escribir la ecuación de diferencia así:

$$196.3 \cdot x[n-1] - 196.277 \cdot x[n-2] = y[n] - 1.8869 \cdot y[n-1] + 0.8869 \cdot y[n-2]$$

En el último paso re-escribimos la ecuación de diferencia en una forma más común, mostrando su naturaleza recurrente del sistema, la cual utilizaremos para implementar en el controlador de nuestra aplicación:

y[n] = 196.3·x[n-1] – 196.277·x[n-2] + 1.8869·y[n-1] - 0.8869·y[n-2]

A continuación, vamos a ver la respuesta de la función de la planta y la de esta con el controlador en el dominio continuo. Primero le metemos una entrada de tipo escalón unitario a la planta y obtenemos la siguiente respuesta:

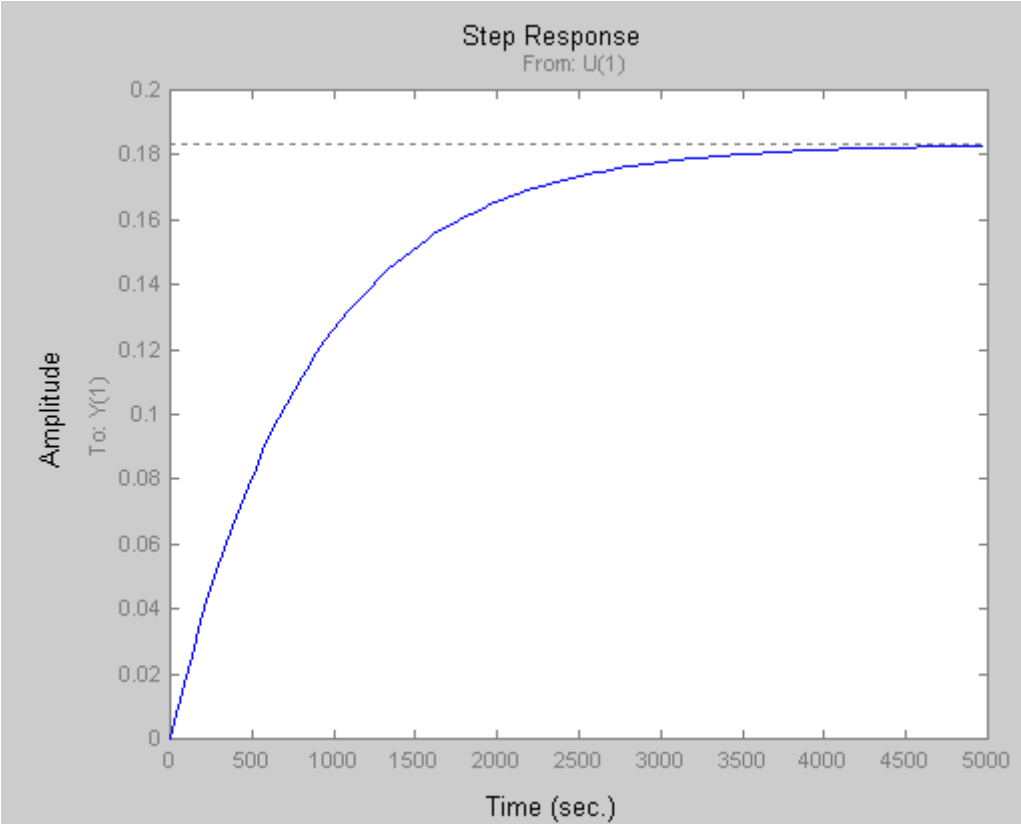


Figura 4.2.2.1 Respuesta de la función de la planta ante una entrada escalón

Como podemos apreciar en la imagen, la respuesta del sistema se aproxima a la de un sistema de primer orden cuyos parámetros característicos son los que se muestran a continuación.

Parámetros	Valor
Tiempo de Pico (t_p)	0 *seg.*
Sobrepico máximo (M_p)	0
Tiempo de crecimiento (t_r)	1890 *seg.*
Tiempo de asentamiento (t_s)	3320 *seg.*

Tabla 4.2.2.1 Parámetros característicos del controlador

Simulamos el sistema con el controlador calculado anteriormente y obtenemos la siguiente respuesta:

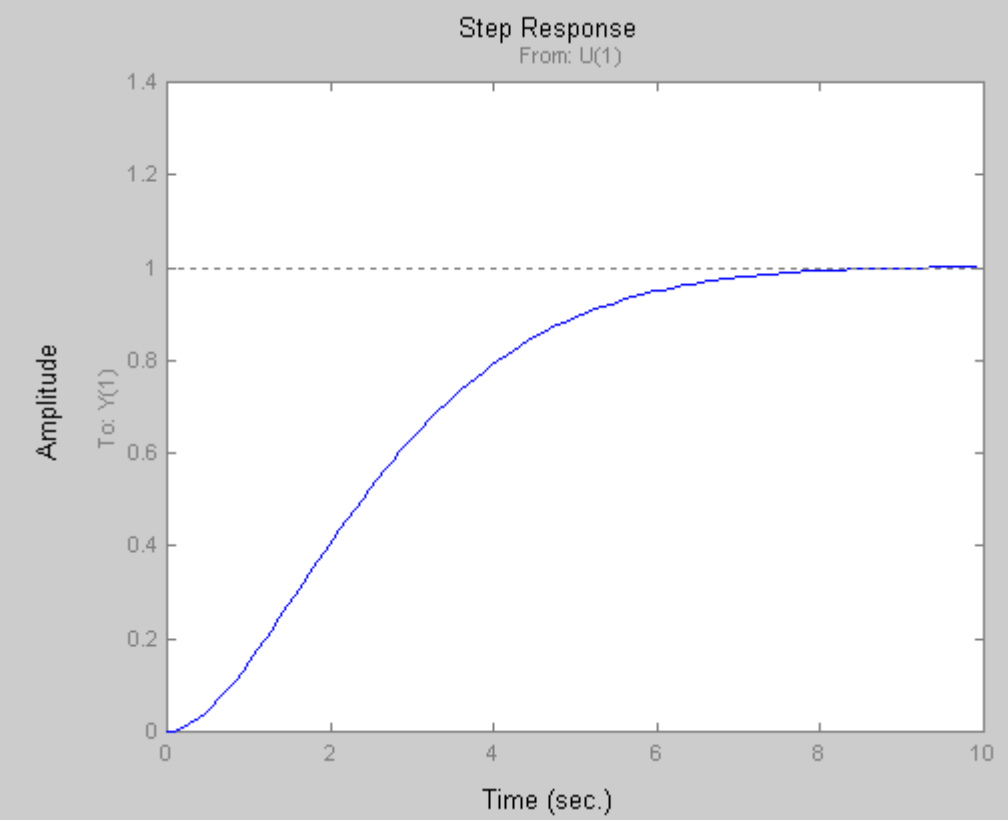


Figura 4.2.2.2 Respuesta de la función de transferencia F (s) ante una entrada escalón unitario

Como podemos observar el sistema es bastante rápido y, además, no presenta ningún sobrepico. En la siguiente tabla podemos ver los parámetros característicos del sistema con el controlador:

Parámetros	Valor
Tiempo de Pico (t_p)	$9.94\,seg.$
Sobrepico máximo (M_p)	0
Tiempo de crecimiento (t_r)	$4.33 seg.$
Tiempo de asentamiento (t_s)	$6.99\,seg.$

Tabla 4.2.2.2 Parámetros del sistema con el controlador

A continuación realizaremos un ensayo en el modelo de SIMULINK con los dos tanques y el controlador ya implementado, cuyo esquema es el siguiente.

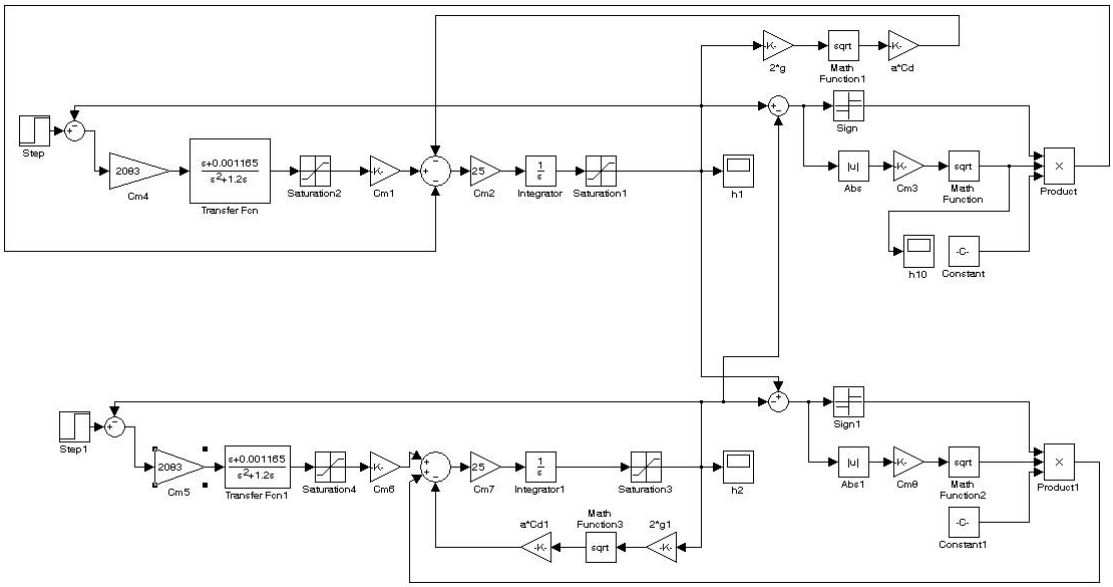


Figura 4.2.2.3 Montaje en 'Simulink' del sistema de tanques con los controladores

Vamos a dar un salto en la referencia de ambos tanques desde su estado estacionario (0.3039 m). En el primer tanque le vamos a dar hasta 0.5 m y en el segundo hasta 0.35 m. En las siguientes gráficas mostramos la respuesta de cada tanque:

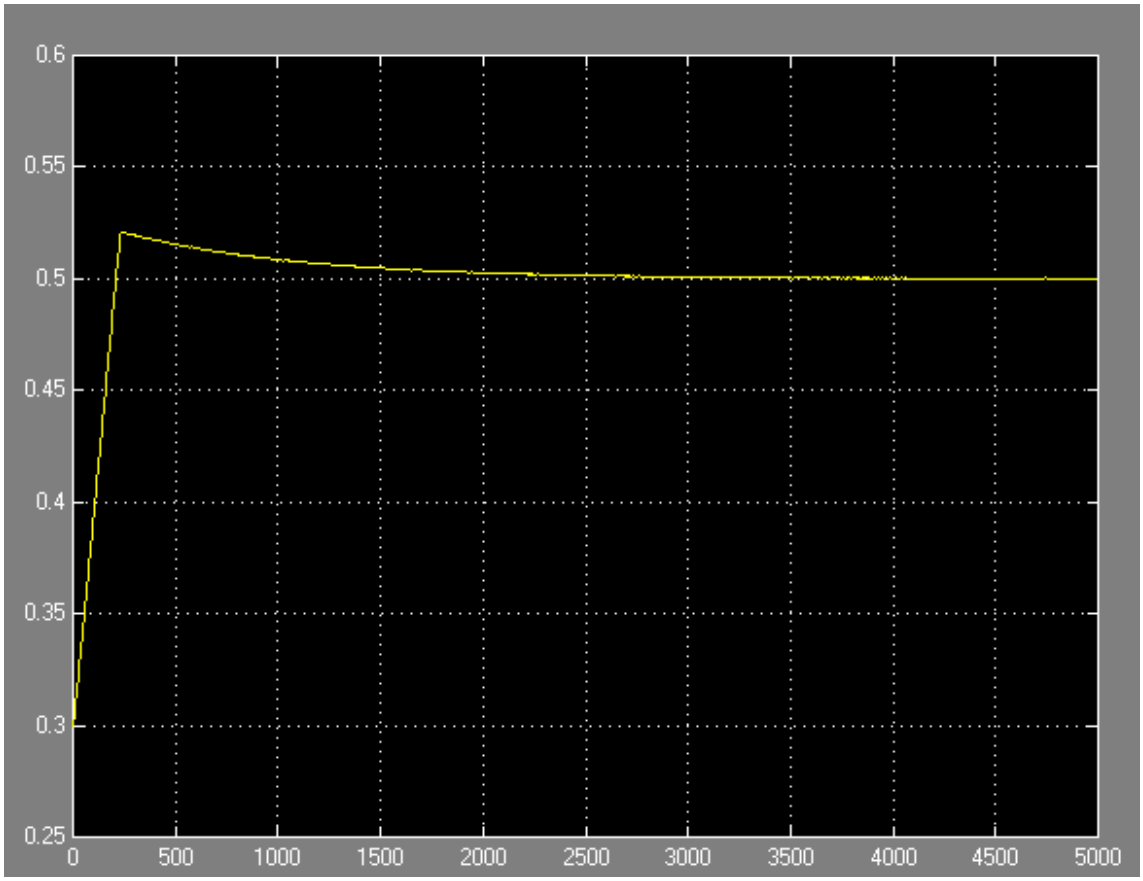


Figura 4.2.2.4 Respuesta del primer tanque

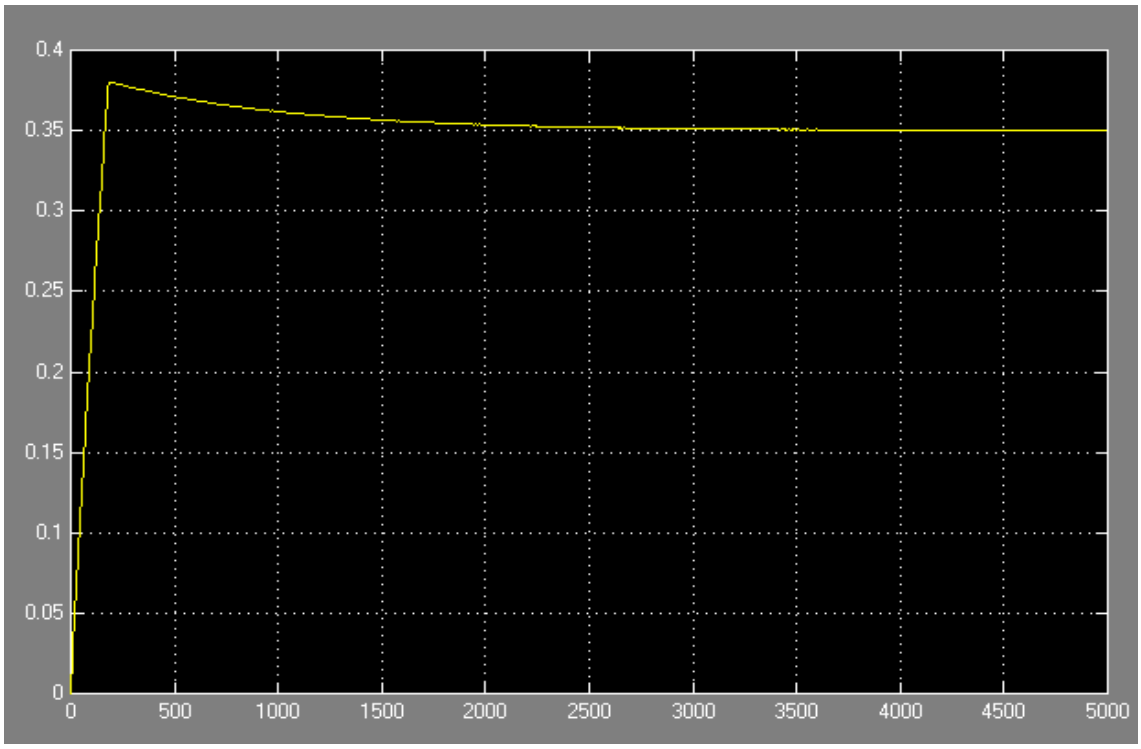


Figura 4.2.2.5 Respuesta del segundo tanque

Como podemos ver ambos sistemas se estabilizan en el punto de referencia aunque ambos presentan un pequeño sobrepico.

A continuación, se muestra la señal de control de ambos controladores ante las entradas de referencia anteriores:

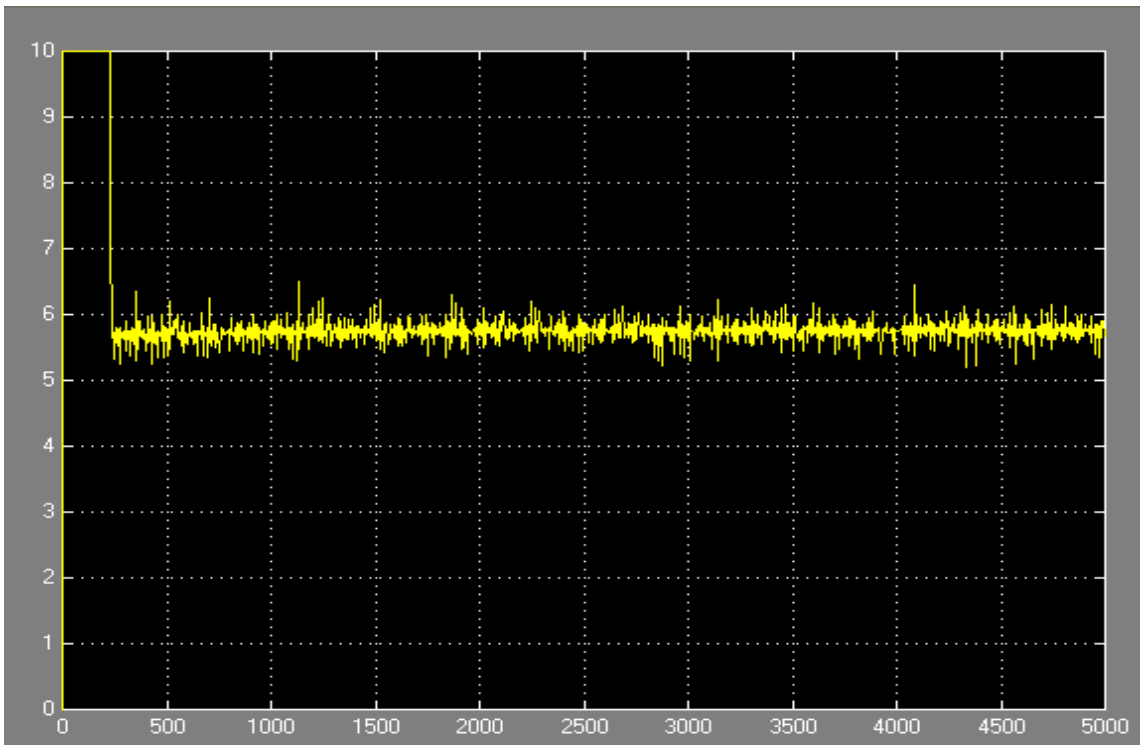


Figura 4.2.2.6 Señal de control del primer tanque

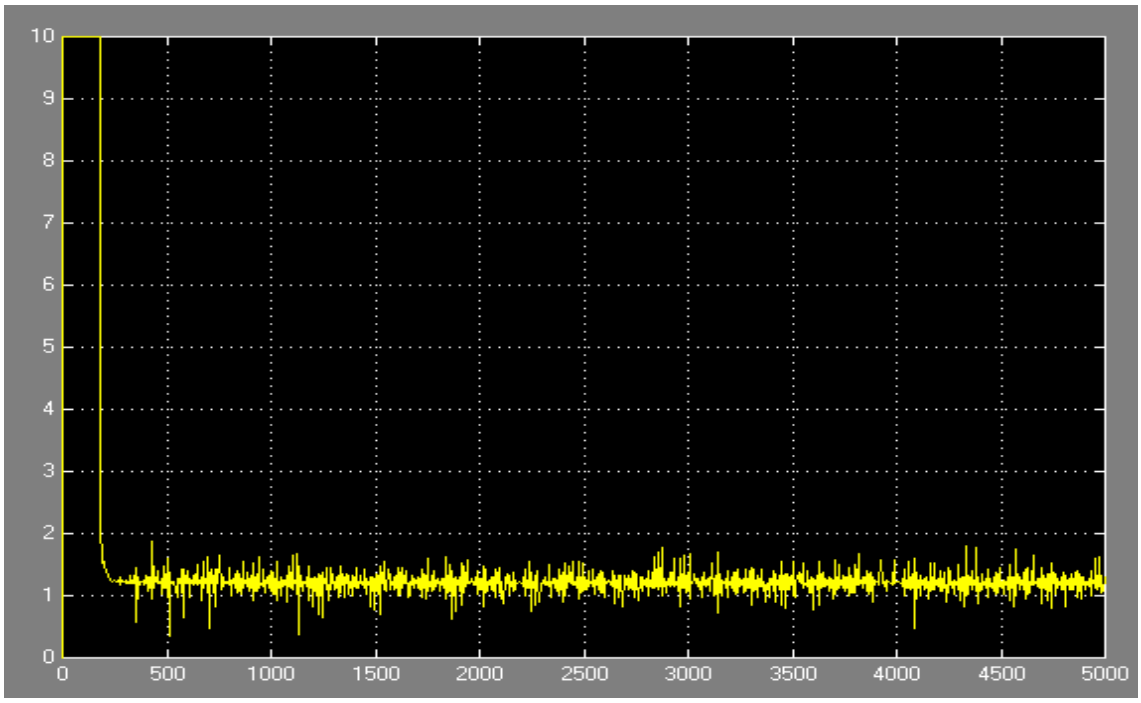


Figura 4.2.2.7 Señal de control del segundo tanque

5. DESARROLLO DE LA APLICACIÓN

5.1 Proyecto en Unity Pro

El comportamiento de la planta del sistema se identifica como un vaciado progresivo de los depósitos por efecto de la gravedad a través de los dos desagües y la tubería que comunica ambos tanques, siendo el flujo que circula por ésta última también dependiente del nivel presente en ambos tanques, la progresividad de esta descarga dependerá en todo caso del grado de apertura de las llaves.

El autómata comienza a ejecutar de manera cíclica la aplicación cargada cuando recibe una tensión de alimentación. La ejecución sólo se detendrá cuando ya no reciba alimentación.

Una vez pulsado el interruptor ON/OFF de la caja de mando mencionada en apartados anteriores, queda alimentada la planta. Los transductores de nivel comienzan a convertir las alturas del líquido de los depósitos en una corriente proporcional que envían al módulo, el cual transforma estas corrientes en tensión, que el autómata lee en sus entradas analógicas y reconoce como el nivel de fluido de los tanques.

En estado de funcionamiento normal este nivel de fluido es introducido en el controlador de la aplicación, el cual lo compara con el nivel óptimo asignado por el operador y en función de la diferencia entre estos niveles actúa sobre las salidas correspondientes a las bombas en caso de ser necesario. Estas salidas del autómata se amplificarán de manera que exciten a las bombas dentro de los rangos de trabajo del modelo utilizado.

Además, en cualquier momento se puede pulsar el botón de emergencia de la caja de mando y, automáticamente, se para el funcionamiento normal de la aplicación, se detiene la actuación sobre las salidas correspondientes a los motores y entra en funcionamiento el protocolo de reseteo necesario para poner de nuevo la aplicación en modo de funcionamiento normal. Este protocolo consistirá primero en la desactivación de la señal de emergencia y, en segundo lugar, la activación del reset externo del sistema, el cual se encuentra también en la caja de mando.

También pueda darse el caso de un funcionamiento interno anómalo debido al envío de medidas de error por parte de alguno de los transductores o a un funcionamiento anómalo de estos que el propio transductor detectará enviando una señal de alarma, en este caso también se paraliza el funcionamiento normal y será necesario de nuevo la activación del reset externo una vez subsanado el error en el transductor para iniciar el funcionamiento normal de la aplicación.

5.1.1 Creación del proyecto

Para crear un proyecto con el software *Unity Pro*, lo primero que tenemos que hacer es crear un fichero nuevo y, posteriormente, seleccionar el tipo de procesador con el que vamos a trabajar. En nuestro caso se tratará del *Modicon M340 BMXP341000.*

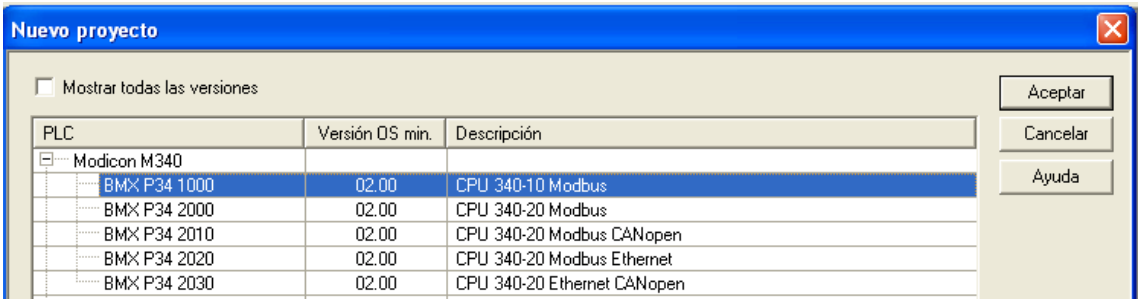


Figura 5.1.1.1 Ventana de selección del procesador

Una vez elegido el procesador, se creará el proyecto y ya no se podrá cambiar de familia de PLC's.

Para guardar los cambios, abrir otro proyecto, crear uno nuevo o exportarlo a otro formato recurriremos a la pestaña '*Ficheros*'.

Figura 5.1.1.2 Detalle del menú 'Fichero'

5.1.2 Configuración

Una vez tenemos creado y definido nuestro proyecto, el paso siguiente es definir los distintos parámetros y configuraciones más acordes con la aplicación que queramos desarrollar, esto se realiza mediante el editor de configuración, donde desde el catálogo de hardware seleccionamos los distintos dispositivos y que inicialmente presenta un aspecto como el que se muestra a continuación.

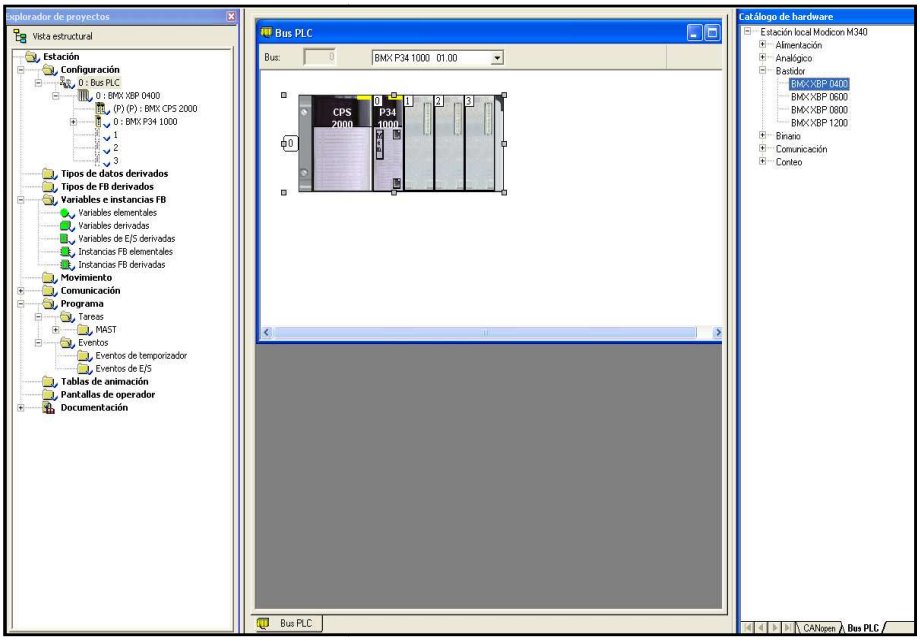


Figura 5.1.2.1 Configuración inicial por defecto

En primer lugar, debemos elegir el bastidor, elemento básico de la plataforma de automatización donde se colocarán el resto de elementos necesarios para la ejecución de la aplicación. En nuestro caso utilizaremos el bastidor *BMX XBP 0400*.

Después debemos seleccionar la fuente de alimentación. El modelo escogido es el BMX CPS 2000.

En la siguiente imagen mostramos el balance de consumo de la fuente de alimentación que presenta el rack que compone la configuración del autómata en las distintas tensiones suministradas por la fuente de alimentación.

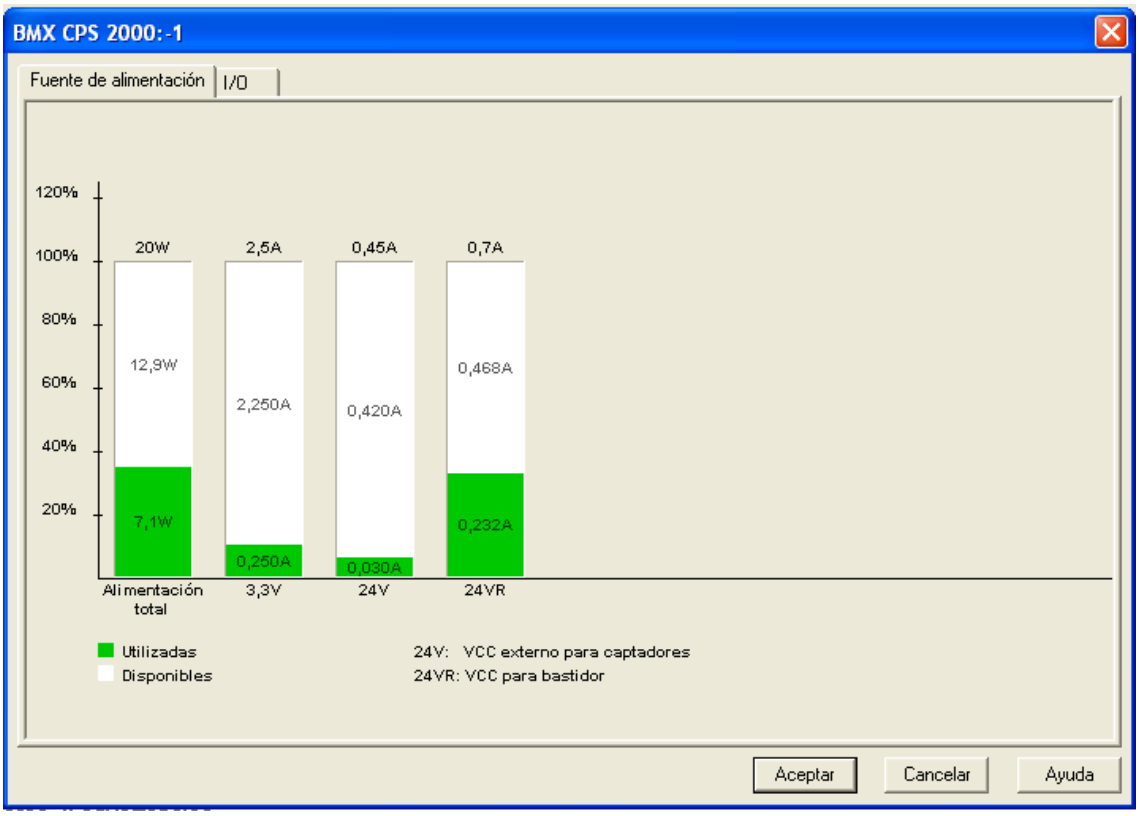
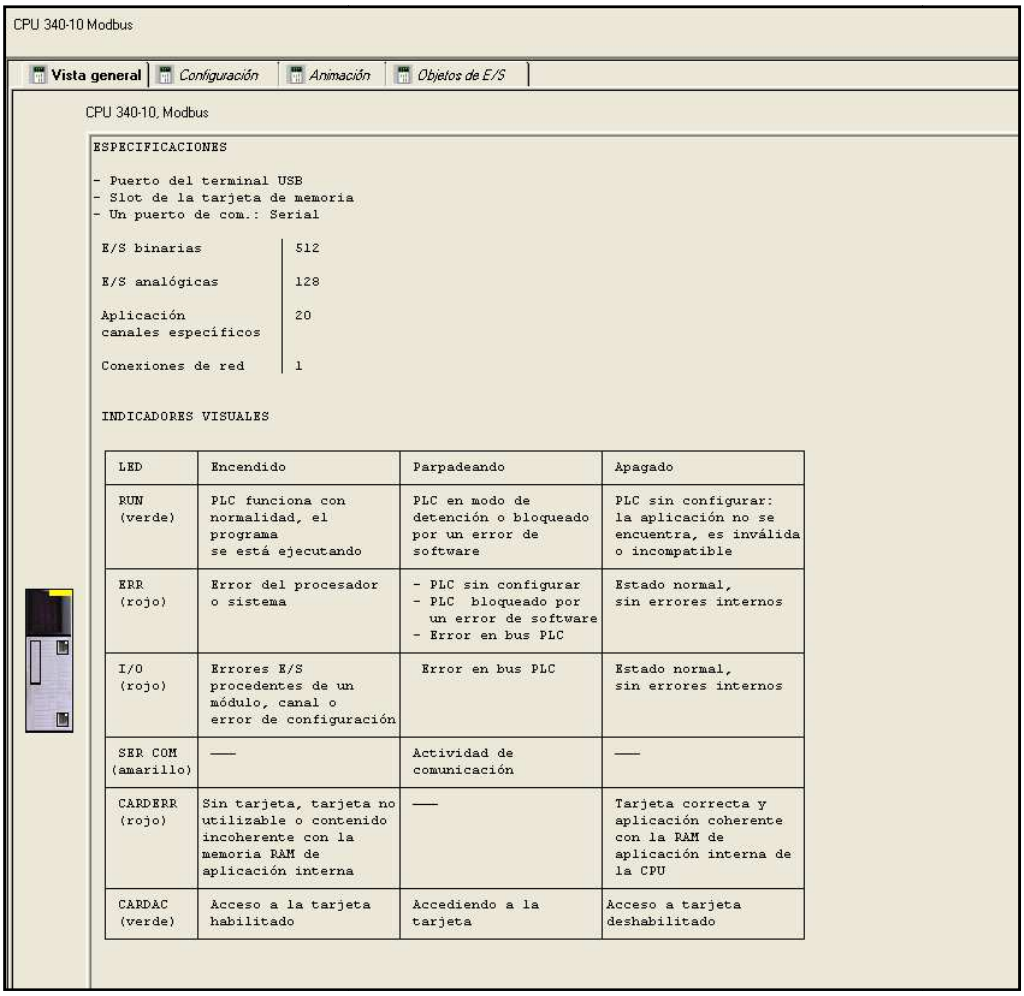


Figura 5.1.2.2 Previsión de alimentación de la fuente de alimentación

Después, escogemos el procesador que en nuestra aplicación será el modelo BMX P34 1000. Las características de dicho procesador se muestran en la siguiente tabla:



CPU 340-10 Modbus

Vista general | Configuración | Animación | Objetos de E/S

CPU 340-10, Modbus

ESPECIFICACIONES

- Puerto del terminal USB
- Slot de la tarjeta de memoria
- Un puerto de com.: Serial

E/S binarias	512
E/S analógicas	128
Aplicación canales específicos	20
Conexiones de red	1

INDICADORES VISUALES

LED	Encendido	Parpadeando	Apagado
RUN (verde)	PLC funciona con normalidad, el programa se está ejecutando	PLC en modo de detención o bloqueado por un error de software	PLC sin configurar: la aplicación no se encuentra, es inválida o incompatible
ERR (rojo)	Error del procesador o sistema	- PLC sin configurar - PLC bloqueado por un error de software - Error en bus PLC	Estado normal, sin errores internos
I/O (rojo)	Errores E/S procedentes de un módulo, canal o error de configuración	Error en bus PLC	Estado normal, sin errores internos
SER COM (amarillo)	——	Actividad de comunicación	——
CARDERR (rojo)	Sin tarjeta, tarjeta no utilizable o contenido incoherente con la memoria RAM de aplicación interna	——	Tarjeta correcta y aplicación coherente con la RAM de aplicación interna de la CPU
CARDAC (verde)	Acceso a la tarjeta habilitado	Accediendo a la tarjeta	Acceso a tarjeta deshabilitado

Figura 5.1.2.3 Especificaciones del procesador BMX P34 1000

El procesador incluye un puerto específico para las comunicaciones mediante puerto serie *Modbus*, configurable desde el explorador de proyectos.

Mediante el menú de configuración del procesador vamos a poder definir una serie de parámetros para cada canal de comunicación:

- Tipo de comportamiento maestro/esclavo.
- Línea de conexión.
- Retardo entre bloques de datos.
- Formato de datos.
- Velocidad de transmisión.
- Bits de parada.
- Paridad.

Esta ventana de configuración tiene el siguiente aspecto:

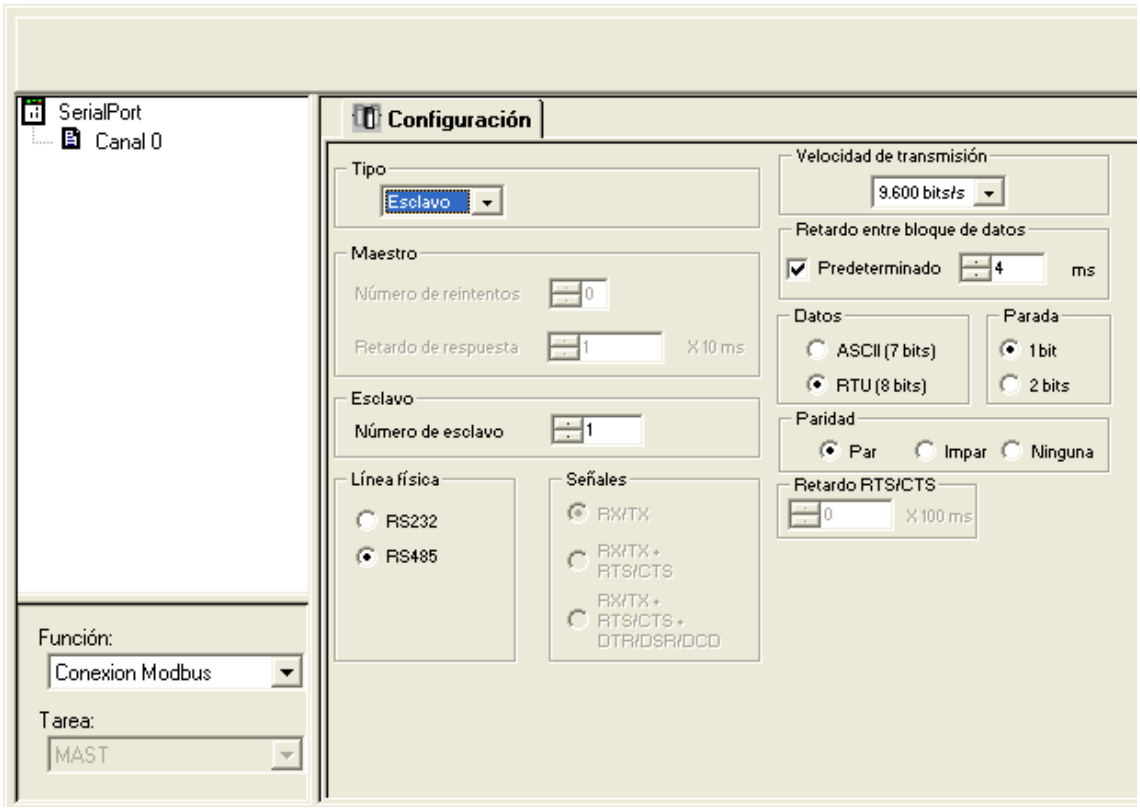


Figura 5.1.2.5 Panel de configuración del procesador

Además para nuestra aplicación necesitaremos un módulo de E/S analógicas y otro de E/S digitales.

Para seleccionar el módulo digital clicamos sobre la pestaña 'Binario' del catálogo de hardware y seleccionamos el modelo BMX DDM 16025. Este modelo dispone de ocho entradas y ocho salidas digitales.

Una vez seleccionada la tarjeta pasamos a asignar las variables de E/S, asignándolas una dirección concreta que nos permitirá acceder a ellas en cualquier momento. Las primeras ocho posiciones se corresponden con las entradas y las ocho restantes con las salidas.

Podemos observar la configuración de E/S digitales de nuestra aplicación en las dos siguientes imágenes:

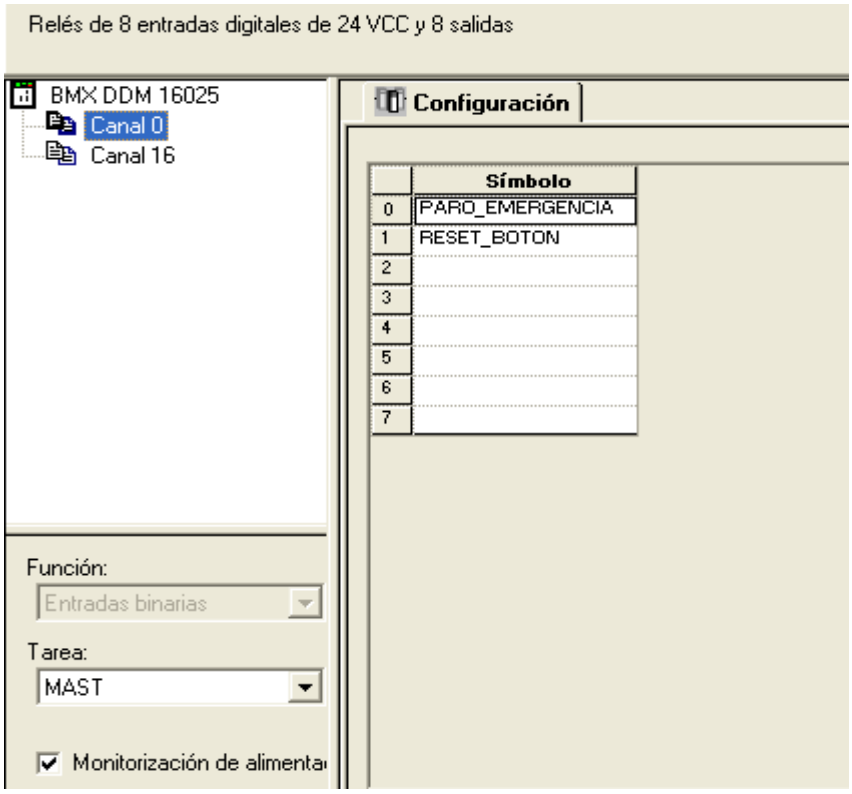


Figura 5.1.2.6 Configuración de las entradas digitales

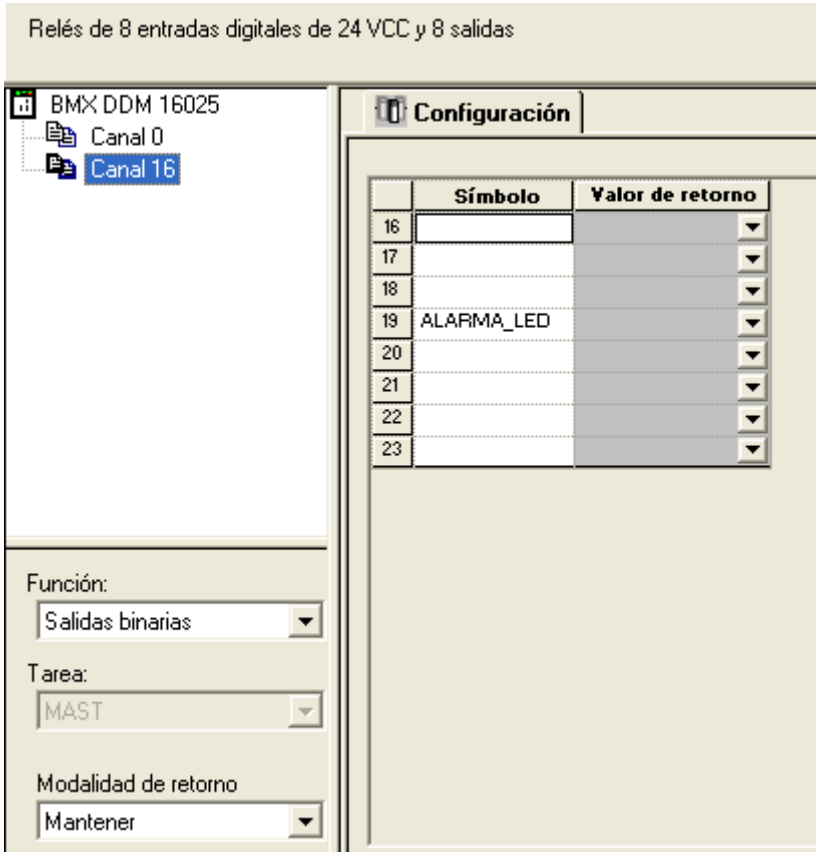


Figura 5.1.2.7 Configuración de las salidas digitales

Para las E/S digitales seleccionaremos el módulo analógico desde la pestaña 'Analógico' del catálogo de hardware. El modelo que vamos a utilizar es el BMX AMM 0600, el cual tiene cuatro entradas y dos salidas analógicas.

Una vez añadido este módulo debemos asignar las variables de entrada y salida a un dirección concreta para poder acceder a ellas en cualquier momento. Las cuatro primeras posiciones se corresponden con las entradas y las dos restantes con las salidas. Podemos también definir el rango en el que estas van a ser leídas o escritas en función de si son entradas o salidas dependiendo de los requerimientos de nuestra aplicación.

En la siguiente imagen aparece la configuración de E/S analógicas del módulo de nuestra aplicación:

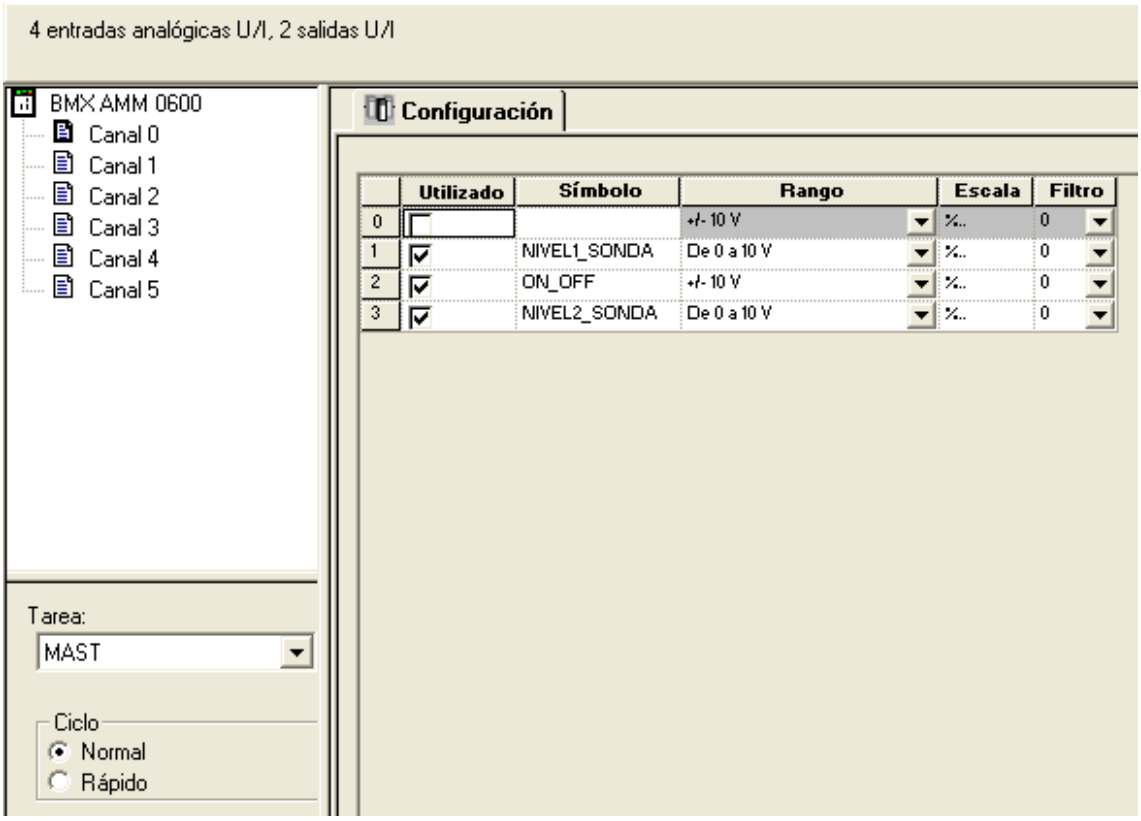


Figura 5.1.2.8 Configuración de las entradas analógicas

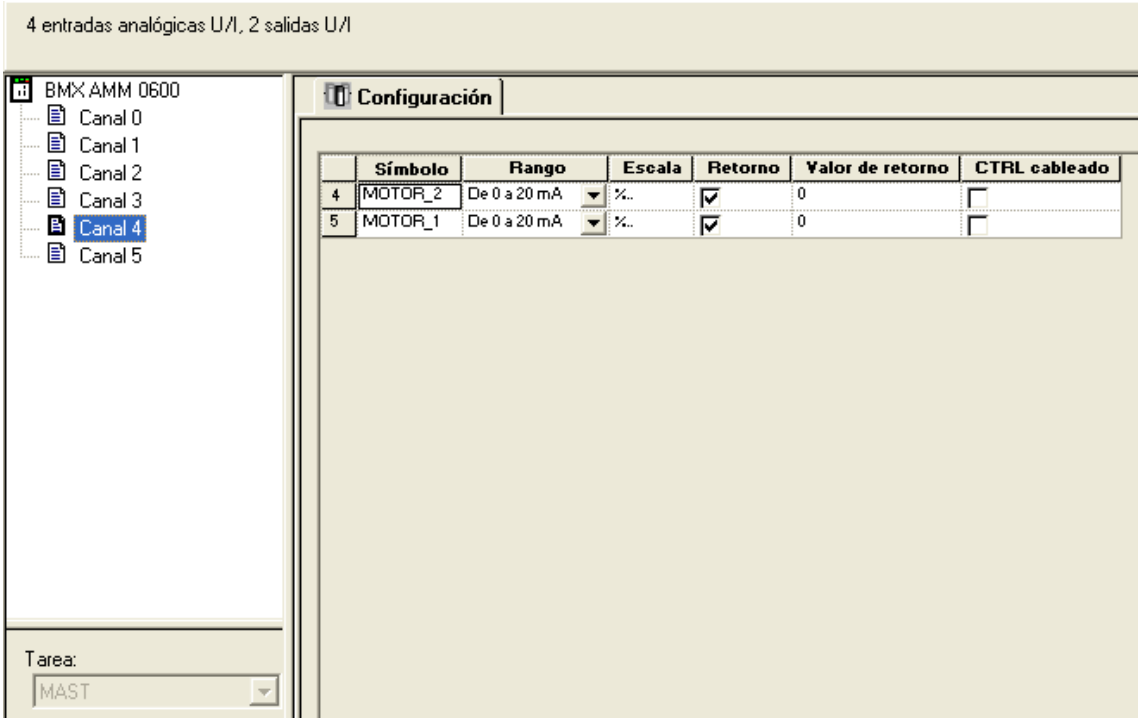


Figura 5.2.1.9 Configuración de las salidas analógicas

Ahora ya tenemos completamente configurada la plataforma de automatización:

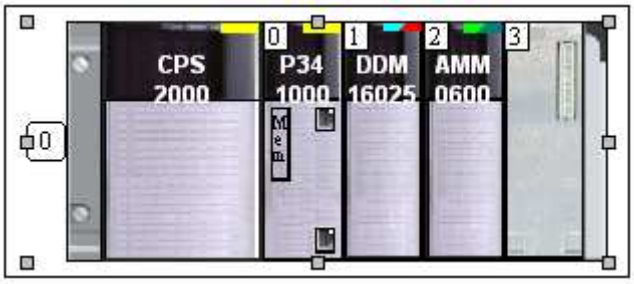


Figura 5.1.2.10 Aspecto final de la configuración

5.1.3 Creación y asignación de variables

Una variable es una entidad de memoria de diferentes tipos cuyos contenidos pueden ser modificados por el programa durante la ejecución. Tenemos dos tipos de variables: alocatadas (direccionada) y no alocatadas (no direccionada).

Crear una variable permite utilizarla dentro del programa, dentro de las tablas de animación, dentro de las pantallas del operador,… .

A través del editor de datos podemos manejar estas variables. Los tipos de variables presentes en nuestro entorno de automatización son los siguientes:

- *Variables*: elementales, derivadas –de un tipo creado por el usuario-, de diagnóstico,… .
- *DDT*: Tipos de datos derivados; estructura y arrays (tablas).
- *Bloques de funciones*: bloques funcionales de la librería o bloques de función de derivados (de usuario o de la librería) insertados en una sección de programación.
- *Bloques DFB*: Bloques de datos derivados.

En las siguientes imágenes se muestran todos los tipos de variables utilizados en nuestra aplicación:

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

Filtro — Nombre: * — EDT DDT IODDT

Nombre	Tipo	Dirección	Valor	Comentario
AAA	BOOL			Todas las válvulas abiertas
AAA_defecto	BOOL			Variable correspondiente al estado de defecto(todas las válvulas abiertas)
AAC	BOOL			Válvula derecha cerrada
ACA	BOOL			Válvula central cerrada
ALARMA_LED	EBOOL	%Q0.1.19		Boton encendido led alarma
CAA	BOOL			Válvula izquierda cerrada
CERO	BOOL		0	0 lógico
CIEN	REAL		100.0	
Desague_c_a	BOOL			Desague central abierto
Desague_c_c	BOOL			Desague central cerrado
Desague_d_a	BOOL			Desague derecho abierto
Desague_d_c	BOOL			Desague derecho cerrado
Desague_i_a	BOOL			Desague izquierdo abierto
Desague_i_c	BOOL			Desague izquierdo cerrado
Dif_nivel12	INT			
Dif_nivel21	INT			
error_sonda	BOOL			Variable booleana correspondiente a un error en la sonda
Estado_motor_1	BOOL			
Estado_motor_2	BOOL			
Igual12	BOOL			
Igual21	BOOL			

Figura 5.1.3.1 Variables de la aplicación

Los bloques de funciones utilizados son los siguientes:

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

Filtro — Nombre: * — EFB DFB

Nombre	Nº	Tipo	Valor	Comentario
FBI_0		TON		
<entradas>				
IN	1	BOOL		Start delay
PT	2	TIME		Preset delay time
<salidas>				
Q	1	BOOL		Delayed output
ET	2	TIME		Internal time
<entradas/salidas>				
<público>				
FBI_1		PI_B		
<entradas>				
PV	1	REAL		Process value
SP	2	REAL		Setpoint
RCPY	3	REAL		Actuator position copy
MAN_AUTO	4	BOOL		Manual (0) or automatic (1) mode
PARA	5	Para_PI_B		Parameter
TR_I	6	REAL		Initialization input
TR_S	7	BOOL		Initialization command
<salidas>				
OUTD	1	REAL		Output variation from PID
MA_O	3	BOOL		Automatic mode (1) or other mode (0)
DEV	5	REAL		Deviation
STATUS	6	WORD		Status word
<entradas/salidas>				
<público>				
FBI_4		TON		
FBI_27		PI_B		
FBI_49		AVGMV		
<entradas>				
MAN	1	BOOL		Mode: 0: automatic, 1: manual
X	2	REAL		Input
N	3	INT		No. of input values loaded in buffer
YMAN	4	REAL		Manual value
<salidas>				
Y	2	REAL		Mean value
RDY	3	BOOL		1: Buffer ready, 0:not ready
<entradas/salidas>				
<público>				
FBI_50		AVGMV		
FBI_56		AVGMV_K		
<entradas>				
MAN	1	BOOL		Mode: 0:automatic, 1:manual
X	2	REAL		Input
K	3	INT		Correction factor
YMAN	4	REAL		Manual value
<salidas>				
Y	1	REAL		Mean value
<entradas/salidas>				
<público>				
FBI_57		AVGMV_K		

Figura 5.1.3.3 Bloques de funciones de la aplicación

El último tipo de variables utilizadas, ya que no se usan tipos de datos derivados (DDT), son los bloques de datos derivados desarrollados para esta aplicación, su estructura interna es genérica variando para cada uno las entradas y salidas y las secciones en función del comportamiento buscado por lo que únicamente se muestra el primero de los bloques, encargado de actualizar los valores de las entradas (‘Actualiza entradas’) realizando la conversión de entero a booleano, dos de ellos (‘Rellenado tanque’ y ‘Comprobación tanque lleno’) dan servicio al estado correspondiente en función del nivel de líquido, otro es el encargado de implementar el controlador de la aplicación (‘Controlador’) y el último es el encargado de estabilizar diferentes valores (‘Estabilizacion medidas’), en este caso la medida de nivel a

introducir en el controlador y la salida de este para su posterior utilización como salida del autómata que activa la bomba.

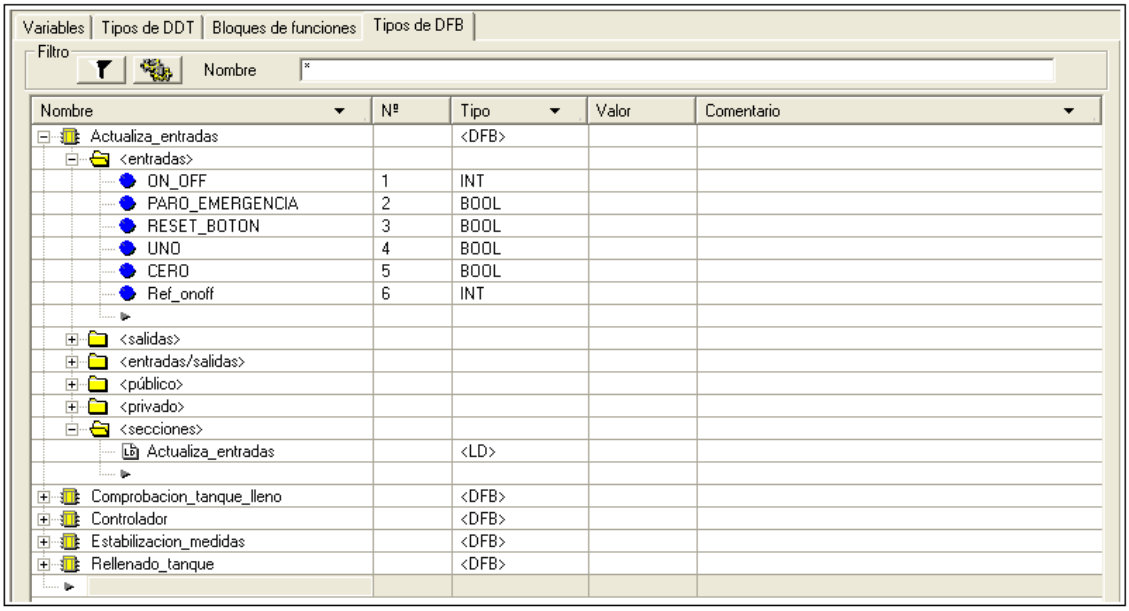


Figura 5.1.3.4 Bloques de funciones de nuestra aplicación

5.1.4 Programación en lenguaje LD y ST

Las diferentes secciones de la aplicación se crean utilizando estos dos tipos de lenguaje, el motivo de esta elección es debido a la mayor experiencia que he desarrollado manejando estos dos lenguajes en la creación de proyectos y aplicaciones anteriores.

El lenguaje de programación se configura inicialmente al generar una nueva sección como primer paso antes de comenzar con la programación en sí.

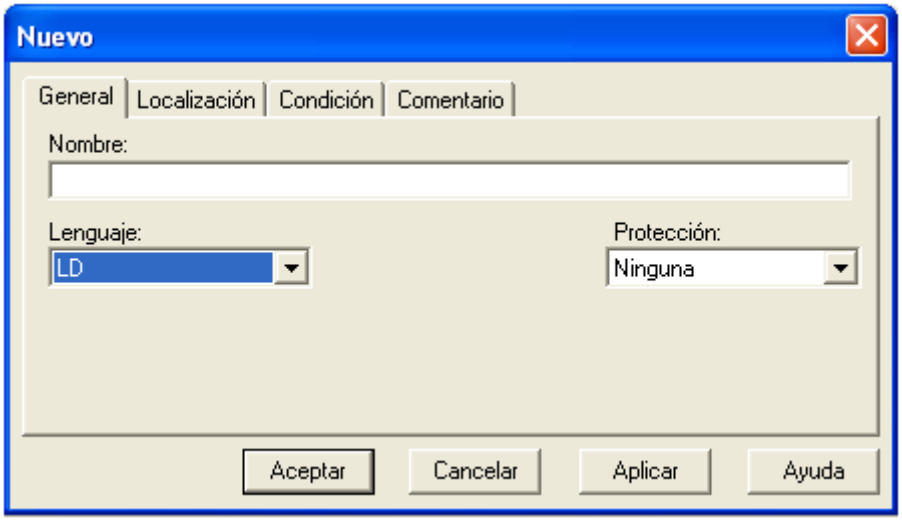


Figura 5.1.4.1 Configuración de sección

Tras este primer paso comienza la programación de la sección en el editor correspondiente dependiendo del lenguaje elegido.

Empezaremos viendo el lenguaje LD o de contactos.

La estructura de un programa en LD equivale a un circuito de corriente para conmutadores de relé.

En el lado izquierdo del editor LD se encuentra la denominada '*barra de alimentación izquierda*'. Esta barra corresponde a la fase (conductor L) de un circuito de corriente. Al igual que en un circuito de corriente, en la programación LD sólo se editan los objetos que estén conectados a una fuente de alimentación. La barra de alimentación derecha equivale al conductor neutro.

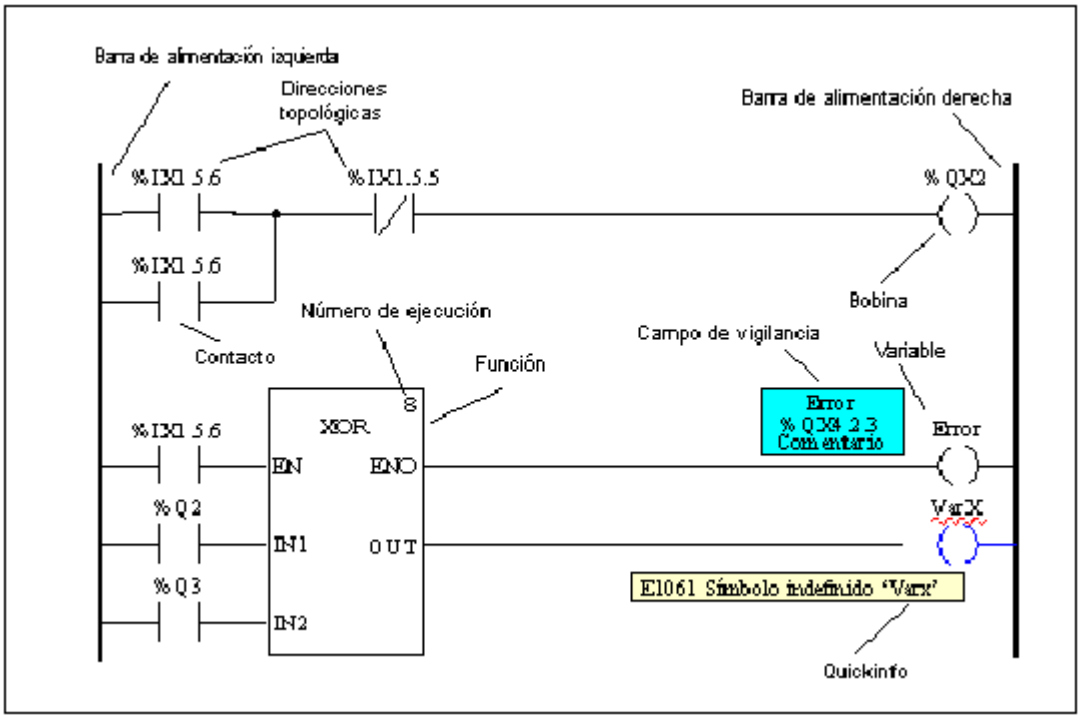


Figura 5.1.4.2 Estructura del lenguaje LD

Una sección LD se compone por una ventana de una sola página, dividida en filas y columnas por medio de una rejilla de fondo como se puede apreciar en la siguiente figura:

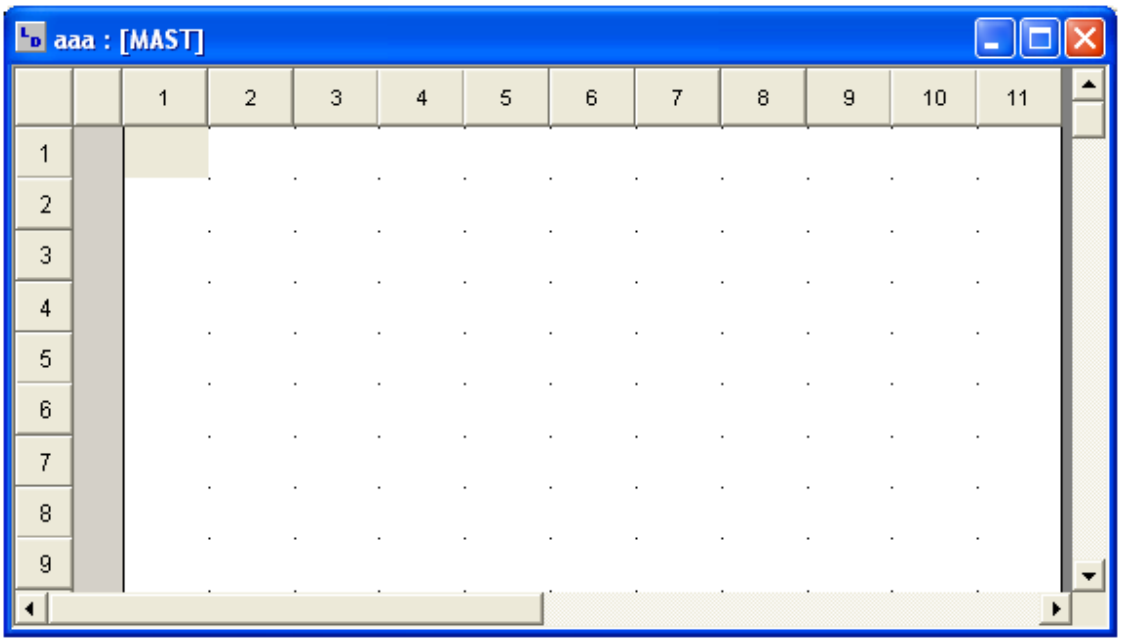


Figura 5.1.4.3 Editor LD

El lenguaje de programación LD se basa en celdas, en cada una de estas se puede colocar un único objeto, los objetos para llevar a cabo la lógica de la sección se deben elegir a partir de una amplia lista en la que se incluyen diferentes elementos:

- Contactos.
- Bobinas.
- Funciones y bloques de funciones elementales.
- Bloques de funciones derivados.
- Procedimientos.
- Elementos de control.
- Bloques de operación y comparación.
- Saltos.
- Llamadas a subrutinas.

Todos estos elementos se manejan de manera gráfica sobre la ventana del editor seleccionándose fácilmente desde la barra de herramientas del mismo cuyo aspecto vemos a continuación.

Figura 5.1.4.4. Barra de herramientas del editor LD

Un contacto es un elemento que transmite un estado a la conexión horizontal de su parte derecha sin modificar el valor del parámetro correspondiente. Puede ser normalmente abierto, normalmente cerrado o de detección de transición positiva.

Una bobina es un elemento que transmite el estado de la conexión vertical de su parte izquierda sin modificar a la conexión horizontal de su parte derecha. El estado se guarda en el parámetro correspondiente. Normalmente suelen seguir a los bloques de funciones. Pueden ser normales, negadas o de detección de transición positiva o negativa.

Las funciones elementales son bloques de datos sin estados internos con varias entradas y una salida que realizan una operación concreta. Los bloques de funciones son lo mismo pero con estados internos y varias salidas.

Los bloques de funciones derivadas son bloques de funciones como los anteriores en los que el usuario crea la operación que realiza internamente en los lenguajes de programación correspondientes. Gráficamente se diferencian de los anteriores por la representación con líneas verticales dobles.

Los procedimientos son funciones elementales especiales que admiten otro tipo de datos específicos que no se pueden operar con los bloques de funciones. Son una ampliación de la norma *CEI 61131-3* y se deben habilitar de forma explícita.

Los elementos de control ejecutan los saltos dentro de la sección y el retorno anticipado a la rutina principal desde una subrutina o un bloque de función derivado.

Por su parte los bloques de operación ejecutan instrucciones y expresiones, mientras que los de comparación realizan operaciones de comparación sencillas.

A continuación vemos en la figura un cuadro resumen de todos estos elementos.

Icono	Función	Teclado
	Modalidad de selección.	
	Contacto normal abierto.	F3
	Contacto normal cerrado.	Mayús.+F3
	Contacto de detención de transición positiva.	Ctrl+F3
	Contacto de detención de transición negativa.	Ctrl+Mayús.+F3
	Bobina.	F5
	Bobina negada.	Mayús.+F5
	Bobina de ajuste.	Alt+F5
	Bobina de restablecimiento.	Mayús.+Alt+F5
	Bobina de detención de transición positiva.	
	Bobina de detención de transición negativa.	
	Detener bobina.	
	Bobina de llamada.	F4
	Conexión booleana.	F7
	Conexión vertical.	Mayús.+F7
	Enlace booleano.	Alt+F6
OPER	Bloque de funcionamiento.	Alt+F7
COMP	Bloque de comparación.	Ctrl+F7
	Salto.	
JL:	Etiqueta de salto.	
	Retorno.	
	Selección de datos.	Ctrl+D
	Asistente de entrada FFB.	Ctrl+I
	Conexión.	F6
	Alternar negación pin.	
	Comentario.	F8
	Ventana de inspección.	F9

Figura 5.1.4.5. Detalle de iconos de la barra de herramientas

Por medio de la asignación de las diferentes variables del programa a estos elementos y las conexiones entre diferentes objetos vamos generando la sección, en todo momento el programa verifica que no existen errores de ningún tipo (sintácticos o semánticos) a la hora de asignar las variables, esta comprobación se puede ver directamente en la sección mediante textos coloreados, como cuadro de información al colocar el puntero sobre un texto con errores o en la ventana de resultados al analizar la sección.

El flujo de estos datos dentro de la sección determina la secuencia de procesamiento de los diferentes objetos. Las redes conectadas al rail izquierdo se procesan de izquierda a derecha y de arriba abajo.

El otro lenguaje utilizado en la programación de la aplicación es el lenguaje ST, o de texto estructurado.

Este lenguaje trabaja con las denominadas "expresiones", que son construcciones compuestas por operadores y operandos que devuelven un valor durante la ejecución. Los operadores son símbolos para las operaciones que se van a ejecutar estos son las variables que se aplican a los operandos. Las distintas instrucciones sirven para asignar a los parámetros actuales los valores devueltos por las expresiones y para estructurar y controlar estas.

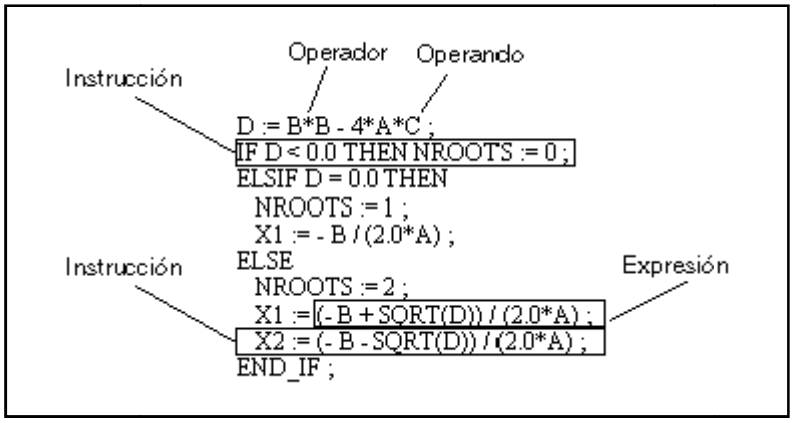
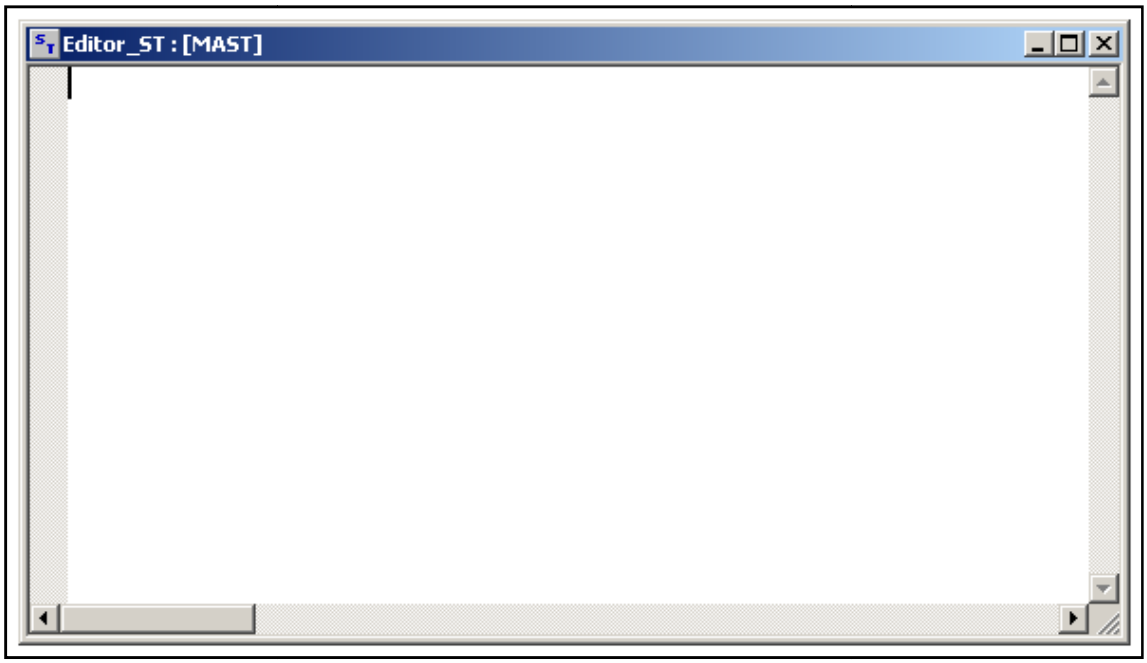


Figura 5.1.4.6 Estructura del lenguaje ST

Una sección ST no está limitada en el entorno de programación, únicamente está limitada por la memoria del PLC. Si que está limitada la longitud de una línea de instrucciones, a 300 caracteres.



5.1.4.7 Ventana del editor ST

La evaluación de una expresión está formada por la aplicación de los operadores sobre los operandos en el orden en el que se haya definido la jerarquía de los operadores, un operador es el símbolo de una operación o de un procesamiento de función, tiene la característica de adaptarse automáticamente al tipo de dato de los operandos.

En la siguiente tabla vemos los operadores disponibles en el lenguaje de programación ST.

Operador	Significado	Operadores	Descripción/Funcionalidad
()	Paréntesis	Expresión	Modificación de la secuencia de ejecución de los operadores.
-	Negación	INT, DINT, UINT, UDINT o REAL	Inversión del signo para el valor del operando.
NOT	Complemento	BOOL, BYTE, WORD, DWORD.	Provoca la inversión del operando por bits.
**	Potenciación	REAL, INT, DINT, UINT, UDINT.	Eleva el valor del primer operando a la potencia del valor del segundo operando.
*	Multiplicación	REAL, INT, DINT, UINT, UDINT.	El valor del primer operando se multiplica por el valor del segundo operando.
/	División	REAL, INT, DINT, UINT, UDINT.	El valor del primer operando se divide entre el valor del segundo operando.
MOD	Módulo	INT, DINT, UINT, UDINT.	El primer operando se divide entre el segundo y el resto de la división se emite como resultado.
+	Adición	INT, DINT, UINT, UDINT, REAL, TIME.	El valor del primer operando se suma al valor del segundo operando.
-	Sustracción	INT, DINT, UINT, UDINT, REAL, TIME.	El valor del primer operando se resta al valor del segundo operando.
<	Menor que	BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DATE	Se compara el valor del primer operando con el valor del segundo operando, si el primero es menor el resultado es un 1 booleano, si no el resultado es un 0 booleano.

Tabla 5.1.4.1 Operadores del lenguaje ST

Mientras que los operandos pueden ser:

- Una dirección.
- Un literal.
- Una variable.
- Una variable de elementos múltiples.
- Un elemento de una variable de elementos múltiples.
- Una llamada de función.
- Una FFB.

Por último, las instrucciones, son los comandos del lenguaje ST, la pieza principal de este tipo de programación.

Puedo tener varias instrucciones en una línea únicamente deben separarse por medio del símbolo de un punto y coma.

Hay varios tipos de instrucciones que seguidamente detallamos:

- Instrucciones de selección:

  - **IF...THEN...END_IF:** La instrucción IF determina que una instrucción o un grupo de instrucciones se ejecute sólo si la expresión booleana correspondiente tiene el valor 1 (verdadero).
    Si la condición es 0 (falso), la instrucción o el grupo de instrucciones no se ejecutará.
    La instrucción THEN marca el final de la condición y el principio de la instrucción o instrucciones.
    La instrucción END_IF señala el final de la instrucción o instrucciones.

  - **ELSE:** sigue siempre a una instrucción *IF…THEN, ELSIF…THEN* o *CASE.* En el caso de seguir a *IF* o *ELSIF* la instrucción se ejecuta solo cuando las expresiones correspondientes tienen valor falso. Si sigue a *CASE* la instrucción solo se ejecuta cuando ninguna marca tiene el valor del selector.
  - **ELSIF…THEN**: sigue siempre a una instrucción *IF…THEN.* Determina que una instrucción o grupo de ellas solo se ejecuta si la expresión booleana correspondiente de la instrucción IF tiene el valor 0 y la expresión booleana correspondiente de la instrucción *ELSIF* tiene el valor 1. En caso contrario la instrucción o grupo de ellas no se ejecuta.
  - **CASE…OF…END_CASE**: está compuesta por una expresión del tipo de datos INT (selector) y una lista de grupos de instrucciones. Cada grupo está provisto de una marca que está compuesta por uno o más números enteros. Se ejecuta el primer grupo de instrucciones cuya marca contenga el valor calculado del selector. De lo contrario, no se ejecuta ninguna de las instrucciones.
    La instrucción *OF* señala el principio de las marcas.
    Dentro de las instrucción *CASE* se puede incluir una instrucción *ELSE* cuyas instrucciones se ejecuten si ninguna marca contiene el valor del selector.
    La instrucción *END_CASE* marca el final de la instrucción o instrucciones.

- Instrucciones de repetición:

  - **FOR…TO…BY…DO…END_FOR**: la instrucción *FOR* se utiliza cuando se puede determinar de antemano la cantidad de repeticiones. De no ser así se utilizan *WHILE* o *REPEAT.*
    La instrucción *FOR* repite una secuencia de instrucciones hasta la instrucción *END_FOR.* La cantidad de repeticiones se determina mediante

el valor inicial, el valor final y la variable de control. Estas variables de control deben ser todas del mismo tipo.

La instrucción *FOR* incrementa el valor de las variables de control desde el valor inicial hasta el final. El valor del incremento está predeterminado a 1, es posible modificar este incremento. El rango de las variables de control se verifica antes de cada nuevo ciclo del bucle. Si éste se encuentra fuera del rango del valor inicial y final, se abandonará el bucle.

La instrucción *DO* marca el final de la definición de repetición y el principio de la instrucción o instrucciones.

La repetición se puede terminar antes de tiempo mediante la instrucción *EXIT*. La instrucción *END_FOR* marca el final de la instrucción o instrucciones.

- **WHILE…DO…END_WHILE**: la instrucción *WHILE* provoca la ejecución repetida de una sección de instrucciones hasta que la expresión booleana correspondiente sea 0, si esta expresión es falsa desde un principio el grupo de instrucciones no se ejecuta en absoluto.

  La instrucción *DO* marca el final de la definición de repetición y el principio de la instrucción o instrucciones.

  La repetición se puede terminar antes de tiempo mediante la instrucción *EXIT*.

  En determinados casos no se puede utilizar esta instrucción ya que se produciría un bucle sin fin que generaría un fallo en la aplicación. Por ejemplo no se puede utilizar para la sincronización entre procesos ni para utilizar en un algoritmo para el que no se pueda garantizar el cumplimiento de la condición final del bucle o la ejecución de una instrucción *EXIT.*
- **REPEAT…UNTIL…END_REPEAT**: la instrucción *REPEAT* provoca la ejecución repetida de una sección de instrucciones hasta que la expresión booleana correspondiente sea 1.

  La instrucción *UNTIL* marca la condición final.

  La repetición se puede terminar antes de tiempo mediante la instrucción *EXIT*.

  La instrucción *END_REPEAT* marca el final de la instrucción o instrucciones.

  En determinados casos no se puede utilizar esta instrucción ya que se produciría un bucle sin fin que generaría un fallo en la aplicación. Por ejemplo no se puede utilizar para la sincronización entre procesos ni para utilizar en un algoritmo para el que no se pueda garantizar el cumplimiento de la condición final del bucle o la ejecución de una instrucción *EXIT.*
- **EXIT**: esta instrucción se utiliza para finalizar las instrucciones de repetición antes de que se cumpla la condición final.

  Si esta instrucción se encuentra dentro de una repetición imbricada, se abandonará el bucle interno. A continuación se ejecutará la primera instrucción después del final de bucle.

- Llamada de subrutina: esta llamada está compuesta por el nombre de la sección de la subrutina y una lista de parámetros vacía. Estas llamadas no devuelven ningún valor.
  La subrutina invocante debe de estar en la misma tarea que la sección ST invocante. También es posible llamar subrutinas ubicadas dentro de otras subrutinas.

- **RETURN**: cada subrutina y bloque de función derivado se abandona de nuevo después de su procesamiento, es decir, se efectúa un retorno a la rutina principal. Si se deben abandonar antes de tiempo se puede forzar el retorno a la subrutina principal mediante *RETURN.*
  No es posible utilizar esta instrucción en la rutina principal.

- Instrucción vacía: el mismo nombre indica su funcionalidad, se represente por medio de un punto y coma.

- Etiquetas y saltos: las etiquetas sirven como punto de destino de los saltos.
  Las etiquetas deben ser el primer elemento de una fila, ser unívocas y cumplir la nomenclatura general.
  Los saltos únicamente se pueden realizar dentro de la sección actual.

- Comentario: comienzan con la cadena de caracteres (* y terminan con *). Entre estas dos cadenas se puede introducir cualquier comentario. Se pueden introducir en cualquier posición del editor ST.

Todos estos también se manejan de manera gráfica sobre la ventana del editor seleccionándose fácilmente desde la barra de herramientas del mismo, cuyo aspecto vemos a continuación.

Figura 5.1.4.8 Barra de herramientas del editor ST

5.1.5. Tabla de animación

Mediante el explorador de proyectos los pasos a seguir para generar nuestra tabla de animación son muy sencillos e intuitivos.

Comenzamos seleccionando desde el propio explorador la pestaña correspondiente a tablas de animación, seguidamente aparece el menú desplegable correspondiente donde mediante la pestaña "Nueva tabla de animación" se genera la tabla en si. Posteriormente aparece la ventana de propiedades de la tabla, donde especificamos el nombre y opcionalmente un breve comentario de la misma si se desea, tras este paso la tabla está generada y guardada bajo el nombre que hemos elegido, en nuestro caso al tratarse de la única tabla de la aplicación no se le ha asignado ningún nombre específico.

Como siguiente paso deberemos ir incluyendo las diferentes variables que deseemos animar posteriormente durante la simulación, se pueden ir introduciendo variable tras variable o simplemente copiarlas a partir de la tabla de variables del programa generada en pasos previos de la aplicación, esta última opción mucho más rápida que la anterior.

Así hasta conseguir completar la tabla con las variables deseadas, para su posterior animación durante la simulación o visualización durante la ejecución del programa, que vemos a continuación en la figura.

Modificación | Forzar

Nombre	Valor	Tipo	Comentario
AAA	0	BOOL	Todas las válvulas abiertas
AAA_defecto	1	BOOL	Variable correspondiente al estado de defecto(todas...
AAC	0	BOOL	Válvula derecha cerrada
ACA	0	BOOL	Válvula central cerrada
ALARMA_LED	0	EBOOL	Boton encendido led alarma
CAA	0	BOOL	Válvula izquierda cerrada
CERO	0	BOOL	0 lógico
CIEN	100.0	REAL	
Desague_c_a	0	BOOL	Desague central abierto
Desague_c_c	0	BOOL	Desague central cerrado
Desague_d_a	0	BOOL	Desague derecho abierto
Desague_d_c	0	BOOL	Desague derecho cerrado
Desague_i_a	0	BOOL	Desague izquierdo abierto
Desague_i_c	0	BOOL	Desague izquierdo cerrado
Dif_nivel12	0	INT	
Dif_nivel21	0	INT	
error_sonda	0	BOOL	Variable booleana correspondiente a un error en la s...
Estado_motor_1	0	BOOL	
Estado_motor_2	0	BOOL	
Igual12	1	BOOL	
Igual21	1	BOOL	
M0	1	EBOOL	Estado inicial
M1_1	0	EBOOL	Estado de funcionamiento normal tanque 1
M1_2	0	EBOOL	Estado de funcionamiento normal tanque 2
M2	0	EBOOL	Estado de emergencia
M3	0	EBOOL	Estado tanque lleno
M4_1	0	EBOOL	Estado rellenado tanque 1
M4_2	0	EBOOL	Estado rellenado tanque 2

Figura 5.1.5.1 Tabla de animación de la aplicación

Estas variables cambiaran durante la ejecución del programa según el comportamiento definido para cada una.

En la modalidad de simulación, los valores de estas variables se podrán modificar o forzar para comprobar los diferentes comportamientos del programa desarrollado.

La operación de forzado únicamente se lleva a cabo sobre variables alocatadas. Para ello es necesario activar el botón de forzado presente en la barra de herramientas de la ventana de animación durante su simulación.

Este forzado consiste en modificar el estado de la variable, siempre hablando de variables digitales, en el caso de tratarse de variables analógicas el valor no será posible modificarlo durante la simulación, manteniéndose en el valor que poseyera la variable al ser generada, si se desea modificar éste será necesario detener la simulación y variar a voluntad el valor de la variable para después iniciar de nuevo la simulación con este nuevo valor.

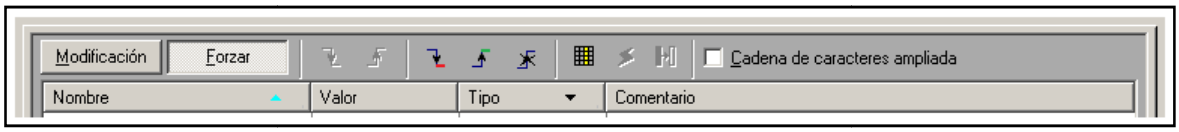


Figura 5.1.5.2. Barra de herramientas de la animación durante el forzado

La operación de modificación se lleva a cabo en variables no alocatadas, variables internas, no se habla de forzado sino de modificación para este tipo de datos. Al igual que en el caso del forzado se activará el botón de modificación de la barra de herramientas durante la simulación, a continuación vemos esta barra de herramientas.

Figura 5.1.5.3. Barra de herramientas de la animación durante la modificación

5.1.6. Pantallas del operador

La pantalla/as del operador se generan nuevamente a partir del explorador de proyectos; al igual que con la tabla de animación la nueva pantalla del operador se crea pulsando en la pestaña que lleva ese mismo nombre, aparecerá el menú desplegable correspondiente y presionando sobre la sección "Nueva pantalla del operador" se crea la pantalla. De nuevo tras esto aparecerá la ventana de propiedades donde asignaremos nombre, localización y comentarios bajo los cuales se guardará a partir de ahora, de nuevo no asignamos ningún nombre específico ya que va a ser la única pantalla disponible en la aplicación.

Después de este primer paso nos encontramos con una pantalla básica y vacía, la cual vamos modelando y añadiendo elementos hasta conseguir el aspecto y la funcionalidad requerida para la aplicación que desarrollamos.

Los elementos y figuras que se deseen incluir se pueden crear manualmente mediante la barra de herramientas del editor, la cual vemos a continuación en la figura que sigue.

Figura 5.1.6.1. Barra de herramientas del editor

Donde cada uno de estos iconos de la barra de herramientas posee una función que a continuación resumimos en la siguiente ilustración.

Icono	Función
	Selección
	Línea
	Rectángulo
	Elipse
	Curva
	Polígono
Aa	Texto
	Imagen
	Examinar pantalla
12	Campo de entrada
	Botón de comando
	Casilla de verificación
	Cuadro de giro
	Indicador de escala
	Intercambio explícito
	Habilitar variables de escritura

Figura 5.1.6.2. Detalle de los iconos de la barra de herramientas

O bien se pueden escoger desde una librería de elementos prediseñados que posee el programa, donde encontraremos multitud de opciones y de objetos de distinta apariencia y con la funcionalidad atribuida con anterioridad por su diseñador, en la pestaña de herramientas con la denominación "Librería de pantallas del operador".

Podemos ver el aspecto de esta librería, las distintas carpetas de que dispone y algunos de sus elementos, a modo de ejemplo, a continuación en la figura:

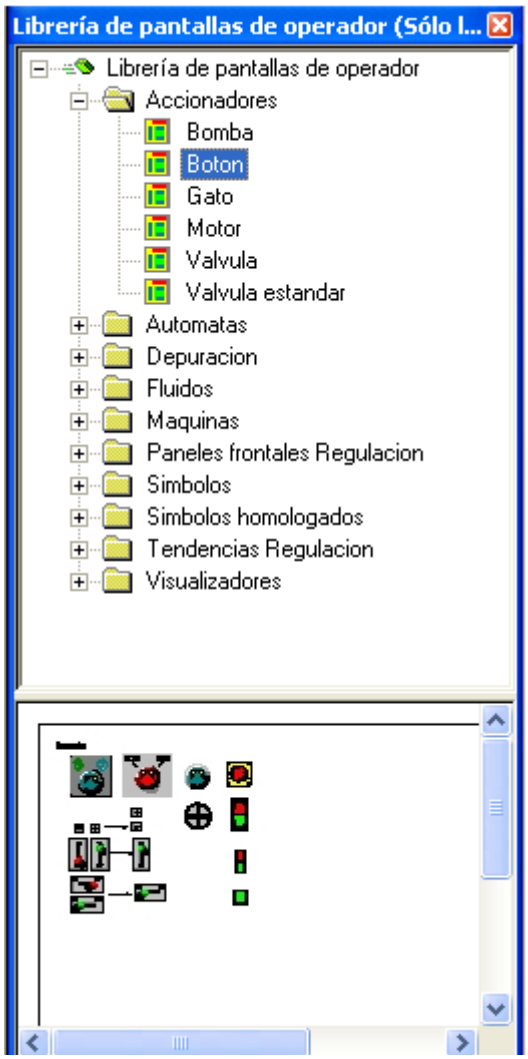


Figura 5.1.6.3 Librería de pantallas del operador

Usando cualquiera de estos dos métodos iremos introduciendo los distintos elementos, a cada uno de ellos se le asignará la variable que le corresponda y la visualización adecuada en cada caso de manera que durante el proceso de ejecución de la aplicación la pantalla tenga un funcionamiento acorde con la aplicación real. Concretamente en nuestro caso se han utilizado ambos métodos, realizando elementos propios mediante la barra de herramientas y escogiendo de la librería otros

que cumplían con los requisitos concretos de la aplicación, así hasta generar las pantallas que podemos ver en las siguientes imágenes.

La primera de las imágenes se corresponde con la pantalla de instrucciones para el operador, la cual simplemente establece una serie de pautas que el operador debe seguir para un correcto funcionamiento de la aplicación.

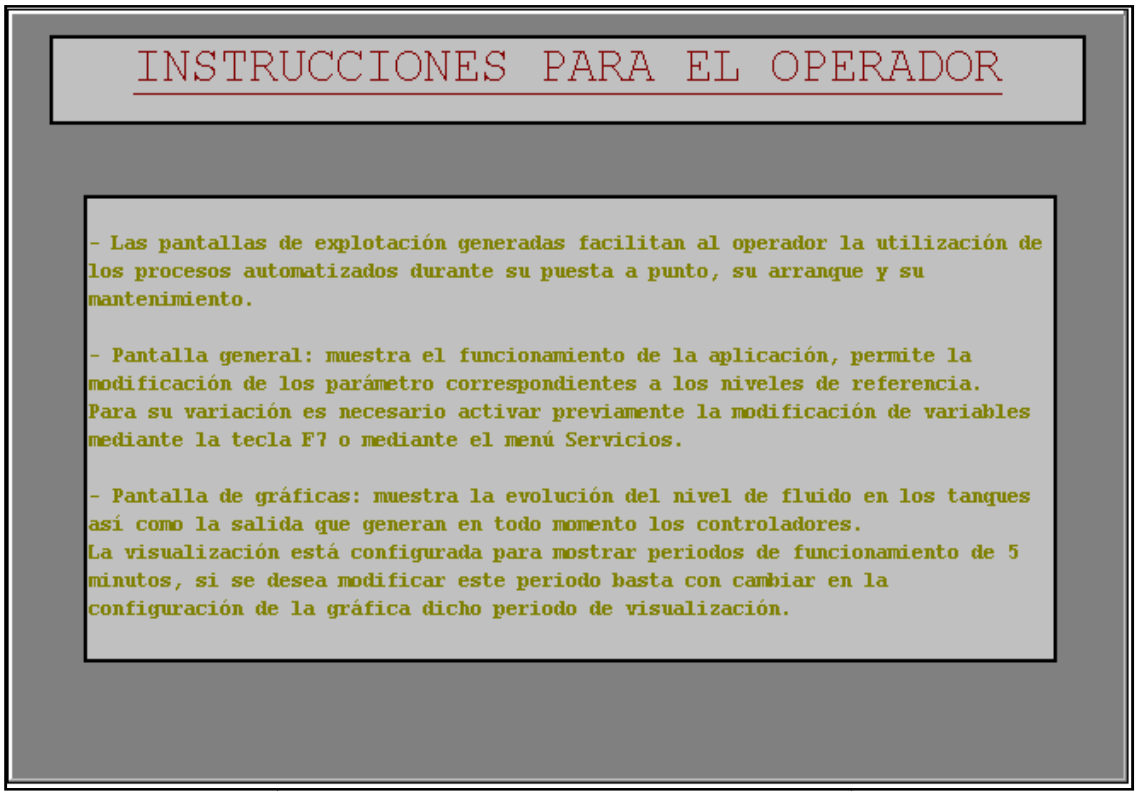


Figura 5.1.6.4. Pantalla del operador inicial

La siguiente figura, llamada pantalla principal, contiene el panel de control con la botonería y LEDs correspondientes, el gráfico del sistema de tanques donde podemos observar la variación de nivel en los tanques y el estado de las válvulas, el bloque que permite el ajuste del nivel de referencia para el nivel de fluido en los tanques y un último bloque que representa el porcentaje de funcionamiento de las bombas.

Cuando el autómata esté ejecutando el programa, observaremos gráficamente, a través de los elementos incluidos en la pantalla, el comportamiento de las variables representadas.

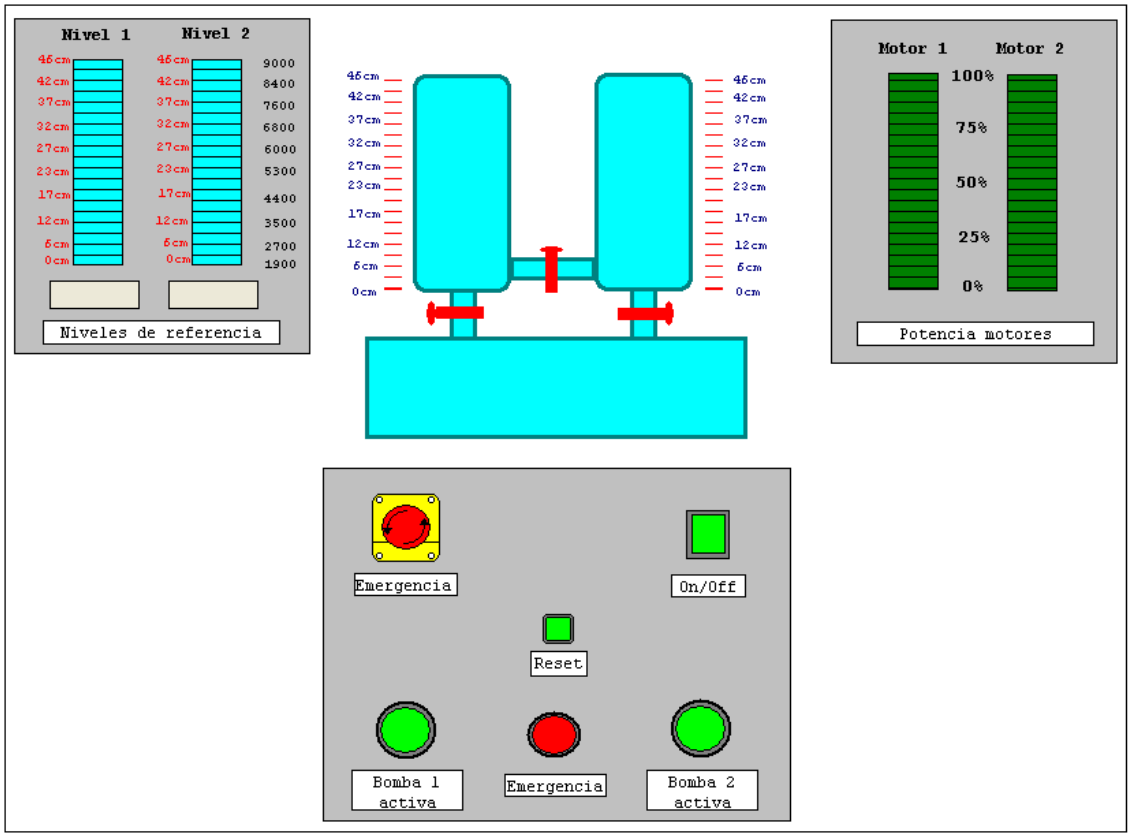


Figura 5.1.6.5 Pantalla del operador de la aplicación

La última de las figuras se corresponde con la ventana de gráficas de tendencias, en ella observamos de manera continua el progreso de nivel del líquido en los dos tanques así como la señal de control que generan los dos controladores

Estas gráficas se pueden utilizar para comprobar funcionamientos anteriores de la aplicación ya que poseen un periodo de visualización variable que puede ajustar el operario a voluntad según el requerimiento que sea necesario.

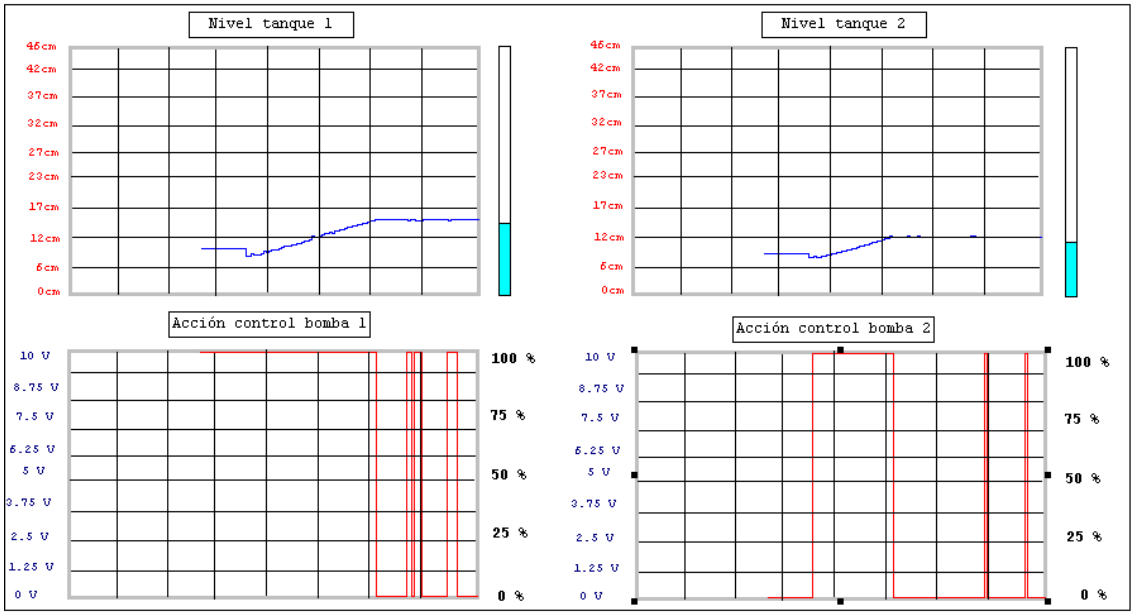


Figura 5.1.6.6 Pantalla de gráficas de tendencias

5.1.7. Generación del proyecto

Una vez terminado el desarrollo de la aplicación se deben llevar a cabo una serie de pasos con el fin de inspeccionar el programa y verificar que todo se ha desarrollado correctamente y no existen errores dentro de éste.

El primer paso consistirá en analizar el proyecto para verificar que no existen errores ni advertencias. Este proceso se lleva a cabo a través del propio icono de análisis (Véase Figura 5.1.7.1.) o a partir del menú "Generar" de la barra de herramientas y la posterior pestaña "Analizar proyecto".

Figura 5.1.7.1. Icono del botón de análisis de la aplicación

En todo momento se puede verificar el estado del proyecto en Unity Pro mediante un recuadro en la parte inferior izquierda acompañado de una bombilla. Durante el desarrollo del programa nos muestra el mensaje "No generado", tras este primer análisis el proyecto pasa a estar "Analizado".

Tras este análisis lo siguiente es generar el proyecto, en este proceso se genera el código que posteriormente transferiremos al autómata. De nuevo podemos hacerlo de forma directa mediante el icono correspondiente (Véase Figura 5.1.7.2.) o a través del menú "Generar" y la pestaña "Regenerar todo el proyecto".

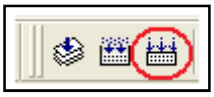


Figura 5.1.7.2. Icono correspondiente al botón "Regenerar todo el proyecto"

Siempre que el proceso se complete correctamente el estado del proyecto pasará a ser "Generado", como veremos en el recuadro inferior de la pantalla. Y la ventana de resultados nos indicará que el proceso has sido satisfactorio como vemos en el ejemplo de la siguiente figura.

Figura 5.1.7.3. Ventana de resultados

El proceso que hemos visto se deberá realizar de nuevo siempre que se produzca modificación alguna en el proyecto, si no se desea regenerar todo el proyecto nuevamente existe también la posibilidad de generar únicamente los cambios realizados. Para ello tenemos el icono que resta en las figuras anteriores o de nuevo a través del menú “Generar” y la pestaña “Generar cambios”.

Figura 5.1.7.4. Icono correspondiente al botón “Generar cambios”

5.1.8. Conexión con el autómata

El último paso es volcar la aplicación desarrollada dentro del autómata. Hasta ahora hemos trabajado desconectados de éste y lo siguiente será tratar de establecer una conexión con el autómata.

Se pueden llevar a cabo dos modalidades de conexión con el PLC:

- Modalidad estándar; conectamos físicamente el ordenador con el autómata mediante cable USB.
- Modalidad simulación; conectamos con el simulador de Unity Pro.

La selección de una u otra se lleva a cabo en el menú "PLC" y mediante la pestaña "Modalidad estándar" o "Modalidad de simulación" según cual sea la requerida o a partir de los iconos de la barra de herramientas que vemos a continuación en la imagen.

Figura 5.1.8.1. Iconos correspondientes a los botones de modalidad "Estándar" y "Simulación"

Nos vamos a fijar en los pasos necesarios para establecer la conexión física con el autómata. Como precaución antes de conectar podemos realizar una comprobación de la conexión que nos determinará si el cable USB está correctamente conectado con la dirección establecida al autómata. Si este paso es correcto podremos realizar la conexión, de nuevo en el menú "PLC" elegiremos la pestaña "Conectar" o si se prefiere mediante el icono correspondiente que se muestra en la siguiente figura.

Figura 5.1.8.2. Icono correspondiente al botón "Conectar"

Una vez llevado a cabo todo esto la conexión se ha realizado con éxito, podemos observar que en la barra inferior izquierda ha desaparecido ya el modo "*OFFLINE*" y la conexión es correcta.

5.1.9. Transferencia del proyecto al autómata

El proyecto generado siempre se ha de transmitir al autómata independientemente de la modalidad de conexión utilizada.

La transferencia se inicia dentro del menú "PLC" en la pestaña "Transferir proyecto a PLC" o haciendo clic en el icono de la barra de herramientas correspondiente que vemos a continuación en la imagen.

Figura 5.1.9.1. Icono correspondiente al botón "Transferir"

A continuación se nos muestra una ventana en la que se define la dirección de origen y la de llegada del proyecto y se nos pide confirmación para realizar la transferencia.

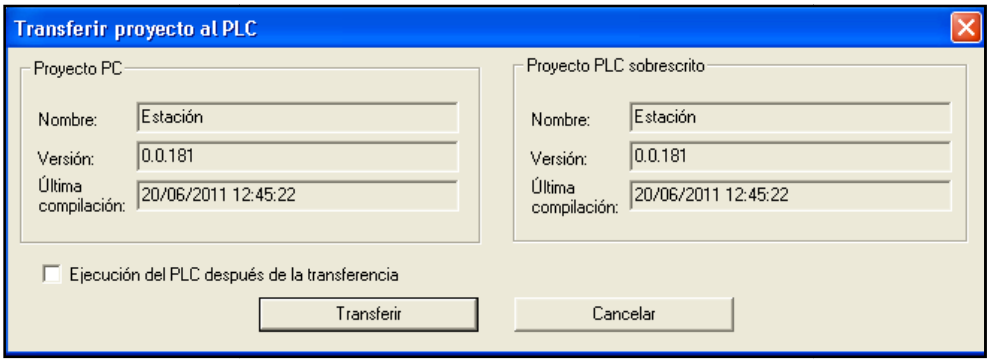


Figura 5.1.9.2. Confirmación de la transferencia

Tras la confirmación comienza el proceso de transferencia, dependiendo de la extensión de la aplicación este proceso podrá durar más o menos tiempo, veremos una ventana que nos mostrará el estado de dicha transferencia.

En el caso de que no hubiéramos generado el proyecto previamente se nos requerirá que lo hagamos antes de comenzar la transferencia mediante ventana emergente que nos permitirá regenerar y transferir posteriormente en un solo paso.

Una vez realizada la transferencia tendremos el mismo proyecto en el autómata y en el ordenador, lo podremos observar nuevamente en la barra inferior izquierda donde aparecerá sobre fondo verde el estado "*IGUAL*", hasta entonces podría haber tenido cualquier otro estado de entre los posibles que relatamos en la siguiente tabla.

Estado	*Descripción*
DIFERENTE	El programa generado no coincide con el del PLC.
IGUAL	El programa generado y el del PLC coinciden.
STOP	PLC parado.
RUN	PLC en modo ejecución.

Tabla 5.1.9.1. Estados posibles

5.1.10. Ejecución

Con el proyecto transferido al PLC el último paso es poner el autómata en modo ejecución para que de esta manera se realice toda la programación generada con anterioridad.

El paso a modo ejecución se puede llevar a cabo mediante el icono propio (Véase Figura 5.1.10.1) o desde el menú “PLC” y la pestaña “Ejecutar”.

Figura 5.1.10.1. Icono correspondiente al botón “Ejecutar”

El otro icono activado indica la ejecución cíclica del programa una vez pulsado el botón de ejecución.

De manera similar detenemos el proceso de ejecución mediante el icono que le sigue al anterior o en el menú “PLC” con la pestaña “Detener”.

Figura 5.1.10.2. Icono correspondiente al botón “Detener”

5.2. Secciones programadas

Las diferentes secciones del programa se definen usando únicamente el tipo de tarea *MAST,* esta tarea maestra representa la tarea principal del programa de aplicación. Es obligatoria y se crea de forma predeterminada.

La tarea *MAST* se compone de secciones y subrutinas, en sus lenguajes correspondientes, que describiremos a continuación.

La ejecución de esta tarea se puede elegir por parte del programador, siendo las opciones posibles ejecución cíclica o ejecución periódica.

A continuación vemos en la figura la ventana de propiedades y configuración general de la tarea *MAST*.

Figura 6.2.1. Propiedades de la tarea MAST

Seguidamente detallaremos la funcionalidad y funcionamiento de cada una de las secciones que componen nuestra aplicación.

5.2.1. Principal

Se trata de la primera sección en ejecutarse tras arrancar la aplicación (durante el primer ciclo después de la puesta en marcha), se denomina estado *M0*.

Actualiza los valores de las entradas digitales y analógicas, convirtiendo estas últimas en variables booleanas para su tratamiento en forma de contactos durante el resto del programa y da servicio al resto de secciones de la aplicación en función de las entradas activadas y el estado del sistema.

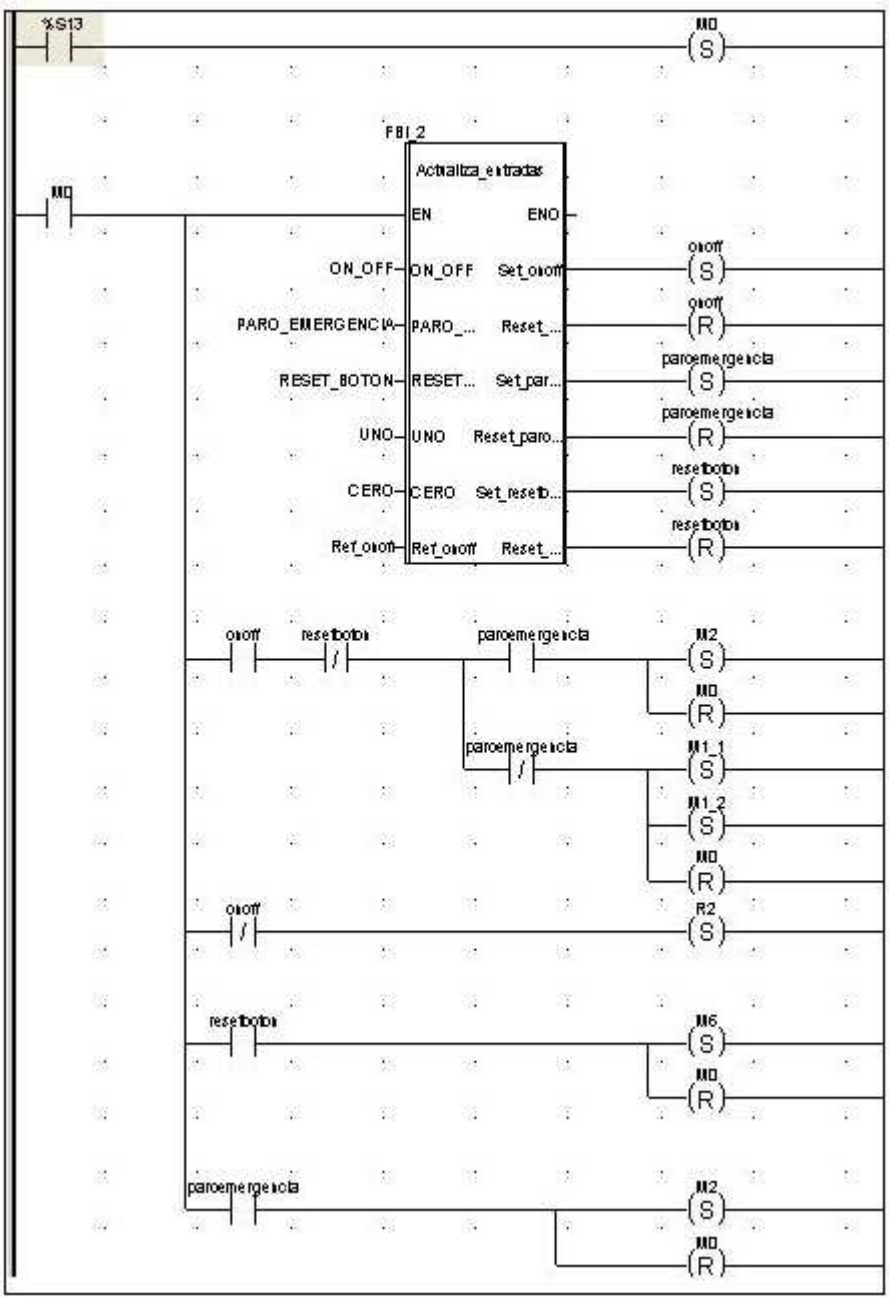


Figura 5.2.1.1 Sección 'Principal'

5.2.2. Funcionamiento 1

Llamaremos a este estado *M1_1* o de funcionamiento normal para el tanque 1, programado en lenguaje LD, a continuación vamos a definir el funcionamiento del mismo.

Este estado se activa desde la sección Principal siempre que durante la anterior no se produjera la activación de la entrada de emergencia ni tampoco la de la entrada de reset externo.

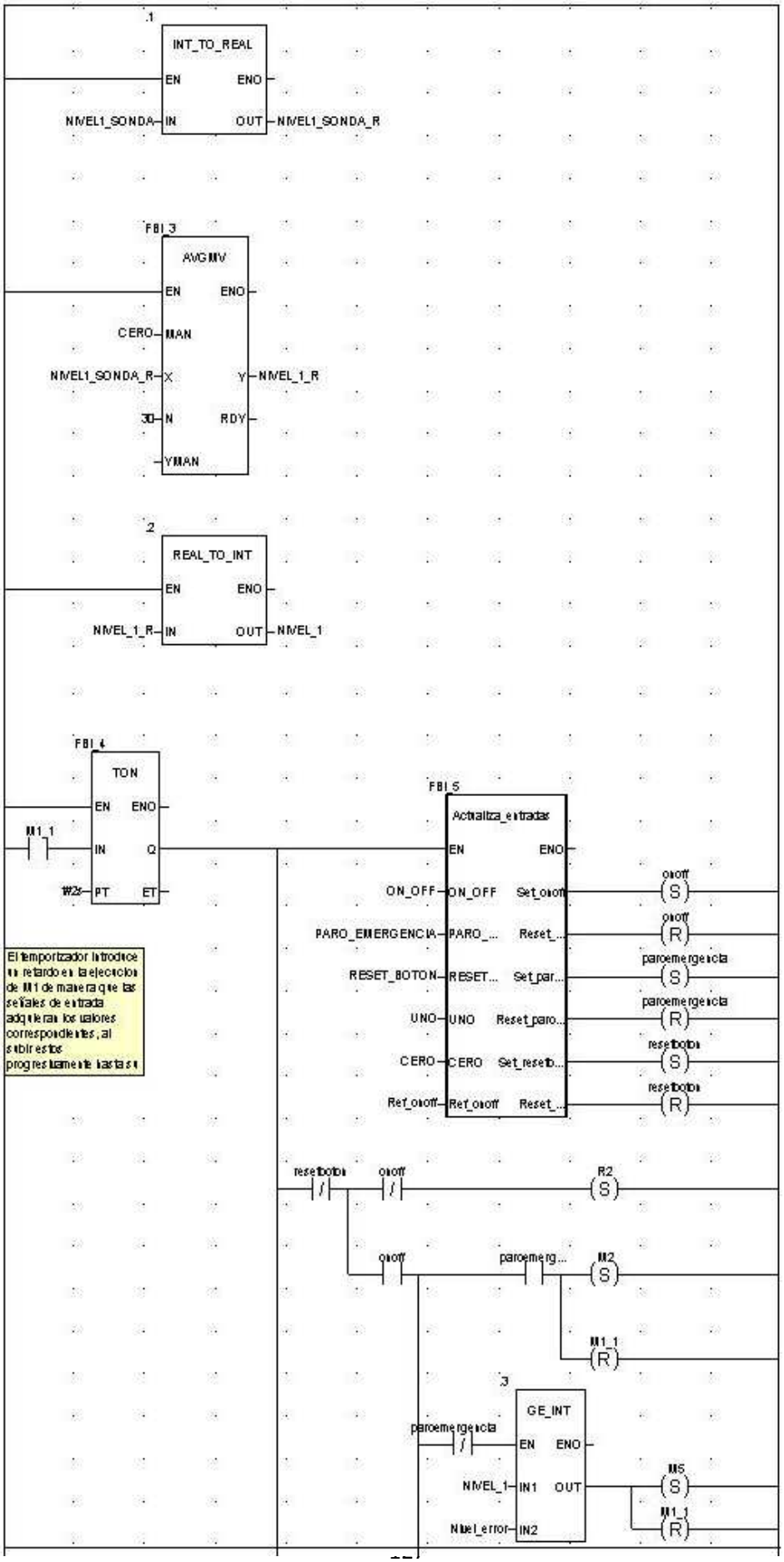
La función de esta sección es activar el siguiente estado correspondiente en función del nivel de líquido presente en el tanque 1. Esta comprobación se realiza mediante distintos bloques de funciones, los cuales detectarán si nos encontramos ante una medida errónea provocada por una mala lectura de la sonda, si el nivel del tanque se encuentra por encima del nivel máximo de funcionamiento o por último si el nivel se encuentra por debajo del nivel máximo de referencia y es necesario el rellenado del tanque.

Al inicio de la sección realizamos un filtro de las medidas que toma la sonda de nivel, esto será necesario para posteriores secciones del programa.

Hemos colocado un bloque temporizador, cuyo cometido es introducir un retardo en el inicio del estado. Esto es necesario ya que la adquisición de la medida de nivel en el tanque es progresiva y debemos tomar esta medida cuando haya alcanzado su valor más alto, de lo contrario estaríamos cometiendo errores que podrían dar paso a estados incorrectos de funcionamiento.

En esta sección también se incluye el bloque actualizador de entradas, por si las mismas cambiasen durante la ejecución de este estado. De la misma forma podemos ver la lógica de activación de diferentes estados en función del estado de las diferentes entradas.

A continuación podemos ver la estructura de esta sección según lo explicado.


1
INT_TO_REAL
EN
ENO
NIVEL1_SONDA
IN
OUT
NIVEL1_SONDA_R
FBI_3
AVGMV
EN
ENO
CERO
MAN
NIVEL1_SONDA_R
X
Y
NIVEL_1_R
30
N
RDY
YMAN
2
REAL_TO_INT
EN
ENO
NIVEL_1_R
IN
OUT
NIVEL_1
FBI_4
TON
EN
ENO
M1_1
IN
Q
#2s
PT
ET
El temporizador introduce un retardo en la ejecución de M1 de manera que las señales de entrada adquieran los valores correspondientes, al subir estos progresivamente hasta su
FBI_5
Actualiza_entradas
EN
ENO
ON_OFF
ON_OFF
Set_onoff
onoff
PARO_EMERGENCIA
PARO_...
Reset_...
onoff
RESET_BOTON
RESET...
Set_par...
paroemergencia
UNO
UNO
Reset_paro...
paroemergencia
CERO
CERO
Set_resetb...
resetboton
Ref_onoff
Ref_onoff
Reset_...
resetboton
resetboton
onoff
R2
onoff
paroemerg...
M2
M1_1
3
GE_INT
paroemergencia
EN
ENO
NIVEL_1
IN1
OUT
M5
Nivel_error
IN2
M1_1

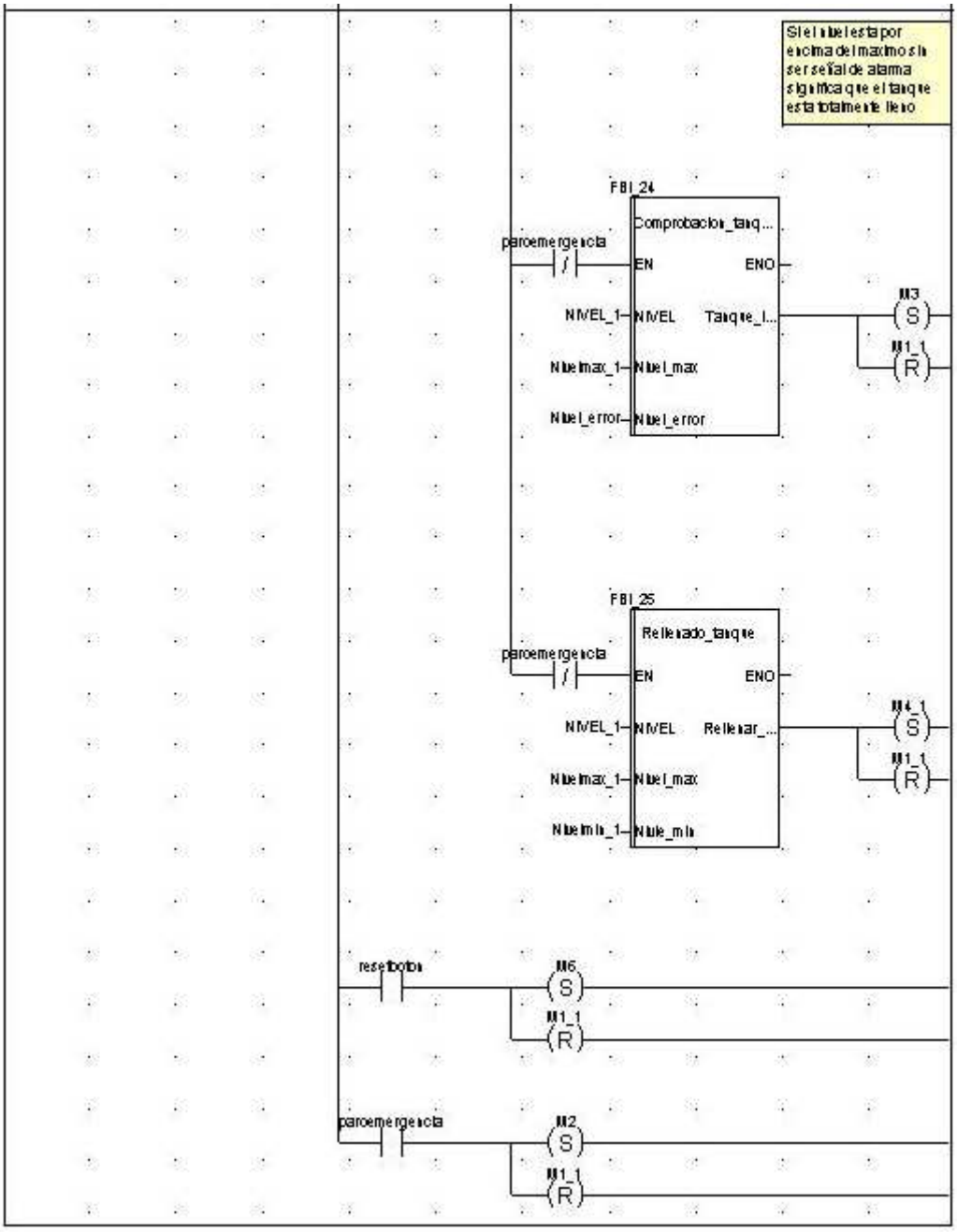


Figura 5.2.2.1 Sección 'Funcionamiento 1'

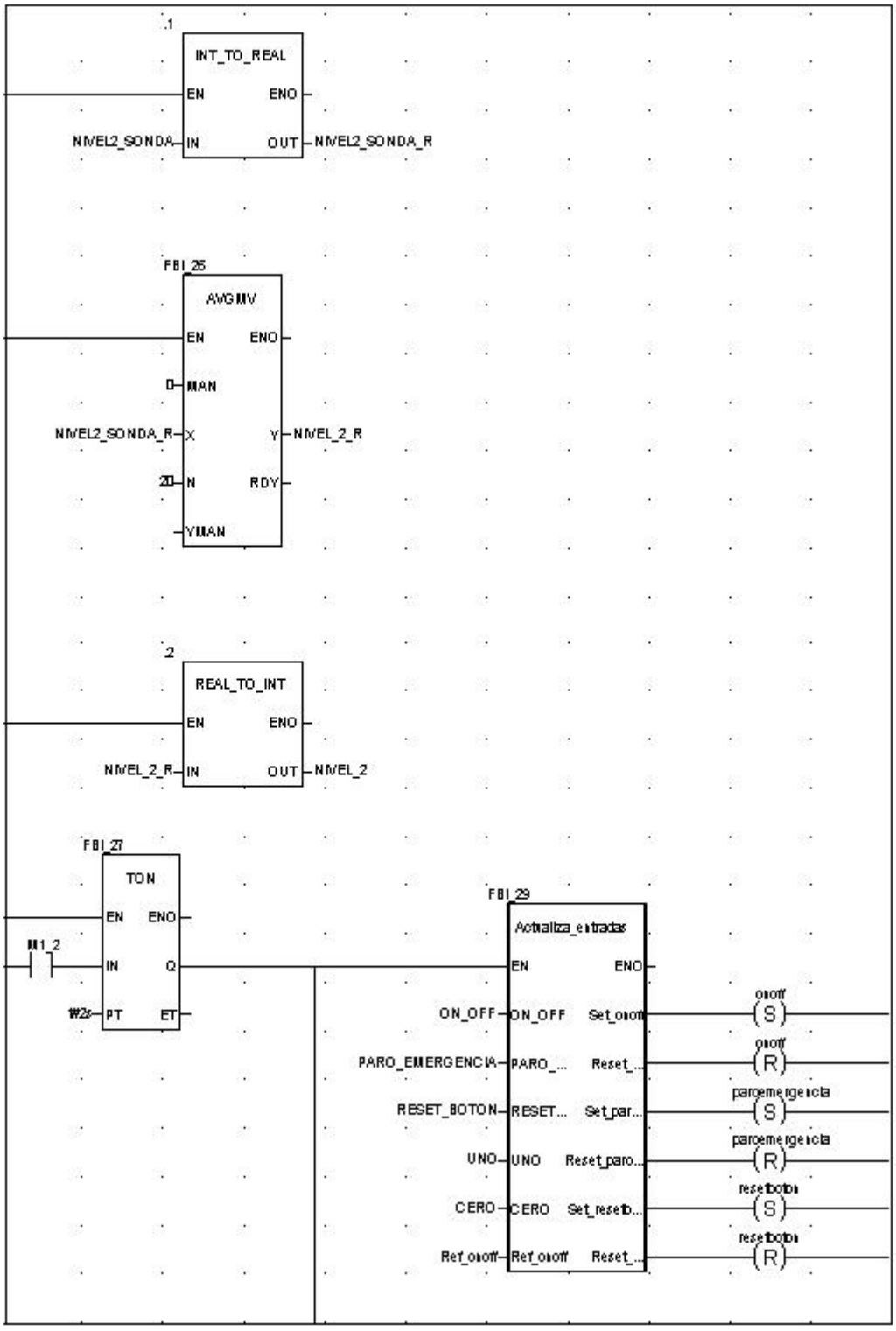
5.2.3. Funcionamiento 2

El funcionamiento y la estructura de este estado al que llamaremos M1_2, es exactamente igual que el de la sección anterior (M1_1) pero referido al tanque 2.

De la misma forma se activa desde la sección Principal siempre que durante la anterior no se produjera la activación de la entrada de emergencia ni tampoco la de la entrada de reset externo.

También incluye el bloque de actualización de las entradas y la lógica necesaria para activar los correspondientes estados en función del estado de las mismas, así como el filtro par el nivel obtenido mediante la sonda.

En las figuras siguientes se muestra la programación de esta sección.


.1
INT_TO_REAL
EN
ENO
NIVEL2_SONDA
IN
OUT
NIVEL2_SONDA_R
FBI_26
AVGMV
EN
ENO
0
MAN
NIVEL2_SONDA_R
X
Y
NIVEL_2_R
20
N
RDY
YMAN
.2
REAL_TO_INT
EN
ENO
NIVEL_2_R
IN
OUT
NIVEL_2
FBI_27
TON
EN
ENO
M1_2
IN
Q
W2s
PT
ET
FBI_29
Actualiza_entradas
EN
ENO
ON_OFF
ON_OFF
Set_onoff
onoff
S
PARO_EMERGENCIA
PARO_...
Reset_...
onoff
R
RESET_BOTON
RESET...
Set_par...
paroemergencia
S
UNO
UNO
Reset_paro...
paroemergencia
R
CERO
CERO
Set_resetb...
resetboton
S
Ret_onoff
Ret_onoff
Reset_...
resetboton
R

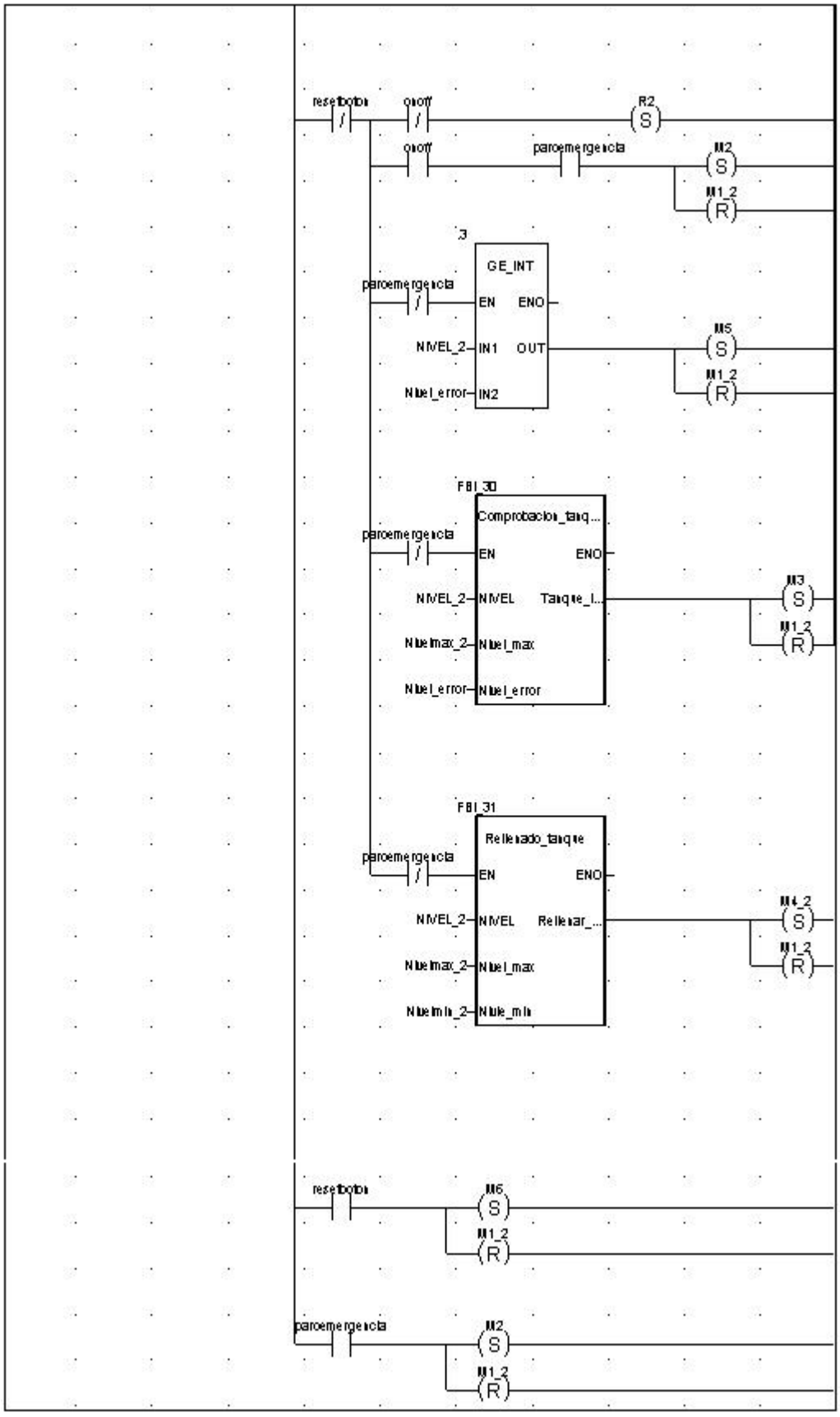


Figura 5.2.3.1 Sección 'Funcionamiento 2'

5.2.4. Emergencia

Esta sección de programa es la llamada sección de emergencia o estado M2, programada el lenguaje LD o de contactos.

Este estado se activa únicamente cuando se pulsa el botón de emergencia situado en el panel de control, activando la variable digital PARO_EMERGENCIA.

El cometido de esta sección es, al activar el botón de emergencia, poner en marcha un proceso de reset por emergencia (detallado en la sección de reseteos) y activar el LED de emergencia de color rojo que posee el panel de control.

También incluye el bloque de actualización de las entradas.

Su estructura de programación la podemos ver a continuación.

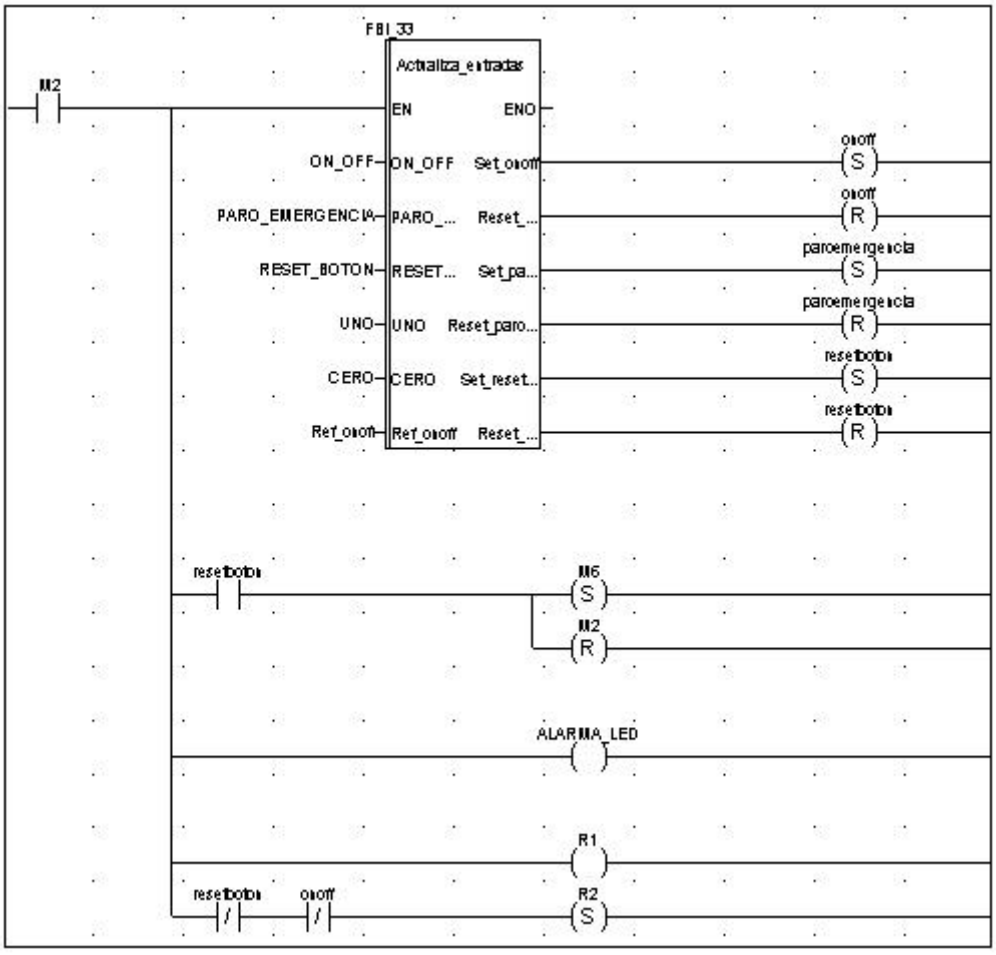


Figura 5.2.4.1 Sección 'Emergencia'

5.2.5. Tanque lleno

Esta sección se corresponde con el estado M3, programado en LD.

Este estado se activa desde las secciones funcionamiento 1 o funcionamiento 2, cuando el bloque que se encarga de determinar si el nivel en el tanque está por encima del nivel de referencia sin ser señal una señal de error de la sonda proporciona una salida positiva.

Esta sección es un estado de reposo, la cual se limita a dar servicio a la sección principal cada vez que se activa, comprobando así si el nivel de los tanques ha sufrido variaciones y debe dejar paso a otros estados de funcionamiento.

Al igual que los estados anteriores incluye el bloque actualizador de entradas y la lógica para la activación de los diferentes estados en función del estado de las entradas. Podemos ver su programación a continuación.

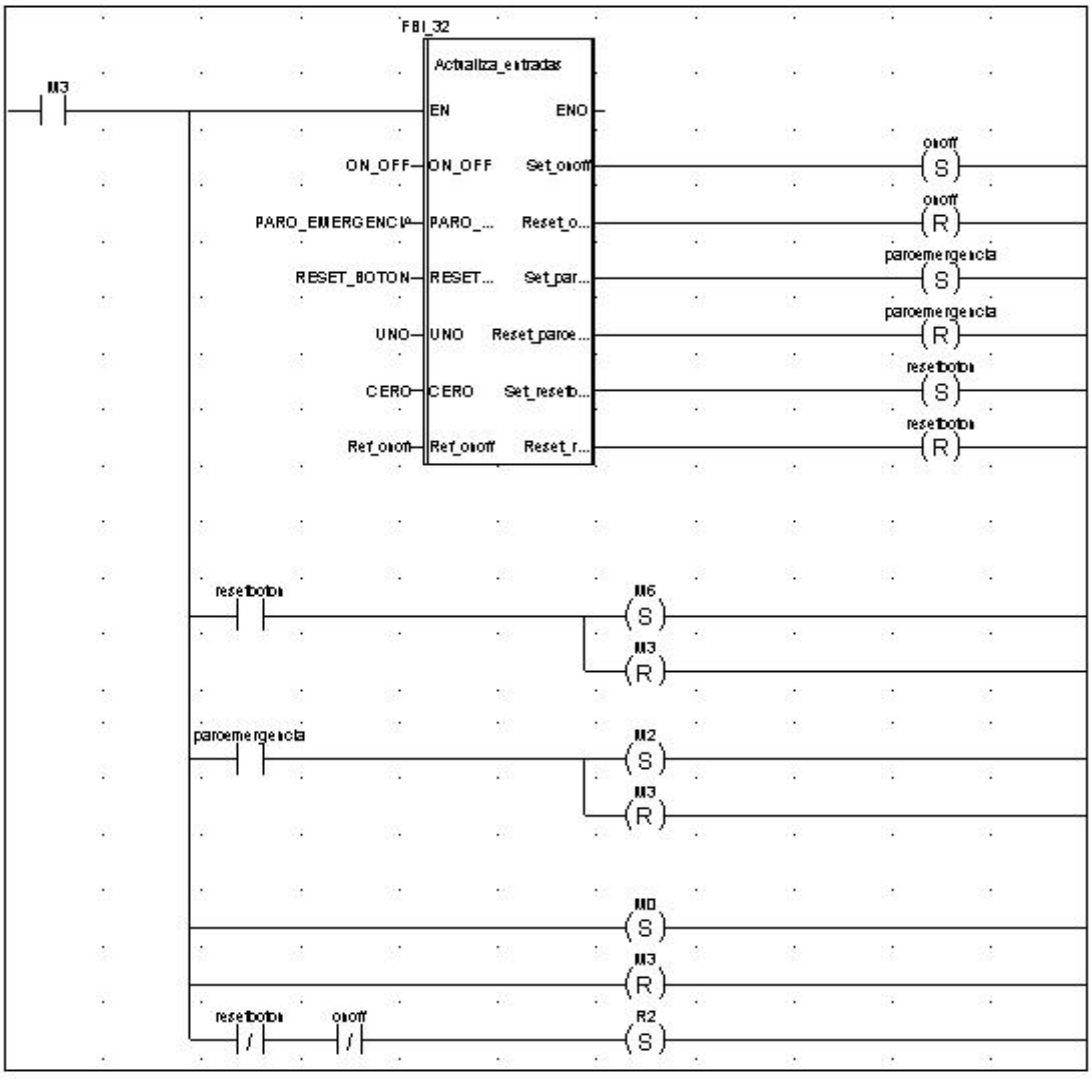


Figura 5.2.5.1 Sección 'Tanque lleno'

5.2.6. Llenado tanque 1

Este estado, denominado M4_1 es el más importante del programa junto con el correspondiente al tanque 2 (M4_2), ambos programados en lenguaje LD.

Este estado se activa desde la sección funcionamiento normal 1 cuando el bloque encargado de determinar si el nivel que presenta el tanque 1 está por debajo del nivel de trabajo seleccionado y por encima del nivel mínimo que es capaz de medir la sonda.

La función de este es la de rellenar el tanque hasta alcanzar el nivel de referencia. Para ello hemos implementado un bloque DFB llamado '*Controlador*', el cual tiene como entradas el nivel de referencia a la entrada que, comparándolo con el nivel actual del proceso, genera la señal de control que almacenamos en la variable **Salida_controlador,** la cual determina la potencia necesaria que suministraremos a la bomba de cada tanque. Este controlador está diseñado en lenguaje ST para poder implementar su función como una ecuación en diferencias de forma que la salida se calcule a partir de los valores de las entradas y salidas en anteriores instantes de tiempo.

A continuación, mostramos el programa del controlador:

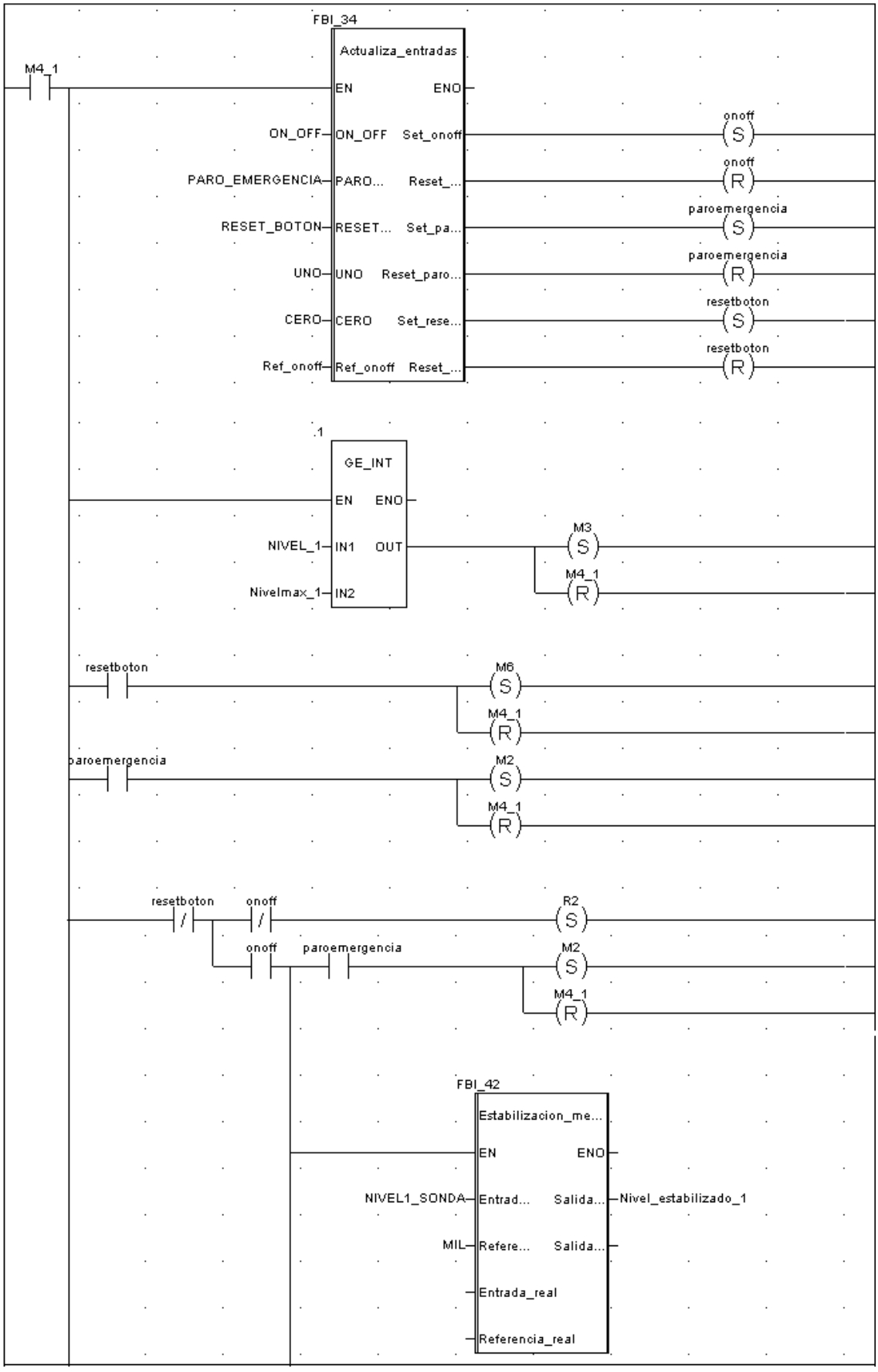
```
n:=0;

FOR n:=0 TO 1000 DO
        y[n]:=0.0;
        x[n]:=0.0;
END_FOR;

FOR n:=0 TO 1000 DO
        IF n=0 THEN
                y[n-1]:=0.0;
                y[n-2]:=0.0;
                x[n-1]:=0.0;
                x[n-2]:=0.0;
        END_IF;

        IF n=1 THEN
                y[n-1]:=0.0;
                y[n-2]:=0.0;
                x[n-2]:=0.0;
                x[n-1]:=Nivel max R;
        END_IF;

        IF n=2 THEN
                y[n-2]:=0.0;
        END_IF;
        IF n>=2 THEN
                x[n-1]:=Nivel max R;
                x[n-2]:=Nivel max R;
        END_IF;

        y[n]:=1.8869*y[n-1]-0.8869*y[n-1]+196.3*x[n-1]-196.277*x[n-2];

        IF y[n]>Limite THEN
                y[n]:=Limite;
        END_IF;

        Salida controlador:=y[n];
END_FOR;
```

Figura 5.2.6.1 Programa del controlador

Para el correcto funcionamiento del controlador es necesario realizar un filtro para el nivel de líquido en el tanque y la salida del controlador, mediante el bloque ***Estabilizacion_medidas*** diseñado para tal fin, de esta forma evitamos los picos que se generan en la señal de control provocados por las variaciones rápidas y oscilantes que se producen al efectuarse la lectura del nivel por parte de la sonda.

Se incluye, como en el resto de estados, el bloque de actualización de variables y lógica de activación de los distintos estados en función del estado de las entradas.

A continuación se muestra la estructura de programación de esta sección:


FBI_34
Actualiza_entradas
M4_1
EN
ENO
ON_OFF
ON_OFF
Set_onoff
onoff
PARO_EMERGENCIA
PARO...
Reset_...
onoff
RESET_BOTON
RESET...
Set_pa...
paroemergencia
UNO
UNO
Reset_paro...
paroemergencia
CERO
CERO
Set_rese...
resetboton
Ref_onoff
Ref_onoff
Reset_...
resetboton
.1
GE_INT
EN
ENO
NIVEL_1
IN1
OUT
M3
M4_1
Nivelmax_1
IN2
resetboton
M6
M4_1
paroemergencia
M2
M4_1
resetboton
onoff
R2
onoff
paroemergencia
M2
M4_1
FBI_42
Estabilizacion_me...
EN
ENO
NIVEL1_SONDA
Entrad...
Salida...
Nivel_estabilizado_1
MIL
Refere...
Salida...
Entrada_real
Referencia_real

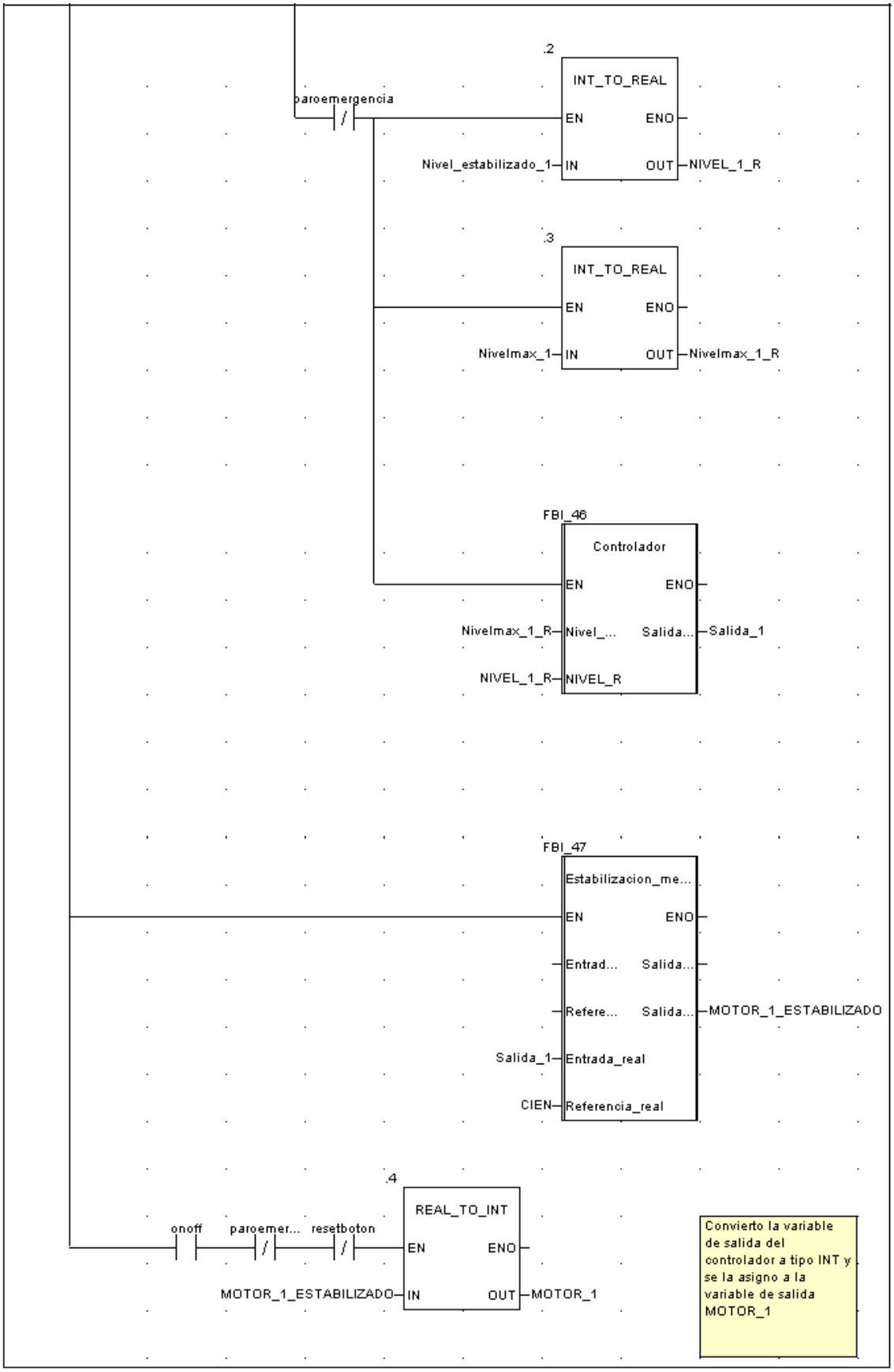
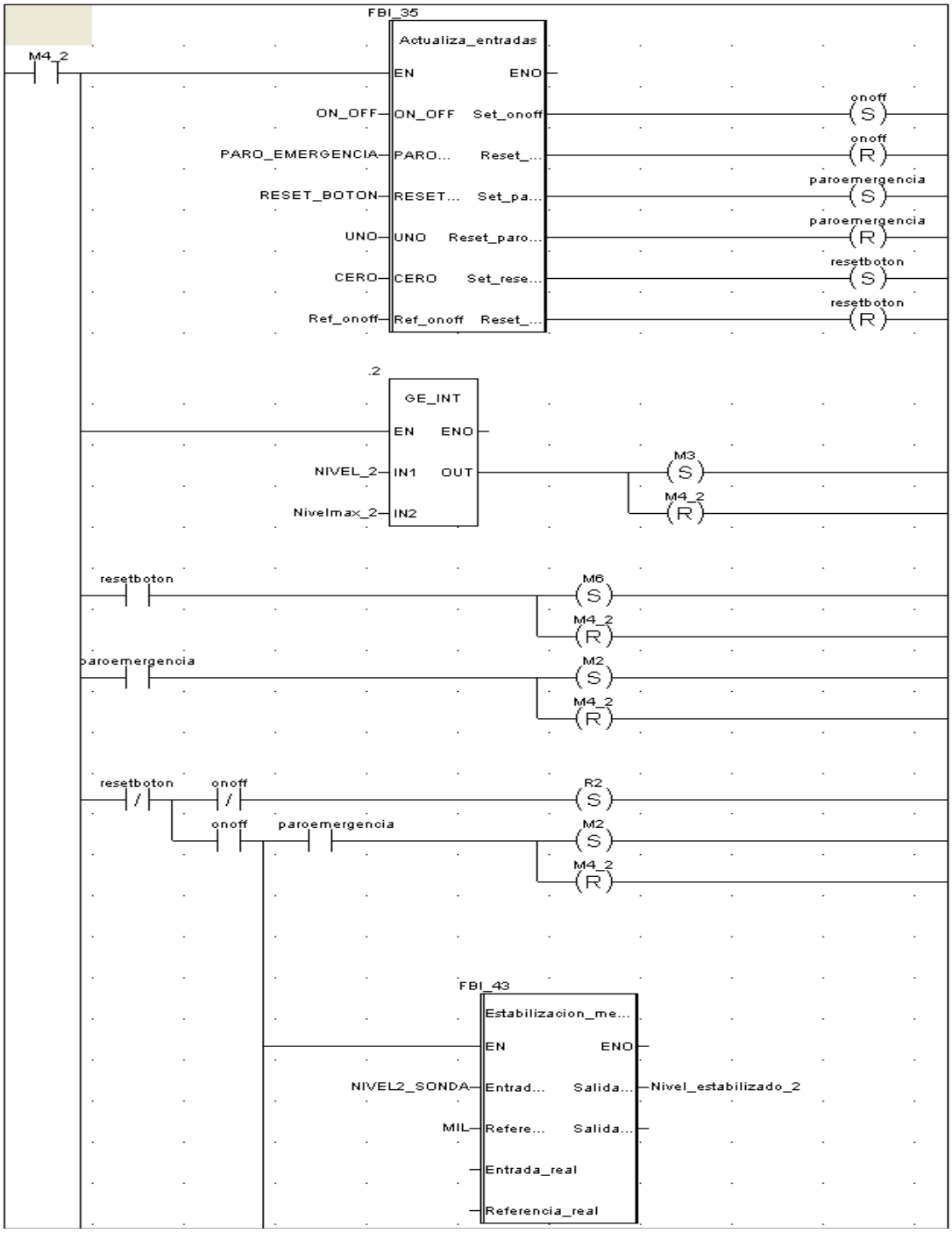


Figura 5.2.6.2 Sección 'Llenado tanque 1'

5.2.7. Llenado tanque 2

A esta sección programada en lenguaje LD, la llamaremos M4_2.Su comportamiento es idéntico al de la sección M4_1 adaptando las entradas y salidas para que funcione correctamente con el tanque 2. En las siguientes figuras podemos ver el comportamiento de la sección.



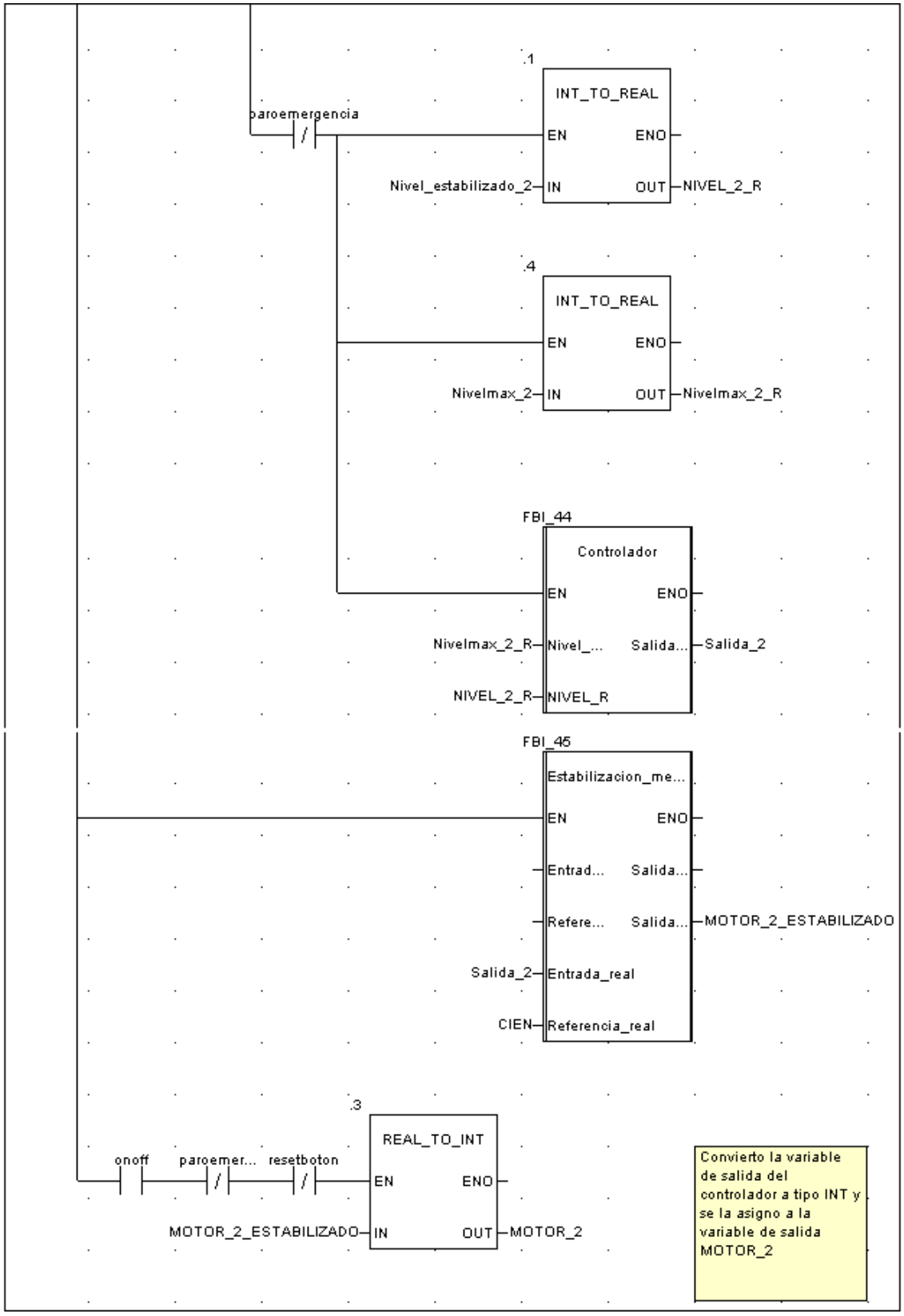


Figura 5.2.7.1 Sección 'Llenado tanque 2'

5.2.8. Error sonda

La sección que vamos a definir a definir a continuación se denomina estado M5 o de error en la sonda.

Consideraremos que existe un error cualquiera de las dos sonda cuando su comportamiento no sea el esperado, cuando esto suceda la propia sonda enviara una señal de alarma consistente en una corriente fuera del rango de *4 – 20 mA* que es el rango considerado para las medidas de nivel, según el fabricante la señal rondará los *22mA.*

Este estado se activa desde la sección funcionamiento normal 1 o funcionamiento normal, cuando el bloque encargado de comparar el nivel del tanque con el valor considerado como señal de alarma de la sonda de un resultado positivo.

La función de esta sección es únicamente la de activar el LED de alarma presente en la parte frontal del panel de control (LED rojo), el cual parpadeara mientras esté presente la señal de error, para diferenciarlo de la señal de emergencia.

Como todas las secciones anteriores, dispone del bloque actualizador de las entradas así como la lógica encargada de actualizar los diferentes estados en función de las entradas.

A continuación se muestra la estructura de programación del sistema.

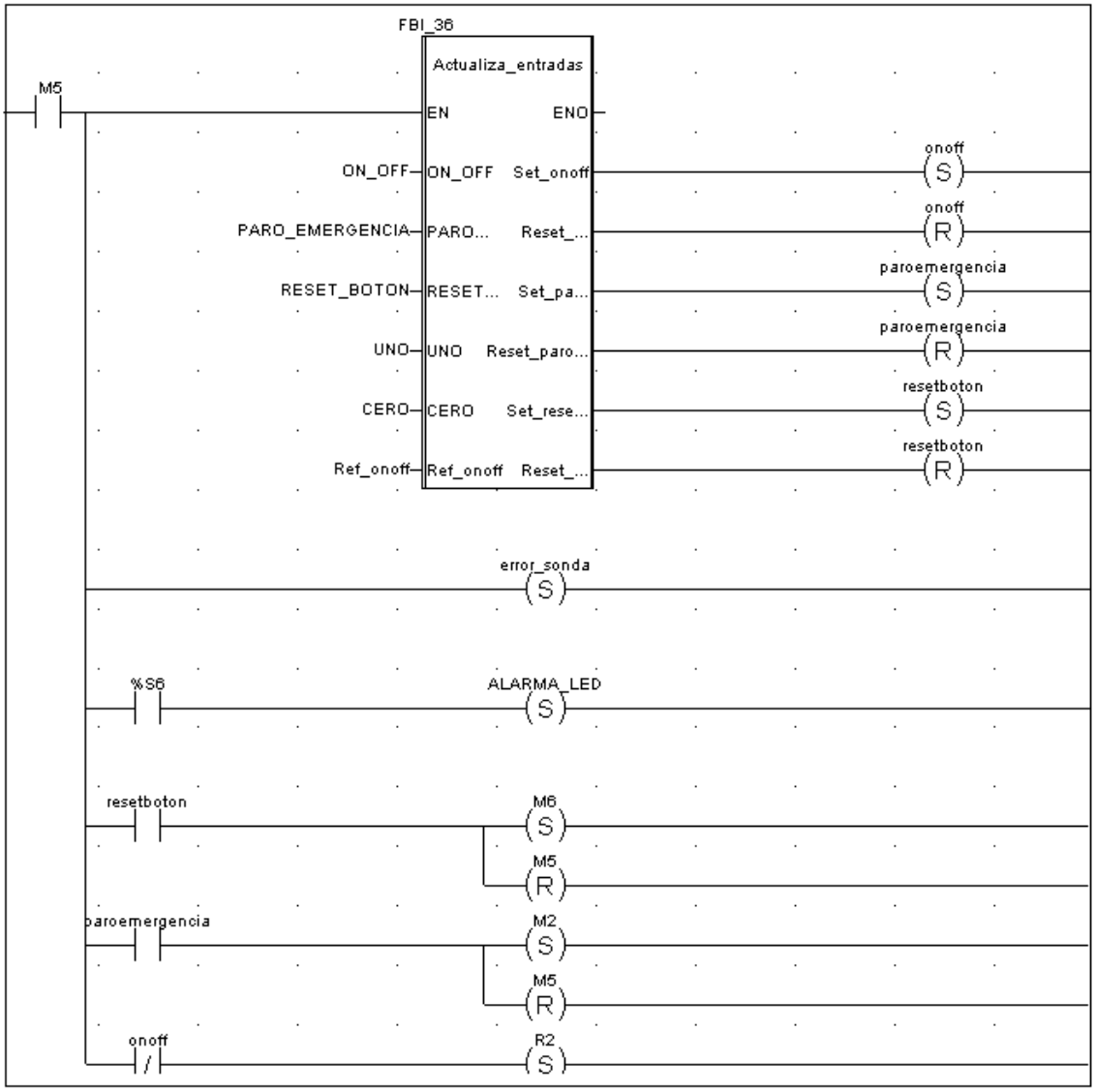


Figura 5.1.8.1 Sección 'Error sonda'

5.2.9. Reseteos

El estado M6 o de reseteos será el encargado de desactivar las diferentes variables o estados necesarios después de que se produzcan reseteos, emergencias o desconexiones durante el transcurso normal de la aplicación y de establecer la secuencia de reactivación que permita a la aplicación continuar con un funcionamiento satisfactorio tras subsanarse estas interferencias.

Existen tres tipos de reset en nuestra aplicación, reset por botón de reset externo, reset por emergencia y reset por botón de apagado, cada una de ellas con un peso determinado dentro de la sección que nos ocupa, siendo la más prioritaria la correspondiente al botón de apagado y la menos prioritaria la de correspondiente al reset.

A continuación definiremos el funcionamiento de cada una de estas subsecciones.

- **Reset por botón de reset externo**

  Esta parte simplemente se limita a desactivar los distintos estados y variables necesarios para reiniciar la aplicación dando servicio a la sección principal del programa (*M0*).

- **Reset por emergencia**

  El reset por emergencia, también llamado estado de reset *R1*, tiene un funcionamiento similar al anterior con la peculiaridad de que la reactivación de le aplicación se realiza cuando se pulsa el botón de reset tras haberse desactivad la emergencia.

- **Reset por desconexión o apagado**

  El último reset que nos queda es el reset por apagado, también llamado estado de reset *R2*, su funcionamiento es similar al reset por botón de reset externo como podemos ver en la siguiente figura.

  A continuación podemos ver está sección con los diferentes reset en el orden según les hemos enumerado:

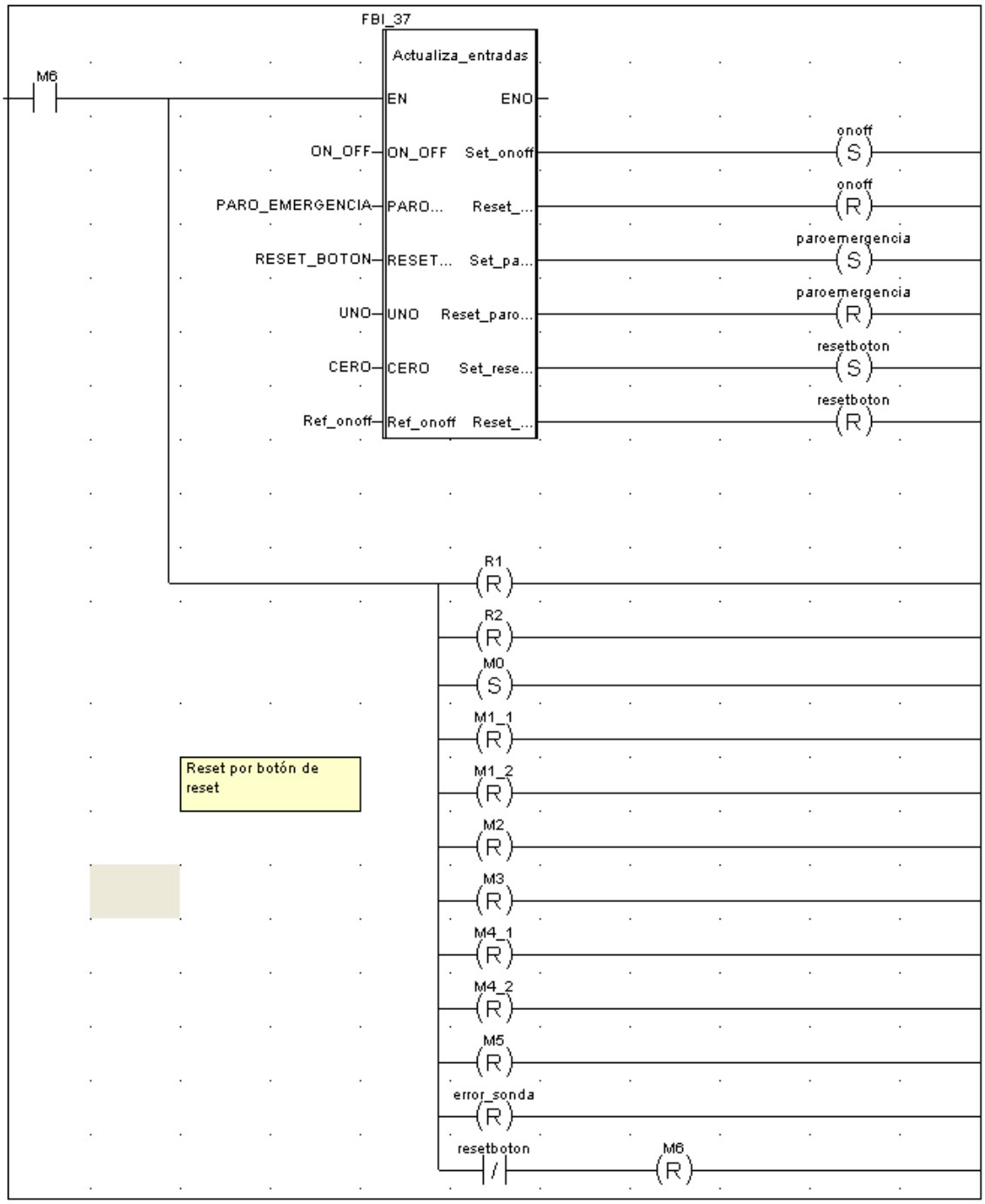


Figura 5.1.9.1 Sección 'Error sonda' (Reset por botón de reset externo)

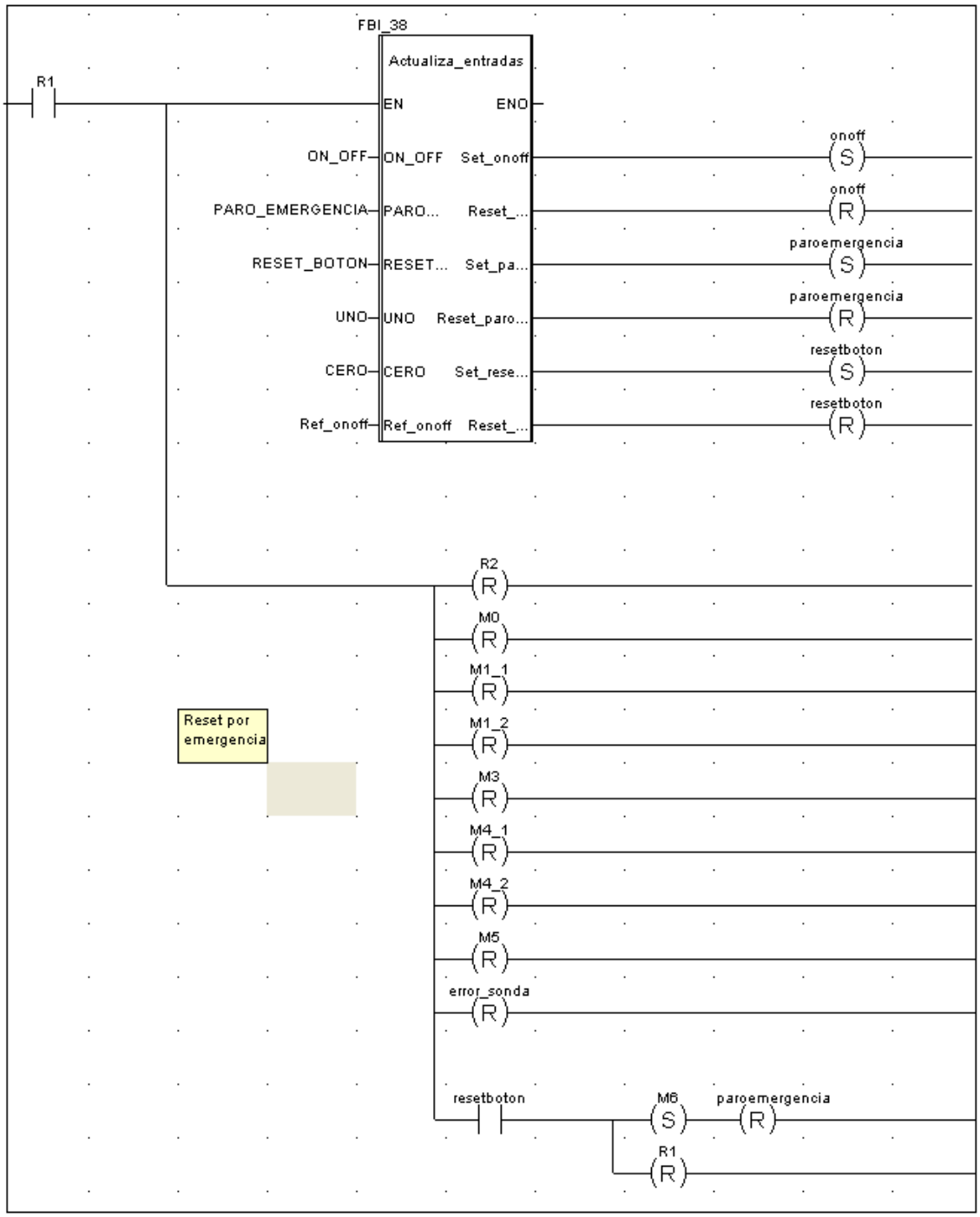


Figura 5.1.9.2 Sección 'Error sonda' (Reset por emergencia)

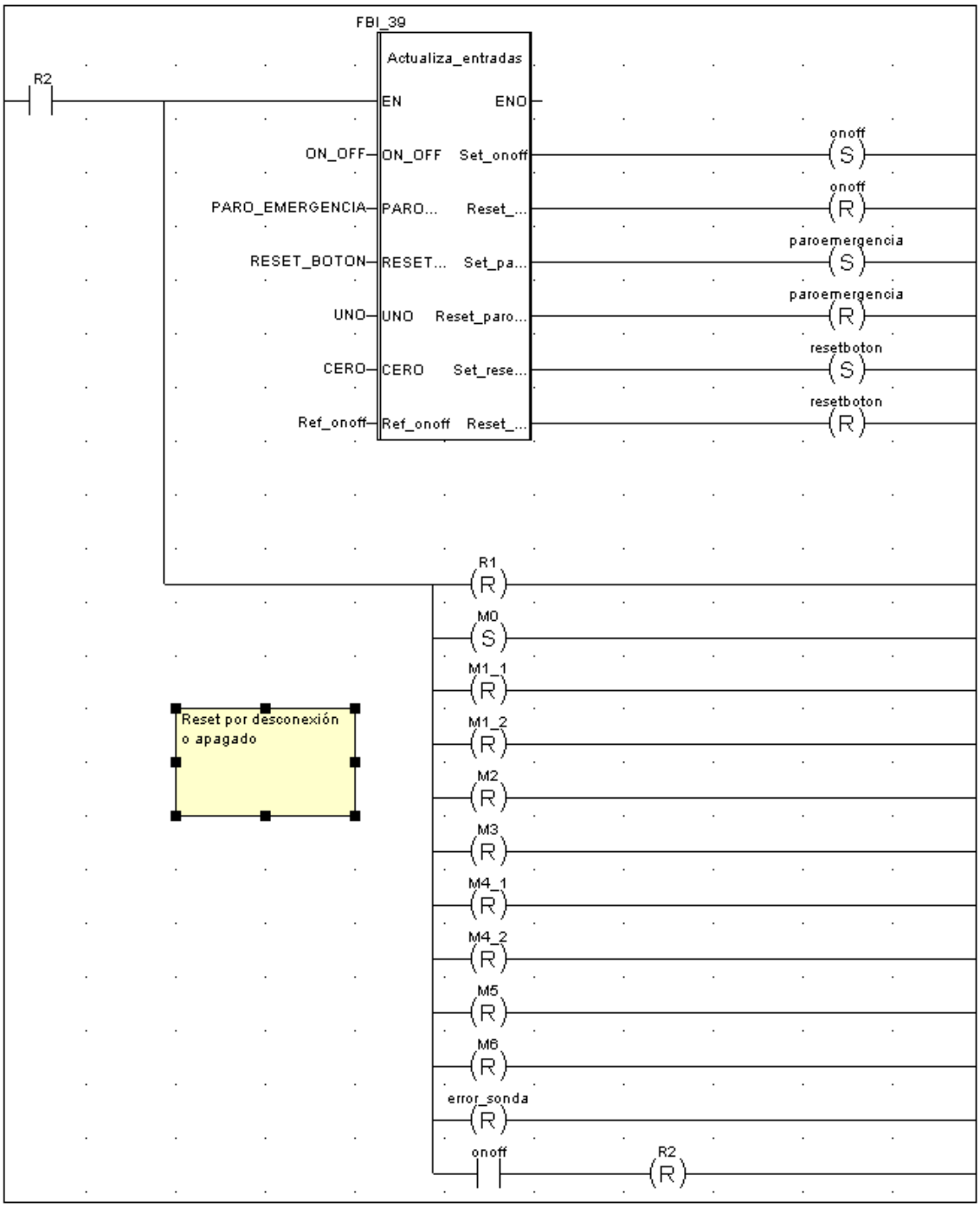


Figura 5.1.9.3 Sección 'Error sonda' (Reset por desconexión o apagado)

5.2.10. Parada del motor

Esta sección denominada parada del motor será la única programada en lenguaje texto estructurado o ST, debido a que este tipo de lenguaje se adapta mejor que el lenguaje de contactos para el tipo de sección que deseamos programar.

El cometido de esta sección no es otro que detener los motores en los instantes que esto sea preciso.

También contiene una solución para evitar los sobrepicos que tiende a generar el controlador en la señal de control para ciertos cambios de nivel, forzando a los motores a trabajar a máxima potencia en estos tramos siempre y cuando el nivel sea menor que el de referencia.

Al igual que en las secciones vistas anteriormente se actualiza en todo momento el valor de las entradas, en esta ocasión en lenguaje ST.

A continuación se muestra el código de programación de esta sección.

```
(*Implemento el bloque Actualiza_entradas en texto estructurado para
actualizar el valor de las variables de entrada y corresponderlas con
sus variables booleanas, como hemos hecho en las demás secciones*)

IF ON_OFF>=1500 THEN
        onoff:=1;
END_IF;

IF ON_OFF<1500 THEN
        onoff:=0;
END_IF;

IF PARO_EMERGENCIA = 1 THEN
        paroemergencia:=1;
END_IF;

IF PARO_EMERGENCIA = 0 THEN
        paroemergencia:=0;
END_IF;

IF RESET_BOTON = 0 THEN
        resetboton:=0;
END_IF;

IF RESET_BOTON = 1 THEN
        resetboton:=1;
END_IF;
```

```
(*Fuerzo las variables MOTOR_1 y MOTOR_2 a cero si se da una de las
 siguientes condiciones: botón de reset pulsado, botón de emergencia
 pulsado o si no está pulsado el botón de encendido*)

IF resetboton OR paroemergencia OR NOT onoff THEN
        MOTOR_1:=0;
        MOTOR_2:=0;
END_IF;

IF resetboton THEN
        ALARMA_LED:=0;
END_IF;

(*Elimino las oscilaciones de la salida del controlador que se producen en
los cambios de nivel para el motor 1*)

IF NIVEL_1 > Nivelmax_1 THEN
        MOTOR_1:=0;
        ELSE
                IF NIVEL_1>1950 AND NIVEL_1 <2050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;

                IF NIVEL_1>2950 AND NIVEL_1 <3050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;

                IF NIVEL_1>3950 AND NIVEL_1 <4050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;

                IF NIVEL_1>4950 AND NIVEL_1 <5050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;

                IF NIVEL_1>5950 AND NIVEL_1 <6050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;
                IF NIVEL_1>6950 AND NIVEL_1 <7050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;

                IF NIVEL_1>7950 AND NIVEL_1 <8050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;

                IF NIVEL_1>8950 AND NIVEL_1 <9050 AND paroemergencia=0 THEN
                        MOTOR_1:=20000;
                END_IF;
END_IF;
```

```
(*Realizo la misma operación con el motor 2*)


IF NIVEL_2 > Nivelmax_2 THEN
        MOTOR_2:=0;
        ELSE
                IF NIVEL_2>1950 AND NIVEL_2 <2050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;

                IF NIVEL_2>2950 AND NIVEL_2 <3050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;

                IF NIVEL_2>3950 AND NIVEL_2 <4050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;

                IF NIVEL_2>4950 AND NIVEL_2 <5050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;

                IF NIVEL_2>5950 AND NIVEL_2 <6050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END IF;
                IF NIVEL_2>6950 AND NIVEL_2 <7050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;

                IF NIVEL_2>7950 AND NIVEL_2 <8050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;

                IF NIVEL_2>8950 AND NIVEL_2 <9050 AND paroemergencia=0 THEN
                        MOTOR_2:=20000;
                END_IF;
END_IF;
```

Figura 5.1.10.1 Sección 'Parada del motor'

5.2.11. Estado de los niveles

Esta sección del programa, programada en lenguaje LD, se encargará de realizar diferentes chequeos y comparaciones para determinar estado de los niveles de fluido en los tanques que posteriormente serán usados en otras secciones del programa.

La sección distingue entre los siguientes estados:

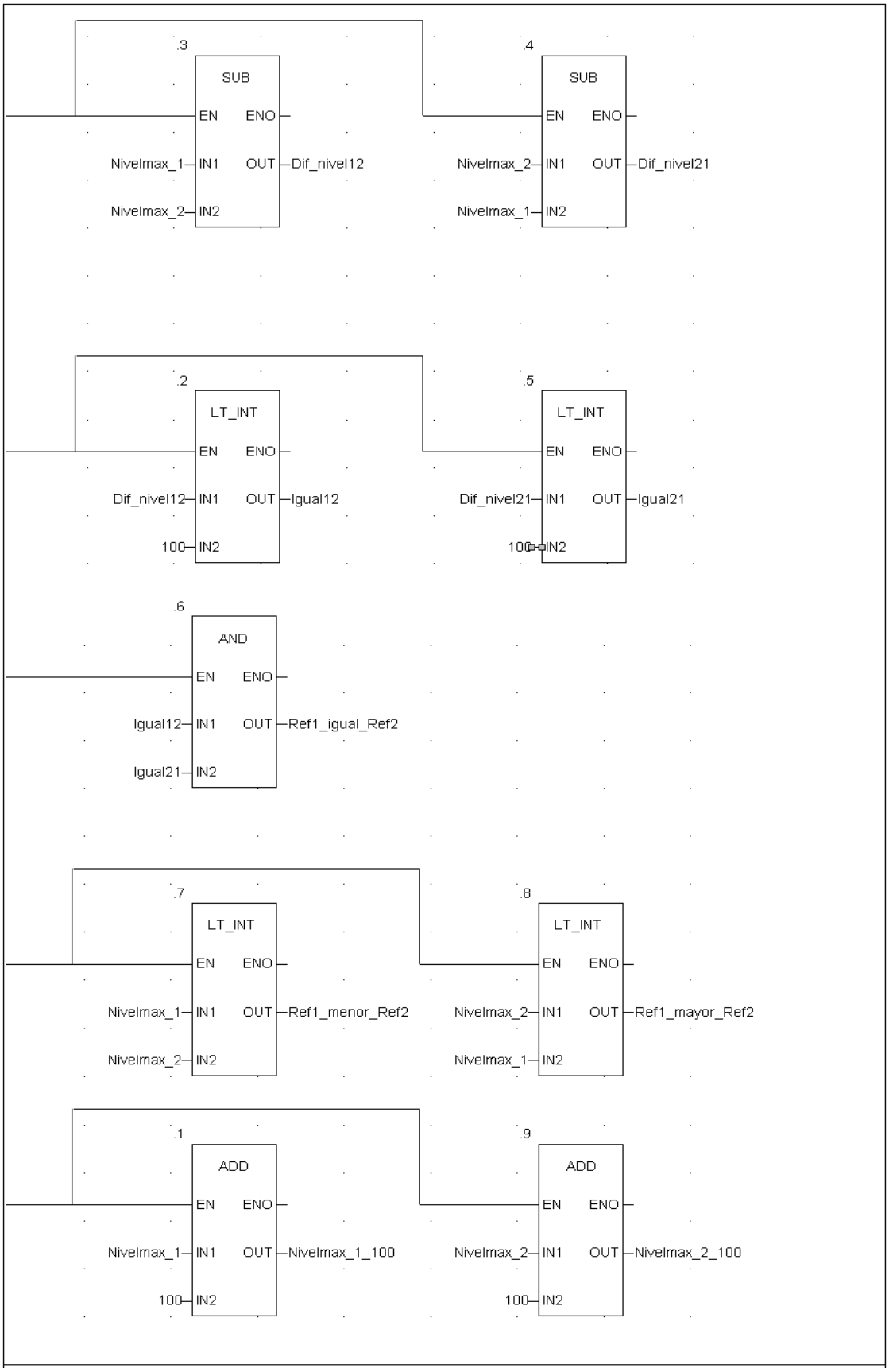
- Determina si la diferencia entre los nivele de referencia de ambos tanques está dentro de una tolerancia los suficientemente pequeña para considerar que el nivel de referencia de ambos tanques es el mismo.

- En caso de no cumplirse lo anterior determina cual de los dos tanques posee el nivel de referencia mayor.

- Distingue también si alguno de los dos tanques ha sobrepasado el nivel de referencia en una cantidad lo suficientemente elevada para considerarse un comportamiento erróneo.

El resultado de todas estas comparaciones se almacenará en distintas variables para su posterior uso.

La programación de esta sección puede verse a continuación.


.3
SUB
EN
ENO
Nivelmax_1
IN1
OUT
Dif_nivel12
Nivelmax_2
IN2
.4
SUB
EN
ENO
Nivelmax_2
IN1
OUT
Dif_nivel21
Nivelmax_1
IN2
.2
LT_INT
EN
ENO
Dif_nivel12
IN1
OUT
Igual12
100
IN2
.5
LT_INT
EN
ENO
Dif_nivel21
IN1
OUT
Igual21
100
IN2
.6
AND
EN
ENO
Igual12
IN1
OUT
Ref1_igual_Ref2
Igual21
IN2
.7
LT_INT
EN
ENO
Nivelmax_1
IN1
OUT
Ref1_menor_Ref2
Nivelmax_2
IN2
.8
LT_INT
EN
ENO
Nivelmax_2
IN1
OUT
Ref1_mayor_Ref2
Nivelmax_1
IN2
.1
ADD
EN
ENO
Nivelmax_1
IN1
OUT
Nivelmax_1_100
100
IN2
.9
ADD
EN
ENO
Nivelmax_2
IN1
OUT
Nivelmax_2_100
100
IN2

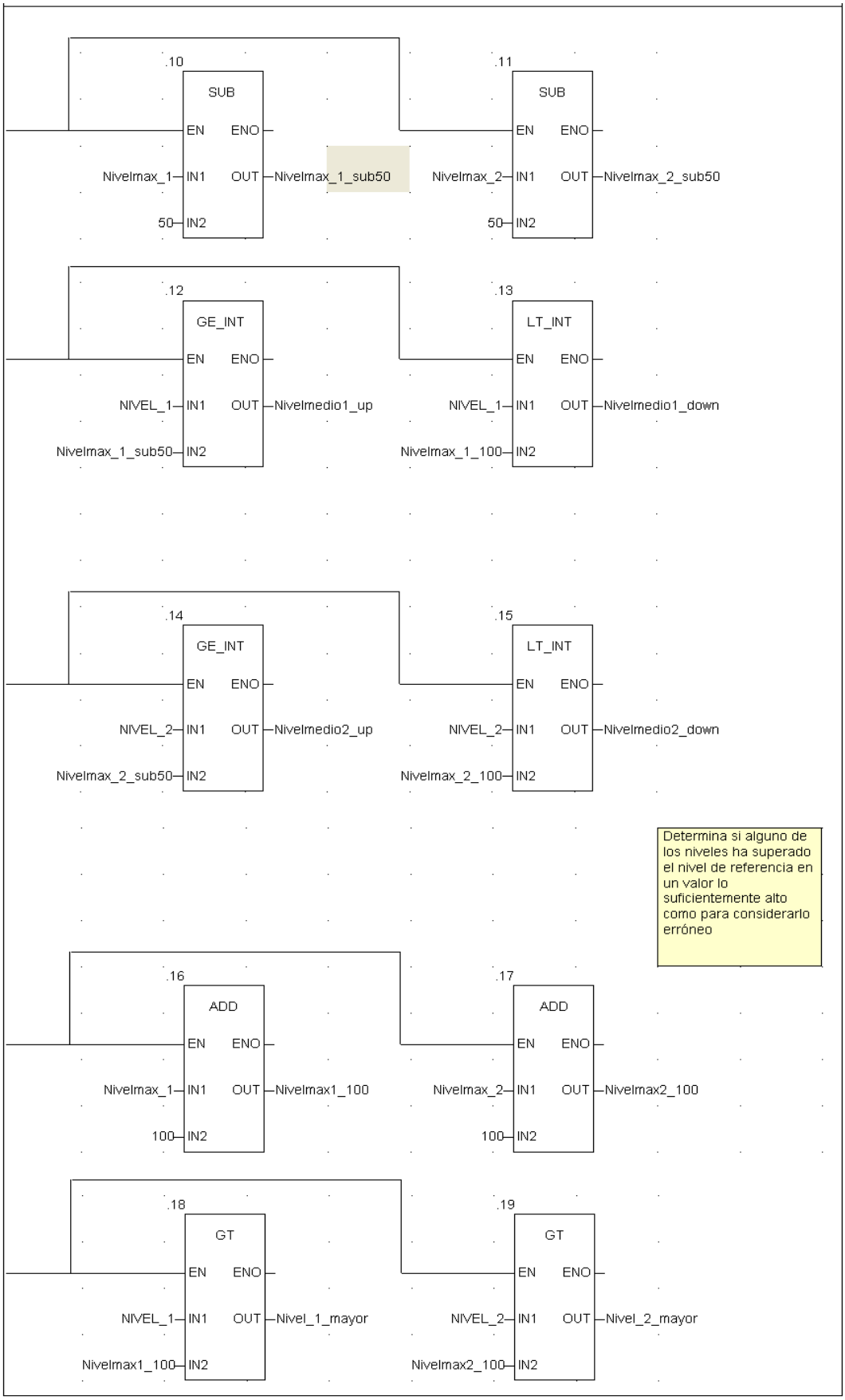


Figura 5.1.11 Sección 'Estado niveles'

5.2.12. Estado de los motores

Esta sección, también programada en lenguaje LD, se encargará de determinar el estado de funcionamiento de las dos bombas mediante diferentes comparaciones y comprobaciones.

Para ello se realiza un filtro de la señal de control que se envía a la bomba, mediante el cálculo de la media de esta en un pequeño periodo de tiempo, y en función de su valor se distingue entre tres modos de funcionamiento distintos, que se definen a continuación.

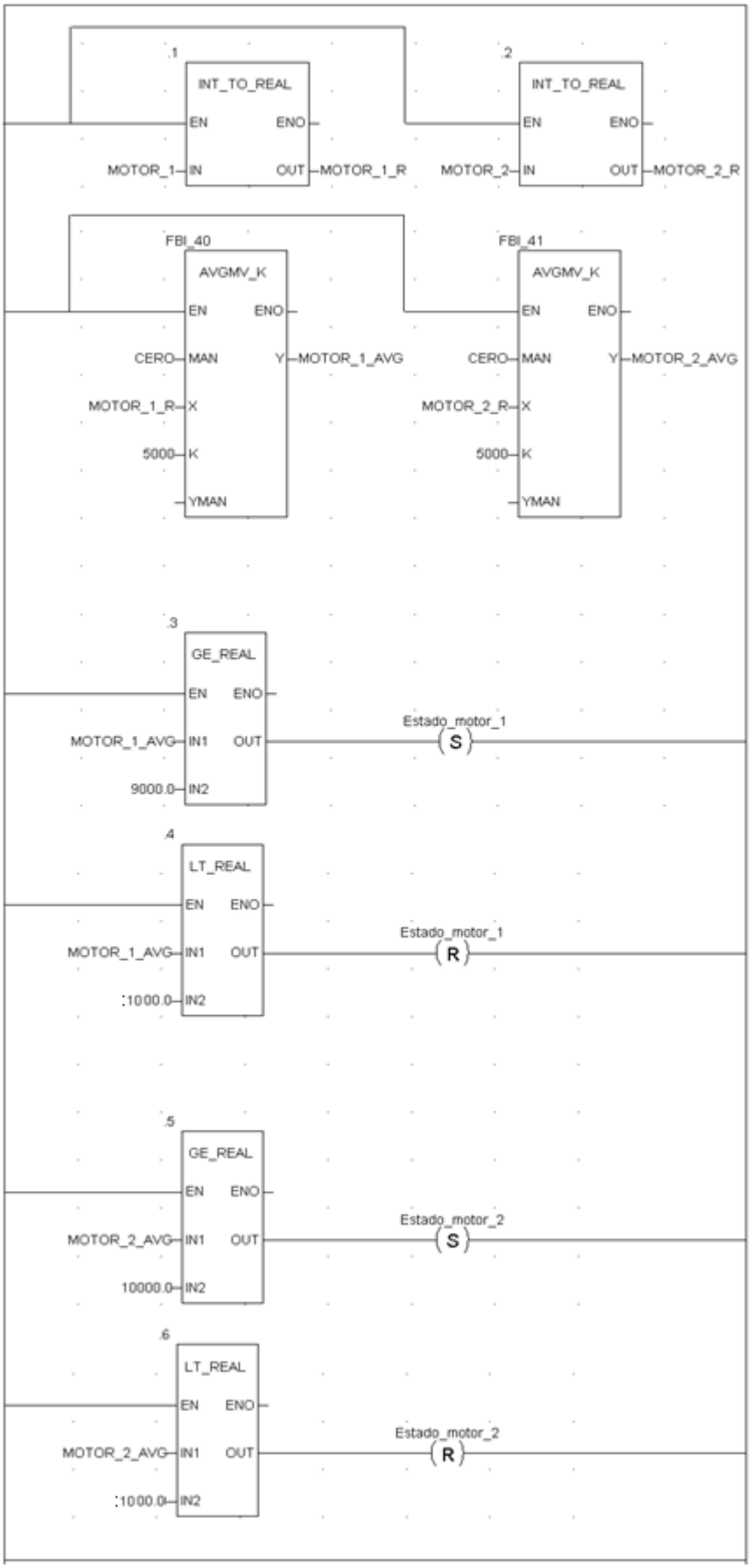
- La bomba no está funcionando o su potencia de funcionamiento es tan baja que puede considerarse nula.

- La bomba está funcionando a pleno rendimiento suministrando el máximo caudal o un valor tan alto que puede considerarse como tal.

- La bomba está funcionando a una potencia media dentro de un rango determinado experimentalmente.

Al igual que en la sección anterior el resultado de las comparaciones se almacena en distintas variables para su posterior utilización por otras secciones del programa.

Su estructura de programación puede verse en las figuras siguientes.


.1
INT_TO_REAL
EN
ENO
MOTOR_1
IN
OUT
MOTOR_1_R
.2
INT_TO_REAL
EN
ENO
MOTOR_2
IN
OUT
MOTOR_2_R
FBI_40
AVGMV_K
EN
ENO
CERO
MAN
Y
MOTOR_1_AVG
MOTOR_1_R
X
5000
K
YMAN
FBI_41
AVGMV_K
EN
ENO
CERO
MAN
Y
MOTOR_2_AVG
MOTOR_2_R
X
5000
K
YMAN
.3
GE_REAL
EN
ENO
MOTOR_1_AVG
IN1
OUT
9000.0
IN2
Estado_motor_1
S
.4
LT_REAL
EN
ENO
MOTOR_1_AVG
IN1
OUT
:1000.0
IN2
Estado_motor_1
R
.5
GE_REAL
EN
ENO
MOTOR_2_AVG
IN1
OUT
10000.0
IN2
Estado_motor_2
S
.6
LT_REAL
EN
ENO
MOTOR_2_AVG
IN1
OUT
:1000.0
IN2
Estado_motor_2
R

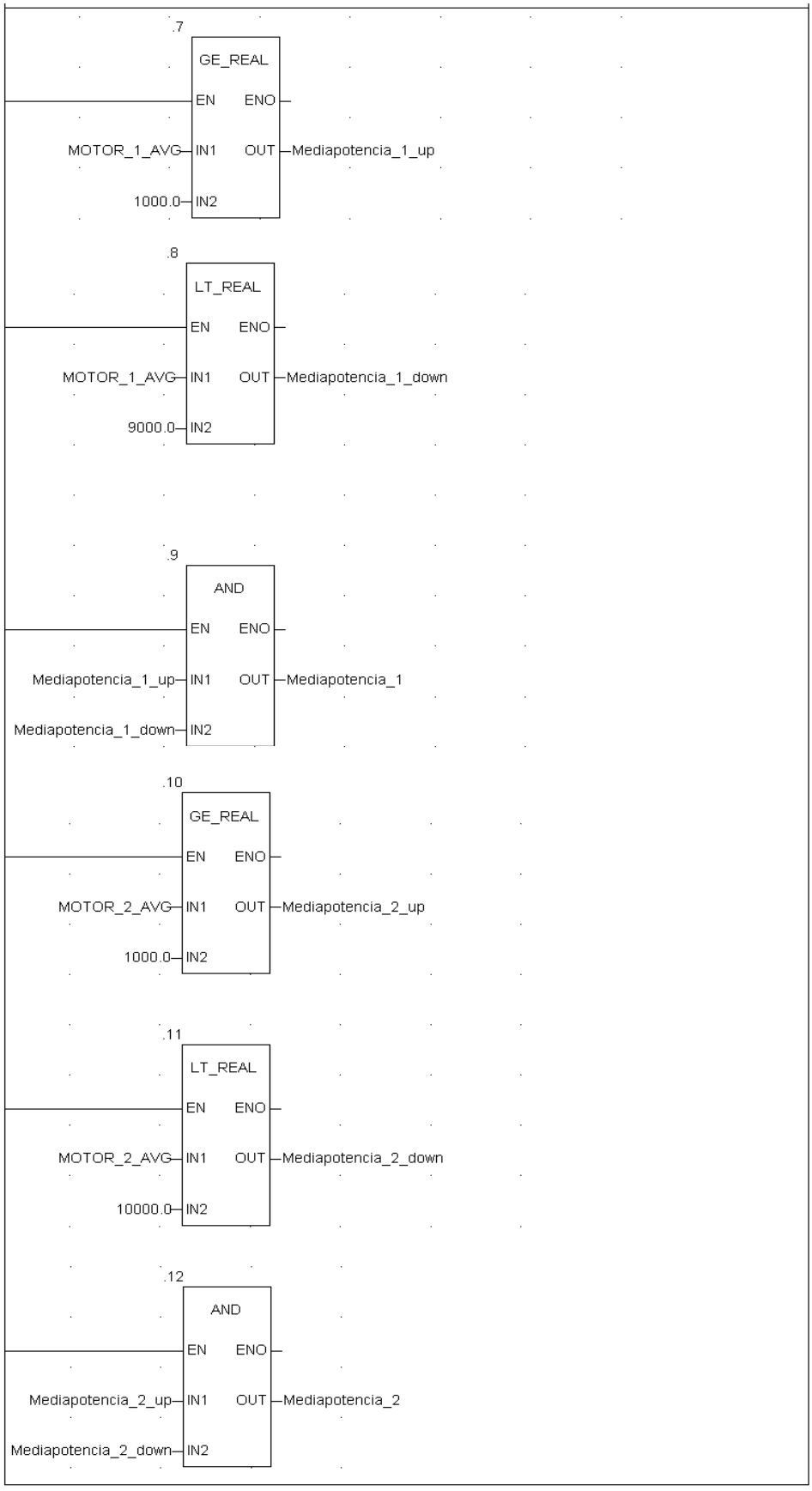


Figura 5.1.12.1 Sección 'Estado motores'

5.2.13. Estado de las válvulas

La sección que nos ocupa, programa en lenguaje LD, tiene por objeto determinar el estado de las tres válvulas manuales que posee el sistema. Para ello se ayuda de las variables de estado calculadas en las dos secciones anteriores que determinan el estado de los niveles de los tanques y las bombas.

El programa es capaz de distinguir entre cinco estados distintos, que se detallan a continuación.

- AAA_Defecto → Es el estado que se considera por defecto, al iniciarse el programa presentara este estado y se suponen las tres válvulas abiertas, si bien es el único estado que no se puede determinar con total certeza,

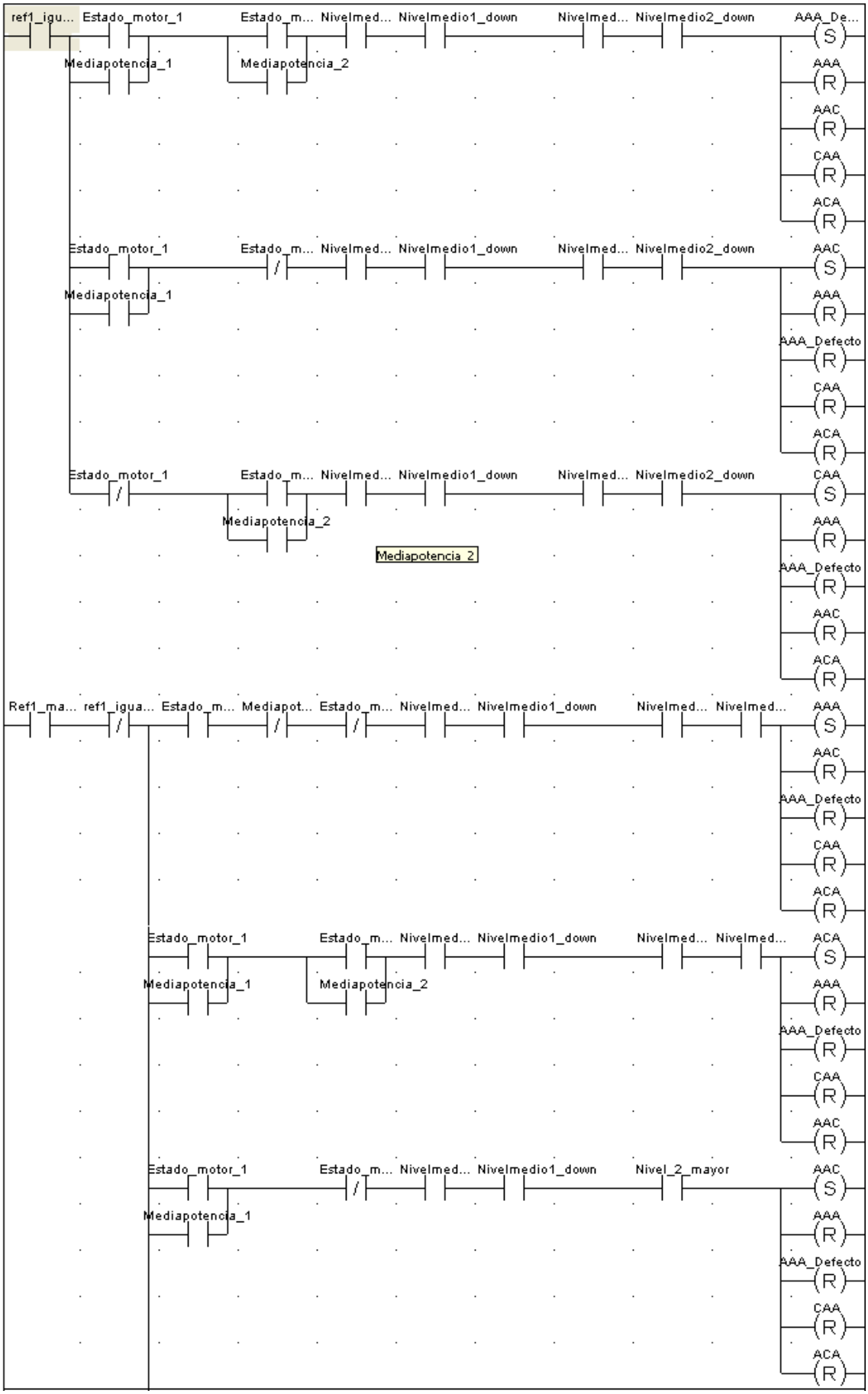
- AAA → Considera las tres válvulas abiertas, este estado a diferencia del anterior si se determina con seguridad.

- CAA → La válvula izquierda (desagüe del tanque 1) está cerrada y las otras dos abiertas.

- ACA → La válvula central (la que comunica los dos tanques), está cerrada mientras que el resto están abiertas.

- AAC → La válvula derecha (desagüe tanque 2) está cerrada, manteniéndose las dos restantes abiertas.

El cálculo de los diferentes estados se realiza comparando entre sí todas las variables de estado calculadas en los dos estados anteriores, la forma en que se realizan dichas comparaciones puede observarse a continuación.


ref1_igu...
Estado_motor_1
Estado_m...
Nivelmed...
Nivelmedio1_down
Nivelmed...
Nivelmedio2_down
AAA_De...
Mediapotencia_1
Mediapotencia_2
AAA
AAC
CAA
ACA
Estado_motor_1
Estado_m...
Nivelmed...
Nivelmedio1_down
Nivelmed...
Nivelmedio2_down
AAC
Mediapotencia_1
AAA
AAA_Defecto
CAA
ACA
Estado_motor_1
Estado_m...
Nivelmed...
Nivelmedio1_down
Nivelmed...
Nivelmedio2_down
CAA
Mediapotencia_2
Mediapotencia 2
AAA
AAA_Defecto
AAC
ACA
Ref1_ma...
ref1_igua...
Estado_m...
Mediapot...
Estado_m...
Nivelmed...
Nivelmedio1_down
Nivelmed...
Nivelmed...
AAA
AAC
AAA_Defecto
CAA
ACA
Estado_motor_1
Estado_m...
Nivelmed...
Nivelmedio1_down
Nivelmed...
Nivelmed...
ACA
Mediapotencia_1
Mediapotencia_2
AAA
AAA_Defecto
CAA
AAC
Estado_motor_1
Estado_m...
Nivelmed...
Nivelmedio1_down
Nivel_2_mayor
AAC
Mediapotencia_1
AAA
AAA_Defecto
CAA
ACA

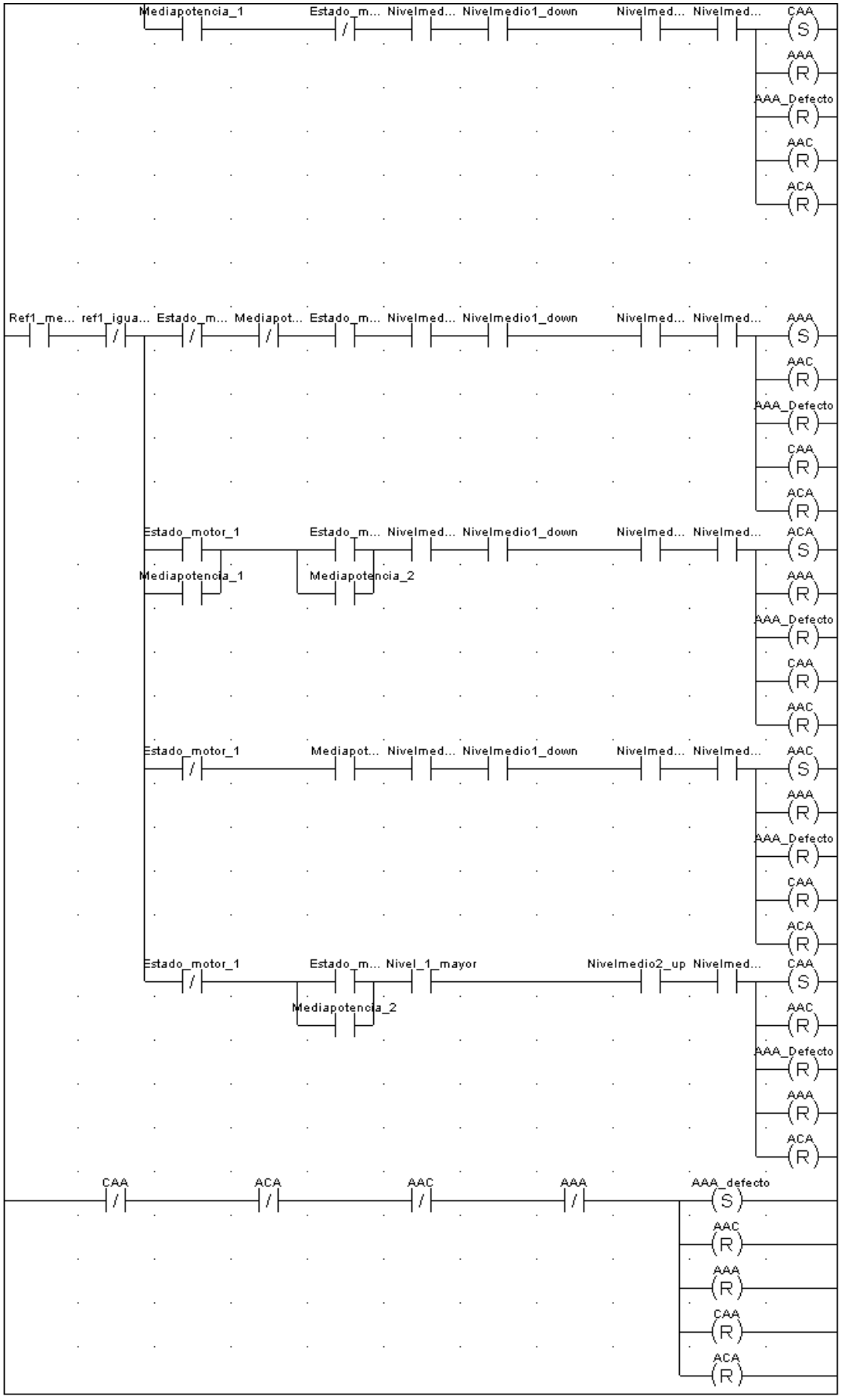
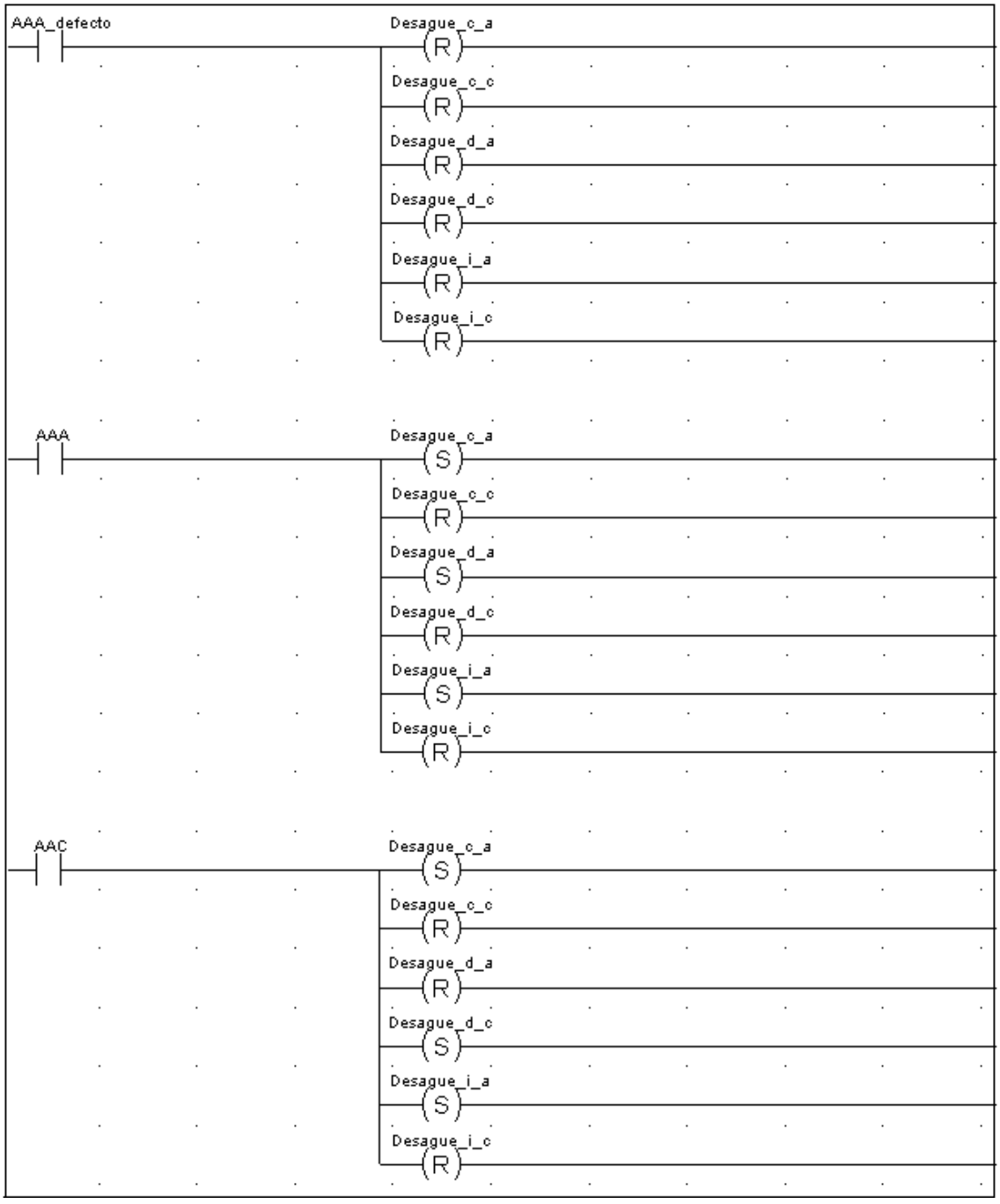


Figura 5.1.13.1 Sección 'Estado válvulas'

5.2.14. Representación de las válvulas

Esta sección, programada también en lenguaje LD, únicamente tiene como objetivo activar o desactivar en cada caso las variables que corresponda para representar correctamente en la pantalla gráfica el estado de las válvulas.

Su programación puede observarse a continuación.



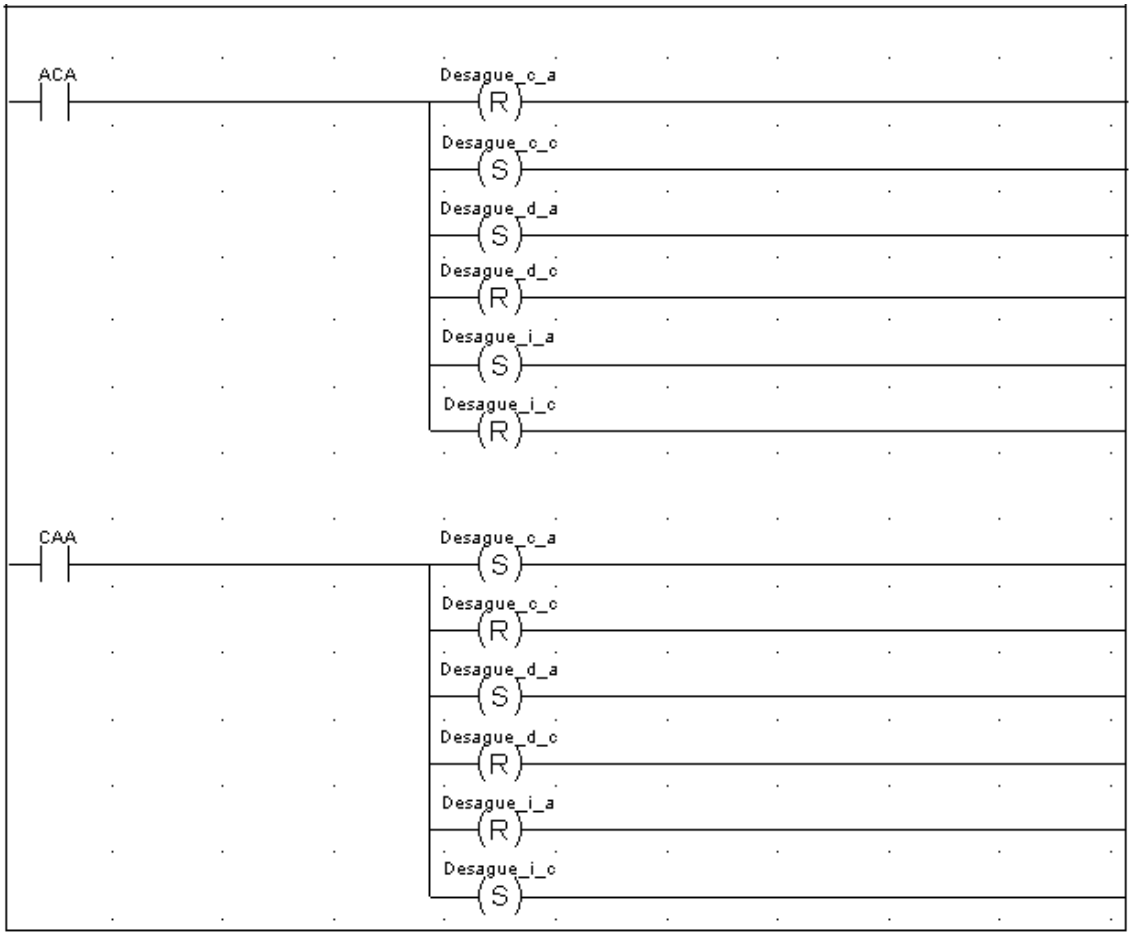


Figura 5.1.14.1 Sección 'Representación válvulas'

5. RESULTADOS Y CONCLUSIONES

5.1 Resultados obtenidos

Los resultados obtenidos han sido bastante satisfactorios, ya que con la aplicación desarrollada conseguimos tanto el control de los niveles de los dos tanques como la detección de atascos en cualquiera de las 3 válvulas que el sistema dispone.

Se ha desarrollado teóricamente la función que define el comportamiento del vaciado del sistema de tanques teniendo en cuenta que se utilizan llaves de paso de apertura manual.

Tras el cálculo de las funciones y modelos hemos diseñado el controlador y lo hemos implementado en forma de una ecuación en diferencias, escribiéndolo en el programa en la sección del controlador realizada en lenguaje literal estructurado (ST). El controlador diseñado es bastante rápido. Además responde muy bien ante cambios pequeños en las referencias de los niveles.

Una vez diseñado el controlador el siguiente paso realizado fue el desarrollo de la aplicación en la plataforma del software propio del autómata Modicon M340, este software se trata de Unity Pro. A la hora de este desarrollo se siguió con las especificaciones requeridas en los cálculos anteriores.

Una vez desarrollado el software y tras comprobar su buen funcionamiento se desarrollo el software encargado de la detección de atascos en las válvulas, este software tuvo un proceso de desarrollo arduo debido a que se debía valorar muchas variables y realizar numerosas pruebas de funcionamiento hasta ajustar diferentes parámetros.

Para ver los resultados del control del nivel sobre los tanques se muestran 2 casos, en el primero se parte de los tanques vacíos y se introducen las referencias de los tanques en el de la izquierda la referencia es 4000 (que equivale aproximadamente a 14.5 cm) y en el tanque de la derecha se introduce como referencia 2500 (que equivale aproximadamente a 5.5 cm); la evolución tanto de los niveles como la acción de control de los motores se muestran a continuación.

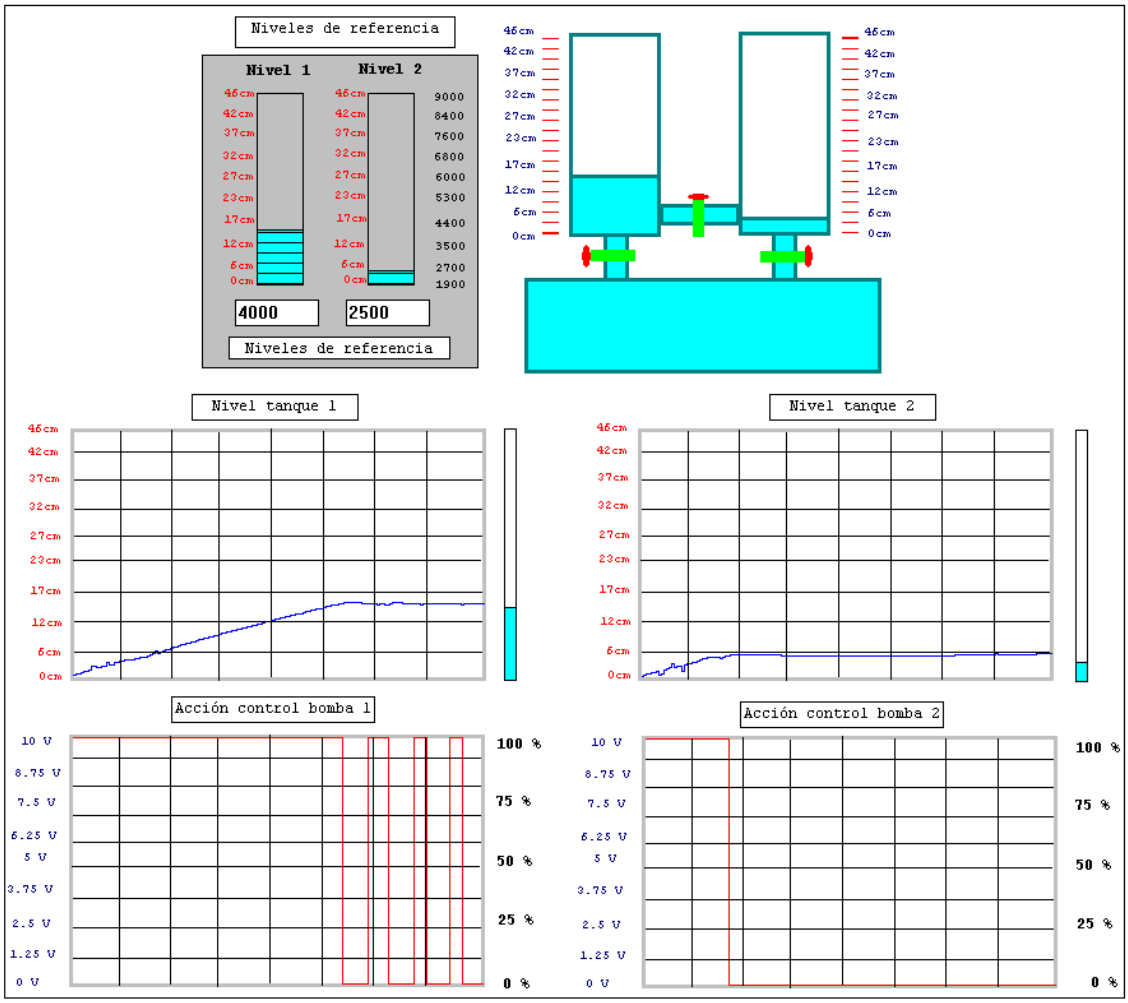


Figura 6.1.1 Resultados pantalla operador y gráficas de tendencias del caso 1

Como se puede ver en las dos gráficas, las bombas trabajan a la máxima potencia hasta alcanzar el nivel de referencia. Una vez alcanzado este nivel, al ser el nivel del tanque de la izquierda mayor que el de la derecha, la bomba 2 deja de funcionar y la bomba 1 sigue trabajando a media carga para que los niveles de los dos tanques continúen en los niveles de referencia establecidos.

En el segundo caso, partimos de el estado anterior y cambiamos el nivel de referencia del tanque de la izquierda a 3500 (12 cm de altura) y el de la derecha a 5000 (22 cm de altura); las evoluciones de los niveles en los tanques como las del control de de los motores se muestran en las siguientes imágenes.

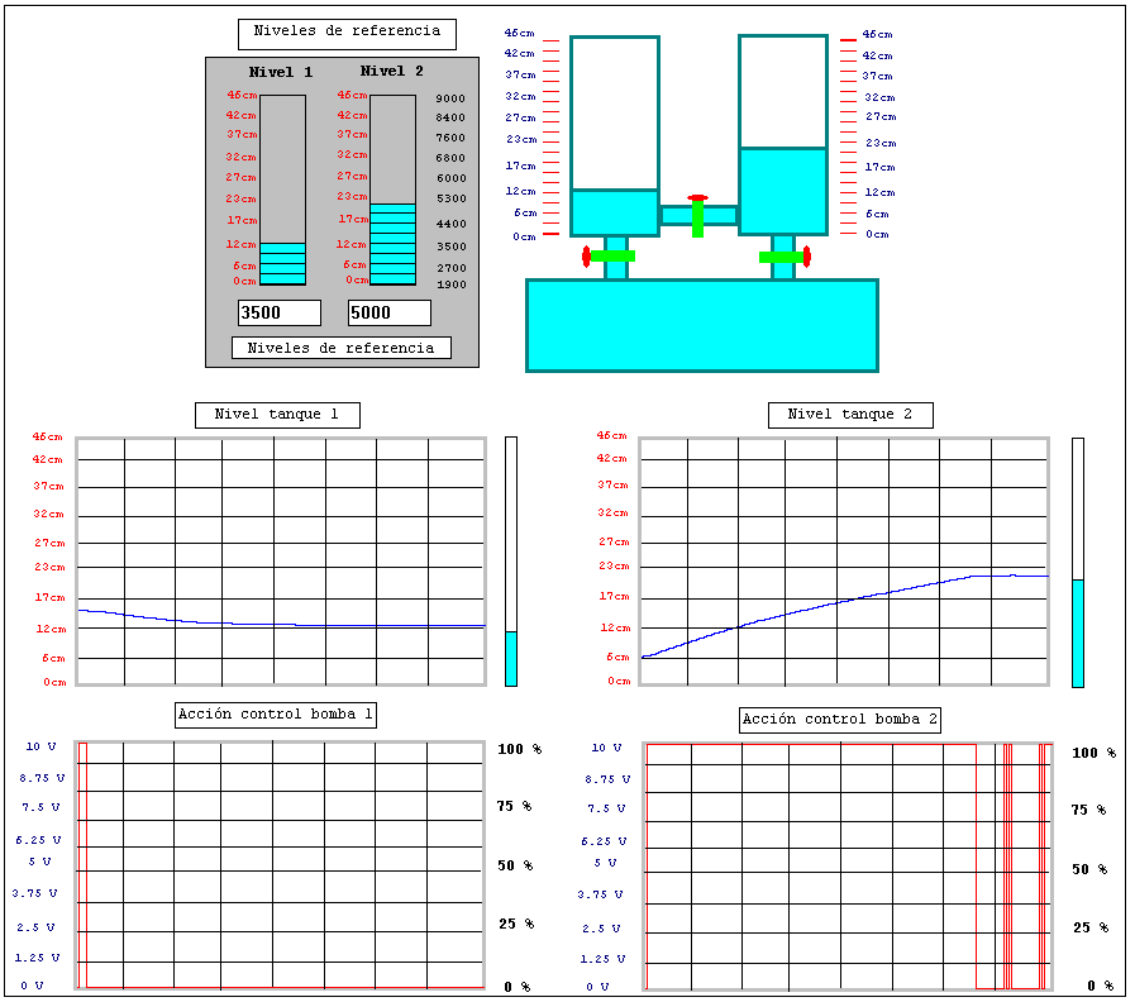


Figura 6.1.2 Resultados pantalla operador y gráficas de tendencias del caso 2

Como se puede ver en las dos gráficas, las bombas trabajan a la máxima potencia hasta alcanzar el nivel de referencia. Una vez alcanzado este nivel, al ser el nivel del tanque de la derecha mayor que el de la izquierda, la bomba 1 deja de funcionar y la bomba 2 sigue trabajando a media carga para que los niveles de los dos tanques continúen en los niveles de referencia establecidos.

En cuanto a la detección de atascos nuestro sistema es capaz de detectar 9 casos. Ya que se cuenta con tres válvulas diferentes y para cada caso y la posibilidad que los niveles de referencia sean iguales o uno mayor que otro.

A continuación expondremos los resultados obtenidos en cada caso.

Para los 2 niveles de referencia iguales:

- Las 3 válvulas abiertas: Los niveles de líquido se mantienen en los niveles de referencia establecidos y las bombas trabajan a media carga.
- La válvula central cerrada: Este caso lo tomamos como el anterior debido a que el flujo que atraviesa la válvula central se considera 0 en ambos casos.
- La válvula de la izquierda cerrada: Los niveles de líquido se mantienen en los niveles de referencia establecidos. La bomba de la izquierda está inactiva y la de la derecha funciona a media carga.
- La válvula de la derecha cerrada: Los niveles de líquido se mantienen en los niveles de referencia establecidos. La bomba de la derecha está inactiva mientras que la de la izquierda trabaja a media carga.

Vamos a mostrar los resultados este último caso, con la válvula de la derecha cerrada y las otras dos abiertas:

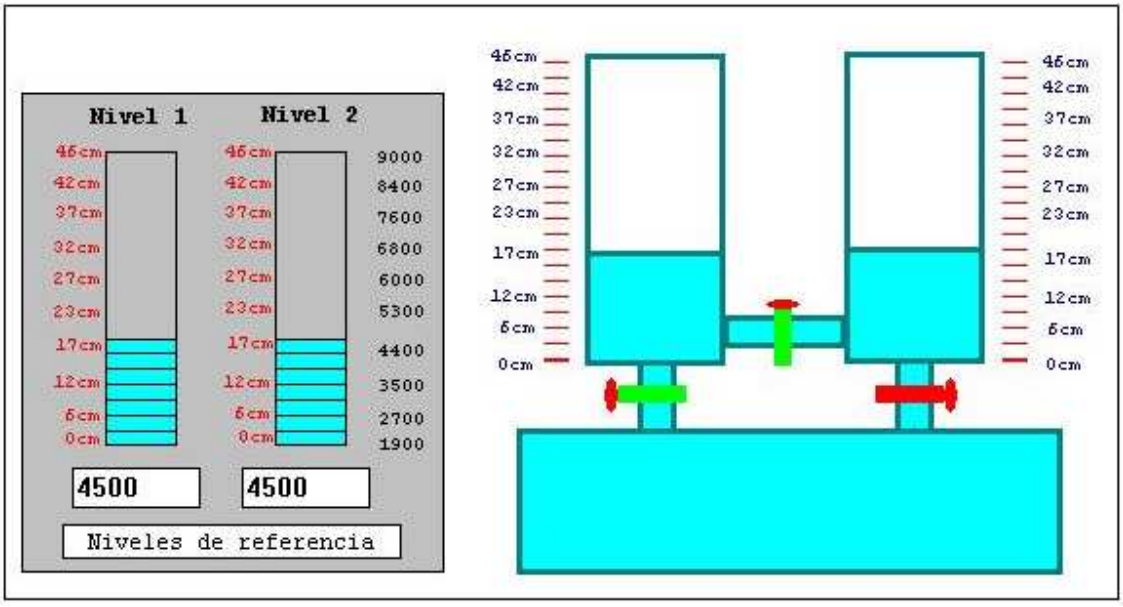


Figura 6.1.3 Estado niveles atascos caso 1

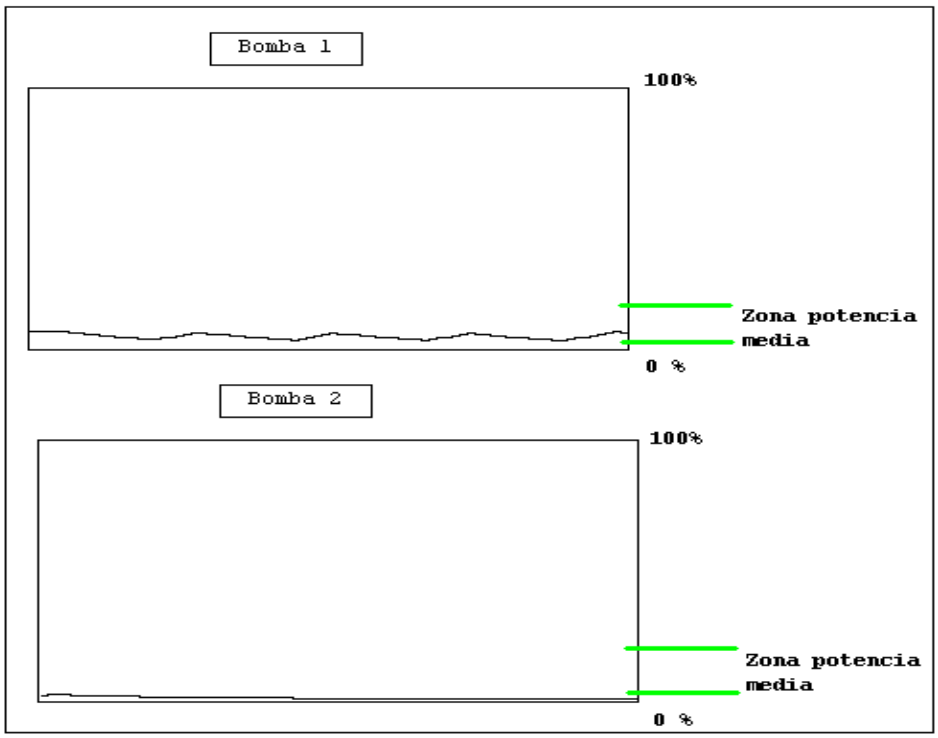


Figura 6.1.4 Gráfica potencia media bombas caso 1

De acuerdo con las secciones 'Estado_valvulas': Si la bomba 2 no está funcionando, la bomba 1 está trabajando a media carga y los niveles de los tanques están comprendidos entre 4450 y 4550, significa que la válvula derecha está cerrada y las otras dos abiertas.

Para el nivel de referencia del tanque de la izquierda mayor que el del tanque de la derecha:

- Las 3 válvulas abiertas: Los niveles de líquido se mantienen en los niveles de referencia establecidos. La bomba de la derecha permanece inactiva mientras que la de la izquierda trabaja a media carga, aunque a un potencia algo superior a cuando los dos niveles eran iguales.
- La válvula de la derecha cerrada: El nivel en el tanque de la izquierda se mantiene en el nivel de referencia establecido mientras que el nivel del tanque de la derecha va aumentando. La bomba de la izquierda trabaja a media potencia y la de la derecha permanece inactiva.
- La válvula de la izquierda cerrada: Los niveles se mantienen en los niveles de referencia establecidos. La bomba de la derecha permanece inactiva mientras que la de la izquierda trabaja a media carga.
- La válvula central cerrada: Los niveles de líquido se mantienen en los niveles de referencia establecidos y las dos bombas trabajan a media carga.

A continuación mostramos los resultados de este último caso, las válvula central cerrada y las otras dos abiertas:

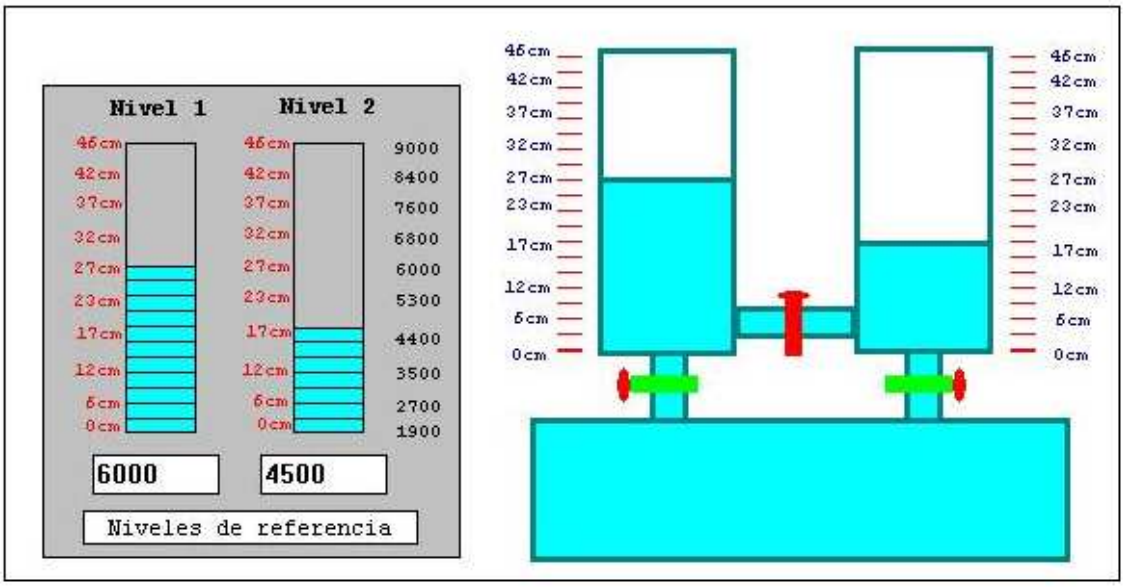


Figura 6.1.5 ***Estado niveles atascos caso 2***

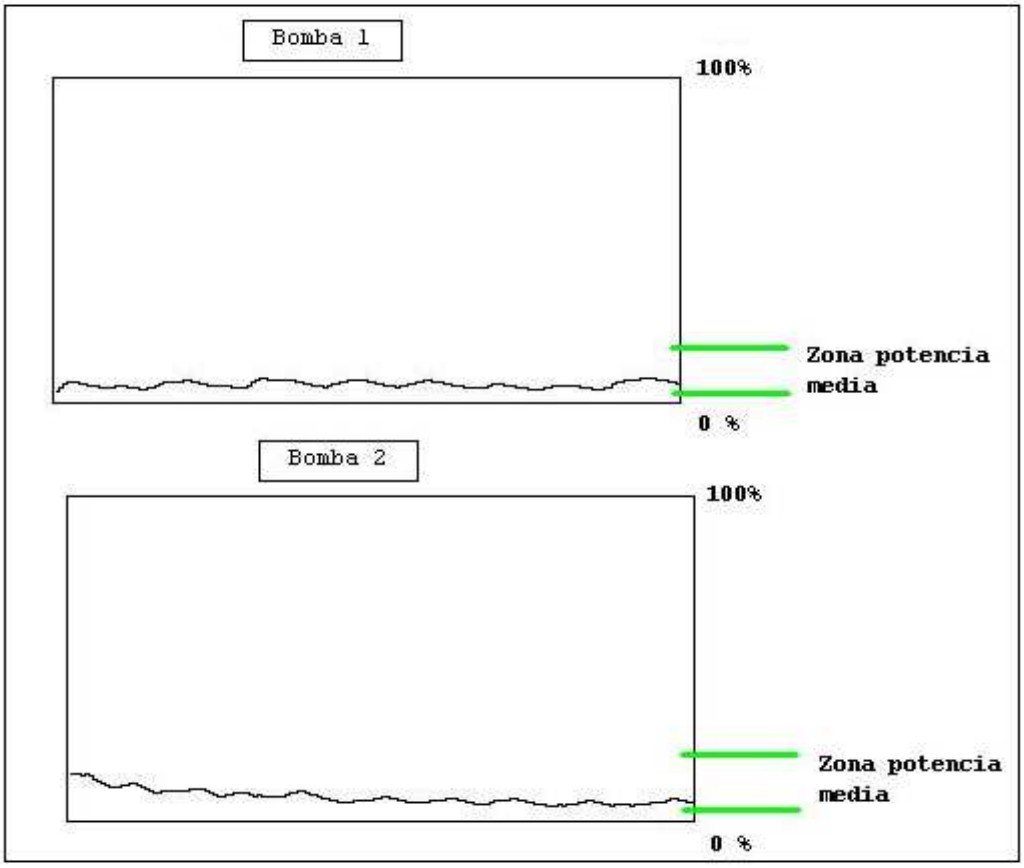


Figura 6.1.6 Gráfica potencia media bombas caso 2

Según la sección 'Estado válvulas': al encontrarse las dos bombas trabajando a media carga y estando el nivel 1 comprendido entre 5950 y 6050 y el nivel 2 entre 4450 y 4550 significa que la válvula central se encuentra cerrada.

Para el nivel de referencia del tanque de la derecha mayor que el del tanque de la izquierda:

- Las 3 válvulas abiertas: Los niveles de líquido se mantienen en los niveles de referencia establecidos. La bomba de la izquierda permanece inactiva mientras que la de la derecha trabaja a media potencia.
- La válvula central cerrada: Los niveles de líquido se mantienen en los niveles de referencia establecidos. Las dos bombas trabajan a media carga.
- La válvula de la derecha cerrada: Los niveles de líquido se mantienen en los niveles de referencia establecidos. La bomba de la izquierda permanece inactiva y la de la derecha funciona a media carga.
- La válvula de la izquierda cerrada: El nivel del tanque de la derecha se mantiene en el nivel de referencia establecido y el tanque de la izquierda aumenta progresivamente su nivel. La bomba de la izquierda permanece inactiva y la de la derecha funciona a media carga.

A continuación mostramos los resultados de este último caso, válvula de la izquierda cerrada y las otras dos válvulas abiertas:

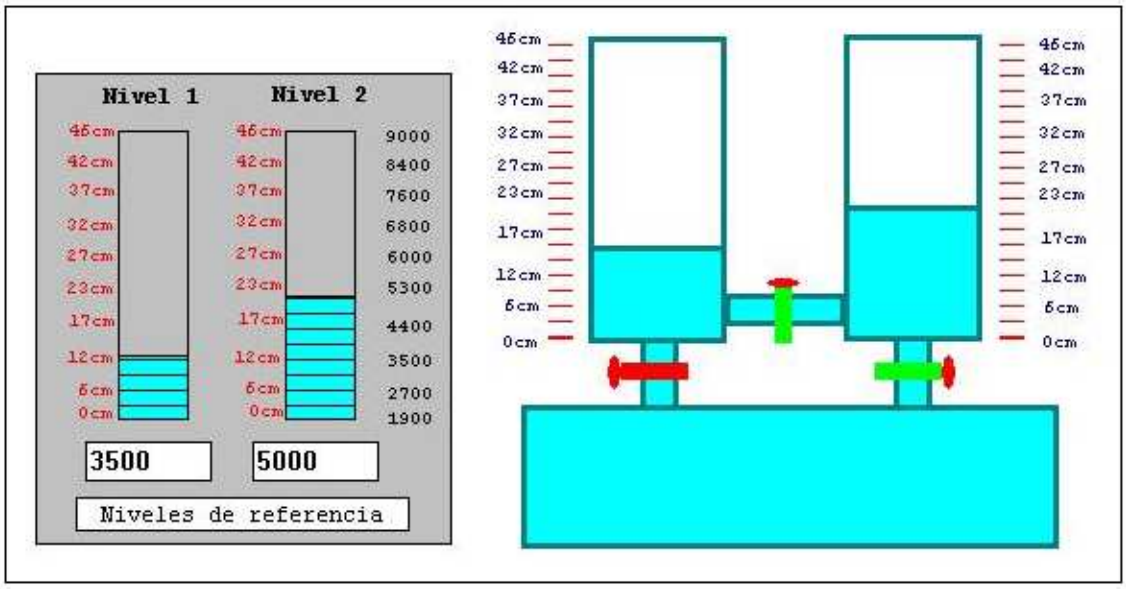


Figura 6.1.7 ***Estado niveles atascos caso 3***

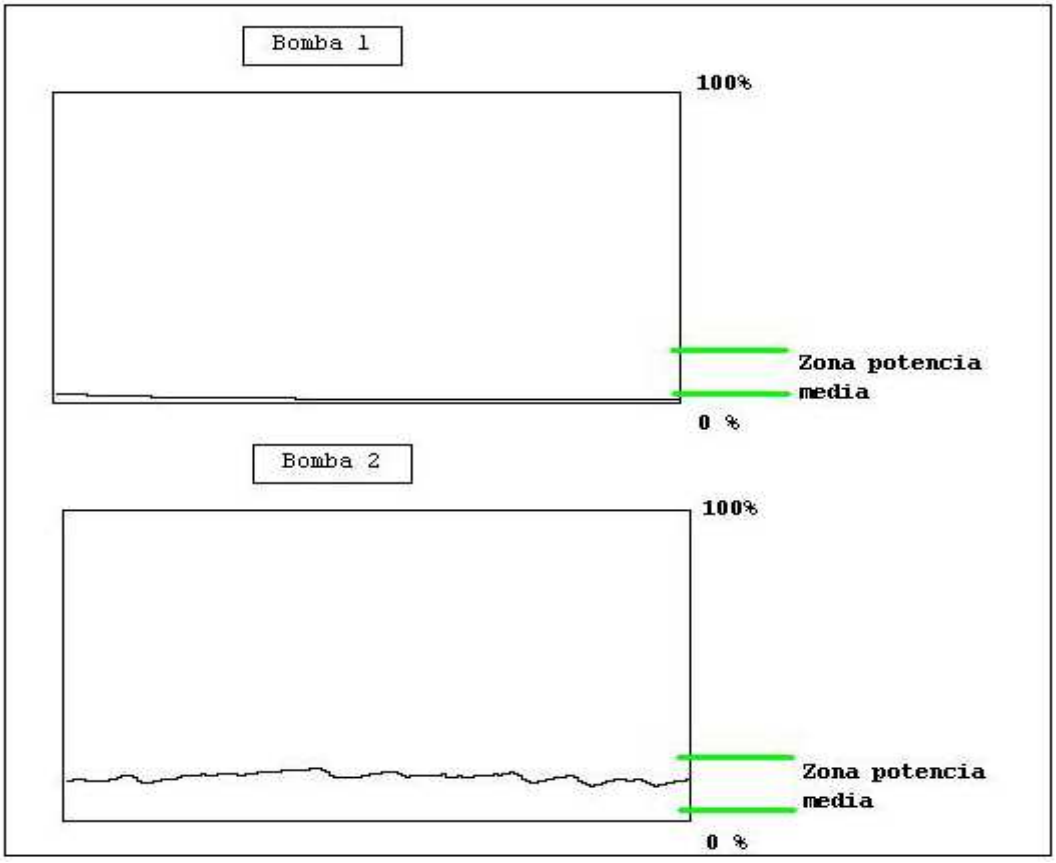


Figura 6.1.8 Gráfica potencia media bombas caso 3

Si la bomba 2 está trabajando a media carga, la bomba 1 está inactiva y el nivel 1 ha sobrepasado el valor 3550, quiere decir que la válvula izquierda se encuentra cerrada y las otras dos abiertas.

6.2. Conclusiones

Las pruebas realizadas al concluir todo el desarrollo de la aplicación han sido totalmente satisfactorias como se ha podido mostrar en el apartado anterior, en cuanto al control del nivel los resultados muestran una más que aceptable capacidad del sistema de controlar el nivel de forma eficiente y relativamente rápida; en cuanto a la detección de atascos los resultados nos permiten determinar que el sistema detecta satisfactoriamente los atascos producidos en cualquiera de las 3 válvulas que cuenta el sistema de la forma más fiable posible.

Por lo que se ha conseguido crear una aplicación la cual es capaz de controlar el nivel de un sistema de tanque comunicantes y también permite detectar atascos en cualquiera de las válvulas de desagüe de los tanques.

Por tanto podemos decir que los objetivos con los que se empezó este Proyecto se han cumplido muy satisfactoriamente. Además con la realización de este Proyecto

Fin de Carrera he aprendido entre otras cosas tanto a trabajar con el autómata Modicon M340, como a programar en lenguaje LD y ST; también he mejorado mi capacidad de resolución de problemas.

6.3. Líneas de desarrollo futuras

La aplicación que se ha realizado esta enfocada al sistema de tanques que hemos utilizado pero la aplicación se puede hacer extensible a cualquier sistema de tanques realizando pequeñas modificaciones; por lo que se puede utilizar fácilmente en cualquier proceso industrial.

Como ampliación de este proyecto y ya que la plataforma de automatización cuenta con un modem de comunicaciones dentro del conjunto se puede pensar en el envío de las señales de emergencia a sistemas remotos vía internet mediante el envío de email o vía telefónica mediante el envío de mensajes de texto a terminales móviles, así mismo se podría ampliar a estos sistemas propuestos la opción de modificar la altura de control del nivel de líquidos mediante la operación inversa, el envío desde los sistemas remotos hacia el modem de la automatización de la información o el código necesario para esta llevar a cabo esta variación.

Además a partir de la detección de atascos realizada por esta aplicación se podría avisar de la incidencia mediante la comunicación vía telefónica, e incluso poder recibir órdenes de actuación en esos casos.

Una gran línea de desarrollo futura puede ser la integración en el sistema de tanques de válvulas reguladoras de caudal, con las cuales permitirían un control más exhaustivo del nivel de los tanques, o la introducción de válvulas automáticas controladas por la aplicación.

7. BIBLIOGRAFIA

- BERNABEU, ENRIQUE J. Y MARTINEZ MIGUEL A.
Diseño algebraico de controladores discretos. Universidad Politécnica de Valencia.

- SCHENEIDER ELECTRIC
Catálogo Modicon M340 2009.
Manual de programación software Unity Pro.

- OGATA, KATSUHIKO
Ingeniería de control moderna. Madrid. Pearson Prentice-Hall, 2003 (4ª ed.).

Sitios web consultados

- Enciclopedia online Wikipedia:
wikipedia.org

- Instituto Schneider Electric de Formación:
instrumentacionycontrol.net

- Plataforma de automatización Modicon M340:
http://www.coevagi.com/Docs/Te_M340.pdf

- Entorno de programación de PLC's Unity:
http://193.146.57.132/depeca/repositorio/asignaturas/201608/T2_UNITY_Parte2.pdf

- PLC's de Schneider:
http://www.infoplc.net/files/descargas/schneider/infoPLC_net_1_PLCs_Schneider.pdf

- Electrónica Unicrom
http://www.unicrom.com

- Manual de usuario Modicon M340 con Unity Pro
http://lra.unileon.es/sites/lra.unileon.es/files/Documents/datasheet_Cel_flex/Modicon_M340.pdf

- RS
http://es.rs-online.com

- Manual Unity Pro
http://www.equiposdidacticos.com/pdf/catalogos/Manual_Unity.pdf

8. ANEXOS

ANEXO I. Archivos .m de matlab.

trimlinmod.m

```
% Función que nos calcula el estado estacionario a partir de un modelo de
% simulación del sistema en SIMULINK, linealiza el sistema y nos devuelve
% sus funciones de transferencia

[X,U,Y,DX]=trim('simulacion_sistema'); %Calcula el estado estacionario del sistema

[A,B,C,D]=linmod('simulacion_sistema',X); %Linealiza el modelo

[num1,den1]=ss2tf(A,B,C,D,1);
[num2,den2]=ss2tf(A,B,C,D,2); % Pasamos de espacio de estados a función de transferencia

G1=tf((num1(1,:)),den1)
G2=tf((num2(2,:)),den2)        %Mostramos las funciones de transferencia
```

Escalonplanta.m

```
%Calculamos la respuesta de uno de los tanques ante una entrada escalon unitario

%Definimos la funcion de transferencia de la planta

gp=tf([0.1833],[858 1])

%Damos una entrada escalon al sistema

step(gp)
```

respuesta_sistema_controlado.m

```
%Calculamos la respuesta de la planta con el controlador dando una entrada escalon
unitario

%Definimos la funcion de la planta

gp=tf([0.1833],[858 1])

%Definimos la funcion del controlador

gc=tf([2083 2.428],[1 1.2 0])

%Juntamos ambas funciones y calculamos la funcion final en lazo cerrado

f=gp*gc/(1+gp*gc)
f=minreal(f)

%Introducimos el escalon unitario

step(f)
```

ANEXO II. Normativa *IEC 61131-3.*

IEC 61131-3 es el primer paso en la estandarización de los autómatas programables y sus periféricos, incluyendo los lenguajes de programación que se deben utilizar. Esta norma se divide en cinco partes principales:

- Parte 1: Vista general.
- Parte 2: Hardware.
- Parte 3: Lenguaje de programación.
- Parte 4: Guías de usuario.
- Parte 5: Comunicación.

La vista general, también denominada de información general, estructura funcionalmente el sistema del autómata programable.

Esta estructura define una serie de partes concretas; función de tratamiento de la señal, función de interfaz con los sensores y actuadores, función de comunicación, función de interfaz hombre-máquina, funciones de programación, puesta a punto, ensayo y documentación, funciones de alimentación de corriente.

La parte del hardware se detiene en el estudio de las especificaciones y de los ensayos de los equipos, especificando requisitos eléctricos, mecánicos y funcionales, informaciones de los fabricantes, métodos y procedimientos de los ensayos, verificaciones.

En cuanto a los lenguajes de programación se especifica la sintaxis y semántica de un conjunto unificado de lenguajes de programación para controladores programables, nos centramos en esta parte ya que toca más de cerca el desarrollo del software de nuestra aplicación.

Éstos controladores están compuestos por dos lenguajes textuales, *IL* (lista de instrucciones) y *ST* (Texto estructurado), y dos lenguajes gráficos, *LD* (diagrama de contactos) y *FBD* (diagrama de bloques de función).

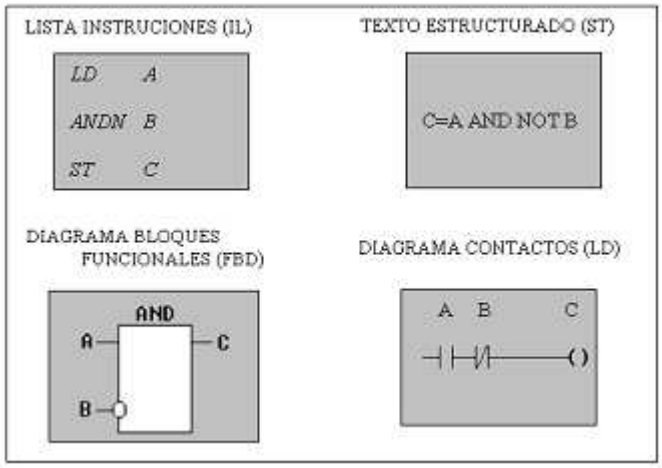


Figura A II.1. Lenguajes de programación IEC 61131-3

Además, los elementos del lenguaje de gráfica de función secuencial (SFC) se definen para estructurar la organización interna de los programas de controladores programables y los bloques de función.

También se definen los elementos de configuración, que admiten la instalación de programas de controladores programables en los sistemas de estos últimos.

Asimismo, se definen las funciones que facilitan la comunicación entre controladores programables y otros componentes de los sistemas automatizados.

La versión actual del sistema de programación Unity Pro admite un subconjunto compatible de los elementos de lenguaje definidos en la norma:

- La norma permite al encargado de implementar un sistema de programación *IEC* elegir o cerrar las funciones de lenguaje específicas o incluso completar lenguajes fuera de las tablas de función que forman parte inherente de las especificaciones; un sistema que solicite conformidad con la norma debe ejecutar las funciones elegidas de acuerdo con las especificaciones de la norma.

- Además, la norma permite al encargado mencionado con anterioridad utilizar los elementos del lenguaje de programación definido en un entorno de programación interactivo. Debido a que la norma afirma explícitamente que la especificación de dichos entornos no está dentro de su competencia, dicho encargado posee cierta libertad para proporcionar una presentación optimizada y procedimientos de manipulación para elementos de lenguaje específicos en beneficio del usuario.

- Unity Pro utiliza esta libertad mediante, para la manipulación combinada de los elementos de lenguaje *IEC* "Configuración" y "Recurso". Igualmente, hace uso de su libertad, para la manipulación de declaraciones de variable o las instancias de bloques de funciones.

La norma también hace referencia a los elementos comunes, dentro de los cuales se definen los tipos de datos. Los tipos de datos previenen de errores en una fase inicial, como por ejemplo la división de un dato tipo fecha por un número entero.

Los tipos comunes de datos son: variables booleanas, número entero, número real, byte y palabra, pero también fechas, horas del día y cadenas (*strings*). Basado en estos tipos de datos, el usuario puede definir sus propios tipos de datos, conocidos como tipos de datos derivados. De este modo, se puede definir por ejemplo un canal de entrada analógica como un tipo de dato.

Las variables permiten identificar los objetos de datos cuyos contenidos pueden cambiar, por ejemplo, los datos asociados a entradas, salidas o a la memoria del autómata programable. Una variable se puede declarar como uno de los tipos de datos elementales definidos o como uno de los tipos de datos derivados. De este modo se crea un alto nivel de independencia con el hardware, favoreciendo la reusabilidad del software.

La extensión de las variables está normalmente limitada a la unidad de organización en la cual han sido declaradas como locales. Esto significa que sus nombres pueden ser reutilizados en otras partes sin conflictos, eliminando una frecuente fuente de errores. Si las variables deben tener una extensión global, han de ser declaradas como globales utilizando la palabra reservada VAR_GLOBAL.

Pueden ser asignados parámetros y valores iniciales que se restablecen al inicio, para obtener la configuración inicial correcta.